Advanced Engineering Informatics 48 (2021) 101286

Contents lists available at ScienceDirect

Advanced Engineering Informatics

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

Smart dispatching and optimal elevator group control through real-time occupancy-aware deep learning of usage patterns

Shu Wang [a,b], Xuejian Gong [a], Mulang Song [a], Cindy Y. Fei [a], Stefan Quaadgras [a], Jianyuan Peng [a], Pan Zou [a], Jerred Chen [a], Wei Zhang [a,c], Roger J. Jiao [a,*]

[a] School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[b] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[c] Dept. of Mechanical Electrical Engineering, Xiamen University, China

ARTICLE INFO

Keywords:
Elevator group control
Occupancy awareness
Elevator dispatching
Decision modeling
Real-time dispatching

ABSTRACT

Passengers spend much time on elevator journeys in high-rise buildings every day, in which unnecessary stops caused by lack of cab capacity take up a certain proportion of the journey time. This study proposes real-time occupancy-aware smart dispatching to avoid pick-up failure by introducing the occupancy information that reflects elevator capacity into the optimization model, thus improving dispatching performance. Occupancy awareness is firstly implemented with deep learning-based object detection to provide estimated capacity. Traffic pattern recognition is implemented with time series analysis and fuzzy logic. Case-based reasoning is applied to recognize the current usage pattern and to deploy specific dispatching strategies. A prioritized A* search model is built to solve dispatching optimization with occupancy information. Discrete event simulation is conducted with Simio and MATLAB to validate the proposed dispatching model.

1. Introduction

Elevators serve as a major vertical transportation method in modern buildings, meeting the transport demands of building occupants for their daily work and living. In high-rise buildings that accommodate a large number of people, passengers spend much time on elevator travel, especially during the rush hour. According to the statistics, there are over 1.1 billion elevators in the United States, and each elevator supports around 3 million passenger trips per year [1]. Thus, there are growing demands for improving elevator dispatching performance. The performance can be evaluated from two perspectives: the quantity of service and the quality of service [2]. The former refers to the passenger handling capacity during peak traffic periods, while the latter includes multiple indicators, such as passenger waiting time and travel time, power consumption, long waiting time percentage, and so on.

To improve transportation efficiency and reduce the operation costs, elevator group control (EGC) techniques are widely studied [3]. EGC refers to managing multiple elevator cabs in a building as a group. It is achieved by allocating hall calls to the most suitable cab to optimize one or multiple criteria, such as average waiting time, round trip time, and building owners' preference, like power consumption [2,4]. Three problems concerned with EGC are what information to use for optimal dispatching, which dispatching optimization algorithm to apply, and how to determine the traffic pattern or the usage pattern to decide the dispatching strategies. An efficient dispatching system needs the support of auxiliary hardware and dispatching algorithms and strategies.

Dispatching information is a set of parameters that describes the current operating states and is the input of dispatching optimization problems. Conventional dispatching information includes the cab position and direction, existing hall calls, registered cab calls, and so on [5]. If more useful information is used, more costs will be for hardware setup and computing time, but better dispatching performance can be obtained.

Dispatching optimization aims at finding the most proper hall call allocations that optimize the selected criteria with the given information. The dispatching optimization problem is essentially a combinatorial optimization problem and falls into the NP-complete problems [6]. Given the characteristics of NP-complete problems, prevailing

* Corresponding author.
E-mail addresses: swang737@gatech.edu (S. Wang), xgong45@gatech.edu (X. Gong), msong84@gatech.edu (M. Song), yfei38@gatech.edu (C.Y. Fei), stefanquaadgras@gatech.edu (S. Quaadgras), pjy@gatech.edu (J. Peng), zoupzp123@gatech.edu (P. Zou), jchen788@gatech.edu (J. Chen), zhangw@stu.xmu.edu.cn (W. Zhang), rjiao@gatech.edu (R.J. Jiao).

https://doi.org/10.1016/j.aei.2021.101286
Received 17 September 2020; Received in revised form 10 February 2021; Accepted 16 March 2021
Available online 11 April 2021
1474-0346/© 2021 Elsevier Ltd. All rights reserved.

optimization algorithms are heuristics-based, which can find a relatively good solution in a short time. Other algorithms like fuzzy logic and neural networks are also used [2,7].

Dispatching strategies regulate elevator operations in different traffic patterns. Based on the forecasted traffic flow, three typical traffic patterns can be clustered: up-peak, down-peak, and inter-floor [8]. Different strategies can be deployed to fit the features of the current traffic pattern, like zoning and sectoring techniques that allocate working regions to different cabs [5,9–10] or specifying the number of cabs for serving upward and downward hall calls.

One problem most prevailing dispatching systems have is that their allocations can cause wasted stops due to the neglect of cab capacity constraints, thus influencing the passenger journey time. The journey time is defined as the time interval between a passenger registers a hall call and he arrives at the destination floor [11]. Dispatching systems usually cease to allocate new hall calls to the overweighed cabs but keep assigning tasks to the cabs without capacity for new passengers. If a cab is not overweighed but has no capacity when it conducts a hall call task, outside passengers cannot enter it and need to wait for another cab. Thus, extra time is spent during this stop.

Cabs' lack of capacity can take place in different scenarios. Sometimes, it is because the cab is fully occupied, and there is no room to accommodate new passengers. It can also be caused by large obstacles that block the entrance. For instance, passengers bring large objects with low mobility into the cab and place them at the entrance, like large and heavy packages. Apart from the above two scenarios, special elevator usages can influence the cab capacity for new passengers. An example is the transfer of a patient that needs first-aid in a high building, and the cab should serve this task as a priority and stop answering other hall calls. Because traditional dispatching systems do not consider occupancy information, the dispatching is made based on the false assumption that the capacity is always enough to answer a hall call. However, capacity is a resource during dispatching, and if the capacity constraint is not satisfied, dispatching performance can be affected.

To avoid pick-up failure caused by the lack of capacity, this study proposes and develops an occupancy information analysis model to implement real-time occupancy awareness for optimal dispatching. The occupancy awareness can be interpreted as context awareness and situational awareness. The former aims to obtain object information in a cab and real-time passenger traffic data, and the latter is to provide the estimated capacity of a cab, the current traffic pattern, and the usage pattern. It uses image data collected from security cameras as inputs and applies deep learning, time series analysis, and intelligent reasoning techniques. Conventional implementation of elevator occupancy awareness mainly relies on sensors, like weight sensors and safety beams [8]. Compared to the proposed method, the sensor-based methods are not only costly but have limitations in precision and functionalities.

To model a dispatching system with occupancy awareness, there are three technical challenges, including the implementation of context awareness and situational awareness, and real-time dispatching optimization with occupancy information.

(1). *Context awareness* aims to track the number of passengers in a cab at a real-time scale and should recognize the category of the other objects and their number. Thus, low computing costs and high accuracy are required for object detection. Because detection results are used as model inputs, fast implementation leaves more time for dispatching. Meanwhile, detection accuracy is important for the accuracy of traffic flow forecast and capacity estimation.

(2). *Situational awareness* is to interpret the latent story of the elevator usage, including estimating the capacity for new passengers and recognizing the current traffic pattern and the usage pattern. While usage pattern recognition can be implemented with case-based reasoning by searching specified objects, capacity estimation should consider multiple factors instead of simply calculating the empty area, and the major challenge is to reflect the influence of non-human objects on capacity, including object areas, mobility, positions, and so on. Also, traffic pattern recognition should firstly obtain and forecast traffic data using the number of transferred passengers during stops, and then address the fuzziness during pattern classification.

(3). *Intelligent dispatching optimization* is to develop the optimization model for allocating hall calls based on specified criteria. Two problems need to be solved. One problem is that the model should generate a high-quality solution at a real-time scale. Because dispatching is made during passengers' waiting, there is a trade-off between computing time and solution quality. When the problem scale is large, methods should be applied to restrict the computing time. The other problem is how to fit occupancy information into the optimization model to reflect the influence of cab capacity on hall call allocation.

In this regard, this paper proposes real-time occupancy-aware smart dispatching for optimal elevator group control and presents its decision modeling. To implement the discussed occupancy awareness, the dispatching system estimates cab capacity with convolutional neural network (CNN) based object detection and introduces it into the dispatching optimization model by modifying the prioritized A* search algorithm. Meanwhile, by using object detection results, traffic data can be obtained, and the current usage pattern can be recognized with case-based reasoning. Using existing time series analysis and fuzzy logic methods, traffic pattern recognition can also be implemented.

The rest of this paper is organized as follows. Section 2 reviews the previous work on EGC techniques, occupancy awareness, real-time dispatching optimization algorithms, and elevator traffic pattern recognition. Section 3 discusses analysis and design of the real-time occupancy-aware dispatching system. The implementation of Mask R-CNN-based context awareness is described in Section 4. Capacity estimation with 3D space consistency modeling and usage pattern recognition are presented in Section 5. Adaptive response rate exponential smoothing (ARRES) based traffic flow forecast and traffic pattern recognition with fuzzy logic are introduced in Section 6. The prioritized A* search optimal dispatching model with occupancy information is discussed in Section 7. In Section 8, discrete event simulation experiments with Simio and MATLAB are conducted to validate the proposed dispatching model. Conclusions are made in Section 9.

2. Related work

2.1. EGC techniques

In the past decades, many EGC technologies have been developed to improve dispatching performance. In 2002, ThyssenKrupp proposes and implements the multi-cab elevator system, where several cabs are placed in a single elevator shaft to reach a greater passenger transportation capacity [12]. KONE proposes a destination control system in 2006, which collects passenger destinations before they enter a cab, and those with the same destination will be assigned to the same group. This method reduces the number of stops during the travel [13]. In 2008, the double-deck elevator system is proposed, in which two cabs are connected vertically in a single shaft to increase the handling capacity [14]. With the image processing and analysis techniques developing, security cameras are also used in EGC to assist dispatching. Toshiba in 2014 proposes an energy-saving algorithm for EGC, which uses cameras to detect the number of passengers waiting for each hall call and analyzes its impact on dispatching performance to decide the number of passengers to be served by different cabs [15]. Similarly, by detecting the traffic demand from the hall area, there is also a study using an expert system that integrates Bayesian networks with fuzzy rules to conduct the decision-making process [16]. This work is further studied by [17], where the effective numbers of passengers and items are estimated with Bayesian networks and then used in the nearest car algorithm.

2.2. Occupancy awareness

Occupancy awareness studies the impacts of occupants' presence and behavior on building operations in the context of smart environments, like space heating and cooling, lighting control, and so on [18]. To assist building energy and comfort management systems with operation decision making, data collected from a sensor-rich environment are analyzed to recognize the occupancy behavior pattern by distinguishable features [19].

In this study, occupancy awareness is to obtain and analyze the occupancy information inside the elevator cab to improve dispatching performance. Conventional methods for occupancy awareness can be divided into sensor-based and image-based methods. The former implementation analyzes data from digital weight sensors, safety beams, and IR sensors [2,8]. For example, the weight sensor-based head-counting detects passengers' in and out during a stop with the total weight change, and IR sensor-based cab capacity estimation roughly calculates the occupied area using the measured distance between the ceiling and the object top surface. There is also a study on using 3D semiconductor sensors for detecting passengers and objects to classify the current running state [20]. The image-based implementation uses security cameras to collect data and applies machine vision or CNN to detect passengers and other objects [5]. CNN-based object detection can identify and localize passengers and other objects simultaneously with high speed and accuracy [21]. Since R-CNN was proposed in 2014, many CNN-based object detectors have been developed and applied. For example, Faster R-CNN can be used to identify defects on product surfaces for inspection [22]. And Mask R-CNN is a flexible two-stage method that implements pixel-level instance segmentation based on the Faster R-CNN structure [23] and therefore has better estimation for object localization.

2.3. Real-time dispatching optimization

Because the elevator dispatching problem is an NP-complete problem, the computing time increases exponentially as the problem scale grows. Consequently, prevailing optimization methods are heuristics, but artifitial intelligence (AI) based methods are also used.

