國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management

College of Management

National Taiwan University

Master's Thesis

結合傳統呼叫系統與目的地呼叫系統之電梯調度最佳化問題

Elevator Dispatching Optimization integrating Conventional and Destination Control Systems

林小喬

Hsiao-Chiao Lin

指導教授: 孔令傑 博士

Advisor: Ling-Chieh Kung Ph.D.

中華民國 114 年 7 月

July, 2025

doi:10.6342/NTU202502477

誌謝

本論文得以順利完成，要衷心感謝在研究過程中所有給予我支持、指導與協助的師長、親友與實驗室的夥伴。

首先，我由衷感謝我的指導教授孔令傑老師。在整個研究過程中，老師不僅給予我學術上的專業指導與，也在我遇到瓶頸時，耐心引導我突破困境，並時常給予我寶貴的建議與鼓勵。老師嚴謹的治學態度和看待問題的方式，將成為我未來學術道路上的榜樣。也要感謝口試委員郭佳瑋教授、余峻瑜教授以及黃奎隆教授，感謝您們撥冗審閱本論文，並提供了許多寶貴的洞見，使本論文的品質得以更加提升。

感謝實驗室的所有夥伴們，在我就讀資管所的兩年當中，在課業、專案或是課外的互相扶持。謝謝佳芊、冠霆，因為你們優秀的能力與負責任的態度，即使我們這屆人數不多，仍效率滿滿的解決各種問題；謝謝學長姊們為實驗室帶來了許多溫暖與歡笑，特別感謝琳瑄、盈穎與元婷，在畢業之後仍給予我支持和鼓勵；謝謝學弟妹們在專案及各種事務的幫忙，特別感謝翊恩、善詩一起完成了電梯派遣專案。

此外，本研究得以進行，也特別感謝翱翔智慧的竣貿學長與祐鈞學長提供寶貴的業界經驗與真實數據支持，使本論文的研究更具實務價值。

最後，謹向我的家人和朋友們表達最深切的感謝。我將永遠緬懷我的父親，在您與病魔搏鬥的這段艱辛時光裡，我在您的病房中完成這份論文；感謝母親在這段時間的理解、支持與包容，您始終是我最堅實的後盾。

林小喬 謹識
于臺大資訊管理學研究所
民國一百一十四年七月

doi:10.6342/NTU202502477

摘要

本研究探討如何解決並最佳化電梯調度問題，旨在平衡能源效率與乘客服務品質。文獻上大多把最佳化電梯調度問題被歸類為 NP-hard 問題，因為其涉及從眾多可行的電梯運行組合中識別最佳調度方案。

我們研究了一種混合控制系統的效率，該系統可將樓層設定為傳統呼叫或目的地呼叫模式。其目標是最大限度地降低電梯運行能耗成本，並減少長時間等待乘客的有效等待時間。

我們透過求解一系列靜態問題來解決動態電梯調度問題，並利用窮舉演算法和基因演算法進行求解。實驗結果表明，相較於傳統呼叫系統，目的地呼叫系統和混合呼叫系統在大多數情境下，於能耗和有效等待時間方面均表現出更佳的性能。此外，基因演算法在計算成本方面提供了顯著優勢，使其成為實際電梯系統更可行的方案。

我們開發了一個與電梯模擬系統串接的電梯調度系統，以模擬現實世界的運作，證實了我們所提出的方法在實務上的可行性。在利用模擬真實乘客需求進行的動態情境案例研究中，結果顯示我們所提出的基於基因演算法的派遣策略，相較於原本基於規則派遣的方法，在能源消耗和有效等待時間方面均顯著降低。

關鍵字：電梯調度問題、目的地呼叫系統、混合呼叫系統、基因演算法

doi:10.6342/NTU202502477

Abstract

In this study, we explores optimization strategies for the Elevator Dispatching Problem (EDP), aiming to balance energy efficiency with passenger service quality. The EDP is classified as an NP-hard problem, involving the identification of optimal scheduling solutions from numerous feasible elevator operation combinations.

We investigate the efficiency of a hybrid control system, which sets floors for either conventional or destination call system. The objective is to minimize the total energy consumption cost of the elevators and the effective waiting time of passengers experiencing prolonged waiting periods.

Our approach addresses the dynamic EDP by solving a sequence of static problems, utilizing both exhaustive and genetic algorithms. Experimental results indicate that, compared to the All-Conventional call system, both All-Destination and Hybrid call systems tend to achieve better performance in energy consumption and effective waiting time across most scenarios. Moreover, the genetic algorithm offers substantial advantages in computational cost, rendering it a more feasible approach for practical elevator systems.

We developed an elevator dispatch system that integrates with an elevator simulation system to replicate real-world operations. Case studies in dynamic scenarios, utilizing simulated real passenger demand, revealed that the proposed GA-based strategy significantly reduces energy consumption and effective waiting time compared to the rule-based method.

Keywords: *Elevator Dispatching Problem, Destination Call System, Hybrid Control System, Genetic Algorithm*

doi:10.6342/NTU202502477

Contents

doi:10.6342/NTU202502477

doi:10.6342/NTU202502477

doi:10.6342/NTU202502477

List of Figures

doi:10.6342/NTU202502477

List of Tables

doi:10.6342/NTU202502477

doi:10.6342/NTU202502477

Chapter 1

Introduction

1.1 Background and Motivation

With the rapid growth of high-rise buildings driven by advancements in construction technologies and increasing demands for space, elevators have become indispensable for ensuring efficient vertical transportation. This critical role highlights the significance of the Elevator Dispatching Problem (EDP), which focuses on optimizing elevator routing and dispatching to serve passengers effectively. EDP addresses complex constraints, including elevator capacity, movement direction, and service zones, while dynamically adapting to real-time and stochastic passenger demands. By enhancing service quality and operational efficiency, EDP ensures a seamless and satisfying experience for building visitors, meeting the rising demands of modern urban infrastructures.

Numerous studies have already been conducted on the EDP. In most papers, the elevator dispatching problem is typically classified as an NP-hard problem because it

doi:10.6342/NTU202502477

involves finding the optimal scheduling solution among numerous feasible elevator operation combinations to achieve specific objectives (Hanif and Mohammad, 2022; Wan et al., 2024). To describe more precisely, the EDP can be considered as a kind of the traveling salesperson problem (TSP)(Tyni and Ylinen, 2006; Zheng et al., 2018). In TSP, a salesperson must visit each city and return the starting point, aiming for the shortest distance; similarly, elevator dispatching problem must determine the best stopping sequence to reduce passenger waiting costs or elevator operating costs. Besides, due to practical application requirements, computation time for getting a better solution is also a critical constraint (Tyni and Ylinen, 2006; Siikonen, 2024), which further increases the complexity of the problem.

There are usually multiple elevators in a high-rise building, and the system that manages the group of elevators is called the elevator group control system (EGCS). EGCS can adapt to changes in passenger demand and building usage patterns, ensuring that service remains efficient under varying conditions. Currently, control systems are mainly divided into two categories: conventional call and destination call. In conventional control systems, the control system only knows the floor from which the passenger's request signal was made and the direction in which the passenger wants to go. It does not have information about the number of passengers or their exact destinations. This type of control system is presently the most mainstream and extensively researched systems (Fujino et al., 1997; Tyni and Ylinen, 2006; Bolat et al., 2013; Tartan and Ciftlikli, 2016; Hanif and Mohammad, 2022; Wan et al., 2024). On the other hand, in destination control systems, the control system collects detailed information from passengers, including their destination floor and the number of people, as passengers declare their desired floor

doi:10.6342/NTU202502477

via a panel or app when calling the elevator. This allows the dispatch policy to increase efficiency by grouping passengers heading to similar destinations. Moreover, it can provide more accurate target value assessments when evaluating a single assignment compared to conventional systems. This advanced system is limited by hardware requirements, and as a result, it is used in fewer buildings and has been investigated by relatively fewer past works (Dai et al., 2010; Ruokokoski et al., 2016; Sorsa et al., 2018).

However, there are still some trade-offs between the two systems. In practical operation, the destination control system usually needs to immediately display the assigned elevator on the panel after the passenger registers the destination. This means that when new passenger requests are registered, even if the initial dispatching is no longer optimal, it cannot be revised. On the contrast, despite the conventional control system's limitation of having less information, it does not need to inform passengers of the assigned elevator. Therefore, when new requests emerge, the algorithm can reconsider the assignments to ensure that a new optimal solution can be determined.

To the best of our knowledge, there is relatively little research focused on hybrid control systems that combine the advantages of both types of systems. We consider that destination control can reduce unnecessary stops and wait times by grouping passengers with similar destinations in the same elevator, which can better meet passenger needs on high-demand floors. In low-demand floors, however, operational costs and efficiency considerations may not be as critical. It might be worth considering using the conventional control system on these floors to retain a certain degree of flexibility and adaptability.

doi:10.6342/NTU202502477

1.2 Research Objectives

In our research, we investigate whether a hybrid control system consisting of conventional and destination control systems can be efficient in serving passengers. Moreover, we develop a dispatch algorithm suitable for all kinds of control system. In a hybrid control system, the type of control system is categorized by floor settings; we predefine certain floors to use conventional calls and others to use destination calls. Each request data has five attributes: the time of birth, the number of passengers, the starting floor, the destination floor. If a request is sent from a floor that accepts conventional calls, the system will know the time of birth and the starting floor of the request but does not know the destination floor and the number of passengers. As the assignment of an elevator to such a request is hidden to that passenger, the dispatch result can be changed until the passengers enter the designated elevator. In contrast, for floors set to accept destination calls, the system will know all the details of the request. Once the algorithm makes the decision, the result cannot be changed.

In the design of the dispatch objective, we primarily focus on enhancing building efficiency while maintaining a certain level of passenger service. We aim to minimize the total energy consumption cost of the elevators and the number of passengers with prolonged waiting periods. By setting a threshold for long waiting times, calls with waiting times exceeding this threshold will result in a penalty on the objective. We hope that our dispatch results will not overly prioritize efficiency at the expense of passenger benefits.

doi:10.6342/NTU202502477

1.3 Research Plan

Our research plan is organized as follows. In Chapter 2, we review some past research regarding the elevator dispatch problem. In Chapter 3, a precise description of our problem is introduced. In Chapter 4, we will propose an algorithm to solve a sequence of static problem in response to dynamic issues, and we will prove that the algorithm can achieve the optimal solution within a single static problem. In Chapter 5, we will simulate real-world demand scenarios and conduct numerical experiments to verify that our proposed solution is applicable in various contexts. In Chapter 6, we will introduce the architecture of our dispatching system implementation. Finally, in Chapter 7, we will summarize the results of the entire research and discuss future developments.

doi:10.6342/NTU202502477

Chapter 2

Literature Review

In this chapter, we explore and analyze prior research on elevator dispatching problems to build a solid theoretical foundation for our study. In Section 2.1, we discuss the elevator dispatching problem by exploring approaches that treat it as a dynamically changing problem as well as examining it as a sequence of static issues. In Section 2.2 investigates various elevator control systems, providing an introduction of their respective strengths and limitations. Section 2.3, we discuss the optimization objectives frequently targeted in elevator dispatching research.

2.1 Static and Dynamic Approaches in EDP

The elevator dispatching problem involves a sequence of decision-making actions based on different environment states. In a study by Wan et al. (2024), the EDP is modeled as a semi-Markov decision process due to the dynamic nature. They took into account that the transition time between states, such as the time for elevators to move between floors

doi:10.6342/NTU202502477

or to wait for passengers boarding or leaving, is not fixed but instead depends on factors like the distance between floors, passenger demands, and traffic patterns. Moreover, there is randomness in state transitions, as passenger arrival floors and destination are random, leading the system to exhibit different traffic patterns at various times. To better address the challenges brought by randomness, they present a reinforcement learning-based solution that uses neural networks to approximate the decision function.

