Passenger Train Unit Scheduling Optimisation

by

Zhiyuan Lin

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

The University of Leeds
School of Computing

September 2014

Material from all the published papers below has been reworked and is distributed throughout the thesis. Zhiyuan Lin contributed the major part of the work and the development of which was directed by Raymond Kwan as the supervisor.

Zhiyuan Lin and Raymond S. K. Kwan, "An integer fixed-charge multicommodity (FCMF) model for train unit scheduling", *Electronic Notes in Discrete Mathematics*, 41:165–172, 2013.

Zhiyuan Lin and Raymond S. K. Kwan, "A two-phase approach for real-world train unit scheduling", *Public Transport*, 6(1):35–65, 2014.

Zhiyuan Lin and Raymond S. K. Kwan, "A two-phase approach for real-world train unit scheduling", the 12th Conferences on Advanced Systems for Public Transport, Santiago, Chile, July 2012.

Zhiyuan Lin and Raymond S. K. Kwan, "An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling", International Network Optimization Conference 2013, Tenerife, Spain, May 2013.

Zhiyuan Lin and Raymond S. K. Kwan, "A convex hull technique and ad hoc branching methods in solving an integer multicommodity flow (IMCF) model for train unit scheduling", poster session at 2013 Mixed Integer Programming Workshop, Madison, Wisconsin, USA, July 2013.

Acknowledgements

First I am most grateful to the Engineering and Physical Sciences Research Council (EPSRC) and Tracsis plc for supporting my PhD study in a Dorothy Hodgkin Postgraduate Award scholarship.

I would like to thank my supervisor Dr Raymond Kwan, who has introduced me into the world of scheduling optimisation and has been always supportive and encouraging with inspiring and insightful guidance and discussions. Moreover, I am sincerely grateful for his full support in initiating a firm starting point for my research career. I would also like to thank my examiners, Prof Ken McKinnon from University of Edinburgh and Dr Mark Walkley from our school, for their valuable and helpful advice and suggestions in correcting my thesis.

This research has greatly benefited from collaborations with the train operating companies First ScotRail and Southern Railway, to whom I would like to express my gratitude. In particular, I would like to thank Jerry Farquharson and Mark Quinn from ScotRail for their valuable advice and expertise, as well as their trust in the goal of automated unit diagramming.

Finally, I really need to thank my wife Zihua for all she has done for me.

Zhiyuan Lin

Leeds, 8 February 2015

Abstract

This thesis deals with optimisation approaches for the train unit scheduling problem (TUSP). Given a train operator's fixed timetables and a fleet of train units of different types, the TUSP aims at determining an assignment plan such that each train trip in the timetable is appropriately covered by a single or coupled units, with certain objectives achieved and certain constraints respected. From the perspective of a train unit, scheduling assigns a sequence of trains to it as its daily workload. The TUSP also includes some auxiliary activities such as empty-running generation, coupling/decoupling control, platform assignment, platform/siding/depot capacity control, re-platforming, reverse, shunting movements from/to sidings or depots and unit blockage resolution. It is also relevant with activities like unit overnight balance, maintenance provision and unit rostering. In general, it is a very complex planning process involving various aspects

Current literature on optimisation methods for the TUSP is very scarce, and for those existing ones they are generally unsuitable for the UK railway industry, either due to different problem settings and operational regulations or simplifications on some critical factors in practice. Moreover, there is no known successful commercial software for automatically optimising train unit scheduling in the world as far as the author is aware, in contrast with bus vehicle scheduling, crew scheduling and flight scheduling. This research aims at taking an initial step for filling the above gaps.

A two-level framework for solving the TUSP has been proposed based on the connection-arc graph representation. The network-level as an integer multicommodity flow model captures the essence of the rail network and allocates the optimum amount of train unit resources to each train globally to ensure the overall optimality, and the station-level process (post-processing) resolves the remaining local issues like unit blockage. Several ILP formulations are presented to solve the network-level model. A local convex hull method is particularly used to realise difficult requirements and tighten LP relaxation and some further discussions over this method is also given. Dantzig-Wolfe decomposition is used to convert an arc formulation to a path formulation. A customised branch-and-price solver is designed to solve the path formulation.

Extensive computational experiments have been conducted based on real-world problem instances from ScotRail. The results are satisfied by rail practitioners from ScotRail and are generally competitive or better than the manual ones. Experiments for fine-tuning the branch-and-price solver, solution quality analysis, demand estimation and post-processing have also been carried out and the results are reported.

This research has laid a promising foundation leading to a continuation EPSRC funded project (EP/M007243/1) in collaboration with FirstGroup and Tracsis plc.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The train unit scheduling problem (TUSP)

A train multiple unit (or *train unit* for short) is a self-propelled non-splittable fixed set of train carriages (or *cars* for short), which is the most commonly used passenger rolling stock in the UK and many other European countries such as Germany, France, the Netherlands and Italy. In contrast with a locomotive hauled carriage, a train unit is able to move in both directions on its own. Train units are classified into different types by characteristics such as power source (electric multiple unit (EMU) or diesel multiple unit (DMU)), number of seats, number of cars, routes to run and home-depots. A train unit can be coupled with other units of the same or compatible types. Table 1.1 gives some examples of different types (classes) of train unit currently used by some train operators in the UK.

In the UK rail industry terms, train trips listed in a timetable are referred to as *trains*. Note that here "train" does not mean a physical carrier of rail transport but a timetabled trip service. A train has four basic attributes: origin, destination, departure time and arrival time. Moreover, a train may be described in more details. For instance, a train belongs to a route group in the rail network which may have restrictions like which types of unit can serve it, the maximum number of cars allowed when units are coupled, and so on. Another important attribute of a train is a passenger demand to be satisfied, usually measured in number of seats.

Table 1.1: Examples of different types of train unit in the UK

Type name	Engine	# of cars
Class 171/7	diesel	2
Class 380/0	electric	3
Class 319	electric	4
Class 442	electric	5
Class 395	electric	6
Class 460	electric	8

Given a rail operator's tentative or fixed timetable consisting of trains on one operational day of a week, a fleet of train units of different types and a rail network of routes, stations and infrastructures, the *train unit scheduling problem* (TUSP) refers to the planning of how timetabled trains are covered by a single or coupled train units from the fleet, with certain constraints and objectives. From the perspective of a train unit, scheduling assigns a sequence of trains to it as its daily workload. The TUSP also includes some auxiliary activities, e.g. empty-running generation, coupling/decoupling control, platform assignment, platform/siding/depot capacity control, re-platforming, reverse, shunting movements from/to sidings or depots and unit blockage resolution. In some cases, maintenance planning is also included into the TUSP, e.g. see [15]. In the UK however, maintenance provision can often be achieved later either within the slacks between scheduled train trips or by swapping physical units, thus maintenance planning is conventionally ignored at the train unit scheduling stage.

Alternative names in the literature for similar or relevant problems of TUSP can be *train unit assignment problem* (Cacchiani et al. [15,18]), *train unit/rolling stock circulation problem* (Schrijver [70], Alfieri et al. [3], Peeters et al. [67], etc.), and so on. We give it the name containing "scheduling" to be consistent with other relevant naming patterns as in "vehicle scheduling", "crew scheduling", "driver scheduling" and so forth.

The process of producing train unit schedules is referred to as "train unit diagramming" in the UK rail industry, where a unit diagram is a documentation of the sequence of trains and other auxiliary activities (e.g. coupling/decoupling, shunting, empty-running, maintenance, etc.) for an individual train unit to serve on a day of operation. Figure 1.1 gives an example of a simplified (where all sub-trips within a train have been omitted) train unit diagram that was one of the ScotRail's diagrams used in the 2012–13 period. It can be seen that the unit type of this diagram is a Class 320 EMU with 3 cars. The locations (stations, sidings, depots, etc.) visited by the unit are listed in the first column. Auxiliary activities are shown in the fourth column and the next train that the unit is going to serve after it arrives at a location in the same row is given in the fifth column. Empty-running trains are identified by an initial "5" in their IDs. The sixth column shows the coupling information when this unit is attached to other unit(s) for some trains. Note that this coupled block is ordered (e.g. GW042/GW033 vs. GW033/GW042) and the change often happens after a reverse activity. The last column lists the crews for each section of this unit's workload. Therefore from this diagram, one can see that the Glasgow (GW) based train unit GW033 starts its service at 05:26 at Motherwell T.M.D. depot and after a series of shunting activities it starts its first passenger train service for 2L01LA at 06:07 at Larkhall. Notice that after serving train 2C47LB from Motherwell to Dalmuir, it departs at 14:18 in an empty-running train 5C47LB to a depot, Yoker C.S., for some maintenance work such as CET (Controlled Emission Toilet), cleaning and washing. Then it is attached to another unit GW042 and the coupled unit block continues to cover the

Diagram ID	**GW GW033**
Days	**MSX**
Class	**320 - 3 Car EMU HAGA**
Duration	**14:16**
Distance	**241.1**

Location	**Arr**	**Dep**	**Activity**	**Train**	**Vehicle Cover**	**Crew Cover**
Motherwell T.M.D.		05:26		5L01LA		ML1624D
Motherwell Sig M368	05:29		Revrse			
Haughhead Jn	05:51		Revrse			
Larkhall	06:02	06:07		2L01LA		
Dalmuir	07:02	07:07		5L08LA		ML1604D
Dalmuir Down Siding	07:09		Revrse			
Dalmuir	07:24	07:31		2L08LA		ML1604D,ML7761T
Larkhall	08:30	08:37		2L11LA		ML1604D,ML7761T
Dalmuir	09:32	09:36		5C56LA		YR1716D,YR1721D,YR1771D
Dalmuir Down Siding	09:38		Revrse			
Dalmuir	09:48	09:53		2C56LA		YD7706T,YR1704D,YR1721D,YR1771D
Motherwell	10:46½	10:50		2F29LA		YD7706T,YR1704D,YR1705D,YR1755D
Milngavie	11:52	12:12		2F20LB		YD7706T,YR1704D,YR1705D,YR1755D
Motherwell	13:11	13:16		2C47LB		YD7706T,YR1704D,YR1705D,YR1755D
Dalmuir	14:11	14:18		5C47LB		YR1704D,YR1705D,YR1755D
Yoker C.S.	14:24		CET			
Yoker C.S.			Clean			
Yoker C.S.			Wash			
			Attach			
Yoker C.S.		16:20		5B06LB	GW042/GW033	YR1744D, YR1792D
Hyndland Loop	16:29½		Revrse			
Anderston	16:58		6 cars #			
Anderston		17:04		1B06LB	GW042/GW033	YD7719T, YR1744D, YR1792D
Lanark	17:54	18:23		2C45LC	GW033/GW042	YD7719T, YR1744D, YR1792D
Partick	19:24	19:26		5C45LC	GW033/GW042	YR1744D, YR1792D
Yoker C.S.	19:39		CET			
Yoker C.S.			Clean			
Yoker C.S.			Wash			
Yoker C.S.						

Figure 1.1: Example of a simplified train unit diagram from ScotRail

remaining services in the evening and finally runs back to Yoker C.S. depot at 19:39 where its day's work is ended by CET, cleaning and washing.

In the UK, once the timetables have been finalised on a half-yearly basis, train operators will proceed to schedule the train unit resources, followed by the crew scheduling and rostering process. Optimisation in train unit scheduling is very important as it generally costs hundreds of thousands of pounds per year to lease and maintain a single train unit, while on the other hand train operating companies have to also provide adequate passenger seat capacities and comply with many complex operational rules required by franchise agreements. Therefore train companies are always trying to make use of their train unit resources as efficiently as possible. For some regions where city commuters heavily rely on railway systems, e.g. the Greater London centred area in southern England and the Edinburgh-Glasgow centred area in Scotland, train unit scheduling is also crucial for optimising the distribution of limited fleet resources, by coupling/decoupling units and/or empty-running. Train unit scheduling will also impact upon the subsequent crew planning/rostering stage, which cannot start without the completion of train unit diagramming.

In real-world practice, a train operator often has one to two thousand timetabled trains to be scheduled and the infrastructure layouts, especially at large stations, are often very complex. This makes train unit scheduling a complicated and sophisticated problem hard to solve. There are multiple systems involved, including train timetables, rail networks, operator's fleet, station layout and infrastructure, sidings and depots, engineering and operational rules and so on. As a result of the interaction of the above systems, there can be various constraints that have to be satisfied, including but not limited to, fleet size limit on each unit type, compatibility relations between traction types and routes, time allowance in connecting two trains by the same unit, passenger demand satisfaction, compatibility relations among traction types when coupled, the maximum number of cars for coupled units, locations banned for coupling/decoupling, insertion of empty-running trains and shunting movements, avoidance of unit blockage, unit overnight balance, unit maintenance and cycles. There are also many objectives reflecting the solution's quality to be achieved, including minimising the fleet size and the operational costs (carriage-kilometre, empty-running movements, shunting movements, etc.), removing redundant/excessive coupling/decoupling operations, preferences among the type-route assignment, inserting long gaps suitable for maintenance, avoiding inappropriate gaps after the evening peak, and so on.

Currently in the UK, all train unit diagrams are manually produced based on a trial-and-error and station-by-station basis. Although this method is able to ensure the operability at the station level, it is time-consuming, error-prone and generally lacks global optimality. Therefore, rail practitioners have always been keen to have the unit diagramming process automated.

1.2 Research motivation and theme

1.2.1 Status of train unit scheduling optimisation

Since the 1980s, thanks to the fruitful research results from the disciplines of integer programming, combinatorial optimisation and computer science, as well as the significant improvements in both hardware and software in achieving more and more powerful computational abilities, in many countries, industries such as rail, bus and air transport have entered the age of automatic scheduling—the tedious work in deriving schedules for crews or transport vehicles that were once accomplished by manual processes have been gradually replaced by automatic scheduling systems that can produce results with comparable and often superior qualities than the manual ones. In the UK for instance, the TrainTRACS system is a commercial package of software for train and bus crew scheduling as a result of more than 40 years' research at University of Leeds [31, 32, 50–53, 55, 82, 83]. It has gained wide acceptance by the UK rail industry where about 70% of rail operating companies are now using it on a regular basis and becomes the de facto standard used in rail franchise bidding. The main advantage in such automatic systems is that they have the optimisation ability to seek possibly the best solutions and to enable many what-if analyses, while a human scheduler may often merely give sub-optimal solutions based on trial-and-error manual heuristics focused on each local station or depot.

Train unit scheduling can be regarded as a special kind of vehicle/transport scheduling problem. Although optimisation methods, either exact or heuristic, have been successfully applied in vehicle/transport scheduling problems such as for buses and aeroplanes with the emergence of some pieces of commercial software that are routinely used, they are not commonly used in train unit scheduling yet. Possible reasons of this phenomenon are listed below:

- Train units are heavily constrained on tracks, giving rise to potential problems such as blockage during platform assignment, track section occupation, shunting, etc.
- A train service may be covered by coupled units in response to higher passenger demands and unit resource redistribution across the network.
- Complex rules imposed by rail systems such as unit type compatibility, maximum number of coupled units, locations banned for coupling/decoupling operations, etc.

These features have greatly increased the difficulty in interpreting, modelling and solving the relevant problems. The relatively few previous researches in train unit scheduling have established important contributions in this area. However, they are generally insufficient in handling the problem instances typical in UK train companies, in terms of the capability to cater for practical problem scales, real-world operational restrictions and station-level complexity. Therefore, although train companies have always been eager for such an automatic optimising system in train unit scheduling, as far as the author is aware, no such system exists that is able to produce high-quality solutions applicable to real-life instances and being routinely used by train operators.

1.2.2 Solution approaches

Many real-world problems, including public transport scheduling, can be modelled as a network flow problem, where flows are to be sent through a network formed by nodes and arcs. When flows are heterogeneous, the problem becomes a multicommodity (network) flow problem. When the flow amounts have to be integers and there are more than one commodity, the corresponding integer multicommodity flow problem is known to be NP-hard [35]. This problem can be solved by ILP using approaches like branch-and-price [8].

In the recent work by Cacchiani et al. [15], it has been proved that the train unit assignment problem, which can be formulated as an integer multicommodity flow problem, is NP-hard when there is more than one type of unit in the fleet. The train unit assignment problem is similar but has fewer real-world requirements than the train unit scheduling problem to be studied in this thesis. Therefore, the train unit scheduling problem, whose network-level part can be formulated as an integer fixed-charge multicommodity problem, is expected to be at least NP-hard.

Exact methods for scheduling based on integer linear programming (ILP) can often derive much better solutions than heuristics. Operational research with relevant techniques such as column generation and branch-and-bound were established and developed as early as in the 1950s and 1960s. Nevertheless, dated back to 1960s–1970s, heuristic algorithms were dominant in scheduling models and systems, because the computers' capabilities were inadequate at that time for the problem sizes in practice if an ILP-based method was applied. From the 1980s when powerful LP and ILP solvers were available, mainly as a result of the significant upgrades in computers' abilities, exact methods began to be increasingly used for various scheduling problems. The report by Wren [82] on forty years' research experience on vehicle and driver scheduling at University of Leeds gives good examples on the evolution of solution approaches used in this area.

To solve an integer linear programme efficiently, a prerequisite should be a powerful LP solver, which is often based on the simplex method. Apart from that, supposing a branch-and-bound based method is used for searching the integer solutions for a minimisation ILP, it is important to strengthen the lower bound given by LP or Lagrangian relaxation through some methods, e.g. valid inequalities. Moreover, well-tuned branching strategies taking advantage of the problem's specific structures would be keys to success in finding integer solutions within practical time.

Currently, it is widely recognised that exact methods however still have the disadvantage that they are usually difficult to be scaled up for real-world cases whose typical sizes are not small. Therefore it is a common practice to consider hybridised methods combining exact and heuristic approaches together. Such hybridisation can often make a huge step change to the system in terms of the problem scales the system can handle. It is also a challenging issue to ensure the solution quality not to be seriously undermined as a result of the hybridisation with heuristics.

1.2.3 Research theme

This research seeks an approach for automatic train unit scheduling that can produce high-quality solutions within a reasonable time especially for problem instances typical for UK train operating companies. There is relatively little previous research in the literature and none of it is suitable for the specific cases in the UK, making the work challenging. Moreover, regarding the solution approaches, it is either the case that exact methods are used to solve models designed for limited problem instances and operational rules that are not commonly seen in other countries or researches have developed models that can be applied to a broader range of instances but the solution approaches are all based on heuristics due to the models' difficulty. This research aims at developing a widely applicable model and solving it mainly by exact methods to ensure the solution quality, which is vitally important. Due to the complexity and sophistication of the TUSP, this research will focus mainly on the following aspects.

1.2.3.1 Problem modelling and ILP formulations

Requirements and objectives of the train unit scheduling problem will be first described in detail. Models will be then constructed to reflect the real-world scenario as much as possible. The focus will be on how to interpret the real-world problem into an appropriate model that can capture the essence of the problem as much as possible while the resulting mathematical formulations are also not too difficult to solve for medium sized instances. A framework based on directed acyclic graphs (DAG) will be discussed. The framework regards the TUSP in a global network-level perspective such that unit resources can be optimally distributed across the entire rail network with the fleet size and the operational costs minimised, which are the most crucial objectives in the TUSP.

Ideally, the above global DAG framework can include everything required by the TUSP. However, in practice, there are some requirements that have to be or can possibly be handled locally in a station-by-station manner. This is because otherwise the global network-level model would be too huge to solve based on the current computational ability, and/or it is found that some requirements are in fact not necessary to be included in the global DAG frameworks as it will be sufficient to relax them to local processing. Based on this principle, the problem will be modelled into a two-phase structure consisting of a global network-level DAG framework and a set of local station-level post-processing tasks. The DAG framework will be able to handle most requirements including the most important ones, and the local post-processing tasks will resolve the remaining issues left by the global model.

The specific modelling strategy of the DAG framework can vary each with its own pros and cons. Different modelling strategies will be discussed and two of them, a connection-arc based DAG and a combination-arc based DAG, will be proposed and formally described. The connection-arc DAG framework and its resulting mathematical formulations are the major targets to be studied in this thesis. Regarding the above DAG framework, several ILP formulations will be proposed. There will be an integer fixed-charge multicommodity flow formulation that tries to include as many requirements as possible into the network-level model. In addition, another integer multicommodity flow formulation that has further relaxed some requirements to the local problems will also be presented, which is more efficient in solving larger scaled problems without undermining satisfying the requirements passed to the local problems. Each of the above formulation has an arc-based and a corresponding path-based formulation variant, linked by Dantzig-Wolfe decomposition.

While the global network-level framework would be the main focus of the thesis, the less important local resolution work will be handled in the experiments using the interactive software TRACS-RS [73] for rolling stock scheduling produced by Tracsis plc, a railway software development and consultancy company spun off from University of Leeds.

1.2.3.2 Solution approaches

Solution approaches with respect to the ILP formulation will be discussed and proposed in detail. An exact method is to be used. Important aspects include:

- Strengthening the LP relaxation by computing explicit convex hulls associated with trains. Additional discussions on the polyhedral combinatoric characteristics and computational algorithms on the train convex hulls will be given in the appendices.
- The design of a customised branch-and-price ILP solver: combine column generation with branch-and-bound taking advantage of problem features.
- Branching strategies: multiple customised branching rules and efficient adaptive node selection methods.

1.2.3.3 Case studies and computational experiments

The feasibility and the applicability of the research to real-world problems are of great importance. The above models and solution methods will be tested using real problem instances of small (100–200 trains) to medium (500 trains) sizes from some train operating companies and the results will be reported and analysed.

1.3 Organisation of the thesis

Firstly, Chapter 2 will review the relevant literature. There will be a brief overview on previous researches relevant to train unit scheduling, locomotive/bus scheduling and train unit shunting. Next an introduction on relevant network optimisation problems will be presented, including the multicommodity flow problem and the more difficult fixed-charge multicommodity flow problem. We will then take a look at some solution techniques for solving LPs/ILPs, such as column generation/branch-and-price, especially for relevant scheduling problems.

In Chapter 3, the train unit scheduling problem will be formally described in detail, including involved entities, restrictive requirements and objectives. In particular, it will be shown that there are several complex requirements typical in the UK rail industry that cannot be solved from the research in previous literature. In addition, there are some objectives that are crucial for a successful schedule but are not reflected in the previous researches. Then, some network-level DAG frameworks in handling the overall TUSP will be introduced. They include the most critical features and requirements of the TUSP at a high-level over the entire rail network. The other lower-level assignments that finally determine the finer detailed operational plans will be handled by post-processing at each location to resolve problems left out by the network-level model, such as unit blockage. The post-processing will also partly decide some auxiliary activities like shunting movements within platforms/sidings/depots. The rationale and validity behind this two-phase framework will be explained and discussed. Finally a connection-arc formulation constructed with more intuitive constraint realisation strategies will be proposed.

In Chapter 4 several further ILP formulations corresponding to the network-level DAG framework will be proposed. First, issues on combination-specific coupling upper bounds will be discussed with two formulations in handling them. Regarding the practical conditions that the constraints at the station-level are unrealistic to be combined into the main model, a reduced arc formulation will be presented to leave some requirements to a post-processing phase. Then Dantzig-Wolfe decomposition will be applied to convert the arc-based formulations to path-based formulations to be more efficiently solved by column generation based methods such as branch-and-price. Relevant theoretical issues on the above Dantzig-Wolfe decomposition process will also be discussed in detail.

Chapter 5 will discuss how to solve the ILP formulations given in Chapter 4. A customised branch-and-price ILP solver will be presented. Relevant column control strategies and parameter settings for the solver will be given in detail. This chapter is ended by presenting the branching strategies embedded into the solver including: (i) a customised multi-rule branching variable/target selection system that can effectively divide the solution space based on the problem's features and (ii) a heuristic node selection system as a combination of depth-first, best-first and breadth-first that uses adaptive tree searching strategies according to the current status to find the optimal solutions quickly.

Chapter 6 reports the computational experiments on real-world instances from two UK train operators solved by the ILP formulations of the connection-arc based models. Analyses on the results will also be given. The feedback from ScotRail shows that the rail practitioners are satisfied with the operability and quality of our schedules.

Chapter 7 generalises a local convex hull method mainly used to strengthen the lower bounds in branch-and-bound based methods and discusses its feasibility range when applied to high dimensional cases, especially within the context of rolling stock scheduling. The structure of the local convex hull will be studied in separating it into a main hull, an up hull and a down hull, if the related points satisfy certain conditions. It will be shown that the main hull has at most two nonzero facets. The number of points outside the main hull will also be investigated mainly on an empirical basis. The computational experiments show that for the problem instances of ordinary train operating companies, even in the most difficult cases, the numbers of points will generally be within the range of QuickHull's [6, 66] capability. Based on the above analysis, a slightly modified QuickHull method will be proposed to compute the local convex hulls taking advantage of their structures.

Chapter 8 presents some investigation directions apart from the main line of the research, including the basic idea of the combination-arc graph representation and a multi-dimensional matching model for the station-level post-processing developed at the early stage of the research.

The thesis will be concluded in Chapter 9 which also proposes future research directions. In particular, inspired from the PowerSolver method [53] used in TrainTRACS, the focus will be given to a promising hybridisation method to make a step change for large problem sizes (1000 or more trains) . Future research topics will also include powerful solution methods for the fixed-charge multicommodity flow ILP such as branch-and-price-and-cut and automated post-processing methods for the station-level tasks.

Chapter 2

Literature Review

This chapter gives a review on previous research relevant to the train unit scheduling problem studied in this thesis, including the circulation, assignment and shunting problems for train units. Then the multicommodity flow problem, often used as a modelling tool for many scheduling problems, as well as its fixed-charge variant, will be mentioned next. This chapter is ended by reviewing some techniques used in solving integer linear programmes, especially for those arising in transport scheduling problems closely related to the approaches proposed or used in this research.

For surveys on operational research methods applied in passenger railway planning and operations, see Huisman et al. [44], Caprara et al. [19, 20], Caprara [21].

2.1 Planning the train unit resources

Train units are widely used in many European countries and the main sources of relevant literature on the train unit scheduling problem are from the Netherlands, Italy and other countries in Europe. It should be noted that the specific problem definitions for the task of assigning train units to a given timetable differ from place to place, in terms of inputs, outputs, requirements, restrictions, network structures, timetable patterns, and other particular issues. This is mainly due to the various planning rules and conventions in different countries. Moreover, different names are used for the phase of planning train unit resources after the timetable is given, such as circulation, assignment and scheduling.

As for earlier research particularly on allocating train units, Ben-Khedher et al. [9] consider the problem of allocating train units of a unique type to the French high-speed trains. The proposed train unit allocation framework is associated with a capacity adjustment model interacting with a seat reservation system. The objective is to maximise the expected profit and the problem is solved by stochastic optimisation, branch-and-bound and column generation. Since the units are all identical, there is no need to consider additional issues that may complicate the problem such as unit type-type compatibility and unit permutation.

In the rest of § 2.1, the researches from the two groups that have mainly focused on train unit resource planning, namely the Dutch group in collaboration with Nederlandse Spoorwegen Reizigers (NSR) and the DEIS group at University of Bologna, will be surveyed.

2.1.1 The train unit circulation problem

The problem of *train unit circulation* has been studied by researchers from the Netherlands, mainly based on one or more lines of the rail operator Nederlandse Spoorwegen Reizigers (NSR). The results are fruitful and have been successfully applied in real-world cases in NSR.

Rolling stock planning at NSR is performed in a two-level/four-stage way (Huisman et al. [44]). The two levels are a central planning level that assigns units to timetabled train services and a local planning level that specifies shunting and platform assignment details at each station. The four stages are strategic planning (10–20 years ahead), tactical planning (about 1 year ahead [44], or more flexibly 2–12 months ahead [61]), operational planning (about 2 months ahead) and short-term planning (daily). The strategic stage only takes place at the central planning level while all the other three stages involve both the central and the local levels.

The rail network of NSR has been decomposed into lines each containing relatively independent and well-patterned train services. The main duty of the central planning at the tactical stage is the *rolling stock allocation problem* that decides the number and types of train unit for each line. Abbink et al. propose a model in [1] for the rolling stock allocation problem, which determines how many train units of each type for each line by minimising the seat shortage for all the 8 o'clock trains. Other issues like fleet size limit and type-type compatibility have also been included into the rolling stock allocation model. Note that although defining and solving rolling stock planning in a different way, the planning horizon for the TUSP in the UK railway industry studied in this research will be mostly close to the tactical planning stage at NSR.

Next, the central planning at the operational planning stage at NSR involves the train unit circulation problem that assigns train units decided by the prior rolling stock allocation to timetabled trains on a line-by-line basis. The objectives of the train unit circulation problem may vary in different literature, such as to minimise fleet size, operational costs, seat shortages, component changes (coupling/decoupling activities). As for the constraints, besides the basic requirements such as time allowances, shunting buffering consideration, passenger demand satisfaction and coupling upper bound, the unit permutation restrictions for yielding feasible coupling/decoupling operations at stations are also imposed, which has increased the problem's difficulty. Some variants in the literature have included the cycling issues as well, i.e. how the unit inventory balance at each location is kept by overnight berthing plans. In [44] the cycling issues are said to be performed as a second step after the rolling stock circulation for each weekday is determined, while in [61,62,67] cycling is partly considered together with the main model. During unit circulation planning at the central level, shunting assignments and maintenance requirements are often left to the local planning level at the same or subsequent stages, e.g. [49,62].

Finally at the short-term planning stage in the central planning, some final modifications on rolling stock circulation have to be carried out due to reasons such as extra trains for sports events and infrastructure maintenance work.

Although sharing a common feature as being a practice to assign train units to a given timetable, the problem definitions and settings for the above train unit circulation problem are different from the TUSP, which will be formally defined in Chapter 3.

At NSR the train unit circulation is solved on a line-by-line basis where the train units needed for each line has already been assigned by the prior rolling stock assignment problem. In the TUSP, neither the line decomposition nor the rolling stock assignment is processed as explicit earlier stages. A timetabled train service at NSR (which will be referred to as an NSR-train here for clarity) can have its unit components changed en route if needed and within each train journey its component non-changeable sections are called trips. Therefore an NSR-train is formed by a sequence of trips. For each trip, a desired passenger demand is to be satisfied and a coupling upper bound in number of cars is imposed. This is natural since coupling/decoupling is allowed at the start/end of trips. In the TUSP, however, the unit configuration is unchangeable at the level of trains, i.e. coupling/decoupling is only allowed during the turnround time between two trains but is totally forbidden en route of a train journey, and passenger demands and coupling/upper bounds are thus specified on individual trains.

Another major difference between the unit circulation problem and the TUSP is the fact that for each NSR-train, its unique predecessor train and successor train are given in advance; for each trip, its unique predecessor trip and successor trip are also prescribed a priori. By predecessor and successor, it means that at least one unit should be operated according to the preset connection between the two relevant trains or trips. Often this pre-sequencing is done locally at each station based on a first-in-first-out (FIFO) basis [29,61]; the fact that trains in timetables are generally well-patterned (departure and arrival times are regulated every hour or periodically) at NSR also makes the FIFO pre-sequencing practically workable. Moreover, at least one unit should follow the complete journey of its belonged NSR-train. This is referred to as the "continuity requirement" [29, 61] and is mainly for passenger convenience as otherwise one has to change the carriages if s/he would like to take the entire train journey. Note that the pre-sequencing and the continuity requirement together will impose a backbone structure in how train units will flow the rail network, leaving relatively less flexibility for unit resource distribution. On the other hand, in problem settings of the TUSP, the connection possibilities among trains are left flexible such that no prescribed connection is imposed. This is also the case for research on the train unit assignment problem to be introduced in § 2.1.2.

Finally when coupling/decoupling is concerned, NSR often uses the rules such that only one of them can be performed during a connection between trips or trains, and a unit that is coupled should be at the front of the block and a decoupled unit should be from the rear of the block. The former rule is due to the need to validate time allowances and the latter rule is to ensure that coupling/decoupling operations are always correct regarding unit permutations. Such simplifications by narrowing down the possible ways in coupling/decoupling operations can effectively guarantee that no violations can occur. There is no such explicit restrictions for the TUSP and all validity verifications for coupling/decoupling operations either due to time allowances, unit permutations or other factors should be imposed either by LP constraints or by local post-processing. This will make sure that all possible operations can take place but at the price of increasing the problem's difficulty.

Relevant literature on the train unit circulation problem is discussed below.

Schrijver [70] first proposes an integer multicommodity flow model and solves it as an ILP for the train unit circulation problem for a single line on a single day with two compatible unit types for NSR. In the model the nodes represent the arrival and departure events at stations and the arcs represent the trips. Flows of different commodities stand for train units of different types. The framework of this model (flow graph or time-space graph) has been kept in most subsequent research on rolling stock circulation by NSR. The objective is to minimise the fleet size. As there are at most two compatible types of unit, the weak passenger demand satisfaction and coupling upper bound knapsack constraints are replaced by explicitly finding their relevant convex hulls in $\mathbb{R}^2$. Some local issues like time allowances and permutation restrictions regarding coupling/decoupling are ignored.

Alfieri et al. [3] consider a problem scenario for a single line and a single day with two compatible types of unit for NSR. The same flow graph framework is used. The objective is to minimise the combination of fixed and variable costs of train units. Two models have been proposed. The first one ignores the unit permutation issue and satisfies the passenger demands and coupling upper bounds directly by constraints in conjunction with a similar local convex hull method as in [70] to strengthen the lower bounds. The second model can handle the unit permutation issue by introducing a transition graph concept and new decision variables representing unit compositions per trip. These new composition variables also have included the conditions to comply with passenger demands and coupling upper bounds implicitly without any explicit constraint formulation. Issues like cycling, shunting conflicts and maintenance are left to other planning stages. Also a pre-processing approach to reduce the sizes of the transition graphs is developed. The model has been applied to the line 3000 of NSR with 12 NSR-trains.

Peeters and Kroon [67] present a model for a train unit circulation problem for two lines of NSR with 15 NSR-trains (182 trips) and 12 NSR-trains (115 trips) respectively. Also based on the framework of a time-space graph, the model uses the aforementioned transition graph where unit compositions at each connection between the trips are explicitly enumerated and the composition transition possibilities are represented by associated decision variables. Moreover, variables and constraints for describing increment or decrement of unit inventories at each station are added to connect all transition graphs together such that the feasibility of the circulation plan is guaranteed by no negative inventories. For solving the ILP formulation for the model, Dantzig-Wolfe decomposition is used where, as the order of the joint flows are considered, the master problem is decomposed with respect to NSR-trains rather than sub-networks per commodity (unit type). Then branch-and-price is applied to get the integer solution. This approach is able to handle real-world instances of NSR in a short time after fine-tuning in the branch-and-price solver.

The models in [3, 67] have employed the idea of transition graph that strongly uses the fact that all NSR-trains are running on lines such that the predecessor and successor of a train or trip are unique. Sometimes, an NSR-train will be associated with a route with a topology other than a "line", but of a "Y" or an "X" shape, indicating that a train/trip can have two predecessors and/or two successors and thus relevant coupling/decoupling operations have to be imposed (however in the prior literature with trains running on lines, no coupling/decoupling is imposed because of the uniquely assigned predecessors/successors). This is called combining/splitting of a train and has been catered for in a customised way by a model proposed by Fioole and Kroon [29]. Real-world problems in the Noord-Oost lines with 665 trips have been tested and several fine-tuning methods are used to speed up the solution process. Near-optimal solutions for the tested instances can be found in 1900–6400 seconds.

Some detailed information and problem solution approaches regarding rolling stock planning at NSR can be found in Maróti's PhD thesis [61].

The major models and solution methods proposed by the Dutch group are tailored specially for the operator NSR according to its planning and operating characteristics. For instance, the rolling stock allocation stage will decompose the rail network into lines with generally compatible unit types such that in the subsequent rolling stock circulation stage the problem scales will be effectively reduced and no type compatibility issue will arise. Moreover, NSR timetables at each line often have well-patterned train services, which makes the pre-sequencing by FIFO workable in practice. The continuity requirement asking at least one unit to following a train's entire route and the fact that predecessors and successors are unique make it possible to employ the transition graph concept that has successfully solved the problem of multicommodity flows involving unit positions. Finally the frequent component change during a train's journey leaves little time for complicated coupling/decoupling activities, making it reasonable and effective to restrict the coupling/decoupling possibilities. This restriction has also eased the difficulty in constructing a transition graph for each train.

Due to the above analysis, the researches from the Dutch group may not be suitable for ScotRail and Southern Railway. It may also be the case for other UK train operators. Taking ScotRail for example, its rail network is not decomposed into detailed lines each with a set of dedicated units to serve and a set of trains repeating the same pattern. Instead, the entire network is first divided into three areas which are relatively independent. Within each area, lots of routes can be further distinguished which may have some crude similarity with the lines at NSR. However no unit type is assigned specifically to any route. In fact it can be the case that a dozen routes share the same group of types of unit. In addition, even for an individual unit of a certain type, it is not meant to be restricted to only cover the trains on a single route. Unit resource interaction among different routes are common. This makes it less meaningful to schedule train units in a line-by-line way at ScotRail and a more applicable approach would be to schedule within each area including all of its routes. Moreover, the pre-sequencing phase required in the models of the Dutch group's researches may not befit ScotRail and Southern Railway whose train services are not as regularly-patterned as the ones at NSR. In particular, after some tests based on ScotRail's datasets, it is observed that a deterministic rule such as the FIFO is often not the best matching plan when one tries to link arrivals to departures in a station's view. Since the pre-sequencing will impose a backbone that must be used in the entire schedule, if one cannot even find any reliable prescribed sequences beforehand in the first place, then it would be better to just let all possible connections be flexible and allow the subsequent optimisation approaches decide the best linking, which would be more suitable for ScotRail and Southern Railway.

2.1.2 The train unit assignment problem

Another kind of train unit resource planning problem, namely the *train unit assignment problem* (TUAP), has been studied by the Italian group DEIS at University of Bologna. The TUAP shares very similar definitions and settings as the train unit scheduling problem that will be described in Chapter 3, especially in the sense that no trains are pre-sequenced in advance and no coupling/decoupling is allowed en route, so differing from the train unit circulation problems at NSR. Cacchiani et al. [15] first present an integer multicommodity flow model for the train unit assignment problem. A major characteristic of this model is that it can handle problems with a fairly large number of distinct train unit types, e.g. 10. For each train, there is a passenger demand in the number of seats and a coupling upper bound on the number of units to be satisfied. Besides the basic requirements, the model can also include additional constraints for maintenance and overnight balance. Other real-world constraints like time allowances involving coupling/decoupling, locations banned for coupling/decoupling, unit type compatibility relations and local shunting planning are not considered.

The model is based on a directed acyclic graph where the nodes represent trains and the arcs represent connection possibilities. Each commodity stands for a unit type and its flow amount at a node gives the number of units of that type to be used for the train of that node. More flexible coupling/decoupling activities can be achieved by multiple flows joining at the same node. A similar DAG framework is used in the research of this thesis.

A path formulation ILP based on the DAG is used in solving the TUAP in [15]. Taking advantage of the instances' feature that no more than two units can be coupled, each weak knapsack constraint to satisfy the demand requirement at a train is replaced by a constraint of its dominant that can be described explicitly. The details of this strengthening method can be found in [14]. An LP-based heuristic is designed for finding the integer solutions. Without using a branch-and-bound tree, this heuristic first solves the LP relaxation to get the dual variables. Then instead of computing the paths to be added for each type directly from the subproblem by the shortest path method, for each type multiple columns are constructed based on the information given by the dual variables and further processed by a subsequent refinement phase to prevent infeasibility. If any of the above constructed paths are shown to be promising, then they are added into the restricted master problem; otherwise some path variables will be fixed to integers according to some rules and the restricted master problem will be re-optimised. The above process will be repeated until certain termination criteria based on solution quality are satisfied or infeasibility is claimed. This heuristic is reported to be crucial in finding feasible solutions in many of the tested instances as the problem is regarded as very difficult.

The above model for the train unit assignment problem has been applied to real-world instances of a regional train operator in Italy, with fleets of up to 10 distinct unit types and timetables of 528–660 trains. The heuristic is coded by C and the LP relaxation is solved by a commercial solver. The customised heuristic is able to find solutions 10–20% better than the manual solutions in practice. The solution time ranges from 1878–1932 seconds without the additional maintenance and overnight balance constraints, and 5897–14105 seconds with those constraints.

Cacchiani et al. [18] later develop a fast and effective heuristic method based on Lagrangian relaxation for the same problem as in [15]. The main aim is to find a suboptimal solution in a very short time such that it can be embedded into real-time operations in conjunction with other planing activities like timetabling. An alternative ILP model is used and the relaxed solution is found by solving a sequence of assignment problems per type. The heuristics consist of a construction phase of solutions and a local search phase for improvement. Computational experiments have been conducted for the same instances as in [15] and some larger "realistic" instances up to around 1000 trains artificially constructed by combining the existing instances. Comparisons with the method given in [15] have been made. Without setting a time limit, the new heuristic finds feasible solutions much faster with worse qualities and poorer lower bounds; while within a time limit of a few minutes, the new heuristic has shown much better performance than [15] in solution quality and the ability to find feasible solutions. The same authors have proposed another fast heuristic for the same train unit assignment problem in [16], whose heuristic component has a similar structure as [18], i.e. a construction phase and a local search phase. The lower bound is given by solving an ILP that only considers a subset of "simultaneous" peak time trains where any two of them cannot be covered by the same unit. This ILP (a cardinality-constrained bounded subset sum problem [63]) is relatively easy to solve by enumeration as the number of unit types and fleet size are small enough. Real-world instances have been tested and comparisons with the approaches in [15, 18] have been made.

Detailed description and other work regarding the research on the train unit assignment problem at the Italian group can be found in Cacchiani's PhD thesis [12], as well as in [13, 17].