For heuristics, an objective function based on the chosen criteria should be determined, including average waiting time, average journey time, the variation in passenger waiting time, round-trip time, power consumption, and so on [5]. Then algorithms like genetic algorithms and particle-swarm optimization generate initial solutions randomly and keep improving the current best solution [24–25]. Heuristics usually find a good but not the optimal solution within an acceptable time. As for artificial intelligence, commonly used methods include the expert system, fuzzy logic, and neural networks. Expert system dispatching uses knowledge and a set of production rules as well as fuzzy rules to allocate hall calls [26]. Fuzzy logic dispatching usually has a fast response but follows a nondynamic dispatch [2]. Fuzzy logic rules, sometimes combined with the reasoning process to select fuzzy rules, are used to evaluate the suitability of each cab for different hall call tasks based on multiple criteria [27–28]. The fuzzy neural network-based method uses the neural network model to predict average waiting time based on the dispatching proposal, and applies fuzzy control to adapt to sudden changes in the traffic flow [29].

2.4. Traffic pattern recognition

Traffic pattern recognition is to classify the current traffic into a predefined pattern, in which different control strategies are deployed. For example, conventional up-peak control ignores downward hall calls or only serves them occasionally [30]. In-time detection of the pattern change can help the system adjust the control strategy and avoid the buildup of long queues in front of the entrance. The recognition of the current traffic pattern is usually based on the forecast of traffic flow, and typical forecast methods include time series models, SVM, and neural networks [8]. Compared to SVM, the neural network has the overfitting problem and cannot escape the local minimum [31]. Fuzzy logic is especially suitable to determine the current pattern, where two types of membership functions are designed to determine the level of traffic components and the traffic intensity, with which fuzzy rules are applied for the recognition [32].

3. System analysis and design for real-time occupancy-aware smart dispatching

This section presents the details and functional analysis of the proposed occupancy information analysis model. System architecture and the workflow of occupancy-aware dispatching are also elaborated.

3.1. Occupancy information analysis model and its functional analysis

The occupancy information analysis model implements both context and situational awareness to provide useful information that can be used during dispatching. The information hierarchy is shown in Fig. 1.

The first level is the hardware level, which collects images inside the cab with security cameras. This level aims at recording the objects in the scene and provides image data for further analysis. The second level is to extract context information. Object detection is implemented to aggregate the number and category of the objects in the cab. The number of passengers in each cab during operation is recorded separately. The third level is to extract situational information by refining the context information as the estimated capacity, the current elevator traffic pattern, and the usage pattern.

The functional analysis diagram (IDEF0) of the model is shown in Fig. 2. The whole process is completed on a microcomputer (M_2). Firstly, object detection (A_1) is performed with a pre-trained Mask R-CNN model (C_2) using in-cab images (I_1) collected from the security camera (M_1). The detection is executed with the predefined frequency or triggers (C_1) to reach a real-time scale and to detect transferred passengers during a stop. The detection result that aggregates the number and category of objects is used for situational awareness, including capacity estimation (A_2), traffic analysis (A3), and usage pattern recognition (A_4). Capacity is estimated based on the occupancy area contributed by each object and their mobility, which are predefined and recorded in the object occupancy information table (I_2). The estimation method is to model the consistency level of the 3D space with a set of predefined rules (C_3). Traffic analysis (A_3) is to analyze the passenger arrival data in different traffic and recognize the current traffic pattern, which requires the time series analysis model (C_4) for passenger arrival rate forecast and fuzzy

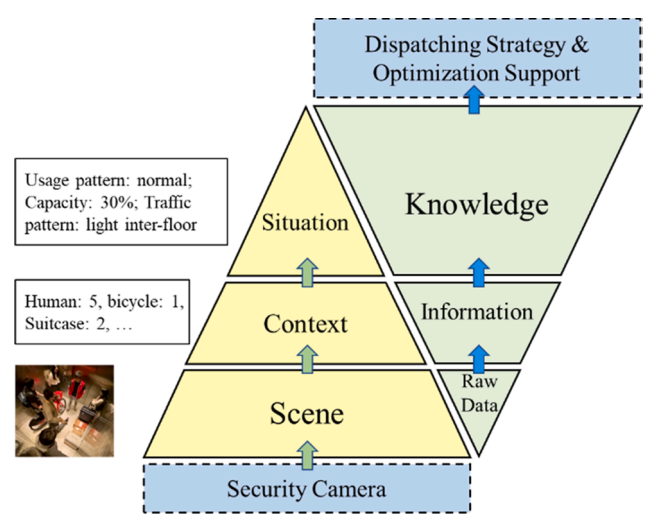


Fig. 1. Occupancy information hierarchy.

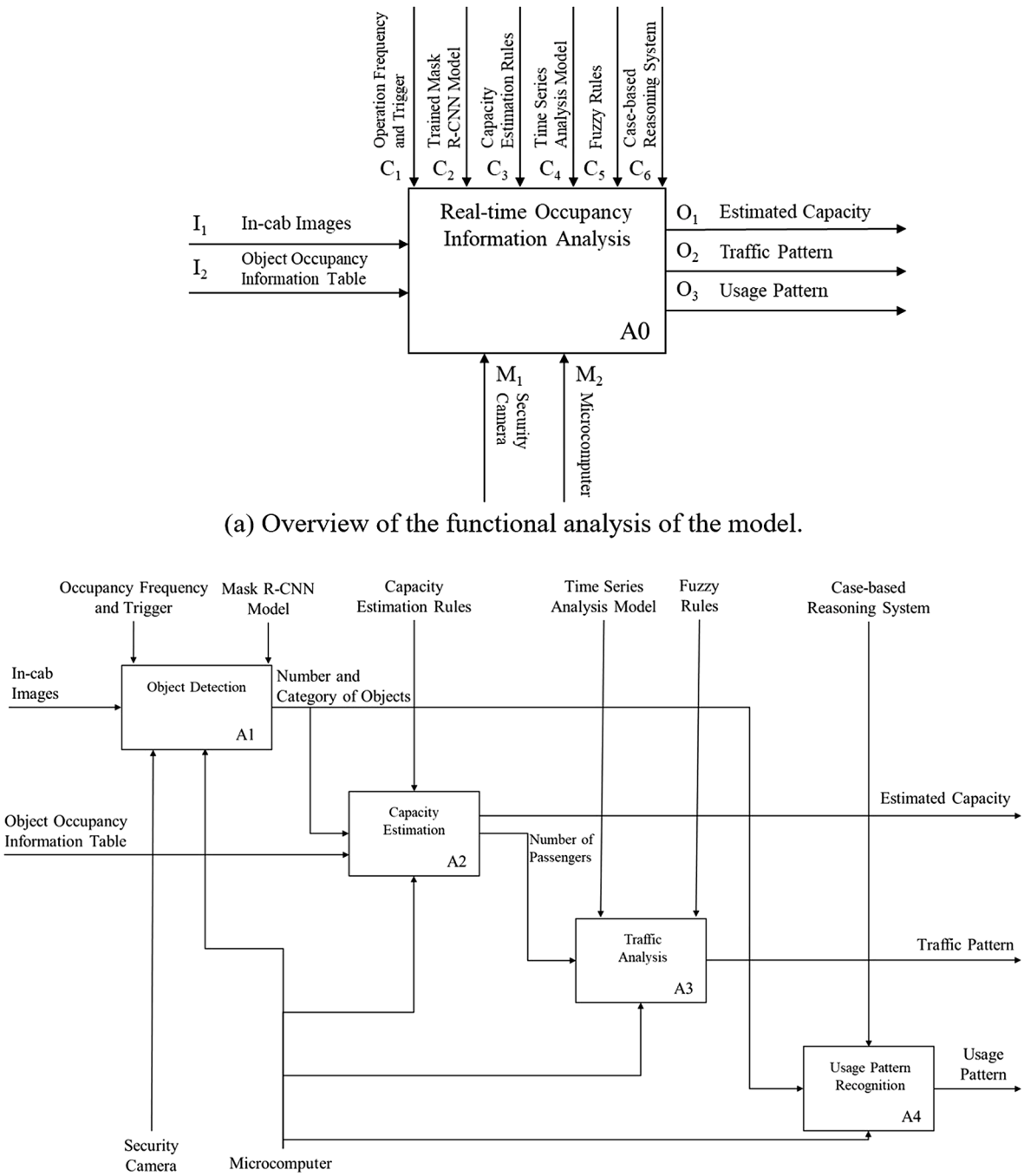


Fig. 2. IDEF0 diagram of the occupancy information analysis model.

rules (C_5) to analyze the traffic intensity and traffic components. And the usage pattern is determined by searching the most similar history case and modifies the corresponding control strategy using case-based reasoning (C_6).

3.2. System architecture and workflow of real-time occupancy-aware dispatching

The system architecture of occupancy-aware dispatching is shown in Fig. 3. It comprises of five modules: the user, the user interface, the database, the knowledge base, and the model base. The user is usually the building manager who operates the system. The manager can read operation data and change dispatching settings through the user interface, and the settings include optimization criteria, the algorithm parameter setting, and dispatching strategies in different traffic patterns. The database stores the data generated during the operation, including the traffic data, registered hall calls, cab call tasks, elevator running states, and so on. The model base consists of mathematical and analytical models for optimal dispatching: the occupancy information analysis model, the traffic analysis model for traffic flow forecast and traffic pattern recognition, the dispatching strategy model that determines the strategies in different traffic patterns, and the real-time dispatching optimization model for optimal hall call allocations. The knowledge base keeps the predefined rules used during dispatching,

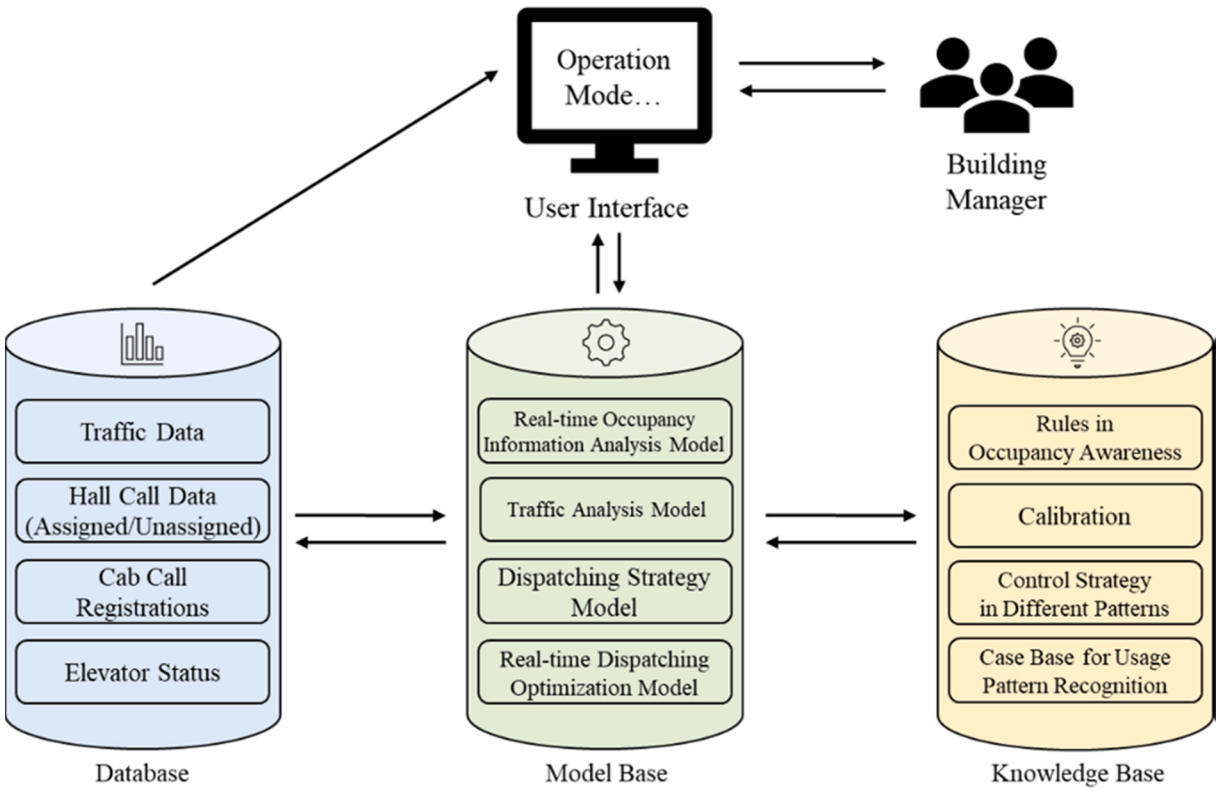


Fig. 3. Architecture of the real-time occupancy-aware smart dispatching.

