Real-Time Passenger Flow Oriented Metro Operation Without Timetables

by

LI HE

A thesis submitted to

The University of Birmingham

for the degree of

DOCTOR OF PHILOSOPHY

Department of Electronic and Electrical Engineering

School of Engineering

The University of Birmingham, UK

November 2022

Abstract

With the increased population in large cities, the demand for urban passenger transport increases every year. Because of their high speed, high capacity and low energy consumption characteristics, urban metro systems are considered to play an essential role in urban transportation. Generally, unpredictable fluctuating passenger flow usually exists in urban metro operations, so traditional predetermined metro timetables cannot always meet the variation of real-time passenger flow, and the service quality of the metro system can be impacted profoundly. Nowadays, many researchers make efforts to propose suitable metro operation strategies to satisfy the constantly changing passenger demands and ensure the system's service quality.

In this situation, the author of this thesis deals with the dynamic metro scheduling problem and proposes a real-time metro operation method according to the variation of passenger flow. An innovative methodology has been proposed to model and solve the dynamic passenger flow oriented metro scheduling and real-time optimisation problem, and derived to propose passenger flow-based real-time operation strategies based on real-life operation without predetermined timetables.

First, a formal mathematical model, the Passenger Flow-Oriented Scheduling Model (POSM) has been proposed, based on nonlinear integer programming to minimise the service quality index (SQI) and also optimise the scheduling strategy with real-time passenger flow variation. An innovative algorithm GA_POSM, based on a genetic algorithm and integrated macroscopic metro and passenger flow simulator, has been designed to solve the scheduling and real-time optimisation problems formulated by POSM.

Then, the performance of GA_POSM has been evaluated based on the system data of Beijing Metro Line 19 with typical passenger flow distribution scenarios and Poisson distribution scenarios. The results show that, compared with traditional periodic timetables, the SQI can be significantly reduced by the scheduling method based on POSM; with real-time passenger flow variation, POSM can also optimise generated scheduling method flexibly.

Based on a field study in the London Underground Bakerloo Line Operation Department, the author also extended the proposed mathematical model to deal with different objectives in real-life operation, and integrated GA_POSM with a decision tree algorithm to improve its calculation speed for real-time application. Based on these extensions, a real-time passenger flow-oriented metro operation method without timetables, RPOM, has been proposed, and the system architecture and infrastructure requirements have been introduced. Compared to traditional timetable-based metro operation, the method can significantly improve the metro operation flexibility and the service quality according to further case studies.

Keywords: Metro scheduling, passenger flow-oriented, genetic algorithm, real-time operation.

Acknowledgements

I am very grateful to my supervisors, Prof. Lei Chen and Prof. Clive Roberts, for their patient, continuous and attentive guidance during my study and research in the University of Birmingham (UOB), for their inspiration and very generous support. I have greatly benefited from their personality, knowledge and enthusiasm in work and life. Without their support and encouragement, my study and research in the UK would not have been possible.

I also would like to give my thanks to Dr David Kirkwood for his kind programming supervision during my study at UOB. I also would like to give my thanks to Dr Ning Zhao for his kind research advice.

Finally, I would like to thank my wife, my parents and my parents-in-law, for their great love, which is my most cherished thing in life.

Contents

List of Figures

List of Tables

List of Abbreviations

AFC	Automated fare collection
BB	Branch and bound methods
CPU	Central processing unit
CSQI	Comprehensive service quality index
DOP	Dynamic optimisation problem
ELP	End left passengers
EUHT-5G	Enhanced ultra-high throughput – 5th generation
FME	Fourier-Motzkin elimination
GA	Genetic algorithm
GA_POSM	Genetic Algorithm for Passenger Flow-Oriented Scheduling
KKT	Karush-Kuhn-Tucker
NLP	Nonlinear programming
POSM	Passenger Flow-Oriented Scheduling Model
PTT	Passenger travelling time
PWT	Passenger waiting time
RAM	Random-access memory

RPOM	Real time passenger flow-oriented operation method
SQI	Service quality index
TLR	Train full load rate

Chapter 1. Introduction

1.1 Research Background and Problem Statement

1.1.1 Background

Nowadays, urban metro systems are considered to play an essential role in public transportation. Because of their high speed, high capacity and low energy consumption characteristics, urban metro systems are regarded as the most popular type of urban transportation, especially in large cities with many inhabitants, such as London, Tokyo, Beijing and Hong Kong (Ding and Xu 2017; Broere 2018; Lin et al. 2021; Lin et al. 2022). With the increased population, passenger demand keeps changing, impacting urban metro systems' service quality and energy consumption. In many situations, changes in passenger flow have a significant effect on a metro system's scheduling plan, and plenty of effort is devoted to optimising the timetable, saving energy and improving the service quality of metro systems.

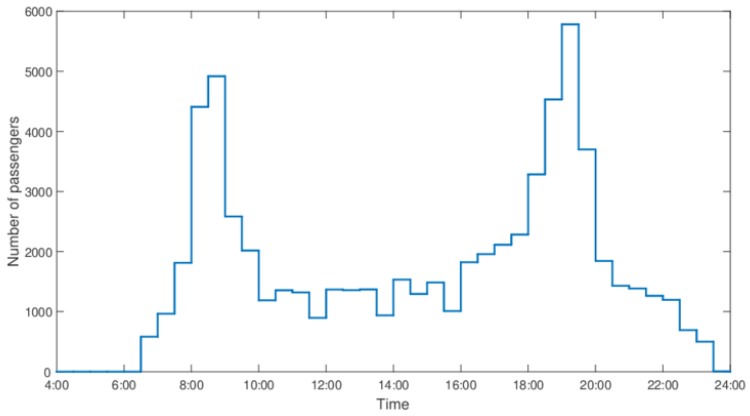


Figure 1-1 Time-variant passenger demands between two stations of the Yizhuang metro line through a single day(Wang et al. 2021)

On urban metro railways, the variations of passenger flow are dramatic. Depending on the time of day, season changes and even various urban events, passenger demand keeps changing. A typical variation of passenger flow, which shows the fluctuation of passengers in one day between two specific metro stations from the Yizhuang metro line, Beijing, is shown in Figure 1-1. The fluctuation shown above is ubiquitous in almost all urban metro systems, especially in developed cities with high population mobility; a well-known example is the London Underground which handles up to five million passenger journeys a day (Transport for London 2021).

Different with a mainline railway system that sells tickets according to the number of seats on the trains and limits the number of passengers boarding, and for which passengers need to plan their journey before boarding, most metro systems do not restrict the number of passengers by ticket, and most passengers usually board the first train arriving at the platform (Kang and Zhu 2016; Zhu et al. 2017). Another difference is that there are established timetables for mainline railway systems, which guide train operation and support passengers in arranging their journey at an early stage. By contrast, most urban metro systems are operated to achieve a periodic timetable with a particular departure interval, e.g. 5 minutes, rather than to obey a specific planned timetable. However, because of the dramatic changes in passenger flow and metro systems' specific operation processes, these periodic urban metro scheduling strategies cannot always satisfy the variation of passenger demands.

The variation of passenger flow can be catalogued into two main kinds: regular variation and irregular variation. Regular variation occurs when some predictable changes in external conditions happen, such as the rush hours, summer holidays, significant events, etc. These regular variations can be forecast in the early stage based on historical

experience, and metro operators have enough time to design specialised timetables for them.

By comparison with regular variation, irregular variations happen in real time, and they are unpredictable in the early stage; for example, because of a sudden change of weather, many tourists choose to travel by the metro instead of walking to a scenery spot. Generally, these irregular variations will invalidate well-designed running strategies, which can cause oversaturation of passengers and deterioration of service quality. As more passengers are stranded in the stations, the dispatching of trains will also be impacted.

This problem of irregular variations in real-time passenger flow associated with a metro system can be appreciated if it is assumed that in a specific time horizon, all the trains in the system are running based on a pre-defined timetable according to the statistical passenger flow data as shown in Figure 1-1. In this scenario, a small variation of real-time passenger flow may cause the pre-defined running strategy to fail. Any big variation of real-time passenger flow can result in poor service quality of the metro system.

1.1.2 Problem Statement

A typical example of a metro train scheduling problem is shown in Figure 1-2. Dispatchers need to consider the departure interval between each train according to different system infrastructure, operation requirements and safety constraints. Normally, in a specific time horizon, dispatchers try to fix the interval for each train. In Figure 1-2, the interval between two trains in peak hours has been set to δ; in off-peak hours the interval is 2δ. This strategy tries to avoid passengers being stranded in stations in peak hours, at the same time reducing the number of trains in off-peak hours to save energy.

Considering all the passengers in the stations in a specific time horizon, the scheduling problem also needs to decide the dwelling time for trains in different stations. Generally, dispatchers will also schedule different dwelling times for trains in different time horizons. In Figure 1-2, trains in peak hours are stopping based on a dwelling time 2δ, to carry more passengers. Conversely, in off-peak hours, the dwelling time is set to δ. Both departure interval and dwelling time are periodical.

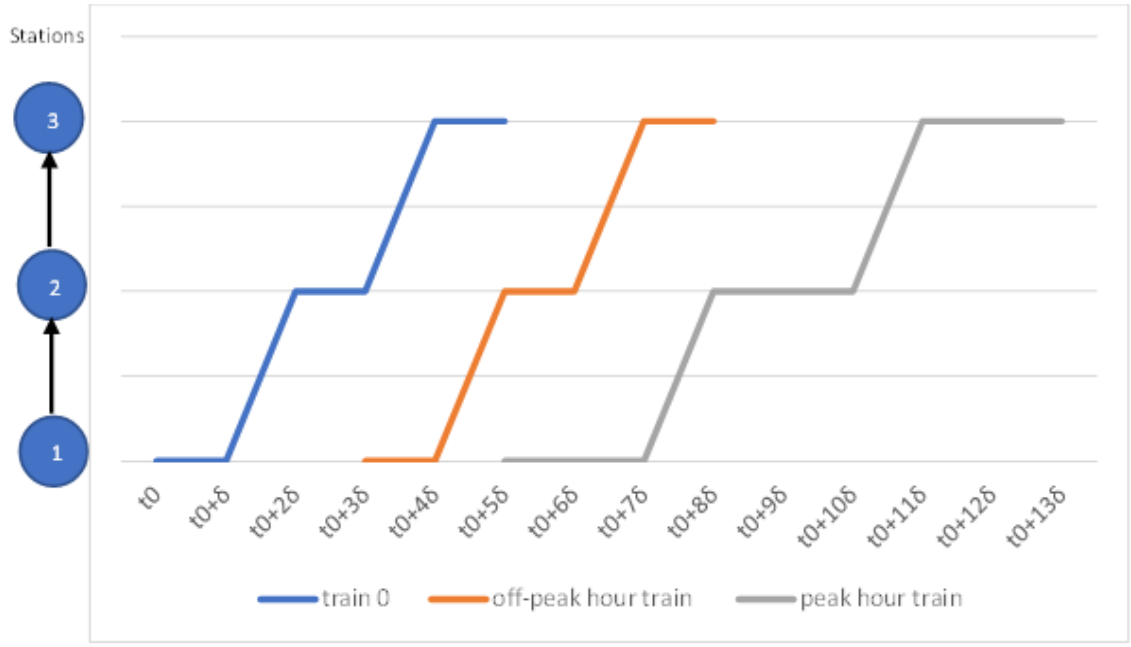


Figure 1-2 Example of train scheduling

The conventional train scheduling approaches mentioned above depend heavily on the dispatcher's experience; in real operation the variation of passenger flow is unstable, and cannot be simply divided into peak and off-peak hours. Sometimes, even in one time horizon, passenger flow changes significantly. As a result, in the last decade, dynamic scheduling strategy research has become very popular; the scheduling process can be represented by optimisation based on a special decision tree with optional intervals and dwelling times for each train according to statistical passenger flow data. As shown in

Figure 1-3, the optional intervals and dwell times constitute branches of the whole tree. Each branch of the tree denotes an interval and a dwelling time for a train in the journey, and all the trains in the same time horizon will have a unique decision tree. Based on the decision tree, the space–time representation of each train, which aims to sketch the train diagram and integrate the physical transportation network with vehicle time-dependent movements for each trains, can be proposed.

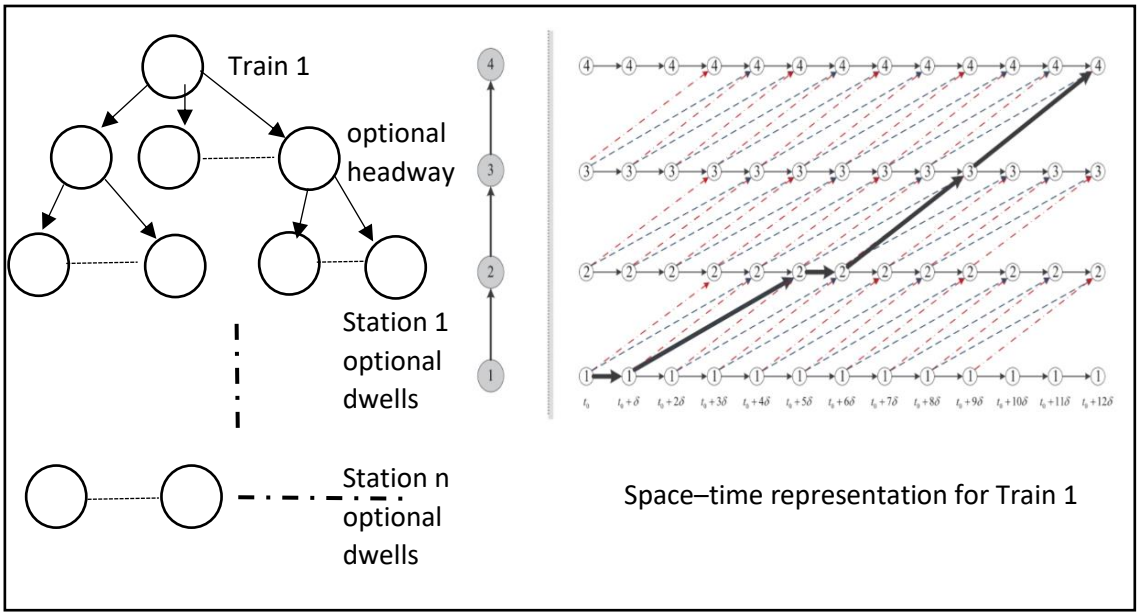


Figure 1-3 Space–time representation of train scheduling process with a special decision tree

But in real operation, the flexibility of both a conventional periodical train scheduling timetable and dynamic scheduling strategy is not enough, because no matter how accurately passenger flow statistical data are recorded, there will always be irregular variations in the real-time operation, so the prepared timetable cannot consistently satisfy real-time passenger demands, sometimes these irregular variations seriously impact the service quality. As a result, to guarantee the quality of scheduling strategies they not only need to be based on dynamic passenger flow, but also must be able to be optimised in real

time according to real-time passenger flow variation. Because in real time some trains have already been operated based on the old timetable, which means some operation strategies have already been determined and implemented, this process will change the problem to a dynamic optimisation problem with an unfixed number of variables, and the modifiable strategies of each train will be different. The associated service quality function may be expressed as passenger waiting time, the average number of passengers stranded in a station, or the average usage of trains or the number of trains in one time horizon, as well as other particular definitions of passenger satisfaction and train efficiency.

Because of the unpredictability of real-time irregular variations in passenger demand, it is essential to optimise the scheduling strategies in a short time, so that the quality of services will not be impacted by variations in passenger flow. Therefore, the optimising method must be fast and flexible enough; any such adjustments must be taken in time, and looking ahead as far as possible, thus making sure to meet the demand of passengers.

1.2 Objectives and Contributions of the Thesis

The author focuses on solving the real-time train scheduling and optimisation problem based on dynamic passenger flow with irregular variation in real-time passenger demands. With regard to the proposed real-time passenger flow-oriented operation problems, the overall objective is to establish a systematic methodology for real-time operation strategy for an urban metro line to replace fixed timetables, including a scheduling strategy based on statistical passenger flow data and a real-time optimisation strategy based on irregular

variations of real-time passenger flow, which is applied to improve the service quality and efficiency of the metro system. This mainly includes the following objectives:

a) To formulate train scheduling optimisation problems which aim to minimise the Service Quality Index (SQI) defined in this research based on dynamic passenger flow demands for a metro line.

b) To extend a train scheduling optimisation mathematical model to solve real-time scheduling and optimisation problems.

c) To develop and integrate rapid algorithms with simulation programs to solve the proposed problems.

d) To extend the mathematical model to propose optimised scheduling strategies to different objectives based on passenger demand and operational requirements based on a real-life field study.

e) To extend the proposed methodology and present an integrated system architecture of a real-time passenger flow-oriented operation method without timetable for an urban metro line.

f) To evaluate the proposed model and methodology using case studies.

As for the objectives above, this thesis mainly contributes as follows:

(1) A formulated model ‘passenger flow-oriented scheduling model (POSM)’ based on nonlinear integer programming for train scheduling optimisation is proposed, dealing with train scheduling based on dynamic passenger flow and real-time scheduling optimisation.

(2) An extension of the proposed mathematical model and for solving real-time passenger flow-oriented metro operation without a timetable problem based on requirements from real-life studies.

(3) An algorithm, GA_POSM, based on a genetic algorithm is introduced for solving the POSM problems proposed in this thesis. The algorithm GA_POSM integrates a generic genetic algorithm and roulette algorithm with modification rules based

on constraint conditions and real-time operation, to be an efficient algorithm for solving POSM problems.

(4) A macroscopic metro simulation program integrated with GA_POSM based on space–time representation has been designed and programmed in Java language, to provide support for solving the model and evaluating results.

(5) An extension of the proposed algorithm and simulation program based on a decision tree algorithm for solving the extended no timetable real-time scheduling problem.

(6) An integrated system architecture of the passenger flow-oriented real-time metro operation method without a timetable. The benefits and shortcomings of the system are discussed.

1.3 Outline of the Thesis

This thesis is structured into six main chapters, as shown in Figure 1-4.

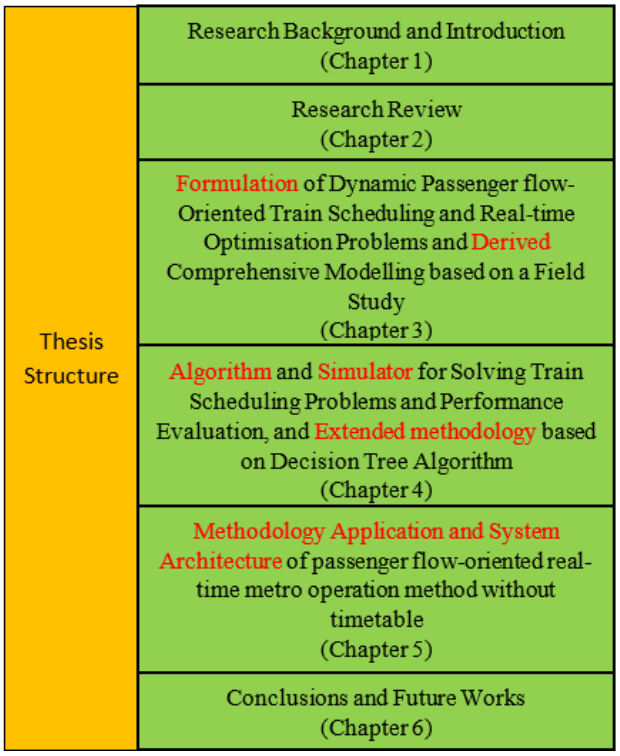


Figure 1-4 Thesis Structure

The overall research motivations and background are introduced in Chapter 1; in particular, the train scheduling problems and real-time optimisation problems to be dealt with in this thesis are presented.

Chapter 2 gives a general introduction and review of related research which has been done all over the world. The research review is divided into three aspects, traditional urban metro railway scheduling, rescheduling and real-time scheduling.

Chapter 3 formulates the dynamic passenger flow demand-oriented scheduling problems and real-time scheduling optimisation problems for urban metro systems. A passenger flow-oriented scheduling model (POSM) based on nonlinear integer programming is proposed. And extends the proposed mathematical modelling with different objectives based on the requirements from real-life field study in London Underground.

Chapter 4 introduces an innovative algorithm GA_POSM based on a generic genetic algorithm and integrated with a metro system simulation program. The algorithm GA_POSM is evaluated using data from Beijing Metro Line 19, and the results are compared with the periodic departure interval running strategy in real life. And the methodology of are extended with a decision tree algorithm to improve its speed to solve the passenger flow-oriented real-time metro operation problem in complex systems.

Chapter 5 describe the process of passenger flow-oriented real-time metro operation method without timetable, RPOM with a large case study. And an integrated system architecture of the urban metro which can be applied for implementation of the proposed passenger flow-oriented real-time metro operation methodology are proposed in this thesis.

Conclusions and future work are presented in Chapter 6.

Chapter 2. Research Review of Rail Traffic Scheduling and Optimisation

Railway scheduling is a complex process of designing and optimising timetables for trains according to different system objectives, In general, metro traffic scheduling and optimisation refers to three types of objectives: (1) increasing timetable robustness to avoid operating incident (Lusby et al. 2018; Ochiai et al. 2019; Restel and Haladyn 2022), such as adding more margin time into a timetable to improve the stability of the timetable (Corman et al. 2012; Corman and Quaglietta 2015; Higgins et al. 1996); (2) increasing operation efficiency to reduce fuel consumption costs (Ning et al. 2015; Zhao et al. 2017; Yang et al. 2022), such as adjusting the running time and acceleration curve (or the number of trains) (Yang et al. 2015); (3) improving the quality of services to satisfy passenger demand (Yang et al. 2019; Pan et al. 2022; He et al. 2022), such as modifying the headway and dwelling time to decrease passengers' waiting and journey time(Zhou and Zhong 2005). With the increased urban population, complicating the fundamental process of metro system operations, more and more attention has been paid to efficient urban metro traffic scheduling and optimisation, especially passenger-oriented methods. Because of the complexity of urban metro systems, systematic methods need to be proposed for timetable scheduling and optimisation. These approaches mainly focus on two aspects: timetable scheduling and rescheduling; some recent studies have also proposed the real-time concept. The objective of this chapter is to present the state of art of the research on urban metro traffic scheduling and optimisation across the world.

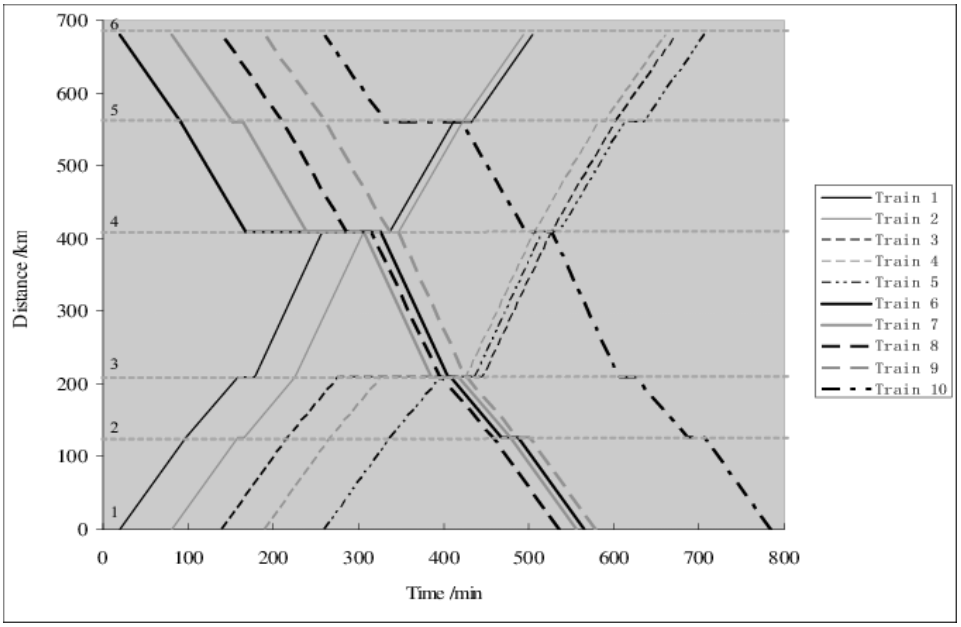


Figure 2-1 Train graph(Yang et al. 2009)

Scheduling is a fundamental process of constructing metro timetables (or in normal operation, it is engaged to propose periodic operation strategies) for trains in the system within a specific time horizon; this process needs to schedule departure intervals and dwelling times (sometimes also the running times) of trains in different stations based on infrastructure and safety requirements. After scheduling, a typical train graph will be generated as shown in Figure 2-1; this graph presents information regarding the running of trains based on a time–distance format. Based on the two processes above, the time of different trains passing and stopping at a specific point on the metro line can be presented, including the arrival and departure time in different stations, allowing systematic timetables to be generated. Also, to ensure the system is safe and can deal with unexpected system variations, margin or recovery times need to be applied to make the timetable robust. However, in real-time operation, with various impacts such as weather, system disturbance and disruption being inevitable, and the normal scheduling process not being able to handle the operation problems in some specific parts of the system, such as

junction areas and terminuses, a targeted scheduling or rescheduling process can be proposed to modified these specific operational problems. The aim of this chapter is to review various approaches to urban metro traffic scheduling and optimisation based on scheduling and rescheduling processes.

2.1 Scheduling

2.1.1 Periodic Scheduling

Generally, urban metro scheduling proposes running strategies for trains to carry passengers across the metro network; For periodic scheduling, trains are scheduled at a specific frequency, and timetables keep repeating in a time horizon. This periodic scheduling method is easy to formulate and optimise based on regular passenger flow variations and periodic events, and because of its periodicity, the system's stability and robustness is increased. Thus, most metro systems in real life apply the periodic method as the operation strategy.

Carey and Crawford (2007) presented a periodic scheduling method that schedules trains at a single train station to avoid conflicts between a large number of trains moving at different speeds between multi-platform stations on conflicting lines, and extended this method to handle a series of complex stations linked by multiple one-way lines in each direction, traversed by trains of differing types and speeds. The calculation time and the solved conflict are two important performance characteristics in this research. They modelled the problem as a nonlinear programming problem, applying heuristic algorithms to schedule the trains. They tested the algorithms based on some draft

timetables constructed with a lot of conflicts, showing that the approach has good performance based on the computation time and the resolved conflicts.

Cordone and Redaelli (2011) studied the reciprocal influence between the quality of the timetable and the amount of transport demand captured by the railway. They proposed the advantages of regular timetables for both passengers and service management, modelling the problem as a mixed-integer nonlinear model with non-convex continuous relaxation and solving it based on a piecewise-linear overestimate of the objective function and a heuristic algorithm. The algorithm was tested by both random instances and a real-world regional network located in north-western Italy. For this research, high demands of passengers is the most important performance characteristics defined.

Li and Lo (2014) considered a scheduling and speed control method to improve a metro rail system's energy efficiency based on a periodical timetable. In this research, they tried to synchronise the accelerating and braking action of trains to maximise the utilisation of regenerative energy based on a periodic timetable constraint. The problem was presented by an integrated energy-efficient operation model which optimises the timetable and trains' speed profile, and a genetic algorithm was applied to solve the model. The system's energy efficiency was considered as the most important characteristics and need to be improved. They evaluated the computational results on a real-world metro system from the Beijing Metro Yizhuang Line in China.

The examples above show that periodic scheduling makes the timetable easy to formulate and robust, especially for some complex parts of a metro system; with a periodic timetable, incidents and disruptions hardly occur. But periodic scheduling also makes the timetable inflexible, which means it is hard to meet variations in demand.

2.1.2 Dynamic Scheduling

Besides periodic scheduling, with the development of passenger counting systems and automatic fare collection systems, more passenger flow data can be analysed (Though in some countries, the passenger counting system may face some legal challenges because of the implemented camera, privacy concerns, discrimination, lack of regulation, system bias and data security, this system can highly improve the accuracy of the recorded passenger flow OD data. Also, in some big cities, the passenger number counting system has already been implemented, such as the Beijing Metro and Shanghai Metro); in recent years, instead of considering periodic timetables, more research has focused on a dynamic scheduling approach based on different demands. This scheduling method converts the traditional scheduling problem to a dynamic optimisation problem with flexible environmental parameters which change over time.

Niu et al. focused on optimising a passenger train timetable in a heavily congested urban rail corridor under time-dependent demand. Based on the origin-to-destination trip records from the automatic fare collection system, the dynamic demands of passenger flow could be collected. Based on the objectives to decrease passenger loading time and waiting time, a nonlinear optimisation model was developed, using a genetic algorithm to find the optimal timetables. The effectiveness of the model and algorithm were evaluated by the real data from the No. 8 subway line in Guangzhou, China. With time-varying origin-to-destination passenger demand matrices, they also developed a train scheduling method to minimise the total passenger waiting time based on skip-stop running patterns of trains, transforming the passenger flow data from the automatic fare collection system into origin–destination (OD) pairs. A set of quadratic and quasi-quadratic objective functions based on quadratic integer programming with linear

constraints was proposed, which synchronise effective passenger loading time windows and train arrival and departure times at each station. Finally, the problem was formulated as a nonlinear mixed-integer problem, and solved by general purpose high-level optimisation solvers. The train dwelling time in each station was considered to be an important variable (Niu and Zhou 2013; Niu et al. 2015).

Halim et al. (2016) proposed a passenger-perspective metro timetable optimisation approach based on simulation models and incomplete passenger flow data which aimed to minimise passenger waiting times and avoid high passenger loads inside the trains. They proposed a linear programming model, solved by an iterative algorithm integrated with a simulation program. The result was simulated and evaluated based on real-world data from the Cairo metro system.

Similar with periodic scheduling, besides passenger demand, some dynamic scheduling also concentrates on energy efficiency. Yin et al. (2016) considered a bi-directional train scheduling approach with dynamic passenger demand, intended to minimise passenger waiting time and energy consumption. They proposed two linear optimisation models, one an integer programming model with the objective to minimise passenger waiting time and energy consumption, the other a mixed-integer programming model to consider the utilisation of regenerative braking energy. They developed a heuristic algorithm based on a Lagrangian relaxation method to solve the models, using a small-scale case study and a real-world instance based on Beijing metro data to evaluate the approach.

Mo et al. (2019) proposed a flexible metro train scheduling approach based on inhomogeneous passenger demand, which aimed to minimise energy cost and passenger waiting time. They proposed a nonlinear integer programming model and integrated a

modified tabu search algorithm with prior enumeration methods to find the approximately optimal solution, applying a case study based on the Beijing Metro Yizhuang Line to evaluate the result.

The examples above show that, as an important time-varying objective function control parameter, dynamic passenger demand is a focus of dynamic scheduling research. Thus, for this approach the synchronisation and interaction between the passenger flow and the metro system become much more significant.

Generally, there are three main techniques for a scheduling modelling approach: linear programming, nonlinear programming and graph theory. Caprara et al. (2002) proposed a graph theoretic formulation modelling approach for train departures and arrivals at a certain station and extracted a linear programming model to determine a periodic timetable for a set of trains that does not violate track and satisfies operational constraints based on a single, one-way track linking two major stations. The result was solved by integrating Lagrange relaxation with a heuristic algorithm and evaluated based on a real-world instance from Ferrovie dello Stato SpA in Italy.

Brännlund et al. (1998) proposed a novel optimisation approach for the timetabling problem of different types of services to obtain a profit-maximising timetable that does not violate track capacity constraints. They modelled the problem as a very large integer programming problem, and a Lagrange relaxation solution approach was applied for track capacity constraints, with their testing work based on an example suggested by the Swedish National Railway Administration which is a single-track railway with 18 passenger trains, 8 freight trains and 17 stations. The result shows the good performance of the approach in terms of computation times and optimality of the obtained timetable.

Sun et al. (2015) proposed a method for optimising train scheduling for metro lines with a train dwell time model according to passenger demand. They formulated this problem based on a nonlinear objective function with some linear equality constraints about headway, passenger and train dwell time. The aim of the optimisation problem was to minimise passenger waiting time and train operation cost. Lagrangian duality theory was adopted to solve the problem with high dimensionality. The result was tested using data from Line 8 of the Beijing metro network.

Wong et al. (2008) presented a mixed-integer programming optimisation model for the schedule synchronisation problem for non-periodic timetables that minimises the interchange waiting times of all passengers and studied the passenger interchanges between different lines in urban transit railways. The optimisation was based on adjusting trains' run times and station dwell times during trips, dispatch times, turnaround times and headway, using an optimisation-based heuristic for the model. The algorithm testing was undertaken on the mass transit railway system in Hong Kong.

Based on graph theory, D'Ariano et al. (2007) adopted a detailed alternative graph model for the train dispatching problem, which aimed to detect the conflicts between different trains, ascertain whether a safe distance headway between trains is respected and provide acceptable speed profiles for each train over an intended time horizon. The final conflict-free scheduling timetable could be generated after a finite number of iterations. A computational study based on an hourly cyclical timetable of the Schiphol railway network was carried out.

2.1.3 Solution Approach

Three typical solution methodologies are applied to scheduling problems: commercial software optimisation solvers, heuristic algorithms and simulation approaches.

Li et al. (2013) proposed a green train scheduling model and fuzzy multi-objective optimisation algorithm; this multi-objective scheduling model could minimise the energy, carbon emission cost and total passenger-time at the same time. They applied a fuzzy multi-objective optimisation algorithm to solve the model based on the nonlinear optimisation software LINGO and obtain a non-dominated timetable. The approach was illustrated and evaluated by two numerical examples based on the Wuhan–Guangzhou high-speed railway.

Normally, heuristic algorithms applied to scheduling problems include evolutionary algorithms (Chen et al. 2015), genetic algorithms (Takagi et al. 2006), greedy algorithms (Krasemann 2010), tabu search (Corman et al. 2010) and simulated annealing (Tomii et al. 2005). As shown by some examples in sections above, due to their good computation time performance, heuristic algorithms have been widely used for dynamic scheduling research in the last decade.

Sun et al. (2019) developed a bi-objective timetable optimisation model to minimise the total passenger waiting time and pure energy consumption based on nonlinear programming. The objective function was solved by a modified genetic algorithm. Numerical examples based on the real-world data from the Beijing Metro Yizhuang Line were used. The results indicate that the approach can improve passenger service and reduce energy consumption efficiently.

Simulation is also an excellent tool for solving railway traffic scheduling problems. Simulation methods have the capability of showing the dynamic characteristics of the

metro system and the performance of passenger flow. Simulation analysis methods are suitable for a system with strong interactions and they also have good extensibility (Halim et al. 2016); they are also a useful tool for validating scheduling strategies.

Xu et al. (2019) proposed a simulation-based optimisation approach for simultaneous locomotive assignment and train scheduling on a single-track railway line. Two mixed-integer programming models were formulated to integrate the final optimisation model. They designed an efficient optimisation approach on the basis of assignment rules and train movement simulation to solve this problem, testing the result on a hypothetical railway system.

Mu and Dessouky (2013) proposed a switchable dispatching policy for a double-track segment, which enables a fast train to pass a slow train by using the track travelled on by trains in the opposite direction if the track is empty. This method is hard to test based on a real-life railway system, thus they tested the results based on an Arena simulation model.

2.2 Rescheduling

Despite there being different scheduling strategies for operational optimisation, in real life, with various impacts such as weather, operational incidents are inevitable (Wang and Zhang 2019) and some incidents will lead to train delays or passengers being stranded, which will decrease the service quality of the system and even cause a system-level failure. Thus, rescheduling processes which mainly focus on modifying train operation when incidents happen are essential. Generally, rescheduling has two objectives, adjusting train operation to deal with unpredictable variations before a problem occur and engaging to modify the timetable back to normal after it has been distributed.

2.2.1 Traditional Rescheduling Strategies

In general, traditional rescheduling strategies can be divided into five categories (Fu et al. 2003): holding, zone rescheduling, short turning, deadheading and skipping.

Holding is applied when some unpredictable variations occur in the pre-defined headways. This method will modify the headway between trains by holding the early-departing train. Eberlein et al. (2001) formulated the holding problem as a deterministic quadratic program in a rolling horizon scheme, in order to minimise passenger waiting time in the situation of large passenger flow. They solved the problem using a heuristic algorithm and tested the result based on the headway data collected from the automated system of MBTA's Green Line in Boston.

A short turning strategy, mainly used in bus operation, divides the rail trips into short-turn and full-length trips; short-turn trips serve a specific zone, while full-length trips run the whole line. This strategy can always be applied when there is high passenger demand within a zone in the line or when there is a big system-level error which leads to serious disruption of the timetable. Delle Site and Filippi (1998) designed an optimisation framework which includes short-turn strategies for intermediate-level planning of bus operations.

The zone rescheduling method is a complex version of the short turning method which divides the whole line into different zones, and undeparted trains are also divided into different groups. Each train will only stop at all the stations within a single zone, then departs without stopping to the terminal station. This method is useful for both scheduling and rescheduling processes, especially when a large passenger flow leads to piling up in a few specific stations. Ghoneim and Wirasinghe (1986) proposed an optimum zone

structure for urban rail lines to deal with the large passenger flow in peak periods. They designed a dynamic programming model to determine the number of zones, the stations that belong to each zone, the headway between trains in the same zone and the fleet size. The applicability of this model was demonstrated through a numerical example using field data.

The deadheading strategy makes some trains skip and run empty through a few stations at the beginning of their trips, which can reduce the headway between all the trains in the system. This method is suitable for dealing with a serious delay situation or when there is high passenger demand in the last stations. Eberlein et al. (1998) proposed a deadheading problem to determine which vehicles to deadhead and how may stations to skip, in order to minimise the total passenger cost in the system. The stop-skipping strategy is an enhanced version of the deadheading strategy, which allows trains that are delayed to skip a number of low-demand stations and increase the running speed. In real life, because of the impact to passengers getting off, stop-skipping is seldom used.

From the Emergency Plan file of the Hefei Urban Rail Transit in real life, most strategies above have been applied (Hefei Urban Rail Transit Co. Ltd. 2021).

2.2.2 Disturbance Rescheduling

Based on the seriousness of the incident, rescheduling research can also be divided into two directions, disturbance management and disruption management. Disturbance management deals with a series of small-scale incidents, such as a short delay. In this situation, the dispatch strategy needs to be modified to reduce the influence on the timetable in the future.

Hou et al. (2019) designed an energy-saving metro train timetable rescheduling model based on ATO profiles and dynamic passenger flow. A mixed-integer programming model was proposed to optimise the total train delay, the number of stranded passengers and the energy consumption of trains. They adopted the commercial optimisation software CPLEX to solve the proposed model; the result could be generated in a short time. Three numerical experiments based on real-world operational data were carried out to verify the effectiveness.

To avoid and modify disturbance, Meng and Zhou (2014) proposed an innovative integer programming model for the problem of train dispatching on an N-track network by means of simultaneously rerouting and rescheduling trains. They decomposed the complex rerouting and rescheduling problem into a sequence of single-train optimisation subproblems and applied a standard label-correcting algorithm to find the time-dependent least-cost time–space path. The results were evaluated based on a set of numerical experiments which indicated the benefits of simultaneous rerouting and rescheduling.

Yang et al. (2014) designed a credibility-based rescheduling model according to a fuzzy optimisation framework in order to reschedule trains in a double-track railway network when capacity was reduced by a low-probability incident. The problem was formulated as a credibilistic two-stage fuzzy 0-1 integer optimisation model to find a reliable operational plan for emergency response guidance. They solved the problem by using the GAMS optimisation software; with numerical experiments, the effectiveness and efficiency of the method was evaluated.

2.2.3 Disruption Rescheduling

Disruption rescheduling research mainly focuses on events which cause a much longer time delay than does disturbance. In this situation, not only modification of the dispatch strategy is needed; other measures such as cancellation of trains, the addition of trains or skipping stops also need to be considered.

Gao et al. (2016) considered a disruption situation in which the planned timetable cannot be operated and a large number of passengers are left stranded in the stations. Considering overcrowding and time-dependent passenger flow, they proposed an optimisation model to reschedule a metro line, in which some stations may be skipped in the recovery period. An iterative algorithm was proposed to solve the model and the result was evaluated based on numerical experiments with data from the Beijing metro.

Louwerse and Huisman (2014) focused on adjusting the timetable of a passenger railway operator in case of major disruptions. Given a disrupted infrastructure situation and a forecast of the characteristic of the disruption, they determined a disposition timetable, specified which trains will still be operated during the disruption and determined the timetable of these trains. An integer programming formulation model was proposed with the main objective to maximise the service offered to the passengers. The result was evaluated based on the Netherlands Railways.

2.2.4 Passenger demand oriented rescheduling

Also, there are some studies that considered passenger demand in rescheduling, most of this research considered about decreasing the impact which caused by the system's disruptions or disturbance on passenger flow.

Wang et al. (2020) propose the collaborative optimisation for train rescheduling combined with passenger flow demand after a metro disruption. They established a train rescheduling model and a passenger flow model, tested the impacts on passenger's travel time and numbers of stranded passengers from rescheduling strategies including short-turnings, fully cancellation and partial cancellation after a metro disruption. They found the short-turning strategy can effectively mitigate the stranded passengers' numbers in disruption, but also increases trains' delays and passenger travel time. Cancelling strategy can reduce passenger travel time but may cause more serious trains' delays. Thus, reasonable number of short-turning and cancellations should be in a combination after a metro disruption.

Zhu and Goverde (2020) integrated timetable rescheduling and passenger reassignment in railway disruptions. They proposed a novel passenger-oriented timetable rescheduling model based on a Mixed Integer Linear Programming to minimise generalized travel times. The model applies re-timing, re-ordering, cancelling, flexible stopping and flexible short-tuning strategies after disruptions, and solved by an Adapted Fix-and-Optimise algorithm. The proposed methodology has been evaluated based on the data from a part of the Dutch railways, and the generalized travel times can be significantly shorten in disruption.

Hong et al. (2021) focused on a train rescheduling problem with large disruptions, where passenger reassignment have to be considered. They proposed a novel mixed-integer linear programming formulation which consider train retiming, reordering, rerouting and reservicing in a large disruption and delt with constraints based on limited seat capacity and track capacity. The objective function is optimised by a weighted-sum method to maximise the number of disrupted passengers reaching their destination stations and

minimise the total train delay for all non-cancelled trains. The proposed method is evaluated based on the data from Beijing-Shanghai high-speed railway line. With the optimal reassignment plan, the railway's real-time efficiency can be achieved in disruptions.

2.2.5 Real-Time Operation Research

With the development of AFC (automated fare collection) ticket systems, passenger counting systems and the proposal of digital twins, real-time metro operation research has become more and more popular in recent decades. In general, real-time operation research includes dynamic scheduling research with statistical real-time passenger flow data and real-time rescheduling research based on a specific situation in a metro system.

Wang (2013) proposed a real-time scheduling problem for urban rail transit systems with the aim of minimising the total passenger travel time. A real-time scheduling model was formulated based on the operation of trains and passenger demand characteristics; with the minimum headway constraints, this problem was transmitted into a nonlinear non-convex programming problem. They solved this problem by sequential quadratic programming. A case study based on the real-time statistical passenger flow data of the Beijing Yizhuang Line was used to demonstrate the performance of the proposed approach.

Chen (2012) modelled and solved real-time train rescheduling problems in railway bottleneck sections to increase the throughput of the sections. Based on the rescheduling problems, a mixed-integer programming model was presented, and solved by an innovation improved algorithm, DE_JRM. The model and algorithm were validated with a case study using Monte Carlo methodology, which demonstrated that the proposed

algorithm can reduce the weighted average delay and satisfy the requirements of real-time traffic control applications.

Zhan et al. (2015) focused on a real-time rescheduling problem of railway traffic on a high-speed railway line in the case of a complete blockage of the railway infrastructure. They considered the situation that trains are blocked by the disruption and wait inside the stations. A mixed-integer programming model was formulated to minimise the total weighted train delay and the number of cancelled trains, while adhering to headway and station capacity constraints. They proposed a two-stage optimisation approach to improve the computation efficiency and tested the model based on the data of the Beijing–Shanghai high-speed railway line.

Flamini and Pacciarelli (2007) proposed a scheduling problem arising in the real-time management of a metro rail terminus which routes incoming trains through the station and schedules their departures. The problem was modelled as a bicriterion job shop scheduling problem with additional constraints and solved by heuristic algorithm. They tested the result based on the rail traffic data at an Italian metro rail terminus.

From the above, there are three main shortages of current real-time research:
1. For scheduling research, instead of proposing a continuous real-time operation methodology, most real-time scheduling research just focus on modifying scheduling method with some specific objectives according to the recorded data of the passenger flow.

2. Most rescheduling research focuses on proposing solutions for some specific situations, such as train delays, crossing junctions. Instead of keeping proposing real-time strategies, this research tries to recover disrupted timetables to the original.

3. Most research has fixed objectives without flexibility.

2.3 Conclusions

In this chapter, the main research on rail traffic scheduling and optimisation, including scheduling research, rescheduling research and real-time research, has been reviewed. In scheduling research, two main aspects, periodical scheduling and dynamic scheduling, have been introduced. Different modelling approaches with a variety of objectives and different solution approaches have also been reviewed. Then the traditional rescheduling method and rescheduling approaches based on different incidents have been reviewed, some passenger flow oriented rescheduling research has been analysed. Finally, some real-time railway research for both scheduling and rescheduling has also been introduced.