In some studies, they obtain this dynamic process decision by solving a sequence of static snapshot problems. A common approach is to set a decision-making point, which can be at fixed intervals to solve the problem or triggered by specific decision events, such as the emergence of a new request. Each snapshot problem includes the elevator information and request information at that particular decision-making point. Elevator information encompasses the position of each elevator, the direction of movement, and the remaining space inside the car, while request information depends on the type of control system. In a research conducted by Tyni and Ylinen (2006), the state changes are segmented into several snapshot scenarios along the time dimension. At the current stage, once a snapshot problem is solved, the solution cannot be altered in the subsequent snapshot problems. The new requests are solved based on the information provided by the previous solution. This approach has the benefit of reducing the complexity of the problem, making it easier to achieve a relatively better solution more quickly. The simulation in the research also demonstrate its feasibility over long-term operations. Another study done by Sun et al. (2009) propose the idea of constructing snapshot problems using a rolling horizon scheme. By setting the length of the time window, the range of information considered during optimization is determined. The hybrid nested partitions

doi:10.6342/NTU202502477

and genetic algorithm method is applied to solve each subproblem. The experiments also demonstrated good performance under different traffic patterns.

As far as we know, most studies solve the elevator dispatch problem using the latter method (Fujino et al., 1997; Tyni and Ylinen, 2006; Dai et al., 2010; Zhang et al., 2010; Bolat et al., 2013; Tartan and Ciftlikli, 2016; Ruokokoski et al., 2016; Sorsa et al., 2018; Hanif and Mohammad, 2022). Our research will follow this mainstream approach, defining the decision-making point for each snapshot problem and solving it accordingly.

2.2 Assignment Evaluation in EDP

In conventional elevator control systems, passengers send signals to the elevator through a simple panel on each floor with just up and down buttons. This means that when a ride request is made, the system can only identify the floor with the request and the direction desired, but cannot know the number of passengers and their destination until they enter the elevator. This causes the algorithm to inaccurately account for the drop-off floors of passengers when simulating the elevator's future routes, leading to deviation in estimating passenger waiting times. In the study by Cortés et al. (2004), only the floors where the elevator stops to pick up passengers are considered in waiting time calculations, disregarding passengers' destinations after boarding. The authors propose a rule-based elevator path planning method, offering the advantage of simplicity in computation. Subsequent studies based on conventional control systems have also estimated elevator paths in the similar approach (Bolat et al., 2013; Tartan and Ciftlikli, 2016). These limitations due to missing information are identified as the major challenges to the

doi:10.6342/NTU202502477

efficiency of elevator dispatching in the study by Zheng et al. (2018). In order to tackle this issue without changing hardware, the authors develop a traffic flow prediction model based on machine learning, aiming to better address the optimization challenges posed by uncertainties.

In contrast, the destination elevator control system is designed to collect more detailed information about passengers' requests. According to the introduction by Koehler and Ottiger (2002), this system provides a more sophisticated panel with multiple number buttons, allowing passengers to select their intended destinations in advance. Thus, the dispatch algorithm can assign passengers with the same destination to the same elevator whenever possible and calculate more precise optimization objective values to meet the application demands. Dai et al. (2010) leverage the advantages of this system, enabling it to estimate passenger waiting time, journey time, and the number of elevators stops more accurately than conventional elevator systems, without relying on additional traffic prediction models. In addition, Ruokokoski et al. (2016) utilized the additional information provided by the destination control system to model the EDP as a mixed-integer programming (MIP), thereby obtaining the theoretical optimal solution.

We have observed that there are relatively few studies on the elevator dispatch problem in hybrid control systems. Therefore, in our research, we attempt to combine different types of control systems to determine if we can achieve better results than with a single system.

doi:10.6342/NTU202502477

2.3 Different Objectives of EDP

Elevator dispatching problems usually aim to improve building efficiency, passenger satisfaction or both. From the perspective of enhancing passenger satisfaction, Fujino et al. (1997) suggest that the optimization objectives for elevators should vary based on floor attributes. High-traffic floors should prioritize minimizing waiting times, while low-traffic floors should focus on minimizing elevator car crowdedness. Similar research target was studied by Tartan and Ciftlikli (2016). They suggest a genetic algorithm-based method to minimize passenger journey time, defined as the time spent waiting for the elevator to arrive plus the time spent traveling to the destination inside the elevator.

On the other hand, Zhang et al. (2010) notice that energy-saving issues are gaining increasing attention. Consequently, they choose to focus on improving building resource efficiency, with the objective of minimizing energy consumption. In the estimation of energy consumption, they considered the energy usage during acceleration and deceleration of the elevator, as well as the energy usage during constant speed movement. They also incorporated the weight of the counterweights in the elevator structure and the load conditions of the elevator car into their calculation formula, providing a clear definition of energy consumption. Both studies propose dispatch solutions based on genetic algorithms.

Additionally, there are some studies take both passenger experience and resource efficiency into account. A bi-objective optimization framework was put forward by Tyni and Ylinen (2006). They believe that passenger waiting time and elevator energy consumption are a trade-off. Considering only one aspect might negatively impact the other. Therefore,

doi:10.6342/NTU202502477

they develop a evolutionary standardized objective weighted aggregation method to find the best balance. Also, Bolat et al. (2013) contemplate a multi-objective optimization. The difference is that they primarily focused on the trade-off between passenger journey time and the balance of elevator workload. They attempt to save passengers' time costs and avoid excessive concentration of tasks on some elevators by using a particle swarm algorithm, thereby improving the efficiency of elevator dispatch policy.

In our research, we chose to use multiple optimization objectives to reduce energy consumption costs without significantly lowering passenger service levels, thereby better meeting practical usage needs.

doi:10.6342/NTU202502477

Chapter 3

Problem Description

In this chapter, we present the formulation for the elevator dispatching problem, discuss control system configurations, and outline the assumptions about elevator travel routes.

3.1 Elevator Dispatching Problem

In this section, we provide a detailed description of the elevator dispatch problem. Here, we refer to passengers' requests to reach desired floors as *calls*. Rather than uniformly applying one control system across the entire building, our approach involves configuring distinct control systems on a floor-by-floor basis. If a passenger's request signal originates from the floors designated as conventional control system, it will be termed a *conventional call*. Conversely, if the request comes from destination control system floors, it will be called a *destination call*. Each elevator can respond to either a conventional call or a destination call, with the difference lying in the level of detail of the passenger information.

We consider three distinct building-wide control system configurations: All-Conventional,

doi:10.6342/NTU202502477

All-Destination, and Hybrid. All-Conventional refers to a setup where all floors are designated as conventional control floors. All-Destination signifies that all floors are configured as destination control floors. The Hybrid configuration combines these two approaches, utilizing destination control for some floors and conventional control for all other floors. Here, we assume that calls from conventional control floors are destined for the top floor (or the bottom floor).

Assuming a building has K floors and N elevators, let the set of all floors be F, with a subset F^C representing the conventional floors and another subset F^D representing the destination floors. The set of all elevators is denoted as $E = \{1, 2, ..., N\}$. Each elevator has the same load capacity L, which is the maximum number of passengers it can carry. It is known that the time required to move one floor is T^M and the time required to open and close the door is T^O. Throughout the entire dynamic process, we can get the n-th elevator's direction of travel $\mathbf{d}_{nt}$, the number of passengers inside the car $\mathbf{p}^E_{nt}$, and the floor it is currently on $\mathbf{f}_{nt}$ at time t. Here, $\mathbf{d}_{nt} \in \{-1, 0, 1\}$ be -1 means the elevator is moving down, 0 means it is stationary, and 1 means it is moving up.

All passengers forms a set $\mathcal{C} = \{1, 2, ..., |\mathcal{C}|\}$, where we can obtain the arrival floor of the i-th group of passengers is $\mathbf{f}^A_i$, and the arrival time as $\mathbf{t}^A_i$, where A stands for *arrival*. The time required for a passenger to enter or exit the elevator is T^S. If $\mathbf{f}^A_i$ belongs to F^C, then the call i belongs to the subset $\mathcal{C}^C$, we can know the passenger's destination direction $\mathbf{d}_i$, where $\mathbf{d}_i \in \{-1, 1\}$. A value of 1 indicates an upward direction, while a value of -1 indicates a downward direction. If $\mathbf{f}^A_i$ belongs to F^D, then the call i belongs to the subset $\mathcal{C}^D$, we can determine not only $\mathbf{d}_i$, but also the number of passengers $\mathbf{p}^C_i$ and their destination floor $\mathbf{f}^D_i$. And we define that $\mathcal{C}$ is the union of $\mathcal{C}^D$ and $\mathcal{C}^C$.

doi:10.6342/NTU202502477

For each call $i \in \mathcal{C}$, we need to assign an elevator $n \in E$ to serve it. Therefore, we define the decision variable $\mathbf{x}_{ni} \in \{0, 1\}$, where a value of 1 indicates that elevator n responds to call i. All decision variables $\mathbf{x}_{ni}$ form a $|E| \times |\mathcal{C}|$ decision matrix $\mathbf{X}$, which is represented as

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \cdots & \mathbf{x}_{1,|\mathcal{C}|} \\ \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \cdots & \mathbf{x}_{2,|\mathcal{C}|} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{|E|,1} & \mathbf{x}_{|E|,2} & \cdots & \mathbf{x}_{|E|,|\mathcal{C}|} \end{pmatrix}. \tag{3.1}$$

Each call must be assigned to exactly one elevator, we have the constraints

$$\sum_{n \in E} \mathbf{x}_{ni} = 1, \forall i \in \mathcal{C}. \tag{3.2}$$

Once a destination call $i \in \mathcal{C}^D$ is assigned to a specific elevator, it cannot be changed. However, a conventional call $i \in \mathcal{C}^C$ can be reassigned to another elevator as long as the task has not yet been executed.

Our optimization objective is to maintain the quality of elevator service while minimizing energy consumption costs. The energy consumption costs $u(\mathbf{X}_{n,.})$ of elevator n operation can be divided into two parts: the energy consumed during full-speed travel and the energy consumed during acceleration and deceleration for stopping (Zhang et al., 2010; Hanif and Mohammad, 2022). We define the former as moving energy consumption $u^M(\mathbf{X}_{n,.})$, where M stands for *moving*, and the latter as stopping energy consumption $u^S(\mathbf{X}_{n,.})$, where S stands for *stopping*.

doi:10.6342/NTU202502477

According to the study by Tyni and Ylinen (2006), the elevator systems is in a balanced state when the weight of the car during operation equals the weight of the counterweight. Typically, the counterweight equals the weight of the empty car plus half of maximum load limit. Therefore, when the passenger weight inside the car is exactly half of the maximum load limit, the elevator is in its most energy-efficient state, meaning $u^M(\mathbf{X}_{n,\cdot})$ will be at its minimum value (Zheng et al., 2018). Thus, we express $u^M(\mathbf{X}_{n,\cdot})$ as a function related to the load weight and moving distance, and directly regard the load weight as the number of passengers. When the number of passengers inside the car accounts for 50% of L, $u^M(\mathbf{X}_{n,\cdot})$ reaches its minimum value, as shown in the Figure 3.1. While $u^S(\mathbf{X}_{n,\cdot})$ is set as a function that is positively linearly related to the number of stops, meaning each stop incurs a fixed unit cost.