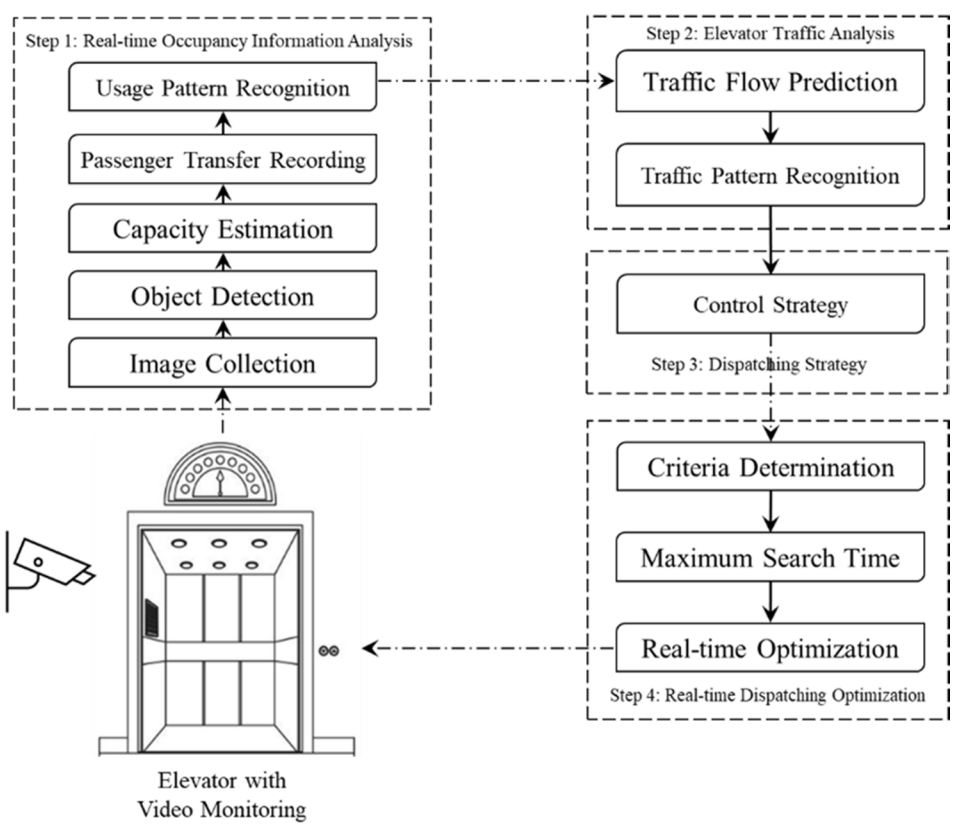


Fig. 4. The workflow of occupancy-aware smart dispatching.

such as the case base that includes history cases of special elevator usages and corresponding dispatching rules, algorithm parameter settings in different operation modes, and control strategies in different traffic patterns.

The workflow of occupancy-aware dispatching is shown in Fig. 4. Firstly, the security camera collects real-time image data and passes the images to the Mask R-CNN model for object detection. Real-time data of passenger numbers, as well as the category and number of other objects, are obtained. Then capacity estimation is conducted with the information and the predefined occupancy information table, and usage pattern recognition is implemented with case-based reasoning. Based on the number of passengers in the cab, the number of transferred passengers during each stop is obtained and the passenger arrival rates in different traffic are computed. Traffic flow forecast is conducted with passenger traffic data using the ARRES model, and fuzzy logic is applied to traffic pattern recognition to analyze the forecasted passenger arrival rates. The control strategy is then chosen according to the traffic pattern. Finally, the optimization criteria and related optimization constraints are set. The modified prioritized A* search algorithm with occupancy information is run to obtain the optimal dispatching result.

4. Object detection with Mask R-CNN for context awareness

4.1. Context awareness objective

The proposed occupancy awareness provides the dispatching system with estimated cab capacity, the current traffic pattern, and the usage pattern. To make it at a real-time scale, fast and accurate implementation of context awareness is required. Thus, a fast and reliable algorithm that can identify the category and number of objects in an image is needed. This is in accordance with the objective of object detection in the computer vision field: to precisely estimate the categories and locations of objects contained in an image from some known classes [33]. In this study, Mask R-CNN is chosen to perform this task.

4.2. Data set construction

Data set construction is an important step in model development. Because most detection targets are everyday objects, like humans, backpacks, and suitcases, an existing dataset can be used to save time from image collection and labeling. In this application, the Microsoft COCO dataset is used, which contains 320,000 labeled images and 90 object categories for object detection [34]. As cardboard boxes are frequently seen in an elevator, corresponding images are also collected and labeled as a new class in the dataset. LabelMe is used to annotate new data. For the convenience of training and testing, the image size is scaled to 640 * 480. The final dataset consists of cardboard boxes and 13 categories of objects from the COCO dataset: backpack, bicycle, cat, chair, dog, handbag, person, snowboard, sports ball, suitcase, surfboard, tennis racket, and umbrella. 2780 images are selected, where 1668 are in the training set and 834 for validation, and the remaining 278 images are in the testing set.

4.3. Mask R-CNN architecture

Mask R-CNN is a simple and flexible model for instance segmentation, which can generate a segmentation mask for each detected instance. The model is based on the framework of Faster R-CNN, which has a region proposal network (RPN) to generate the region of interest (RoI) with features extracted from the backbone structure. The backbone consists of a bottom-up pathway and a top–bottom pathway. The former is usually a CNN for feature extraction, like ResNet or VGG. The latter is a feature pyramid network that generates semantic features at various resolution scales.

The difference between Faster R-CNN and Mask R-CNN is that Mask R-CNN adds an RoI Align layer and a branch for predicting high-quality segmentation masks. Compared to the Faster R-CNN model whose recognition is after segmentation, the mask branch is in parallel with the branches for classification and bounding box prediction, adding only a small computing cost. Because classification, bounding boxes, and masks are computed simultaneously, the computation time becomes smaller. Furthermore, the RoI Align layer solves the misalignment between the RoI and extracted features introduced by quantization in RoIPool operations, making it possible to predict masks at the pixel scale. Fig. 5 presents the architecture of Mask R-CNN.

4.4. Model training with transfer learning

Transfer learning is to initiate the network parameters with a pre-trained model, because the first several convolutional blocks extract the general features that can be used for other classification tasks. Thus, the parameters in these layers can be frozen, and only parameters in the classification layer and the last several layers should be trained. In this way, the training time is largely saved, and less data are needed to develop a new classification model [35].

In this study, ResNet-50 is chosen for backbone feature extraction, and transfer learning is implemented by firstly freezing the backbone layers and only training the head layers. Then the entire model is fine-tuned. An example of object detection in an elevator cab is presented in Fig. 6.

5. 3D space consistency modeling and case-based reasoning for elevator situation comprehension

To improve dispatching performance, situational information is extracted to equip the optimization model with the comprehension of the current environment. In elevator dispatching, three types of situational information can be useful for obtaining high-quality dispatching solutions: the estimated capacity, the cab usage pattern, and the traffic pattern.

Capacity estimation is to understand the space utilization of each cab. It enables the optimization model to consider the capacity constraints during dispatching in heavy traffic, so the hall call allocation can be more balanced to avoid pick-up failure caused by the lack of capacity. The objective of usage pattern recognition is to recognize the transport objective of the trip. If the cab is in a special usage, it may deploy certain rules and perform only the specified tasks. Traffic patterns indicate the major directions of passenger trips and the overall traffic intensity. Traffic pattern recognition aims to deploy certain dispatching strategies to fit the characteristics of the current traffic flow, and the strategies are usually a set of hall call allocation rules. In this section, the implementation of capacity estimation and usage pattern recognition is introduced, while the details of traffic pattern recognition are presented in Section 6.

5.1. 3D space consistency modeling with object mobility for capacity estimation

Cab capacity refers to the area that can accommodate new passengers. It is different from the unoccupied space, because sometimes passengers cannot find a path when large obstacles get in the way. Considering the influence of object mobility on the capacity, this study proposes to utilize the object occupancy area, S, and the object mobility, P, to model the 3D space consistency level, L, and then to estimate capacity with L.

The object occupancy area is the area that the object actually occupies and is estimated based on the crowd density approximation method by fully occupying the cab with the object [4]. For example, the ground of a cab is measured as $2m \times 2m$. Fig. 7 shows the static crowd density from 1 person/m^2 to 5 people/m^2. According to the crowd density and risk analysis control, 5 people/m^2 is the upper limit for

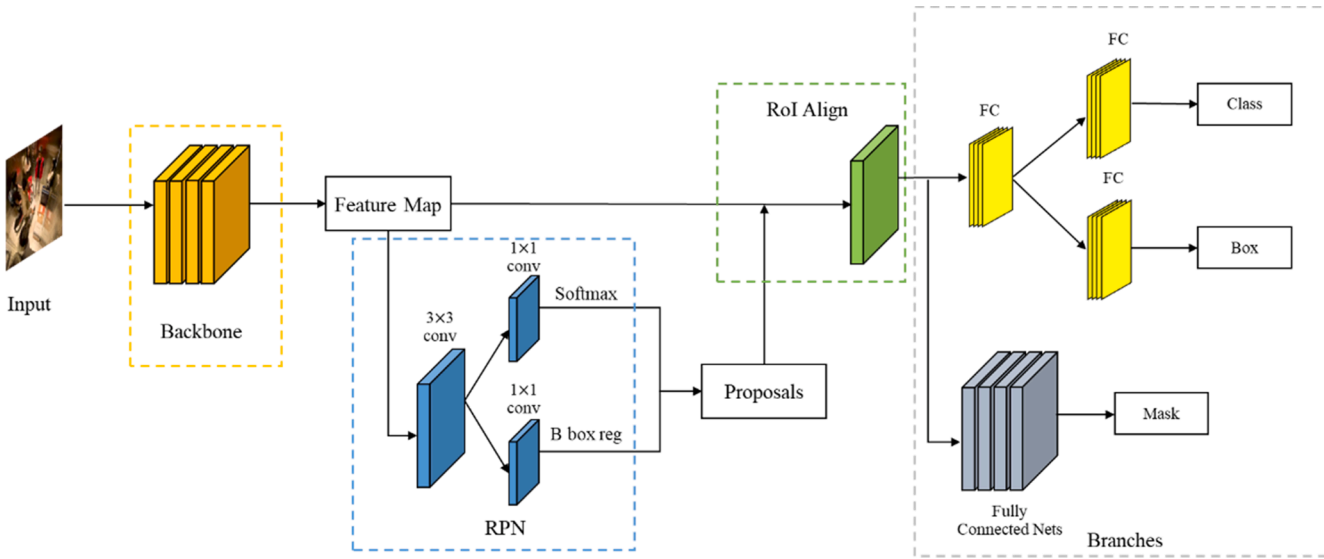


Fig. 5. Mask R-CNN architecture.

Fig. 6. Object detection in the elevator cab.