As is apparent from the above review, most metro systems in real life still apply a periodical timetable to meet the demand of regular passenger flow variations and periodic events. Most of the existing scheduling research, including real-time scheduling research, focuses on generating fixed optimised timetables based on recorded real-time or hypothetical passenger OD data, which helps deal with the dynamic variations in passenger flow or deal with a specific operation circumstance. Most real-time rescheduling research concentrates on modifying timetables to normal only after some incident happens. And most of the passenger flow oriented rescheduling researches are also based on the situation which the system has already been suffered from disruption or disturbance and aim to decrease these impacts to the passenger flow, instead of changing dispatch strategies initiatively based on the real-time passenger flow variation. However, in real life, with irregular variations in passenger flow always occurring in real time, the

metro operation strategies also needs to be optimised swiftly even before the disruption or disturbance occurs. We considers transforming scheduling into a real-time optimisation process and modifying the traditional scheduling problem to a dynamic optimisation problem with a variable number of solutions and combining heuristic algorithms with a simulation method to increase the flexibility of the objective function, putting forward a new scheduling method which keeps generating and optimising operational strategies according to passenger flow in real time.

The main innovations of this research are:

(1) To propose an innovative concept of real-time scheduling optimisation. Instead of only optimising the timetable based on statistical or predicted passenger flow data before operation as in traditional research, this research keeps modifying dispatching strategies whole trains in the operational time horizon according to real-time passenger flow variation to ensure the metro system's real-time performance and flexibility.

(2) To be independent of established timetables. Unlike most traditional scheduling and rescheduling research, which focuses on proposing optimising strategies based on old timetables before operation, this research generates and modifies dispatching strategies for both dispatched trains and non-dispatched trains in real time without an established timetable.

(3) To design an innovative algorithm to solve the proposed problem, which increases the flexibility of the objective function and improve the computation time.

Chapter 3. Formulation and Extension of Dynamic Passenger Flow-Oriented Metro Scheduling and Real-Time Optimisation Problems

In the previous chapters, the problems of dynamic passenger flow-oriented real-time metro scheduling and optimisation have been described; the defects in the early research and the innovation of this research have also been proposed. The main objective of this research is to provide a method which can generate metro scheduling strategies based on dynamic passenger demands and optimise the strategies in real time based on passenger flow variations.

In this chapter, a mathematical train scheduling model for dynamic passenger flow scheduling and real-time optimisation, which is named passenger flow-oriented scheduling model (POSM), is presented to formulate the proposed scheduling problems by nonlinear programming (NLP) based on dynamic optimisation. In the first section, the general concepts of mathematical programming and dynamic optimisation are introduced. Then the formulation of POSM using NLP and the concept of dynamic optimisation is described in detail with a case explanation.

3.1 Introduction to Mathematical Programming

3.1.1 General Definitions of Mathematical Programming

The earliest application of mathematical programming can be traced back to the Second World War; at that time, there was an urgent need to allocate scarce resources to various military units. From 1947, when the American mathematician George Dantzig proposed

the simplex method for numerically solving linear programming problems (Dantzig 1948), to the genetic algorithm proposed by John Holland in the 1970s and the CPLEX Optimisation Studios developed by IBM in 1988, the development of mathematical programming has been rapid. Nowadays, mathematical programming is widely used in the manufacturing industry, finance, military, transportation, etc.

Generally, mathematical programming is used to determine several variables which can lead to a maximum or a minimum result in an objective function; these determinations have to satisfy a number of constraints based on the system's infrastructure. The mathematical form of a mathematical programming can be described as an optimisation problem with different constraints as shown in Equation 3-1.

$$\begin{cases} maximise\ f(x) \\ subject\ to: \\ a1_i(x) \leq b_1 \quad i = 1, \dots, m \\ a2_j(x) > b_2 \quad j = 1, \dots, n \\ a3_k(x) = b_3 \quad k = 1, \dots, o \\ x \in S \in \mathbb{R}^n \end{cases}$$

Equation 3 − 1

The function $f(x)$ is the objective function which determines the objective to match the desired result of the optimisation. Generally, these objectives include maximising and minimising the objective function. $a1_i(x) \leq b_1 \quad i = 1, \dots, m$, $a2_j(x) > b_2 \quad j = 1, \dots, n$ and $a3_k(x) = b_3 \quad k = 1, \dots, o$ are sets of constraints based on the optimised system which include equality constraints and inequality constraints.

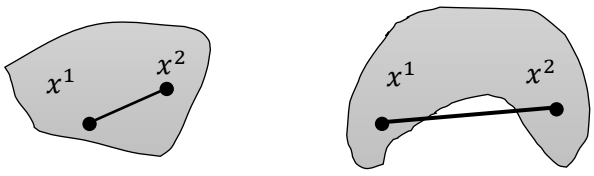
$\mathbb{R}^n$ represents the Euclidean space with n dimensions of this problem, S is a smaller mathematical set contained in the Euclidean space and x is the vector variable or a set of variables, such as $\{x_1, x_2, \dots, x_n\}$, which need to be calculated and lead to an optimised

result. Thus $x \in S \in \mathbb{R}^n$ defines the value range of the variables in the optimisation problem.

For the set S of a mathematical program, if and only if Equation 3-2 can be satisfied, the set S can be considered as convex.

$$\begin{cases} \lambda x^1 + (1-\lambda)x^2 \in S \\ \quad for\ all: \\ \quad \forall x^1 \in S \\ \quad \forall x^2 \in S \\ \quad \forall \lambda \in [0,1] \end{cases} \qquad \textbf{Equation } 3-2$$

The graphical representation of a convex set and a non-convex set is shown in Figure 3-1. Convex optimisation problems can be solved quickly and reliably up to a very large size, as they only have global optimum solutions. Contrarily, with different local optimum solutions, the solving processes are more complicate for non-convex problems.



Convex Non-convex

Figure 3-1 Graphical representation of a convex set and a non-convex set

The maximisation optimisation problem can be easily converted to a minimisation problem by converting the symbol of the objective function. The objective function is given by $maximise\ f(x) = minimise\ f'(x) = -f(x)$.

For the mathematical programming in Equation 3-1, if a set of variables x in the value range satisfies the constraints $a1_i(x) \leq b_1 \quad i = 1, \dots, m$, $a2_j(x) > b_2 \quad j = 1, \dots, n$,

$a3_k(x) = b_3 \quad k = 1, \dots, o$, this variable vector x can be called a solution of the objective function. If a solution in the value range can maximise the objective $f(x)$, this solution is a global optimal solution.

However, if a variable vector x^0 can only maximise the problem within its neighbourhood $N(x^0)$, this vector is a global optimum of the problem in a smaller range as shown in Equation 3-3. For the original problem, this vector is a local optimum solution. Generally, most mathematical problems aim to find the global optimum solution instead of the local optimum solution. Examples of a global optimum solution and local optimum solution are shown in Figure 3-2.

$$\begin{cases} maximise\ f(x) \\ subject\ to: \\ a1_i(x) \le b_1 \quad i = 1, \dots, m \\ a2_j(x) > b_2 \quad j = 1, \dots, n \\ a3_k(x) = b_3 \quad k = 1, \dots, o \\ x \in N(x^0) \in \mathbb{R}^n \end{cases}$$

Equation 3 − 3

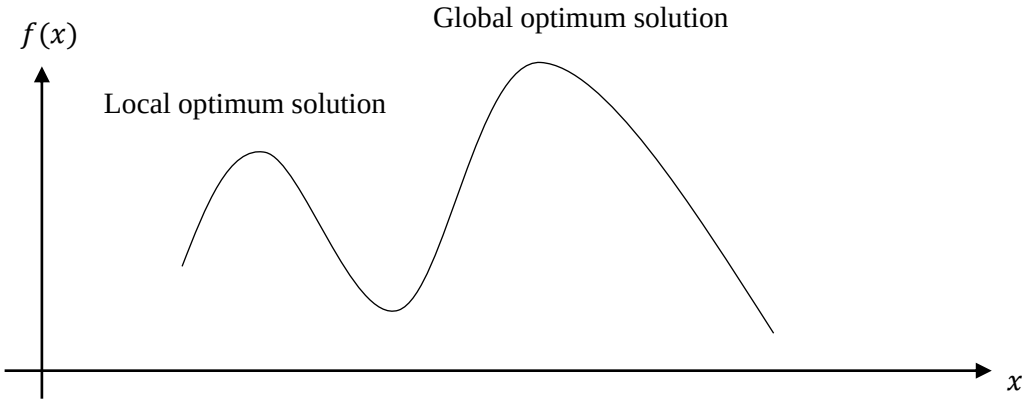


Figure 3-2 Examples of global and local optimum solutions

Generally, it is more difficult to calculate the solution for a mathematical problem with a non-convex set, as the global optimal solution is hard to characterise (Chen 2012; Jeter 2018; Hillier 2021) .

3.1.2 Classification of Problems in Mathematical Programming

Generally, mathematical programming problems can be classified into different types according to the characteristics of the different properties which include the objective function, the constraint functions and the definition of the value range. The main classifications are listed below:

1. Based on the linearity of the objective function and the constraint function, problems can be classified as linear programming or nonlinear programming.
2. Based on the convexity of the solutions' value range, problems can be classified as convex programming or non-convex programming.
3. Based on the continuity of the solutions' value range, problems can be classified as continuous programming or non-continuous programming.
4. Some mathematical programming problems do not have a constraint function; based on whether the constraint function is contained in the problem, the problem can be classified as unconstrained programming or constrained programming.
5. The solutions of some mathematical programming problems can only be integral values; these problems can be regarded as integer programming.

There are also different classifications based on other properties of mathematical programming problems, such as quadratic programming, conic programming, deterministic programming, stochastic programming, etc.

Most mathematical programming problems have two or more properties, for example the constraint function of a mathematical programming problem is nonlinear and the value range of the problem is an integer. Thus, the problem can be called nonlinear integer

programming. Classification of the main types of mathematical programming problems is shown in Table 3-1.

Classification	Objective Function	Constraint Function	Value Range
Continuous Programming	N/A	N/A	$S \in \mathbb{R}^n$ Continuous
Integrated Programming	N/A	N/A	$S \in \mathbb{Z}^n$ Discrete
Linear Programming	Linear	Linear	$S \in \mathbb{R}^n$
Nonlinear Programming	At least one function should be nonlinear		$S \in \mathbb{R}^n$
Unconstrained Programming	N/A	No	$S \in \mathbb{R}^n$
Constrained Programming	N/A	Yes	$S \in \mathbb{R}^n$
Convex Programming	Convex	Convex	$S \in \mathbb{R}^n$ Convex

Table 3-1 Main types of mathematical programming problems

3.1.3 Linear Programming

As the earliest mathematical programming method, the research on linear programming can be dated back to 1827 (Schrijver 2011); at that time Fourier published a method(known as Fourier-Motzkin elimination) to solve it (Sierksma and Zwols 2015). During 1946 and 1947, George B. Dantzig independently developed the simplex method (Dantzig 1948), which can tackle most linear programming problems. Since the Second World War, more and more research and applications based on linear programming have been implemented. The main feature of linear programming is that both the objective function and constraint function need to be linear. Thus, a linear programming problem can be presented as Equation 3-4:

$$\begin{cases} Maximise\ c^T x \\ subject\ to: \\ a_i^T x \le b_i\ i = 1, \dots, m \\ a_j^T x \le b_j\ j = 1, \dots, n \\ \dots \\ x \in S \in \mathbb{R}^n \\ c, a_i, a_j, \dots \in \mathbb{R}^n \\ b_i, b_j, \dots \in \mathbb{R}^n \end{cases} \qquad \textbf{Equation } 3-4$$

By adding slack variables (additional introduced variables to transform inequality constraints to equality constraints), all the 'less-than' equations can be transformed to 'equal' equations. Thus, the linear programming problem in Equation 3-4 can be transformed into a standard form as shown in Equation 3-5.

$$\begin{cases} Maximise\ z = c^T x \\ subject\ to: \\ Ax = b \\ x \ge 0 \end{cases} \qquad \textbf{Equation } 3-5$$

where z is the objective to be maximised, x is the variables, c is the vector of cost coefficients, A is a real $m \times n$ matrix, $b = (b_1, \dots, b_m)^T$ is the right-hand-side vector, and m and n are the number of constraints and variables, respectively. Generally, most linear programming problems will be transformed into the standard form before being solved.

The simplex method has been applied to solve linear programming problems from early on. A specific introduction and explanation of the simplex method can be found in the work of Minoux (2004). In 1984, a new interior point method (Adler et al. 1989) was proposed to solve the linear programming problem by Narendra Karmarkar, and this method can also be applied to solve nonlinear convex programming problems.

3.1.4 Nonlinear Programming

In contrast to linear programming, if there is at least one nonlinear function in a problem (including the objective function and constraint function), it can be considered as nonlinear programming.

Compared with linear programming, nonlinear programming is hard to solve. Thus, for different nonlinear programming problems, the solution method is different. There are two main types of nonlinear programming, unconstrained and constrained, which will be introduced separately.

3.1.4.1 Unconstrained Nonlinear Programming

The mathematical definition of an unconstrained nonlinear programming problem can be shown by Equation 3-6. As there is no constraint function, the objective function $f(x)$ must be nonlinear.

$$\begin{cases} Maximise\ f(x) \\ x \in \mathbb{R}^n \end{cases} \qquad \textbf{Equation } 3-6$$

Most of the methods for solving an unconstrained nonlinear programming problem aim to find a starting point x_0, then try to find the optimal solution x^* by iteration. The performance of the solving process, such as the calculation speed and quality of the result, will be significantly impacted by the starting point x_0. Generally, the steepest decent method, Newton's method, quasi-Newton's method and some other methods without the calculation of derivatives are applied (Powell 1964).

Sometimes, the objective function of some unconstrained nonlinear programming problems is concave or convex; however it may not be everywhere differentiable. This specific kind of unconstrained nonlinear programming problem can be dealt with by the decomposition methods (Tai and Espedal 1998).

3.1.4.2 Constrained Nonlinear Programming

The mathematical definition of constrained nonlinear programming is shown in Equation 3-7. It is similar to constrained linear programming, but has at least one nonlinear function. Optimisation problems that have constraints in practice are normally represented by constrained nonlinear programming.

$$\begin{cases} Maximise\ f(x) \\ subject\ to: \\ a1_i(x) \le b_1 \quad i = 1, \dots, m \\ a2_j(x) = b_1 \quad j = 1, \dots, m \\ x \in S \in \mathbb{R}^n \end{cases} \qquad \textbf{Equation } 3-7$$

The necessary conditions to solve a nonlinear programming problem are the Karush–Kuhn–Tucker conditions (Kuhn–Tucker or KKT conditions) which were proposed by William Karush in his Master's thesis in 1939 as the necessary conditions for solving constrained nonlinear programming problems and originally named after Harold W. Kuhn and Albert W. Tucker in 1951.

Generally, two big categories of methods are usually applied to solve constrained nonlinear programming problems, direct methods and methods based on the concept of duality.

The normal direct methods include the method of changing the variables, method of feasible directions, the reduced gradient method, Newton's method, etc. (Schenk 1998). Firstly, these methods will generate a number of solutions which can satisfy the constraint function, then choose the optimum solutions step by step.

In contrast, methods based on the concept of duality will convert the proposed constrained nonlinear programming problem into am unconstrained nonlinear programming problem first, and then solve this problem based on the methods for unconstrained nonlinear

programming. These methods mainly include penalty function methods and classical Lagrange methods.

With the computation time requirements for the iterative process of direct methods, and due to time limits in real-life operation, in general, direct methods can only generate a suboptimal solution instead of the optimal solution. Contrary to direct methods, methods using the concept of duality are more robust and obtain global convergence more easily, but their operating process is much more complicated.

3.1.5 Integer Programming

An integer programming problem is a mathematical programming problem with some or all of the variables restricted to being integers. Based on this restriction, problems can be divided into pure integer programming and mixed-integer programming problems. All the variables need to be taken as integer values for a pure integer programming problem. In contrast, in a mixed-integer programming problem, only some of the variables are constrained to be integers while other variables are allowed to be non-integers, which means this type of problem can also include some continuous variables. There is a special type of integer programming called 0-1 programming or binary integer programming which only takes values of 0 or 1 for its variables. In recent years, because of its accessibility, integer programming has been widely applied in different areas, such as production planning, scheduling, territorial partitioning and telecommunications networks. The mathematical definition of a pure integer programming problem can be expressed as in Equation 3-8:

$$\begin{cases} Maximise\ f(x) \\ subject\ to \\ a1_i(x) \le b_1 \quad i = 1, \dots, m \\ a2_j(x) = b_1 \quad j = 1, \dots, m \\ x \ge 0 \\ x \in S \in \mathbb{Z}^n \end{cases}$$

Equation 3 − 8

Compared with normal linear or nonlinear programming problems, the values of a pure integer programming problem are restricted to being integers which are larger or equal to zero. Generally, there are three main types of method for solving an integer programming problem, branch and bound methods, cutting-plane methods and meta-heuristic methods.

Branch and bound methods

The branch and bound method (BB, or BnB) was firstly proposed by Ailsa Land and Alison Doig whilst carrying out research at the London School of Economics sponsored by British Petroleum in 1960; it has become the most commonly used tool for solving integer programming and has been improved by many authors (Land and Doig 1960) . Generally, branch and bound methods mainly include these steps:

Branching: In the branching process, the whole value range $\boldsymbol{S}$ of the variables will be divided into some smaller value set $\{S_1, S_2, \dots, S_k\}$, $\cup_1^k S_i = \boldsymbol{S}$. A tree structure with the subset S_i as its nodes can be defined.

Pruning: In this process, based on the objective, for example, maximise the value of $f(x)$, the upper and lower bounds will be calculated for each subset S_i. If the upper bound of a subset S_k is smaller than the lower bound of a subset S_j, then the subset S_k will be discarded. This process is similar to cutting the bad branches from a tree.

These branching and pruning steps will continue to discard the bad values of $f(x)$ until the set $\boldsymbol{S}$ is reduced to a single set which can maximise the value of the objective function $f(x)$.

The calculation time of the branch and bound method is impacted by the selection of criteria for branching, bounding and pruning, and the stopping criteria; in the worst case, these may lead to exponential time complexities. In practice, the branch and bound method is a systematic method for solving programming problems, especially for integer programming (Clausen 2003).

Cutting-plane methods

Cutting-plane methods are a class of methods that iteratively refine (cut) the objective function by means of linear inequalities; they were introduced by Ralph E. Gomory in the 1950s and are commonly used to find integer solutions for mixed-integer linear programming problems (Gilmore and Gomory 1961; Marchand et al. 2002).

Generally, cutting-plane methods will relax the integer restriction of variables for the mathematical programming problem first, and solve the relaxed problems to generate a basic feasible solution; the solution is considered to be a vertex of the convex polytope in geometry that consists of all the feasible points. If the vertex is an integer, the solution of the original problem has been found; otherwise, we need to find a hyperplane with the vertex and all feasible integer points on each side and add it as an additional constraint to create a modified linear programming problem, then solve the new problem and keep iterating the process until the integer solution can be found. Cutting-plane methods can also be extended to solving nonlinear programming problems (Konno et al. 2003).

Meta-heuristic methods

Most mathematical programming problems in real life are hard to solve using traditional methods. As the size of the problem grows, the value range may be larger, and also the number of variables will be increased. These practical problems will highly impact the computation time of a traditional method.

With the development of computer science, meta-heuristic methods have become popular in recent years. A meta-heuristic is a higher-level procedure or heuristic designed to find, generate, or select a heuristic that may provide a sufficiently good solution to a mathematical programming problem, especially with incomplete or imperfect information or limited computation capacity. The meta-heuristic method was proposed in the 1950s and is widely used for solving integer programming problems; it was defined as "an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions" by Osman and Laporte in 1996 (Osman and Laporte 1996).

Generally, meta-heuristic methods implement some form of stochastic optimisation and deal with the optimisation problem with a discrete value space, the solution being dependent on the set of random variables generated. Thus, most meta-heuristic methods do not guarantee that a globally optimal solution can be found. The commonly used meta-heuristic methods include: the tabu search algorithm (Glover 1986) which uses a local or neighbourhood search procedure to iteratively move from one potential solution to an improved solution; the simulated annealing algorithm (Kirkpatrick et al. 1983) which is based on the notion of slow cooling, the temperature progressively decreasing from an initial positive value to zero – the algorithm randomly selects a solution in each time step, measures its quality, and moves to it according to the temperature-dependent probabilities

of selecting better or worse solutions; the genetic algorithm (Holland 2010) which is inspired by the process of natural selection – a population of candidate solution will be generated randomly and evolved toward better solutions based on the objective; the solutions of genetic algorithms are generally represented in binary as strings of 0s and 1s, thus this method is extremely suitable for 0-1 programming or binary integer programming problems. The other widely used meta-heuristic methods include scatter search (Glover 1977), colony optimisation (Dorigo 1992), particle swarm optimisation (Kennedy 1995) and differential evolution (Storn and Price 1997).

3.1.6 Dynamic Programming

Many real-world optimisation problems are dynamic. The field of dynamic optimisation deals with such problems where the search space or the parameters of objective function change over time. Generally, an unconstrained single-objective DOP (dynamic optimisation problem) can be defined by Equation 3-9:

$$\begin{cases} f\left(\vec{x}, \vec{a}^{(t)}\right) \\ x \in \mathbb{R}^n \end{cases} \qquad \textbf{Equation 3 - 9}$$

where f is the objective function, $\vec{x}$ is a solution in the search space, $\vec{a}$ is a vector of time-varying objective function control parameters and t is the time index. In the real world, $\vec{a}$ could be a series of environmental parameters which change over time; as a specific example, in a metro system, it can be the passenger flow entry rate which keeps changing with time. It can also include other time-variant environmental parameters like the domain of variables, number of variables and the interactions between variables.

Most existing DOP research considers that parameter changes only happen in a discrete time interval. For example, $t \in \{1, 2, \ldots, T\}$, the series of the objective functions, is:

$$\langle f(\vec{x}, \vec{a}^{(1)}) | f(\vec{x}, \vec{a}^{(2)}) | \dots | f(\vec{x}, \vec{a}^{(T)}) \rangle \qquad \textbf{Equation3} - \textbf{10}$$

In normal research, the change of parameters between successive functions is small, i.e., keeping the passenger flow entry rate the same for a specific time interval, all the functions in the series can be presented by one equation. In cases where the control parameter changes are unstable, especially where there are connections between parameters, i.e., passenger flow entry rate keeps changing with time, and it will impact the number of passengers who can board or alight from a specific train, each objective needs to be considered independent and accumulate the objective function in each time point to the final objective function, as shown in Equation 3-11 (Yazdani et al. 2021).

$$\begin{cases} F(\vec{x}) = \sum_{t=1}^{T} f(\vec{x}, \vec{a}^{(t)}) \\ t \in \{1, 2, \dots, T\} \\ x \in R^n \end{cases} \qquad \textbf{Equation 3} - \textbf{11}$$

3.2 Formulation of Passenger Flow-Oriented Scheduling Model (POSM) Based on Nonlinear Integer Programming

In order to get a better understanding and solve passenger flow-oriented scheduling and real-time optimisation problems, the problems need to be formulated into mathematical models. A Passenger Flow-Oriented Scheduling Model (POSM) is proposed in this section.

Based on other rail scheduling research (Wang et al. 2013; Chen et al. 2015), the objectives for the scheduling optimisation must be given first. According to a random passenger interview from the Wuhan Metro system, most passengers are concerned about

their waiting time in the station to judge the service quality. Also, from the field study on London Underground, operators consider passenger waiting time the most crucial index to satisfy passengers. Passenger waiting time directly impacts the overall passenger experience and satisfaction. Long waiting times can lead to frustration and inconvenience, potentially causing passengers to seek alternative transportation. Additionally, it can affect the operational efficiency of the metro system, as passengers may become overcrowded at stations. By reducing waiting times, metro operators can improve the reliability and overall convenience of the service, making it a more attractive option for passengers.

And according to the discussion with the China Academy of Transportation Science. The full load rate, also known as the "train load factor" or “train capacity occupation ratio” is considered an essential index in metro service because it measures the utilisation of the available capacity on each train. The full load rate indicates how effectively the metro system is utilising its resources, such as rolling stock, and how well it is meeting the demand for transportation. A high full load rate indicates that the metro service operates at maximum efficiency. In contrast, a low full load rate may indicate the underutilisation of resources, which can result in higher operational costs and reduced revenues. In order to optimise their operations, metro operators need to regularly monitor the full load rate and make adjustments as needed to ensure that their resources are being used effectively and efficiently.

Thus, in this research, a definition of SQI is used which integrates passenger waiting times and train capacity occupation ratio. The objective is to find the minimum passenger waiting time and maximum train capacity occupation ratio. The mathematical representation of the objective function is shown in Equation 3-12.

$$\min \text{SQI} = \omega_1 \sum_{i,s} w_{i,s} + \omega_2 \sum_{i,s} \frac{1}{cr_{i,s}} \qquad \textbf{Equation3} - \textbf{12}$$

where ω_1 denotes the weighting of passenger waiting time in the operation, while ω_2 denotes the weighting of capacity occupation ratio. The detailed ratios of weightings are determined based on different optimisation requirements. Generally, in peak time, to satisfy high passenger demand, passenger waiting time will be assigned a bigger weighting value; in off-peak time, to increase occupation ratio, passenger waiting time will be assigned a smaller value. $w_{i,s}$ is the waiting time for different trains in each station. $cr_{i,s}$ is each train's capacity occupation ratio after it departs from different stations.

In this objective function, the aim is to increase train capacity occupation ratio, while passenger waiting time should decrease. In this situation, the reciprocal value of capacity occupation ratio is applied as shown in Equation 3-12, which modifies the objective function to a standard form and unifies the objectives to minimise.

Let S be the set of stations in the system, T be the discrete operation time horizon considered and I be the set of trains scheduled in the considered time horizon. The index s is associated with a specific station, t denotes a specific time in operation time horizon and i represents a specific train number in the system.

$$s \in S = \{0, 1, \dots, |S|\} \qquad \textbf{Equation 3} - \textbf{13}$$

$$t \in T = \{0, 1, \dots, |T|\} \qquad \textbf{Equation 3} - \textbf{14}$$

$$i \in I = \{0, 1, \dots, |I|\} \qquad \textbf{Equation 3} - \textbf{15}$$

Let H be the set of optional departure time intervals between two trains and E be the set of optional dwelling times in stations for each train. $ld_{i,i-1}$ represents the choice of

departure time interval between train i and train $i-1$ from the optional set, while $le_{i,s}$ represents the choice of dwelling time for train i at station s from the optional set. $hd_{i,i-1}$ denotes the departure time interval between train i and train $i-1$ at the first station; this time interval should be included in the collection of optional intervals. And $e_{i,s}$ denotes the dwelling time for train i at station s; dwelling time should also be chosen from the optional dwelling time set.

$$hd_{i,i-1} \in H(ld_{i,i-1}) \qquad \textbf{Equation } 3-16$$

$$e_{i,s} \in E(le_{i,s}) \qquad \textbf{Equation } 3-17$$

Let $h_{i,i-1,s}$ denote the time interval (departure time of the first train minus arrival time of the next train) between train i and train $i-1$ at station s.

$$h_{i,i-1,0} = hd_{i,i-1} \qquad \textbf{Equation } 3-18$$

For systematic operational safety, the minimum headway h_{min} should be provided.

$$h_{i,i-1,s} \geq h_{min} \qquad \textbf{Equation } 3-19$$

Generally, in a metro system, the scheduling process mainly focuses on five parameters: arrival times, departure times, running times, dwelling times and departure time intervals. For this research, the dwelling times and departure intervals are defined as the decision variables; the relationship between these and other parameters should be provided.

The arrival time $a_{0,0}$ for the first train at the first station is set to a constant value. The arrival time $a_{i,0}$ of train i at the first station equals the departure time $d_{i-1,0}$ of train $i-1$ plus the departure time interval from the optional set.

$$a_{i,0} = d_{i-1,0} + hd_{i,i-1} \qquad \textbf{Equation } 3-20$$

The departure time $d_{i,s}$ of train i at station s is equal to the sum of the arrival time $a_{i,s}$ and the dwell time $e_{i,s}$.

$$d_{i,s} = a_{i,\mathrm{s}} + e_{i,s} \qquad \textbf{Equation } 3-21$$

For train i, planned running time $r'_{i,(s-1,s)}$ added to the departure time $d_{i,\mathrm{s}-1}$ equals its planned arrival time $a'_{i,s}$; if $a'_{i,s}$ minus the last train's departure time $d_{i-1,s}$ is larger than the minimum headway, its running time $r_{i,(s-1,s)}$ is equal to the planned running time, else its running time is equal to the minimum headway added to last train's departure time in station s minus the departure time in station $s-1$ for train i.

$$a'_{i,s} = r'_{i,(s-1,s)} + d_{i,\mathrm{s}-1} \qquad \textbf{Equation } 3-22$$

$$r_{i,(s-1,s)} = \begin{cases} r'_{i,(s-1,s)} & a'_{i,s} - d_{i-1,s} \geq h_{min} \\ h_{min} - d_{i,\mathrm{s}-1} + d_{i-1,s} & a'_{i,s} - d_{i-1,s} < h_{min} \end{cases} \qquad \textbf{Equation } 3-23$$

The arrival time is equal to the last train's departure time added to the running time.

$$a_{i,\mathrm{s}} = d_{i,\mathrm{s}-1} + r_{i,(s-1,s)} \quad s > 0 \qquad \textbf{Equation } 3-24$$

As for passenger flow, the number of passengers $p_{i,s}^{wait}$ at station s waiting for train i can be represented by the passengers $p_{i-1,s}^{left}$ who were left by train $i-1$ and newly arrived passengers based on the time-dependent arrival rate $\lambda_s(t)$ between the departure times of train $i-1$ and train i.

$$p_{i,s}^{wait} = p_{i-1,s}^{left} + \sum_{d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t) \qquad \textbf{Equation } 3-25$$

The number of passengers $p_{i,s}^{train}$ on train i before it arrives at station s must be smaller than or equal to its maximum capacity $c_{i,max}$. The remaining capacity $c_{i,s}$ of train i at

station s after passengers alight equals the train's maximum capacity minus $p_{i,s}^{train}$ plus the spare capacity $p_{i,s}^{alight}$ because of passengers alighting. With more and more metro systems have been implemented with platform screen doors and safety officers on the platform, the impacts from passenger alighting and boarding to journey have been reduced. Thus, this research does not consider the alighting and boarding impacts.

$$c_{i,s} = c_{i,max} - p_{i,s}^{train} + p_{i,s}^{alight} \qquad \textbf{Equation 3-26}$$

The number of passengers $p_{i,s}^{board}$ who can board train i at station s cannot be larger than the remaining train capacity.

$$p_{i,s}^{board} = \min\left(c_{i,s}, p_{i,s}^{wait}\right) \qquad \textbf{Equation 3-27}$$

If a train's capacity is enough, all passengers can board the train. Otherwise, some passengers will be left. The number of passengers $p_{i,s}^{left}$ left by train i at station s is given by:

$$p_{i,s}^{left} = \max\left(p_{i,s}^{wait} - c_{i,s}, 0\right) \qquad \textbf{Equation 3-28}$$

The number of passengers on train i after station s is equal to the number of passengers on the train before it arrives minus alighting passengers plus boarding passengers.

$$p_{i,s+1}^{train} = p_{i,s}^{train} - p_{i,s}^{alight} + p_{i,s}^{board} \qquad \textbf{Equation 3-29}$$

Thus the passenger waiting time part $w_{i,s}$ in the objective function Equation 3-14 can be transformed into the form of Equation 3-30.

$$w_{i,s} = p_{i,s}^{left}\left(d_{i,s} - d_{i-1,s}\right) + \sum_{t = d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t)\left(d_{i,s} - t + 1\right) \qquad \textbf{Equation 3-30}$$

The trains' full load rate $cr_{i,s}$ can be transformed into the form of Equation 3-31.

$$cr_{i,s} = \frac{p_{i,s}^{train}}{c_{i,max}} \qquad \textbf{Equation } 3-31$$

The objective function presented in this thesis is to minimise the SQI shown in Equation 3-12. For the POSM, the detailed form of Equation 3-14 can be transformed into Equation 3-32.

$$\text{SQI} = \omega_1 * \sum_{i=0}^{I}\sum_{s=0}^{S-1}\left[p_{i-1,s}^{left}\left(d_{i,s} - d_{i-1,s}\right) + \sum_{t\,=\,d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t)\left(d_{i,s} - t + 1\right)\right] +$$

$$\omega_2 * \sum_{i=0}^{I}\sum_{s=1}^{S}\left[\frac{p_{i,s}^{train}}{c_{i,max}}\right]$$

$$\forall d_{-1,s} = 0;\ \forall l_{-1,s} = 0 \qquad \textbf{Equation3} - \textbf{32}$$

where ω_1 and ω_2 are the weightings of passenger waiting time and capacity occupation ratio in the operation, the values of which are based on service quality requirements. $p_{i-1,s}^{left}$ is the number of passengers left by the last train $i-1$; $d_{i,s}$ and $d_{i-1,s}$ are the departure times of train i and train $i-1$ in station s. t represents a specific time in the operation time horizon; $\lambda_s(t)$ is the dynamic passenger flow arrival rate based on time variation. $p_{i,s}^{train}$ is the number of passengers on train i when it arrives in station s; $c_{i,max}$ is the maximum capacity of train i.

The aim of passenger flow-oriented scheduling is to find the optimal departure time $d_{i,s}$ for each train to minimise the SQI presented in Equation 3-12. With Equations 3-20 to 3-24, a train's departure time $d_{i,s}$ can be represented by departure time interval $hd_{i,i-1}$ and dwelling time $e_{i,s}$, as in Equation 3-33.

$$d_{i,s} = \sum_{1}^{s}\left[r_{i,(s-1,s)} + e_{i,s}\right] + \sum_{i=1}^{i}(e_{i-1,0} + hd_{i,i-1}) + a_{0,0} + e_{i,0} \qquad \textbf{Equation } 3-33$$

Thus, the problem presented above is a nonlinear integer programming problem; the variables that need to be optimised are: $hd_{i,i-1}$ and $e_{i,s}$. For this research, we do not modify planned running time actively. The values of all the other variables are given or can be calculated in operations. The constraint conditions shown in the equation above are the system constraints based on safe operation and infrastructure which need to be observed in practical metro operations. In some specific operation situations, more constraints should be added to the POSM, based on actual operational and safety requirements.

As presented above, the proposed problem of train scheduling based on dynamic passenger demand can be formulated with nonlinear integer programming as follows:

Objective:

Minimise

$$\text{SQI} = \omega_1 \sum_{i=0}^{I}\sum_{s=0}^{S-1}\left[p_{i-1,s}^{left}\left(d_{i,s} - d_{i-1,s}\right) + \sum_{t\,=\,d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t)\left(d_{i,s} - t + 1\right)\right]$$

$$+\ \omega_2 \sum_{i=0}^{I}\sum_{s=1}^{S}\left[\frac{c_{i,max}}{p_{i,s}^{train}}\right]$$

Subject to:

$$\forall d_{-1,s} = 0;\ \forall l_{-1,s} = 0$$

$$d_{i,s} = \sum_{1}^{s}\left[r_{i,(s-1,s)} + e_{i,s}\right] + \sum_{i=1}^{i}(e_{i-1,0} + hd_{i,i-1}) + a_{0,0} + e_{i,0}\ \ hd_{i,i-1} \in H;\ e_{i,s} \in E$$

$$h_{i,i-1,0} = hd_{i,i-1} \quad hd_{i,i-1} \in H$$

$$h_{i,i-1,s} \geq h_{min}$$

and

$$p_{i,s}^{wait} = p_{i-1,s}^{left} + \sum_{d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t)$$

$$c_{i,s} = c_{i,max} - p_{i,s}^{train} + p_{i,s}^{alight}$$

$$p_{i,s}^{board} = min\ (c_{i,s}, p_{i,s}^{wait})$$

$$p_{i,s}^{left} = max\left(p_{i,s}^{wait} - c_{i,s}, 0\right)$$

$$p_{i,s+1}^{train} = p_{i,s}^{train} - p_{i,s}^{alight} + p_{i,s}^{board}$$

As passenger waiting time is hard to be calculated by statistics in operation and this model is difficult to be tested in real life, we can analyse the parameters in the objective function to validate the correction of this model. $p_{i-1,s}^{left}$ defines the left passenger in the station after departing, the waiting time of this passenger will directly be impacted by the departure time between two trains, which is $\left(d_{i,s} - d_{i-1,s}\right)$, thus the left passenger waiting time is represented by $p_{i-1,s}^{left}\left(d_{i,s} - d_{i-1,s}\right)$. New passenger waiting time is directly impacted by the passenger arrival time which is defined in the passenger arrival rate $\lambda_s(t)$, and the difference between the arrival time and the next train's departure time $\lambda_s(t)\left(d_{i,s} - t + 1\right)$. For the train full load rate, we compare the passenger number on the train after train departing from each station $p_{i,s}^{train}$ with the train's maximum capacity $c_{i,max}$ We also implemented the macroscopic passenger and metro simulator, which was proposed in

Chapter 4 to validate our model by calculation, the results from simulation can correspond with the calculating results from the mathematical model.

The presented nonlinear integer programming problem is engaged to find the optimal departure interval $hd_{i,i-1}$ between two trains and the dwell time $e_{i,s}$ for each train at each station with dynamic passenger flow parameter $\lambda_s(t)$ which keeps changing with time.

3.3 Derived POSM for Real-Time Optimisation Model

Based on the POSM shown above, a metro timetable can be optimised according to collected time-varying passenger flow data. However, in actual operation, because of irregular variations, passenger flow data will keep changing in real time. With the development of passenger counting systems, these irregular data can be detected and predicted in real time. Thus, the timetable generated by POSM also needs to be optimised in real time. However, it is impossible to adjust the entire timetable of a service which has been dispatched and is running in the network. Thus, to optimise the timetable based on detected variations in real time, the departure time generated $d_{i,s}^{old}$ should be compared with the detection time of the variation t_D. If $d_{i,s}^{old}$ is earlier than or equal to the time at which the variation detected, it cannot be optimised and these departure times will be regarded as parameters in the objective function in future optimisation, else it is still a variable $d_{i,s}^{new}$ which can be optimised in real time:

$$d_{i,s}^{real-time} = \begin{cases} d_{i,s}^{fixed} & t_D \leq d_{i,s}^{old} \\ d_{i,s}^{new} & t_D > d_{i,s}^{old} \end{cases} \qquad \textbf{Equation } 3-34$$

Based on the new equation proposed above, this problem turns into a DOP with a flexible number of variables. The number of variables in real-time scheduling optimisation will

keep varying with the detecting time. Thus the POSM with real-time optimisation can be modified as follows:

Objective:

Minimise

$$\text{SQI} = \omega_1 \sum_{i=0}^{I} \sum_{s=0}^{S-1} \left[p_{i-1,s}^{left} \left(d_{i,s}^{real-time} - d_{i-1,s}^{real-time} \right) + \sum_{t=d_{i-1,s}^{real-time}+1}^{d_{i,s}^{real-time}} \lambda_s(t) \left(d_{i,s}^{real-time} - t + 1 \right) \right] + \omega_2 \sum_{i=0}^{I} \sum_{s=1}^{S} \left[\frac{c_{i,max}}{p_{i,s}^{train}} \right]$$

Subject to:

$$\forall d_{-1,s} = 0;\ \forall l_{-1,s} = 0$$

$$d_{i,s} = \sum_{1}^{s} \left[r_{i,(s-1,s)} + e_{i,s} \right] + \sum_{i=1}^{i} (e_{i-1,0} + hd_{i,i-1}) + a_{0,0} + e_{i,0} \quad hd_{i,i-1} \in H;\ e_{i,s} \in E$$

$$h_{i,i-1,0} = hd_{i,i-1} \quad hd_{i,i-1} \in H$$

$$h_{i,i-1,s} \geq h_{min}$$

and

$$p_{i,s}^{wait} = p_{i-1,s}^{left} + \sum_{d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t)$$

$$c_{i,s} = c_{i,max} - p_{i,s}^{train} + p_{i,s}^{alight}$$

$$p_{i,s}^{board} = min\ (c_{i,s}, p_{i,s}^{wait})$$

$$p_{i,s}^{left} = max\left(p_{i,s}^{wait} - c_{i,s}, 0\right)$$

$$p_{i,s+1}^{train} = p_{i,s}^{train} - p_{i,s}^{alight} + p_{i,s}^{board}$$

and

$$d_{i,s}^{real-time} = \begin{cases} d_{i,s}^{fixed} & t_D \leq d_{i,s}^{old} \\ d_{i,s}^{new} & t_D > d_{i,s}^{old} \end{cases}$$

3.4 Explanation of POSM With a Typical Case

In the last section, the formulation of POSM and real-time optimisation is presented based on integer programming; in this section, the model is explained in detail with a typical case.

Figure 3-3 shows the layout of a typical single-line metro system, where the trains depart from the original station to their destination based on a given departure time interval and dwelling time at each station.

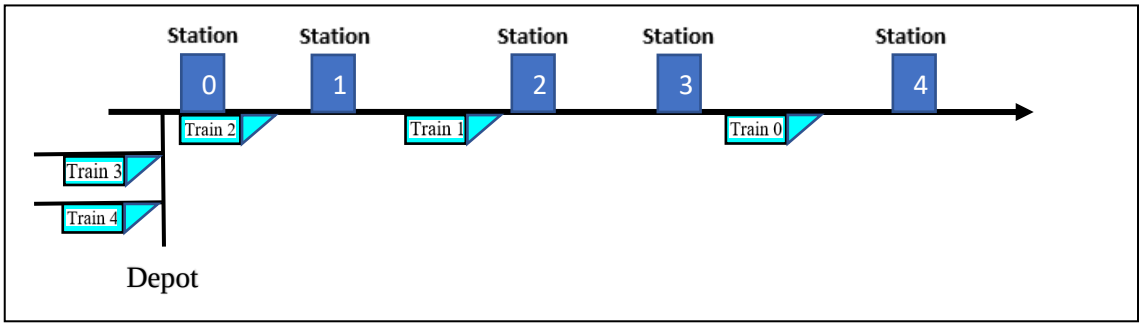


Figure 3-3 Layout of a typical single-line metro system

It is assumed that there are five trains in a five-station single-line system in the research time horizon. Here the stations and trains are assigned with a unique number as shown in Equations 3-35 and 3-36.

$$s \in S = \{0, 1, \dots, |S|\}, |S| = \mathbf{4} \qquad \textbf{Equation } \mathbf{3-35}$$

$$i \in I = \{0, 1, \dots, |I|\}, |I| = \mathbf{4} \qquad \textbf{Equation } \mathbf{3-36}$$

Three optional departure time intervals and two dwelling time choices are used in the system, as shown in Equations 3-37 and 3-38:

$$H = \{3, 5\} \qquad \textbf{Equation } \mathbf{3-37}$$

$$E = \{1, 2\} \qquad \textbf{Equation } \mathbf{3-38}$$

Because train 0 is the first train operating in the research time horizon, the departure time interval of train 0 does not need to be considered, and as the final station, station 4, will not impact the operation result, the dwelling times at station 4 also do not need to be considered. Thus, possible departure and dwelling strategy choices are shown as follows; the solution of departure interval and dwelling time based on the choices are listed in Tables 3-2 and 3-3:

$$ld_{i,i-1} = \{0,1,1,0\} \qquad \textbf{Equation } \mathbf{3-39}$$

i	1	2	3	4
$hd_{i,i-1}$	3	5	5	3

Table 3-2 Departure interval choices

$$le_{i,s} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad s < 4 \qquad \textbf{Equation } \mathbf{3-40}$$

	i \ s	**0**	**1**	**2**	**3**
$e_{i,s}$	**0**	**1**	**1**	**1**	**2**
	1	**2**	**1**	**2**	**1**
	2	**2**	**2**	**1**	**1**
	3	**1**	**2**	**2**	**2**
	4	**1**	**1**	**1**	**1**

Table 3-3 Dwelling time choices

The planned running time $r'_{i,(s-1,s)}$ between different stations can be assumed to be the following:

$$r'_{i,(s-1,s)} = \{2, 3, 2\} \quad 0 < s < 4 \qquad \textbf{Equation } 3-41$$

Based on Equations 3-37 to 3-43, the arrival time $a_{i,s}$ and departure time $d_{i,s}$ of different trains at different stations can be calculated; the results are listed in Tables 3-4 and 3-5.

	i \ s	0	1	2	3
$a_{i,s}$	0	0	3	7	10
	1	4	8	12	16
	2	11	15	20	22
	3	18	21	26	30
	4	22	25	29	32

Table 3-4 Arrival time $a_{i,s}$

	i \ s	0	1	2	3
$d_{i,s}$	0	1	4	8	12
	1	6	9	14	17
	2	13	17	21	23
	3	19	23	28	32
	4	23	26	30	33

Table 3-5 Departure time $d_{i,s}$

Setting the minimum headway between two trains $h_{min} = 2$, based on Equations 3-24 and 3-25, the arrival time in the green box of Table 3-4 needs to be modified to ensure operational safety. Thus, the arrival and departure times after modification are shown in Tables 3-6 and 3-7.

	i \ s	0	1	2	3
$a_{i,s}$	0	0	3	7	10
	1	4	8	12	16
	2	11	15	20	22
	3	18	21	26	30
	4	22	25	30	34

Table 3-6 Arrival time $a_{i,s}$ after safety modification

	i \ s	0	1	2	3
$d_{i,s}$	0	1	4	8	12
	1	6	9	14	17
	2	13	17	21	23
	3	19	23	28	32
	4	23	26	31	35

Table 3-7 Departure time $d_{i,s}$ after safety modification

Assuming the time-dependent arrival rate $\lambda_s(t)$ based on passenger-flow arrival rate, OD matrices are shown in Tables 3-8 and 3-9. At 20 minutes, there is a variation to the arrival rate. For now, these research passenger flow data need to be assumed, modified and forecasted based on the metro system's historical statistics, but with the development of passenger counting systems and AFC systems, more accurate passenger flow data can be collected and analysed in future.

