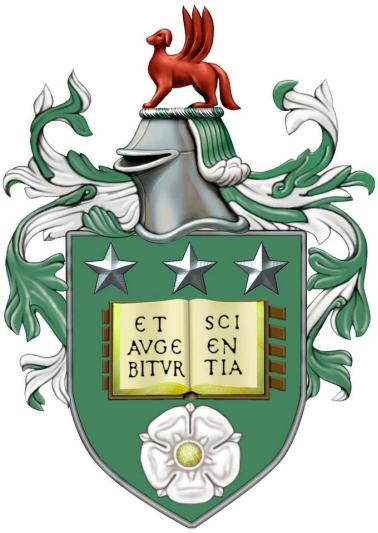
Train Unit Scheduling Optimization with Station Level Resolution

Li Lei

Submitted in accordance with the requirements
for the degree of Doctor of Philosophy



The University of Leeds
School of Computing
September 2020

The candidate confirms that the work submitted is her own, except where work which has formed part of a jointly authored publication has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

Materials from all publications on conferences or journals below have been reorganized and distributed through-out the thesis. Li Lei contributed the major part of these works under the supervision of Professor Raymond Kwan. During the first year of her PhD study, Dr. Zhiyuan Lin acted as the co-supervisor.

Li Lei, Raymond S.K. Kwan, Zhiyuan Lin, Pedro J. Copado-Mendez, "Station level refinement of train unit network flow schedules", the 8th International Conference on Computational Logistics, Southampton, UK, October 2017.

Li Lei, Raymond S.K. Kwan, Zhiyuan Lin, Pedro J. Copado-Mendez, "Resolution of Station Level Constraints in Train Unit Scheduling", Conference on Advanced Systems in Public Transport and TransitData 2018, Brisbane, Australia, July 2018.

Li Lei, Raymond S.K. Kwan, Zhiyuan Lin, Pedro J. Copado-Mendez, "Resolution of coupling order and station level constraints in train unit scheduling", Public Transport. Under review.

Li Lei and Raymond S.K. Kwan, "Evaluation of Objective Function Designs Through an Auxiliary Heuristic Scheme", the 13th Metaheuristics International Conference, Cartagena, Colombia, July 2019.

Li Lei and Raymond S.K. Kwan, "An auxiliary heuristic approach for objective function design evaluation", Operations Research Perspectives. Under review.

Acknowledgements

Here, I would like to express my gratitude to all who helped me in accomplishing my research and this thesis.

First of all, I am most grateful to Professor Raymond Kwan, my supervisor, who gave me this great opportunity to be his PhD student. During the last few years, he has been very supportive with encouraging and inspiring guidance and discussions to lead me to finish my PhD research and to be a better young researcher. He shared my joy and also helped me out through tough times. He always patiently corrected my bad-writing drafts and explained to me why he did those changes and how to improve writing skills. He helped me to get in contact with related industrial companies that provided me with valuable experimental datasets and also feedback on my research. He supported me in attending winter school and academic conferences, where to build research connections and broaden my research horizon. He showed me how to work efficiently, and encouraged me to balance work and life. I also want to express my sincere thank to him for the academic, emotional, and financial supports especially during the thesis writing time when the University is locked down. My great gratitude also goes to Dr. Zhiyuan Lin, Dr. Pedro J Copado-Mendez, Dr. He Wang, Dr. Mustafa Bakr. They gave me a lot of valuable help and suggestions on my research, programming, academic writing, career, and life. As my co-supervisor during the first-year study, Dr. Zhiyuan Lin showed great support to my research even after he moved to the University of Manchester. They and their families brought me a lot of great times and memories as friends. This research has also greatly benefited from collaborations with Tracsis plc and train operating companies Great Western Railway, TransPennine Express, and Greater Anglia (UK), to whom I would like to express my gratitude. In particular, I want to thank Ian Smith, Mark Quinn, Mark Withall, Stathis Agrapidis, Russ Cunningham, Gareth Rawlings for their valuable advice and expertise.

I want to thank the University of Leeds, especially the School of Computing for granting me the fees and maintenance studentship. Without this scholarship, I could not manage to study at this great University, which has provided me with

a very great and joyful study and research environment. It enables me to meet many knowledgeable researchers whom I learned from. I also want to express sincere thanks to Professor Kristina Vušković, Professor Eric Atwell, Dr. Isolde Adler, Dr. Natasha Shakhlevich, Dr. Brandon Bennett, Dr. Samuel Wilson, who gave me chance to work as a teaching assistant to practise and improve my teaching skills. Besides, I want to greatly thank the administrative team, especially Mrs. Judi Drew, Mrs. Gaynor Butterwick, Miss Kathrine Garnett, who provided efficient administrative supports and also enjoyable team-up events. I am also grateful to Professor Peter Jimack, who gave me chances to help with the meeting between the University of Leeds and the Joint School of SWJTU (both are my alma maters) as an 'unprofessional' interpreter. My great gratitude also goes to my supervisors during my postgraduate and undergraduate, Professor Qiyuan Peng, and Professor Gang Wu, for their support and help. I want to thank my English and Chemistry teachers at high school, Changsheng Peng, Hui Chen, for their kindness and edification.

I want to thank my parents deep from my heart. My parents do not speak English and have no idea about my research, but they tried their best to show great support and encouragement although what I did is far from their expectation. I probably could not do better if I were in their position. Without their consistent support, I would never have had a chance to study as a PhD student in the UK. I am also grateful to my aunties, uncles, and cousins who made my parents' life happier and easier without me next to them. In addition, I want to give many thanks to my friends in China who gave me a lot of emotional support, especially to Yong Yang, Li Fan, Xiaodan Yan, Dan Li, Xiaoyun Zhang, Nianci Wen, Daben Yu, Yanbo Wang, Jingwei Guo, Qiaochu Fan, Qingwei Zhong, and many more. I appreciate everyone I met who gave me good memories in life. In the end, I also want to give thanks to dear me, who did not give up when many hard times encountered in my life.

Li Lei

Leeds, 23rd September 2020

Abstract

The train unit scheduling optimization (TUSO) problem aims at seeking a conflict-free operational plan for a set of train units to serve all trips defined in a fixed timetable with minimum operational costs. TUSO is addressed at two levels: the network level and the station level. The network level focuses on determining the serving sequence of trips for each train unit, where the stations are simplified as single points. The station level deals with the issues left in a network-level solution with detailed infrastructure restored. Prior to this research, TUSO at the network level, specific on the UK railway operating system, has been tackled as a multi-commodity network flow problem. Whereas train unit flows are balanced and optimised over the service network, potential operational conflicts due to layouts in individual train station have been ignored. This research mainly concerns resolving such operational conflicts at the station level. However, this research has also made contributions in improving the network flow model.

This research follows the two-phase approach [60] to tackle TUSO at these two levels. TUSO is first solved at the network level in Phase I, where two solvers have been developed, namely RS-Opt [57] and SLIM [26]. Given a solution from the network level, two operational aspects are left undetermined: coupling order issues and linkage feasibility. To finalize these two aspects, an adaptive approach expanding Phase I to Phase II is proposed. Phase II takes a further step of station-level resolution and attempts to complete a fully operable schedule. The logistics of coupling/decoupling activities and tentative linkages are determined in detail to prevent conflicts where possible, particularly focusing on developing an operable schedule without conflicts of coupling order or crossing linkages in train stations. If the unresolvable station-level conflicts still exist at Phase II, the process loops back to Phase I with added constraints to avoid the identified conflicts. Through these two phases, a global optimal solution that is also operable considering station-level layouts will be secured. Moreover, the observation on the network-level experimental results from the existing RS-Opt and SLIM has inspired the research on improving the network flow model from the perspective of considering additional terms in the objective function such as the slack time and the number of cars. It is extended as a new methodology to evaluate the effectiveness of alternative objective function designs.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Railway transport planning

The 'Transport Statistics Great Britain 2018 Report' shows that track transport (including national rail, underground, light rail and tram) plays a more and more important role in people's daily life through the last 30 years [1]. As shown in Figure 1.1, the journeys completed by track modes increase continuously while the bus journeys outside the London area drop steeply. In terms of operational kilometers, shown in Figure 1.2, the national rail has a significant rise, but the

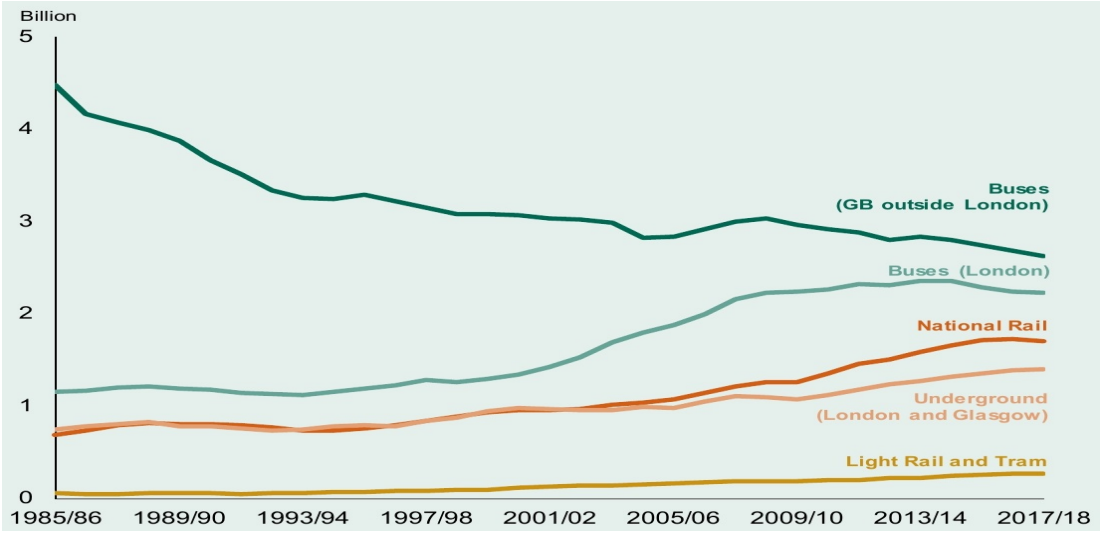


Figure 1.1: Passenger journeys

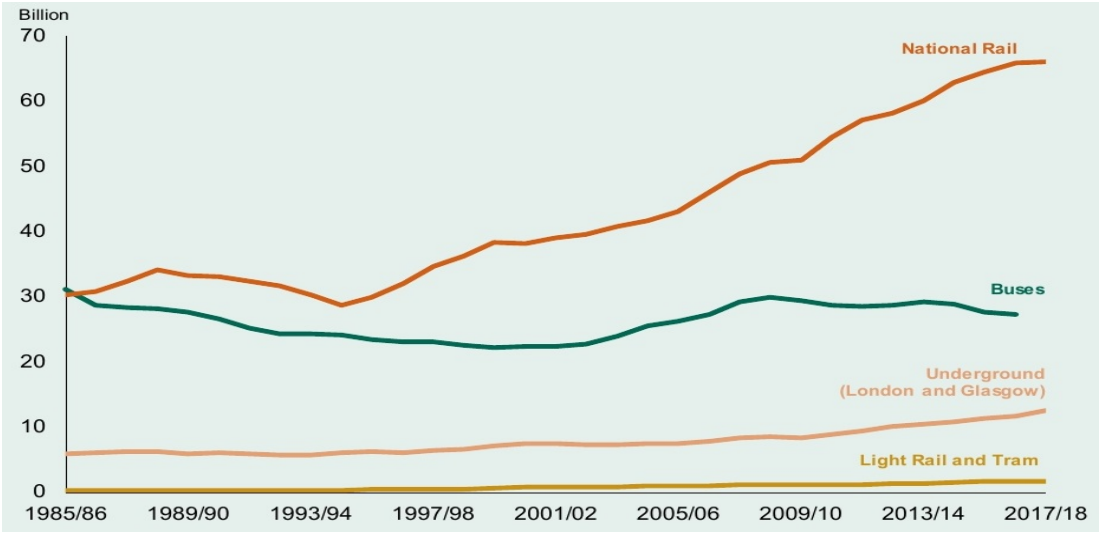


Figure 1.2: Passenger kilometers

kilometer variation of bus services is rather small. Compared to buses, train services have the advantage of medium-to-long-distance transport. The increasing demand of track transport raise new challenges: how to manage the massive railway system efficiently? With the development of computing capacity, the railway industry gradually turns to invest in computer-aided systems at the planning and operational stages. For instance, the Canadian National Railway uses a decision support system (DSS) to successfully decrease the incidence of derailments while other companies were experiencing an increase [71]. Lots of literature also focus on automatic planning and operating methods. For instance, Kroon et al. develop a DSS that can assist the planners of Railned in generating timetables for railway trips [46]; Ingolotti et al propose a DSS to efficiently and quickly solve and plot the single-track railway scheduling problem [43].

Public railway transport planning is an intractable and complicated process that needs tremendous work of many interacting aspects. Thus, the entire planning process is usually divided into a few hierarchical stages, as shown in Figure 1.3 [34, 87, 70]. In the classic sequential planning process, the results of the previous planning stage are considered as pre-fixed information for the next stage. This strategy leads to a sub-optimal operation for the railway system [75]. First is *demand analysis*, focusing on the determination of passengers' traveling demands

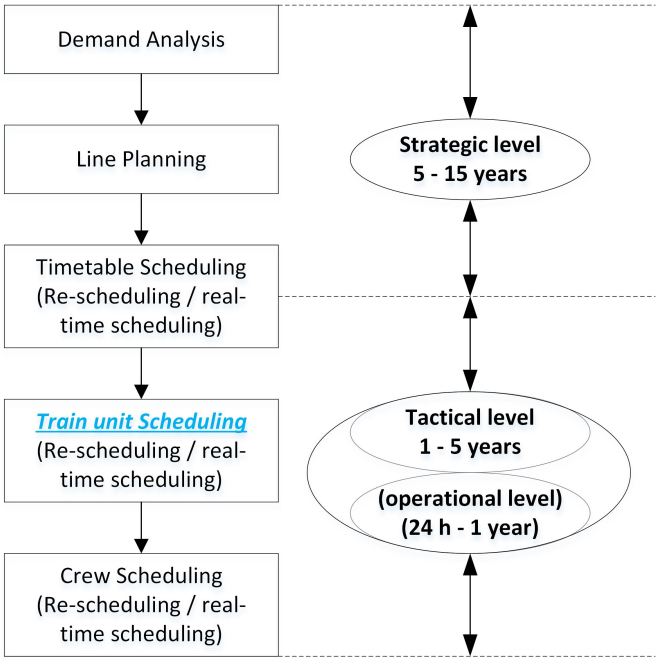


Figure 1.3: Hierarchical railway transport planning process

and their O-D pairs. The *line planning* aims at determining a set of routes to be convenient for the potential passenger demands with some frequencies and stop patterns for trip services [74]. These two stages are usually classified as the *strategic level* with a time scale of 5 to 15 years, which are objective to resource acquisition. Generally speaking, timetabling is the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives [14]. The *timetable scheduling* for railway transportation aims at determining a periodic timetable for a set of trips that do not violate track capacities and satisfies some operational constraints [20]. Thereafter comes *train unit scheduling* where two subproblems are derived: train unit assignment, train unit shunting. Train unit assignment is to determine the utilization plan of limited train units to cover all the trips defined by a given timetable under some constraints. Train unit shunting takes care of the shunting operations in stations or depots. The *crew scheduling* stage aims to distribute the personnel to ensure every trip has enough staff. Two levels are considered for theses three scheduling stages: the tactical level has a time

scale of 1 - 5 years to deal with resource allocation; the operational level of 24 hours to 1 year is about daily operations. At the operational level, rescheduling is also necessary when some unexpected disruptions are encountered, for example, strike, severe weather, train unit breakdown. This research focuses on the train unit scheduling problem, marked as blue with an underline in Figure 1.3, which is the pre-planning of daily operations in advance of a new timetable going live. In the UK, the schedules planned usually serve for a duration of over six months, therefore, optimization is significantly important.

1.2 Train unit scheduling optimization (TUSO)

Currently, the UK railway system is operated by 17 franchises [2], such as National Express, Northern Rail, First Great Eastern, TransPennine Express (TPE), Great Western Railway (GWR). Each operator possesses/leases a certain number of train units whose leasing, operating, and maintenance costs are quite expensive. Railway operators have to provide adequate train units to satisfy passenger demands and comply with many complex operational rules required by franchise agreements. Thus, railway operators always try to make use of limited train unit resources as efficiently as possible. For some regions where many commuting trips are conducted, train unit scheduling is even more crucial for the distribution of limited fleet resources, for instance, the Greater London centred area in southern England and the Edinburgh-Glasgow centred area in Scotland.

Compared to traditional locomotive trains, a train multiple unit (referred to as train unit for short) has a fixed number of carriages, e.g., 2-car and 4-car units. It has a built-in engine(s) capable of moving in either direction. Figure 1.4 gives some train unit examples of one to four cars. Train units are commonly used in modern railway passenger transportation of many countries, such as the UK, China, Japan, and many other European countries. Train units can be classified into different types according to many characteristics, for instance, the power source (electric multiple unit (EMU), diesel multiple unit (DMU), and bi-mode multiple unit (BMU)), number of carriages, number of seats. Based on physical configurations of train units and special panning rules, a train unit family is defined as a set of train unit types that are allowed to be coupled together. Table

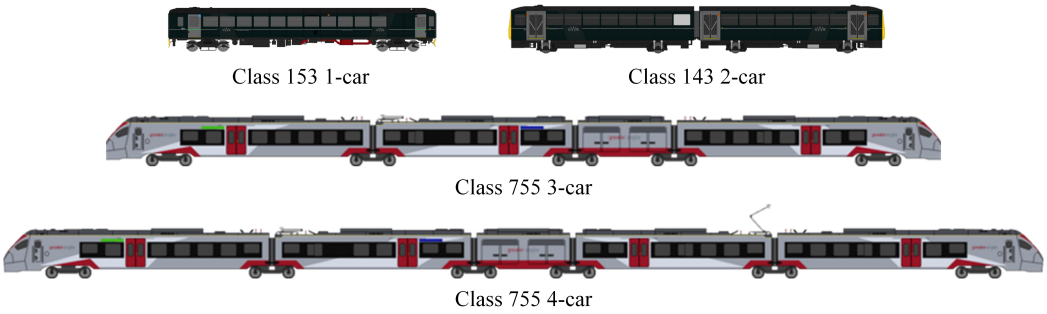


Figure 1.4: Train unit examples

1.1 shows some families of different types of train units used in the UK railway system. In the UK railway industry terms, a timetable defines a set of logical

Table 1.1: Example train unit families used in the UK

Family	Unit type	Power source	Cars	Seats	Example operator
1	Class 170/1	DMU	2	109	CrossCountry
	Class 170/3		3	182	
2	Class 317/5	EMU	4	291	Greater Anglia
	Class 317/6		4	268	
3	Class 755/3	BMU	3	153	Greater Anglia
	Class 755/4		4	205	
4	Class 158/0	DMU	2	138	Great Western Railway
	Class 158/9		3	142	
5	Class 350/4	EMU	4	205	TransPennine Express
6	Class 802/2	BMU	5	326	TransPennine Express

trips operated on the railway network with a series of attributes: origin and destination (O-D pair), departure and arrival times, route, intermediate stations, and parking platforms. A few notable restrictions for each logical trip are the passenger demand to be satisfied, the train unit types and the maximum number of train units or carriages that are allowed to serve this trip.

Train unit scheduling optimization, referred to as TUSO for short, aims at seeking an optimized operable plan of utilizing the limited number of train units at the railway tracks concerning specific objectives and constraints. The features of TUSO in this thesis also include coupling/decoupling operation, empty-running,

mileage, re-platforming, reversal, coupling order issues, station feasibility. Some research of TUSO includes maintenance planning as well, for example, [17, 36, 86, 48]. In the UK, however, maintenance provision can often be achieved later either within the slacks between scheduled train trips or by swapping physical units of the same type. Therefore the train unit scheduling stage conventionally does not consider maintenance planning. TUSO takes three pieces of information as inputs: a timetable describing trips, a fleet of train units, railway infrastructure, and operational rules. The solution of TUSO is normally the sequence of trips that are consecutively served by every utilized train unit, which demonstrates its working loads on an operational day. The train unit(s) assigned to serve a timetabled trip is defined as a train unit block, which could be a single unit or multiple coupled units. In the UK, train unit scheduling is also called train unit diagramming. A diagram describes the sequence of serving trips and other auxiliary activities (e.g., coupling/decoupling, reversal) for a certain train unit.

Table 1.2 is an example diagram (referred to as AK 219 FO) from the TPE company, used from 20/12/2019 to 15/05/2020 [3]. This train unit is of type class 185/0 with three carriages. The locations visited by this train unit are listed in the first column. The second and third columns give the arrival and departure times. The fourth column demonstrates the serving trips during an operational day. The specific route occupied by each trip is listed in the fifth column to avoid ambiguity when there is more than one route between two locations. The auxiliary activities are shown in the sixth column, including reverse, attach, and detach. The last column shows the train unit formation and permutation when a unit block of multiple coupled units serves one trip, where '1' and '2' represent the front and the rear, respectively. A reverse activity flips the front-rear sequence of a formation of multiple coupled units. For instance, trip '1B66' has a reverse operation at location 'ManchstrP', referring to Manchester Piccadilly. As a result of this reverse activity, the coupled train unit sequence is flipped from '218(1)/219(2)' to '219(1)/218(2)'.

Currently, the train unit schedule is produced manually in the UK, on the basis of trial-and-error and ad-hoc manner. The manual method can guarantee the feasibility of a train unit schedule at the railway infrastructure. However, this process is very time consuming and cannot ensure the optimality. Once the

timetable is finalized, the operators proceed to the train unit scheduling process, followed by the crew scheduling and rostering process. Train unit scheduling is a bottleneck before train crew scheduling. Savings in train unit schedules will

Table 1.2: A train unit diagram from TransPennine Express (TPE)

Diagram: AK 219 FO. Fleet : 185/0 LTP
Start Date: 20/12/2019; End Date: 15/05/2020
Miles from Fuel: Start 700.00, End 1366.12
Miles: Loaded 645.37, Empty 20.75, Total 666.12

Location	Arr	Dep	WTT	Route	Activities	Coupled
Sheffield		03:25	1B59	Main		
Huddrsfld	04:16	04:17	1B59	Gdb		
ManchstrP	04:52	04:56	1B59		Reverse	
Mancr Air	05:11				Attach	
Mancr Air		06:53	1B66			218(1)/219(2)
ManchstrP	07:11	07:18	1B66	Stkp	Reverse	219(1)/218(2)
Sheffield	08:11	08:12	1B66	Main		219(1)/218(2)
Cleethpes	09:05				Detach	
Cleethpes		11:26	1B77			
Doncaster	12:04	12:42	1B77	Main		
Sheffield	13:08	13:11	1B77	Stkp		
ManchstrP	14:02	14:07	1B77		Reverse	
Mancr Air	14:23	14:53	1B82			
ManchstrP	15:11				Attach	
					Reverse	
ManchstrP		15:18	1B82	Stkp		219(1)/227(2)
Sheffield	16:09	16:11	1B82	Main		219(1)/227(2)
Doncaster	16:35	16:37	1B82			219(1)/227(2)
Cleethpes	17:49	18:26	1B91		Reverse	227(1)/219(2)
Doncaster	19:39	19:43	1B91	Main		227(1)/219(2)
Sheffield	20:09	20:11	1B91	Stkp		227(1)/219(2)
ManchstrP	21:01				Detach	
					Reverse	
ManchstrP		21:09	1B91			
Mancr Air	21:25	21:53	1B96			
ManchstrP	22:15				Attach	
					Reverse	
ManchstrP		22:02	1B96	Romly		219(1)/220(2)
Sheffield	23:01	23:12	1B96	Wdbrn		219(1)/220(2)
Rothram C	23:32	23:32	1B96			219(1)/220(2)
Doncaster	23:51	24:05	5B96			
CroftnDep	24:59					

lead to savings in crew. Thus, this research focuses on deriving an automated scheduling method, particularly on the optimization objectives and solution quality, and the resolution of station-level conflicts, based on fundamental research [57, 26].

1.3 TUSO at two levels

In practice, a timetable from a railway operator could have up to two thousand trips operated in a weekday. Train unit block serving each trip must be scheduled at the complex railway network as well as detailed station infrastructure. TUSO is a complicated and sophisticated problem that is hard to solve because it involves several systems, e.g., timetables, rail networks and station layout, train unit fleet, engineering and operational rules. They lead to a large number of complex constraints to be satisfied, for instance, passenger demands, fleet size, type-route compatibility, turnaround time, coupling/decoupling activities, coupling order, reversal en-route, track capacities. These complicated constraints make TUSO as a whole a challenging problem to solve. Therefore, TUSO is addressed at two levels: the network level and the station level [56, 47]. The network level allocates train unit resources to obtain the serving sequence of trips in a fixed timetable with minimum operational costs. This level has been studied in literature [60, 17]. At the network level, stations are simplified as single points where the detailed layouts are not considered. This simplification means all the shunting movements required for connections between arrivals and departures are assumed feasible, no matter what schedule is given by the network level. The station level deals with the remaining issues left in the network-level solution with station infrastructure restored, mainly coupling order assignment and the finalization of tentatively linkages that are assigned at the network level.

1.3.1 Research synopsis for the network level

Motivation In literature [56, 17], the network-level problem is modeled as an integer linear programming (ILP) of multi-commodity flows based on a directed acyclic graph (DAG), which is shown as an NP-hard problem [76, 16, 55]. In

the DAG, nodes represent trips with the source and the sink added as usual, and linkages represent potential connections between any two nodes. An exact solver is developed based on branch-and-price [57] and convex hull techniques [58], named 'rolling stock optimizer (RS-Opt)'. RS-Opt is an exact method but only suitable for small to medium problem instances. Thus, a size-limited iterative method (SLIM) boosting RS-Opt to deal with large instances is also derived [26]. As SLIM is supposed to extract good structural properties contributing to the solution such that a solution whose structure is highly overlapped with the 'optimal' solution obtained by RS-Opt solely is expected. However, the experimental results show that the solution schedule converged through SLIM is very different from that obtained by RS-Opt alone, even when their final objective function values are very close or even identical. Although the branch-and-price process in RS-Opt may have influenced the arc selection from the original DAG, the design of objective function may have contributed to the unexpected behavior such that alternative objective function designs are considered at the network level. Moreover, the correlations amongst the hierarchical optimization criteria are complicated. One might synergize/conflict with another, or there may be some more unrevealed or unclear correlations. Thus, 'how much a given objective function can be trusted?' and 'what optimization criteria should be considered in the objective function?' are significant questions. These considerations have driven the expanding research on the methodology of evaluating objective function effectiveness, seen in Chapter 5.

Research approach The research of this part considers the scheduling problems that can be modeled as network flow problems and their main optimization criteria have a clear hierarchy such that the objective function can be formulated as a weighted sum. Integer Linear Programming (ILP) models and their exact solvers have the advantage of delivering '(near-) optimal' solutions. For a complex real-world scheduling problem, the solution usually contains many combinatorial properties, for instance the schedule structure. Using small problem instances that an exact ILP method can solve to optimality, we propose a methodology to benchmark the effectiveness of alternative objective function designs, including three main steps: design alternative objective functions, obtain solutions and

their features, evaluate the effectiveness of alternative objective function designs. The main measure of effectiveness is in the structural similarity between the auxiliary heuristic converged solutions and the exact solver found solution. In conjunction with other solution features, the objective function effectiveness is further quantified as an aggregation of several derived elements.

Contribution While the research on multi-criteria optimization and parameter control are extensive and fruitful in the literature, they mainly focus on the performance of solution algorithms rather than establishing high confidence in the objective function or contributing to the 'definition' of what is considered as optima in the objective function, which are blank in the literature. The methodology proposed at the network level bridges these two gaps to quantify the objective function effectiveness and identify what is optimal to be considered in the objective function. For a real-world scheduling problem, consider an exact Integer Linear Programming (ILP) solver (P) as a black-box, used to deliver a benchmark solution s^*. An auxiliary heuristic approach ($\bar{P}$) which iteratively calls P with an improved reduced input is established to deliver a set of heuristic solutions $\{s_1, s_2, ..., s_z\}$ compared to s^*. Through the comparisons, a series of features are derived to quantify the effectiveness of alternative objective function designs. Although we may still need to rely on the judgement of domain experts, considering structural properties in the objective function design may help better covering the 'hidden' criteria considered by practitioners and establish higher confidence in the objective function.

1.3.2 Research synopsis for the station level

Motivation Station-level layouts and infrastructure are significant when implementing a network-level schedule at stations, which are the main place terminating arrival trips and generating new departure trips. In a network-level solution schedule, there are two things assigned: the train unit block serving each trip, the linkages between arrivals and departures. However, the simplification of station layouts at the network level results in some uncertainties that may cause operational conflicts, classified into two operational aspects. The first

aspect is the sequence of multiple train units (referred to as the coupling order). Coupling/decoupling activities can sometimes only be operated in a specific sequence. The second aspect is linkage finalization at platforms that have been assumed to be operable at the network level. Each linkage implies a series of station operations. The tentatively assigned linkage at the network level may invalidate the solution at the station level. These two aspects will be finalized at the station level, seen in Chapter 6.

Research approach An adaptive approach is proposed to expand the network level (Phase I) to the station level (Phase II), attempting to finalize a fully operable schedule. The basic Phase I model optimizes train unit assignment at the network level and tentatively allocates unit blocks to trips and their connections. The incompleteness of undetermined coupling orders and unfinalized tentative linkages may cause severe operational conflicts at the stations. Phase II is to take a further step of station-level resolution and to fix the incompleteness. Station level constraints would be very complex and difficult to generalize in an all-encompassing TUSO model, therefore, they are hidden/ignored in Phase I. Phase I has the benefit that its solution may have already automatically satisfied many of the station level constraints although they have not been explicitly modeled. This adaptive approach incrementally inserts constraints for Phase II against station-level conflicts identified in the solution from Phase I. The main loop of this method is between Phase I core solver (RS-Opt, developed by Lin and Kwan [57]) and Phase II including the process of station-level conflict detection and coupling order assignment. This method resolves the station-level constraints and assigns feasible coupling orders. A new scheduling solution is sought if Phase II has encountered unresolvable conflicts, and newly formulated critical station constraints are fed back to RS-Opt to eliminate those inoperable conflicts. The experiments show the correctness and effectiveness of this approach.

Contribution The problems at the network level and the station level are usually considered separately. The network-level model has the limitation of ignoring station-level constraints, which leads to an incomplete solution. This defect restricts the operability when it is implemented at the station level because of a set

of undecided factors, for instance, the coupling order impacted by station layouts, timings, and unit movement directions, etc. This piece of research expands the network level research to the station level and scrutinizes the potential problems in a solution of the network level. An integer linear programming with station-level constraints eliminating possible conflicts is proposed. Based on the research about the network level of TUSO, an adaptive approach is proposed to consider the station-level constraints efficiently producing a complete and operable solution by systematically analyzing and iteratively adding station-level conflict constraints. Through this method, the linkages given by the basic network-level model can be finalized and feasible coupling orders can also be assigned if there is no conflict detected.

1.4 Outline of thesis

In chapter 2, the existing research relevant to this thesis is briefly reviewed. Firstly, a brief overview of the previous research relevant to TUSO is given, which has been addressed at two levels: the network level and the station level. The review on the network level is presented in three subsections: train unit assignment, train unit scheduling, and train unit circulation. The station-level mainly focuses on the review of trip and train unit planning within a station/depot. Also, the research bridging the network level and the station level are discussed. Thereafter, the topic related to the objective function evaluation is reviewed, including multi-criteria optimization, parameter tuning and control.

Chapter 3 describes TUSO at both the network and the station levels in detail, including restrictive constraints, optimization criteria. The requirements at the network level are discussed as below: the fleet size of each train unit type, passenger demands, compatibility between train unit and route/depot, turnaround time, and coupling/decoupling issues at the network. Requirements, when a network-level solution is applied to the station level, are discussed from the following aspects: coupling order, linkage slack time, crossing linkages, platform type and capacity, re-platforming shunting, interchangeability between the same type of train units. The optimization criteria and solution qualities are discussed from six aspects: fleet size, running mileage, empty running, number of cou-

pling/decoupling events, connection preference, and unit type preference.

Chapter 4 firstly introduced the directed acyclic graph (DAG) framework, where DAG generation, decision variables, and arc costs are elaborated. Based on the DAG, a multi-commodity fixed-charge integer linear programming is established, where the constraints for both the network and the station levels are contained. Later on, the two existing solvers, RS-Opt [57] and SLIM [26], for the network level are further introduced. The experimental observations through these two solvers are discussed to elicit one of the main research topics of this thesis: effectiveness evaluation for the objective function designs. In the end, the complexity of the station-level constraints is also discussed.

Chapter 5 proposes a methodology to evaluate objective function effectiveness. Firstly, an introduction on the objective function is presented. The methodology includes three parts: design alternative objective functions, heuristic approach, obtain solutions and their features, and evaluate objective function effectiveness. The methodology is tested by real-world TUSO instances. Four alternative objective functions are evaluated. The experimental results are discussed at the end.

Chapter 6 proposes an adaptive method of expanding the network level to the station level. Firstly, the definition of coupling order is formalized, followed by relevant function and operator definitions. Thereafter, coupling order propagation on the network is introduced, in which the propagating boundaries are defined. The adaptive approach includes two parts: coupling order assignment and conflict detection. The methods of conflict resolution are hereafter introduced. The experiments are conducted based on both artificial and real-world TUSO instances.

Chapter 7 gives conclusions of this research and presents future research directions.

Chapter 2

Literature review

This chapter firstly gives an overall review of the TUSO problem studied in this thesis, including an overview on train planning, the train unit planning at the network level, the trip and train unit planning at the station level, bridging between the network and the station levels. Then the objective function evaluation problem, which is scarce in the literature, is discussed and its relevant aspects of multi-criteria optimization and parameter control are reviewed.

2.1 Overview on train planning

In the overall review of [7, 27], the whole spectrum from infrastructure-related issues to passenger and freight transportation problems, solved by operations research methods, has been discussed. Since the big difference between passenger and freight railway transport, Huisman et al. [39] specifically surveys on operational research in passenger railway transportation specific for the characteristics of the European Union, in which the train planning process at Nederlandse Spoorwegen Reizigers (NSR, which is a company operating Netherlands railway) are classified as two levels and four stages. The two levels are the central level and the local level. The central level is mainly about the entire network involved operations such as timetabling, train unit assignment and circulation. The local level is mainly about local-size-impact operations including shunting, platform assignment and routing of train units at each station. The four stages include

strategic planning, tactical planning, operational planning, and short-term planning. The strategic stage only takes place at the central planning level while all the other three stages involve both the central and local levels.

The train planning process in the UK, especially for busy passenger rail systems, is normally broken down into several stages [82]. The first stage consists of train operators drawing up outlines or drafting train plans (timetables) for running train services. This drafting process is mainly based on a few estimated factors: travel demands between origins and destinations, demand distribution throughout the day, promising revenues and costs, available rolling stock and crews, existing contracts and commitments, business or marketing plans, and the knowledge that they may have about competitors. The draft train plans are likely to contain various types of conflicts, for instance, train times, headways, line and platform usages. The second stage adjusts and fixes the conflicts that may exist in the first stage plan. The plan refinement at this stage also considers the compatibility with the plans made by other operators, because some lines and stations will be operated by more than one railway operator. More information about the passenger railway planning, scheduling, and operations is as per literature [52, 39, 21, 22, 19, 25]. Some extending surveys on AI planning are as per Wilkins [83].

As the consecutive stage of the timetabling process, the train unit scheduling problem is also addressed as two levels: the network level and the station level [60, 47]. The network level aims to solve train sequencing and fleet assignment by considering stations as simple points and temporarily ignoring station layouts. The coupling and decoupling activities are a distinct feature to satisfy passenger demands at the network flow level and redistribute train unit resources. The station level copes with finalizing the tentative plan given by the network level to fix undetermined issues because of the simplification of the station layouts. One of the research topics of this thesis focuses on how to deal with the station-level resolution to seek an operable schedule while considering detailed infrastructure. This topic has not been scrutinized in literature. Considering the problems researched in this thesis, the train unit planning problem is reviewed at two levels: the network level and the station level. The network level includes train unit assignment, train unit scheduling, and train unit circulation. The station level

contains routing trips through stations, station shunting scheduling, and depot shunting scheduling. Thereafter, some research bridging these two levels are reviewed.

2.2 Train unit planning at the network level

During the process of dealing with the train unit planning at the network level, the train unit planning at the station level is usually ignored and derived as a separate problem. The train unit planning problem at the network level determines the plan of utilizing a set of train units to serve the pre-described trips in a given timetable. There are three similar but different approaches to describe this problem, namely the train unit assignment problem (TUAP), the train unit scheduling problem (TUSP), and the train unit circulation problem (TUCP). They are attributable to the research groups in Italy, UK, and the Netherlands, respectively.

2.2.1 Train unit assignment

The train unit assignment problem (TUAP) is mainly researched by the Italian group DEIS at the University of Bologna. Given a set of timetabled trips to be performed every day, and a set of train units of different types, TUAP calls for the specification of train units to be used, and, for each of these train units, of the associated trips. Cacchiani et al. [17] first present an integer multi-commodity flow model based on a pre-generated directed acyclic multi-graph (DAG) (including arc formulation and path formulation), on which nodes represent trips (a dummy start node and a dummy end node are added) and arcs represent possible connections. Each commodity stands for a train unit type, used to partition the arcs in the DAG into subsets. The flow amount at a trip node gives the train unit combination serving that trip. For each trip, there is a passenger demand counted as the number of seats and a coupling upper bound on the number of units to be satisfied. The constraints for maintenance and overnight balance are also included. An LP-based heuristic is developed for the path formulation, where the weak knapsack constraints of coupling upper bound (no more than 2 units,

$\sum x_i \leq 2$) and passenger demands ($\sum a_i x_i \geq b$) are replaced by tighter convex hull constraints [15]. Three instances of 528, 662, 588 trips from regional Italian train operators are experimented with and without handling maintenance and overnight balance constraints. Besides, five sub-instances are derived from the original three instances by train unit type are also tested. Without maintenance and overnight balance constraints, the original three instances are solved by this heuristic method in 1932s, 2309s, 1878s, respectively. The five sub-instances are solved in a time range of 18-1590s. With these two additional constraints, the original and their sub-instances are solved in 5897-14105s and 18-8519s. It shows this heuristic can find 10-20% better solutions than the practitioners' solutions for the majority of the cases.

For the same problem described in [17], Cacchiani et al. [18] present a fast and effective heuristic method based on Lagrangian relaxation. The heuristic consists of a construction phase of solutions and a local search phase for improvement. The same instances from [17] but with fewer train unit types are tested. When there is no execution time limit, the new heuristic can find solutions for all the instances in a short time, which is seven times faster than the method proposed in [17]. However, the solution quality is 6% worse on average. When various execution-time limits are imposed, the new heuristic turns out even much faster and still provide good-quality solutions. This new heuristic is suitable for all cases in which the problem either must be solved many times (e.g., when it is integrated with other phases of railway planning) or when it must be solved very quickly (e.g., real-time planning).

2.2.2 Train unit scheduling

The train unit scheduling problem (TUSP) is mainly researched by the British group from the University of Leeds. Considering more comprehensive real-world features in the UK railway system, the train unit scheduling problem (TUSP) is firstly addressed as two levels: the network level and the station level [60, 47]. TUSP at the network level shares very similar definitions and settings with TUAP, for example, no line-based network decomposition, no pre-sequenced trips, etc. Their mathematical models are both established based on a pre-

generated directed acyclic graph. Compared to the research from DEIS, the UK group considers more real-world constraints specifically on the UK railway system: banned/restricted locations for coupling/decoupling operations, unit type compatibility, various coupling upper bound, complex turnaround time, and the avoidance of excessive and/or redundant coupling/decoupling operations. In terms of solving approaches, the DEIS group mainly uses heuristic-based methods. The British research group proposes an exact method and a hybrid method. The following gives a brief review of the research on the network-level problem of TUSP.

An integer multi-commodity flow model [56] based on a pre-generated DAG is established for the network level. An exact method, named rolling stock optimizer (referred to as RS-Opt), is proposed based on the customized branch-and-price method [57], where convex hull techniques [58] and dynamic column generation are applied. Except for normal variable branching, customized branching strategies are also designed to handle some constraints, e.g., train-family compatibility, locations banned on coupling/decoupling operations. The static and dynamic orders of applying different branching strategies are presented. Three constraints (passenger demands, train unit coupling upper bound, and car upper bound) are converted as tighter convex hull constraints to narrow down the solution space. The column generation is applied to both root and leaf nodes of the branch-and-bound tree. This is because if only the root node of the BB tree is solved by column generation, then the root may not contain all the columns needed for finding an optimal integer solution. To avoid the search trapped to local optima, a jump mechanism is embedded in RS-Opt. The computational experiments are conducted based on two groups of real-world instances from First ScotRail, a major passenger train operation in Scotland, UK. The results after applying different branching strategy orders are compared. The best strategy is to give variable branching the lowest priority, and the worst is to give it the highest priority. RS-Opt can find competitive solutions to the manual solution in a reasonable time. Also, it has good performance in reducing over-provision and empty running.

Through the observation of the solutions obtained by the model in [57], it is noticed that the solutions contain some unnecessary coupling/decoupling activi-

ties. Thus, a heuristic branch-and-bound approach exploiting flow potentials to reduce coupling/decoupling redundancy in the network-level model is proposed [59], which can be understood as a new branching strategy on coupling and decoupling activities. In the experiments, unnecessary coupling/decoupling activities are efficiently reduced. The solution obtained by this method has the potential of obtaining the same number of coupling/decoupling events as in the manual solution with better quality in other aspects. To further reduce coupling/decoupling events and solve complex turnaround time, fixed-charge variables are introduced to the basic model proposed in [57], expanding to a branch-and-price-and-cut method [54]. The complex turnaround time is considered as dynamic cuts added to the basic model. Since the newly introduced fix-charge variables largely increase the problem complexity, the basic method [57] is pre-solved and its solution is used as a warm-start to the speed up the solving process. Another feature in the UK railway system is bi-level passenger demands, classified as the desirable capacity and targeted capacity which are extracted from historic capacity provisions and passenger count surveys. Thus, a new integer multi-commodity flow model based on the previous research is proposed [53]. A branch-and-price method similar to the method in [57] is applied. Computational experiments conducted on First ScotRail class 334 instance show the effectiveness of solving the new feature.

Although the exact method is capable to deliver the optimal solution for the given objective function, it can only deliver solutions for small to medium size problem instances. According to the experiment observation, RS-Opt without fix-charge variables hardly delivers solutions when there are more than 500 trips. With fixed-charge variables, even 100 trip instances are hard to solve. Thus, a size-limited iterative method (referred to as SLIM) is proposed to amplify the power of RS-Opt [26]. SLIM has an arc controller to shrink the size of original DAG into a relatively small size as iterative input driving RS-Opt to seek a new solution. At the convergence, a sub-optimal solution will be delivered. Also, concurrent computing is applied to speed up the algorithm. Real-world datasets from Great Western Railway (a British train operating company owned by First-Group that operates the Greater Western railway franchise) have been tested. The experimental results show that when there are 4 cores, SLIM has the best performance. The comparisons between the solutions obtained by RS-Opt and

SLIM have been made. SLIM can converge to the same fleet size with RS-Opt solution in a much shorter time. Also SLIM can solve instances that RS-Opt fails to deliver the optimal solution within 12 hours.

2.2.3 Train unit circulation

Similar to TUAP and TUSP, the train unit circulation problem (referred to as TUCP) also seeks an appropriate allocation plan for train units covering all trips in a fixed timetable. The relevant objective pursued by TUCP is to provide efficient and robust trip services to passengers. Robustness means that the factors that have the potential to cause any distributions should be avoided as much as possible. For example, coupling and decoupling train units to adjust the capacity of a trip are usually considered as one of such factors. The Netherlands researchers deeply studied TUCP based on the railway network operated by NSR. Their research has been successfully applied in practice for NSR cases as a DSS. The railway networks considered in TUAP and TUSP are treated as a whole, however, NSR network is usually operated line-by-line and per day of the week, which contains relatively independent trip-service patterns. After the train unit circulation of a single day in a week has been determined, the second step is to make these single day plans to fit each other, defined as the cycling problem. TUAP and TUSP do not consider the train unit cycling problem. Therefore, TUCP is simplified to decide the number and type of train unit for trips running an individual line or a few lines and afterward to guarantee the train unit circulation is balanced over a week.

Schrijver [76] studies on the real-world TUCP for a case in Nederlandse Spoorwegen (NS) of a single line operated on a single day and firstly proposes a network model based on a directed graph $D = (V, A)$. For each location $x \in X$ and for each time t when any trip leaves or arrives at x, a vertex is formed (x, t), referred to as the arrival and departure events at stations. For any stage of any trip ride, leaving place x at time t and arriving at place y at time t', an arc from (x, t) to (y, t') is constructed, referring to trips. Flows of different commodities stand for train units of different types. The optimization target is to minimize the total number of train units concerning the given bounds on demands and capaci-

ties. Some local issues, for example, time allowances and train unit permutation restrictions regarding coupling/decoupling are not considered. The commercial mixed-integer programming solver CPLEX is used to solve this network model. Two scenarios that are single type and two types have experimented, and the result comparisons have been shown. The couping order problem has been addressed but has not been considered in their circulation model.

A concept of 'transition graph' has been introduced in the research of [6, 69] to capture coupling/decoupling and coupling order problems in TUCP. These two aspects strongly increase the complexity of TUCP. An ordered sequence of train units serving a trip is defined as a composition. A transition graph represents all the feasible compositions for each trip and the potential transition connections between compositions at the successive locations along the journey of the trip. [6] focuses on the determination of appropriate numbers of train units of different types together with their efficient circulation on a single line. An integer multi-commodity flow model with several additional constraints related to station shunting is proposed and a heuristic solution approach is developed to solve this model. The solution obtained by solving the model without composition constraints is used as a strong lower-bound for the full model. Real-life instances from NSR have experimented. [69] describes a model and a branch-and-price algorithm to solve the TUCP on two interacting lines. Variables and constraints describing unit inventories at each station are introduced to ensure there are no negative inventories. Two real-world cases are conducted, and the outcome is outperforming the commercial solver CPLEX 8.0. [31] extends the problem addresses in [69] to take into account the underway combining and splitting of trains. A line of 167 timetabled trips, which have been divided into 665 trips according to possible changes of composition underway, has experimented by CPLEX. The planners at NSR agreed that the solutions from their model are in any respect better than the manually created plans.