The problem settings and models from the DEIS group share some common features with the TUSP to be studied in this research, mainly in the fact that there is no network decomposition and pre-sequencing beforehand. However, the models proposed by the DEIS group cannot be directly applied to the railway system in the UK due to the following reasons. First, there is no consideration on banned/restricted locations for coupling/decoupling operations. Second, there is no concern on the unit type compatibility issues. Moreover, their coupling upper bound is set to be 2 units for all trains while the scenarios for the UK operators are more complex, where the bounds have multiple restrictive factors and can be combination-specific. In addition, there is no consideration on the time allowances involving coupling/decoupling. Finally, there is no consideration on the avoidance of excessive and/or redundant coupling/decoupling operations or unit permutation/blockage issues at the station level. The solution methods proposed by the DEIS group used for solving the relevant ILP models for the TUAP are mainly based on heuristics. In this research exact methods based on customised branch-and-price will be designed and tested.

2.1.3 The train unit shunting problem

Train unit shunting refers to the detailed local planning on how train units move within rail infrastructures such as platforms and sidings to avoid blockage and reduce operational costs, as well as to reposition units for subsequent assignments. Usually shunting assignment is carried out after some higher-level planning over the rail network has been done which provides a rough input for the shunting to be further specified and modified.

The Dutch group aforementioned has done some research on train unit shunting based on instances from NSR, where the problem setting is a bit different from what will be described in § 3.1.7. In these researches, the focus is on linking the arrival and departure units that have to be shunted to shunt yards (shunt tracks, sidings) with connection time gaps of medium lengths while the matching for the arrival and departure units of "through trains" with short time gaps are regarded to be already given in the previous train unit circulation stage as in § 2.1.1.

Freling et al. [33] propose a two-phase approach to solve the train unit shunting problem at a station within a 24-hour time range. The main focus is on shunting train units that are temporarily unused between two trips to sidings of the station. The first phase will do a rough matching between the arrival and departure units at the station. The second phase will determine detailed shunting assignment plan at each siding for each unit. The approach is based on a set-partitioning integer linear programming model solved by root-only column generation plus branch-and-bound 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 prevents capacity overflow and unit blockage at all sidings, and also minimises the number of coupling/decoupling operations. Instances for the station with up to 80 train units to be parked at 19 sidings have been tested where near-optimal solutions can be found with small gaps to global optimality.

Kroon et al. [49] later propose an integrated approach with four models for the same train unit shunting problem as in [33] where unit matching and shunting assignment are processed together, hence the global optimality is guaranteed. The paper first considers the LIFO shunt tracks with a basic model (Model 1) that contains an excessive number of constraints for eliminating crossing possibilities. To overcome this, a second model (Model 2) is given where the crossing constraints are replaced by aggregating them into clique constraints. Then, a third model (Model 2a) is presented to implicitly describe the clique inequalities in an efficient way by additional variables and employing some ideas from comparability graphs [37]. A Model 3 is also proposed including the "virtual tracks" such that the size of the problem can be further reduced. Finally the models for FILO shunt tracks have been extended to Model 4 to suit the scenarios involving free tracks. Computational experiments on two stations of the NSR network have been done. One station has 15 free shunt tracks and 4 LIFO shunt tracks with 50–125 non-through trains to be parked and the other has 11 free shunt tracks and 2 LIFO shunt tracks with 30–45 non-through trains. 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.

Again since the planning methods and problem settings in NSR are different from those in the UK rail industry, particularly as the way how train units are assigned to timetabled trains are different as shown in § 2.1.1, the approaches in resolving the local shunting problems designed specifically for NSR from the Dutch group cannot be applied directly to the UK train operators.

2.2 Other kinds of vehicle scheduling problems

2.2.1 Locomotive scheduling

There have been some researches on locomotive scheduling, where the most commonly seen task is to assign locomotives and/or cars to timetabled trains. The requirements and problem settings for locomotive scheduling are quite different from a standard train unit scheduling problem. First, shunting restrictions for locomotive-hauled carriages (cars) are different from the ones for train units, e.g. the shunting operations for train units can be more flexible due to their bidirectional nature and shorter shunting time. Second, the orders and types of train unit can have significant impact on the schedule, especially when coupling/decoupling is involved, while they are less important for locomotives scheduling. Third, the operational regularities and the timetable patterns of networks served by train units are also different from those for locomotives and cars.

The scheduling research at University of Leeds was started in early 1960s on locomotive scheduling. This problem can be formulated as a standard assignment problem but the problem size was too large for the computers at that time to be solved by the Hungarian algorithm. A heuristic method was designed by Wolfenden and Wren [80] that can resize the problem scale small enough such that the Hungarian algorithm would be repeatedly used for updated problem instances that converge to the optimum. This method was applied to real-world instances of British Railways and became operational in May 1963, reducing the number of locomotives from 15 to 12 and giving 28% less empty-running. It is believed to be the world's first computer-produced working schedule [82].

In other countries, Ziarati et al. [84] propose a model to assign locomotives to trains such that sufficient power can be used to pull the cars. It is solved by a "branch-first, cut-second" approach, where valid inequalities are inserted to strengthen the LP lower bound and to prevent the case that more than 3 types of locomotives are assigned to the same train leg. Computational experiments have been conducted for real-world instances from Canadian National Railway.

Cordeau et al. [24–26] present some models on the locomotive and car assignment problem, employing solution methods such as Benders decomposition and heuristic branch-and-bound based on column generation. The concept of "train consist" is used which is defined as a group of compatible locomotive-car combinations that travel along some part of the rail network. Later various practical constraints have been added to the model, such as the maintenance issue. Computational experiments have been performed based on the instances from VIA Rail, Canada.

Lingaya et al. [58] study the problem of assigning cars to timetabled trains for VIA Rail, Canada. A complex model including a master plan to additional information regarding passenger demands is used. Coupling/decoupling activities among cars is allowed and notably the order of the cars has been explicitly considered. Some real-world requirements such as maintenance are considered as well. Dantzig-Wolfe decomposition in conjunction with a heuristic branch-and-bound procedure is employed to solve the relevant ILP. The model has been tested based on the instances from VIA Rail, Canada.

Brucker et al. [11] study the problem of locomotive hauled car circulations based on a given timetable. Since the compositions of timetabled trains in terms of cars have been given in advance, the focus is on finding efficient repositioning trips for the cars. The problem is solved by a simulated annealing method. Computational results for a real-world instance “Württemberg winter term 1996/97” in Germany are given.

2.2.2 Bus scheduling

Bus scheduling shares some features with rolling stock scheduling but is generally less complicated due to its nature, e.g. buses can move much more freely than rolling stock which is constrained by tracks and there is no auxiliary activities like coupling/decoupling and shunting which may involve issues like time allowance violation and blockage. Since each bus trip can only be assigned with one bus whatever the passenger demand is, the issue of redistributing vehicle resources across the network according to passenger demand no longer exists. Mature commercial software packages for bus scheduling have been developed long ago since the 1970s in the UK, mainly based on the researches at University of Leeds. We will focus only on the relevant literature on bus scheduling from the Leeds group.

In the 1970s, Smith and Wren [72] developed a multi-depot and multi-type bus scheduling system named VAMPIRES, which originated from the aforementioned locomotive scheduling system [80] in the 1960s. To deal with the multi-depot issue, a heuristic method extended from the heuristics for locomotive scheduling was devised. A multi-stage heuristic process was applied by repeatedly solving a classical Transportation Problem to give dual variables to guide the heuristics. VAMPIRE consistently gave savings of at least 5% of the fleet size. There were several bus companies routinely using it, including Greater Manchester Transport and Yorkshire Traction.

When an operator’s routes have been decomposed into smaller groups with well-establish regular services, the ability in VAMPIRES becomes superfluous. A simpler system, TASC [72], was later designed for such cases. In 1983, VAMPIRES was converted for running on microcomputers with some additional improvements such as a three-way swapping procedure. This upgraded version was used by more than forty bus operators and was said to be still in commercial use at the time the survey by Wren [82] was written.

In the 1990s, Kwan and Rahin [54] propose an object oriented approach for bus scheduling called BOOST (Basis for Object Oriented Scheduling of Transport), which embraces the object-oriented paradigm for better conceptualisation, extensibility and reusability. BOOST incorporates a modification on the VAMPIRES algorithm (and its commercialised version as the BUSPLAN component in BUSMAN [22]), using inputs of flexible definitions of the routes and service frequencies. BOOST is used in-house by several bus companies, including First, the UK's largest bus operator.

2.3 Vehicle scheduling and vehicle routing

Although vehicle scheduling and vehicle routing share a feature that a set of events (pick-up/delivery services, train/bus trips, etc.) needs to be covered by a vehicle or vehicles, the major difference between them is that in vehicle scheduling the event times would have been fixed beforehand (e.g. by a timetable) such that there are strict sequential and connection relations among the events, while for a typical kind of vehicle routing the event times are usually unspecified.

Bodin and Golden give a detailed review on vehicle routing and scheduling in [10]. In this survey a "vehicle route" is defined as an ordered sequence of events traversed by a vehicle, starting and ending at a depot; a "vehicle schedule" is defined as a sequence of events with preset arrival and departure times traversed by a vehicle in the designated order at the specified times. Sometimes a problem can have a more complex flavour, such as the requirements on time windows or precedence for vehicle routing. The authors regard those mixed problems as combined vehicle routing and scheduling problems. In addition, they have designed a taxonomy system in classifying the variations of vehicle routing and scheduling problems. For example, single- or multi-depot, homogeneous or heterogeneous vehicle fleet, etc. Moreover, a classification on solution approaches has also been defined, such as cluster/route/schedule first/second etc.

2.4 The integer multicommodity flow problem and its fixed-charge variant

The multicommodity flow problem can be generically characterised by a set of commodities to be flowed through a network, each with specified origins and destinations. Different types of the problem may differ from other features such as whether there are minimum supply requirements and/or imposed capacities restrictions shared by all commodities on arcs and/or nodes, whether there are upper bounds on flows' total amount to be used for each commodity, the components of the objective, and other possible side constraints. This problem can be traced back to the pioneering work by Ford and Fulkerson [30]. Due to its structure with block-angular matrices and linking constraints, it has triggered the emergence of some of the most important discoveries in operational research, for example, Dantzig-Wolfe decomposition [27]. Surveys on multicommdity flow problems can be found for example in Hu [43], Assad [5], Kennington [48], Ahuja et al. [2], Alvelos [4], Shetty [71].

In spite of many researches on the linear (continuous) version of the multicommodity flow problem where the variables can take continuous values, the corresponding integer version has received relatively less attention, especially for the versions whose variables can take general integral values rather than only binary ones, as well as the cases where a commodity can use multiple paths, as reported in [4]. In Barnhart et al. [7], an origin-destination integer multicommodity flow problem is solved by branch-and-price-and-cut, where the path variables are binary and each commodity can only use one path. This work is one of the pioneering ones in maturely using branch-and-price [8]. Notably, taking advantage of the problem's structure, the branching rule is applied without adding constraints but only deleting columns, and it prevents the generation of invalid columns in the subproblems. Moreover, the branching is more effective in eliminating the symmetry of arc variables in the branch-and-bound tree that may severely slow down the branching process. Lift cover inequalities are also used in conjunction with branch-and-price. Near-optimal solutions can be found in reasonable time in problems arising in telecommunication. In Alvelos [4], a solution approach based on branch-and-price for general integer multicommodity flow problem is proposed, where the invalid columns from the subproblems are prevented explicitly by adding branching constraints. Assuming the network is not acyclic, potential problems due to negative cost cycles are also eliminated in a formulation with path variables and cycle variables. Another branch-and-price-and-cut approach has also been proposed for binary multicommodity flow problems. Acceleration approaches, different decomposition approaches are also discussed and computational experiments are reported.

The fixed-charge multicommodity flow (FCMF) problem variant is characterised by adding an extra binary variable (the fixed-charge variable) for each arc to indicate whether the associated arc is used or not, meanwhile adding a term in the objective as the sum of all fixed-charge variables to optimise alongside with the original arc or path variable. Similarly, a mixed integer version and an all integer version can be further categorised, where in the former the arc or path variables take continuous values while in the latter they have to take integral values. There is somewhat a lack of uniformity in the terminologies for network flow problems with "fixed-charge" and "multicommodity" features. In fact most pieces of literature simply refer "fixed-charge multicommodity flow problem" as "network design problem", e.g. [2]. But the term "network design problem" itself can also be used for occasions having or not having the features of "fixed-charge" and/or "multicommodity" in some cases.

The FCMF problems are difficult compared with a generic multicommodity flow problem without the fixed-charge variables, even for the mixed integer version. Most of the current solution methods for FCMF problems are (meta)heuristic-based. Moreover, while there are plenty of researches on the mixed integer FCMF, the all-integer FCMF version has received much less attention. Hewitt et al. have proposed two hybridisation methods in [40, 41] for the all-integer FCMF problems where the arc or path flow variables are binary.

It will be shown in Chapter 3 that the TUSP can be formulated either as an integer multicommodity flow problem or as an integer fixed-charge multicommodity flow problem, depending on to what extent the various practical requirements are to be satisfied by the models. Each version has its advantages and disadvantages and the choice between them should be made in a problem-specific basis. This point will be revisited in Chapter 3 later. If the integer (fixed-charge) multicommodity flow models are to be formulated and solved as ILPs, column generation based methods such as branch-and-price are commonly used exact methods.

2.5 Pre-/dynamic-column generation

If branch-and-price [8] is chosen to solve an ILP, then a column generation process is needed, where except the initial columns providing a feasible solution to trigger the subsequent process, generally all new columns to be added to the restricted master problem (RMP) have to be generated by some methods. There are three usual ways for generating those columns.

The first kind, which is commonly seen and is taking advantage of the structure of the original problem, is to generate the columns dynamically from the decomposed subproblems until no promising candidate is available. Theoretically this approach will guarantee the optimality of the node LP. However, it also has two disadvantages, both due to the dynamical nature of how new columns are generated. One is the difficulty in satisfying complex rules imposed on individual columns; the other, arising in solving the problem by branch-and-price, is that dynamically generated columns may have properties inconsistent with the current branching status (Barnhart et al. [8]). The above disadvantages lead to the need for designing very complicated and sophisticated subproblems or customised branching strategies to prevent the generation of invalid columns. Constrained shortest path subproblems in crew scheduling (Irnich and Desaulniers [45]), generalised assignment problem (Savelsbergh [69]) and origin-destination integer multicommodity flow problem (Barnhart et al. [7]) are well-known examples.

The second kind is to perform a pre-generation in advance trying to construct all or the essence of possible candidates into an initial set, and then to carry out the column generation only based on this set. If, for example, the simplex method and reduced costs are to be used in the column generation, then candidates are priced out by explicitly computing and comparing their reduced costs, where it is convenient and flexible to add several candidates into the RMP from the same decomposed subproblem at one time.

Moreover, the above two disadvantages in dynamic-generation are unlikely to occur in pre-generation based column generation, making it more preferable for solving problems whose columns have very strict rules on validity such that it is basically impossible to generate them dynamically. Crew scheduling is a very good example for such problems where there are strict and sophisticated rules imposed on individual columns (which represents a shift of one driver's daily workload) as a result of union's agreement, operational regularities and so on such that it is only realistic to construct those shifts beforehand. The Generate-and-Select (GaS) method by Kwan [51] for crew scheduling is an example of pre-generation, as well as the other researches from the Transport Scheduling group at University of Leeds (Kwan et al. [50, 53, 55], Kwan [52], Fores et al. [31, 32], Wren et al. [83]) for bus and train driver scheduling. Hennig et al. [39] present a model for maritime routing where the columns are also pre-generated due to the complexity of the validity rules restricted on individual columns.

The third kind is somewhat between the above two, where the columns are constructed by heuristics guided by some rules as well as taking into account the dual information from the RMP and the subproblem structure. The construction and refinement phases for generating paths from Cacchiani's work for train unit assignment aforementioned in § 2.1.2 are examples of this category. Note that in these examples the column generation is not embedded into a BB tree framework to find integer solutions; instead, a diving heuristic by repeatedly fixing appropriate path variables is used.

Chapter 3

The train unit scheduling problem

This chapter gives an elaborated description on the train unit scheduling problem typical in the UK as introduced in Chapter 1 as well as the way it is going to be modelled in this thesis. Finally an “intuitively-constructed” ILP formulation associated with its modelling frameworks will be presented.

It is worth mentioning that the relevant knowledge on the TUSP in the UK is not well-documented—it is obtained mainly from the practitioners from railway industry based on their own experiences. Detailed problem description may also differ from operator to operator, even for train companies within the UK. In this chapter, the knowledge is mainly obtained from two train operators in the UK: First ScotRail and Southern Railway, as well as from the developers and consultants at Tracsis.

3.1 Requirements and restrictions

The problem description is started with the requirements and restrictions that a feasible and operable train unit schedule must comply with, since any violations on them will result in an invalid plan.

3.1.1 Fleet size limit

Each train operator has a fleet of units of limited numbers per type. Indeed a schedule whose used unit number exceeds the fleet size limit is invalid. Beyond this basic requirement, the operators usually would like to use as few units as possible to achieve the same or similar service provision for the input timetable.

3.1.2 Type-route/depot compatibility

There are compatibility relations among unit types and routes. Type-route compatibility relates to tracks and therefore sub-route sections, i.e. a train is constrained by the most restrictive sub-route it goes through. As each train belongs to a certain route, this finally turns out to be a compatibility relation among unit types and trains. Taking ScotRail for instance, there are 10 unit types in the fleet and more than 2000 trains in the timetable. However, for each train, there will not be more than 4 types permitted to serve it. The scenario for Southern Railway is slightly different. There are 11 unit types in the fleet where although for most trains not more than 4 types are suitable, there are some routes with trains that can be served by up to 7 types of unit.

The similar compatibility relations also exist for unit types and their home-depots. They may have an impact on the beginning/ending trains that a type of unit can serve and the assignment plan on relevant maintenance provision.

The type-route/depot compatibility can be realised in the directed acyclic graph (DAG) construction process such that certain type-graphs can only contain certain nodes and arcs (cf. § 3.6).

3.1.3 Time allowances

A prerequisite for two trains to be consecutively connected by the same unit would be that the arrival time of the previous train should be earlier than the departure time of the next train. However, extra buffering time has to be imposed on this time gap, which is conventionally called *turnround time* in the UK rail industry. Note that the turnround time only applies to the period connecting two trains, but not the intermediate stops within a train en route. According to the rules of ScotRail, the minimum turnround time for most stations is 5 minutes. Examples of exceptions include a 10 minutes turnround time at Edinburgh Waverley Station and a 1–2 minutes turnround time at Motherwell Station. Also, at Glasgow Central Station (High Level), it is desired that the minimum turnround time is 5 minutes during peak time (usually 7:00–9:30 and 16:30–19:00) and 10 minutes during off-peak time. If an empty-running or a shunting movement is concerned, its estimated time consumption should also be taken into account. When there is no coupling/decoupling operation involved, the time allowance requirement can be imposed in the construction process of the DAG such that only arcs implying valid gaps between two train nodes can be created (cf. § 3.6).

On the other hand, when coupling/decoupling operations occur during a time gap, the time allowance validity will be complicated, especially for tight connection gaps of 5–15 minutes. A single coupling/decoupling operation may take 2–5 minutes and it is possible for a unit to decouple first and then couple again to another unit within a gap. A non-trivial issue here would be that coupling/decoupling operations are in fact *unknown* beforehand and *dynamic* during the solution process such that the method of presetting the node and arc construction in the DAG as aforementioned becomes invalid in ensuring time allowances at gaps involving coupling/decoupling operations. In an approach to be proposed in this thesis, such time allowance issues involving coupling/decoupling will be treated either by explicit constraints in relevant ILP formulations or by a post-processing method as a subsequent step.

3.1.4 Passenger demands

Each train in the timetable should be covered by a unit or a block of coupled units whose total capacity satisfies a passenger demand expected for the train, which is often measured in number of seats.

The passenger demand is a very important input for an automatic scheduling system, since it has significant impact on train unit resource distribution over the network and objectives such as fleet size. On the other hand, its fuzzy nature and inadequate documentation by train operators make it not easy to obtain reliable and precise passenger demand data.

The analyses below are based on the unit diagramming knowledge from First ScotRail, the major train operator in Scotland. These may also be representative more or less in other UK train operating companies.

3.1.4.1 Minimum demand requirement

There is a minimum passenger demand level in terms of unit combinations required by the authorities or franchise agreements which has to be satisfied. Converting the minimum required unit combination into seat numbers, this requirement is usually easy to satisfy because the demand levels are much lower than the real diagrams used in practice. It is also noteworthy that the unit combinations actually used in manual diagrams can differ from the required ones as long as the level of seat provision is not reduced.

3.1.4.2 Passenger number count survey

Every year, for some of the timetabled trains, surveys of the actual passenger number count (referred to as “PAX” at ScotRail) will be recorded and given to the resource planning department. These PAX number surveys are very important data to consider for the next year’s unit diagramming as they reflect the real passenger flow distribution. It is observed that in the manual diagrams not all trains’ demands are satisfied against the PAX number, especially at peak hours. Those trains are referred to as “under-provided”. This is mainly due to the lack of unit resources available at peak hours, and the coupling upper bound restriction (cf. § 3.1.5.2) is also another minor reason for it. Moreover in the manual diagrams there are many trains whose capacity provisions exceed their PAX number, e.g. one unit would be sufficient to satisfy the actual passenger demand but the schedule uses two. Those trains, referred to as called "over-provided", are mainly caused by two reasons. One is that there is no place available for decoupling (cf. § 3.1.5); the other is that excessive capacity provision may be used to relocate unit resources to satisfy other trains later elsewhere.

3.1.4.3 Historical diagrammed capacity

Historical diagrammed capacity datasets in the past few years will be available when the scheduling on a newly published timetable begins. Since a large proportion of trains in the new timetable will remain unchanged compared with the previous year's timetable, the historical records can be used for reference. On the other hand, note that there can be updates and alterations in a new timetable every year, e.g. with new trains added and/or new routes concatenated/extended to original trains. In these cases, the historical records may be less important.

As for the value and reliability of the historical diagrammed capacity data, for one thing, it may contain the information on some implicit agreement/expectation with transport authorities; for another, it may include unnecessary over-provision as a by-product of manual scheduling on some trains having not been reviewed and checked over many years.

3.1.4.4 Fuzzy nature of passenger demands

As an important input for an automatic scheduling system, passenger demands in number of seats for each train have a very fuzzy nature. So far, manual practitioners in the UK usually schedule the units based on some rather rough seat number demands for each train, where three major factors are concerned: (i) the minimum level required, (ii) historical data in previous diagrams, (iii) actual passenger number counts and other local knowledge. For some train operating companies, the targeted capacity demands are not accurately defined and documented. Therefore it is often the case that for many trains in the timetable the "ideal" or "true" demands are actually unknown.

The estimation on the "true" demands is not easy. Simply using the historical records is not sufficient, even assuming most trains remain unchanged in the new timetable (not to mention this is not true). Such historical records may not be flawless in reflecting the true capacity requirements and some of their capacity strengthening is due to the need to redistribute the unit resources over the network rather than to satisfy the demand of the current train. From the manual schedulers' point of view, since the process is basically modifying previous schedules subject to the changed parts in the new timetable, the backbone of the new schedule will be heavily analogous to the previous ones and the overall unit resource distribution over the network are unlikely to change much. Therefore if there is a drastic change in the demand input, e.g. from recent PAX numbers, then it is not easy to reflect it in the new schedule due to the above reasons. In addition, if there have been some unreasonable local patterns in the historical diagrams, they are likely to remain in the new schedule year after year, making the diagrammed capacities difficult to be adjusted to the “true” demands.

On the other hand, directly using the PAX number is very likely to be unrealistic, since the PAX number for many peak hour trains are far too large such that satisfying all of them is impossible by the limited fleet size. In other words, if to use the limited number of train units to cover the timetabled trains, there have to be some trains whose capacity provisions are less than the PAX number. It is tricky to decide which subset of those high-PAX trains to satisfy and which not, to obtain some overall optimality, e.g. to have as many peak hour trains as possible with their PAX demand achieved.

3.1.5 Coupling and decoupling activities

A challenging issue which distinguishes train unit scheduling from other vehicle scheduling is that more than one train unit can be attached together to serve the same train, which is known as *coupling* and the reverse operation is called *decoupling*. Coupling/decoupling activities are often essential for satisfying high passenger demands at peak time by providing more capacities. They can also be used as a way to redistribute unit resources across the network, i.e. to provide unit capacities to some trains later elsewhere rather than for the current train whose passenger demand requirement may be actually very low. This ability is very important for optimising the overall schedule by minimising the fleet size and the number of empty-running trains, as the optimisation system can detect unit resource imbalance over the network and rebalance it globally preferably by coupling units to existing passenger trains.

For a set of coupled units as a *unit block*, both its *composition* and *permutation* are important. This is because in many occasions coupling/decoupling can only be performed in a certain order and corrective repositioning may not be possible within the time gap.

There are some rules imposed on coupling/decoupling activities that will further complicate the train unit scheduling problem. Also as an operation taking certain costs and resources, optimisation on coupling/decoupling is necessary. These issues will be discussed in the following.

3.1.5.1 Type-type compatibility and train unit family

Train units of the same type are permitted to couple and some different types may also be allowed to attach. This relation is referred to as type-type compatibility. With respect to type-type compatibility, the *train unit family* is defined such that coupling compatible unit types belong to the same family. So far in all the problem instances that have been studied, train unit families partition the fleet into mutually exclusive subsets, that is, a type can only belong to one and only one family and there is no overlapping of types among any families.

For most UK train operators, it is observed that the type-type compatibility relation will partition the fleet into families such that the number of types within each family is relatively small. Table 3.1 shows the unit type and family information of Southern Railway and ScotRail. It can be seen that in the Southern Railway fleet the 11 types are partitioned into 6 families and in the ScotRail fleet the 10 types are partitioned into 6 families. Some family can have as many as 4 members while some only have a unique member.

Table 3.1: Train unit types and their associated families of the fleets of ScotRail and Southern Railway ("c156" stands for "Class 156" and so on)

Operator	Family	Type	Capacity (seats)	# of cars
ScotRail	SR.I	c156	145	2
	SR.II	c158	136	2
		c170	189	3
		c170S	198	3
	SR.III	c314	212	3
	SR.IV	c318	219	3
		c320	230	3
	SR.V	c334	183	3
	SR.VI	c380/0	208	3
		c380/1	282	4
Southern Railway	SN.I	c171/7	107	2
		c171/8	241	4
	SN.II	c455/8	316	4
		c456/0	152	2
	SN.III	c313/1	194	3
	SN.IV	c460/0	366	8
	SN.V	c442/1	320	5
	SR.VI	c377/1	223	4
		c377/2	223	4
		c377/3	160	3
		c377/4	243	4

3.1.5.2 Coupling upper bounds

When train units are coupled into a unit block, the total number of cars is restricted by an upper bound. Since only units of the same family can be coupled, it is reasonable to investigate and assign the bounds in a family-by-family way. The coupling upper bound restriction rule is mainly determined by the following factors:

- *Unit combination*: Different type combinations have to follow different coupling upper bounds. Regardless of the other factors below, this is the basic rule in determining the maximum number of coupled cars. Table 3.2 gives the upper bounds solely as a result of type combinations in the fleets of ScotRail and Southern Railway. The coupling upper bounds' combination-specific feature sometimes may seem strange.

For example, the members in Family SN.I can only form a 2-unit block whatever types used, as c171/7+c171/7 (4-car), c171/8+c171/8 (8-car) and c171/7+c171/8 (6-car), although all individual cars in either c171/7 or c171/8 have the same *car* length of 23.62 metres.

- *Routes*: Routes may further impose restrictions on coupling upper bounds, due to reasons like platform lengths, infrastructure requirements, operational / engineering rules, etc. Taking the example of Family SN.I again, on most routes the coupling upper bound is 2 units of whatever combinations as discussed above. However, in the route between Ashford and Hastings, no coupling is allowed at all so that only a single unit of either c171/7 or c171/8 can cover the trains running there. Another example from ScotRail is that, for types of c156, c158 and c380/1 running on the route between Glasgow Central and Edinburgh Waverley via Shotts, the coupling upper bound will be reduced to 4 cars while normally these types can make a formation up to 6 or 7 cars. It is possible to create sub-family variants by routes within the same family, in order to make it more convenient for data processing.

- *Time band*: For some part of the rail network, different time bands will have different coupling upper bounds for the same route and combination. For instance, type c158 and c170 from Family SR.II can cover the route from Aberdeen to Inverurie and usually the coupling upper bound is 5 cars (i.e. c158(2 car)+c170(3 car)). However, during the evening peak time a maximum of 7 cars is allowed, thus more combinations such as 3×c158, 2×c170 and 2×c158+c170 are permitted during that time. Similar to the Routes, it is possible to create sub-families as a result of time band within the same family.

It is worth mentioning that in Table 3.2 as well as in the documented rules from many train operators, the fact that the coupling upper bound is measured in the number of cars does not mean the car number is the only reason or restrictive factor for yielding a coupling upper bound. In fact, it can be a result of multiple restrictive factors imposed concurrently such as the number of cars/units, platform lengths, engineering facilities, etc., and the most restrictive one will take effect. This also explains why the upper bounds can be irregular or combination-specific when they are measured only in the number of cars as one can observe in Table 3.2.

3.1.5.3 Locations banned/restricted for coupling/decoupling

Due to reasons in engineering/operational rules, facility/infrastructure restrictions and so on, at some locations coupling/decoupling operations are either totally forbidden or restricted under certain conditions. For example, Dalmuir Station from the ScotRail network belongs to such a category. In addition, coupling/decoupling may be banned/restricted

Table 3.2: Coupling upper bounds in number of cars associated with type combinations, regardless of other factors as routes etc., from fleet of ScotRail and Southern Railways

Operator	Family	Combination	Upper bound (in # of cars)
ScotRail	SR.I	c156	6
	SR.II	any combinations by c158, c170, c170S	6
	SR.III	c314	6
	SR.IV	any combinations by c318, c320	6
	SR.V	c334	8
	SR.VI	any combinations by c380/0,1	7
Southern Railway	SN.I	c171/7 only	4
		c171/8 only	8
		c171/7 and c171/8	6
	SN.II	c455/8 only	8
		c456/0 only	6
		c455/8 and c456/0	8
	SN.III	c313/1	3
	SN.IV	c460/0	8
	SN.V	c442/1	10
	SN.VI	any combinations by c377/1,2,3,4	12

since some large stations are too busy such that speedy smooth throughput is often required to avoid any disruptions. Edinburgh Waverley Station is such an example.

For a location where coupling/decoupling is banned, no such operations are permitted at any time and any place in that location at all. For a location where coupling/decoupling is restricted, the operation permission is subject to various conditions, e.g. time band, platform, train directions ("up" or "down"[1]), route, passenger/empty-running train, etc., and perhaps a combination of the above. Moreover, for some stations, only one of the operation in coupling and decoupling is permitted. One particular potential difficulty in automatic train unit scheduling regarding the platform-specific ban is the fact that not every train operator will have the detailed platform information assigned to each train service prior to the unit diagramming stage. Although some important, well-established routes may have conventionally allocated platforms, there are always some trains whose platforms are left undecided. This is undesirable as it makes the information needed for forbidding coupling/decoupling incomplete at the unit diagramming stage.

Here we give some examples from the rules regarding banned/restricted locations for coupling/decoupling at ScotRail to illustrate their complexity:

- *At Edinburgh Waverley Station: Coupling/decoupling is banned from 6:30 to 19:30 for trains in the GB area on weekdays.*
- *At Ardrossan South Beach Station: Coupling is permitted in up direction only; de-*

[1]For a rough guide, a train's direction is "up" if it is heading from London and is "down" if it is heading towards London. The "up" and "down" designation is mainly to distinguish the two possible directions of travel on a track.

coupling is permitted in down direction only.

- *At Kilmarnock Station: Coupling is only permitted for the trains from Barassie to Kilmarnock route; no other coupling permitted; decoupling is not restricted.*
- *At Dalmuir Station: Coupling is not permitted; decoupling is only permitted at Platform 5 (Bay Platform); all other coupling and decoupling must take place in Dalmuir Down Siding subject to maximum 6 cars.*
- *At Bathgate Station: Coupling is not permitted; decoupling is only permitted for empty-running movements arriving from Bathgate Depot.*

The proportion of such locations is not small. Taking ScotRail for example, among the 75 locations that can be the end points of trains, coupling/decoupling is totally banned at 49 of them.

3.1.5.4 Coupling/decoupling en route

Sometimes coupling/decoupling may not be only allowed at the origin or the destination of a complete train journey but at some intermediate stations, i.e. train units may be attached or detached en route. These scenarios are sometimes due to the topological structure of the rail network like hubs or junctions; they can also be a result of more detailed and sophisticated operations. For instance, at NSR in the Netherlands, coupling/decoupling can take place en route such that a passenger has to carefully choose the carriage according to different locations s/he is going to get off. In the UK, this sort of operation is very rare and therefore will not be considered in the thesis. Finally, although sharing some features, coupling/decoupling en route is not entirely the same activity as the “combining/splitting” of trains described by Fioole et al. [29] and Maróti [61]. The main difference is that the former is not necessarily a required operation while the latter is imposed to take place. When a coupling/decoupling operation en route is due to network structure reasons and is required, it may be regarded as a joining/splitting activity as defined at NSR. See further details in § 2.1.1.

3.1.5.5 Optimisation on coupling/decoupling

Although appropriate coupling/decoupling activities may contribute to an optimal schedule in terms of fleet size minimisation and demand satisfaction, they would also consume resources of tracks, shunting operations, crews, time, etc. More seriously, our tests on real dataset show that excessive/redundant coupling/decoupling operations are one of the major reasons for unit blockage at stations if station layout is ignored (cf. § 3.1.7). Therefore, it is important to minimise those operations. In other words, if potential coupling/decoupling activities cannot make a contribution in optimising the schedule, it is preferable to keep a unit block together as much as possible. This issue may seem trivial and obvious for manual scheduling but special attention has to be taken for automatic scheduling based on ILP.

3.1.6 Empty-running

An advantage of redistributing a train unit across the network by means of coupling to a timetabled passenger train is that no conflicts in terms of track usage will be expected giving ease to the scheduling process. On the other hand, generating an empty-running train without passengers may be more flexible in terms of its timing. However there are disadvantages as well. The track path and timing have to be checked and approved by authorities such that no conflict with other track users may occur. Moreover, empty-running does not directly contribute to passenger carrying capacity but has significant additional operational costs. Therefore, empty-running should be avoided if an alternative way suffices in achieving the same effect.

Empty-running trains often occur at medium or large time gaps between passenger trains where sufficient time is available. They are also occasionally accompanied by additional activities such as returning to depots during off-peak time and maintenance work. Empty-running trains may be planned on non-passenger routes, which may either be a shortcut journey or to allow the train units to use places that passengers usually will not reach.

Note that in unit diagrams, activities of shunting a unit from/to a depot at the beginning/end of the day will also be regarded as empty-running. They are essentially unavoidable and thus are different from the aforementioned empty-running that connects passenger trains. Empty-running may also include coupling/decoupling from/to depots that can further complicate the problem. Some other shunting activities like re-platforming, shunting from/to sidings, etc. are also recorded as empty-running in unit diagrams. These shunting-related empty-running will be discussed in detail later in § 3.1.7.

In an automatic scheduling system, empty-running trains are usually planned using a predefined collection of possible empty-running journey time allowances between location pairs. There may also be time bands when empty-running is prohibited. Another workable way for the system is to only use the existing empty-running trains found in manual schedules. At the end of the scheduling process, empty-running trains are planned in detail and ensured to be conflict free. At that stage, it is possible that some adjustments to the schedule are required because some empty-running trains may turn out to be infeasible.

3.1.7 Shunting movements, unit blockage and rail infrastructures

Sometimes the connection between an arrival train and a departure train by the same train unit can only be accomplished by certain shunting movements. In railway operations, *shunting* generally refers to empty-running movements between platforms, sidings and depots during a connection. Appropriate shunting plans enable train units to be sent to the right place at the right time without causing blockage with each other.

Different shunting movements require different lengths of connection time gaps. For example, shunting between platforms and sidings usually takes shorter time than shunting from/to a depot. When an available time gap is large, any type of shunting can be feasible; as the gap gets shorter, it becomes more restrictive. For some train operators the arrival and departure platforms for trains have already been assigned at the timetabling stage while others may not give the assignment at all or only have tentative plans. The assignment of which sidings or depots units will be shunted from/to are only planned at the train unit scheduling stage. Since depot shunting usually happens during a long idle time for the unit, it is generally less constraining in producing unit schedules.

A distinctive characteristic that makes vehicle scheduling on railway different from vehicle scheduling on road (e.g. bus) is that movements of rolling stock vehicles are strictly restricted by tracks and other rail infrastructures, which will give various blockage problems. The blockage problem can be further worsened by excessive or redundant coupling/decoupling operations, e.g. two units are supposed to attach but a third unit is in the middle. Some researches in the literature refer to the blockage problem as *crossing* (Freling et al. [33], Kroon et al. [49]). Detecting and eliminating potential blockage requires the collection of comprehensive station-level infrastructure knowledge and operation details, including track and platform layout, train directions, unit permutation (when coupled), and platform assignment, to name only a few of them. Our computational tests based on real-world datasets show that a train unit schedule solely based on timetable, fleet and network/route knowledge without complete station-level details may often be inoperable due to unit blockage.

Now we give two simple examples to show how unit blockage can invalidate a schedule when some station-level details are ignored.

Example 3.1

Table 3.3 gives a timetable of four trains i, j, m, n to be scheduled. Suppose that the destination of i, j and the origin of m, n are the same station and all the trains have been assigned to the same Platform 1 which is a long platform subdivided into 1a and 1b with bidirectional track. There are two train units ① and ② available and are both compatible with all the four trains where each train's demand would be satisfied with any one of the two units. Suppose Unit ① serves i and Unit ② serves j, then applying the FIFO rule matching arrival trains to departure trains would yield the following solution:

(i) Unit ①: $i \rightarrow m$; Unit ②: $j \rightarrow n$.

However solution (i) is infeasible since when departing at 10:10, Unit ① will be blocked by Unit ② whose departure time is 10:20. Note that however it is impossible to make this judgement without station infrastructure and train direction knowledge. It is easy to see,

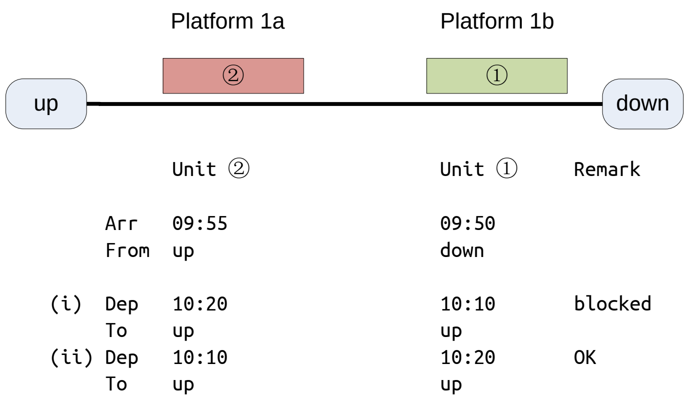
Table 3.3: Four trains associated with the long Platform 1 (Example 3.1)

Train	From	To	Dep. time	Arr. time
i	down	–	–	09:50
j	up	–	–	09:55
m	–	up	10:10	–
n	–	up	10:20	–

if the above missing information is known in advance, that an alternative feasible solution can be found as:

(ii) Unit ①: $i \to n$; Unit ②: $j \to m$.

The two solutions are illustrated in Figure 3.1.



```
            Unit ②              Unit ①     Remark

      Arr   09:55               09:50
      From  up                  down

(i)   Dep   10:20               10:10      blocked
      To    up                  up
(ii)  Dep   10:10               10:20      OK
      To    up                  up
```

Figure 3.1: The scenarios at Platform 1 (Example 3.1)

Example 3.2

Consider four trains to be scheduled in Table 3.4 with three compatible units available as Unit ①, ② and ③ all allowed to cover any trains. In this case coupling/decoupling is involved in a unit blockage issue. Assume that a block of ① and ② has been assigned to i which arrives at Platform 2 later and ③ has been assigned to j which arrives at the same platform earlier. Also take the assumption that ② is at the front of i. There are two trains m, n need to be linked with the units from the arrival trains i, j, where m needs 2 units and departs earlier while n requires 1 unit and departs later. Without the infrastructure knowledge, one can propose three solutions for the linking, which are

(i) ① $i \to m$; ② $i \to m$; ③ $j \to n$.

(ii) ① $i \to m$; ② $i \to n$; ③ $j \to m$.

(iii) ① $i \to n$; ② $i \to m$; ③ $j \to m$.

However after an analysis one may find that Solution (i) and (ii) are actually infeasible without additional shunting operations due to blockage. Unfortunately given the relatively

Table 3.4: Four trains associated with the long Platform 2 (Example 3.2)

Train	From	To	Dep. time	Arr. time	demand (units)
i	up	–	–	13:16	2
j	up	–	–	13:10	1
m	–	down	13:25	–	2
n	–	down	13:30	–	1

short time gap, shunting those units to sidings or other platforms is unlikely to be possible. Solution (i) asks the block ①+② to continue to cover m which also requires 2 units, but m will be obstructed by n served by ③ departing later. Solution (ii) unrealistically asks ① to couple with ③ while ② is in the middle. Only Solution (iii) yields a feasible schedule where ② is coupled with ③ to serve the earlier train m and then ① will cover n that departs later. Figure 3.2 illustrates the platform scenarios for the three units in Example 3.2.