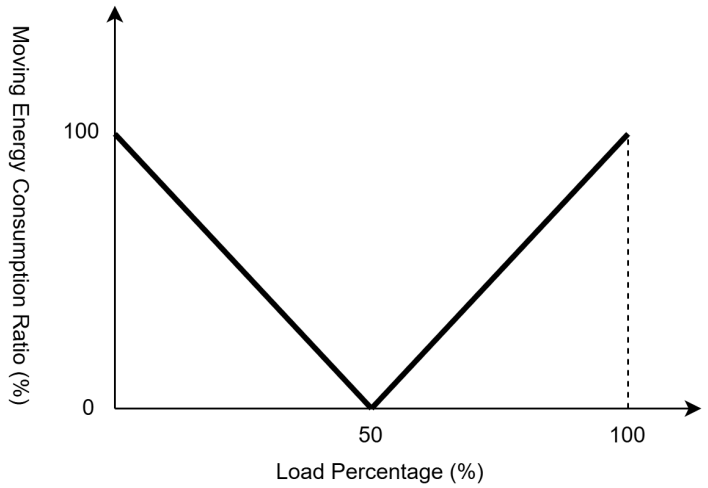


Figure 3.1: Relation of Load and Moving Energy Consumption

In addition, we evaluate the quality of elevator service by passengers' waiting time, which is the period from when a passenger arrives at the lobby and makes a call to when they enter the elevator. There is a trade-off between passengers' waiting time and elevators' operation costs (Tyni and Ylinen, 2006). We consider that most passengers

doi:10.6342/NTU202502477

find short waiting time acceptable, and only become dissatisfied when the waiting time is too long. Therefore, we set a waiting time threshold θ to distinguish between reasonable waiting and excessive waiting. We aim to minimize the number of passengers experiencing excessively long waiting times. Let $E(i)$ as the elevator that serves call i, and $w(\mathbf{X}_{E(i),.})$ represent the actual waiting time of the passenger in the i-th call, which is a function. We use a variable y_i indicate whether this passenger's waiting time exceeds θ, then

$$y_i = \begin{cases} w(\mathbf{X}_{E(i),.}) - \theta & \text{if } w(\mathbf{X}_{E(i),.}) > \theta \\ 0 & \text{otherwise} \end{cases}. \tag{3.3}$$

In conclusion, the objective function can be expressed as

$$\min \quad \alpha \cdot \frac{1}{|E|} \sum_{n \in E} u(\mathbf{X}_{n,.}) + \beta \cdot \frac{1}{|\mathcal{C}|} \sum_{i \in \mathcal{C}} y_i, \tag{3.4}$$

where the $u(\mathbf{X}_n, .)$ is the energy cost generated by the elevator n after serving all the calls assigned to it, which is also a function. α and β are the weights that balance energy cost and excessive waiting penalties.

For this dynamic process problem, similar to most papers (Fujino et al., 1997; Tyni and Ylinen, 2006; Dai et al., 2010; Zhang et al., 2010; Bolat et al., 2013; Tartan and Ciftlikli, 2016; Ruokokoski et al., 2016; Sorsa et al., 2018; Hanif and Mohammad, 2022), we will derive dispatch decisions for each call by solving a sequence of static problems. We hope that through this method, we can achieve a satisfactory long-term solution.

We summarize the notations mentioned in Table 3.1.

doi:10.6342/NTU202502477

Sets	
F	Set of floors. $F = F^C \cup F^D$
E	Set of elevators.
$\mathcal{C}$	Set of calls. $\mathcal{C} = \mathcal{C}^C \cup \mathcal{C}^D$
Parameters	
K	Number of floors.
N	Number of elevators.
L	Load capacity of elevators.
T^M	The time required of elevators to move one floor.
T^O	The time required of elevators to open and close door.
T^S	The time required of a passenger to enter or exit the elevator.
$\mathbf{f}_i^A$	Arrival floor of the i-th call, $\mathbf{f}_i^A \in F$.
$\mathbf{t}_i^A$	Arrival time of the i-th call.
$\mathbf{d}_i$	Destination direction of the i-th call, $\mathbf{d}_i \in \{-1, 1\}$.
$\mathbf{f}_i^D$	Destination floor of the i-th call, $\mathbf{f}_i^D \in F$.
$\mathbf{p}_i^C$	Number of passengers in the i-th call.
Decision variables	
$\mathbf{x}_{ni}$	1 if assign elevator n to serve call i or 0 otherwise.
$\mathbf{d}_{nt}$	Moving direction of elevator n at time t, $\mathbf{d}_{nt} \in \{-1, 0, 1\}$.
$\mathbf{f}_{nt}$	Position of elevator n at time t, $\mathbf{f}_{nt} \in F$.
$\mathbf{p}_{nt}^E$	Number of passengers in elevator n at time t.

Table 3.1: List of Sets, Parameters and Decision Variables

doi:10.6342/NTU202502477

3.2 Assumption for Elevator Routing

To evaluate the service performance of each call, we must simulate the elevator's route in advance. The behavior of elevator service for calls follows several constraints mentioned in the paper by Closs (1970), which are also followed by many related studies (Tartan and Ciftlikli, 2016; Sorsa et al., 2018; Siikonen, 2024).

Specifically, when passengers are inside, the elevator must not change direction. All calls will be serviced in the sequence of the elevator's direction of travel. Passengers may not allowed to enter elevators that are moving in a direction opposite to their intended destination, and the elevator must not bypass the destination floors of passengers inside. Moreover, for all calls belong to destination calls ($i \in \mathcal{C}^D$), passengers have to enter the prescribed elevator given by dispatcher.

In our research, we do not optimize the service sequence. When an elevator receives multiple assigned calls, it first serves the calls that are in the same direction and along its current path sequentially. It then turns to serve calls in the opposite direction, and finally serves the remaining calls that are in the same direction but not along its current path (Cortés et al., 2004).

For example, as shown in the Figure 3.2 (a), consider an elevator currently on the 2nd floor, moving upward. It needs to serve three calls: c_1 for the 1st floor going up, c_2 for the 6th floor going up, and c_3 for the 8th floor going down. Under this rule, the elevator will first serve c_2, which is in the same direction and along its upward path, then c_3, which is in the opposite direction, and finally return to serve c_1. Another similar example, but

doi:10.6342/NTU202502477

with the elevator initially moving downward, is shown in Figure 3.2 (b). Thus, once the calls assigned to an elevator are determined, the elevator's operating route for servicing these calls is also determined.

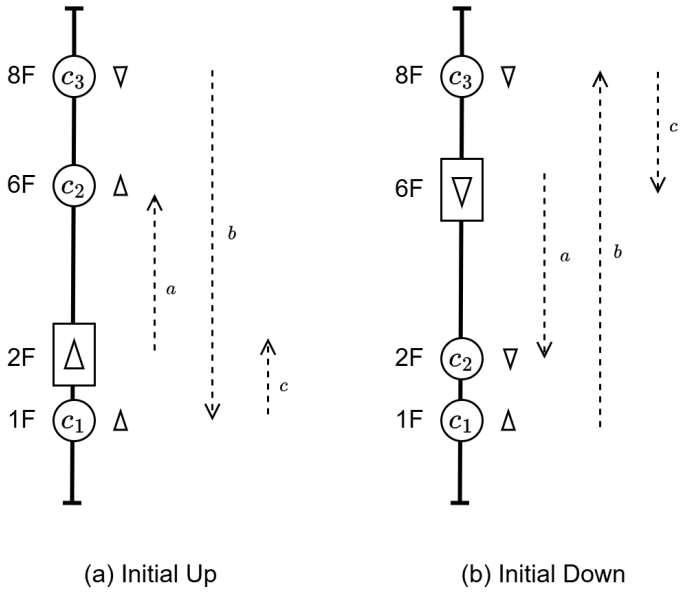


Figure 3.2: Service Sequence of Elevator

In the following chapters, we will discuss the decision-making time points, such as at fixed intervals or when new passengers arrive, forming snapshot problems and modeling them to find a good solution at each snapshot.

doi:10.6342/NTU202502477

Chapter 4

Solution Approach

Given a dynamic instance, our solution approach involves forming and solving a sequence of static instances. This chapter will discuss how we define a static problem, define the solution structure and algorithms for solving it, and finally measure the experimental performance of our proposed solution method in static instances.

4.1 Static Problem Formulation

To transform a dynamic problem into a sequence of static problems, we need to determine the decision events (e.g., a new passenger arrival) or fixed decision intervals (e.g., every 5 seconds) at which the system state will be captured—known as the decision point. At each decision point, we record all system information to create a static snapshot of the problem. In this problem, related to elevators, we can get the current position f_n, direction of movement d_n, and number of passengers p_n^E of the n-th elevator. Regarding passengers, we collect the newly generated calls during the time interval from the previous

doi:10.6342/NTU202502477

decision point, forming the request set $C = \{1, 2, ..., |C|\}$. We can get the desired direction d_j and the arrival floors f_j^A of the j-th calls in C. Additionally, we can obtain information about the number of passengers p_j^C and their destination floor f_j^D from the calls in C^D. For the calls in C^C, we assume p_j^C as 1, and the f_j^D remains unknown.

Moreover, we refer to the set of calls that were generated prior to the previous decision point and still exist in the system as $\tilde{C} = \{1, 2, ..., |\tilde{C}|\}$. The decision results $\tilde{x_{ni}}$ for the i-th call in $\tilde{C}$ from solving the previous snapshot problem are also given parameters for this snapshot problem. We define *outer calls* as the set $\tilde{C}^O$, representing the passengers' requests for an elevator made in the lobby; and *inner calls* as the set $\tilde{C}^I$, representing the passengers' requests for a destination floor made inside the elevator. When passengers enter the elevator, their outer call is converted to an inner call, indicating the elevator has picked them up. Thus, $\tilde{C}$ can be expressed as $\tilde{C}^C \cup \tilde{C}^D$ or $\tilde{C}^O \cup \tilde{C}^I$. When a passenger arrives at their destination, they disappear from the system and are simultaneously removed from the set $\tilde{C}$. The relationship between call sets C, $\tilde{C}$, and decision points is shown in Figure 4.1.