2.3 Trip and train unit planning at the station level

The problem of trip and train unit planning at the station level is considered as a relatively independent problem from the network level. The previous scholars normally study this problem based on a local-impact size, e.g., within a station/depot. The trip and train unit planning within a station refers to the detailed local plans on how the arrival objectives move among rail infrastructures within a location, i.e., platforms, sidings, and other tracks, to be conflict-free accomplishing necessary works and re-generate departures appropriately (correct time, correct position) with minimum operational costs. This section is going to review the literature from the following aspects: routing trips through stations, station shunting scheduling, and depot shunting scheduling.

2.3.1 Routing trips through stations

A railway station is bounded by so-called entry points and leaving points. The routing problem aims at routing all trains in the most appropriate way going through the station, given the timetable of arrivals and departures, station layouts, and the safety system. For an arriving train, the safety system reserves an inbound route from the entry point to the arrival platform; similarly, the safety system reserves an outbound route from the departure platform to the exit point for a departing train. In practice, the safety system ensures that any two reserved routes are conflict-free. In this problem, coupling/decoupling operations should be taken into consideration as well, i.e. finding two inbound routes to and one outbound route from the same platform for a coupling operation and vice versa for a decoupling operation. This problem is described in a decision support system (DSS), called DONS, developed by the Dutch researcher group, which mainly contains two complementary modules: CADANS and STATIONS. The CADANS module is developed by Schrijver and Steenbeek [77], assisting the planners to generate cyclic hourly timetables. The STATIONS module assists the planners to check whether the timetable generated by CADANS is feasible with respect to the routing of trips through railway stations and to generate

operational routing plans for trips within stations [88, 46].

To find conflict-free routes for trips, a mathematical model is established in [88] based on the Node Packing Problem (NPP), whose formal definition is as follows [62]: "Let $G = (V, E)$ is an undirected graph, where V is the nodes set and E is the edges set. Thus, a 'node packing' is a set $Q \subseteq V$ such that no edge joins two elements in Q. Then the NPP is to find a node packing of maximum cardinality." The model is solved by a branch-and-cut approach. The experiments are conducted based on a Dutch railway station with 15 platforms, roughly 100 signals and 135 sections, about 60 of which contain a switch. Both real-world timetables and randomly generated timetables are tested, and the results show the effectiveness in solving the routing problem. The approach proposed by [88] mainly focuses on the algorithms that can be incorporated into the module STATIONS, however, it is not sufficient for solving the routing problem for large Dutch railway stations such as Amsterdam Central Station (CS) and Utrecht CS. Thus, Kroon et al. [46] make a complementary, mainly focusing on computational complexity. An extensive description of this problem is analyzed regarding the safety system, and it is concluded that some sections and routes of a railway station can be ignored and only a subset of the track sections and the routing possibilities needs to be taken into account. They show that the routing problem is NP-complete as soon as each trip has three routing possibilities. However, if each train has only two routing possibilities, the problem can be solved in polynomial time. This result can be extended to the case where coupling and uncoupling of trains, certain service considerations, and a cyclic timetable have to be taken into account.

Zwaneveld et al. [89] describe this routing problem as a weighted node packing problem to improve the methods proposed by [88, 46], containing extra features of shunting decisions and route preferences. An improved branch-and-cut approach based on preprocessing is developed to solve the problem. The preprocessing identifies superfluous nodes that can be removed from the problem instance. The experiments show that this method can handle the routing problem for all the railway stations in the Netherlands efficiently, and it has been implemented in DONS. Lusby et al. [62] extend the NPP definition to be the weighted node packing problem (WNPP) by introducing weights on the nodes. The maximum node packing is the packing with maximum total weight. Thus the routing problem

is formulated as a large set-packing model, maximizing the total benefit in routing all trains. A branch-and-price approach is developed to solve this problem, focusing on dual representations of some feasible solutions. The computational experiments are conducted based on Pierrefitte-Gonesse junction in Germany with 64 block sections, 118 track sections, and track-length range 100-3900m interacting with freight, intercity, and express. The results show that this approach can give fairly good results for all 5 randomly generated timetables, which set that the trains arrive at the junction every 90-144s on average, with a horizon of 1 hour.

2.3.2 Station shunting scheduling

Shunting schedules guarantee that the assigned train units to the arrival and departure trips are operable at stations in terms of fixed timings and layouts. Shunting schedules play an important role in the railway operational system particularly on the realizability of the train unit schedule. A timetable defines arrival/departure times and platforms for each trip, and a train unit schedule assigns train unit blocks to serve each trip. And a shunting schedule ensures the train unit blocks can appear exactly where they are expected.

Tomii et al. [78] define station shunting problem: the objective of the shunting scheduling problem is to decide for trains that need shunting where the sidetracks should be assigned considering shunting times. This problem is considered as a resource-constrained project scheduling problem. An occupation of a track is regarded as a work, and a series of occupations of tracks/sidings form the project. Tracks and sidetracks are considered as resources. This problem has been divided into two subproblems: resource allocation and shunting time decision. A two-stage-search algorithm is proposed to solve this problem combining with probabilistic local search and PERT (Program Evaluation Review Technique). The candidate shunting schedules produced by the local search will be evaluated by PERT. The experiments are conducted on two actual train schedule in a middle size station with 6 tracks and 2 sidings. The results show that the proposed method succeeded in obtaining a practical solution.

Different from the shunting problem for trains defined in [78], Freling et al.

[32] defines the shunting scheduling for train units in a railway station, consisting of matching the arriving and departing shunt units, and parking these shunt units on the shunt tracks, such that the total shunting costs are minimal. The main focus is on shunting train units that are temporarily unused between two trips to sidings. A two-phase approach is proposed to solve this problem within a 24-hour time range. The first phase gives a rough match between the arrival and departure train units. The second phase further determines the detailed shunting plan at each siding for each train unit. The approach is based on a set-partitioning integer linear programming model solved by branch-and-bound together with root-only column generation, where each column corresponds to a feasible assignment plan for a siding. Two types of sidings, i.e. First-In-Last-Out (FILO) sidings (unidirectional) and free sidings (bidirectional) are considered. The model minimises the number of coupling/decoupling operations and prevents capacity overflow and unit blockage at all sidings. Instances for a station with up to 80 train units to be parked at 19 sidings have been tested. The results show that near-optimal solutions can be found, which have small gaps from the global optimality.

Kroon et al. [45] consider the two subproblems (train unit matching and siding shunting assignment) of train unit shunting planning, divided by [32], as an integral problem. An integrated approach with four models is proposed to achieve global optimality. Firstly, the FILO shunt tracks are considered in a basic model that contains an excessive number of constraints for eliminating crossing possibilities. To reduce the number of constraints for eliminating crossing possibilities, a second model aggregating crossing constraints into clique constraints is given. Then, a third model is presented to implicitly describe the clique inequalities efficiently by introducing additional variables and employing comparability graphs [35]. Thereafter, a model including the "virtual tracks" is presented such that the problem size can be further reduced. The models for FILO shunt tracks have been extended to another new model to suit the scenarios involving free tracks. Commercial mixed-integer programming software CPLEX is used to solve these models. Computational experiments on two stations of the NSR network have been conducted. One station has 15 free shunt tracks and 4 FILO shunt tracks with 50-125 train units to be parked and the other has 11 free shunt tracks and

2 FILO shunt tracks with 30-45 shunting train units. The results have shown that the aggregation methods for the crossing constraints are successful and the models are able to produce solutions of acceptable qualities with reasonable computational time.

For a single station, the research of Freling et al. [32] and Kroon et al. [45] only consider the linking between the arrival and departure train units that have to be shunted to shunt yards (shunt tracks, sidings) with connection time gaps of medium lengths. The matching for the arrival and departure units with short time gaps are supposed to be already given in the previous train unit circulation stage. The coupling order in a coupled train unit block and the parking order on a track are considered, but the influence of fixed orders in one station on other connected stations are not considered. If a train unit block with a fixed order arrives at the destination, this order may not be consistent with the order required by the destination. If this train unit block has a medium to long time gap to serve the next trip, it can be regarded as a shunting problem described in [32, 45]. But when the time gap is short, there might be a coupling order conflict that may disrupt the normal railway network operation.

The second phase of TUSP described in [60] is also presented as shunting scheduling. The first phase of [60] is a train unit assignment problem which gives train unit combination serving each trip and tentative linkages between arrivals and departures for each station. The most important optimization target is to minimize the fleet size. The second phase respects the objectives of the first phase and modifies the first phase results by resetting some linkage pairs to preserve the global optimality. The second phase models the train unit shunting problem as a 6-dimensional matching problem with a mixed-integer linear programming (MILP) formulation. The second phase includes the following elements: the matching between arrival and departure units, unit positions, parking berths (including the platforms and siding lines), parking methods, conflicts between a pair of shunting plans. Thus, a shunting plan is presented as $\pi = (u, p, v, q, b, m)$, and $a(\pi) = (T(u), T(v))$ is on behalf of a unique valid linkage $a \in A$. u and v mean arrival and departure unit; p and q denote positions and their possible choice sets for arrival unit u and departure units v; b is on behalf of parking berth and m stands for the parking method. π and $a(\pi)$ are corresponding to each

other. For each shunting plan, there is a shunting plan cost c_π and the objective function aims at minimizing the total shunting plan costs with a conflict-free principle. Due to the deterministic feature, the conflict relations between every pair of fixed shunting plans are considered to be pre-computed as input data. A promising solution approach is described for the second phase which is based on the pre-generation approach and the research of Muter et al.[66] about the column-and-dependent-row generation.

2.3.3 Depot shunting scheduling

The depot shunting problem is mainly about making a schedule that can guide train units to have their designated works such as inspection, cleaning of the cars, maintenance, etc. In the research of Nagasaki et al. [67], a system that can generate car shunting schedules in a depot automatically has been created, which contains two steps. Three features are determined at the first step by using a rule-based algorithm: the order of shunts and works, the routes of shunts, and the place to conduct the works. The second step firstly detects violations by Programming Evaluation and Review Technique (PERT) and resolves the detected condition violation. This system changes the schedule by using different methods, such as changing a track to hold, changing a route to shunt, and swapping the order of works. Depending on the conditions of the violations, the system searches for the appropriate method to resolve these violations step by step. The rule-based algorithm is suitable for reflecting the scheduler's experience, and PERT enables the system to easily and quickly manipulate a shunting schedule. A greedy algorithm is used as a trail to deal with this combinational optimization problem, in which strategy of jumping out of the local optima is applied. Three datasets are tested, however, their system can only solve one case out of three.

Wang et al. [81] establish an integer programming model to handle the depot yard arrangement whit respect to the constraints of occupation compatibility of resources, EMU operation and routine maintenance plans. The optimization targets are to minimize the unnecessary time occupying critical track sections and the total costs of shunting paths. This depot shunting problem has been transformed into a shop scheduling problem based on a topological graph of shunting

tasks with additional space and time constraints. A solution approach based on the MAX-MIN ant system is designed to solve this model and it can obtain favourable results for numerical examples.

Jacobsen et al. [44] define a shunting problem in a workshop area with different equipment, where workshops to repair trains and depot tracks to park trains are assigned considering the timings of relevant activities. The solution is a set of shunting plans for trains. The plan states when and where each train has to be shunted (including to the workshops) and ensures that sufficient crew is available for each repair. The shunting plans cannot invalidate corresponding time constraints and should be conflict-free to each other. Three heuristics are applied: Guided Local Search (GLS), Guided Fast Local Search (GFLS), and simulated annealing (SA). To generate a complete high-quality schedule, the solutions found by heuristics are used as an initial feasible solution to launch the model solved by CPLEX. The experiments are conducted for three workshop areas: the first one has 4 depot tracks and 3 workshops; the second one has 6 depot tracks and 5 workshops; the third one has 8 depot tracks and 6 workshops. Artificial instances in these three workshop areas are tested. The results show that GLS has the best performance throughout all three heuristics, also the solutions from GLS have the smallest gaps (1.68%, 1.35%, 1.21% for three workshop areas respectively) from the optimal solution found by CPLEX.

Wang et al. [80] model depot shunting problem as a 0-1 programming to determine an optimal shunting schedule for electric multiple units depot (SSED). The objective is to minimize the number of shunting movements. The constraints include track occupation conflicts, shunting routes conflicts, time durations of maintenance processes, and shunting running time. They propose an enhanced particle swarm optimization (EPSO) algorithm to solve the optimization problem. And their empirical study shows the EPSO algorithm performing better than the traditional particle swarm optimization algorithm in the aspect of optimality.

2.4 Bridging the network and the station levels

The train unit scheduling problem at the network level and the station level are not isolated. They are connected by running trips on the railway network. In re-

cent years, some researchers study bridging the two levels together. [60] proposes models for both the network flow level and the station level. The network level is a fixed-charge multi-commodity flow model, and the station level is a multi-dimensional matching model. These two models can communicate through arc variables. The network level temporarily ignores station layouts and generates a tentative train unit scheduling solution which may not be operable at the station level. Taking this tentative solution as input, the station level will re-match the linkages considering a specific shunting plan with respect to station infrastructure. Conceptually, the linkages re-matched by the station-level model will be encouraged if they are chosen by the network-level solution. Ideally, the good quality and operable solution at both levels can be achieved through this communication. A branch-and-price solver [57] for the network level is developed, and a heuristic wrapper reducing the input size as input to drive the branch-and-price solver to tackle large instances is also developed. However, the station level is still left undetermined. There is a series of very complex constraints at the station-level model which is about eliminating conflict shunting plans. Unfortunately, generating the sets of conflicting shunting plans in advance as input for the station level to apply these complex constraint is impossible because of tremendous possible combinations need to be checked. Thus, this thesis proposes new thoughts and methods to further determine the parts that have been left open by the network level to ensure a more deterministic and operable solution at the station level.

Haahr and Lusby consider integrating the rolling stock scheduling problem and the train unit shunting problem within railway depots with a simplified assumption that depot tracks are all dead-end, referred to as IRSUSP. Two similar branch-and-cut based approaches are proposed to solve the IRSUSP. The first approach is extended from the branch-and-price framework for the rolling stock scheduling problem proposed in [37], where a new routine is introduced to assess the feasibility of rolling stock schedules from a shunting perspective at all depots. The second model is inspired by the work in [31], and an additional routine to test the shunting feasibility of rolling stock schedules is added. These two approaches are referred to as route first method (RFM) and route last method (RLM) respectively. In addition, a comparison of different methodologies for solv-

ing a subproblem of the train unit shunting problem, track assignment problem, is presented. Several real-life case studies provided by DSB S-tog, a suburban train operator in the greater Copenhagen area, are tested. Computational results for the two proposed integrated frameworks highlight the need for integration on an operational planning level, where the problem is more constrained due to the initial positions of the units. It is shown that both the RFM and the RLM are capable of quickly finding good solutions and that, if optimality is required, the RLM is superior.

Zhong et al [86] consider the problem of rolling stock scheduling with maintenance requirements. A heuristic approach is proposed to solve this problem as two main stages. The first stage deals with the rolling stock scheduling problem where the maintenance restrictions are ignored. A conventional mixed-integer programming model is used and CPLEX is used to solve this model to generate multiple ranked candidates of rolling stock schedules for the first stage. The maintenance restrictions are restored at the second stage, and candidate rolling stock schedules are checked whether they are feasible. The candidate checking at the second stage is performed as an assignment problem, written in C#. These two stages are iteratively performed to avoid generating all schedules in the first stage before proceeding to the second stage. If a solution is feasible with respect to the maintenance constraints, the algorithm will be terminated. The performance of this approach is analyzed based on the computational results of real-life instances provided by the Chinese High-Speed Railway. These instances focus on the railway network within the Zhengzhou Group, which is one of the busiest railway networks in China. Compared to the manual schedule that is currently in use, the proposed approach yields far superior schedules. The optimized schedules improve rolling stock efficiency and lead to a reduction in operating cost of approximately 10.5%.

2.5 Multi-criteria optimization

To solve a real-world problem by mathematical programming, model simplification is usually applied to simplify the complexity of the problem to be easier to solve, for instance, transforming nonlinear expression to linear expression, ig-

noring non-significant factors, and converting time-related parameters to static parameters. Many real-world problems have multiple optimization criteria that can be modeled as a set of mathematical terms, denoted as $Z = \{z_1, z_2, \ldots, z_n\}$, to be considered in the objective function. These optimization criteria are often in conflict and their priorities are not easy to be determined such that these problems are usually regarded as multi-criteria optimization [29]. The studies on multi-criteria optimization are extensive in literature. Mostly, the objective function(s) of fixed optimization terms are used in multi-criteria optimization. The main focus is the performance of solution algorithms rather than establishing the confidence of objective functions, which is scarce in the literature.

There is no definite ranking order for the solutions of a multi-criteria optimization problem, i.e., no single solution can simultaneously optimize every criterion. This is because the multiple cost functions are often in conflict and do not have an optimization hierarchy. Consequently, the multi-criteria problems usually do not consider a unique optimal solution but a set of representative trade-off solutions. The classical methodology is to use priori methods [24]. Two well-known priori methods are the constraint method and the weighted summation method. The constraint method is to optimize one of the objectives and consider the others as constraints with estimated cost bounds [38]. The weighted summation method adds all terms together to be considered as a single objective function (2.1) by introducing weights λ_i, and possibly an additional condition (2.2) is added [63]. The single objective function is easier and more deterministic because it uses a trade-off numerical value to measure the solution quality. Unfortunately, the estimated cost bounds and scaled weights are hard to define. And the solution generated based on given bounds/weights may not be preferred by users. On the other hand, the Pareto front technique aims to compute a set of dominant trade-off solutions for users to choose [68]. However, the number of solutions required to accurately represent the Pareto front of a given problem increases exponentially with the number of objectives; the storage or time requirement of some quality indicators, such as hypervolume, diversity measure, and hyperarea difference, also increases exponentially with the number of objectives [51].

Many train planning and scheduling problems can also be considered as multi-criteria optimization problems. A multi-objective model is proposed to deal with

the passenger train scheduling problem [34]. Two conflicting optimization criteria are considered: railway company-view optimization criteria, passenger-view optimization criteria. They minimize fuel consumption and traveling time respectively. A two-objective integer programming model is established to describe the train utilization and operation problem in a subway system, which is solved by a genetic algorithm [85]. A multi-objective formulation is proposed to assess mainline train services considering the interests of multiple stakeholders, such as journey times, customer waiting times, punctuality, and crowdedness [23]. Similarly, a genetic algorithm is applied. The trade-off between conflicting objectives is illustrated through the Pareto analysis.

$$min \left\{ \sum_{i=1}^{n} \lambda_i z_i(x) : x \in X \right\} \tag{2.1}$$

$$\sum_{i=1}^{n} \lambda_i = 1, (0 < \lambda_i < 1, \forall i \in \{1, 2, \ldots, n\}) \tag{2.2}$$

Multi-criteria optimization in literature mainly concerns Pareto fronts and their analysis, in which conflict multi-objectives have individual objective functions that are not merged. However, the multiple objectives considered in this research are not in strong conflict and have a discernible importance hierarchy to be merged into a single weighted sum objective function.

2.6 Parameter tuning and control

The 'errors' or 'uncertainties' arises when simplification and approximations have been applied to mathematical models and solution approaches such that the reduction of uncertainties is considered during the model calibration process summarized as parameter tuning and control. After mathematical models and solution approaches are established in which the optimization terms are fixed, the studies on parameter tuning/control also draw a lot of attention. This section briefly reviews the literature on parameter control. The main focus of parameter tuning is on how to obtain a set of parameter setting which gives the algorithm good performance.

Parameter tuning and control are usually studied after objective functions are determined and solution approaches are developed. It tunes the parameters affecting the time needed of an algorithm to find an optimal solution but does not contribute to the 'definition' of what is considered as optimal. Conventionally, most mathematical algorithms have default parameter settings that are manually set in an ad-hoc manner based on the considerable effort of experiments and experience [65]. For instance, there are 80 parameters in CPLEX affecting the search mechanism that can be controlled by users [41]. During the last few decades, many automatic methods of parameter tuning and control are reported in the literature, particularly in (meta-)heuristic methods. They are classified into four types [30]: (i) sampling methods, (ii) model-based methods, (iii) screening methods, (iv) meta-evolutionary methods. Sampling methods reduce the search effort by cutting down the number of parameters. However, this leads to the challenge of predicting a limited number of parameters that have the best performance and robustness. A sampling method to systematically tune parameters (up to five) is proposed, which is based on statistical analysis and local search techniques [5]. The second type of method establishes models based on the parameter data to reduce the total number of experiments. For instance, the sequential model-based algorithm configuration (SMAC) method based on random forests is model-based [40], which expands this method to general algorithm configuration problems. Screening methods identify the best parameters with a minimum number of experiments. '*-Race' approaches are commonly used [12, 9]. Meta-evolutionary methods consider parameter tuning as an optimization problem, for example the parameter iterated local search (ParamILS) method [41] and the focused iterated local search (FocusedILS) method [42], which have inspired the configuration studies of the multi-objective problem [13].

However, only a little research discusses weight tuning for objective function terms. The analytical hierarchy process is used to assign the weights of the penalized terms in the objective function which are some soft constraints of the nurse scheduling problem [8]. Based on historical data, a method to automatically determine the relative importance of soft constraints is proposed [64]. In train planning and scheduling research, the weights of objective function terms are mostly set as pre-fixed parameters, but the method of deriving those weights are

not explained [57, 79, 31].

One focus of this thesis is to explore the effectiveness of an objective function based on an established mathematical model that has been properly simplified and a developed exact solver whose parameters are fixed. The alternative objective function designs containing different optimization terms are established. The weight of each term is set based on their importance hierarchy. However, weight tuning is beyond the discussion of this thesis.

Chapter 3

TUSO with station-level resolution

This chapter elaborates on the train unit scheduling optimization problem in detail based on the UK railway industry cases. Firstly a brief introduction on the restrictions of TUSO at the network level is presented, followed by the station-level concerns for a network-level solution. The optimization criteria for TUSO and solution qualities are then discussed. The relevant information about the TUSO problem in the UK is not well documented. The description in this research is mainly obtained from intensive communications and discussions with some railway practitioners and a software company providing solution packages for public transport management, whose staff are very familiar with the UK railway market and have years of railway operating experiences. They are Great Western Railway(GWR), TransPennine Express (TPE), Greater Anglia, as well as Tracsis Plc. However, some details may still be different from one operator to another, even for the train operating companies within the UK.

3.1 Brief requirements for the network level

Since TUSO at the network level is already well established in literature [57, 58, 55, 53, 54, 59, 26], this section only briefly presents the requirements and restrictions considered in the network level, including fleet size, passenger de-

mands, type-route/depot compatibility, turnaround time, coupling/decoupling issues. More details can be found in Lin's thesis [52].

3.1.1 Fleet size

Every train operator possesses a fleet of train units in different types. Each type has a limited number of train units. For example, TransPennine Express (TPE) has a fleet listed in Table 3.1 to operate main routes of North West England, Yorkshire and Humber, North East England, Scotland. TPE also has 14 Class 68 locomotives plus 66 Mark 5A carriages. If a schedule uses more train units

Table 3.1: TransPennine Express (TPE) fleet

Train unit type	Class 185	Class 350/4	Class 397	Class 802/2
Fleet size	51	5	12	19

than what the operator has, it is an invalid schedule. Considering the other complex auxiliary activities that a train unit must have, e.g., maintenance, and their expensive operational fees, operators would like to use train units as few as possible to satisfy the demands efficiently.

3.1.2 Passenger demands

A timetable only defines a set of logical trips, to which physical train unit(s) need to be assigned. The assigned train unit(s) are supposed to satisfy the passenger demands of every trip described in the timetable, which is an important constraint for the automatic scheduling system. The distribution of passenger demands strongly affects the allocation of train unit resources over the railway network. Thus, the utilization of train units to efficiently cover all passenger demands directly affects some optimization criteria, for instance, fleet size, mileage.

In the UK, passenger demands are usually obtained based on historical data measured as the number of seats. Every year, some surveys on the actual seats requested by some timetabled trips are carried out. The survey results are considered as a valuable factor to generate a new schedule for the next year, thus, the features of real passenger demand distribution are reserved. If a trip is covered by

some train units that are more than needed, it is defined as an over-provision trip, vice versa, an under-provision trip. In a manual schedule, it is quite often to run some under-provision trips during peak hours because of a shortage of train units. In this research, passenger demand for each trip is considered as hard constraints that are not allowed to be invalidated. In other words, the under-provision trips will not appear in the schedule. However, over-provision trips are acceptable to provide diversification for train unit allocation, which is useful to balance train unit resources over the railway network.

3.1.3 Type-route/depot compatibility

Train units and routes/depots have many configurations such that the compatibility between them must be considered. Railway routes are built based on different standards. EMU cannot run on the routes which are only designed for DMU, but BMU can run on both the electrified routes and the routes for DMU. In the UK railway industry, the maintenance work for each train unit is usually assigned to certain depots, i.e., it should return to the specified depots when it needs maintenance. Besides, some planning rules also define the matches between train units and routes/depots. Consider the TPE franchise and fleet as an example. The train units of class 185 run all routes in the areas of North and South TransPennine & TransPennine North West. The train units of class 350/4 and class 397 operate on the routes in TransPennine North West. Class 802/2 train units are used on two routes: Liverpool Lime Street - Edinburgh via Newcastle, Manchester Airport - Newcastle. And the locomotive hauled stocks (Class 68 + Mark 5A) run on the route of Liverpool Lime Street - Scarborough via Manchester Victoria. These units/vehicles are assigned to many depots. Class 185 and class 350/4 are maintained by Siemens at Ardwick depot in Manchester with a smaller facility in York. Scottish stabling points for both stocks include Corkerhill C.S.M.D. (Glasgow) and Craigentinny T.&R.S.M.D. (Edinburgh). Class 802/2 is maintained by Hitachi at Doncaster Carr and Craigentinny. Class 397 and loco-hauled sets are maintained by Alstom (on behalf of TPE) at Longsight (Manchester), Edge Hill (Liverpool), and Polmadie (Glasgow). The nature of feasible matches between train units and routes/depots can be converted as the

compatibility relations among train unit types and trips.

3.1.4 Turnaround time

Turnaround time only applies to the period connecting two trips, but not to the intermediate stops within a trip en-route. Thus, a prerequisite for any two trips to be consecutively served by the same train unit is that the time gap between the arrival time of the terminating trip and the departure time of the leaving trip should be long enough for the train unit to accomplish a series of operations at the corresponding station. For example, cleaning, watering, coupling/decoupling, and other station shunting movements. Turnaround time can be considered as four parts: basic turnaround time, formation length time, coupling/decoupling time, and station shunting time. The basic turnaround time is location and O-D based. Most train stations in the UK accept 3-5 minutes of basic turnaround time if there is no other specification about the O-D or other operations. Table 3.2 gives some exceptional examples of basic turnaround time requirements. The minimum

Table 3.2: Exceptional examples of basic turnaround time from GWR

Destination	Location	Origin	Power type	Turnaround time
			HST	7 min
OLDOHST	PADTON		HST	7 min
	PENZNCE	BRSTLTM	HST	25 min
	PLYMTH	BRNSTPL	DMU	15 min
	WSMARE	CRDFCEN	DMU	10 min

turnaround times requested by different length of train unit blocks, measured by the number of carriages, are classified differently. In GWR, 3 minutes are needed for 1-3 carriages, and 1 more minute is needed for every 3 more carriages. The formation length time should be applied if it is longer than the basic turnaround time.

On top of the basic turnaround time or the formation length time, extra buffering time may need to be allocated for coupling/decoupling operation and some other station shunting movements. The time to perform a coupling or a decoupling operation is usually defined in operational rules. In GWR for example, a coupling operation needs 5 minutes, and a decoupling operation needs 4

minutes. Figure 3.1 gives an example to calculate the minimum time needed for the coupling/decoupling operations of linkage (i, j). In this example, there are

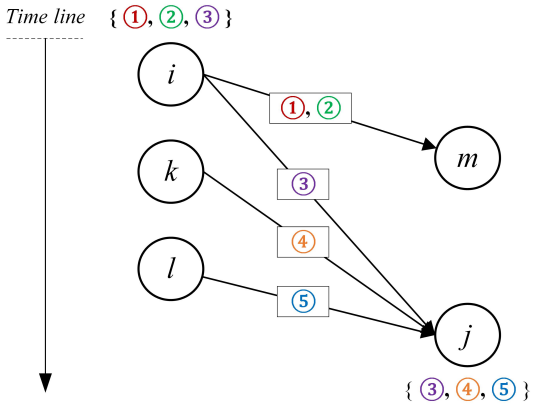


Figure 3.1: Turnaround time calculation for the connection (i, j)

5 trips and 4 connections in total. To form {③, ④, ⑤} to serve j, the train unit block {①, ②, ③} after serving trip i firstly needs to be decoupled into two shorter train unit blocks:{①, ②}, and {③}. Thereafter, train unit block {③} needs to be coupled with train unit block {④} and {⑤} sequentially. Thus, one decoupling operation (①②||③) and two coupling operations (③+④, and ③+⑤) are needed to form {③, ④, ⑤} that is used to serve j. Denote the time for a coupling operation as τ_{cpl}, and the time for a decoupling operation as τ_{dpl}. Thus, time needed for the coupling and decoupling operations (τ_{cd}) for linkage (i, j) is shown in expression (3.1).

$$\tau_{cd} = \tau_{dpl} + 2 * \tau_{cpl} \tag{3.1}$$

The basic turnaround time is considered during the process of generating DAG because it is static. The formation length and coupling/decoupling time are dynamic while forming the solution at the network level, which are considered as dynamic cuts. However, the station shunting time is omitted at the network level since the station details are ignored, which will be discussed in section 3.2.2.

3.1.5 Coupling/decoupling issues

Multiple train unit blocks can be attached to form a longer train unit block where coupling activities occur. Reversely, a multi-unit block can be detached as two or more shorter train unit blocks, where decoupling activities occurr. Coupling and decoupling activities aim to redistribute limited train units to satisfy passenger demands and balance them around the railway network. This section introduces the coupling/decoupling issues that are considered at the network level: coupling compatibility, coupling upper bounds, location restrictions on coupling/decoupling, and coupling/decoupling en-route.

Coupling compatibility: The train units of different families (examples seen in Table 1.1) are not permitted to have coupling operation. According to the observation of all problem instances studied in this thesis, train unit families partition the entire fleet into multiple mutually exclusive subsets, i.e., a train unit type is contained in a sole family as well as there is no type overlapping among any two families. Usually, the number of train unit types in a family is relatively small, and some families have only one train unit type.

Coupling upper bound: When multiple train units coupled together, the length of coupled train unit block must be considered because of the capacity of station infrastructure and some special planning rules. Many factors affect coupling boundaries, e.g. unit combination, coupling length, time band, etc. They are measured by the number of train units or carriages. Normally, the train unit coupling upper bound is 3-4, and the carriage upper bound is 12. When both unit and carriage numbers are restricted, the tightest boundary should be applied.

Location restrictions on coupling/decoupling: Coupling/decoupling activities may not be allowed at some locations because of many reasons, e.g., operations rules, infrastructural restrictions, management efficiency. Some locations may entirely ban on coupling/decoupling such that there is no such operation at any time at any place at these locations. Some locations only ban coupling/decoupling operations during certain time bands or at certain places, e.g., some platforms or a certain direction. Some locations may only allow only coupling or decoupling operated.

Coupling/decoupling en-route: In the UK railway operating process, it is nor-

mal to have coupling/decoupling operations at some intermediate stations, which change the train unit formation of a trip en-route. There are two possible reasons: the topological structure of the rail network like hubs or junctions, the unbalanced passenger demands distribution through the subsections of an O-D pair. Consider the unit diagram given in Table 1.2 as an example. The O-D of trip 1B82 and trip 1B91 is Doncaster - Manchester Airport, on which Manchester Piccadilly is an intermediate station. Trip 1B82 has a coupling en-route activity at Manchester Piccadilly, and reversely trip 1B91 has a decoupling en-route activity also at Manchester Piccadilly. Because the passenger demands of subsection Doncaster - Manchester Piccadilly are higher than that of subsection Manchester Piccadilly - Manchester Airport.

3.2 Station-level concerns for a network-level solution

Given a network-level solution, this section describes the concerns that need to be resolved and finalized when station infrastructure is restored to the model, including coupling order, linkage slack time, crossing linkages, platform type and capacity, re-platforming shunting, interchangeability between train units of the same type, and other features.

3.2.1 Coupling order

We define a coupling order as a permutation of a train unit block of multiple train units. Coupling orders are significant to be determined at the station level because many coupling/decoupling activities can only be operated in a specific order under the specific station shunting environment. Thus, the coupling/decoupling activities assigned by the network level solution can be used to infer operable coupling orders based on moving directions, timings, and station layouts. If reordering is inevitable for the sake of avoiding operational conflict, the feasibility of corresponding slack time must be considered, details seen in section 3.2.2.

Coupling order formation

If the station-level infrastructure is not considered, the coupling orders of the assigned train unit block of a trip have $n!$ possibilities, where n is the number of train units in the block. Consider the trip i in Figure 3.1 as an example, whose assigned train unit block is {①, ②, ③}, such that there are 6 potential coupling orders: [①②③], [①③②], [②①③], [②③①], [③①②], [③②①]. However, not all of them are operable at the station level. A decoupling operation must be done after the train unit block {①, ②, ③} serving trip i, and the two decoupled train unit blocks ({①, ②}, and {③}) are going to serve trips m and j respectively. Suppose trips i, m, j are at the same 'dead-end' platform, which has only one accessible end. Without considering the other coupling operations for trip j, the simplified illustration of infeasible and feasible coupling orders of train unit block {①, ②, ③} operated at a dead-end platform is demonstrated in Figure 3.2. For the infeasible coupling order case, train unit ③ blocks the

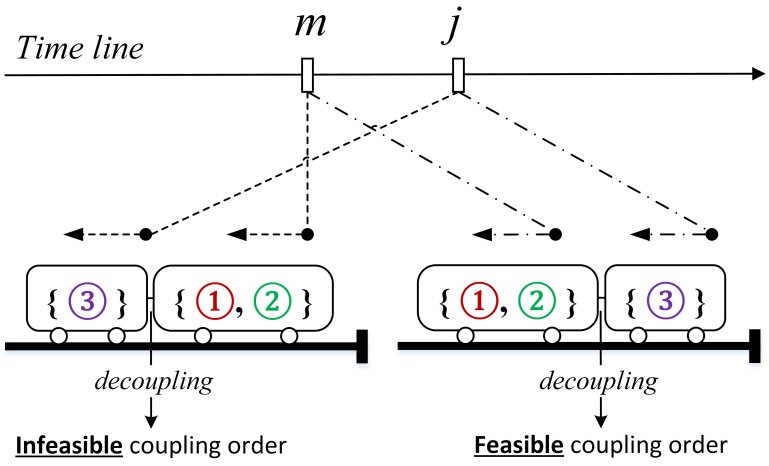


Figure 3.2: Coupling order at a dead-end platform

departure of train unit block {①, ②}, which needs to leave earlier to serve trip m. However, the latter coupling order is feasible to support the assignment that using {①, ②} to serve trip m and {③} to serve trip j, which is conflict-free. The coupling order of train unit block {①, ②} is not significant in this circumstance, because it does not cause any conflict in this decoupling operation. Thus, only two

coupling orders of train unit block {①, ②, ③} are feasible out of six potential coupling orders: [③②①], [③①②]. It is worthy to mention that these two feasible coupling orders at this circumstance may not be feasible globally, which will be explained in the paragraph of 'Coupling order propagation'.

Coupling order propagation

Train stations are not isolated in a rail network but connected by routes. Running trips concatenate the operations at the starting, intermediate, and terminating stations together. Coupling order is normally formed at the origin station of a trip, and its influence is not confined within the origin but can be propagated to other stations because of running trips and connections among trips. Thus, a coupling order formed at a specific station must be operable at other influenced stations at any time. For instance, station A proposes a locally feasible coupling order for a train unit block based on its current shunting environment. However, it may be invalid to other stations or even to station A since the shunting environment is dynamically changing by time. If a conflict caused by coupling order arises at a station, additional operations, e.g., re-ordering shunting, must be adopted to fix this invalidity. This not only increases operational costs but also may disturb/delay other trips. Thus, coupling order decisions at stations must be sophisticated.

Coupling order reversal en-route

Usually, no coupling or decoupling operation happens at intermediate stations, but one feature of the coupling order captured while considering moving directions is *front-rear-reversal*. If the arrival and departure directions of a coupled unit block are opposite at some platforms of intermediate stations, the front and rear of the unit block will be reversed; otherwise, the coupling order keeps the same. Figure 3.3 illustrates scenarios that exhibit coupling order reversal en-route of a coupled unit block based on the specific platform structure and trip directions. Thus, a trip may have different unit sequences at its origin, intermediate stations, and destination.

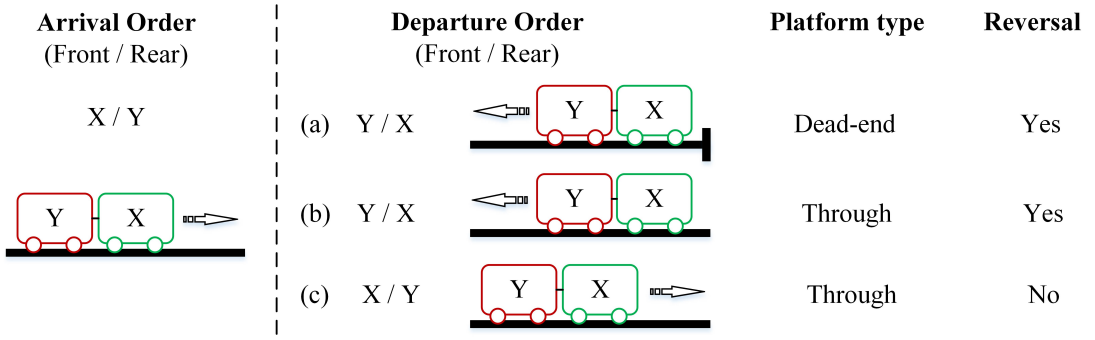


Figure 3.3: Coupling order scenarios at intermediate stations

3.2.2 Linkage slack time

The slack time of a linkage (directed arc in the DAG) is defined as the time gap between the departure time of its head node trip j (τ_j^{dep}) and the arrival time of its tail node trip i (τ_i^{arr}). The slack time ($\tau_{(i,j)}$) of linkage $a = (i, j)$ is shown in expression (3.2), where A represents all the trip-to-trip connections in the solution.

$$\tau_{(i,j)} = \tau_j^{dep} - \tau_i^{arr}, \ \forall (i,j) \in A \tag{3.2}$$

This limited slack time is the total time for the train unit block assigned on the linkage to finish all the involved manoeuvre, such as cleaning, watering, coupling/decoupling, re-ordering, re-platforming. The turnaround time needed for train unit(s) is classified as basic turnaround time (τ_a^1), formation length time (τ_a^2), coupling/decoupling time (τ_a^3), and station shunting time (τ_a^4), seen in section 3.1.4. Thus, the total turnaround time needed for a trip-to-trip linkage a is shown as expression 3.3.

$$trt_a = \{\tau_a^1, \tau_a^2\}_{max} + \tau_a^3 + \tau_a^4, \ \forall a \in A \tag{3.3}$$

The first three parts ($\tau_a^1, \tau_a^2, \tau_a^3$) are captured at the network level. τ_a^3 only counts the time consumed by the number of coupling/decoupling operations, which can be calculated by expression (3.1). The time consumed by station shunting movements (τ_a^4) is considered at the station level. The station shunting movements, e.g., forming appropriate coupling orders and re-platforming, ensure that train

units can be operated smoothly (conflict-free) in stations. At the network level, the turnaround time consumed by those auxiliary station-level operations is not accurately accounted for, i.e. insufficient slack time may invalidate a network-level solution. A linkage is locally feasible at a station if there is sufficient slack time to accomplish essential manoeuvre to avoid station conflict.

$$\Delta\tau_a = \tau_a - trt_a \geq 0, \ \forall a \in A \tag{3.4}$$

Hence, the slack time of each trip-to-trip arc in a network-level schedule must be checked by constraints (3.4) if some station-level operations are inevitable. $\Delta\tau$ must be not less than 0. Otherwise, the connection is infeasible.

3.2.3 Crossing linkages

To illustrate, suppose four single unit trips i, j, m, n are linked at a through platform at station B with no problem of unit type compatibility and seat capacities. Two possible network-level solutions are abstracted, as shown in Figure 3.4, referred as first-in-first-out (FIFO) connection and first-in-last-out (FILO) connection respectively. However, these two solutions may turn out to be in-

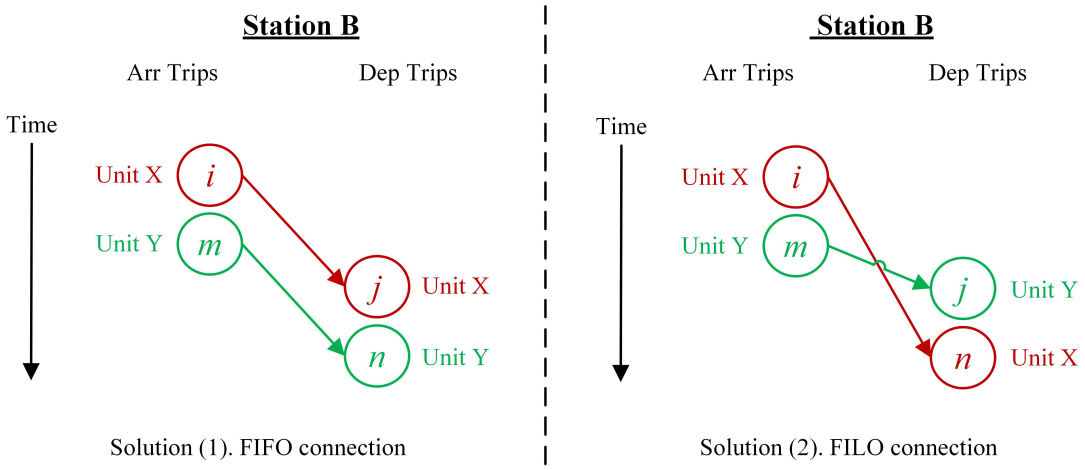


Figure 3.4: Two Possible Solutions

feasible while considering unit-block moving direction on the restricted station

Figure 3.5: Example snapshot of two units prior to their next assigned departures

tracks. Figure 3.5 shows a possible parking position of these two units at the platform before the departure of j and n. Concerning the departure time of j and n, solution (1) will be feasible if and only if j departs to the "up direction" and the solution (2) will be feasible if and only if j departs to the "down direction". Otherwise, crossing occurs as the unit assigned to n blocks the unit serving j, which is supposed to travel earlier than n. If no station information is introduced in the model, the feasibility of tentatively assigned arcs cannot be finalized. This example illustrates how crossing linkages can seriously affect the validity of a network-level schedule.

3.2.4 Platform type and capacity

During daytime scheduling, coupling and decoupling operations are often executed at platforms in the UK. Hence, the configuration and accessibility of platforms are significant to be considered. There are two main types of platforms: dead-end platform and through platform. To some degree, the platform accessibility locally decides the moving directions of train units since their movements are rigidly restricted by tracks. Dead-end platform can only be approached from one end of the track, i.e. the train units accumulated on it must follow the rule of FILO. The moving directions of approaching and leaving a dead-end platform must be opposite to each other. On the other hand, through platform can be reached via both ends, which has more complex utilization in the real-world railway management. It is common in the UK railway industry to logically divide a through platform into a few sub-platforms, for instance, Cardiff Central Station through platform 3 and 4 are used as 3A, 3B, and 4A, 4B respectively. This logical dividing gives through platforms a lot more flexibility but also increase the problem complexity. In practice, the trips are operated by many train operating companies, and platforms are often divided by train operations companies at a

station. In this thesis, we only consider the trips operated by a single company, although the incompatibilities between the trips operated by different train operating companies may theoretically cause conflicts. At the network level, the coupled unit upper bound and carriage upper bound for each trip are considered based on the limitation of the physical length of the corresponding platform. At the station level, the unit accumulation on the platform with time is considered to assure the units parking at the same platform will not invalidate the schedule.

3.2.5 Re-platforming shunting

Some train units arrive at a platform and finish all the necessary operations and later leave the station from the same platform without shunting away to any other places, for example, sidings or other platforms. However, some train unit blocks may leave from a different platform from its arrival platform, thus, re-platforming shunting must be operated. At the network level, station-level shunting operations are not realized because of the ignorance of station layouts. In the UK railway industry, schedulers try to avoid re-platforming movements and accomplish the station operations for an arrival train unit at the same platform. When a re-platforming operation is inevitable, there are two essential constraints to be considered. One is if there is enough slack time for this re-platforming operation, which goes back to τ_a^4 defined in constraints (3.3). How long this re-platforming operation takes is measured as ten minutes after discussing with railway operators. The other is that feasible time boundaries for the re-platforming train units should be considered to ensure that there will be no blockage caused by this re-platforming operation at an inappropriate time. Given timetable defines the arrival time and departure time for logical trips, however, the departure times and arrival times for the re-platforming operations on the train units to be prepared to serve another trip at stations are flexible. The existing train units parking on the platforms may result in that the involved re-platforming operations can only be operated at a specific time range. Also, re-platforming operations at different time ranges may result in different coupling orders. Considering the coupling order propagation on the network, only a specific time range may provide the feasible coupling order.

3.2.6 Interchangeability between train units of the same type

Each railway operator usually has many types of train units. The train units of the same type share common configurations such that they are interchangeable. In a schedule, each train unit has a unique diagram with detailed routes and serving trips. When a train unit is not available, a substitute of the same type can replace its mission, if the changed mission does not affect their necessary maintenance. This feature gives the shunting operations at the station level great flexibility by swapping the diagrams of the same type of train units. In this thesis, the coupling order of the train unit blocks consisting of same-type train units can be considered as conflict-free because the potential blockages can be easily avoided by swapping the missions of these train units.