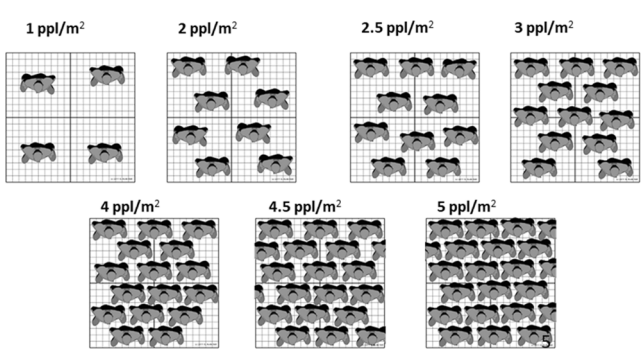


Fig. 7. Crowd density estimation.

standing spaces. Therefore, the maximum capacity for humans is 20, and the object occupancy area is 0.2 sqm.

The object mobility is defined as a categorical variable that describes how easily it can be moved. The mobility has three levels: low, medium, and high. If an object is too large or heavy to move, or it is not supposed to move much in the cab, its mobility is defined as low. For example, the mobility of a wheelchair is low, because crowdedness should be avoided when the disabled are in the cab.

The 3D space consistency level is defined to describe the ability of a cab to make room for new passengers, which can be regarded as the conversion efficiency between space and capacity. The object position factor should be considered. For example, if all the objects in the cab have high mobility, it is easier to reach a high space utilization rate, because the position of passengers and other objects can be adjusted easily. But if an object with low mobility is placed near the cab entrance, it will be inconvenient for passengers to enter it. However, given that the relationship among capacity, object mobility, and position is ambiguous to model, and that the object position obtained from Mask R-CNN is influenced by the camera angle, this study uses the mobility factor to reflect the effect of object position on capacity. Table 1 presents part of the object occupancy information used in this study.

In this study, the mobility value P corresponds to the predefined mobility level of each object. P is an ordinal variable, and a larger value means lower mobility:

$$P = \begin{cases} 1, \text{level} = \text{high} \\ 2, \text{level} = \text{medium} \\ 3, \text{level} = \text{low} \end{cases} \tag{1}$$

The space consistency level is modeled as the weighted average of the mobility P of the objects in the cab, with the object occupancy area S being the weight:

$$L = \left(\sum_{i=1}^{N} S_i \cdot P_i \Big/ \sum_{i=1}^{N} S_i \right) \tag{2}$$

where N is the number of detected objects. If more area is occupied by objects with low mobility, the space consistency level will be higher, making the capacity smaller than the empty area. In this sense, the estimated capacity C is calculated as the empty area divided by the space consistency level:

Table 1
Object occupancy information.

Object	Mobility	Occupancy Area	Occupancy Percentage
Human	High	0.2 sqm	5%
Backpack	High	0.02 sqm	0.5%
Suitcase	High	0.2 sqm	5%
Scooter	Medium	0.214 sqm	6%
Cardboard Box	Medium	0.3 sqm	7.5%
Wheelchair	Low	1 sqm	25%
Bicycle	Low	0.5 sqm	12.5%

$$C = \frac{1}{L} \cdot \left(S_{\text{cab}} - \sum_{i=1}^{N} S_i \right) \tag{3}$$

where S_{cab} is the ground area of the cab. If all objects have high mobility, the capacity will equal the empty area. Otherwise, it is smaller.

5.2. Case-based reasoning for elevator usage pattern recognition

Because elevators can be in special usages during operations, timely detection of a special usage can help the dispatching system adjust its dispatching rule and bring convenience to passengers. For example, if an elevator cab is often used to transport freight, a corresponding usage pattern can be set. If specific freight carriers or devices are detected, the usage will be changed. In this scenario, a special strategy or rule can be used, like to clear the other hall call tasks until the current task is finished. Different dispatching strategies can be set to satisfy the demands of special elevator usages from the management perspective.

In this application, case-based reasoning is used to implement usage pattern recognition. There are five steps: case representation and indexing, new case creation, case retrieval, case reuse or case adaptation, and case retention.

Case representation and indexing are the preparation work for further operations. Because the objective is to detect special usages, the number and type of key items can be used to describe a case. For example, stretchers and cleaning carts are set as the target object for the first-aid usage and cleaning tool transport. And the number of passengers is also of interest. Then $< c_1, c_2, c_3, ID, ans >$ can be used to describe a case, where c_1 and c_2 are Boolean variables (true or false) representing the existence of stretchers and cleaning carts, and c_3 is a numeric variable representing the number of passengers in the cab. *ID* is the index of a case, and *ans* is its recorded solution. To ease the case retrieval process, the indexing can use the usage as the case category and a number that represents the sequence of that case in this category. Table 2 presents two cases for the transport of a cleaning cart and freight.

Case retrieval aims to find similar cases in the case library, and similarity measuring between the new case and old cases is needed. A conventional approach is to use the nearest neighbor matching with the following equation:

$$\frac{\sum_{i=1}^{n} w_i * similarity(c_i^{new}, c_i^{old})}{\sum_{i=1}^{n} w_i} \tag{4}$$

where c_i is the case feature, and w_i is the weight of the feature. Because the usage detection is largely determined by the existence of specific items, the corresponding feature variable is either Boolean or integral, making the similarity measuring easier.

The objective of case adaptation or reuse is to find the solution to the current case. Based on different elevator usages and detected items, there can be several types of solutions. For example, if the elevator is used for first aid, the current hall call and cab call assignment will be eliminated, and no new hall call will be allocated to this cab until this task is performed. If the elevator is used for cleaning cart transport, it will keep its current cab call assignment, but whether to execute existing hall call tasks can be decided by other information, like the floor difference between the cart destination and its starting floor. Production rules can be used to evaluate the case similarity and decide whether to modify or reuse the original solution. After the solution is found, the solved new case can be stored in the case library for future reference.

Table 2
Examples of case representation.

(a) Cleaning cart transport
ID: Cleaning_003
Cleaning Cart: True; Freight: False; Psg_Number: 4; Capacity: 30%; Direction: upward
Solution: The cab becomes unavailable for new calls till the cart leaves.
(b) Freight transport
ID: FreightTransport_015
Cleaning Cart: False; Freight: True; Psg_Number: 0; Capacity: 30%; Direction: downward
Solution: Clear current hall call tasks and the cab becomes unavailable for new calls till the freight leaves.

6. Traffic pattern recognition with ARRES and fuzzy logic

Elevator traffic can be categorized into three components based on the travel purpose: incoming traffic is the ascending travel from the main floor; outgoing traffic is the descending travel to the main floor; inter-floor traffic is the travel from an inside floor to another inside floor. Traffic flow analysis is to forecast the passenger arrival rates of three traffic components. Then the forecast is used to recognize the current traffic pattern with fuzzy logic.

The conventional method has three steps: passenger traffic data analysis, traffic flow forecast, and traffic pattern recognition.

6.1. Passenger traffic data analysis

To determine the passenger arrival rates in different traffic, the data that record the number of passengers in each cab should be transformed to the number of passengers in different traffic.

Context awareness provides the number of passengers in a cab at a real-time scale. Considering that passengers only enter or exit the cab during a stop, a frequency should be determined to capture the number of passengers in and out. Assume passengers do not enter the cab until all the passengers whose destination is at this floor exit. High-frequency object detection can have higher accuracy of transferred passengers. Let n_0, n_{min}, n_1 be the number of passengers before a stop, the minimum number of passengers during a stop, and the number of passengers after a stop. The number of passengers exiting the cab is $n_0 - n_{min}$, while the number of new passengers is $n_1 - n_{min}$.

Then the number of transferred passengers in different traffic is obtained. For incoming traffic, the number of passengers who enter the cab on the main floor is recorded. For inter-floor traffic, if the cab moves upwards, the number of passengers entering the cab between the second floor and the next highest floor is recorded. If the cab moves downwards, the number of passengers who exit the cab on these floors are also recorded. For outgoing traffic, the number of passengers who exit the cab on the main floor is recorded. The recorded data are then aggregated with a fixed granularity to calculate the passenger arrival rates in different traffic.

6.2. Traffic flow forecast with ARRES

Traffic pattern recognition aims to detect the pattern change in advance to avoid the form of long waiting queues. Thus, traffic flow forecast is implemented to predict the passenger arrival rates. In this study, the ARRES model is chosen to solve this problem, which was validated in previous work [36]. The learning of traffic flow is not limited to a single day, and the past data will be incorporated into the long-term statistics.

Compared to single exponential smoothing, ARRES continuously optimizes the smoothing factor. The prediction formula is illustrated below.

$$F_{t+1} = \alpha_t Y_t + (1 - \alpha_t) F_t \tag{5}$$

$$\alpha_t = |E_t / AE_t| \tag{6}$$

$$E_t = \beta e_t + (1 - \beta) E_{t-1}, 0 < \beta < 1 \tag{7}$$

$$AE_t = \beta |e_t| + (1 - \beta) AE_{t-1} \tag{8}$$

$$e_t = Y_t - F_t \tag{9}$$

where F_{t+1} is the forecast smoothed value, Y_t is the observation in the current period, E_t is the smoothed average error, and AE_t is the smoothed absolute error. α_t is the smoothing factor that keeps being optimized. β is a predefined constant parameter between zero and one. And e is the error term. After the forecast, the future passenger arrival rates in different traffic, $\lambda_{inc}, \lambda_{out}, \lambda_{int}$, can be obtained.

6.3. Traffic pattern recognition with fuzzy logic

Traffic pattern recognition is implemented with fuzzy logic that analyzes the traffic intensity and the distribution of different traffic components in the building. It is based on published research in KONE [8]. The proposed method defines four traffic factors related to traffic components and the overall traffic intensity. Fuzzy sets are obtained with two types of membership functions for traffic components and the traffic intensity. Fuzzy rules are then applied to determine the traffic pattern.

Traffic component analysis and traffic intensity evaluation are firstly done, with four traffic factors, u_1, u_2, u_3, u_4, defined to grade their level. The component analysis aims to reflect the proportion of three traffic components. The levels for incoming traffic, outgoing traffic, and inter-floor traffic, u_1, u_2, u_3, are shown in Equation (10)-(12).

$$u_1 = 100^*\lambda_{inc}/(\lambda_{inc} + \lambda_{out} + \lambda_{int}) \tag{10}$$

$$u_2 = 100^*\lambda_{out}/(\lambda_{inc} + \lambda_{out} + \lambda_{int}) \tag{11}$$

$$u_3 = 100^*\lambda_{int}/(\lambda_{inc} + \lambda_{out} + \lambda_{int}) \tag{12}$$

The overall traffic intensity level u_4 is to scale the total arrival rate to the passenger handling capacity HC, where HC is defined as the number of served passengers in a five-minute period during up-peak traffic:

$$u_4 = 100^*(\lambda_{inc} + \lambda_{out} + \lambda_{int})/HC \tag{13}$$

Then fuzzy sets {low, medium, high}$\in \mathscr{F}_1$ are used to describe u_1, u_2, u_3, and the membership functions are shown in Equation (14)-(16), in which u represents u_1, u_2, or u_3. Fig. 8 presents the fuzzy set membership functions of the traffic components.

$$\mu_{low}(u) = \begin{cases} 1, if\ u < 25 \\ \dfrac{35-u}{10}, if\ 25 \le u < 35 \\ 0, if\ u \ge 35 \end{cases} \tag{14}$$

$$\mu_{medium}(u) = \begin{cases} 0, if\ u < 30 \\ \dfrac{u-30}{20}, if\ 30 \le u < 50 \\ \dfrac{70-u}{20}, if\ 50 \le u < 70 \\ 0, if\ u \ge 70 \end{cases} \tag{15}$$

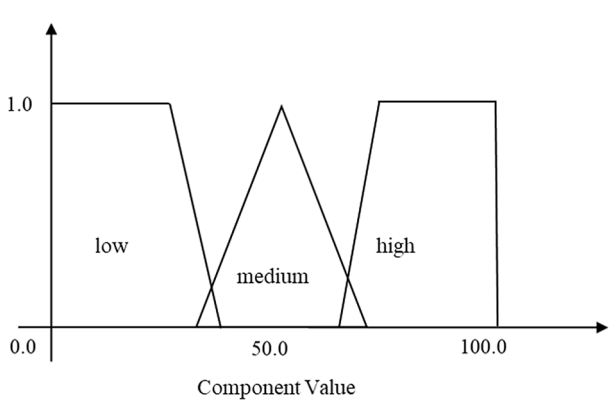


Fig. 8. Membership functions for different traffic components.

$$\mu_{high}(u) = \begin{cases} 0, if\ u < 65 \\ \dfrac{u-65}{10}, if\ 65 \le u < 75 \\ 1, if\ u \ge 75 \end{cases} \tag{16}$$

The obtained results are used to determine the grade of the traffic type with the equation below, and corresponding fuzzy rules are presented in Table 3. For example, a rule for determining the traffic type's being incoming can be expressed as "incoming≈high and outgoing≈low and inter-floor≈low". The traffic type will influence the parameter setting in the membership functions of the traffic intensity.

$$u_j(u_1, u_2, u_3) = \mu_i(u_1) \wedge \mu_i(u_2) \wedge \mu_i(u_3) = \min\{\mu_i(u_1), \mu_i(u_2), \mu_i(u_3)\} \tag{17}$$

where $i \in \mathscr{F}_1, j \in J$. J is the space of all traffic types. The grade obtained with Equation (17) is added to the corresponding rule. The rule with the highest truth value determines the traffic type.