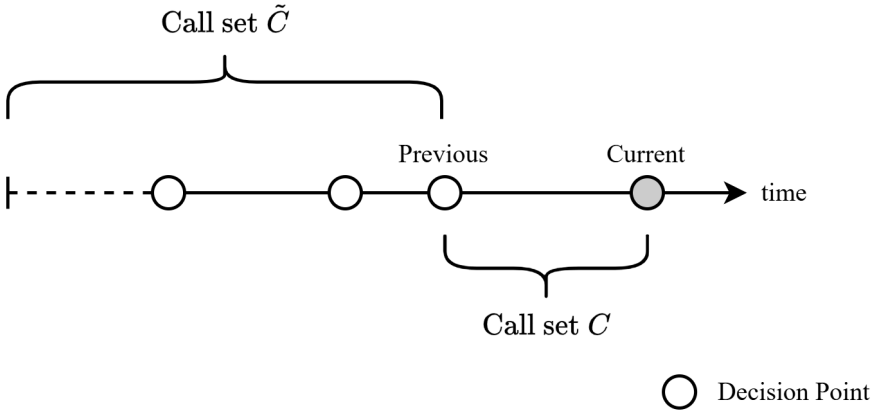


Figure 4.1: Illustration of Call Collection

doi:10.6342/NTU202502477

In our algorithm, we can modify the previous dispatching results for passengers who have not entered the elevator yet and belong to conventional calls. Furthermore, due to mechanical and physical operation constraints, the scheduled stops within two floors of the elevator's current position while it is moving cannot be altered. We denote $\tilde{n}(i)$ as the index of elevator n when $\tilde{x}_{ni} = 1$ for a given call i. Consequently, for the i-th call in $\tilde{C}^O \cap \tilde{C}^C$, if the difference between f_i^D and $f_{\tilde{n}(i)}$ is more than two floors, and the elevator's operating state $d_{\tilde{n}(i)}$ is stationary, the dispatch result $\tilde{x}_{ni}$ can be re-optimized. We denote the re-dispatchable demand set within $\tilde{C}$ as $\tilde{C}^R$. The definition of $\tilde{C}^R$ can be written as

$$\tilde{C}^R = \{i \in \tilde{C}^O \cap \tilde{C}^C \mid (d_{\tilde{n}(i)} \in \{1, -1\} \wedge |f_i^D - f_{\tilde{n}(i)}| > 2) \vee (d_{\tilde{n}(i)} = 0)\}. \tag{4.1}$$

		Conventional	**Destination**
$\tilde{C}$	**Outer**	Eq.(4.1) is satisfied ($\tilde{C}^R$): O	X
	Inner	X	X
C	**Outer**	O	O

Table 4.1: Table of Dispatchable Attribute

In summary, we have segmented the call sets C and $\tilde{C}$ into two dimensions, as organized in Table 4.1. In this table, 'O' indicates that the call set is dispatchable, while 'X' signifies that it is not dispatchable. Thus, for the current snapshot problem, we have to determine the decision variable x_{nk} for the k-th calls in the dispatchable call set, written as $C \cup \tilde{C}^R$. Based on the given information, we need to ensure that the final solution x_{nk} does not result in the number of passengers in the elevator exceeding its maximum load

doi:10.6342/NTU202502477

limit L. Therefore, the capacity constraint can be expressed as

$$p_n^E + \sum_{i \in \tilde{C}^O \setminus \tilde{C}^R} \tilde{x}_{ni} \cdot p_i^C + \sum_{k \in C \cup \tilde{C}^R} x_{nk} \cdot p_k^C \leq L, \forall n. \tag{4.2}$$

Additionally, we need to guarantee that each passenger is served by only one elevator. Therefore, the service constraint can be written as

$$\sum_{n \in E} x_{nk} = 1, \forall k \in C \cup \tilde{C}^R. \tag{4.3}$$

In our evaluation, we primarily consider two indicators: the energy consumption of elevator operations and passenger waiting times. When calculating the indicators, we estimate the elevator's movement path based on the rules outlined in Section 3.2. We need to account for the fact that when the elevator stops to pick up passengers, its direction of movement must be consistent with the passengers' desired direction. Therefore, for the indicators up to the point when passengers enter the elevator, we calculate until the designated elevator's stop floors and operating direction match the specified demand. After passengers enter the elevator, the indicators no longer need to consider the consistency of the direction. We aim for these computation indicator values to be as close as possible to the actual performance results of the simulation system. These computation indicator values are derived from a formula denoted as $\hat{w}(x_{E(i),\cdot})$ and $\hat{u}(X_{n,\cdot})$.

For the sake of comparison, we will use a composite formula, referred to as the objective function, to combine these two indicators into a single comprehensive evaluation value. We calculate the waiting time excess penalty $\hat{y}_i$ for the i-th call within a static

doi:10.6342/NTU202502477

problem using Equation (3.3). The $\hat{y}_i$ is written function as

$$\hat{y}_i = \begin{cases} \hat{w}(X_{E(i),.}) - \theta & \text{if } \hat{w}(X_{E(i),.}) > \theta \\ 0 & \text{otherwise} \end{cases}. \tag{4.4}$$

The objective function is written as

$$\min \quad \alpha \cdot \frac{1}{|E|} \sum_{n \in E} \hat{u}(X_{n,.}) + \beta \cdot \frac{1}{|C \cup \tilde{C}^R|} \sum_{k \in C \cup \tilde{C}^R} \hat{y}_i. \tag{4.5}$$

4.2 Solution Structure and Constraint Handling

In our proposed method, based on a study by Tartan and Ciftlikli (2016), the structure of the solution is an array of length l, with each element contains the identifier of dispatchable calls. The length l is related to the number of unique dispatchable calls in a problem, and the definition of a unique call varies depending on the call mode. In the conventional call setting, if a call from the same floor and in the same direction already exists and has not yet been served, any new passengers appearing during this period will be considered part of the same call group until it is served, after which a new call can occur. In the destination call setting, regardless of whether a call from the same floor and in the same direction exists, all new passengers will be considered an independent call.

To generate different solutions, we write the identifier of the n-th elevator into each cell, representing the assignment of the n-th elevator to the floor, while meeting its

doi:10.6342/NTU202502477

direction requirement. This structure naturally satisfies the service constraint mentioned in Equation (4.3). For example, if there are five unique dispatchable calls waiting for elevators in the system, these calls are respectively: floor 1 going up, floor 2 going down, floor 5 going up, floor 5 going down, and floor 6 going down. These calls are sequentially indexed as $c1$, $c2$, $c3$, $c4$, and $c5$, respectively. There are a total of 3 elevators, with identifiers $e1$, $e2$, and $e3$. Our method will then construct a one-dimensional array of length 5, as shown in Figure 4.2 (a), and will generate multiple solutions by writing the elevator identifiers into cells, as shown in Figure 4.2 (b).

$c1$	$c2$	$c3$	$c4$	$c5$

(a) Solution Structure

$c1$	$c2$	$c3$	$c4$	$c5$
$e1$	$e2$	$e3$	$e3$	$e1$

(b) Example Solution

Figure 4.2: Solution Structure in a Static Problem

After generating a group of solutions, we need to avoid assigning too many calls to a single elevator. If an elevator reaches the specific floor and the number of passengers exceeds the maximum capacity limit L, passengers will be unable to enter the elevator, resulting in an invalid stop. This not only increases the waiting and travel time for all passengers but also causes unnecessary energy consumption costs. To minimize the possibility of this issue, we estimate the number of passengers that each candidate solution will bring to each elevator and use Equation (4.2) to check whether the load exceeds the

doi:10.6342/NTU202502477

maximum limit L.

However, in practical scenarios, if an elevator is temporarily unable to pick up a designated passenger due to being full, it can simply proceed to deliver the current passengers to their destinations and then return to pick up the previously missed passenger. Therefore, such a situation is not strictly an infeasible solution but rather one that requires more time to complete the task. Consequently, if a solution fails to satisfy Equation (4.2), we will assign a sufficiently large penalty value M to the objective function to reduce the probability of the algorithm selecting this solution as the final one. We let a binary variable $\hat{o}_n$ represent whether the estimated future load of elevator n exceeds the load limit, expressed as

$$\hat{o}_n = \begin{cases} 1 & \text{if} \quad p_n^E + \sum_{i \in \tilde{C}^O \setminus \tilde{C}^R} \tilde{x}_{ni} \cdot p_i^C + \sum_{k \in C \cup \tilde{C}^R} x_{nk} \cdot p_k^C > L \\ 0 & \text{otherwise} \end{cases}. \tag{4.6}$$

Then, the objective function can be rewritten as

$$\min \quad \alpha \cdot \frac{1}{|E|} \sum_{n \in E} \hat{u}(X_{n,\cdot}) + \beta \cdot \frac{1}{|C \cup \tilde{C}^R|} \sum_{k \in C \cup \tilde{C}^R} \hat{y}_i + M \cdot \sum_{n \in E} \hat{o}_n. \tag{4.7}$$

In the next section, we will elaborate on obtaining an acceptable solution for a static problem using different algorithms.

doi:10.6342/NTU202502477

4.3 Algorithms for Solving the Static Problem

This section details the two algorithmic approaches employed to solve the static elevator dispatching problem. We utilize both an exhaustive algorithm (EA), which guarantees the optimal solution for small-scale problems by exploring the entire solution space, and a genetic algorithm (GA), a metaheuristic approach designed to find a good enough solutions within a acceptable computation time range for more complex scenarios.

4.3.1 Exhaustive Algorithm

The elevator dispatching problem is a combinatorial optimization problem with NP-hard characteristics. Additionally, our objective function, Equation (4.7), cannot be expressed in an explicit mathematical form, making it impossible to calculate the gradient, thus making gradient methods difficult to use.

Due to the nature of combinatorial optimization problems, in small-scale instances, we can exhaustively enumerate all possibilities to form a finite solution space. By calculating the objective function value for each solution within this space, we can identify the solution that yields the minimum objective function value as the theoretical optimal solution. Based on the solution structure defined in Section 4.2, the size of this space is related to the number of unique dispatchable calls and the number of elevators. Assuming a problem with k unique dispatchable calls and n available elevators, the total number of possibilities will be n^k. The time it takes for the algorithm to find the optimal solution will increase with the size of the search space.

doi:10.6342/NTU202502477

Within problem scales where the computation time is acceptable, we can employ an exhaustive algorithm to determine the optimal dispatching performance for the same set of demands across different control systems, allowing for a comparison of their advantages and disadvantages.

4.3.2 Genetic Algorithm

When dealing with large-scale problems, it may require exponential time to solve by exhaustive algorithm. Given that it is impractical for passengers to bear the time cost of waiting for EGCS to compute the optimal solution, we choose to use metaheuristic algorithms. While these algorithms cannot guarantee an optimal solution, they typically find sufficiently good solutions within a reasonable time frame, meeting the needs of real-world applications.

Among the many classical metaheuristic algorithms, we apply the genetic algorithm (GA) to our problem. In the Elevator Dispatching Problem (EDP), a single decision (e.g., reassigning a passenger request to another elevator) can lead to drastic and nonlinear chain reactions affecting the overall elevator operation paths, passenger waiting times, and energy consumption . Consequently, a "neighboring solution" that appears promising locally (i.e., a solution derived from only minor modifications to the current one) may not perform as well as expected globally. This characteristic makes algorithms that overly rely on iterating through neighboring solutions for optimization, such as local search or greedy algorithms, unsuitable for EDP. By utilizing crossover and mutation to explore different directions, GA increases the chance of finding the optimal solution.

doi:10.6342/NTU202502477

We aim to use GA to find a sufficiently good solution from a vast number of possibilities within a short time frame. Our chromosome structure adheres to the solution structure defined in Section 4.2. In our proposed method, during crossover, we randomly select a cut point and exchange gene segments between two chromosomes, effectively changing elevator identifiers to generate new candidate solutions. For example, assuming we have two chromosomes, P_1 and P_2, each with a length l. We randomly select a cut point c and exchange the gene segments after this point to generate two new offspring chromosomes, O_1 and O_2. These new offspring can be expressed as[1]

$$O_1 = (P_1[1:c], P_2[c+1:l]) \text{ and} \tag{4.8}$$

$$O_2 = (P_2[1:c], P_1[c+1:l]). \tag{4.9}$$

Subsequently, we define a parameter, k, the mutation rate, which dictates that each offspring has a $k\%$ probability of undergoing mutation. During mutation, we randomly select a gene locus within a chromosome and change its elevator identifier to a different elevator identifier to produce new candidate solutions. For example, assuming we have an offspring O, we randomly select a gene locus, i, and change its elevator identifier to a different elevator identifier e', thereby generating a new individual O', which can be represented as

$$O' = (O[1], O[2], \ldots, O[i-1], e', O[i+1], \ldots, O[l]). \tag{4.10}$$

[1]The [ ] operator, in the expression for equation, serves to access and extract gene segments from the chromosomal array.

doi:10.6342/NTU202502477

In our implementation of the genetic algorithm, we employ roulette wheel selection as the method for choosing parent chromosomes for the crossover operation. This probabilistic selection strategy assigns each chromosome a probability of being chosen proportional to its fitness value, effectively giving fitter chromosomes a higher chance of contributing to the next generation. The algorithm continues to iterate through the evolutionary process until a predefined maximum number of iterations is reached, at which point the search for a satisfactory solution is terminated.

