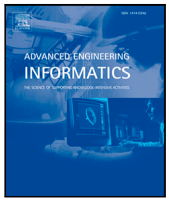
Advanced Engineering Informatics 61 (2024) 102497

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei



Traffic pattern-aware elevator dispatching via deep reinforcement learning

Jiansong Wan, Kanghoon Lee, Hayong Shin *

Department of Industrial and Systems Engineering, KAIST, 291 Daehak-ro Yuseong-gu, Daejeon 34141, Republic of Korea

ARTICLE INFO

Keywords:
Elevator dispatching
Semi-Markov decision process
Deep reinforcement learning
Traffic pattern awareness

ABSTRACT

This study addresses the elevator dispatching problem using deep reinforcement learning, with a specific emphasis on traffic pattern awareness. Previous studies on reinforcement learning-based elevator dispatching have largely focused on training separate models for single traffic patterns, such as up-peak, down-peak, lunch-peak, and inter-floor. This separate training approach not only introduces practical complexities by requiring an auxiliary model to predict traffic patterns for guiding dispatching decisions but is also computationally burdensome. In contrast, our goal is to develop a unified, traffic pattern-aware dispatching model. We formulate the elevator dispatching problem as a Semi-Markov Decision Process (SMDP) with novel state representation, action space, and reward function designs. To solve the formulated SMDP, we propose a Dueling Double Deep Q-Network (D3QN) architecture associated with the training algorithm. To ensure traffic pattern awareness, we train our model in a unified 'All in One' traffic scenario, employing two practical techniques to enhance the training process: (1) temporal grouping with gradient surgery and (2) incorporation of passenger arrival information. Empirical evaluations confirm the superiority of our model over multiple benchmarks, including those relying on separate, pattern-specific models. Remarkably, our unified model demonstrates robust performance across unseen traffic scenarios and performs exceptionally well in single traffic patterns despite being trained solely on the unified 'All in One' scenario. The short inference time for decision-making further solidifies the model's practical viability. Additionally, the incremental benefits contributed by each of our introduced techniques are also investigated. Our code is available at https://github.com/jswan95/RL-based-traffic-pattern-aware-elevator-dispatching

1. Introduction

Elevator systems have become integral components of high-rise buildings, facilitating vertical transportation for occupants. Typically, these systems consist of multiple elevator cars coordinated by a centralized Elevator Group Control System (EGCS). The EGCS's responsibility is to meet all vertical transportation demands while optimizing various performance metrics, such as Average Waiting Time (AWT), the percentage of long-waiting calls, and energy efficiency [1]. Among these metrics, minimizing the AWT has been a long pursuit for elevator industry practitioners and academic researchers.

In a traditional up-down elevator system, passengers first release a transportation request by pressing the hall buttons on each floor. The EGCS then dispatches the request to a specific elevator car, also known as the 'landing call assignment' process. The assigned elevator car proceeds to the requested floor and picks up the passengers. Finally, the elevator car drops the passengers off at their destination floor, as notified by passengers through the car button inside the cabin. Recent advancements in vertical transportation technologies have given rise to more sophisticated systems, including destination control [2], double-deck [3], and multi-car elevator systems [4], to satisfy the growing vertical transportation needs.

The effectiveness of elevator dispatching algorithms is widely acknowledged as a determinant factor in reducing AWT. Ruokokoski et al. [2] categorizes these algorithms into two primary types: immediate assignment and delayed assignment. Immediate assignment promptly allocates an elevator to a transportation request as soon as passengers initiate it. On the other hand, delayed assignment defers the final assignment decisions until the last feasible moment for making changes. Conventional approaches to elevator dispatching encompass a spectrum of methodologies including heuristics [5], genetic algorithms [6,7], fuzzy logic control [8], and multi-agent systems [9]. Nevertheless, these conventional methods struggle to achieve an even near-optimal policy due to the complex characteristics of elevator systems and the NP-hard nature of the elevator dispatching problem. Therefore, a promising direction to explore is the development of self-learning methods that enable elevator systems to autonomously reinforce their dispatching policy [10].

* Corresponding author.
E-mail addresses: jswan@kaist.ac.kr (J. Wan), leehoon@kaist.ac.kr (K. Lee), hyshin@kaist.ac.kr (H. Shin).

https://doi.org/10.1016/j.aei.2024.102497
Received 2 November 2023; Received in revised form 12 March 2024; Accepted 17 March 2024

Reinforcement Learning (RL) is a well-established learning paradigm that refines agent behavior through trial and error [11]. While RL's applicability to elevator dispatching has been explored in seminal works such as those by Markon et al. [12], Crites and Barto [13], their use of a simple neural network and Q-learning algorithm leaves room for improvement, especially considering the recent advancements in deep reinforcement learning techniques. Subsequent investigations by Ikuta et al. [4], Jansson and Uggla Lingvall [14] further underscored the utility of RL, developing agents to adaptively choose appropriate dispatching rules from a heuristic pool. However, the performance of such an approach heavily relies on the quality of the heuristic pool. A poorly designed pool can considerably restrict the agent's ability to achieve optimal performance. Recently, deep learning has become the frontier for artificial intelligence, and its combination with RL has also gained a lot of interest [15]. In a work by Wei et al. [10], a deep RL framework based on Asynchronous Advantage Actor-Critic (A3C) is proposed to address the elevator dispatching problem, leveraging the powerful feature extraction capabilities of deep neural networks to enhance the agent's understanding of the complex and dynamic state of the elevator system, thereby improving the overall system performance. However, their work assumes the availability of perfect information, such as the number of people waiting behind the hall call, which is often difficult to access in conventional elevator systems. This assumption limits the practicality of their approach for real-world implementation. Liu et al. [16] fused the information from image and voice recognition technologies, along with the Internet of Things to enable more nuanced RL-based decision-making.

Moreover, existing RL-based elevator dispatching studies have largely confined their focus to single traffic patterns for both training and testing. In real-world deployment, these pattern-oriented separate training approaches require an auxiliary model to predict current traffic patterns, which then selects a corresponding pre-trained model for decision-making. Therefore, the effectiveness of the dispatching algorithm becomes closely tied to the accuracy of the pattern-predicting model. Notably, even if the pre-trained models exhibit high performance under single traffic patterns, inaccuracies in pattern identification can significantly degrade the system's overall effectiveness. Additionally, traffic patterns do not always adhere to rigid categories. Various traffic patterns often share overlapping characteristics within certain time frames. This renders a pattern-predicting model, relying on strict discrete categorization, ill-suited for handling the mixed and dynamic nature of real-world traffic patterns.

To bridge the aforementioned research gap, this study introduces a novel traffic pattern-aware model for elevator dispatching. Our model first formulates the elevator dispatching problem as a Semi-Markov Decision Process (SMDP). We then present a corresponding Dueling Double Deep Q-Network (D3QN) architecture and training algorithm to solve the formulated SMDP, enabling the central controller to derive an effective elevator dispatching policy. To ensure traffic pattern awareness, we train the model in a unified 'All in One' traffic scenario. Moreover, we propose two practical techniques: temporal grouping with gradient surgery and incorporating passenger arrival information, to improve the training process under the unified traffic scenario. Empirical evaluations demonstrate our model's superiority against existing benchmarks, including those relying on separate, pattern-specific models. Our unified model also exhibits robust performance across various unseen traffic scenarios, further solidifying the model's practical viability. Notably, it still demonstrates outstanding performance in single traffic patterns despite being trained exclusively in a unified 'All in One' scenario. Additionally, we analyze the incremental benefits contributed by each of our introduced techniques. The main contributions of this study can be summarized as follows:

- We formulate the elevator dispatching as an SMDP, including novel designs of state representation, action space, and reward function.
- We propose a corresponding D3QN architecture associated with its training algorithm to optimize the elevator dispatching policy.
- We achieve traffic-pattern-aware elevator dispatching by training the proposed model on a unified 'All in One' traffic scenario, incorporating two effective practical techniques to enhance the training process.

The remainder of this paper is structured as follows: Section 2 prepares the preliminary knowledge for this study. Section 3 provides a detailed explanation of the SMDP formulation, including designs for state representation, action space, and reward function. Section 4 elaborates on the model architecture, training algorithm, and two practical techniques. Section 5 offers an exhaustive analysis of the empirical results, while Section 6 draws final conclusions.

2. Preliminaries

In this section, we provide an introductory overview of semi-Markov decision processes and value-based deep reinforcement learning to offer readers some preliminary knowledge before delving into the details of our model. Following that, we present a literature review regarding traffic analysis in elevator systems.

2.1. Semi-Markov Decision Process

The Semi-Markov Decision Process (SMDP) serves as a continuous-time extension of its discrete-time counterpart, the Markov Decision Process (MDP). A distinguishing feature between the two frameworks is the nature of state sojourn time: in SMDP, this time duration is represented by a general continuous random variable, as opposed to the fixed time steps used in MDP [17]. Due to its continuous-time characteristics, SMDP is often more well-suited for modeling decision-making problems that occur in continuous time. Several classical algorithms initially designed for solving MDPs, such as Temporal-Difference Learning with eligibility traces and Q-learning, have been adapted to address SMDPs by Bradtke and Duff [18]. The concept of 'reward-to-go,' also known as 'return,' is also redefined in the SMDP context. While in MDPs the return is calculated as a discounted sum of future rewards in discrete time, SMDP employs a discounted integral of continuous-time future rewards represented as follows:

$$\sum_{t=0}^{\infty} \gamma^t r_t \quad \text{becomes} \quad \int_0^{\infty} e^{-\beta\tau} r_\tau \, d\tau,$$

where r_t is the immediate reward in discrete time step t, $\gamma \in [0, 1)$ is the constant discount factor; r_τ is the instantaneous reward at continuous time τ, and β controls the rate of exponential decay.