Destination station / **Origin station**	**0**	**1**	**2**	**3**	$\lambda_s(t)$
0	**x**	**2**	**3**	**2**	**7**
1	**x**	**x**	**4**	**3**	**7**
2	**x**	**x**	**x**	**5**	**5**
3	**x**	**x**	**x**	**x**	**0**

Table 3-8 Passenger arrival rate OD matrix and time-dependent arrival rate $\lambda_s(t)$ between 0 and 20 minutes

Destination station / Origin station	0	1	2	3	$\lambda_s(t)$
0	**x**	**5**	**3**	**5**	**13**
1	**x**	**x**	**6**	**6**	**12**
2	**x**	**x**	**x**	**2**	**2**
3	**x**	**x**	**x**	**x**	**0**

Table 3-9 Passenger arrival rate OD matrix and time-dependent arrival rate $\lambda_s(t)$ after 20 minutes

The capacity of trains $c_{i,max}$ is a given value based on the vehicle in the system. With the $\lambda_s(t)$ data from the table above and $c_{i,max}$, according to Equations 3-27 to Equation 3-31, the passenger flow parameters required in the objective function can be calculated.

Regarding the presented example, in the research time horizon, there are five trains running in the system with two optional dwelling time in four stations, and four trains departing with two optional departure time intervals (the arrival time of the first train at the first station is predetermined). The number of possible dwelling and departing strategy decisions is $2^4 \times (2^4)^5 = 2^{24} = 16777216$. Because of the huge number of decisions, a special decision array can be determined to generate the decision tree (Table 3-10).

$le_{0,0}$	$le_{0,1}$	$le_{0,2}$	$le_{0,3}$	$ld_{1,0}$	$le_{1,0}$	$le_{1,1}$	$le_{1,2}$	$le_{1,3}$	$ld_{2,1}$	$le_{2,0}$	$le_{2,1}$	$le_{2,2}$	$le_{2,3}$
0	0	0	1	0	1	0	1	0	1	1	1	0	0
$ld_{3,2}$	$le_{3,0}$	$le_{3,1}$	$le_{3,2}$	$le_{3,3}$	$ld_{4,3}$	$le_{4,0}$	$le_{4,1}$	$le_{4,2}$	$le_{4,3}$				
1	0	1	1	1	0	0	0	0	0				

Table 3-10 Example of decision array table

Green cells in the decision array table indicate the departure interval choice between two trains, while yellow cells indicate the dwelling time choice in each station for each train.

In the presented example above, the decision array is set to [0,0,0,1,0,1,0,1,0,1,1,1,0,0,1,0,1,1,1,0,0,0,0,0].

The dispatching process for the five trains in the five-station system can be represented by the decision tree shown in Figure 3-4.

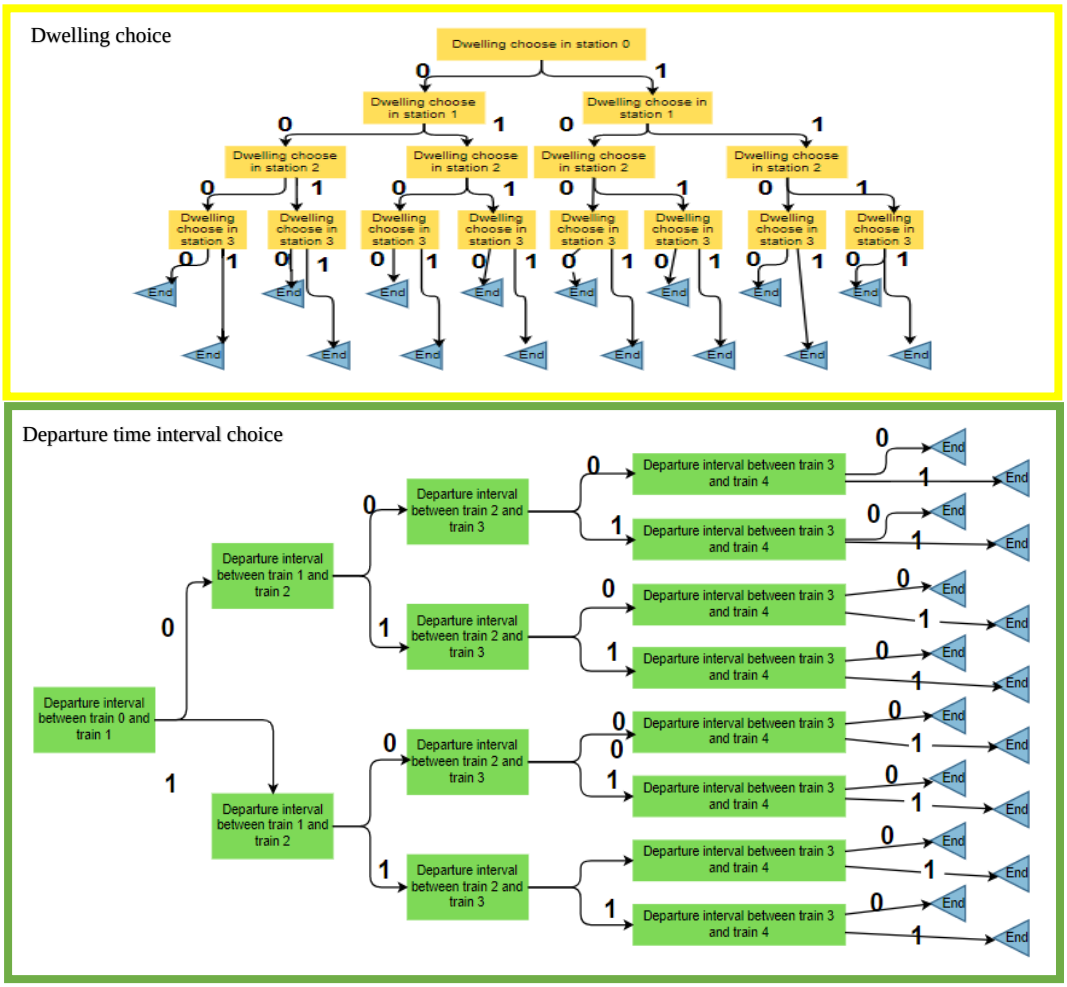


Figure 3-4 Decision tree representing the scheduling process

Thus, in decision tree representation, the object is to find the best decision branch to satisfy dynamic passenger demand. Besides that, in a real-time optimisation process, based on Equation 3-36, the detection time of the real-time variation needs to be compared with the generated timetable to indicate whether the scheduled strategy is adjustable. In this example, the variation is assumed to be detected at time 20. As listed in Tables 3-11 and 3-12, the arrival and departure times in the red cells are fixed, because the detection

time is later than them. Trains have already been dispatched following the old timetable earlier than the detection time.

$a_{i,s}$	i \ s	0	1	2	3
	0	0	3	7	10
	1	4	8	12	16
	2	11	15	20	22
	3	18	21	26	30
	4	22	25	29	32

Table 3-11 Fixed and adjustable arrival times in real-time optimisation

$d_{i,s}$	i \ s	0	1	2	3
	0	1	4	8	12
	1	6	9	14	17
	2	13	17	21	23
	3	19	23	28	32
	4	23	26	30	33

Table 3-12 Fixed and adjustable departure times in real-time optimisation

Conversely, the scheduled arrival and departure times in the blue cells are later than the detection time which means the trains still have not been dispatched based on the scheduled time, so these arrival and departure times are adjustable. Corresponding these times to the decision array, as shown in Table 3-13, only the dwelling time and departure time choices in the cells outlined in blue can be adjusted.

$le_{0,0}$	$le_{0,1}$	$le_{0,2}$	$le_{0,3}$	$ld_{1,0}$	$le_{1,0}$	$le_{1,1}$	$le_{1,2}$	$le_{1,3}$	$ld_{2,1}$	$le_{2,0}$	$le_{2,1}$	$le_{2,2}$	$le_{2,3}$
0	0	0	1	0	1	0	1	0	1	1	1	0	0
$ld_{3,2}$	$le_{3,0}$	$le_{3,1}$	$le_{3,2}$	$le_{3,3}$	$ld_{4,3}$	$le_{4,0}$	$le_{4,1}$	$le_{4,2}$	$le_{4,3}$				
1	0	1	1	1	0	0	0	0	0				

Table 3-13 Adjustable dwelling and departure time choices

For the decision tree in this example, only the dwelling time branches of train 2 at stations 2 and 3, train 3 at stations 1, 2 and 3, and the departure time interval branch and all the dwelling time branches of train 4 can be modified. Also, for the objective function, the number of decision variables is changed from 24 to 10; all the fixed choices are turned from variables into parameters in real-time optimisation.

From the example above, we find that for dynamic passenger demand scheduling, as the number of trains in a time horizon increases, the number of possible dwelling and departure decisions will rise sharply. Even a small example with five trains and five stations has a huge decision tree. This leads the POSM problem to be an NP-hard problem, which means it is quite hard to find an optimal solution in polynomial time. In addition, for real-time scheduling optimisation, the number of decision variables keeps changing based on the time at which passenger flow variation is detected, which leads to it being a DOP with an uncertain number of variables. Efficient algorithms for solving POSM for both passenger demand scheduling and real-time optimisation are required.

3.5 Filed Study for Application Requirements for Real life Metro Operation

In the previous sections, the formulation of dynamic passenger flow-oriented scheduling and real-time optimisation problems was presented, For the real-life metro application, we conducted field studies in the London Underground Bakerloo Line Operation Department to investigate the essential requirements for real-time metro operation to enhance our methodology for the real life application as shown in Figure 3-5. After the field studies, the following requirements have been proposed.

1. Real-time dynamic operation is not necessary for all-day metro operation; the operation strategy is more suitable for a time horizon with dramatic passenger flow variations, or when some unpredictable passenger flow variations occur, such as the time between peak hours and off-peak hours. In a normal situation, a periodic operation strategy should be applied to improve the system's robustness.

2. Generally, modifying the number of trains in a specific operation time horizon will severely affect the economic operation of the metro system and lead to more operational problems, such as staff redeployment. Thus, for real-time operation, the number of trains should be confirmed before the operation starts.

3. In different operation time horizons, different objectives may be proposed, such as improving trains' loading factor for off-peak hours and decreasing passenger waiting time for peak hours.

4. The computation time must be fast enough to ensure metro operators have enough time to modify the operation strategies.

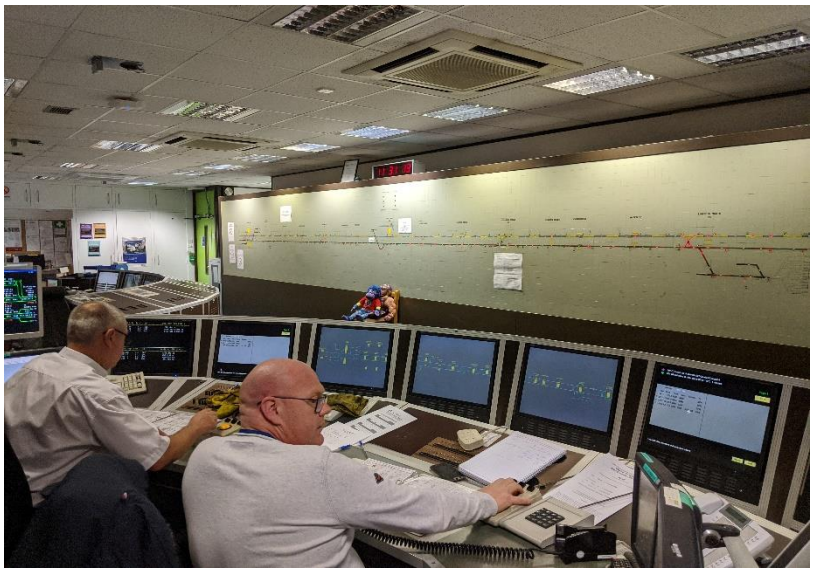


Figure 3-5 Field study in London Underground Bakerloo Line Operation Department

Based on these requirements, a modified comprehensive objective function which integrates multiple objectives will be introduced.

3.6 Passenger Flow-Oriented Comprehensive Objectives Function

During the field studies in the London Underground Bakerloo Line Operation Department, after introducing the base level POSM scheduling method, the metro operators of the London Underground mainly proposed three requirements for the operational strategies based on three typical different operational scenarios.

(1) An operational strategy to balance passenger waiting time and passenger travelling time in peak hours

For a metro system, the peak hours are always in the weekday mornings and evenings, with a large number of passengers on the way to work and home. In these time horizons,

because of the huge passenger flow, satisfying passenger requirements becomes the most important objective. In peak hours, passengers intend to wait a short time in the stations and arrive at their destinations as early as possible; compared with the other situation, more trains will be dispatched in these periods. In general, with a longer dwelling time in a station, the passenger waiting time in a specific station will be decreased; however, the passenger waiting time in the other stations and the boarded passenger travelling time will be increased, and the arrival time of the next train will also be impacted. Thus, an operation strategy which balances the waiting time and travelling time for all passengers in this time horizon is necessary.

(2) An operation strategy to balance passenger waiting time and train full load rate in off-peak hours

Based on the field study, at weekends or in off-peak hours, such as weekday afternoons, the number of passengers is significantly less than the peak hour passenger flow. In real-life operation, we can always observe the scenario that only a few passengers are waiting at stations and most cars of a train are still empty, but the trains still depart at short intervals. Normally, passengers in these off-peak hours are more casual about the waiting and travelling time. With the original short dwelling time and departure interval, though, the passengers' waiting time and travelling time can be decreased; the trains' full load rate will also be decreased which impacts the system's efficiency. In this situation, an operation strategy is needed which balances passenger waiting time and train full load rate, to satisfy passenger demand and metro system efficiency at the same time.

(3) A faster operation strategy to take all passengers to their destination in the time horizon before the closing time

A faster operation strategy which can take all passengers to their destination is helpful near to the closing time of the metro system. Based on our field study, for the London Underground, sometimes because only a few passengers arrive before the closing time, trains will run with empty cars and still keep fixed dwelling times at empty stations, which significantly impacts the system efficiency. Thus, an operation strategy which can take all passengers in the system's closing time horizon to their destination in the shortest time is helpful.

Besides the three requirements above, in real-life operation, the number of trains in the operation time horizon is normally fixed. Modifying the number of trains will cause invalid scheduling, staff redeployment and other operational problems. Based on the experience of the operators from the London Bakerloo Line, they will only change the number of trains in extreme circumstances. Thus, as introduced above, in this research, the number of trains will not be modified in the real-time operation.

According to the objectives proposed above in three different scenarios, in the research time horizon, the modified objective function can be divided into four parts and described by the proposed notations in the beginning of this chapter:

Passenger waiting time (PWT): An evaluation object needs to be minimised based on requirements 1, 2 and 3. PWT includes the waiting time of newly arrived passengers and that of passengers left by the last train, as shown in Equation 3-42.

$$\mathbf{PWT} = \sum_{i=0}^{I}\sum_{s=0}^{S-1}[p_{i,s}^{left}(d_{i,s} - d_{i-1,s}) + \sum_{t\,=\,d_{i-1,s}+1}^{d_{i,s}} \lambda_s(t)(d_{i,s} - t + 1)] \quad \mathbf{Equation\ 3-42}$$

Passenger travelling time (PTT): An evaluation object needs to be minimised based on requirement 1. PTT includes the dwelling time and running time for all the passengers on the trains as shown in Equation 3-43.

$$\mathbf{PTT} = \sum_{i=0}^{I}\sum_{s=0}^{S-1}[(p_{i,s}^{train} - p_{i,s}^{alight})(d_{i,s} - a_{i,s}) + p_{i,s+1}^{train}(a_{i,s+1} - d_{i,s})] \quad \mathbf{Equation\ 3-43}$$

Train full load rate (TLR): An evaluation object needs to be maximised based on requirement 2. As introduced in the previous sections, TLR is the ratio between the passengers on the train and the train's maximum capacity, as shown in Equation 3-44.

$$\mathbf{TLR} = \sum_{i=0}^{I}\sum_{s=1}^{S}\left[\frac{p_{i,s}^{train}}{c_{i,max}}\right] \qquad \mathbf{Equation\ 3-44}$$

End left passengers (ELP): A punishment factor needs to be minimised based on requirements 1 and 2, and a constraint factor needs to equal zero based on requirement 3. ELP is the number of passengers left by the system at the end of the research time horizon. In a normal situation, passengers will keep entering the stations, thus some passengers will always be left at the end of a research time horizon except the system's ending time horizon. These passengers left behind are usually regarded as a punishment factor as it will be allocated to next time horizon as the increment of passenger entry rate. However, at the ending time, the system needs to take all passengers to their destination, so it is regarded as a constraint. ELP is equal to the number of passengers left by the last train and the passengers arriving after the last train departs in the specific time horizon, as shown in Equation 3-45.

$$\mathbf{ELP} = p_{I,S}^{left} + \sum_{t\,=\,d_{I,S}+1}^{T}\lambda_s(t)(T - t + 1) \qquad \mathbf{Equation\ 3-45}$$

As for the number of trains, the modification is done with strict limitations and the optimisation can only be processed based on historical data before real-time operation.

Integrating the four parts together, a comprehensive objectives function which engages to improve the service quality according to the three different requirements can be proposed, as shown in Equation 3-46.

$$\mathbf{CSQI} = \boldsymbol{w_{t1}}\mathbf{PWT} + \boldsymbol{w_{t2}}\boldsymbol{r_1}(\mathbf{t})\mathbf{PTT} + \boldsymbol{w_{t3}}\boldsymbol{r_2}(\mathbf{t})\frac{\mathbf{1}}{\mathbf{TLR}} + \boldsymbol{w_{t4}}\boldsymbol{r_3}(\mathbf{t})\mathbf{ELP} \qquad \textbf{Equation } 3-46$$

where **CSQI** is the comprehensive service quality index which integrates the three requirements at different times proposed above; as it needs to be minimised in the operation, we use the reciprocal of TLR. $\boldsymbol{w_{t1}}$ to $\boldsymbol{w_{t4}}$ are the weighting parameters of different requirements; please note, these weightings are designed based on specific cases. According to different generalised journey times and system norms, the weighting factor could be different. In real-life operation, operators can choose suitable weightings based on different system requirements..

Only PWT is the essential object in the three requirements. Three switching functions $\boldsymbol{r_1}(\mathbf{t})$ to $\boldsymbol{r_3}(\mathbf{t})$ have been added to the other objects; these functions can be presented as Equations 3-47 to 3-49.

$$\boldsymbol{r_1}(\mathbf{t}) = \begin{cases} 0 & \mathrm{t} \notin \text{peak hours} \\ 1 & \mathrm{t} \in \text{peak hours} \end{cases} \qquad \textbf{Equation } 3-47$$

$$\boldsymbol{r_2}(\mathbf{t}) = \begin{cases} 0 & \mathrm{t} \notin \text{off peak hours} \\ 1 & \mathrm{t} \in \text{off peak hours} \end{cases} \qquad \textbf{Equation } 3-48$$

$$\boldsymbol{r_3}(\mathbf{t}) = \begin{cases} 1 & \mathrm{t} \notin \text{system ending time} \\ +\infty & \mathrm{t} \in \text{system ending time} \end{cases} \qquad \textbf{Equation } 3-49$$

where $\mathbf{t}$ is the time in the research time horizon. As the multiple objective function aims to minimise CSQI, $\boldsymbol{r_1(t)}$ and $\boldsymbol{r_2(t)}$ only switch to 1 when PWT and TLR need to be counted, which is based on the operation situations in real life. $\boldsymbol{r_3(t)}$ equals 1 in the normal situation which adds the passengers left behind to the final result as a punishment, but at the system ending time, because all passengers need to be taken, $\boldsymbol{r_3(t)}$ equals $+\infty$; if there are passengers left, the result of the objective function will be invalid. In addition, all the constraints for the real-time operation in previous section should also be met.

3.7 Conclusions

It is important to build an applicable mathematical model to better understand passenger flow-oriented metro scheduling and real-time optimisation problems. In this chapter, first, the concepts and theory of mathematical programming were introduced. Then, the author explained the formulation of train scheduling problems with dynamic passenger demand based on mathematical programming technologies; a nonlinear mathematical model named POSM was proposed, and this model was also derived to solve a real-time scheduling optimisation problem based on the variation of dynamic passenger demand. The example of metro scheduling and real-time optimisation based on dynamic passenger flow demonstrates the application of the POSM model. At last, we also extended the POSM to a comprehensive model based on the requirements from the field study in London Underground.

In the next chapter, the method for solving POSM is proposed. An innovative algorithm based on a genetic algorithm, named GA_POSM, is introduced and a macroscopic metro simulator is presented. Before being implemented to real-time passenger flow oriented

operation, the validity of POSM and, the performance of the algorithm and simulator is evaluated based on some small case studies. After these case studies, we also improve the solving method with decision tree algorithm.

Chapter 4. An Innovative Algorithm, GA_POSM, for Solving POSM Problems

From the mathematical model of POSM proposed in detail in the previous chapter, we can see that the POSM problem is a nonlinear integer dynamic programming problem, which has dwelling and departure interval time as decision variables. Before we implement the comprehensive model to the system-level application, we need to validate the basic POSM first. If we consider the real-time optimisation function of POSM, it changes to a dynamic programming problem with uncertain variables. It is challenging to find an efficient algorithm to solve the proposed POSM problems. Thus, in this chapter, an innovative algorithm named GA_POSM is presented. GA_POSM is derived from a general genetic algorithm, and modified to be an effective tool for solving the POSM problem for both passenger flow-oriented scheduling and real-time scheduling optimisation. In this chapter, the general concepts and definition of genetic algorithms are introduced. Then a macroscopic metro simulator will be introduced and integrated with the modified genetic algorithm, and the modifications and improvements of GA_POSM are described in detail. To validate the efficiency of the proposed model and algorithm, the performance of GA_POSM is evaluated with a case study based on the Beijing Metro Line 19; also, the evaluation results are analysed. At last, based on the requirements from the real life field study, we also derive the proposed method with decision tree algorithm for boarder application according to different system's infrastructures.

4.1 Innovative Algorithm GA_POSM and Metro Simulator

4.1.1 New Algorithm Requirement Consideration

The basic POSM models presented in the previous chapter can be divided into two problems. For dynamic passenger flow-oriented scheduling, the problem is a nonlinear integer programming problem with a time-varying dynamic parameter $\lambda_s(t)$; for real-time scheduling optimisation, the problem is also a nonlinear integer programming problem with dynamic parameters but because of the variation of detection time t_D, the number of decision variables in the objective function is uncertain. Both of these problems optimise the dwelling time and departure time interval according to passenger flow for each train in the research time horizon.

For the first problem, generally, all of the possibilities of the dwelling time and departure time interval setting decisions must be enumerated. In terms of every possible dwelling $e_{i,s}$ and departure decision $hd_{i,i-1}$, the results from the objective function are compared to find the best decision. For the second problem, the objective function is modified whenever the number of decision variables changes; all the unmodifiable variables are replaced with fixed parameters and all the parameters changing before the variation happens are recorded; then decisions are enumerated for the remaining variables. Obviously, neither of the methods mentioned above can solve the problems easily. First, the computational time will increase exponentially with the size of the problem. It is not possible to apply this method to real-time operation, such as real-time scheduling and optimisation. Second, there is a time-varying dynamic parameter $\lambda_s(t)$ in the objective function which will keep changing with time. Third, in a real-time optimisation problem, the number of decision variables is uncertain, and it is difficult to always keep generating new objective functions.

As seen from the POSM, the problem has the following features:

1. The search space of variables is large and increases significantly when the size of the problem increases.

2. In the objective function, a parameter keeps varying with time with increases the difficulty of solving.

3. In real-time optimisation, the number of decision variables changes with different detection times; it is hard for traditional heuristic algorithms to deal with unfixed decision variables.

4. As the problem refers to real-time optimisation applications, the objective function must be solved in a short time limit.

5. Nearly optimised solutions are acceptable.

Considering the features mentioned above, the author compares advantages and disadvantages of some general algorithms, as shown in Table 4-1.

Algorithm	**Advantage**	**Disadvantage**
Exhaustive algorithm	• It is simple to be understanded and implemented • It is the most straight forward approach to find a solution for optimisation • It can guarantee the correct optimised solution will be found	• It is computationally intensive and has exponential time complexity • It can consume significant resources • It is not suitable for dynamic problems, since it requires a complete re-run each time that problem changes
Gradient Descent algorithm	• It is simple to be implemented • It can handle large-scale convex problem • It is computationally efficient	• It can get stuck in local optimisation • It can converge slowly for problems with high dimensional parameters • It is sensitive to the initialisation of the parameters
Greedy algorithm	• It is fast, since it does not require searching the entire solution space	• It may only find a locally optimal solution

	• It is simple to be implemented and does not require complex mathematical algorithm • It generates good approximate solutions	• It is not suitable for problems that require backtracking and changing previous decisions, since it makes decisions based on the current state only • It does not guarantee an optimal solution in some situations
Genetic algorithm	• It can handle large and complex problems, as non-linear and multi-modal problems. • It can handle noise, uncertainty, and irrelevant information in the data • It's 0-1 code is appropriate to represent the dwelling and departure choice in this problem • It can record the optimised result from each generation.	• It may converge prematurely to a sub-optimal solution • It's performance can be impacted by the choice of hyperparameters • It requires a good understanding of the problem domain and the properties of the solution space
Simulated annealing	• It is capable to find the global optimisation • It is a robust algorithm which can avoid getting stuck in local optimisation • It can provide good approximate solutions	• It can converge slowly for problems with large numbers of variables • It requires the setting of hyperparameters which can impact the quality of solutions • It is computationally intensive

Table 4-1 Advantages and disadvantages of considered algorithm

Compared above algorithms, the exhaustive algorithm is not appropriate for this problem since it has exponential time complexity; as the size of the problem increase, the exhaustive algorithm will quickly become infeasible. Also, compared with discrete problems, the gradient descent algorithm works better with well-behaved objective functions with continuous variables and requires additional techniques to handle constraints. Compared with the genetic algorithm, the greedy algorithm is easier to get stuck in local optimisation, and constraints and penalties are more difficult to be defined by the greedy algorithm; moreover, as this problem needs backtracking and changing previous decisions, the greedy algorithm is also not appropriate. In this problem, compared to simulated annealing, the gene sequence in the genetic algorithm is more

suitable to represent the dwelling and departure interval choices for a metro system, especially when it is based on peak and off-peak strategy, and with time constraints the gene sequence is also easier to be recorded and modified to update the real-time operation.

Thus, the author proposes an innovation algorithm named GA_POSM which is based on a genetic algorithm (GA) and integrated with a macroscopic passenger–metro interaction simulator to solve POSM problems. In the next section, the general genetic algorithm is introduced, the details of the modification for the algorithm GA_POSM are described and the algorithms to simulate passengers and metro systems in the simulator are expressed. GA_POSM will also be evaluated based on the Beijing Metro Line 19 at the end of this chapter.

4.1.2 Introduction to Genetic Algorithms

Direct search methods are widely used for solving optimisation problems that do not require information about the objective function's gradient. Unlike most traditional optimisation methods which search for an optimal point based on gradient, direct search methods search optimal solution points around the current point and look for a better value of the objective function than the value at the current point. Based on this feature, a direct search method can be applied to nonlinear, non-differential or even discontinuous objective functions. The general principle of direct search methods is to generate variations of the solution vectors and select better solutions from the variations. Some selection methods, such as greedy criteria, make the algorithms converge sufficiently fast; however, this can lead to the algorithms being trapped in a local optimum. Thus some other methods apply probabilities to solutions for the selection process to forbid a local optimum.

Genetic algorithms are a family of direct search methods which mimic the natural evolutionary process, so they are also classified as evolutionary algorithms. This type of algorithm was proposed as early as 1954 based on computer simulation of the evolution research of Nils Aall Barricelli. As computer evolution simulation became more popular, the genetic algorithm was described in detail by Fraser and Burnell in the 1970s; Fraser's simulations included all the essential elements of the modern genetic algorithm. This method has been applied to solve complex engineering problems and became a widely recognised optimisation method from Ingo Rechenberg's research in the 1970s.

As an evolutionary algorithm, a genetic algorithm also includes three critical operations: selection, crossover and mutation. Consider the problem presented in Equation 3-14, that is a typical nonlinear programming problem. Generally, for a genetic algorithm we use binary numbers to encode the decision variables in the objective function, which means the decision variables will be transformed from decimal form to binary form (also called gene fragments) as shown in Equation 4-1; this encoding method makes genetic algorithms suitable for dealing with discrete integer programming problems. After transformation, the number of the binary decision variables may be increased. We assume the number of decision variables for this problem is *T*, and the number of binary decision variables after encoding is *D*.

$$\{x_h | h = 1,2,3, \dots, T\} \rightarrow \{b_k | k = 1,2,3, \dots, D\} \qquad \textbf{Equation 4-1}$$

We can use a *D*-dimensional parameter $B = (b_1, b_2, \dots, b_D)$ (also known as a gene sequence) to represent the vector of the binary decision variables. The optimisation problem is to adjust this parameter to maximise or minimise the objective function and meet the constraint conditions at the same time.

As an evolutionary algorithm, we also need to define the maximum generation *G*, and the population size *PS* in every generation. Each population represents an individual *D*-dimensional parameter for the vector of decision variables. Thus the *D*-dimensional parameters contained in each generation can be written as Equation 4-2:

$$\{\mathbf{B}_{l,g} | l = 1,2,3,\ldots,PS \quad g \in G\} \qquad \textbf{Equation 4-2}$$

Normally, an initial population will be chosen randomly if there is no specific requirement for the problem. In some research, because the decision variables for most problems have an upper and lower bound, to forbid a local optimum, the initial population is generated based on a uniform distribution or normal distribution between the upper and lower bounds, in order to maximise the variety of the decision variables' vectors in the searching spaces. After the initial population, the genetic algorithm will keep choosing better decision vectors and generating a new population based on chosen decisions for the next generation; this process will only be stopped when satisfactory solutions have been found or the maximum number of generations has been counted. The general flow chart of the genetic algorithm method is shown in Figure 4-1, Selection, crossover and mutation are three important steps of genetic algorithms.

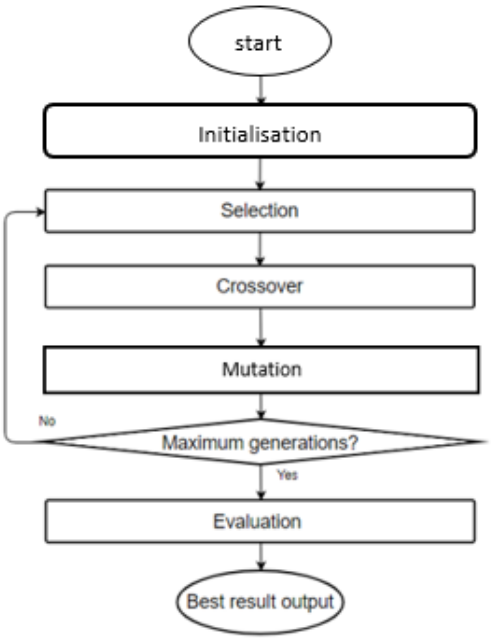


Figure 4-1 Flow chart of classic genetic algorithm method

(1) Selection

A genetic algorithm is a large loop process; after the initial population is input, the selection processing will begin. In this operation, all the trial decision vectors from $\mathbf{B}_{l,g}$ will be evaluated and the decision vectors B_p leading to better solutions will be selected (which indicates the survival of the fittest in biological evolution) to generate parent vectors to construct a new *D*-dimensional parameter $\mathbf{P}_{l,g+1}$ as shown in Equation 4-3. The best decision variable and solution for this generation will be recorded.

$$\mathbf{P}_{l,g+1} = \{B_{p_1}, B_{p_2}, \ldots, B_{p_l}\} \qquad \textbf{Equation 4-3}$$

(2) Crossover

As with biological evolution, after better parent decision vectors are selected, they will cross over and change a part of their vectors $B_{p_1} = (b_{1,p_1}, b_{2,p_1}, \ldots, b_{D,p_1})$, $B_{p_2} = (b_{1,p_2}, b_{2,p_2}, \ldots, b_{D,p_2})$ to generate two child decision vectors $B_{c_1} =$

$(b_{1,c_1}, b_{2,c_1}, \ldots, b_{D,c_1})$ and $B_{c_2} = (b_{1,c_2}, b_{2,c_2}, \ldots, b_{D,c_2})$ for the next generation. As the decision variables have already been binary-encoded, this process can be shown as Equation 4-4:

$$b_{k,c_1} = \begin{cases} b_{k,p_1} & 0 < k < r \\ b_{k,p_2} & r \leq k \leq D \end{cases} \quad \& \quad b_{k,c_2} = \begin{cases} b_{k,p_1} & r \leq k \leq D \\ b_{k,p_2} & 0 < k < r \end{cases} \qquad \textbf{Equation 4-4}$$

where *r* is a random number between 0 and the vector length ***D***; after crossover, the first child B_{c_1} is inherited from the first parts of the first parent B_{p_1} until the variable b_{r,c_1}; the remaining parts after *r* are inherited from the second parent B_{p_2}. Contrarily, the second child B_{c_1} inherits the opposite parts. This process emulates the gene exchange in genetics. After crossover, the same number of child decision vectors with population vectors can be generated. A new *D*-dimensional parameter $\mathbf{C}_{l,g+1}$ is shown in Equation 4-5.

$$\mathbf{C}_{l,g+1} = \{B_{c_1}, B_{c_2}, \ldots, B_{c_l}\} \qquad \textbf{Equation 4-5}$$

(3) Mutation

Finally, all the decision vectors in the child *D*-dimensional parameter $\mathbf{C}_{l,g+1}$ will mutate to generate decision vectors for the next generation by using the strategy shown in Equation 4-6.

$$b_{k,g+1} = \begin{cases} 1 - b_{k,c} & m_1 \leq k \leq m_2 \\ b_{k,c} & k < m_1, k > m_2 \end{cases} \quad m_r = 1$$

$$b_{k,g+1} = b_{k,c} \quad m_r = 0 \qquad \textbf{Equation 4-6}$$

where m_1 and m_2 are two mutation length indices which indicate the mutated binary decision variables (the mutated gene fragments in a gene sequence); normally, these two indices are generated according to a given mutation length. m_r is a mutation index equals

to 0 or 1 based on a given mutation rate; mutation only happens when the mutation index is equal to 1. Because binary decision variables are used in a GA, when mutation happens, we can use 1 minus the original variable to mutate it to the opposite value. After the mutation process, the next generation's collection of *D*-dimensional parameters $\mathbf{B}_{l,g+1}$ can be generated.

The genetic algorithm will continue the selection, crossover and mutation loop until the maximum generation. The best decision variables in each generation will be recorded and compared; finally it will evaluate the best result B_{best} and transform the result to decimal form to output.

4.1.3 Introduction to Macroscopic Metro Simulators

The previous section introduced the basic concepts and operation process of a genetic algorithm. However, because of the features of POSM presented in Chapter 3.2, the classic genetic algorithm introduced in the previous section cannot be operated directly for solving the POSM problem. First, as the decision variables of our objective function are the choices of the dwelling time $le_{i,s}$ at each station and the choices of departure time $ld_{i,i-1}$ between two trains, we need to consider the transformation between these choices and binary decision variables and train timetables. Second, we need to consider how to represent the dynamic parameter $\lambda_s(t)$ which represents passenger flow rate based on a time-variant OD matrix. Third, we need to integrate decreasing passenger waiting times and increasing train capacities in the object; the evaluation and selection processes in the genetic algorithm also need to be modified with the objective function. Last but not least, in real-time operation, the number of decision variables keeps changing with the detection time, which means the parameters related to passenger flow in the objective function and

the length of binary decision variable vectors (gene sequences) B also need to be modified according to the detection time. With comprehensive consideration, a multifunctional macroscopic metro simulator has been designed.

The structure of the macroscopic simulator is shown in Figure 4-2. There are four sections in the simulator: the metro infrastructure section, passenger flow section, timetable section and passenger–train interaction section, which are introduced separately below. Based on this simulator, network passenger flow distribution can be tracked at any specific time in the operation time horizon.

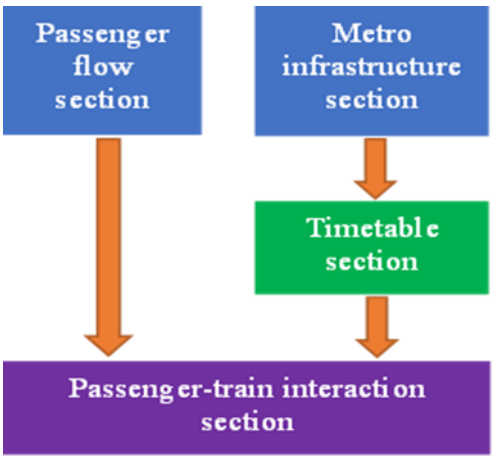


Figure 4-2 Structure of macroscopic metro simulator

(1) Metro infrastructure and passenger flow sections

These sections mainly focus on data input and a description of passenger flow and the metro system. For passenger flow, the most critical data set is the entry rate OD matrix which describes passenger journeys between specific stations; these matrices will simulate the dynamic passenger arrival rate $\lambda_s(t)$ in the objective function. For metro infrastructure, the input data mainly focus on the constraint conditions based on the

mathematical model POSM, the planned running time $r'_{i,(s-1,s)}$, optional departure interval time H, optional dwelling time E and maximum train capacity $c_{i,max}$.

(2) Timetable section

Based on different optional departure intervals and dwell times settled in the metro system section, there are many combinations for the choices of interval and dwell time. The timetable section generates the arrival times and departure times for all trains by assembling the combinations from different trains. To ensure the safety constraints, i.e. the minimum headway, some of the arrival times and departure times need to be modified according to the system constraints shown in Chapter 3. After modification, an extensive timetable for the system can be established. The algorithm for the timetable section is shown below:

Algorithm 1: Timetable generator	
Input: Departure interval $ld_{i,i-1}$ and dwell choice $le_{i,s}$ for trains in the time horizon	
Output: Arrival and departure time based on departure interval and dwell	
1	for train i $\in$ $set\ of\ trains\ I$ do
2	for station $s \in set\ of\ stations\ S$ do
3	if i = 0 and s = 0
4	$a_{i,s} = 0$
5	$d_{i,s} = a_{i,s} + E(le_{i,s})$
6	else if i = 0 and s != 0
7	$a_{i,s} = d_{i,s-1} + r'_{i,(s-1,s)}$
8	$d_{i,s} = a_{i,s} + E(le_{i,s})$
9	else if i != 0 and s = 0
10	$a_{i,s} = d_{i-1,s} + H(ld_{i,i-1})$
11	$d_{i,s} = a_{i,s} + E(le_{i,s})$
12	else if $d_{i-1,s} - d_{i,s-1} - r'_{i,(s-1,s)} \geq h_{min}$
13	$a_{i,s} = d_{i,s-1} + r'_{i,(s-1,s)}$
14	$d_{i,s} = a_{i,s} + E(le_{i,s})$
15	else
16	$a_{i,s} = d_{i,s-1} + h_{min}$
17	$d_{i,s} = a_{i,s} + E(le_{i,s})$
18	return $a_{i,s}$, $d_{i,s}$
19	end if

20	end for
21	end for

(3) Passenger–train interaction section

This part mainly concentrates on the interaction between the metro system and passenger flow based on the objective function, thus this section should calculate the waiting time ω_t of all passengers and the train full loading rate $cr_{i,s}$. Trains are dispatched based on the timetables generated in the timetable section, and passengers arrive at each station according to the passenger flow entry rate OD matrix generated by the passenger flow section. With trains departing and stopping based on the chosen departure interval and dwell time, passengers get off firstly to make more space for passengers preparing to board. The maximum train capacity $c_{i,max}$ should be defined by the metro infrastructure section; to be fair, if there is enough room on the train, passengers will board in sequence according to different destinations until the train reaches maximum capacity. Based on Equations 3-28 to 3-32, the remaining passengers will keep waiting for the next train in the station. With the implementation of platform screen doors and safety personnel on platforms, we do not need to consider impacts of alighting and boarding for the journey, and assume the dwelling time is enough for passengers to fill an empty train. The algorithm for the passenger–train interaction simulator is shown below:

Algorithm 2: Passenger–train interaction simulator	
Input: Passenger data from passenger flow section, system data from metro infrastructure section, departure and arrival time from timetable section	
Output: Total passenger waiting time WAT and total reciprocal CPR of train capacity occupation ratio $cr_{i,s}$ at each station	
1	WAT = 0
	CPR = 0
2	for time t ≤ operation time T do, t++
3	for station $s \in$ set of stations S do

4	$\omega_t = \omega_t$ + the waiting passengers $p_{i,s}^{station}$
5	end for
6	for train i $\in$ $set\ of\ trains\ I$ do
7	for station s $\in$ set of stations S do
8	if arrival time $a_{i,s}$ = t
9	for alighting passengers $p_{i,s}^{alight} > 0$ do
10	passengers on the train $p_{i,s}^{train}$--
11	train's remaining capacity $c_{i,s}$++
12	end for
13	end if
14	if departure time $d_{i,s}$ = t
15	left passengers $p_{i,s}^{left}$ = waiting passengers $p_{i,s}^{wait}$
16	for waiting passengers $p_{i,s}^{wait} > 0$ && train′s capacity $c_{i,s} > 0$ do
17	passengers on the train $p_{i,s+1}^{train}$++
18	left passengers $p_{i,s}^{left}$--
19	train's capacity $c_{i,s}$--
20	end for
21	CPR = CPR + train's maximum capacity c_{max} /passengers on the train $p_{i,s+1}^{train}$
22	else
23	continue
24	end if
25	end for
26	end for
27	end for
28	return WAT CPR

From the above introduction, the simulator can simulate macroscopic passenger behaviour and metro system operation in the research time horizon; the distribution of passenger flow can be observed at each specific time point. With this simulator, the generated timetable can even be evaluated by different objective.

4.1.4 Improved Genetic Algorithm GA_POSM for Solving POSM Problems

The macroscopic metro simulator introduced in the last section can be integrated with a classic genetic algorithm to solve the problems in dynamic passenger flow-oriented

scheduling problem and the real-time scheduling optimisation proposed above. In this section, by integration with the simulator, improvement of the genetic algorithm will be explained from following perspectives: transformation between binary decision vectors from the genetic algorithm and available timetables, representation of dynamic passenger flow arrival rate, real-time binary decision vector length modification and avoiding local optimisation.

(1) Transformation between binary decision vectors and available timetables

Because of the complexity of metro systems and special features of genetic algorithm, the binary decision vectors must be transformed to available timetables to be evaluated. Thus, before the selection step of the genetic algorithm, the timetable section of the simulator is operated; the binary decision vector (gene sequences) for each train can be transformed to the choices of departure interval and dwelling time, and the choices will be modified by the timetable algorithm presented above based on the constraints from the objective function, and generates the available timetable. The detailed process is shown in Figure 4-3.

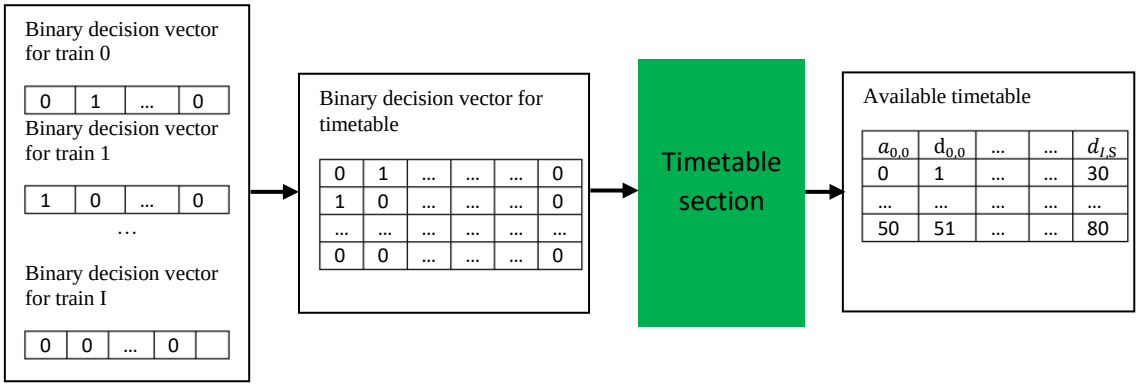


Figure 4-3 Approach to generating available timetable by GA_POSM

(2) Dynamic passenger flow rate representation

As a DOP, a time-varied parameter for passenger arrival rate $\lambda_s(t)$ is in the objective function, which leads to a complicated evaluation process. However, by integrating it with the simulator, the result can be evaluated directly by the passenger–train interaction section. Based on the passenger flow section, the dynamic parameter $\lambda_s(t)$ will be simulated by time-varied OD matrices of passenger flow entry rate. The passenger–train interaction section simulates the interaction between the dynamic passenger flow rate OD matrices and the metro system directly, and generates the evaluation result for the objective function based on the simulation. The approach is shown in Figure 4-4.