The fuzzy sets for the traffic intensity are {light, normal, heavy, intense}$\in \mathscr{F}_2$, as shown in Fig. 9.

The membership functions for different traffic intensities are presented in the following:

$$\mu_f(u_4) = \begin{cases} 0, if\ u_4 < a_j \\ \dfrac{u_4 - a_j}{b_j - a_j}, if\ a_j \le u_4 < b_j \\ 1, if\ b_j \le u_4 < c_j \\ \dfrac{d_j - u_4}{d_j - c_j}, if\ c_j \le u_4 < d_j \\ 0, if\ u_4 \ge d_j \end{cases} \tag{18}$$

where $a_j \le b_j \le c_j \le d_j, f \in \mathscr{F}_2$. Different set limits, a_j, b_j, c_j, d_j, are set according to the traffic type j.

Finally, to determine the grade of each traffic pattern, the following formula is used:

$$\mu_z(u_1, u_2, u_3, u_4) = min\{\mu_i(u_1), \mu_i(u_2), \mu_i(u_3), \mu_f(u_4)\} \tag{19}$$

where z $\in Z$. Z is the space of all traffic patterns.

Table 4 presents the 36 fuzzy rules used to determine traffic patterns. The rule with the highest truth value determines the traffic pattern.

7. Real-time dispatching optimization with prioritized A* search considering occupancy information

The elevator dispatching problem falls into the NP-complete problems and requires real-time responses. In a building with $\mathscr{N}$ elevators with p hall calls, the number of possible solutions is $\mathscr{N}^p$. During peak

Table 3
Fuzzy rules to determine the traffic type.

Incoming	Outgoing	Inter-floor	Traffic Type
high	low	low	incoming
medium	low	low	incoming
low	high	low	outgoing
low	medium	low	outgoing
low	low	high	inter-floor
low	low	medium	inter-floor
medium	medium	low	two-way
medium	low	medium	mixed
low	medium	medium	mixed

Note: Reprinted from "Elevator group control with artificial intelligence", M. Siikonen, 1997.

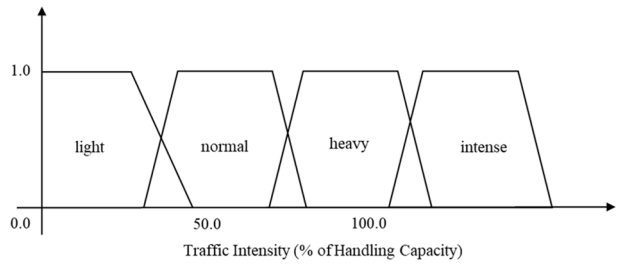


Fig. 9. Membership functions for the traffic intensity.

Table 4
Fuzzy rules for traffic pattern recognition under heavy intensity.

Intensity	Incoming	Outgoing	Inter-floor	Traffic Pattern
intense	high	low	low	intense up-peak
intense	low	high	low	intense down-peak
intense	low	low	high	intense inter-floor
intense	medium	low	low	intense up-peak
intense	low	medium	low	intense down-peak
intense	low	low	medium	intense inter-floor
intense	medium	medium	low	intense mixed
intense	medium	low	medium	intense mixed
intense	low	medium	medium	intense mixed
heavy	high	low	low	up-peak
heavy	low	high	low	down-peak
heavy	low	low	high	heavy inter-floor
heavy	medium	low	low	heavy up-peak
heavy	low	medium	low	heavy down-peak
heavy	low	low	medium	heavy inter-floor
heavy	medium	medium	low	mixed
heavy	medium	low	medium	heavy mixed
heavy	low	medium	medium	heavy mixed
normal	high	low	low	up-peak
normal	low	high	low	down-peak
normal	low	low	high	inter-floor
normal	medium	low	low	up-peak
normal	low	medium	low	down-peak
normal	low	low	medium	inter-floor
normal	medium	medium	low	mixed
normal	medium	low	medium	mixed
normal	low	medium	medium	mixed
light	high	low	low	light up-peak
light	low	high	low	light down-peak
light	low	low	high	light inter-floor
light	medium	low	low	light up-peak
light	low	medium	low	light down-peak
light	low	low	medium	light inter-floor
light	medium	medium	low	light mixed
light	medium	low	medium	light mixed
light	low	medium	medium	light mixed

Note: Reprinted from "Elevator group control with artificial intelligence", M. Siikonen, 1997.

traffic, the dispatching system needs to make hall call allocations in a short time with high quality. In this study, a modified prioritized A* search algorithm is used to find the optimal allocation considering occupancy information [7].

7.1. Prioritized A* search algorithm

Compared to other heuristics-based methods, the A* search algorithm is more suitable in this study. Firstly, prevailing heuristics cannot be stopped during running unless the convergence criteria are met, but hall call allocations should be done within very few seconds after a new hall call is registered. Secondly, population-based heuristics usually have a thorough search horizon. However, dispatching information keeps changing, and optimal allocations of the hall calls far from cab location are less important, once new hall calls are generated in between. Thus, it is advantageous if the algorithm can limit the search horizon and interrupt the search on given conditions. Thirdly, the cost of each hall call task is reflected in the objective function of the A* search algorithm, making it easier to fit the occupancy information into the problem.

As a local search heuristic algorithm, A* search can be understood as a path-finding algorithm, in which each path connects an initial state to a goal state [37]. During the search, the path is evaluated through a heuristic state evaluation function, f, which is the sum of two cost functions: the cost of reaching the current state from the initial state, g, and the cost of reaching the goal state from the current state, h. The search direction is based on the heuristic state evaluation function. The key feature of A* search is that if h is never overestimated, the solution is optimal in terms of g [37]. The workflow of A* search is presented in Table 5.

In the context of elevator dispatching, each level addresses one hall call assignment in the A* search algorithm. When the number of hall calls exceeds the number of cabs, this algorithm will not terminate until all hall calls are assigned. However, in this scenario, it is more important that each cab has at least one hall call task. Moreover, if the algorithm is interrupted because of the limit of computing time, there might be cabs with no hall call assignment. In this regard, the prioritized A* search [7] is proposed and prioritizes the search sequence according to the elevator usage. This means the objective of the search at each tree level is to allocate a hall call task to a specified cab. Because the search logic forces the assignment of cabs to hall calls instead of the reverse, the search can be stopped after each cab has no less than one hall call task.

7.2. Elevator dispatching optimization problem formulation

To introduce the occupancy information into the problem formulation, the capacity C_n and cab ground area S_n are utilized, where n is the index of cabs. Based on the influence of the capacity on pick-up failure, the cost of performing a hall call task should be smaller if a cab has more capacity, and vice versa. Thus, the occupancy factor o_n is set as the coefficient of the task cost, and it is calculated as below:

$$o_n = \begin{cases} \sqrt{S_n/C_n}, \, if \;\; C_n > 0.01S_n \\ 10, \, if \;\; C_n \leq 0.01S_n \end{cases} \tag{20}$$

The effect of capacity on dispatching is enlarged to some extent when the capacity is larger than 1% of the ground area. To avoid o_n being too large, its maximum is set as 10. Furthermore, the occupancy factor o_n should only influence the hall call tasks performed before the cab changes the direction, because usually a cab will not change its direction until it finishes all the tasks in the original direction.

The passenger waiting time is chosen as the criterion to evaluate dispatching performance. Correspondingly, the calculation of g and h are designed to reflect the waiting time. Two look-up tables are prepared to calculate g and h. The first table records the cost for each cab to answer a hall call as its first hall call task from the current position,

Table 5
A* search algorithm workflow.

Initialization
(1) Set F as the number of tree levels
(2) Initialize the closed set $C = \varnothing$ and the open node set $M_i = \varnothing$ for unsearched nodes in the level i, $i = 1, \cdots, F$

Search
(1) Generate the descendant nodes of the node m_i, which has the lowest heuristic state function value in the node set M_i, and add its descendant nodes to M_{i+1}
(2) Calculate the f value for the nodes in M_{i+1}
(3) Sort the nodes in M_{i+1} in increasing order by f values
(4) Move the node m_i to the closed set C

Termination
(1) If the first node in M_{i+1} is the goal state, end.
(2) Else, let $i = i+1$ and return to the search stage.

taking the cab call tasks into account. And the second table records the cost of trips between different pairs of hall calls for each cab, in which both hall call and cab call tasks are considered. The cost estimation is based on the worst-case scenario to ensure no overestimation is made on h. Let d_n be the cost to answer a hall call as its first hall call task for the cab n, and $s_n(p_i, p_{i+1})$ be the cost of a trip between the hall call p_i and p_{i+1}, where p_i refers to an assigned hall call of the cab n. Let P_n be the number of assigned hall calls to it. Thus, the first table records $d_n(p_1)$, referring to the cost of answering its first hall call task p_1. The second table records $s_n(p_i, p_{i+1})$, meaning the cost of answering the hall call p_i followed by p_{i+1}. Let v be the number of hall call tasks before it changes the direction. For each hall call m performed before the cab changes its direction, the cost $g_n(m)$ is given by:

$$g_n(m) = o_n{}^* d_n(p_1),\ when\ m = 1 \tag{21}$$

$$g_n(m) = o_n{}^* d_n(p_1) + \sum_{i=1}^{m-1} o_n^{\frac{1}{3i}} s_n(p_i, p_{i+1}),\ when\ m = 2, \cdots, v \tag{22}$$

It should be noted that the current occupancy information has less influence on the task cost as the task index m becomes larger. On the other hand, if the hall call m is performed after the cab n changes its direction, the task cost will not be influenced by the current capacity. For a hall call m which is in the opposite direction of the cab n, the cost $g_n(m)$ is calculated as below:

$$g_n(m) = d_n(p_1),\ when\ m = 1 \tag{23}$$

$$g_n(m) = d_n(p_1) + \sum_{i=1}^{m-1} o_n^{\frac{1}{3i}} s_n(p_i, p_{i+1}),\ when\ m > v = 0 \tag{24}$$

$$g_n(m) = o_n{}^* d_n(p_1) + \sum_{i=1}^{v-1} o_n^{\frac{1}{3i}} s_n(p_i, p_{i+1}) + \sum_{j=v}^{m-1} s_n\left(p_j, p_{j+1}\right),\ when\ m > v > 0 \tag{25}$$

Thus, g_n is the sum of $g_n(m)$ for all the assigned hall calls of a cab, and g is the sum of g_n for all the cabs at the current node. They are expressed as below:

$$g_n = \sum_{m=1}^{P_n} g_n(m) \tag{26}$$

$$g = \sum_{n=1}^{\mathscr{N}} g_n \tag{27}$$

where $\mathscr{N}$ is the number of elevator cabs in the system.