3.2.7 Other features

Some factors can help determine coupling orders and ease/remove station-level conflicts. Firstly, some flexible timings for re-platforming, depot/siding shunting, and empty running can be utilized to determine the feasible time slot producing the conflict-free coupling order. This will be re-visited in section 6.1.3. Secondly, some linkage re-matching can be applied locally to resolve the local conflicts. Thirdly, introducing extra essential shunting movements within the slack times of the involved linkages can also resolve some local conflicts.

3.3 Optimization criteria and solution qualities

It is difficult to use a single objective function to satisfy all real-world optimization perspectives, but the comprehensive analysis of these perspectives is helpful to design the objective functions. In the practice of real-world scheduling, the optimization goals are multiple and complex. For a timetable that is served by a single type of train unit, the number of total needed train units plays the most important role in the objective function. When multiple types of train units are involved, operators may also care about the utilization balance between different

types. Besides, some schedulers may prefer to optimize the number of carriages or the seat over-provision. Through the fruitful discussion with railway operators, this thesis summarises the optimization criteria of TUSO as follows: fleet size, running mileage, empty running, number of coupling and decoupling, connection preference, and unit type preference.

3.3.1 Fleet size

In the UK, a train operating company is a business entity operating passenger trips on the franchised railway network with a limited number of train units. As the high costs of buying/leasing, maintaining train units, operators expect to use fewer train units to cover timetabled trips. Thus, minimizing the fleet size is the most significant criterion. Sometimes the minimization of carriage numbers is also considered when multiple train unit types of large differences in capacities are used.

3.3.2 Running mileage

Running mileage is the main operating cost of a fleet, which refers to the total distances done by the fleet to finish the one-day workload. If one train unit is sufficient to serve some trips but two or more train units (referred to as over-provision) are assigned, we consider this assignment as an inappropriate plan because train unit mileages are wasted. In practice, over-provision is allowed for some specific reasons, e.g., it is a good way of relocating train unit resources over the railway network to reduce the number of empty running and save crew resources. If the mileage minimisation is strongly encouraged, it is possible to affect the fleet size minimisation, because the extreme case of mileage minimisation is that many train units only serve one trip. Thus, the mileage minimisation is generally encouraged but there is no need to forbid the over-provision. Besides, running mileage has different calculation methods, for instance, train unit mileage, carriage mileage, and seat/passenger mileage.

3.3.3 Empty running

Empty running is necessary for many reasons such as maintenance, train unit resource balance, thus, empty running is defined when train units run from one location to another without carrying passengers. Generating new empty running may have the advantage of flexible timings. However, the new empty running must be checked and approved in advance to ensure that it does not result in any conflict on tracks at any time with other movements, e.g., timetabled trips. As the empty running is a pure cost for operators, it should be avoided if an alternative train unit utilization is available, for example, re-distribute a train unit through the network by coupling to the train unit block assigned to serve a timetabled trip such that no conflict will occur.

3.3.4 Number of coupling and decoupling

Coupling and decoupling operations are normally used to distribute unit resources across the rail network to satisfy various passenger demands with a fewer number of train units. However, these operations should not be largely encouraged as they complicate the station-level shunting operations and may lead to blockages [50, 47]. Besides, from the observation of the experiments based on the existing RS-Opt, the scheduling results at the network level will often yield many unnecessary coupling/decoupling events (further introduced in section 4.1.3), which will not occur in a manual schedule. These unnecessary coupling/decoupling operations imply considerable wastes on operational costs and unpredictable potential conflicts if the plan were implemented directly at the station level. Therefore, coupling and decoupling events should be avoided if it is evitable.

3.3.5 Connection preference

In practice, compact diagrams, in which the time gap between any pair of two consecutive trips are mostly small and medium, is preferred. Under a compact schedule, train units can be operated more efficiently covering more trips and occupying less capacity of station infrastructure. Besides, the FIFO pattern is encouraged, i.e., the train unit that arrives at a location earlier is expected to

leave from this location earlier if there is no conflict.

3.3.6 Train unit type preference

Each operator possesses a certain number of train units in different types, and each type is suitable for some specific routes according to physical configurations and operational rules. Although a train unit type may be operable on multiple routes, there usually a type-route preference defined, which is considered as the type-trip preference because of trips running on the routes. In other words, a trip can be served by multiple types of train units, but these types are not equally preferred. For example, unit types class 156 and class 158 are both permitted for running on the route between Glasgow Central and Edinburgh Waverley via Shotts, but class 156 is preferred. Besides, it has been observed from the experimental results that the existing RS-Opt is inclined to utilize the train unit with higher passenger capacity. Thus, it is significant to consider the preference of different train unit types.

3.3.7 Summary

These criteria will be converted into measurable expressions at the modeling and formulating stage. Although there are multiple optimization criteria, TUSO considers multiple criteria in a single sum weighted objective function, because these criteria are not in conflict and have a discernible importance hierarchy that can be maintained by the well-designed weights of terms in the objective function. Compared to the multi-criteria optimization problem, the single objective function is more deterministic and easier to implement. The relations between optimization objectives and solution qualities are complex. The optimization criteria are derived based on the business targets that can also be used to measure solution qualities. In return, the solution qualities imply the effectiveness of objective functions. Further relations among objective functions and solution qualities will be discussed in chapter 5.

Chapter 4

Modeling and formulating for TUSO at the network level

This section introduces the modeling strategies and mathematical formulation for TUSO at the network level. Firstly, a directed acyclic graph (DAG) is described which is generated based on a given timetable and some constraints. Based on the pre-generated DAG, a multi-commodity fixed-charge integer linear programming is established. The terms that may be considered in the objective function are modeled separately. In the end, the constraints at the network level are introduced. In the literature [57, 26], the optimization criteria of Term 1a and Term 3 are mainly modeled. In this research, many more potential optimization criteria are modeled (for instance, number of cars, slack time) and how the Term 1b, Term 2, and Term 4 play a role in delivering a good solution will be discussed in Chapter 5. Besides, how to assign the weights for different optimization criteria to maintain the discernible optimizing importance is also discussed. The constraints for the network level are mainly modeled in [57, 26]. In this research, the constraints for eliminating the station-level constraints when a network-level solution is applied at the station level are newly modeled. The method of detecting the potential conflicts located in a network-level solution will be discussed in Chapter 6.

4.1 A directed acyclic graph (DAG)

4.1.1 DAG generation and decision variables

Using a directed acyclic graph (DAG) to model TUSO at the network level is implemented in literature [15, 56]. A DAG consists of a set of nodes and a set of directed arcs such that no cycle exists, denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. The nodes set $\mathcal{N}$ consists of the timetabled trips (N), and the source and the sink ($\{s, t\}$) that are added to represent siding/depot, i.e. $\mathcal{N} = N \cup \{s, t\}$. The arcs in $\mathcal{A}$ represent potential connections between any two nodes. This representation is defined as connection-arc graph representation. DAG is generated on a given timetable and a series of real-world requirements and constraints as follows:

- Basic turnaround time: as introduced in section 3.1.4, the operational rules define some basic turnaround times, which are static such that they are considered as hard constraints while generating arcs of DAG. Usually, this basic turnaround time is 3-5 minutes, however, there are also some exceptional cases that need a longer time. For example, the trips from 'BRSTLTM' at location 'PENZNCE', the basic turnaround time needed for the train units of the power type of 'HST' is 25 minutes, seen in Table 3.2. Any arc that does not obey the basic turnaround time constraints will not be generated. The basic turnaround time may not be feasible because the dynamic turnaround time, defined in section 3.1.4 including formation length time, coupling/decoupling time, and station shunting time, changes during the train unit assignment process.

- O-D pairs of connected trips: as we defined beforehand, arcs represent potential connections between any two trips, whose origin and destination are described in the given timetable. If the tail-trip destination of an arc is the same as the head-trip origin, a trip-to-trip arc can be generated. Otherwise, this arc will not be generated or regarded as an empty running from the tail-trip destination to the head-trip origin. A sufficient time window for the empty running must be counted, for instance, the running time between two locations, the shunting time for involved operations, and

some buffering time waiting for a proper gap to insert this empty running without any congestion with other timetabled trips.

– Permitted train unit types: the compatibility between railway route and train unit type results in a set of permitted train unit types for each trip. If there is no common train unit type for two consecutive trips, the connection between these two trips cannot be generated.

Table 4.1 gives a timetable of 5 trips labeled as T1 to T5, in which O-D pair, origin, destination, departure time, arrival time, permitted unit type, and demands of each trip are provided. Two train units of compatible type X and Y are available, where X has 100 seats, and Y has 200 seats. Based on the information given in Table 4.1, an original DAG can be generated, as shown in Figure 4.1. The arcs in a DAG can be classified into three types:

Table 4.1: An example timetable of 5 trips

Trip	Origin	Destination	Dep Time	Arr Time	Permitted Type	Demands
T1	S1	S2	6:00	10:00	X	300
T2	S3	S4	8:00	9:00	Y	200
T3	S2	S4	10:15	13:30	X, Y	100
T4	S4	S5	14:00	15:00	Y	200
T5	S2	S4	15:30	17:00	X, Y	200

(i) Trip-to-trip arcs (A): this type of arcs connect the arrivals to the departures at a location, which are the most common arcs in a solution DAG.

(ii) Sign-on (A_0) and sign-off (A_∞) arcs: they connect trip nodes with the source and the sink.The arcs coming out from the source are called sign-on arcs representing the start of a train unit diagram. The arcs terminating at the sink are defined as sign-off arcs demonstrating the end of the one-day workload. Each diagram of a specific train unit starts with a sign-on arc and ends up with a sign-off arc.

(iii) Empty running arcs (A_e): in this thesis, empty running is represented as arcs in DAG. Three types of empty running are considered: empty running moving within the same station (e.g., re-platforming/siding shunting movement), empty running going to the depot and returning to the same location, and empty

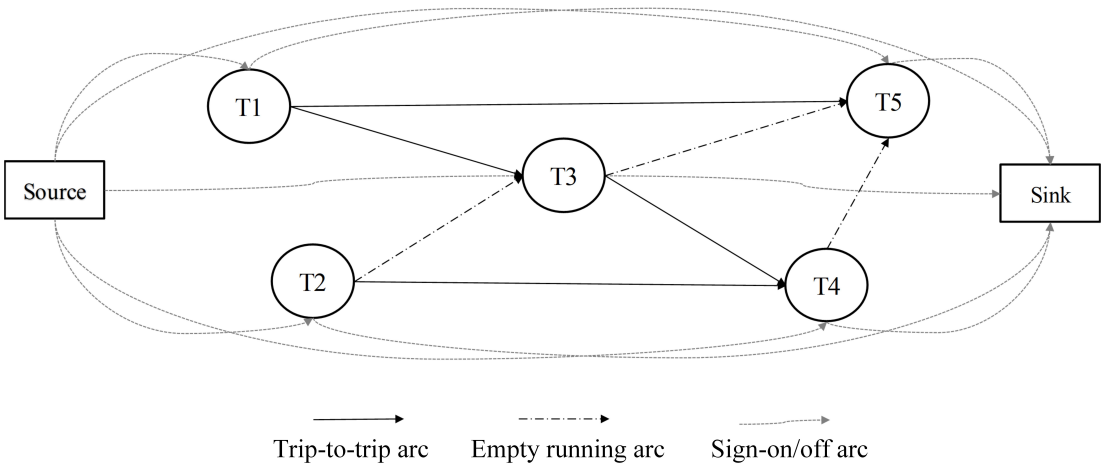
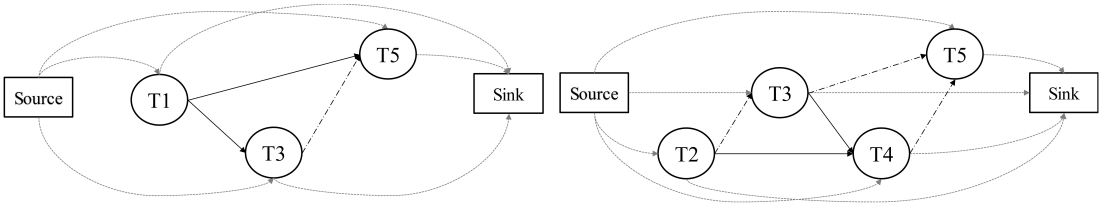


Figure 4.1: The original DAG generated from the timetable given in Table 4.1

running connecting two passenger trips at different stations. It is noticeable that generating a large number of empty running arcs will increase the complexity of the original DAG. Thus, an alternative way is to extract existing empty running from manual diagrams and insert to the DAG.

Thus, the arc set of the original DAG is $\mathcal{A} = A \cup A_0 \cup A_\infty \cup A_e$, where $A_0 = \{(0,i)|i \in N\}$, $A_\infty = \{(i,\infty)|i \in N\}$, $A = \{(i,j)|i,j \in N,$ and $s_a^i = s_d^j\}$, and $A_e = \{(i,j)|i,j \in N,$ and $s_a^i \neq s_d^j\}$. s_a^i and s_d^j represent the arrival location of trip i and the departure location of trip j respectively. Let use $\mathcal{K}$ to denote the set of train unit types used in a given timetable. Based on the compatibility between train unit type and route, type graphs $(\mathcal{G}^k, \forall k \in \mathcal{K})$ are constructed based on the original DAG $(\mathcal{G})$. Each $\mathcal{G}^k$ is a subgraph of $\mathcal{G}$. Two type graphs can be derived for train unit types of X and Y according to the original DAG in Figure 4.1 and permitted type information in Table 4.1, as shown in Figure 4.2. We use $\mathcal{P}^k$ and $\mathcal{A}^k$ to represent the path set and arc set on $\mathcal{G}^k$ respectively. A path $p \in \mathcal{P}^k$ represents a train unit diagram of type k, which contains a collection of consecutive arcs and serving trips starting from the source and ending at the sink. Based on $\mathcal{G}$ and $\mathcal{G}^k$, decision variables are defined as follows:

- arc-type-flow variables: $x_a \in \mathbb{N}$, $\forall a \in \mathcal{A}^k$, $\forall k \in \mathcal{K}$, representing the number of train units of type k flowing on arc a.



(a). Type-DAG for type X (b). Type-DAG for type Y

Figure 4.2: Type graphs derived from Figure4.1

- path-type-flow variables: $x_p \in \mathbb{N}$, $\forall p \in \mathcal{P}^k$, $\forall k \in \mathcal{K}$, representing the number of train units of type k flowing on path p.
- arc-selection (fixed charge) variables: $y_a \in \{0, 1\}$, $\forall a \in \mathcal{A}$, indicating if an arc $a = (i, j)$ is selected in the solution.

The arc-type-flow and path-type-flow variables are inter-convertible, seen in expression (4.1), where $\mathcal{P}_a$ represents all the paths containing type arc a.

$$x_a = \sum_{p \in \mathcal{P}_a} x_p, \forall a \in \mathcal{A}^k, \forall k \in \mathcal{K} \tag{4.1}$$

As the arcs in $\mathcal{G}$ carry many features in TUSO, the consideration of arc costs should be carefully exquisite. Two types of arc costs, including arc usage cost and slack time cost, are discussed as follows.

4.1.2 Arc usage cost

The arc usage cost refers to the cost of selecting one arc, denoted by c_a ($\forall a \in \mathcal{A}$). It is defined as constants according to the classifications of arc types. We consider the cost of a trip-to-trip arc as the base cost because it only implies some station-level operations, but the other types of arcs need many other operations, for example, depot shunting and empty running. Selecting a sign-on arc means the fleet size will be increased at least by one train unit. Selecting a sign-off arc means the corresponding train unit(s) will terminate its workload and do not serve other trips. Besides, sign-on and sign-off arcs are usually associated with depot/siding shunting operations. If a train unit terminates its workload too early, more train

units may be needed to cover other trips. To minimize the fleet size, it is expected to use fewer train units to satisfy all the trips in the given timetable. Thus, the costs for sign-on and sign-off arcs are higher than a trip-to-trip arc. Similarly, empty running arcs represent some trips that do not carry any passenger but a pure cost for operators such that the cost of empty running arcs should also be higher than the base cost. This type of cost is considered in the existing RS-Opt [57], in which the encouragement for long-gap arcs during off-peak time for inserting maintenance plan is also considered.

4.1.3 Slack time cost

Normally, an arrival train unit should not park at a platform for a long time. It should serve another departure trip as soon as possible or be shunted to the depot/siding. Thus, the slack time cost is newly considered to encourage the selection of the arcs with short/medium time gaps to achieve the objective of compact diagrams that enable train units to efficiently satisfy trips, denoted as $c'_{a=(i,j)}, \forall i, j \in N$. The longer the slack time is, the higher the cost is, i.e., the arcs with shorter slack time are encouraged. In the UK, maintenance works for train units are usually inserted into some long-gap connections during the off-peak time. Besides, some slack connections is helpful to reserve the robustness. Thus, some long turnaround time connections located at the off-peak time ranges are also encouraged, which has been considered in the existing RS-Opt [52] such that it will not be discussed in this thesis. As we do not consider the scheduling in depot/siding, the slack time cost is only counted for trip-to-trip arcs and empty running arcs ($A \cup A_e$). Firstly, we define the slack time cost of an arc as $\frac{\tau_{a=(i,j)}}{\tau_{max}}$, where the actual slack time (τ_a) is normalized by the maximum slack time (τ_{max}) in the original DAG. Considering that FIFO connections are more preferred than other connections (e.g., FILO connections), examples seen in Figure 3.4, the slack time cost is further defined as expression (4.2).

$$c'_{a=(i,j)} = \left(\frac{\tau_{a=(i,j)}}{\tau_{max}} \right)^2 \tag{4.2}$$

The square in expression (4.2) is to encourage FIFO connections that can be firstly proved for the connections between any four trip nodes (2-to-2) with time relation of $\tau_i^{arr} < \tau_m^{arr} < \tau_j^{dep} < \tau_n^{dep}$, as shown in Figure 4.3. Suppose these trips

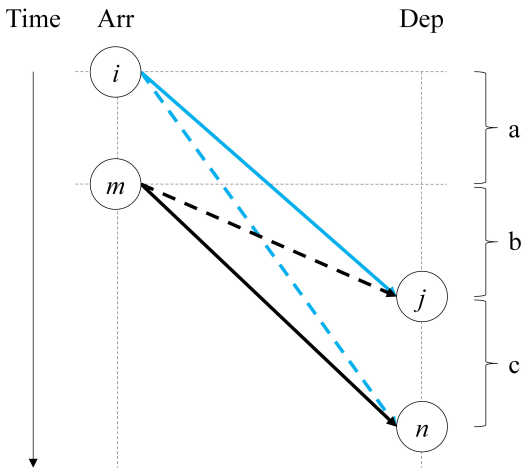


Figure 4.3: Trigonometry for four trip nodes

can be satisfied with a single train unit of the same type such that the connection pairs between these four trips are classified as two types: uncrossing pair (solid arcs) and crossing pair (dashed arcs).

Theorem 1. *If the slack time is not squared, the uncrossing pair and the crossing pair in Figure 4.3 are regarded as the same quality, i.e., they have the same summation value of corresponding slack times,* $\tau_{(i,j)} + \tau_{(m,n)} = \tau_{(i,n)} + \tau_{(m,j)}$.

Proof. $\tau_{(i,j)} + \tau_{(m,n)} = \tau_j^{dep} - \tau_i^{arr} + \tau_n^{dep} - \tau_m^{arr}$;

$\tau_{(i,n)} + \tau_{(m,j)} = \tau_n^{dep} - \tau_i^{arr} + \tau_j^{dep} - \tau_m^{arr}$;

Thus, $\tau_{(i,j)} + \tau_{(m,n)} = \tau_{(i,n)} + \tau_{(m,j)}$;

Theorem 1 holds. □

Theorem 2. *If the slack time is squared, the uncrossing pair in Figure 4.3 is encouraged, i.e., the summation value of the squared slack times of the uncrossing pair is smaller than that of the crossing pair,* $\tau_{(i,j)}^2 + \tau_{(m,n)}^2 < \tau_{(i,n)}^2 + \tau_{(m,j)}^2$.

Proof. Based on the Pythagoras formula, we have equations ① and ②;

$\tau^2_{(i,n)} - \tau^2_{(i,j)} = (a+b+c)^2 - (a+b)^2 = c^2 + 2ac + 2bc$, equation ①;

$\tau^2_{(m,n)} - \tau^2_{(m,j)} = (b+c)^2 - b^2 = c^2 + 2bc$, equation ②;

① - ②: $(\tau^2_{(i,n)} + \tau^2_{(m,j)}) - (\tau^2_{(i,j)} + \tau^2_{(m,n)}) = 2ac > 0$;

Theorem 2 holds. □

The denominator (τ^2_{max}) is not considered in the proof because it is a fixed value for a specific DAG. Based on the proof for '2-to-2' connections, it can be extended to 'n-to-n' connections, i.e., the FIFO connections always have a smaller sum of squared slack times than any other feasible connections, as long as the passenger demands of n arrival trips and n departure trips can be satisfied by one train unit such that no coupling/decoupling operations are involved. Let us denote the arrival trips by $\{i_1, i_2, \ldots, i_n\}$ and the departure trips by $\{j_1, j_2, \ldots, j_n\}$, both sorted in an ascending order by time. Suppose that the feasible FIFO connection exists for the 'n-to-n' nodes, denoted as A_{FIFO} in expression (4.3). Let $A_{notFIFO}$ be any other feasible arc connection set other than A_{FIFO}. Thus, we have Theorem 3 to be proved.

$$A_{FIFO} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_n, j_n)\} \tag{4.3}$$

Theorem 3. *For n arrival trips $i_1, i_2, \ldots, i_n$, and n departure trips $j_1, j_2, \ldots, j_n$ with the arrival time relations $\tau^{arr}_{(i_1)} < \tau^{arr}_{(i_2)} < \cdots < \tau^{arr}_{(i_n)}$ and departure time relations $\tau^{dep}_{(j_1)} < \tau^{dep}_{(j_2)} < \cdots < \tau^{dep}_{(j_n)}$, the following relations holds:*

$$\sum_{a \in A_{FIFO}} \tau^2_a < \sum_{a \in A_{notFIFO}} \tau^2_a, \quad \forall A_{notFIFO} \tag{4.4}$$

Proof. Theorem 2 shows that the uncrossing pair always has a smaller sum of squared slack time than the crossing pair. By definition, A_{FIFO} does not contain any crossing, but any $A_{notFIFO}$ contains at least one crossing, which means at least one crossing pair can be swapped by uncrossing pair with reduced the sum of squares. Therefore, A_{FIFO} has the lowest sum of squared slack times. Theorem 3 holds. □

Figure 4.4 gives an example of converting a $A_{notFIFO}$ for $n = 5$ into the A_{FIFO}. Firstly, a crossing pair (marked as red) is swapped by an uncrossing pair resulting

in $A'_{notFIFO}$. A second swap on $A'_{notFIFO}$ leads to the FIFO connection. Each swap, the sum of squares on slack time is reduced. However, the slack time cost without square cannot distinguish the differences between A_{FIFO} and $A_{notFIFO}$. Thus, we say the squared slack time cost has the advantage of encouraging FIFO connections.

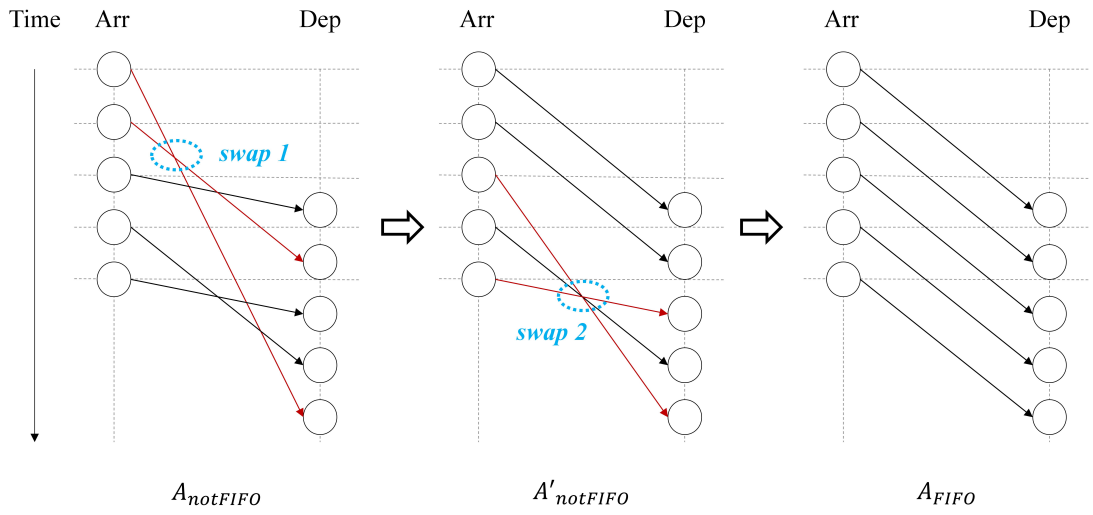


Figure 4.4: Connection swapping ($n = 5$)

For the connections involving multiple compatible train units, both types of connections demonstrated in Figure 4.5 are feasible. Although Figure 4.5(a) shows the connections we expected, the connections of Figure 4.5(b) often appear in the experimental results produced by the existing RS-Opt. We define the case of Figure 4.5(b) as 'unnecessary coupling/decoupling'. The unnecessary coupling/decoupling may be caused by the particular algorithm in RS-Opt based on the path-type-flow variables and lacking station layout information [59]. Train schedulers prefer to keep coupled train units together if there is no need to operate coupling/decoupling because any extra operation consumes resources, e.g., time, station capacity, crew. Besides, unnecessary coupling/decoupling operations increase the complexity of station shunting that may invalidate the network-level connections because of limited slack time, or that may be infeasible to operate since those operations can only be executed in a specific coupling sequence [49]. The final solution is a sub-graph of the original DAG, on which the number of

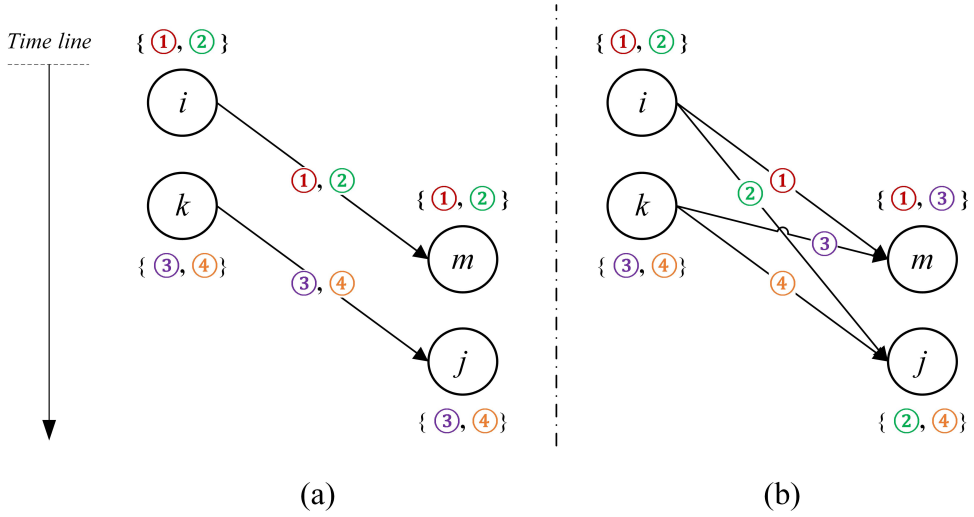


Figure 4.5: Unnecessary coupling and decoupling operations

coupling and decoupling operations can be captured by counting the get-in and the get-off arcs of each trip node. By definition, coupling/decoupling operations involve more arcs such that the consideration of arc usage cost and slack time cost can support the minimization of the total number of coupling/decoupling events, including unnecessary coupling/decoupling events.

4.2 Mathematical model

An arc at the network level implies a series of shunting operations at the station level. We have to make sure that all arcs/flows of a solution schedule are operable on the station-level infrastructure. Based on the original DAG, a multi-commodity fixed-charge integer linear programming with the station-level conflict elimination is established. The station-level constraints eliminate potential conflicts by not selecting the solutions containing infeasible arc(flow) combinations. It is an integrated model covering the constraints for TUSO at both the network level and the station level, which is an expansion based on the path-variable-based model for the network level [56, 57]. For the convenience of modeling the constraints for station-level conflicts of infeasible arc(flow) combinations, this model

uses arc-type-flow variables x_a introduced in section 4.1.1. The mathematical model includes three parts: optimization terms that may be considered in the objective function, weight assignment, and constraints.

4.2.1 Optimization terms

The optimization terms that are supposed to be considered in the objective function are formulated based on the actual optimization criteria discussed in section 3.3. Terms will be discussed one by one instead of introducing an established objective function because one research topic in this thesis is about exploring the solution quality improvement by customizing the optimization terms considered in the objective function, seen in chapter 5.

Fleet size (Term 1)

Usually, the fleet size is the most important optimization criterion for railway operators because of the expensive fees of leasing, operating, and maintaining. This criterion can be modeled from two considerations. One is to minimise the total number of train units only; the other is to consider the minimisation of train units and carriages. When there is only one train unit type, minimisation of train units is sufficient, seen in expression (4.5). The total number of train units can be obtained by adding up the flows of different types of train units on all sign-on arcs.

$$\text{Term 1a: } \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}_0^k} x_a \tag{4.5}$$

While multiple types of train units are in use, the second consideration has the advantage of minimizing the number of carriages under the condition without increasing the total number of train units. On the other hand, it can keep a balance for the utilization among different types, especially when they have large capacity differences to avoid excessive over-provision of capacities. The expression (4.6) is formulated for the second consideration, in which c_k represents the number of carriages of train unit type k. w_1 and w_2 are the weights for train units and

train carriages.

$$\text{Term 1b: } \sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}_0^k}(w_1 x_a + w_2(c_k x_a)) \tag{4.6}$$

Let us consider the usage of train units as the reference such that $w_1 = 1$, the relative value of w_2 is designed as expression (4.7) to ensure the minimisation of carriages will not affect the minimisation of train units. c_{max} refers to the maximum carriage number of a train unit. $\triangle$ is a small bias value to adjust the weight of carriage minimisation. In the UK, a train unit can have 12 carriages at the most, and $\triangle$ is usually set as 2 in this thesis.

$$w_2 = \frac{1}{c_{max} + \triangle} \tag{4.7}$$

Running mileage (Term 2)

As each arc connects with a tail node and also a head node, running mileage of train units can be captured by counting the mileage of the head node (or the tail node) of each selected arc ($m_{j(a)}$), as shown in expression (4.8). We assume that the mileage of the source and the sink node is 0. To be more flexible in terms of the operator's preference, carriage mileage, seat mileage can also be considered as optimization terms.

$$\text{Term 2: } \sum_{k\in\mathcal{K}}\sum_{a=(i,j)\in\mathcal{A}^k}(m_{j(a)} x_a) \tag{4.8}$$

Arc usage (Term 3)

According to the arc classification in section 4.1.1, the arc usage costs of sign-on/off arcs, trip-to-trip arcs, empty running arcs are formulated in this part, as shown in expression (4.9), where c_a is the arc usage cost defined in section 4.1.2.

$$\text{Term 3: } \sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}^k}(c_a x_a) \tag{4.9}$$

Slack time (Term 4)

This term is corresponding to the optimization criterion of connection preference: a compact diagram is preferred. Minimizing the total slack time is an option to guide the solver to choose short/medium connections under the condition of sufficient turnaround time. As explained in section 4.1.3, c'_a is used to encourage FIFO connections such that this term is established as expression (4.10).

$$\text{Term 4: } \sum_{k\in\mathcal{K}}\sum_{a\in\mathcal{A}^k}(c'_a x_a) \tag{4.10}$$

Coupling and decoupling events (Term 5)

This term is formulated based on the arc-selection variables y_a (defined in section 4.1.1), whose value range is given in expression (4.11). Thus, the total number of coupling and decoupling operations can be obtained by expression (4.12), where A_j^{in} and A_j^{out} represent the get-in arc set and the get-out arc set of trip j. For the locations that do not allow any coupling/decoupling operations at some time ranges, every arrival trip can only choose one get-out arc, and every departure trip can only have one get-in arc.

$$y_a = \begin{cases} 1, & \text{if } \sum_{k\in\mathcal{K}:a=a(k)} x_a \geq 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

$$\text{Term 5: } \sum_{j\in N}(\sum_{a\in A_j^{in}} y_a + \sum_{a\in A_j^{out}} y_a) \tag{4.12}$$

Train unit type preference (Term 6)

When a trip can be served by not equally preferred multiple types of train units, type preference should be introduced, which can be considered from route wise. Let use α_j^k to represent the cost of using type k to serve trip j referring a specific route. Thus, the type-cost for every trip can be demonstrated as a matrix, where the train unit type with a higher type-route preference has a lower cost. Five trips (T1-T5) that can be served by three train unit types (k_1-k_3) are considered as an example. The type-cost matrix is shown in Table 4.2. For the types that are not

Table 4.2: Cost matrix for train unit type preference

α_j^k	T1	T2	T3	T4	T5
k_1	0.1	0.3	0.3	0.2	0.3
k_2	0.2	0.1	0.5	0.5	0.4
k_1	0.7	0.6	0.2	0.3	0.3

allowed to serve some trips, the corresponding preference costs in the matrix can be considered as '∞'. As the decision variable used in the mathematical model is arc-based, the type preference costs attaching to trips can be modeled as arc cost. For each arc, we only count the type-cost for the head node or the tail node. Therefore, the optimization term of type preference cost is established as expression 4.13.

$$\text{Term 6: } \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}^k} \alpha_j^k x_{a=(i,j)} \tag{4.13}$$

Other ways of type-preference consideration include capacity wise and inventory wise. The first differs the preference based on the passenger-seat capacity, where the train unit type with a larger number of seats costs more. The second differs the preference based on the inventory of each type owned by the operator. Except for considering this criterion in the objective function, hard constraints of lower bounds for each train unit type can also be set by experienced schedulers.

4.2.2 Weights assignment

For TUSO, six terms are extracted from business optimization criteria and modeled based on the DAG. As these terms have discernible optimization hierarchy to railway operators, the objective function is constructed in a weighted summation formulation. The weight for each term can be measured based on the approximate value range of the respective term to maintain the hierarchy. Thus, we consider the lower bound of a higher hierarchy term as the upper bound of the lower hierarchy term. In this section, the weights for the first four terms (term 1 to term 4) are discussed in detail because they are the most important terms to railway operators, and they are all based on the arc-type-flow variable x_a. The objective function consisting of these four terms is written as expression (4.14), where W_1

to W_4 are the weights for the respective terms. To maintain the optimization hierarchy, the relation $W_1 \cdot Term1 > W_2 \cdot Term2 > W_3 \cdot Term3 > W_4 \cdot Term4$ should hold.

$$minimize\ W_1 \cdot Term1 + W_2 \cdot Term2 + W_3 \cdot Term3 + W_4 \cdot Term4 \tag{4.14}$$

The fleet size (the total number of train units) is the most important term and we consider it as a reference such that W_1 is set up as '1'. As the linear relaxation value of Term 1 (R) is easy to obtain and it is the lower bound of the fleet size. R is supposed to be quite close or equal to the solution fleet size (integral solution) R^*, i.e., $R \leq R^*$ always holds. Thus, we consider R as the base to derive appropriate weights of the other three terms, which will not invalidate the optimization hierarchy.

With a timetable with passenger demands, the approximate running mileage range of train units can be measured based on the train unit bounds $[u_j^{min}, u_j^{max}]$, where u_j^{min} is the minimum number of train units satisfying the passenger demand of trip j and u_j^{max} is the maximum number of train units allowed to serve trip j. Let use m_j to present the actual mileage of trip j. Thus, the approximate unit mileage range is shown in expression (4.15). Let us denote $\frac{\sum_{j\in N} u_j^{min} m_j}{\sum_{j\in N} u_j^{max} m_j}$ as γ.

$$\sum_{j\in N} u_j^{min} m_j \ < \ Term2 \ < \ \sum_{j\in N} u_j^{max} m_j \tag{4.15}$$

Relation (4.16) can be obtained.

$$\gamma \cdot R < \frac{R}{\sum_{j\in N} u_j^{max} m_j} \cdot Term2 < R \tag{4.16}$$

As $R \leq R^* = W_1 \cdot Term1$ holds, we can assign $W_2 = \frac{R}{\sum_{j\in N} u_j^{max} m_j}$. The arc usage is simplified as the number of arcs selected in the solution, denoted as $|A^*|$, whose approximate value range can be obtained by considering two extreme cases. (i) Suppose that there is only one path on which all trips in the given timetable are satisfied by a single train unit such that $|N|+1$ arcs are in use. (ii) The worst case is that there are $|N|$ paths, where each path covers a single trip such that

$2|N|$ arcs are selected. We assume the maximum allowed flow on an arc is u^{max}. Therefore, the maximum number of arcs in the solution is $2u^{max}|N|$. Since only a small portion of arcs have multiple flows and $2|N|$ is already a very extreme case, we use $|N| < |A^*| < 2|N|$ as the approximate arc usage range, which is evidenced by our experiments that all the total number of solution arcs are in this range and still quite far from $2|N|$. By doing a similar inference with the mileage wight derivation, expression (4.17) can be obtained. Thus, the weight for arc usage is $W_3 = \dfrac{\gamma \cdot R}{2|N|}$.

$$\frac{\gamma \cdot R}{2} < \frac{\gamma \cdot R}{2|N|} \cdot Term3 < \gamma \cdot R < W_2 \cdot Term2 < W_1 \cdot Term1 \tag{4.17}$$

Slack time cost for arc a is $c'_a = \left(\dfrac{\tau_a}{\tau_{max}}\right)^2$, seen in expression (4.2). Thus, term 4 can be written as $\dfrac{1}{\tau^2_{max}} \sum_{k \in \mathcal{K}} \sum_{a \in \mathcal{A}^k} \tau_a^2 x_a$. Most of the arcs selected in the solution are short to medium but τ^2_{max} is usually a very big number, e.g., it is very common for τ_{max} to be more than 1200 minutes. Therefore, W_4 is set as $\dfrac{1}{\tau^2_{max}}$ directly, where $W_4 << W_3$ such that the optimization hierarchy can be well maintained.

4.2.3 Constraints

Fleet size constraints (C_1)

For each type $k \in \mathcal{K}$, there is an upper bound b_k. To schedulers, a lower bound (b'_k) can also be considered. These constraints are as shown in expression (4.18). The upper bound constraints can be relaxed to easily find a feasible solution to start the algorithm but penalized with a big M in the objective function.

$$b'_k \leq \sum_{a \in \mathcal{A}^k_0} x_a \leq b_k, \ \forall k \in \mathcal{K} \tag{4.18}$$

Flow conservation constraints (C_2)

The flow conservation constraints are to ensure the flow of every train unit type on each trip node is balanced, where A_j^{in} and A_j^{out} are the get-in type-arc set and get-out type arc set of trip j.

$$\sum_{a \in A_j^{in}} x_a - \sum_{a \in A_j^{out}} x_a = 0, \ \forall j \in N, \ \forall k \in \mathcal{K} \tag{4.19}$$

Convex hull constraints (C_3)

Each trip $i \in N$ has to satisfy three basic hard constraints: passenger demands (C_{3a}), unit coupling upper bound (C_{3b}) and carriage upper bound (C_{3b}), presented as expressions (4.20), (4.21), and (4.22) respectively.

$$\sum_{k \in \mathcal{K}_j} \sum_{a \in A_j^{in}} q^k x_a \geq q_j, \ \forall j \in N \tag{4.20}$$

$$\sum_{k \in \mathcal{K}_j} \sum_{a \in A_j^{in}} u^k x_a \leq u_j, \ \forall j \in N \tag{4.21}$$

$$\sum_{k \in \mathcal{K}_j} \sum_{a \in A_j^{in}} v^k x_a \leq v_j, \ \forall j \in N \tag{4.22}$$

These constraints can be converted into equivalent convex hull constraints that are tighter, as shown in expression 4.23. F_j is the convex hull facets for trip j. $H_{(f,k)}^j$ and h_f^j are coefficients at type k position and the corresponding RHS of facet f for trip j. More proofs can be found in [58].

$$\sum_{k \in \mathcal{K}_j} \sum_{a \in A_j^{in}} H_{(f,k)}^j x_a \leq h_f^j, \ \forall f \in F_j, \ \forall j \in N \tag{4.23}$$

Consistency constraints between x_a and y_a (C_4)

For each arc $a \in \mathcal{A}$, if it is selected in the solution: $y_a = 1$ and $x_a \geq 1$; otherwise: $y_a = 0$ and $x_a = 0$. Here, u_a is the biggest unit number that an arc can flow.

$$\sum_{k \in \mathcal{K}: a = a(k)} x_a \leq u_a y_a, \ \forall a \in \mathcal{A} \tag{4.24}$$

Coupling/decoupling time constraints (C_5)

These constraints are to ensure that there is sufficient time for the coupling and decoupling operations, referring back to the example demonstrated in Figure 3.1. Empty-running arcs (between different locations) usually have a very long slack time where the time for coupling/decoupling operations can be assumed feasible. C_5 is not applied to sign-on and sign-off arcs as the scheduling of sidings/depots is not considered in TUSO. Moreover, some long trip-to-trip arcs have long slack times that are sufficient to operate the maximum number of coupling (C_i) and decoupling (D_j) operations allowed by the corresponding trips. Thus, if a trip-to-trip arc $a = (i, j)$ satisfies condition (4.25), there is no necessity to consider coupling/decoupling time constraint, where ζ_{ij} is the time allowance available to operate coupling/decoupling operations of arc (i, j). The trip-to-trip arcs that do not satisfy condition (4.25), denoted as A', need to dynamically consider constraints (4.26).

$$\tau_{dcpl} \cdot D_i + \tau_{cpl} \cdot C_j \leq \zeta_{ij} \tag{4.25}$$

$$\tau_{dcpl}(\sum_{a \in A_i^{out}} y_a - 1) + \tau_{cpl}(\sum_{a \in A_j^{in}} y_a - 1) \leq \zeta_{ij}, \ \forall a \in A' \tag{4.26}$$

Constraints for eliminating conflicting arc combinations (C_6)

While considering station-level infrastructure in arc selection, a selected arc is feasible if and only if it is operable within permitted time and station layouts, and a train unit schedule is feasible if every arc in the schedule is a feasible arc. Moreover, some related arcs when considered together may cause a conflict because of the factors discussed in section 3.1 and 3.2. To be more precise, a conflict is caused by the unit-type-quantity flow on a set of arcs. Therefore, the

constraints for eliminating station-level conflicts (C_6) can be modeled from two aspects, called ArcSelection constraints (C_{6a}) and TypeFlow constraints (C_{6b}) respectively. There are some advantages and disadvantages of constraints C_{6a} and C_{6b}. The ArcSelection constraints are straightforward and easy to apply. However, these constraints are slightly over-tight and they might rule out some feasible solution because they do not consider any flow combination on the target infeasible arc set. On the other hand, the TypeFlow constraints do not have that side effect and they are just tight enough to eliminate the infeasible unit-type-quantity-flow combinations on the target arc set. However, these constraints will largely increase the problem size since new variables need to introduced. For station-level conflict elimination, these two types of constraint can be applied separately or mixed. The boundaries of addressing a conflict arc sub-set will be discussed in section 6.3.

(i) ArcSelection constraints (C_{6a}): a ArcSelection conflict contains a set of arcs which together are not compatible at the station level, denoted by $\bar{A}$. If the arcs in $\bar{A}$ are not all selected in a solution, the specific conflict will not appear in the solution. Let $\mathcal{Z}_1$ be the set of possible arc selection conflicts. A conflict-free schedule must not contain any arc set $\bar{A} \in \mathcal{Z}_1$.

$$\sum_{a \in \bar{A}} y_a \leq \mid \bar{A} \mid -1, \ \forall \bar{A} \in \mathcal{Z}_1 \tag{4.27}$$

(ii) TypeFlow constraints (C_{6b}): a TypeFlow conflict represents an infeasible combination of the unit-type-quantity flow on a set of arcs due to physical railway structure. Let $\mathcal{Z}_2$ represents the set of possible type flow conflicts. These conflicts actually are integral points in the solution space. The integer cut technique proved by [10] can be applied to eliminate infeasible binary integral points, represented as $p = (x_1, \cdots, x_i, \cdots x_n)$. The infeasible integral point is divided into two sets shown in expression (4.28).

$$B = \{i \mid x_i = 1\}, \ Q = \{i \mid x_i = 0\} \tag{4.28}$$

The logical relation of this event can be converted to expression 4.29 equiv-

alent to expression 4.30.

$$\neg[(\wedge x_i, (i \in B)) \wedge (\wedge x_i, (i \in Q))] \tag{4.29}$$

$$(\vee \neg x_i, (i \in B)) \vee (\vee x_i, (i \in Q)) \tag{4.30}$$

Since the variable x_i is binary, expression (4.30) can be restrictively mapped to the integer constraints (4.31) to avoid a certain integral point.

$$\sum_{i \in B}(1 - x_i) + \sum_{i \in Q} x_i \geq 1 \tag{4.31}$$

This technique is only valid for binary variables. Necessary adaptation is derived below to apply it on an integer model. A set of working binary variables are introduced based on arc-type-flow variables, denoted as follows:

- arc-type-flow binary variables: $x_a^q \in \{0, 1\}, \forall a \in \mathcal{A}^k, \forall k \in \mathcal{K}, \forall q \in \mathcal{U}$. $\mathcal{U}$ denotes the set of natural values which can be assigned to corresponding arc-type-flow integer variable x_a.