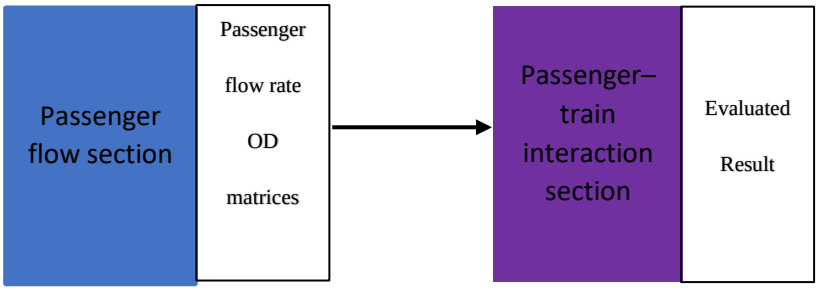


Figure 4-4 Approach to simulating dynamic parameter $\lambda_s(t)$ in the objective function

(3) Real-time modification of binary decision vector length

For a traditional genetic algorithm, the length of the binary decision vector (or gene sequence) is rigid. However, for real-time scheduling optimisation research, with a different detection time of irregular passenger flow variation, some trains have already departed before the detection time; thus the number of operational dwelling times and departure time intervals keeps varying, which leads the length of the binary decision vector to keep changing in real-time optimisation. By integrating the macroscopic simulator with a genetic algorithm, these irregular variations can be recorded in the passenger flow section and the non-optimised timetable can be recorded in the metro infrastructure section. The non-optimised timetable will be compared with the detecting

time and, based on this comparison, a process to modify the length of binary decision vectors can be applied in the genetic algorithm for real-time scheduling research. Then the genetic algorithm will generate new decision vectors and process as normal; the only difference is that, in the selection step, after a new binary decision vector is generated, it will be relinked with the unmodifiable part of old decisions to generate a completed timetable. This process is shown in Figure 4-5.

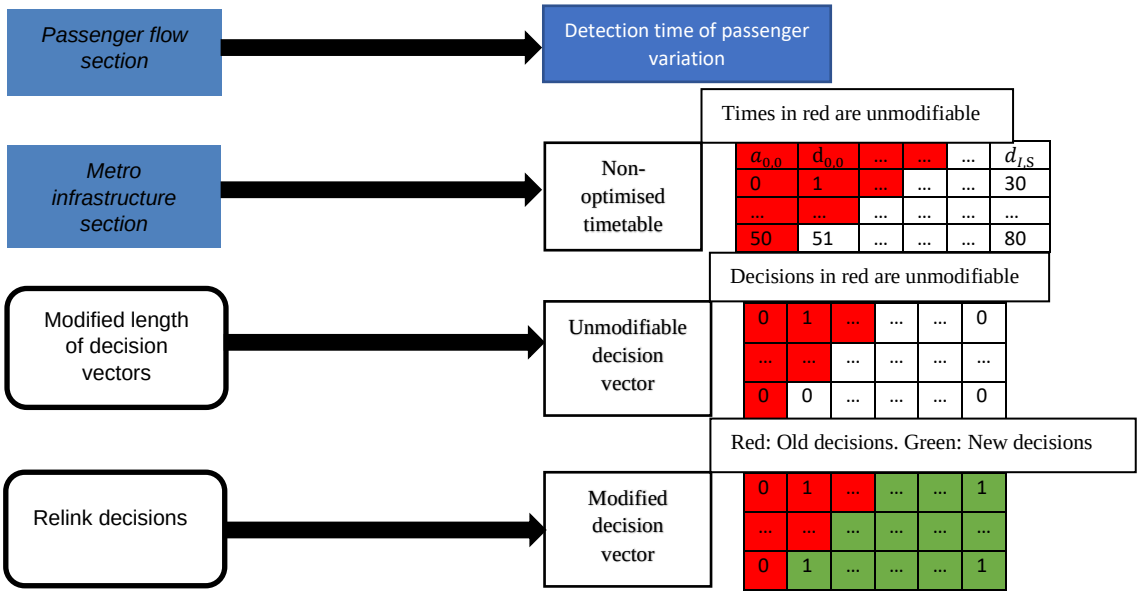


Figure 4-5 Approach to modifying binary decision vectors in real-time optimisation

(4) Approach to avoid local optimisation

Though for this research the nearly optimised solutions are acceptable, and in the real-time optimisation, instead of the best timetable, a new timetable better than the old one is adequate, we still engage to avoid local optimisation caused by modifying operation of the genetic algorithm. A roulette wheel selection algorithm (also known as a fitness proportionate section algorithm) is a method for selecting potentially good solutions to avoid local optimisation (Lipowski and Lipowska 2012).

In a roulette wheel selection algorithm, instead of just selecting the binary decisions with better evaluated results, we assign a score s_d to each possible decision vector d according to its evaluated result, then the total score $\sum_{d=1}^{D} s_d$ will be added and a random number R between 0 and the total score will be generated. Each decision vector's score will contribute a range in the total score; if the random number R is in the range, the decision vector will be selected. Thus, the possibility p_d of a decision vector being selected is

$$p_d = \frac{s_d}{\sum_{d=1}^{D} s_d} \qquad \textbf{Equation 4 – 7}$$

This process can be imagined as a roulette wheel where the whole wheel is the total score, each decision vector is distributed on a different part of the wheel according to its score and the random number R is the indicator on the wheel. When the wheel spins, a binary decision vector with a higher score has a higher possibility of being selected, but it still can be eliminated because the possibility is less than 1; contrarily, a decision vector with a lower score also has a chance of being selected, as shown in Figure 4-6. In a genetic algorithm, some characteristics of the weak decision vectors could prove useful after further crossover or mutation; by applying a roulette wheel selection algorithm, these weak decision vectors also get a chance to be selected, which helps to avoid local optimisation.

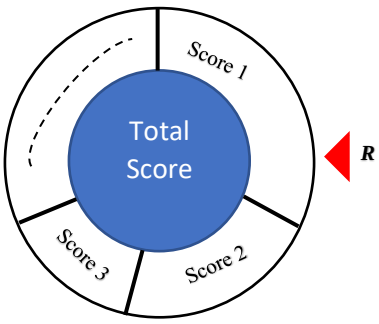


Figure 4-6 Explanation of roulette wheel selection algorithm

After the above steps, the interface between the genetic algorithm and the simulator can be shown by Figure 4-7.

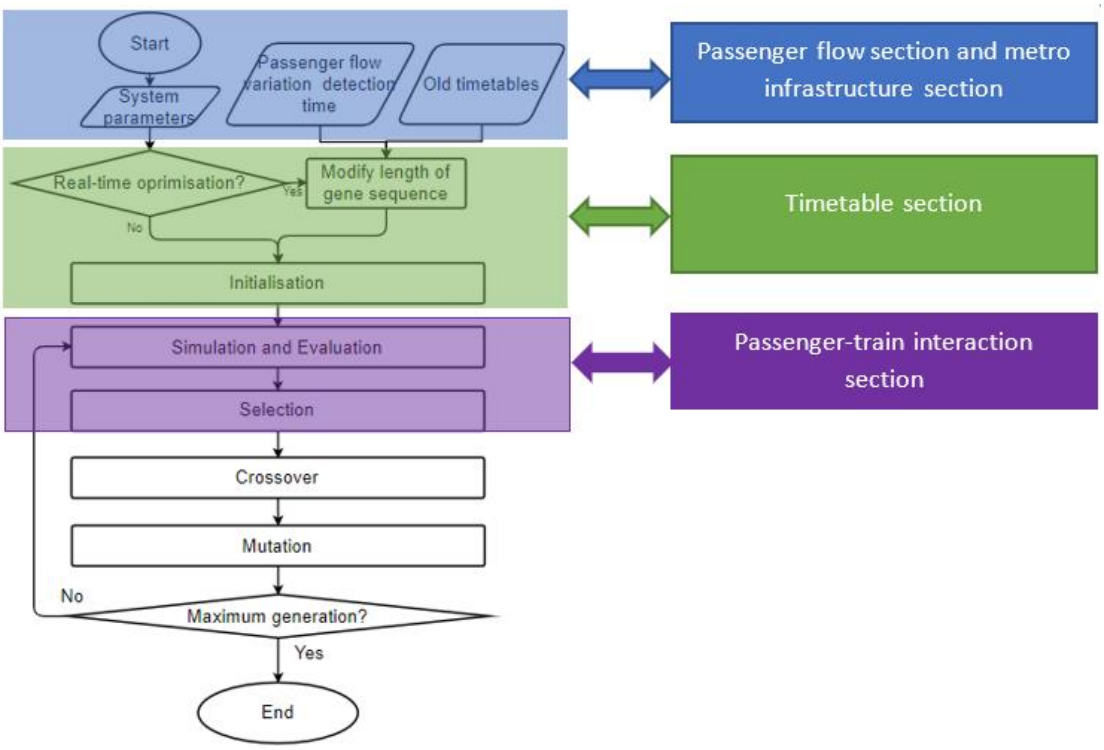


Figure 4-7 Interfaces between GA_POSM and simulator

4.2 Performance Evaluation of GA_POSM

4.2.1 Systematic Approach

In the previous section, the basic concept of the genetic algorithm was introduced, and a modified genetic algorithm, GA_POSM, which is integrated with a macroscopic metro simulator was presented in detail to solve passenger flow-oriented scheduling and real-time optimisation problems. In this section, the performance of GA_POSM is evaluated by a systematic approach described in the following sections. The overview of the systematic approach for this problem is shown in Figure 4-8.

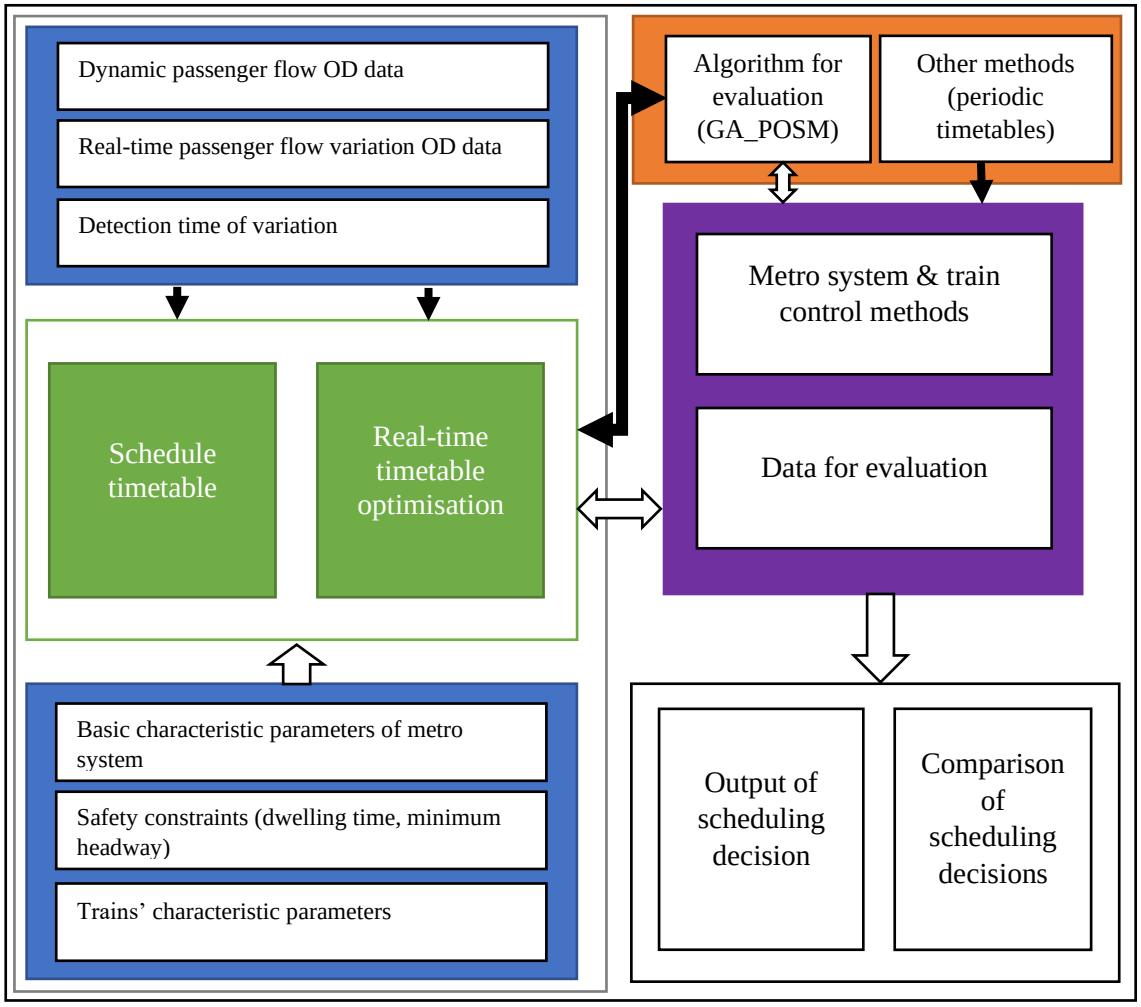


Figure 4-8 Systematic approach for evaluation of GA_POSM

From the figure above, it can be seen that this systematic approach indicates the data inputs and outputs for different parts of the simulator, and presents the interaction between the algorithm and simulator. There are six main parts of this system architecture:

(1) The metro infrastructure section, storing basic metro infrastructure and rolling stock data, including the number of stations, running time between stations, trains' maximum capacity, safety constraints, optional dwelling time and departure interval, and the repository of generated timetables.

(2) The passenger flow section, holding the dynamic passenger flow entry speed OD data for scheduling, real-time passenger variation detection time and varying entry speed of passenger flow.

(3) The timetable section, interacting with the genetic algorithm and metro infrastructure section, transforming the binary decision variables to operational timetables, modifying the length of decision vectors based on the detection time.

(4) The scheduling repository section, storing different scheduling strategies including GA_POSM and other strategies (such as traditional periodical timetables).

(5) The passenger–train interaction section, simulating the metro operation based on different dispatching strategies, outputting the evaluation result for the strategies based on the objective function.

(6) The comparing section, collecting and comparing the results of different scheduling strategies.

All of the parts were programmed in the Java programming language and established as Java files. The data flow between different parts is also shown in Figure 4-8. The main function of the metro infrastructure section and passenger flow section is to prove the infrastructure data of metro system and passenger flow. Different binary decision variables generated in the scheduling simulation section are based on different scheduling strategies. The timetable section generates timetables based on binary variables and inputs the generated timetables to the passenger–train interaction section to simulate and evaluate them; it also modifies the length of binary variables for real-time time optimisation. For GA_POSM, the results from the passenger–train interaction section also

become an input to select the binary vectors for the next generation. Finally, the results of different scheduling strategies are collected and compared in the comparing section.

4.2.2 Case Study

The Beijing Metro Line 19 is used to evaluate the performance of the proposed approach. Though this metro line is still under construction when writing this thesis, new 5G network communication and passenger counting technology from Beijing Infrastructure Investment Co Ltd will be allocated to this metro line. Before passengers enter the platforms, they need to choose their origin and destination from a ticket machine. When they pass the platform gates, the AFC system can collect and generate real-time OD data and EUHT communication units will be set up in all stations to transfer the OD data; this collection process is shown in Figure 4-9. Based on the EUHT-5G technology shown in Figure 4-10, the passenger flow data in stations and trains can be transferred to the metro control centre in real time, which enables analysis of passenger volume for both scheduling and real-time optimisation.

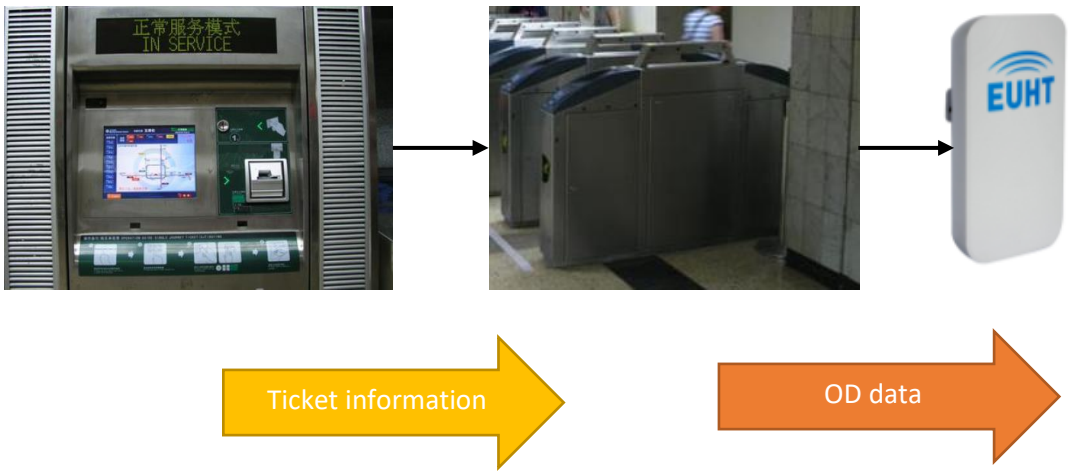


Figure 4-9 Passenger OD data collection process

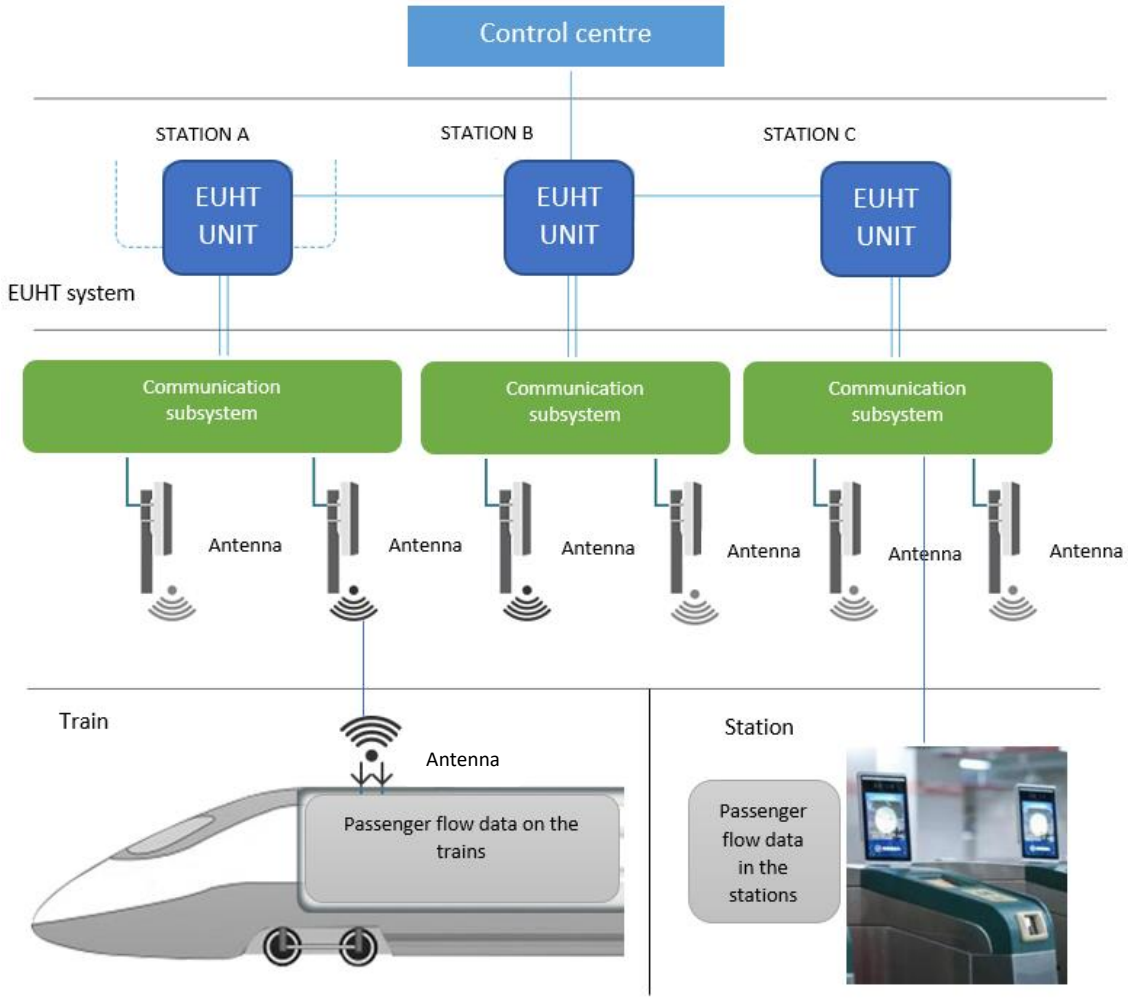


Figure 4-10 EUHT-5G communication technology

The total length of Beijing Metro Line 19 is 22.4 km, with 10 stations which are fully underground; four stations are still under construction. This metro line begins at Mudanyuan station in Haidian District and ends at Xingong station in Fengtai District. As its stations are located through most busy areas in Beijing, it is envisioned to relieve overcrowding and link the Beijing Financial Street commercial area. A geographic map of the metro line is shown in Figure 4-11.

Figure 4-11 Geographic map of Beijing Metro Line 19

This metro line uses eight-car type A rolling stock CCD5034/SFM80 designed by CRRC Changchun and CRRC Qingdao Sifang. The basic characteristics and an image of CCD5034/SFM80 train sets are shown in Table 4-2 and Figure 4-12. Based on the data currently available, the first train will depart at 05:30 and the last train will depart at 23:25 in one day. Based on a periodical scheduling strategy, two departure time intervals are operated, 6 minutes in peak time and 8 minutes in off-peak time. Normally, the dwelling time will be 1 minute at each station; shielded gates and alarms will be used in platforms to avoid operational disturbance caused by rushing passengers. The planned running times between different station are shown in Figure 4-13.

Model	CCD5034/SFM80
Power Collection	1500 V DC
Vehicle Formulation	6M2T （Tc+Mp+M+Mp+M+M+Mp+Tc）
Total Length	185.68 m Tc – 23.58 m Mp, M – 21.88 m
Rated Capacity	2520 passengers
Traction Output	6000 kW
Max Speed	100 km/h

Table 4-2 Configuration of CCD5034/SFM80 trainsets

Figure 4-12 CCD5034 trainset at Mudanyuan station

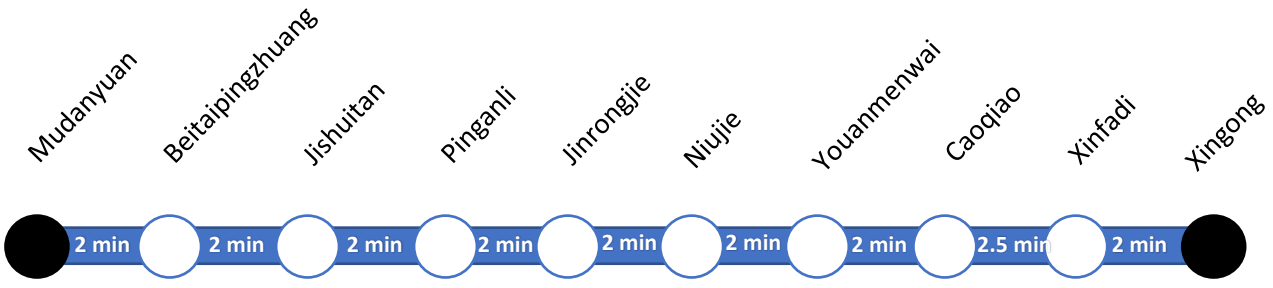


Figure 4-13 Planned running time between stations

4.2.3 Preparation for Case Study

Because the Beijing Metro Line 19 is still under construction, we need to assume and modify some data for both the metro system and passengers. For this case study, a single-direction line from Mudanyuan to Xingong is demonstrated and tested, the 10 stations in the system marked by S0 to S9. In macroscopic terms, the passenger flow variation in every second is hard to detect. And based on our field observation study in the London Euston metro station, generally, a person needs at least 3 seconds to pass through the platform's gate. Thus, for the macroscopic metro operation simulation, we set 30 seconds as a basic time-unit. To ensure the system's safety, the minimum headway between two trains arriving at the same station should be larger than 4 time-units (2 minutes). Based on the data from the last section, the optional departure time intervals between trains are set to 12 time-units (6 minutes) and 16 time-units (8 minutes); the optional dwelling time is set to 1 time-unit (0.5 minutes) and 3 time-units (1.5 minutes). Nowadays, most metro systems in China applied with shield gates (platform screen doors) and safety personnel to ensure passengers and other disturbances will not impact trains' operation, passengers have enough time to embark and disembark by order; compared with old metro systems, the dwelling time and headway can be highly decreased, shown in Figure 4-14. However, the 0.5 minute dwelling time is still very tight in real life, but as a case study based on the simulation environment, we engage in using some extreme situations to implement a better evaluation of the proposed methodology to ensure the validity of the methodology with extreme situations, these extreme situations may not suitable for a real-life operation. Seven trains are applied for this case study; according to the departure, dwelling and running data, the time horizon to cover a seven-train case study is 210 time-units (105 minutes). The modified infrastructure data are shown in Table 4-3.

Figure 4-14 Shield gates (platform screen doors) and a safety personnel

Parameter	***Data***
Basic time-unit	30 seconds
Minimum headway	4 time-units
Number of stations	10 stations (S0 to S9)
Research time horizon	210 time-units (105 minutes)
Train capacity	2520 passengers
Number of trains	7
Optional dwelling time	1 time-units, 3 time-units
Optional departure interval	12 time-units, 16 time-units

Table 4-3 Configuration of CCD5034/SFM80 trainsets

The passenger data in the case studies were mocked up based on early-stage desk research of the metro operator and the pre-designed capacity target of this line considering its connections with other commuting systems. Also, the passenger flow data from the other four typical Beijing metro stations, as shown in Figure 4-15, are also an important reference for the mocked-up passenger flow data; these data can represent the typical

passenger flow distribution in busy and unbusy stations. At busy stations, such as the stations around the starting point, the entry speed of passenger flow is set around 25 persons/minute in peak time and 8 persons/minute in off-peak time. At unbusy stations, such as the stations around the finishing point, the entry speed of passenger flow is set around 8 persons/minute in peak time and 3 persons/minute in off-peak time.

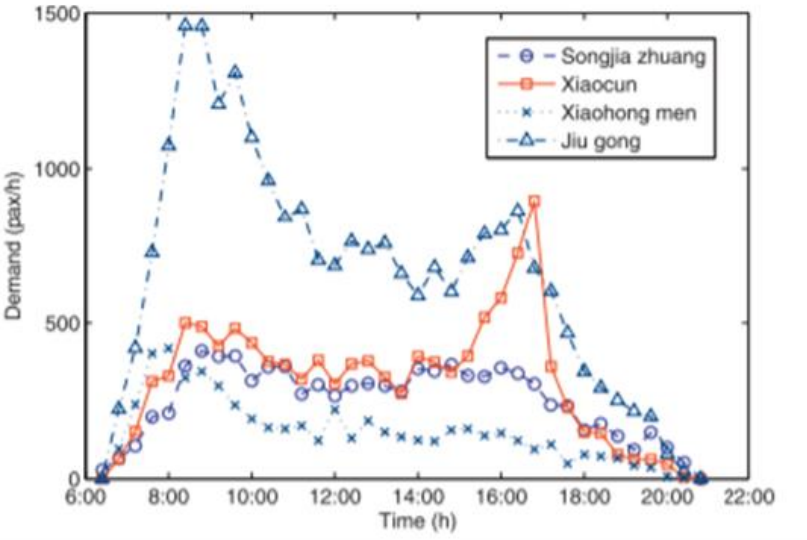


Figure 4-15 Passenger flow data from four typical Beijing metro stations

Based on early-stage research and estimation of the distribution of passengers getting off at different stations, combined with the passenger flow data above, different OD data of passenger flow entry speed data can be generated. Normally, it takes 15 minutes to calculate and transfer the average passenger flow data, thus for dynamic passenger flow research, the dynamic OD speed matrix will keep refreshing in every 15 minutes, this time limitation is mainly linked to the measurement and transmission technologies applied nowadays, and the calculation time is also an important limitation for the data updating resolution since the increase of resolution may lead to more computation time. Without these technical limitations, the passenger flow rate record data could be updated

with a higher frequency (such as per second), but at this stage, enough time is necessary for calculating and proposing a new strategy. Also, because the future is unpredictable, for this research, we combine the statistical data to generate the passenger distribution in the whole research time horizon and use the real-time data to keep refreshing the statistical data; the passenger flow data leads to an extremely large number of OD matrices, an example of which is shown in Table 4-4. The first column represents the original stations, while the first row represents the destinations; each number represents the entry speed of passengers with units of persons/time-unit, in this case study, we only consider a single-direction operation.

O \ D	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
S0	X	1	3	2	1	2	2	3	1	1
S1	X	X	1	2	3	1	2	2	2	3
S2	X	X	X	2	1	2	2	1	0	3
S3	X	X	X	X	1	2	3	3	2	1
S4	X	X	X	X	X	2	1	2	2	2
S5	X	X	X	X	X	X	2	1	1	2
S6	X	X	X	X	X	X	X	4	3	2
S7	X	X	X	X	X	X	X	X	3	5
S8	X	X	X	X	X	X	X	X	X	4
S9	X	X	X	X	X	X	X	X	X	X

Table 4-4 Example of passenger flow entry speed OD matrix

According to the basic POSM, if dynamic passenger-oriented scheduling were to be considered in this 10-station Beijing Metro Line 19 scenario, with seven trains with two optional dwelling times and departure time intervals, the number of possible dwelling and departure times will be $2^9 \times 2^{10^6} = 5.902958104 \times 10^{20}$. In the computing and simulation environment programmed by the author (CPU Intel Core i5 2.5 GHz, 12 G of RAM, programming software: Java 11, Eclipse 2021), for every scheduling decision, it

takes on average about 0.2 ms to calculate the result. That means if we use an exhaustive algorithm to enumerate all the possible dwelling and departure decisions, it will take about 3.3×10^{13} hours to get the optimal solution. Though with operation, the number of optional decisions will decrease, this calculation time is still unacceptable. As nearly optimised solutions are acceptable in this problem (we only need to get a timetable better than the previous one), an exhaustive algorithm is also unsuitable to be applied.

With the presented Beijing Metro Line 19 scenario as a case study, the performance of GA_POSM is evaluated in terms of both objective solution and computational time. Two original periodic scheduling strategies (with 6-minute departure intervals and 0.5-minute dwelling, 8-minute departure intervals and 1.5-minute dwelling time) are also tested to compare the results between dynamic scheduling and traditional periodic scheduling.

4.2.4 Scheduling Evaluation with Typical Passenger Flow Scenarios

To evaluate the scheduling performance of GA_POSM with typical passenger flow scenarios, 10 typical dynamic passenger flow distribution scenarios shown in Appendix B were chosen, including a large passenger flow distribution in peak time, normal passenger flow distribution in off-peak time, a large variation of distribution in the different times of the research time horizon, etc. For each distribution scenario, the GA_POSM algorithm is applied to generate scheduling strategies and the SQI based on the objective function of the system is calculated; meanwhile, the SQI calculated by the two original periodic strategies is also calculated. Then the results of the three different strategies are compared.

The basic parameters for the GA_POSM algorithm and the objective function are listed below:

- The ratio of weight ω_1 is set to 0.01 and ω_2 is set to 10^7.
- The number of populations in GA_POSM is set to 200.
- Mutation factors are set to 0.2.
- Crossover factor CR is set to 4.

After considering the weightings choices in Generalised Journey Time (Wheat and Wardman, 2017), we engage in defending relatively balanced weightings ω_1 and ω_2, and let the choice a little bit biased to the service quality. As the results shown in Figure 4-16, a test is implemented to compare the contribution in percentage according to different weightings from both objectives to the final result based on the average value of these typical cases. Base on this result and also for the convenience of observation, ω_1 is set to 0.01 and ω_2 is set to 10^7.

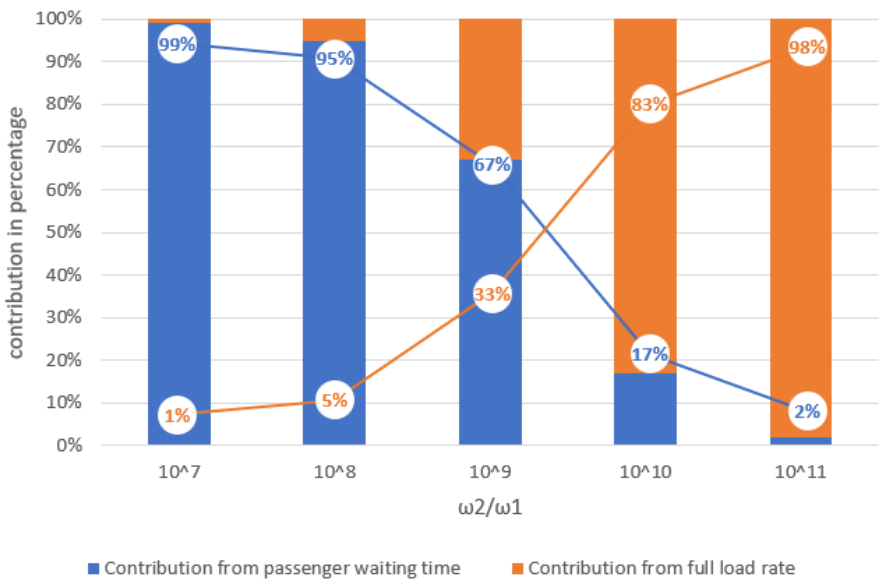


Figure 4-16 Percentage of contributions based on different weightings

One of the typical convergence graphs is shown in Figure 4-17. Based on the hardware and programming environment used by the author (CPU Intel Core i5 2.5 GHz, 12 G of

RAM, programming software: Java 11, Eclipse 2021), the average computational time is around 20 to 25 seconds which can satisfy the time restriction for real-time metro scheduling operation. As the periodic scheduling method generates decisions directly based on a predetermined departure interval and dwelling time without searching processes for optimisation, the computational time of these two traditional periodic scheduling strategies can be ignored.

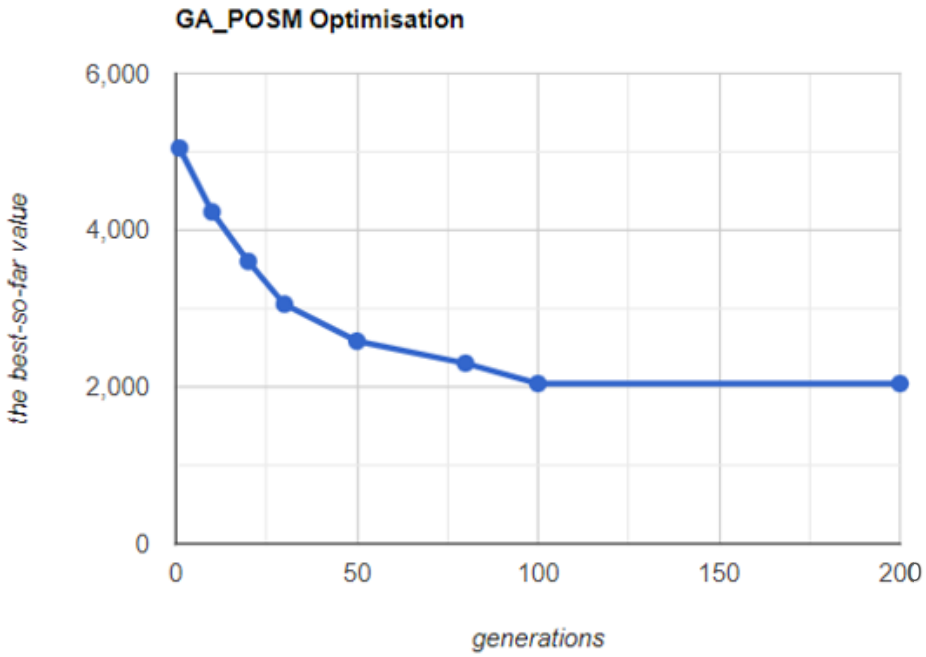


Figure 4-17 Convergence graph of GA_POSM

Table 4-5 shows the GA_POSM strategy scheduling decisions for the 10 typical evaluation scenarios; the numbers after AT and DT represent the specific arrival time and departure time based on the scheduling strategies. Please note, some dwelling time is tight in the table above, these strategies may not suitable for real-life application and can only be implemented in simulation environment for the extreme evaluation.

SN 1	Station 0	Station 1	Station 2	Station 3	Station 4	Station 5	Station 6	Station 7	Station 8
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/21	25/26	30/33	37/38	42/43	47/48	52/53	58/61
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/53	57/58	62/63	67/68	73/76
Train 4 AT/DT	39/40	44/47	51/54	58/61	65/66	70/73	77/78	82/85	90/93
Train 5 AT/DT	52/53	57/58	62/65	69/72	76/79	83/86	90/93	97/98	103/104
Train 6 AT/DT	65/66	70/71	75/76	80/83	87/90	94/95	99/100	104/105	110/111
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 2	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/8	12/15	19/22	26/29	33/36	40/41	45/46	51/52
Train 2 AT/DT	13/14	18/21	25/28	32/35	39/40	44/47	51/52	56/59	64/67
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/53	57/58	62/65	69/72	77/78
Train 4 AT/DT	39/40	44/47	51/52	56/59	63/64	68/69	73/76	80/81	86/89
Train 5 AT/DT	52/53	57/58	62/63	67/70	74/75	79/82	86/87	91/92	97/100

Train 6 AT/DT	65/66	70/71	75/76	80/81	85/86	90/91	95/96	100/103	108/111
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 3	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/3	7/8	12/13	17/18	22/23	27/28	32/33	37/38	43/44
Train 2 AT/DT	19/22	26/27	31/32	36/39	43/44	48/51	55/56	60/61	66/67
Train 3 AT/DT	38/41	45/46	50/53	57/60	64/67	71/72	76/77	81/82	87/88
Train 4 AT/DT	57/60	64/65	69/72	76/79	83/84	88/89	93/96	100/103	108/111
Train 5 AT/DT	76/79	83/84	88/91	95/98	102/103	107/110	114/117	121/122	127/130
Train 6 AT/DT	95/98	102/103	107/110	114/117	121/124	128/131	135/138	142/145	150/151
Train 7 AT/DT	114/117	121/124	128/131	135/138	142/145	149/152	156/159	163/166	171/174
SN 4	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/19	23/24	28/29	33/34	38/39	43/44	48/51	56/57
Train 3 AT/DT	26/27	31/32	36/37	41/44	48/49	53/54	58/61	65/66	71/72

Train 4 AT/DT	39/40	44/45	49/50	54/57	61/62	66/67	71/72	76/79	84/85
Train 5 AT/DT	52/53	57/58	62/63	67/70	74/75	79/80	84/85	89/92	97/98
Train 6 AT/DT	65/66	70/71	75/78	82/83	87/88	92/93	97/98	102/105	110/111
Train 7 AT/DT	78/79	83/86	90/91	95/98	102/103	107/110	114/115	119/122	127/128
SN 5	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/19	23/24	28/29	33/34	38/39	43/44	48/49	54/55
Train 3 AT/DT	26/27	31/32	36/37	41/42	46/47	51/52	56/57	61/62	67/68
Train 4 AT/DT	39/40	44/45	49/50	54/55	59/60	64/65	69/70	74/77	82/85
Train 5 AT/DT	52/53	57/58	62/63	67/68	72/75	79/80	84/87	91/94	99/100
Train 6 AT/DT	65/66	70/73	77/80	84/87	91/94	98/101	105/106	110/111	116/119
Train 7 AT/DT	82/85	89/92	96/99	103/106	110/111	115/116	120/121	125/126	131/134
SN 6	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42

Train 2 AT/DT	13/14	18/19	23/26	30/31	35/36	40/41	45/46	50/53	58/59
Train 3 AT/DT	26/27	31/32	36/39	43/46	50/51	55/56	60/63	67/68	73/76
Train 4 AT/DT	39/40	44/47	51/52	56/59	63/66	70/73	77/80	84/87	92/95
Train 5 AT/DT	52/53	57/58	62/65	69/72	76/79	83/86	90/93	97/98	103/104
Train 6 AT/DT	65/66	70/73	77/80	84/85	89/92	96/97	101/102	106/107	112/113
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 7	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/3	7/8	12/13	17/18	22/23	27/28	32/33	27/28	43/44
Train 2 AT/DT	19/22	26/27	31/32	36/37	41/42	46/47	51/54	58/59	64/67
Train 3 AT/DT	38/41	45/56	50/51	55/56	60/61	65/66	70/71	75/78	83/84
Train 4 AT/DT	57/60	64/65	69/70	74/75	79/80	84/85	89/90	94/95	100/101
Train 5 AT/DT	76/79	83/84	88/89	93/96	100/103	107/110	114/117	121/124	129/130
Train 6 AT/DT	95/98	102/105	109/112	116/119	123/126	130/133	137/140	144/145	150/151
Train 7 AT/DT	114/117	121/124	128/131	135/138	142/145	149/152	156/159	163/166	171/174

SN 8	Station 0	Station 1	Station 2	Station 3	Station 4	Station 5	Station 6	Station 7	Station 8
Train 1 AT/DT	0/3	7/10	14/17	21/24	28/31	35/38	42/43	47/48	53/54
Train 2 AT/DT	19/22	26/29	33/36	40/43	47/48	52/53	57/58	62/65	70/73
Train 3 AT/DT	38/41	45/48	52/55	59/60	64/65	69/70	74/77	81/84	89/92
Train 4 AT/DT	57/60	64/65	69/72	76/77	81/84	88/91	95/98	102/103	108/109
Train 5 AT/DT	76/79	83/84	88/91	95/98	102/105	109/110	114/117	121/124	129/132
Train 6 AT/DT	95/98	102/105	109/110	114/117	121/124	128/131	135/138	142/145	150/151
Train 7 AT/DT	114/117	121/124	128/131	135/138	142/145	149/152	156/159	163/166	171/174
SN 9	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/3	7/8	12/13	17/18	22/23	27/28	32/33	37/38	43/44
Train 2 AT/DT	19/22	26/29	33/36	40/43	47/50	54/57	61/64	68/71	76/77
Train 3 AT/DT	38/41	45/48	52/55	59/62	66/69	73/76	80/83	87/90	95/98
Train 4 AT/DT	57/60	64/67	71/74	78/81	85/88	92/95	99/102	106/107	112/115
Train 5 AT/DT	76/79	83/86	90/93	97/100	104/107	111/114	118/121	125/126	131/132

Train 6 AT/DT	95/98	102/105	109/112	116/119	123/126	130/133	137/140	144/147	152/155
Train 7 AT/DT	114/117	121/124	128/131	135/138	142/145	149/152	156/159	163/166	171/174
SN 10	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/19	23/24	28/29	33/34	28/29	43/46	50/51	56/67
Train 3 AT/DT	26/27	31/32	36/37	41/42	46/49	53/56	60/61	65/66	71/72
Train 4 AT/DT	39/40	44/45	49/50	54/57	61/62	66/69	73/74	78/79	84/87
Train 5 AT/DT	52/53	57/58	62/63	67/68	72/73	77/80	84/85	89/90	95/98
Train 6 AT/DT	65/66	70/71	75/76	80/81	85/86	90/91	95/98	102/103	108/111
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/111	115/116	121/124

Abbreviation: SN + number: different scenarios; AT/DT: arrival/departure time in each station.

Table 4-5 GA_POSM scheduling results for ten typical scenarios

Table 4-6 shows the SQI values of the ten typical scenarios based on the scheduling strategies of GA_POSM and the original two periodic strategies. Each scenario's GA_POSM computational time is also listed in the table.

Scenario ID	GA_POSM		Periodic scheduling 1	Periodic scheduling 2
	SQI	Time (s)	SQI	SQI
1	3015	19	3294	3557
2	1248	20	2764	1323
3	2114	22	2724	2131
4	4132	21	4308	4721
5	3594	21	3609	4099
6	1124	20	1432	3815
7	2558	20	2952	2606
8	2419	19	2956	2420
9	2933	19	3090	2942
10	4012	20	4013	4710

Table 4-6 SQIs of different scheduling strategies for ten typical scenarios

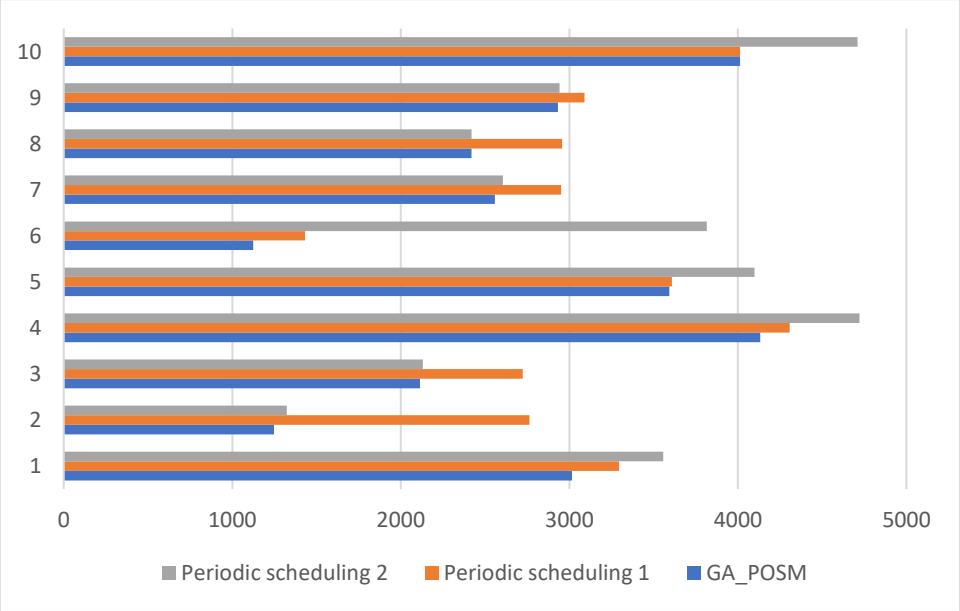


Figure 4-18 Comparison of SQIs between GA_POSM and periodic timetables

Figure 4-18 shows the comparison of GA_POSM with two periodic timetables. As we can see, for different passenger distribution scenarios, such as a large amount of passengers in peak times, a small amount of passengers in off-peak times, a large variation in passenger flow, etc., GA_POSM improves the service quality significantly compared with the normal periodical scheduling strategies in terms of the definition of SQI in this thesis, especially with dramatic variations in passenger flow.