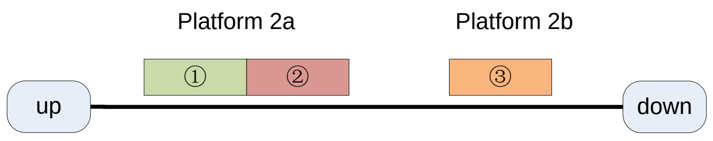


Figure 3.2: The scenarios at Platform 2 (Example 3.2)

The above unit blockage problems may seem to be unlikely to happen if a human scheduler is planning the diagrams in a trial-and-error way station by station. However they are indeed commonly observed in the solutions given by optimisation models we have developed on real-world datasets if no infrastructure knowledge of each station is considered.

It is also worth mentioning some issues related to infrastructures such as facility capacity overflow (e.g. at platforms/sidings/depots, etc.), compatibility between traction types and facilities, unit permutation (especially when performing coupling/decoupling and reversing), which are also crucial in forming a feasible unit schedule.

As for the TUSP, the process of train unit assignment based on timetable, fleet knowledge and route information over the entire network can be regarded as a kind of high-level (global) planning. On the other hand, shunting movements planning and blockage removal based on each station's infrastructure knowledge can be regarded as a kind of low-level (local) planning. Manual schedulers' experiences indicate that the two levels of TUSP can be dealt with separately. This topic will be further discussed in § 3.4.

3.1.8 Unit overnight balance

Generally, train units are not allowed to stay overnight on the main tracks at stations. After finishing a day's work, a train unit has to return to a depot or a siding for overnight berthing by empty-running and starts the next day's work from the berthing place to the departure location of the first train also by empty-running. The rules on how train units are assigned to overnight berths differ from operator to operator. At ScotRail for instance, each end point station has its designated overnight berths for the train units finishing the daily work there, which is called “Stabling Restrictions”. Most stations in the ScotRail network cannot accommodate any overnight unit at all (e.g. units from the GB area finishing daily work at Edinburgh Waverley Station have to be shunted to the nearby Bathgate Depot) while a small number of stations can be used for berthing by themselves (e.g. Helensburgh Central Station). On the other hand, at Southern Railway, train units are assigned to overnight berths by routes and/or unit types. In this thesis, we assume that overnight berthing follows the rule from ScotRail, i.e. overnight berths are assigned according to the beginning/end stations from train units’ workloads, and from now on we uniformly call all types of overnight berths (depots, sidings, stations) as depots.

Based on the above scenario, there is a requirement that diagrams on two consecutive weekdays have to be linked with each other such that the number of units per type at each depot at the end of the first day should be the same as that at the beginning of the second day. This is referred to as *overnight balance*. The ways the balance is realised are somewhat different according to which two weekdays are involved. For many train operators in the UK, unit schedules on Monday, Tuesday, Wednesday, Thursday and Friday are almost identical and are denoted as “SX” diagrams, while diagrams on Saturday (“SO”) and Sunday (“SU” or “SUN”) can be very different from each other and from the SX days. For the SX days, one only needs to balance the beginning and the ending units of the identical SX schedule and this pattern can be repeatedly used for linking any consecutive SX days, e.g. Monday to Tuesday, Thursday to Friday. On the other hand, when an SO or SU schedule is involved, it requires the knowledge of two kinds of diagrams among SX, SO and SU. For instance, linking a Friday’s schedule to a Saturday’s needs to balance the SX ending units with the SO beginning units; linking Saturday to Sunday needs to balance the SO ending units with the SU beginning units. In addition, SO/SX trains may be less frequent than SX trains such that balancing from Friday to Saturday and from Sunday to Monday may not be difficult.

In the UK railway industry, the overnight balancing is usually not included into the main process of train unit scheduling but as a subsequent post-processing. The manual schedulers will take the raw results of train unit scheduling as the inputs of the balancing work and try to keep everything balanced either by modifying the tentative unit schedule or even inserting additional empty-running trains between depots after all services. Often there is sufficient flexibility for them to find feasible plans in this way. Nevertheless, the unit overnight balancing on the SX days will be considered in this thesis as a supplementary function to the scheduling models, mainly due to the practitioner’s wish to ease the subsequent manual process.

3.1.9 Unit maintenance and cycles

There are several kinds of maintenance work for train units. The first kind is the major maintenance called "big service" that each unit has to undergo within a certain range of mileage accumulated. It often needs a long time to perform such a kind of service. The second kind is the daily inspection checks processed in a relatively shorter time. The third kind is the cleaning/washing work. For diesel units, an additional refuelling operation has to be performed, which is also associated with a mileage accumulation like the "big service". Maintenance operation is often performed at depots, non-stabling sidings or specially constructed facilities away from station tracks during time gaps at off-peak hours. For a train unit assigned to maintenance, it generally needs to run to/from the maintenance location by empty-running. The mileage limits and permitted maintenance locations may differ according to the factors such as unit types, routes, time band, etc.

A physical train unit does not need to perform the same diagram of train sequence every day. Indeed as the units of the same type are interchangeable, a physical unit can take tasks of different diagrams of its type on different days. Often this is realised by unit cycles (or unit rostering). For instance, there are 20 units of a unit type (excluding backups etc.) corresponding to 20 diagrams. Often a physical unit of this type will perform daily workloads of different diagrams among the 20 units in a given order such that a unit cycle is formed from coming out of a major maintenance until the unit goes in for the major maintenance again. Therefore unit cycles are closely related with the above maintenance issue. Note that the long gaps suitable for a major maintenance check cannot be available in every diagram. Therefore it is important to ensure all physical units have the opportunity to perform the diagrams containing long gaps, which can be realised by setting appropriate unit cycles.

Like unit overnight balancing, in UK railway industry unit maintenance and cycle planning is usually performed in a trial-and-error approach after the main unit scheduling process. Experiences from the manual schedulers show that this way is usually applicable due to the sufficient flexibility. Based on this fact, maintenance and cycling will not be considered in depth in this thesis.

3.2 Objectives and solution qualities

In this section the objectives of the train unit scheduling problem that have close relation with the solution quality issue are discussed. Those objectives should be converted into measurable quantities to be processed by optimisation models. Often many of the quality requirements and preferences are complex to model and difficult to optimise and evaluate, however their great importance in helping the operators to accomplish the agreement or win franchise bidding heightens the solution quality issue.

It may be worth mentioning that for most public transport scheduling (and possibly other sorts of scheduling) the definition of "solution quality" itself is a tricky issue. Except in some really simple situations or simplified models designed for theoretical interest, the "best/optimal solution" for a real-world problem often does not exist—it may not even be possible to accurately define it in the first place. It could be influenced by both subjective and objective factors, and one solution considered to be "better" for one instance or scheduler could be deemed "worse" for another instance or scheduler. As commented by Wren [82] on forty years' experience on developing scheduling systems, "A scheduling system does not produce *the* correct schedule; there is no such thing". Therefore regarding solution quality, what the designer of such systems can do is to give the system the ability to reflect different quality evaluation requirements as much as possible. As for models based on ILPs, this could be done preferably through the quantitatively measurable objective function. With careful weight setting and parameter tuning, the ILP can reflect the user's wish in yielding solutions with their preferred qualities.

The following attempts to convey the main aspects on the solution quality issue in the train unit scheduling problem.

3.2.1 Fleet size

Because of the high costs associated with leasing and maintaining a train unit, minimising the fleet size is the most important objective, given the basic fleet limit requirement has been satisfied by compulsory constraints. In a multi-objective environment, the fleet size will be given the largest weight to ensure its priority in the optimisation.

3.2.2 Operational costs

Common operational costs include carriage-kilometre (mileage), empty-running movements, shunting movements and so on. Their weights in the objective function of an optimisation method should be carefully tuned to achieve the most desired effects. Moreover, different users may need to alter the weights according to their own needs on which cost is more desirable to minimise than others.

Carriage-kilometre refers to the total mileage incurred by all train units or all unit cars. It is natural to think that if one unit suffices in serving a train then it is a waste to use a coupled two-unit block. However in real-world practice things may be more complicated. For instance, as aforementioned, over-provision may be used to relocate unit resources across the network such that a smaller fleet size and/or less empty-running can be achieved. If the cost for an over-provision instance is set less than an empty-running movement and than the use of a train unit, then the system will make a schedule with more over-provided trains to reduce the fleet size and the number of empty-running trains, which is more reasonable. Another tricky problem called "short diagrams" due to minimising carriage-kilometre will be explained in § 3.2.4.4.

3.2.3 Redundant/excessive coupling/decoupling

In a manual scheduling process, the redundant/excessive coupling/decoupling could be carefully avoided. However, as aforementioned, based on the observation from our computational experiments, the scheduling results given by an automatic optimisation system will often yield too many redundant or excessive coupling/decoupling operations if no special attention is taken into account. Those undesirable operations will make considerable waste on operational resources and are generally too impractical to be accepted by train operators. They may also lead to unit blockage.

3.2.4 Preferences

Various preferences can be found in forming a real-world train unit schedule. Here are a few examples.

3.2.4.1 Unit types

Beyond the mandatory compatibility requirements between unit types and routes/depots, there can be preferences for permitted combinations among them too. For instance, unit types c156 and c158 are both permitted for running in the route between Glasgow Central and Edinburgh Waverley via Shotts, but c156 is more preferred.

3.2.4.2 Long gaps for maintenance

A schedule will contain time gaps of different lengths where a unit is not serving a train. Although such idle time is usually not productive, as aforementioned, some long gaps would be desirable to ease the subsequent maintenance planning such that maintenance tasks can be inserted into appropriate long gaps. Finding the correct pattern and balance of such long gaps are not a trivial work. Generally those long maintenance gaps should be between the morning and evening peaks. However the accurate peak time ranges are not always clearly defined and can be type-/train-specific. Figure 3.3 gives an example of a desirable long gap in the off-peak time of the day.

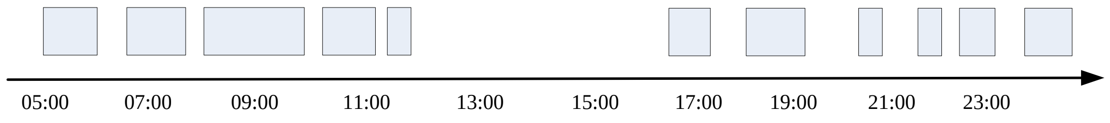
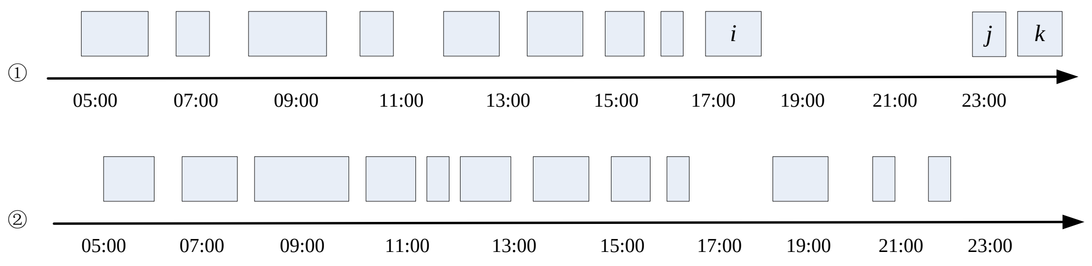


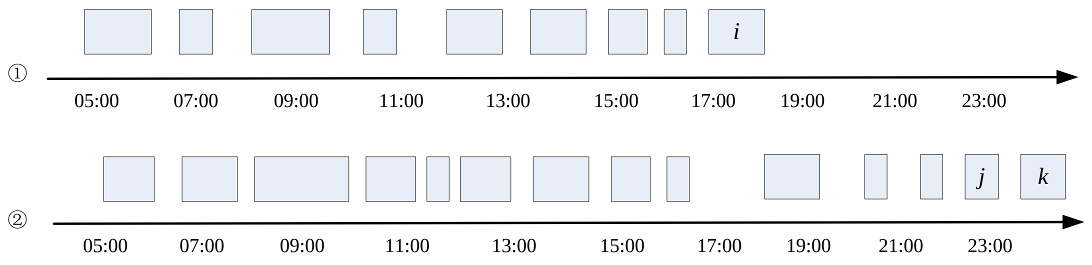
Figure 3.3: A desirable long gap for maintenance in the off-peak time

3.2.4.3 Gaps at night

On the other hand, some medium and long gaps are not desirable in some occasions. For instance, it would be undesirable to have a long gap in the evening followed by only a small number of remaining trains late at night to finish a day's work. A better alternative (with all other solution qualities such as fleet size and empty-running unchanged) would be to have a unit finish its work early in the evening and let another unit serve the remaining trains in a tight pattern until late night. Figure 3.4 shows a comparison between a less preferred night pattern where after serving train i Unit ① has to wait a long time to finish its work by covering j and k and a more preferred night pattern where Unit ① finishes its work at train i while Unit ② will cover j and k at late night.



(a) Less preferred pattern at night



(b) More preferred pattern at night

Figure 3.4: Medium and long gaps at night

3.2.4.4 Short diagrams

There is a tricky issue due to minimisation of carriage-kilometres and imbalanced passenger demand distribution among peak and off-peak time bands: It is often observed that the scheduling system may produce a schedule with a small number of "short diagrams"—diagrams only containing one or two trains (mostly peak hour trains). The reason is that the number and passenger demands of peak trains are much larger than off-peak ones. Those short diagrams cannot include more off-peak trains since the demands there have already been satisfied by other units, thus adding them can only increase the carriage-kilometres which, at that point no more savings on fleet size or empty-running can be made by over-providing those off-peak trains, is less preferable. Schedulers may find those short diagrams unacceptable—having a train unit only serve one or two trains a day is certainly a waste of resources. However adding some off-peak trains to them may not be a perfect resolution either, as this is also a waste in running units on off-peak services that do not need extra capacity strengthening. Here a kind of dilemma is in the way. This issue also has a close relation with the estimation and setting of passenger demands. Perhaps a modification on passenger demands of the off-peak trains would give a better result. But it is also not easy to decide which trains to have their demands strengthened beforehand.

This topic will be revisited in Chapter 6.

3.3 Current manual train unit scheduling process

In the UK railway industry, train unit scheduling (or train unit diagramming as it is called by the manual practitioners) is currently processed manually. Although details may vary among schedulers and/or train operators, the basic principle is similar.

Generally speaking, the manual practice is to modify from the existing diagrams subject to the updated timetable on a station-by-station basis. At each station, given the new timetable and fleet/route information, there will be a list of arrival and departure trains and how they were matched in the existing diagrams. When a modification is needed due to timetable updates, schedulers will try to rematch arrival trains to departure trains in affected parts, with circumspection to ensure no violation will occur on compatibility, demand satisfaction, coupling/decoupling, unit blockage etc. A train with higher demand may need to be covered by multiple units and thus may have more than one arrival or departure train to match. If an arrival train cannot be paired with a departure train, then it indicates that a unit is stabled at the depot associated with the station with this train as its last service in the day; similar for a departure train and a unit taking it as its first service in the day. The process often requires several iterations by a trial-and-error at the same or different locations to get a feasible schedule with acceptable quality.

This manual approach does have some advantages. First, as it is based on the previous schedules, there will be less drastic change in the new diagrams which may be helpful in ensuring robustness. Second, the manual circumspection for complex restrictive requirements straightforwardly satisfies the constraints that are difficult to realise by optimisation models, e.g. unit blockage, coupling/decoupling optimisation, various preferences.

The disadvantages of the manual approach are as follows. First as it is carried out at each station, the solution is usually not globally optimal. For instance, finding out the best capacity provision for each train is quite tricky, as capacity strengthening for a train may only be used for redistributing unit resources rather than satisfying the passenger demand for that train. Only a global approach can fully capture the unit flow balance over the entire network and determine the right provision for each train. Second, this local method can sometimes trap itself into a loop, as modification in one place may cause relevant changes in other places and eventually the problem will be bounced back to the origin. Sometimes the chain-effect may be quite serious making a number of previously finished places infeasible again. Third, the new diagrams given by this method may be adversely affected by historical schedules. Some scheduling decisions were passed down year after year without being challenged or reconsidered. For instance, there were some unreasonable patterns in previous diagrams in using more coupling operations to satisfy some peak time trains, but the new diagrams cannot dispose of them simply because of the "inertia" from the old schedule. Finally, if compared with an automatic system, a manual process is time-consuming—it may take months to produce a new schedule after the timetable has been finalised.

3.4 Network-level and station-level scheduling

The entire train unit scheduling problem would be too huge-sized to be tractable if all detailed station-level restrictions are also considered in a process of solving the entire problem once for all. Therefore, a two-phase modelling strategy is proposed to allow some of the station-level requirements to be left from a main network-level framework, whose basic duty is to globally assign train units to trains temporarily ignoring some station layout details. The network-level framework can be regarded as an upgrade of the DAG framework in Cacchiani et al. [15]. However what makes the new framework different is that it is capable of handling the following critical real-world requirements and constraints:

- Non-negligible time consumptions taken by coupling/decoupling during turnround connection gaps (may also be handled by post-processing)
- Unit type compatibility when coupled
- Combination-specific coupling upper bounds
- Locations banned/restricted for coupling/decoupling
- Avoidance of unnecessary/excessive coupling/decoupling (may also be handled by post-processing)

The aim of the network-level framework is to produce solutions that are near to being operable. The station-level post-processing finally determines the detailed operational plans in a station-by-station manner taking into account the layout constraints such as unit blockage removal, precise track assignments on platforms/sidings/depots and the order of coupled units.

Traditionally human schedulers compile train unit schedules based on individual train stations. Because of the input timetable, all train arrivals and departures are predetermined and a human scheduler would try to make feasible links between them at each station. This manual process is analogous to the post-processing phase. However, the consideration of empty-running across stations, coupling and decoupling possibilities and passenger demands would force the scheduler to switch their thought process between stations on a trial-and-error basis; the network-level model takes a holistic network-wide approach instead. The rationale for the proposed two-phase approach is partly due to the observation on the manual process in the UK that once the cross-station flows are satisfactorily or nearly sorted, the final logistics at the station level is usually resolvable. This decomposition also shares some similarity as what the practitioners at NSR in the Netherlands have been doing according to [61]. Finally this observation has been further reaffirmed by our experiments in post-processing the first phase results using TRACS-RS [73], a piece of interactive software which aims at facilitating human schedulers' manual process by visualizing and resolving blockage and shunting plans at the station level. Results of the network-level model have been uploaded into the TRACS-RS system and operable final results can be easily obtained by some straightforward modification in all such experiments. The post-processing basically realises the same tasks as TRACS-RS, and it is our wish that in the future it will be a totally automatic process.

3.5 Different modelling strategies for network-level framework

The job of the network-level framework is to assign train units to timetabled trains taking a global view in distributing unit resources across the entire network with most of the constraints satisfied and most of the objectives optimised.

Methodologically, there can be several alternative strategies in modelling the network-level framework. The most commonly seen ones are often based on a network flow structure consisting of nodes and arcs in directed graphs. This section will briefly discuss three of such possible modelling strategies, where the first two have been seen in previous literatures while the last one will be proposed as a novel modelling strategy for the train unit scheduling problem in this thesis.

To better illustrate the connections and differences of the three strategies, a very simple timetable with 5 train services is given in Table 3.5. The trains are labelled as i, j, k, m, n involving 4 stations A, B, C, D and 2 available unit types ① (capacity 280 seats), ② (capacity 260 seats). The passenger demands are given in number of seats. The subsequent descriptions will show how the scenarios in this timetable will be represented in different modelling strategies.

Table 3.5: A timetable containing 5 train services

Train	Dep	From	Arr	To	Demand	Unit type
i	08:05	A	09:30	B	200	①, ②
j	08:30	C	09:55	D	500	①, ②
k	09:45	B	11:30	C	300	①, ②
m	11:55	C	13:05	A	150	①
n	12:05	D	13:55	C	390	②

3.5.1 Flow graph representation

The flow graph representation has been used by the Dutch group [3,29,61,70] for studying the NSR networks. A flow graph (also called a time-space graph) consists of nodes and directed arcs. Figure 3.5 gives an illustration of the major aspects of such a flow graph corresponding to the timetable in Table 3.5. Note that a node represents an event of either an arrival or a departure of a train at a station such that the events at the same station are aligned in the same dashed line. There are two kinds of arcs. A trip arc connects two nodes from different stations (shown in the oblique arcs between different stations) and a station arc links two consecutive nodes of the same station (shown in the horizontal arcs within the same station). At each node of the flow graph, the flow balance has to be ensured such that the amount of incoming flows is the same as outgoing flows for each unit type. This flow graph representation can be formulated as an integer multicommodity flow problem where each commodity corresponds to a unit type, e.g. see [3,70].

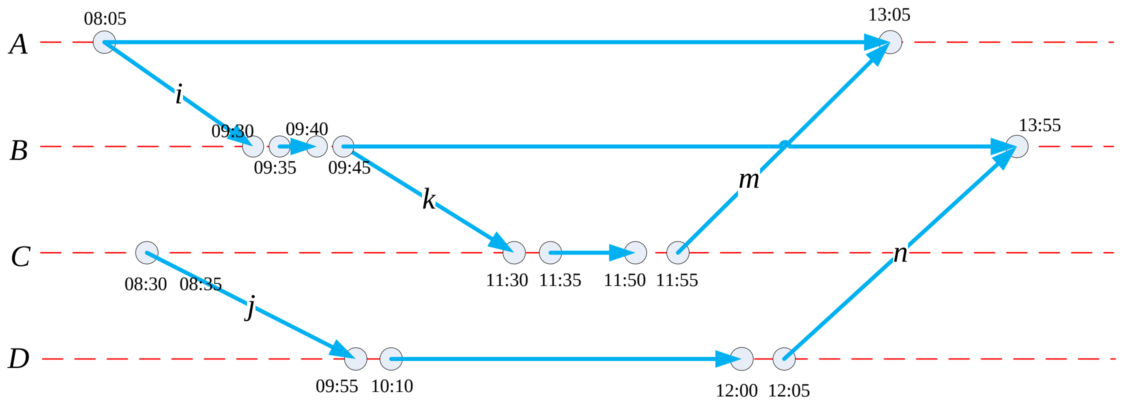


Figure 3.5: A flow graph corresponding to the timetable in Table 3.5

The flow graphs are suitable for a planning method where (i) most train connections are given in advance, and (ii) the rail network and timetable have some well-established patterns such as regular arrival/departure times and decomposed line groups (e.g. usually Figure 3.5 will imply that trains $i, \ldots, n$ are all from the same line where the stations are exactly ordered as $A-B-C-D$). There is no report on how this kind of representation can handle rail networks with a more complex topology than lines or how the possibility of empty-running movements can be inserted into it, e.g. how to connect train k to train n by empty-running from C to D. If it is forbidden to create new empty-running trains, it is possible to regard existing empty-running trains from historical records as passenger trains and insert them as trip arcs in the flow graph.

3.5.2 Connection-arc graph representation

The second modelling strategy for the network-level framework would be the node-arc construction method commonly seem in many vehicle scheduling problems, which will be referred to as connection-arc DAG representation here. It is the modelling strategy used in Cacchiani et al. [15]. In such a representation, there is a source node and a sink node or sometimes there can be multiple source and sink nodes. Apart from the source/sink node, each of the other nodes stands for a train service. There are two kinds of arcs in the connection-arc graph: A sign-on/-off arc links a train node from/to the source/sink respectively, and a connection-arc links two train nodes. Moreover, if a connection-arc implies a turnround action with an additional empty-running movement from the arrival location of the previous train to the departure location of the next train, then it also has an attribute of empty-running. The connection-arc graphs can be either cyclic [18] or acyclic [15] depending on different purposes. A connection-arc based acyclic directed graph associated with the timetable in Table 3.5 is illustrated in Figure 3.6.

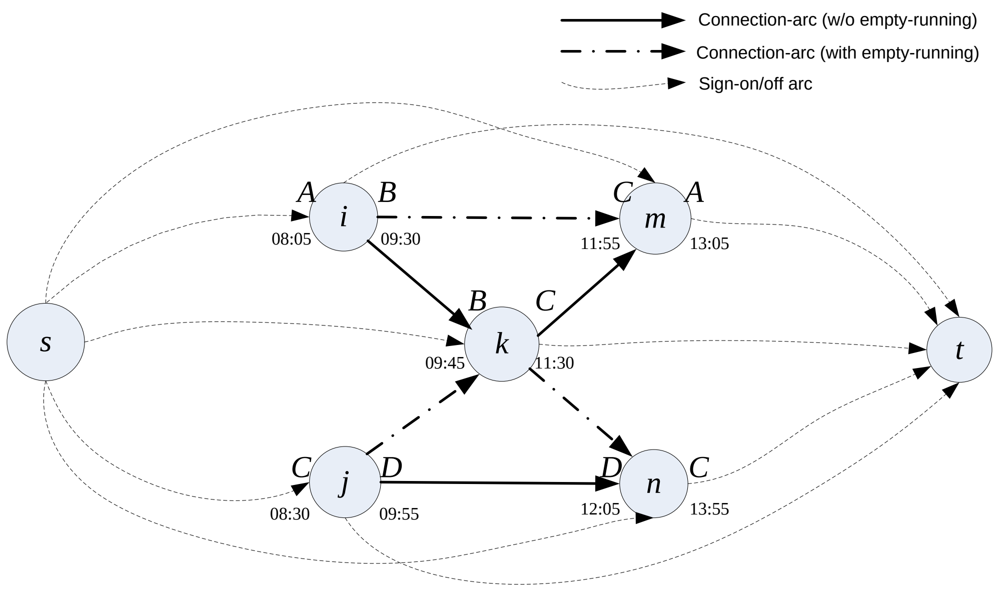


Figure 3.6: The connection-arc graph corresponding to Table 3.5

The connection-arc graph representation is more flexible compared with the flow graph representation since it does not directly related to any special network topology, timetable patterns or pre-sequenced connections. Therefore, it is more suitable to accommodate additional variables and/or constraints in the associated mathematical formulations to achieve sophisticated and complex real-world constraints. On the other hand, the flexibility may make the resulting mathematical formulations more difficult to solve than the ones derived from a flow graph representation.

3.5.3 Combination-arc graph representation

In the above connection-arc graph representation, each connection-arc only represents the relation between two train nodes whether or not they can be served by the same unit consecutively. Usually in a multicommodity flow framework, each connection-arc will be further associated with a set of arc variables of all permitted commodities (types) at this arc. The specific unit combination information within this connection is up to the value assignment on the arc variables of different types at this connection.

A novel modelling strategy named *combination-arc graph representation*, which is still at an early stage of development, will be proposed in § 8.1. This representation has some similarities with the connection-arc graph but takes a rather different approach to creating and defining arcs such that each arc stands for a possible unit combination during the turnround action from one train to another train. Figure 3.7 gives a DAG of the combination-arc graph representation corresponding to the trains in Table 3.5 with the sign-on/-off arcs omitted. Note that each combination-arc stands for a possible unit combination connecting two consecutive trains. By assigning each combination arc a single binary variable, the possible flow distributions can be captured by an ILP formulation. This representation may often use more arcs than the connection-arc representation but in certain circumstances its unique advantage over the connection-arc representation will be important. Details of the combination-arc graph representation can be found in § 8.1.

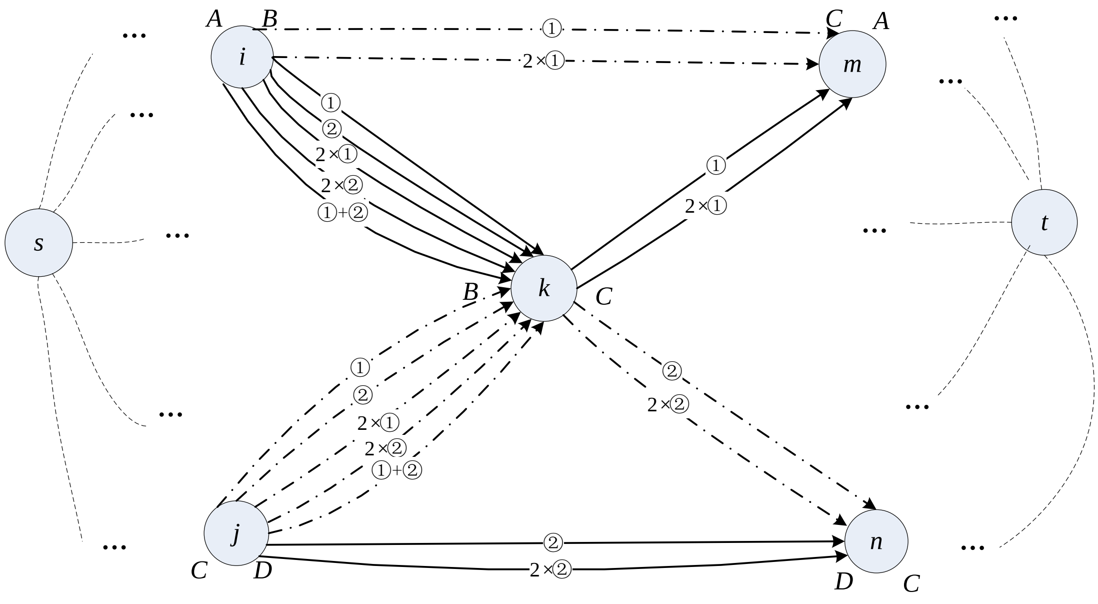


Figure 3.7: The combination-arc DAG corresponding to Table 3.5

3.6 A network-level framework based on the connection-arc graph representation

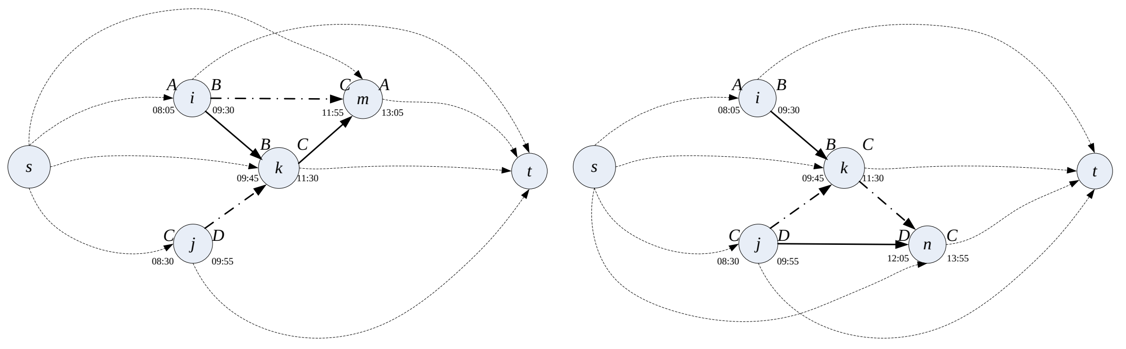
We propose a network-level modelling strategy that belongs to the category of the so-called connection-arc graph representation as described in § 3.5.2. Compared with the framework used by Cacchiani et al. [15] which is also connection-arc graph based, this strategy can realise more crucial and practical real-world requirements by additional block-arc variables and solution methods such as customised train-family and banned location branching.

This network-level framework is based on a directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. We define the node set $\mathcal{N} = N \cup \{s, t\}$, where N is the set of *train nodes* representing train services in the timetable, and s and t are the *source* and *sink* node. The arc set is defined as $\mathcal{A} = A \cup A_0$, where A is the *connection-arc* set and A_0 the *sign-on/off arc* set. A connection-arc $a \in A$ links two train nodes i and j ($a = (i, j),\ i, j \in N$), representing a potential turnround connection such that after serving train i a unit can continue to serve train j as its next task. A sign-on arc $(s, j) \in A_0$ starts from s and ends at a train node $j \in N$; a sign-off arc $(j, t) \in A_0$ starts from a train node $j \in N$ and ends at t. Generally all train nodes have a sign-on arc and a sign-off arc. We use $\delta_-(j)$ and $\delta_+(j)$ to denote all arcs that terminate at / originate from node j respectively. Finally, an s-t path $p \in P$ in $\mathcal{G}$ represents a sequenced daily workload (the train nodes in the path) for a unit.

For the fleet, we denote K the set of all unit types, corresponding to the commodities in a multicommodity flow framework. As for type-route compatibility, type-graphs $\mathcal{G}^k$ representing routes each type $k \in K$ can serve are constructed based on $\mathcal{G}$, as well as the components of $\mathcal{G}$, e.g. P^k and $\mathcal{A}^k$ refers to the set of all paths and arcs in type-graph $\mathcal{G}^k$ respectively. Figure 3.8 shows two type-graphs corresponding to type ① and ② respectively derived from the DAG in Figure 3.6 based on the instance in Table 3.5. We call an arc in a type-graph $\mathcal{G}^k, k \in K$ a *type-arc*; on the other hand, we call an arc in the original graph $\mathcal{G}$ a *block-arc*, since when used it represents a unit block of coupled units of the same or different types that are not split during the coupling/decoupling/shunting/empty-running process between two trains.

Based on the above network-level framework, an intuitive way in deriving an ILP formulation will be an arc formulation based on integer variables associated with the connection-arcs. Ideally to include as many requirements as possible, three kinds of decision variables will be set, namely the *type-arc variables*, *block-arc variables* and *overnight balance variables* defined as the following.

- Type-arc variables $x_a \in \mathbb{Z}_+, \forall a \in \mathcal{A}^k, \forall k \in K$: indicating the flow amount (number of units) in type-arc a of type-graph $\mathcal{G}^k$.
- Block-arc variables $y_a \in \{0, 1\}, \forall a \in \mathcal{A}$: indicating whether block-arc $a \in \mathcal{A}$ is used.
- Overnight balance variables $z^+_{d,k}, z^-_{d,k} \in \mathbb{R}_+, \forall d \in D, k \in K$: calculate the inventory



(a) Type-graph $\mathcal{G}^{\textcircled{1}}$ for type $\textcircled{1}$ (b) Type-graph $\mathcal{G}^{\textcircled{2}}$ for type $\textcircled{2}$

Figure 3.8: Type-graphs with respect to type $\textcircled{1}$ and $\textcircled{2}$ derived from the DAG in Figure 3.6

imbalance at each depot $d \in D$ for each type $k \in K$.

The introduction of the block-arc variables is to satisfy the three requirements mentioned in § 3.1, i.e. time allowance validity involving coupling / decoupling, banned / restricted locations for coupling / decoupling and eliminating redundant/excessive coupling/decoupling. It is understandable that all the above requirements are relevant with coupling/decoupling operations, because to get the detailed information of those operations, the model needs to focus on unit blocks reflected by block-arc variables, rather than individual units reflected by type-arc variables. The overnight balance variables are used to keep unit overnight balance at each depot as much as possible. Note that as mentioned in § 3.1.8, it is not mandatory to keep overnight inventory balanced at an early stage of unit diagramming. When it is assumed to be unnecessary to include this issue at this stage, one may remove all variables and constraints related to $z^{+}_{d,k}, z^{-}_{d,k}$.

3.6.1 A connection-arc formulation (AF_0)

We will introduce the mathematical formulations that are going to be used in this thesis by first considering a connection-arc ILP formulation with constraints constructed in a relatively more "intuitive" way based on the network-level framework as an integer fixed-charge multicommodity flow problem with side constraints. Here we first assume that for each train there is a shared coupling upper bound for whatever the unit combination is (which is however not true for many real-world cases). The formulation can then be constructed in a more intuitive form, as follows

$$(AF_0) \quad \min \quad C_1 \sum_{k \in K} \sum_{a \in \mathcal{A}^k} c_a x_a + C_2 \sum_{a \in \mathcal{A}} y_a + C_3 \sum_{d \in D} \sum_{k \in K} \left(z^{+}_{d,k} + z^{-}_{d,k} \right) \tag{3.1}$$

$$
\begin{aligned}
&\sum_{a\in\delta_+^k(s)} x_a && \le b_0^k, && \forall k \in K && (3.2)\\
&\sum_{a\in\delta_-^k(j)} x_a - \sum_{a\in\delta_+^k(j)} x_a && = 0, && \forall j \in N, \forall k \in K && (3.3)\\
&\sum_{k\in K_j}\sum_{a\in\delta_-^k(j)} q^k x_a && \ge r_j, && \forall j \in N && (3.4)\\
&\sum_{k\in K_j}\sum_{a\in\delta_-^k(j)} v^k x_a && \le u_j, && \forall j \in N && (3.5)\\
&\sum_{k\in K: a'=a(k)} x_{a'} && \le u_a y_a, && \forall a \in \mathcal{A} && (3.6)\\
&\tau^D_{\mathrm{arr}(i)}\left(\sum_{a\in\delta_+(i)} y_a - 1\right) + \tau^C_{\mathrm{dep}(j)}\left(\sum_{a\in\delta_-(j)} y_a - 1\right) && \le e_{ij}, && \forall (i,j) \in A^* && (3.7)\\
&\sum_{a\in\delta_-(j)} y_a && = 1, && \forall j \in N_B^- && (3.8)\\
&\sum_{a\in\delta_+(j)} y_a && = 1, && \forall j \in N_B^+ && (3.9)\\
&z^+_{d,k} - z^-_{d,k} - \left(\sum_{a\in(A_d^k)^{\mathrm{on}}} x_a - \sum_{a\in(A_d^k)^{\mathrm{off}}} x_a\right) && = 0, && \forall d \in D, \forall k \in K && (3.10)\\
&x_a && \in \mathbb{Z}_+, && \forall a \in \mathcal{A}^k, \forall k \in K && (3.11)\\
&y_a && \in \{0,1\}, && \forall a \in \mathcal{A}; && (3.12)\\
&z^+_{d,k}, z^-_{d,k} && \in \mathbb{R}_+, && \forall d \in D, \forall k \in K && (3.13)
\end{aligned}
$$

In the above connection-arc formulation (AF_0) (3.1) – (3.13), Constraints (3.2) ensure the number of deployed units for each type $k \in K$ is within its fleet size limit b_0^k. Constraints (3.3) are to force the flow conservation balance at each train node. Constraints (3.4) are to satisfy the passengers' demands where q^k is the unit capacity of type k and r_j is the passenger demand for train j, both measured in number of seats. v^k is the number of cars for unit type k, and Constraints (3.5) require, for each train, that the total car number is within the coupling upper bound, which is assumed to be of the same value u_j for all possible unit combinations in formulation (AF_0). The situation involving combination-specific coupling upper bounds as described in § 3.1.5.2 will be discussed later. Constraints (3.6) are the so-called forcing constraints, which together with the second term in the objective, will ensure the right values for the binary block-arc variables such that $y_a = \begin{cases} 1, & \sum_{k\in K_a: a'=a(k)} x_{a'} > 0, \\ 0, & \sum_{k\in K_a: a'=a(k)} x_{a'} = 0, \end{cases}$ where $a(k)$ stands for the type-arc of type k associated with block-arc a. Although meant to be a sufficiently "large" number, u_a for arc $a = (i,j)$ would be ideally set as small as possible, i.e. to be the possibly largest number of units coupled at i and j.

In the following we will explain the rest of (AF_0) that is not covered in the above paragraph.

3.6.1.1 The objective function

The objective (3.1) defines the optimisation targets to be achieved, where C_1, C_2 and C_2 are the weights for the three major terms and generally set so that $C_1 > C_2$ and $C_1 > C_3$ by one or two degrees of magnitude since the first term contains the value on the fleet size. It is up to the user's wish to set a relation between C_2 and C_3 reflecting his/her preferences. In the first term, c_a is the cost for type-arc a, which can be further elaborated into a weighted linear combination of several sub-costs to achieve multiple effects and/or preferences with different sub-weights $V_1, \ldots, V_6$:

$$c_a^k = V_1\delta_a + V_2 m_{j(a)}^k + V_3 e_a - V_4 g_a^k - V_5 \pi_{j(a)}^k - V_6 \pi_a^k. \tag{3.14}$$

The above cost of type-arc c_a is re-written as c_a^k to emphasise the type k implied by type-arc a. In (3.14), the first term minimises the total number of units used, where $\delta_a = 0$ if a is a connection-arc or a sign-off arc and $\delta_a = 1$ if a is a sign-on arc. The second term minimises the total passenger train unit-mileage, where $j(a) \in N$ is the train node as the endpoint node of arc a and $m_{j(a)}^k$ is the mileage of train $j(a)$ used by a single unit of type k. The third term minimises the total number of empty-running units where e_a is the empty-running cost of arc a ($e_a = 0$ if a contains no empty-running). The fourth term encourages the insertion of long gaps between two trains during which maintenance can be done. Parameter $g_a^k = 0$ if arc a does not imply a long gap or this gap is not suitable for type k; $g_a^k > 0$ on the other hand. There are also preferences among positive values of g_a^k as some long gaps are more suitable for types k than others. The fifth term encourages type-train (route) preferences, where $\pi_{j(a)}^k \geq 0$ is a preference weight for a type-train pair $(k, j(a))$: the higher the more suitable. The sixth term encourages sign-on/off arc preferences (usually related to home depots) for each type of units, where π_a^k stands for a preference parameter for a type/sign arc pair (k, a): the higher the more suitable. If a certain type k is not suitable for a sign-on/off arc, this arc will not have been inserted into $\mathcal{G}^k$ in the first place. In practice the above six terms can be used selectively according to the users' specific needs. Finally the sub-weights $V_1, \ldots, V_6$ should be appropriately set to ensure that $c_a^k > 0$ for all type-arcs. If this cannot be realised, then the last three terms should be changed to positive values with the definition of $g_a^k, \pi_{j(a)}^k, \pi_a^k$ modified accordingly.