For example, suppose the unit coupling upper bound is 3, i.e., $\mathcal{U} = \{0, 1, 2, 3\}$, and type graph $\mathcal{G}^{k_1}$ is in use. For each arc-type-flow integer variable x_a, there would be four corresponding arc-type-flow binary variables: x_a^0, x_a^1, x_a^2, and x_a^3. The binary values for them are defined as expression (4.32)

$$x_a^q = \begin{cases} 1, & \text{if } x_a = q,\ q \geq 1. \\ 0, & \text{otherwise.} \end{cases} \tag{4.32}$$

A conflict should be isolated from the entire solution if we only target to eliminate a specific type flow conflict structure. Let $\breve{A}$ represent the set of arcs containing the type flow conflict structure; B contains all the type-flow binary variables with value 1, and Q holds the other variables over the arc set $\breve{A}$. The *TypeFlow constraints* are shown in equation (4.33), where $\mathcal{Z}_2$ is the set of $\breve{A}$.

$$\sum_{B}(1 - x_a^q) + \sum_{Q} x_a^q \geq 1,\ \forall \breve{A} \in \mathcal{Z}_2 \tag{4.33}$$

The domains for each type of variables are shown at the places where they are defined.

4.3 Existing solvers for the network level of TUSO

The network level of TUSO has been researched during the last few years and many results have been achieved. This section mainly introduces the established two solvers, called RS-Opt (rolling stock optimizer) and SLIM (size-limited iterative method).

4.3.1 RS-Opt

RS-Opt is established by Zhiyuan Lin and Raymond Kwan and the related research can be found in [56, 60, 57, 58, 53, 55, 54, 59]. This section gives a brief on the mainline of RS-Opt research. The arc-based formulation is intuitive and straightforward, however, the path-based model is usually more preferable than the former because the path-based model can be solved more efficiently by applying the column generation technique [57]. Thus, an equivalent path-based formulation (4.34), denoted as (PF), is converted from the arc-based formulation. Theoretically, a path-based formulation is the result of applying Dantzig-Wolfe decomposition to a corresponding arc-based formulation although a path formulation itself is self-explanatory without the corresponding arc formulation (seeing proofs in [57]). As the transform between the arc-based model and the path-based model is only related to x_a and x_p, only the terms and constraints related to the arc-type-flow variable (x_a) are converted into (PF).

$$(PF)\ minmize \sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k} c_p x_p \tag{4.34a}$$

$$\text{subject to} \sum_{p\in\mathcal{P}^k} x_p \le b_k, \forall k \in \mathcal{K} \tag{4.34b}$$

$$\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}^k_j} H^j_{(f,k)} x_p \le h^j_f,\ \forall f \in F_j,\ \forall j \in N \tag{4.34c}$$

The consistency constraints between x_a and y_a (expression (4.24)) is converted as expression (4.35). The remaining constraints are kept the same.

$$\sum_{k\in\mathcal{K}}\sum_{p\in\mathcal{P}_a^k} x_p \leq u_a y_a,\ \forall a \in \mathcal{A} \tag{4.35}$$

The path cost c_p consists of three parts: flow cost, nodes cost, and arc cost, as shown in expression (4.36).

$$c_p = c_p^0 + \sum_{j\in N_p} c_j + \sum_{a\in A_p} c_a \tag{4.36}$$

RS-Opt is an exact method solving the problem of TUSO at the network level by a customized branch-and-price method [11], which is a technique for solving large-scale ILP problems by combining branch-and-bound (BB) method [84] and column generation [61, 28]. The branch-and-bound method is a traditional approach to solve ILPs based on partial enumeration and divide-and-conquer. The column generation technique is a well-established method to solve large-scale ILPs. It only considers a subset of variables to form a restricted master problem (RMP), and iteratively adds new variables that may improve the objective value by inspecting the results of subproblems. For the network level of TUSO, it is not possible to generate all the paths in advance because of the tremendous possibilities of node-arc combinations based on the original DAG with a large number of timetabled trips. Thus, combining the BB method and column generation technique can deal with this situation as only a small subset of potential paths need to be considered at each iteration.

Figure 4.6 shows the basic mechanism of the customized branch-and-price method of solving the path-based model (PF) [52]. The fixed-charge binary variables (y_a) largely increase the problem complexity such that the optimization terms and constraints related to y_a are firstly ignored. The LP relaxation of (PF) is regarded as the RMP, denoted as $(\bar{PF})$, which is solved by column generation at each active BB tree node. Let $\phi_k \leq 0$, $\forall k \in \mathcal{K}$ and $\psi_{(f,j)} \leq 0$, $\forall f \in F_j$, $\forall j \in N$ be the optimal dual variables associated with constraints (4.34b) and (4.34c) respectively. The subproblems can be performed for each

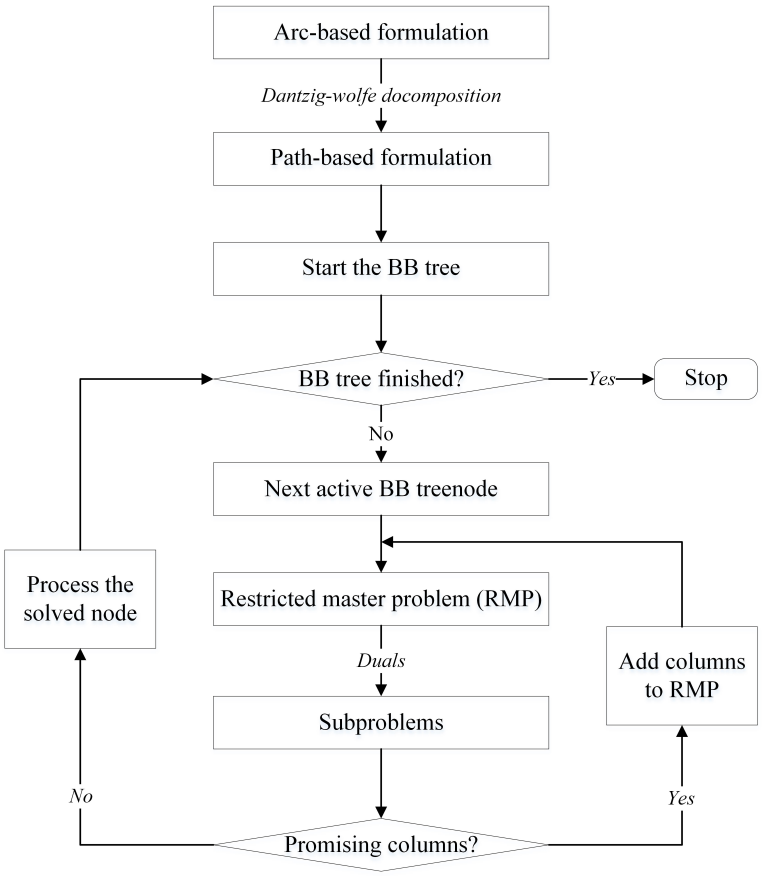


Figure 4.6: Basic mechanism of RS-Opt

k by solving the kth shortest path problem defined as expression (4.37), where $X^k = \{x \in \mathbb{Z}_+^{A^k} | M^k x^k \leq b_k\}$. $M^k x^k \leq b_k$ is a uniformed expression of constraints (4.18) and (4.19). Proofs can be found in literature [57].

$$\bar{c}_k^*(\phi, \psi) = \min_{x \in X^k} \left\{ \frac{1}{b_k} \Big(\sum_{a \in \mathcal{A}^k} c_a x_a - \sum_{j \in N, f \in F_j} \sum_{a \in A_j^{in}} H^j_{(f,k)} \psi_{(f,j)} x_a \Big) - \phi_k \right\} \tag{4.37}$$

At each iteration, a pair of upper bound and lower bound will be updated. Recording bounds has the advantage of terminating the column generation process when the gap between the upper and lower bound is regarded as acceptable. Four customized branching strategies are applied, including train-family branching, banned location branching, arc-type-flow variable branching, arc-selection

variable branching. The train-family branching is used to eliminate the incompatible matches between train unit types and routes. The banned location branching deals with some locations that do not allow coupling/decoupling operations thus each arrival can only be connected to one departure. These two branching strategies are designed to deal with the requirements discussed in section 3.1.3 and section 3.1.5. The other two branching strategies are traditional 'fractional to integral' branchings. The binary branching on the fixed-charge variables are also associated with coupling and decoupling time constraints C_5. They largely increase the problem complexity, such that the dynamic cut generation scheme is applied that will only add the specific constraints when needed [54]. The solver without y_a is considered as a warm-start for the branch-and-price-and-cut method to solve constraints C_5 [54]. Besides, the order of applying the first three branching strategies is also investigated. The experiments show that the performance is better if the arc-type-flow variable branching is executed after the other two strategies. In terms of BB tree exploration, depth- and best-first search methods are considered. Moreover, a 'jump' strategy is designed to avoid the search trapped at the middle level of the BB tree.

Except for the research of establishing RS-Opt, some other works at the network level have also been investigated. To deal with the fuzzy passenger requirement which is a very common case in the UK railway industry, a new integer multi-commodity flow model with bi-level capacity requirements is proposed based on the previous research [53]. A heuristic branch-and-bound approach based on the core RS-Opt is proposed to reduce coupling/decoupling redundancy (seeing the example demonstrated in Figure 4.5). This method exploits arcs that are not fully utilized and discovers better flow potentials to improve the unnecessary coupling/decoupling operations [59].

4.3.2 SLIM

TUSO at the network level is an NP-hard problem that is hard to solve [33, 76, 16, 57]. RS-Opt is an exact method, which has the advantage of delivering an 'optimal' solution with the price of high computational cost. The experiments show that RS-Opt without considering the fixed-charge variables can solve small

to medium size problem instances (up to 500 trips) [57]. With the fixed-charge variables, RS-Opt may fail to deliver a solution for even small-size instances (e.g., 100 trips) [54]. However, the number of trips in a real-world timetable could be very large (e.g., more than one thousand trips) such that the corresponding original DAG will be enormous.

The complexity of TUSO is because of the great number of arc-flow combinations in the original DAG. As the optimizing solution is a subgraph of $\mathcal{G}$, a heuristic wrapper named as arc controller is designed to produce compact and reduced graphs, which is used as iterative inputs for RS-Opt to seek a new solution. The hybrid method of arc controller and RS-Opt is called SLIM [26]. The framework of SLIM is shown in Figure 4.7. At iteration i, the arc controller

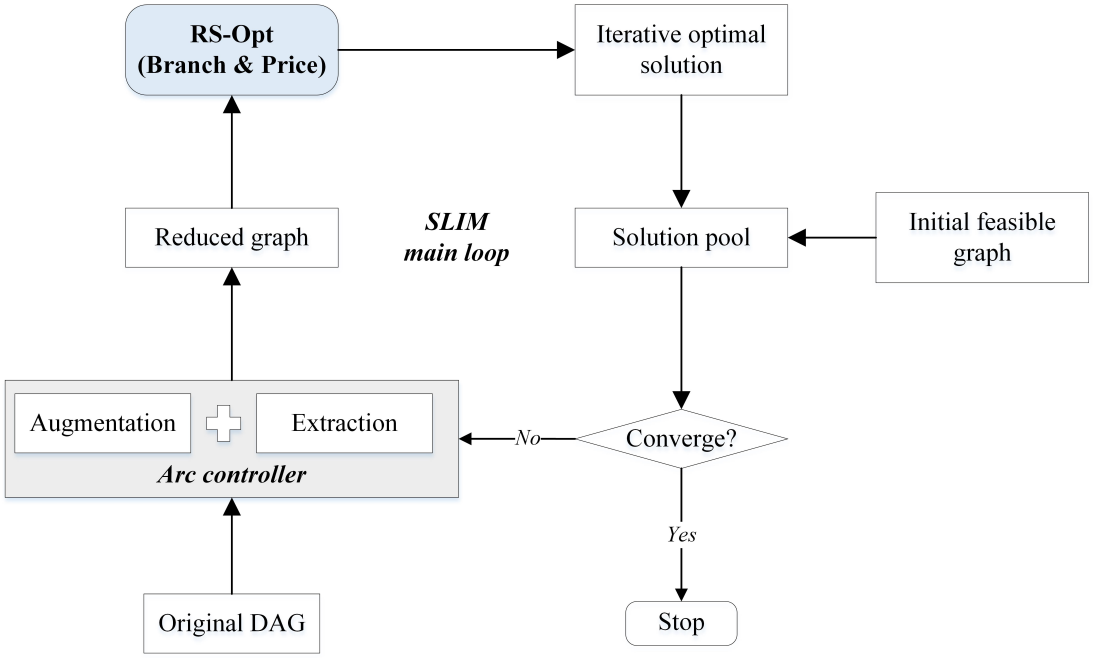


Figure 4.7: Basic mechanism of SLIM

launches RS-Opt with a reduced graph $\mathcal{G}'$. As $\mathcal{G}'$ is of small size, RS-Opt can yield an optimal solution graph $\mathcal{G}_i^*$ quickly. This new solution will be stored in the solution pool for the next iteration. Based on the solution pool and the original DAG, a new $\mathcal{G}'$ will be formed for the next iteration by the arc controller. The SLIM process is started with an initial feasible solution. The arc controller

aims at constructing a reduced graph that is sufficient for RS-Opt to produce a near-optimal solution. The arc controller contains two main processes to refine the full DAG to be reduced: Extraction and Augmentation.

(i) Extraction: this process is based on the solution pool in which all the 'optimal solutions' found by SLIM iterations are stored. At each iteration, this process extracts an essential graph $\bar{\mathcal{G}}^*$ from the solution pool to make sure there must be a feasible solution. The essential graph is the base for constructing the reduced DAG. The extraction methods can be problem-specific. For the TUSO problem, choosing the best existing solution and choosing a random solution are in consideration.

(ii) Augmentation: this process takes a small portionof arcs from the original DAG, which is called augmented arc set $\bar{\mathcal{A}}$. The reduced DAG is obtained by merging the essential DAG and the augmented arcs, i.e. $\mathcal{G}' = \bar{\mathcal{G}}^* + \bar{\mathcal{A}}$. The augmentation process secures some chances for RS-Opt to find a better solution. As the reduced DAG is much smaller than the original DAG, it is expected to deliver an 'optimal solution' very quickly. Three methods of forming augmented arcs are proposed and applied in [26]: location-based heuristic (LBP), time-based heuristic (TBH), and path-based heuristic (PBH).

SLIM has three stop criteria: maximum running time, maximum iterations, maximum iterations without improvement for objective function values. SLIM can find solutions with better objective function values than the one found by RS-Opt solely because RS-Opt allows stops with an acceptable gap between the lower bound and the upper bound. SLIM splits the original DAG into many small sub-graphs, and the optimal solution will be sought for each sub-graph. This allows SLIM to find many more acceptable integral solutions, especially near the convergence. In the experiments, SLIM can solve real-world instances that RS-Opt solely fails to deliver solutions. For some instances that are solvable by RS-Opt alone, SLIM can converge to the same or very close objective function values to the solution conveyed by RS-Opt itself.

4.4 Discussion

4.4.1 Experiment observations on the existing solvers

Given a timetable containing a set of trips, RS-Opt alone yields a single optimal solution s^*. SLIM calls RS-Opt and seeks the sub-optimal solution iteratively to form a solution set $S = \{s_1, s_2, ..., s_z\}$. Not that the solving mechanism of SLIM is the same with that of RS-Opt, and its convergence relies on the objective function value improvement that indicates if the given reduced graph contains a better solution compared to the last iteration. At the convergence of SLIM, the gap of objective function values between s_z and s^* should be very small, and these two solution DAGs should contain a high percentage of the same arcs. The percentages of arcs in s^* that are also present in the iterative solutions in S have been compared. The experiments show that SLIM and RS-Opt can usually turn out very similar objective function values, but the solution arcs are dissimilar. For some datasets, e.g., GWR-EMU, the overlapping arc percentages are only approximately 40%. The common arc percentage is not as high as our expectation, even when the iterations were leveling off with the objective values quite close to the optimal objective value yielded by RS-Opt solely. Although the branch-and-price process in the RS-Opt solver and the arc controller heuristics in SLIM may have influenced the graph arc selection, the design of objective function may have contributed to the observed behavior as well. The most likely reason could be that the existing objective function is not sensitive enough to differentiate the graph structure. This observation drives a generic work of objective function effectiveness evaluation, where the objective functions are formulated as weighted summation and the optimization criteria have discernible optimization hierarchy, seen in chapter 5.

4.4.2 Station-level constraints

Section 4.2 introduces a full model for TUSO including the constraints of the network and station levels. The two alternative constraints of eliminating station-level conflicts (shown in expressions (4.27) and (4.33)) are based on fixed-charge variables that largely increase the complexity of solving the model. Besides, com-

puting the potential conflict arc(flow) sets ($\mathcal{Z}_1$ and $\mathcal{Z}_2$) in advance as inputs based on the station-level structures and the original DAG is a massive work, because numerous of arc-selection/type-flow combinations are supposed to be checked. This makes solving the model as a whole almost impossible. On the other hand, the coupling order for coupled train unit blocks is also left to be further determined. Since many of the station-level constraints could have already been satisfied implicitly by the network flow solution, an adaptive method is proposed to determine feasible coupling orders and resolve the station-level conflicts based on the basic network flow model, seen in chapter 6.

Chapter 5

Objective function evaluation

Real-world scheduling problems, e.g., train unit scheduling, are virtually all NP-hard. Often, only near-optimal solutions could be delivered in practice. For instance, the 'exact' solver for an ILP model may have to incorporate rules to cut short a branch-and-bound search. For an objective function, the ability to differentiate as fine-grain as possible the near but sub-optimal solutions is very important. Another hurdle in designing the objective function is that real-world schedules typically have numerous possible structural properties that the practitioners would be concerned about. However, it would be impractical to elicit and incorporate all such considerations as optimisation criteria in designing the objective function. Many detailed optimisation criteria in real-world scheduling would inevitably remain 'hidden' and not explicit in the objective function.

Large and complex instances of real-world scheduling problems often need an auxiliary heuristic (but not necessarily resorting to entirely heuristics) to help an exact ILP solver to derive near-optimal solutions within a practical time. It is important for the auxiliary heuristic searches not to be wandering over a large 'poorly differentiated' solution space because of an ineffective objective function. Using small problem instances that an exact ILP method can solve to optimality, we propose a methodology to benchmark the effectiveness of alternative objective function designs. The main measure of effectiveness is in the structural similarity between the auxiliary heuristic converged solutions and the exact solver found solution. In conjunction with other solution features, the objective function effec-

tiveness is further quantified as an aggregation of several derived elements. This methodology is explained and demonstrated on a train unit scheduling problem with four alternative objective functions designed. The results show that two of them are significantly more effective than others.

5.1 Introduction

The objective function is usually a trad-off amongst a series of explicit and hidden optimization criteria and also a trade-off between complexity and accuracy. A perfectly defined objective function taking into account all aspects does not exist except for some idealized cases designed for theoretical interests. In practice, the ideal case that the designed objective function can rank all the feasible solutions precisely is hardly achieved. The real case is usually that the solutions of very similar or even the same objective function value have very different structural properties. This phenomenon also appeared in the experiments of TUSO at the network level (further explained in section 5.2). The solution approaches for a scheduling problem can be classified into two types: exact methods, and pure/hybrid heuristic methods. The exact models are normally solved using black-box commercial optimizers such as Gurobi, CPLEX, FICO. Although they claim to deliver 'exact optima', they are often computationally practical only for relatively small problem instances because of the complex combinatorial nature. The claimed 'exact optima' are usually only near but sub-optimal solutions because of not exhausted branch-and-bound tree and other simplifications to deliver the solution within a practical time. The heuristic methods can only claim 'sub-optimal' solutions, whose (near) optimality is hard to ensure. Nevertheless, heuristic methods have the advantage of delivering multiple near-optimal solutions through customized search and converging conditions.

The final solution is delivered based on the combinatorial correlations between the main optimization criteria modeled in the objective function. The complex structures in a schedule solution representing detailed operational plans. It is vital for an objective function to have the ability to differentiate solutions as fine-grain as possible especially when they are near-optimal. Thus, we propose a systematical approach to benchmark the effectiveness of alternative objective

function designs for a real-world scheduling problem that satisfying the following features and assumptions: ① Only a few main criteria that are not in conflict but have a discernible importance hierarchy to be formed in a weighted sum objective function are identified to satisfy the other hidden criteria. ② It can be modeled as a network-flow problem where the structure of how to connect nodes matters and the practitioners prefer some certain patterns of connections, for example, first-in-and-first-out connections. ③ Assume there is a ready-to-use exact solver for the considered problem and we trust the solution obtained by the exact solver is the best near-optimal solution to be considered as a solid benchmark to compare. The correlations amongst the hierarchical optimization criteria are complicated. One might synergize/conflict with another, or there may be some more complicated correlations unrevealed. Thus, what optimization criteria should be considered in the weighted-sum objective function is also significant. This question can be explained by comparing the effectiveness of alternative objective function designs that contain different main optimization criteria.

Using an exact ILP method that can solve a scheduling problem to optimality, we propose a methodology with an auxiliary heuristic of differentiating and promoting good structural properties in the solutions to investigate the effectiveness of alternative objective function designs, including three stages. ① Design alternative objective functions: the main optimization criteria potentially to be formulated into the objective function are obtained through intensive discussions with practitioners. ② Obtain solutions and their features: an exact ILP solver (P) taking the entire input is used to deliver an 'optimal' solution as the benchmark, and an auxiliary heuristic approach ($\bar{P}$) is developed to iteratively call P with a reduced input that is updated and optimized during the convergence to the final heuristic solution. Through the comparisons between the benchmark and heuristic solutions, a series of elements reflecting the objective function effectiveness are exacted. ③ Evaluate the effectiveness of alternative objective function designs: the main measure for the effectiveness is in the structural similarity between the auxiliary heuristic converged solutions and the exact solver found solution. In conjunction with other solution features, the effectiveness of an objective function is quantified as an aggregation of the solution features derived at the second stage based on the analytic hierarchy process [72].

This methodology is explained and demonstrated on the TUSO problem for which four alternative objective function designs are evaluated. The results show that the objective functions of higher effectiveness can guide $\bar{P}$ to differentiate good structural properties and converge to the solutions that have very high structural similarities with their corresponding benchmark solutions obtained by P. However, the objective functions of lower effectiveness lead $\bar{P}$ to wander over poorly differentiated solutions whose structures are far from the benchmark solution, even when the objective function value of the final solution obtained by $\bar{P}$ is very close to or identical to the optimum claimed by P. Through the effectiveness analysis for the alternative objective function designs containing different main optimization criteria, the criterion that promotes/reduces the objective function effectiveness the can also be identified.

5.2 Methodology

This section describes a methodology of evaluating the effectiveness of alternative objective function designs, which is inspired by the experimental investigation of the TUSO problem. TUSO is modeled as an integer multi-commodity flow problem on a pre-generated directed acyclic graph (DAG) that is considered as the input for solution approaches. The solution is a subgraph of the full DAG, which depicts only the train unit flows determined. Therefore, the size of the full DAG directly affects the computational complexity of solving the network flow problem because of enormous arc-and-flow combinations. An exact solver called RS-Opt has been developed for small/medium instances [57]. A size-limited iterative method called SLIM has also been developed for larger instances [26]. RS-Opt takes the entire DAG as input and delivers an 'optimal' solution s^*. SLIM aims at iteratively extracting and refining the full DAG into a small subgraph such that RS-Opt can yield a solution quickly. At convergence, a solution s_z carrying similar structural properties with s^* is expected. However, the structural properties of s^* and s_z are very different although their objective function values are very close or even the same. Although the branch-and-price process in RS-Opt may have influenced the graph arc selection, the design of objective function not agile to differentiate solution structures may have contributed to the unexpected

phenomenon such that alternative designs are considered.

Thus, a methodology with an auxiliary heuristic of promoting good structural properties in the solutions is established to evaluate the effectiveness of alternative objective functions as shown in Figure 5.1. The first stage is to design alternative

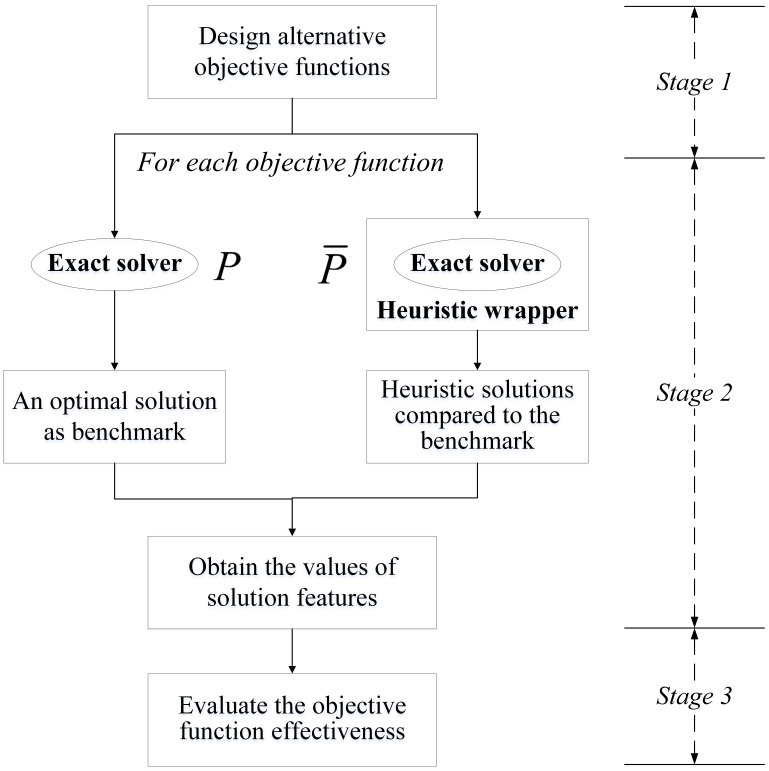


Figure 5.1: Methodology flowchart

objective functions for a specific scheduling problem. For each objective function design, this methodology utilizes an exact solver P and an auxiliary heuristic $\bar{P}$ that iteratively forms reduced inputs from the entire input for P to obtain a series of heuristic solutions. The solution from P is considered as a benchmark and the heuristic solutions obtained during the iterations of $\bar{P}$ are compared to the benchmark. Through the comparisons between the solutions obtained from these two streams, the values of solution features reflecting the effectiveness of each objective function design are obtained. In the third stage, the effectiveness of each objective function design is calculated by hierarchically integrating all

the solution features. The practitioners are involved in this stage to review if the evaluation results from the systematical methodology cooperate with their judgement.

5.2.1 Alternative objective function designs

Stage 1 aims at forming several candidate objective functions through the problem investigation. The participation of practitioners is important at this stage, especially on identifying main optimization criteria with discernible optimization importance. These optimization criteria are only a few from the complete list of criteria for the considered problem. The business optimization criteria for real-world problems are often descriptive and macroscopical. Modeling them as mathematical terms is a vital step to enable the problem to be solved in mathematical programming. As the main optimization criteria we considered respect a discernible importance hierarchy, they can be formulated in a weight-sum objective function and the weights of corresponding mathematical terms can be assigned to maintain the importance hierarchy. Considering the most important optimization criterion as the basis, the weights for the other terms can be derived based on the approximate value ranges of their respective optimization criteria. Denote the set of terms as $Terms = \{f_1, f_2, \ldots, f_n\}$, where n is the total number of optimization terms converted from the main optimization criteria. Thus, the alternative objective functions are designed based on different combinations of these optimization criteria in the weighted sum formulation. A candidate objective function can be expanded as below:

$$F_k = w_a f_a + w_b f_b + \cdots + w_m f_m, 1 \leq m \leq n \tag{5.1}$$

where m terms from $Terms$ are considered. $a, b, \ldots, m$ are chosen from the range of $[1, n]$ without repetition such that there would be C_n^m candidates with m terms. A set of alternative objective functions including different terms can be generated, represented as set $F = \{F_1, F_2, \ldots, F_k, \ldots\}$.

5.2.2 Heuristic approach

Let us define the original graph of a scheduling problem (referring to the entire input) as the entire problem space outlined by all the nodes and arcs, which maps to the entire solution space. Given a problem space, a solution is often a rather small subset of the entire input scattered in different corners of the entire problem space. This means the entire input contains plenty of information that does not contribute to the final solution but increases the computational complexity. Thus, deriving optimized reduced inputs containing good structural properties is an effective method to reduce the complexity of the entire problem. This heuristic aims at extracting an optimal reduced input enabling the exact method to deliver a solution as good as the 'optimal' solution obtained by the exact method solely. A solution containing all the nodes and arcs to convey a feasible solution flow is defined as an essential graph. An entire problem space can be divided into many small regions. Each region contains a small set of arcs that are in a certain relation, for instance they are located in the same time range. At each iteration, one/more regions are augmented to the essential graph to increase the potential of deriving some better structural properties to the solution. Suppose we arrange those regions on a wheel. Augmenting different regions on the wheel to the essential graph is defined as rotation on regions.

Thus, a heuristic approach $\bar{P}$ iteratively extracting the essential graph and rotating on regions to promote good structural properties in the reduced inputs for an exact solver P to converge the final solution is designed, as shown in Algorithm 1. $\bar{P}$ requires the entire input ($\mathcal{G}$) and a ready-to-use exact solver (P) to ensure a set of iterative solutions. Let use $soList$ to store the solutions found during the iterations of $\bar{P}$. $\bar{P}$ starts with an initial feasible solution s_0 that can be obtained by some simple heuristics for instance greedy algorithm. The $Extraction()$ method takes the $solList$ as input to extract an essential graph g^* to ensure the reduced input formed at this iteration has a feasible solution. $Extraction()$ can be defined in many customized methods. For instance, extracting the best solution in $solList$ could be an option which is helpful to speed up the convergence of $\bar{P}$. Randomly extracting any solution is also an option that has the advantage of helping $\bar{P}$ jump out of the potential local optimal solutions. According to the characteristics of a

Algorithm 1 Pseudo code of $\bar{P}$

Require: $\mathcal{G}$, P
Ensure: $solList$
1: $s_0 := initialFeasibleSolution$
2: $solList.add(s_0)$
3: **repeat**
4: $g^* := Extraction(solList)$
5: $regionList := RegionDivideMethod(g^*, \mathcal{G})$
6: **for all** g **in** $regionList$ **do**
7: $\hat{g} := g^* \cup g \longrightarrow Augmentation$
8: $s_i := P(\hat{g})$
9: $solList.add(s_i)$
10: **end for**
11: **until** $reachStopCriteria()$

specific network-flow problem, $\mathcal{G}$ can be organized in many ways to divide regions stored and arranged in $regionList$.

The loop on $regionList$ represents the rotation on regions to ensure that every corner of the entire problem space is reached. The merge between g^* and a region g is defined as $Augmentation$ and the region g is defined as an augmented region. The $Augmentation$ results in a reduced input $\hat{g}$. In each iteration, a reduced input is used to launch P such that a series of iterative solutions can be found and stored in $solList$. The augmented regions are applied systematically and the scheme for dividing the problem space is to group combinatorial features for promoting good structural properties to the solutions. The intensified search within the augmented region applied (in conjunction with the essential graph) aims at finding the suitable replacements from the augmented region for some parts of the essential graph resulting in an improved solution s_i. $\bar{P}$ will stop when one of the stopping criteria is reached, for example, maximum time/iteration, none improvement on the objective function value for a certain number of iterations. The size of a region can be controlled through a parameter μ. Logically, we usually set $0 < \mu << 1$ to make sure $|\hat{g}| << |\mathcal{G}|$ such that P can deliver an 'optimal' solution quickly. On the other hand, the benchmark solution s^* is obtained by considering the entire graph $\mathcal{G}$ as input for P, i.e. $s^* = P(\mathcal{G})$. A reduced input that is enough for P to deliver a solution as good as the benchmark solution s^* is

defined as an optimal reduced input. At convergence, $\bar{P}$ feeds an optimal reduced input to P to produce the final solution s_z. Before convergence, the reduced input at each iteration is sub-optimal but is of a very small size such that P can be executed very quickly to claim an 'optimal' solution s_i, which may be extracted as an essential graph for the next iteration. Given the entire input $\mathcal{G}$, $\bar{P}$ can find a set of iterative solutions $S = \{s_1, s_2, \ldots, s_i, \ldots, s_z\}$ during the process of skimming $\mathcal{G}$ into the optimal reduced input.

Consider a scheduling problem as an example to illustrate the working mechanism of $\bar{P}$, seen in Figure 5.2. The entire input $\mathcal{G}$ that could be massive is shown in Figure 5.2(a). Figure 5.2(b) gives an essential graph g^* that is a naive solution where every node is covered by the arcs related to the source and the sink. Figure 5.2(c) is the reduced input in which the augmented region of blue arcs are added by *Augmentation* to the essential graph shown in Figure 5.2(b). Figure 5.2(d) shows an improved solution graph obtained by P considering Figure 5.2(c) as input.

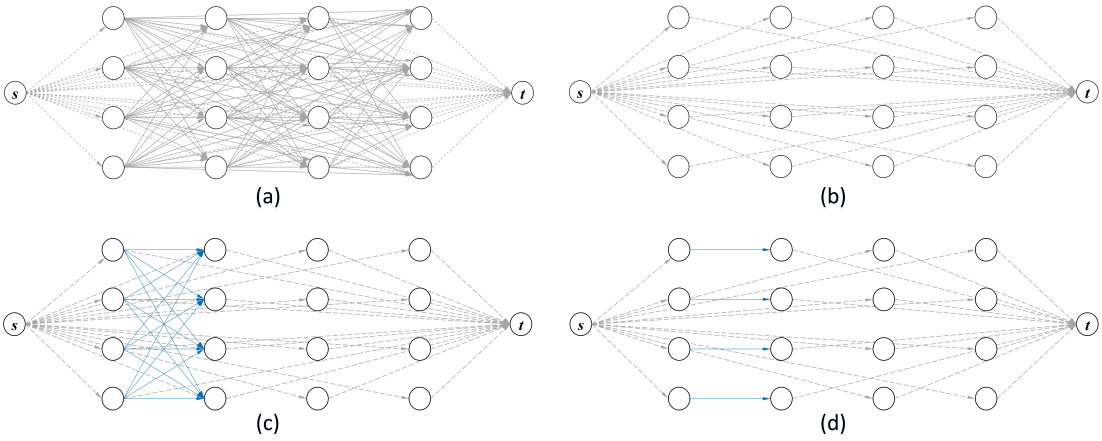


Figure 5.2: An example to illustrate the working mechanism of $\bar{P}$

5.2.3 Solution features

For a comparison between the solutions obtained from P and $\bar{P}$, four main features reflecting the objective function effectiveness are extracted: the quantified

values of the main optimization criteria, structural similarities, the number of different solutions obtained during the iterations of $\bar{P}$, the objective function values of the final solutions.

(i). The quantified values of the main optimization criteria: the objective function value guides the convergence of search algorithms, but the trade-off numerical value of multiple mathematical terms may be meaningless to practitioners. However, the actual values of the main optimization criteria carried in the solution are very important to the practitioners because they are the business targets. Thus, the optimization criteria values reflect if the corresponding objective function performs well. Some of the main optimization criteria may not play a big role in the effectiveness evaluation process. This is because they are modeled/synergized by the mathematical terms in the objective function such that they are supposed to be well minimized/maximized, i.e., a criterion may have the same value in the different solutions converged by different objective function designs. The values of all the main optimization criteria quantified from the solutions with respect to alternative objective function designs are stored in the set TV.

(ii). Structural similarities: as we assumed in section 5.1, the benchmark solution s^* found by P solely is the best near-optimal solution we can obtain. $\bar{P}$ divides the entire input into many small-size reduced inputs corresponding to many sub solution spaces. An ineffective objective function may lead $\bar{P}$ to wander over a large 'poorly differentiated' sub solution space, resulting in a heuristic solution whose structural property is far from s^* even if they have very similar or even the same objective function value. The solution to a scheduling problem is often a subgraph of the original graph. A solution schedule usually contains many diagrams where how the diagrams connect are also very important because the structural properties of a solution schedule convey the detailed operational plans. Thus, investigating the structural similarities between the heuristic solutions and s^* is very significant to benchmark the objective function effectiveness. For an objective function, all the heuristic solutions found by $\bar{P}$, $\{s_1, s_2, ..., s_z\}$, are compared to the benchmark s^*, presented in Figure 5.3.

For the structural comparisons, an effective objective function is endorsed by two indicators: very high structural similarity between s_z and s^*, roughly getting

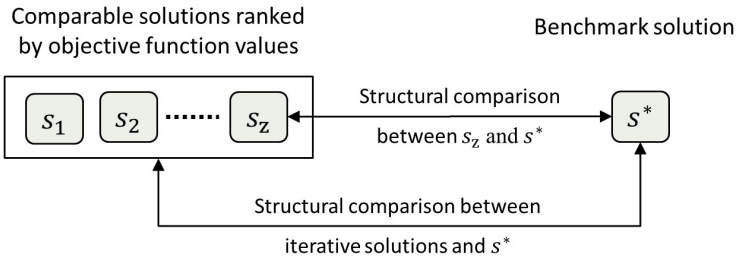


Figure 5.3: Structural comparisons

higher structural similarities between the ranked iterative solutions and s^*. Three aspects are derived from the structural comparisons. First is the structural similarity range, defined as $r = \xi^+ - \xi^-$, where ξ^+/ξ^- represents the similarity margin between the iterative solution with the best/worst objective function value and s^*. r shows how much the similarities has been increased through the iterations of $\bar{P}$. All the alternative objective functions start with the same initial feasible solution such that the lower bounds (ξ^-) of the ranges (corresponding to each objective function design) are valid to be compared. Another two aspects are the slope and the R-squared value of the linear regression fitting curve for the scatter figures. The slope demonstrates how quickly the similarity is improved through the iterative solutions of $\bar{P}$. The similarities between the ranked iterative solutions from $\bar{P}$ and s^* are supposed to be improved in line with the improvement of the objective function value such that the solutions in the scatter figures are supposed to be located close to the fitting curve. The R-squared value shows the deviations between the iterative solutions and the fitting curve. It is worthy to point out that 'R-squared equals 1' is hard to achieve for most real-world scheduling problems because of the complex nature and the trade-off during the modeling process. All the features related to the structural similarities are stored in the set DC.

(iii) The number of different solutions obtained during the iterations of $\bar{P}$: this feature investigates the capability of a given objective function to differentiate schedule properties during $\bar{P}$ iterations where the essential graphs and regions rotate on the entire problem space. We define the solutions that carry the same objective function value (with respect to the same objective function design)

but different structures as *homogeneous solutions* (*HS*). *HS* are regarded as the same quality by the numerical objective function value. Let us use an example

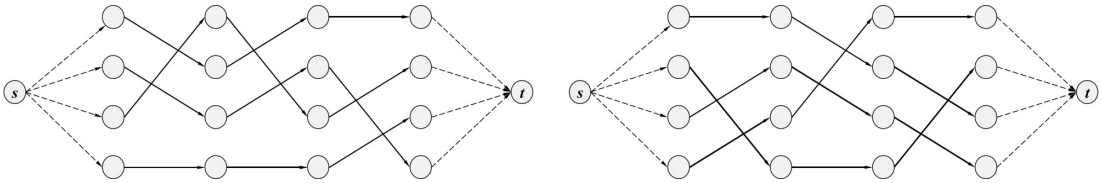


Figure 5.4: An example of homogeneous solutions

to illustrate *HS*. Suppose the nodes are customers who need to be served, and the source and the sink are added as usual. The directed arc between any two customers means they can be served consecutively. Suppose the optimization target only considers the minimum number of workers to serve all customers. For this example, the two solutions, shown in Figure 5.4, need four workers but these customers are served in very different sequences, regarded as two *HS*. When the connection lengths matter, these two *HS* could be of very different quality to practitioners. For such a case, the designed objective function is supposed to be able to differentiate as fine-grain as possible the structural properties in the solutions especially when the solutions are near but sub-optimal.

The objective function values partition and rank the solution space. Suppose the complete solution space is known. Figure 5.5 (a) and (b) show an example of a solution space in which all feasible solutions are ranked by two alternative objective functions respectively. The gray area represents the solution sector car-

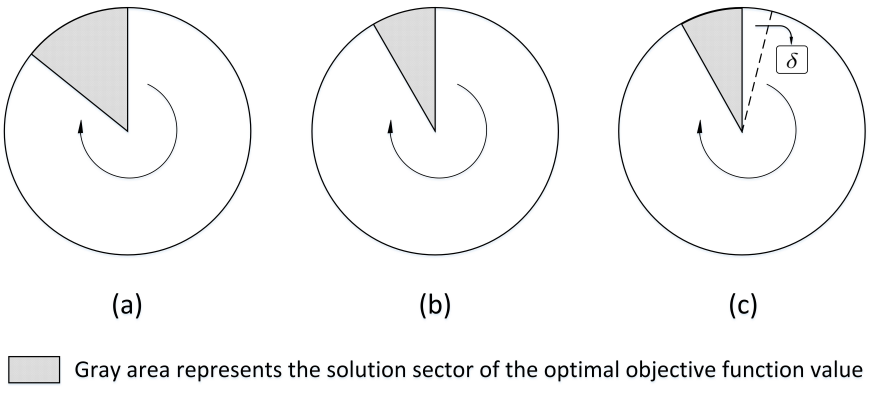


Figure 5.5: Ranked solution space

rying the optimal objective function value. The other solutions are getting worse along with the arrow direction. The size of the top-rank solution sector and hence the number of optimal *HS*, of Figure 5.5(b) is smaller than that in Figure 5.5(a), indicating that the second objective function can differentiate the structural properties better. To find all *HS* in the top sector, some exhaustive (or brute-force) search methods can be used to explore the entire solution space. However, it is very computationally expensive. Usually, solvers terminate if any solution in the top sector is found, which means the other *HS* are abandoned without further investigation. Moreover, the real optimality (upper bound = lower bound) is hard to achieve in practice because of the computational complexity. Thus, some relaxed stopping criteria must be utilized. For instance, the 'relative gap' from the optima, denoted as δ, is often calculated by $\delta = \frac{ub - lb}{lb}$, where ub and lb are upper bound and lower bound respectively. For such a case, even the exact method may return only a sub-optimal solution whose objective function value is worse than that of the solutions in the top sector. 'How much worse it could be' depends on the value of δ shown in Figure 5.5(c). This relaxation allows heuristic methods to find some solutions which have better objective function values than s^*.

In each iteration, $\bar{P}$ rotates to a different region of the entire problem space based on an essential graph to form a reduced input. Each reduced input corresponds to a solution subspace. Equivalently, $\bar{P}$ extracts many subspaces from the complete solution space. Each solution subspace is searched by P to claim one '(near-) optimal' solution as the representative in the top-rank solution sector of this subspace. Thus, $\bar{P}$ iteratively ranks the representative solutions from different solution subspaces. For some ineffective objective functions, the representative solutions for many different solution subspaces may have the same objective function value although their schedule structures are significantly different. This may lead to a quick convergence, however, too many *HS* are not a good sign for objective function effectiveness. In conclusion, for an effective objective function, it is expected to find many different iterative solutions in terms of objective function value and schedule structure, and 'less *HS* for each objective function value' is expected. These features are stored in the set SN.

(iv) The objective function values of the final solutions: the objective function

values of the final solutions from $\bar{P}$ is compared to that from P respective to each objective function design. The comparison results could be three values: CG ={*better, same, worse*}. The reason why it could be '*better*' is because of some relaxed stopping criteria, for example, the 'relative gap' introduced in Figure 5.5(c) considering $\delta > 0$.

5.2.4 Evaluation

Every solution feature reflects the effectiveness of the objective function to some extent, therefore, a scheme of integrating all solution features together is proposed to benchmark objective function effectiveness, including two steps.

(i) First is to estimate the relative importance of all the solution features based on the understanding of real-world problem and algorithm behaviors, where the analytical hierarchy process [72] is applied. This process is a widely used systematic approach for quantitatively measuring the relative importance among different factors to support the decision making for multi-criteria problems. The relative importance between any two same-layer features is measured on the basis of a 1 to 9 scale, where 1 means they are equally important and 9 means one is extremely important compared to the other one. For instance, if feature i is w ($1 \leq w \leq 9$) times more important than feature j, the relative importance of i over j is $w_{ij} = w$. Accordingly, the relative importance of j over i is $w_{ji} = 1/w$. Hence, a comparison matrix can be obtained, generally written as a square matrix W below, in which $w_{ij} = 1/w_{ji}$. To obtain the importance weight vector, each element in W is firstly normalized by its column summation to be converted as W^{nor}, i.e. $w'_{ij} = w_{ij} / \sum_{i=1}^{n} w_{ij}$.

$$W = \begin{bmatrix} 1 & w_{12} & \dots & w_{1n} \\ w_{21} & 1 & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{n1} & w_{n2} & \dots & 1 \end{bmatrix} \longrightarrow W^{nor} = \begin{bmatrix} w'_{11} & w'_{12} & \dots & w'_{1n} \\ w'_{21} & w'_{22} & \dots & w'_{2n} \\ \dots & \dots & \dots & \dots \\ w'_{n1} & w'_{n2} & \dots & w'_{nn} \end{bmatrix}$$

The estimated weight vector is calculated by $w_i = \dfrac{\sum_{j=1}^{n} w'_{ij}}{n}$ based on W^{nor}. The consistency ratio (CR) refers to the reliability of the measured weights through

the pairwise comparison method. Generally, the measured weights are acceptable if $CR \leq 10\%$ [73]. The features reflecting the effectiveness of objective function have two layers, and each feature at the first layer has some sub-features, as explained in section 5.2.3. The global weight of each sub-feature is its local weight multiplied by the local weight of its parent feature, denoted as $\hat{w}_i$.