2.2. Deep reinforcement learning

Traditional value-based RL methods, like Q-learning and SARSA, commonly learn a policy by updating a tabular value function for all possible state–action pairs. However, these methods suffer from the so-called 'curse of dimensionality' when applied to complex problems with numerous state–action pairs. Deep Reinforcement Learning (DRL), incorporating deep neural networks into RL, has attracted considerable interest in recent years.

For value-based RL methods, DRL replaces the tabular value function with deep neural networks whose parameters implicitly represent all possible state–action values. DeepMind proposed the Deep Q-Network (DQN), which remarkably mitigates the instability of DRL by introducing two techniques: experience replay and a separate target Q-network [19]. The target Q-network (parameterized by θ^-) with the same structure as the main Q-network (parameterized by θ) is to generate learning targets for the main Q-network. For training stability, the parameters of the target Q-network are updated periodically by copying from the main Q-network. The parameters of the main Q-network can be optimized by minimizing the empirical mean squared

Temporal Difference (TD) error regarding the experience batch of size m sampled from the experience replay buffer:

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{(s,a,r,s')} \left(r + \gamma \max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a) \right)^2.$$

Following that, numerous efforts have been made to further improve DRL with novel architectures or training algorithms, such as double DQN [20], dueling architecture [21], prioritized experience replay [22], and deep deterministic policy gradient [23]. DRL finds applications in a diverse array of industries, such as manufacturing, transportation, energy management, and internet traffic management [24–30].

2.3. Traffic analysis in elevator systems

In contrast to dispatching problems in other transportation systems [31,32], elevator systems exhibit distinct traffic patterns. Therefore, traffic analysis is a crucial area of study in elevator systems because different traffic flows result in various traffic patterns, necessitating different dispatching policies. According to Barney and Al-Sharif [33], traffic patterns in an office building during a typical day can be classified into four types based on the main traffic flow feature: up-peak (ascending from the lobby), down-peak (descending toward the lobby), lunch-peak (both), and inter-floor (no clear predominant flow). Accurately recognizing the traffic pattern of the elevator system contributes to supporting EGCS dispatching decision-making, thereby enhancing passengers' user experience.

Numerous studies have delved into the analysis of traffic patterns and passenger arrivals in elevator systems. So et al. [34] developed a model recognizing five types of traffic patterns using Artificial Neural Networks (ANNs) with nine system attributes as inputs. Subsequently, a multi-value support vector machine-based elevator traffic pattern classifier was proposed, showing superior performance than that of simple ANNs [35,36]. Cortés et al. [37] proposed a fuzzy logic-based peak traffic detection by considering information related to car load and car direction. Sorsa et al. [38] found that modeling uncertainties benefits solving elevator dispatching problems, and the geometric Poisson process outperforms the Poisson process in forecasting uncertain demand. Sorsa et al. [39] explored the size of social groups among passengers in office, hotel, and residential buildings, providing insights into modeling passenger group arrivals in elevator traffic simulations.

Not only have these studies focused on analyzing traffic, but many have also attempted to use traffic-related information to enhance dispatching decision-making processes. Utgoff and Connell [40] introduced a Minimize Vexation, version 10 (MV10) dispatcher, which models individual users in the system to infer more information regarding the current state, such as the number of users in the hallway and car, destination of users in the hall and car, to aid decision-making. Their model also projects hall calls in the near future for idle car placement. Wang et al. [41] suggested integrating elevator car occupancy information to avoid pick-up failures. Zheng et al. [42] employed Dynamic Time Warping (DTW) and K-means clustering to model the statistical distribution of traffic data, facilitating the optimization of dispatching rules via simulation. Furthermore, Zhang et al. [43] predicted future passenger arrivals using a Transformer network, subsequently leveraging this data for predictive elevator dispatching.

However, a common limitation observed in these studies is the disconnect between the traffic information extraction and decision-making modules, introducing additional complexities in effectively leveraging the extracted traffic information within existing dispatching models. Our model differs from these studies because the traffic information capturing and decision-making are trained in an end-to-end manner without any further burden of considering how to effectively use traffic-related information for decision-making.

3. Semi-Markov Decision Process formulation

Elevator dispatching involves decision-making that affects the availability of elevator cars and passengers waiting in the future. Additionally, passengers continuously enter the elevator system and release transportation requests. This requires us to make sequential decisions with a potentially infinite optimization horizon. Therefore, to make optimal dispatching decisions, it is essential to adopt a farsighted approach that considers the impact of present decisions on future outcomes. In this regard, we address the elevator dispatching problem by formulating it as a Semi-Markov Decision Process (SMDP). Given that the time intervals between discrete events are real-valued variables and the passenger arrival rates are also dynamically changing, SMDP is more suitable for our purposes compared to an MDP. In the following subsections, we provide detailed explanations of the state representation, action space, and reward function of our SMDP formulation.

3.1. State representation

The state summarizes the system's current situation and aids the agent's decision-making. As illustrated in Fig. 2, our original state representation contains two main categories of information: hall information (red, four columns) and elevator car information (yellow, $3 \times C$ columns). Formally, the state is represented as a matrix of dimensions $F \times (4 + 3 \times C)$, where F denotes the number of floors and C represents the number of cars. Each element located at row f and column k of this matrix encodes the kth feature of the fth floor, defined as follows:

- $k = 1$ (or 2): binary, whether the up (or down) button at floor f is pressed
- $k = 3$ (or 4): real, specifies the elapsed time since the up (or down) button at floor f was pressed
- $k = 4 + 3 \times j + 1$: binary, whether passengers will get off at floor f from car j
- $k = 4 + 3 \times j + 2$ (or 3):binary, whether car j is currently at floor f with a moving up (or down) state

3.2. Action space

The action space represents all possible behaviors of the agent at a specific decision-making moment. In our formulation, each elevator car has five possible actions: stopping at up-floor, stopping at down-floor, passing up-floor, passing down-floor, and staying at the current floor. We consider a single-agent view instead of a multi-agent view in our formulation. Therefore, from the agent's view, the total number of possible actions amounts to $5 \times C$. Serving as the central controller for all elevator cars, the agent determines actions for each car as needed. At a decision-making moment, actions corresponding to the non-decision-needed cars will be masked out. The remaining actions corresponding to the unique decision-needed car may also be masked out by following the constraints considered in previous works. For example, reversing is not allowed until the car has served all the car buttons in the present direction; a car cannot stop at a floor if another car has already stopped there. The set of unmasked actions is termed the *admissible actions*, denoted as A_t. Also, we define the discrete event (when an elevator car arrives at a floor and finishes loading or unloading passengers if any) as a *decision event* if the cardinality of the admissible action set exceeds one, i.e., $|A_t| > 1$. Upon the occurrence of a decision event, the agent must select a particular action a_t from the set A_t. To illustrate, consider Fig. 2, where actions denoted in red are associated with the car requiring a decision. Actions corresponding to other cars (purple dashed) are automatically masked. Certain actions (red dashed) may also be masked due to pre-established rules. The final action (green outline) is selected from the remaining admissible actions (red solid).

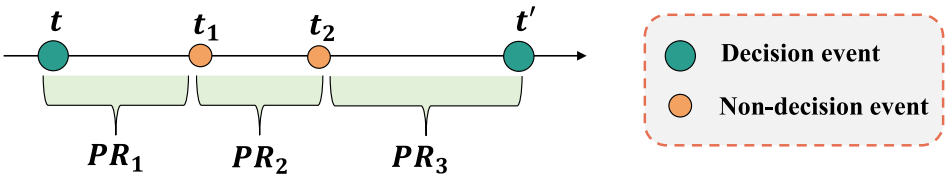


Fig. 1. Illustration of reward calculation. In this example, the reward is the sum of three partial rewards, each of which is calculated by integrating the instantaneous reward rate between two successive discrete events.

3.3. Reward function

The reward function is designed to evaluate the agent's action selection and optimize the decision-making process. In our framework, the reward is computed as the sum of partial rewards between two consecutive decision events, which occur at times t and t', respectively. Each partial reward is a discounted integral over the instantaneous reward r_τ, evaluated at every time point τ between the last event time t^l and the current event time t^c. Suppose n non-decision events occur during this transition, the total reward is mathematically represented as:

$$R(s,a) = \sum_{(t^l,t^c)\in\{(t,t_1),(t_1,t_2),\ldots,(t_n,t')\}} \underbrace{\int_{t^l}^{t^c} e^{-\beta(\tau-t)} r_\tau d\tau}_{\text{Partial reward}} \tag{1}$$

The key is how to define r_τ. Here we consider three variants of r_τ with different exponents:

$$r_\tau = -\left(\underbrace{\sum_{p\in HP} \left(\tau - t_p^{arrive}\right)^i}_{\text{WT signal}} + const \cdot \underbrace{\sum_{p\in CP} \left(\tau - t_p^{board}\right)^i}_{\text{TT signal}} \right), \tag{2}$$

where $i = 0, 1, 2$. The negative sign ensures that the reward function becomes a quantity subject to maximization. The first term within the parenthesis corresponds to the Waiting Time (WT) signal, and the second term refers to the Travel Time (TT) signal. The constant $const$ serves to balance these two reward signals. For $i = 0$, each passenger p in the hall passenger set HP or the car passenger set CP produces an identical instantaneous reward of -1 regardless of their arrival time t_p^{arrive} or boarding time t_p^{board}. When $i = 1$, the instantaneous reward scales linearly with the time a passenger has waited or traveled. For $i = 2$, the reward decreases quadratically with increasing wait or travel time, thus amplifying the urgency from the agent's perspective. The effects of these different definitions of instantaneous reward are empirically examined in Section 5, where readers can find more analysis and discussions.