The second term in (3.1) is a cost associated with the total number of used block-arcs, which penalises coupling/decoupling activities, thus eliminating unnecessary ones; it also partly drives y_a to desired binary values. We will further explain the rationale of it in § 3.6.1.4. The third term is the cost of the unit overnight imbalance at each depot for

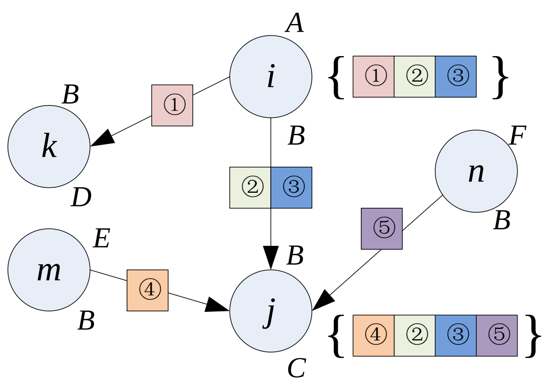


Figure 3.9: Coupling/decoupling time calculation at (i, j)

normal weekdays. Its details will be discussed in § 3.6.2.

3.6.1.2 Time allowances involving coupling/decoupling

Figure 3.9 shows an example of time allowance calculation at a connection-arc $(i, j) \in A$. There are five train nodes i, j, k, m, n, four *used* connection-arcs $(i, j), (i, k), (m, j), (n, j)$ and five train units ① to ⑤. Train i is from location A to B (and so on for other trains) and the five units (can be of the same or different types) are all compatible. First, units ①,②,③ are coupled together to serve train i. Then at the arrival station B of i, unit ① is detached from the block and continues to serve train k while unit ② and ③ remain together at B. Meanwhile unit ④ arrives at B from E by train m and unit ⑤ arrives at B from F by train n, and both are attached to the unit block ②+③ to form a new block ④+(②+③)+⑤ to serve train j later. Note that as explained in § 3.1, the permutation of coupled units (e.g. ④+(②+③)+⑤ against (②+③)+④+⑤ etc.) should also be carefully considered to avoid potential inoperable plans, which is left to post-processing in the two-phase approach. We note the fact that impractical operations such as units ② and ③ shortly decouple with each other and then couple with ④ and ⑤ to form a permutation "②+⑤+③+④" would be very unlikely to happen in practice. We also take the assumption that it takes $n - 1$ times of single coupling/decoupling operation when n blocks are coupled/decoupled, which is realistic since coupling/decoupling three or more units simultaneously will generally violate safety regulations. Note that we here only calculate the time taken solely by coupling/decoupling operations, because the time consumed by other relevant activities like shunting or empty-running has been already included into the connection information and subtracted from the time gap (as they can be determined a priori). Come back to the Figure's instance, there are one decoupling (①|②+③) and two coupling (④ $\rightarrow$ ②+③, ②+③ $\leftarrow$ ⑤) operations occurring at station B. Assuming that the time gap of connection (i, j) is 12 minutes, and that each single coupling/decoupling operation takes 3 minutes, and there has to be at least 5 minutes buffering (turnround) time, and no shunting is needed (and no empty-runing), then the turnround time is $3 \times 1 + 3 \times 2 + 0$ which leaves only 3 minutes idle time, which is less than the buffering time required. The conclusion is that there is not sufficient time to perform such a coupling/decoupling operation over connection (i, j).

To model the above calculation by linear constraints, block-arc variables $y_a, a \in \mathcal{A}$ are used. Let $\tau^D_{\text{arr}(i)}$ be the time for a single decoupling operation at the arrival location of train i, $\tau^C_{\text{dep}(j)}$ be the time for a single coupling operation at the departure location of train j, and e_{ij} be the time slack between i and j modified with potential buffering/shunting/empty-running time (generally all of which can be determined in advance by the two trains i, j). Constraints (3.7) are used to ensure the turnround time in between i and j not to be violated by potential coupling/decoupling there. It is worth mentioning that often if long distance empty-running (i.e. not within the same station) is implied by (i, j), then the time gap there should be sufficiently large such that the time consumed by coupling/decoupling can be ignored. Therefore, Constraints (3.7) are mainly applied to connection arcs without long distance empty-running. Moreover, Constraints (3.7) are not needed for any sign-on/off arcs; it is also not necessary to require it for all connection-arcs, but only those short ones having a potential of violating turnround time allowance, which we denote as a set A^*. Let M_i and M_j be the possible maximum number of coupled units for train i and j respectively. If $e_{ij} \geq \tau^D_{\text{arr}(i)}(M_i - 1) + \tau^C_{\text{dep}(j)}(M_j - 1)$, then $(i, j) \notin A^*$; otherwise $(i, j) \in A^*$.

3.6.1.3 Banned/restricted locations for coupling/decoupling

The rules regarding banned/restricted locations for coupling/decoupling are explained in § 3.1.5.3. Most of the conditions restricted by those rules can be sufficiently determined by relevant train nodes and arcs containing information on routes, time bands, train directions, and so on. However if relevant trains have not been assigned to specific platforms before unit scheduling, then the platform restrictions on coupling/decoupling cannot be applied due to the lack of information. For the instances we have been working on, not all trains have had their platforms fixed before the train unit scheduling stage. Therefore the banned location information in our computational experiments at the moment does not contain the cases where certain platforms are banned for coupling/decoupling since the trains to be assigned to them are unknown. In fact it is possible to temporarily ignore this restriction at the network-level framework and to leave it to the subsequent post-processing stage where the platform assignment as a local issue can be accomplished.

To forbid those potential rule-breaking coupling / decoupling activities at banned / restricted locations by linear constraints, we define $N_B^- \subseteq N$ as a set of trains whose departure locations are banned for coupling, and $N_B^+ \subseteq N$ for trains with decoupling banned arrival locations. Note that this is sufficient to also include some special locations where only one of the coupling/decoupling operations is banned. From Figure 3.9, it can be seen that for a train j, if only *one* incoming block-arc in $\delta_-(j)$ is used, then there is no coupling at its departure location, and if only *one* outgoing block-arc in $\delta_+(j)$ is used, then there is no decoupling at its arrival location. Thus it is possible to realise the banned location requirements by Constraints (3.8) and (3.9). Note that when arcs with long distance empty-running are involved in $\delta_+(j), \delta_-(j)$, empty-running train nodes may need to be explicitly inserted into the DAG to ensure some coupling/decoupling possibilities will not be excluded by Constraints (3.8) and (3.9). For example, two units can couple at another location allowing coupling and empty-running together to the departure location of j where coupling is banned though. In this context, some empty-running train nodes overlapping in time and locations may also need to be merged into the same node. Another way to preserve those possibilities is to exclude some empty-running arcs from $\delta_+(j), \delta_-(j)$ in Constraints (3.8) and (3.9) on an ad hoc basis, which is especially convenient when only the existing ECS movements in the manual schedules are allowed in the optimisation model.

Alternatively, there is also an option to exclude all rule-braking coupling/decoupling activities at banned/restricted locations by branching in the solution process for the relevant ILP such that no Constraints (3.8) and (3.9) are needed. This will be discussed in detail in § 5.3.1.3.

3.6.1.4 Redundant/excessive coupling/decoupling

As illustrated in § 3.6.1.2 and 3.6.1.3, block-arcs can be used to measure the number of coupling/decoupling operations. If we sum up all the coupling/decoupling operation numbers for both locations of all train nodes,

$$\sum_{j\in N}\left[\left(\sum_{a\in\delta_-(j)} y_a - 1\right) + \left(\sum_{a\in\delta_+(j)} y_a - 1\right)\right] = 2\left(\sum_{a\in\mathcal{A}} y_a - |N|\right), \tag{3.15}$$

we get the total number of coupling/decoupling operations during the whole day of the entire rail network, so by minimising $\sum_{a\in\mathcal{A}} y_a$, we would minimise the number of coupling/decoupling in the day. In fact, the term $\sum_{a\in\mathcal{A}} y_a$ refers to the number of all used block-arcs. Since each used block-arc represents a unit block formed by coupled units, then we have got another interpretation of (3.15): to use as few blocks as possible during the day, or equivalently, to keep the units together as much as possible.

The above is in fact a fairly reasonable and even basic requirement for a schedule of good quality, which has been somehow ignored in previous literature. We give an example below to show its importance.

In Figure 3.10, a local comparison of two different schedules is given involving four trains i, j, m, n. Suppose either of train i, j can be linked to either of train m, n at the common location B, and all of the trains can be served by any of the four units ① to

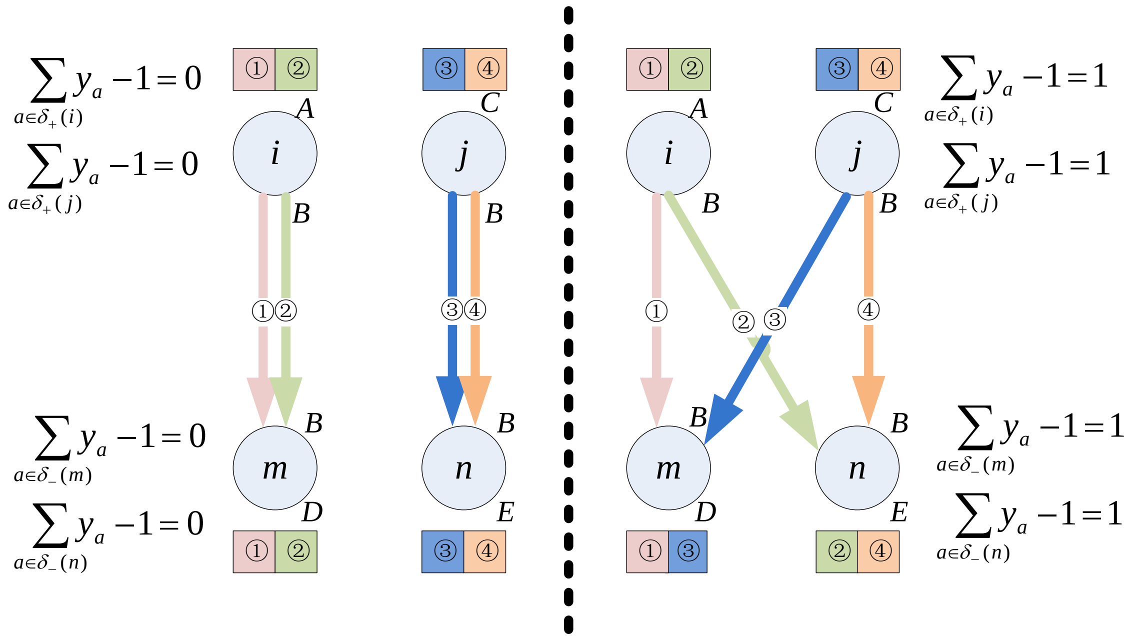

$\sum_{a\in\delta_+(i)} y_a - 1 = 0$
$\sum_{a\in\delta_+(j)} y_a - 1 = 0$
$\sum_{a\in\delta_-(m)} y_a - 1 = 0$
$\sum_{a\in\delta_-(n)} y_a - 1 = 0$
$\sum_{a\in\delta_+(i)} y_a - 1 = 1$
$\sum_{a\in\delta_+(j)} y_a - 1 = 1$
$\sum_{a\in\delta_-(m)} y_a - 1 = 1$
$\sum_{a\in\delta_-(n)} y_a - 1 = 1$
i
j
m
n
A
B
C
D
E
①②
③④
①
②
③
④
①③
②④

Figure 3.10: Redundant/excessive coupling/decoupling

④. The left shows a reasonable arrangement such that unit block ①+② (or ③+④) continues to serve train m (or n) after finishing train i (or j), while the right illustrates a clumsy "swapping" where block ①+② (or ③+④) decouples first and then each single unit re-attaches to one single unit from the other block. Consequently, the left involves no coupling/decoupling at all but the right has two decoupling and two coupling operations. Interestingly, while the preference on the left cannot be more obvious for a human scheduler's point of view, the automatic optimisation process based on ILP simply cannot distinguish the two unless the use of block-arcs is also penalised as in (3.15). It is clear that on the left two block-arcs ((i, m) and (j, n)) are used while on the right four block-arcs ($(i, m), (i, n), (j, m), (j, n)$) are used; however the arc variables themselves cannot achieve this measurement as the flow amounts in both cases are the same of 4.

The above is a very simplified instance with only four nodes. As shown from our computational experiments based on real-world dataset, if the use of block-arcs is not penalised, the overall consequence for the entire DAG will be much worse, resulting in a large amount of unnecessary coupling/decoupling activities during the day such that the solutions are very unlikely to be acceptable by train operators. Moreover, as mentioned before, excessive coupling/decoupling is a major source of potential unit blockage in the network-level results, making the subsequent post-processing tasks more difficult to solve. Therefore, it is very important to eliminate excessive/redundant coupling/decoupling activities, either in the network-level stage or in post-processing.

3.6.2 Overnight balancing on normal weekdays

In § 3.1.8, it has been mentioned that the requirements on unit overnight balancing is usually not involved in the main part of unit scheduling but as a subsequent task to be fulfilled after raw unit diagrams have been produced. Nevertheless, sometimes the practitioners would ideally like raw unit schedules to have some characteristics that may ease the subsequent planning for cycles and maintenance. For instance, we were told that ideally it is better to ensure overnight balance for the normal weekdays' (usually Tuesday, Wednesday and Thursday) schedules. Here overnight balance means the number of units per type at each depot at the beginning of the day is the same as the end of the day.

To include the ILP with an ability in handling the overnight balancing on normal weekdays, we assume that each station has a unique depot designated to it. Let $d \in D$ be the available depots in the rail network, each having a set of associated sign-on arcs $(A_d^k)^{\text{on}} \subset A_0^k, \forall k \in K$ which link to the trains depart from stations belonging to d, and a set of associated sign-off arcs $(A_d^k)^{\text{off}} \subset A_0^k, \forall k \in K$ which link to the trains arrive at stations belonging to d. In a connection-arc based formulation, the following constraints can be applied to keep the overnight balance at each depot, similar as what has been used in [15],

$$\sum_{a \in (A_d^k)^{\text{on}}} x_a = \sum_{a \in (A_d^k)^{\text{off}}} x_a, \quad \forall d \in D, \forall k \in K. \tag{3.16}$$

Here we will use a slightly different strategy that gives more flexibility to the scheduler, as it is found that the strict balancing imposed by (3.16) may sometimes undesirably increase the fleet size. In this case, the scheduler may prefer having less number of deployed units even if the balance is not kept at some depots because they can simply add extra empty-running trains to desired depots at night after all services are finished to recover the imbalance, which costs much less than using additional train units.

Define new decision variables $z_{d,k} \in \mathbb{R}, \forall d \in D, \forall k \in K$ as the unit imbalance amount for type k at depot d, that is

$$z_{d,k} = \sum_{a \in (A_d^k)^{\text{on}}} x_a - \sum_{a \in (A_d^k)^{\text{off}}} x_a, \quad \forall d \in D, \forall k \in K. \tag{3.17}$$

Then a new term can be added to the objective to minimise the summed absolute values of imbalances per depot per type, as "min $\cdots + C_3 \sum_{d \in D} \sum_{k \in K} |z_{d,k}|$". The weight C_3 controls the flexibility how much importance should be given to overnight balancing compared with minimising the fleet size and the number of coupling/decoupling operations. When it is required to keep an overnight balance without any additional after-work empty-running, then one can set $C_3 = +\infty$ to realise it.

To overcome the nonlinearity caused by the absolute value $|z_{d,k}|$ in the objective, we use a common practice by replacing $z_{d,k}$ with two positive variables $z_{d,k}^+, z_{d,k}^- \geq 0$ which gives

$$z_{d,k} = z^+_{d,k} - z^-_{d,k}, \tag{3.18}$$

such that (3.17) and (3.18) yield (3.10) in (AF_0), and substituting the absolute value term by $|z_{d,k}| = z^+_{d,k} + z^-_{d,k}$ in the objective. An alternative approach would be to change the problem into a (linearly constrained) quadratic program by minimising " $\cdots + C_3 \sum_{d\in D} \sum_{k\in K} z^2_{d,k}$" which is also a well-studied optimisation problem. Since the sizes of D and K are relatively small, the former method by using $z^+_{d,k}, z^-_{d,k}$ may be more advantageous as it will not increase the problem size much while keeping its LP structure.

To sum up, the third term in the Objective (3.1) in conjunction with Constraints (3.10) is used to penalise the unit overnight imbalance if one would like to also have some consideration on overnight balance, which is not critically needed at the network-level stage though. Note that the integrality requirement on $z^+_{d,k}$ and $z^-_{d,k}$ should be automatically satisfied due to Constraints (3.10) and the fact that all x_a's are integral.

Chapter 4

Further ILP formulations

Formulation (AF_0) is intuitive and straightforward. Nevertheless, it has a drawback that it is incapable of handling the combination-specific coupling upper bounds as described in § 3.1.5.2, which are commonly seen in the fleets of Southern Railway and ScotRail. The first part of this chapter will be devoted to the remedies for this problem, where two methods are going to be presented, i.e. the explicit multi-restrictive constraints and the train convex hull methods.

Solving the network-level ILP including many requirements may have some disadvantages especially when the problem scale is medium or large. Therefore, some of the requirements are removed from the network-level framework to the station-level post-processing, leading to a reduced network-level formulation suitable for medium/large sized instances. The rationale behind this relaxation will also be justified.

Finally the Dantzig-Wolfe decomposition method will be applied to the connection-arc formulations, leading to path formulations that are more suitable for column generation based methods such as branch-and-price. We will leave the details of the branch-and-price solver to Chapter 5.

4.1 Combination-specific upper bounds

4.1.1 Multiple restrictive factors and explicit multi-restrictive constraints

In formulation (AF_0), the coupling upper bound restriction upon each train is satisfied by Constraints (3.5), because we have assumed that for each train j there is a shared bound u_j for whatever the unit combination is. It is the case in the instances in Cacchiani et al. [15, 17, 18] where $v^k = 1, u_j = 2, \forall k \in K, j \in N$. This assumption, however, is not

Table 4.1: Two coupling upper bounds for Family SN.II (c455/8 and c456/0)

Combination	UB in car#	UB in unit#
c455/8 (4-car)	8 (2×c455/8)	2
c456/0 (2-car)	6 (3×c456/0)	3
mixed	8 (c455/8+2×c456/0)	3

true for some instances at Southern Railway and ScotRail, as shown in Table 3.2, where one can see that for some unit families in the fleets of both operators the coupling upper bounds are clearly combination-specific.

Often this kind of nonconformity on coupling upper bounds is caused by multiple restrictive factors on how train units can be coupled for a train. For example, an operational regulation requires the number of coupled units is less than a certain number, an engineering rule requires the number of coupled cars is less than a certain number, and a platform length requires the total length of a coupled block is less than certain metres, and so forth. Those requirements combined together can result in very complex combination-specific upper bounds.

When the number of restrictive factors is small and all the candidate units available for the train are from a single family, it is possible to explicitly use multiple coupling upper bound constraints with respect to each factor to achieve the desired restrictions. For instance, for all families in the Southern Railway fleet, two restrictive factors—unit number and car number—have been identified such that it is sufficient to set two sets of coupling upper bound constraints according to them to achieve the targets.

Suppose a connection-arc formulation is used, and let N' be the set of trains such that only a single family of unit is available, then we have the following explicit multi-restrictive constraints

$$\sum_{k\in K_j}\sum_{a\in\delta_-^k(j)} v^k x_a \le u_j, \quad \forall j \in N'; \tag{4.1}$$

$$\sum_{k\in K_j}\sum_{a\in\delta_-^k(j)} x_a \le \tilde{u}_j, \quad \forall j \in N', \tag{4.2}$$

where v^k is the number of cars for unit type k, u_j is the coupling upper bound in the number of cars, and $\tilde{u}_j$ is the coupling upper bound in the number of units.

We will use c455/8 and c456/0 of unit family SN.II from the Southern Railway fleet to illustrate the possibility in using the above explicit constraints. Table 4.1 shows the coupling upper bounds in car number and unit number for this family. In this case, for the trains that are solely served by Family SN.II, setting $u_j = 8$ and $\tilde{u}_j = 3$ would be sufficient.

On the other hand, as discussed in § 3.1.5, it is possible for a train to have candidate units belonging to different families. In a solution given by the connection-arc graph framework, a train may be potentially served by more than one family, which violates the type-type compatibility requirement. Therefore, there should be some way to eliminate such possibilities for those trains. Since the upper bounds are generally defined within each family, one of the approaches is to combine the type-type compatibility issue with the multiple restrictive factors among available families. In this case, additional decision variables and constraints have to be introduced to indicate whether a family at a train is used or not and only one family is allowed at a train. Let $\zeta_j^\varphi \in \{0,1\}$ be the train-family decision variables for the trains having more than one family of unit available (the set of such trains are denoted as N'') such that $\zeta_j^\varphi = 0$ if family φ does not cover train j and $\zeta_j^\varphi = 1$ otherwise. Let u_j^φ and $\tilde{u}_j^\varphi$ be the upper bounds in car number and unit number for family φ at train j respectively and Φ_j be the set of families available at j. Then we can use the following constraints to cater for the trains in N'':

$$\sum_{k\in\varphi}\sum_{a\in\delta_-^k(j)} v^k x_a \le u_j^\varphi \zeta_j^\varphi, \quad \forall j \in N'', \forall\varphi \in \Phi_j; \tag{4.3}$$

$$\sum_{k\in\varphi}\sum_{a\in\delta_-^k(j)} x_a \le \tilde{u}_j^\varphi \zeta_j^\varphi, \quad \forall j \in N'', \forall\varphi \in \Phi_j; \tag{4.4}$$

$$\sum_{\varphi\in\Phi_j} \zeta_j^\varphi = 1, \qquad \forall j \in N''. \tag{4.5}$$

We will call the ILP formulation derived by replacing Constraints (3.4) and (3.5) by (4.1)–(4.2) and (4.3)–(4.5) the "extended arc formulation" (AF_1). Note that although the above strategy by using additional train-family decision variables can realise the unit type-type compatibility requirement, it was observed later that the train-family branching method to be introduced in § 5.3.1.1 would be more preferable than having additional variables and constraints as in (4.3)–(4.5). Nevertheless (4.1)–(4.2) have been applied to the computational experiments performed at the early stage of this research for Southern Railway as described in § 6.1.

When the number of restrictive factors gets large, the above explicit method may bring too much burden to the ILP with lots of variables and constraints. Moreover, notice that constraints in an ILP have a *conjunctive* relation which may make the above explicit method invalid. We will use an artificially created example to show this possibility. Suppose when c455/8 and c456/0 are coupled, the only configuration allowed is c455/8+c456/0, which gives an upper bound in car number of 6 and unit number of 2 as shown in Table 4.2. This gives no available value for u_j and $\tilde{u}_j$ since assigning $u_j = 8$ and $\tilde{u}_j = 3$ may yield a wrong configuration for mixed cases involving both types while assigning $u_j = 6$ and $\tilde{u}_j = 2$ may exclude the possibilities in having valid combinations such as 2×c455/8 and 3×c456/0.

The phenomenon shown in the artificial example in Table 4.2 is often due to the fact that there might be further factors behind. As long as all the factors are found and included, the upper bound requirements will likely be still satisfied. However, generalising

Table 4.2: An artificial example for the upper bounds on Family SN.II (c455/8 and c456/0)

Combination	UB in car#	UB in unit#
c455/8 (4-car)	8 (2×c455/8)	2
c456/0 (2-car)	6 (3×c456/0)	3
mixed (artificial)	6 (c455/8+c456/0)	2

such factors may require many more constraints to be added to the ILP. In addition, different operators may have different groups of factors such that the explicit constraints are not sufficiently robust. Moreover, if an operator has some even more complex rules not only relying on quantitatively restrictive limits, linear inequalities may not be able to handle them. For instance, an operator may have a rule stating that for safety and/or engineering reasons, the maximum number of coupled units has to be reduced when units from different types are coupled, as what happens in Table 4.2. The remedy for such more complex combination-specific upper bounds will be given in § 4.1.2.

4.1.2 Train convex hulls and a tightened arc formulation (AF_2)

A simple and somewhat universal method for satisfying the combination-specific coupling upper bound is proposed. For each train, the method straightforwardly enumerates all possible unit combinations according to the combination-specific rules and the passenger demand, computes an associated local convex hull and converts it to convex hull constraints to expand the ability of formulation (AF_0) by replacing Constraints (3.4) and (3.5) with the new constraints. Note that at the same time, the lower bound given by the LP relaxation of the ILP formulation will often be tightened, which is desirable for solving an ILP.

In the railway vehicle scheduling literature, similar ideas in constructing local convex hulls or set dominants with respect to individual trains can be found in [3, 15, 29, 70, 84] etc. However, in the above literature, local convex hulls are solely used to strengthen LP relaxation. In the method to be presented here, besides the same purpose in tightening lower bounds, more importantly local convex hulls are used to satisfy the complex coupling upper bound requirements that are difficult to be realised by linear constraints. It can also remove part of the type-type incompatible combinations.

Let K_j be the set of permitted types for train $j \in N$, and $w^j = (w_1^j, w_2^j, \cdots, w_{|K_j|}^j)^T \in \mathbb{Z}_+^{K_j}$ be a unit combination vector for j, where w_k^j stands for the number of units of type k used for j. We define a *unit combination set* by enumerating all valid unit combination for each train:

$$W_j := \left\{ w^j \in \mathbb{Z}_+^{K_j} \middle| \forall w^j : \text{a valid unit combination for train } j \right\}, \forall j \in N, \tag{4.6}$$

such that

(i) $\sum_{k \in K_j} q_k w_k^j \geq r_j$, where q_k is the capacity of unit type k and r_j is the passenger demand requirement, both measured in numbers of seats.

(ii) $\sum_{k \in K_j} v_k w_k^j \leq u(w^j), \exists w^j \in W_j$ being used, where v_k is the number of cars for unit type k and $u(w^j)$ is the coupling upper bound in number of cars allowed for combination w^j.

(iii) the used types $(k : w_k^j > 0)$ are compatible.

To justify the advantage of this enumeration approach, note that while it is easy to satisfy (i) by linear constraints, it is generally very difficult or impractical for (ii) and (iii) in the same way. For instance, to use the constraints in (ii) directly, the relation among the $|W_j|$ expressions has to be *disjunctive*, i.e. one and only one of them should be present if and only if its corresponding combination w^j is chosen by the schedule, rather than conjunctive like the relations that constraints have in an LP. This is because it is possible for some constraints of different combinations to be contradictory with each other. Moreover, the knapsack constraints from (i) and (ii) can also make the LP relaxation very weak if used directly, as the similar observations reported in [15] and our own experiments. Finally, this enumeration generally suits any validity rules imposed for a train, not limited to the above three. Apart from fully satisfying the passenger demands and coupling upper bounds, the convex hull enumeration can also preclude a certain proportion of combinations with incompatible types. The remaining incompatible type combinations will be further removed in the branching stage by a branching rule to be introduced in § 5.3.1.1.

For all W_j in our problem instances, due to their small dimensions (e.g. $|K_j| \leq 4, \forall j \in N$ in the ScotRail datasets), and numbers of points (e.g. $|W_j| \leq 9, \forall j \in N$ in the ScotRail datasets), it is possible to explicitly compute the convex hulls of all unit combination sets by some well-established convex hull computation algorithms, e.g. the QuickHull algorithm [6]. Such convex hulls associated with each train are referred to as *train convex hulls* as

$$\text{conv}(W_j) = \left\{ w^j \in \mathbb{R}_+^{K_j} \middle| H^j w^j \leq h^j \right\}, \quad \forall j \in N, \tag{4.7}$$

described by a set of nonzero facets $f \in F_j$ in $\mathbb{R}_+^{K_j}$, with $H^j \in \mathbb{R}^{F_j \times K_j}$ and $h \in \mathbb{R}^{F_j}$. Finally, by variable conversion $w_k^j = \sum_{a \in \delta_-^k(j)} x_a$, we can replace Constraints (3.4) and (3.5) in formulation (AF_0) by the following train convex hull constraints:

$$\sum_{k \in K_j} \sum_{a \in \delta_-^k(j)} H_{f,k}^j x_a \leq h_f^j, \quad \forall f \in F_j, \forall j \in N, \tag{4.8}$$

where $H_{f,k}^j$ is the entry of nonzero facet f and type k in H^j and h_f^j is the entry of nonzero facet f in h^j.

Although Constraints (4.8) do not necessarily describe the facets of the corresponding original convex hulls in the ILP with respect to variables x, they do capture the solution spaces in a tighter form. We will call this more powerful connection-arc based ILP formulation formed by replacing Constraints (3.4) and (3.5) with Constraints (4.8) in (AF_0) the “tightened arc formulation” (AF_2).

Note that the combination enumeration and convex hull computation are all processed before solving the ILP such that no extra burden will be added to the solution process.

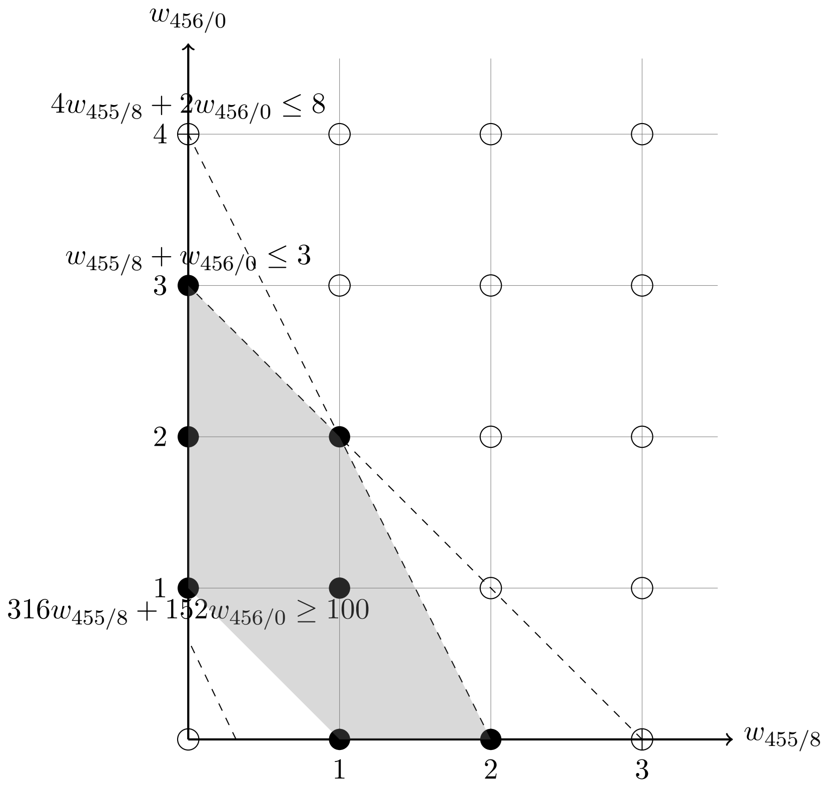

$w_{456/0}$
$4w_{455/8} + 2w_{456/0} \leq 8$
$w_{455/8} + w_{456/0} \leq 3$
$316w_{455/8} + 152w_{456/0} \geq 100$
$w_{455/8}$
4
3
2
1
1
2
3

Figure 4.1: The convex hull from Example 4.1

Example 4.1 We first use an example based on Family SN.II from Southern Railway (cf. Table 4.1) to illustrate the above convex hull preprocessing approach. For a train “1A06” with a passenger demand of 100 seats, train units of c455/8 (4-car, 316 seats) and c456/0 (2-car, 152 seats) are permitted, and they are compatible for coupling. There are two restrictive factors for the coupling upper bounds, as shown in Table 4.1. If solved by Formulation (AF_1), then two constraints are needed for ensuring the coupling upper bounds (before the variables are converted from w to x) as $4w_{455/8} + 2w_{456/0} \leq 8$ and $w_{455/8} + w_{456/0} \leq 3$. Another constraint $316w_{455/8} + 152w_{456/0} \geq 100$ is used for satisfying the passenger demand. On the other hand, we can enumerate all valid unit combinations as:

$$W_{1A06} = \left\{(w_{455/8}, w_{456/0}) | (1,0), (2,0), (0,1), (1,1), (0,2), (1,2), (0,3)\right\},$$

and compute its corresponding train convex hull:

$$\operatorname{conv}(W_{1A06}) = \left\{ w \in \mathbb{R}^2_+ \;\middle|\; \begin{array}{l} f_1 : 2w_{455/8} + w_{456/0} \le 4 \\ f_2 : w_{455/8} + w_{456/0} \le 3 \\ f_3 : w_{455/8} + w_{456/0} \ge 1 \end{array} \right\},$$

which is a polytope with three nonzero facets $\{f_1, f_2, f_3\} = F_{1A06}$, giving three corresponding train convex hull constraints for train 1A06. Figure 4.1 gives an illustration on the above example. The filled points indicates the valid unit combinations and the blank points are the invalid ones. The dashed lines give the constraints if Formulation (AF_1) is used and the shaded area is the convex hull of the valid combination points if Formulation (AF_2) is used. One can see that compared with the explicit multiple restrictive factor constraints the convex hull has narrowed down the solution space shown at the bottom-left corner area in the figure.

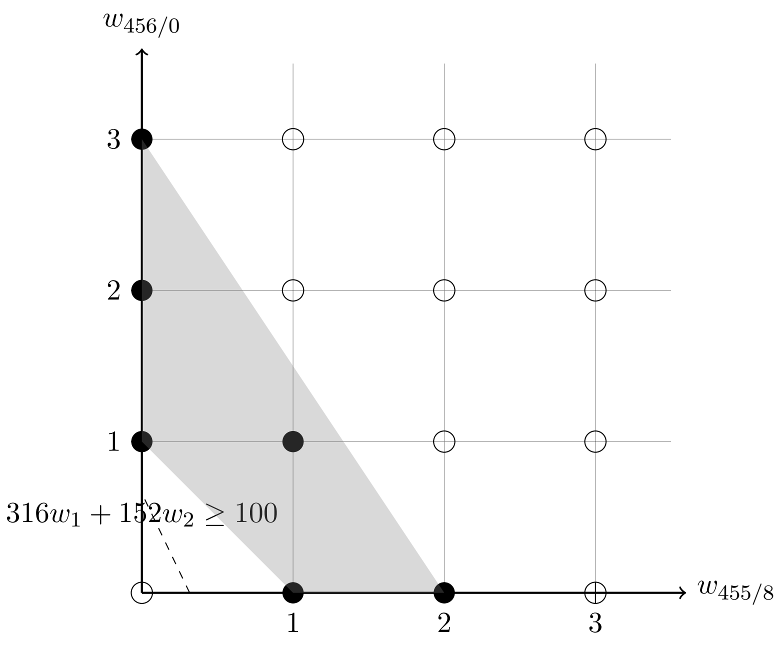


Figure 4.2: The convex hull from Example 4.2

Example 4.2 Now consider another example based on the artificial instances in Table 4.2 with the same demand 100 seats at train 1A06. In this case the multiple restrictive factor constraints are unable to represent combination-specific coupling upper bounds. We have to enumerate all possible combinations:

$$W_{1A06} = \Big\{ (w_{455/8}, w_{456/0}) | (1,0), (2,0), (0,1), (1,1), (0,2), (0,3) \Big\},$$

and compute its corresponding train convex hull:

$$\operatorname{conv}(W_{1\text{A06}}) = \left\{ w \in \mathbb{R}_+^2 \;\middle|\; \begin{array}{l} f_1 : 3w_{455/8} + 2w_{456/0} \leq 6 \\ f_2 : w_{455/8} + w_{456/0} \geq 1 \end{array} \right\}, \tag{4.9}$$

to be further processed into appropriate constraints. Figure 4.2 shows the convex hull from the valid combinations from Example 4.2. Note that in Example 4.2 if we assume that c455/8 and c456/0 are incompatible for coupling, then the unit combination set will be changed to $\{(1,0),(2,0),(0,1),(0,2),(0,3)\}$. However, one can check that its corresponding convex hull would be exactly the same as the one given in (4.9), which contains an invalid combination $(1,1)$. Those invalid combinations appearing in the convex hulls will be eliminated by a branching method to be introduced in § 5.3.1.1.

Exploring the local convex hull method In the previous research [3, 15, 29, 70, 84] and the train convex hull method proposed in this section, local convex hulls are easy to compute using some well-established algorithms. This is because the dimensions and numbers of points of the convex hulls are quite small or because there are special characteristics such that the convex hulls can be described explicitly (e.g. in [15] the coupling upper bound is 2 units for all trains). It is known that when the numbers of dimensions and points increase, it would be more difficult to compute the relevant convex hulls [66]. As for train unit scheduling, Southern Railway has some routes that can be allowed to use any of at least 7 unit types with the coupling upper bounds of more than 2 units, which gives a more difficult problem than any from the previous literature. It is also of theoretical interest, either within rolling stock scheduling or in a more generic integer multicommodity flow framework, to see to what extent these local convex hulls are still practical, by former and/or new algorithms, to compute the local convex hulls.

Some tentative theoretical and experimental exploration regarding the local convex hull method for rolling stock scheduling has been conducted and some results have been obtained. However given the main research theme of this PhD project, the results on the more generalised local convex hull method are only preliminary. They may be useful for future researchers in relevant fields to carry out further studies. We will leave the details of the results to Chapter 7 for interested readers.

4.2 A reduced arc formulation: leaving some requirements to post-processing

Formulations $(AF_0), (AF_1)$ and (AF_2) try to include as many requirements as possible at the global network-level framework. However, they correspond to integer fixed-charge multicommodity flow problems which are known to be hard to solve even for some small-to-medium scale instances [40]. Note that, even the "mixed-integer version" of a fixed-charge multicommodity flow problem, i.e. the arc variables x are continuous whilst only the fixed-charge variables y are binary, is regarded as difficult. Our computational experiments for Southern Railways (AF_1) have also proved that: the formulation cannot solve the problem to optimality generally for the instances with the number of trains greater than 200. Details on the experiments can be found in § 6.1.

In order to deal with instances with more than 200 trains, which represent medium-to-large scale instances in the UK railway, it is necessary to use different approaches in solving the network-level model. One way would be to relax some requirements imposed on the network-level framework to the subsequent post-processing stages such that the network-level ILP can have the ability to solve medium/large sized problems without undermining the global optimality too much. This gives the idea in only using a "standard" integer multicommodity flow formulation by removing all block-arc variables y_a as well as all associated constraints. At the same time, the overnight balance variables and constraints will also be removed although they may not bring too much burden to the problem due to their small numbers. The reason to not include them is mainly because they are not mandatory at the current unit diagramming stage. Another minor reason is for the ease of description in the subsequent theoretical analysis such as Dantzig-Wolfe decomposition. By doing this, the network-level framework loses its ability to eliminate excessive/redundant coupling/decoupling, to ensure the time allowances in a turnround connection involving coupling/decoupling and to keep the unit inventory balanced at each depot. Bear in mind those requirements will have to be satisfied at later stages. Note that although, taking formulation (AF_0) for instance, Constraints (3.8) and (3.9) are also removed, it will not affect the resulting ILP's ability to forbid coupling/decoupling operations at banned locations, thanks to the banned location branching method to be introduced in § 5.3.1.2. Moreover, the train convex hull constraints in formulation (AF_2) are unaffected since they do not involve the block-arc and overnight balance variables.

Therefore, we propose a reduced arc formulation (AF_3) by removing all variables and constraints associated with y_a and $z^-_{d,k}, z^+_{d,k}$ from the tightened connection-arc formulation (AF_2), which gives a standard integer multicommodity flow problem:

$$(AF_3) \qquad \min \quad \sum_{k\in K}\sum_{a\in\mathcal{A}^k} c_a x_a \tag{4.10}$$

$$\text{s. t.} \quad \sum_{a\in\delta^k_+(s)} x_a \le b^k_0, \quad \forall k\in K; \tag{4.11}$$

$$\sum_{a\in\delta^k_-(j)} x_a - \sum_{a\in\delta^k_+(j)} x_a = 0, \quad \forall j\in N, \forall k\in K; \tag{4.12}$$

$$\sum_{k\in K_j}\sum_{a\in\delta^k_-(j)} H^j_{f,k} x_a \le h^j_f, \quad \forall f\in F_j, \forall j\in N; \tag{4.13}$$

$$x_a\in\mathbb{Z}_+, \quad \forall a\in\mathcal{A}^k, \forall k\in K;. \tag{4.14}$$

(AF_3) is generally easier and more efficient to solve than (AF_2), despite the price on temporarily leaving some requirements from the network-level stage to the station-level stage etc. A strong supportive evidence for such a split approach is that our computational experiments have shown that it was always possible to recover any potential time allowance violation locally in post-processing for all the instances we tested. Moreover, in those experiments minimising coupling/decoupling locally at stations can always be realised without increasing the fleet size. Note that it is up to the branching stage in solving (AF_3) to satisfy the requirements on unit type-type compatibility and coupling/decoupling banned locations.

4.3 Applying Dantzig-Wolfe decomposition to the reduced arc formulation (AF_3)

In this section, the Dantzig-Wolfe decomposition method [27] will be applied to the reduced connection-arc formulation (AF_3). This will give a path formulation that is more suitable to be solved by column generation based methods. The same arguments developed below will also be applicable to $(AF_0), (AF_1), (AF_2)$.

Dantzig-Wolfe decomposition is a technique for handling linear and integer programming problems with special embedded decomposable structures that can provide efficient solutions. Originated from a paper by Dantzig and Wolfe [27] in 1960, it has become a mature and widely applied technique in optimisation. Instead of dealing with a difficult original problem directly, this technique allows one to solve a set of relatively smaller sized, easier and typically well-structured subproblems and a reformulated master problem with the column generation technique [28,59]. In the context of integer programming, Dantzig-Wolfe decomposition can often additionally capture some integrality properties of part of the problem and provide a tighter LP relaxation bound. According to the ways in reformulating the original problem, the application of Dantzig-Wolfe decomposition in integer programming can be further categorised as discretization and convexification [74]. Regarding different views in interpreting this technique, first there is a strong relation between Dantzig-Wolfe decomposition and Lagrangian relaxation via the master problem and its dual [36]. In addition, it can also be viewed as a special case of the variable redefinition method proposed by Martin [64]. Further details on Dantzig-Wolfe decomposition can be found in [27,59,74,75]

When the number of arcs is large, the reduced arc formulation (AF_3) may become difficult to solve due to the excessive number of arc variables. By applying Dantzig-Wolfe decomposition to (AF_3), a path formulation can be obtained which allows us to explore the embedded network structure in the subproblems while solving the path formulation ILP more efficiently and implicitly by column generation.