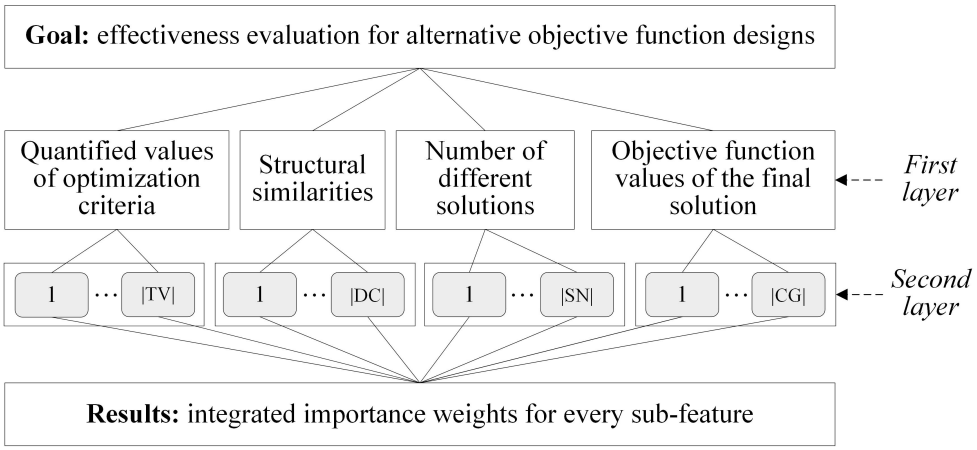


Figure 5.6: Hierarchical structure of solution features

(ii) The second is to quantify the integrated effectiveness for alternative objective function designs based on the derived solution features. The magnitudes of solution features may be different such that their values should be normalized. On the other hand, each feature reflects the objective function effectiveness in either a positive or negative way. Consider the 'minimization' problem as a reference. If the statement 'smaller feature value endorses better objective function effectiveness' is true, this feature reflects the objective function effectiveness in a positive way, stored in the set PF. On the contrary, this feature is in a negative way, stored in the set NF. The integrated effectiveness of objective function k is calculated by unifying all the features in both PF and NF together, as shown in expression (5.2), where v_i is the normalized value of feature i. Thus, the smallest value of P_k indicates the best effectiveness of objective function F_k.

$$P_k = \sum_{i \in PF} \hat{w}_i v_i - \sum_{i \in NF} \hat{w}_i v_i, \forall k \in F \tag{5.2}$$

5.3 Computational experiments

5.3.1 Problem preparation and initialization

A train unit has a fixed number of passenger carriages, which cannot be split but can be coupled with other train units. TUSO assigns a limited number of train units to cover all the trips in a timetable satisfying a series of constraints such as passenger demands, unit-route compatibility, turnaround time, etc. The optimization target is to obtain a set of unit diagrams (referring to schedule) with minimal operational costs, describing the serving sequences of trips and some auxiliary activities, for example, coupling/decoupling operations. TUSO is described as an integer multi-commodity network flow model based on a directed acyclic graph (DAG) [57], in which multi-commodity refers to different types of train unit. Nodes represent trips, a source, and a sink. Arcs represent potential connections among nodes that are generated according to real-world requirements and constraints. A path on the DAG represents a unit diagram from the source to the sink. The objective function candidates including different potential optimization criteria are shown in Table 5.1. F_1 is the base objective

Table 5.1: Objective function candidates

Objectives	Fleet size	Arc usage	Mileage	Compact diagram
F_1	Yes	Yes	No	No
F_2	Yes	Yes	Yes	No
F_3	Yes	Yes	No	Yes
F_4	Yes	Yes	Yes	Yes

* F_1 is the base objective function embedded in the exact solver P

function embedded in the developed exact solver P. Compared to F_1, every other objective function contains some extra term(s). Fleet size is usually the most important target because of expensive leasing and maintaining fees. Mileage implies fuel consumption. A compact diagram means there should not have many dramatic long turnaround time arcs chosen in the solution. The weights are assigned based on their approximate ranges. The importance hierarchy is maintained by considering the minimum value of a more important term as the maximum value of a less important term.

5.3.2 Solution feature information

The solution features of TUSO are listed in Table 5.2. TUSO is modeled based on DAG and its solution is a sub-graph of the full DAG. Thus, the structural similarity of the TUSO problem is measured by the arc overlapping between the compared two solutions. The integrated weights of sub-features (column $\hat{w}_i$) and how each sub-feature reflects the objective function effectiveness (column PF/NF) are also given. The consistency ratio for the first layer features is CR = 9.5%, and CR values for the second layer features are 0%, 2.6%, 0.47%, 1% respectively. They are in the acceptable range indicating the reliability of the estimated weights.

Table 5.2: Details of solution features

NO.	Solution features		PF/NF	$\hat{w}_i$
1	TV	Fleet size	PF	0.2479
2		Mileage		0.1239
3		Arc usage		0.0620
4		Compact diagram		0.1239
5	SC	Structural similarity	NF	0.1490
6		Range		0.0673
7		R^2		0.0259
8		Slop		0.0259
9	SN	In terms of OF value	NF	0.0163
10		In terms of structure		0.0306
11		Ave # of HS	PF	0.0862
12	CG	Better	PF	0.0225
13		Same		0.0134
14		Worse		0.0068

5.3.3 Experimental results

This section reports on the computational experiments for investigating the effectiveness of alternative objective function designs for TUSO. Although $\bar{P}$ is practical for large instances, experiments using small instances are conducted so that the exact solver P alone is able to yield benchmark solutions. Thus, three small real-world datasets shown in Table 5.3 are used. These datasets are from

Greater Anglia operating the East Anglia franchise in the UK and providing many commuter/regional services throughout the East of England. The exact

Table 5.3: Datasets information

Dataset	No. of trips	Full DAG (arc #)	No. of O/D	Unit type
D1	109	1080	11	class 755/3
D2	133	1684	12	class 755/4
D3	137	1514	12	class 360/4

solver P (RS-Opt [57]) is coded in FICO Xpress-MP 8.5 with Mosel, and the auxiliary heuristic $\bar{P}$ (SLIM [26]) is written in C#. RS-Opt and SLIM have been upgraded to carry the features for the objective function effectiveness evaluation, for instance, multiple alternative objective function designs, well-measured wights to maintain the discernible importance of different optimization criteria, compact diagram, and mileage. The experiments are conducted on a 64-bit workstation with 16G RAM and an Intel Core i7-6700 CPU. RS-Opt has a customized branch-and-price strategy and only utilizes the simplex solver of Xpress-MP to solve LP-relaxation during the column generation process without employing the default integer programming solver provided by Xpress-MP [57]. One of the stopping criteria of RS-Opt is defined as the 'relative gap' explained in Figure 5.5(c) such that SLIM may converge to the solutions whose objective function values are better than the benchmark solution claimed by RS-Opt solely. SLIM is set to run 10 times and each time runs a maximum of 3000 iterations. A greedy method is employed to construct the initial feasible solution to start the process ensuring that each candidate objective function has the same start point.

Solution feature results

For SLIM, running time is not considered as an indicator for the objective function effectiveness because a random method is used in $Extraction()$ attempting to obtain iterative solutions as many as possible, which may lead SLIM to stop at the maximum iteration. However, running time is significant to benchmark objective function effectiveness because it shows how quick the alternative objective function designs guide RS-Opt to find the benchmark solution s^*. The average running times of the three datasets with respect to different objective

function designs are given in Figure 5.7. The results show that F_3 can quickly guide RS-Opt to find the '(near-) optimal' solution, but F_2 takes almost 4 times of F_3 running time to converge.

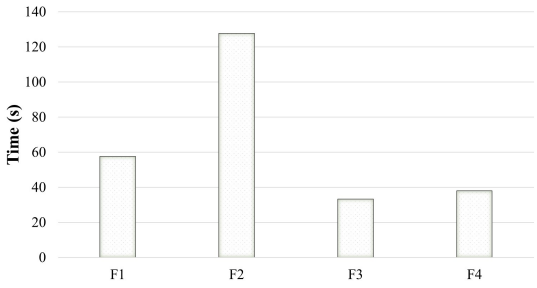


Figure 5.7: RS-Opt average running time

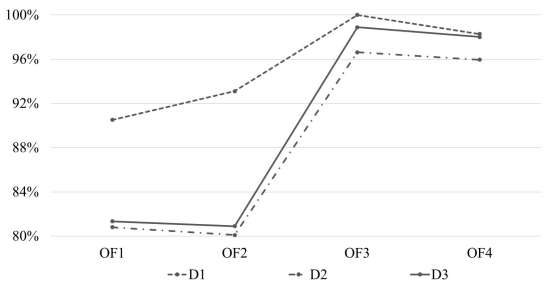


Figure 5.8: Structural similarities

(1) Quantified values of main optimization criteria Four main optimization criteria are considered: fleet size, arc usage, mileage, and compact diagram. The compact diagram is modeled by the time length (slack time) of arcs. At the convergence of RS-Opt and SLIM according to each objective function candidate, the quantified values of all the main optimization criteria can be derived from the final solution. All alternative objective function designs can always lead RS-Opt and SLIM to converge to the solutions which have the same values for the first three main optimization criteria, as shown in Table 5.4. The first and the third

Table 5.4: Quantified values of the first three optimization criteria

Convergence	Fleet size	Mileage (mile)	Arc usage
D1	7	6902.6	116
D2	17	13382.1	190
D3	20	13691.2	230

optimization criteria are always modeled in all alternative objective function designs, thus, it is reasonable to have no difference because they are supposed to be minimized. We notice that there is no difference in the second optimization criterion as well no matter whether it is considered into the objective function. One reason may be that this optimization criterion is positively supported by the optimization of some other criteria.

Table 5.5 shows the quantified values of slack time for three datasets. 'B/A/W' represents the 'best/average/worst' slack time value carried by the final solutions obtained from multiple runs of SLIM. The 'Max gap' row shows the maximum difference between the maximum and minimum values from RS-Opt and SLIM based on the four alternative objective function designs. The results show that the last optimization criterion, modeled as slack time, has a significant differ-

Table 5.5: Slack time (mins)

Dataset		D1		D2		D3	
Solver		RS-Opt	SLIM (B/A/W)	RS-Opt	SLIM (B/A/W)	RS-Opt	SLIM (B/A/W)
OFs	F_1	2034	2000	8782	7693/8346/8946	9899	9507/9993/10930†
	F_2	2160†	2000	9197	7913/8384/9634†	9597	9872/10015/10501
	F_3	1813*	1843	7180	7046/7193/7309	8897	8905/9020/9099
	F_4	1813*	1813*/1828/1843	7257	6716*/6993/7223	9064	8789*/8993/9133
Gap		347	187/172/157	2017	1197/1391/2411	1002	1083/1022/1831
Max gap			347		2918		2141

Notes: $\star$/$\dagger$ marks the minimum/maximum slack time for each dataset; OFs: objective functions

ence amongst the solutions from alternative objective function designs, where the biggest 'max gap' for D2 is almost 3000 minutes. For each dataset, the final solutions from RS-Opt and SLIM always carry the same values for the first three optimization criteria no matter which objective function is in use such that these three solution features reflect the effectiveness of alternative objective function designs to the same degree. On the other hand, the last optimization criterion (slack time) that varies a lot throughout the solutions converged by different objective functions can act as a significant feature to indicate the objective function effectiveness.

(2) Structural similarities Figure 5.8 demonstrates the results of structural similarity comparisons between the final solutions obtained by RS-Opt and SLIM. F_3 guides SLIM to obtain the solution that has the highest similarity to the benchmark solution for all the three tested datasets. Comparing F_1 to F_3, F_2 to F_4, the different optimization criterion in the objective function is 'compact diagram'. F_3 and F_4 generally guide SLIM to obtain solutions that have much higher similarities with the corresponding benchmark than F_1 and F_2. This indicates that 'compact diagram' plays an important role in differentiating the structural

properties. Considering Figure 5.7 and Figure 5.8 together, F_3 always takes the shortest time and leads to the highest similarity.

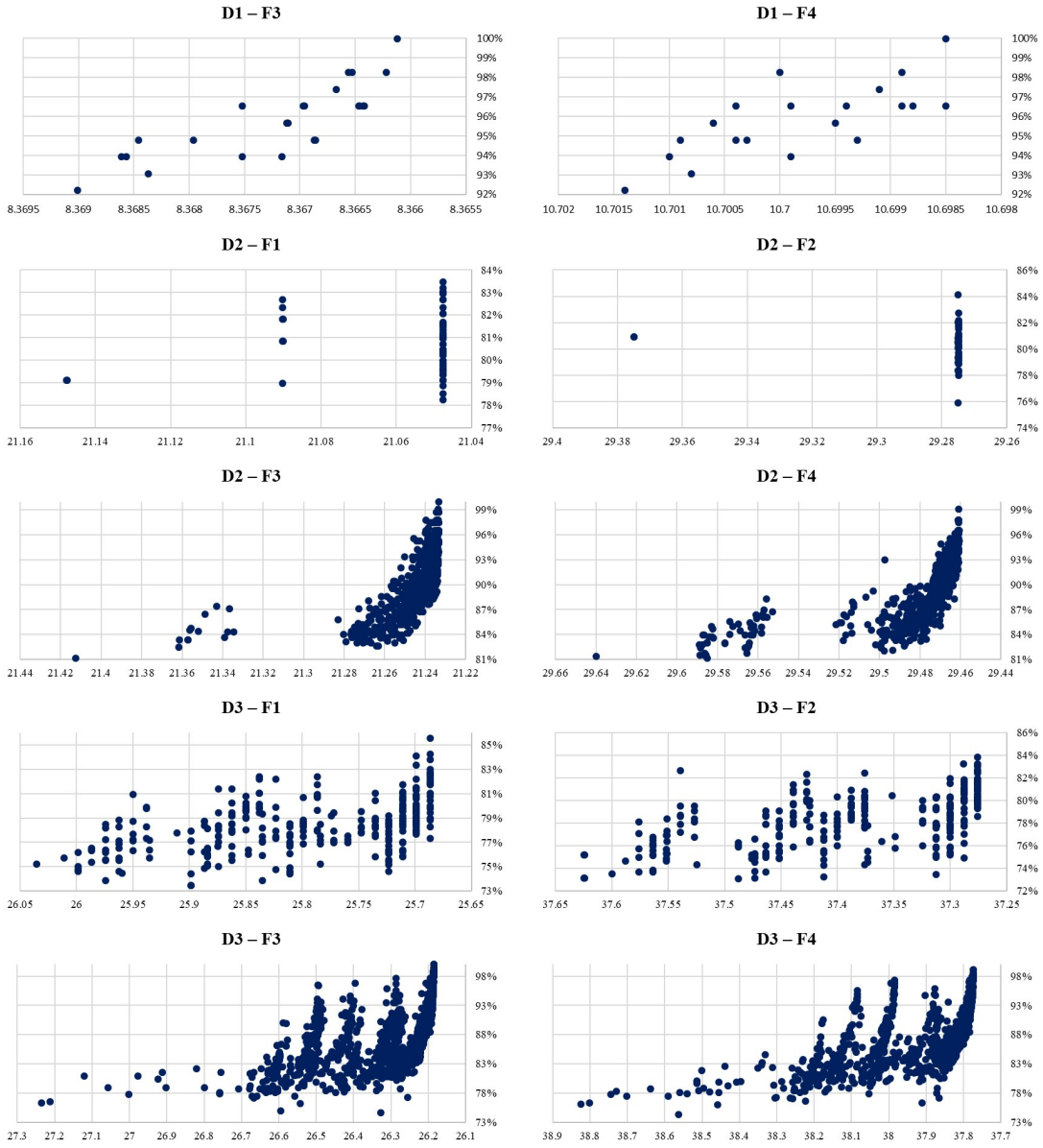


Figure 5.9: Structural similarities between the iterative solutions and the benchmark solution

Figure 5.9 shows the structural similarities between the iterative solutions and the corresponding benchmark solutions for three datasets. The figures of F_1 and

F_2 for D1 are not shown because SLIM only finds one solution whose structural similarity to the corresponding s^* can be found in Figure 5.8. Compared to F_1 and F_2, a few aspects endorsing a better objective function effectiveness are observed for F_3 and F_4: (i) many more iterative solutions are found; (ii) the solution space is ranked more precisely; (iii) the R-squared values are larger; (iv) the ranges of structural similarity are larger which means the similarities have been largely increased via the process of objective function value getting better.

(3) The number of different solutions obtained during the iterations of $\bar{P}$ Table 5.6 shows the sub-feature values of SN, including the number of solutions in terms of objective function value, the number of solutions in terms of schedule structure, and the average number of HS (seen Table 5.2). F_1 and F_2 can only find very few solutions in terms of the objective function value through many runs, and even only one solution found for D1. However, for each objective function value, there are many more solutions in different structures. This means many different subspaces derived by SLIM converge to the same objective function value but in different structures, i.e., the given objective function leads SLIM to wander over a large 'poorly differentiated' solution space. Consider the results marked as bold in Table 5.6 as an example. F_2 found 4 distinct objec-

Table 5.6: Number of iterative solutions

Dataset	D1			D2			D3		
Features No.	9	10	11	9	10	11	9	10	11
F_1	1	1	–	6	58	9.667	40	272	6.8
F_2	1	1	–	**4**	**44**	**11**	41	269	6.561
F_3	26	27	1.039	279	973	3.488	895	1190	1.298
F_4	17	21	1.235	436	918	2.106	814	1069	1.245

tive function values, however, these 4 objective function values corresponds to 44 different schedule structures [1]. This phenomenon confuses SLIM to distinguish better structural properties and converge to a better solution. On average, each

[1]Note: in Figure 5.9 of 'D2-F2', it seems that only 'two' objective function values are found. This is because some values are very close. The 'four' values are *29.2749, 29.2750, 29.3749, 29.3750* respectively. The same reason applies to other countable values in Figure 5.9 which looks like not consistent with Table 5.6.

objective function value has 11 *HS*. If the entire solution space is searched, this number would be even larger which means F_2 does not rank the solutions well. F_1 and F_2 can only lead SLIM to find a single solution whose structure is very different from but objective function value is identical to the benchmark solution. This is considered as a bad indicator for objective function effectiveness, thus, the values of the eleventh feature for F_1 and F_2 are empty for D1. On the other hand, F_3 and F_4 can find a lot more different solutions in terms of objective function value and schedule structure. Besides, the average number of *HS* for each objective function value is also much smaller. For D1 solutions found by F_3, only one objective function value has a *HS*. The others are all unique (one objective function value corresponds to one detailed schedule), which is desirable. According to the structural comparisons, the effectiveness of F_3 and F_4 is much superior to F_1 and F_2.

(4) Objective function values of the final solutions Table 5.7 gives the results of the objective function values of final solutions from SLIM compared to that from RS-Opt for each dataset over four alternative objective function designs. The comparison results of F_1 and F_2 have three cases: better, same, and worse. However, F_3 and F_4 can always lead SLIM to find solutions whose objective

Table 5.7: Convergence comparison of three datasets

Convergence	F_1	F_2	F_3	F_4
D1	Same	Same	Better	Better
D2	Better	Better	Better	Better
D3	Worse	Worse	Better	Better

function values are better than their corresponding benchmark solutions. For F_3 and F_4, the solutions with the same objective function value of s^* are also found by SLIM and their schedule structures are the same to the corresponding benchmark structures. The feature values for the comparison of objective function values are converted as a matrix with binary values to calculate the integrated objective function effectiveness, seen in the last three columns of Table 5.8.

Integrated effectiveness

Through the analysis of solution features, generally F_3 and F_4 have better effectiveness than F_1 and F_2. This section evaluates the integrated effectiveness for alternative objective function designs by employing the method proposed in section 5.2.4. To compute the integrated effectiveness according to expression (5.2), the weights and normalized values of all the features are needed, in which weights are demonstrated in Table 5.2. To obtain comparable v_i, the quantified value of

Table 5.8: Normalized values for features

Data	F	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
D1	F_1	1	1	1	.942	.905	0	0	0	.039	.037	.8	0	1	0
	F_2	1	1	1	1	.931	0	0	0	.0385	.037	.8	0	1	0
	F_3	1	1	1	.84	1	1	1	1	1	1	.841	1	0	0
	F_4	1	1	1	.84	1	1	.802	.868	.654	.778	1	1	0	0
D2	F_1	1	1	1	.955	.808	.277	0	0	.0138	.06	.879	1	0	0
	F_2	1	1	1	1	.801	.435	0	0	.009	.045	1	1	0	0
	F_3	1	1	1	.781	1	1	1	1	.64	1	0.317	1	0	0
	F_4	1	1	1	.789	.99	.949	.115	.0236	1	.943	.191	1	0	0
D3	F_1	1	1	1	.905	.847	.478	.492	.483	.0447	.229	1	0	0	1
	F_2	1	1	1	1	.831	.426	.711	.622	.046	.226	.965	0	0	1
	F_3	1	1	1	.848	1	1	.867	1	1	1	.191	1	0	0
	F_4	1	1	1	.863	.991	.98	1	.928	.909	.898	.183	1	0	0

each feature is normalized by the largest value across the four candidate objective functions, shown in Table 5.8. The first three columns are the normalized values of fleet size, mileage, and arc usage. Referring back to Table 5.4, their quantified values across four candidate objective functions are the same. Thus, they equally reflect the effectiveness of each candidate, i.e. the effectiveness is mainly determined by the other features. The last three columns are the matrix converted from the comparisons of the objective function values of the final solutions from SLIM and RS-Opt, in which each element is binary. For instance, the matrix for D1 means: F_1 and F_2 guide SLIM to converge at a solution with the same objective function value of benchmark solution; F_3 and F_4 converge to a solution with a better objective function values than the benchmark solution.

Table 5.9 demonstrates the results of integrated effectiveness. It shows that the effectiveness of F_3 and F_4 is remarkably better than F_1 and F_2, which is consistent with the analysis of the quantified values of solution features. The

effectiveness of F_3 is slightly better than F_4, and similar phenomena is observed between F_1 and F_2. This indicates that 'mileage' does not significantly contribute to the objective function effectiveness and may have some negative influence on the effectiveness. On the other hand, the effectiveness of F_3 is much better than

Table 5.9: Integrated effectiveness

	F_1	F_2	F_3	F_4
D1	0.4717	0.4751	**0.2734**	0.3081
D2	0.4647	0.4717	**0.2268**	0.2660
D3	0.4344	0.4399	**0.2217**	0.2288
Rank	$F_3 > F_4 > F_1 > F_2$			

F_1. Similarly, F_4 performs much better than F_2. This implies that 'compact diagram' largely promote SLIM to differentiate better structural properties to the solutions and further boosts objective function effectiveness. Moreover, these three datasets have the same effectiveness rank for four alternative objective function designs: $F_3 > F_4 > F_1 > F_2$, which is also consistent with the effectiveness ranked by the average running times of RS-Opt shown in Figure 5.7. According to the feedback from the practitioners, F_3 is the most effective objective function using only three main optimization criteria to cover other 'hidden' criteria considered by practitioners. And the solutions found by F_3 and F_4 are significantly better than that found by F_1 and F_2 in practice. This feedback endorses that the proposed methodology can effectively and systematically establish confidence in alternative objective function designs. Through this investigation and discussion with the practitioners, it is concluded that considering fleet size, arc usage, and compact diagram in the objective function is the most effective combination of main optimization criteria. Through the comparison between the alternative objective function designs containing different optimization criteria, we identify mileage slightly reduces objective function effectiveness but compact diagram largely increases objective function effectiveness.

Additional experiments on the number of coupling/decoupling operations and carriages

This section presents some additional experimental results on the number of coupling/decoupling operations and carriages concerning different terms in the objective function. These experiments are carried out by RS-Opt solely without considering fixed-charge variables. In the model with fixed-charge variables, the number of coupling/decoupling operations is considered as a term in the objective function. As the fixed-charge variables largely increase the problem complexity, we try to avoid to use the model with fixed-charge variables. Theoretically, considering the slack time cost can positively reduce the total number of unnecessary coupling/decoupling operations without using fixed-charge variables, as the analysis in section 4.1.3.

Table 5.10: Number of coupling and decoupling operations of D2 and D3

	F_1	F_2	F_3	F_4
D2	28	37	19	21
D3	45	38	31	35

Table 5.10 gives the results of the total number of coupling/decoupling operations under different objective functions. D1 is not included in this table because its solution does not contain any coupling/decoupling operation. For D2 and D3, the total number of coupling/decoupling operations in the solutions obtained by F_3 and F_4 is much fewer than the other two objective functions. Besides, the solution delivered by F_3 has the lowest number of coupling/decoupling operations. These results support the theoretical analysis in section 4.1.3 and comply with the evaluation results of the objective function effectiveness.

Figure 5.10 to Figure 5.13 give part of the solution connections at station Ipswich (visualized by Tracs-RS [4]) that are obtained through different objective functions for D1. Comparing F_1 to F_3, F_2 to F_4, slack time is added. These station view comparisons enable us to intuitively investigate that considering the slack time in the objective function can effectively decrease the total number of coupling/decoupling operations. Let us consider the connections marked in the red square in Figure 5.10 and Figure 5.11 as an example. We can notice that

Figure 5.10 contains many unnecessary coupling/decoupling operations. On the other hand, the connections for those trips in Figure 5.11 are very neat in which no unnecessary coupling/decoupling operations are contained. Similar results are also observed in the other comparisons in these figures. Besides, the first-in-first-out connections in the solution obtained by the objective functions considering slack time are obviously encouraged.

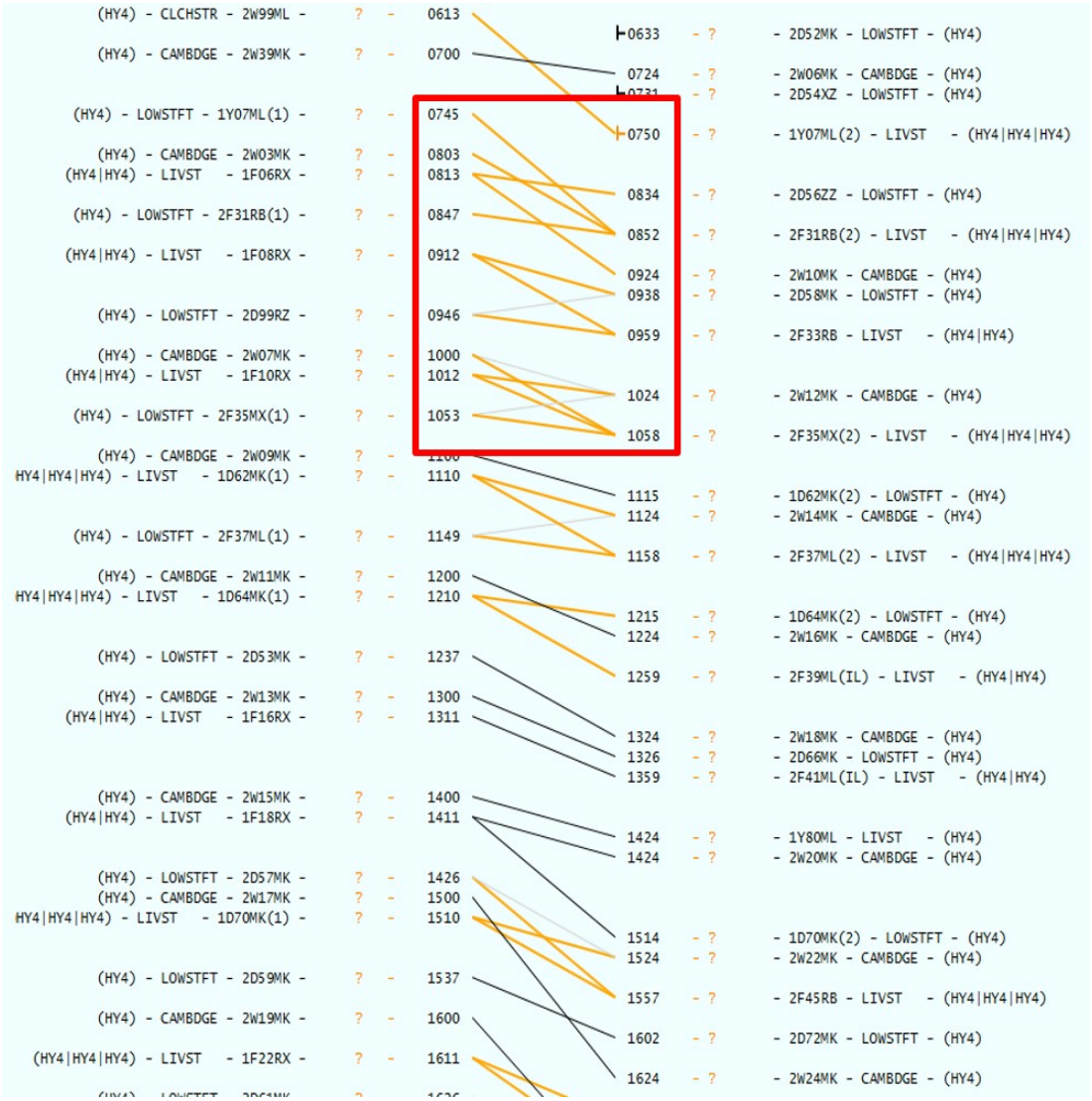


Figure 5.10: D2 Ipswich station view obtained by F_1

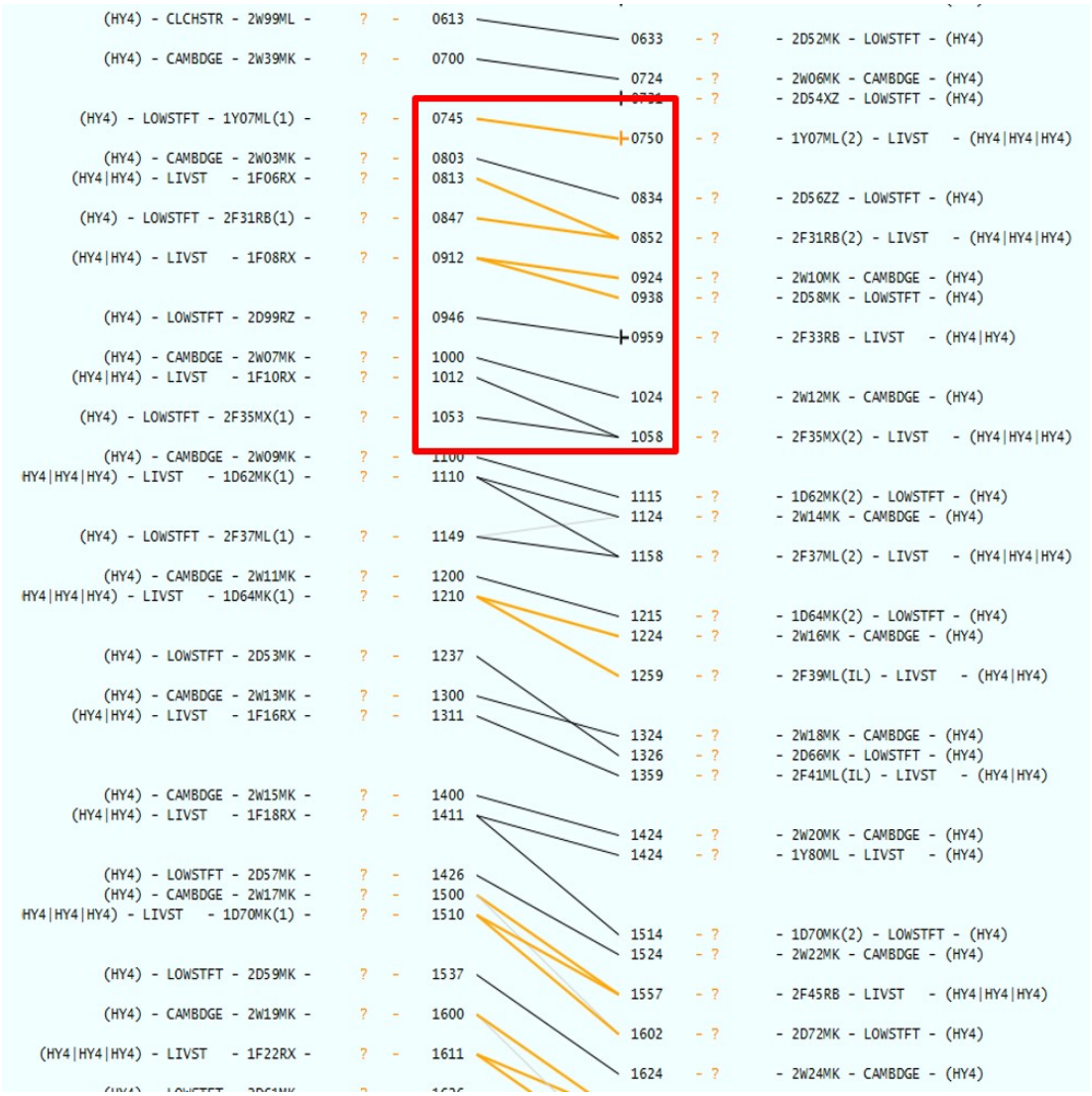


Figure 5.11: D2 Ipswich station view obtained by F_3

As D1, D2, and D3 has only a single train unit type, D4 and D5 are used to carry out the experiments on carriages. D4 has 243 trips and two train unit types; D5 contains 579 trips and seven types of compatible train units. Table 5.11 gives the computational time of RS-Opt for these two datasets under the alternative objective functions. The objective function effectiveness reflected by running time comply with the experiments on D1, D2, and D3. The running times under F_3 and F_4 are significantly shorter than that under F_1 and F_2. Same

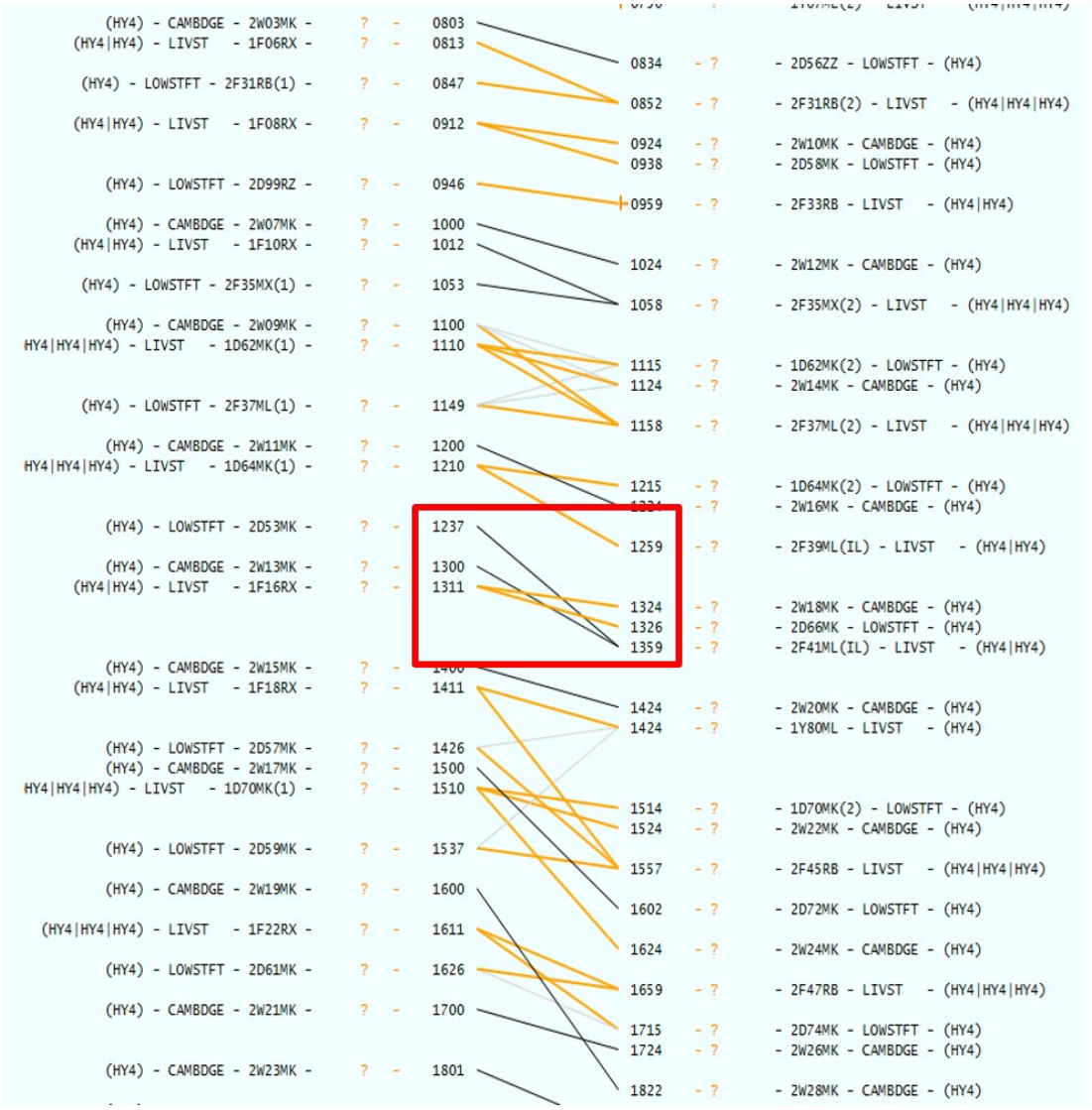


Figure 5.12: D2 Ipswich station view obtained by F_2

with the total number of coupling and decoupling. Besides, the effectiveness of F_3 is superior to the others. Thus, we carry out the experiments on the carriage minimization (Term1b) based on F_3 and F_4, and the results are shown in Table 5.12.

Notice that the total number of train units are always the same no matter if Term1b is considered. However, the total number of carriages can be further

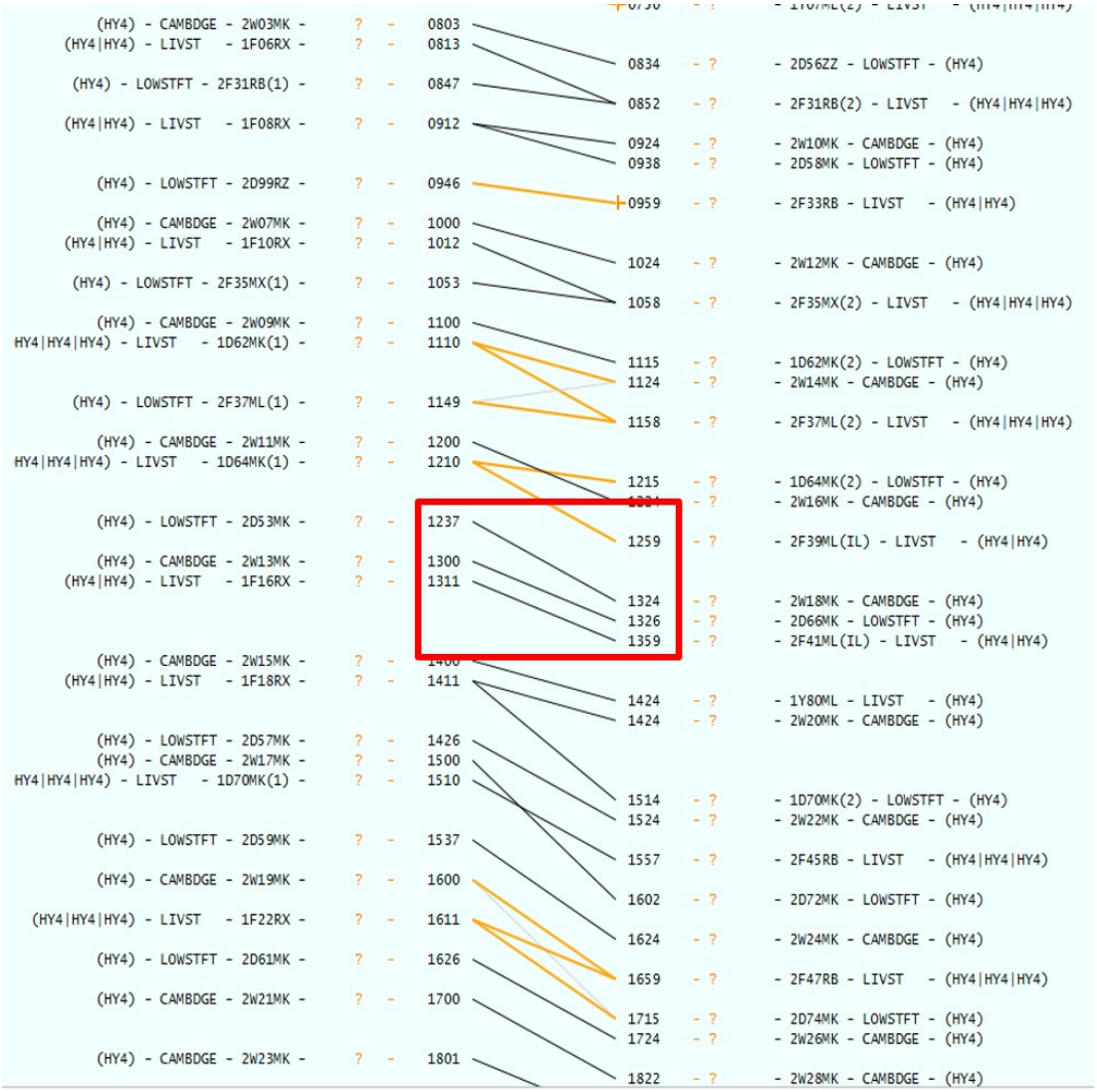


Figure 5.13: D2 Ipswich station view obtained by F_4

Table 5.11: Running times and coupling/decoupling operations of D4 and D5

		F_1	F_2	F_3	F_4
D4	Running time (s)	1099	1166	663	795
	C/D	34	40	27	29
D5	Running time (s)	699	952	143	165
	C/D	45	38	31	35

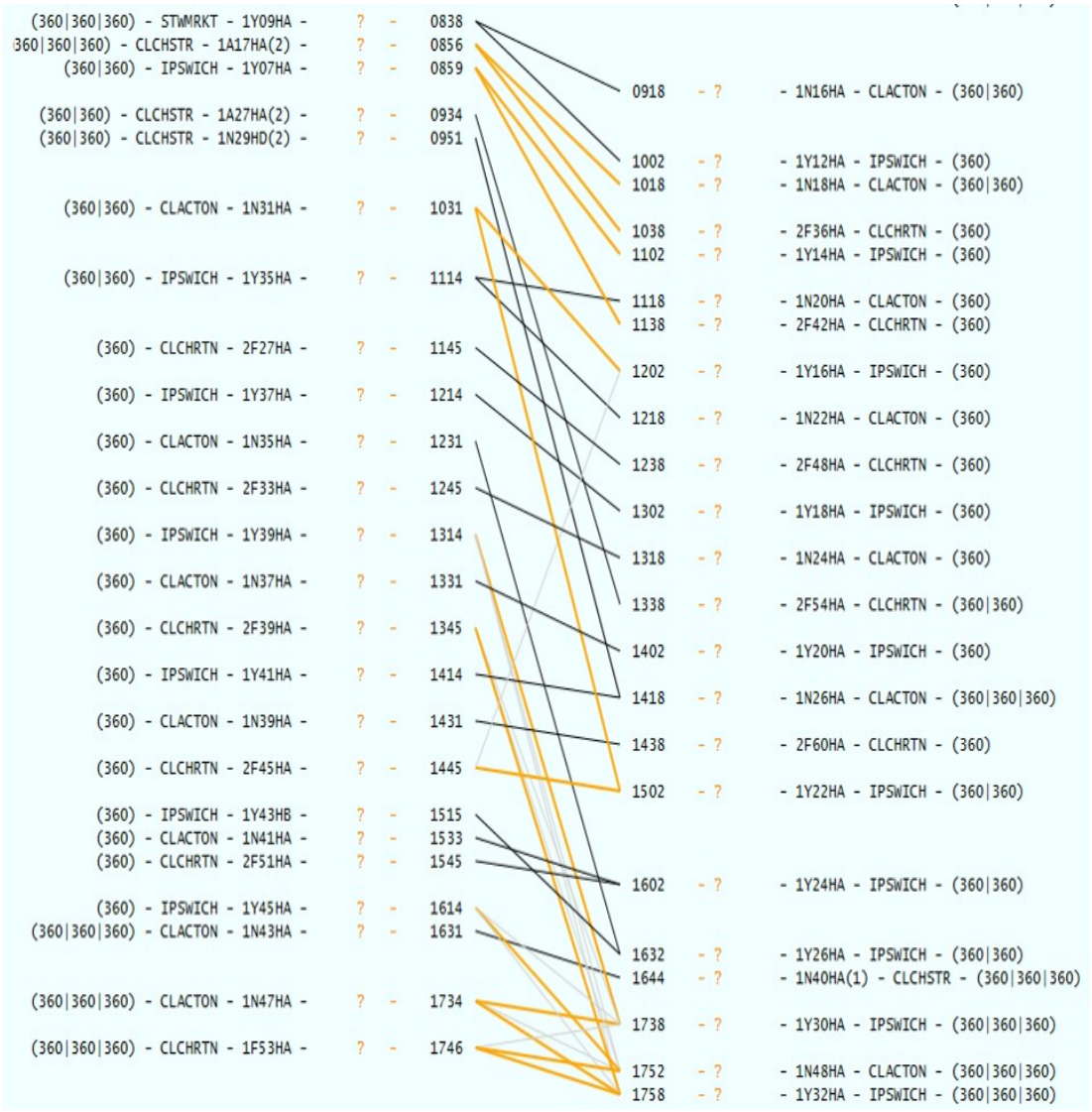


Figure 5.14: D3 Liverpool street station view obtained by F_1

optimized when Term1b is considered. For instance, the solutions obtained by F_3 and F_4 use 93 carriages, which are 3 of class 755/3 and 21 of class 755/4; on the other hand, the solutions considering Term1b contain only 89 carriages in which 7 of class 755/3 and 17 of class 755/4 are in use. It is noticeable that different train unit types with distinct capacities (number of carriages) in the solution is more balanced when Term1b is considered. The carriage optimization will not affect the

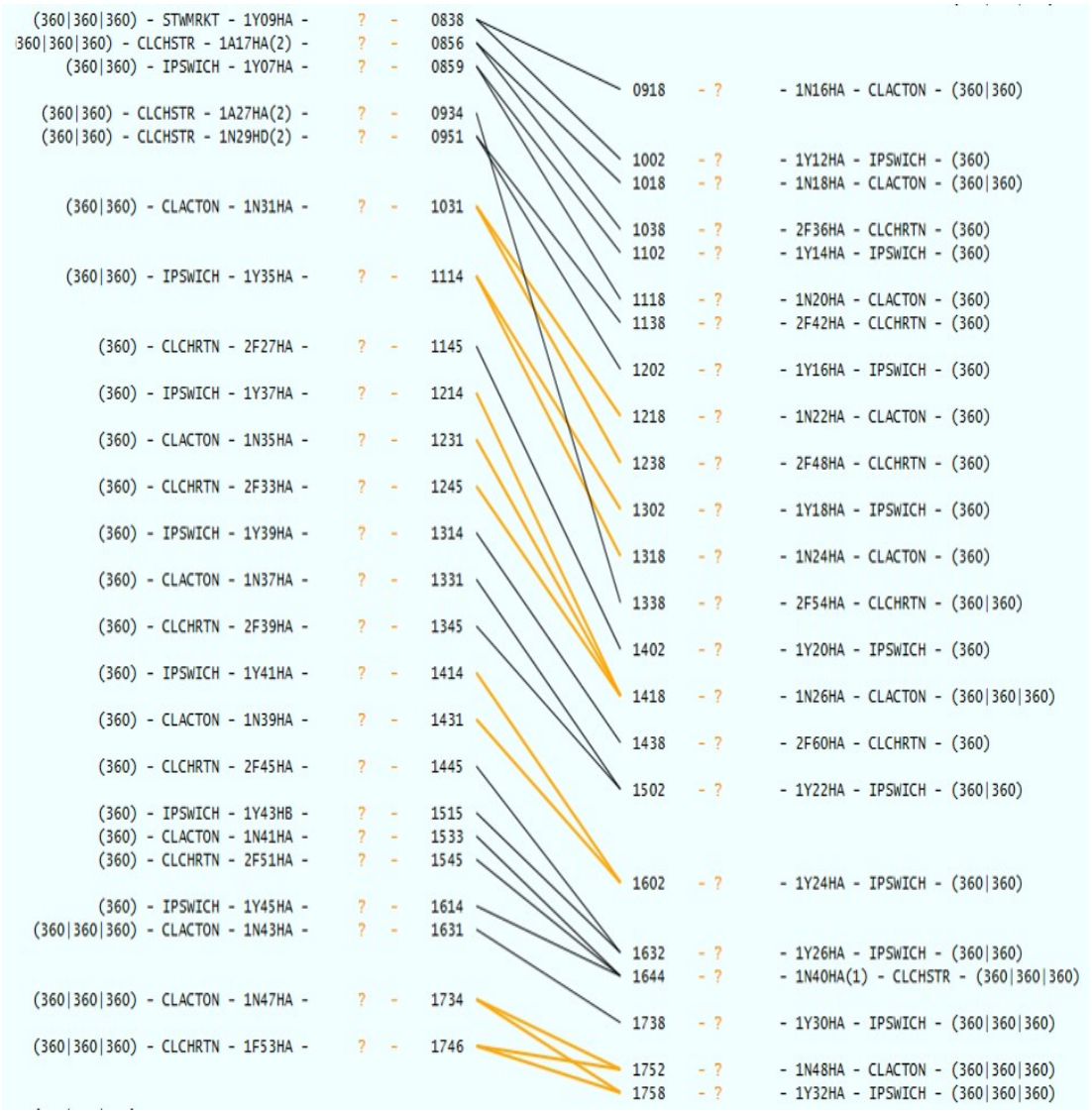


Figure 5.15: D3 Liverpool street station view obtained by F_3

optimization of train unit numbers because of the well-measured weights, details seen in section 4.2.2. The carriage minimization can save operational costs and carriage mileages because running a longer train unit costs more.