Fig. 1 presents an example of how reward computation is performed. The agent selects an action a in state s at t, with the next decision event occurring at t'. This interval incorporates two non-decision events such as passenger arrivals and boardings. Consequently, the total reward for this transition can be expressed as:

$$R(s,a) = PR_1 + PR_2 + PR_3, \tag{3}$$

where PR represents the partial reward. To calculate PR, we need the current event time, last event time, and last decision time. In the example,

$$PR_3 = \int_{t_2}^{t'} e^{-\beta(\tau-t)} r_\tau d\tau \tag{4}$$

For further elaboration on the calculation of PR under the three different variants of instantaneous reward, readers may refer to Appendix A.

4. Model architecture and training algorithms

This section elucidates the architecture and training algorithm of our model in detail. In particular, we elaborate on the three techniques that improve the stability and effectiveness of the training process for a unified model.

4.1. Foundational training algorithm and architecture

Fig. 2 offers a comprehensive overview of the proposed architectural design. We adopt a Deep Double Dueling Q-Network(D3QN) for approximating the state–action value function to avoid the 'curse of dimensionality'. The state representation matrix initially passes through a convolutional block, which consists of two one-dimensional convolution layers. These one-dimensional convolutions operate along the floor axis, allowing for effective feature extraction across all floors. The convolutional block output is then flattened and fed into the state value head and action advantage head separately. Ultimately, we compute the estimated state–action values based on the combined outcomes of the state values and action advantages.

To optimize the model parameters, we customize the D3QN algorithm within our formulated SMDP. During training, partial rewards accumulate following each discrete event. We then determine whether the event is a decision event. If so, we store the new experience segment $(s, a, t, R(s,a), s', A_{t'}, t')$ into a replay buffer D. Later we sample a batch of experience segments with size m from the replay buffer for model training. The Temporal Difference (TD) target is calculated using the Double Q-learning algorithm as follows:

$$q' = R(s,a) + e^{-\beta(t'-t)} Q_{\theta^-}(s', \text{argmax}_{a'\in A_{t'}} Q_\theta(s', a')). \tag{5}$$

It is worth noting that we employ a variable discount factor $e^{-\beta(t'-t)}$, which depends on the time interval $t' - t$ between two successive decision events. A longer time interval results in a higher discount, thus providing a more nuanced approach compared to the traditional use of a fixed discount factor within the conventional MDP framework.

The loss function $\mathcal{L}(\theta)$ is the mean square of the TD error:

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_M \left(q' - Q_\theta(s,a)\right)^2. \tag{6}$$

We can optimize the loss function by performing gradient descent steps with respect to the parameter θ.

$$\theta \leftarrow \theta + \eta \underbrace{\frac{1}{m} \sum_M \left(q' - Q_\theta(s,a)\right) \nabla Q_\theta(s,a)}_{\nabla_\theta \mathcal{L}(\theta)}. \tag{7}$$

4.2. Special efforts for 'All in One' scenario: Two practical techniques

The foundational training algorithm described above is effective enough for training agents under single traffic patterns. Moreover, when applied directly to the 'All in One' scenario, it does indeed result in an agent that exhibits a certain level of traffic pattern awareness, owing to the experience segments from various traffic patterns stored in the replay buffer. However, there are still several challenges because

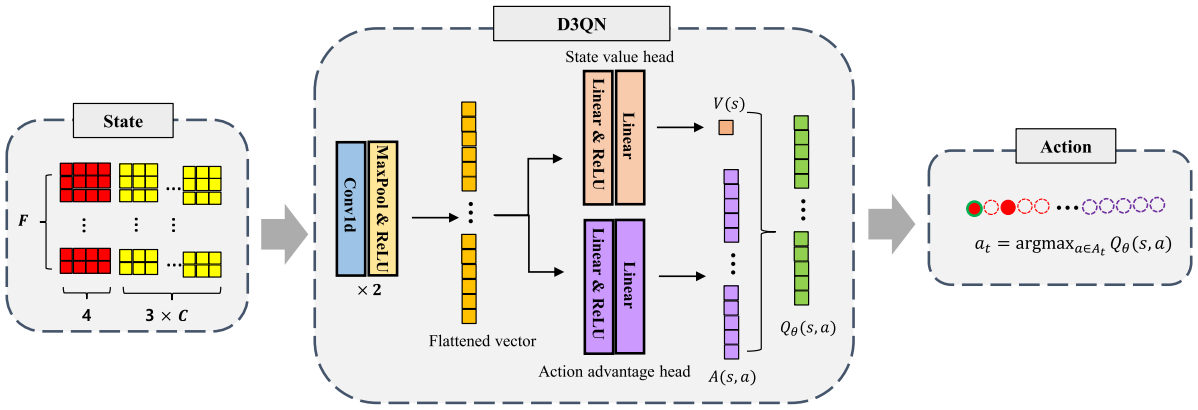


Fig. 2. Overview of the proposed model.

Algorithm 1: Training algorithm of the proposed model

Initialize: replay memory D_k for each temporal group k;
the parameter of main Q-network θ ;
the parameter of target Q-network $\theta^- \leftarrow \theta$;
while discrete event occurs
  Calculate partial reward according to Equation(1);
  $R(s,a) \leftarrow R(s,a) + PR$; /* Accumulate partial reward */
  $t^l \leftarrow t_{now}$; /* Update last event time */
  if *decision event* **then**
    $t' \leftarrow t_{now}$;
    Get admissible actions $A_{t'}$;
    Execute action a' selected according to ϵ-greedy policy;
    Store segment $(s, a, t, R(s,a), s', A_{t'}, t')$ into corresponding D_k ;
    $s \leftarrow s'$; /* Update previous state */
    $a \leftarrow a'$; /* Update previous action */
    $R(s,a) \leftarrow 0$; /* Reset reward */
    $t \leftarrow t'$; /* Update previous decision event time */
    Sample m segments from each D_k ;
    Calculate TD target according to Equation(6) ;
    Calculate losses according to Equation (7) ;
    Gradient surgery among different temporal groups as shown in Figure3 ;
    Perform gradient descent with respect to θ according to Equation(8);
    Every T steps: $\theta^- \leftarrow \theta$;
until training ends;

of the dynamically changing and mixed nature of the traffic scenario, indicating a need for improvement.

Subsequently, we introduce two highly effective practical techniques aimed at enhancing the training process, thereby improving the model's performance within this unified scenario. Algorithm 1 presents the details of the complete training procedure. The complexity of the proposed algorithm linearly depends on the number of temporal groups. This dependence arises due to the necessity of calculating the gradient of each temporal group through backpropagation. It should be noted, however, that this aspect primarily impacts the training time. Once the network is trained, the inference computation time for decision-making is demonstrated to be sufficiently fast for real-time applications, as indicated in our computation time analysis experiment. Other contributing factors, such as the number of elevator cars and floors, may be considered negligible for the computation time because they only affect the design of the network architecture.

4.2.1. *Temporal grouping with gradient surgery*

The elevator system within the unified 'All in One' traffic scenario operates within a dynamic and evolving environment, where dispatching policies may vary across different time periods. This is the big challenge for training a unified agent under the 'All in One' traffic scenario because the experiences collected at different time moments might lead to different parameter updates thus harming the overall performance. We resolve this issue via temporal grouping with gradient surgery.

Firstly, we uniformly divide the 'All in One' traffic scenario into different time intervals. During training, we store and sample experiences corresponding to the time interval in which the experiences happened. This temporal grouping is essential because considering each individual time moment independently can introduce noise and inefficiency into the training process.

Secondly, we perform gradient surgery before updating the network parameters to resolve the potential conflicting gradient that may arise from different temporal groups. When gradients are derived from training samples of different temporal groups, these gradients may conflict with each other, interfering with the performance of each other. The situation is exacerbated when there is a significant inconsistency in gradient scales. The performance associated with the smaller gradient tends to be overshadowed by the one with the larger gradient. To mitigate this issue, we suggest projecting a temporal group's gradient onto the orthogonal plane of the gradient of any conflicting temporal group as shown in Fig. 3. This projection strategy effectively neutralizes the conflicts in gradient scale and direction, resulting in more balanced and stable training outcomes. Employing this technique offers marked gains in training efficiency and stability for agents working within the 'All in One' scenario. For more details of the gradient surgery algorithm, readers can refer to the work of Yu et al. [44].

4.2.2. *Incorporation of passenger arrival information*

Moreover, we propose enriching the state representation by incorporating passenger arrival information into the original state representation matrix as shown in Fig. 4. The augmented state representation matrix is supposed to sharpen the agent's understanding of traffic dynamics under the 'All in One' scenario.