Finally, for fitness evaluation, we directly use Equation (4.7) as the fitness function, carrying over the objective of minimizing the objective function value to minimizing the fitness value, thereby selecting chromosomes with relatively superior performance.

In the next section, we will conduct experiments with different sets of parameters to evaluate the performance of the algorithm.

4.4 Performance Evaluation for the Static Problem

4.4.1 Experiment Setting

In our experimental environment, the basic setup is a building with 15 floors, equipped with several elevators controlled by the same Elevator Group Control System (EGCS). Passengers have varying space, origin floor, and destination floor requirements. The walking time for each individual passenger to enter or exit the elevator follows a uniform distribution between 1 and 2 seconds. Detailed parameter settings for the building and elevators are shown in Table 4.2. The energy consumption of an elevator moving one

doi:10.6342/NTU202502477

floor is related to the direction of travel and the load inside the elevator. We simplify the elevator's load limit to a maximum passenger capacity L, and consider the movement energy consumption as an array of length $L+1$. If the current number of passengers inside the elevator is x, the $(x+1)$-th element of this array is used as the energy consumption value for moving one floor.

Building Configuration	
Number of available floors	15
Lowest floor	1F
Highest floor	15F
Elevator Configuration	
Maximum capacity	9 persons
Door open time	3 seconds
Door close time	3 seconds
Move time / floor	4 seconds
Energy consumption / floor (up)	[10.5, 8.5, 6.5, 5.6, 5.2, 5.6, 6.8, 8.3, 10.6, 13] units
Energy consumption / floor (down)	[13, 10.6, 8.3, 6.8, 5.6, 5.2, 5.6, 6.5, 8.5, 10.5] units
Energy consumption / stop	2 units

Table 4.2: Building and Elevator System Specifications for Static Simulation

We have three different control variables: 'Number of Elevators (E),' 'Probability of Passengers Inbound on the Ground Floor (PI),' and 'Probability of Passengers Outbound on the Ground Floor (PO).' Since the ground floor cannot be both the origin and destination of the same trip, these probabilities are defined conditionally. Specifically, PI represents the probability that a passenger's origin floor is the ground floor, given

doi:10.6342/NTU202502477

that their destination is not the ground floor. Conversely, *PO* represents the probability that a passenger's destination floor is the ground floor, given that their origin floor is not the ground floor.

Based on different combinations of these variables, we define a set of distinct experimental scenarios, each designed to simulate specific traffic patterns and system configurations. The sets of scenario experiments as shown in Table 4.3. Starting with a baseline scenario (Scenario 0), which is designed based on the building environment of the case study presented in a later chapter, we individually adjusted the value of each variable to create a total of seven scenarios. For example, the first set of scenarios focuses on the impact of elevator quantity: in Scenario 1-Low (1L), we reduced the number of elevators from 3 to 2, while in Scenario 1-High (1H), we increased the number from 3 to 6. Each scenario comprises 50 independent instances, with each instance involving 8 passengers simultaneously appearing on different floors requiring elevator service.

Scenario	E	PI	PO
0	4	0.5	0.5
1L	**2**	0.5	0.5
1H	**6**	0.5	0.5
2L	4	**0.2**	0.5
2H	4	**0.8**	0.5
3L	4	0.5	**0.2**
3H	4	0.5	**0.8**

Table 4.3: Experimental Scenarios

doi:10.6342/NTU202502477

We will use the solution obtained by the exhaustive algorithm as the optimal solution to measure the difference between the genetic algorithm's solutions and the optimal solution. Since the instances are independent of each other, there is no re-dispatching factor present in this set of experiments. The objective function of the exhaustive algorithm and the fitness function of the genetic algorithm are equivalent, as shown in Table 4.4, both aiming to minimize the objective value.

Parameter	Setting
θ	30
α	1
β	10

Table 4.4: Parameter of the objective function for Single Static Problem

For ease of comparison, our experimental results will be presented not only as raw values but also with the calculated percentage gap between them, using the formula,

$$\text{GAP} = \frac{\text{Result}^{\text{Observed}} - \text{Result}^{\text{Reference}}}{\text{Result}^{\text{Reference}}} \times 100\%. \tag{4.11}$$

We consider three metrics to evaluate results: objective value, total energy, and effective waiting time. The objective value is derived by applying the theoretically optimal dispatching strategy, identified by the dispatch algorithm through fitness value comparison, to the elevator simulation environment. This simulation then yields the actual measured objective value, calculated using Equation (3.4). Therefore, the ideal fitness value derived during the optimization process may not always perfectly align with the actual objective value observed during simulation execution. Energy consumption refers to the actual

doi:10.6342/NTU202502477

energy consumption cost for the entire elevator system to serve all passengers. Effective waiting time is defined as the average time spent by passengers whose actual waiting time exceeds θ seconds, specifically measuring the duration beyond θ.

4.4.2 Performance in the Exhaustive Algorithm

This section details the experimental results obtained for the three elevator control systems: 'All-Conventional', 'All-Destination', and 'Hybrid', under various control parameter settings. These results were derived using an exhaustive algorithm. By comparing the performance achieved by the exhaustive algorithm for each system, we aim to evaluate their respective strengths and weaknesses.

The experimental results for the All-Conventional system are presented in Table 4.5. Utilizing its performance as a reference, we computed the gap for the All-Destination and Hybrid systems relative to All-Conventional across three metrics, as defined by Equation (4.11). The results for the All-Destination and Hybrid systems are listed in Table 4.6.

Across most scenarios and metrics, both the All-Destination and Hybrid systems demonstrate superior performance compared to the All-Conventional system, typically reflected in negative percentage gaps (indicating a reduction in cost or time) for energy consumption and effective waiting time, and lower objective values. This aligns with the expected benefits of more sophisticated elevator control.

All-Destination demonstrates substantial improvements in effective waiting time across most scenarios. Notably, All-Destination shows a significant improvement of -27.7% in Scenario 1H (more elevators). This may suggest that when more elevators are available

doi:10.6342/NTU202502477

for dispatching, All-Destination can utilize resources more efficiently, especially in terms of effective waiting time. Furthermore, All-Destination similarly exhibits superior performance in Scenario 1L (fewer elevators). We postulate that in resource-constrained environments, the value derived from detailed passenger information becomes more pronounced, which could explain the less optimal performance observed in the relatively balanced state of Scenario 0.

Additionally, experimental results reveal that All-Destination performs better in Scenario 3L (low outbound probability) than in Scenario 3H (high outbound probability). This suggests that the value of destination information is leveraged more effectively when fewer passengers boarding on non-ground floors are destined for the ground floor. Furthermore, Scenario 3H exhibits superior performance compared to Scenario 0. This improvement can be attributed to its continued ability to utilize the benefits of destination information for upward-bound passengers, even when outbound traffic to the ground floor is high.

Despite its overall superior performance, the All-Destination system still exhibits positive gaps in certain scenarios, for example, an objective gap of 6.5% in Scenario 2L and 4.5% in Scenario 0. These cases suggest that under specific traffic conditions, the complexity of the All-Destination system' s dispatching may occasionally result in slightly higher costs or waiting times compared to the simpler All-Conventional system. We hypothesize that this could be due to the All-Destination system's lack of decision flexibility; in highly optimized planning scenarios, any momentary deviation in elevator operation from the intended plan could lead to a decrease in overall efficiency.

The Hybrid system also shows improvements, though its gaps are often less pro-

doi:10.6342/NTU202502477

nounced than All-Destination's for objective value, indicating that in most scenarios, the advantage of having more information from the destination floor allows it to perform better than All-Conventional. Notably, in Scenario 0, 2L, and 3L, it slightly outperforms All-Destination. This may suggest that when passenger arrivals and departures are less concentrated on specific floors, Hybrid can leverage the advantage of conventional floor redispatching to achieve better performance than All-Destination. We can find that even by designating only one floor as the destination floor, Hybrid can gain advantages in most scenarios and does not cause significant losses in less favorable conditions.

Scenario	Objective Value	Energy consumption	Effective Waiting Time
	All-Conventional	All-Conventional	All-Conventional
0	171.8	349.9	8.4
1H	118.9	300.5	6.9
1L	460.0	326.0	29.7
2H	128.3	308.4	5.1
2L	195.7	368.6	10.4
3H	180.9	322.3	10.0
3L	178.3	321.2	9.8

Table 4.5: Experimental Results for All-Conventional (EA) in Static Instances

4.4.3 Convergence Analysis of Genetic Algorithm

In each configured static instance, the most extreme scenario necessitates the simultaneous processing of 8 demand requests. This implies a maximum solution space equivalent to the number of elevators raised to the power of 8. To establish a more rational termi-

doi:10.6342/NTU202502477

Scenario	Objective Value		Energy consumption		Effective Waiting Time	
	All-Desti.	Hybrid	All-Desti.	Hybrid	All-Desti.	Hybrid
0	179.6 (4.5%)	177.4 (3.2%)	348.7 (−0.3%)	340.9 (−2.6%)	9.2 (9.6%)	9.2 (9.2%)
1H	86.0 (−27.7%)	107.2 (−9.9%)	298.3 (−0.7%)	296.8 (−1.2%)	3.6 (−47.3%)	5.8 (−16.2%)
1L	426.6 (−7.3%)	440.0 (−4.4%)	325.5 (−0.1%)	325.7 (−0.1%)	26.4 (−11.2%)	27.7 (−6.7%)
2H	111.0 (−13.5%)	124.8 (−2.7%)	306.5 (−0.6%)	300.4 (−2.6%)	3.4 (−32.9%)	5.0 (−2.9%)
2L	208.5 (6.5%)	195.9 (0.1%)	372.2 (1.0%)	371.8 (0.9%)	11.5 (11.5%)	10.3 (−0.7%)
3H	157.6 (−12.9%)	182.4 (0.8%)	329.9 (2.4%)	321.4 (−0.3%)	7.5 (−25.1%)	10.2 (1.7%)
3L	148.5 (−16.7%)	143.8 (−19.3%)	337.4 (5.1%)	326.6 (1.7%)	6.4 (−34.5%)	6.2 (−36.5%)

Table 4.6: Experimental Results for All-Destination and Hybrid (EA) in Static Instances

nation criterion for the genetic algorithm, we selected a single instance from Scenario 0 and subjected it to 100 repeated optimizations using the genetic algorithm. Across these 100 trials, the population size was set to 60, the mutation rate to 0.3, and the maximum iterations to 500. Our primary objective was to observe the generation at which the optimal solution typically converged across the majority of these experiments. The results are presented as a histogram in Figure 4.3.