Let Q be the number of hall calls that have not been assigned and q_u is one of the hall calls, where $u = 1, \cdots, Q$. To limit the optimization time when Q is large, an upper limit can be set on the number of hall calls to be assigned. The cost $h_n(k)$, where k means the k_{th} hall call to be assigned in the current node, refers to the cab n answering the k_{th} hall call before it changes its direction and is calculated as below:

$$h_n(k) = o_n{}^* \min[d_n(q_u)],\ when\ P_n = 0 \tag{28}$$

$$h_n(k) = \min\left[\sum_{m=1}^{P_n} g_n(m) + o_n^{\frac{1}{3P_n}} s_n\left(p_{P_n, q_u}\right)\right],\ when\ P_n > 0,\ u = 1, \cdots, Q \tag{29}$$

The objective is to find the hall call that results in the minimum $h_n(k)$. And if the k_{th} hall call is performed after the cab n changes the direction, the cost $h_n(k)$ is given by:

$$h_n(k) = \min[d_n(q_u)],\ when\ P_n = 0 \tag{30}$$

$$h_n(k) = \min\left[\sum_{m=1}^{P_n} g_n(m) + s_n\left(p_{P_n, q_u}\right)\right],\ when\ P_n > 0 \tag{31}$$

In this regard, $h(k)$ is the smallest value from $h_n(k)$ for all the cabs, and h is the sum of $h(k)$ for all the newly assigned hall calls at this node. In the meantime, h is used as the termination criterion. If $h = 0$ at a node, the current state will be the optimal solution.

8. Simulation experiments and system validation

In this section, the discrete event simulation model for elevator dispatching is developed, and simulation experiments are conducted to validate the proposed smart dispatching with real-time occupancy awareness. To give prominence to the effects of dispatching with occupancy information and to simplify the simulation process, only capacity estimation is implemented, meaning that the object occupancy information is known and recorded once it is generated, and no special usage pattern is set during the experiments. Furthermore, objects in the experiments are designed to have relatively large occupancy area but small weight.

8.1. Experiment environment setup and assumptions

Simulation for elevator dispatching requires three types of support from the software perspective: one for building the discrete event simulation model, one for implementing the dispatching optimization algorithm, and one for data exchange between the simulation model and the algorithm. In this study, Simio is used to model the behavior of elevator cabs and passengers. MATLAB is used for running dispatching optimization. And Microsoft SQL Server is the data exchange platform. A user-defined Simio process is developed with C# to call MATLAB functions from Simio, so that Simio can run the MATLAB optimization algorithm during the simulation. Some assumptions are made based on the elevator usage in real life:

(i). Passengers cannot enter a cab if it does not have enough capacity or they make it overweighed.
(ii). If passengers cannot enter a cab when it arrives, they will repress the hall call button after it leaves.
(iii). If a cab is fully occupied, it still stops at the floor where its allocated hall call tasks are registered.
(iv). Passengers only enter the cab that goes towards their destinations.
(v). Passengers arrive one by one, instead of in a group.
(vi). A cab will park on the ground floor if it has no tasks.
(vii). A cab will not change the direction until all the hall call and cab call tasks are fulfilled in the original direction.
(viii). The time of traveling to the adjacent floor is fixed.
(ix). The time is the same for performing either a hall call task, a cab call task, or both on a floor.

8.2. Simulation model development with Simio

In Simio, the basic object unit is the entity. Built-in entities like vehicles and workers in the standard library have predefined properties, processes, and functions. Considering the complexity of the elevator operation logic, this study develops the elevator and passenger objects from scratch to model their behaviors.

(1). ***Model state variable setup:*** To guarantee the accessibility of key variables in the process of all the entity objects, also for the convenience of data exchange, they are defined as the model state variables. These variables include the weight and capacity of each elevator cab, arrays that record the allocated hall calls and the committed cab calls of each cab, and an array recording the unassigned hall calls. Meanwhile, total passenger waiting time, total passenger journey time, and the number of the served passenger are defined here.
(2). ***Passenger object:*** Passenger processes can be grouped according to whether the passenger enters a cab or not. When the passenger is waiting, the process is to ensure that the passenger information is initialized, and the unassigned hall call array is updated. After

the passenger enters the cab, the process is to update the cab variables and to check if the cab arrives at the passenger's destination floor.

The key process is shown in Fig. 10. After a passenger is generated from the source, related state variables are initialized, including traffic data like the birth floor and destination floor, and data for calculating the weight and cab capacity. Then the passenger is transferred to the waiting area of that floor, and the unassigned hall call array is updated in the corresponding index. After the preparation, there are conditional statements to check if any cab arrives at the floor and if the passenger can be taken. If so, the passenger is transferred into the cab. Otherwise, re-dispatching might be implemented. After the passenger enters the cab, the array that records the cab call tasks is updated. The passenger waits until the cab arrives at the destination floor and is then transferred to the exit.

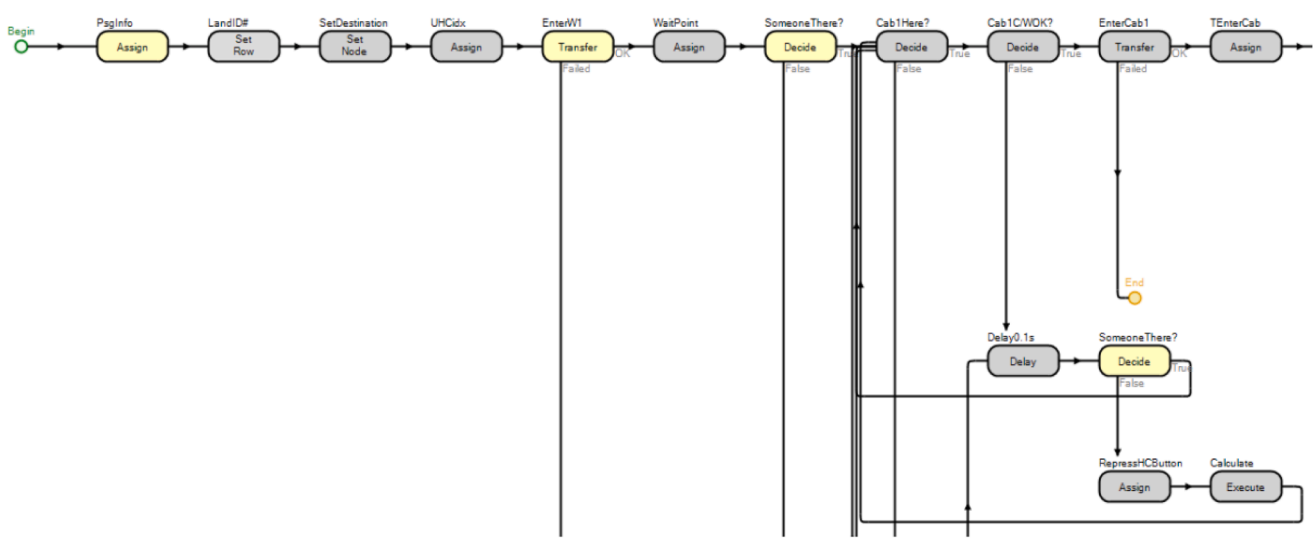
(3). ***Elevator object:*** The difficulty of modeling elevator objects is to determine the destination of the next task. Because the dispatching operation is dynamic, and traffic data keeps changing, assigned hall calls can be reassigned to another cab. Thus, it is necessary to update the next task whenever it arrives at a floor or after it performs a task. In this study, elevator processes are grouped as the floor-related process and the task-related process. The former defines the operation logic when a cab arrives at a floor, while the latter is to find its next task.

Fig. 11(a) presents the process when a cab arrives at a floor. It first determines if the next task is on this floor or if a hall call from this floor is assigned to it when it approaches. If not, the next task will be updated in case of any change of hall call allocations. If yes, the current direction will be determined, and the cab will stop at this floor to drop off or pick up passengers. The corresponding task arrays are then updated. After that, the next task is determined. Fig. 11(b) shows the process to determine the next task after a task is performed. Firstly, it checks if the cab has any task. If not, the cab will park on the ground floor. If yes, the current direction and the number of unperformed tasks in this direction will be determined, and these two steps will decide if a change of direction is needed. Then the nearest task in the current direction is set as the next task.

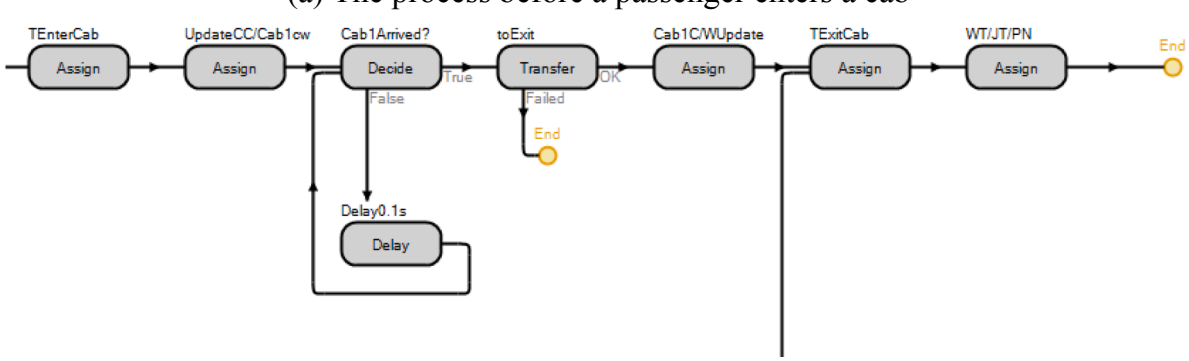
8.3. Experiment design

The experiments simulate elevator dispatching in a 10-floor building with two elevator cabs at different traffic intensity levels, and the objective is to analyze the effect of cab capacity on dispatching optimization. The facility view of the simulation model is presented in Fig. 12.

In the experiments, passenger-related variables include the weight and occupancy area of passengers and their belongings, and the passenger arrival rates on the ground floor and upper floors. Table 6 presents the weight distribution of passengers as well as their belongings, and the probability of occurrence for each scenario is also given. The corresponding object occupancy information can be found in Table 1. The destination floor of a passenger is randomly selected from all other