Each element of the two added columns means the arrival rate from(to) floor f. The passenger arrival rate is computed according to Eq. (8), which integrates both historical and real-time data. Historical data offers a prior estimation of the passenger arrival rate, denoted by λ^{prior}. In our experiment, λ^{prior} is employed to simulate the generation of passengers. On the other hand, real-time data is to reflect the actual rate of passenger arrivals.

$$\lambda_f^{\text{post}} = \alpha \times \lambda^{\text{prior}} + (1-\alpha) \times \frac{NP_f}{\Delta t} \tag{8}$$

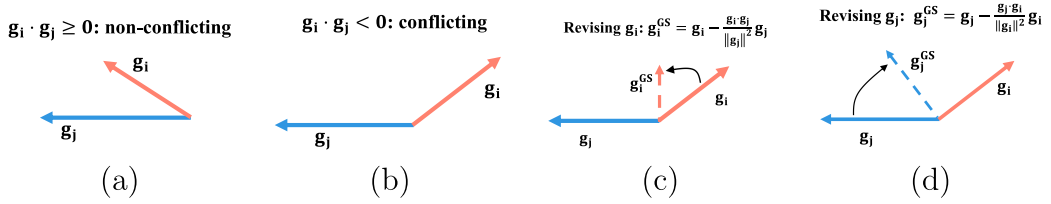


Fig. 3. Gradient surgery resolving conflicting gradients.

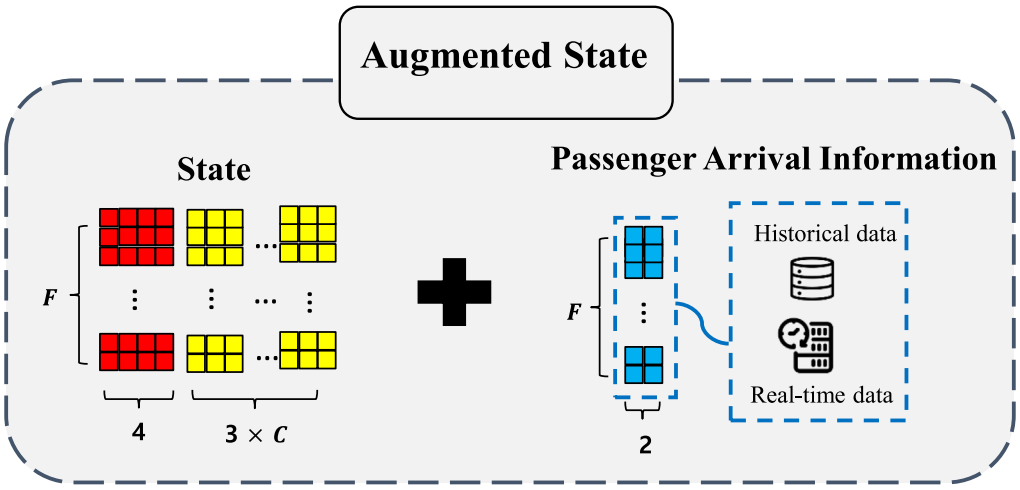


Fig. 4. The augmented state incorporates passenger arrival information into the original state representation.

Table 1
Summary of simulator settings.

System component	Settings
Building	Number of floor: 20 Floor height: 3 m Population: 1200
Car	Number of cars: 4 Capacity: 20 Door opening time: 1 s Door closing time: 1 s Boarding time: [1.6, 7] Alighting time: [1.6, 7] Full speed: 3 m/s Acceleration: 1.5 m/s^2 $t_{0,0}$: $2\sqrt{2}$ s $t_{0,F}$: 2 s $t_{F,0}$: 2 s $t_{F,F}$: 1 s

Here, NP_f refers to the number of passengers arriving whose origins (or destinations) are floor f during the most recent time window of length Δt. The constant α serves as a weighting factor to balance the contributions of the two types of data.

5. Numerical experiment

In this section, we first describe the simulator settings for our experiments. Then, a detailed analysis of the experiment results is presented.

Several elevator simulators have been developed in previous literature. The Building Traffic Simulator (BTS) stands out as a tool developed by KONE Corporation, offering a comprehensive platform where diverse building configurations and transportation devices can be specified. Notably, BTS enables the assessment of transportation devices' capabilities to manage passenger traffic in various scenarios, including exceptional situations such as building evacuations. Another noteworthy simulator is ELEVATE [45], a widely acknowledged simulation tool developed by Dr. Peter and commercialized by Peters Research, Ltd. Cortés et al. [46] introduced SimMP, a tool designed for planning and simulating dynamic vertical traffic. Miyamoto and Yamaguchi [47] developed MceSim, a multi-car elevator system simulator specifically capturing the moving dynamics of multiple elevator cars within a single shaft.

However, several constraints impede the utilization of existing elevator simulators for our study. Primarily, the majority of renowned elevator simulators are implemented in C++, posing challenges for integration with our reinforcement learning (RL) model, which is coded in Python. Furthermore, most of these simulators are not publicly accessible, creating further barriers to their utilization. To address these limitations, we have released our Python-based elevator simulator code to facilitate its use in future RL-based elevator dispatching studies, thereby contributing to the accessibility and versatility of simulation tools in this domain.

5.1. Simulation settings

Table 1 provides an overview of our simulator environment, featuring a 20-floor building equipped with four elevator cars. The building's floor height measures 3 m. Each elevator car possesses a maximum speed of 3 m/s and accommodates up to 20 passengers. The door opening time and closing time default as 1 s. The boarding time and alighting time for each passenger follow a truncated normal distribution from 1.6 to 7 with a standard deviation of 1. Our simulator takes into account the influence of acceleration and deceleration, a crucial factor that impacts travel times. This is accomplished by modeling the elevator cars as uniformly accelerating from a standstill to their maximum speed (and conversely, decelerating to a stop) over a distance equivalent to the height of a floor. This approach enables us to calculate the acceleration associated with different travel scenarios. For instance, when a car passes a floor at full speed and halts at the subsequent floor, the travel time is 2 s, denoted as $t_{F,0} = 2$ s. In contrast, $t_{0,0}$ signifies the

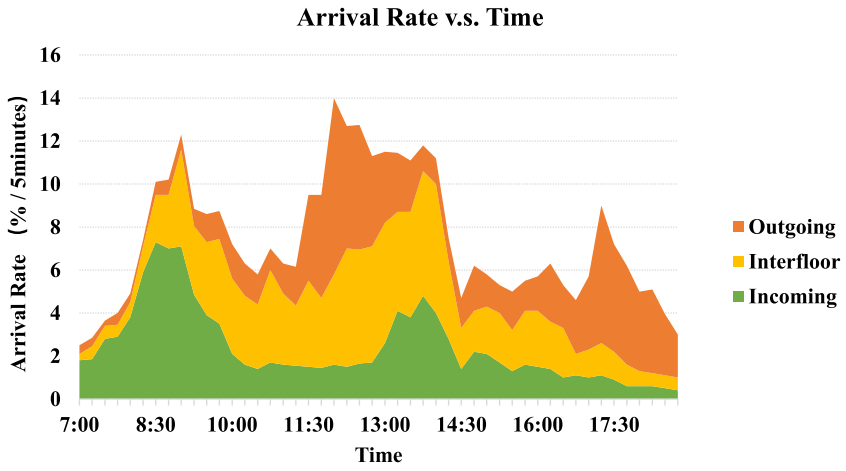


Fig. 5. The daily traffic profile used in our experiments. The arrival rate of passengers in a 5-minute period is given as a percentage of the total population of the building.

travel time when a car that has recently halted at a floor proceeds to stop at the next one.

Within our simulator, we utilize a daily traffic profile, adapted from the work of Hautamäki et al. [48]. As illustrated in Fig. 5, this profile captures the traffic dynamics within an office building from the hours of 7:00 to 19:00. It comprises three distinct traffic flows: incoming (representing passengers moving from the lobby to other floors), outgoing (representing passengers moving from other floors to the lobby), and inter-floor (encompassing passengers traveling between various floors in both upward and downward directions). These time-varying traffic flows align with the classical elevator traffic theory and correspond to four typical traffic patterns: up peak (7:00–10:00), inter-floor (10:00–11:30, 14:30–16:00), lunch peak (11:30–14:30), and down peak (16:00–19:00). We refer to the ensemble of these four traffic patterns as the 'All in One' scenario.

5.2. Experiment and result analysis

Through the experimental evaluation, we aim to answer the following research questions:

- **Analysis of Instantaneous Reward**: How does our model perform with the three different instantaneous reward definitions?
- **Performance Under Single Traffic Patterns**: How does the performance of our model contrast with benchmark rules under typical single traffic patterns, including up-peak, inter-floor, lunch-peak, and down-peak?
- **Performance Under the 'All in One' Scenario**: How well does our model perform under 'All in One' scenario, how well our model is traffic pattern aware, and how do the two practical techniques benefit our model?
- **Running Time Analysis**: How long it takes for our model to make a dispatching decision? Is it fast enough for real-time dispatching?
- **Evaluation of Robustness**: How robust is our model under different unseen population levels and unseen traffic profiles?
- **Effect of Combining Historical Data and Real-Time Data**: What is the advantage of combining historical data and real-time data in our model?

The general settings for our experiments are as follows. We train the agents under four single traffic patterns for 80 simulation days each. The 'All In One' scenario, being a mixture of the four single traffic patterns, requires a training time of 320 simulation days. The constant β determining the discounting degree defaults as 0.01. The constant β determining the discounting degree defaults to 0.01, and the balancing factor $const$ in the instantaneous reward definition defaults to 0. We use Adam as the parameter optimizer for the neural networks, with an initial learning rate of 5×10^{-5}. The value of α in Eq. (8), which calculates the arrival rate, is set to 1 in all experiments, except for experiment 5.