4.3.1 The node-arc incidence matrix and total unimodularity

Regarding formulation (AF_3), Constraints (4.12) are the flow conservation constraints. Often in a standard presentation in the network flow theory, the coefficients from Constraints (4.12) together with (4.11) are associated with a so-called *node-arc incidence matrix*, for cases where the "$\leq$" signs in Constraints (4.11) are replaced by "$=$", i.e. the amount of flow to be routed though the network is fixed to b_0^k for each commodity k. Nevertheless the discussion below will show that the extension from "$=$" to "$\leq$" as in (4.11) will still be suitable for the application of Dantzig-Wolfe decomposition. The name node-arc incidence matrix will thus be inherited from the standard cases noting the slight difference as above. Moreover, sometimes a flow amount bound constraint $\sum_{a\in\delta_-^k(t)} x_a \leq b_0^k$ at the sink t may also be present for denotational completeness. However this constraint is redundant because the rank of an $m \times n$ node-arc incidence matrix cannot be greater than $m - 1$, and one would expect the flow conservation is kept throughout the network by Constraints (4.12).

Let $Mx \leq b$ be the matrix form of Constraints (4.11) – (4.12) such that M is the node-arc incidence matrix of (AF_3) and let $M^k x \leq b^k$ be the part from $Mx \leq b$ corresponding to type k, i.e. $b^k = (b_0^k, 0, \ldots, 0, b_0^k)^T$. Also let H and h be the coefficient matrix and right-hand side vector associated with Constraints (4.13). A more concise form of (AF_3) can be written as

$$(AF_3') \quad \min_{x\in\mathbb{Z}_+^{\mathcal{A}}} \{c^T x | Mx \leq b, Hx \leq h\}, \tag{4.15}$$

where M has the typical block-angular structure suitable to be decomposed with respect to each type k, i.e.

$$M = \begin{pmatrix} M^1 & & & \\ & M^2 & & \\ & & \ddots & \\ & & & M^{|K|} \end{pmatrix}, \quad b = (b^1, \ldots, b^{|K|})^T.$$

Figure 4.3 gives a network as a directed acyclic graph with 7 nodes and 8 arcs, which has a single source s and single sink t corresponding to the connection-arc graph defined in § 3.5.2 for train unit scheduling. The corresponding node-arc incidence matrix is given in Table 4.3. Certain patterns can be recognised for a generic node-arc incidence matrix. The rows and columns in the matrix correspond to the nodes and arcs in the DAG respectively. Also in a column of arc (i, j), there is a -1 from the row of node i, a 1 from the row of node j and all the other entries are 0's.

There are several important properties regarding the node-arc incidence matrices M and M^k. The fundamental one would be they are totally unimodular.

Definition 4.1. *A matrix M is totally unimodular if the determinant of each of its square*

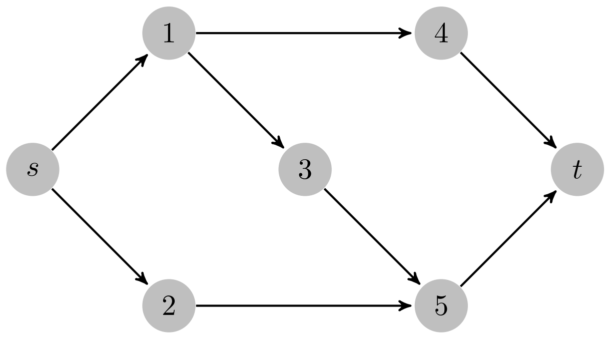


Figure 4.3: A DAG with 7 nodes, 8 arcs, a single source s and a single sink t

Table 4.3: The node-arc matrix associated with the network in Figure 4.3

Nodes \ Arcs	$(s,1)$	$(s,2)$	(1,3)	(1,4)	(2,5)	(3,5)	$(4,t)$	$(5,t)$
s	-1	-1	0	0	0	0	0	0
1	1	0	-1	-1	0	0	0	0
2	0	1	0	0	-1	0	0	0
3	0	0	1	0	0	-1	0	0
4	0	0	0	1	0	0	-1	0
5	0	0	0	0	1	1	0	-1
t	0	0	0	0	0	0	1	1

submatrix is either 0, $+1$ *or* -1.

One would notice by definition if a matrix is totally unimodular, then in particular, each of its entries should be either 0, +1 or −1. Theoretical exposition on total unimodularity is out of the scope of this thesis. Here a simple sufficient condition for determining the total unimodularity property of node-arc incidence matrices is given.

Theorem 4.1 (Poincaré [68]). *Let M be a matrix with entries in $\{0, +1, -1\}$. If each column of it has at most one $+1$ and at most one -1, then M is totally unimodular.*

Thus the result below is straightforward.

Corollary 4.1 (Poincaré [68]). *The incidence matrix M of any directed graph is totally unimodular.*

The following theorem characterises the polyhedron defined by a totally unimodular matrix.

Theorem 4.2 (Hoffman and Kruskal [42], Veinott and Dantzig [77]). *If M is an $m \times n$ matrix with integer entries, the following are equivalent:*

(i) *M is totally unimodular.*

(ii) *The extreme points of $\{x \in \mathbb{R}^n_+ | Mx \leq b\}$ are integral for all integral vectors b.*

(iii) *Every nonsingular submatrix of M has an integral inverse.*

Thus it is natural to link a matrix's total unimodularity with the optimal solutions of the linear program it represents. Particularly, when the LP is bounded, a stronger necessary and sufficient relation can be guaranteed as the following theorem [81].

Theorem 4.3. *The linear program $\min_{x\in\mathbb{R}^n_+}\{c^Tx|Mx \leq b\}$ has an integral optimal solution for all rational vectors c and for all integral vectors b for which it has a finite optimal objective value if and only if M is totally unimodular.*

The above states that for a bounded LP defined by a totally unimodular matrix, it will always give an integral optimal solution for any objective coefficients and integral right-hand sides, and vice versa. Note the this "iff." condition is based on the requirement that it holds for all $c \in \mathbb{R}^n$ and for all $b \in \mathbb{Z}^n$. Particularly, there could be LP instances defined by non-unimodular matrices that always yield integer optima either with any c but certain b or with any integer b but certain c [56].

Given an $m \times n$ totally unimodular matrix M such that all extreme points of $Q = \{x \in \mathbb{R}^n_+|Mx \leq b\}$ are integral and Q is bounded, if a discrete set $X = \{x \in \mathbb{Z}^n_+|Mx \leq b\}$ is defined, then the convex hull of X is known to be exactly $\text{conv}(X) = \{x \in \mathbb{R}^n_+|Mx \leq b\}$ [81].

4.3.2 Paths and extreme points

Now come back to the directed acyclic graph underneath formulation (AF_3'). The node-arc structure of each type-graph $\mathcal{G}^k$ has been encoded into the corresponding constraints $M^kx \leq b^k$. With respect to any type-k flows routed through $\mathcal{G}^k$ (i.e. disregard other restrictions like $Hx \leq h$ and x_a's are integers), a *feasible flow* in $\mathcal{G}^k$ is defined as an arc variable solution $x \in \mathbb{R}^{\mathcal{A}^k}_+$ that satisfies $M^kx \leq b^k$. Therefore $\{x \in \mathbb{R}^{\mathcal{A}}_+|M^kx \leq b^k\}$ is the set of all feasible flows of $\mathcal{G}^k$.

For the type-graph $\mathcal{G}^k = (\mathcal{N}^k, \mathcal{A}^k)$ defined over the node-arc incidence matrix from $M^kx \leq b^k$, apart from using the arc-based variable $x \in \mathbb{R}^{\mathcal{A}^k}_+$ to represent feasible flows, another way would be to use path variables and cycle variables. Here a path is restrictively defined as a source-sink simple path joining s and t. As the networks of the train unit scheduling DAGs are all acyclic, cycles will be omitted here. Given a set of path variables, they will satisfy the flow conservation requirement automatically by definition. All path solutions can be uniquely converted into feasible flows based on arcs solutions by $x_a = \sum_{p\in P^k}\delta^p_a x_p = \sum_{p\in P^k_a} x_p, \forall a \in \mathcal{A}^k$, where $x_p \in \mathbb{R}_+$ is the magnitude of the flow along path p, P^k is the set of paths in $\mathcal{G}^k$, $\delta^a_p = 1$ if arc a is in path p and 0 otherwise, and P^k_a is the set of paths passing through arc a. On the other hand, an arbitrary arc solution may not represent a feasible flow across the network. However when an arc solution is feasible, the following Flow Decomposition Theorem [2] indicates that the arc flows from a feasible arc solution can be decomposed into a set of path flows (and cycle flows, which shall be omitted in the acyclic networks for train unit scheduling).

Theorem 4.4 (Flow Decomposition in DAG)**.** *In a directed acyclic graph, every path flow solution has a unique representation as an arc flow solution that satisfies the flow conservation requirement. Conversely, every arc flow solution that satisfies the flow conservation requirement can be decomposed as the sum of path flows, where the decomposition is not necessarily unique.*

In fact the above decomposition is a special case of the well-known Minkowski-Weyl's Theorem which shows that every point in a polyhedron can be expressed as a convex combination of the polyhedron's extreme points (paths) plus a non-negative combination of the polyhedron's extreme rays (cycles) [28].

It has been mentioned that the polyhedron $Q^k = \{x \in \mathbb{R}_+^{\mathcal{A}^k} | M^k x \leq b^k\}$ is the set of all feasible arc flows of $\mathcal{G}^k$. There is a slight difference in this polyhedron from an ordinary network flow case such that the flow amount is only upper bounded rather than fixed (the ordinary one is sometimes called a shortest path polyhedron). One may consider adding a dummy node and dummy arcs in $\mathcal{G}^k$ without any demands/costs to replace all "$\leq$" by "$=$" and to route all superfluous flows to them. Nevertheless, the standard analysis is still analogously workable on the new case due to "$\leq$" with some modification. Noting that the problem defined by $M^k x = b^k$ corresponds to an uncapacitated network, first we give a result for the standard case [34, 78].

Theorem 4.5. *A feasible solution to the system $\{x \in \mathbb{R}_+^{\mathcal{A}^k} | M^k x = b^k\}$ is an extreme point if and only if it corresponds to a path flow.*

Assume that the system $M^k x = b'^k$ always has feasible solutions for any total flow amount b'^k_s, b'^k_t in the range of $[0, b^k_0]$. Proposition 4.1 shows that some properties in Theorem 4.5 for the standard case can be extended to the new case with "$\leq$".

Proposition 4.1. *Each extreme point of $Q^k = \{x \in \mathbb{R}_+^{\mathcal{A}^k} | M^k x \leq b^k\}, b^k = (b^k_0, 0, \ldots, 0, -b^k_0)^T$ is either the null point $\mathbf{0}$ or a feasible solution corresponding to a single path with a magnitude of b^k_0. Each feasible solution corresponding to a single path with a magnitude of b^k_0 is an extreme point of Q^k.*

Proof. First the null point $\mathbf{0} \in \mathbb{R}_+^{\mathcal{A}^k}$ is an extreme point of Q^k, since $M^k \mathbf{0} \leq b^k$ and it is impossible to find any other two points $x^1, x^2 \in Q^k$ with a scalar $0 < \alpha < 1$ such that $\mathbf{0} = (1-\alpha)x^1 + \alpha x^2$ and $x^1, x^2 \neq \mathbf{0}$.

Then let x^* be an extreme point containing positive entries. Such a point exists as it is assumed that $M^k x = b'^k$ is always feasible for any $0 \leq b'^k_s, b'^k_t \leq b^k_0$. Theorem 4.4 shows that x^* can be decomposed into a set of paths P^* (and there is no cycle due to the graph being acyclic). Suppose there are two paths $p_1, p_2 \in P^*$. Then two feasible solutions x^1, x^2 can be constructed such that x^1 is formed by transferring a small amount of flow $0 < \varepsilon < \min\{\mu(p_1), \mu(p_2)\}$ (with $\mu(p)$ the magnitude of the flow along path p) from p_1 to p_2 and x^2 by transferring the same amount of flow from p_2 to p_1. Then we have $x^* = \frac{1}{2}x^1 + \frac{1}{2}x^2$ which contradicts the assumption that x^* is an extreme point. The same principle can be applied to cases with more than two paths. Thus there can be one and only one path in P^*. The flow amount of its arcs should be the same and be no greater than b_0^k. Suppose an extreme point x^* based on a single path has a flow amount $0 < b' < b_0^k$. Then there exist two feasible points $x^1 = \mathbf{0}$ and x^2 formed by increasing the flow amount of x^* from b' to b_0^k, and a scalar $0 < \alpha = \frac{b'}{b_0^k} < 1$ such that $x^* = (1-\alpha)x^1 + \alpha x^2$, yielding a contradiction. The only option left would be the path flow with a magnitude of b_0^k and it can be verified that it is impossible to construct it with any other two path flows along the same path.

According to Theorem 4.4, a path flow is the most elementary entity after a feasible flow is decomposed. Thus a path flow cannot be formed by feasible flows from different paths. Within the same path p, although a path flow x'_p with $\mu(x'_p) < b_0^k$ can be formed by the path flow $x_p, \mu(x_p) = b_0^k$ and the null point $\mathbf{0}$, it is impossible to construct x_p in the same way by any other two flows along p. $\square$

Proposition 4.1 shows that with respect to the extreme points the polyhedron $Q^k = \{x \in \mathbb{R}_+^{\mathcal{A}^k} | M^k x \leq b^k\}$ from the extended case has only one additional point $\mathbf{0}$ compared with the polyhedron $\{x \in \mathbb{R}_+^{\mathcal{A}^k} | M^k x = b^k\}$ from the standard case. The implication is that a similar equivalence relation to the shortest path problem as from the standard case can also be defined by Q^k. Consider the following LP

$$\min_{x \in \mathbb{R}_+^{\mathcal{A}^k}} \{c^T x | M^k x \leq b^k\}, \tag{4.16}$$

where the cost c is an arbitrary rational vector based on arcs. The LP (4.16) corresponds to the basic structure of the subproblem in selecting promising candidates for the RMP if the arc formulation (AF'_3) (4.15) is solved by Dantzig-Wolfe decomposition or Lagrangian relaxation. If we replace the "$\leq$" in (4.16) with "$=$" then it is well-known that the resulting LP is equivalent to a shortest path problem with arc cost c such that a path with a flow amount of b_0^k has to be found to minimise the total cost[1]. In (4.16), although the flow amount is only bounded rather than fixed, since the nonzero extreme points of Q^k should be all at the maximum value of b_0^k, then solving it will still be equivalent to finding a shortest path of amount b_0^k in $\mathcal{G}^k$ as long as $\mathbf{0}$ is not the optimal solution. When $\mathbf{0}$ happens to solve the LP (e.g. $c_a \geq 0, \forall a \in \mathcal{A}^k$), no shortest path can be found by solving (4.16).

[1]In many occasions in illustrating such an equivalence the flow amount b_0^k is simply set as 1 since it is essentially irrelevant with the finding of the shortest path. In fact a similar shortest path problem can still be constructed even there is no flow upper bound constraint. In this case the polyhedron will be unbounded and paths corresponding to the extreme rays that can take any values are expected.

4.3.3 Decompose arc formulation to path formulation

Now come back to the efficient connection-arc formulation (AF_3') (4.15). Let $X^k = \{x \in \mathbb{Z}_+^{\mathcal{A}^k} | M^k x^k \leq b^k\}$ and c^k, H^k, x^k the corresponding components of costs, convex hull coefficients and arc variables with respect to type k, (4.15) can be rewritten based on each k into formulation (AF_3'') (4.17) – (4.19),

$$(AF_3'') \quad \min \sum_{k \in K} c^{kT} x^k \tag{4.17}$$

$$\text{s. t.} \quad \sum_{k \in K} H^k x^k \leq h; \tag{4.18}$$

$$x^k \in X^k, \forall k \in K. \tag{4.19}$$

The arc variable x should be understood in an $x \in X^{k_1} \times \cdots \times X^{k_{|K|}}$ way and we simplify this denotational rigidity such that x^k implies an original variable $x = (0, \ldots, x^k, \ldots, 0)^T$. Assuming that each $c^{kT} x^k$ is bounded for any feasible solutions in X^k, then replacing each X^k with $\text{conv}(X^k)$ in (4.19) will not change the objective. It is known that $\text{conv}(X^k) = \{x \in \mathbb{R}_+^{\mathcal{A}^k} | M^k x \leq b^k\}$ as aforementioned. Hence if we let the finite set $\{x^p\}_{p \in P^k}$ be the nonzero extreme points of $\text{conv}(X^k)$, then each point in $\{x^p\}_{p \in P^k}$ corresponds to a path in $\mathcal{G}^k$. The only remaining extreme point in $\text{conv}(X^k)$ apart from $\{x^p\}_{p \in P^k}$ is $\mathbf{0} \in \mathbb{R}^{\mathcal{A}^k}$ which will be particularly denoted as $\mathbf{0}^k$. From the Minkowski-Weyl's Theorem [65], each point x^k in X^k can be expressed as a convex combination of the above extreme points, i.e.,

$$\begin{gathered} x^k = \lambda_0^k \mathbf{0}^k + \sum_{p \in P^k} \lambda_p x^p, \quad \lambda_0^k + \sum_{p \in P^k} \lambda_p = 1, \\ \lambda_0^k \geq 0, \lambda_p \geq 0, \forall p \in P^k, \forall k \in K. \end{gathered} \tag{4.20}$$

Replacing x^k by (4.20) in (AF_3'') will yield a so-called extensive formulation (EF_3) (4.21) – (4.24) based on the extreme points of all $\text{conv}(X^k), k \in K$ while the original arc formulation is called a compact formulation. The extensive formulation reads,

$$(EF_3) \quad \min \sum_{k \in K} \sum_{p \in P^k} \left(c^{kT} x^p \right) \lambda_p \tag{4.21}$$

$$\text{s. t.} \quad \sum_{k \in K} \sum_{p \in P^k} (H^k x^p) \lambda_p \leq h; \tag{4.22}$$

$$\lambda_0^k + \sum_{p \in P^k} \lambda_p = 1, \qquad \forall k \in K; \tag{4.23}$$

$$\sum_{p \in P^k} \lambda_p x^p \in \mathbb{Z}_+^{\mathcal{A}^k}, \quad \forall k \in K. \tag{4.24}$$

Formulation (EF_3) is based on the λ-variables as the "weights" or "fractions" for each extreme point in a convex combination for a solution. Now for each path $p \in P^k, k \in K$, we define a new variable $\xi_p = b_0^k \lambda_p$. Its natural interpretation would be the flow amount along path p contributed to a solution made up of all paths. From Proposition 4.1 each extreme point in $\text{conv}(X^k)$ has the same maximum amount of flow b_0^k in the corresponding path. Thus for each $p \in P^k$ let $x^p = b_0^k \hat{x}^p$, i.e. $\hat{x}^p$ shares the same nonzero entries as x^p except that they are all 1's. Then we have $x^k = \sum_{p \in P^k} \lambda_p b_0^k \hat{x}^p = \sum_{p \in P^k} \hat{x}^p \xi_p$, and its equivalent component-wise form $x_a^k = \sum_{p \in P^k} \delta_a^p \xi_p = \sum_{p \in P_a^k} \xi_p$ which is a familiar one in converting path variables to arc variables. Now substitute the extreme points in (EF_3) by $x_p = b_0^k \hat{x}^p$ and $\xi_p = b_0^k \lambda_p$ and note that $\sum_{p \in P^k} \lambda_p = 1 - \lambda_0^k \leq 1$ implies $\sum_{p \in P^k} b_0^k \lambda_p \leq b_0^k$, a new formulation (PF_3) based on path variable ξ can be derived as

$$(PF_3) \quad \min \sum_{k \in K} \sum_{p \in P^k} \left(c^{kT} \hat{x}^p \right) \xi_p \tag{4.25}$$

$$\text{s. t.} \quad \sum_{k \in K} \sum_{p \in P^k} (H^k \hat{x}^p) \xi_p \leq h; \tag{4.26}$$

$$\sum_{p \in P^k} \xi_p \leq b_0^k, \qquad \forall k \in K; \tag{4.27}$$

$$\sum_{p \in P^k} \hat{x}^p \xi_p \in \mathbb{Z}_+^{\mathcal{A}^k}, \quad \forall k \in K. \tag{4.28}$$

In the above path formulation, the original convexity constraints (4.23) become the fleet size bounding constraints (4.27). Constraints (4.28) have inherited the original integrality requirement imposed on the individual arc variables x_a^k which can be expressed in terms of ξ_p as

$$x_a^k = \sum_{p \in P^k} \delta_a^p \xi_p = \sum_{p \in P_a^k} \xi_p. \tag{4.29}$$

This precaution against changing the integrality requirement from the compact formulation is expected when Dantzig-Wolfe decomposition is applied in a convexification way [74]. The following statement will ensure that anyway it is sufficient to let all ξ_p to be integral to achieve (4.28).

Proposition 4.2. *For the arc formulation (AF_3) and the path formulation (PF_3), each integral path solution ξ corresponds to a unique integral arc solution x. On the other hand, each integral arc solution x will have at least one corresponding integral path solution ξ.*

Proof. The first part is trivially true according to (4.29). The second part can be proved by the procedure of flow decomposition, e.g. see [2]. During the procedure each time a path with the magnitude of the smallest flow amount from its belonging arcs is identified and subtracted from the network until no path is available and the network has no flow. A path identified in this way should be always integral as its magnitude is either from an original arc or from an updated arc with its flow amount reduced by another arc's original or updated amount, which are all of integral values. $\square$

Being able to use the sufficient integrality requirement imposed on individual paths as

$$\xi_p \in \mathbb{Z}_+, \quad \forall p \in P^k, \forall k \in K, \tag{4.30}$$

we will redefine the path formulation (PF_3) based on ξ as (4.25) – (4.27) plus (4.30). Formulation (PF_3) can be re-expressed into a more straightforward and intuitive form. First it is possible to define a cost for each path as $c^{kT}\hat{x}^p = \sum_{a\in\mathcal{A}^k} c_a \delta_a^p = \sum_{a\in\mathcal{A}_p} c_a = c_p$. Then we can write Constraints (4.26) in their component-wise form as,

$$\sum_{k\in K_j, p\in P^k} (H^k\hat{x}^p)_{fj}\xi_p \le h_f^j, \quad \forall f \in F_j, \forall j \in N. \tag{4.31}$$

To find out what $(H^k\hat{x}^p)_{fj}$ is, it is helpful to trace back to the arc formulations (AF_3) and (AF_3''). Consider the component-wise form of Constraint (4.18) in the (f,j)-th entry, i.e. $\sum_{k\in K_j}(H^k x^k)_{fj} \le h_f^j$, and compare it with (4.13). Then we have

$$(H^k x^k)_{fj} = \sum_{a\in\delta_-^k(j)} H_{fk}^j x_a.$$

Thus

$$\begin{aligned}(H^k\hat{x}^p)_{fj} &= \frac{1}{b_0^k}(H^k x^p)_{fj}\\ &= \frac{1}{b_0^k}\sum_{a\in\delta_-^k(j)} H_{fk}^j x_a^p\\ &= \sum_{a\in\delta_-^k(j)} H_{fk}^j \delta_a^p.\end{aligned}$$

Let P_j^k be the set of paths passing via node j in type-graph $\mathcal{G}^k$, and let $\mathcal{A}_p^k$ be the set of arcs in path p in type-graph $\mathcal{G}^k$. It is possible to rewrite Constraints (4.26) with the coefficients from the original train convex hull constraints in (4.8) and the arc formulations as,

$$\begin{aligned}\sum_{k\in K_j, p\in P^k} (H^k\hat{x}^p)_{fj}\xi_p &= \sum_{k\in K_j, p\in P^k}\sum_{a\in\delta_-^k(j)} H_{fk}^j\delta_a^p\xi_p\\ &= \sum_{k\in K_j, p\in P^k}\Bigg(\sum_{a\in\delta_-^k(j)\setminus\mathcal{A}_p^k} H_{fk}^j\cdot 0 + \sum_{a\in\delta_-^k(j)\cap\mathcal{A}_p^k} H_{fk}^j\cdot 1\Bigg)\xi_p\\ &= \sum_{k\in K_j, p\in P_j^k} H_{fk}^j\xi_p \le h_f^j.\end{aligned}$$

Therefore we have reformulated the arc formulations into path formulations using Dantzig-Wolfe decomposition and have re-expressed it into a more intuitive form using the original coefficient from the train convex hull constraints in the arc formulations. These path formulations will be introduced formally in the next section.

4.3.4 The path formulations

The arc formulations use variables based on connection-arcs. Alternatively, one may consider using the alternative path formulations instead. A path here is restrictively defined as an s-t path as a collection of consecutive arcs starting from source s and ending at sink t. Theoretically a path formulation is the result of applying Dantzig-Wolfe decomposition to a corresponding arc formulation as shown in the previous parts, although a path formulation per se has a natural interpretation to be self-explanatory without its corresponding arc formulation.

Define decision variables $x_p \in \mathbb{Z}_+, \forall p \in P^k, \forall k \in K$ as the path variables representing the flow amount along path p. Each arc formulation from $(AF_0), (AF_1), (AF_2)$ and (AF_3) has an equivalent path formulation. For example, AF_3 has the following corresponding path formulation (PF_3') (expressed in a "more intuitive" form):

$$
\begin{aligned}
(PF_3') \quad \min \quad & \sum_{k\in K}\sum_{p\in P^k} c_p x_p && (4.32)\\
\text{s. t.} \quad & \sum_{p\in P^k} x_p \le b_0^k, \quad \forall k \in K; && (4.33)\\
& \sum_{k\in K_j}\sum_{p\in P_j^k} H_{f,k}^j x_p \le h_f^j, \quad \forall f \in F_j, \forall j \in N; && (4.34)\\
& x_p \in \mathbb{Z}_+, \quad \forall p \in P^k, \forall k \in K. && (4.35)
\end{aligned}
$$

Note that in the path formulations one needs to be clear about the path costs c_p in the objective (4.32), with respect to different sub-weights $V_1, \ldots, V_6$ by:

$$
\begin{aligned}
\sum_{k\in K}\sum_{p\in P^k} c_p x_p = \; & V_1 \sum_{k\in K}\sum_{p\in P^k} x_p + V_2 \sum_{k\in K}\sum_{j\in N^k}\sum_{p\in P_j^k} m_j^k x_p \\
& + V_3 \sum_{k\in K}\sum_{a\in \mathcal{A}^k}\sum_{p\in P_a^k} e_a x_p - V_4 \sum_{k\in K}\sum_{a\in A}\sum_{p\in P_a^k} g_a^k x_p \\
& - V_5 \sum_{k\in K}\sum_{j\in N^k}\sum_{p\in P_j^k} \pi_j^k x_p - V_6 \sum_{k\in K}\sum_{a\in A_0^k}\sum_{p\in P_a^k} \pi_a^k x_p.
\end{aligned}
\tag{4.36}
$$

Let N_p, $\mathcal{A}_p$, A_p and A_{0p} be the set of train nodes, arcs, connection arcs and sign-on/off arcs contained in path p, and $k(p)$ the type of path p, the overall cost c_p for a path p can thus be determined according to (4.36) as:

$$
c_p = V_1 + V_2 \sum_{j\in N_p} m_j^{k(p)} + V_3 \sum_{a\in \mathcal{A}_p} e_a - V_4 \sum_{a\in A_p} g_a^{k(p)} - V_5 \sum_{j\in N_p} \pi_j^{k(p)} - V_6 \sum_{a\in A_{0p}} \pi_a^{k(p)}, \tag{4.37}
$$

which has a natural interpretation how different terms are combined to form a path's cost. In most of the experimented instances, it is set that $V_1 = 1$ and the other $V_i, i \neq 1$ some values smaller than 1 accordingly.

The additional variables and constraints arising to cater for the unit overnight balance can be easily fitted into the path formulation by

$$z^{+}_{d,k} - z^{-}_{d,k} = \sum_{a\in(A^k_d)^{\text{on}}} \sum_{p\in P_a} x_p - \sum_{a\in(A^k_d)^{\text{off}}} \sum_{p\in P_a} x_p, \quad \forall d \in D, \forall k \in K; \tag{4.38}$$

It is worth mentioning that the lower bound given by the LP relaxation of the path formulation (PF'_3) is the same as the LP relaxation of the original arc formulation (AF_3), since the decomposed parts $X^k, k \in K$ in Dantzig-Wolfe decomposition have integral convex hulls. Due to the same reason, the lower bound given by the Lagrangian relaxation on (AF_3) by relaxing Constraints (4.13) is the same as the above two bounds.

Despite the same lower bound given by LP or Lagrangian relaxation, the path formulation (PF'_3) is usually more preferable to its corresponding arc formulation (AF_3), mainly because the former can be solved more efficiently by column generation based methods where the subproblems are standard shortest path problems easy to solve. In Chapter 5, a detailed description will be given on how to solve the path formulations by a column generation based method named branch-and-price.

Chapter 5

A branch-and-price solver

In this chapter, a branch-and-price ILP solver for solving the path formulation (PF_3) will be presented in detail, from which many of the techniques used can also be applied to the path formulations derived from (AF_0)–(AF_2) either directly or with certain modifications.

5.1 Branch-and-price

Branch-and-price [8] is a high-level framework for solving large-scale ILPs by combining branch-and-bound (BB) [81], a traditional way for solving ILPs by partial enumeration and divide-and-conquer, and column generation [28,59], a well-established method in solving large-scale LPs by only considering a subset of variables forming a restricted master problem (RMP) and iteratively adding new variables that may improve the objective value by inspecting results from solving subproblems (separation problems, or pricing problems in the view of the simplex method), until no such candidate is available. In a generic branch-and-bound framework, if the BB tree's root node is solved by column generation, then the root may not contain all the columns needed for finding an optimal or sub-optimal integer solution and it may not even contain sufficient columns for getting an integer solution. Thus branch-and-price calls for the need to also perform column generation not only at the root, but also at the leaf nodes of the BB tree. Figure 5.1 illustrates the basic mechanism of branch-and-price, where at each active node the LP relaxation of the corresponding RMP is solved by column generation.

Several issues have to be taken care of when one tries to combine BB with column generation at each tree node. For instance while the invalid existing columns in the RMP due to branching can be directly deleted, there should be a mechanism to prevent the

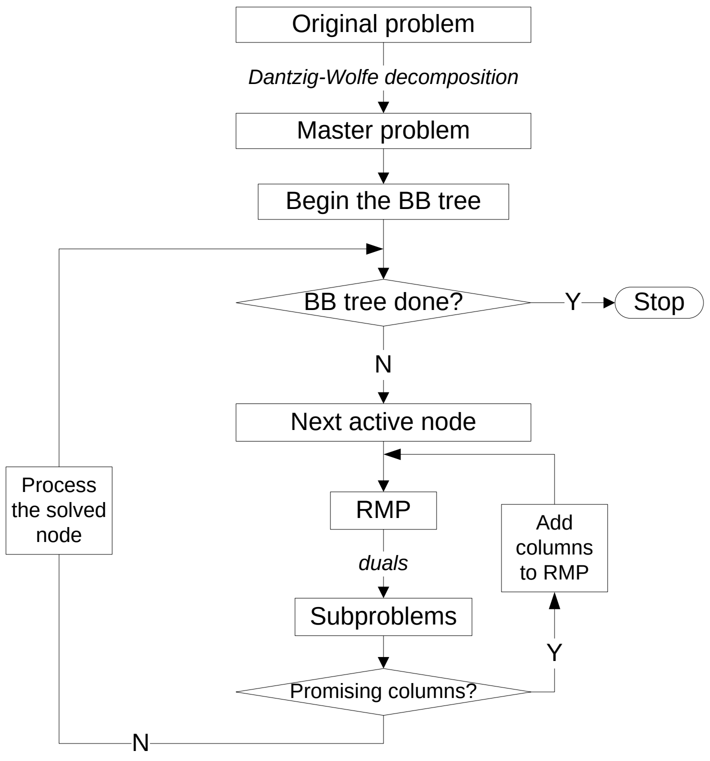


Figure 5.1: A flow chart on the basic mechanism of branch-and-price

generation of such columns in the subproblems. Customised branching rules are thus necessary. Also as the size of the BB tree can be quite huge, it may not be affordable to perform complete column generation throughout the BB tree. Therefore it may be desirable to terminate the column generation early at some leaf nodes. Finally the "tail-off" effect is often observed in column generation process such that the LP objective is improved very slowly at the final stage for a large number of iterations. This behaviour is believed to have close connection with dual variables and still remains a challenging issue for theoretical researchers [28]. Nevertheless, in the branch-and-price solver designed in this research, the "tail-off" effect is prevented by a strategy named column inheritance that is going to be introduced later in this chapter.

Further theoretical elaboration on branch-and-price, branch-and-bound and column generation is beyond the scope of this thesis and interested readers may see [8, 28, 59, 81] for details.

5.2 Column generation

5.2.1 Restricted master problem and subproblems

We will take the path formulation (PF_3) (and its equivalent form $(PF_3'))$ as an example while introducing the branch-and-price solver. Since (PF_3) is an ILP, one may need to solve its LP relaxation ($\overline{PF_3}$) many times to get the optimal integer solution, e.g. in a branch-and-bound tree. Although having the same lower bound from LP relaxation, the path formulation (PF_3) can be more preferable over the arc formulation (AF_3) such that it allows one to solve its LP relaxation ($\overline{PF_3}$) only based on a subset of its variables iteratively, and to only generate new variables if they are likely to improve the objective, hence the idea of "column generation", although sometimes this term is used interchangeably with "Dantzig-Wolfe decomposition" involving both a compact and an extensive formulation.

Column generation can be interpreted in many different ways, for instance, in pricing out columns with negative reduced costs if the simplex method is used to solve the LP relaxation. The essence of it is to suppose only a subset of variables (e.g. here a subset of paths $\widetilde{P}^k$ in each $\mathcal{G}^k$) has been included in solving the LP relaxation problem which is referred to as the restricted master problem (RMP). Let $\phi_k \leq 0, \forall k \in K$ and $\psi_{fj} \leq 0, \forall f \in F_j, \forall j \in N$ be the optimal dual variables associated with Constraints (4.27) and (4.26) of the RMP of ($\overline{PF_3}$), whose dual problem can be written as

$$\max \quad \sum_{k \in K} b_0^k \phi_k + h^T \psi \tag{5.1}$$

$$\text{s. t.} \quad \phi_k + (H^k \hat{x}^p)^T \psi \leq c_p \quad \forall p \in \widetilde{P}^k, \forall k \in K; \tag{5.2}$$

$$\phi, \psi \leq \mathbf{0}. \tag{5.3}$$

Hence there are $|K|$ separation problems from the dual in finding constraints associated with $p \in P^k$ in each $\mathcal{G}^k$ such that $\phi_k + (H^k \hat{x}^p)^T \psi > c_p$. This can be performed for each k by solving the k-th subproblem

$$\overline{c}_k^*(\phi, \psi) := \min_{p \in P^k} \{c_p - \phi_k - (H^k \hat{x}^p)^T \psi\}. \tag{5.4}$$

and checking the sign of $\overline{c}_k^*$. If $\overline{c}_k^* \geq 0$ for all $k \in K$, then the procedure terminates as the primal-dual pair reaches optimality for all potential primal variables. If a $\overline{c}_k^* < 0$ is identified in the k-th subproblem, then the corresponding path $p \in P^k$ is added to the RMP and the updated restricted master problem will be re-optimised. This interaction between the RMP and the k subproblems will be repeated until no promising candidate is available to be added. From the view of the simplex method, $\overline{c}_k^*$ is the smallest reduced cost over all paths in P^k and if $\overline{c}_k^* \geq 0, \forall k \in K$ one can claim the optimality on the primal problem ($\overline{PF_3}$).

The subproblem (5.4) can be regarded as a shortest path problem with a node weight

$\sum_{f\in F_j} H^j_{fk}\psi_{fj}$ at each j plus a source-sink weight $c_p - \phi_k$. To be more specific, we have

$$\begin{aligned}\bar{c}_p &= c_p - \sum_{f\in F_j, j\in N} (H^k\hat{x}^p)_{fj}\psi_{fj} - \phi_k \\ &= c_p - \sum_{f\in F_j, j\in N} \Big(\sum_{a\in\delta^k_-(j)} H^j_{fk}\delta^p_a\psi_{fj}\Big) - \phi_k \\ &= c_p - \sum_{f\in F_j, j\in N} \Big(\sum_{a\in\delta^k_-(j)\cap\mathcal{A}_p} H^j_{fk}\psi_{fj}\Big) - \phi_k \\ &= c_p - \sum_{j\in N_p}\sum_{f\in F_j} H^j_{fk}\psi_{fj} - \phi_k.\end{aligned}$$

The above expression on reduced cost can also be directly obtained from (PF'_3). This is consistent with the observation in (4.16) such that there is an equivalence relation between an ILP under a network structure embedded in X^k and a shortest path problem defined in the same network, because if we take a deeper insight into (5.4), it would be understood as

$$\begin{aligned}\bar{c}^*_k(\phi,\psi) &= \min_{p\in P^k} \{c^{kT}\hat{x}^p - (H^k\hat{x}^p)^T\psi - \phi_k\} \\ &= \min_{x\in X^k} \left\{ \frac{1}{b^k_0}[c^{kT}x - (H^k x)^T\psi] - \phi_k \right\} \\ &= \min_{x\in X^k} \left\{ \frac{1}{b^k_0}\Big(\sum_{a\in\mathcal{A}^k} c_a x_a - \sum_{j\in N, f\in F_j}\sum_{a\in\delta^k_-(j)} H^j_{fk}\psi_{fj}x_a\Big) - \phi_k \right\}.\end{aligned} \tag{5.5}$$

From the discussions over (4.16), the problem in (5.5) is equivalent to a shortest path problem in $\mathcal{G}^k$ with modified arc costs and its LP relaxation itself automatically solves this ILP. Those modified costs are the same as the ones in the shortest path problem defined by (5.4). Solved as a linear program, as long as there are candidates with negative reduced costs as the path lengths, $\mathbf{0}^k$ will not solve (5.5), although it is an extreme point of X^k. When there is no path with a negative length, the ILP is finished.

As for solving the subproblems as shortest path problems, it may worth mentioning that as the underlying graph $\mathcal{G}^k$ is a DAG, the shortest path problem can be solved very efficiently by topological sorting in the linear rate of $O(|N^k| + |\mathcal{A}^k|)$ [2].

5.2.2 Bounds and stopping criterion

If one solves $(\overline{PF_3})$ by column generation, then during each iteration, a pair of bounds on the master problem's objective value z^*_{MP} can be obtained, and is totally driven by the optimal duals ϕ_k, ψ_{fj} of the current RMP. It is clear that the optimal objective of the RMP $z^*_{\text{RMP}}(\phi,\psi) = \sum_{k\in K} b^k_0\phi_k + h^T\psi$ always provides an upper bound on z^*_{MP}. Moreover Theorem 5.1 will show that a lower bound can be obtained by considering the value of the subproblem results $\bar{c}^*_k, k\in K$ and assuming an upper bound on the sum of the optimal master problem variables can be found [28, 59].

Theorem 5.1. *For each $k \in K$, let $\kappa_k \geq \sum_{p \in P^k} \xi_p^*$, where $\xi_p^*, p \in P^k$ are the elements of type k in the optimal solution of the master problem $(\overline{PF_3})$. Then*

$$z^*_{\text{RMP}}(\phi, \psi) + \sum_{k \in K} \kappa_k \bar{c}^*_k(\phi, \psi) \leq \sum_{k \in K, p \in P^k} c_p \xi^*_p = z^*_{\text{MP}}. \tag{5.6}$$

Proof. Noting that $\sum_{p \in P^k} \xi_p^* \leq b_0^k, \sum_{k \in K, p \in P_j^k} H^j_{fk} \xi^*_p \leq h^j_f$ and $\bar{c}^*_k, \phi, \psi \leq 0$, we have

$$\begin{aligned}
z^*_{\text{MP}} - z^*_{\text{RMP}}(\phi, \psi) &= \sum_{k \in K, p \in P^k} c_p \xi^*_p - \Big(\sum_{k \in K} b_0^k \phi_k + \sum_{f \in F_j, j \in N} h^j_f \psi_{fj} \Big) \\
&\geq \sum_{k \in K, p \in P^k} c_p \xi^*_p - \Big(\sum_{k \in K, p \in P^k} \xi^*_p \phi_k + \sum_{f \in F_j, j \in N, k \in K, p \in P_j^k} H^j_{fk} \xi^*_p \psi_{fj} \Big) \\
&= \sum_{k \in K, p \in P^k} c_p \xi^*_p - \Big(\sum_{k \in K, p \in P^k} \xi^*_p \phi_k + \sum_{f \in F_j, j \in N_p, k \in K, p \in P^k} H^j_{fk} \xi^*_p \psi_{fj} \Big) \\
&= \sum_{k \in K, p \in P^k} \xi^*_p \Big(c_p - \phi_k - \sum_{f \in F_j, j \in N_p} H^j_{fk} \psi_{fj} \Big) \\
&\geq \sum_{k \in K, p \in P^k} \xi^*_p \bar{c}^*_k(\phi, \psi) \\
&\geq \sum_{k \in K} \kappa_k \bar{c}^*_k(\phi, \psi),
\end{aligned}$$

which finishes the proof. □

Corollary 5.1. *At iteration i in the column generation process in solving $(\overline{PF_3})$, let ϕ^i, ψ^i be the corresponding optimal dual variables. A pair of bounds on the optimal objective value of the master problem $(\overline{PF_3})$ can be obtained as*

$$z^*_{\text{RMP}}(\phi^i, \psi^i) + \sum_{k \in K} \kappa_k \bar{c}^*_k(\phi^i, \psi^i) \leq z^*_{\text{MP}} \leq z^*_{\text{RMP}}(\phi^i, \psi^i) \tag{5.7}$$

From Corollary (5.1), one would expect that at iteration i if $\bar{c}^*_k(\phi^i, \psi^i) = 0, \forall k \in K$ then the upper bound equals to the lower bound and the column generation terminates, which is the case if the process is viewed as a simplex method or as a separation problem.