In our observations from 100 experiments, approximately 80% achieved the optimal solution within 250 iterations, even when the maximum iterations were set to 500. Considering that additional iterations incur increased computational cost with limited marginal

doi:10.6342/NTU202502477

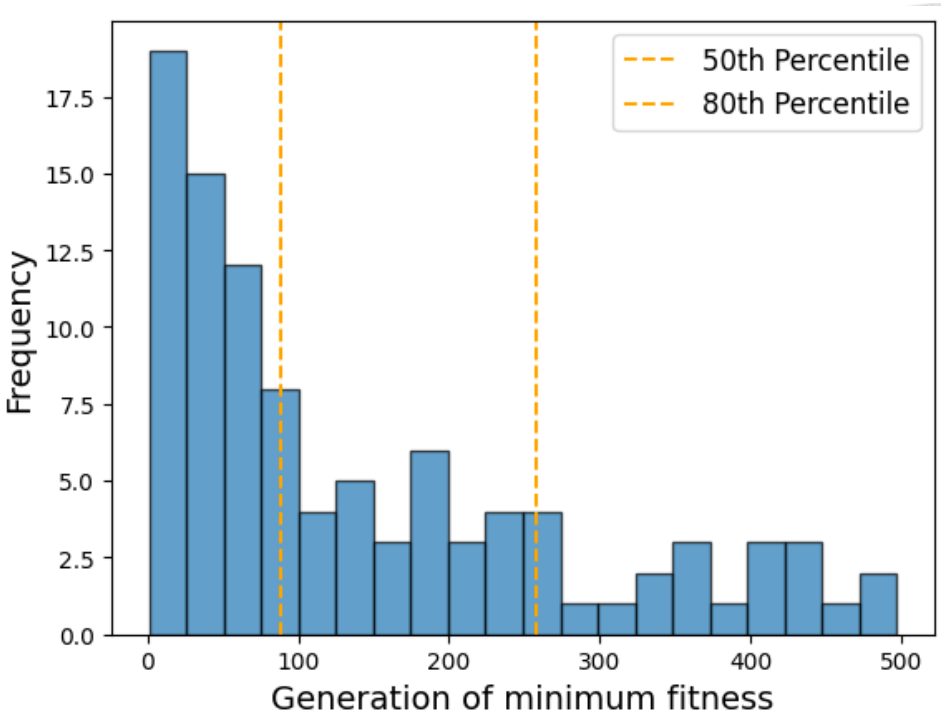


Figure 4.3: Convergence Behavior of Genetic Algorithm over Generations

benefit, we've decided to set the maximum iterations to 250. In summary, the parameters designed for the genetic algorithm are presented in Table 4.7.

Parameter	Setting
Population size	60
Maximum iterations	250
Mutation rate	0.3
Selection method	Roulette Wheel Selection
Termination condition	Maximum Iterations Reached

Table 4.7: Genetic Algorithm Parameter Settings for Single Static Problem

doi:10.6342/NTU202502477

4.4.4 Performance in the Genetic Algorithm

While All-Destination and Hybrid systems demonstrated promising performance in passenger service improvements with the exhaustive algorithm, they incur significant computational costs when dealing with larger problem scales. Therefore, following the preceding experimental scenarios, we transitioned to employing the genetic algorithm to find solutions. Maintaining All-Conventional in exhaustive algorithm as the reference, we calculated the gap for various performance metrics, as presented in Table 4.8.

For the All-Destination system, the GA-driven results are largely competitive with the exhaustive algorithm's performance in most scenarios when compared to the All-Conventional baseline. However, in all scenarios, the energy consumption is higher compared to the results obtained using the exhaustive algorithm, implying that the GA might not always be as efficient as EA in minimizing energy. Nevertheless, it still maintains an advantage in effective waiting time.

Similarly, the Hybrid system optimized by GA shows strong performance, with negative gaps in most scenarios. Its performance, characterized by robustness and balanced optimization, suggests it can be a practical choice, often delivering competitive or even superior results to All-Destination's GA-optimized solutions in specific scenarios, while still harnessing the significant computational efficiency of the GA.

Crucially, the minor deviations from optimality observed in the GA's results are the direct consequence of its primary advantage: computational speed. The very nature of a genetic algorithm means it explores the solution space heuristically, aiming for a "good enough" solution within a feasible time frame, rather than exhaustively searching for the

doi:10.6342/NTU202502477

Scenario	Objective Value		Energy consumption		Effective Waiting Time	
	All-Desti.	Hybrid	All-Desti.	Hybrid	All-Desti.	Hybrid
0	176.5 (2.7%)	170.7 (−0.6%)	355.7 (1.6%)	345.1 (−1.4%)	8.8 (3.8%)	8.4 (0.1%)
1H	67.0 (−43.6%)	100.3 (−15.7%)	303.5 (1.0%)	299.7 (−0.2%)	1.6 (−76.1%)	5.0 (−26.9%)
1L	434.9 (−5.5%)	432.6 (−6.0%)	326.8 (0.2%)	325.9 (−0.0%)	27.2 (−8.6%)	27.0 (−9.2%)
2H	112.0 (−12.7%)	134.5 (4.9%)	313.3 (1.6%)	308.0 (−0.1%)	3.4 (−34.2%)	5.8 (12.4%)
2L	199.2 (1.8%)	186.8 (−4.6%)	377.8 (2.5%)	373.8 (1.4%)	10.5 (1.2%)	9.3 (−9.8%)
3H	177.4 (−1.9%)	158.1 (−12.6%)	330.6 (2.6%)	319.7 (−0.8%)	9.5 (−5.6%)	7.8 (−22.1%)
3L	152.4 (−16.7%)	159.4 (−10.6%)	339.8 (5.8%)	339.9 (5.8%)	6.7 (−31.1%)	6.7 (−24.0%)

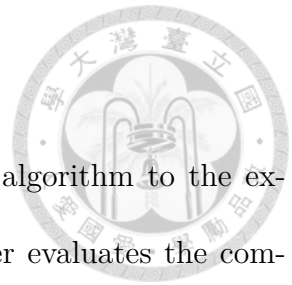
Table 4.8: Experimental Results for All-Destination and Hybrid (GA) in Static Instances

absolute best. This makes it indispensable for real-world applications where obtaining a precise optimal solution might take days, weeks, or even years for complex problems.

Therefore, while the GA may sacrifice a marginal degree of optimality in certain cases, it provides solutions that are highly practical and efficient to compute. The magnitude of this reduction in computational time cost will be specifically detailed in the next section, demonstrating the GA's significant practical value.

doi:10.6342/NTU202502477

4.4.5 Analysis for Computation Cost



Having demonstrated the comparable performance of the genetic algorithm to the exhaustive algorithm in the preceding subsection, this section further evaluates the computational time benefits offered by genetic algorithm. Given that the solution space in elevator dispatching problems is strongly influenced by the number of elevators and the number of unique calls, we build upon a new scenario with 8 unique calls and incrementally increase the number of elevators from 1 to 10. In another set of experiments, we gradually increase the number of unique calls from 1 to 15 while keeping the number of elevators at 3.

We then test the computation time performance of genetic algorithm and exhaustive algorithm for both All-Destination and Hybrid control systems under these varying elevator counts and unique call counts. The results can be visualized in the line graphs as Figure 4.4 and Figure 4.5, respectively. It can be observed from both figures that the curves representing the exhaustive algorithm, regardless of whether it is the All-Destination or Hybrid control system, exhibit an exponential growth in computation time beyond a certain threshold as the x-axis value increases. This time cost is particularly pronounced when the number of elevators increases. In contrast, the genetic algorithm's computation time remains comparatively negligible.

These results demonstrate that while the exhaustive algorithm might be feasible for a very small number of elevators or unique calls, its computational cost becomes prohibitive as the solution space grows. This escalating computational time would directly translate into increased passenger waiting times in a real-world dynamic system. Conversely,

doi:10.6342/NTU202502477

the genetic algorithm, with its inherent limitations on population size and maximum iterations, ensures that computation time remains controlled even as the problem scale expands. While this might involve some sacrifice in solution quality, it prevents the computational overhead from substantially compromising passenger experience. This reinforces the necessity of using the genetic algorithm for real-world elevator systems.

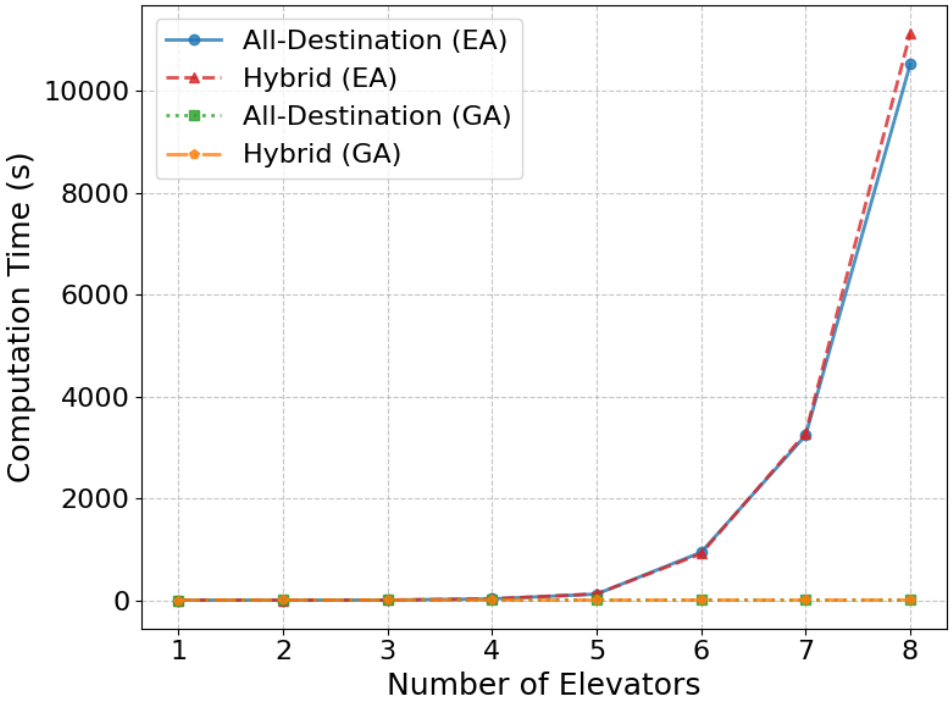


Figure 4.4: Algorithm Computation Time by Elevator Counts

4.4.6 Analysis for Benefit of Destination Information

This subsection aims to quantify the advantages that destination information provides to elevator dispatch systems. We base our experimental data from Scenario 0, with the control system set to All-Destination, and all other parameters remaining consistent with previous settings. The distinction lies in two specific experimental groups: in one, referred to as speculative information, the dispatch system is not provided with passengers' actual

doi:10.6342/NTU202502477

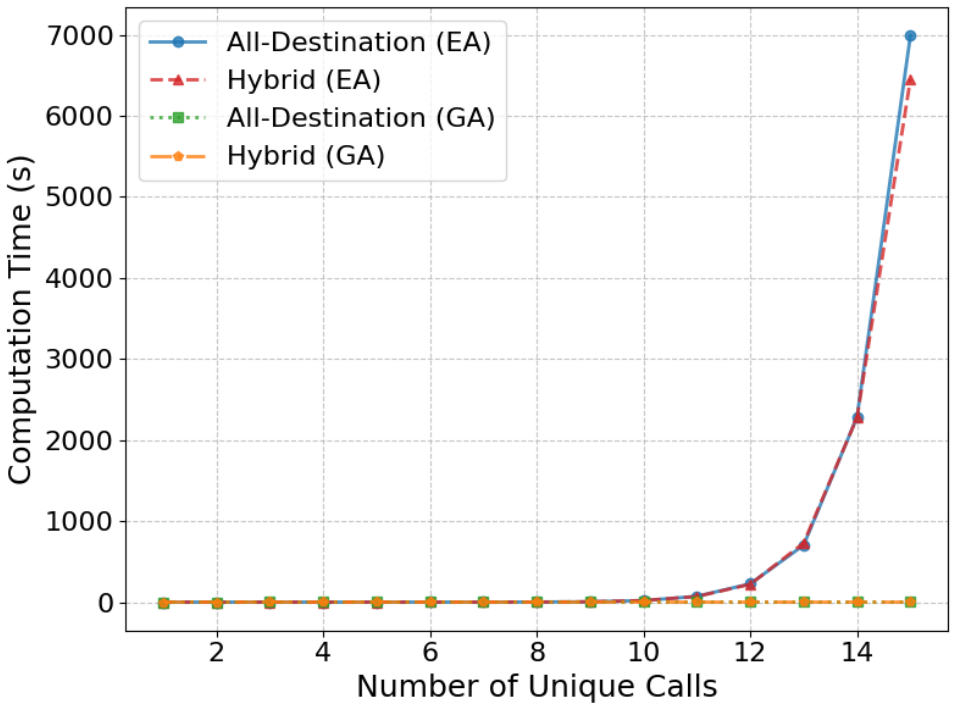


Figure 4.5: Algorithm Computation Time by Unique Call Counts

destination information; instead, it is programmed to assume passengers are traveling to the highest floor (for upward travel) or the lowest floor (for downward travel). The other group, referred to as precise information, provides the dispatch system with complete destination details. By comparing the results from these two experimental groups, we can estimate the benefits that destination information contributes to the dispatch system. The experimental results are presented in Table 4.9, where the percentage gap is calculated based on the performance of the All-Conventional system optimized by the exhaustive algorithm.