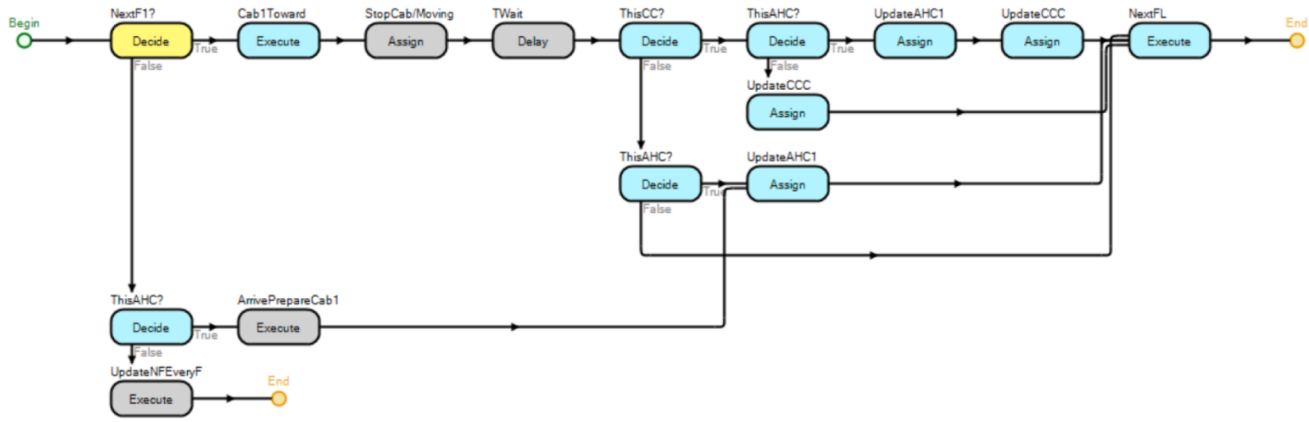


(a) The process before a passenger enters a cab

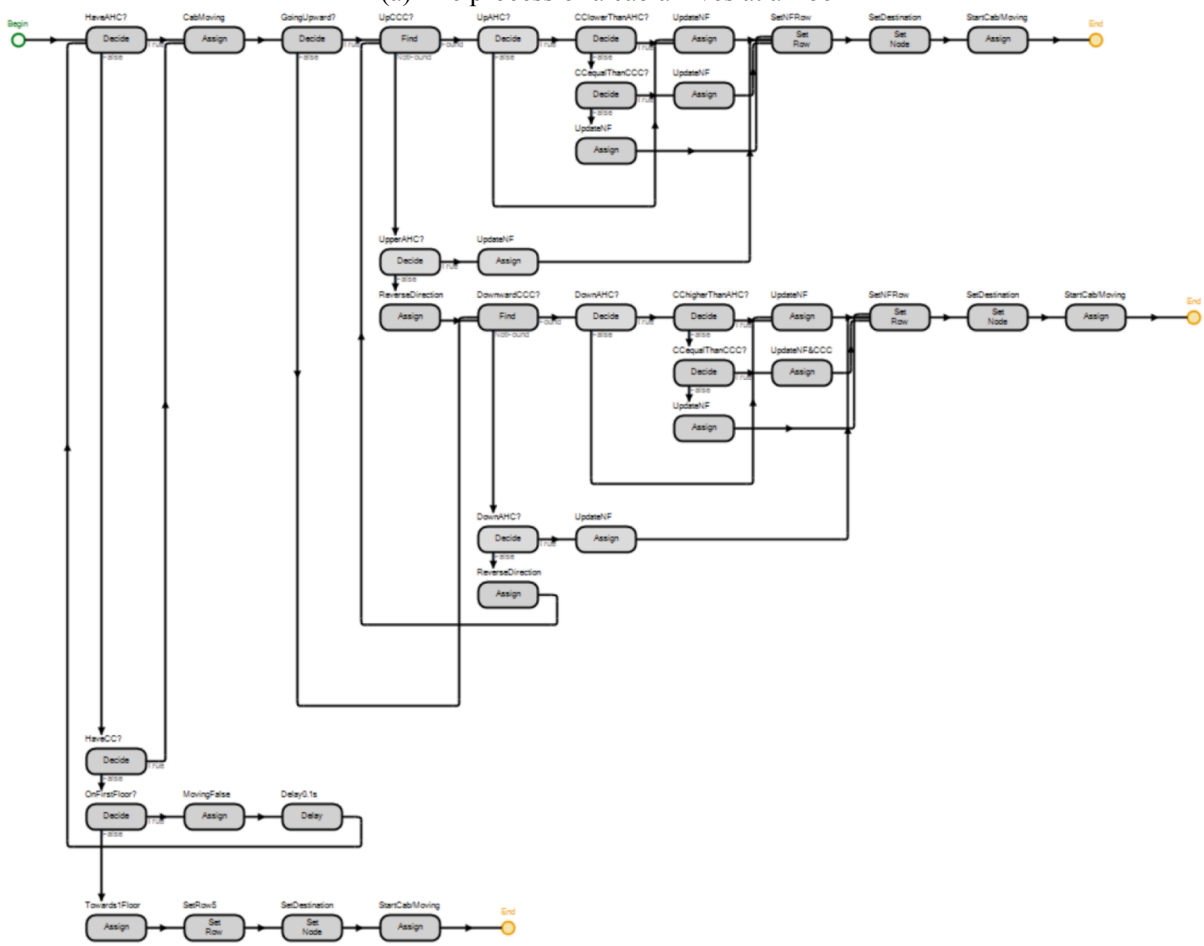


(b) The process after a passenger enters a cab

Fig. 10. The key process of passenger objects.



(a) The process of a cab arrives at a floor



(b) The process of determining the destination of the next task

Fig. 11. The key processes of elevator objects.

floors with the same probability. Five travel intensity levels are set, including one scenario for light traffic, normal traffic, heavy traffic, and two for intense traffic. Each level has different passenger arrival rates and experiment time, as shown in Table 7. And elevator-related variables are fixed. The occupancy area of the cab ground is $4m^2$, and the load capacity is 1000 kg. It takes 4 seconds for a cab to go to an adjacent floor and 8 seconds to perform a task.

The prioritized A* search algorithm is used to solve the optimization problem. Two optimization models are developed: one without the occupancy factor in the heuristic state evaluation function, representing the conventional dispatching system; and one with the occupancy factor, representing the occupancy-aware dispatching system. The heuristic state evaluation function is calculated based on the estimated time of arrival.

Four indicators are chosen for performance measuring: average waiting time, average travel time, average journey time, and the number of served passengers. There are two reasons to choose the four indicators. Firstly, because the algorithm is expected to minimize the waiting time, the effects of using capacity information on the traffic time should be analyzed. Furthermore, capacity information conveys the space utilization of a cab, so the quantity of service is another affected factor that should be considered.

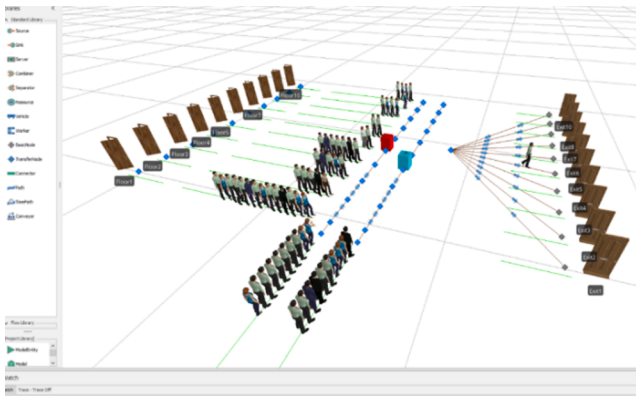


Fig. 12. Discrete-event simulation in Simio.

Table 6
Weight distribution and probability of passengers and belongings.

Combination	Simio Expression		Probability
	Human Weight	Belonging Weight	
Human	Random.Triangular (30, 65, 90)	0	65%
Human + Bicycle	Random.Triangular (30, 65, 90)	Random.Triangular (7.5, 8.5, 10)	2%
Human + Suitcase	Random.Triangular (45, 65, 90)	Random.Triangular (20, 23, 30)	5%
Human + Cardboard Box	Random.Triangular (45, 65, 90)	Random.Triangular (8, 15, 25)	7%
Human + Backpack	Random.Triangular (30, 65, 90)	Random.Triangular (3, 5.5, 8)	20%
Human + Wheelchair	Random.Triangular (50, 65, 90)	Random.Triangular (30, 40, 50)	1%

Table 7
Passenger arrival rate settings at different traffic intensity levels.

Traffic Intensity	Intensity Level	Simio Expression		Average Number of Entities per Hour	Running Time [minutes]
		Ground Floor [minutes]	Upper Floor [minutes]		
I1	Light	Random. Exponential (0.3)	Random. Exponential (1)	740	30
I2	Normal	Random. Exponential (0.2)	Random. Exponential (2/3)	1110	20
I3	Heavy	Random. Exponential (0.15)	Random. Exponential (0.5)	1480	15
I4	Intense	Random. Exponential (0.12)	Random. Exponential (0.4)	1850	15
I5	Intense	Random. Exponential (0.1)	Random. Exponential (1/3)	2220	15

8.4. Results and analysis

The experiment results are recorded in Table 8, showing the values of the four performance indicators from two optimization models at five different traffic intensity levels. The comparison charts of the four indicators between the two models are also presented in Fig. 13.

Compared to the traditional dispatching method, the proposed occupancy-aware dispatching fits the occupancy factor into the original evaluation function as the coefficient of a task cost. This change introduces the trade-off between the time cost of a hall call task and space utilization of the assigned cab. In Fig. 13(a), occupancy-aware dispatching suggests lower average waiting time in light, heavy, and intense traffic. When passenger arrival rates are low, the constraints on capacity can always be satisfied, and pick-up failure will not happen. In this case, the model with the occupancy factor is influenced by cab capacity during decision making and might miss the optimal decision. As passenger arrival rates become larger, capacity is sometimes not enough to accommodate new passengers. Those who fail to enter the cab will spend extra time waiting for the next cab. In this case, occupancy-aware dispatching helps avoid pick-up failure by skipping the original optimal hall call allocation. However, if the traffic is too heavy, cabs are nearly always fully occupied, and the occupancy factor will be similar among different cabs, making the capacity information less effective. Fig. 13(b) presents the comparison of the average travel time, where occupancy-aware dispatching has slightly better performance in normal, heavy, and intense traffic. The reduction of travel time can be due to the reduction in the probability of pick-up failure, which reduces unnecessary stops in a trip. Fig. 13(c) suggests occupancy-aware dispatching improves the quality of service in light, heavy, and intense traffic.

The number of served passengers is an indicator for the quantity of service, and Fig. 13(d) presents the comparison of experiment results. It is observed that occupancy-aware dispatching improves the quantity of service in normal, heavy, and intense traffic. This is because the occupancy factor helps balance the space utilization. Because the utilization difference is reflected in the decision-making process, cabs with more capacity can perform more tasks with lower costs. However, if the passenger arrival rate is low, cabs always serve the new passengers in a short time, and the quantity of service will not be improved.

To summarize, occupancy-aware dispatching improves both the quality of service and quantity of service to some extent. In most cases, it improves the quantity of service if the passenger arrival rate is not too low. It also reduces the average waiting time when the traffic is not too heavy. In practical applications, if the traffic intensity is light or intense, where the capacity is always enough or always not enough, the occupancy factor can be set smaller.

8.5. Managerial implications

In practice, the proposed occupancy-aware dispatching can help building managers improve both the quality and quantity of elevator service. Because cab capacity is considered during dispatching, fewer allocations that result in pick-up failure will be generated. Therefore, passengers spend less time waiting during the rush hour, and elevators can transport more passengers. Moreover, deep learning-based object detection and case-based reasoning enable the dispatching system to identify special elevator usages during operation and to change the task allocation rule correspondingly, bringing convenience to elevator users. Therefore, the occupancy-aware dispatching is expected to be useful in buildings where passengers carry large-size objects or cabs use specific control strategies in different usages, like hospitals. Therefore, the occupancy-aware dispatching system will become a positive contributor to smart buildings.

9. Conclusions

In this paper, real-time occupancy-aware smart dispatching is proposed, and its decision modeling is presented. Compared to prevailing dispatching systems, real-time occupancy information analysis is conducted to provide useful information for optimal dispatching, including the estimated capacity, the traffic pattern, and the usage pattern. The contribution of this work is the design of the occupancy-aware dispatching system. A novel method is proposed to estimate the cab capacity by modeling the 3D space consistency level, and case-based reasoning is applied to usage pattern recognition to adjust dispatching rules according to the cab usage. Moreover, the cab capacity constraint is formulated as the occupancy factor and is integrated into the objective

Table 8
Experiment results at different traffic intensity levels.