Table 2
Experimental analysis of three instantaneous reward.

	r_0	r_1	r_2
Avg WT	21.99 ± 0.18	22.00 ± 0.22	22.29 ± 0.21
Max WT	248.41 ± 42.62	222.18 ± 27.16	203.07 ± 28.55
Avg TT	44.79 ± 0.23	44.75 ± 0.40	45.15 ± 0.31
Max TT	278.56 ± 44.94	253.92 ± 20.86	239.58 ± 19.34

The benchmark rules used for comparison with our model include Round Robin (RR), Scan, Look, and a Genetic Algorithm (GA)-based dispatching proposed by Tartan and Ciftlikli [6], who improved upon the method proposed in Cortés et al. [7] by replacing the existing binary encoding with decimal encoding. This proposed decimal encoding is easier to implement and demonstrates better performance. Another benchmark is a Particle Swarm Optimization(PSO)-based dispatching approach proposed by Bolat et al. [49]. The last benchmark is an RL-based dispatching model that learns to assign newly arrived requests to the most suitable car. This model is adapted from the RL-based Elevator Group Control (RL-EGC) model proposed by Wei et al. [10].

To ensure the reliability of our results, we conducted each test experiment ten times with different random seeds, each spanning a simulation day. Each simulation is run with a 12-hour warming-up period and a 12-hour fade-out period during which the data is not collected for more solid analysis.

5.2.1. Analysis of instantaneous reward

This experiment empirically examines the effect of the three instantaneous rewards defined in Section 3. To do so, we trained three agents using the three reward definitions under a down-peak traffic pattern. The results, as depicted in Fig. 6, unveil a consistent decline in Average Waiting Time (AWT) as the training progresses, underscoring the effectiveness of all three instantaneous reward definitions. In addition, Table 2 offers insights into the performance of these reward definitions based on both 'Avg' metrics (Average Waiting Time and Average Travel Time) and 'Max' metrics (Maximum Waiting Time and Maximum Travel Time). Notably, the r^0 reward definition performs best regarding the 'Avg' metrics due to its alignment with these metrics. Conversely, r^1 and r^2 exhibit superior performance in the 'Max' metrics, primarily because they consider the passenger arrival and boarding times. This consideration results in passengers with earlier arrival yielding fewer rewards when using r^1 and r^2 as the instantaneous reward, thus forcing the agent to make more equitable decisions. r^2

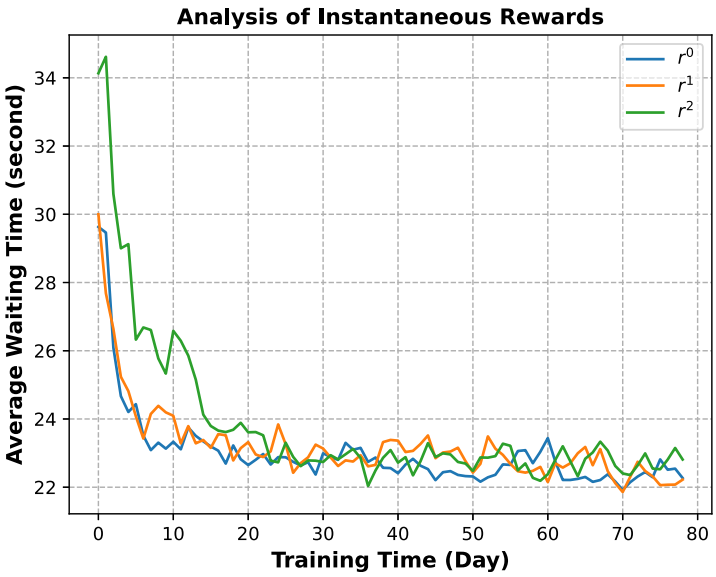


Fig. 6. Training process of three instantaneous reward definitions.

Table 3
Average waiting time of various scenarios.

Methods		AllInOne	UpPeak	InterFloor	LunchPeak	DownPeak
Benchmarks	RR	–	–	44.53 ± 0.95	–	36.65 ± 0.64
	Scan	123.28 ± 15.69	220.93 ± 9.76	33.99 ± 0.41	141.01 ± 9.24	33.95 ± 0.22
	Look	104.73 ± 12.19	170.54 ± 17.05	30.34 ± 0.30	136.48 ± 8.24	30.45 ± 0.31
	GA	78.89 ± 5.11	115.73 ± 4.87	24.68 ± 0.33	120.12 ± 6.42	23.58 ± 0.24
	PSO	76.11 ± 3.21	110.34 ± 5.14	24.53 ± 0.25	112.09 ± 3.11	23.32 ± 0.14
	ETA	66.59 ± 3.52	95.36 ± 8.04	24.72 ± 0.32	98.00 ± 6.40	23.44 ± 0.32
	RL-EGC	63.01 ± 5.54	89.18 ± 8.07	23.95 ± 0.22	92.22 ± 6.06	22.51 ± 0.26
Ours	4Separate	59.47 ± 2.89[a]	81.49 ± 8.33[b]	22.99 ± 0.15[b]	82.66 ± 5.71[b]	22.00 ± 0.22[b]
	Unified(Basic)	61.23 ± 4.32	88.94 ± 10.42	24.30 ± 0.31	85.11 ± 5.58	25.40 ± 0.61
	Unified(+GS)	52.18 ± 2.84	69.35 ± 7.45	22.95 ± 0.31	77.72 ± 1.00	21.52 ± 0.19
	Unified(+GS+PAI)	47.62 ± 3.73	61.69 ± 7.61	22.78 ± 0.40	67.21 ± 1.16	21.32 ± 0.21

–: Unmeaning result because of overloading.
[a] Performance of RL-4Separate agent which combines the four specialized RL-Separate agents.
[b] Performance of RL-Separate agent for single traffic pattern.

achieves an even smaller value than r^1 regarding the 'Max' metrics due to its quadratic relationship. However, out of the three instantaneous reward definitions, r^2 is the farthest from our objective of minimizing AWT. Consequently, for the subsequent experiments, we opt for r^1 as the instantaneous reward definition. This choice strikes a balance by maintaining consistency with our AWT minimization objective while retaining fairness in decision-making processes.

5.2.2. Performance under single traffic patterns

This experiment presents a comparative analysis of our proposed method against established benchmark rules under four single traffic patterns. This is what most existing RL-based elevator dispatching studies have done in their experiment section. We separately trained and evaluated four specialized RL agents, referred to as RL-Separate, each optimized for a particular traffic pattern. As detailed in Table 3, our model consistently demonstrates a remarkable performance edge over the benchmark rules across all four traffic patterns, demonstrating the effectiveness of our proposed approach. Fig. 7 illustrates the training process of the four RL-Separate agents under each single traffic pattern.

5.2.3. Performance under 'All in One' scenario

In this experiment, we evaluate the effectiveness of our model under 'All in One' Scenario. we first trained a Unified(Basic) agent without incorporating either of the two proposed techniques: temporal grouping with Gradient Surgery (GS), and Passenger Arrival Information (PAI). These enhancements were then incrementally added to the training of the other two agents. In this experiment, we evenly divide the traffic scenario into 10 temporal groups and perform gradient surgery. As illustrated in Table 3, we observed a marked enhancement in performance upon the incremental inclusion of the specified techniques, demonstrating the effectiveness of each technique. Fig. 8 graphically represents the training process of the three agents.

Furthermore, our enhanced Unified agent displayed exceptional performance when tested under other single traffic patterns, demonstrating its traffic pattern awareness. During the process of training the Unified(+GS+PAI) agent, we segmented the performance related to different patterns and compared it with the training performance of RL-Separate corresponding to each pattern. We can clearly see from Fig. 7 that during training, the ability of our model to handle the four patterns simultaneously increases and performs even better than the corresponding RL-Separate agents. Most importantly, we prepared another agent, RL-4Separate, for comparison with our Unified agent. The RL-4Separate is a combined agent that selects corresponding pre-trained four RL-Separate agents adaptively according to traffic patterns. Our Unified agent with two techniques added greatly outperforms the RL-4Separate, demonstrating that our model can be free from using an auxiliary model to predict traffic patterns while showing superior performance, thereby showcasing the high potential for real-life applications.

We have also trained and tested our model under different configurations and three well-recognized traffic profiles (CIBSE, Strakosch, and Siikonen). Additional results can be found in Appendix B.

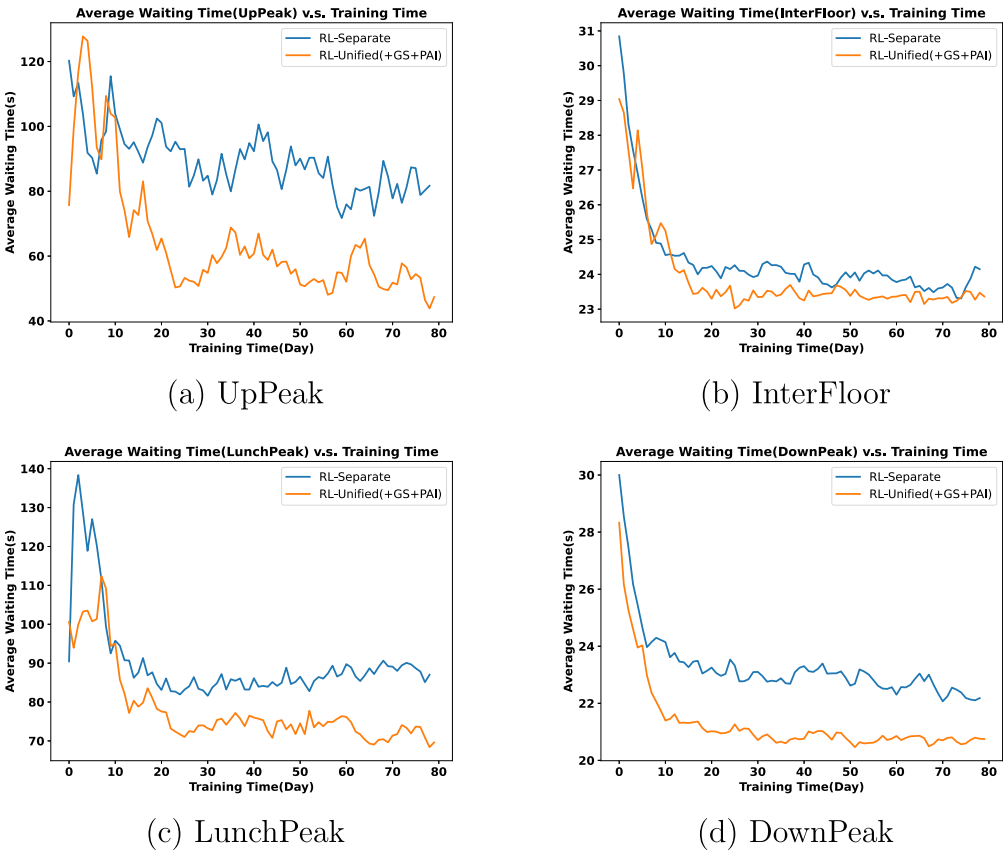


Fig. 7. AWT decreases during training time under four traffic patterns.