Note that although the upper bound is always monotonically decreasing as the column generation proceeds, it is not true that the lower bound is always increasing. Oscillation somehow may occur, e.g. the lower bound may decrease at some iterations and increase again. Therefore a tighter pair of bounds at iteration i would be to always use the best lower bound so far among all iterations, as

$$\max_{l=1,\ldots,i} \Big\{ z^*_{\text{RMP}}(\phi^l, \psi^l) + \sum_{k \in K} \kappa_k \bar{c}^*_k(\phi^l, \psi^l) \Big\} \leq z^*_{\text{MP}} \leq z^*_{\text{RMP}}(\phi^i, \psi^i). \tag{5.8}$$

As for choosing appropriate values for parameters κ_k, here two strategies will be discussed. There may be more alternatives apart from these.

Setting $\kappa_k = b_0^k$ on all iterations First it is reasonable to set $\kappa_k = b_0^k$ for each $k \in K$ on all iterations during the column generation process. The validity is ensured by the fleet size bounding Constraints (4.27). Then the lower bound becomes $z^*_{\mathrm{RMP}}(\phi,\psi) + \sum_{k\in K} b_0^k \bar{c}_k^*(\phi,\psi)$. This is the same lower bound if the original arc formulation (AF_3'') is solved by Lagrangian relaxation with the same dual variables ψ associated with the penalised Constraints (4.18). To see this, let $L(\psi)$ be the Lagrangian lower bound and $\mathcal{L} = \max_{\psi\le 0} L(\psi)$ be the optimal solution of the Lagrangian dual problem. Then we have

$$\begin{aligned} L(\psi) &= \min_{x^k\in X^k, k\in K} \left\{ \sum_{k\in K} c^{kT} x^k - \psi^T \Big(\sum_{k\in K} H^k x^k - h \Big) \right\} \\ &= \sum_{k\in K} b_0^k \min_{x^k \in X^k} \left\{ \frac{1}{b_0^k} [c^{kT} x^k - \psi^T (H^k x^k)] \right\} + \psi^T h \\ &= \sum_{k\in K} b_0^k \min_{p\in P^k} \{ c^{kT} \hat{x}^p - \psi^T (H^k \hat{x}^p) \} + \psi^T h. \end{aligned} \tag{5.9}$$

The resulting Lagrangian dual in finding $\mathcal{L}$ is often solved by some subgradient methods. However it can be understood as an LP in the following way by adding extra variables ϕ_k for all k, as used in [28], such that

$$\mathcal{L} = \max \quad \sum_{k\in K} b_0^k \phi_k + \psi^T h \tag{5.10}$$

$$\text{s. t.} \qquad \phi_k \le c^{kT}\hat{x}^p - \psi^T(H^k \hat{x}^p), \quad \forall p \in P^k, \forall k \in K; \tag{5.11}$$

$$\phi, \psi \le \mathbf{0}. \tag{5.12}$$

One can verify the Lagrangian dual problem (5.10)–(5.12) is exactly the dual problem of the path formulation (PF_3). Thus given the shared variables ψ plus another variable ϕ that can be regarded as a constant, the Lagrangian lower bound at ψ has the same value as the lower bound given by (5.6) with $\kappa_k = b_0^k$, i.e.,

$$\begin{aligned} L(\psi) &= \sum_{k\in K} b_0^k \min_{p\in P^k} \{ c^{kT}\hat{x}^p - \psi^T (H^k \hat{x}^p) - \phi_k \} + \psi^T h + \sum_{k\in K} b_0^k \phi_k \\ &= \sum_{k\in K} b_0^k \bar{c}_k^*(\psi,\phi) + \psi^T h + \sum_{k\in K} b_0^k \phi_k. \end{aligned} \tag{5.13}$$

Setting $\kappa_k = \sum_{p\in P^k} \tilde{\xi}_p^*$ at each iteration Noting that the sum of path variables is only upper bounded rather than fixed, a better lower bound can be found dynamically at each iteration of the column generation process. Let $\tilde{\xi}^*(\phi,\psi)$ be the optimal solution of the RMP with dual variables ϕ, ψ, then it is true that $\sum_{p\in P^k} \xi_p^* \le \sum_{p\in P^k} \tilde{\xi}_p^*(\phi,\psi) \le b_0^k$ for each $k \in K$. Therefore, by setting $\kappa_k = \sum_{p\in P^k} \tilde{\xi}^*(\phi,\psi)$ promptly in each round of column generation LP, a tighter lower bound on z^*_{MP} can be obtained as $z^*_{\text{RMP}}(\phi,\psi) + \sum_{k\in K}\sum_{p\in P^k}\tilde{\xi}^*_p(\phi,\psi)\bar{c}^*_k(\phi,\psi)$.

Having the lower bound at hand at each LP iteration during column generation can benefit the solution process in many ways. One important aspect is that it allows one to stop column generation before the master problem is solved to optimality. To be specific, column generation requires solving the RMP iteratively until no subproblem with negative valued $\bar{c}^*_p$ is observed. This can often cause an excessive number of LP iterations to get an optimal solution, especially at the later part with very slow improvement on the objective. In practice, especially during a branch-and-bound process for finding integer solutions, one may stop the process before all subproblems have yielded non-negative reduced costs, on condition that the lower bound and upper bound given by (5.8) are close enough and/or the number of LP iterations exceeds a given threshold. Then the lower bound can be used as the node value (a lower bound on the potential integer value from its offspring) of the BB tree which is still a valid value anyway.

5.2.3 Applying column generation to other formulations

The framework of the above process of column generation applied to solve the LP relaxation of (PF_3) can be borrowed in solving the LP relaxation of other formulations. Take (AF_2) for example. It has a corresponding path formulation (PF_2) as

$$(PF_2)\quad \min \quad C_1\sum_{k\in K}\sum_{p\in P^k} c_p x_p + C_2\sum_{a\in\mathcal{A}} y_a + C_3\sum_{d\in D}\sum_{k\in K}\left(z^+_{d,k}+z^-_{d,k}\right) \tag{5.14}$$

$$\sum_{p\in P^k} x_p \le b^k_0, \quad \forall k\in K \tag{5.15}$$

$$\sum_{k\in K_j}\sum_{p\in P^k_j} H^j_{f,k}x_p \le h^f_j, \quad \forall f\in F_j, \forall j\in N \tag{5.16}$$

$$\sum_{k\in K_a}\sum_{p\in P^k_a} x_p \le u_a y_a, \quad \forall a\in\mathcal{A} \tag{5.17}$$

$$\tau^D_{\text{arr}(i)}\left(\sum_{a\in\delta_+(i)} y_a - 1\right) + \tau^C_{\text{dep}(j)}\left(\sum_{a\in\delta_-(j)} y_a - 1\right) \le e_{ij}, \quad \forall (i,j)\in A^* \tag{5.18}$$

$$\sum_{a\in\delta_-(j)} y_a = 1, \quad \forall j\in N^-_B \tag{5.19}$$

$$\sum_{a\in\delta_+(j)} y_a = 1, \quad \forall j\in N^+_B \tag{5.20}$$

$$z^{+}_{d,k} - z^{-}_{d,k} - \left(\sum_{a \in (A^k_d)^{\text{on}}} \sum_{p \in P_a} x_p - \sum_{a \in (A^k_d)^{\text{off}}} \sum_{p \in P_a} x_p \right) = 0, \quad \forall d \in D, \forall k \in K \tag{5.21}$$

$$x_a \in \mathbb{Z}_+, \quad \forall a \in \mathcal{A}^k, \forall k \in K \tag{5.22}$$

$$y_a \in \{0, 1\}, \quad \forall a \in \mathcal{A}; \tag{5.23}$$

$$z^{+}_{d,k}, z^{-}_{d,k} \in \mathbb{R}_+, \quad \forall d \in D, \forall k \in K \tag{5.24}$$

Suppose an RMP of the master problem of (PF_2) is based on a subset of paths (with the complete set of arcs included), then it is sufficient to only detect whether there are promising paths from each $\mathcal{G}^k$. Let the dual variables $\phi_k \leq 0, \psi_{fj} \leq 0$ be associated with Constraints (5.15) and (5.16) respectively. A set of dual variables $\eta_a \leq 0$ linked with Constraints (5.17) is also needed. Moreover, note that the depots $d \in D$ would partition the sign-on/-off arcs in a way such that each sign-on/-off arc, as well as each path, will be associated with one sign-on depot (the depot linked with the path's sign-on arc) and one sign-off depot (the depot linked with the path's sign-off arc). Let $d^{\text{on}}_p, d^{\text{off}}_p$ denote the sign-on/-off depot belonged to path p, and let $\theta_{dk} \lesseqgtr 0$ be the dual variables associated with Constraints (5.21). Then the reduced cost of path p in the k-th subproblem can be written as

$$\bar{c}_p = c_p - \phi_k - \sum_{j \in N_p, f \in F_j} H^j_{fk} \psi_{fj} - \sum_{a \in \mathcal{A}_p} \eta_a - \theta_{d^{\text{on}}_p, k} + \theta_{d^{\text{off}}_p, k}, \tag{5.25}$$

finding the smallest value of which is also a shortest path problem with a train node weight $-\sum_{f \in F_j} H^j_{fk} \psi_{fj}$ for j, an arc weight $-\eta_a$ for a, a path weight $c_p - \phi_k$ for p plus a sign-on arc weight $-\theta_{d^{\text{on}}_p, k}$ and a sign-off arc weight $\theta_{d^{\text{off}}_p, k}$.

5.3 Branching strategies

In order to get integer valued solutions, a popular method is to use the branch-and-bound (BB) tree. A detailed description on a generic BB method can be found in many optimisation/operational research books, e.g. [65] and [81]. A major difference in the BB method that is going to be presented here, however, is that besides making the fractional variables integral, it also eliminates coupling among incompatible unit types for all trains and forbids coupling/decoupling operations at banned/restricted locations. Thus the purpose of the extended BB method will be referred to as "to find an optimal *operable* solution" where an operable solution means a solution such that (i) all (path or arc) variables are integral, (ii) each train is served by a unit block made of compatible types, and (iii) there is no coupling/decoupling activity at banned/restricted locations. Moreover, the term "feasible" will be solely used in the context for showing whether an LP (relaxation) problem is feasible or infeasible, and may be explicitly called “LP-feasible”. It should not be confused with the definition on the above “operable” whose counterpart in a generic BB method is “integral”.

Branch-and-bound is based on a divide-and-conquer framework using implicit enumeration. Take formulation (PF_3) as an example. First the root of the BB tree represents its LP relaxation ($\overline{PF_3}$) without any branching. Then several leaf nodes[1] as the children of the root will be generated such that the solution space is partitioned by the children to get rid of some inoperable solutions while preserving all potential operable solutions. This process is called branching. To further partition the solution space, children nodes will have grandchildren and grandchildren will have their descendants and so forth. Each child will inherit the branching from its parent plus its own solution space partitioning scheme.

Every node that is LP-feasible provides a lower bound on the values of all its descendants, including any possible operable ones. On the other hand, each node that is operable always provides an upper bound on the optimal operable solution. A global upper bound on the optimal operable solution is the one with the smallest value among all operable solutions and the associated solution is referred to as the incumbent operable solution. Initially at the root the global upper bound can be set as $+\infty$, although in practice this setting can be more flexible. A node will be cut-off (which means no branching will ever occur there) if any of the three conditions is met:

(i) The node is LP-infeasible (cut-off by infeasibility).

(ii) The node is operable (cut-off by operability).

(iii) The node is LP-feasible but has a larger value than the global upper bound (cut-off by bound).

If a node does not fall into any of the above criteria, i.e. it is LP-feasible with a smaller value than the global upper bound but not yet operable, then it is put into the active node queue and may be branched later. The active nodes with the smallest value will provide a best lower bound on the optimal operable solution. The BB process is continued by selecting a node from the active queue and branching on it. Here come the two main issues to be concerned in making an efficient BB method, i.e. branch design and node selection, which are often very much problem-specific. Branch design refers to how to partition the solution space at an active node. Node selection refers to how to select active nodes from the active queue. Both of them will have a significant impact on the efficiency or even the workability of a BB method. The details of the branch design and node selection of the branch-and-price solver for the train unit scheduling problem will be discussed later in § 5.3.1.

[1]The term “node” here on the BB tree, which can be either a leaf or a root, should not be confused with the “node” in a network.

Normally the BB process will stop when one of the two conditions is met:

(i) The global upper bound is equal to the best lower bound

(ii) The active queue is empty without any operable solution found

Condition (i) means an operable node is found and its value is the same as the best (smallest) lower bound thus this operable solution is also an optimal solution. Condition (ii) means the BB tree has been exhaustively searched and there is no solution satisfying the problem thus one can claim the entire problem is inoperable ("ILP-infeasible" in a generic BB method). Other strategies may also be used. For example, when a problem is extremely difficult, one may consider to be just satisfied with finding an operable solution and stop the BB process. On the other hand, when a problem is relatively easy, one can let the BB search continue even after an operable solution has been found while the active queue is not empty yet, in order to find more optimal alternatives.

5.3.1 Branch design

There can be many ways to form branches. For example, if it is required that $x_1 \in \mathbb{Z}_+$ and a fractional solution $x_1 = 0.56$ is found in a parent node. Then two branches can be made such that the left children requires $x_1 \leq \lfloor 0.56 \rfloor = 0$ (thus simply $x_1 = 0$) and the right child requires $x_1 \geq \lceil 0.56 \rceil = 1$. The most straightforward way to realise such requirements would be to add the above constraints explicitly into the LP. However, a large number of constraints may seriously slow down the solution process. One may often consider better alternatives to realise the same effect. For instance, to impose $x_1 = 0$ it is equivalent to just delete the object "1" representing x_1 both in the RMP and the subproblems. This alternative not only avoids adding new constraints but also reduces the sizes of the RMP and the subproblems.

The essence of branching is to partition the solution space by discarding inoperable solutions without losing any potential operable ones. Therefore, one may not necessarily form the branches explicitly on individual fractional variables. Suppose a tiny-sized problem with 500 variables that all have to be binary. Then in the very worst case one may need to form $2^{500} \approx 3.28 \times 10^{150}$ branches. Even if branch-and-bound uses partial enumeration, the resulting average number will still be huge. In commonly seen practical problems the number of variables could be from thousands to millions, and may not be merely binary but as general integers. Branching solely on fractional variables will thus be too impractical. This is the famous "curse of dimensionality" arising in many discrete problems. Therefore, one is forced to seek alternative branching objects according to problem features.

In a traditional BB framework, branching is generally employed to make fractional variables integral. Nevertheless, in this research for train unit scheduling, branching is also used as a method to realise the requirements that are difficult to satisfy by ILP constraints.

If used appropriately, such branches can both satisfy those tough requirements as well as effectively divide the solution space such that they can make the "fractional-to-integer" process to be performed more easily and efficiently.

A multi-rule branching system reflecting the above issues is designed for the branch-and-price solver, where certain priorities are given to each branching rule and they can be altered according to the problem status. The branching rules are designed to follow the principles below as much as possible,

- The solution space is partitioned by deleting columns rather than adding LP constraints.
- Multiple columns can be removed as a result of one branch.
- The rules are compatible with the branch-and-price framework.

There are three branching rules involved in this system, namely train-family branching, banned location branching, and arc variable branching.

5.3.1.1 Train-family branching

In forming the train unit combination set W^j in (4.6), the combinations with incompatible types are not included in this discrete set. However as discussed in § 4.1.2 some of the invalid combinations can be still inside the continuous set $\text{conv}(W^j)$ and thus may appear in a relevant LP relaxation solution.

Therefore a branch design named train-family branching is presented to eliminate the remaining type-incompatible combinations. To form such branches, the solution of the relevant LP relaxation at a BB tree node will be checked and a train covered by more than one family will be identified. Currently the checking order among trains for finding out such a multi-family train is set to follow the time line, i.e. trains are sorted according to their departure times and the checking is performed from the earliest train to the latest train. If a train is found to be multi-family covered, then the checking is stopped and the train to be branched is thus identified. Suppose such a train j is detected with families $\varphi_1, \dots, \varphi_n$ (n is usually not a large number) serving it and let $\mathit{\Phi}_j$ be the set of all families allowed to serve j. Then $n+1$ branches will be formed in the following way.

- For the first n branches $1, \dots, n$, say at the i-th branch where $i \in \{1, \dots, n\}$, only family φ_i is allowed to serve train j. To achieve this, in the RMP all paths indicating any family in $\mathit{\Phi}_j \setminus \{\varphi\}$ serves j will be deleted; in the shortest path subproblem of type k not belonging to φ_i, node j will be deleted from the shortest path network.
- For the last $(n+1)$-th branch, if $\mathit{\Phi}_j \setminus \{\varphi_1, \dots, \varphi_n\} \neq \emptyset$, then families $\varphi_1, \dots, \varphi_n$ will be forbidden to serve j, which can be realised by similar path/node deleting schemes as described above; if $\mathit{\Phi}_j \setminus \{\varphi_1, \dots, \varphi_n\} = \emptyset$, then the $(n+1)$-th branch is no longer needed.

Algorithm 1 Train-family branching with heuristics

```
Given:
  a current branching status using train-family branching
  an LP relaxation solution at a BB tree node $n_{\mathrm{BB}}$
  a list $L$ of sorted trains in $N$ according to their departure times
Result: the latent branches (“buds”) stored in $n_{\mathrm{BB}}$ for future use
Begin:
initialise $\Phi \leftarrow \emptyset$
for all $j \in L$ do
  initialise $\Phi_j \leftarrow \emptyset$
  for all $\varphi \in \Phi_j$ do
    if $\varphi$ serves $j$ then
      $\Phi_j \leftarrow \Phi_j + \{\varphi\}$
    end if
  end for
  if $|\Phi_j| \geq 2$ then
    $\Phi \leftarrow \Phi_j$
    break
  end if
end for
if $\Phi = \emptyset$ then
  stop and switch to another branching rule having a lower priority
else
  set $0 \leq \rho_T \leq 1$
  initialise $\Phi' \leftarrow \emptyset$
  for all $\varphi \in \Phi$ do
    if $\frac{\sum_{k \in K_j : \mathrm{family}(k) = \varphi} w_k^j}{\sum_{k \in K_j} w_k^j} \geq \rho_T$ then
      $\Phi' \leftarrow \Phi' + \{\varphi\}$
    end if
  end for
  for all $\varphi \in \Phi'$ do
    form a bud for $n_{\mathrm{BB}}$ such that only $\varphi$ serves $j$
  end for
end if
```

This branch design rule does not add any extra constraints to the RMP. Moreover, it can reduce the number of columns in the RMP and the scales of the subproblem shortest path networks. Although its major aim is to force all trains to be served by compatible families, a by-product would be it may also effectively partition the solution space such that the later fractional-to-integral process can be eased.

In practice, heuristics can be embedded into the train family branching process, particularly for harder problems with a large number of families available for each train. A parameter ρ_T is set to let the process only consider families with certain amount of coverage in serving j, rather than all of the used families. This parameter can also be set dynamically according to the current status of the BB tree. The scheme of the train-family branch design with heuristics is given in Algorithm 1.

5.3.1.2 Banned location branching

The banned location branching rule aims at ensuring no coupling/decoupling operation takes place at trains banned/restricted for coupling/decoupling in $N_B^- \cup N_B^+$. This branching rule is very important for ILP formulations without the block-arc variables y_a and relevant constraints. It will be also useful for the ILP formulations with variables y_a since this branching rule basically realises similar effect in forcing the fractional block-arc variables involved with banned location trains into binaries.

The main idea of banned location branching has some similarity with train-family branching. It will check the LP relaxation solution at a BB tree node and select a banned location train j among N_B^- or N_B^+ that has multiple used block-arcs in $\delta_-(j)$ or $\delta_+(j)$ in the LP relaxation solution. The checking order is also set to follow the time line of the banned location trains on their departure times since experience has shown that it is efficient. Once such a train has been found the checking stops. Suppose train j has been identified with multiple flowed block-arcs $a_1, \ldots, a_n$ (either incoming or outgoing but not both) and let $A_j = \delta_-(j)$ (when $j \in N_B^-$) or $A_j = \delta_+(j)$ (when $j \in N_B^+$). Then in principle $n+1$ branches can be formed such that

- For the first n branches $1, \ldots, n$, say at the i-th branch where $i \in \{1, \ldots, n\}$, among all arcs in A_j, only block-arc a_i is allowed to be flowed. To achieve this, in the RMP all paths containing any arcs in $A_j \setminus \{a_i\}$ will be deleted; in the shortest path subproblems for all types, arcs in $A_j \setminus \{a_i\}$ will be deleted from the shortest path network.

- For the last $(n+1)$-th branch, if $A_j \setminus \{a_1, \ldots, a_n\} \neq \emptyset$, then block-arcs $a_1, \ldots, a_n$ will be all removed by similar path/node deleting schemes as above; if $A_j \setminus \{a_1, \ldots, a_n\} = \emptyset$ (which is very unlikely to happen in practice), then the $(n+1)$-th branch is no longer needed.

Similar to train-family branching, banned location branching does not add any extra constraints and can reduce the problem sizes both in the RMP and the subproblems. It forces all banned location trains to have only one incoming and/or outgoing block-arc flowed and is also able to effectively partition the solution space. Similar heuristics as for train-family branching can also be embedded into banned location branching with a parameter ρ_B to only consider arcs with a certain flow magnitude. The scheme of the banned location branch design with heuristics is given in Algorithm 2. Note that the banned location branching is sufficient to eliminate all violated coupling/decoupling operations for the formulations with or without block-arc variables $y_a, \forall a \in \mathcal{A}$.

Algorithm 2 Banned location branching with heuristics

```
**Given:**
  a current branching status using banned location branching
  an LP relaxation solution at a BB tree node $n_{\mathrm{BB}}$
  a list of banned location trains $L \subset N_B^- \cup N_B^+$ sorted according to their departure times
**Result:** the latent branches (“buds”) stored in $n_{\mathrm{BB}}$ for future use
**Begin:**
initialise $A \leftarrow \emptyset$; let $b(a)$ be the block-arc that a type-arc $a$ belongs to and let $T(a)$ be the set of the corresponding type-arcs a block-arc $a$ associated with
**for all** $j \in L$ **do**
  initialise $A_j \leftarrow \emptyset$
  **if** $j \in N_B^-$ **then**
    **for all** $a \in \delta_-^k(j), k \in K$ **do**
      **if** $x_a \geq 0$ **then**
        $A_j \leftarrow A_j + \{b(a)\}$
      **end if**
    **end for**
  **else**
    **for all** $a \in \delta_+^k(j), k \in K$ **do**
      **if** $x_a \geq 0$ **then**
        $A_j \leftarrow A_j + \{b(a)\}$
      **end if**
    **end for**
  **end if**
  **if** $|A_j| \geq 2$ **then**
    $A \leftarrow A_j$
    break
  **end if**
**end for**
**if** $A = \emptyset$ **then**
  stop **and** switch to another branching rule having a lower priority
**else**
  set $0 \leq \rho_B \leq 1$
  initialise $A' \leftarrow \emptyset$
  **for all** $a \in A$ **do**
    **if** $\frac{\sum_{t \in T(a)} x_t}{\sum_{t \in T(a), a \in A} x_t} \geq \rho_B$ **then**
      $A' \leftarrow A' + \{a\}$
    **end if**
  **end for**
  **for all** $a \in A'$ **do**
    form a bud for $n_{\mathrm{BB}}$ such that only $a$ is kept in $\delta_-(j)$ or in $\delta_+(j)$
  **end for**
**end if**
```

5.3.1.3 Arc variable branching

As for finally finding a solution with all-integer variables, it is generally not enough to drive all variables to integers solely by train-family branching and/or banned location branching. When both of them have been thoroughly explored, that is, all trains have been served by compatible types and all coupling/decoupling operations have been banned at certain locations, and there are still fractional variables in the LP relaxation, then the branching rule will be switched to arc variable branching on individual type-arc variables. Even if the RMP is a path formulation, variable branching can still be performed based on arcs whose values can be accessed by $x_a = \sum_{p \in P_a^k} x_p$. Usually it is not easy and is not encouraged to design a branch rule based on individual paths within a column generation context.

When the branch design comes to arc variable branching, it is still a complex issue to decide which arc(s) to be branched among all fractional arcs. One principle would be to select the arc that may yield a greater change in the objective value if it is branched. However, it is often very difficult to make a precise prediction on which arc(s) will give the greatest change. Experiences suggest one may branch on the "most fractional" arc whose decimal part is nearest to 0.5. Alternatively, one may branch on the arc with the largest pseudo-cost [57]. In the particular case of train unit scheduling, one may give priority to those arcs associated with the peak hour range, or one may first branch on the arcs associated with the train nodes allowing coupling/decoupling operations (therefore allowing the use of multiple incoming/outgoing arcs). As for realising such branches in the LP, one may have to insert explicit constraints. For instance, one can use the method given by Alvelos [4], where a fractional valued arc $a \in \mathcal{A}^k$ is branched by two constraints as

$$\sum_{p \in P_a^k} x_p \leq \lfloor x_a \rfloor, \quad \sum_{p \in P_a^k} x_p \geq \lceil x_a \rceil. \tag{5.26}$$

Using the above branching constraints does not totally eliminate the possibility of some short cuts. For example, if $\lfloor x_a \rfloor = 0$ then arc a can be deleted by removing the associated paths in the RMP and the associated arcs in the subproblem networks instead of using the constraints explicitly.

5.3.1.4 The order of the branching rules

Employing several branching rules, it is necessary to control their order to be used during the branching process. For instance, train-family branching is considered more advantageous than arc variable branching and thus if both of them can be used, the system would like to choose the former. In this case it is said that the rules have an order that "train-family branching $\succ$ arc variable branching". Another issue in such an ordering system is the computational efficiency in checking the rules' availabilities. Taking the example of train-family and arc variable branching, it is not necessary to have an overall check in finding a train served by multiple families *and* an arc variable that is fractional. If train-family branching is put before arc variable branching in the rule order, until all trains are served by single-family covered unit blocks such that it is not possible to use train-family branching, there is no need to check for fractional variables. After the train-family branching has been exploited, the checking has to be switched to focusing on another rule, e.g. fractional arc variables. Therefore, the order imposed on different branching rules are in fact the order how the checking on different rules are going to perform such that once available branching objects are found, the checking in the current round will stop. If no such an object is available, the checking will have to be switched to the next inferior rule as the current rule has been temporarily exploited.

The (checking) order on branching rules can either always remain the same (static) or have some change according to the current status on the BB tree (dynamic). In either case, an initial list L_0 representing an order at the beginning state upon the rules $r_1, \ldots, r_n$ is set as

$$L_0 = \langle r_1, r_2, \ldots, r_n \rangle, \tag{5.27}$$

which states that $r_1 \succ r_2 \succ \cdots \succ r_n$. The initial list L_0 can be devised according to the general preferences on the rules, for example, "train-family $\succ$ banned location $\succ$ arc variable". Based on that, three kinds of branching rule ordering systems will be introduced.

(i) *Static rule ordering*

If the branching rule order always remains the same as the initial list L_0, then the ordering is static, which is simple and straightforward. This static rule ordering is specified in Algorithm 3. In that algorithm, `check`(r) is defined as a function to check whether there are available objects regarding rule r in the LP relaxation solution at BB tree node n_{BB}. If there are objects available, it will return to TRUE; otherwise it will return to FALSE.

Algorithm 3 Static branching rule order

```
Given:
  a list of ordered branching rules $L_0 = \langle r_1, \ldots, r_n \rangle$
  an LP relaxation result at a BB tree node $n_{\text{BB}}$
Result: the chosen branching rule $r^*$ stored in $n_{\text{BB}}$ for later use
Begin:
initialise $r^* \leftarrow null$
for all $r \in L_0$ do
  if check($r$) = TRUE then
    $r^* = r$
    break
  end if
end for
if $r^* = null$ then
  an operable solution found at $n_{\text{BB}}$
end if
```

The static ordering in Algorithm 3 can always ensure the most desirable rules are used as long as they are available, at the price of possible slower computational efficiency though. This is because when a rule r has once been (temporarily) used up and thus the system will have to use the next inferior rule of r in L_0 (denoted by $I_0(r)$) for some time, r will still be checked before $I_0(r)$ every time, which is inefficient as time used to search a branching object can be considerable when the network size is large. Note that when a rule r is said to be “exploited” or “used up”, it is for the current checking round only. It is possible for some branching objects on r to be available again later; r can be used up again, and available objects reappear, and so forth. Despite the fact that available objects may later pop up again, since rule r has been exploited once before, such occasions are likely to be much less frequent than the presence of a branching object on other rules including its next inferior rule $I_0(r)$. Therefore the efficiency in branching object searching might be improved if the positions of r and $I_0(r)$ are swapped in L_0 as long as the first occasion that rule r is used up is detected. This gives the idea of the following dynamic rule ordering named “switch@new-rule”.

(ii) *Dynamic rule ordering: switch@new-rule*

Based on the above observation, a dynamic branching rule order system is presented which is referred to as “switch@new-rule”. In this system, the rule order in the list is frequently updated according to the current status. In a checking round based on list L_j, if the chosen rule r^* is different from the head of L_j, then L_j will be updated to L_{j+1} such that r^* is placed as the head of L_{j+1} and the rest rules will be ordered according to their priority relation in L_0 and concatenated to r^*, i.e. $L_{j+1} = \langle r^*, L_0 \setminus \langle r^* \rangle \rangle$. The checking order stored in L_{j+1} will be then locally passed down to all the children nodes of the current node being checked and continued to be inherited by the descendants for some generations, as long as no new rule pops up to update L_{j+1}. Such a change can also be globally imposed on all selected active nodes beyond the current node’s descendants. However, restricting the updated list only on the descendants may be more reasonable when a node selection method making frequent jumps at different levels of the BB tree is used, e.g. methods based on the best-first searching.

A simple example on switch@new-rule with five rules is given in (5.28), where the rule underlined refers to the one chosen differing from the head rule such that an old list is going to be updated to a new list.

$$\begin{aligned}
L_0 &= \langle r_1, \underline{r_2}, r_3, r_4, r_5 \rangle \\
L_1 &= \langle r_2, r_1, \underline{r_3}, r_4, r_5 \rangle \\
L_2 &= \langle r_3, r_1, r_2, \underline{r_4}, r_5 \rangle \\
L_3 &= \langle r_4, r_1, r_2, r_3, \underline{r_5} \rangle \\
L_4 &= \langle r_5, \underline{r_1}, r_2, r_3, r_4 \rangle \\
L_5 &= \langle r_1, r_2, r_3, r_4, r_5 \rangle
\end{aligned} \tag{5.28}$$

The dynamic switch@new-rule ordering is specified in Algorithm 4.

(iii) *Dynamic rule ordering: switch@stagnation*

Algorithm 4 Dynamic switch@new-rule

```
Given:
  a list of ordered branching rules $L_j = \langle r_1, \dots, r_n \rangle$
  an LP relaxation result at a BB tree node $n_{\mathrm{BB}}$
Result: the chosen branching rule $r^*$ stored in $n_{\mathrm{BB}}$ for later use and perhaps an updated rule list
Begin:
initialise $r^* \leftarrow null$
for all $r \in L_j$ do
   if check($r$) = TRUE then
      $r^* := r$
      if $r^* \neq head(L_j)$ then
         $L_{j+1} = \langle r^*, L_0 \setminus \langle r^* \rangle \rangle$
      end if
      break
   end if
end for
if $r^* = null$ then
   an operable solution found at $n_{\mathrm{BB}}$
end if
```

The second dynamic rule ordering uses a different point of view and may only be suitable for problems with some special characteristics. It can also be regarded as an extension of "switch@new-rule" with an additional protective ability against stagnation on a rule. Having multiple branching rules and an initial preference order dictated by L_0, sometimes the branching can be "trapped" into a subset of rules. For instance, banned location branching is regarded to be more advantageous than arc variable branching. However sometimes the number of banned location nodes is huge and there are too many used arcs associated with them in an LP relaxation solution which makes the process at banned location branching seem to be endless—the process stagnates at banned location branching. When such a case occurs, switching to another rule rather than fixating on the current rule may be more helpful. For instance, at this moment if one switches the rule to arc variable branching and uses it for some rounds, then it is likely that as the solution space has been reduced by more and more integral arc variables, the number of possibilities in banned location branching objects have also been greatly reduced such that when one comes back to banned location branching (after or before the exploitation on arc variable branching), it is possible to make all coupling/decoupling as required within reasonable time.

To establish the above idea, for each rule r in the current list L_j, when r is the head of L_j, the number of times that r has been used without any natural change (i.e. changes due to r is used up thus another rule has to be triggered) is recorded as *unchanged*(r). If any natural change happens, then *unchanged*(r) will be reset to 0 and the list will be updated like in "switch@new-rule". If *unchanged*(r) exceeds a preset threshold value M_{UC}, then the system will force the use of a new list L_{j+1} such that the head is changed from r to $I_0(r)$, the second rule is r and the rest rules are sorted as in L_0 following $I_0(r)$ and r, i.e. $L_{j+1} = \langle I_0(r), r, L_0 \setminus \langle r, I_0(r) \rangle \rangle$. The algorithmic description is given in Algorithm 5.

Algorithm 5 Dynamic switch@stagnation

```
  Given:
    a list of ordered branching rules $L_0 = \langle r_1, \dots, r_n \rangle$
    an LP relaxation result at a BB tree node $n_{\mathrm{BB}}$
    head rule's $unchanged(head)$ number and the threshold $M_{UC}$
  Result: the chosen branching rule $r^*$ stored in $n_{\mathrm{BB}}$ for later use and perhaps an updated rule list
  Begin:
  initialise $r^* \leftarrow null$
  for all $r \in L_j$ do
    if check($r$) = TRUE then
      $r^* := r$
      if $r^* \neq head(L_j)$ then
        $L_{j+1} = \langle r^*, L_0 \setminus \langle r^* \rangle \rangle$
        $unchanged(head) := 0$; $unchanged(r^*) := 0$
      else
        $unchanged(r^*) := unchanged(r^*) + 1$
        if $unchanged(r^*) > M_{UC}$ then
          $L_{j+1} := \langle I_0(r^*), r^*, L_0 \setminus \langle r^*, I_0(r^*) \rangle \rangle$
          $unchanged(r^*) := 0$; $unchanged(I_0(r^*)) := 0$
        end if
      end if
      break
    end if
  end for
  if $r^* = null$ then
    an operable solution found at $n_{\mathrm{BB}}$
  end if
```

The above dynamic rule ordering mechanism is called "switch@stagnation". It is not as widely applicable as the "static" or "switch@new-rule" for the train unit scheduling instances tested in this research. Only when the branching process really stagnates at a certain rule will this method be called and once the process has succeeded in jumping out from the troublesome rule, the rule ordering should be driven back to "static" or "switch@new-rule".

5.3.2 Node selection method

Node selection in a branch-and-bound process refers to the decision making on how to select a next active node to solve from the active queue after the current node has been solved and the BB process needs to be continued. Here "to solve" means to get the node's objective value by, e.g., LP relaxation, to determine its latent branches with a kind of branching rule and store them in the node as buds (if possible), and to put the node into the active queue or to prune it depending on the node's solution status. Since the active queue can often contain a large number of active nodes with different attributes such as depths, values and branch design assignments, it is important to use an efficient node selection method that will give faster convergence to the optimal operable solutions.

While designing a node selection method for a specific problem is never a trivial issue, two basic rules are often mentioned as a start point for further devising more specific and sophisticated methods, i.e.,

(i) Best-first: always select the active node with the best objective value as the next node to solve (e.g. the smallest value in a minimisation problem). This strategy can be regarded as a special kind of generalised breadth-first in the context of generic tree searching algorithms where objective values rather than depths are used for grouping the nodes at the same "levels". In order to facilitate a best-first scheme, it is helpful to let the active queue be maintained ascendingly by the node objective values (for a minimisation problem) or descendingly (for a maximization problem) each time after a new active node has been inserted into it.

(ii) Depth-first: let the current node to give a child (if possible) and select it as the next node to solve; if no child can be given when the current node has just been cut-off, do a backtracking in a last-in-first-out (FILO) stack order to one of the current node's ancestors. This depth-first method is similar as the depth-first search in a context for generic tree searching algorithms. It can lead to many "diving actions" or "dives" that are defined as a branching process performed consecutively and directly from a parent to one of its children and to its grandchildren, etc., such that the process proceeds deeper and deeper level-by-level without any interruption. Such dives can lead the searching process to very deep places on the BB tree and are often the only way to obtain operable solutions. The active queue for the depth-first method is generally formed as an FILO stack which is simpler to maintain compared with the sorted active queue for best-first.

The above two node selection methods each focuses on one aspect for finding an optimal operable solution—the former the optimality and the latter the operability. When an operable solution is found by best-first, its quality is often guaranteed to be the best, as suggested by the name. However a major drawback of this method is that it often takes quite long time to find an operable solution, because it tends to select nodes at the top levels of the BB tree such that effective dives are less frequently triggered. Often, it is not until the moment most of the best nodes at the top-levels are exploited that the process can start to search in deeper places. Also best-first can make the "cut-off-by-bound" option be hardly useful as it is rare to find sub-optimal operable solutions to provide upper bounds at the early and middle stages during the entire searching. On the other hand, depth-first can often find a lot of operable solutions in a relatively short time due to its frequent diving actions, but the qualities of those operable solutions found in early rounds are often very poor, because no concern on the values of the nodes is accounted. Anyway, having the global upper bound frequently updated by many operable solutions, one is able to get rid of lots of low-quality nodes by cut-off-by-bound. Nevertheless, even being more "diving-prone" than best-first, without additional mechanism, depth-first alone may still be unable to provide sufficient dives. This is often due to the possibility that searching can be trapped at the middle levels of the BB tree in an oscillatory manner and there is no "exit" for the current search to be relocated to somewhere else on the tree. Since the possibility in finding operable solutions at the middle-levels is still not high, once trapped there, the efficiency of the search would be seriously undermined.

Regarding the above issues, a node selection system combining both best- and depth-first for the branch-and-price ILP solver for train unit scheduling will be proposed. The principle of the system is to try to achieve as many dives towards high quality operable solutions as possible, while avoiding being trapped at the top- and middle-levels of the tree. Its mechanism can be briefly summarised in the four points listed below.

(i) Initialise the BB tree with the best-first node selection method within the first *iniL* levels of the BB tree where *iniL* is a preset parameter controlling how many levels on the BB to be explored by best-first at the initialisation stage.

(ii) After the initialisation stage, switch the node selection method to depth-first.

(iii) During the depth-first search, if a *jump condition* is triggered, the search will jump to the active node with the best or "extended-best" value, where the definition on "better" might be extended from being a smaller objective value. The active nodes before the selected best or "extended-best" node in the active queue will be repositioned after the current tail in their original order.

(iv) After such a jump, continue to search the BB tree starting from the selected best or "extended-best" node with depth-first.

5.3.2.1 Initialisation

If a BB search starts directly at a chosen node by depth-first, then it may go deep along a single path[2] from this node only to find later that this path is an undesirable direction. To avoid such "blind" searches, it is better to have a preliminary exploration over the BB tree at a few of its top-levels. Having more information about the tree, experience may indicate among the few top-level nodes which one may yield a better chance in finding an operable solution with high quality. Then the depth-first search will start from this chosen top-level node. Moreover, this initialisation phase will be mainly responsible for preparing top-level nodes available as the targets for later "jumps" (cf. § 5.3.2.2).

The above gives the basic idea of the initialisation phase. A parameter *iniL* is set to let the tree search perform an initialisation phase on the nodes of the first *iniL* levels. One can use the enumerative breadth-first strategy to solve all nodes within the first *iniL* levels. However, noticing the fact that often optimal operable solutions will belong to tree paths with top-level nodes having small objective values, the initialisation thus can

[2]A path here refers to a sequence of tree nodes from the root to the current leaf node on the BB tree. It is unrelated to the paths $p \in P$ in the DAG network.

be partially enumerative, i.e. only focusing on the more promising nodes—the nodes with small objective values. This exploration by best-first tries to make the initialisation more effective and selective compared with the breadth-first. It also allows the initialisation to explore a few more levels compared with breadth-first, as when the potential size of a BB tree is huge, only a small number of additional initialisation levels by breadth-first may bring a greater burden to the later process.

Therefore the proposed initialisation phase uses the best-first strategy in the first *iniL* levels. In order to know whether a level l is exploited by initialisation (only after its previous level $l-1$ has been exploited), one needs to check if there is still any node in the active queue from level l and if none of such node can be found, level l would be regarded as exploited. This initialisation phase will provide many top-level active nodes with good qualities waiting to be jumped to in the later process. The larger the parameter *iniL* is set, the more comprehensive the system will know about the BB tree to decide where to start the depth-first search, and more nodes will be available later for possible jumps. But if one sets *iniL* too large, then the searching process will be more like a "best-first" and the overall efficiency will be thus compromised.