Based on the result, periodic timetables with a shorter departure time interval and dwelling time perform better than the long-interval periodic timetables in peak hours; with a huge amount of passenger flow, the system urgently decreases passenger waiting time. Contrarily, in off-peak hours, the trains' capacity occupation ratio needs to be increased to save energy, thus periodic timetables with a long departure time interval are more suitable. However, the dynamic scheduling strategies generated by GA_POSM are more flexible and can satisfy the two objectives at the same time; trains' departure interval and dwelling time can be modified based on the variation of passenger flow automatically.

4.2.5 Real-Time Optimisation Evaluation with Typical Irregular Variation of Passenger Flow

In this section, the real-time optimisation function of GA_POSM will be evaluated based on the three original passenger flow scenarios from the last section with nine typical irregular variations, such as a dramatic increase or decrease of passenger flow, or some gradual variations of passenger flow; these variations are distributed at different times in the research time horizon. With the passenger counting cameras which are built outside the stations and the EUHT communication units of Beijing Metro Line 19, normally, these variations can be detected and transferred to the control centre 15 minutes (30 time-units) before these irregular passengers enter the platform, so dispatchers have enough time to

estimate the trend of the further OD matrix after the variations and reschedule the timetable based on the estimation.

In this situation, the GA_POSM algorithm will compare the old scheduling strategies generated in last section with the detection time, to determine the modifiable dwelling time and departure time interval firstly. Then, GA_POSM will generate optimised scheduling strategies for the modifiable parts of the original strategies based on the SQI. Finally, the optimised scheduling strategies will be connected with the unmodifiable parts of the original strategies. Nine irregular passenger flow variations scenarios for this case study are shown in Appendix C.

Table 4-7 shows GA_POSM's real-time modified scheduling decisions for the evaluation scenarios; the decisions in yellow cells are the modified scheduling decisions. Please note, some dwelling time is tight in the table above, these strategies may not suitable for real-life application and can only be implemented in simulation environment for the extreme evaluation.

SN 1 DV: 30	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/21	25/26	30/33	37/38	42/43	47/48	52/53	58/61
Train 3 AT/DT	26/27	31/34	38/39	43/46	50/51	55/56	60/63	67/68	73/76
Train 4 AT/DT	39/40	44/45	49/52	56/59	63/64	68/69	73/74	78/81	86/87
Train 5	52/53	57/58	62/65	69/70	74/75	79/80	84/87	91/94	99/100

AT/DT									
Train 6 AT/DT	65/66	70/71	75/76	80/81	85/86	90/91	95/98	102/103	108/109
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 1 DV: 60	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/21	25/26	30/33	37/38	42/43	47/48	52/53	58/61
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/53	57/58	62/63	67/68	73/74
Train 4 AT/DT	39/40	44/47	51/54	58/61	65/66	70/71	75/76	80/81	86/87
Train 5 AT/DT	52/53	57/58	62/63	67/70	74/75	79/82	86/87	91/92	97/98
Train 6 AT/DT	65/66	70/71	75/76	80/81	85/86	90/91	95/96	100/101	106/109
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 1 DV: 90	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/21	25/26	30/33	37/38	42/43	47/48	52/53	58/61
Train 3	26/27	31/34	38/41	45/48	52/53	57/58	62/63	67/68	73/74

AT/DT									
Train 4 AT/DT	39/40	44/47	51/54	58/61	65/66	70/71	75/76	80/81	86/87
Train 5 AT/DT	52/53	57/58	62/65	69/72	76/79	83/86	90/93	97/98	103/104
Train 6 AT/DT	65/66	70/71	75/76	80/83	87/90	94/95	99/100	104/105	110/111
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 1 IV: 30	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/21	25/26	30/33	37/38	42/45	49/50	54/55	60/63
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/55	59/62	66/69	73/74	79/80
Train 4 AT/DT	43/46	50/53	57/60	64/67	71/72	76/79	83/86	90/93	98/99
Train 5 AT/DT	62/65	69/72	76/79	83/86	90/93	97/98	102/103	107/110	115/116
Train 6 AT/DT	81/84	88/91	95/98	102/105	109/110	114/115	119/122	126/129	134/137
Train 7 AT/DT	100/103	107/110	114/117	121/124	128/131	135/138	142/145	149/152	157/160
SN 1 IV: 60	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42

AT/DT									
Train 2 AT/DT	13/14	18/19	25/26	30/33	37/38	42/43	47/48	52/53	58/61
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/53	57/58	62/65	69/70	75/76
Train 4 AT/DT	39/40	44/47	51/54	58/61	65/66	70/71	75/76	80/81	86/87
Train 5 AT/DT	52/53	57/58	62/65	69/72	76/77	81/82	86/87	91/92	97/98
Train 6 AT/DT	65/66	70/71	75/76	80/83	87/88	92/93	97/98	102/103	108/109
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 1 IV: 90	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/6	10/11	15/16	20/21	25/26	30/31	35/36	41/42
Train 2 AT/DT	13/14	18/21	25/26	30/33	37/38	42/43	47/48	52/53	58/61
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/53	57/58	62/63	67/68	73/76
Train 4 AT/DT	39/40	44/47	51/54	58/61	65/66	70/73	77/78	82/85	90/93
Train 5 AT/DT	52/53	57/58	62/65	69/72	76/79	83/86	90/93	97/98	103/104
Train 6 AT/DT	65/66	70/71	75/76	80/83	87/90	94/95	99/100	104/105	110/111
Train 7	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120

AT/DT									
SN 2 IV: 30	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/8	12/115	19/22	26/29	33/36	40/41	45/46	51/54
Train 2 AT/DT	13/14	18/21	25/28	32/35	39/42	46/47	51/54	58/61	66/69
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/55	59/62	66/69	73/76	81/84
Train 4 AT/DT	43/46	50/53	57/60	64/67	71/72	76/79	83/86	90/93	98/101
Train 5 AT/DT	62/65	69/72	76/77	81/84	88/91	95/98	102/105	109/112	117/120
Train 6 AT/DT	81/84	88/91	95/98	102/103	107/110	114/117	121/124	128/131	136/139
Train 7 AT/DT	100/103	107/110	114/117	121/124	128/131	135/138	142/145	149/152	157/160
SN 2 IV: 180	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/1	5/8	12/15	19/22	26/29	33/36	40/41	45/46	51/52
Train 2 AT/DT	13/14	18/21	25/28	32/35	39/40	44/47	51/52	56/59	64/67
Train 3 AT/DT	26/27	31/34	38/41	45/48	52/53	57/58	62/65	69/72	77/78
Train 4 AT/DT	39/40	44/47	51/52	56/59	63/64	68/69	73/76	80/81	86/89
Train 5	52/53	57/58	62/63	67/70	74/75	79/82	86/87	91/92	97/100

AT/DT									
Train 6 AT/DT	65/66	70/71	75/76	80/81	85/86	90/91	95/96	100/103	108/111
Train 7 AT/DT	78/79	83/84	88/89	93/94	98/99	103/104	108/109	113/114	119/120
SN 3 DV: 30	**Station 0**	**Station 1**	**Station 2**	**Station 3**	**Station 4**	**Station 5**	**Station 6**	**Station 7**	**Station 8**
Train 1 AT/DT	0/3	7/8	12/13	17/18	22/23	27/28	32/33	37/38	43/44
Train 2 AT/DT	19/22	26/27	31/32	36/37	41/42	46/47	51/52	56/67	62/65
Train 3 AT/DT	34/35	39/40	44/45	49/50	54/57	61/62	66/67	71/72	77/78
Train 4 AT/DT	47/48	52/53	57/58	62/63	67/70	74/75	79/80	84/85	90/91
Train 5 AT/DT	60/61	65/66	70/73	77/78	82/85	89/90	94/95	99/102	107/110
Train 6 AT/DT	73/76	80/83	87/90	94/97	101/102	106/109	113/114	118/121	126/127
Train 7 AT/DT	92/93	97/100	104/107	110/114	118/121	125/126	130/131	135/136	141/144

Abbreviation: SN + number: different scenarios; IV/DV + number: increasing/decreasing variations of passenger flow + the detection time-units of variations; AT/DT: arrival/departure time in each station.

Table 4-7 Real-time optimisation results for nine typical scenarios

Table 4-8 records the SQI values after the scenarios with real-time irregular passenger flow variations, according to the real-time optimisation strategies and the original

scheduling strategies based on GA_POSM. The computational time of real-time optimisation is also listed in the table.

Scenario ID	GA_POSM Real-time optimisation		GA_POSM Original
	SQI	Time (s)	SQI
1	2246	19	2294
2	2722	19	2760
3	3015	19	3015
4	2648	20	2984
5	3075	19	3120
6	3014	20	3014
7	2726	20	2916
8	1248	20	1248
9	4280	20	4327

Table 4-8 SQIs after nine real-time modifications

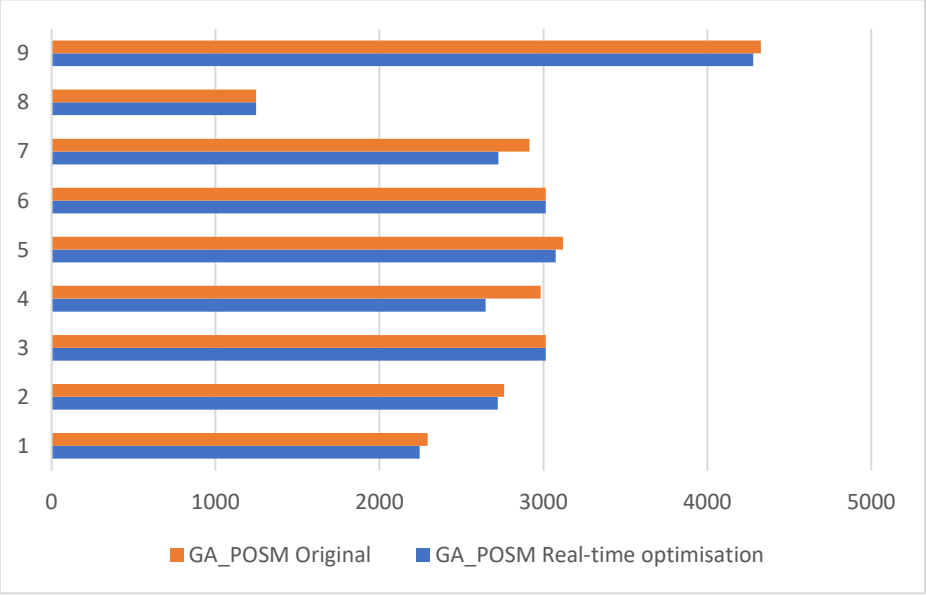


Figure 4-19 Comparison of SQIs after modifications and original scheduling

Figure 4-19 shows the comparison of SQI results for the irregular passenger flow variation, after GA_POSM's real-time optimisation and original dynamic scheduling strategies. As can be seen, after the detection time, the scheduling strategies have been changed to satisfy the irregular passenger flow variation using the real-time optimisation strategies. With a sudden increase of passenger flow in some specific stations, most trains are modified to a shorter departure time interval and dwelling time in other stations to take the large amount of passengers in these specific stations. By Contraries, with an unexpected decrease in passengers, most trains will be modified with a long departure interval and dwelling time to increase the trains' capacity occupation ratio. According to the modifications in Table 4-7 and the SQI result from Table 4-8, we find that the earlier the irregular variation is detected, the better the modified result. Thus, updating passenger flow data in time is an essential process for real-time scheduling optimisation. The limitation is, in some situation when the variation was detected late, the improvement is minor, but this method can increase the overall flexibility of the system.

4.2.6 Poisson Distribution Evaluation

Because of the randomicity of passenger flow variation, the selected 10 entry rates of the passenger flow distribution scenarios are too few to cover all the typical distribution scenarios for evaluation. It is impossible to test all the passenger flow entry rate distribution scenarios and it is also very difficult to evaluate the operation strategy in real life. Thus, in order to further evaluate the performance of the GA_POSM algorithm, in this section, a statistical evaluation process based on passenger flow rate Poisson distribution has been carried out to evaluate the performance of GA_POSM, in terms of an average SQI.

Poisson distribution is a probability distribution that models scenarios that have a default behaviour and cumulative possible deviations from that behaviour. It is a discrete probability distribution that expresses the probability of a given number of events which are independent of the time since the last event occurring in a fixed time interval. Poisson distribution is suitable to be applied to a system with a large number of possible events during a fixed time interval, such as the passenger flow distribution and variation in the research time horizon. The probability mass function of a Poisson distribution is shown in Equation 5-1. This function calculates the possibility of occurrence k happening in the research time interval.

$$Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!} \qquad \textbf{Equation 4-8}$$

where λ is the expected rate of occurrences, k is the number of occurrences and e is Euler's number.

For this research, Poisson distribution is used to generate more passenger flow rate data based on typical situations to update a passenger flow rate with more randomness in every detecting period for more case studies. The passenger flow rate from four typical Beijing metro stations shown in Figure 4-15 is implemented as the data bound for the Poisson distribution since they can represent the typical passenger flow rate distribution in daily life.

Based on the passenger flow data from the previous section, the entry rate of passenger flow in each station is assumed to be between [3, 17], where in the Poisson distribution, the parameter λ is set to 10. The probability mass function diagram and cumulative distribution function diagram are shown in Figures 4-20 and 4-21.

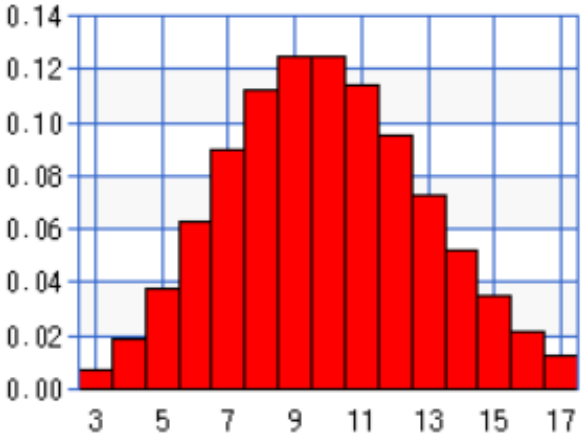


Figure 4-20 Probability mass function diagram of passenger entry rate

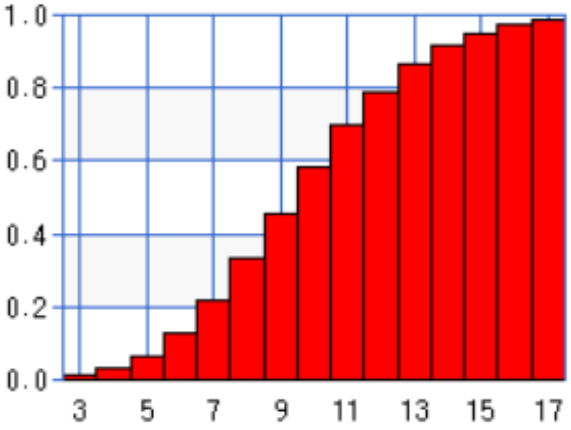


Figure 4-21 Cumulative distribution function diagram of passenger entry rate

The aim of applying the Poisson distribution in this research is to generate a large amount of passenger flow rate OD data which can simulate different passenger distribution scenarios in different stations. For each passenger distribution scenario, the dynamic scheduling method of GA_POSM and the two traditional periodic timetables are applied to the simulator to calculate the SQI results. With the large numbers of passenger distribution scenarios, for evaluation and comparison of different scheduling methods, the statistical average value of SQI is calculated. A lower statistical SQI value shows better performance of the scheduling strategies based on the objectives' solutions. Besides

that, to evaluate the real-time optimisation function of GA_POSM, irregular passenger flow variations based on the Poisson distribution are also generated; the comparison of SQI based on different scheduling strategies after the variations is also calculated. The procedure of the evaluation is shown in Figure 4-22.

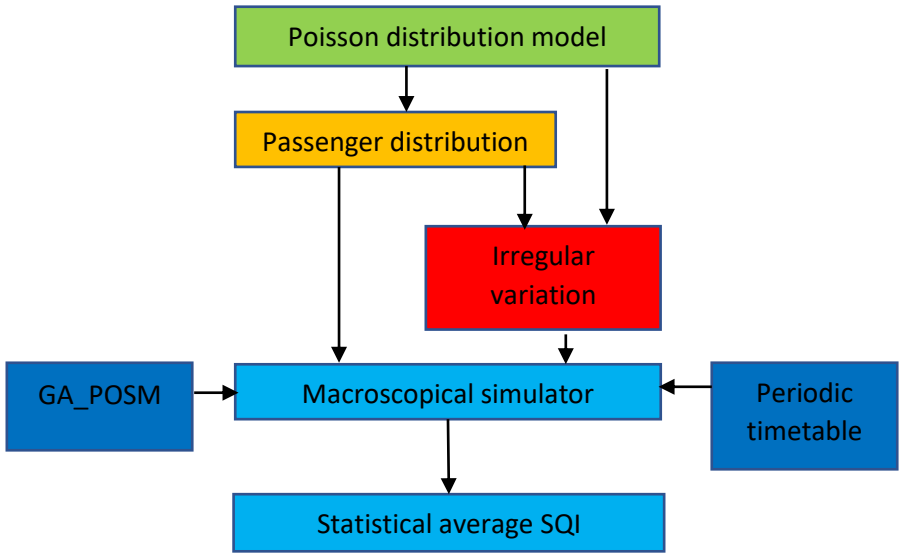


Figure 4-22 Statistical evaluation procedure

For each scheduling strategy, 100 simulation passenger distribution scenarios have been generated and applied for evaluation. The statistical SQI results of GA_POSM, short periodic timetable and long periodic timetable are listed in Table 4-9.

Scheduling strategy / Result	GA_POSM scheduling	GA_POSM Real-time optimisation	Short periodic timetable	Long periodic timetable
Average SQI results of scheduling	2080		2867	2799
Average SQI results of real-time optimisation	2872	2715		

Table 4-9 Average SQIs for different scheduling strategies under passenger flow Poisson distribution evaluation

From the table above, it can be seen that compared with the periodic scheduling timetables, GA_POSM can improve the service quality on average by 30% based on the SQI determined in this article. For the real-time optimisation function, compared with the original dynamic scheduling strategies, the service quality can be improved by around 7% after real-time modification.

4.3 A Faster Real-Time Metro Operation Method Without Timetable Based on GA_POSM and Decision Tree Algorithm

Based on the above case study, we validated GA_POSM in dynamic passenger flow-based scheduling and real-time scheduling optimisation. For the 10-station metro system, Beijing Metro Line 19, because of the integrated modified genetic algorithm and macroscopic metro simulator, GA_POSM can propose the optimisation of operation strategies in a short time.

However, different to other optimisation problems, for different metro systems, there are dramatic variations in the number of stations. For example, the Tunnel metro system in Istanbul, Turkey has only two stations, while the New York City Subway has 472 stations. The number of trains needing to be optimised is also variable according to different requirements. Some operators only need an optimised strategy for the trains in peak hours; however, some systems may need an optimised strategy for the whole day. In different metro systems, the number of optional dwelling times and departure time intervals is also different. Moreover, for this problem, we may need to satisfy different number of objectives in different time horizon (as the comprehensive objective function). Thus,

compared with other optimisation problems, the total number of solutions for this research is fluctuating, which will lead to variation of the gene sequence length in the genetic algorithm. Table 4-10 shows the variation of the number of solutions and gene sequence length according to different numbers of stations, trains and optional stop strategies.

Number of stations considered	Number of trains considered	Optional dwelling and departure level	Number of solutions	Length of gene sequence
2	5	2	2^{10}	10
4	6	2	2^{24}	24
8	7	2	2^{56}	56
10	8	4	2^{160}	160
20	9	4	2^{360}	360
30	10	4	2^{600}	600

Table 4-10 Number of solutions and variation of gene sequence length based on different infrastructure

Generally, the efficiency of a genetic algorithm-based optimising method is seriously impacted by the number of variables: more variables will lead to more solutions and a longer gene sequence, larger population and more iterations in the optimising process. Because of the increase of iterations, a longer calculation time is inevitable. As a real-time operation method need to be applied for most metro systems, operational strategies should be proposed in a short time without being impacted by the infrastructures of different systems. Thus, we aim to integrate the proposed GA_POSM with the decision

tree learning method, in order to provide operation strategies more faster without the impacts of different infrastructures.

4.3.1 Introduction to Decision Tree Learning

As a non-parametric supervised machine learning method, decision tree learning is always considered to be the most straightforward algorithm in machine learning. Other supervised, unsupervised and reinforcement machine learning approaches could also be considered. However, as a problem with proposed mathematical model and implemented optimising algorithm, instead of applying with other complicated learning algorithms, the decision tree algorithm has better interpretability and computational efficiency. Moreover, the decision tree learning only requires little data preparation and also has good performance with large datasets. Duo to its various advantages, this method has been commonly used in various application scenarios for decades.

A decision tree algorithm generates the results by splitting the training data set based on different training features; the training data can be formed as in Equation 4-9, where the vector $\boldsymbol{x}$ is the set of different training features and Y is the target variables based on the training data. A decision tree mainly contains three types of node, the root node, which is the original tree node; the internal nodes, which contain the evaluation of variables; and the leaf nodes, which are the last nodes in the decision tree and will help to make decisions when new data is inputted. Based on a recursive partitioning process, a tree model as shown in Figure 4-23 can be built.

$$(\boldsymbol{x}, Y) = (x_1, x_2, x_3, \dots, y) \qquad \textbf{Equation } 4-9$$

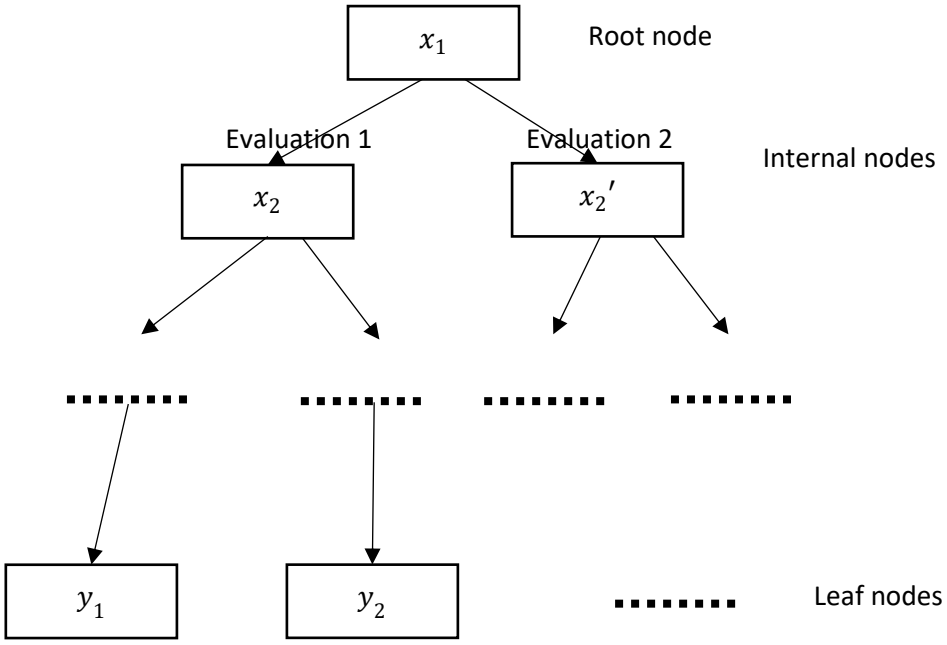


Figure 4-23 Tree model of the decision tree learning process

Normally, decision tree learning is a method commonly used in data mining, which engages to generate a tree model that predicts the value of a target variable based on several input variables. For example, Crosby et al. (2016) introduced a novel approach based on decision tree learning to predict UK-wide daily traffic counts on all roads in England and Wales. Liu et al. (2017) utilised a machine-learning algorithm based on decision tree to predict future copper prices.

In decision analysis, decision tree learning can be used for decision making, especially with discrete decision values. In this situation, the tree model is called a classification tree; the leaves in the classification tree represent the class labels of the decisions and the branches represent conjunctions of features that lead to those decisions. For example, Su and Shiue (2003) developed an intelligent scheduling controller to support a shop floor control system to make real-time decisions based on decision tree learning. Bian and Wang (2021) studied a new school enterprise cooperation mechanism based on an improved decision tree algorithm.

For the real-time metro operation problem in this research which can accept suboptimal discrete solutions, compared with continual optimisation of the system in real time, building a decision tree for each train at different stations according to the optimised dwelling time levels and departure time intervals based on a large number of results can help the operator make decisions faster in real-time operation and avoid the impacts from the infrastructures in different systems. For this research, the internal nodes of the decision tree can be labelled with system parameters and requirements based on the objective function, such as passenger arrival rate at different stations and the operation time. The branches of the decision tree represent the different inputs of the parameters which will lead to different results. The last leaves represent the decisions for the level of dwelling time and departure time interval based on different situations; as all the learning inputs for the decision tree have already been optimised, these decisions can be applied to the system directly.

4.3.2 Decision Tree Learning Features for the Real-Time Train Operation Problem

As the important elements which severely impact the learning process and the decision result, appropriate learning features of the decision tree should be selected meticulously. For the real-time train operation problem, we aim to build dwelling time-level decision trees to make the dwelling time decision for each train at different stations and also build the departure time interval decision trees to decide a suitable departure time interval between two trains; the target is to minimise the CSQI in the comprehensive objective function. Because the optimisation process is based on proposed multiple objective function, the learning features should also be based on the function's parameters; selected essential learning features and statements are given as follows:

The time of making decisions: As an important parameter in the objective function, in the research time horizon, the passenger flow entry rate, the adjustable dwelling times and departure intervals and the choice of objectives are closely connected with the time of making decisions.

Passenger flow entry rate: The rate with which passengers enter different stations is also an essential parameter in the objective function. Based on the optimised results in Chapter 4, this feature will impact trains' dwelling time choices at different stations. Normally, trains will stop for longer at stations with a high passenger entry rate.

The passengers left at each station: The number of passengers left at each station is also an important learning feature. In the operation process, the entry rate of passengers in some stations may be slow, but the number of left passengers could be large; in this situation, a train also needs to stop for longer.

Based on the learning features listed above, a sample learning table for a train's dwelling operation strategies for the second station in a three-station system can be shown as in Table 4-11.

Station Number: 2				Train Number: 2			
Time horizon	**Entry rate of passengers in different stations (persons/minute)**			**Passengers left at different stations**			**Dwelling decision**
	1	**2**	**3**	**1**	**2**	**3**	
1	2	2	1	0	0	0	Long
1	1	10	2	0	0	0	Long
1	2	3	8	1	0	0	Short
2	1	3	3	0	0	10	Short
2	3	1	3	2	8	1	Long
3	4	3	3	0	0	0	Long
3	8	2	7	0	1	5	Short
3	1	8	1	1	2	2	Long

Table 4-11 Sample learning table for a train in a three-station system

4.3.3 Explanation of the Learning Process of Decision Trees

To build a decision tree, a method is needed to split the learning samples; in this research, we used Gini impurity as the splitting index. A Gini impurity is a number between 0 and 0.5 which can help to classify random data based on a specific feature from a dataset. The formula of Gini impurity is given by Equation 4-10:

$$Gini = 1 - \sum_{i=0}^{j} P\left(\frac{i}{N}\right)^2 \qquad \textbf{Equation 4-10}$$

where N represents the number of datasets, i represents the number of data in a specific label or decision, and P represents the ratio of a decision. In the process, we aim to choose suitable features to decrease the Gini impurity to make decisions. For example, a decision

tree based on the sample learning data in Table 4-11 is shown in Figure 4-24; ES represents the passenger entry rate while LP is the number of passengers left in different stations.

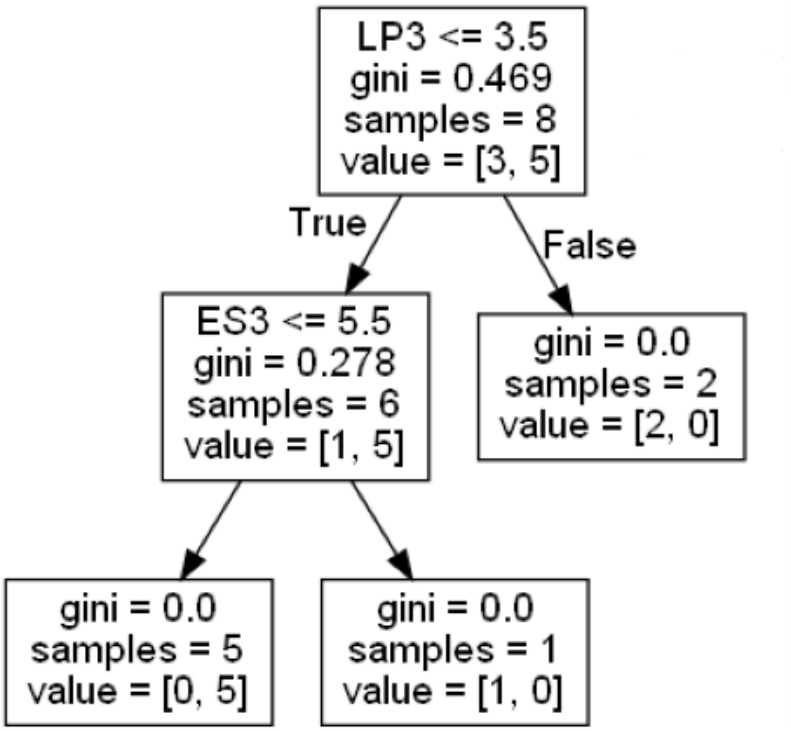


Figure 4-24 Decision tree based on the sample data

To build this decision tree, at first the Gini impurity based on all the data needs to be calculated. The elements in each step are explained as follows.

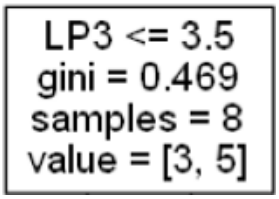


Figure 4-25 Root node of the decision tree example

Samples = 8 means that all eight decisions are left at this point in the decision process as this is the root node.

Value = [3, 5] means three of the decisions are operated based on the short dwelling strategy and five with the long strategy.

Gini = 0.469 the calculation of Gini impurity for this node can be given as: $1 - \left(\frac{3}{8}\right)^2 - \left(\frac{5}{8}\right)^2 = 0.469$.

LP3 ≤ 3.5 means the number of passengers left in station 3 is the feature to decrease the Gini impurity in this step; the node should follow this rule to split.

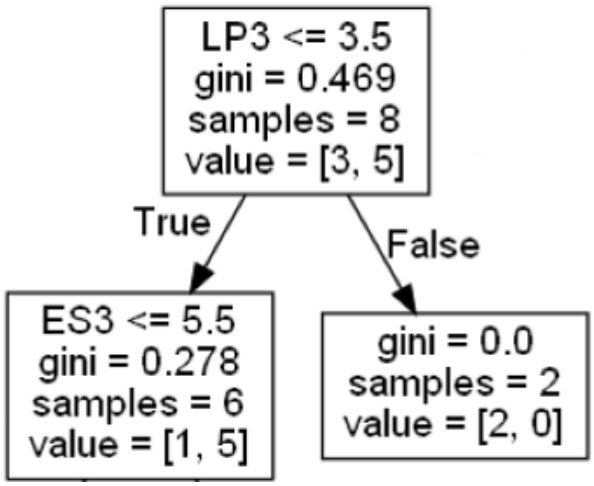


Figure 4-26 First splitting process of the decision tree

Based on the splitting rule above, two decisions are classified as 'False', and follow the right-hand route; as both of these decisions are operated with short dwelling strategy, the Gini impurity is decreased to zero. Six decisions are classified as 'True' and follow the left-hand route.

Samples = 6 means that all six decisions are left at this point in the decision process as their number of passengers left in station 3 is less than 3.5.

Value = [1, 5] means one of the decisions is operated based on the short dwelling strategy and five with the long strategy.

Gini = 0.278 the calculation of Gini impurity for this node can be given as:$1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = 0.278.$

ES3 ≤ 5.5 means the passenger entry rate at station 3 is the feature to decrease the Gini impurity in this step; the node should follow this rule to split.

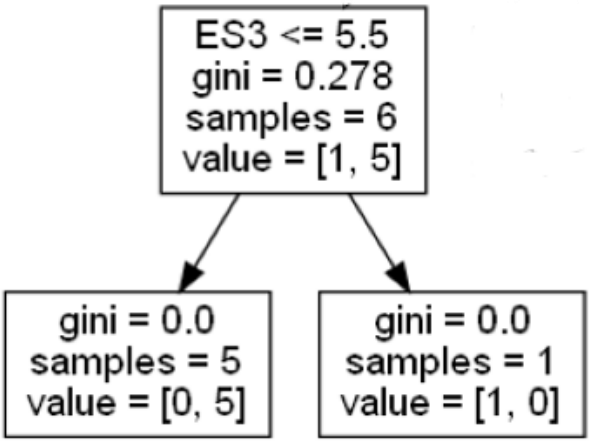


Figure 4-27 Second splitting process of the decision tree

Based on the rule above, the data with a passenger flow entry rate less than 5.5 have been divided to a new left node; the others are directed to the right node. As the data in both final nodes have the same decisions, all the Gini impurities of this decision tree have been decreased to zero.

The example above introduces the basic generation process of a decision tree. Theoretically, with appropriate learning features, the decision tree's Gini impurity can always be decreased to zero. Otherwise, some learning features may not have any connection with the decision result or there are not enough input learning data. As an algorithm based on a large number of optimising results, in some situations, compared with using the genetic algorithm directly, the decisions generated based on a decision tree

may not be the best, but the calculation time based on a decision tree is negligible, which makes this method suitable for macroscopic real-time operation.

4.4 Conclusions

In first part of this chapter, an innovative algorithm GA_POSM has been proposed for solving the passenger flow-oriented scheduling and real-time optimisation problems formulated with POSM. Based on a macroscopic metro simulator, the binary arrays in the traditional genetic algorithms can be converted into operational strategies directly, and an innovation operation 'gene length modification' has been derived for real-time optimisation. The addition of the gene length modification operation makes GA_POSM an efficient tool to optimise trains' operation strategies in real time. Some performance evaluations based on typical passenger flow from the Beijing metro system were presented to evaluate the performance of GA_POSM for both scheduling and real-time optimisation, the basic POSM model has been validated. Also, a stochastic performance evaluation based on Poisson distribution has been applied. Two periodic timetables based on a real-life metro system's data were compared. As the performance evaluation results show, under different passenger flow distributions, compared with the periodic operation, the scheduling method can decrease the SQI defined in this article and improve the service quality significantly. Furthermore, the real-time optimisation can modify the operation strategies in real time to satisfy the variations of passenger flow, which significantly improves the flexibility of metro operation. Based on the test, earlier detection leads to a better optimised result.

In the second part of this chapter, we considered the complex requirements of real life metro systems and the comprehensive model based on our field study. Then, we introduced and integrated a decision tree algorithm with proposed methodology to improve its applicability for different systems and objectives. The system architecture and operation process of the integrated methodology will be introduced in next chapter.

Chapter 5. Methodology Application to Real life Metro Operation

In the previous chapters, the formulation of dynamic passenger flow-oriented scheduling and real-time optimisation problems was presented, a comprehensive model based on field study for real-time operation was extended. the modified genetic algorithm GA_POSM was designed. Different evaluation results can prove the good performance of GA_POSM in decreasing passenger flow waiting time and improving service quality compared with the traditional periodic metro operation. And a faster real-time metro operation method without timetable with higher applicability based on the proposed GA_POSM integrated with a decision tree algorithm has been presented. In this chapter, an integrated system architecture is introduced for real-time metro operation, to integrate the proposed operation method with traffic management and train control systems in a real-life metro system. And a macroscopic case study for the real-time metro operation method without timetable will be demonstrated.

5.1 Integrated System Architecture for Real-Time Metro Operation Without Timetable

In the previous chapters, the methodology GA_POSM for scheduling metro trains based on dynamic passenger flow was presented and validated, and it was also derived with a decision tree algorithm to provide real-time operation decisions for the metro system without a prepared timetable. To implement the proposed methodology with real metro systems, an integrated system architecture must be proposed, as two important

components for real-time operation, the requirements and structures of the real-time passenger flow data collection system and the passenger flow-based operation strategy system need to be considered. In this part, based on the passenger monitoring system applied in the Chinese metro system and the proposed real-time passenger flow-based decision methodology, the operational processes of these components will be introduced and integrated. The objective is to deliver an integrated metro control process which can provide operational strategies in real time without a timetable.

5.1.1 Structure of the Real-Time Data Collection System

Based on the requirement of proposed methodology, the real-time data collection system mainly contains three modules, which will be introduced separately as follows:

(1) Data collection module

This module mainly collects the passenger flow data, station data and metro network data; the following units should be included in this system:

- Ticket data collection unit: connect with the AFC system in stations, collect the data of passenger flow and tickets which include the entry station and the destinations.
- Passenger number data collection unit: connect with the monitoring system in each station, collect data including the number of passengers and the entry rate of passengers in each station and platform.
- Station and metro network data collection unit: collect the infrastructure data of the metro system based on the metro's Geographic Information System.

(2) Data processing module

This module receives and processes the passenger flow data collected by the data collection module; the following units should be included in this system:

- Passenger OD matrix unit: integrate each ticket's entry and destination data from the ticket data collection unit with the passenger flow entry rate data from the passenger number data collection unit and generate a real-time passenger entry rate OD matrix.
- Operation data unit: generate operation data including the running time between stations, optional dwelling time and departure time intervals for the metro system based on the station and network data.
- Passenger data recording unit: record the passenger entry rate OD matrix generated. The recorded historical data can be used to pre-optimise the scheduling before daily operation.

(3) Data transmission interface module

This module analyses and transfers the processed passenger flow and metro system data for the proposed methodology to the operation decision system to generate operational strategies. Based on existing metro data collection and transmission systems (such as the EUHT-5G technology introduced in the Chapter 4), this process takes 15 minutes in a normal situation. Thus, the average real-time passenger flow data will continue to be kept updated and transmitted in a specified period.

The flow chart of the whole real-time data collection system is shown in Figure 5-1.

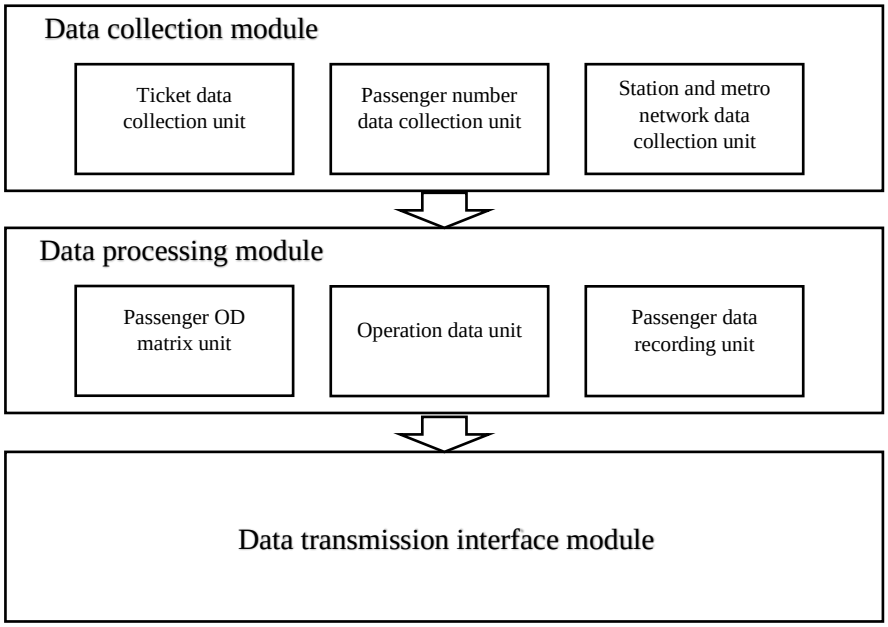


Figure 5-1 Flow chart of real-time data collection system

5.1.2 Structure of the Real-Time Passenger Flow-Oriented Operation Decision System

Based on the passenger flow data updated by the real-time data collection system, the real-time passenger flow-oriented operation method (RPOM) should propose appropriate operational decisions according to the proposed approaches, including the algorithm GA_POSM and the decision tree algorithm.

Before the operation starts, as there will not be big modifications of the number of trains in real-life operation, the algorithm GA_POSM is able to test and propose a optimised scheduling strategy based on the historical passenger flow data for the system with different numbers of trains, and the optimisation results will become the learning data for the decision tree algorithm. The flow chart of this pre-operational process is shown in Figure 5-2. After the operation starts, the number of trains is fixed; the flow chart of GA_POSM in real-time application is shown in Figure 5-3.