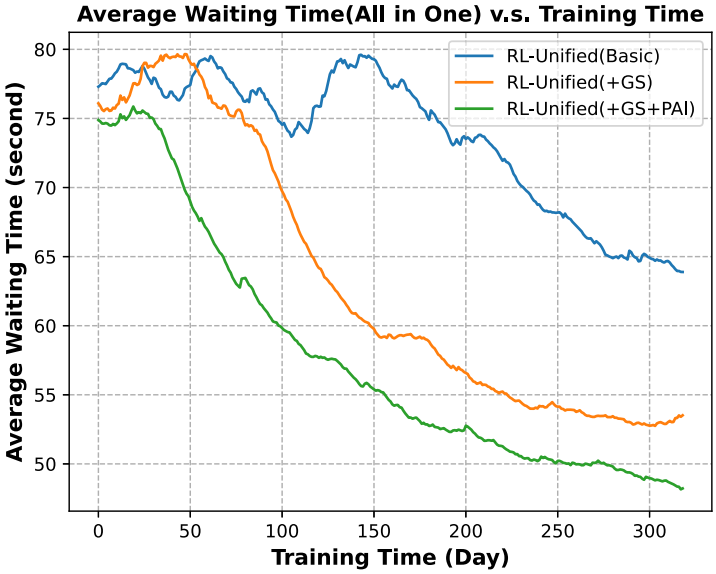


Fig. 8. Training process of four variants of RL-Unified agents.

5.2.4. Running time analysis

As shown in Table 4, GA and PSO are very slow due to the large search space of decisions. ETA, as a competitive benchmark rule, exhibits the fastest inference speed, taking only 2.59×10^{-4} seconds to make a dispatching decision. Our proposed model requires 5.64×10^{-4} seconds, which is twice the inference time of ETA. Nevertheless, we believe it is sufficiently fast for making real-time elevator dispatching decisions, confirming its practical applicability.

5.2.5. Evaluation of robustness

In this experiment, we initially trained our model with a building population of 1200 under the Hautamäki et al. [48] traffic profile and tested it across various unseen population levels and traffic profiles to assess its robustness.

As illustrated in Fig. 9, our enhanced Unified model consistently outperforms both the ETA rule and the RL-4Separate model across different population levels. Additionally, we tested the trained model

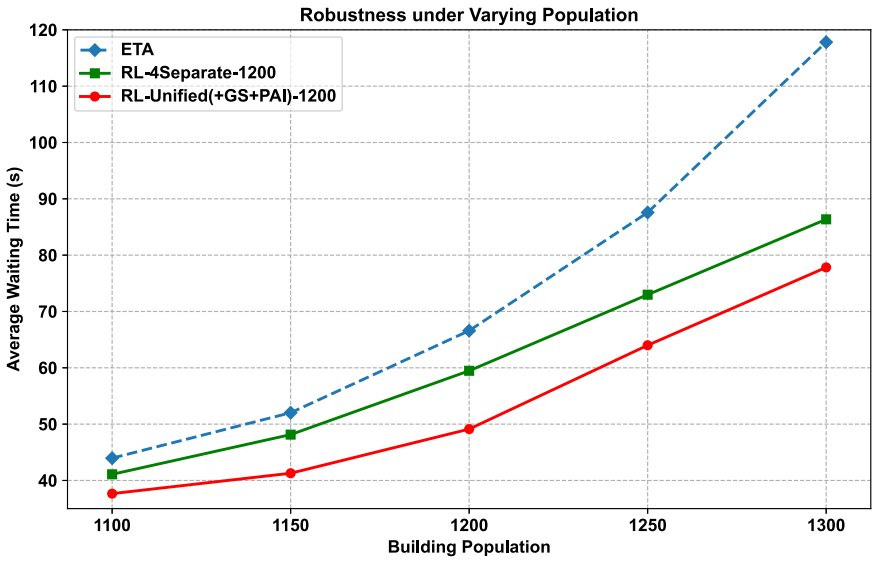


Fig. 9. Performance under different unseen population levels.

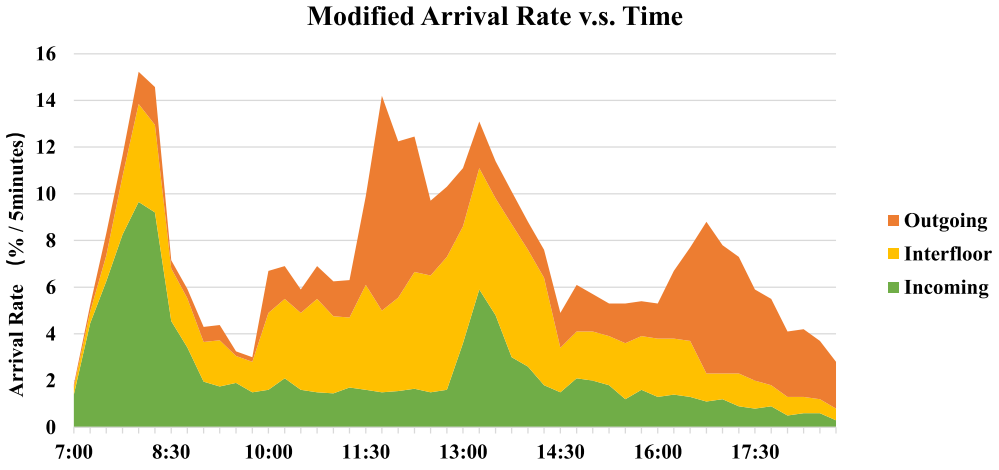


Fig. 10. Modified traffic profile for real passenger generation.

Table 4
Comparison of running time per decision.

	Running time (second)
GA	0.398
PSO	0.203
ETA	2.59×10^{-4}
Ours	5.64×10^{-4}

Table 5
Test performance on three well-recognized traffic patterns without additional training.

	CIBSE	Strakosch	Siikonen
ETA	212.13 ± 14.99	87.44 ± 6.80	36.97 ± 3.97
GA	181.73 ± 6.34	81.02 ± 3.92	39.46 ± 3.60
PSO	178.52 ± 9.11	83.86 ± 3.92	31.78 ± 2.60
4Seperate	197.52 ± 13.16	81.53 ± 5.56	33.39 ± 3.27
Unified(+GS+PAI)	171.24 ± 4.74	71.23 ± 3.45	25.63 ± 0.93

against three other unseen traffic profiles. The results are presented in Table 5, indicating that the 4Seperate model cannot robustly handle changing traffic profiles due to its poor generalizability. However, our unified model consistently outperforms other methods across these three unseen traffic profiles, where the model is not trained. This demonstrates its more robust performance in the face of changing traffic profiles. This robustness further highlights its potential for real-life deployment.

5.2.6. Benefits of combining real-time data and historical data

In the experiments conducted thus far, we assumed that the actual passenger arrival is the same as the prior knowledge derived from historical data. However, the actual passenger arrival might have some gap with the prior knowledge due to various unpredictable factors. Thus we propose to use real-time data to bridge the gap between prior knowledge and actual passenger arrival. During the testing phase, we deviated from the traffic profile illustrated in Fig. 10 for passenger generation, introducing slight modifications to the traffic file. This was done to simulate variations in actual passenger arrivals. We try to combine the prior knowledge and real-time data as the passenger arrival information. The result is demonstrated in Fig. 11. We can see that combining prior knowledge and real-time data achieves better performance than solely using prior knowledge or real-time data. The result demonstrates that both two sources of data benefit the model's traffic pattern-aware ability, enhancing the agent's decision-making.