5.3.2.2 Jump condition

After the BB tree has been initialised by best-first for the *iniL* levels, the searching strategy will be switched to depth-first that will lead to frequent dives. However this depth-first alone would not always guarantee an efficient searching—the searching process solely driven by depth-first may often experience oscillations trapped at the middle levels of the BB tree without any effective dive at all, mainly due to frequent cut-off actions, which significantly reduces the chance in finding an operable solution. The above "trap" phenomenon on depth-first has been frequently observed from the instances in the TUSP. Moreover, empirically and heuristically though, when the current node yields an operable solution, the probability of finding other operable solutions in its neighbourhood on the BB tree is likely to be low. In these two cases, a jump condition should be triggered to relocate the searching process such that the depth-first search will be stopped at the current location and the next node to be solved will be an active node somewhere else in the tree, which makes a "jump". To be specific, the jump condition will be triggered at the current node n as long as the Boolean expression below is TRUE:

$$\left[(noJump(n) \geq G) \bigwedge \texttt{notInDiving}(n) \bigwedge \frac{d_{\mathrm{BB}} - d(n)}{d_{\mathrm{BB}}} > \varepsilon_{BB}\right] \bigvee \texttt{oprSol}(n). \tag{5.29}$$

As for the first term in (5.29), *noJump*(n) is a counter recording how many nodes have been searched since the last jump action until the current node n; G is a parameter as a jump gap that controls the possibly minimum distance in number of traversed nodes between two jump actions. Each time a jump action takes place, $noJump(n)$ will be reset to 0.

The second term $\texttt{notInDiving}(n)$ detects whether the current searching status is in a diving or not. It returns to TRUE if the status is not in diving and FALSE otherwise. Two parameters are set in determining the diving status, i.e. the backward check range R and the rising action limit L. To be specific, to check if a current depth-first is in diving or not, R times of comparisons will be made among the recently searched nodes starting from the latest searched node n to node $n-R$, i.e. the R-th earlier searched node before n. For $i = n \dots n-R$, each time $\texttt{notInDiving}(n)$ compares the depths between the pair of consecutively searched nodes i and $i-1$ and if node i has a depth strictly higher (closer to the root) than its former node $i-1$, i.e. $d(i) > d(i-1)$, then a rising action is identified. If within the R times of comparisons, rising actions are detected more than L times, then $\texttt{notInDiving}(n)$ will regard the current status as being not in diving and return TRUE. In practice the effective check range should be the minimum of R and G since if $R > G$ it may be invalid to check more than G times backwards.

Figure 5.2 and 5.3 give an example of how $\texttt{notInDiving}(n)$ determines whether the status with the current node $n = 11$ is in diving or not. The numbers in the nodes indicate the order how they are searched (i.e. node 1, 2, ... 12). Suppose from node 1 to node 11 the depth-first is in use. Assume that at the current node $n = 11$ and $\texttt{oprSol}(n) = FALSE$ while both the other conditions in the first disjunction term except $\texttt{notInDiving}(n)$ are satisfied as in (5.29). Also assume that the parameter setting is $R = 8, L = 2$, which means to check backwards in the previous 8 searched nodes and if 2 rising events are identified, the status will be regarded as not in diving and the depth-first has to be stopped by jumping to a “best” node. As shown in Figure 5.2, there are two rising events within the range, i.e. between node 6 and 7, as well as between node 10 and 11. They are both caused by the fact that the children of node 4 and node 8 are cut-off thus the diving cannot continue (see the corresponding Figure 5.3). Therefore, at node 11, a jump is triggered to relocate the search to node 12 which is the currently best node in the active queue.

The third term $\frac{d_{\text{BB}}-d(n)}{d_{\text{BB}}} > \varepsilon_{BB}$ ensures the jump action will not take place when the current node n is within an ε-percent distance from the BB tree's deepest places. The rationale is when the current node is at deep levels of the tree, where the possibility in finding operable solutions is generally much higher than other parts, the search would better continue without being interrupted by a jump action, even if the current status is not in diving and/or there has been more than G nodes searched since the last jump action.

The last term $\texttt{oprSol}(n)$ returns to TRUE if the current node yields an operable solution and FALSE otherwise. It ensures a jump action will immediately take place if an operable node has just been found.

A jump action will relocate the current search to an active node with the best objective

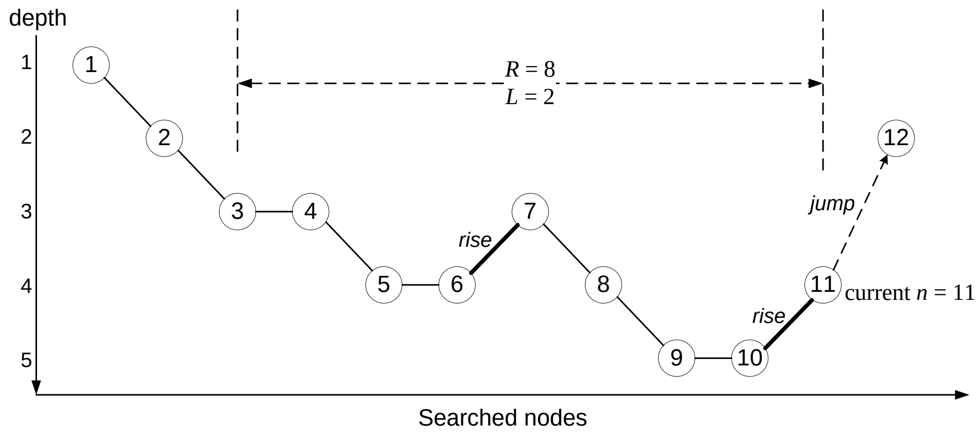


Figure 5.2: Searched nodes and their depths for the `notInDiving`(n) example

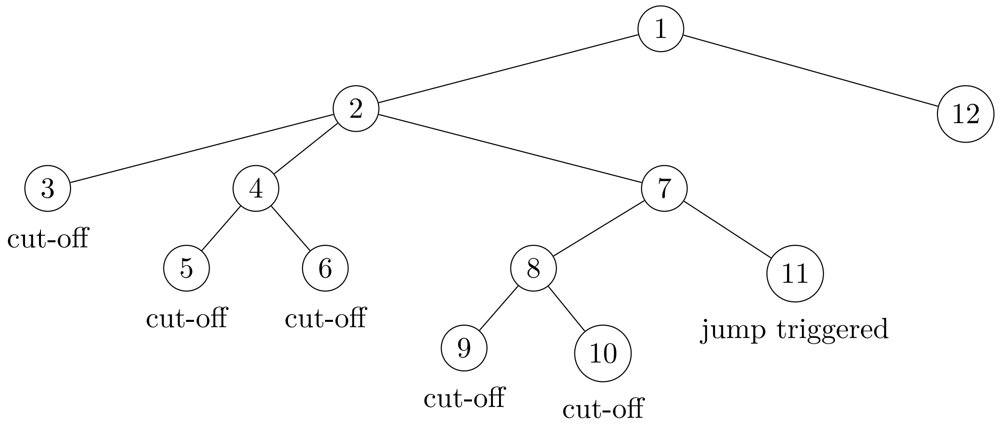


Figure 5.3: The BB tree for the `notInDiving`(n) example

value and conventionally "best" means the smallest value in a minimisation problem. In fact the definition on the "best" can be more flexible and comprehensive, as what the next part will discuss.

5.3.2.3 The "extended-best" node to jump to

When an active node is regarded as "better" than another one in selecting the active node to jump to, the implication would be the "better" active node is more promising in finding an operable solution with high quality in the subsequent few steps by, e.g., depth-first. It is impossible to design a method that can always tell which one is better than another in this context. However, experience suggests that often the optimal operable solution will originate from a BB tree path whose top-level nodes tend to have "better" qualities in terms of smaller objective values and/or higher completion ratios and/or quicker objective change rates than other nodes at the same level. Here for "best"-first on determining which active node to jump to when a jump action is triggered, the meaning of "best" will be extended by combining the following three criteria, i.e. objective value, global slope and completion ratio.

(i) *Objective value* $z^*(n)$ *of the LP relaxation at* n

In the BB tree of a minimisation ILP problem, a smaller objective value usually indicates a better quality the node may possess, which is a conventional way in evaluating an active node. As for the problem instances for train unit scheduling tested in this research, experience suggests that it is still the most reliable and robust criterion compared with the other two and it is likely to have the dominant impact among the three.

Jumping to an active node with a smaller objective value usually means the search will be relocated from a deep place to somewhere at the top or middle levels of the BB tree. This can be regarded as a fresh-start if the current search is being trapped into a dead-end. Those fresh-starts are necessary in a sophisticated searching system as they have a global view on the tree by "rescuing" the search from local stagnations.

(ii) *Completion ratio*

Completion ratio refers to the degree of completeness on targets' branching rules. For instance, how many trains are served by single-family units, how many arc variables are integral, etc. For this criterion it is straightforward that the more complete the better, as it is what the BB is heading for. Often the nodes with the highest completion ratio are located at the lower parts of the BB tree as the deeper a node is situated, the more branches are accumulated to make the completion ratio higher. However, solutions at the same level generally can have different completion ratios, sometimes being significantly different, because a considerable part of the branching targets are *automatically* accomplished by the LP relaxation rather than by branches—after all one cannot expect all variables are made integral solely as a result of branches. Thus the nodes with a high completion ratio often have a large proportion of branching targets realised automatically by LP relaxation. Experiments show that those high completion ratio nodes are generally more promising in ending up with an operable solution by subsequent branching compared with the ones with low completion ratios.

There can be different ways in defining the completion ratio, e.g. for a node where 80% of the trains are served by single families, 100% of the banned location nodes have no multiple used arcs, and 60% of the arcs are integral, an average completion ratio can be calculated as $\frac{1}{3}(80\% + 100\% + 60\%) = 80\%$ if one regards the three branching rules as equally important. Completion ratio provides a complementary way in evaluating an active node's quality. Used in conjunction with the best objective value, it can make the selection more intelligent. For example, if there are two active nodes with the same smallest objective value but different completion ratios, then the node with a higher completion ratio can be regarded as more "promising" than the other.

(iii) *Global slope*

The global slope from the root to node n is defined as

$$z'(n) = \frac{z^*(n) - z^*(0)}{l(n)},$$

where $z^*(n)$ is the objective value of n, $z^*(0)$ is the objective value of the root and $l(n)$ is the level number of n. The rationale of this criterion is that observation from experiments in train unit scheduling suggests that a node belonging to a path from the root that has significant changes in the objective values is more promising in keeping some preferred characteristics from the ascendents. On the other hand, a node originated from a path where the objective values hardly change from root to n may be less promising as some undesirable features may continue to be kept in the descendants.

Note that the above criteria can be contradictory with each other, particularly the objective value is very likely to be opposite to the completion ratio and the global slope. Often locally along the same branch, as the completion ratio gets higher the objective value would be larger. In practice, one may use only one or a combination of part or all of the above three criteria, depending on the problem's specific features. Regarding the strategy in using a combination of them, two strategies are proposed.

(i) Use a weighted linear combination of the three criteria. For each criterion, a score will be calculated and the overall score would be derived from the weighted scores of the above two or three. The weight of each criterion can be adjusted according to its significance.

(ii) Use a "max max" strategy. For instance, first select all the nodes with the best objective values within some tolerance. Among them, a second round of selection will choose the one with the highest completion ratio. This has the advantage that the computational efficiency may not be seriously compromised. For example, it is easy to collect all the active nodes with the best values, if the active queue is maintained with the nodes sorted ascendingly according to their objective values when best-first is in use. Therefore the best active nodes are simply the few nodes at the front of the queue with the same value. Then computing the completion ratio among the few front nodes in the active queue will be much less time-consuming than trying to calculate both the completion ratios and the objective values among all active nodes, get the scores of all active nodes and find out the one with the best score, as what will be the case if strategy (i) is going to be used.

5.4 Other issues regarding the branch-and-price solver

5.4.1 Initial feasible solutions at root: primal heuristics and naive columns

In order to construct an initial feasible solution to trigger the column generation at the root node of the BB tree, a primal heuristic based on a greedy algorithm is used. The main idea of this heuristic is to select the tightest connection arc (with respect to turnround time) for connecting the next train at each train node. The resulting solutions are integer feasible where passenger demands and coupling upper bounds are satisfied during the construction process while requirements like type-type compatibility and banned location restrictions are ignored as they can be realised in the branching stage. There is also an option in the primal heuristic to not require explicit restriction on the fleet size and it is usually switched on. Coupling/decoupling time allowance, however, is imposed in the primal heuristic. This greedy method has the ability of self-rescuing/reconstructing when it is trapped in a loop of repeatedly encountering infeasible solutions. For some instances, this primal heuristic can provide an initial feasible solution with an acceptable quality for subsequent stages. For example, for one instance from Southern Railway with 102 trains to be served by train unit class 171/7 and 171/8, the greedy method has produced 15 paths while the fleet size in practice given by manual process is 13 units. Note that the initial feasible solutions may not provide a global upper bound at the start point for the branch-and-price process since the requirements on type-type compatibility and banned locations for coupling/decoupling are not included, unless these constraints are not violated by the primal heuristic solution, which however is rarely seen from the experiments on the TUSP.

When the above greedy method (with or without the fleet size limit) fails to give a feasible solution, three alternatives are employed.

The first one is to discard the fleet size constraints (e.g. Constraints (4.33) as in formulation (PF_3')) in the ILP. The primal heuristic works in a way such that as long as it can produce a solution, the requirements of passenger demands, coupling upper bounds and coupling/decoupling time allowance are all satisfied without any consideration of the fleet size. Therefore, failure in giving a feasible solution is often due to a violation in the fleet size if it is required. Experiments on the current train unit scheduling show that in many cases the solver can find an optimal operable solution with a fleet size less than the maximum. Thus discarding the associated constraints often will not have any impact on the entire solution process.

If the fleet size has to be retained (e.g. there are tests on how much additional capacity can be provided keeping the current fleet size unchanged), artificial variables $\beta^k \geq 0$ associated with each k for Constraints (4.33) are added with the big-M costs in the objective. To be specific, we have the following constraints

$$\sum_{p \in P^k} x_p \leq b_0^k + \beta^k, \quad \forall k \in K \tag{5.30}$$

to replace the fleet size constraints and an item $\sum_{k\in K} M\beta^k$ to be added in the objective. A technical issue should be noticed in setting an appropriate value on M . It is sometimes observed that β^k can be positive in a root optimal LP solution with a not too large objective value and an incorrect actual fleet size. In these cases, some fine-tuning has to be done on the values of the big-Ms.

Finally when the primal heuristics cannot yield a feasible solution for the root when even if the fleet size constraints are removed, a naive construction process will be invoked. This approach simply constructs a set of $\sum_{j\in N} |K_j|$ paths each containing one and only one train $j \in N$ served by a type of unit. Without the fleet size constraints, this set of columns can always give an initial feasible solution to trigger the column generation at the root. This will bring a large number of columns into the RMP that may never be used again later. Therefore some column management strategies will be called to remove some or all of such naive columns later. Nevertheless, it is often observed that the naive construction yields no worse subsequent performance for the entire branch-and-price than the greedy construction.

5.4.2 Initial feasible solutions at leaves: column inheritance and column filter

The primal heuristics used to construct an initial feasible solution for the root is not used at leaves, mainly because this greedy method cannot consider the many branching enforcements imposed on leaf nodes. Instead, a *column inheritance* strategy is used such that each new-born leaf node inherits all or part of the columns directly from its parent as its initial columns to trigger the subsequent column generation. This strategy avoids designing complicated heuristics taking consideration of branches at leaves and also prevents from acquiring irrelevant columns from other inappropriate nodes—e.g. taking columns from its parent's siblings, where the branching setting is quite different. The most significant advantage in using the column inheritance strategy would be the subsequent column generation process often becomes very easy to solve, usually requiring only a very small number of RMP iterations, as shown by the computational experiments on train unit scheduling. Moreover, it was observed from the experiments that the troublesome "tail-off" effect was hardly found at leaf nodes while applying the column inheritance strategy.

Nevertheless, a disadvantage of the column inheritance strategy would be the number of columns at leaves in deeper places of the BB tree would be quite large as the tree grows, which may seriously slow down the branch-and-price process. In this case a *column filter* option is available such that when the number of column generation iterations at a node exceeds a preset number it_F, the filter will delete some non-basic columns whose reduced costs are larger than a preset threshold ρ_{RC} in the optimal LP relaxation solution at the parent node.

When the inherited columns from the parent fail in constructing an initial feasible solution, it does not necessarily indicate the infeasibility of the current leaf—it is possible that there exists a feasible solution but the limited existing columns simply cannot construct it. When this happens, naive columns similar as those described in § 5.4.1 will be generated to construct a feasible solution. If even the naive columns fail in constructing a feasible solution, then the infeasibility of the current leaf node will be claimed and the node will be cut-off by infeasibility.

Some more drastic strategies are possible for column management at leaves, but have to be used cautiously. One of such strategies is to only allow each leaf node to get the columns that are in the basis or having small reduced costs in its parent's optimal solution. This strategy however often leads to poor overall performance. Another one would be to construct the initial feasible solution simply from naive columns at each leaf node. There is somewhat a trade-off relation in how to construct the initial column at leaf nodes such that one needs to carefully choose between inheriting columns from the parent and fresh-starting from naive columns. The former will burden leaf nodes with many columns but has less column generation iterations; the latter may have for each leaf node less columns (if most naive columns are removed later) but requires more column generation iterations with possible tail-off effects. Achieving a trade-off point can be tricky and problem-specific. Nevertheless, as for the instances tested for the train unit scheduling problem, there is yet no evidence showing that the strategy acquiring initial columns to be all from naive columns will improve the overall performance.

5.4.3 Column generation early stop

There is an option in the branch-and-price solver to stop the column generation process early before it is fully solved to optimality, as suggested in [8]. This is especially useful when column generation at the current node is triggered by naive columns rather than inherited columns from the parent as the tail-off effect is frequently encountered when the naive columns are used. At the end of each RMP LP iteration, a pair of upper and lower bounds UB, LB on the optimal objective of the master problem can be obtained from (5.8). Therefore if this gap is smaller than a tolerance ε_{CG} before the column generation is strictly solved by zero reduced costs from all subporoblems, the LP can be regarded as almost optimal and the lower bound will become the node's LP relaxation value. Options are set such that the gap can be measured absolutely like $UB - LB \leq \varepsilon_{CG}$ or relatively like $\frac{UB-LB}{UB} \leq \varepsilon_{CG}$ or $\frac{UB-LB}{UB+1} \leq \varepsilon_{CG}$. The absolute strategy is used in the experiments reported in Chapter 6.

An alternative way to achieve such an early stop is to set a maximum number M_{CG} on column generation iterations. Setting such a parameter would be more advantageous if the column inheritance strategy were used, as often by limiting the maximum number of column generation iterations the early stop will only take place at the root node while most of the leaf nodes can still be solved to optimality by a few numbers of CG iterations.

5.4.4 Estimated global upper bounds and node reservation

In order to avoid nodes with bad qualities at the early stage in the BB search, an artificial global upper bound UB^0 is set based on experience in estimating the gap between the root's objective value and the optimal operable solution. In the train unit scheduling problem using Model (PF_3') with $V_1 = 1$ and $V_i < 1, i \neq 1$, often it is quite reliable to set $UB^0 = \lceil z^*(0) \rceil + \mathit{\Delta}^0$ with $\mathit{\Delta}^0 = 1$ where $z^*(0)$ is the root's objective value and $\mathit{\Delta}^0$ is the initial increment value. Then the BB search will be initiated with the global upper bound UB^0. Note that nodes that are "cut-off" by UB^0 will not be really abandoned but instead they are reserved in a reservation list. Letting UB be the real global upper bound given by operable solutions, as long as $UB^0 < UB$, the nodes in the reservation list will not be regarded as part of the active queue and only the nodes cut-off by UB will be truly given up.

If optimal operable solutions can be found within the active nodes derived by UB^0, then the BB search terminates and the reserved nodes will never be used. If UB^0 fails in finding optimal operable solutions, which means all active nodes other than the ones in the reservation list have been used up and the active queue is temporarily empty, the estimated upper bound will be updated by $UB^1 = UB^0 + \mathit{\Delta}^1$, where $\mathit{\Delta}^1$ is the increment value for the first update. Accordingly, the nodes in the reservation list having objective values between UB^0 and UB^1 will become the new active queue. Analogously for subsequent $k = 1, 2, \dots,$ when an estimated upper bound UB^k fails in finding optimal operable solutions, similar process will take place on the BB tree by letting $UB^{k+1} = UB^k + \mathit{\Delta}^{k+1}$ and making the nodes in the reservation list with objectives between UB^k and UB^{k+1} as the members of the new active queue. Note that each time the increment parameter $\mathit{\Delta}^{k+1}$ can be set independently according to the current status.

Chapter 6

Computational experiments

This chapter reports the computational experiments for the train unit scheduling problem. Part of the experiments are early work based on the real-world instances from Southern Railway. The main part of the results that are going to be reported is based on the real-world instances from First ScotRail, whose network has been manually divided into three main areas (GA, GB and GC) by the practitioners to ease the scheduling and planning. The GA and GB areas were tested in this research.

At the beginning a brief summary on the earlier work with the datasets from Southern Railway will be reported. Then a detailed description of the dataset and problem instances of ScotRail will be given. These are the bases of the computational experiments. Some overall results on the newest experiments on the GB and GA areas of the ScotRail 2014 May timetables with passenger demands from December 2013 will be reported next, followed by some overall results on the earlier experiments on the GB area of the ScotRail 2011 December timetables with manually diagrammed unit capacities as passenger demands. With appropriate post-processing by TRACS-RS (see § 3.4), the results are satisfied by rail practitioners from ScotRail. Generally the solutions' qualities are comparable to or better than the manual ones. Next, the focus will be switched to reporting the fine-tuning process on the branch-and-price ILP solver by different parameter settings. Then a series of experiments on estimating the "ideal" passenger demands by varying the required demands taken by the solver as inputs.

This chapter will be closed by a discussion on using TRACS-RS for post-processing. This process resolves the station-level unit blockage problems and locally realises some relaxed requirements such as coupling/decoupling time allowances and minimising the number of used block-arcs from the network-level framework. Note that the network-level framework has already achieved the crux of a practical solution. Based on the experiences from both the rail practitioners and our computational experiments, the remaining station-level adjustment work is generally easier to solve than the network-level model either manually or automatically. At the moment TRACS-RS, a reliable interactive visualisation tool with its accurate infrastructure knowledge of all the stations across the ScotRail network, is able to fill the gap between the network- and station- level frameworks. In the future, given time and developmental resources, the station-level adjustments can be automated either as an extension of our optimisation algorithms or as a utility in TRACS-RS.

Most computational experiments solved by the customised branch-and-price ILP solver have been performed on a 64 bit Xpress-MP suite (latest version 7.6) on a Dell workstation with 8G RAM and an Intel Xeon E31225 CPU. As for the realisation of the branch-and-price process, the solver has only utilised the default simplex solver of Xpress-MP to solve LP relaxations during the column generation processes without employing the default integer programming solver of Xpress-MP. The main components such as the column generation process and the branch-and-bound tree are realised by customised codes designed by the author. Most of the post-processing tasks have been done based on TRACS-RS of version 1.0.177.0.

For clarity of presentation, where many result figures are involved, the figures will be recorded in Appendix A and referred as appropriate in the main thesis.

6.1 Experiments on Southern Railway

At the earlier stage, our computational experiments on train unit scheduling were mainly based on the real-world problem instances from Southern Railway Ltd, which is a very large passenger train operator in the south of England, covering routes from London Bridge/Victoria to Banstead, Beckenham Junction, Epsom Downs, Epsom, Leatherhead, East Croydon, Sutton, East Grinstead, Uckfield, Caterham, Tattenham Corner, Redhill, Gatwick Airport, Crawley, Horsham, Littlehampton, Bognor Regis, Southampton, Portsmouth, Brighton, Eastbourne and Ore. Each year, there are two timetables published in Southern Railway. In the year of 2011, an overall Southern Railway timetable had over 2400 trains on a typical weekday, and 11 different types of train unit were in use (see Table 3.1).

Some subsets of the December 2011 timetable of Southern Railway, including their actual unit diagrams derived by manual schedulers, were used for developing our earlier models and for the relevant experiments. The subset contained 1346 passenger trains and 6 unit types all from the “TWTh” (Tuesday, Wednesday, Thursday) diagrams.

Unfortunately, no passenger demands collected from real-life surveys could be provided and we had to use the manually diagrammed unit capacities as the passenger demands

Table 6.1: Results from preliminary experiments on Southern Railway

Instance	Train#	Unit# manual	Unit# solver	Type	Time (sec)	root col. only
1	102	13	13	c171/7,c171/8	5.065	no
2	164	17	12	c442/1	15.652	yes
3	207	22	41	c456/0	94.979	yes
4	377	49	50	c377/4	> 7200	yes
5	496	42	56	c455/8	> 7200	yes

to be input into the solver. This seriously limited the solver's potential in finding better solutions than the manual diagrams. In addition, the idea of using the train convex hulls had not been developed yet at that stage. Instead, a set of natural constraints explicitly stating the various requirements as given in Chapter 4 was used for the solver. Those "knapsack constraints" can significantly weaken the LP relaxation such that the performance of the solver can be compromised. In general, the solver's ability was quite poor at the early stage, such that it was not capable of handling the full-sized real-world instances from Southern Railway.

Due to the above reasons, only five subsets extracted from the entire Southern Railway December 2011 timetable were tested and some simplifications were made. The five subsets accounted for around 67% of the entire timetable, which covered some of the busiest routes in the Southern Railway network including the entire diesel unit network. Each of them corresponded to the trains in the real diagrams served by compatible unit type(s). Sometimes when the instance was too difficult for the solver, one had to allow the branch-and-bound to be only based on the columns generated by the root, making it no longer a branch-and-price process. The ILP formulation used was the extended natural formulation (AF_1). The objective in the solver was to only minimise the fleet size and the number of used block-arcs. Since each subset consisted of only a single family of unit, only Constraints (4.1)–(4.2) for the single-family trains with respect to N' were present. There was no information about banned locations for coupling/decoupling provided by the train operator at that time; therefore, Constraints (3.8)–(3.9) were not included. As a result of the above simplification or lack of input knowledge, neither the train-family branching nor the banned location branching was needed in the branching stage. The minimum turn-round time and the single coupling/decoupling operation time was set to be 5 minutes and 2 minutes for all connection arcs respectively. Since no practical post-processing method had been developed at that stage, there was not any effective method to prevent the unit blockage at that stage.

The major results from the experiments on the five subsets based on the December 2011 Southern Railway timetable are given in Table 6.1.

No operable solutions for problem instances with more than 500 trains could be found. For problem instances within fewer than 500 trains, operable solutions with various qualities could be found, and if the number of trains was no more than 200, solutions could be found in a relatively short time. Only Instance 1 found the fully optimal operable solution by branch-and-price. Instance 2–3 found the optimal solutions based on the root columns only but these were suboptimal for the full problem. For Instance 4–5, the experiments had to be stopped after the computational time exceeded 2 hours and the solutions that were likely to be sub-optimal had been found.

The above results had been analysed by the experts from both Southern Railway and Tracsis plc. The major point from their feedbacks was the fact that frequent unit blockage problems were observed at some stations, making all the results basically invalid for practical use (cf. § 3.1.7 for the unit blockage issue). Note that this sort of blockage problem does not exist in most of the other vehicle scheduling problems such as bus scheduling. It was then realised that it might be necessary to decompose the problem into a network-level and a station-level to ensure the unit blockage issues can be resolved and therefore the post-processing phase was included in the subsequent experiments for ScotRail.

With respect to the ILP solver, it was still immature at that stage. There was no way to strengthen the poor LP relaxation as a result of the knapsack constraints. It needed the constraints corresponding to two factors (car number and unit number) to satisfy the combination-specific coupling upper bounds and there was still the possibility for this method to be invalid for more complicated scenarios. No sophisticated branching strategies such as the adaptive node selection method was developed either, making the branching stage very inefficient.

Nevertheless, the collaboration with Southern Railway has been helpful and inspiring. The importance of interactions and mutual understanding between the academic and the industry can never be underestimated. The huge difference between practical problems for daily operations and simplified models for academic researches was realised and analysed. Several meetings were made between the research group and the experts from Southern Railway to better fulfil the academic-industry gap. One of the most impressive experiences gained by the author was how to access and understand the expert knowledge on the train unit scheduling problem. There was no well-documented, systematic description, no technical book or reference available, and the information found from other relevant academic literature was extremely limited and unsuitable for the UK railway system. To acquire the very knowledge that is needed to produce workable results in practice, the only approach would be the frequent interactions via meetings and email correspondences with the rail experts. This has drawn a significant distinction between this research and many other studies that are more theoretical-flavoured.

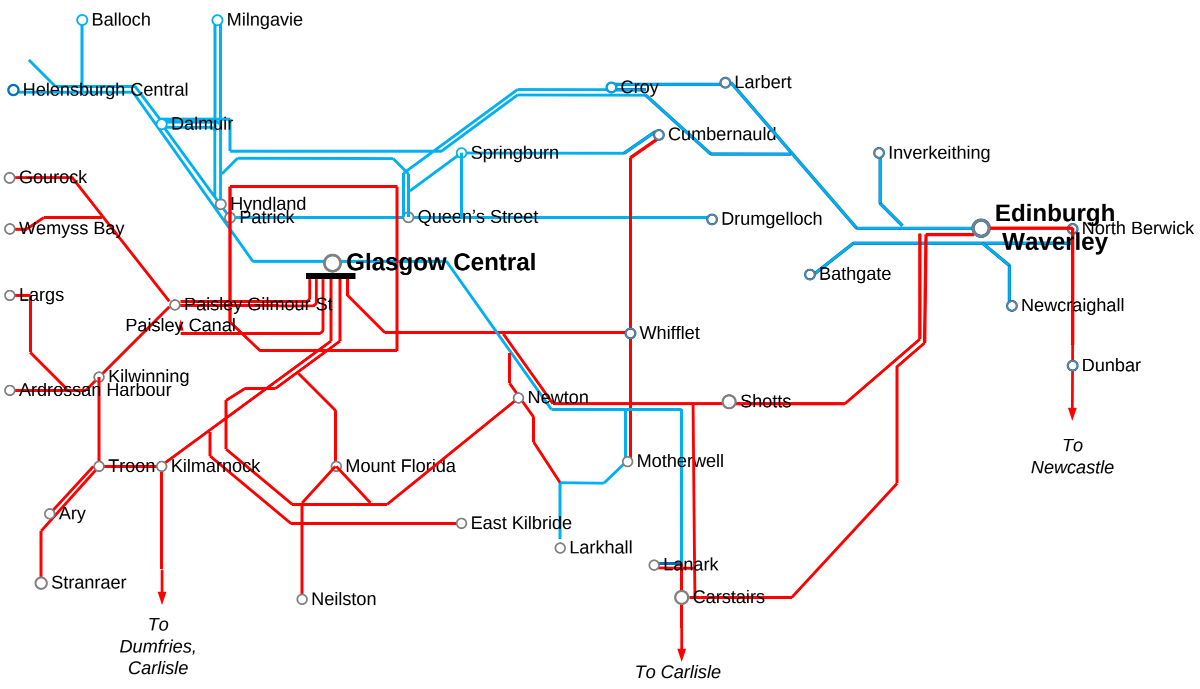


Figure 6.1: The GA (red) and GB (blue) areas of the ScotRail network

6.2 ScotRail dataset

6.2.1 Routes and services of ScotRail

The major part of the research benefits from the collaboration with First ScotRail, the main passenger train operator company in Scotland. The fleet information of ScotRail regarding train unit types and their compatibility relations and default coupling upper bounds, etc. has already been given in Table 3.1. This largest train operator in Scotland has a daily timetable of around 2250 trains. The more than two thousand services has been traditionally divided into three relatively independent areas by the manual schedulers, namely the GA, GB and GC areas. The three areas are not entirely divided geographically, but as an artificial subdivision of routes/services by ScotRail for scheduling/planning purposes. GA/GB are associated with those frequent commuter trains mainly in the busiest central belt part of Scotland connecting Edinburgh and Glasgow, while GC includes all other trains in Scotland not in GA/GB.

Figure 6.1 gives an illustrative network map of the central belt where the lines in red are from GA and the lines in blue are from GB. Some routes of GA can be extended beyond the central belt to some southern places like Stranraer, Carlisle and Newcastle. The computational experiments to be reported will focus on the GA and GB areas.

The following sections will give detailed introductions on the GA and GB areas.

6.2.1.1 The GA area

Table 6.2 gives the route information of the GA area. Being centred at Glasgow Central Station, the routes from GA area mainly connect services from Glasgow to places at the western/southern directions roughly separated by the River Clyde, such as Paisley, Gourock, Wemyss Bay, Largs, Ardrossan, Neilston, Newton, Ayr, Stranraer, Barrhead, Kilmarnock, as far as to locations as Dumfries, Carlisle and Newcastle. It also connects services from Glasgow Central to Edinburgh via Shotts and Castairs. Cumbernauld at the north-eastern direction is also connected to Glasgow Central via Whifflet as part of the GA services. Moreover, the Cathcart Circle Line around Glasgow City is also a part of GA.

Only the subsets served by electric train units of GA (known as GA Electrics) have been involved in the numerical experiments as suggested by the practitioners from ScotRail. This excludes the trains served by the diesel units of c156 and c158 which are relatively independent. In the May 2014 timetable, there are 703 timetabled trains in GA Electrics. The manual solution uses 14 units of c314, 20 units of c380/0 and 15 units of c380/1 to cover the 703 trains in their proposed diagrams. c380/0 and c380/1 are coupling-compatible while c314 can only be coupled with its own types. Note that the services in GA Electrics generally have very dense patterns since on average each unit covers about 14.3 trains per day according to the manual diagrams.

6.2.1.2 The GB area

Table 6.3 gives the route information of the GB area, which is known as Northern Electrics at ScotRail, as the routes are mainly related to the locations north to the River Clyde. Three subsets can be further identified in GB as shown in Table 6.3: (i) From Lanark in the south-east direction to Balloch in the north-west direction; (ii) From Edinburgh in the east direction to Helensburgh in the north-west direction; (iii) From Springburn in the north direction to Helensburgh. Moreover, in the new May 2014 timetable, a new route has been added to the GB area, i.e. some services originally terminated at Springburn have been extended to Cumbernauld. From the two route tables 6.2 and 6.3, it can be observed that the GA services have more complex coupling upper bound restrictions, while for GB the upper bound relation is much simpler such that it can be shortly summarised as for whatever the case is, at most 2 units can be coupled if the permitted 3-car types c318, c320 and c334 are used.

The GB area is the part mainly focused in the computational experiments to be reported. In the May 2014 timetable, there are 510 timetabled services in the GB area, to be served by 19 units of c318, 20 units of c320 and 36 units of c334 in the proposed manual diagrams by ScotRail. c318 and c320 can be coupled with each other while c334 defines a unit family on its own. The services in GB have much less dense patterns compared with GA, since on average each unit only serves 6.8 trains according to the manual diagrams.

Table 6.2: Routes of the GA area of ScotRail

Route	Permitted types	Coupling upper bound (car)
Glasgow Central to Gourock and Wemyss Bay	c314, c380/0, c380/1	6
Glasgow Central to Ardrossan/Largs/Kilwinning	c380/0, c380/1	7
Glasgow Central to Irvine and Ayr	c380/0, c380/1	7
Glasgow Central to Paisley Canal	c314, c380/0, c380/1	4, c380/x: 3 on consecutive return trips
Glasgow Central to Glasgow Central (Cathcart Circle))	c314, c380/0, c380/1	c380/x: 6, c314: 4, c380/x: 3 on consecutive return trips
Glasgow Central to Newton	c314, c380/0, c380/1	c380/x: 6, c314: 4, c380/x: 3 on consecutive return trips
Glasgow Central to Neilston	c314	6
Glasgow Central to Stranraer	c156	4
Kilmarnock/Ayr to Girvan/Stranraer	c156	4
Glasgow Central to Barrhead	c156	4
Glasgow Central to East Kilbride	c156	6
Glasgow Central to Kilmarnock	c156	6, 4 if services call at Priesthill and Darnley, Kennishead and / or Nitshill
Glasgow Central / Dumfries to Carlisle / Newcastle	c156	4
Glasgow Central / Motherwell to Edinburgh via Shotts	c156 (preferred), c158	4
Glasgow Central / Motherwell to Edinburgh via Carstairs	c380/1	Maximum 4 vehicles and must be fitted for Conductor operation (only 8 units fitted). Last Service ex Edinburgh, also some SO services worked by Class 156 and / or Class 158 units
Glasgow Central to Whifflet	c380/0, c380/1	4
Motherwell to Cumbernauld	c156 (preferred), c158	4
Edinburgh to Dunbar	c380/0, c380/1	8
Edinburgh to Ayr	c380/0, c380/1	7
Glasgow Central/Edinburgh/Ayr to North Berwick	c380/0, c380/1	4

Table 6.3: Routes of the GB area of ScotRail

Route	Permitted types	Coupling upper bound (car)
Lanark / Motherwell / Coatbridge Central / Larkhall to Partick / Hyndland / Milngavie / Garscadden / Dalmuir / Balloch	c318, c320, c334	6
Edinburgh / Bathgate to Airdrie / Partick / Milngavie / Garscadden / Dalmuir / Balloch / Helensburgh	c334	6
Airdrie / Springburn / Bellgrove / High Street / to Partick / Milngavie / Garscadden / Dalmuir / Balloch / Helensburgh	c318, c320 ,c334	6

Table 6.4: Instances from ScotRail tested in the numerical experiments

Instance	GB-entire	GB-334R	GB-318(320)R	GA-314	GA-380
Timetable	Dec 2011	May 2014	May 2014	May 2014	May 2014
Unit types	c318, c320, c334	c334	c318 (c320)	c314	c380/0, c380/1
Train#	483	184	326	278	427
Multi-family train#	352	–	–	–	–
$N_B^{\pm}$#	362	156	275	218	137
Block-arc#	10641	3252	5986	3483	5242
Type-arc#	17779	3253	5986	3483	10457
Demand input	diag	PAX	PAX	PAX	PAX

6.2.1.3 Tested instances

There were several different problem instances being tested in the computational experiments, each with different properties and purposes. Table 6.4 gives the details of those instances.

In Table 6.4, the names of the instances are listed in the first row. The first column gives the properties to be shown for the instances. Train# gives the number of trains in the instance, which equals to the number of train nodes in the corresponding DAG. Multi-family train# gives the number of trains that have more than one family of unit permitted to serve (thus have to be eliminated by train-family branching if multiple families are identified there in relevant LP relaxation). $N_B^{\pm}$# gives the number of banned location nodes, i.e. $|N_B^-| + |N_B^+|$ in the corresponding DAG. Block-arc# and Type-arc# are the number of block-arcs and type-arcs in the DAG. For the Demand input, "diag" means the demands were set as the manually diagrammed, and "PAX" means they were set as the surveyed actual passenger number counts. For the May 2014 timetable, the PAX# data were from the December 2013 survey.

There were 5 instances mainly tested in the computational experiments. GB-entire includes the entire services from the GB part of the December 2011 timetable with all the three permitted types c318, c320 and c334. GB-334R is an instance of rerunning the

GB services based on the May 2014 new timetable; it contains the 184 trains that were supposed to be served by c334 in the May 2014 manual diagrams and in the experiments only c334 was present. On the other hand, GB-318(320)R consists of the remaining trains in GB from the May 2014 timetable and manual diagrams. This decomposition on GB according to unit families was justified by practitioners from ScotRail such that although the two families of c334 and c318/c320 share a large proportion of permitted routes, it is in fact more preferable and more sound to keep the original route assignment plan. The practitioners also pointed out that c318 and c320 can almost be regarded as the same type of unit after some installation works on additional toilets etc. to be taken place on all c318 units. Therefore, in GB-318(320)R, the two types have been regarded as one type. Finally, GA-314 and GA-380 are also decomposed instances for the GA area based on families, although for the GA-380 case, c380/0 and c380/1 cannot be merged into one single type.

Admittedly, for instances like GB-334R, GB-318(320)R and GA-314, if one solves them by the reduced formulations (AF_3) or (PF_3), their LP relaxation would actually become single-commodity flow problems that can be solved efficiently as minimum-cost flow problems after some parametric conversions, although a branch-and-bound tree would still be needed for banned location branching. Nevertheless, the ultimate target of the research is to develop a solver that will solve instances with multi-types and multi-families. This makes the investigation on minimum-cost flow methods digressive. Moreover, the behaviours observed from solving such single-type instances by conventional multicommodity flow ILP methods would be very helpful and inspiring for multi-type cases. Pragmatically, it would be a good idea to insert a minimum-cost flow based routine as a complementary component for single-type instances that are detected in a commercial product in the future, which is however out of the scope of this thesis.

6.3 Experiments in the branch-and-price solver on the ScotRail datasets

In this section we will report the experiments on the ScotRail datasets using the customised branch-and-price solver introduced in Chapter 5. The main aims of the experiments are to test the feasibility of the two-level framework (i.e. the network-level and the station-level) and the operability of the branch-and-price solver, both being subject to real-world instances. At the same time, it was expected to get a deeper insight and better understanding on the branch-and-price solver by a series of fine-tuning experiments with different parameter settings. Moreover, the possibility for the solver to contribute to "ideal" passenger demand estimation will be discussed and the relevant experiments will be reported. Finally feedback and suggestions from the experts at ScotRail were analysed and taken to make possible improvements regarding the issues like solution qualities etc. Some experiments regarding the feedback on solution qualities will be given.