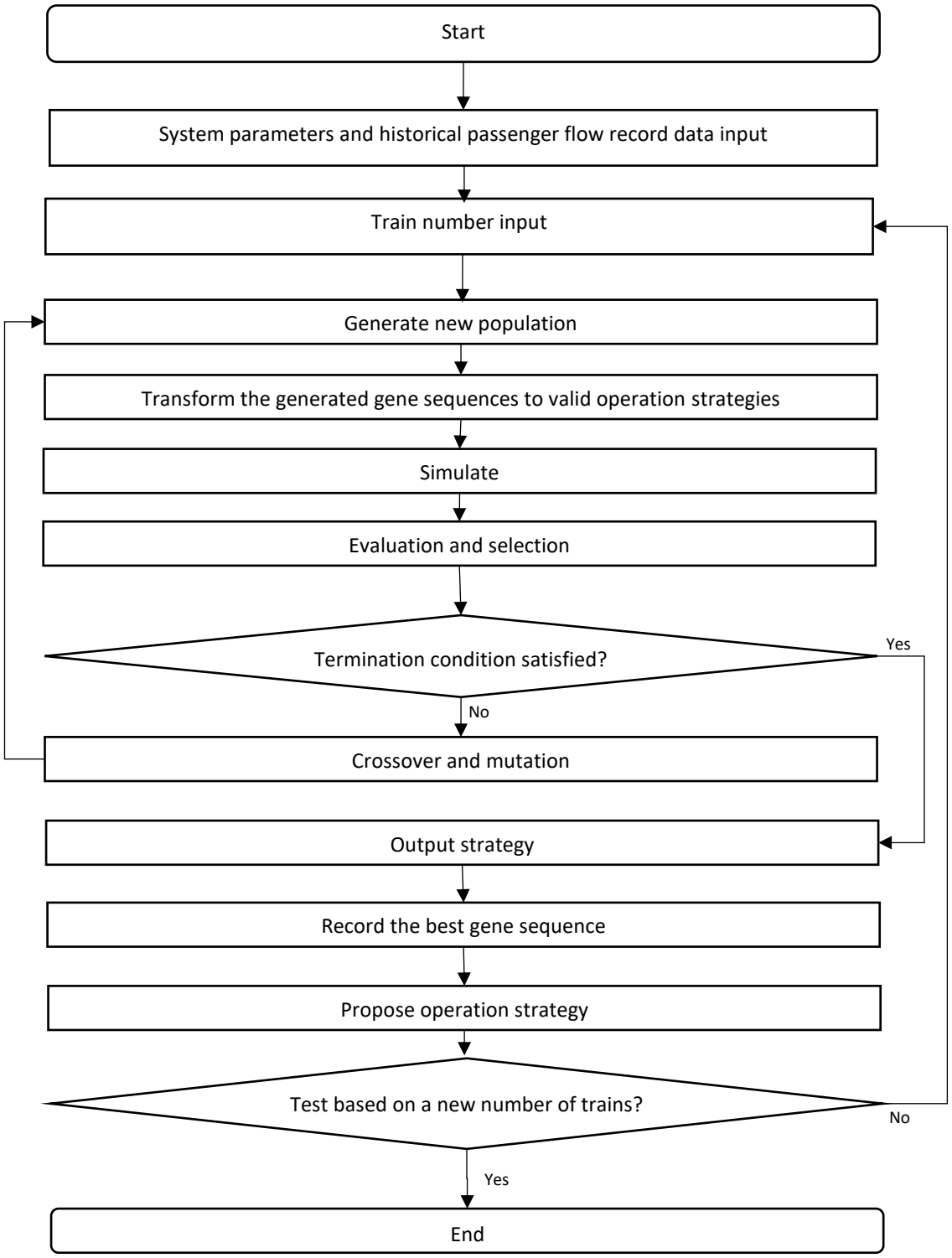


Figure 5-2 Flow chart of GA_POSM in pre-operational process

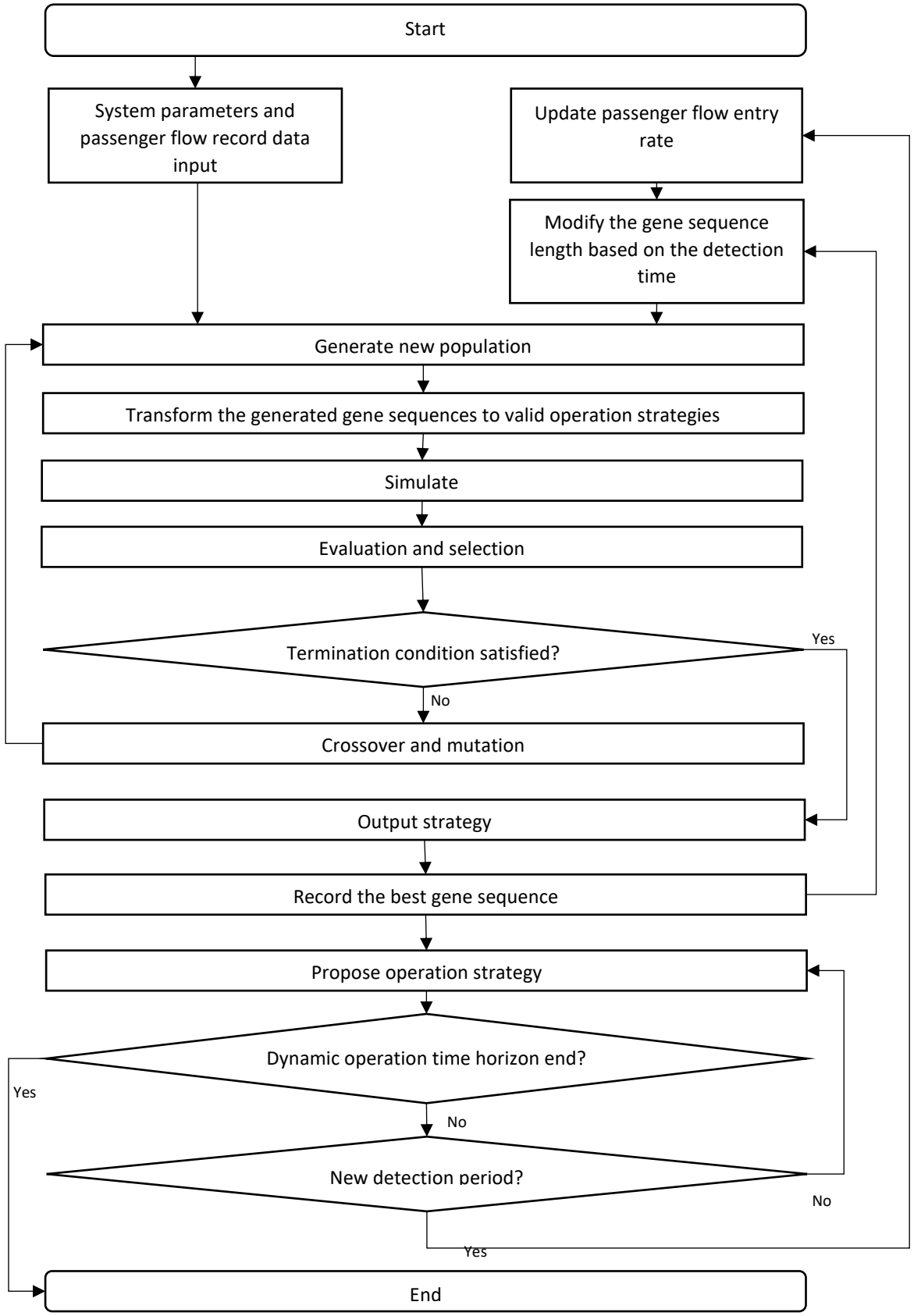


Figure 5-3 Flow chart of GA_POSM in real-time operation

In RPOM, all the decisions generated by the algorithm GA_POSM will be regarded as the learning data and inputted to the decision tree algorithm. In real-time operation, the operation decisions will be made by the decision tree algorithm directly to save time. The whole system structure including the interfaces between different systems, operation process and data flow is shown in Figure 5-4.

The specific application process can be summarised by the following steps:

1. Based on the passenger OD data and entry rate data from the AFC ticket system and passenger counting system, record and collect dynamic passenger flow entry rate OD data for the operation time horizon.

2. Based on the infrastructure data from the GIS system, generate the operation constraint data for the operation algorithm.

3. Based on the historical passenger flow OD data stored in the data collection system, propose different dynamic passenger flow-oriented metro scheduling plans according to a specific object in the operation time horizon. Test different numbers of trains, optimise the departure intervals and dwelling times with the GA_POSM algorithm and input all the optimised plans to the decision tree algorithm as the learning data.

4. Based on the real-time update frequency of the AFC ticket system and passenger number counting system, keep calculating and updating the real-time detected passenger flow entry rate OD data. Based on the real-time data, keep proposing new operation decisions according to the decision tree generated. The GA_POSM algorithm will also keep proposing optimising strategies and input these strategies to the decision tree algorithm as learning data in real time.

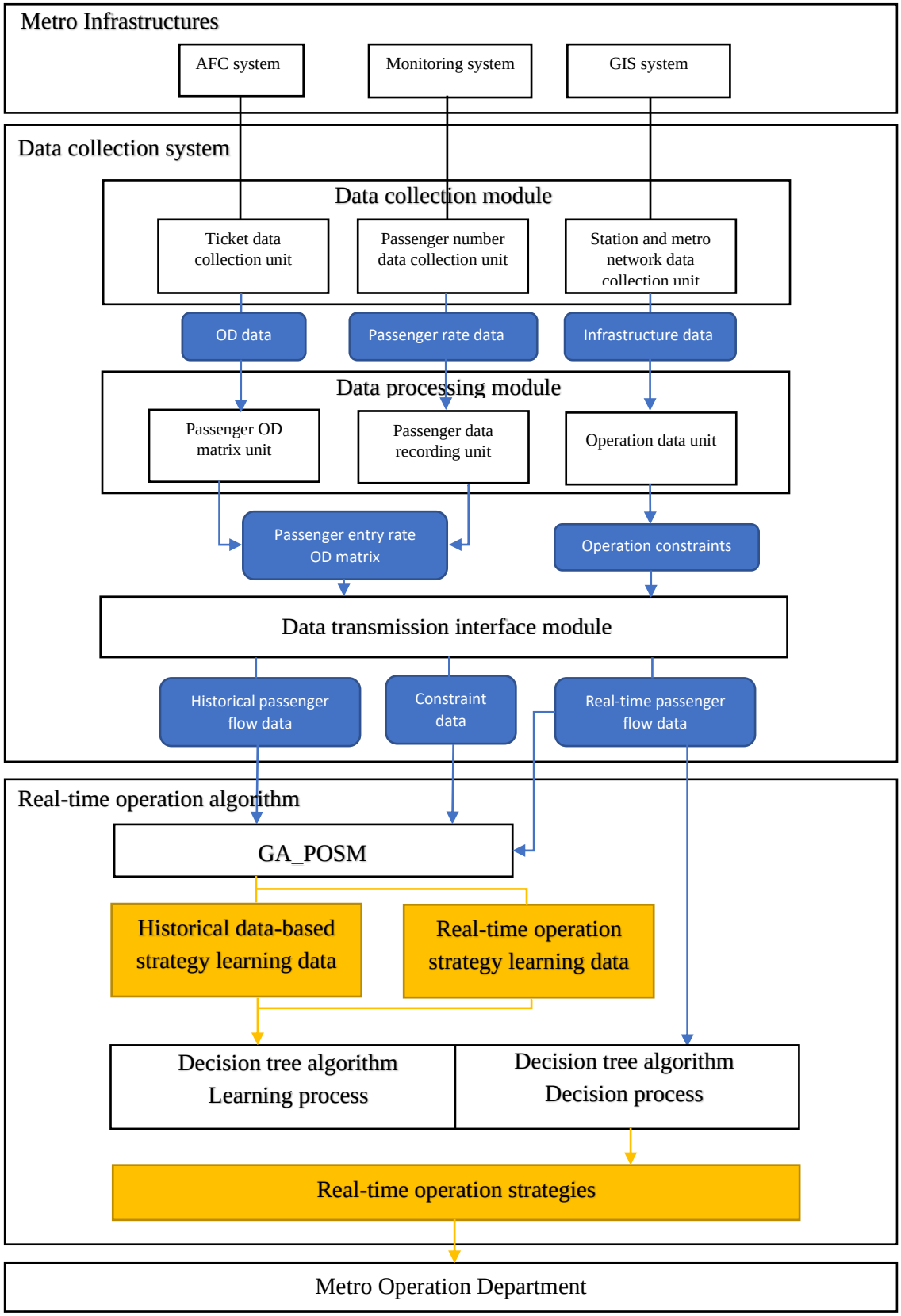


Figure 5-4 Real-time metro operation process and data flow

5.2 Real-Time Passenger Flow-Oriented Metro Operation Without Timetables Case Study

In the previous section, we introduced the concept of the innovative real-time passenger flow-oriented metro operation method without timetable. In this section, a large-scale real-time passenger flow-oriented metro operation case study will be carried out and the result of RPOM will be evaluated.

As described in the previous chapter, Beijing Metro Line 19, which will be allocated with new 5G network communication and passenger counting technology from Beijing Infrastructure Investment Co Ltd, is still used to demonstrate the real time passenger flow-oriented metro operation method and evaluate its performance. Generally, the detection and transfer process will take 15 minutes with existing EUHT communication units. Thus, 15 minutes is regarded as the detection period, as introduced earlier, this limitation is mainly related to the measurement and transmission technologies applied now. With the application of the decision tree algorithm, the real-time calculation time for the macroscopic system will be highly reduced. This case study applied with a double-direction line from Xingong station to Mudanyuan station. For the optimisation process, this double-direction line system can be transformed to a single-direction system with twice the number of stations, including a disembark-only station and an embark-only station. As the depot of Beijing Metro Line 19 is beside Xingong station, we can transform the system as shown in Figure 5-5.

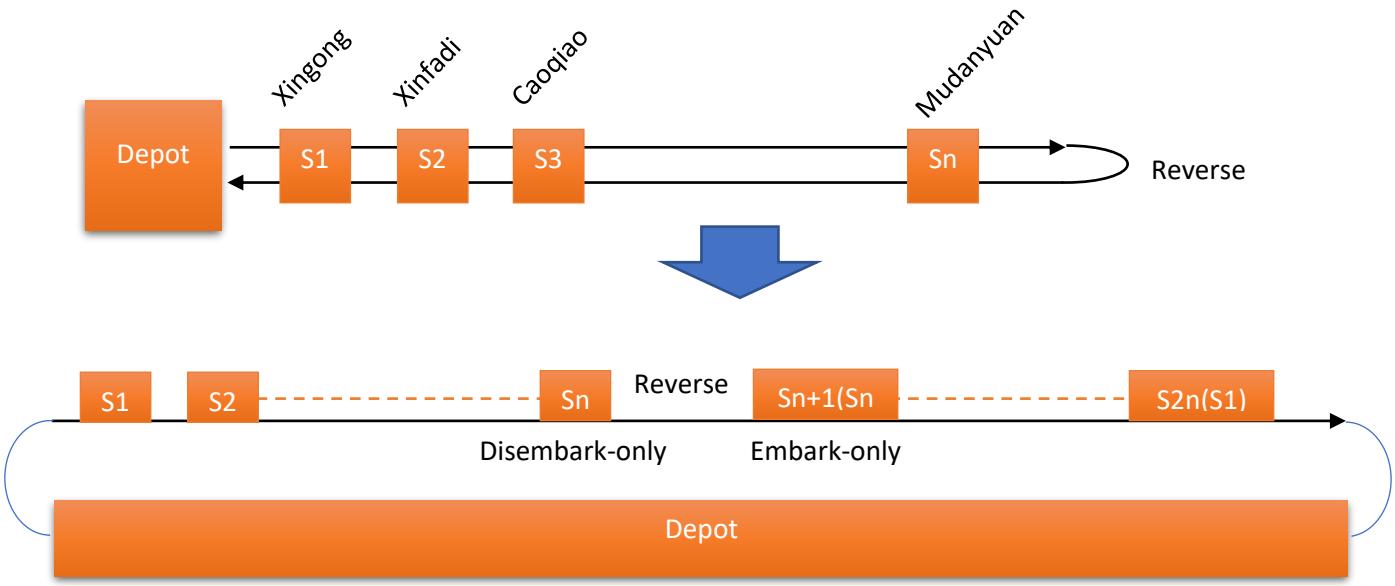


Figure 5-5 Transformation of a double-direction system to a single-direction system

The infrastructure data of this case study are shown in Table 5-1. The research time horizon is divided into three shorter time horizons to simulate variation of objectives in real-life operation, which are the peak hour horizon from 0 to 60 minutes with huge passenger flow, the off-peak hour horizon from 61 to 120 minutes with normal passenger flow and the system-ending hour horizon from 120 to 180 minutes with fewer passengers. This 10-station double-direction system will be considered as a 20-station single-direction system. For each train, there are two optional dwelling strategies and two optional departure strategies.

Parameter	Data
Passengers entering research time horizon	180 minutes
Objectives	Balance the passenger waiting time and the passenger travelling time from 0 to 60 minutes; Balance the passenger waiting time and the train full load rate from 60 to 120 minutes; Take all passengers to their destination with the shortest waiting time from 120 to 180 minutes.
Number of stations	10 stations for double-direction
Number of test trains	28 trains, 29 trains, 30 trains
Minimum headway	2 minutes
Capacity of each train	2520 people
Optional dwelling time	Short dwelling: 0.5 minutes, Long dwelling: 1.5 minutes
Optional departure interval	Short interval: 4 minutes, Long interval: 5 minutes
Real-time passenger flow entry rate updating period	15 minutes
Reverse time	4 minutes

Table 5-1 Infrastructure parameters of the case study

Before the real-time operation, the GA_POSM method will schedule and calculate optimised scheduling for a 28- to 30-train system based on the assumed historical passenger flow entry rate data as shown in Figure 5-6. Generally, passengers can only enter the system later than the system start and stop entering earlier before the system close. The test results are shown in Table 5-2.

Number of vehicles in system	CSQI
28	N/A (cannot take all passengers)
29	16887
30	14348

Table 5-2 Test result for different number of vehicles in a system

From above results based on historical data, a 30-train system will be chosen for the real-time operation. Before the operation starts, 30 different historical passenger flow data will be optimised by GA_POSM; the results will be the learning input for the decision tree algorithm.

In the real-time operation, passenger flow entry rate will be updated and transferred every 15 minutes. Different to the optimisation based on GA_POSM, the decision tree

algorithm in RPOM can rapidly make operational strategy decisions according to the latest updated passenger flow entry rate data without the impacts from infrastructures. The assumed real-time variation of passenger flow entry rate data is shown in Figure 5-7.

Compared with the historical passenger flow data, real-time passenger flow varies slightly from 15 to 60 minutes. From 60 to 90 minutes, passenger flow increases and decreases dramatically; it stays the same as the historical data from 105 to 150 minutes then increases again from 135 to 150 minutes.

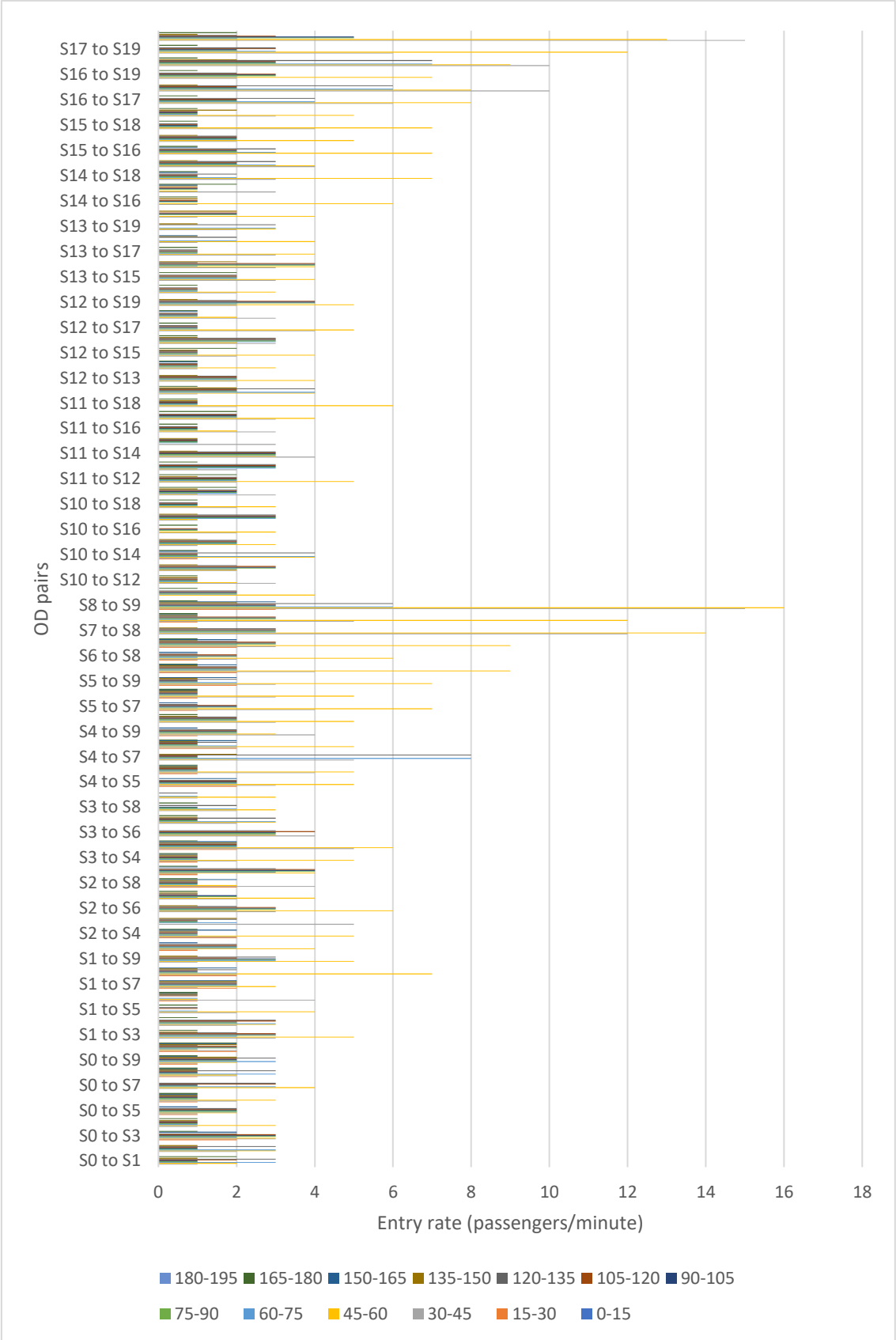


Figure 5-6 Assumed historical passenger flow entry rate OD data

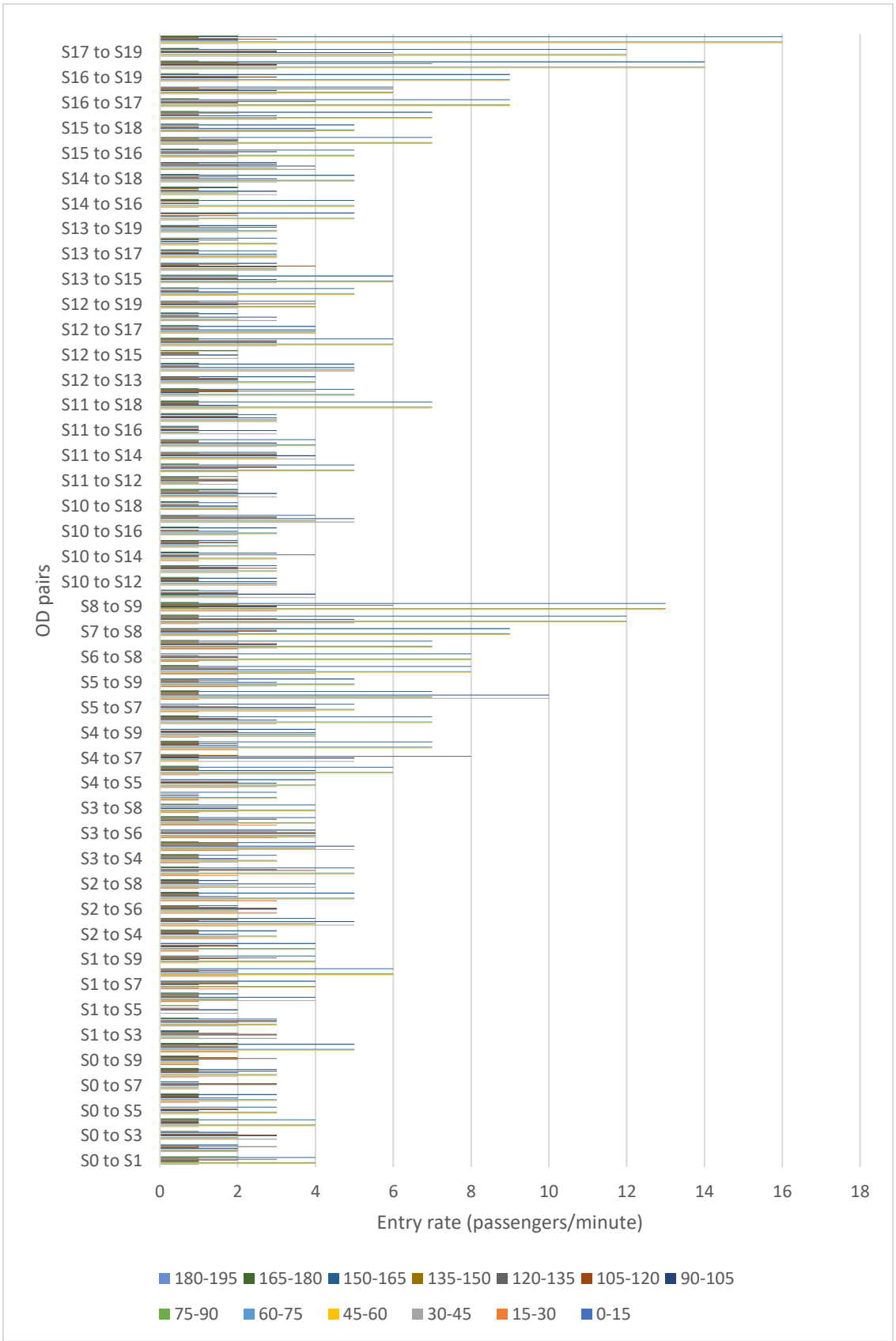


Figure 5-7 Assumed real-time passenger flow entry rate OD data

Because of the large number of results, the operation strategies based on RPOM and the comparison of the CSQI from this method and periodic timetables are shown in Appendix D.

From the table, the RPOM method-based GA_POSM data and decision tree algorithm can optimise and modify any later operation strategy for both dispatched trains and follow-up trains after every detection period, and the calculation time is negligible. In the 180-minute research time horizon, the operation strategy optimisation process has been implemented 7 times and 183 operation strategies have been modified and rescheduled. Typically, these modifications were implemented after noticeable passenger flow variations happened, especially when a sudden massive passenger flow occurred. In most cases, these modifications are better than the old operation strategies. However, because the decision tree algorithm has been applied, we cannot ensure the modification is always better, such as the situation at 45 and 90 minutes, where the CSQI of the modified operation strategy is a slightly larger than the old strategy but compared with worse results of periodic strategies these flaws can be accepted. Furthermore, unlike periodic strategies, RPOM keeps modifying its operation strategies in real time, increasing the system's flexibility. With a high degree of flexibility, in most situations, the performance of RPOM is much better than the periodic timetables which can be reflected by the value of CSQI. Also, RPOM can always improve the efficiency of the trains; by using a short time periodic strategy, the system needs at least 40 trains to cover the research time horizon. Moreover, as there are always passenger flow variations in real-life operation, a flexible operation method is more valuable than a perfectly prepared fixed timetable.

In the view of metro operators, with a fixed number of trains, compared with periodic strategies like ST and LT, there is no significant increase in train running costs with

RPOM, and it is easy to apply in practice. Of course, extra operational training for the metro operators and drivers is necessary. However, compared with the experience-based operation method, more accurate indicators can be provided to the operators and drivers by RPOM.

Chapter 6. Conclusions and Future Work

6.1 Conclusions

In this research, a systematic methodology has been proposed for modelling and optimising metro systems' scheduling strategies based on dynamic passenger flow demands. The methodology has also been derived to generate real-time passenger flow-oriented operation decisions without timetables. The modelling process, innovative solution algorithms, performance evaluation, derivation process and macroscopic case studies have been presented in this paper. The aim of this research is to propose a more flexible method to help metro operators make decisions to satisfy objectives based on real-time passenger flow.

First, to better understand the metro scheduling problems based on dynamic passenger flow, a nonlinear integer programming mathematical model, passenger flow-oriented scheduling model (POSM) is proposed.

To translate the proposed methodology into real-life implementation, a field study was carried out in the London Underground Bakerloo Line Operation Department. Based on the field study, we investigated the essential requirements for real-time metro operation, collected the main objectives of daily operation and modified the objective function according to the CSQI to meet different objectives in different operational times.

To solve the dynamic passenger flow-oriented scheduling problem, firstly, a macroscopic metro-passenger simulator has been built based on the objective function to simulate and record the interaction between passengers and metro systems. Then, an innovative

algorithm GA_POSM, based on genetic algorithms has been proposed for solving the dynamic passenger flow demand scheduling problem; by integration with the simulator, GA_POSM can solve the dynamic passenger flow-oriented scheduling problem efficiently and even optimise the scheduling strategies for trains that have already been dispatched in real time.

In addition, a systematic approach to evaluate the performance of GA_POSM has been introduced; two typical scenarios based on data from Beijing Metro Line 19 and a stochastic scenario based on Poisson distribution have been evaluated for both the dynamic passenger flow oriented scheduling function and the real-time optimisation function of GA_POSM, and the results have been compared with two typical periodic scheduling strategies. Based on the objective function in our mathematical model, compared with traditional periodic scheduling strategies, for all the cases, the SQI can be significantly reduced by GA_POSM. And for real-time variation of passenger flow, the real-time optimisation function of GA_POSM can modify scheduling strategies rapidly. In this case study, the average computation time of the GA_POSM algorithm is around 20 to 25 seconds, making it possible to support metro operators to make operational strategies in real time.

Based on the field study, in real-time operation, the computation time requirement is very strict; for different systems, there will be huge variations in the number of variables, which will highly impact the computation time of GA_POSM as it is based on a genetic algorithm. Thus, a decision tree algorithm which can propose decisions rapidly has also been introduced and integrated with GA_POSM. A real-time passenger flow-oriented operation method without timetable, RPOM, has been proposed.

Finally, the system architecture and infrastructure requirements have been introduced to integrate all the proposed methodologies together and demonstrate the data flow in the system. A macroscopic case study which can show implementation of the proposed methodology in real-time passenger flow-oriented metro operation without timetables has also been presented.

6.2 Limitations and Future Work

At this stage, the author has focused on modelling and solving the passenger flow-oriented scheduling and real-time optimisation without timetable for metro systems, there are still some limitations which can be optimised in the future work. To extend this research, further tasks are suggested:

(1) Further evaluation of the RPOM operation method for more scenarios will be useful before implementation. Other optimisation algorithms are worth comparing with the algorithm.

(2) As the Beijing Metro Line 19 is still under construction, a limitation of this research is that non-real-life passenger flow data were collected for the test from the metro line. It will be worth collecting passenger flow data after the Beijing Metro Line 19 is completely open to the public and designing more test instances for the proposed method.

(3) The methodology presented in this thesis has been demonstrated in a laboratory environment with a programming simulator. With the limitations of mathematical modelling and software simulating, there are still some requirements from metro operators in real life that have not been considered and modelled. A practical test

of the methodology could be necessary, and the methodology proposed in this thesis should be able to guide metro operators to modify operation strategies in real metro systems in future.

(4) More optional dwelling time and departure interval choices can by considered for further research.

(5) Future uncertainty in the operation can be considered, some discount factors may be applied.

(6) Other machine learning method, including supervised learning, unsupervised learning and reinforcement learning could be considered and compared for real time operation to improve the service quality.

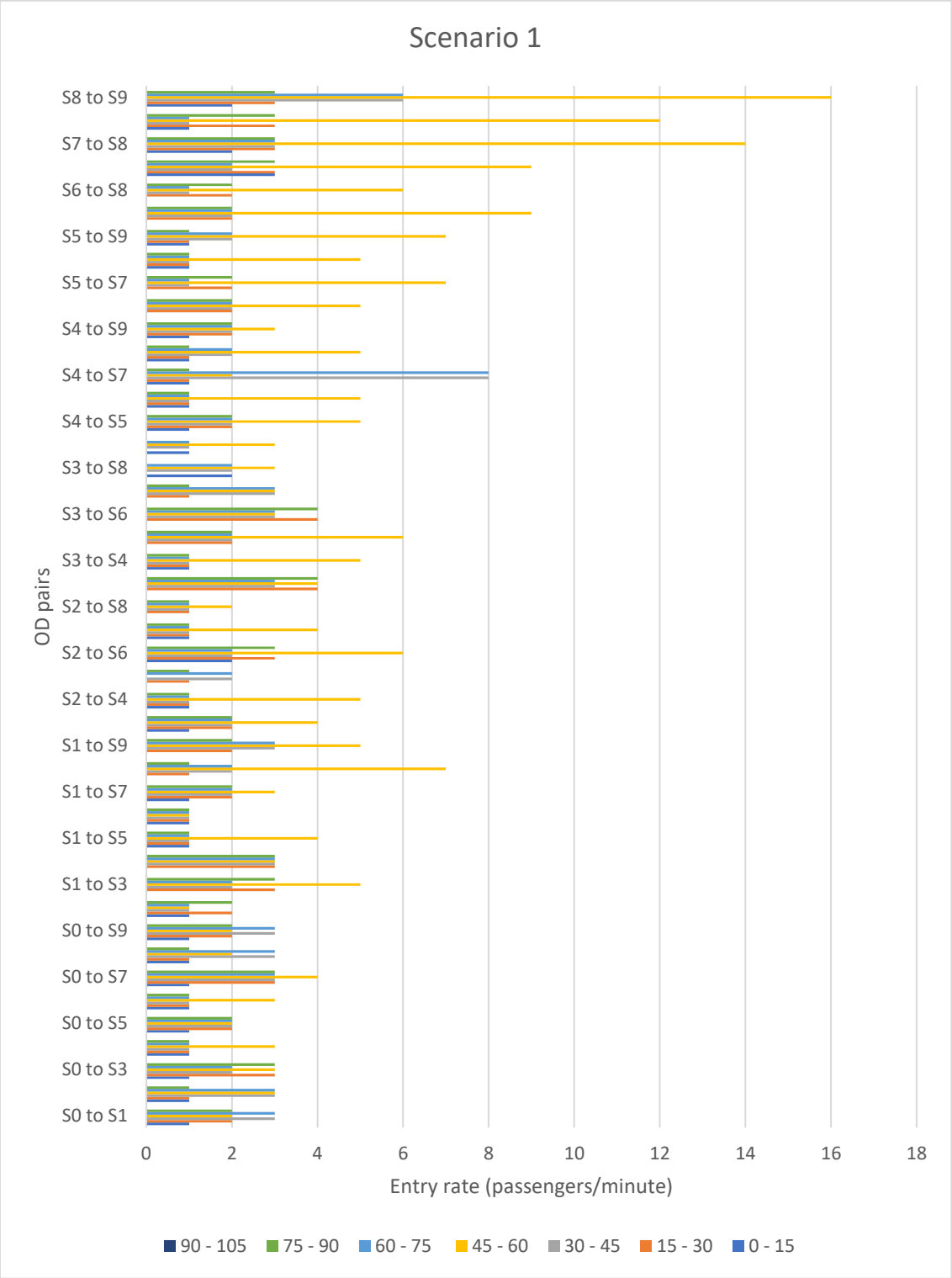
(7) Further research based on different passenger flow data updating resolution without technical limitations will be useful to investigate its impacts to the results.

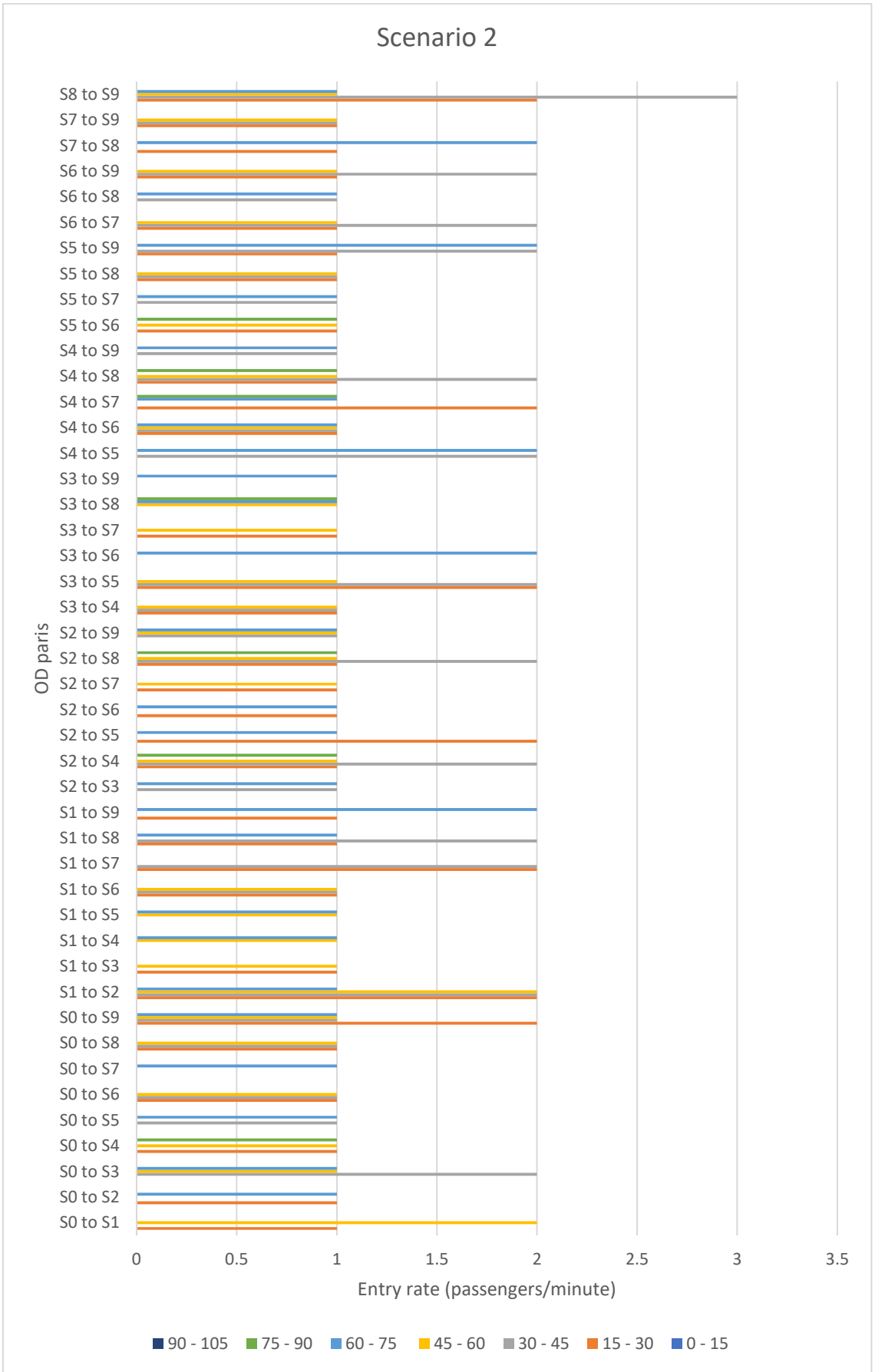
(8) Instead of using real-time data to refresh the statistical passenger flow data in operation, some prediction methods for passenger flow are worth to be considered.

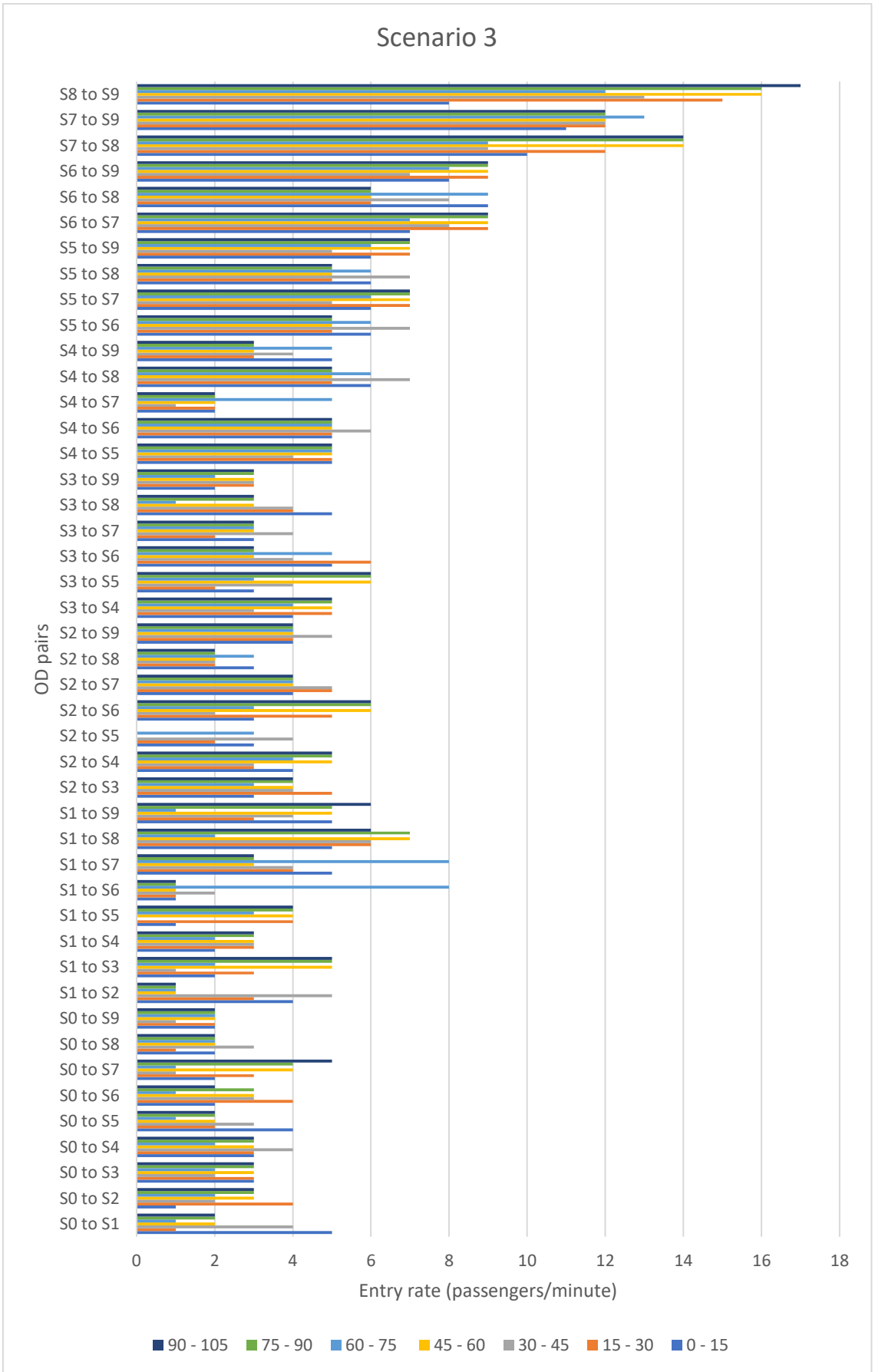
Appendix A. Publication During PhD Research

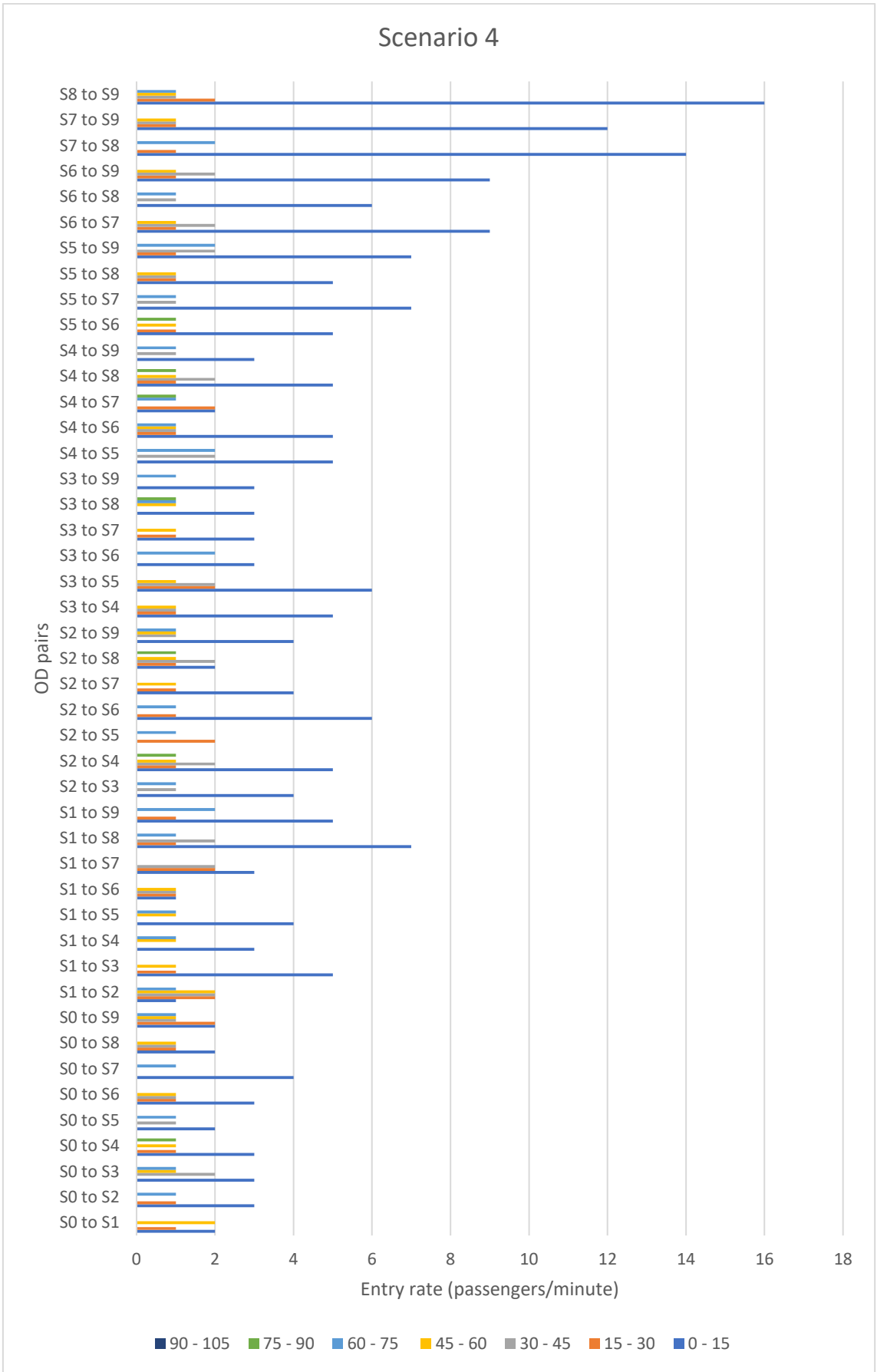
He, L., Chen, L., Liu, J., Roberts, C., Yu, S. and Feng, X. (2022) Passenger flow-oriented metro operation without timetables. Applied Sciences, 12: 4999. https://doi.org/10.3390/app12104999