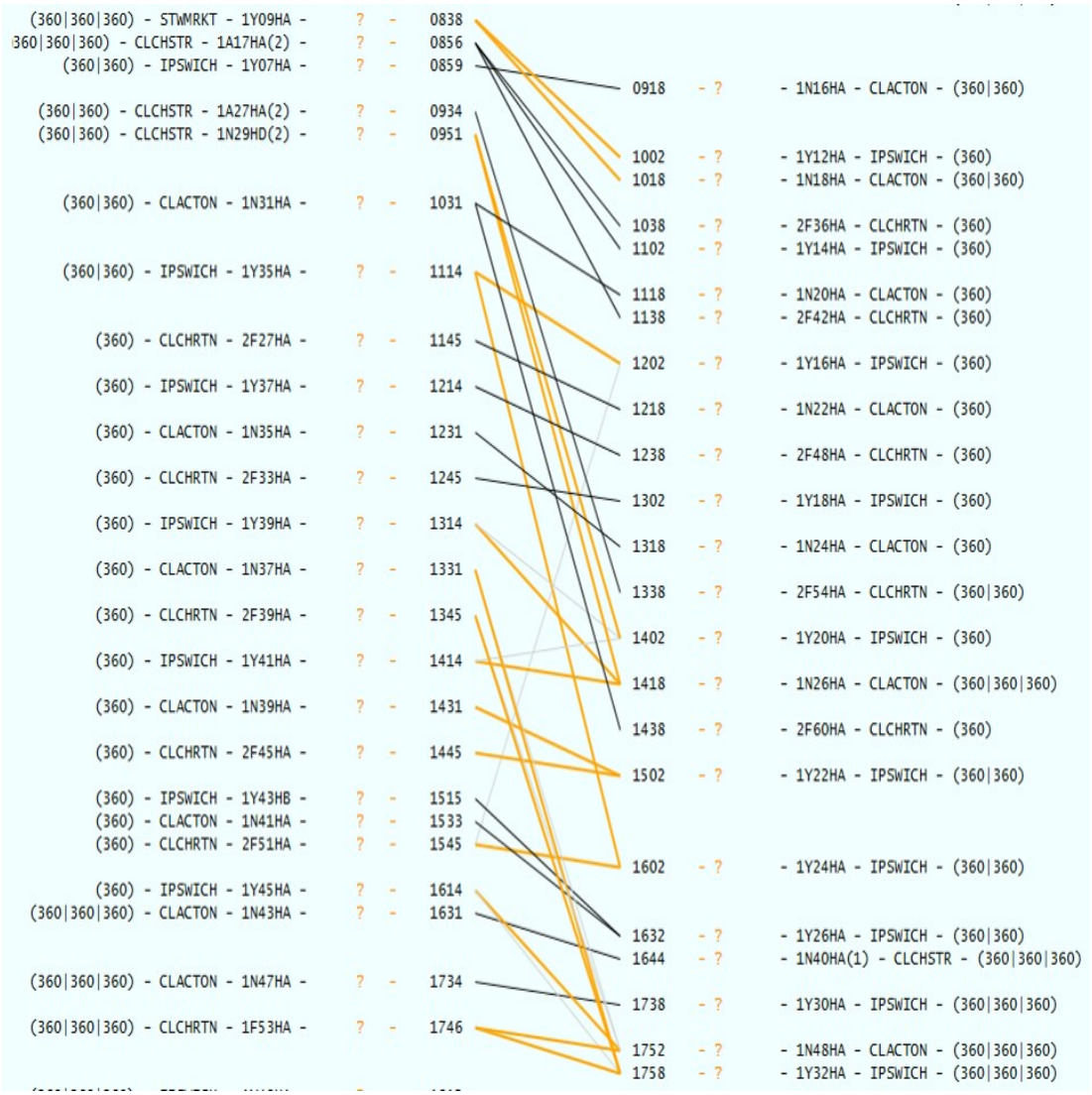


Figure 5.16: D3 Liverpool street station view obtained by F_2

Table 5.12: Number of units and carriages of D4 and D5

		F_3	F_3 + Term1b	F_4	F_4 + Term1b
D4	Unit	24	24	24	24
	Carriage	93	89	93	89
D5	Unit	70	70	70	70
	Carriage	139	138	139	138

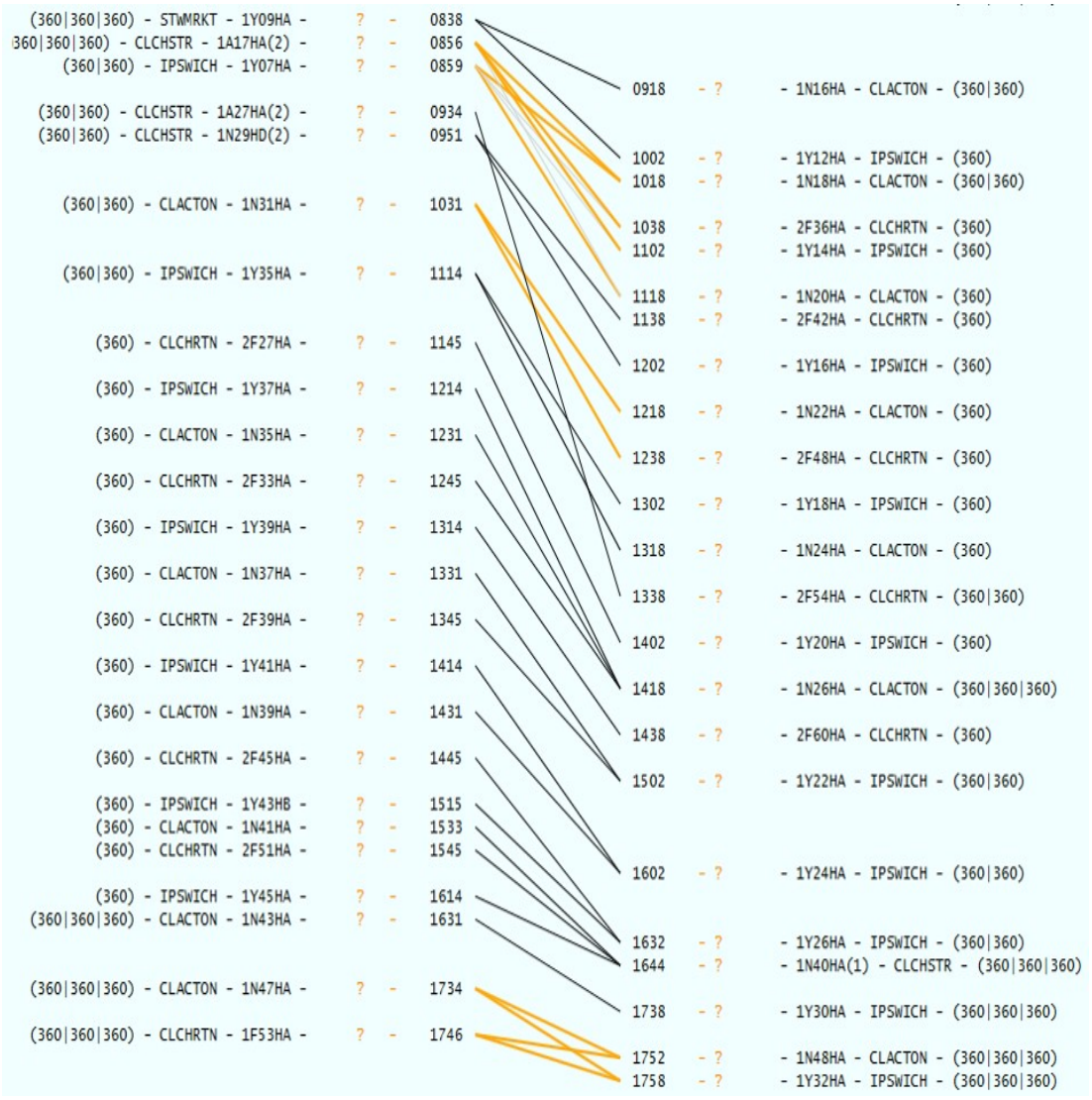


Figure 5.17: D3 Liverpool street station view obtained by F_4

5.4 Discussions

For a complex real-world scheduling problem that is virtually NP-hard and has numerous possible structural properties, it is not easy to establish confidence in the effectiveness of objective function designs. The study on this topic is very scarce in the literature. There is a lot of research studying real-world multi-criteria optimization and automatic methods for algorithm parameter control,

seen in the literature review in sections 2.5 and 2.6. However, they mostly pay attention to promote the performance of algorithms to deliver better quality solutions or shorten the computational time. In this research, a methodology evaluating objective function effectiveness through the comparisons between the solutions obtained from an exact ILP method and an auxiliary heuristic method is presented. A set of features reflecting objective function effectiveness are derived, where the main measure of effectiveness is in the structural comparisons between the heuristic solutions and the exact solution. A hierarchical scheme of integrating all features is devised to quantify the objective function effectiveness. The experiments carried out with TUSO instances strongly support the effectiveness of this methodology. Besides, some additional experimental results on improving the optimization of the total number of coupling/decoupling events and carriages through objective functions are also presented. The TUSO problem researched at the University of Leeds [57, 58, 26, 49] has inspired this work. In ongoing work, we are investigating methods to improve the auxiliary heuristic method to perform better in deriving reduced inputs. Another direction is how to evaluate the objective function effectiveness if no practical exact method solver is available but only a heuristic method is available. We may consider a dynamic benchmark that can be updated once a better solution is found until no improvement can be achieved.

Chapter 6

Station level resolution

Train unit scheduling at the network level focuses on assigning the vehicle flows to cover a timetable satisfying seat demands, which left coupling orders undetermined and tentatively assigned linkages unfinalized. The research in this chapter aims at expanding the network level (Phase I) to the station level (Phase II) to resolve these two aspects to obtain a conflict-free solution at the specific station layouts. While non-operable linkages may not be difficult to recognise by human schedulers, precise algorithms are needed for the computer to detect conflicts through the entire network based on complex station-level constraints. The coupling order, coupling order propagation, and also coupling/decoupling operations need to be precisely formalized to decompose the complex graph into some fundamental parts which are easier for applying the logic of detecting conflict and finalize the tentative linkages systematically. This chapter includes the following parts: coupling order definition, function and operator definitions, coupling order propagation boundaries, an adaptive approach for the station-level resolution, and experimental results.

6.1 Coupling order definition

The solution from Phase I is a set of paths each corresponding to a certain train unit sequentially serving a set of trips. This solution gives a collection of units for each trip and connections among trips. There is no defined unit sequence

in a coupled unit block because of the simplification of station level constraints. Hence, a multi-set can be defined to represent the collection of units for each trip, denoted by U_j. The elements in U_j are the units of same or different types serving trip j. The number of elements of the same unit type in U_j is the type flow of trip j. For instance $U_j = \{X, X, Y, Y\}$ means trip j is served by 4 units composed of 2 units of type X and 2 units of type Y. In a network flow solution, two attributes for a trip j are important to introduce the concept of coupling order, which are the predecessor and successor node sets (I_j and R_j). The predecessor/successor node set of trip j contains all the nodes that have an arc to/from trip j. Consider Figure 6.2 as an example, the predecessor node set of trip j is $\{i_1, i_2, ..., i_m\}$ and the successor node set of trip j is $\{r_1, r_2, ..., r_m\}$. In addition, the trip timing and direction information related to a given timetable and corresponding station structure are also important in later discussion. Their notations are shown in table 6.1. Here, $dp(j)$ and $ap(j)$ are binary values since each platform has two

Table 6.1: Notations of some trip attributes

Notations ($j \in N$)	Definitions
I_j	predecessor node set for trip j in a given solution
R_j	successor node set for trip j in a given solution
$dp(j)$	departing direction of trip j
$dt(j)$	departing time of trip j at its departure platform
$ap(j)$	arriving direction of trip j
$at(j)$	arriving time of trip j at its arrival platform

notional directions of approach/travel. Let us define them as $\{up, down\}$, which will be used in the rest of this part. One value has to be crossed out for the dead-end platforms since they only have one end accessible. Thus the general unit-block collection for each trip can be described as expression (6.1). Normally, the unit coupled together can be up to 4 in real-life, i.e. $m \leq 4$, $|I_j| \leq 4$, and $|R_j| \leq 4$.

$$U_j = \{\ u_1\ u_2\ ...\ u_m\ \}, \forall j \in N \tag{6.1}$$

For trips served by a single unit ($m = 1$), there is no need to consider their coupling order. Besides, the unit sequence of unit blocks of same type is also not

critical. This is based on the assumption that the diagram assignment among same-type units can be easily swapped since unit circulation and maintenance are not considered in this research.

Day-time coupling and decoupling operations are mostly done at platforms in the UK. Those activities are critical because they form/request a certain unit sequence. This feature together with trip attributes in table 6.1 can be utilized to tentatively assign locally feasible coupling orders to some trips. Let S_j denote the set of stopping stations of trip j. To formalize the coupling order concept, let us define a sequenced multi-set O_j^s to represent the coupling order for trip j at location $s, s \in S_j, j \in N$. Start of a sequence is the front of a certain moving direction followed by sequenced unit blocks as the rear. During the coupling order assignment process, it has three status: fixed, unfixed, and semi-fixed, which can be described in a general expression (6.2). The elements in O_j^s are sequenced subsets of U_j. The union of all the elements in O_j^s must be the same with U_j, because O_j^s and U_j express the unit formation of trip j, as shown in expression (6.3). Generally, $O_j^s = U_j$ if O_j^s is "unfixed".

$$O_j^s = [\ v\ | v \subseteq U_j\ ] = [\ v_1\ v_2\ ...\ v_{m'}\ ],\ \forall s \in S_j,\ \forall j \in N \tag{6.2}$$

$$\bigcup_{i=1}^{m'} v_i = U_j,\ \forall j \in N,\ m' \leq m \tag{6.3}$$

In the context of arrival and departure, a unit block has a front and a rear with respect to its travel direction. As shown in Figure 6.1, the coupling of two unit blocks can either be a front-front (Figure 6.1a) or front-rear (Figure 6.1b and c) attachment. Rear-rear (Figure 6.1d*) attachment is physically impossible unless one of the unit blocks reverses into the platform, in which case the reversed approaching end is regarded as the front. Similarly, after the decoupling of a unit block, the two resulting unit blocks may next travel in the same direction with front-rear facing or in the opposite directions with rear-rear facing (front-front facing is physically impossible).

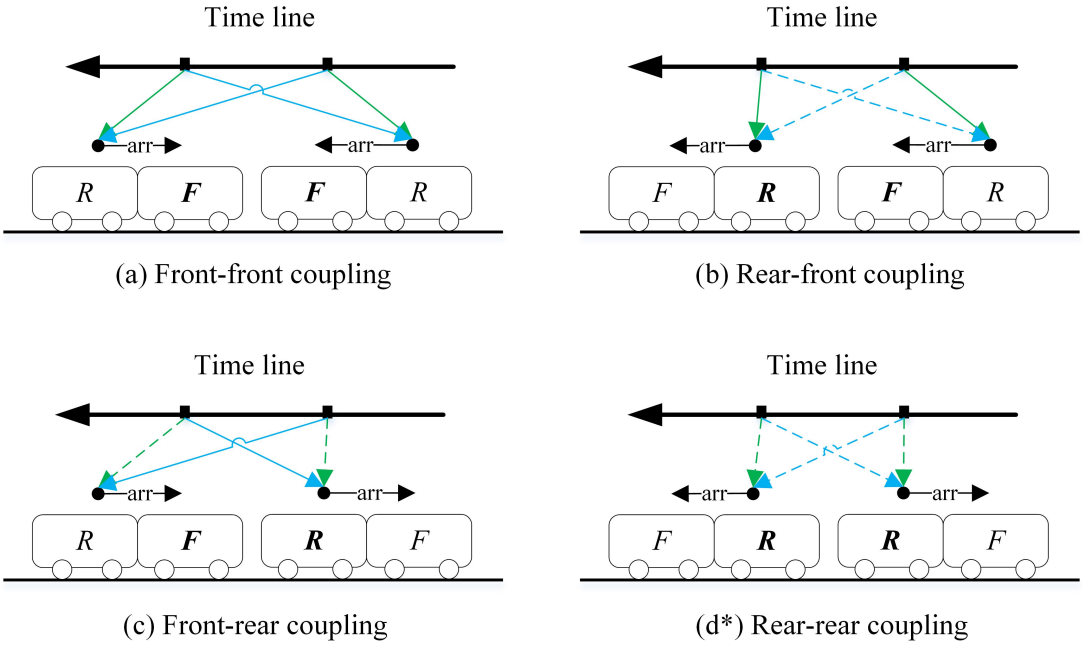


Figure 6.1: Coupling position

6.2 Function and operator definitions

In real life, the moving direction of a train journey may be reversed at some en-route stations. With a known unit sequence in a trip at a certain station, the unit sequences at other stations are all known by counting the number of en-route reversal operations. For a given network flow schedule, it is worthy to tentatively seek if there is a feasible coupling order assignment existing. Coupling and decoupling operations can locally fix some feasible coupling order. Running trips can spread the influence of coupling order out to the whole network. Accordingly, a front-and-rear reversal function is defined to express the en-route reversal movement, which will be applied once the moving direction is reversed. In addition, coupling and decoupling operators are also defined based on the assumption that the trips involving coupling/decoupling events are operated at the same platform. Their notations are shown in table 6.2. The operations related to different platforms, for example re-platforming, will be further explained in section 6.4.1.

Table 6.2: Function and operator

Symbols	Remarks
$Rev(\delta)$	front-and-rear reversal function
" + "	coupling operator
" − "	decoupling operator

(1) Front-and-rear reversal function $Rev(\delta)$: the parameter δ is a coupling order. This function can be applied multiple times, defined as $Rev^n(\delta)$, $n \in \mathbb{N}$. While n is odd, δ will be reversed; otherwise, the same order is kept. Testing n is odd or even is a matter of computational implementation.

(2) Coupling operator: a coupling event is referring to two (sequenced-)sets joining together to form a longer (sequenced-)set. The arcs involving this operation are called coupling linkages. Let us define the coupling operator $(+)$ on two unit blocks with the resultant coupling order as shown in expression (6.4). That is, the second operand unit block w is attached to the rear of the first operand unit block v.

$$u = v + w \implies O_u = [O_v O_w] \tag{6.4}$$

Let i_1, i_2 denote train unit blocks arriving at the same platform to be attached and i_1 arrives first, $at(i_1) \leq at(i_2)$. Suppose i is the resultant unit block after the attachment. As shown in Figure 6.1, i_1 and i_2 can arrive from the same/opposite direction(s). Considering $ap(i_1)$ as the *reference direction*, the coupling order of the resultant unit block is shown in expression (6.5).

$$\begin{aligned} i &= \begin{cases} i_1 + i_2, & \text{if } ap(i_1) = ap(i_2) \\ Rev(i_2) + i_1, & \text{otherwise} \end{cases} \\ &\implies O_i = \begin{cases} [O_{i_1} O_{i_2}] \\ [Rev(O_{i_2}) O_{i_1}] \end{cases} \end{aligned} \tag{6.5}$$

Once the coupled formation O_i based on the reference direction is fixed,

the next step is to assign O_i to trip j according to the relative directions between $dp(j)$ and $ap(i_1)$, seen in expression (6.6).

$$O_j^{ori} = \begin{cases} O_i, & \text{if } dp(j) = ap(i_1) \\ Rev(O_i), & \text{otherwise} \end{cases} \tag{6.6}$$

(3) Decoupling operator: Similar to the discussion on the coupling operator in (2), let us define the decoupling operator $(-)$ on two unit blocks with the resultant coupling order of first operand as shown in expression (6.7), in which the second operand unit block w is detached from the rear of the first operand unit block v.

$$u = v - w \implies O_v = [O_u O_w] \tag{6.7}$$

Let r, r_1, r_2 denote train unit blocks such that r_1 and r_2 depart from the same platform after being detached from r. Suppose r_1 departs first ($dt(r_1) \leq dt(r_2)$), and using $dp(r_1)$ as the *reference direction*. The coupling order of unit block r can be obtained according to expression (6.8).

$$\begin{aligned} r_1 &= \begin{cases} r - r_2, & \text{if } dp(r_1) = dp(r_2) \\ r - Rev(r_2), & \text{otherwise} \end{cases} \\ \implies O_r &= \begin{cases} [O_{r_1} O_{r_2}] \\ [O_{r_1} Rev(O_{r_2})] \end{cases} \end{aligned} \tag{6.8}$$

With the arrival direction of r, the requested coupling order at the destination of trip j can be assigned as expression (6.9).

$$O_j^{dest} = \begin{cases} O_r, & \text{if } ap(j) = dp(r_1) \\ Rev(O_r), & \text{otherwise} \end{cases} \tag{6.9}$$

For the dead-end platform, the unit blocks can only arrive from the same direction and the departure direction must be opposite to the arrival direction.

When there is no coupling ($|I_j| = 1$) or decoupling ($|R_j| = 1$) related to form-

ing/decomposing some new unit blocks, coupling order is temporarily considered as not constrained. The coupling-order significance of unfixed trips will be restored at the stage of coupling order propagation through the whole network. Another case shown in Figure 6.2 refers that trips involve more than one coupling/decoupling operations, i.e., $|I_j| \geq 3$ or $|R_j| \geq 3$. This case is considered in an iterative way as shown in algorithm 2. Only one coupling operation is applied at each iteration. The temporary coupling order during the iteration process is stored in O_{temp} till all the unit block of trips in I_j is traversed. Thus the final O_i can be obtained to be assigned to O_j^{ori}. Similarly, this method can be applied to multi-decoupling operations as well. Let use D to represent the size of $|I'_j|$. The complexity of algorithm 2 is $O(D)$.

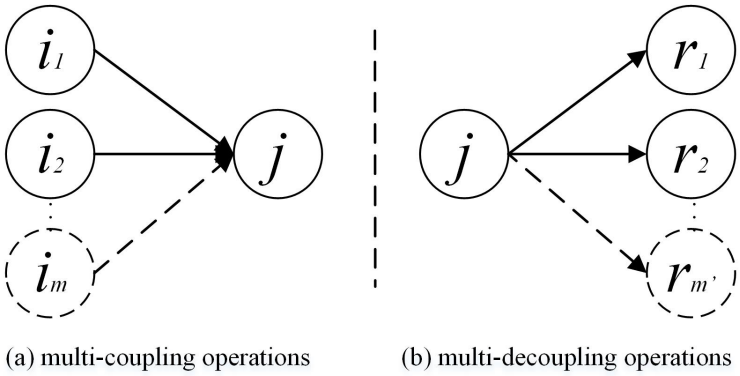


Figure 6.2: Multiple (de-)coupling operations

6.3 Coupling order propagation boundaries

In this section, we shall systematically extract concise parts of the DAG where coupling order conflicts could arise in a given Phase I solution because the coupling order has not been maintained along with network flow level. With respect to the full DAG $\mathcal{G}$, a Phase I solution is a sub-graph $\mathcal{G}^* \subset \mathcal{G}$ with train unit flows assigned. Disregarding the source and sink and arc directions, $\mathcal{G}^*$ is decomposed into one or more disjoint connected graphs, which can be classified into two types as follows, represented as sets $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively.

Algorithm 2 Multi-coupling operations

Require: I_j
Ensure: O_j^{ori}
1: $I_j^{'} := sorted(I_j)$; $i_1 := I_j^{'}.firstTrip$; $O_{temp} := O_{i_1}^{dest}$
2: **for all** idx **in** $I_j^{'} \setminus i_1$ **do**
3: $O_{temp} \leftarrow$ "+" for O_{temp} and $O_{i_{idx}}$
4: **end for**
5: $O_i := O_{temp}$
6: **if** $dp(j) = ap(i_1)$ **then**
7: $O_j^{ori} := O_i$
8: **else**
9: $O_j^{ori} := Rev(O_i)$
10: **end if**

- We do not consider the coupling order issue for a sub-graph if it has only a single unit type or it has multiple unit types but does not involve any coupling/decoupling operation, stored in $\mathcal{G}_1$.
- The sub-graphs who do not satisfy the conditions for $\mathcal{G}_1$ will be stored in $\mathcal{G}_2$ because their coupling order issues matter.

In a $g \in \mathcal{G}_2$, there may be some trips that are served by only one single unit type, denoted as set N_1. The coupling order issue for those trips are immaterial, thus, g can be further decomposed to smaller sub-graphs to analyze the coupling order issue by trimming off all the trips in N_1. Let use $\mathcal{G}_3$ to denote the set of sub-graphs decomposed from all the graphs in $\mathcal{G}_2$. Figure 6.3, shows an example graph $g \in \mathcal{G}_2$ in which the trips that are free of coupling order issue (single unit type) are represented as dashed circles. Two independent sub-graphs (marked out by blue boxes) can be extracted from g when those dashed nodes are trimmed off. A graph in $\mathcal{G}_3$ is the smallest unit to analyze the coupling order issue on the network. The coupling order issue of a $g_1 \in \mathcal{G}_3$ has nothing to do with that of another $g_2 \in \mathcal{G}_3$, but is sealed within g_1 such that the coupling orders for the trips contained in g_1 are affected to each other, defined as coupling order propagation. If the coupling orders of the trips in a graph g_1 in $\mathcal{G}_3$ are not compatible to each other, all the get-in and get-out arcs of the nodes in g_1 in $\mathcal{G}_3$ will be collected as an infeasible arc combination. Take the second sub-graph in Figure 6.3 as

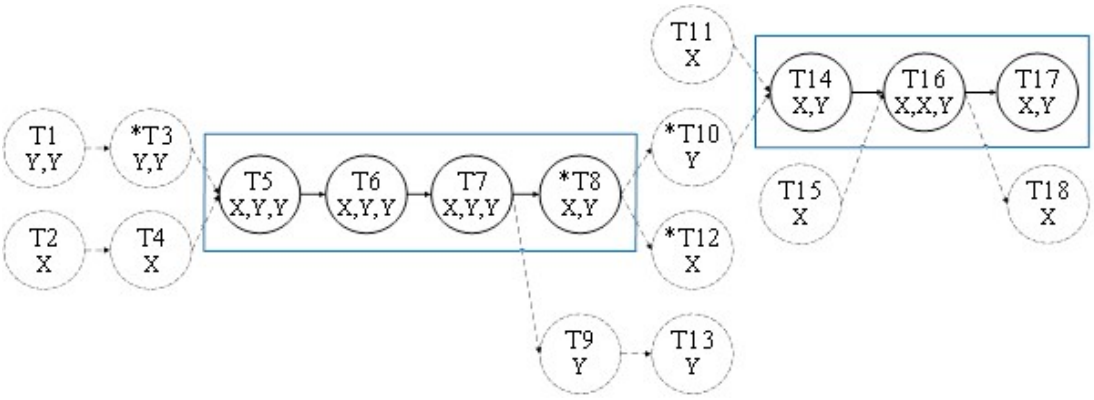


Figure 6.3: Two sub-graphs extracted from a graph in $\mathcal{G}_2$

an example, if the coupling orders of T14, T15, T17 are not compatible to each other, the arcs {T11-T14, T10-T14, T14-T16, T15-T16, T16-T17, T16-T18} will be collected as an infeasible arc combination.

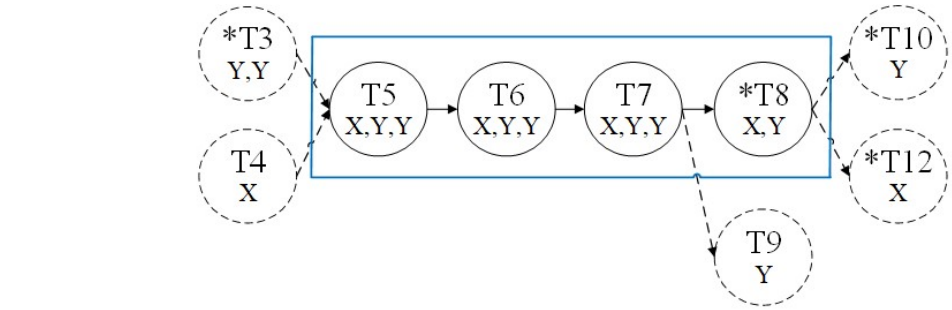
To illustrate how coupling order propagates through a graph in G_3, the first sub-graph in Figure 6.3 is taken as an example. At its fringe, all the get-in and get-out arcs and their connected nodes (T3, T4, T9, T10, and T12) are restored. Figure 6.4a shows the DAG with one coupling and two decoupling operations. One en-route reversal operation (marked as *) will happen to T3, T8, T10 and T12. Figure 6.4b is the corresponding schematic space-time diagram and their serving units assigned at the network flow level, in which (de-)coupling operations are marked as black circles. Stations A, B, C have only one platform and their platform types are also indicated. There may be other trains visiting as intermediate stops, which are not shown here. They are usually planned at the timetabling stage such that they will not be in conflict with other terminating trips.

First of all, assign local feasible coupling orders by coupling/decoupling operations introduced in section 6.2 and the results are as follows:

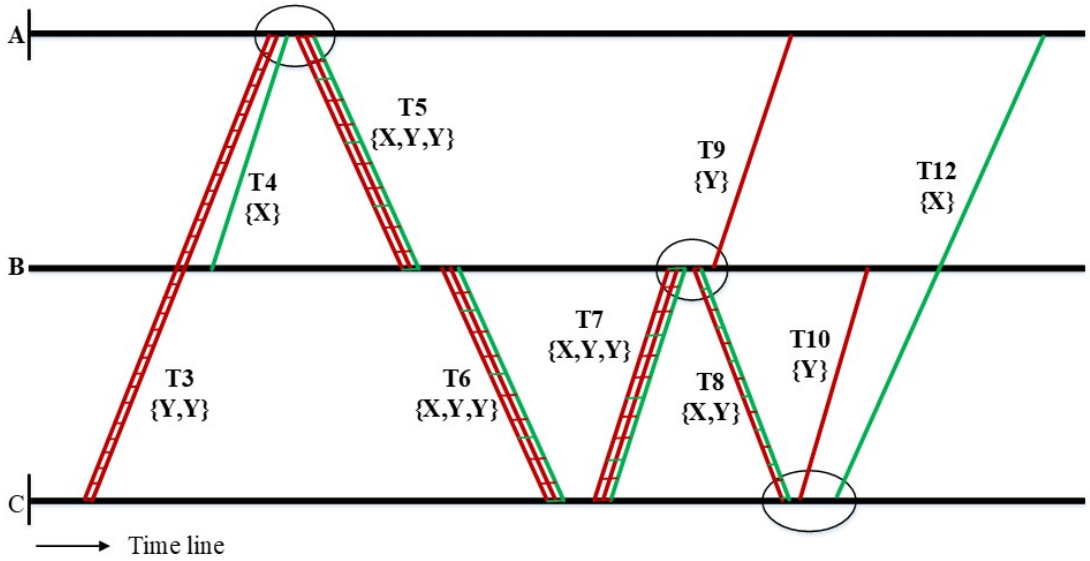
(1) Coupling operation to form T5 with regard to arcs (T3,T5) and (T4,T5) by considering $ap(T3)$ as the reference direction.

Since, $ap(T3) = ap(T4)$ and $at(T3) < at(T4)$

We have, $i = i_3 + i_4 \Longrightarrow O_i = [O_3 O_4] = [\{Y, Y\}\{X\}] = [YYX]$



(a) DAG



(b) Space-time diagram

Figure 6.4: A sub-graph extracted from Figure 6.3

Since $dp(T5) \neq ap(T3)$, $O_5^{ori} = Rev(O_i) = [XYY]$

Thus, $O_5^{dest} = O_5^{ori} = [XYY]$

Since, no en-route reversal operation for T5, $O_5^{ori} = O_5^{dest} = [XYY]$.

(2) Decoupling operation for T7 through arcs (T7,T8) and (T7,T9) to serve T8 and T9 with consideration $dp(T8)$ as the reference direction.

As $dp(T8) \neq dp(T9)$, $r_8 = r' - Rev(r_9)$

Thus, $O_{r'} = [O_8 Rev(O_9)] = [\{X, Y\}\{Y\}] = [\{X, Y\}Y]$

Since $ap(T7) = dp(T8)$, $O_7^{dest} = O_{r'} = [\{X, Y\}Y]$

Then, $O_7^{ori} = O_7^{dest} = [\{X, Y\}Y]$

(3) Decoupling operation for T8 to serve T10 and T12 over arcs (T8,T10) and (T8,T12). Let $dp(10)$ be the reference direction.

Since, $dp(T10) = dp(T12)$ and $dt(T10) < dt(T12)$

We get, $r_{10} = r - r_{12} \Longrightarrow O_r = [O_{10}O_{12}] = [\{Y\}\{X\}] = [YX]$

Because of $ap(T8) \neq dp(T10)$, $O_8^{dest} = Rev(O_r) = [XY]$

Thus, $O_8^{ori} = Rev(O_8^{dest}) = [YX]$

Through the operations above, the coupling orders of T5 and T8 are fixed and the coupling order of T7 is semi-fixed, but the coupling order of T6 is still unfixed. Three possible tracking methods are considered for the coupling order propagation: early to late tracking, late to early tracking, and tracking starting with any intermediate trip. These methods may result in conflict-free or coupling order collision at different locations on the network.

Method 1: T5 → T6 → T7 → T8.

- T5 → T6: since there is no en-route reversal happening to T6 and $ap(T5) = dp(T6)$, thus $O_6^{ori} = O_6^{dest} = O_5^{dest} = [XYY]$.

- T6 → T7: because $ap(T6) \neq dp(T6)$ and no en-route reversal for T7 as well, thus $O_7^{ori} = O_7^{dest} = Rev(O_6^{dest}) = [YYX]$.

- Compare the propagated coupling order to the semi-fixed coupling order requested by the decoupling operation: $[YYX]$ is not compatible with $[\{X, Y\}Y]$.

- Conclusion: coupling order collision at T7.

Method 2: T8 → T7 → T6 → T5.

- T8 → T7: this involves a decoupling operation which is considered at the stage of partially finalizing the locally feasible coupling order of T7. With

known coupling order of T8, $O_7^{dest} = [O_8^{ori} Rev(O_9^{ori})] = [YXY]$. This coupling order must be compatible with the semi-fixed coupling order of T7 because this propagation only uses the fixed coupling order of the unit block serving T8 to replace the unfixed part in $[\{X, Y\}Y]$.

- T7 $\rightarrow$ T6: $O_6^{ori} = O_6^{dest} = Rev(O_7^{ori}) = [YXY]$.
- T6 $\rightarrow$ T5: $O_5^{ori} = O_5^{dest} = O_6^{ori} = [YXY]$.
- Compare the propagated coupling order to the fixed coupling order formed by the coupling operation: $[YXY]$ is not compatible with $[XYY]$.
- Conclusion: coupling order collision at T5.

Method 3: tracking starts with any intermediate unfixed trip, T6 in this example. The coupling order of T6 can be fixed via T5 or T7.

- T5 $\rightarrow$ T6: $O_6^{ori} = O_5^{dest} = [XYY]$.
- T7 $\rightarrow$ T6: $O_6^{dest} = Rev(O_7^{ori}) = [Y\{X, Y\}]$.
- O_6^{ori} and O_6^{dest} are not compatible to each other.
- Conclusion: coupling order collision at T6.

As shown in the example above, although the coupling order collision might be located in different places because of different propagation tracking methods, the same set of arcs are involved, which are all the get-in and get-out arcs of the nodes in a graph $g \in G_3$. For a given network-flow-level schedule, there may results in some individual coupling order collisions, and it is also possible that there are some collisions with overlapping arcs. For the latter case, the collisions with overlapping arcs can be considered via two strategies: consider them individually such that each conflict contains the overlapping arcs, or group them together as a combined conflict. Some experiments on these strategies will be shown in section 6.5.

6.4 An adaptive approach

This part proposes an adaptive method to resolve the station level constraints based on the basic network flow model. Station-level conflicts are detected from a given network flow schedule, which are translated into additional constraints of RS-Opt. Some conflicts can be detected by the visualization tool, *TRACS-RS* [4]. Moreover, a more analytical method of detecting conflicts for a given Phase I solution will be discussed in section 6.4.1. Figure 6.5 shows the flowchart of this method. The basic Phase I containing constraints C_1 to C_4 (referring

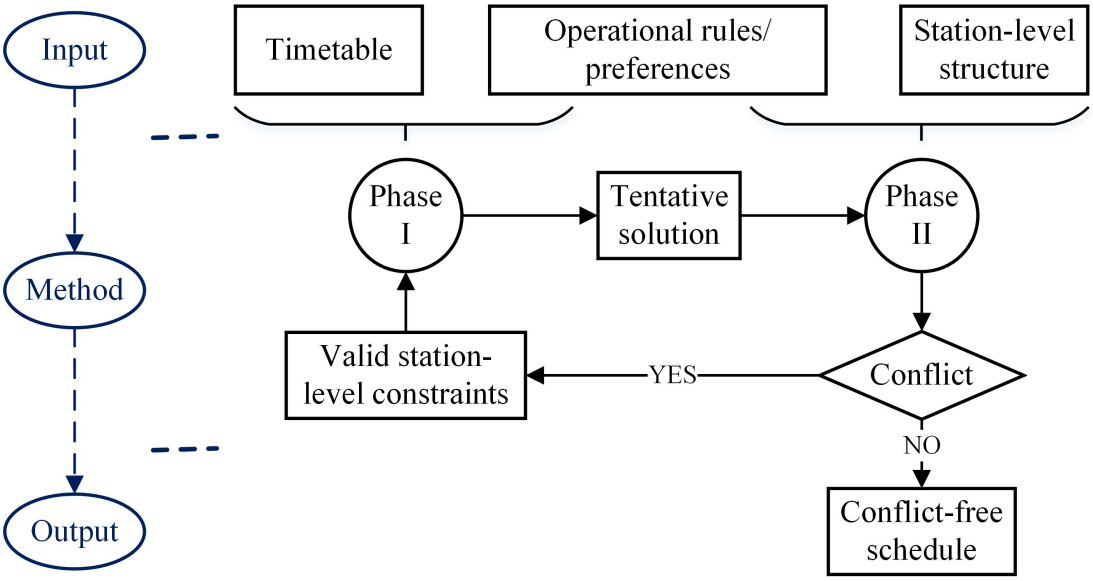


Figure 6.5: Flowchart for the adaptive approach

back to Chapter 4.2.3) is solved by RS-opt in [57]. Phase II attempts detecting potential conflicts and assigning feasible coupling orders. Once an unresolvable station-level conflict is encountered, a corresponding station-level constraint C_5 (seen in Chapter 4.2.3) will be formulated and added to RS-Opt. A new solution is to be sought once some new station-level constraints are added. If no more station-level blockage is detected, corresponding feasible coupling order of trips served by multi-units will also be given such that a solution with richer coupling order information will be delivered.

Phase II contains two core parts. The first part is the conflict detection and

coupling order assignment which is based on a series of factors, for instance, physical structure of railways and stations, timing and moving directions of trips, tentatively assigned arcs and unit flows for trips etc. For a given station operation environment, conflicts are detected when a tentative linkage is not operable, or feasible coupling order cannot be found at a certain platform, or a preassigned coupling order is not feasible to another operation environment. The second part is to extend Phase I and resolve the detected conflicts.

This method does not have the risk of getting stuck in a dead loop and we will discuss this from two aspects: ① The conflicts located in a network-level solution are sparse because many of the station level constraints could have already been satisfied implicitly by the basic network flow solution in Phase I. This is endorsed by the findings from the practitioner. ② Once a conflict is detected in a network-level solution, all the solutions that contain this conflict will not be considered as feasible such that they will never become a solution delivered by Phase II leading to an endless loop.

6.4.1 Coupling order assignment and conflict detection

A two-stage method is proposed to assign coupling order and detect potential infeasibility caused by crossing linkages or coupling order collision. The first stage is based on each platform which has two purposes: one is to assign locally a feasible coupling order with regard to coupling/decoupling operations introduced in 6.2; the other is to verify related linkage feasibility. The second stage is to find out the compatibility among locally fixed coupling order within each $g \in \mathcal{G}_3$ and also further fix some unfixed/semi-fixed coupling orders by spreading the locally feasible coupling order assigned by the first stage to the network.

Platform-based stage

This stage mainly assigns locally feasible coupling order and detects potential conflicts mainly caused by crossing linkages. The platform function can be modeled as a data structure behaving similar to a double-ended queue for storing and processing unit blocks, in which unit blocks are sequenced by "push" and "pop" operations, shown in Figure 6.6. A "push" operation refers to one unit

block getting into the platform with a "journey" and a "pop" operation refers to one unit block getting out of the platform with a "journey" after some necessary operations. The "journey" here can be either a timetabled trip or a shunting

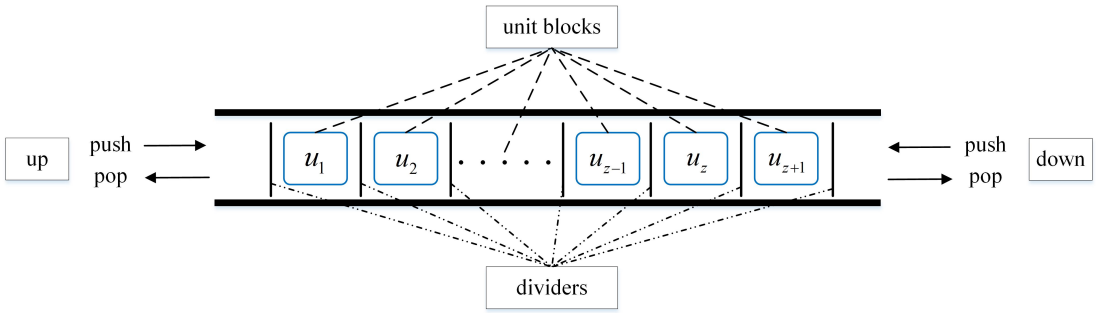


Figure 6.6: Data structure for *unitStore*

movement, for example a re-platforming movement. The two ends for "push/pop" operations correspond to two approaching directions to a through platform, which can be mapped to *up* and *down* directions. The dead-end platform can also be imitated by disabling one end. Besides, this data structure has capacity limitation regarding to the platform length. While pushing each arrival unit block, virtual "dividers" are added to isolate it from the existing unit blocks since unit blocks are physically separate when they arrive. This data structure is denoted as *unitStore*.