Traffic Intensity	Running Time [minutes]	Whether Consider Occupancy factor	Average Waiting Time [seconds]	Average Travel Time [seconds]	Average Journey Time [seconds]	Number of Served Passengers
I1	30	No	59.93	39.90	99.83	358
		Yes	55.29	41.37	96.66	358
I2	20	No	63.36	43.76	107.12	351
		Yes	70.45	42.83	113.28	364
I3	15	No	83.97	42.96	126.93	308
		Yes	82.16	42.57	124.73	320
I4	15	No	92.97	44.85	137.82	366
		Yes	87.52	41.85	129.37	358
I5	15	No	92.06	43.52	135.58	431
		Yes	92.49	43.96	136.45	450

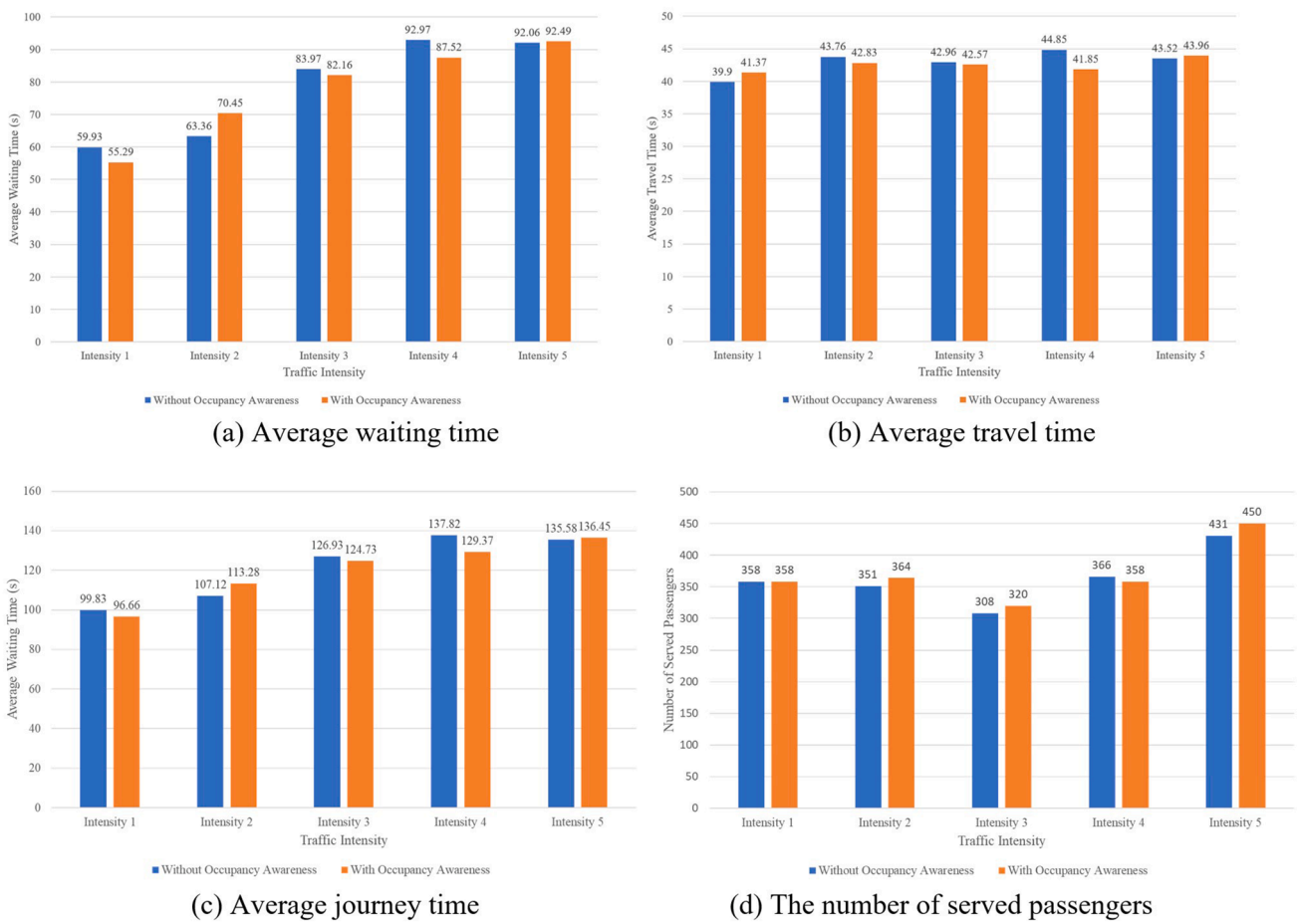


Fig. 13. Performance comparison of four indicators.

function of the dispatching optimization model. Discrete event simulation experiments are conducted, suggesting the occupancy-aware dispatching system can improve the quantity of service and quality of service in most cases.

However, limitations exist in this study. In capacity estimation, the influence of object position is not considered. The relationship between object position and cab capacity is difficult to model, and the camera placement may bring errors to object localization. Also, current formulation of the occupancy factor is fixed at different traffic intensity levels. Dynamic formulation should be studied to adapt to the features of different traffic intensities.

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.

Acknowledgements

This material is based upon work supported by the National Science Foundation NSF Future of Work at the Human-Technology Frontier Big Idea under Grant No. 1928313.

References

[1] B. Gaille, 22 Elevator Industry Statistics, Trends & Analysis (2018).
[2] J.R. Fernandez, P. Cortes, A survey of elevator group control systems for vertical transportation: a look at recent literature, IEEE Control Systems Magazine 35 (4) (2015) 38–55, https://doi.org/10.1109/MCS.2015.2427045.
[3] C.B. Kim, K.A. Seong, H. Lee-Kwang, J.O. Kim, Y.B. Lim, A fuzzy approach to elevator group control system, IEEE Transactions on systems, man, and cybernetics 25 (6) (1995) 985–990, https://doi.org/10.1109/21.384263.
[4] G. Barney, L. Al-Sharif, Elevator traffic handbook: theory and practice, Routledge, 2015.
[5] L. Al-Sharif, Introduction to elevator group control (METE XI), Lift Report 42 (2) (2016) 59–68.
[6] M.R. Garey, D.S. Johnson, Computers and intractability, Vol. 174, freeman, San Francisco, 1979.
[7] M. Hamdi, D.J. Mulvaney, Prioritised A* search in real-time elevator dispatching, Control Engineering Practice 15 (2) (2007) 219–230, https://doi.org/10.1016/j.conengprac.2006.06.005.
[8] M. Siikonen, Elevator group control with artificial intelligence, (1997).
[9] W.L. Chan, A.T. So, K.C. Lam, Dynamic zoning for intelligent supervisory control, International Journal of Elevator Engineering 1 (1996) 47–59.
[10] Z. Li, Y. Zhang, H. Tan, Particle swarm optimization for dynamic sectoring control during peak traffic pattern, In International Conference on Intelligent Computing Springer, Berlin, Heidelberg (2007) pp. 650-659.
[11] B. Bolat, P. Cortés, Genetic and tabu search approaches for optimizing the hall call—car allocation problem in elevator group systems, Applied Soft Computing 11 (2) (2011) 1792–1800, https://doi.org/10.1016/j.asoc.2010.05.023.
[12] J. Gale, ThyssenKrupp's TWIN lift system, Elevator World 51 (7) (2003) 51–53.
[13] J. Sorsa, H. Hakonen, M. Siikonen, Elevator selection with destination control system, Elevator World 54 (1) (2006) 148.
[14] K. Hirasawa, T. Eguchi, J. Zhou, L. Yu, J. Hu, S. Markon, A double-deck elevator group supervisory control system using genetic network programming, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38 (4) (2008) 535–550, https://doi.org/10.1109/TSMCC.2007.913904.
[15] J. Wang, Y. Shen, S. Wang, Q. Zhao, K. Nakamura, H. Yamada, T. Tanaka, Energy-saving algorithm for elevator group control system with cameras, in: IEEE In Proceeding of the 11th World Congress on Intelligent Control and Automation, 2014, pp. 2654–2658, https://doi.org/10.1109/WCICA.2014.7053144.
[16] V. Zarikas, N. Tursynbek, Intelligent elevators in a smart building, FTC 2017-Future Technologies Conference (2017).
[17] Y. Bapin, K. Alimanov, V. Zarikas, Camera-driven probabilistic algorithm for multi-elevator systems, Energies 13 (23) (2020) 6161, https://doi.org/10.3390/en13236161.
[18] J. Page, D. Robinson, N. Morel, J.L. Scartezzini, A generalised stochastic model for the simulation of occupant presence, Energy and buildings 40 (2) (2008) 83–98, https://doi.org/10.1016/j.enbuild.2007.01.018.
[19] B. Dong, B. Andrews, Sensor-based occupancy behavioral pattern recognition for energy and comfort management in intelligent buildings, Proceedings of building simulation (2009) 1444–1451.
[20] R. Deplazes, R. Cortona, Three-dimensional monitoring in the area of an elevator by means of a three-dimensional sensor, U.S. Patent (2006) 7,140,469.
[21] Y. Pi, N.D. Nath, A.H. Behzadan, Convolutional neural networks for object detection in aerial imagery for disaster response and recovery, Advanced Engineering Informatics. 43 (2020), 101009, https://doi.org/10.1016/j.aei.2019.101009.
[22] Y. Wang, M. Liu, P. Zheng, H. Yang, J. Zou, A smart surface inspection system using faster R-CNN in cloud-edge computing environment, Advanced Engineering Informatics. 43 (2020), 101037, https://doi.org/10.1016/j.aei.2020.101037.
[23] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: In Proceedings of the IEEE international conference on computer vision, 2017, pp. 2961–2969.
[24] P. Cortés, J. Larrañeta, L. Onieva, Genetic algorithm for controllers in elevator groups: analysis and simulation during lunchpeak traffic, Applied Soft Computing 4 (2) (2004) 159–174, https://doi.org/10.1016/j.asoc.2003.11.002.
[25] B. Bolat, O. Altun, P. Cortés, A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems, Applied Soft Computing 13 (5) (2013) 2633–2642, https://doi.org/10.1016/j.asoc.2012.11.023.
[26] S. Tsuji, M. Amano, S. Hikita, Application of the expert system to elevator group-supervisory control, In IEEE Proceedings, The Fifth Conference on Artificial Intelligence Applications (1989) 287–294, https://doi.org/10.1109/CAIA.1989.49165.
[27] M. Amano, Elevator control apparatus using evaluation factors and fuzzy logic, U. S. Patent (1993) 5,233,138.
[28] C.B. Kim, K.A. Seong, H. Lee-Kwang, J.O. Kim, Design and implementation of a fuzzy elevator group control system, IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 28 (3) (1998) 277–287, https://doi.org/10.1109/3468.668960.
[29] J. Liu, Y. Liu, Ant colony algorithm and fuzzy neural network-based intelligent dispatching algorithm of an elevator group control system, 2007 IEEE International Conference on Control and Automation (2007) pp, https://doi.org/10.1109/ICCA.2007.4376773.
[30] G.C. Barney, Uppeak revisited, Elevator technology 4 (1992) 39–47.
[31] F. Luo, Y. Xu, J. Cao, Elevator traffic flow prediction with least squares support vector machines, 2005 IEEE International Conference on Machine Learning and Cybernetics 7 (2005) 4266–4270, https://doi.org/10.1109/ICMLC.2005.1527686.
[32] M. Siikonen, Elevator traffic pattern recognition, In Proc. of 4th IFSA Congress, 1991.
[33] Z. Zhao, P. Zheng, S. Xu, X. Wu, Object detection with deep learning: A review, IEEE transactions on neural networks and learning systems 30 (11) (2019) 3212–3232, https://doi.org/10.1109/TNNLS.2018.2876865.
[34] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft coco: Common objects in context, In Springer, Cham European conference on computer vision (2014) 740–755, https://doi.org/10.1007/978-3-319-10602-1_48.
[35] H. Shin, H.R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, R. M. Summers, Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning, IEEE transactions on medical imaging 35 (5) (2016) 1285–1298, https://doi.org/10.1109/TMI.2016.2528162.
[36] S.G. Makridakis, S.C. Wheelwright, V.E. McGee, Forecasting: methods and applications, 2nd Edition, Wiley, New York, 1983.
[37] R.E. Korf, Search: A survey of recent results, Morgan Kaufmann Exploring artificial intelligence (1988) 197–237, https://doi.org/10.1016/B978-0-934613-67-5.50010-1.