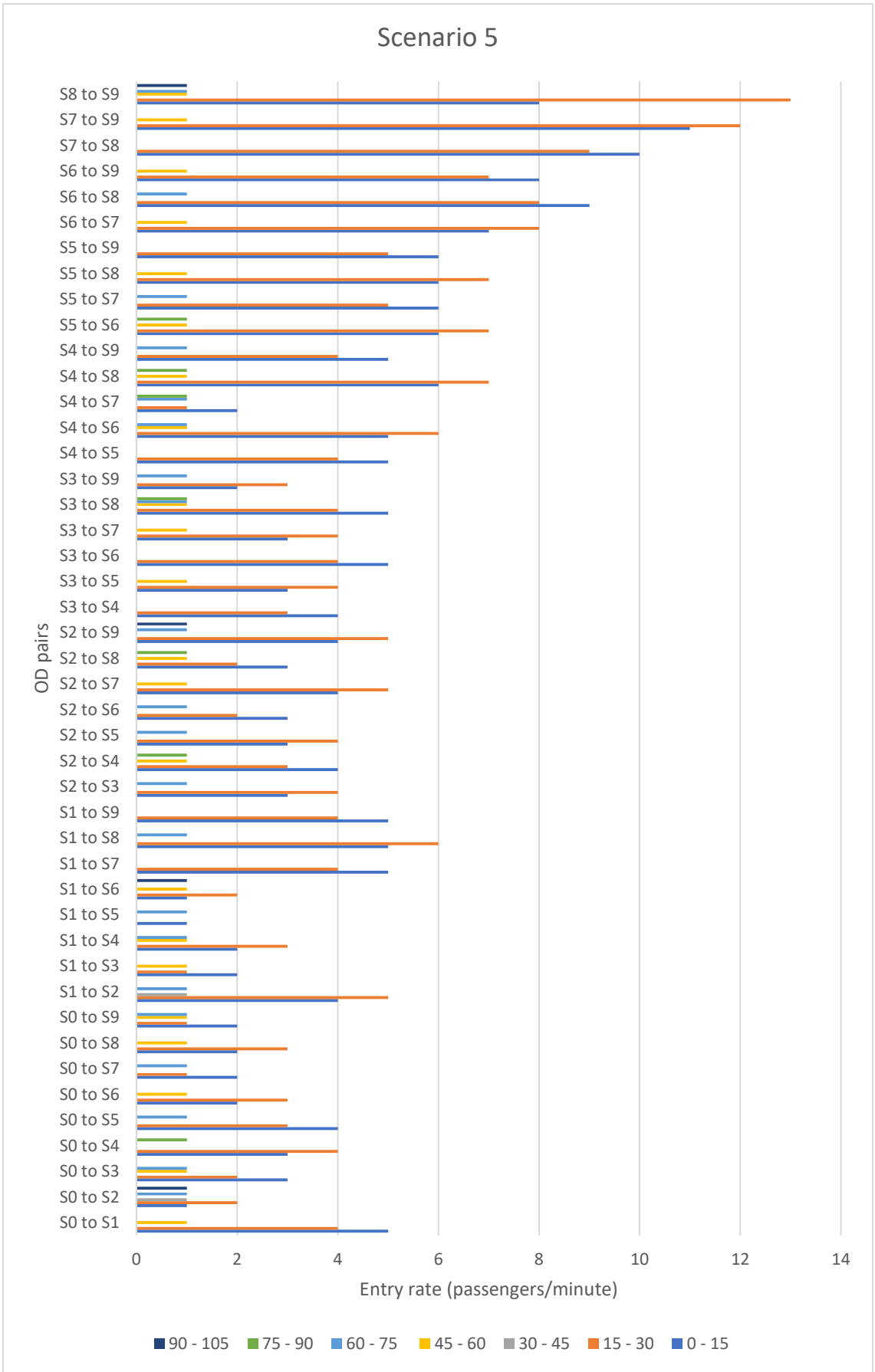
Appendix B. Passenger Flow Scenarios for Scheduling Evaluation

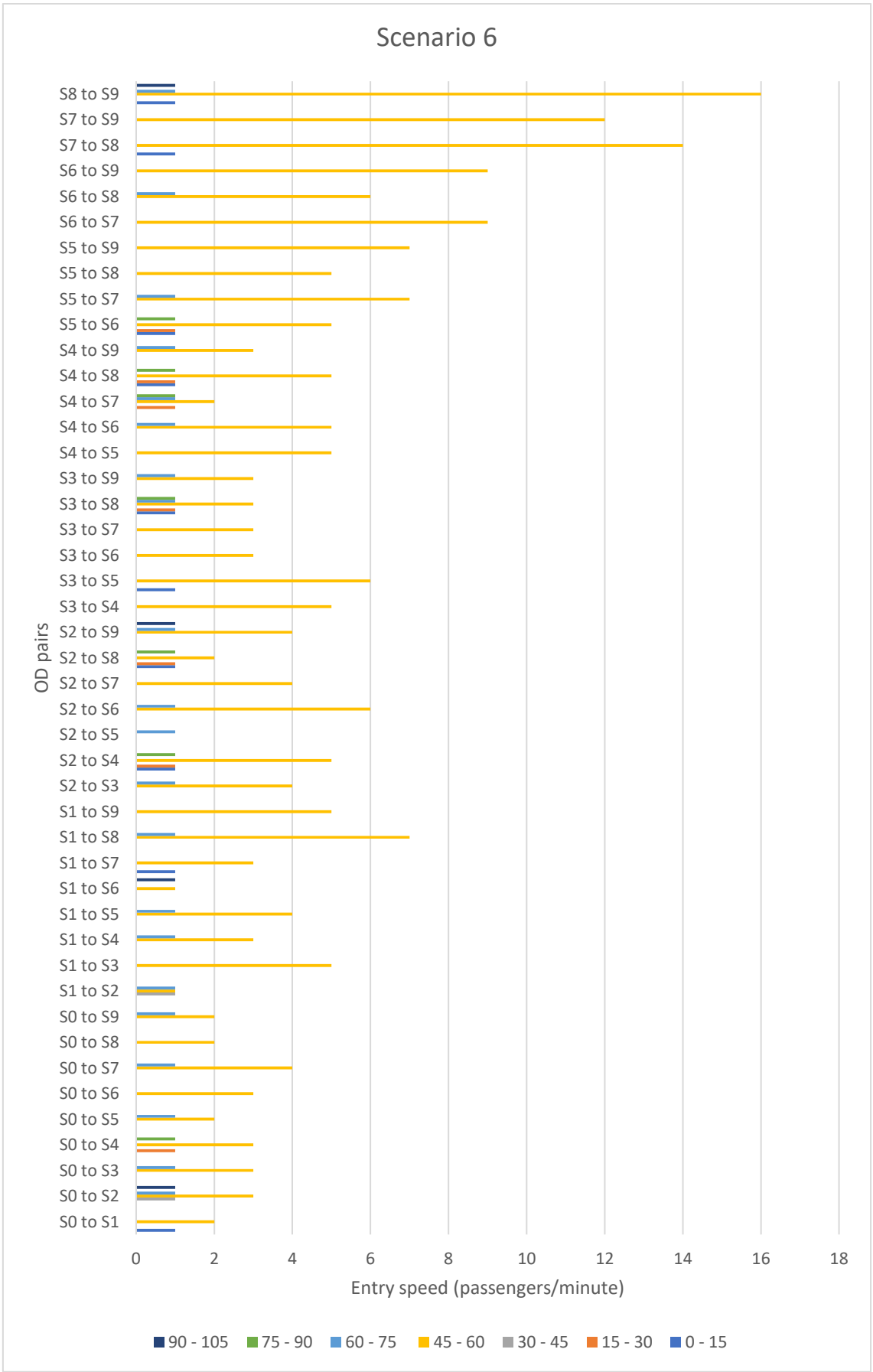


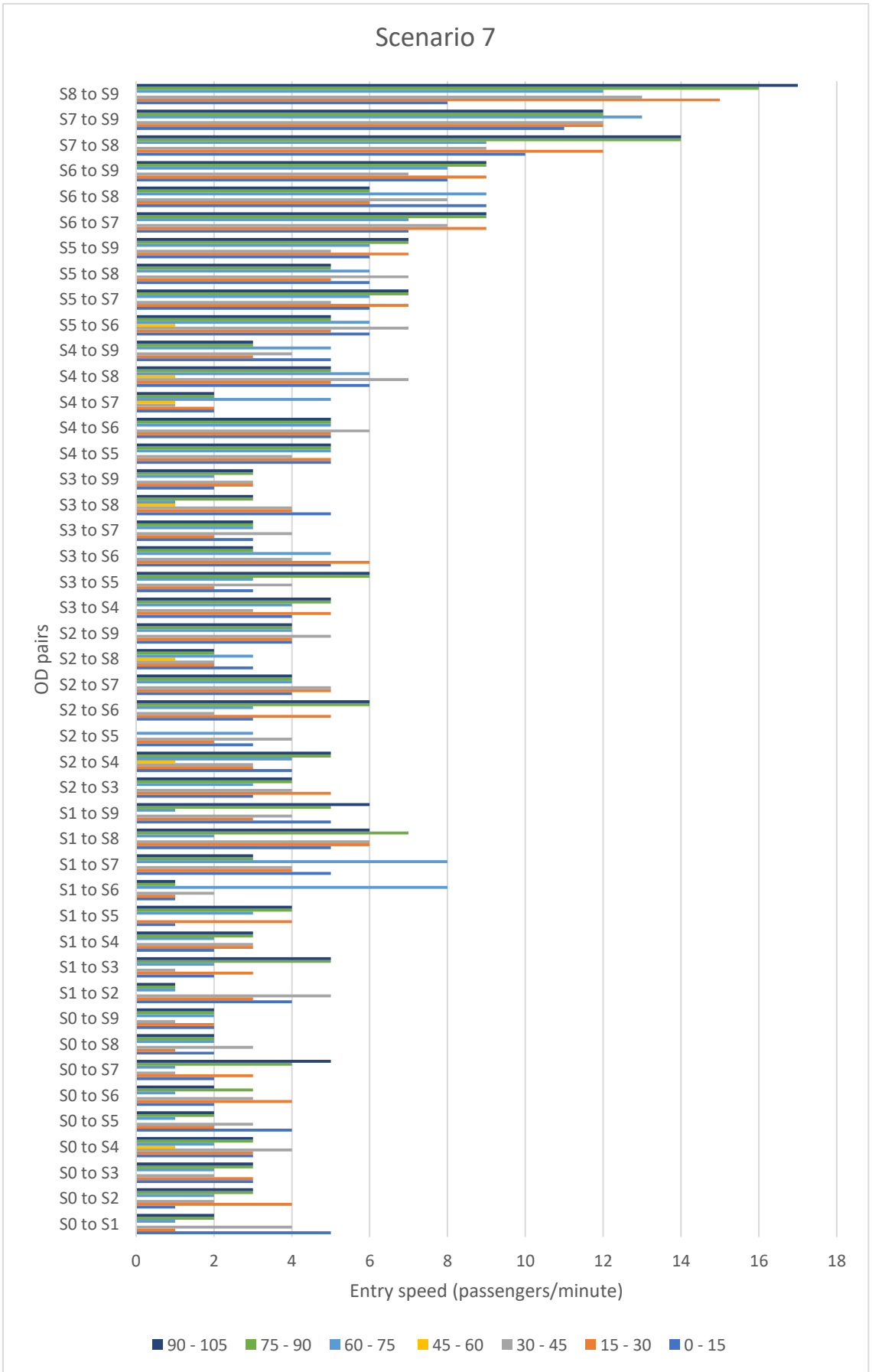

Scenario 2
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
OD paris
0
0.5
1
1.5
2
2.5
3
3.5
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

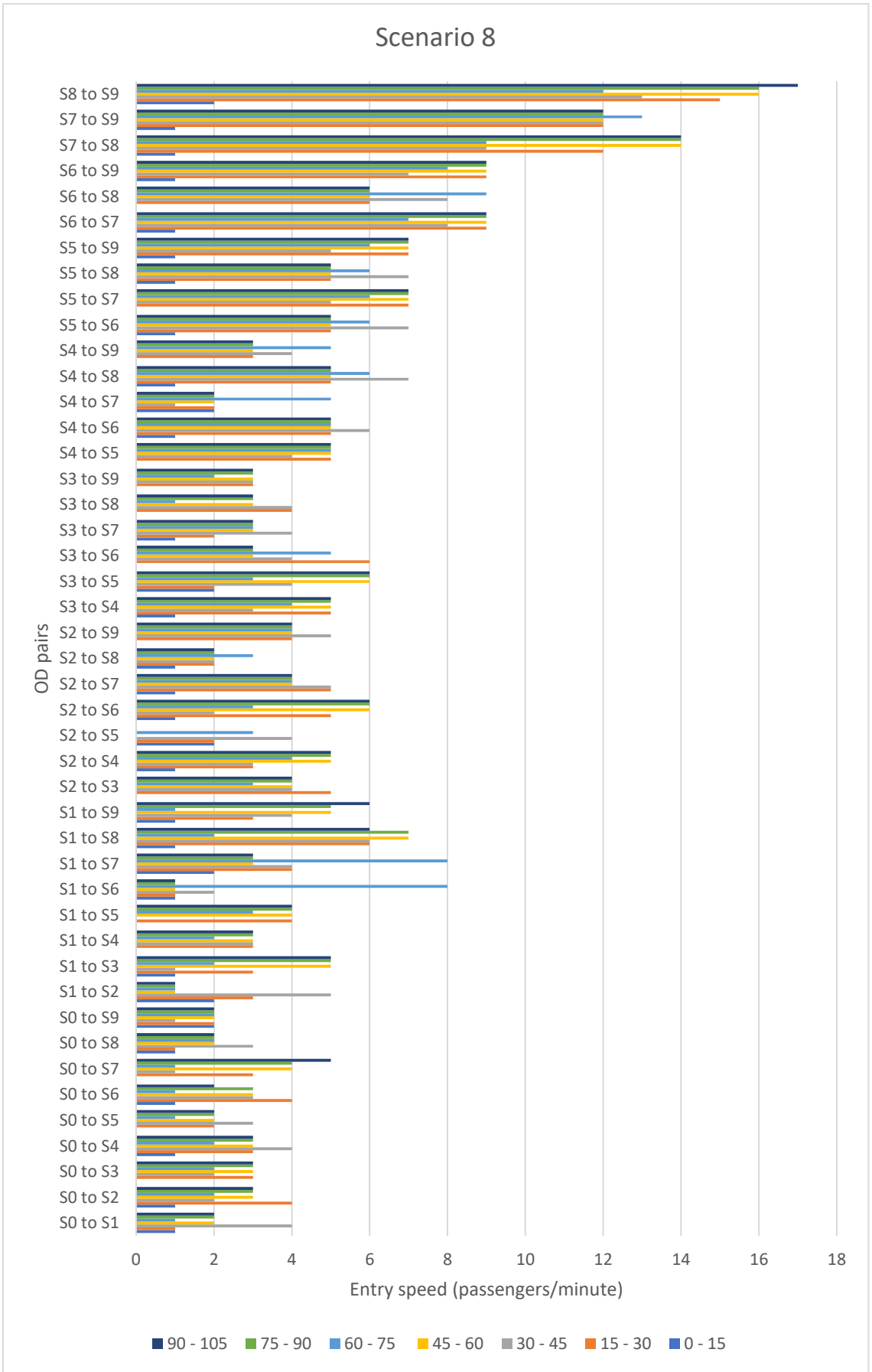

Scenario 3
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
OD pairs
0
2
4
6
8
10
12
14
16
18
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

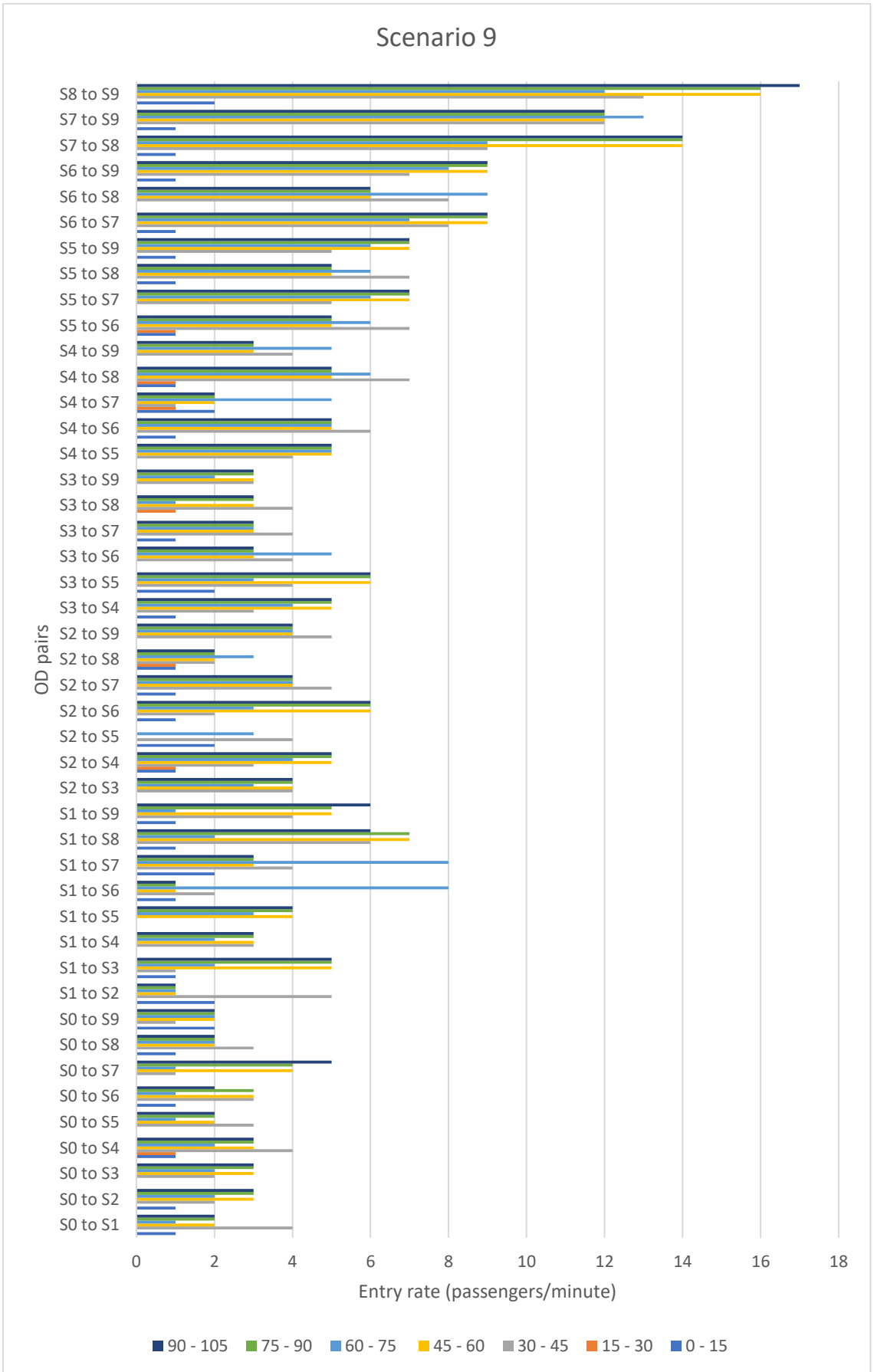

Scenario 4
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

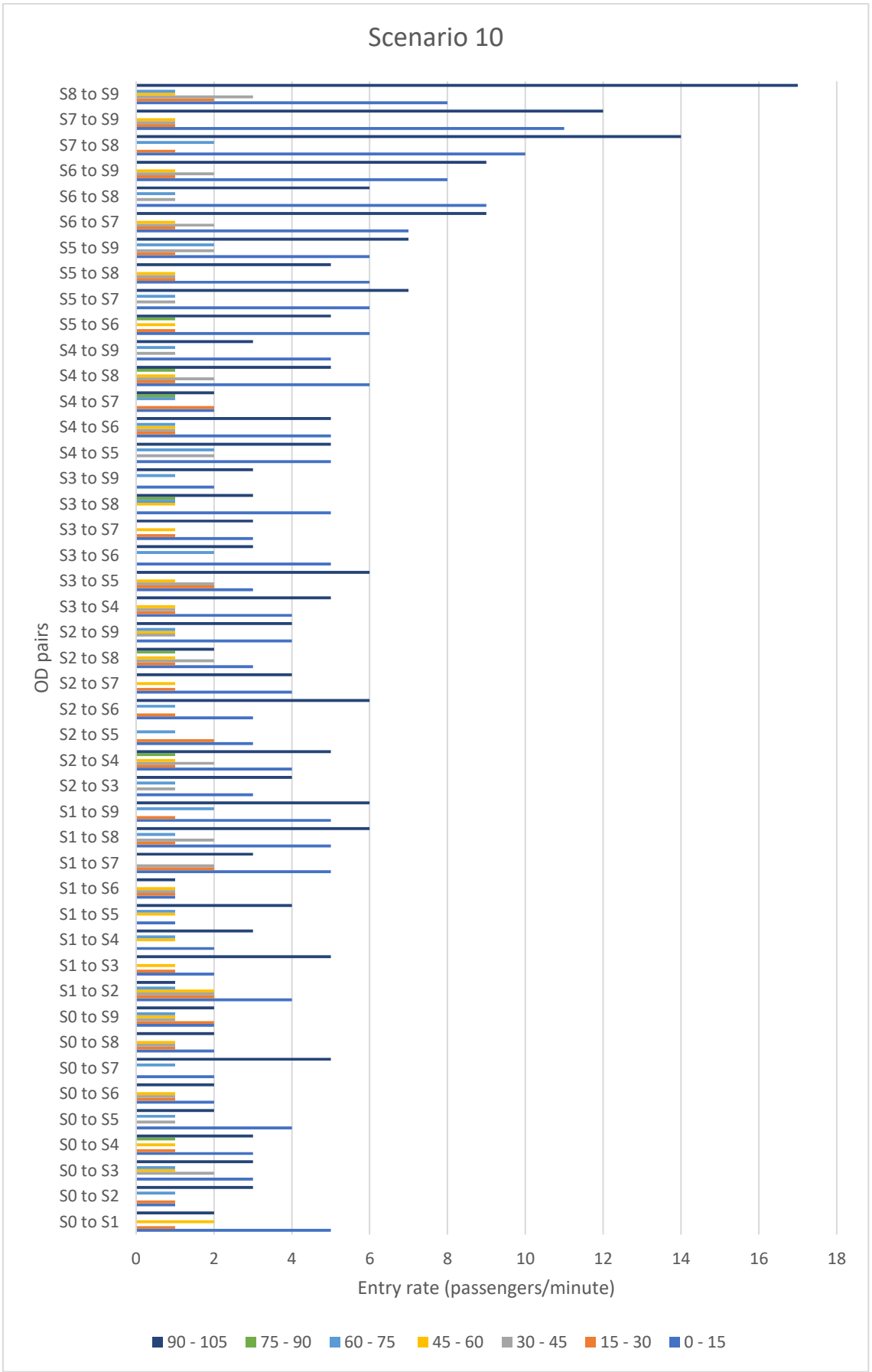

Scenario 5
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15


Scenario 6
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

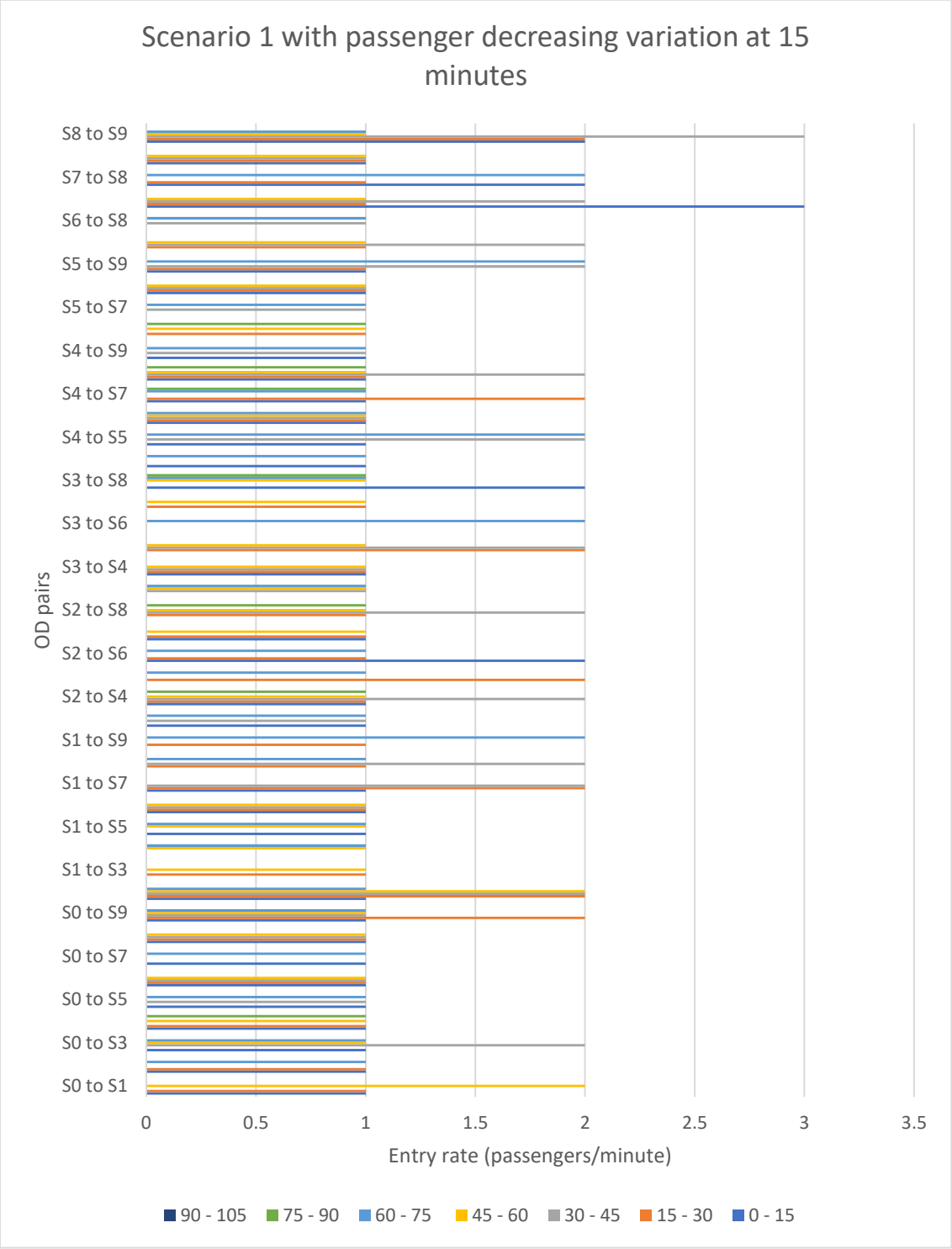

Scenario 7
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

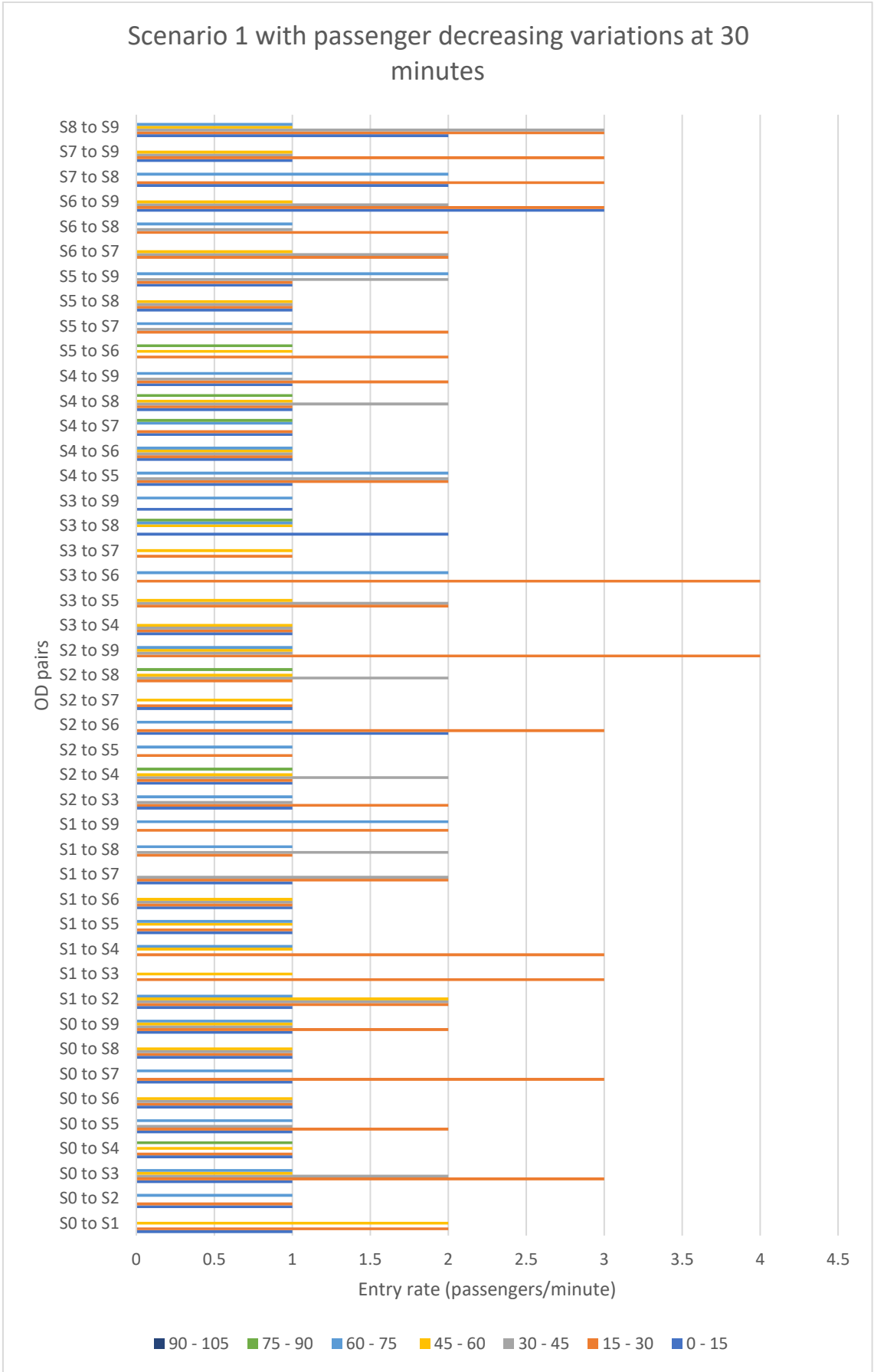

Scenario 8
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

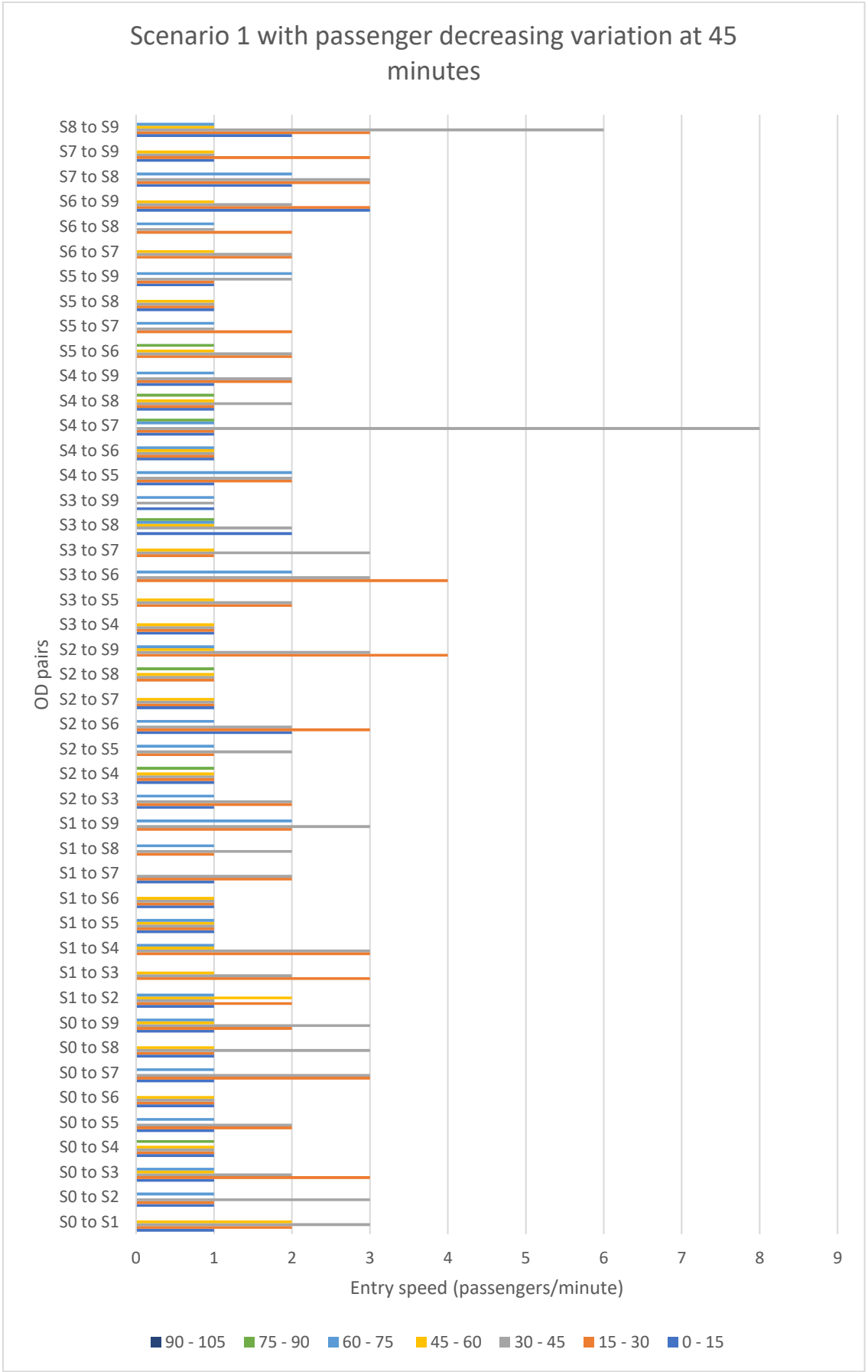

Scenario 9
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

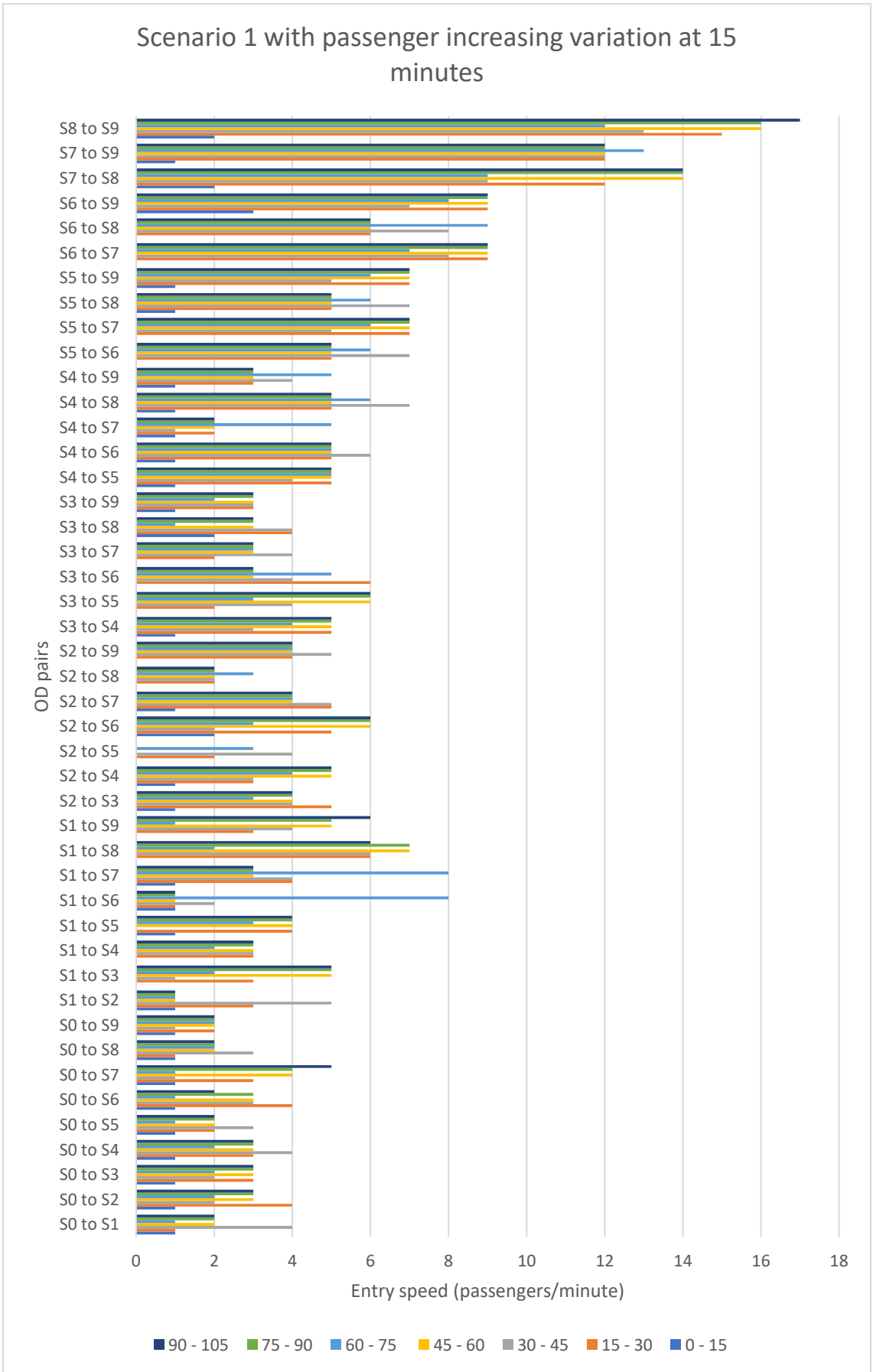

Scenario 10
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
OD pairs
0
2
4
6
8
10
12
14
16
18
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

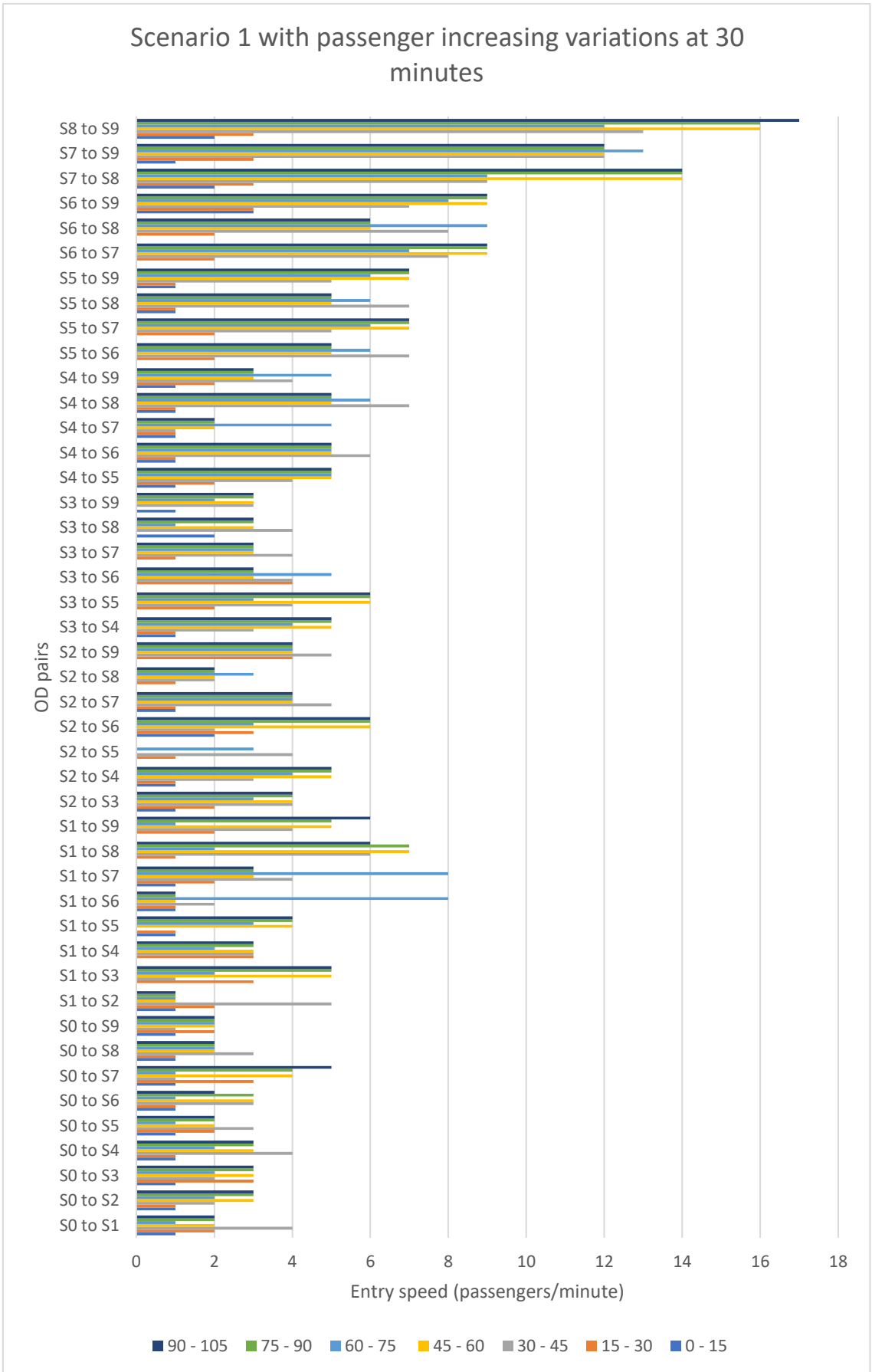
Appendix C. Passenger Flow Irregular Variation Scenarios for Real-Time Optimisation Evaluation

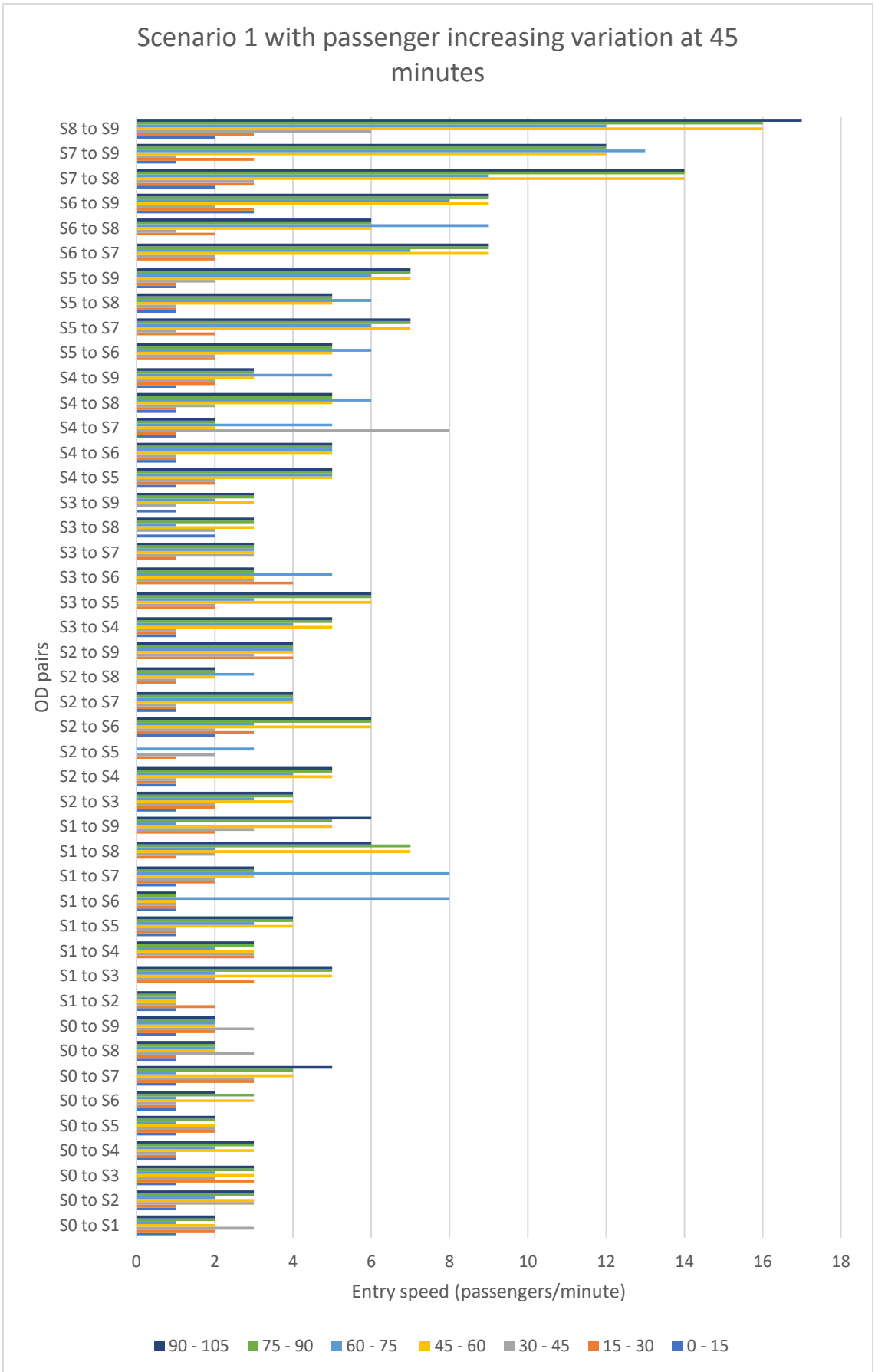


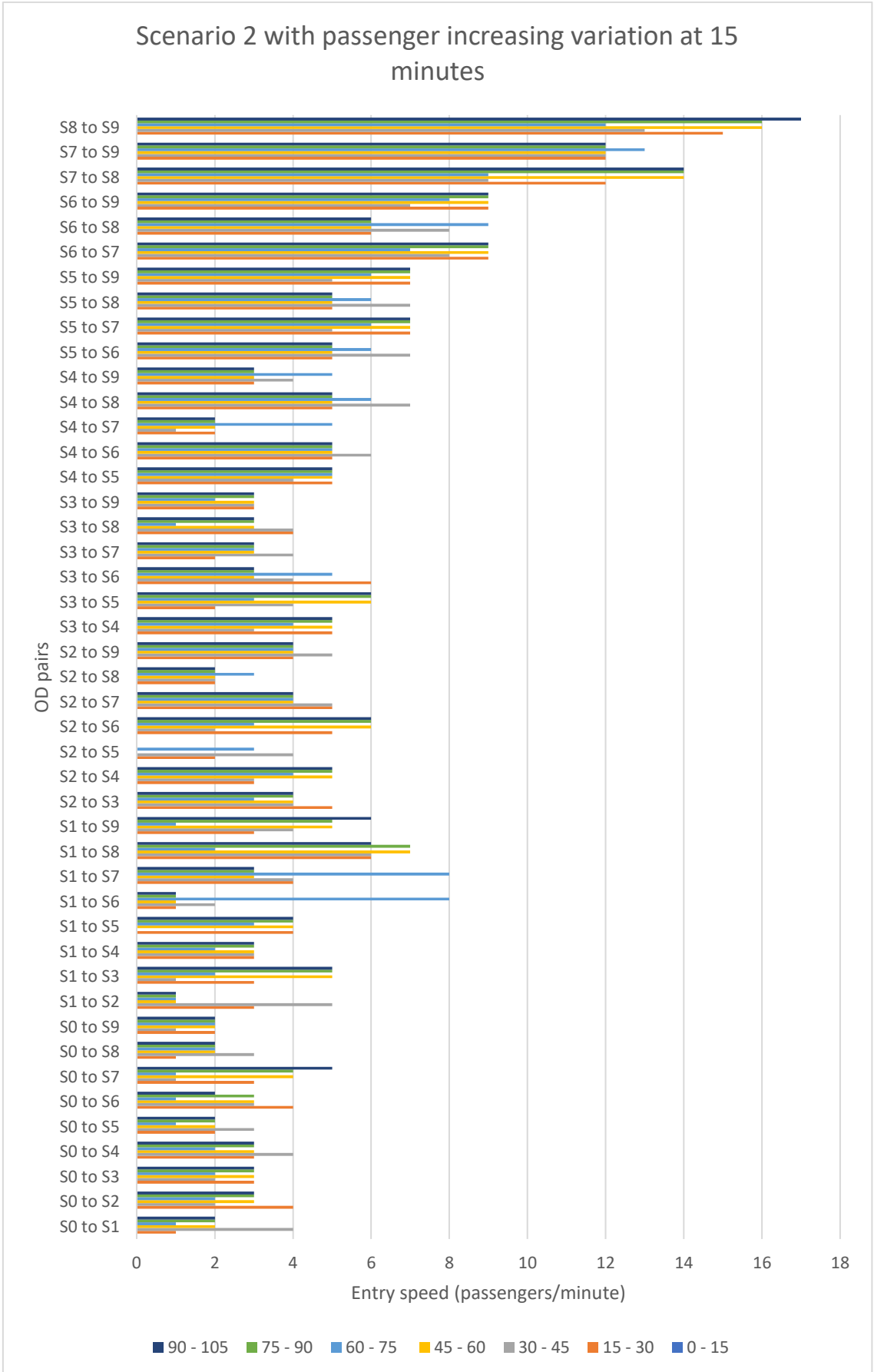

Scenario 1 with passenger decreasing variations at 30 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