Algorithm 3 describes the process at a platform. It takes a platform h and basic solution s of Phase I as input. The output is *a richer train unit schedule* (s') with locally feasible coupling order assigned at each platform and verified linkages. Conflicts will be collected in CF_1 which will be further resolved. Three lists are formed for platform h: arrival list, departure list and linkage list connecting the arrivals to departures. This algorithm starts with the following initializations: dL and aL are time sorted departure and arrival trips respectively; *dtrip* and *time* is assigned with the first departure trip and its departure time respectively; *unitStore* is initialized as empty. Once *time* is assigned a new value, all the trips in aL whose arrival times are smaller than *time*, defined as set A_{dt}, will be attempted to push in *unitStore* once there is enough space. If there is not enough space, *shuntingAway*() function will be called to verify all the existing

Algorithm 3 Platform-based stage

Require: $s,\ h, (\forall h \in \mathcal{H})$
Ensure: s' *and* CF_1

$s' := s$; $dL := sortedDepTripList$; $aL := sortedArrTripList$; $dtrip := dL.firstTrip()$; $time := dtrip.depTime$; $unitStore := empty$
repeat
  for all $atrip$ **in** $A_{dt} = (aL \mid atrip.arrTime < time)$ **do**
    if $(unitStore.length + atrip.length \leq max)$ **then**
      $unitStore.push(atrip.composition)$
    else if $(\exists i \in unitStore:\ shuntAway(i)$ **and** $i.length \geq atrip.length)$ **then**
      $sh := shuntingMove$; $s'.update(sh)$; $unitStore.remove(i)$;
      $unitStore.push(atrip.composition)$
    else
      $dt := \{unitStore.depTrips, link(atrip).headNodes\}$
      $c := OCconflict$; $CF_1.add(c)$. // Over capacity conflict
      $dL := dL.remove(dt)$; $aL := aL.remove(atrip)$; $unitStore.clear()$
      if $dL.isEmpty() = false$ **then**
        $dtrip := dL.firstTrip()$; $time := dtrip.depTime$; **go to** for loop
      else
        break;
      end if
    end if
    $aL.remove(atrip)$
  end for
  $links := dtrip.getInLinkages$
  $verify := linkImplement(links, unitStore)$
  if $verify.operable = true$ **then**
    $orders := verify.orderList$
    $s'.update(orders)$; $unitStore.pop(dtrip.composition)$
  else
    $c := verify.LIconflict$
    $CF_1.add(c)$. // Linkage implementation conflict
    $u_c := c.linkedUnitblocks$; $unitStore.remove(u_c)$
    update *element sequence in unitStore*
  end if
  $dL.remove(dtrip)$; $dtrip := dL.firstTrip()$; $time := dtrip.depTime$
until $dL.isEmpty()$

unit blocks to find out if some of them can be shunted away to secure enough space for the newly push-in unit block. The linkage slack time will be verified during this process. If $shuntingAway(i) = true$, the newly push-in unit block can be stored in $unitStore$ and accordingly shunting movement will be saved in s'. Otherwise, all the arcs linked with those unit blocks will be considered as a conflict stored in CF_1 because they invalidate the platform capacity constraint, denoted as $OCconflict$. All the unit blocks will be cleared out from $unitStore$, and the departure trips corresponding to the unit blocks in $unitStore$ and $atrip$ will be removed from dL, and new $dtrip$ and $time$ will be assigned to restart the for loop.

After all the arrival unit blocks arriving earlier than current $time$ are successfully pushed into $unitStore$, the algorithm starts to deal with the unit blocks which are supposed to be popped out to serve the trip whose departure time is $time$. A function called $linkImplement()$, shown in algorithm 4, is applied to verify the feasibility of $links$ which are the get-in linkages of this departure trip with respect to the current $unitStore$ status. If all the get-in linkages are verified as operable, the locally feasible coupling orders will be updated to s' and the unit block serving this trip will be popped out of $unitStore$. Otherwise, the conflict related to the linkage implementation, denoted as $LIconflict$ will be saved into CF_1 to be resolved later and the unit blocks related to this conflict u_c will be removed from $unitStore$. Then $dtrip$ and $time$ will be renewed for next iteration. A conflict will not break this algorithm until all the departure trips is verified because this strategy tries to collect as many conflicts as it can.

For Algorithms 3, the 'repeat' operation is executed based on list dL. Under the 'repeat' loop, there are three parts whose complexities need to be considered: ① a loop on aL to find out A_{dt} is firstly executed; ② a loop based on A_{dt} to judge if the arrival trips in A_{dt} can be parked on platform h is launched, under which another loop on $unitStore$ is needed to check if any shunting movements can be inserted; ③ Algorithm 4 to finalize the corresponding linkages is called by Algorithms 3. Thus, the complexity of Algorithm 3 is $O(|dL| * (|aL| + |A_{dt}| * |unitStore| + O(Algorithm4)))$. The complexity of Algorithm 4 is $O(|ubs.arrTrips|)$ because it has only one loop based on $ubs.arrTrips$. The sizes of $|dL|$ and $|aL|$ depend on the given timetable and railway network. $|A_{dt}|$

Algorithm 4 *linkImplement*()

Require: *links*, *unitStore*
Ensure: *orderList*, *LIconflict*, *operable*
1: $c := links.count()$; $ubs := unitStore(dir, c)$
2: $i := ubs.firstArrTrip$; $j := ubs.lastArrTrip$; $operable = true$
3: **if** $links.tailTrips = ubs.arrTrips$ **then**
4: **if** $c = 1$ **and** $i.getOutArc.count() \geq 2$ **then**
5: $orderList.add(O_i^{dest})$. // Assigned by " − ", case ②
6: **else if** $c \geq 2$ **then**
7: $counter := 0$
8: **for all** at **in** $ubs.arrTrips$ **do**
9: **if** $at.getOutArc.count() = 1$ **then**
10: $counter = counter + 1$
11: **end if**
12: **end for**
13: **if** $counter = c$ **then**
14: $orderList.add(O_{dtrip}^{ori})$. // Assigned by " + ", case ③
15: **else if** $counter = c - 1$ **and** $j.getOutArc.count() \geq 2$ **then**
16: $orderList.add(O_{dtrip}^{ori}, O_j^{dest})$. // Assigned by both " + " and " − ", case ④
17: **else**
18: $operable = false$; $LIconflict$. // No feasible order assigned
19: **end if**
20: **end if**
21: **else**
22: $operable = false$; $LIconflict$. // Crossing linkages
23: **end if**

and $|ubs.arrTrips|$ are usually smaller than 4 in practice representing less than 4 train units allowed to be coupled together to serve next trip and $|unitStore|$ is bounded by the platform length that normally can park no more than 12 units. Let us use J, E, L, V to denote $|dL|$, $|aL|$, $|A_{dt}|$, $|unitStore|$ respectively. Therefore, the complexity of Algorithm 3 is $O(J * (E + L * V))$. As this algorithm is executed for every platform in $\mathcal{H}$, thus, the complexity is $O(J * \mathcal{H} * (E + L * V))$.

Algorithm 4, *linkImplement*(), is a sub-part of algorithm 3 and the coupling and decoupling operations ("+", "−") defined in section 6.2 are applied in this algorithm. There are two types of input for this algorithm: one the the *links* obtained at line 19 in algorithm 3, the other is the current status of *unitStore*.

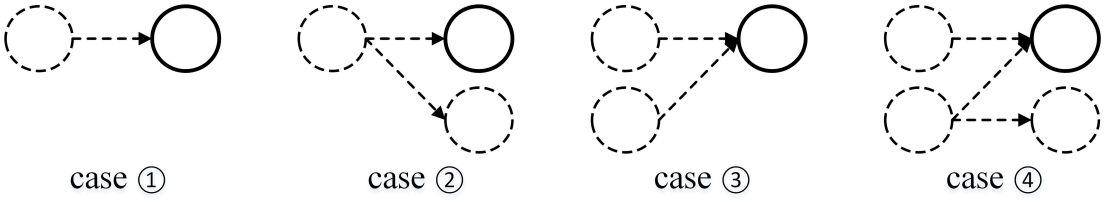


Figure 6.7: 4 possible cases of operations to form a departure trip

This algorithm focuses on verifying if the linkages in *links* of a departure trip (*dtrip*) are operable under the corresponding *unitStore* circumstances and returns an assigned locally feasible coupling order or a conflict. Let us define *dir* as the end (*up* or *down*) of *unitStore* that complys with the departure direction of *dtrip* and c is the total number of linkages in *links*. Thus, we define a function $unitStore(dir, c)$ to obtain c unit blocks counted from the end *dir*, stored in *ubs*. i and j are the arrival trips delivering the first and last unit blocks in *ubs* respectively. Let us consider an example by using the *unitStore* status shown in Figure 6.6. Suppose $c = 3$, $dir = down$, thus, $ubs = [u_{z+1}u_zu_{z-1}]$, and i and j refers to the arrival trips delivering u_{z+1} and u_{z-1} respectively. This algorithm firstly justifies if the unit blocks in *ubs* are delivered by the arrival trips which are actually linked to *dtrip*. If this condition is satisfied, the operations converting the linked arrival trips to *dtrip* will be investigated, which can be classified into four cases: ① no coupling or decoupling, ② decoupling only, ③ coupling only, ④ both coupling and decoupling, as shown in Figure 6.7. For case ① it will be claimed as operable if this condition is satisfied because it only requests standard turnaround time that has been considered while generating DAG. The case ② ③ ④ can be differentiated by counting c and get-out arcs of each arrival trips for the unit blocks in *ubs*. The feasibility of these cases will be ensured if a corresponding locally feasible coupling order can be assigned. Otherwise, this algorithm claims the given *links* as inoperable and returns the corresponding conflict.

Network-based stage

The result from the platform-based stage is the basis of the network-based coupling order propagation process to finalize more feasible coupling orders or to

detect conflict caused by coupling-order collision. As station operations are connected by running trips some unfixed/semi-fixed trips can be further determined taking advantage of the coupling order propagation on the network based on the solution graph of Phase I and the railway-network structure, and the coupling order conflict can also be captured during the propagation process over $g \in \mathcal{G}_3$. The issues considered at this stage include: coupling order propagating, en-route reversal of a unit block, and flexible timings for some empty shunting movements (discussed in section 6.4.1).

Algorithm 5 Network-based stage

Require: s', θ
Ensure: s'' *and* CF_2
1: $\mathcal{G}_3 := graphSplit(s')$; $s'' := s'$
2: **for all** g **in** $\mathcal{G}_3$ **do**
3: $i := g.fixedL.anyTrip$
4: $backwardSearch(eL_i)$; $forwardSearch(lL_i)$
5: **if** $propatingToEdge = true$ **then**
6: $orders := finalizedOrder(g)$
7: $s'' := s'.update(orders)$
8: **else**
9: $c := COconflict$; $CF_2.add(c)$. // Coupling order collision in g
10: **end if**
11: **end for**

Algorithm 5 focuses on the coupling order propagation through each $g \in \mathcal{G}_3$ that may contain some trips with fixed coupling orders assigned at the platform stage. Let define $fixedL$ as the fixed-order trip list for a $g \in \mathcal{G}_3$. For a trip $i \in fixedL$, considering i as a divider, the trips within g can be split into two sorted lists. One includes the trips earlier than i, called *earlier trip list* and denoted by eL_i; the other contains the trips later than i, called *later trip list* and denoted by lL_i. Starting from i, two directions coupling order propagation search will be launched, as shown in Figure 6.8, to further assign coupling order and detect coupling order collision. Trip i can be the first trip, the last trip, or any other trip, corresponding to one of those three different tracking methods of coupling order propagation that have been analyzed in section 6.3. The $backwardSearch(eL_i)$ and $forwardSearch(lL_i)$ start with either O_i^{ori} or O_i^{dest}. It is possible that the

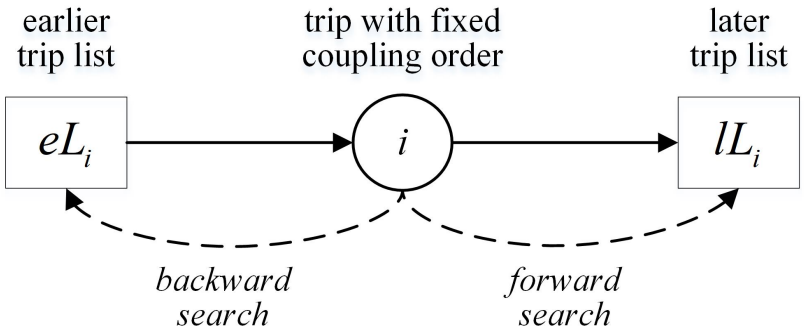


Figure 6.8: Backward and forward search within a $g \in \mathcal{G}_3$

platform stage assigns two incompatible orders to the origin and destination of trip i such that this g is claimed as a coupling order conflict directly. Otherwise, the coupling order of trip i at certain location will be propagated to other trips in g and compared to their coupling orders that may have been assigned at the platform stage. If the propagated order and the assigned order of each trip in g are compatible, g is claimed as conflict-free. The coupling orders of trips in g are finalized and updated to s''. Otherwise, all the linkages related to the trips in g will be collected as a coupling order collision to be stored in CF_2 for the further resolution. Let use M, Q, R to represent $|\mathcal{G}_3|$, $|eL_i|$, and $|lL_i|$. The complexity of Algorithm 5 is $O(M * (Q + R))$.

Flexible timings

The time available for unit blocks moving from arrivals to departures are limited, which is rigidly constrained by the timetable. Within this limited time range, a series of time-consuming operations must be accomplished at the corresponding stations such as coupling, decoupling, re-platforming, shunting to depot/siding, cleaning, equipment inspection etc. Suppose that a unit block u arrives with trip i at platform h and finishes all necessary operations which consume time of tp_h and later leaves from h to another platform h' to serve trip j. A dummy trip $\tilde{u}_{(i,j)}$ from h to h' is generated and its departure time $dt(\tilde{u})$ and arrival time $at(\tilde{u})$ are flexible. However, the duration time of dummy trip u is restricted by the clock time of timetable such that time boundaries for the departure and arrival times of

dummy trips must be considered to ensure no blockage caused at inappropriate time. Normally, $\tilde{u}$ can be moved away from h after the necessary operations are finished but it must be shunted away before the next arriving trip, and its departure time range is shown in expression (6.10). Besides, $\tilde{u}$ must arrive at h' earlier than the departure time of trip j minus necessary departure operations and later than the last departure trip at h', and its arrival time range is shown in expression (6.11).

$$at(i) + pt_h < dt(\tilde{u}) < at(i+1), \ \forall \tilde{u} \tag{6.10}$$

$$dt(j-1) < at(\tilde{u}) < dt(j) - pt_{h'}, \ \forall \tilde{u} \tag{6.11}$$

Figure 6.9 shows a simple example of avoiding blockage by manipulating flexible time boundaries together with coupling order decision. Suppose that h_1 and h_4 are dead-end platforms and h_3 is a through platform and the directions of $T3$ and $T4$ are the same. $T1$ is a re-platforming trip from h_2 to h_1 The following two procedures explain what time boundary is feasible.

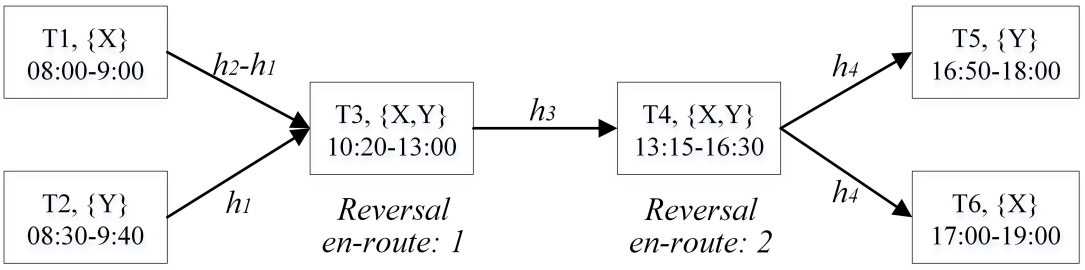


Figure 6.9: Avoid blockage by manipulating the flexible timings boundaries

Procedure 1. $at(T2) < at(\tilde{u}_1) < dt(T3) - tp_{h_1} \rightarrow O_3^{ori} = [XY] \rightarrow O_3^{dest} = Rev(O_3^{ori}) = [YX] \rightarrow O_4^{ori} = [YX] \rightarrow O_4^{dest} = Rev^2(O_4^{ori}) = [YX] \rightarrow u_1$ blocks the departure of $u_2 \rightarrow$ infeasible time boundaries.

Procedure 2. $at(\tilde{u}_1) < at(T2) \rightarrow O_3^{ori} = [YX] \rightarrow O_3^{dest} = Rev(O_3^{ori}) = [XY] \rightarrow O_4^{ori} = [XY] \rightarrow O_4^{dest} = Rev^2(O_4^{ori}) = [XY] \rightarrow O_4^{ori} = [Y]$ and $O_4^{ori} = [X] \rightarrow$ feasible time boundaries.

6.4.2 Conflicts resolving

The station-level resolution includes two main parts: coupling order assignment and conflict detection, and conflicts resolving. The first part is very important because it is the basis for the second part. If the first part claims a conflict-free solution, there is no need to launch the second part anymore. These two parts are complementary to each other. After the conflicts are detected by the first part, the second part converts these conflicts into linear constraints and feeds new constrains to RS-Opt. The method to form new constraints of eliminating the solutions that contain conflicts is described in the mathematical model part, seen in section 4.2.3 constraints C_6. However, generating the full conflict set of $\mathcal{Z}_1$ or $\mathcal{Z}_2$ in an original full DAG is very complicated as numerous arc/flow combinations. Through the proposed adaptive approach, the complexity of generating $\mathcal{Z}_1$ or $\mathcal{Z}_2$ can be avoided, and we only need to detect the conflicts existing in a Phase I solution. According to the observation on the Phase I solution, the potential conflicts at the station level are sparse, thus, the complexity of automatically assigning coupling orders and detecting potential conflicts is much smaller than that of generating the full conflict set $\mathcal{Z}_1$ or $\mathcal{Z}_2$ in an original DAG. The method of assigning feasible coupling order and detecting conflict have been systematically designed in section 6.4.1. Given a solution from Phase I, the detected conflicts are stored in CF_1 and CF_2 such that the constraints introduced in expressions (4.27) and (4.33) can be converted as expressions (6.12) and (6.13), defined as valid cuts.

$$\sum_{a \in \bar{A}} y_a \leq \mid \bar{A} \mid -1, \ \forall \bar{A} \in CF_1 \cup CF_2 \text{ (Arc selection)} \tag{6.12}$$

$$\sum_{B} (1 - x_a^q) + \sum_{Q} x_a^q \geq 1, \ \forall \breve{A} \in CF_1 \cup CF_2 \text{ (Type flow)} \tag{6.13}$$

Figure 6.10 shows the logistic of feeding valid cuts to the solver (RS-Opt) of Phase I. RS-Opt considers the original DAG $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ as input and gives a tentative solution $\mathcal{G}^* = (\mathcal{N}, \mathcal{A}^*)$. The station level attempts to assign feasible coupling orders to $\mathcal{G}^*$ and finalizes tentative linkages. During this process, the conflicts are detected at the platform stage and the network stage. Note that some conflicts at the platform stage can be locally resolved, which will not be

recorded in CF_2. Three strategies are introduced to resolve local conflicts, including swapping part of the unit diagrams for the same type of train units, inserting extra station shunting movements within the bearable time of corresponding linkages, manipulating the flexible timing of re-platforming/depot-return train units. The resolved platform-based results will be passed to the network-based stage to assign further coupling orders and detect coupling order conflict stored in CF_2. The conflict in CF_1 and CF_2 will be formed as valid cuts to be added to RS-Opt. RS-Opt will be launched again with detected valid cuts to deliver a new solution $\mathcal{G}^* = (\mathcal{N}, \mathcal{A}^*)$. The working mechanism of the valid cuts is to eliminate all the solutions containing the conflicts formed in the valid cuts. Thus, RS-Opt with added valid cuts will not consider a solution which contains any conflict that has been constrained in RS-Opt as a feasible solution. For a given schedule, if and only if no conflicts or only local resolvable conflicts are detected, the schedule will be reported as the final conflict-free schedule with assigned coupling order and finalized linkages.

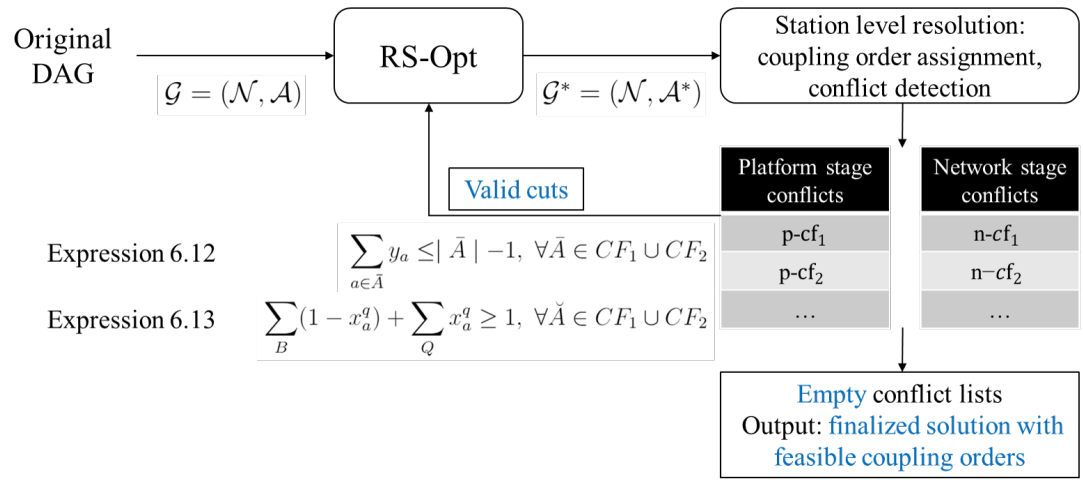


Figure 6.10: Logistic of feeding valid cuts to Phase I

6.5 Dataset description

The main goals in this part are to assign feasible coupling orders and finalize tentative linkages and the conflicts will be detected if the goals fail. In this research, coupling order assignment at the platform, coupling order propagation on the network, flexible times for re-platforming connections, and also reversal en-route are all featured. Real-world datasets may not contain all the features at the same time in a single dataset, or some of the features may not be present in every real-world dataset. However, these features are significant to be considered because there would be some severe consequences affecting the entire network operation when one of these cases occur, even though they may be infrequent. On the other hand, RS-Opt hardly delivers solutions for real-world datasets while considering the fixed-charge variables. However, the coupling order and linkage finalization are significant to be resolved at the station level to ensure an operable solution for the TUSO problem in practice. Thus, we design small but complex artificial datasets to capture all the features we considered in this Chapter.

6.5.1 Artificial datasets

For proof of concepts and verifying the correctness of the results, the experiments are firstly conducted based on a relatively small size artificial station-level structure, shown in Figure 6.11. This structure contains three terminating stations which are S1, S4, and S8, and their platform information are indicated in Figure 6.11. The other stations are intermediate stations where the coupling order

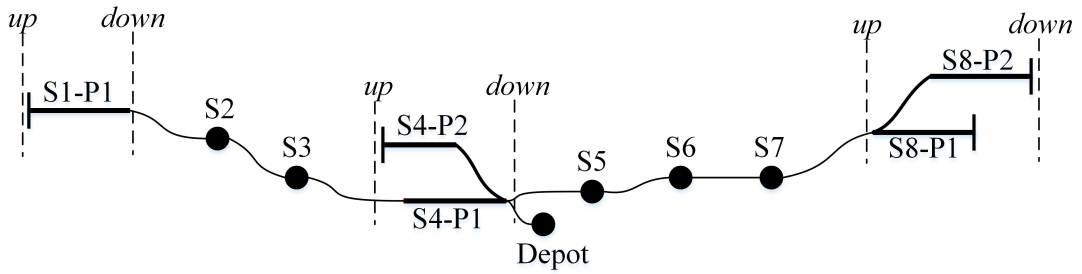


Figure 6.11: Station-level structure

reversal en-route may occur. $\{up\}$ and $\{down\}$ directions for each platform are marked to notify the relative approaching and leaving directions of unit blocks. Only one direction is available for dead-end platform, but two directions are usable for through platform. One depot connects with the *down* direction of two platforms of station S4, which means the units shunt to depot at this station can only leave their platforms through the *down* direction. For the purpose of manually verifying the correctness of solutions given by the approach proposed in this part which are claimed as conflict-free solutions, four small artificial datasets, containing trips running between all three O-D pair (S1-S4, S1-S8, S4-S8), are well designed for different purposes for instance coupling order assignment, conflict detection, conflict resolution, etc. Table 6.3 gives a summary of these four testing datasets. Datasets D3 and D4 have the same timetable such that they

Table 6.3: Summary of artificial data sets

Dataset	DAG	Unit types	Unit compatibility	Fleet size
D_1	(17, 73)	X,Y	Yes	2X,4Y
D_2	(25, 111)	X,Y	Yes	2X,5Y
D_3	(21, 91)	X,Y	Yes	5X,5Y
D_4	(21, 91)	X,Y	Yes	5X,5Y

have the same number of generated arcs. However, some trips in this timetable have different passenger demands, thus, their solutions could be different as well. Two types of compatible units are used on the network to satisfy diverse passenger demands.

6.5.2 Real-world datasets

The real-world datasets used in this research are derived from TransPennine Express (TPE), which is a British train operating company owned by FirstGroup that operates the TransPennine Express franchise. TPE runs regional and intercity rail services between the major cities of Northern England and Scotland, as shown in the map in Figure 6.12 [3]. Usually, TPE has a daily timetable of around 500 trips. RS-Opt is set as the full model containing the arc-selection variables (y_a), which struggles to deliver a solution for even small instances. Thus,

three sub-datasets are derived from the complete timetable as shown in Table 6.4, which is divided based on locations. Class 185 and Class 350 are usually not coupled together in real-life scheduling, but we consider them can be coupled for testing the coupling orders as the train unit sequence is much more significant when two different types of train units are coupled.

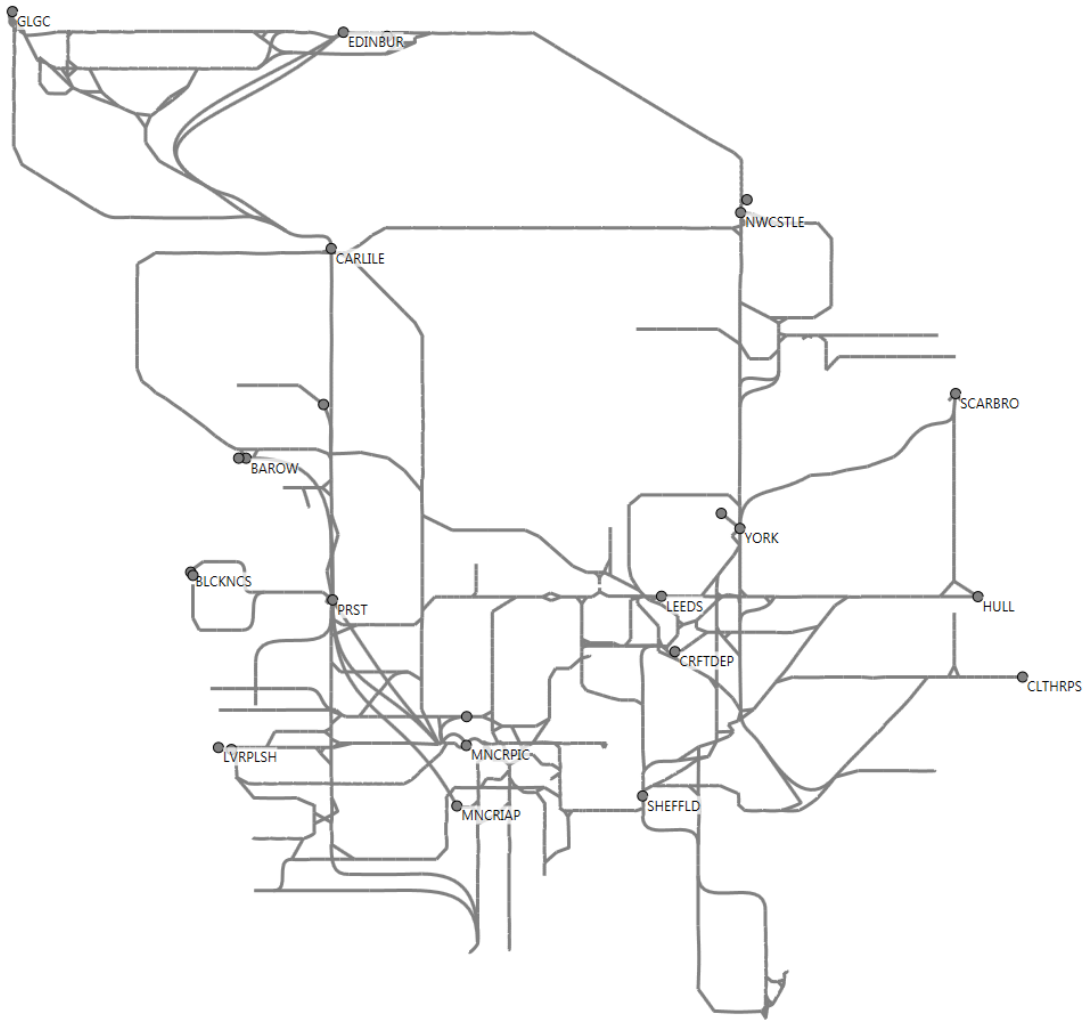


Figure 6.12: Rail map of TPE

Table 6.4: Summary of real-world data sets

Dataset	DAG	Unit types	Unit compatibility	Fleet size	Stations
$Sub1$	(97, 2671)	c185, c350	Yes	20, 21	9
$Sub2$	(126, 1913)	c185, c350	Yes	11, 17	7
$Sub3$	(65, 569)	c185, c350	Yes	6, 12	4

6.6 Experimental results

The experiments are applied by the station-level constraints C_{5a} taking use of the fixed-charge variables (y_a) because C_{5b} is very hard to be implemented in the existing RS-Opt. Phase I solver, RS-Opt, is written in FICO Xpress-MP 8.5 with Mosel, and the coupling order assignment and linkage finalization at Phase II are coded in C#. The experiments are conducted on a 64-bit workstation with 64G and an Intel Core i7-6700HQ CPU.

6.6.1 Results of artificial datasets

The solution of dataset D_1 from RS-Opt without any station-level constraints is conflict-free. The solution graph with assigned feasible coupling orders is shown in Figure 6.13. The coupling orders for T1, T8, and T10 are fixed at the platform-based stage, marked as red in Figure 6.13. The coupling order of T7 is semi-fixed by the platform-based stage $[Y\{XY\}]$, which means unit Y is the front but the sequence of $\{XY\}$ can not be assigned at the platform-based stage. Based on the results of the platform-based stage, the network-based stage further assigns the coupling orders for another 2 trips including T7 through coupling order propagation, marked as green in Figure 6.13. Trip T14 is formed by a coupling operation of two units in different types and unit Y served T6 involves a re-platforming operation. As T14 is the last trip served by unit block $\{XY\}$, its feasible coupling order can be either $[XY]$ or $[YX]$. The time of arriving at the departure platform of T14 is flexible but just early enough to complete the coupling operation with unit X from T12 at station S4 platform P2 to be ready to serve T14.

Dataset D2 runs three iterations to find a conflict-free solution, and detailed

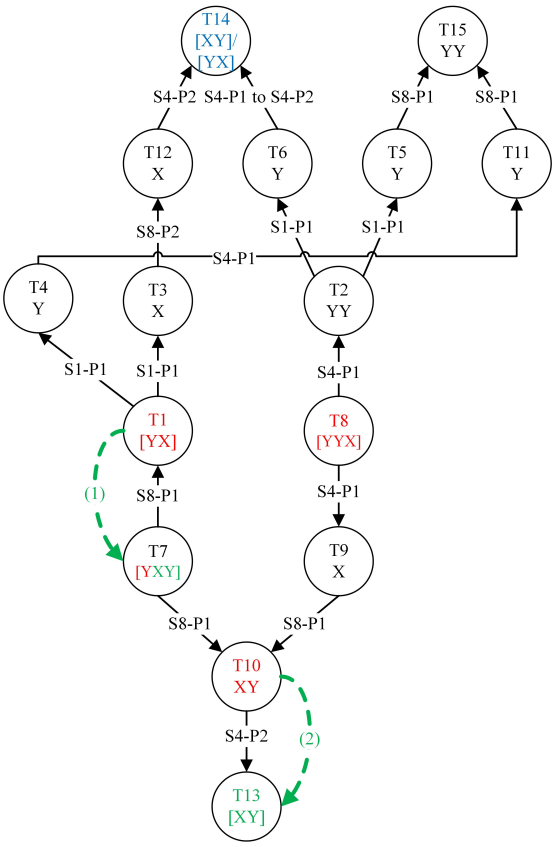


Figure 6.13: Solution graph with assigned feasible coupling order for D1

running information is shown in Table 6.5. We can notice that the solution fleet size of each iteration is the same and the objective function has a slight increase as we expected for compromising of producing a conflict-free schedule. In this process, 4 conflicts are detected during the platform-based stage, in which 2 of them are locally resolvable. Hence, only 2 station-level constraints (crossing linkages) are added back to Phase I which are shown in the fifth column in Table 6.5. Since all the conflicts are detected during the platform-based stage, the arc and flow changing through two iterations are visualized in Figure 6.14 where the red arrows represent crossing linkages. The conflicts are resolved by some arc and unit type changes for T14 and T20.

The experimenting information for dataset D3 is shown in Table 6.6 and the corresponding iterative solution graphs are presented in Figure 6.15. As shown in

Table 6.5: Information on each iteration for D2

Iteration	Objective	Fleet	Time	Conflicts	New constraint arc set
1	8.09951	4X,4Y	10.1s	0	No
2	8.09954	4X,4Y	11.3s	2	{80,85}
3	8.10041	4X,4Y	9.4s	2	{83,85}

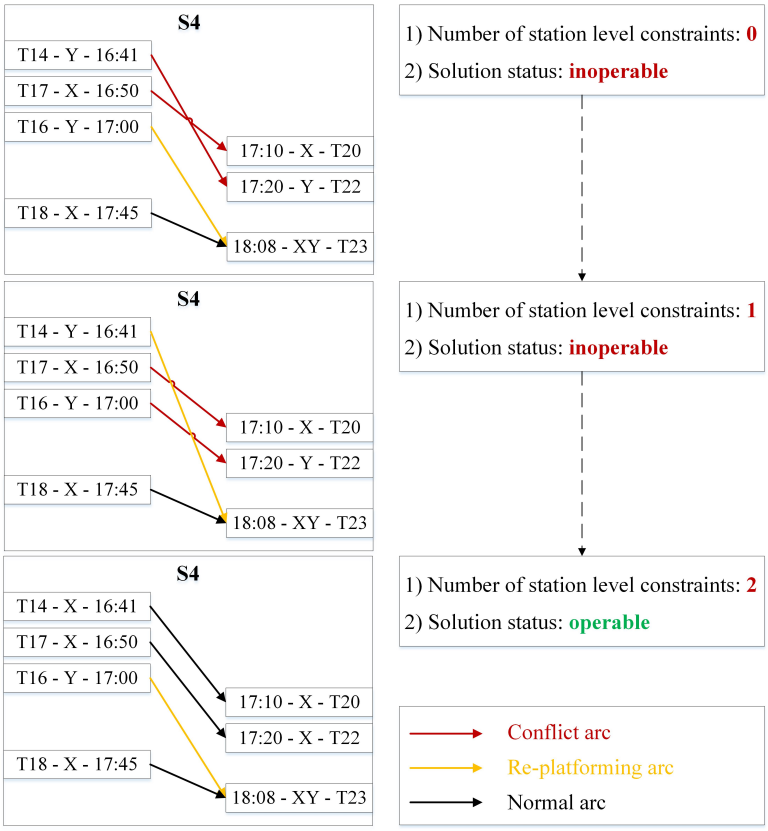


Figure 6.14: Station-level arc-flow changing

Table 6.6: Information on each iteration for D3

Iteration	Objective	Fleet	Time	Conflicts	New constraint arc set
1	6.08601	3X,3Y	11.5s	0	No
2	6.08603	3X,3Y	10.9s	1	{20,27,38,40,47,48}
3	7.08598	3X,4Y	207.1s	1	{34,43,51,52,61,62}

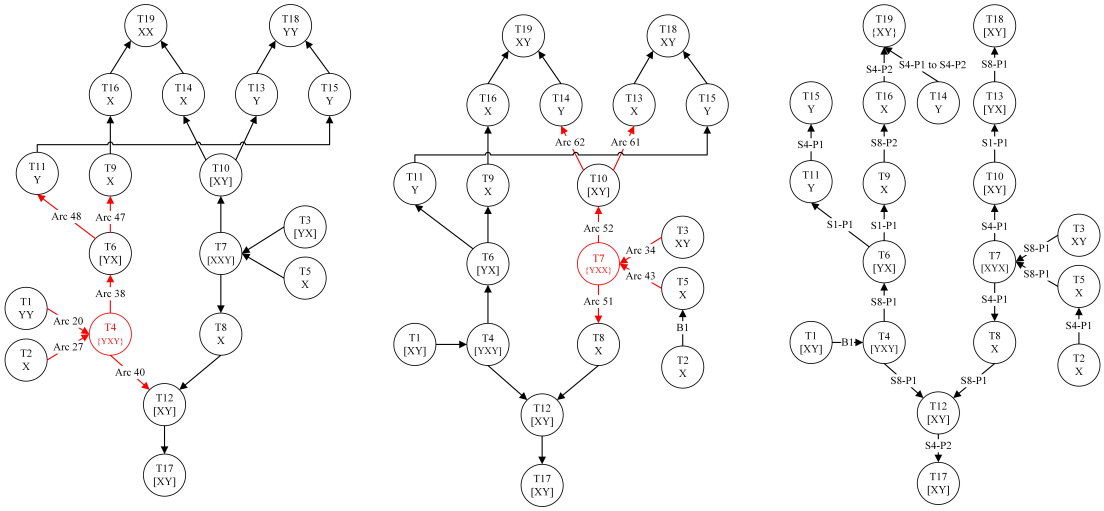


Figure 6.15: Iterative solution graphs for D3

Table 6.6, the fleet size used in the final solution has increased by one unit Y. It is because the problem size is small and there is not a lot of other feasible solutions with the same fleet size. For the real-world dataset, increasing some units to find a feasible solution at the station level would not be frequent since there would have more feasible solutions corresponding to the minimum number of train unit. For the solution from Phase I of each iteration, there is no conflict detected at the platform-based stage but one conflict is detected at the network-based stage, which has been marked as red arrows in Figure 6.15. In the first solution, a coupling order collision is located in the arc set $\{20, 27, 38, 40, 47, 48\}$, which is converted as constraint (6.14) such that all the solutions that contain these arcs are eliminated. With constraint (6.14), RS-Opt deliveries a new solution as shown in the second graph in Figure 6.15. However, the second solution still contains a coupling order collision in the arc set $\{34, 43, 51, 52, 61, 62\}$, converted as constraint (6.15). RS-Opt has been launched again with constraints (6.14) and (6.14) to deliver another solution which is finalized as conflict-free with assigned coupling orders seen in the third solution graph in Figure 6.15.

$$y_{20} + y_{27} + y_{38} + y_{40} + y_{47} + y_{48} \leq 5 \tag{6.14}$$

$$y_{34} + y_{43} + y_{51} + y_{52} + y_{61} + y_{62} \leq 5 \tag{6.15}$$

For dataset D4, two individual conflicts are detected for the solution without any station-level constraints and two new constraints are added through the first Phase II iteration. However, the new solution turns out two overlapping conflicts and the conflict arc sets are {20,27,38,39,47,48} and {34,39,51,52}. As it is seen here, arc 39 belongs to two conflicts. Hence, two strategies have been taken to tackle this type of overlapping conflicts. One is to treat them as two constraints since no matter which set of arcs are selected simultaneously will definitely invalid the solution. Two more iterations are needed to find a feasible solution. The other is to consider them as an integrated constraint since it is possible to have a feasible solution if the flows on any detected incompatible arc set have a slight change. The experiment results show the solution following this strategy is conflict-free directly after adding this integrated constraint. This experiment shows that two strategies can produce feasible results but the second strategy gives more flexibility in terms of arc selection, which means the second strategy is not as over-tight as the first strategy. Compare to the first-strategy solution, this solution has higher unit-usage efficiency in terms of fleet size, besides, the solution graph is much simpler. We also tried to combine these two strategies, but the results are still not as efficient as the second strategy.

Table 6.7 shows the information of arc selection of all datasets, where Arcs(1) represent the number of arcs selected in the solution without station-level constraints and Arcs(2) is on behalf of the number of arcs chosen in the final conflict-free solution. The arc overlapping percentage between Arcs(1) and Arcs(2) are

Table 6.7: Information on arc selection for D1, D2, D3 and D4

Dataset	Arcs(1)	Arcs(2)	Overlapping(%)	Total conflicts	Iterations
D1	24	24	100%	0	1
D2	31	31	87.1%	4	3
D3	27	28	85.2%	2	3
D4-strategy 1	27	28	85.2%	4	5
D4-strategy 2	27	31	92.6%	8	3

very high. Considering the solution structure, we notice that the conflict-free

solution only changes a small portion of arcs around detected arcs and tries to reserve most of the other low cost arcs. The conflict-free optimal solution is always found within reasonable time and iterations. Besides, the feasible coupling orders and flexible timings can be determined simultaneously.

6.6.2 Results of real-world datasets

The experiments on the artificial datasets endorse the correctness and effectiveness for the proposed method. The artificial datasets are purposed designed to capture all the features described in this method and there are only very limited number of feasible solutions corresponding to the optimal number of train units. These two reasons lead the artificial datasets to be more complicated to solve and obtain a conflict-free solution at the station level. In other words, real-world datasets may be easier (less iterations) to be solved.

Table 6.8 gives a summary of the results of datasets $Sub1$ and $Sub2$. These two datasets do not involve any coupling or decoupling operations, thus, there is no coupling order conflict detected at the network propagation process. As

Table 6.8: Information on each iteration for $Sub1$ and $Sub2$

Data	Iteration	Objective	Fleet size	Time (s)	Conflicts
$Sub1$	1	31.839	20, 21	16	3
	2	32.2474	20, 21	26696	4
$Sub2$	1	34.0061	11, 17	39	3

we can see, the conflict-free solution schedule of $Sub1$ needs two iterations to be finalized, and the details of each iteration are shown in Table 6.9. At the first iteration, three platform conflicts are detected, in which the first two conflicts at the platform 1 of MNCRIAP (Manchester Airport station) are locally resolvable by taking advantage of the interchangeability between the same type of train unit. However, the third platform conflict at the platform 1 of LVRPLSH (Liverpool Lime Street station) is unresolvable, marked as bold and underlined. Thus, this conflict is converted as a new constraint (6.16) added to RS-Opt to seek another solution. In the new iteration, RS-Opt gives a solution with the same fleet

Table 6.9: Iterative conflicts of $Sub1$

Iteration	Location	Arc ID	Remark	Local resolvable
1	MNCRIAP_1	{467,525}	Crossing linkages	Yes
	MNCRIAP_1	{655,667}		Yes
	LVRPLSH_1	{314,357,430}		**No**
2	NWCSTLE_2	{123,351,154}	Crossing linkages	Yes
	NWCSTLE_2	{556,590}		Yes
	NWCSTLE_2	{220,293}		Yes
	LVRPLSH_1	{605,665}		Yes

size but slightly higher objective function value. At Phase II, another four platform conflicts (located at NWCSTLE_2 and LVRPLSH_1) are detected. As all of them are locally resolvable, the locally resolved solution is the final conflict-free schedule.

$$y_{314} + y_{357} + y_{430} \leq 2 \tag{6.16}$$

However, the solution of $Sub2$ obtained at the first iteration can be finalized directly because only three local-resolvable platform conflicts located at MNCRIAP_1 are detected, seen in Table 6.10 that can be resolved by using interchangeability.

Table 6.10: Platform conflicts of $Sub1$

Iteration	Location	Arc ID	Remark	Local resolvable
1	MNCRIAP_1	{501,558}	Crossing linkages	Yes
	MNCRIAP_1	{1511,1542}		Yes
	MNCRIAP_1	{1603,1628}		Yes

The solution of $Sub3$ from the network level is finalized as conflict-free at the station level, where three coupling operations are assigned with feasible coupling orders, as shown in Figure 6.16 and Figure 6.17. The coupling operation in Figure 6.16 is operated at Edinburgh Waverley station. The coupling order in this Figure is not significant because of the interchangeability between the same type of train unit. Figure 6.17 contains one coupling operation and one decoupling operation, operated at GLGC (Glasgow Central Station). The coupling order

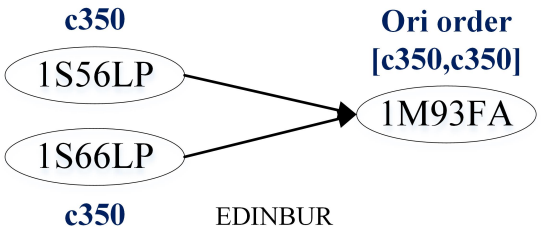


Figure 6.16: Coupling operation at Edinburgh Waverley station

of trip 1M96FP at its origin is $[c350, c185]$, formed by the coupling operation between the train units serving trips 1S35LP and 1S35LL. The coupling order of trip 1S71LP at its destination is $[c185, c350]$, requested by the decoupling operation involving 1M94FA and 1M94LL. As 1M96FP and 1S71LP have no reversal en-route, the assigned coupling orders is the same during the whole journeys. The coupling orders between 1M96FP and 1S71LP are compatible as this arc

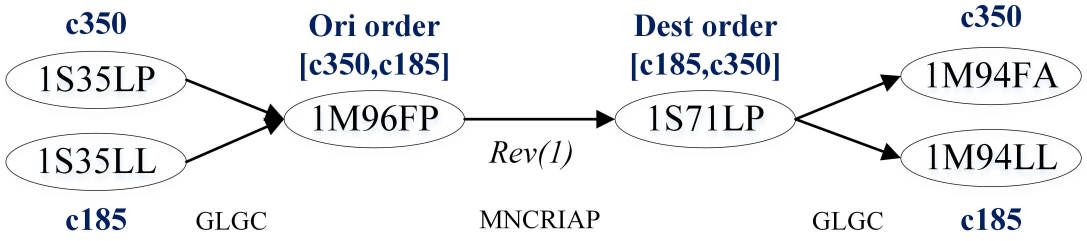


Figure 6.17: Coupling order propagation

is operated at a dead-end platform of MNCRIAP (Manchester Airport station) such that there must be a reversal operation. Besides, the solution has five re-platforming operations that do not involve any coupling/decoupling operations. Therefore, they are fixed by assigning the same platforms to the involved trips.

6.7 Discussions

The network-flow model has the limitation of ignoring station level constraints, which leads to an incomplete solution. This defect restricts the operability when

it is implemented at the station level because of a set of undecided factors, for instance, the coupling orders impacted by station layouts, timings, and unit movement directions. This research scrutinizes the potential problems in a solution from the network level. A multi-commodity fixed-charge integer linear programming including station-level conflict elimination is proposed. Based on the research on the network level of TUSO, an adaptive approach is proposed to selectively consider detected station-level constraints to efficiently produce a complete and operable conflict-free solution with assigned feasible coupling orders and finalized linkages. The experiments on the artificial and real-world datasets show the correctness and efficiency of the proposed method. Although we have already discussed that the proposed method is hardly stuck in a dead loop (seen in the third paragraph of section 6.4), we still discuss a potential backup method that can avoid a dead loop. Taking advantage of SLIM, a set of ranked near-optimal solutions can be initially generated as tentative solutions to be feed to Phase II to assign conflict-free solutions, and Phase II picks up the best conflict-free solution as its output if there is any. In theory, this process can be repeated until a conflict-free solution is found, or the (good-enough) candidate solutions in Phase I are all checked by Phase II.