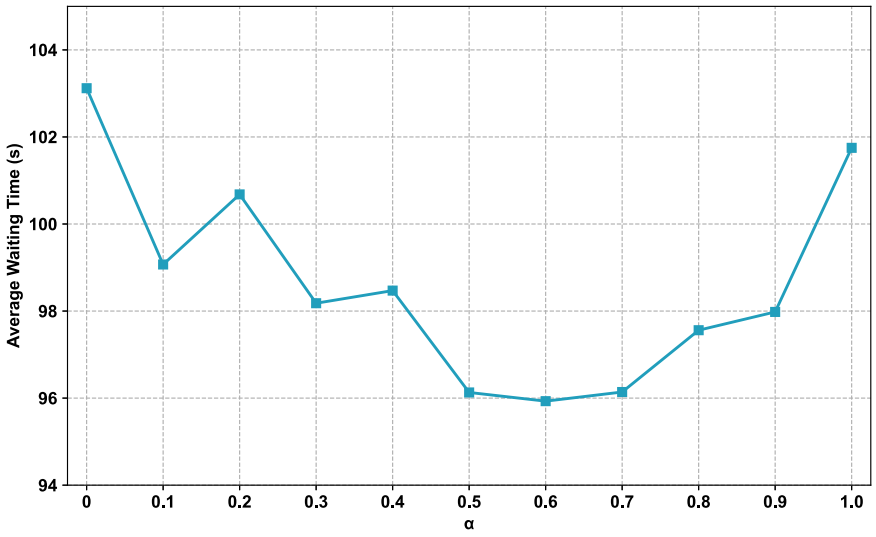


Fig. 11. Performance under different α values.

6. Conclusion

In this study, we tackle the elevator dispatching problem by incorporating an awareness of various traffic patterns into a single unified agent. In contrast to existing reinforcement learning-based elevator dispatching studies, which have largely focused on individual traffic patterns, we achieved a unified agent with the ability to make traffic pattern-aware dispatching decisions for mixed traffic patterns. Our empirical experiments validate that even though our model is trained exclusively in a mixed 'All in One' scenario, it still maintains outstanding performance across individual traffic patterns. Additionally, our unified agent consistently outperforms alternative approaches that rely on multiple, pattern-specific trained models in terms of average waiting time. Importantly, it exhibits robust performance even when faced with different traffic dynamics (e.g., unseen population levels and traffic profiles).

Future investigations may seek to integrate a more sophisticated passenger arrival prediction module, thus augmenting the model's ability to perceive and adapt to dynamic traffic conditions. Additionally, one can also expand the scope of the reward definition by incorporating considerations related to energy consumption, thus striking a balance between waiting time and energy optimization. Furthermore, our model can be adapted for use in multi-car systems, where two or more elevator cars operate within a single elevator shaft. We hope that our work could serve as a foundational reference, inspiring future efforts aimed at developing RL-based traffic pattern-aware dispatching systems that are better suited for practical, real-world deployment.

CRediT authorship contribution statement

Jiansong Wan: Writing – review & editing, Writing – original draft, Methodology, Conceptualization. **Kanghoon Lee:** Writing – review & editing, Methodology, Conceptualization. **Hayong Shin:** Supervision, Project administration, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

We thank the anonymous reviewers for their careful reading of the manuscript and their many constructive comments and suggestions, which greatly improve the quality of the paper.

This research was supported by the National Research Foundation of Korea(NRF) funded by Ministry of Science and ICT (NRF-2022M3J6A1063021 and NRF-5199990113928).

Appendix A. Partial reward calculation

Using the example illustrated in Fig. 1, we show how to calculate the PR_2.

Case1: i = 0

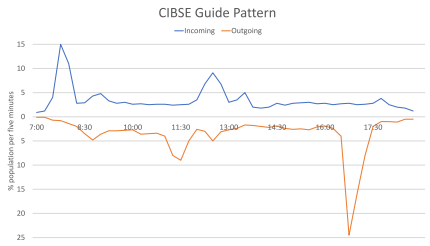
$$\begin{aligned} PR_2 &= \int_{t_1}^{t_2} e^{-\beta(\tau-t)} r_\tau^0 d\tau \\ &= \int_{t_1}^{t_2} e^{-\beta(\tau-t)} \sum_p -(\tau - t_p^{arrival})^0 d\tau \\ &= -\sum_p \int_{t_1}^{t_2} e^{-\beta(\tau-t)} 1 d\tau \\ &= -\sum_p \left(e^{-\beta(t_1-t)} \left(\frac{1}{\beta}\right) - e^{-\beta(t_2-t)} \left(\frac{1}{\beta}\right) \right) \end{aligned}$$

Case2: i = 1

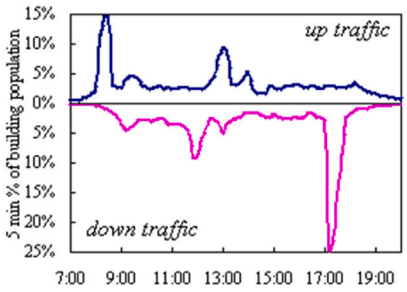
$$\begin{aligned} PR_2 &= \int_{t_1}^{t_2} e^{-\beta(\tau-t)} r_\tau^1 d\tau \\ &= \int_{t_1}^{t_2} e^{-\beta(\tau-t)} \sum_p -(\tau - t_p^{arrival}) d\tau \\ &= -\sum_p \underbrace{\int_{t_1}^{t_2} e^{-\beta(\tau-t)} (\tau - a_p^{arrival}) d\tau}_{\text{Integrate by parts}} \\ &= -\sum_p \left(e^{-\beta(t_1-t)} \left(\frac{(t_1 - t_p^{arrival})}{\beta} + \frac{1}{\beta^2} \right) \right. \\ &\quad \left. - e^{-\beta(t_2-t)} \left(\frac{(t_2 - t_p^{arrival})}{\beta} + \frac{1}{\beta^2} \right) \right) \end{aligned}$$

Case3: i = 2

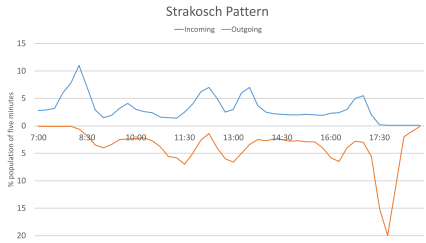
$$PR_2 = \int_{t_1}^{t_2} e^{-\beta(\tau-t)} r_\tau^2 d\tau$$



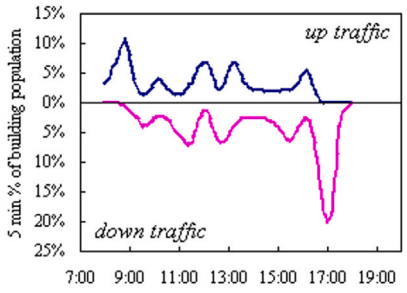
Redrawn CIBSE Pattern



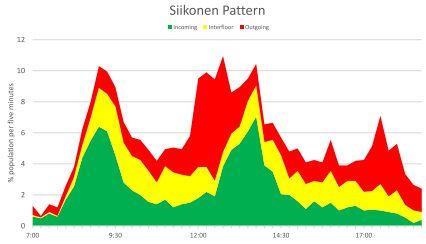
CIBSE source: Peters et al. (2000)



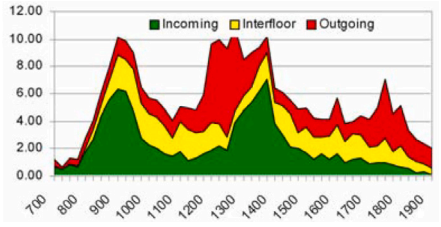
Redrawn Strakosch Pattern



Strakosch source: Peters et al. (2000)



Redrawn Siikonen Pattern



Siikonen source: Siikonen (2000)

Fig. B.1. Three well-recognized traffic patterns [50,51].

$$= \int_{t_1}^{t_2} e^{-\beta(\tau-t)} \sum_p -(\tau - t_p^{arrival})^2 d\tau$$

$$= -\sum_p \underbrace{\int_{t_1}^{t_2} e^{-\beta(\tau-t)} (\tau - t_p^{arrival})^2 d\tau}_{\text{Integrate by parts}}$$

$$= -\sum_p \left(\left[-\frac{1}{\beta} e^{-\beta(\tau-t)} \left(\tau - t_p^{arrival}\right)^2 \right]_{t_1}^{t_2} + \frac{2}{\beta} \underbrace{\int_{t_1}^{t_2} e^{-\beta(\tau-t)} \left(\tau - t_p^{arrival}\right) d\tau}_{\text{Integrate by parts}} \right)$$

$$= -\sum_p e^{-\beta(t_1-t)} \left(\frac{\left(t_1 - t_p^{arrival}\right)^2}{\beta} + \frac{2\left(t_1 - t_p^{arrival}\right)}{\beta^2} + \frac{2}{\beta^3} \right)$$

$$- \sum_p e^{-\beta(t_2-t)} \left(\frac{(t_2 - t_p^{arrival})^2}{\beta} + \frac{2(t_2 - t_p^{arrival})}{\beta^2} + \frac{2}{\beta^3} \right)$$

Appendix B. Three well-recognized traffic profiles

B.1. Redrawn traffic profiles

See Fig. B.1.

B.2. Experiments of different configurations

See Tables B.1 and B.2.

Table B.1
Performance under the setting of 12 floors, 4 elevator cars, 2.5 m/s car speed, 900 population.

Methods	CIBSE	Strakosch	Siikonen
ETA	44.09 ± 5.11	21.93 ± 1.68	15.57 ± 0.28
GA	39.79 ± 5.54	18.82 ± 0.52	15.57 ± 0.24
PSO	36.44 ± 4.22	19.37 ± 1.94	15.29 ± 0.09
Unified(+GS+PAI)	33.85 ± 1.04	15.90 ± 0.28	15.12 ± 0.08

Table B.2
Performance under the setting of 16 floors, 5 elevator cars, 3 m/s car speed, 1200 population.