Scenario	Objective Value	Energy consumption	Effective Waiting Time
0-Precise	179.6 (4.5%)	348.7 (−0.3%)	9.2 (9.6%)
0-Speculative	186.1 (8.3%)	364.8 (4.2%)	9.5 (12.5%)

Table 4.9: Impact of Information Precision on All-Destination (EA)

doi:10.6342/NTU202502477

Based on the results, the ”0-Precise” scenario consistently exhibits lower gap values across all evaluated metrics compared to ”0-Speculative.” This observation strongly indicates that providing elevator dispatch systems with precise destination information leads to tangible improvements in overall operational performance.

doi:10.6342/NTU202502477

Chapter 5

Performance Evaluation

As mentioned in Chapter 3, we will derive the dispatching strategy for the dynamic problem by solving a sequence of static problems. Furthermore, Chapter 4 detailed how we determine the decision points to segment the dynamic problem into these static instances. Building upon these preceding chapters, this chapter now applies our proposed solution to practical scenarios and examines its effectiveness.

5.1 Solution Architecture

Having established the formulation and solution methods for a static problem, we will now apply this framework to the dynamic scenarios encountered in real-world applications. This involves solving a sequence of static problems at discrete decision points to derive the dispatching strategy for the overarching dynamic problem, as illustrated in Figure 5.1. Here, we adopt a fixed time interval of 1 second for segmenting the dynamic problem into static snapshots. Starting from time zero, we capture the snapshot information

doi:10.6342/NTU202502477

within each 1-second interval, treat it as a static problem, and then find its solution. It is important to note that the time elapsed from a passenger's arrival to the decision point is also included in their waiting time.

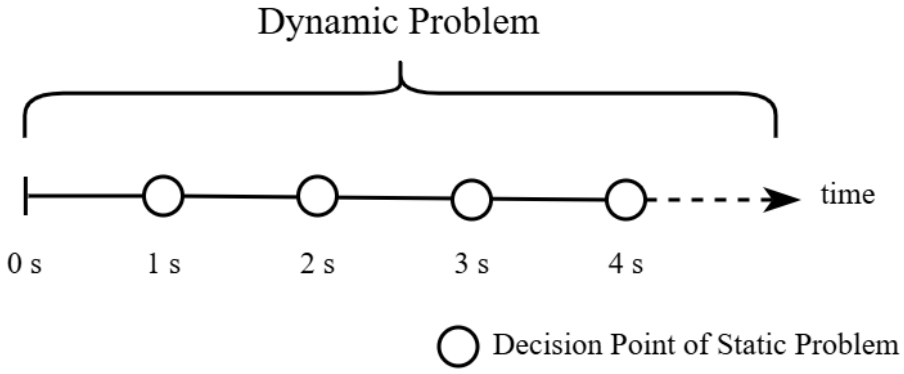


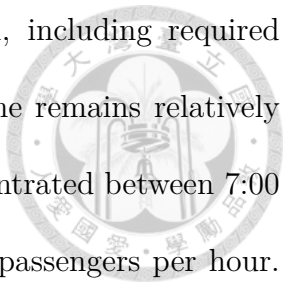
Figure 5.1: Solution Architecture for a Dynamic Problem

5.2 Experiment Setting

To more accurately evaluate the effectiveness of our proposed solution, we collaborated with a company specializing in intelligent elevator solutions to obtain a simulated passenger dataset estimated based on real-world data distributions collected via image recognition, encompassing 40,863 passenger requests over a seven-day period.

In our experimental building environment, we also configured the parameters according to the building information from which this simulated dataset was derived, with the detailed settings presented in Table 5.1. This building, a hotel, comprises fourteen above-ground floors and two subterranean levels. The floor numbering scheme omits the 3rd, 4th, and 13th floors. Vertical transportation within the building is facilitated by four elevators.

doi:10.6342/NTU202502477



This simulated dataset comprises diverse demand information, including required space, origin floor, and destination floor. The overall traffic volume remains relatively consistent across the seven days of the week. Peak demand is concentrated between 7:00 a.m. and 8:59 a.m. (UTC+8) daily, averaging approximately 553 passengers per hour. Off-peak periods occur between 10:00 p.m. and 6:59 a.m. (UTC+8), with an average of around 73 passengers per hour. During the remaining hours, the average traffic volume is approximately 314 passengers per hour.

Regarding the statistical analysis of inter-floor traffic flow, the floor with the highest outbound traffic is the 1st floor, with approximately 62% of passengers having the 1st floor as their destination. This is followed by the 17th floor, where about 13% of passengers have the 17th floor as their destination. Similarly, the 1st floor also registers the highest inbound traffic, with approximately 55% of passengers originating from the 1st floor. The 17th floor again ranks second, with about 13% of passengers starting their journey from the 17th floor. It can be concluded that the building's traffic flow is heavily concentrated on the 1st floor, with a significant disparity in traffic volume compared to the second most active floor.

Subsequently, we statistically analyzed the number of passengers appearing within each one-second interval, which represents the scale of the static problems we aim to solve. Our analysis revealed that in 37,637 instances, only one passenger appeared within a one-second window. Two passengers appeared simultaneously in 1,550 instances, and three passengers appeared concurrently in 42 instances. This distribution indicates that, when segmenting the dynamic problem into one-second intervals, the size of each static problem remains relatively small.

doi:10.6342/NTU202502477

Building Configuration	
Number of accessible floors	16
Lowest floor	B2F
Highest floor	17F
Elevator Configuration	
Number of elevators	4
Maximum capacity	9 persons
Door open time	3 seconds
Door close time	3 seconds
Move time / floor	4 seconds
Energy consumption / floor (up)	[10.5, 8.5, 6.5, 5.6, 5.2, 5.6, 6.8, 8.3, 10.6, 13] units
Energy consumption / floor (down)	[13, 10.6, 8.3, 6.8, 5.6, 5.2, 5.6, 6.5, 8.5, 10.5] units
Energy consumption / stop	2 units

Table 5.1: Building and Elevator System Specifications for Dynamic Simulation

Based on the analysis of the simulated dataset, the Hybrid control system is configured such that the ground floor utilizes destination calls, while all other floors maintain conventional calls. Furthermore, incorporating insights and recommendations from the company's practical operational experience, the passenger waiting time threshold θ was set at 30 seconds. For the genetic algorithm, the parameters are listed in Table 5.2.

doi:10.6342/NTU202502477

Parameter of the objective function	
θ	30
α	1
β	10
Parameter of the genetic algorithm	
Population size	50
Maximum iterations	100
Mutation rate	0.3
Selection method	Roulette Wheel Selection
Termination condition	Maximum Iterations Reached

Table 5.2: Algorithm Parameter Settings for the Dynamic Problem

5.3 Experiment Result

5.3.1 Performance Compared to Rule-Based Strategy

This subsection, we utilized the building's existing conventional control system, where the group control system only registers upward or downward requests from each floor and employed the building's original rule-based strategy as a benchmark against our genetic algorithm-based strategy.

The rule-based strategy consists of four rules, each applied when a call is generated to identify a suitable elevator. The system progresses to the next rule only if the current rule are not satisfied. These rules are prioritized as follows: 'Same Stop', which designates an elevator precisely at the same floor as the pending call; 'Free Ride', which identifies an

doi:10.6342/NTU202502477

elevator whose projected travel path includes the floor of the pending call; 'Idle', which selects an elevator currently in a non-operational state; and 'Nearest Away', which selects the elevator closest to the pending call.

The results are presented in Table 5.3. We found that the GA-based dispatch strategy outperforms the rule-based strategy by achieving a considerable reduction in total energy consumption and average effective waiting time. This suggests that the application of our proposed approach can lead to a more efficiency dispatching strategy for the All-Conventional system.

Strategy	Energy consumption	Effective Waiting Time
Rule-based	2713706.1	30.4
GA-based	2261410.8	27.1
Gap	−16.7%	−10.9%

Table 5.3: Performance for GA-based strategy compared to Rule-based Strategy

Consequently, by retaining the existing control system and merely substituting the dispatching algorithm with our proposed GA-based strategy, it is possible to overcome the rigidities inherent in rule-based approaches and capitalize on the advantages of re-assignment. This enables a simultaneous reduction in both energy consumption and effective waiting time, leading to a comprehensive improvement in the building's transportation efficiency.

doi:10.6342/NTU202502477

5.3.2 Performance with Different Control Systems

We investigate the benefits offered by the All-Destination and Hybrid control systems. In this phase, we conduct the dynamic experiments using the genetic algorithm. We utilize the All-Conventional system, also run with the genetic algorithm, as our comparative baseline to calculate the percentage difference for each metric, with the results tabulated in Table 5.4. Beyond overall performance, we further analyze and compare their respective performance during specific time periods: peak, off-peak, and regular traffic hours.

Overall, both the All-Destination and Hybrid systems show a slight increase in total energy consumption compared to the All-Conventional system (1.5% and 1.1% respectively). All-Destination exhibit higher effective waiting times overall (14.8%), indicating that the average impact on passengers experiencing longer waits. These results are consistent with the performance observed in static instances, particularly mirroring the findings for Scenario 0 as detailed in Table 5.4. Given that this dataset has a PI of 0.55, a PO of 0.62, and uses 4 elevators, which are parameters closely approximating those of Scenario 0, shows that both the All-Destination and Hybrid systems perform less favorably under these conditions.

The off-peak period highlights a different strength. The Hybrid system shows the most substantial improvement in effective waiting time, though it also results in a slight increase in energy consumption. In contrast, the All-Destination system achieves modest improvements in both energy consumption and effective waiting time. This suggests that during low-demand periods in this dataset, the advanced dispatching capabilities of these systems are highly effective. This efficacy might stem from the reduced need for high

doi:10.6342/NTU202502477

control system responsiveness during off-peak conditions, thereby providing ample scope for the benefits of more sophisticated control system designs to manifest.

Even though our overall performance in this particular case study did not fully meet ideal expectations, the observed advantages in certain periods nonetheless affirm the practical feasibility of our proposed solution. Furthermore, based on our experimental results across various traffic scenarios, as presented in Table 5.4, All-Destination and Hybrid call systems consistently achieve superior performance in most instances. Consequently, building managers could potentially observe the building's specific traffic patterns and prioritize key performance indicators in advance, subsequently selecting the most suitable control system to maximize operational benefits.