Chapter 7

Conclusions and future work

7.1 Conclusions

This research contributed a major step forward in deriving fully operable optimized train unit schedules, which started new research to follow on from the network level to the station level scheduling. This research provided a strong basis for future research in this area. Moreover, a novel approach for evaluating the effectiveness of objective function designs is extended from the research of train unit scheduling optimization.

In this thesis, the train unit scheduling optimization (TUSO) problem, particularly for the UK railway industry, has been studied focusing on the effectiveness of alternative objective function designs and station-level resolution. To the best of the author's knowledge, the studies of these two aspects of TUSO are very scarce in the literature. Conventionally, the network-level and station-level problems of TUSO are researched separately. For the network-level problem, the researchers from the Netherlands and Italy have achieved fruitful outcomes. The station-level problem, defined as a shunting problem within a large station/depot, has been studied by the Netherlands and Denmark research groups. The differences between TUSO studied in this thesis and other similar problems are discussed in chapter 2 and chapter 3. Before this research, there is no study considering the feasibility of a network-level solution when it needs to be implemented at the station level. As far as the author is aware, there is no known

successful commercial software optimizing TUSO even at the network level only in real-life scheduling, in contrast with the bus/train crew scheduling and the flight scheduling. A solver automatically solving TUSO at the network (RS-Opt) is proposed and developed by Lin and Kwan [57]. However, there are still some parts left open to be further finalized. On this basis, this research aims to bring the station level into the big picture and produce a more operable solution. This research is carried out with close collaborations with a software company (Tracsis Pcl) and two UK railway operators (GWR and TPE). They provided valuable suggestions based on their experience of real-world train unit scheduling, and experimental datasets. Besides, this research is also benefited from the interactive unit diagramming software TRACS-RS, supported by Tracsis plc, particularly in the visualization of schedule results. This chapter summarises the research in this thesis as follows.

7.1.1 Problem description

To the author's knowledge, there is no previous literature available for the TUSO problem combining the network level with the station level. This thesis gives a detailed description of TUSO at both levels. As the network level has been described thoroughly in literature [56, 57], this thesis mainly focuses on the concerns that need to be resolved and finalized at the station level containing the following aspects: coupling order formation, coupling order network propagation, coupling order reversal en-route, sufficient slack time, crossing linkages, platform type and capacity, re-platforming shunting, and interchangeability between the same type of train unit. Besides, using customized objective function designs to improve solution quality is discussed.

7.1.2 Modeling

The original DAG, generated based on a timetable as well as some basic constraints, is firstly described on which decision variables are defined. The arc usage cost assigned according to the type of arc and the slack time cost calculated based on the arc time gap are introduced. A novel multi-commodity fixed-charge integer linear programming for TUSO is proposed, covering the constraints at both

the network and the station levels. The optimization criteria are formulated one by one for the convenience of evaluating the effectiveness of alternative objective functions considering different terms. There is an optimization hierarchy among the optimization criteria such that the weights can be accordingly assigned to maintain this hierarchy. The station-level constraints prevent the solution from choosing any conflicts, referring to $ArcSelection$ conflicts or $TypeFlow$ conflicts. This model does not assign coupling orders directly, but the station-level constraints can be used to eliminate the arc/flow combination that carries infeasible coupling orders if they can be captured in advance.

7.1.3 Objective function evaluation

Inspired by the investigation on the experimental results from RS-Opt [57] and SLIM [26], a generalized methodology to establish confidence in the effectiveness of alternative objective function designs, where different optimization criteria are considered, is proposed. Firstly, a detailed description of this problem is given that is different from the multi-criteria optimization and automatic methods for algorithm parameter control. The proposed methodology contains three stages: design alternative objective functions, obtain solutions and their features, evaluate objective function effectiveness. The optimization criteria are firstly investigated for a specific scheduling problem, and they are converted into mathematical terms formed into alternative objective functions. The second stage relies on two algorithms to obtain solutions. One is an exact method, and the other is a hybrid method with a heuristic wrapper iteratively launching the exact method. Then, the features indicating the effectiveness of objective function are analyzed, including quantified values of optimization criteria, schedule structure, the number of different solutions obtained during the iteration of the hybrid heuristic method, the comparisons between the converged objective function values obtained through two solvers. Those features are considered into the integrated evaluating system designed based on the analytical hierarchy process [72]. The experiments carried out by TUSO instances support the effectiveness of this methodology. The criterion of slack time plays an important role in improving the effectiveness of the objective function. On the contrary, the term of mileage

may have a negative impact on objective function effectiveness.

7.1.4 Station-level resolution

Chapter 6 proposes an adaptive approach to solve TUSO at both the network and the station levels. Phase I is the network-level problem, and RS-Opt [57] is used as the base solver producing the network solution. Phase II tackles the concerns left in the solution given by Phase I. The main features handled in Phase II are summarised as follows.

- The features relevant to the coupling order are formally defined: coupling order definition, coupling/decoupling operation, coupling order network propagation, coupling order reversal en-route.
- Locally feasible coupling orders are assignment based on the coupling and decoupling operations together with the timings and moving directions at the operating station (platform) of involved trips. When these locally feasible coupling orders propagate through the network, more feasible coupling orders can be assigned.
- Conflicts are detected through two steams. One is to detect coupling order conflict through the coupling order propagating process. The other is to detect conflicts during the platform operating including crossing linkages, platform capacity violation, shortage of operational time.
- The detected conflicts are classified into two types: local resolvable conflicts, and un-resolvable conflicts. The un-resolvable conflicts are converted into new constraints added to Phase I to seek a new solution. While there are no any un-resolvable conflicts detected, the complete conflict-free solution will be delivered.

The experiments are carried out with artificial datasets and real-world datasets. Artificial datasets are designed to cover all features that need to be considered and small at the same time to verify the correctness of the proposed method. The real-world datasets results show the proposed method can efficiently assign coupling orders and produce more operable solution at the station level.

7.2 Future work

Although the work of this thesis has achieved some novel and useful results that improve the existing solver for TUSO (RS-Opt) performance and more station-level features are captured as well, the research on the TUSO problem at both the network and the station levels in the UK is still far from being mature. Here, a series of aspects that would be worth exploring to improve the established research are summarised as follows.

7.2.1 A heuristic for TUSO at the network level

This part introduces a heuristic method to construct a feasible solution for TUSO at the network level based on the original DAG generated in advance, seen in section 4.1.1. The solution graph is a sub-graph of the original DAG, and the flow on it should be all integers, which are the train unit combinations. This method firstly generates all the feasible train unit combinations for each trip, defined as candidates, thereafter, constructs connections among these candidates.

Candidate generation

For each trip, we need to consider five hard constraints, including passenger demands, unit coupling upper bound, carriage upper bound, permitted train unit types, and compatibility between types. Based on these constraints, all feasible candidates of train unit combinations for each trip can be obtained in advance. A fix-size array with integer values is defined to represent a candidate. The array size is the total number of train unit types used in the given timetable. Each index of this array indicates a specific unit type, and the element value at each position is the number of the corresponding train unit type used in the candidate. If one unit type is not allowed to serve a trip, its element value in the arrays of all candidates for this trip will be always zero. Besides, if some unit types are not compatible to be coupled together, the candidates containing these types at the same time will not be generated. For instance, there are three unit types (X, Y, Z) used in a timetable, and their information is as seen in Table 7.1. Type X and Z are not compatible, i.e., they are not allowed to be coupled together. Suppose

Table 7.1: Information of different unit types

Type	Seats	Cars	Can serve	Not compatible with
X	122	2	t_1, t_2	Z
Y	147	3	t_1, t_2, t_3	–
Z	160	3	t_2, t_3	X

that the train unit coupling upper bound is 3, and the carriage upper bound is 6. The seat demands of trips t_1, t_2, and t_3 are 244, 269, and 307 respectively. The feasible candidates for these trips are listed in Table 7.2.

Table 7.2: Feasible candidate train unit combinations for each trip

Trip	Demands	Unit combinations
t_1	244	[2,0,0] [1,1,0] [0,2,0] [3,0,0]
t_2	269	[1,1,0] [0,2,0] [0,1,1] [0,0,2] [3,0,0]
t_3	307	[0,1,1] [0,0,2]

Notations

The notations and remarks used to illustrate the heuristic are presented in Table 7.3. The feasible candidates for each trip is denoted as V_i, $i \in N$, where N is the trip set. The passenger demand for each trip is denoted as d_i. Similar to the definition in the DAG described in section 4, A_i^{in} and A_i^{out} are defined as get-in and get-out arc sets of trip i respectively. S is used to store the constructed arcs and flows. The others are the lists used in the arc-constructing process.

Heuristic for arc construction

Based on the generated candidates for each trip, this part introduces a heuristic method to construct the arcs among candidates according to a set of optimization criteria. This method starts with the first trip of a working day, to which a desirable candidate is firstly assigned. A directed arc connects two trips where the trip starts the arc is defined as the tail trip and the trip ends the arc is defined as the head trip. At each step, one arc connecting a candidate e_n ($e_n \in V_i$) of the

Table 7.3: Notations

Notations	Remarks
V_i	Set of candidates for trip i, $\forall i \in N$
d_i	Passenger demands of trip i, $\forall i \in N$
A_i^{in}	Get-in arc set of trip i, $\forall i \in N$
A_i^{out}	Get-out arc set for trip i, $\forall i \in N$
S	List to store constructed arcs flows
FB	List of flow-balanced trips
UE	List of unexplored trips sorted by departure times
$EUSF$	List of explored trips that have not been satisfied nor flowed
$ESUF$	List of explored and satisfied trips that have not been flowed
$EFUS$	List of explored and flowed trips that have not been satisfied

tail trip (i) to a candidate e_m ($e_m \in V_j$) of the head trip (j) is constructed, until all trips are assigned with appropriate candidates.

Algorithm 6 gives the main logic of the heuristic of constructing a feasible solution, which requires the original DAG $\mathcal{G}$, candidates V_i and passenger demands d_i for each trip as input to ensure a feasible integer solution (S) as output. At the beginning of this algorithm, no trip has been assigned any candidate thus UE is initialized as N. In the algorithm, the elements in UE will be gradually removed, and it will be an empty list at the end of this algorithm. The other lists are initialized as empty, but temporarily store trips at different statuses. $EUSF$, $ESUF$, $EFUS$ will be empty again when the algorithm finishes. For a trip in $EUSF$, a priority for 'satisfying train unit' is given over 'flowing away train unit', i.e., get-in arcs have the priority to be constructed. $FB = N$ is the stopping criterion. For a trip i which has been assigned a feasible candidate, there are four statuses during the arc constructing process: ① the flow of this candidate is satisfied by the flows from the get-in arcs, ② the flow of this candidate is partially satisfied by the flows from the get-in arcs, ③ the flow of this candidate is completely flowed out through the get-out arcs, ④ the flow of this candidate is partially flowed out through the get-out arcs. These four statuses are denoted as $usComb$, $sComb$, $ufComb$, $fComb$ respectively. When the condition $i.usComb + i.sComb = i.ufComb + i.fComb = i.comb$ is hold,

Algorithm 6 Arc-constructing heuristic

Require: $\mathcal{G}(\mathcal{N}, \mathcal{A})$, V_i, d_i $(\forall i \in N)$
Ensure: S
1: $UE := N$; $EUSF, ESUF, EFUS, FB, S := emptyList$
2: **repeat**
3: **if** $(EUSF = EFUS = ESUF = emptyList)$ **then**
4: $i = UE.firstTrip$
5: $i.comb = MostFitComb(V_i)$
6: **update** $i.usComb$, $i.sComb$, $i.ufComb$, $i.fComb$
7: $UE.Remove(i)$; $EUSF.Add(i)$
8: **else if** $(EUSF.count \neq 0$ **or** $EFUS.count \neq 0)$ **then**
9: **for all** i **in** $EUSF \cup EFUS$ **do**
10: $BackwardConstruct(i)$
11: **end for**
12: **else if** $(ESUF.count \neq 0)$ **then**
13: **for all** i **in** $ESUF$ **do**
14: $ForwardConstruct(i)$
15: **end for**
16: **end if**
17: **until** $FB = N$

the flow on i is balanced. If $i.sComb \neq i.comb$ and $i.fComb \neq i.comb$, i is stored in $EUSF$; If $i.sComb = i.comb$ and $i.fComb \neq i.comb$, i is stored in $ESUF$; If $i.sComb \neq i.comb$ and $i.fComb = i.comb$, i is stored in $EFUS$; If $i.sComb = i.comb$ and $i.fComb = i.comb$, i is stored in FB. If lists $EUSF$, $ESUF$, $EFUS$ are all empty, the trip i in UE with the earliest departure time will be chosen as the start trip and the method $MostFitComb(V_i)$ assigns the candidate with the lowest seat over-provision to i, where $i.usComb$ and $i.ufComb$ are updated as $i.comb$. Thus, i is removed from UE and added to $EUSF$. This step triggers the arc-constructing process. For every trip in $EUSF$ and $EFUS$, a $BackwardConstruct()$ method is designed to construct the get-in arcs, seen in the algorithm 7. For each trip in $ESUF$, a $ForwardConstruct()$ method is designed to construct get-out arcs. A trip whose flow on its get-in and get-out arcs are balanced with the assigned candidate will be added into FB. The algorithm will stop when all trips are satisfied by appropriate candidates, i.e., $FB = N$. If $UE.firstTrip$ is used more than once, this means the solution contains multiple

independent fully connected sub-graphs if the arc direction, sign-on and sign-off arcs are ignored.

The $BackwardConstruct(i)$ method in algorithm 7 constructs get-in arcs for the given trip i where the candidate of trip i has been fixed. This algorithm takes $\mathcal{A}$ as input and ensures the arc, arc flow, and a candidate for the tail node

Algorithm 7 $BackwardConstruct(i)$

Require: $\mathcal{A}$
Ensure: a_{ji}, $a_{ji}.comb$, $j.comb$
1: **repeat**
2: $a_{ji} = ShortestArc(A_i^{in})$
3: **if** (a_{ji} is a sign-on arc) **then**
4: $a_{ji}.comb = i.usComb$
5: **else if** ((j **in** $EUSF \cup ESUF$) **and** $Intersection(i.usComb, j.ufcomb) \neq \emptyset$) **then**
6: $a_{ji}.comb = Intersection(i.usComb, j.ufcomb)$
7: **else if** ((j **in** UE) **then**
8: $j.comb = MostFitComb(V_j, i.usComb)$
9: **if** $Intersection(i.usComb, j.comb) \neq \emptyset$ **then**
10: $a_{ji}.comb = Intersection(i.usComb, j.comb)$
11: **end if**
12: **else**
13: $A_i^{in} = A_i^{in}.Delete(a_{ji})$; $A_j^{out} = A_j^{out}.Delete(a_{ji})$
14: **go to line 3**
15: **end if**
16: $S.Add(a_{ji}, a_{ji}.comb)$
17: **update** $i.sComb$, $i.usComb$, $j.fComb$, $j.ufComb$
18: $A_i^{in} = A_i^{in}.Delete(a_{ji})$; $A_j^{out} = A_j^{out}.Delete(a_{ji})$
19: **until** $i.sComb = i.comb$
20: **update** UE, $EUSF$, $ESUF$, $EFUS$, FB

of the constructed arc a_{ji} as output. Only one get-in arc is constructed at each iteration. If the main loop is operated more than once to satisfy trip i, i needs coupling operations. At each iteration, the arc a_{ji} with the shortest slack time is chosen by the $ShortestArc()$ method because a compact graph is preferred by real-world scheduling. Besides, short arcs have the advantage of encouraging a train unit to cover more trips that decreases the total number of train units in the solution.

The tail node j may be at three statuses stored in different lists or the source. If a_{ji} is a sign-on arc, we will assign $i.usComb$ to $a_{ji}.comb$ directly as we assume the source node can provide any candidate. If j is in lists $EUSF$ or $ESUF$, j has been explored by some arcs and its candidate has been assigned. Therefore, the new constructed arc a_{ji} is feasible if the intersection between $j.ufComb$ and $i.usComb$ is not empty, and the flow on arc a_{ji} is the intersection value. While j is in list UE, the candidate of j is not assigned yet. The $MostFitComb()$ method taking two parameters is called to find the most appropriate candidate for j from V_j to suit $i.usComb$. For each pair (e_m, $i.usComb$), $\forall e_m \in V_j$, a value indication how the suitness between e_m and $i.usComb$ is calculated by expression (7.1), where w_1, w_2 are the weights for the corresponding terms.

$$p_m = w_1 x_m + w_2 y_m, \forall m \in |V_j| \tag{7.1}$$

The first term (x_m) is the cost of using the train units in e_m to serve $i.usComb$, classified into three categories: (i) if $e_m = i.usComb$, there will be no decoupling operation for j, therefore, cost 1 can be used; (ii) if $e_m \subset i.usComb$ ($i.usComb \subset e_m$), this connection is associated with at least one coupling operation for i (decoupling operation for j), thus, cost 2 can be assigned; (iii) if $e_m \cap i.usComb \neq \emptyset$ and the former two conditions are not satisfied, this connection indicates at least one decoupling for j and one coupling for i such that a cost of 3 can be used. The second term is the over-provision of seats if use e_m to serve trip j. If the first three conditions in $BackwardConstruct(i)$ do not suit to a_{ji}, a new shortest arc will be sought until an arc is successfully constructed to S. At each iteration, $i.sComb$, $i.usComb$, $j.fComb$, $j.ufComb$ will be updated. Also, a_{ji} needs to be removed from the associated arc lists. After trip i is satisfied, involved lists will be updated according to the attributes values of trip i and trip j. The $ForwardConstruct(i)$ method constructing get-out arcs for trip i is similar to the $BackwardConstruct(i)$ process.

Discussion

A heuristic constructing a feasible integer solution for TUSO at the network level is proposed. Constraints considered in this method include passenger demands,

permitted type, train unit type compatibility, unit coupling upper bound, and carriage upper bound. Based on the feasible candidates for each trip, solution arcs and flows are constructed one by one. This part discusses some potential advantages of this method. TUSO is hard to solve because of its combinatorial nature. RS-Opt needs a relatively long time to solve some complicated instances or might even fail. This heuristic aims at generating a good-quality feasible solution in a short time to provide a quick reference for the scheduler. Currently, RS-Opt uses two simple methods to generate initial potential paths, and the BB-tree upper bound is unknown until the first integer feasible solution is found. Potentially, we can use the heuristic solution, a feasible integer solution, as a better starting point for RS-Opt and give a better upper bound for the BB tree from the beginning to reduce the computing time of RS-Opt. Besides, it is not very hard to consider the station-level constraints in the heuristic: the station-level resolution method proposed in chapter 6 can be called whenever an arc is constructed, to check if it is compatible with the other constructed arcs. However, the above analysis is still preliminary lack of deep thinking and experiment support due to the time limitation. These could be future research directions.

7.2.2 A multi-graph flow model

The connection-arc graph representation described in Chapter 4 contains a set of nodes and arcs, in which nodes represent trips as well as the source and sink, and arcs represent potential connections between two nodes that can be consecutively served by the same train unit block. The flows on arcs are associated with the quantity of each type of train unit, defined as decision variables. This way of modeling does not consider coupling orders that are affected by specific station layouts. This part presents a multi-graph to capture the coupling order feature of TUSO, defined as permutation-arc multi-graph representation. This presentation shares some similarities with the connection-arc graph representation but takes a relatively different method to define the nodes and arcs.

Permutation-arc multi-graph

We use an example to illustrate the permutation-arc multi-graph. Table 7.4 gives a timetable of five trips, whose permitted train unit types are X and Y. Types X and Y have 100 and 200 passenger seats. In this table, the origin, destination, departure time, and arrival time of each trip are presented. Let T denote all the trips in the given timetable.

Table 7.4: Timetable of 5 trips

Trip	Origin	Destination	Departure time	Arrival time	Demand
i	S1	S3	07:00	10:30	200
j	S2	S3	09:00	10:40	300
l	S3	S4	10:50	12:30	500
m	S4	S1	10:55	12:00	300
n	S4	S5	10:55	12:00	200

Firstly, we define the nodes of the permutation-arc multi-graph representation. Suppose the source node (0) can provide any train unit permutation to any trip, and any train unit permutation can be terminated in the sink node (∞). The nodes for trips in this representation are similar to the candidates defined in section 7.2.1 with the extra consideration of coupling order such that a train unit combination may correspond to multiple train unit permutations. Therefore, there is a set of permutation candidates for each trip, denoted as N_t, $\forall t \in T$. The set of all the permutation candidates of all trips is denoted as $N = \bigcup_{t=1}^{|T|} N_t$. Thus, the node-set in this representation is defined as $\mathcal{N} = N \cup \{0, \infty\}$. Suppose the maximum number of coupled train units is 3. Therefore, trip l in Table 7.4 for instance can be served by 4 train unit permutations: $[XYY]$, $[YYX]$, $[YXY]$, $[YYY]$, which are the permutation candidates for trip l. Let use $\mathcal{K}$ to represent the set of train unit types, and q_i^k, $i \in N$, $k \in \mathcal{K}$ to represent the train unit quantities of type k. A cost $(c_i, i \in N)$ is assigned to each permutation candidate, which is an integrated cost of train unit type preference and seat over-provision. In the solution, only one permutation candidate for each trip can be used.

Each permutation candidate can derive more than one potential sub train unit

permutation, defined as sub-element, to serve another permutation candidate. Let use E_i, $i \in N$ to represent the set of sub-elements derived from permutation candidate i. The arcs are generated based on the three conditions. (i) The destination of i is the same as the origin of j. (ii) The arrival time of i is earlier than the departure time of j, and the time gap must be larger than the basic turnaround time. (iii) i and j must contain some same sub-element. Using a sub-element from i to serve j constructs a permutation-arc connection such that a set of connections from i to j based on using common sub-elements, denoted as A_{ij}. Figure 7.1 gives an example of all the potential permutation-arc connections between i and j with train unit permutations of $[XY]$ and $[XYY]$. Node i can

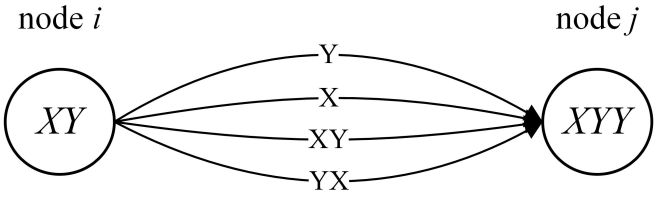


Figure 7.1: Potential permutation-arc connections between two nodes

provide four alternative sub-elements, which are the flows on the corresponding arcs, to serve node j. Let us use A to denote the arc set between any two permutation candidates, defined as permutation arcs. Similar to the connection-arc graph presentation, the arcs get out from the source node are called sign-up arcs, denoted as A_0; the arcs go into the sink node are defined as sign-off arcs, represented as A_∞. The arc set for the permutation-arc multi-graph is a union of permutation arcs, sign-on arcs, and sign-off arcs, denoted by $\mathcal{A} = A \cup A_0 \cup A_\infty$. Let us use $q_a^k, a \in \mathcal{A}, k \in \mathcal{K}$, to represent the number of train units of type $k, k \in \mathcal{K}$. A cost $(c_a, a \in \mathcal{A})$ is assigned to each arc, based on the cost of generating the corresponding sub-element from i to serve j.

Based on the definition of nodes and arcs, Figure 7.2 gives an example DAG of the permutation-arc multi-graph representation according to the trips in Table 7.4. Every trip is represented as a rectangular in which every node is a permutation candidate with a specific number of train units of each type and coupling order. The arcs stand for the potential permutation connections between two

consecutive nodes.

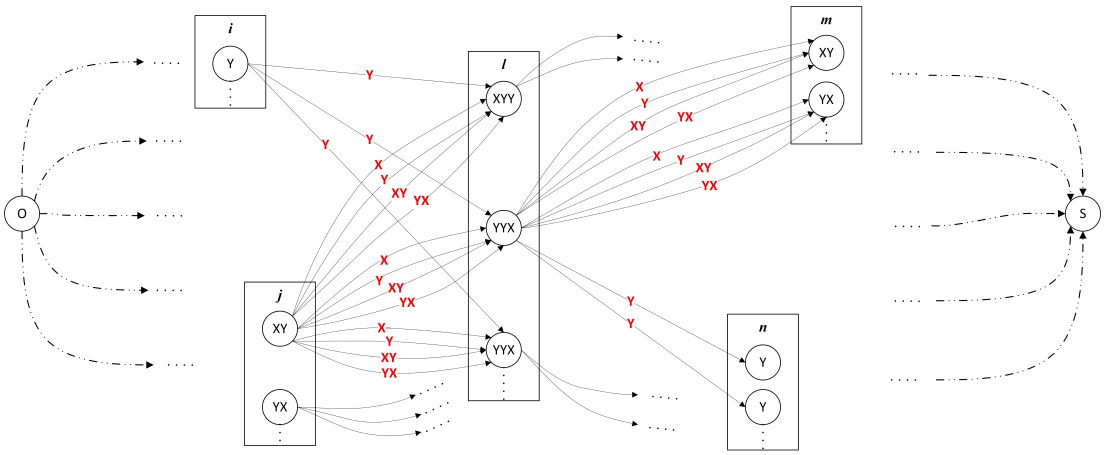


Figure 7.2: Permutation-arc multi-graph representation

A binary linear formulation

Based on the permutation-arc multi-graph representation, the train unit distribution can be captured through a binary linear formulation by assigning each node and arc a binary variable, defined as two types of binary decision variables. (i) Node-selection variables: x_i, $\forall i \in N$, representing if a node is selected. (ii) Arc-selection variables: y_a, $\forall a \in \mathcal{A}$, representing if an arc is selected. Thus, the train unit scheduling with the coupling order problem can be formulated as follows.

$$x_i = \begin{cases} 1, & \text{if node } i \text{ is selected} \\ 0, & \text{otherwise.} \end{cases} \tag{7.2}$$

$$y_a = \begin{cases} 1, & \text{if arc } a \text{ is selected} \\ 0, & \text{otherwise.} \end{cases} \tag{7.3}$$

Expression 7.4 is the objective function of minimizing the general operational costs. The first term is the total number of train units used to cover all trips.

The second and third terms are node costs and arc costs.

$$\min \sum_{a \in A_0} (y_a \cdot \sum_{k \in \mathcal{K}} q_a^k) + \sum_{i \in N} c_i x_i + \sum_{a \in \mathcal{A}} c_a y_a \tag{7.4}$$

Constraints (7.5) ensure that only one permutation candidate can be selected for each trip. Constraints (7.6) ensure only one sub-element between two nodes can be selected.

$$\sum_{i \in N_t} x_i = 1, \forall t \in T \tag{7.5}$$

$$\sum_{a \in A_{ij}} y_a = 1, \forall i \in \mathcal{N}, \forall j \in \mathcal{N} \tag{7.6}$$

Expression (7.7) represents the node flow balance constraints, where A_i^{in} and A_i^{out} are the sets of get-in and get-out arcs of node i. For each permutation candidate, the get-in flow of each train unit type must be equal to that on the get-out arcs.

$$\sum_{a \in A_i^{in}} y_a q_a^k = \sum_{a \in A_i^{out}} y_a q_a^k, \forall k \in \mathcal{K}, \forall i \in N \tag{7.7}$$

A railway network may contain a set of locations where the coupling and decoupling are forbidden. Expression (7.8) and (7.9) are used to present these constraints, where N_{dep}^b and N_{arr}^b are the set of nodes whose corresponding trips are banned on coupling/decoupling operations at the departure station and arrival station respectively.

$$\sum_{a \in A_i^{in}} y_a = x_i, \forall i \in N_{dep}^b \tag{7.8}$$

$$\sum_{a \in A_i^{out}} y_a = x_i, \forall i \in N_{arr}^b \tag{7.9}$$

Constraints (7.10) ensure that the time consumed by coupling/decoupling will not invalid the time allowance (ζ_{ij}) available to operate coupling/decoupling operations of arc $a = (i, j)$.

$$\tau_{dcpl}(\sum_{a \in A_i^{out}} y_a - 1) + \tau_{cpl}(\sum_{a \in A_j^{in}} y_a - 1) \leq \zeta_{ij}, \forall a \in A \tag{7.10}$$

Constraints 7.11 prevent that there will be no conflict combinations of the permutation arcs selected in the final solution. $\bar{A}$ represents a permutation-arc set that is not operable at the station layouts. $\mathcal{Z}_3$ denotes a set of $\bar{A}$.

$$\sum_{a \in \bar{A}} y_a \leq |\bar{A}| - 1, \forall \bar{A} \in \mathcal{Z}_3 \tag{7.11}$$

The domain of the decision variables are given where they are defined.

Discussion

Compared to the connection-arc graph representation, this permutation-arc multi-graph representation has some advantages and disadvantages. Firstly, we talk about the advantages. The permutation-arc formulation can cover the feature of the coupling order that cannot be realized by the connection-arc graph representation. In the permutation-arc multi-graph representation, the permutation on each node and arc are pre-generated in advance. Station layouts can be considered in the arc cost while generating the elements on arcs. Consider the four permutation arcs presented in Figure 7.1 as an example. Suppose the operating platform is dead-end. Table 7.5 shows how to construct the arc costs between two nodes considering platform type and moving directions. Let us use the departure

Table 7.5: Arc costs for the example connection in Figure 7.1

Use depDir as the reference			depDir $\neq$ arrDir	
Tail node i	Gen_cost	Element	Form_cost	Head node j
$[XY]$	Decoupling	$[X]$	Coupling	$[XYY]$
	Decoupling	$[Y]$	Coupling	
	Reversal	$[XY]$	Coupling	
	None	$[YX]$	Reversal+Coupling	

direction of the head node j (depDir in Table 7.5) as a reference. If $[XY]$ only generates $[X]$ or $[Y]$ to form $[XYY]$, both coupling and decoupling operations are in need. On the other hand, if i uses $[XY]$ or $[YX]$ to form $[XYY]$, reversal shunting and coupling operation are necessary. Based on these essential station shunting movements and their time consumptions, arc costs can be assigned ac-

cordingly. The conflict preventing constraints (7.11) are specific on a particular permutation-arc combination, referring to train unit flows. However, the conflict preventing constraints 4.27 based on the connection-arc graph presentation described in Chapter 4 cannot be specific on the flows on a particular arc structure, resulting in over-tight. Constraints 4.33 can be specific on the flow combination, however, they need to introduce another type of binary variable making the problem very hard to solve.

Secondly, we discuss some challenges of the permutation-arc multi-graph representation. Although the candidate permutations for each trip and permutation arcs can be enumerated in advance, they make the original DAG for the latter representation much larger than that of the former for the same timetable. The size of the original DAG is a significant indicator of the problem complexity. That means the latter model may take a much longer time to solve for the same instance. On the other hand, considering station layouts and coupling orders increase the challenges for generating the initial permutation-arc multi-graph. Besides, the latter representation is no longer an ordinary network flow problem due to the irregular "flow balance" constraints because the decision variables x_i are not the real flow amount but only an indicator of selection. This also brings challenges in how to design appropriate solution approaches.

Due to the time limitation, it is difficult to make more insightful comparisons between these two representations without further theoretical analysis and computational experiments. Besides, the special graph structure and problem definition make the permutation-arc multi-graph representation a unique network flow problem. The solution approaches may need special requirements that need further investigation in future research.

7.2.3 Other potential directions

RS-Opt without fixed-charge variables, as a solver for TUSO at the network level, currently can solve small to medium size instances. With fixed-charge variables, it struggles to solve even small size instances. To deal with large scale problem instances, SLIM [26] is developed with three methods of forming compact graphs from the original DAG as inputs for RS-Opt. However, the experimental

results show these methods are not effective to extract good subgraphs. Thus, one future research direction would be to develop more efficient and accurate methods to extract better subgraphs to feed RS-Opt, where slack time and probability may be useful to be considered. Combining the research in this thesis and literature [56, 57, 58, 54, 55, 59], most issues in TUSO have been finalized, including the train unit block for each trip, the coupling order for each multi-unit train unit block, and connections between arrivals and departures, coupling/decoupling operations, and some necessary station movements for example re-platforming. However, the specific shunting plans within each station are still left open to be finalized. In the future, developing the method of how an arrival train unit to be shunted at the specific station infrastructure and to be a departure train unit will be considered. In terms of the objective function evaluation, a potential research direction is how to evaluate the objective function effectiveness if there is no practical exact solver available but only a heuristic solver available. It would be worth considering a dynamic benchmark that can be updated once a better solution is found until no improvement can be achieved.

Bibliography

[1] Transport Statistics Great Britain 2018, UK government. https://www.gov.uk/government/statistics/transport-statistics-great-britain-2018, 2018.

[2] Passenger rail franchising in Great Britain, https://en.wikipedia.org/wiki/Passenger_rail_franchising_in_Great_Britain, 2019.

[3] Communication with Mr. Mark Quinn from First Group, 17.

[4] Tracsis Plc, Tracs-RS. https://tracsisops.com/software/operations-planningsoftware/tracs-scheduling-software/, 2017.

[5] B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations research*, 54(1):99–114, 2006.

[6] A. Alfieri, R. Groot, L. Kroon, and A. Schrijver. Efficient circulation of railway rolling stock. *Transportation Science*, 40(3):378–391, 2006.

[7] A. A. Assad. Models for rail transportation. *Transportation Research Part A: General*, 14(3):205–220, 1980.

[8] M. N. Azaiez and S. S. Al Sharif. A 0-1 goal programming model for nurse scheduling. *Computers & Operations Research*, 32(3):491–507, 2005.

[9] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In *International workshop on hybrid metaheuristics*, pages 108–122. Springer, 2007.

[10] E. Balas and R. Jeroslow. Canonical Cuts on the Unit Hypercube, 1972.

[11] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[12] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, pages 11–18. Morgan Kaufmann Publishers Inc., 2002.

[13] A. Blot, H. H. Hoos, L. Jourdan, M.-É. Kessaci-Marmion, and H. Trautmann. Mo-paramils: A multi-objective automatic algorithm configuration framework. In *International Conference on Learning and Intelligent Optimization*, pages 32–47. Springer, 2016.

[14] E. K. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266–280, 2002.

[15] V. Cacchiani, A. Caprara, G. Maróti, and P. Toth. On integer polytopes with few nonzero vertices. *Operations Research Letters*, 41(1):74–77, 2013.

[16] V. Cacchiani, A. Caprara, and P. Toth. Scheduling extra freight trains on railway networks. *Transportation Research Part B: Methodological*, 44(2):215–231, 2010.

[17] V. Cacchiani, A. Caprara, and P. Toth. Solving a real-world train-unit assignment problem. *Mathematical programming*, 124(1-2):207–231, 2010.

[18] V. Cacchiani, A. Caprara, and P. Toth. A lagrangian heuristic for a train-unit assignment problem. *Discrete Applied Mathematics*, 161(12):1707–1718, 2013.

[19] A. Caprara. Almost 20 years of combinatorial optimization for railway planning: from lagrangian relaxation to column generation. In *OASIcs-OpenAccess Series in Informatics*, volume 14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

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

[21] A. Caprara, L. Kroon, M. Monaci, M. Peeters, and P. Toth. Passenger railway optimization. *Handbooks in operations research and management science*, 14:129–187, 2007.

[22] A. Caprara, L. Kroon, and P. Toth. Optimization problems in passenger railway systems. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[23] A. H. Chow and A. Pavlides. Cost functions and multi-objective timetabling of mixed train services. *Transportation Research Part A: Policy and Practice*, 113:335–356, 2018.

[24] J. L. Cohon. *Multiobjective programming and planning*, volume 140. Courier Corporation, 2004.

[25] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[26] P. Copado-Mendez, Z. Lin, and R. Kwan. Size limited iterative method (slim) for train unit scheduling. In *Proceedings of the 12th Metaheuristics International Conference*. MIC, 2017.

[27] J.-F. Cordeau, P. Toth, and D. Vigo. A survey of optimization models for train routing and scheduling. *Transportation science*, 32(4):380–404, 1998.

[28] J. Desrosiers and M. E. Lübbecke. A primer in column generation. In *Column generation*, pages 1–32. Springer, 2005.

[29] M. Ehrgott. *Multicriteria optimization*, volume 491. Springer Science & Business Media, 2005.

[30] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[31] P.-J. Fioole, L. Kroon, G. Maróti, and A. Schrijver. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174(2):1281–1297, 2006.

[32] R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261–272, 2005.

[33] M. R. Gary and D. S. Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[34] K. Ghoseiri, F. Szidarovszky, and M. J. Asgharpour. A multi-objective train scheduling model and solution. *Transportation research part B: Methodological*, 38(10):927–952, 2004.

[35] M. C. Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, 2004.

[36] J. Haahr and R. M. Lusby. Integrating rolling stock scheduling with train unit shunting. *European Journal of Operational Research*, 259(2):452–468, 2017.

[37] J. T. Haahr, R. M. Lusby, J. Larsen, and D. Pisinger. A branch-and-price framework for railway rolling stock rescheduling during disruptions. 2014.

[38] Y. Haimes. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE transactions on systems, man, and cybernetics*, 1(3):296–297, 1971.

[39] D. Huisman, L. G. Kroon, R. M. Lentink, and M. J. C. M. Vromans. Operations research in passenger railway transportation. *Statistica Neerlandica*, 59(4):467–497, 2005.

[40] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[41] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36:267–306, 2009.

[42] F. Hutter, H. H. Hoos, and T. Stützle. Automatic algorithm configuration based on local search. In *Aaai*, volume 7, pages 1152–1157, 2007.

[43] L. Ingolotti, P. Tormos, A. Lova, F. Barber, M. A. Salido, and M. Abril. A decision support system (dss) for the railway scheduling problem. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 465–474. Springer, 2004.

[44] P. M. Jacobsen and D. Pisinger. Train shunting at a workshop area. *Flexible services and manufacturing journal*, 23(2):156–180, 2011.

[45] L. G. Kroon, R. M. Lentink, and A. Schrijver. Shunting of passenger train units: an integrated approach. *Transportation Science*, 42(4):436–449, 2008.

[46] L. G. Kroon, H. E. Romeijn, and P. J. Zwaneveld. Routing trains through railway stations: complexity issues. *European Journal of Operational Research*, 98(3):485–498, 1997.

[47] R. S. K. Kwan, Z. Lin, P. J. Copado-Mendez, and L. Lei. Multi-commodity flow and station logistics resolution for train unit scheduling. In *Proceedings of the 8th Multidisciplinary International Conference on Scheduling: Theory and Applications*, pages 321–324. MISTA, 2017.

[48] Y.-C. Lai, D.-C. Fan, and K.-L. Huang. Optimizing rolling stock assignment and maintenance plan for passenger railway operations. *Computers & Industrial Engineering*, 85:284–295, 2015.

[49] L. Lei, R. Kwan, Z. Lin, and P. Copado-Mendez. Resolution of station level constraints in train unit scheduling. In *Proceedings of the 14th International Conference on Advanced Systems in Public Transport (CASPT)*. CASPT, 2018.

[50] L. Lei, R. S. Kwan, Z. Lin, and P. J. Copado-Mendez. Station level refinement of train unit network flow schedules. In *Proceedings of the 8th International Conference on Computational Logistics (ICCL 2017)*. ICCL, 2017.

[51] M. Li, S. Yang, and X. Liu. Diversity comparison of pareto front approximations in many-objective optimization. *IEEE Transactions on Cybernetics*, 44(12):2568–2584, 2014.

[52] Z. Lin. *Passenger Train Unit Scheduling Optimisation*. PhD thesis, University of Leeds, 2014.

[53] Z. Lin, E. Barrena, and R. S. Kwan. Train unit scheduling guided by historic capacity provisions and passenger count surveys. *Public Transport*, 9(1-2):137–154, 2017.

[54] Z. Lin, P. J. Copado-Méndez, and R. S. K. Kwan. A branch-and-price-and-cut method for train unit scheduling with complex minimum turnround time requirements. In *14th International Conference on Advanced Systems in Public Transport and TransitData*, 2018.

[55] Z. Lin and R. Kwan. Multicommodity flow problems with commodity compatibility relations. In *ITM Web of Conferences, Volume 14, 2017 The 12th International Conference Applied Mathematical Programming and Modelling–APMOD 2016*, volume 14. EDP Sciences, 2017.

[56] Z. Lin and R. S. Kwan. An integer fixed-charge multicommodity flow (fcmf) model for train unit scheduling. *Electronic Notes in Discrete Mathematics*, 41:165–172, 2013.

[57] Z. Lin and R. S. Kwan. A branch-and-price approach for solving the train unit scheduling problem. *Transportation Research Part B: Methodological*, 94:97–120, 2016.

[58] Z. Lin and R. S. Kwan. Local convex hulls for a special class of integer multicommodity flow problems. *Computational Optimization and Applications*, 64(3):881–919, 2016.

[59] Z. Lin and R. S. Kwan. Redundant coupling/decoupling in train unit scheduling optimization. *Electronic Notes in Discrete Mathematics*, 64:45–54, 2018.

[60] Z. Lin and R. S. K. Kwan. A two-phase approach for real-world train unit scheduling. *Public Transport*, 6(1-2):35–65, 2014.

[61] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.

[62] R. Lusby, J. Larsen, D. Ryan, and M. Ehrgott. Routing trains through railway junctions: a new set-packing approach. *Transportation Science*, 45(2):228–245, 2011.

[63] R. T. Marler and J. S. Arora. The weighted sum method for multi-objective optimization: new insights. *Structural and multidisciplinary optimization*, 41(6):853–862, 2010.

[64] M. Mihaylov, P. Smet, W. Van Den Noortgate, and G. Vanden Berghe. Facilitating the transition from manual to automated nurse rostering. *Health Systems*, 5(2):120–131, 2016.

[65] E. Montero, M.-C. Riff, and N. Rojas-Morales. Tuners review: How crucial are set-up values to find effective parameter values? *Engineering Applications of Artificial Intelligence*, 76:108–118, 2018.

[66] I. Muter, $. $. Birbil, and K. Bülbül. Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. *Mathematical Programming*, pages 1–36, 2013.

[67] Y. Nagasaki and S. Takahashi. Automatic generation of car shunting scheduling in railway car depots. *WIT Transactions on The Built Environment*, 103:585–594, 2008.

[68] V. Pareto. Manuel d'économie politique. *Bull. Amer. Math. Soc*, 18(462-474):3, 1912.

[69] M. Peeters and L. Kroon. Circulation of railway rolling stock: a branch-and-price approach. *Computers {&} operations research*, 35(2):538–556, 2008.

[70] N. J. Rezanova and J. Clausen. *The train driver recovery problem-Solution method and decision support system framework*. 2009.

[71] F. M. Rono. Challenges facing the implementation of decision support systems in loan allocation among commercial banks in kenya. *Unpublished MBA Project, University of Nairobi*, 2010.

[72] T. L. Saaty. Decision making with the analytic hierarchy process. *International journal of services sciences*, 1(1):83–98, 2008.

[73] T. L. Saaty and L. G. Vargas. *Models, methods, concepts & applications of the analytic hierarchy process*, volume 175. Springer Science & Business Media, 2012.

[74] A. Schöbel. Line planning in public transportation: models and methods. *OR spectrum*, 34(3):491–510, 2012.

[75] A. Schöbel. An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research Part C: Emerging Technologies*, 74:348–365, 2017.

[76] A. Schrijver. Minimum circulation of railway stock. *Cwi Quarterly*, 6(3):205–217, 1993.

[77] A. Schrijver, A., Steenbeek. Dienstregeling ontwikkeling voor railned (timetable development for railned). Technical report, Amsterdam, 1994.

[78] N. Tomii, L. J. Zhou, and N. Fukumura. An algorithm for station shunting scheduling problems combining probabilistic local search and pert. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 788–797. Springer, 1999.

[79] Y. Tsuji, M. Kuroda, Y. Kitagawa, and Y. Imoto. Ant colony optimization approach for solving rolling stock planning for passenger trains. In *2012 IEEE/SICE International Symposium on System Integration (SII)*, pages 716–721. IEEE, 2012.

[80] J. Wang, B. Lin, and J. Jin. Optimizing the shunting schedule of electric multiple units depot using an enhanced particle swarm optimization algorithm. *Computational intelligence and neuroscience*, 2016, 2016.

[81] Z. Wang, T. Shi, W. Zhang, and H. Wang. Model and algorithm for optimized formulation of scheduled shunting operation plans of electric multiple units depots. *Journal of the China railway society*, 35(8):1–9, 2013.

[82] R. Watson. The effect of railway privatization on train planning: A case study of the uk. *Transport Reviews*, 21(2):181–193, 2001.

[83] D. E. Wilkins. *Practical planning: extending the classical AI planning paradigm*. Morgan Kaufmann, 2014.

[84] L. A. Wolsey. *Integer programming*. Wiley, 1998.

[85] X. Yang, B. Ning, X. Li, and T. Tang. A two-objective timetable optimization model in subway systems. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1913–1921, 2014.

[86] Q. Zhong, R. M. Lusby, J. Larsen, Y. Zhang, and Q. Peng. Rolling stock scheduling with maintenance requirements at the chinese high-speed railway. *Transportation Research Part B: Methodological*, 126:24–44, 2019.

[87] U. Zimmermann and T. Lindner. Train schedule optimization in public rail transport. In *Mathematics—key technology for the future*, pages 703–716. Springer, 2003.

[88] P. J. Zwaneveld, L. G. Kroon, H. E. Romeijn, M. Salomon, S. Dauzere-Peres, S. P. M. Van Hoesel, and H. W. Ambergen. Routing trains through railway stations: Model formulation and algorithms. *Transportation science*, 30(3):181–194, 1996.

[89] P. J. Zwaneveld, L. G. Kroon, and S. P. M. Van Hoesel. Routing trains through a railway station based on a node packing model. *European Journal of Operational Research*, 128(1):14–33, 2001.