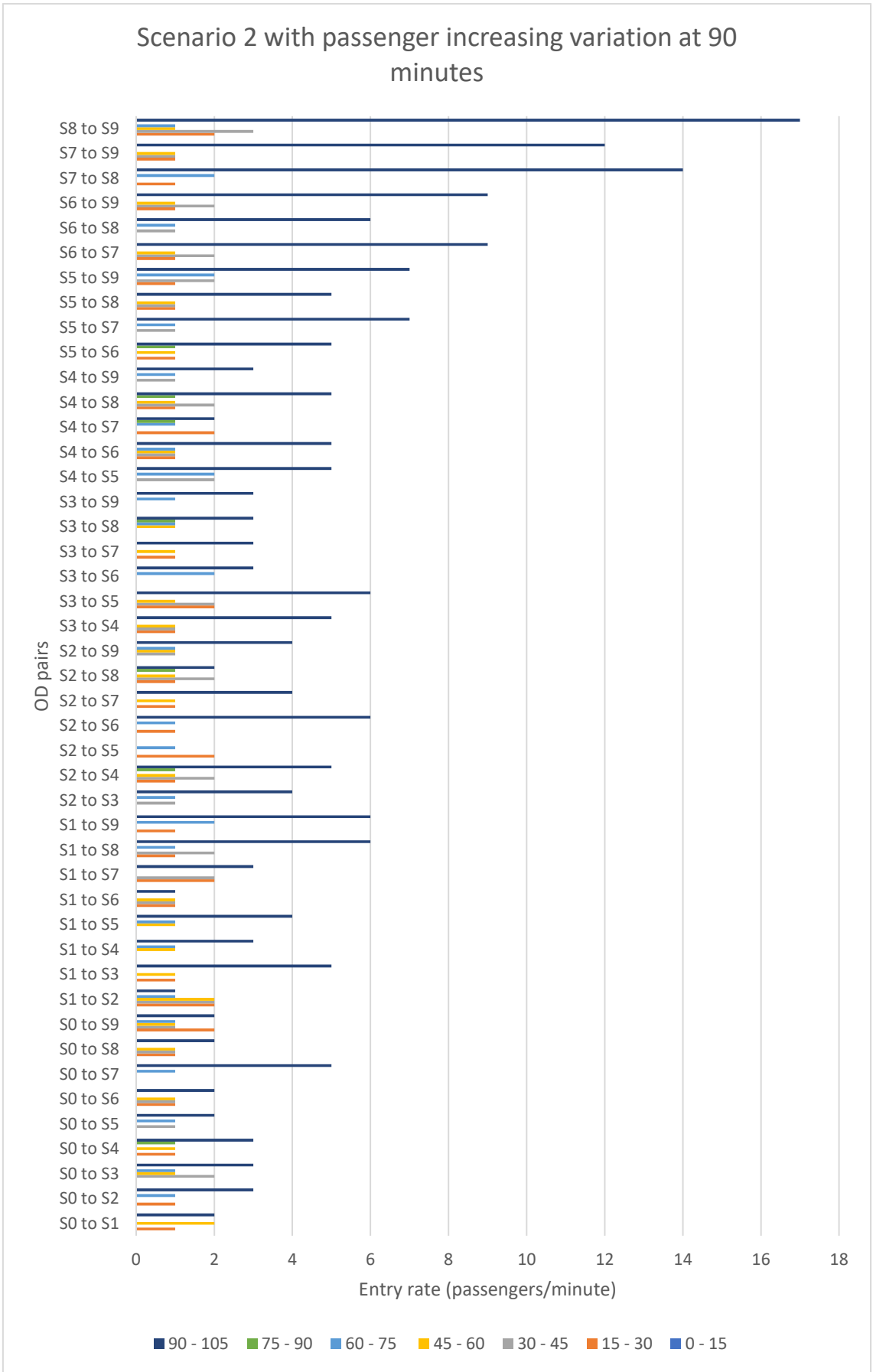

Scenario 1 with passenger decreasing variation at 45 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
1
2
3
4
5
6
7
8
9
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

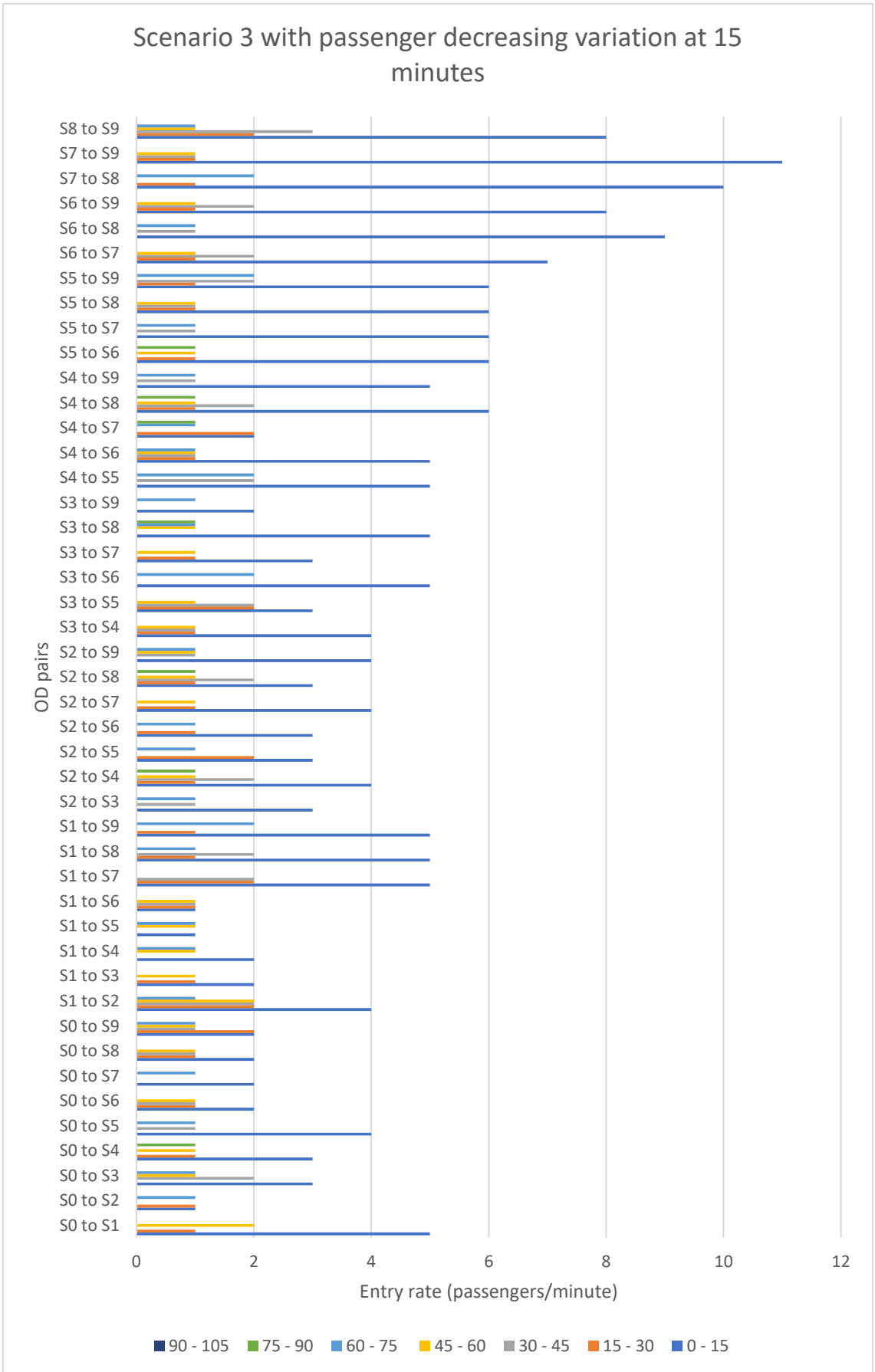

Scenario 1 with passenger increasing variation at 15 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15


Scenario 1 with passenger increasing variations at 30 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15


Scenario 1 with passenger increasing variation at 45 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15


Scenario 2 with passenger increasing variation at 15 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry speed (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15


Scenario 2 with passenger increasing variation at 90 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
14
16
18
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15


Scenario 3 with passenger decreasing variation at 15 minutes
OD pairs
S8 to S9
S7 to S9
S7 to S8
S6 to S9
S6 to S8
S6 to S7
S5 to S9
S5 to S8
S5 to S7
S5 to S6
S4 to S9
S4 to S8
S4 to S7
S4 to S6
S4 to S5
S3 to S9
S3 to S8
S3 to S7
S3 to S6
S3 to S5
S3 to S4
S2 to S9
S2 to S8
S2 to S7
S2 to S6
S2 to S5
S2 to S4
S2 to S3
S1 to S9
S1 to S8
S1 to S7
S1 to S6
S1 to S5
S1 to S4
S1 to S3
S1 to S2
S0 to S9
S0 to S8
S0 to S7
S0 to S6
S0 to S5
S0 to S4
S0 to S3
S0 to S2
S0 to S1
0
2
4
6
8
10
12
Entry rate (passengers/minute)
90 - 105
75 - 90
60 - 75
45 - 60
30 - 45
15 - 30
0 - 15

Appendix D. Operation strategies generated by RPOM

Detection time		0 minutes		
Strategy		RPOM	ST	LT
CSQI		14348	N/A	17419
	DI	DW1 to DW20		
T1	X	LLLSLLSSLSSLSSLSLLSL		
T2	L	LSLLLLSSLSSLSLLLLSLL		
T3	L	LLLSLSLLLSSSLLSSLLLL		
T4	L	LLLLLLLLSLLLLLLLLLSL		
T5	L	LLLLLLLLLLLLLSLLLSLL		
T6	L	LLLLLLLLLLLLLLLLLLLL		
T7	L	LLLSLLLLLLLLLLLLLLL		
T8	L	LLLLLLLLLLLLSLLLLLLL		
T9	L	LLLLLLLLLLLSLLLLLLLL		
T10	L	LLLLLLLLLLLLLLLLLLSL		
T11	L	LLLLLLLLLLLLLLLLLLLL		
T12	L	LLLLLLLLLLLLLLLSLLLL		
T13	L	LLLLSLLLLLLSLSLLLLLL		
T14	L	LLLLLLLLLSSLLLSLLLLL		
T15	L	LSLLLLLLLLLLLSLLLLLL		
T16	L	LLLSLLLLLLSSLLLLSLSL		
T17	L	LLLSLLLLLLLLLLLLLLSL		
T18	L	LLLLLLLLLLLLLLSLLLSL		

T19	L	LLLSLLSLLSLLLLLLLLL
T20	S	LLLLSLSSLSSLLSSLSLLL
T21	S	LLLLLLLLSLSLLLLLLLL
T22	S	LLLLLLLLLLLLLLSLLLL
T23	L	SSLLLLSLLLLLLSLSLLLL
T24	L	LLLLSSSLLLLLSLLLLLLL
T25	L	LLLLSSLLLLLLLLLLLLL
T26	L	LLSLLSLLLSSLLLSLLSLL
T27	L	LLLSLLLLLSLLLLLLSLL
T28	S	LLSLLLLLSLSSLSLSLSLL
T29	L	LLLLLSSSLLLSSLLLLSLL
T30	L	LLLLLLSLSLSLLLLSSLLL

Detection time	15 minutes			
Strategy	RPOM	Last RPOM	ST	LT
CSQI	14234	14472	N/A	17547

	DI	DW1 to DW20
T1	X	LLLSLSLSLLLSSLSSSSSL
T2	L	LSLSLLSLLLSSSSSSLSLL
T3	L	LSLLSLSLSSSLSSLLLSSL
T4	L	LSSSSSLLSSLSSSLLLLSL
T5	L	SLLSLLLSSLLSSLSSLSSL
T6	L	LLLLSLSLSSLLSLLSLLSL
T7	S	LLSLSSLLLLSSLLLSLLLL

T8	L	LSLLSLSSSLSLSSSLLLLL
T9	S	SLSSSLLLSSLSLLSLSLLL
T10	L	LLLLLLLLSSLLSSLLLLLL
T11	S	LLLLLLLLSSLLLLLSSSLL
T12	L	LLLLLLSLLLSSSLSLLLLL
T13	L	LLLLLSSSLLSLLSLSSSSL
T14	L	LLSSLLLLLSLLLSSSSLLL
T15	L	LLSLSSSLSLSSSSSLSLSL
T16	S	LSSLLSSSLSLLSLSSSSLL
T17	L	LLSLLSSLSLSSSLSLSLSL
T18	L	LLLLSLSSSLLSLSLLLLLL
T19	L	LSSLSLLSSSLSLSLLLSSL
T20	S	LLLSLLLSLSSLLSLSSLLL
T21	L	LLSSSLSLLLLLLSSSSLLL
T22	L	LSLSLLSSSLSLLSSLSLSL
T23	L	LLLSLSLSSLSSLSLSLLLL
T24	L	LSSSSLSSLSSLSSSLSLSL
T25	L	LLSSSSSLLSSSSSLLLSLL
T26	L	LSSLSSSLLLLSLLLLLLLL
T27	L	LLSSLSSLSSSLLLLSSLSL
T28	L	LSLSSSLLLSLLLLLSSLSL
T29	L	LSSLSSLSSLLLSLSSLLLL
T30	L	LSSLLLLSLSSLSLLSSLSL

Detection time		30 minutes			
Strategy		RPOM	Last RPOM	ST	LT
CSQI		13309	13922	N/A	16352
	DI	DW1 to DW20			
T1	X	LLLSLSLSLLLSLSSSLLSSL			
T2	L	LSLSLLLSLLLSLSSSLSSSSL			
T3	L	LSLLLSLLLSSSLSSSLSLLSL			
T4	L	LSSSLLLSSLLLLSLSSLSLL			
T5	L	SLLLSLSSSLSLSLSLSLLL			
T6	L	LSLLSSLSSLLLSLLSLLLL			
T7	L	LLSSLLSLLSLSLLLSSSLL			
T8	L	LLLLSLSSLSSLLSSSLLLL			
T9	L	SLSSLLSLLSSSLLLLLSLL			
T1S	L	LLSLSLLLLSLLLSSSLSLL			
T11	L	LSSLLLSLSLSLLLLLLLSL			
T12	L	LLLSLLLSLSSLSSSSLLLL			
T13	L	SSSLLSSLLLSLLSLLSSSL			
T14	L	LSSSLSLSLLSLLSSLLLSL			
T15	L	LLLSLSSLSLSLLLSLSLSL			
T16	L	SLLLLLSSSSLLSLLLLSLL			
T17	L	LSSSLSSSLLLSSLSLSLLL			
T18	L	LSLSSSSLLLSSLSSSLSSL			
T19	S	SLSSSSSLSLSLSLLSLSLL			

T2S	L	SLLSSLSSLLLLLLSLSSLL
T21	L	LLSSLSSSSLLSLSSLSLSL
T22	L	LLSLSSLLLSSLLSLLSSLL
T23	S	LLLLSLSSLLSLLSLLLLSL
T24	L	LSSSLSLLLLLSSLLLSSSL
T25	L	SSLLSSLLLLLLLSLLSSLL
T26	L	LLSLSSLLSLLLLLSLLSSL
T27	L	LLLSSSLLSSLSSLLLLLLL
T28	L	LSLLLLSSLSLSLLSSLLSL
T29	L	LLSLLLLSSSSSLLLSLSLL
T30	L	LLLSLLSLLSSLLLSLLLSL

Detection time	45 minutes			
Strategy	RPOM	Last RPOM	ST	LT
CSQI	13668	13271	N/A	16529

	DI	DW1 to DW2S
T1	X	LLLSLSLSLLSLSSLSLSSL
T2	L	LSLSLLSLLSLSLLSSLSSL
T3	L	LSLLSLLSSSSSSLLSLLSL
T4	L	LSSSLLSSLLLSLLLLLLLL
T5	L	SLLLSLSSLLLSSLLLLSLL
T6	L	LSLLSLLLSSLLSSLLLLLL
T7	L	LLSSSSLSLLSSLSSLSSSL
T8	L	LLLSSSLLLLLSSSSSLSSL

T9	L	LSLLSLSSLSSSSLSSLLSL
T1S	L	LLLLSSSSLSLLLLLSSLLL
T11	L	LSLSLSLLSLLSLLLSLSLL
T12	L	LSSSLSSLLSLLSSSLSLLL
T13	L	LLSSSLLSLSSSLSLSLLSL
T14	L	SSLLLSLSSLSSSLLLSLSL
T15	L	LLSLSSLSSLSSSLSSLLLL
T16	L	LLSSLLSLLLSSSSLSSSLL
T17	L	LLLSSLSSSLLLLLLSSSLL
T18	L	LSLLLSSLLSSSSSSLLLL
T19	L	LSSSLLLLLLLSLSLSSLLL
T20	S	LSSLLLSLSLLSLLSSSLLL
T21	L	SSSSSLLLSLSLLLSLSLSL
T22	L	LSSSSSLLSLSSLSLSSSLL
T23	L	LLLSLSSSSSLSSSSLSLSL
T24	S	LSSLSLSLSSSSLSSSSSSL
T25	L	LLLLLSLLSLSLLLSSSSLL
T26	L	LLSSLLSLSLSLSSLLSSLL
T27	S	LLLSSLSLLLSSSLLLSSSL
T28	L	LLLSSLLLLSSSLSSSSLLL
T29	L	LSSSSSLSSSLSLSLSLLSL
T30	L	LSLSSSSLLLLLSLLSSSSL
Detection time		60 minutes

Strategy		RPOM	Last RPOM	ST	LT
CSQI		13006	13054	N/A	16059
	DI	DW1 to DW2S			
T1	X	LLLSLSLSLLLSLSSLSLSSL			
T2	L	LSLSLLLSLLLSLSLLLSSLLLSL			
T3	L	LSLLLSLLLSSSSSLLLLLLL			
T4	L	LSSSLLLSSLLLSLLLSSLSLL			
T5	L	SLLLSLSSLLLLLLLLSLSLL			
T6	L	LSLLLSLLLLSSLLLLSSLSLSL			
T7	L	LLSSSSLSSLSSSLSSLLSL			
T8	L	LLLSSSSSLSSSSLSSSLSL			
T9	L	LSLLLSLLLSSSSSLSLLLLLL			
T1S	L	LLLLLLSLLLSLLSSLSSLLLL			
T11	L	LLLLLSSSSSLLLSLLSSSSL			
T12	L	SLSSLSSSLLLLLSSLSLSSL			
T13	L	LLLSLLLLSSSSSLSSLLLSL			
T14	L	LSSLSLSSLLLLSLSSSLSLL			
T15	L	LSLLLSLLLSSSLSLLLSSSLLL			
T16	L	LLSLSLSLSSSLSLSLLLSLL			
T17	L	LSLLLLSSLSSLLSSSSLLLL			
T18	L	LLSLSLSSLSLLLSSSSLLSL			
T19	L	LSLLLLSSLSSLLLSLLLLLL			
T2S	S	LLLLLSLSLSLLLSLLLLLSL			

T21	L	LLLSLLLSSSLSLSSLLSSL
T22	L	LSSSSLSLLLLSLLLSSSLL
T23	L	LLLSSSLLSSSLSLLLLSLL
T24	L	LLLSSSSLLSSLSLLSSLSL
T25	L	LSSLLSLLLSLLSSSSLSLL
T26	L	LSSSLSLSSSLSSSLSSLSL
T27	L	LSLLSLSLLSSSLLSLSLSL
T28	L	SSLSSLLLLLLSLSSLSLSL
T29	L	LLSSSSLSSLLSSSLLLSSL
T30	S	LSLLSSLLLSSLSLSLLSSL

Detection time	75 minutes			
Strategy	RPOM	Last RPOM	ST	LT
CSQI	13371	13383	N/A	16059

	DI	DW1 to DW2S
T1	X	LLLSLSLSLLSLSSLSLSSL
T2	L	LSLSLLSLLSLSLLSSLLSL
T3	L	LSLLSLLSSSSSSLLLLLLL
T4	L	LSSSLLSSLLLSLLSSLSSL
T5	L	SLLLSLSSLLLLLLLSSLSL
T6	L	LSLLSLLLSSLLLSSLLSLL
T7	L	LLSSSSLSSLSSLSSLLLSL
T8	L	LLLSSSSSLSLSLLLSSLSL
T9	L	LSLLSLLSLLLSLLSLLLLL

T1S	L	LLLLLSLSLLLSSSLLLSLL
T11	L	LLLLSLLSLSLSSLSLLSLL
T12	L	SLLLSSLLLLSLSLSLLLLL
T13	L	LLLSSLLLLSSSLLSSSLLL
T14	L	LLLLSSLSSLSSLLLLSLLL
T15	S	SLLLSLLLLSSLLLLSSSSL
T16	L	LLLLSLLLSLSSLLLSLLLL
T17	L	LLLSLSSSSSLLSLSSLLLL
T18	S	LSSSSSLLLSSLLSLLLSSL
T19	L	SSSLSLSSLLLSSSLSSLLL
T2S	L	LSSSSLSSSLSSLSSSLSSL
T21	L	LSLSSSLLSLSSSLLLLSLL
T22	L	LSLLSLLSLSSSSLSSLLLL
T23	S	LSLLSLSSLSLLSSSLSSLL
T24	L	LLSSLSLLSLLLSLSLLLSL
T25	L	LSLSSLLLSLSSLLLLLLSL
T26	S	LLLSSSLLLLSLSSLSLSSL
T27	S	LSLSLLSLSLLLLSSLSLSL
T28	L	LSLSLLLSLSSLLLLSSSSL
T29	L	SLLLLLSSLLSLLSLSLLLL
T30	L	LSLSLLSLSLLSSSSSLSSL

Detection time	90 minutes			
Strategy	RPOM	Last RPOM	ST	LT

CSQI		13156	13135	N/A	17480
	DI	DW1 to DW2S			
T1	X	LLLSLSLSLLSLSSLSLSSL			
T2	L	LSLSLLSLLSLSLLSSLLSL			
T3	L	LSLLSLLSSSSSSLLLLLLL			
T4	L	LSSSLLSSLLLSLLSSLSSL			
T5	L	SLLLSLSSLLLLLLLSSLSL			
T6	L	LSLLSLLLSSLLLSSLLSLL			
T7	L	LLSSSSLSSLSSLSSLLLLL			
T8	L	LLLSSSSSLSLSLLLSSSSL			
T9	L	LSLLSLLSLLLSSSLSLLLL			
T1S	L	LLLLLSLSLLLSSSSLSLSL			
T11	L	LLLLSLLSLSSLLLSSSLSL			
T12	L	SLLLSSLSLSSSSSLLSSLL			
T13	L	LLLSSLSLSLLLSSSSSSLL			
T14	L	LLLLSSLSSLSSLLLSSSLL			
T15	S	SLLSLLSLLLSSSLSLSSLL			
T16	L	LSSLLLSLLSSLSLSSSSSL			
T17	L	LSLSSLSSSSSLSSSLSLSL			
T18	L	LSLSSLLSSLLLSSSLSSSL			
T19	S	LSSLLLSSSLLLSSSSLSSL			
T2S	L	SLSLSSSSLSLSSLSSSLSL			
T21	L	LLSLLSSLSLLLLSLSLSLL			

T22	L	LLSSLSSSSLSLLLLSSSSL
T23	L	LSSLLLSSLSLSLSSLSSSL
T24	L	LSSSSLSLSLSLLLLLSSLL
T25	L	LSLLSSSLSLSSLLSSLLSL
T26	L	LSSSLLLSLLLSLLSLLLSL
T27	L	LLLLSSSLLLLSLSSSLSLL
T28	L	LSLLLSSSLLLSLSSLLLSL
T29	L	LLLSLLSLSLSSLLSLLSSL
T30	L	LSSLLLLLSLSSSLLSSSLL

Detection time	150 minutes			
Strategy	RPOM	Last RPOM	ST	LT
CSQI	13791	13862	N/A	16828

	DI	DW1 to DW2S
T1	X	LLLSLSLSLLSLSSLSLSSL
T2	L	LSLSLLSLLSLSLLSSLLSL
T3	L	LSLLSLLSSSSSSLLLLLLL
T4	L	LSSSLLSSLLLSLLSSLSSL
T5	L	SLLLSLSSLLLLLLLSSLSL
T6	L	LSLLSLLLSSLLLSSLLSLL
T7	L	LLSSSSLSSLSSLSSLLLLL
T8	L	LLLSSSSSLSLSLLLSSSSL
T9	L	LSLLSLLSLLLSSSLSLLLL
T1S	L	LLLLLSLSLLLSSSSLSLSL

T11	L	LLLLSLLSLSSLLLSSSLSL
T12	L	SLLLSSLSLSSSSSLLSSLL
T13	L	LLLSSLSLSLLLSSSSSSLL
T14	L	LLLLSSLSSLSSLLLSSSLL
T15	S	SLLSLLSLLLSSSLSLSSLL
T16	L	LSSLLLSLLSSLSLSSSSSL
T17	L	LSLSSLSSSSSLSSSLSLSL
T18	L	LSLSSLLSSLLLSSLSLLSL
T19	S	LSSLLLSSSLLLSSSLLSLL
T2S	L	SLSLSSSSLSLLSSSSSLLL
T21	L	LLSLLSSLSSLSSSSLLLSL
T22	L	LLSSLSSSSSSSLLSLLSSL
T23	L	LSSLLSSLLLSSSSLSSLSL
T24	L	LSSLLSSSSLSLLLSSLSSL
T25	L	LSSSLSLSLSSSSLLLLSLL
T26	L	SLLSSLLSSLSLSLLLSLSL
T27	S	LLLLSLLSLSLLSLSLSLSL
T28	L	LSSLSSLSLLLSSSSSLSLL
T29	L	SSSLLSSLSSSSLLSLSSSL
T30	L	SSLSLSLSLLLSSSLLLSSL

Abbreviations: DW + number: dwelling time + station number; DI: departure interval; T + number: train number; S: short time strategy; L: long time strategy; ST: short periodic timetable; LT: long periodic timetable.

References

Adler, I., Resende, M., Veiga, G. et al. (1989) An implementation of Karmarkar's algorithm for linear programming. Mathematical Programming, 44 (1–3): 297–335.

Bian, F. and Wang, X. (2021) School enterprise cooperation mechanism based on improved decision tree algorithm. Journal of Intelligent & Fuzzy Systems, 40 (4): 5995–6005.

Brännlund, U., Lindberg, P., Nõu, A. et al. (1998) Railway timetabling using Lagrangian relaxation. Transportation Science, 32 (4): 358–369.

Broere, W. (2018). Urban Problems - Underground Solutions. ACUUS 2012 Advances in Underground Space Development.

Caprara, A., Fischetti, M. and Toth, P. (2002) Modeling and solving the train timetabling problem. Operations Research, 50 (5): 851–861.

Carey, M. and Crawford, I. (2007) Scheduling trains on a network of busy complex stations. Transportation Research Part B: Methodological, 41 (2): 159–178.

Chen, L. (2012) Real time traffic management in junction areas and bottleneck sections on mainline railways. PhD thesis, University of Birmingham.

Chen, L., Roberts, C., Schmid, F. et al. (2015) Modeling and solving real-time train rescheduling problems in railway bottleneck sections. IEEE Transactions on Intelligent Transportation Systems, 16 (4): 1896–1904.

Clausen, J. (2003). Branch and Bound Algorithms-Principles and Examples. [online] www.semanticscholar.org.

Cordone, R. and Redaelli, F. (2011) Optimizing the demand captured by a railway system with a regular timetable. Transportation Research Part B: Methodological, 45 (2): 430–446.

Corman, F., D'Ariano, A., Pacciarelli, D. et al. (2012) Bi-objective conflict detection and resolution in railway traffic management. Transportation Research Part C: Emerging Technologies, 20 (1): 79–94.

Corman, F., D'Ariano, A., Pacciarelli, D. et al. (2010) A tabu search algorithm for rerouting trains during rail operations. Transportation Research Part B: Methodological, 44 (1): 175–192.

Corman, F. and Quaglietta, E. (2015) Closing the loop in real-time railway control: Framework design and impacts on operations. Transportation Research Part C: Emerging Technologies, 54: 15–39.

Crosby, H., Davis, P. and Jarvis, S. (2016) Spatially-intensive decision tree prediction of traffic flow across the entire UK road network. 2016 IEEE/ACM 20th International Symposium on Distributed Simulation and Real Time Applications (DS-RT).

Dantzig, G. (1948) Programming in a linear structure. Bulletin of the American Mathematical Society, 54 (11): 1074–1074.

D’Ariano, A., Pranzo, M. and Hansen, I. (2007) Conflict resolution and train speed coordination for solving real-time timetable perturbations. IEEE Transactions on Intelligent Transportation Systems, 8 (2): 208–222.

Delle Site, P. and Filippi, F. (1998) Service optimization for bus corridors with short-turn strategies and variable vehicle size. Transportation Research Part A: Policy and Practice, 32 (1): 19–38.

Ding, L. and Xu, J. (2017). A review of metro construction in China: Organization, market, cost, safety and schedule. Frontiers of Engineering Management, 4(1), p.4.

Dorigo, M. (1992) Optimization, learning and natural algorithms. PhD thesis, Polytechnic University of Milan.

Eberlein, X., Wilson, N., Barnhart, C. et al. (1998) The real-time deadheading problem in transit operations control. Transportation Research Part B: Methodological, 32 (2): 77–100.

Eberlein, X., Wilson, N. and Bernstein, D. (2001) The holding problem with real–time information available. Transportation Science, 35 (1): 1–18.

Flamini, M. and Pacciarelli, D. (2008) Real time management of a metro rail terminus. European Journal of Operational Research, 189 (3): 746–761.

Fu, L., Liu, Q. and Calamai, P. (2003) Real-time optimization model for dynamic scheduling of transit operations. Transportation Research Record, 1857 (1): 48–55.

Gao, Y., Kroon, L., Schmidt, M. et al. (2016) Rescheduling a metro line in an over-crowded situation after disruptions. Transportation Research Part B: Methodological, 93: 425–449.

Ghoneim, N. and Wirasinghe, S. (1986) Optimum zone structure during peak periods for existing urban rail lines. Transportation Research Part B: Methodological, 20 (1): 7–18.

Gilmore, P.C. and Gomory, R.E. (1961). A Linear Programming Approach to the Cutting-Stock Problem. Operations Research, 9(6), pp.849–859.

Glover, F. (1977) Heuristics for integer programming using surrogate constraints. Decision Sciences, 8 (1): 156–166.

Glover, F. (1986) Future paths for integer programming and links to artificial intelligence. Computers & Operations Research, 13 (5): 533–549.

Halim, H., Sakr, M. and Aly, W. (2016) Metro timetable optimization from passenger perspective based on simulation models and incomplete data of passenger flow. 2016 IEEE Symposium Series on Computational Intelligence (SSCI).

He, L., Chen, L., Liu, J., Roberts, C., Yu, S. and Feng, X. (2022). Passenger Flow-Oriented Metro Operation without Timetables. Applied Sciences, [online] 12(10), p.4999.

Hefei Urban Rail Transit Co. Ltd. (2021). Hefei Metro Emergency Response Standards.

Higgins, A., Kozan, E. and Ferreira, L. (1996) Optimal scheduling of trains on a single line track. Transportation Research Part B: Methodological, 30 (2): 147–161.

Hillier, F.S. (2021). Introduction to operations research. Dubuque: Mcgraw-Hill Education.

Holland, J. (2010) Adaptation in natural and artificial systems. Cambridge, Mass.: MIT Press.

Hong, X., Meng, L., D'Ariano, A., Veelenturf, L.P., Long, S. and Corman, F. (2021). Integrated optimization of capacitated train rescheduling and passenger reassignment under disruptions. Transportation Research Part C: Emerging Technologies, 125, p.103025. doi:https://doi.org/10.1016/j.trc.2021.103025.

Hou, Z., Dong, H., Gao, S. et al. (2019) Energy-saving metro train timetable rescheduling model considering ATO profiles and dynamic passenger flow. IEEE Transactions on Intelligent Transportation Systems, 20 (7): 2774–2785.

Jeter, M. (2018). Mathematical Programming. Routledge.

Kang, L. and Zhu, X. (2016). A simulated annealing algorithm for first train transfer problem in urban railway networks. Applied Mathematical Modelling, 40(1), pp.419–435.

Karmarkar, N. (1984) A new polynomial-time algorithm for linear programming. Combinatorica, 4 (4): 373–395.

Karush, W. (1939) Minima of functions of several variables with inequalities as side constraints. MSc thesis, University of Chicago.

Kennedy, J. and R. Eberhart (1995). "Particle swarm optimization." 1995 Ieee International Conference on Neural Networks Proceedings, Vols 1-6: 1942-1948.

Kirkpatrick, S., Gelatt, C. and Vecchi, M. (1983) Optimization by simulated annealing. Science, 220 (4598): 671–680.

Konno, H., Kawadai, N. and Tuy, H. (2003) Cutting plane algorithms for nonlinear semi-definite programming problems with applications. Journal of Global Optimization, 25: 141–155.

Krasemann, J. (2010) Greedy algorithm for railway traffic re-scheduling during disturbances: A Swedish case. IET Intelligent Transport Systems, 4 (4): 375.

Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. Proceedings of 2nd Berkeley Symposium. Berkeley: University of California Press; pp. 481–492.

Land, A.H. and Doig, A.G. (1960). An Automatic Method of Solving Discrete Programming Problems. Econometrica, 28(3), p.497.

Li, X. and Lo, H. (2014) An energy-efficient scheduling and speed control approach for metro rail operations. Transportation Research Part B: Methodological, 64: 73–89.

Li, X., Wang, D., Li, K. et al. (2013) A green train scheduling model and fuzzy multi-objective optimization algorithm. Applied Mathematical Modelling, 37 (4): 2063–2073.

Lin, D., Broere, W. and Cui, J. (2022). Metro systems and urban development: Impacts and implications. Tunnelling and Underground Space Technology, 125, p.104509.

Lin, D., Nelson, J.D. and Cui, J. (2021). Exploring influencing factors on metro development in China from urban and economic perspectives. Tunnelling and Underground Space Technology, 112, p.103877.

Lipowski, A. and Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. Physica A: Statistical Mechanics and its Applications, 391(6), pp.2193–2196.

Liu, C., Hu, Z., Li, Y. et al. (2017) Forecasting copper prices by decision tree learning. Resources Policy, 52: 427–434.

Louwerse, I. and Huisman, D. (2014) Adjusting a railway timetable in case of partial or complete blockades. European Journal of Operational Research, 235 (3): 583–593.

Lusby, R.M., Larsen, J. and Bull, S. (2018). A survey on robustness in railway planning. European Journal of Operational Research, 266(1), pp.1–15.

Marchand, H., Martin, A., Weismantel, R. and Wolsey, L. (2002). Cutting planes in integer and mixed integer programming. Discrete Applied Mathematics, 123(1-3), pp.397–446.

Meng, L. and Zhou, X. (2014) Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables. Transportation Research Part B: Methodological, 67: 208–234.

Minoux, M. (2004) Mathematical programming. Ann Arbor: UMI Books on Demand.

Mo, P., Yang, L., Wang, Y. et al. (2019) A flexible metro train scheduling approach to minimize energy cost and passenger waiting time. Computers & Industrial Engineering, 132: 412–432.

Mu, S. and Dessouky, M. (2013) Efficient dispatching rules on double tracks with heterogeneous train traffic. Transportation Research Part B: Methodological, 51: 45–64.

Ning, B., Xun, J., Gao, S. and Zhang, L. (2015). An Integrated Control Model for Headway Regulation and Energy Saving in Urban Rail Transit. IEEE Transactions on Intelligent Transportation Systems, 16(3), pp.1469–1478.

Niu, H. and Zhou, X. (2013) Optimizing urban rail timetable under time-dependent demand and oversaturated conditions. Transportation Research Part C: Emerging Technologies, 36: 212–230.

Niu, H., Zhou, X. and Gao, R. (2015) Train scheduling for minimizing passenger waiting time with time-dependent demand and skip-stop patterns: Nonlinear integer programming models with linear constraints. Transportation Research Part B: Methodological, 76: 117–135.

Ochiai, Y., Masuma, Y. and Tomii, N. (2019). Improvement of timetable robustness by analysis of drivers' operation based on decision trees. Journal of Rail Transport Planning & Management, 9, pp.57–65.

Osman, I. and Laporte, G. (1996) Metaheuristics: A bibliography. Annals of Operations Research, 63 (5): 511–623.

Pan, H., Yang, L. and Liang, Z. (2022). Demand-oriented integration optimization of train timetabling and rolling stock circulation planning with flexible train compositions: A column-generation-based approach. European Journal of Operational Research.

Powell, M. (1964) An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal, 7 (2): 155–162.

Restel, F. and Haładyn, S.M. (2022). The Railway Timetable Evaluation Method in Terms of Operational Robustness against Overloads of the Power Supply System. Energies, 15(17),

Schenk, H. (1998). "An elementary introduction to direct methods." Direct Methods for Solving Macromolecular Structures 507: 19-26.

Schrijver, A. (2011). Theory of linear and integer programming. Chichester ; Weinheim: Wiley.

Sierksma, G. and Zwols, Y. (2015). Linear and integer optimization : theory and practice. Boca Raton: Chapman & Hall/CRC.

Storn, R. and Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11(4): 341–359.

Su, C. and Shiue, Y. (2003) Intelligent scheduling controller for shop floor control systems: A hybrid genetic algorithm/decision tree learning approach. International Journal of Production Research, 41 (12): 2619–2641.

Sun, H., Wu, J., Ma, H. et al. (2019) A bi-objective timetable optimization model for urban rail transit based on the time-dependent passenger volume. IEEE Transactions on Intelligent Transportation Systems, 20 (2): 604–615.

Sun, X., Zhang, S., Dong, H. et al. (2015) Optimization of metro train schedules with a dwell time model using the Lagrangian duality theory. IEEE Transactions on Intelligent Transportation Systems, 16 (3): 1285–1293.

Tai, X. and Espedal, M. (1998) Rate of convergence of some space decomposition methods for linear and nonlinear problems. SIAM Journal on Numerical Analysis, 35 (4): 1558–1570.

Takagi, R., Weston, P., Goodman, C. et al. (2006) Optimal train control at a junction in the main line rail network using a new object-oriented signalling system model. WIT Transactions on The Built Environment, 88: 479–488.

Tomii, N., Tashiro, Y., Tanabe, N. et al. (2005) Train operation rescheduling algorithm based on passenger satisfaction. Quarterly Report of RTRI, 46 (3): 167–172.

Transport for London (2021). Annual Report and Statement of Accounts. Transport for London.

Wang, D., Wang, Y., Zhu, S., Meng, L. and Tang, T. (2020). Train rescheduling for minimizing passenger travel time under disruption for metro lines. [online] IEEE Xplore.

Wang, P. and Zhang, Q. (2019) Train delay analysis and prediction based on big data fusion. Transportation Safety and Environment, 1 (1): 79–88.

Wang, Y. (2013) Real-time scheduling for single lines in urban rail transit systems. In IEEE International Conference on Intelligent Rail Transportation. Beijing. 2013. IEEE; p. 7.

Wang, Y., Zhu, S., D'Ariano, A., Yin, J., Miao, J. and Meng, L. (2021). Energy-efficient timetabling and rolling stock circulation planning based on automatic train operation levels for metro lines. Transportation Research Part C: Emerging Technologies, 129, p.103209.

Wong, R., Yuen, T., Fung, K. et al. (2008) Optimizing timetable synchronization for rail mass transit. Transportation Science, 42 (1): 57–69.

Wheat, P. and Wardman, M. (2017). Effects of timetable related service quality on rail demand. Transportation Research Part A: Policy and Practice, 95, pp.96–108.

Xu, X., Li, K. and Lu, X. (2019) Simultaneous locomotive assignment and train scheduling on a single-track railway line: A simulation-based optimization approach. Computers & Industrial Engineering, 127: 1336–1351.

Yang, L., Li, K. and Gao, Z. (2009). Train Timetable Problem on a Single-Line Railway With Fuzzy Passenger Demand. IEEE Transactions on Fuzzy Systems, 17(3), pp.617–629.

Yang, L., Zhou, X. and Gao, Z. (2014) Credibility-based rescheduling model in a double-track railway network: A fuzzy reliable optimization approach. Omega, 48: 75–93.

Yang, S., Liao, F., Wu, J. and Chen, Y. (2022). An Efficient Train Timetable Scheduling Approach With Regenerative-Energy Supplementation Strategy Responding to Potential Power Interruptions. IEEE Transactions on Intelligent Transportation Systems, [online] 23(9), pp.14267–14282.

Yang, X., Chen, A., Li, X. et al. (2015) An energy-efficient scheduling approach to improve the utilization of regenerative energy for metro systems. Transportation Research Part C: Emerging Technologies, 57: 13–29.

Yang, X., Wu, J., Sun, H., Gao, Z., Yin, H. and Qu, Y. (2019). Performance improvement of energy consumption, passenger time and robustness in metro systems: A multi-objective timetable optimization approach. Computers & Industrial Engineering, 137, p.106076.

Yazdani, D., Cheng, R., Yazdani, D. et al. (2021) A survey of evolutionary continuous dynamic optimization over two decades—Part A. IEEE Transactions on Evolutionary Computation, 25 (4): 609–629.

Yin, J., Chen, D., Yang, L. et al. (2016) Efficient real-time train operation algorithms with uncertain passenger demands. IEEE Transactions on Intelligent Transportation Systems, 17 (9): 2600–2612.

Zhan, S., Kroon, L., Veelenturf, L. et al. (2015) Real-time high-speed train rescheduling in case of a complete blockage. Transportation Research Part B: Methodological, 78: 182–201.

Zhao, N., Roberts, C., Hillmansen, S., Tian, Z., Weston, P. and Chen, L. (2017). An integrated metro operation optimization to minimize energy consumption. Transportation Research Part C: Emerging Technologies, 75, pp.168–182.

Zhou, X. and Zhong, M. (2005) Bicriteria train scheduling for high-speed passenger railroad planning applications. European Journal of Operational Research, 167 (3): 752–771.

Zhu, Y. and Goverde, R.M.P. (2020). Integrated timetable rescheduling and passenger reassignment during railway disruptions. Transportation Research Part B: Methodological, 140, pp.282–314.

Zhu, Y., Koutsopoulos, H.N. and Wilson, N.H.M. (2017). Inferring Left Behind Passengers in Congested Metro Systems from Automated Data. Transportation Research Procedia, 23, pp.362–379.