Period	Energy consumption			Effective Waiting Time		
	All-Conven.	All-Desti.	Hybrid	All-Conven.	All-Desti.	Hybrid
Overall	2261410.8	2294390.1 (1.5%)	2286282.6 (1.1%)	27.1	31.1 (14.8%)	26.8 (−0.9%)
Peak (7:00-8:59)	258312.2	256629.3 (−0.7%)	258504.1 (0.1%)	38.5	40.7 (5.6%)	37.3 (−3.1%)
Regular (9:00-21:59)	1635563.3	1673160.2 (2.3%)	1659051.1 (1.4%)	22.3	28.0 (25.9%)	22.9 (2.9%)
Off-Peak (22:00-6:59)	367535.3	364600.6 (−0.8%)	368727.4 (0.3%)	23.1	21.8 (−5.6%)	19.6 (−15.4%)

Table 5.4: Experimental Results for Different Control System in the Dynamic Instance

doi:10.6342/NTU202502477

Chapter 6

Implementation of Elevator Dispatching System

This chapter details the implementation framework that integrates our proposed elevator dispatching solution with the simulation environment. We will introduce the modular decomposition of the system architecture and the interaction methods between these modules, demonstrating how we establish a bridge between theoretical algorithms and practical application within a simulated environment.

6.1 System Architecture

As mentioned in Chapter 5, our collaboration with a company specializing in intelligent elevator solutions not only provided us with a realistic passenger traffic distribution but also led to the co-development of a comprehensive elevator dispatching experimental system. This system is functionally divided into two main parts based on their operational

doi:10.6342/NTU202502477

objectives: the Elevator System and the Dispatching System.

The Elevator System, primarily provided by the collaborating company, is built upon their professional expertise to model the real-world operation of an elevator group control system. This includes simulating elevator movement patterns, passenger behavior, and maintaining environmental status information. In the other hand, the Dispatching System is developed by us. It takes environmental information provided by the Elevator System as input and produces elevator operation command that the Elevator System understands, thereby controlling the elevators within the Elevator System to pick up and drop off passengers at various floors.

The system framework is shown as Figure 6.1. Our experiments leverage the interactive operation between the Elevator System and the Dispatching System. It serves the passenger demand dataset we provide, collects records such as passenger boarding times and elevator travel paths recorded by the Elevator System, and then calculates performance indicators like elevator energy consumption and passenger waiting time. This architecture, which separates the Elevator System and the Dispatching System, advantageously allows us to interchange the core logic within the Dispatching System. This flexibility enables us to evaluate different dispatching strategies using consistent assessment standards for measuring the merits of various dispatch solutions.

Within the Dispatching System, functionality is separated into three distinct modules: the Instance Generator, the Optimizer, and the Router. The Instance Generator is responsible for directly acquiring information from the Elevator System and preprocessing it into a format readily consumable by other modules. The Optimizer serves as the core module for finding dispatching strategies; it is designed to be interchangeable, allow-

doi:10.6342/NTU202502477

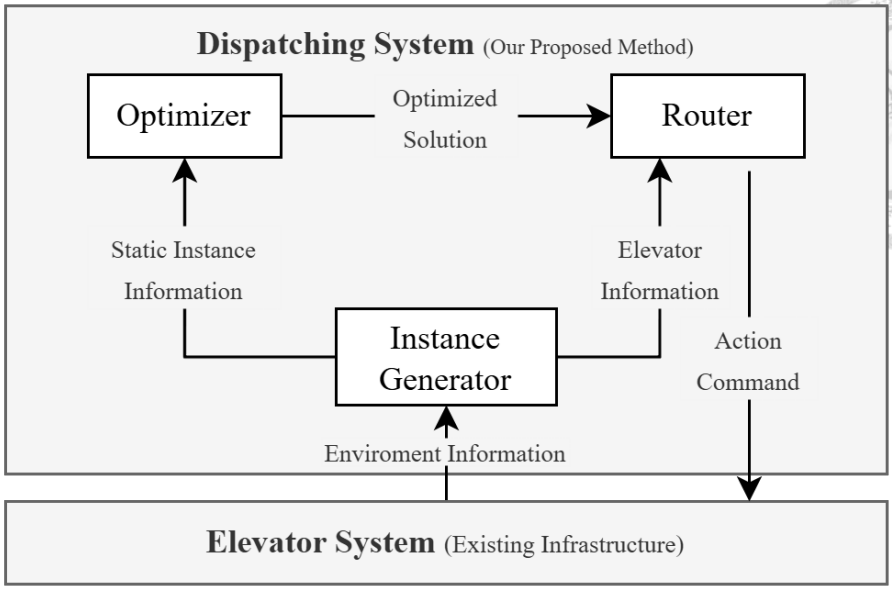


Figure 6.1: System Framework

ing for the use of either the exhaustive algorithm or the genetic algorithm to determine optimal solutions under a specific algorithmic approach. The Router is responsible for directly issuing elevator action commands to the Elevator System at appropriate timings, thereby influencing the elevator's subsequent travel paths.

6.2 Module Interaction

We focus on events within the Elevator System that alter the environment's state, such as call generation, elevator door opening/closing, passenger boarding, and elevator movement. When these events occur, the Instance Generator collects the latest environmental information to update the state. For instance, upon a call generation event, the Instance Generator records that signal in the passenger information and sets its status to 'unassigned'. When passengers board an elevator, since the system cannot precisely identify

doi:10.6342/NTU202502477

individual boarding passengers in practice, the Instance Generator updates the status of all passengers at the elevator's current floor and direction to 'aboard'.

Based on our defined decision criteria, the Instance Generator transmits the latest elevator information and passenger information, structured as static instance information, to the Optimizer at each decision point. Simultaneously, it changes the status of these passengers to 'assigned'. The Optimizer then invokes the algorithmic logic discussed in Chapter 4 to find a dispatching solution, which is subsequently transmitted to the Router.

Since the elevator system operates in a dynamic environment, the Instance Generator provides the Router with the latest elevator information when environmental changes occur. This allows the Router to determine if the current state is suitable for issuing commands. For instance, if all elevators are fully loaded, a command to pick up passengers should not be executed immediately; instead, the command to stop at the passenger's origin floor would be issued only after space becomes available in a specific elevator. Furthermore, command specificity varies with call type: for Destination calls, beyond directing an elevator to a specific floor for pickup, the Router must also inform the passenger, via the Elevator System, which elevator to board. Finally, given our re-dispatching mechanism for conventional calls that have not yet boarded, if the originally assigned elevator for a specific conventional call changes, the Router will also issue a command to the Elevator System to cancel that stop, thereby preventing the elevator from making unnecessary door operations.

By systematically processing information from the Elevator System, these three modules collaborate to issue timely action commands, facilitating dynamic decision-making.

doi:10.6342/NTU202502477

Chapter 7

Conclusion

This study presents a comprehensive framework for optimizing elevator dispatching strategies. We began by formulating the elevator dispatching problem as a combinatorial optimization challenge, amenable to solution through a sequence of static instances.

In Chapter 2, we reviewed relevant literature on elevator dispatching problems, encompassing static and dynamic approaches, assignment evaluation, and various optimization objectives. Chapter 3 meticulously describes the elevator dispatching problem, defining parameters, variables, and an objective function aimed at minimizing elevator operational energy costs and reducing the number of passengers experiencing prolonged waits.

Chapter 4 delves into the static problem formulation and solution methods. We decomposed the dynamic problem into a sequence of static snapshots and proposed two algorithms for solving them: an exhaustive algorithm (to find optimal solutions for small-scale instances) and a genetic algorithm (to find sufficiently good solutions within a reasonable timeframe for large-scale instances). Experimental results demonstrated that,

doi:10.6342/NTU202502477

compared to All-Conventional control systems, both All-Destination and Hybrid control systems exhibited improvements in objective value across most scenarios. Concurrently, the genetic algorithm offered substantial advantages in computational cost, making it a more viable approach for practical elevator systems.

Chapter 5 applied the proposed solution to dynamic scenarios, validating its effectiveness through simulations using simulated data based on real-world distributions. Experimental results found that the proposed genetic algorithm-based dispatching strategy outperformed the traditional rule-based method in terms of both energy efficiency and effective waiting time. The All-Destination and Hybrid control system exhibited a less favorable result under certain traffic conditions.

Finally, Chapter 6 provides a detailed overview of the elevator dispatching system's implementation framework, including the modular decomposition of the Elevator System and Dispatching System, and their interactions. This architectural flexibility allows researchers to evaluate the merits of different dispatching strategies using consistent assessment standards.

In conclusion, this research developed and validated an elevator dispatching algorithm suitable for hybrid control systems, demonstrating the feasible performance in balancing energy efficiency and passenger service quality. Future research can further explore the potential of more complex traffic patterns, real-time data integration, and machine learning techniques in elevator dispatching to achieve smarter and more adaptive systems.

doi:10.6342/NTU202502477

Bibliography

Bolat, B., O. Altun, P. Cortés. 2013. A particle swarm optimization algorithm for optimal car-call allocation in elevator group control systems. *Applied Soft Computing* **13**(5) 2633–2642.

Closs, G.D. 1970. The computer control of passenger traffic in large lift systems. *Ph. D. Dissertation, University Manchester* .

Cortés, P., J. Larrañeta, L. Onieva. 2004. Genetic algorithm for controllers in elevator groups: analysis and simulation during lunchpeak traffic. *Applied Soft Computing* **4**(2) 159–174.

Dai, D., J. Zhang, Z. Yin W. Xie, Y. Zhang. 2010. Elevator group-control policy with destination registration based on hybrid genetic algorithms. *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 12. IEEE, V12–535.

Fujino, A., T. Tobita, K. Segawa, K. Yoneda, A. Togawa. 1997. An elevator group control system with floor-attribute control method and system optimization using genetic algorithms. *IEEE Transactions on Industrial Electronics* **44**(4) 546–552.

doi:10.6342/NTU202502477

Hanif, M., N. Mohammad. 2022. Performance analysis of particle swarm optimization and genetic algorithm in energy-saving elevator group control system. *Proceedings of the International Conference on Big Data, IoT, and Machine Learning: BIM 2021*. Springer, 497–511.

Koehler, J., D. Ottiger. 2002. An ai-based approach to destination control in elevators. *AI Magazine* **23**(3) 59–59.

Ruokokoski, M., J. Sorsa, M.-L. Siikonen, H. Ehtamo. 2016. Assignment formulation for the elevator dispatching problem with destination control and its performance analysis. *European Journal of Operational Research* **252**(2) 397–406.

Siikonen, M.-L. 2024. Current and future trends in vertical transportation. *European Journal of Operational Research* **379**(2) 361–372.

Sorsa, J., H. Ehtamo, J.-M. Kuusinen, M. Ruokokoski, M.-L. Siikonen. 2018. Modeling uncertain passenger arrivals in the elevator dispatching problem with destination control. *Optimization Letters* **12** 171–185.

Sun, J., Q.-C. Zhao, P.B Luh. 2009. Optimization of group elevator scheduling with advance information. *IEEE Transactions on Automation Science and Engineering* **7**(2) 352–363.

Tartan, E.O., C. Ciftlikli. 2016. A genetic algorithm based elevator dispatching method for waiting time optimization. *IFAC-PapersOnLine* **49**(3) 424–429.

Tyni, T., J. Ylinen. 2006. Evolutionary bi-objective optimisation in the elevator car routing problem. *European Journal of Operational Research* **169**(3) 960–977.

doi:10.6342/NTU202502477

Wan, J., K. Lee, H. Shin. 2024. Traffic pattern-aware elevator dispatching via deep reinforcement learning. *Advanced Engineering Informatics* **61** 102497.

Zhang, J., J. Tang, Q. Zong, J. Li. 2010. Energy-saving scheduling strategy for elevator group control system based on ant colony optimization. *2010 IEEE Youth Conference on Information, Computing and Telecommunications*. IEEE, 37–40.

Zheng, J., H.C.T. Thomas, Y. HuaiBing. 2018. Traffic prediction for efficient elevator dispatching. *TENCON 2018-2018 IEEE Region 10 Conference*. IEEE, 2232–2236.

doi:10.6342/NTU202502477