Methods	CIBSE	Strakosch	Siikonen
ETA	75.89 ± 6.83	27.20 ± 2.01	15.94 ± 0.46
GA	71.70 ± 6.06	29.01 ± 3.09	15.76 ± 0.23
PSO	71.11 ± 3.55	27.99 ± 2.77	15.65 ± 0.37
Unified(+GS+PAI)	66.63 ± 4.06	19.24 ± 1.17	15.28 ± 0.23

References

[1] J.R. Fernandez, P. Cortes, A survey of elevator group control systems for vertical transportation: A look at recent literature, IEEE Control Syst. Mag. 35 (4) (2015) 38–55.

[2] M. Ruokokoski, J. Sorsa, M.L. Siikonen, H. Ehtamo, Assignment formulation for the elevator dispatching problem with destination control and its performance analysis, European J. Oper. Res. 252 (2) (2016) 397–406.

[3] P. Cortes, J. Munuzuri, A. Vazquez-Ledesma, L. Onieva, Double deck elevator group control systems using evolutionary algorithms: Interfloor and lunchpeak traffic analysis, Comput. Ind. Eng. 155 (2021) 107190.
[4] M. Ikuta, K. Takahashi, M. Inaba, Strategy selection by reinforcement learning for multi-car elevator systems, in: 2013 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2013, pp. 2479–2484.
[5] A. Rong, H. Hakonen, R. Lahdelma, Estimated Time of Arrival (ETA) Based Elevator Group Control Algorithm with More Accurate Estimation, Turku Centre for Computer Science, 2003.
[6] E.O. Tartan, C. Ciftlikli, A genetic algorithm based elevator dispatching method for waiting time optimization, IFAC-PapersOnLine 49 (3) (2016) 424–429.
[7] P. Cortés, J. Larrañeta, L. Onieva, Genetic algorithm for controllers in elevator groups: Analysis and simulation during lunchpeak traffic, Appl. Soft Comput. 4 (2) (2004) 159–174.
[8] J. Fernández, P. Cortés, J. Muñuzuri, J. Guadix, Dynamic fuzzy logic elevator group control system with relative waiting time consideration, IEEE Trans. Ind. Electron. 61 (9) (2013) 4912–4919.
[9] Q. Zong, L. Dou, W. Wang, Elevator group control scheduling approach based on multi-agent coordination, in: 2006 6th World Congress on Intelligent Control and Automation, vol. 2, IEEE, 2006, pp. 7249–7253.
[10] Q. Wei, L. Wang, Y. Liu, M.M. Polycarpou, Optimal elevator group control via deep asynchronous actor–critic learning, IEEE Trans. Neural Netw. Learn. Syst. 31 (12) (2020) 5245–5256.
[11] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[12] S. Markon, H. Kita, Y. Nishikawa, Adaptive optimal elevator group control by use of neural networks, Trans. Inst. Syst., Control Inf. Eng. 7 (12) (1994) 487–497.
[13] R.H. Crites, A.G. Barto, Elevator group control using multiple reinforcement learning agents, Mach. Learn. 33 (2) (1998) 235–262.
[14] A. Jansson, K. Uggla Lingvall, Elevator control using reinforcement learning to select strategy, 2015.
[15] S. Dong, P. Wang, K. Abbas, A survey on deep learning and its applications, Comp. Sci. Rev. 40 (2021) 100379.
[16] X. Liu, J. Huang, X. Tang, Intelligent elevator scheduling algorithm based on image recognition and voice recognition, in: International Conference on Neural Networks, Information, and Communication Engineering, Vol. 12258, NNICE, SPIE, 2022, pp. 306–312.
[17] M. Baykal-Gürsoy, Semi-Markov decision processes, in: Wiley Encyclopedia of Operations Research and Management Science, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2010.
[18] S.J. Bradtke, M.O. Duff, Reinforcement learning methods for continuous-time Markov decision, Adv. Neural Inf. Process. Syst. 7 7 (1995) 393.
[19] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[20] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, (no. 1) 2016.
[21] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, N. Freitas, Dueling network architectures for deep reinforcement learning, in: International Conference on Machine Learning, PMLR, 2016, pp. 1995–2003.
[22] T. Schaul, J. Quan, I. Antonoglou, D. Silver, Prioritized experience replay, 2015, arXiv preprint arXiv:1511.05952.
[23] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, 2015, arXiv preprint arXiv:1509.02971.
[24] S. Yang, J. Wang, Z. Xu, Real-time scheduling for distributed permutation flowshops with dynamic job arrivals using deep reinforcement learning, Adv. Eng. Inform. 54 (2022) 101776.
[25] J. Deng, S. Sierla, J. Sun, V. Vyatkin, Reinforcement learning for industrial process control: A case study in flatness control in steel industry, Comput. Ind. 143 (2022) 103748.
[26] Z. Qin, J. Tang, J. Ye, Deep reinforcement learning with applications in transportation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 3201–3202.
[27] W. Qin, Z. Zhuang, Z. Huang, H. Huang, A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem, Comput. Ind. Eng. 156 (2021) 107252.
[28] J. Xie, H. Dong, X. Zhao, A. Karcanias, Wind farm power generation control via double-network-based deep reinforcement learning, IEEE Trans. Ind. Inform. 18 (4) (2021) 2321–2330.
[29] A. Perera, P. Kamalaruban, Applications of reinforcement learning in energy systems, Renew. Sustain. Energy Rev. 137 (2021) 110618.
[30] S. Dong, Y. Xia, T. Peng, Network abnormal traffic detection model based on semi-supervised deep reinforcement learning, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4197–4212.
[31] J. Wan, H. Shin, Predictive vehicle dispatching method for overhead hoist transport systems in semiconductor fabs, Int. J. Prod. Res. 60 (10) (2022) 3063–3077.
[32] X. Zhang, M. Yan, B. Xie, H. Yang, H. Ma, An automatic real-time bus schedule redesign method based on bus arrival time prediction, Adv. Eng. Inform. 48 (2021) 101295.
[33] G. Barney, L. Al-Sharif, Elevator Traffic Handbook: Theory and Practice, Routledge, 2015.
[34] A.T. So, J. Beebe, W. Chan, S. Liu, Elevator traffic pattern recognition by artificial neural network, in: Elevator Technology 6 Proceedings of Elevcon 1995, vol. 6, (no. 1) 1995, p. 122.
[35] Y. Xu, F. Luo, Pattern recognition of traffic flows in elevator group control systems based on SVM, Chinese. J. South China Univer. Technol. (Nat. Sci.) 33 (6) (2005) 32–35.
[36] Y.G. Xu, F. Luo, Traffic pattern recognition method for novel elevator system, Kongzhi Lilun yu Yingyong/ Control Theory Appl. 22 (6) (2005) 900–904.
[37] P. Cortés, J.R. Fernández, J. Guadix, J. Muñuzuri, Fuzzy logic based controller for peak traffic detection in elevator systems, J. Comput. Theor. Nanosci. 9 (2) (2012) 310–318.
[38] J. Sorsa, H. Ehtamo, J.-M. Kuusinen, M. Ruokokoski, M.L. Siikonen, Modeling uncertain passenger arrivals in the elevator dispatching problem with destination control, Optim. Lett. 12 (2018) 171–185.
[39] J. Sorsa, M.L. Siikonen, J.M. Kuusinen, H. Hakonen, A field study and analysis of passengers arriving at lift lobbies in social groups in multi-storey office, hotel and residential buildings, Build. Serv. Eng. Res. Technol. 42 (2) (2021) 197–210.
[40] P.E. Utgoff, M.E. Connell, Real-time combinatorial optimization for elevator group dispatching, IEEE Trans. Syst., Man, Cybern.-Part A: Syst. Hum. 42 (1) (2011) 130–146.
[41] S. Wang, X. Gong, M. Song, C.Y. Fei, S. Quaadgras, J. Peng, P. Zou, J. Chen, W. Zhang, R.J. Jiao, Smart dispatching and optimal elevator group control through real-time occupancy-aware deep learning of usage patterns, Adv. Eng. Inform. 48 (2021) 101286.
[42] J. Zheng, H.C.T. Thomas, Y. HuaiBing, Traffic prediction for efficient elevator dispatching, in: TENCON 2018-2018 IEEE Region 10 Conference, IEEE, 2018, pp. 2232–2236.
[43] J. Zhang, A. Tsiligkaridis, H. Taguchi, A. Raghunathan, D. Nikovski, Transformer networks for predictive group elevator control, in: 2022 European Control Conference, ECC, IEEE, 2022, pp. 1429–1435.
[44] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, C. Finn, Gradient surgery for multi-task learning, Adv. Neural Inf. Process. Syst. 33 (2020) 5824–5836.
[45] R.S. Caporale, Elevate™ traffic analysis software (eliminating the guesswork), Elevator World 48 (6) (2000) 118–124.
[46] P. Cortés, J. Muñuzuri, L. Onieva, Design and analysis of a tool for planning and simulating dynamic vertical transport, Simulation 82 (4) (2006) 255–274.
[47] T. Miyamoto, S. Yamaguchi, MceSim: A multi-car elevator simulator, IEICE Trans. Fundam. Electron., Commun. Comput. Sci. 91 (11) (2008) 3207–3214.
[48] T. Hautamäki, et al., Multiobjective optimization model for elevator call allocation, 2021.
[49] B. Bolat, O. Altun, P. Cortés, A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems, Appl. Soft Comput. 13 (5) (2013) 2633–2642.
[50] R. Peters, P. Mehta, J. Haddon, Lift passenger traffic patterns: Applications, current knowledge and measurement, Elevator World 48 (9) (2000) 87–94.
[51] M.L. Siikonen, On traffic planning methodology, Elevator Technol. 10 (2000) 267–274.