Four kinds of experiments were performed. The first kind was on two groups of standard schedules on the GB and GA areas. The first group (GB-entire-1a) was based on the entire December 2011 GB timetable with the passenger demands set as the manually scheduled unit capacities and the second group was formed by four experiments (GB-334R-1a, GB-334(320)R-1a, GA-314-1a and GA-380-1a) based on the decomposed May 2014 timetables on GA and GB with the passenger demands extracted from actual passenger counts. The focus of the above standard experiments was on the comparison between the manual and the solver's results and the operability of the solver's solutions in handling real-world instances.

The second kind was a series of experiments in fine-tuning the branch-and-price solver with various parameter settings, including eight groups as: (i) the maximum column generation iteration number M_{CG}, (ii) the column generation gap tolerance ε_{CG}, (iii) the initialisation level $iniL$, (iv) the column filter parameters it_F and ρ_{RC}, (v) the branching rule orders and strategies, (vi) the banned location branching parameter ρ_B, (vii) the jump gap G, (viii) the initial estimated upper bound increment $\varDelta^0$. The focus was on the theoretical aspect in better understanding the branch-and-price solver.

The third kind was a series of tests on "ideal" passenger demand estimation. The last kind was the solution quality analysis based on the feedbacks from experts at ScotRail. Issues on preferences on connection time gaps, short diagrams will be discussed and relevant experiments will be presented. Finally we will report the experiments on the post-processing in resolving the remaining problems in the network-level solutions using TRACS-RS on a station-by-station basis.

6.3.1 General parameter settings

The parameter settings that generally remain unchanged for all the experiments reported in § 6.3 are described here. The minimum and maximum connection time durations for two consecutive trains were set to be 5 min and 720 min respectively. The objective weights on fleet size, arc cost and empty-running cost were set to be 1, 0.001 and 0.1. Only the existing empty-running movements found in the manual schedulers' diagrams were allowed to be used in the solver. This is because creating a new empty-running requires lots of activities and efforts far beyond the simple tasks of satisfying temporal and spatial allowances, e.g. one should check the track occupation with Network Rail and other train companies. Except for the cases otherwise stated, the jump criterion was always set as regarding the active node with the lowest objective value as the "best" one. The backward check range in detecting dives was set to be $R = 10$, and the rising action limit was set to be $L = 1$, i.e. once a rising action is found, the process will be deemed non-diving promptly (see § 5.3.2.2). The parameter controlling the distance from the bottom of the BB tree for deciding jumping actions was set to be $\varepsilon_{\text{BB}} = 0.02$. The primal heuristic was generally disabled except when otherwise stated as for most of the instances tested in our experiments it did not outperform much other than generating the naive columns. Each leaf node was set to inherit its parent's columns with or without enabling the filter option. If the inherited columns could not yield an initial feasible solution, naive columns as described in § 5.4.1 would be used. As for the estimated upper bound, a strategy in defining the next bound by a constant increment amount $\varDelta' = 0.05$ was used such that $\varDelta^{k+1} = \varDelta^k + \varDelta', \forall k$.

As for the convergence on the solver's termination criterion, for all the experiments reported, the branch-and-price processes were set to terminate as soon as the gap between the BB tree's global lower bound (i.e. the smallest node value among the active queue) and global upper bound (i.e. the smallest value given by the operable solutions) is less than 0.5. This ensures no operable solution with a smaller fleet size exists and the objective value cannot be improved by a decrease of 0.5. This tolerance is proved to be sufficient and appropriate for the current instances from ScotRail. Terminating at an operable solution with a very strict tolerance might take too long a time for large-scale cases like GB-entire. In the future, when the ILP solver's ability is further enhanced, it may be possible to apply smaller tolerances than 0.5.

Other non-general parameter settings that are going to vary in the series of experiments will be specified on individual bases.

6.3.2 Overall scheduling on the GB area

This experiment (GB-entire-1a) was based on the entire timetabled trains and fleet of the GB area of the December 2011 timetable with passenger demands set to be the seat capacities of the train units in their existing diagrams given in the December 2011 manual diagrams. Using the diagrammed capacities as the demands may not give solutions truly reflecting the actual passenger flows as surveyed by PAX. However, more straightforward comparisons between the manual and automatic solutions could be made. Details of the input and non-general parameter settings of this experiment is given in Table 6.5. This set of parameters was not fine-tuned, although its performance in the experiment was satisfactory. ε_{CG} is the stopping tolerance for the gap between the upper bound from the RMP and the lower bound given by (5.8). M_{CG} is the maximum iteration number threshold for all column generation processes. "Filter" indicates whether the filter option (cf. § 5.4.2) is enabled at each tree node with the threshold of column number it_F and reduced cost value ρ_{RC}. ρ_B and ρ_T are the parameters for banned location and train-family branching. In "rule order", T, B and A stand for "train-family", "banned location" and "arc variable" respectively, followed by the rule switch strategy. $\varDelta^0$ and $\varDelta'$ give the initial and updated increment values for estimated upper bounds. "Jump" indicates whether jump actions are used followed by the jump gap G. Finally, $iniL$ gives the initialisation level. Similar tables as Table 6.5 will be given for other experiments. The maximum total number of BB tree nodes was set to 5000 for this experiment.

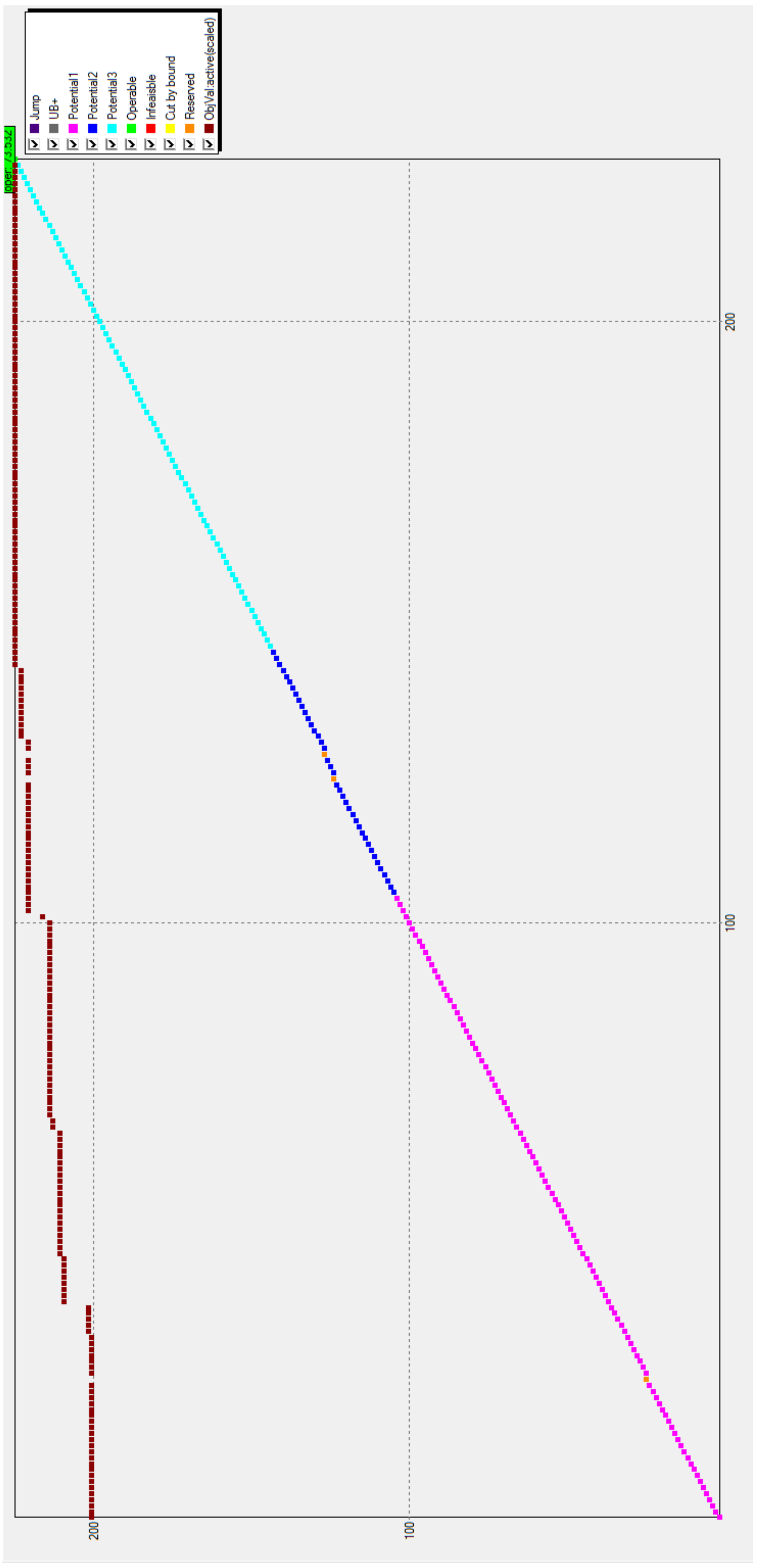


Figure 6.2: Node-depth graph for experiment GB-entire-1a

Table 6.5: Input and non-general parameter setting of experiment GB-entire-1a

instance	ε_{CG}	M_{CG}	filter	ρ_B	ρ_T	order	strategy	Δ^0	jump/G	$iniL$
GB-entire	0	890	no	0.01	0.01	T$\succ$B$\succ$A	switch@new-rule	0.1	yes / 3	0

Table 6.6: Result on experiment GB-entire-1a

	unit#	oper-obj	root-obj	root-LB	ECS#	rule% (T/B/A)	BB#	time
manual	72	73.921[a]	–	–	12	–	–	–
1a	72	73.532	73.5263	73.4312	8	46/18/36	227	3802

[a] Equivalent value converted from the manual diagrams.

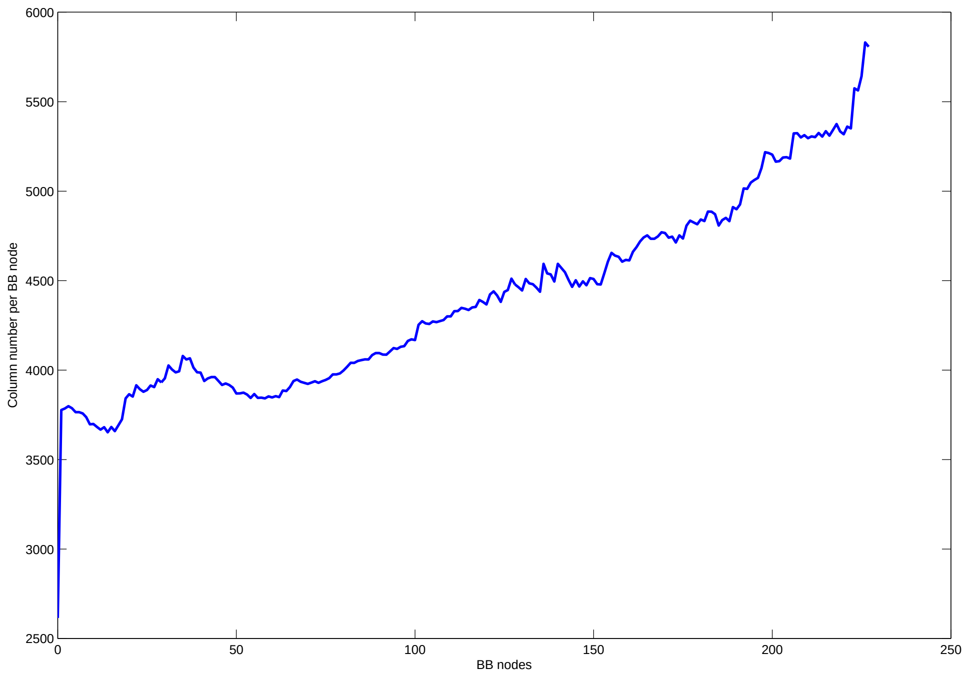


Figure 6.3: Column numbers per tree node for experiment GB-entire-1a

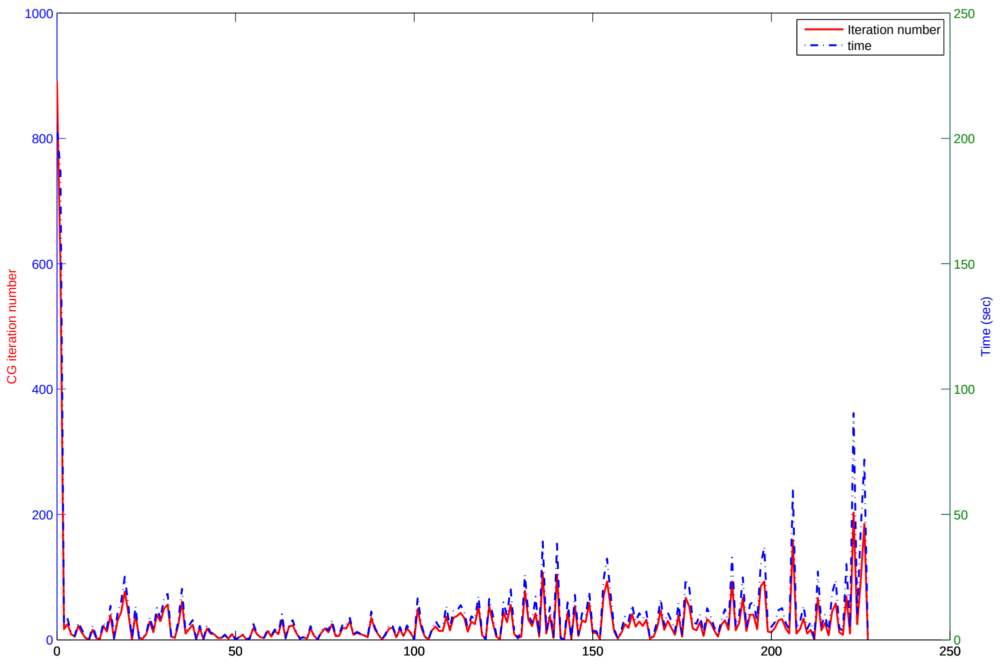


Figure 6.4: CG iteration numbers against CG time per tree node for experiment GB-entire-1a

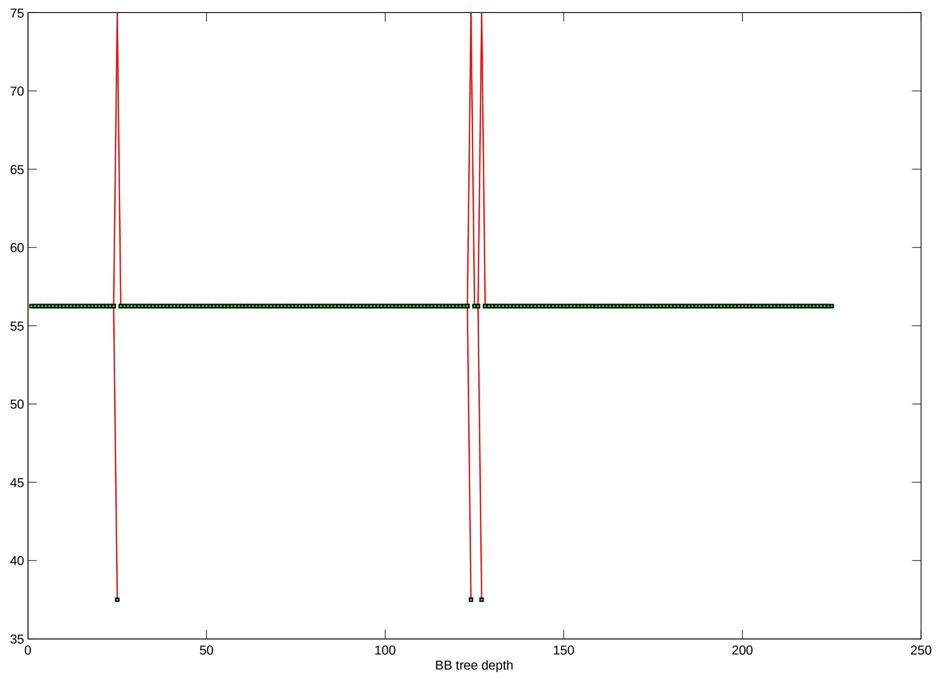


Figure 6.5: BB tree for experiment GB-entire-1a

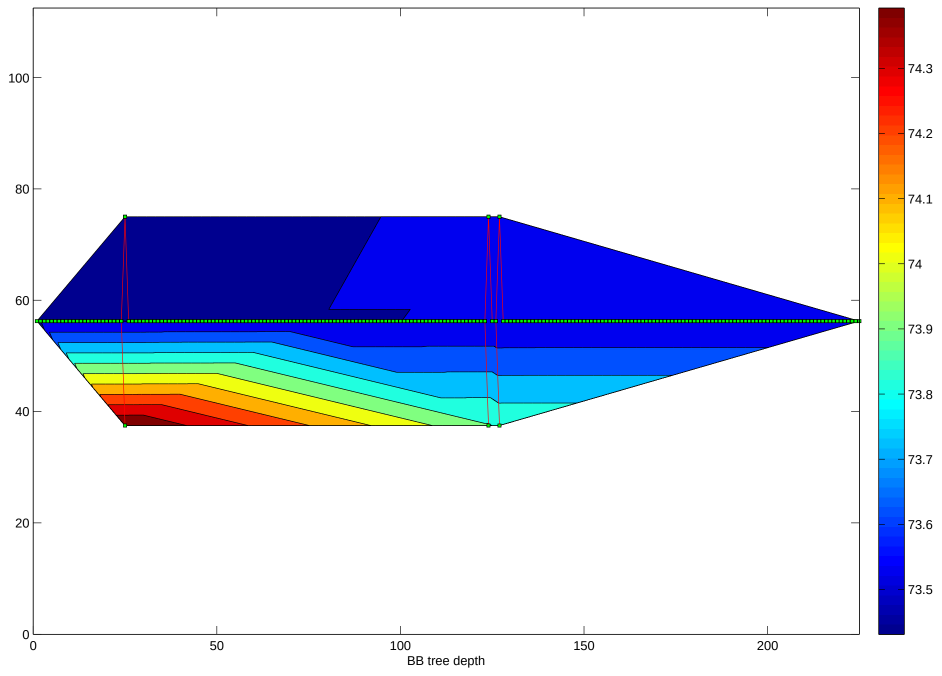


Figure 6.6: BB tree with node values for experiment GB-entire-1a

Table 6.6 gives the result on experiment GB-entire-1a against the manual solution. The number of units used (unit#), operable objective value (oper-obj), LP relaxation objective value at the root (root-obj), the lower bound value at the root (root-LB), number of ECS movements (ECS#), percentage of the three branching rules (rule% (T/B/A)), number of nodes in the BB tree (BB #) and computational time in seconds (time) are listed. The solver gave a solution with the same fleet size as the manual one of 72 units, possibly due to the fact that the input demand was set as the diagrammed unit capacities. Although using the same number of units, the solver's solution outperformed the manual diagrams in quality by having a smaller operable objective value: solver's 73.532 against 73.921, which is the equivalent objective converted from the manual solution. Note that the root was not solved to optimality as there is still a gap between the root objective (73.5263) and the root lower bound (73.4312), due to the fact that the column generation iteration number at the root had reached the maximum allowed value $M_{CG} = 890$. The solver used only 8 of the empty-running movements from the existing 12 ones in the manual diagrams, thanks to the minimisation on ECS numbers. The mostly frequently used branching rule was the train-family branching (46%) compared with banned location (18%) and arc variable (36%). For a network with 485 nodes and 17779 arcs with many branching targets, only 227 BB tree nodes were used to find the optimal operable solution. The computational time was 3802 seconds.

After the post-processing phase that eliminated all unit blockage issues without changing the objective value of the solution, the practitioners from ScotRail were satisfied with the above results given by the solver, which had a better quality than the manual diagrams and was basically operable especially in having no unit blockage at any stations. In addition, the solver also detected a case with 8 min turnround time at Edinburgh Waverley from the manual diagrams which had violated the 10 min minimum turnround time restrictions there. The solver also gave a solution with more strengthened capacity provision compared with the manual diagram such that 12 trains were served by 2 units while they had been served by 1 unit in the manual results. Note that those strengthened trains did not result in the solver having a higher objective than the manual one, because from Table 6.6, one can see the fact is the opposite. The main reason of it would be in the objective the empty-running movements were given a weight 100 times larger than the weight on "carriage-kilometre". Therefore the solver would use more strengthened services to realise the same effect that were once achieved by ECS in the manual diagrams. This is also a good example to show how coupling/decoupling can be used to redistribute unit resources as a better alternative to empty-running.

There were several kinds of graphs used in the computational experiments to reflect and help in better understanding the solution results. We will use the GB-entire-1a experiment as an example to introduce those graphs that will also be used to explain and analyse the subsequent experiment in this chapter.

6.3.2.1 Node-depth graph

Figure 6.2 gives a *node-depth graph* for GB-entire-1a. Such a graph is used as a way to analyse the behaviour of the branch-and-price processes in this research. Typically in a node-depth graph, the horizontal axis gives the traversed BB nodes during the branch-and-price and the vertical axis marks the depth of each node. Note that contrary to a conventional BB tree which is rooted at the top and grows downwardly, in a node-depth graph a higher value on the vertical axis means a deeper depth. For a node that is put into the active queue (potential), three different colours, magenta, blue and cyan, are used to indicate the determined branching rules at the potential nodes—train-family ("potential1" with magenta), banned location ("potential2" with blue) and arc variable ("potential3" with cyan). For a node that is not put into the active queue, an orange/yellow/red colour means it is reserved/cut-off-by-bound/infeasible, and a green square shows a node that is operable labelled with its objective value. For other events, a purple square shows a jump at the current node, and a grey square shows an event of an increment in the estimated upper bound. Sometimes, the scaled objective value of each potential node is also displayed alongside the node depth information.

The node-depth graph is very useful for observing the branch-and-price solver's behaviours. In particular, dives and jumps can be clearly identified and analysed from such a graph. Other characteristics such as the distribution of different branching rules, the development of operable/infeasible solutions can also be interpreted from a node-depth graph. Moreover, it is a convenient and powerful tool for checking and debugging when one designs customised sophisticated branching strategies. In Figure 6.2, one can see the entire branch-and-price process for GB-entire-1a was almost one single dive except the interruption from three reserved nodes (node 24, 124 and 128) (cf. § 5.4.4). No jump was ever triggered, although this option was enabled. Note that there was not a single recurrence for any of the three used branching rules. For example, after the train-family branching was exploited at node 105, it was never used again in the subsequent process, although theoretically there was a possibility in reusing it. Such behaviour would be partly as a result of the appropriately used dynamic strategy “switch@new-rule” (cf. § 5.3.1.4).

6.3.2.2 Column number graph

Figure 6.3 gives the column numbers of the traversed nodes in the BB tree. This kind of graph is mainly used to verify the effect of the column filter described in § 5.4.2, as well as to get an overall understanding of the BB tree. As the filter was disabled in experiment GB-entire-1a, the numbers of columns were basically increasing. The slight oscillation in the curve is mainly due to the jumps to different places of the tree and some columns were deleted as a result of the train-family and banned location branching.

6.3.2.3 CG iterations numbers and computational time graph

Figure 6.4 shows the relation of column generation iteration numbers and the corresponding total time consumption of column generation iterations at each BB tree node. In most occasions, due to the use of the column inheritance strategy, both the iteration number and the time consumption at the leaf nodes will be much smaller compared with the root, avoiding the disturbing tail-off effect. This is clearly shown in Figure 6.4. It took a large number of iterations with a long time at the root—in this case the column generation was even stopped due to the maximum iteration threshold $M_{CG} = 890$ rather than exact optimality. On the other hand, since each leaf node will inherit all of its parent’s columns, the subsequent column generation processes at leaves were much easier to solve than at the root, with far less number of iterations and much less time, and they were all solved to optimality since none of them had reached the limit of 890 iterations. As for the two scales of iteration number and time, there basically exists a positive correlation between them and when such a consistency is broken, it will indicate something worth noticing in the BB tree. In Figure 6.4, the consistency changed in several occasions in the later nodes, mainly after the branching rule had switched to the arc variable branching at node 146. Indeed, the first two rules would only remove columns while the third rule would add constraints to the ILP, which might have brought a considerable burden to it—e.g. nearly 100 seconds were needed to reach the LP optimality at a few later nodes.

6.3.2.4 BB tree graphs

Figure 6.5 gives the BB tree of the GB-entire-1a experiment and Figure 6.6 gives the same BB tree with the objective value of each tree node illustrated in contours. Note that in both of them, compared with a conventional way in visualising a branch-and-bound tree where the root is at the top and the tree grows downwardly, the trees shown in Figure 6.5, Figure 6.6 and similar kinds of figures later are rotated 90° anticlockwise, where the root is on the middle-left and the tree grows towards the right direction. The horizontal axis gives the depths of the tree nodes; the vertical axis merely shows the distances of the nodes at the same level, and is thus only meaningful for visualisation purpose and is actually ignorable for interpreting the BB tree. In Figure 6.6, the coloured contours are derived according to the node values of the tree nodes as shown by the legend on the right of the graph. They will give an overall idea on the values of the nodes in different parts of the BB tree, especially under different parameter settings and branching strategies. Note that the contours from between the nodes do not show values of any specific entities.

Since the branch-and-price process from GB-entire-1a was almost a simple dive, the structure of the BB tree was accordingly simple. The three reserved nodes 24, 124 and 128 (orange coloured in Figure 6.2) are clearly shown in Figure 6.6 with their high values. Such graphs on the BB tree, which can be used in conjunction with the other aforementioned graphs, are found to be helpful in better understanding the branch-and-price process, especially in analysing different parameter settings in the fine-tuning experiments to be reported in § 6.3.4.

6.3.2.5 Conclusions on experiment GB-entire-1a

In Experiment GB-entire-1a the customised branch-and-price successfully solved the real-world problem instance of the entire GB area with nearly 500 trains, 3 unit types (2 unit families) and a large number of banned locations for coupling/decoupling. Although the input passenger demands were converted from the unit capacities from manual diagrams which limited the solver's potential in finding much better solutions than the manual diagrams, with respect to more accurately reflecting the actual passenger flows, the results given by the optimisation process were slightly better than the existing diagrams. The practitioners from ScotRail were satisfied with the results. The feasibility of the two-level framework and the operability of the solver were verified. There might be still some potential in improving the solver's efficiency. Also, some issues regarding the solution qualities were identified by the practitioners and will be discussed in detail in § 6.5.

6.3.3 May 2014 timetables on GA and GB

This series of four experiments (GB-334R-1a, GB-318(320)R-1a, GA-314-1a, GA-380-1a) was based on the timetabled trains and fleet of the GA and GB areas of the May 2014

Table 6.7: Inputs and non-general parameter settings of the May 2014 experiments

instance	ε_{CG}	M_{CG}	filter	ρ_B	order	strategy	Δ^0	jump/G	*iniL*
GB-334R	0	890	no	0.01	B≻A	switch@new-rule	0.5	yes/3	0
GB-318(320)R	0	890	no	0.01	B≻A	switch@new-rule	0.1	yes/3	0
GA-314	0	500	no	0.01	B≻A	switch@new-rule	0.5	yes/25	0
GA-380	0	800	no	0.0001	B≻A	switch@new-rule	1	yes/2	0

Table 6.8: Result on experiment GB-334R-1a

	unit#	oper-obj	root-obj	root-LB	fit[a]	OP[a]	UP[a]	ECS#	rule%(B/A)	BB#	time
manual	36	–	–	–	111	68	4	3	–	–	–
solver	33	33.2435	33.232	33.232	168	15	1[b]	0	96/4	707	584

[a] There was one train whose December 2013 passenger count survey data was unknown.
[b] There was one train whose PAX#> $2 \times q_{c334}$ (unit capacity of c334) thus can never be satisfied.

timetable with demands extracted from actual passenger counts surveyed in December 2013. It gave the solver an opportunity to provide better solutions than the manual diagrams. As aforementioned, the timetabled trains had been decomposed with respect to the type-compatibility relations according to the manual schedulers' wish such that no train-family branching was needed. Details of the input and non-general parameter settings of this series of experiments are given in Table 6.7. There was no guarantee that this set of settings would be the best one. In GB-334R-1a and GB-318(320)R-1a, the minimum and maximum connection time durations were set to be 5 min and 720 min for most locations, while in GA-314 and GA-380 the maximum connection time was changed to be 120 min due to the routes' denser patterns. The primal heuristic was enabled in the two experiments on GA. The maximum total number of BB tree nodes was set to be 5000 for all the four experiments.

6.3.3.1 GB-334R-1a

Table 6.8 gives the results on experiment GB-334R-1a against the manual solution. The solver saves 3 units compared with the manual diagrams and after post-processing the solver's fleet number is still kept to be 33. Since the actual passenger numbers PAX# were available for this new round of experiments, three more criteria comparing the unit capacity provided by the manual diagrams and the solver with the actual passenger numbers have been added to the table as "fit", "OP" and "UP". "fit" gives the number of trains whose unit capacity provided can satisfy the passenger demand and for each train if one decreases the unit number by 1, the new provision will fail to satisfy the demand. "OP" gives the number of trains whose unit capacity are over-provided such that for each train if one appropriately decreases the unit number by at least one unit, the passenger demand can still be satisfied. "UP" gives the number of under-provided trains such that for each train the unit capacity provision cannot satisfy the passenger demand. From Table 6.8, it can be seen that although three units were saved by the solver, it still gave better capacity

Table 6.9: Result on experiment GB-318(320)R-1a

	unit#	oper-obj	root-obj	root-LB	fit[a]	OP[a]	UP[a]	ECS#	rule%(B/A)	BB#	time
manual	39	–	–	–	230	63	9	14	–	–	–
solver	$40 \to 39$[b]	40.897	40.8818	40.8643	284	17	1[c]	$5 \to 9$[b]	88/12	713	5034

[a] There were 24 trains whose December 2013 passenger count survey data was unknown. They were mainly the newly added trains in the May 2014 timetable.

[b] Post-processing has reduced the fleet size from 40 to 39 by adding 4 newly created ECS movements that were not in the manual existing diagrams.

[c] There was one train whose PAX#$> 2 \times q_{c318}$ (unit capacity of c318) thus can never be satisfied.

provision than the manual diagrams with more fitted trains (111:168), less over-provided trains (68:15) and less under-provided trains (4:1). Note that the solver will always satisfy the input demand. The presence of a single UP train in the solver's result is because this train's PAX# is higher than the capacity of 2 units of c334 but 2 units is the maximum coupling number for this type. Therefore, we had to modify the demand for that train from its actual PAX# to $2 \times q_{c334}$. Moreover, although the manual diagrams uses 3 empty-running movements, the solver uses none of them. The better capacity provision and the fact that no ECS movement was used from the network-level ILP solver have been retained after post-processing. As for the distribution of the branching rules used, one can see that the banned location branching, which is more preferable than arc variable branching, was used for most of the time, mainly because the rule order was set to be B$\succ$A. The practitioners from ScotRail was particularly satisfied with the result given by experiment GB-334R-1a due to its higher qualities in many aspects compared with the manual diagrams.

Figure 6.7 – 6.11 give the relevant figures with respect to experiment GB-334R-1a. From the node-depth graph many dives and jumps can be observed, as well as frequent node reservation events that had terminated most of the dives. As most of the tree nodes were at the top levels of the tree with frequent relocation by jumps, the curve reflecting the number of columns behaves in a strong oscillation pattern. The CG iteration numbers and CG times are basically consistent with each other, since most tree nodes were branched by banned location branching. Again it took many CG iterations at the root but much less at all the leaves. From the two graphs of BB tree, one can see how most nodes with undesirable values had been reserved by the estimated upper bound at the top levels of the BB tree, followed by a long dive with nodes of good objective values that made the process finally reach an optimal operable solution.

6.3.3.2 GB-318(320)R-1a

Table 6.9 gives the result on experiment GB-318(20)R-1a against the manual solution. There was again one train whose surveyed actual passenger number was higher than what the unit type can possibly provide thus can never be satisfied. Note that the manual solution has another 8 trains under-provided beside this one. Being so, the manual solution

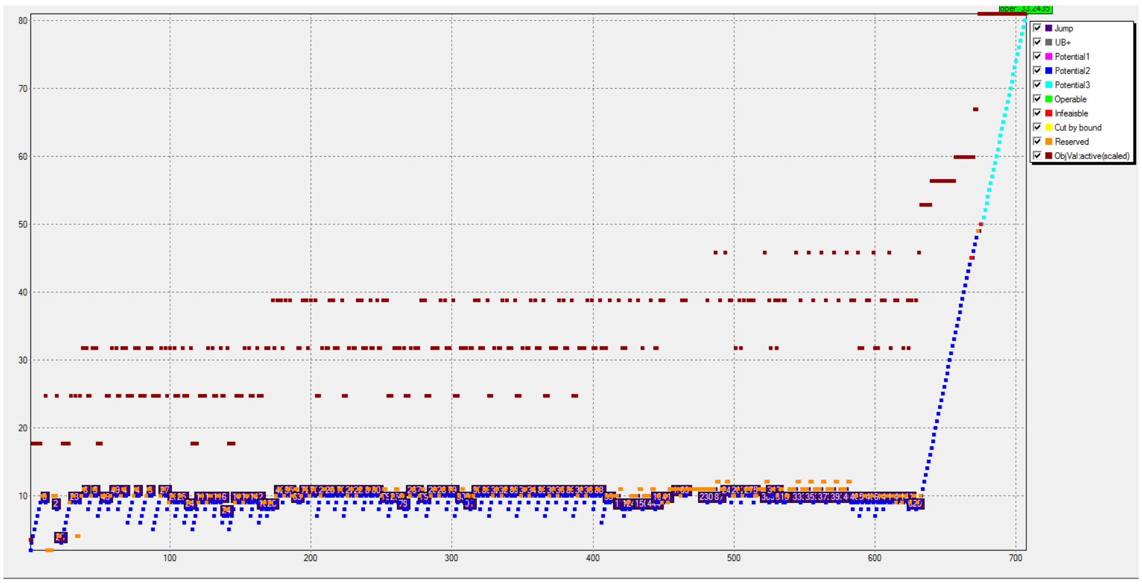


Figure 6.7: Node-depth graph for experiment GB-334R-1a

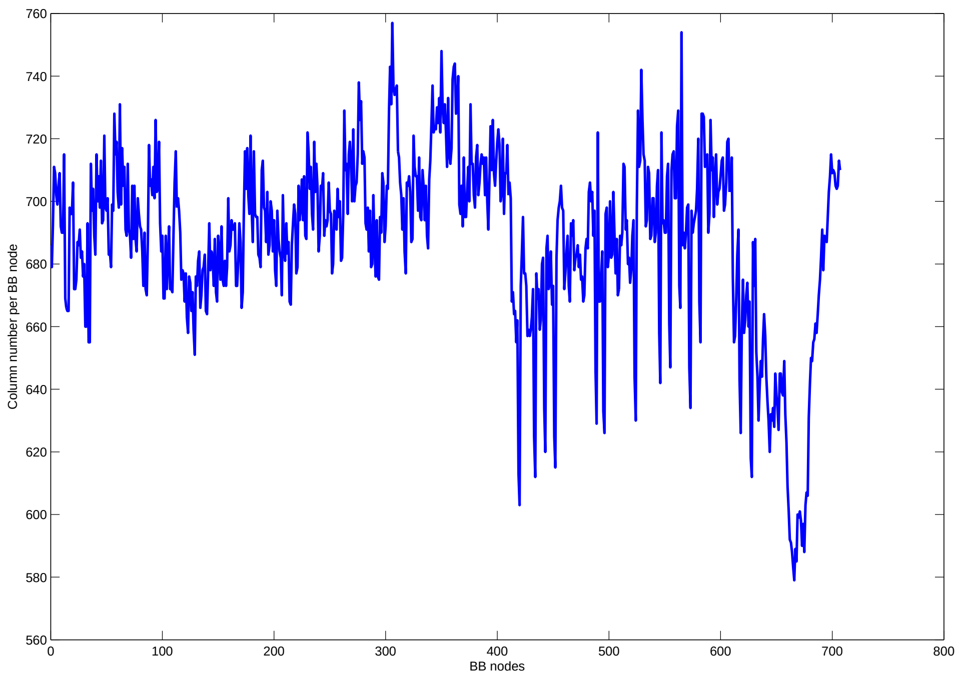


Figure 6.8: Column numbers per tree node for experiment GB-334R-1a

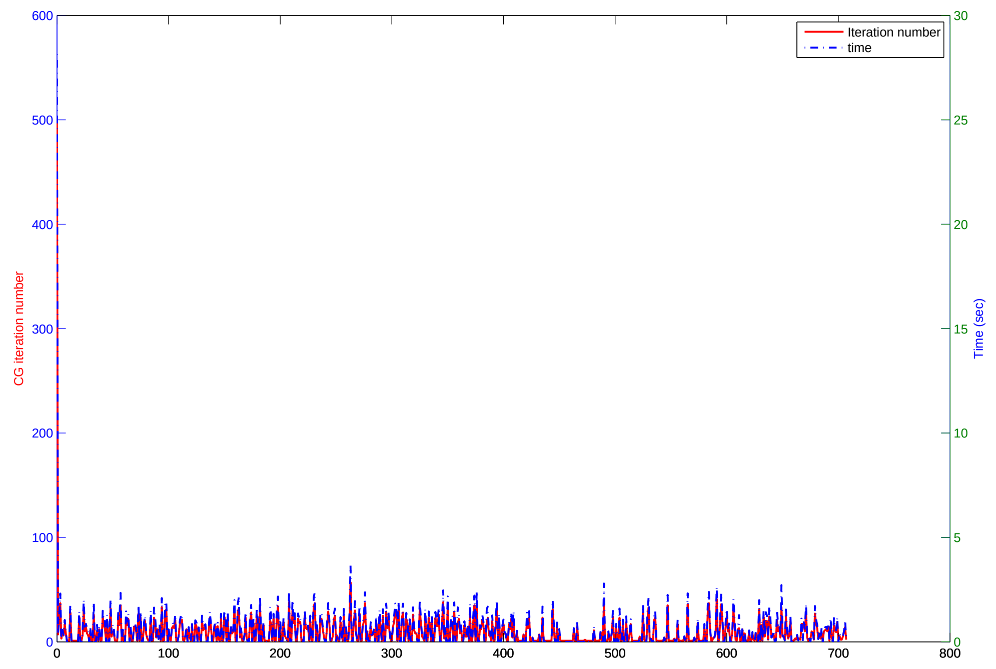


Figure 6.9: CG iteration numbers against CG time per tree node for experiment GB-334R-1a

uses 39 units. Having the above 8 trains' demand satisfied, the solver has to use 40 units, which seems reasonable. The solver uses 5 out of the 9 existing empty-running movements. Interestingly, when the solver's solution was under post-processing, 4 newly created empty-running trains other than any of the existing 14 ECS in the manual diagrams could be identified manually such that after adding them the fleet size returned back to 39. The overall solution turned out to be of the same fleet size, better capacity provision in terms of Fit, OP and UP trains, and less number of ECS trains. Nevertheless, the validity of the 4 newly added empty-running movements has to be further carefully verified by the practitioners.

Figure 6.12 – 6.16 give the relevant figures on experiment GB-318(320)R-1a. Again, the branching rules had no recurrence as shown in the node-depth graph. Note that in Figure 6.13, from node 0 to node 636, the search was at the stage of banned location branching with frequent jumps, giving the oscillation of the curve. On the other hand, since node 637, the rule changed into arc variable branching, giving a sudden increasing in the number of columns.

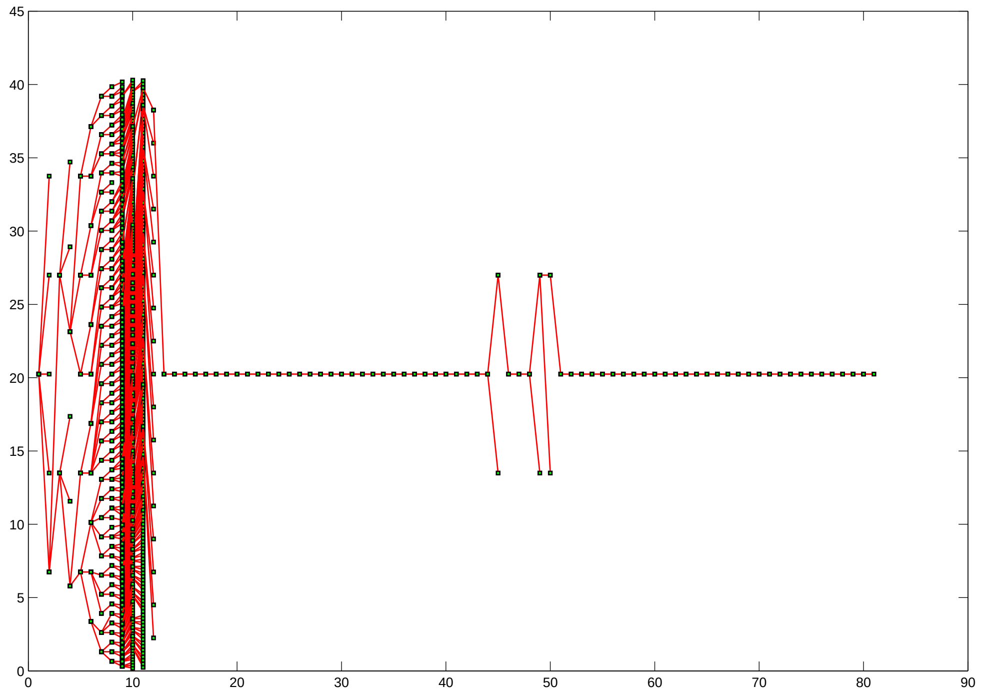


Figure 6.10: BB tree for experiment GB-334R-1a

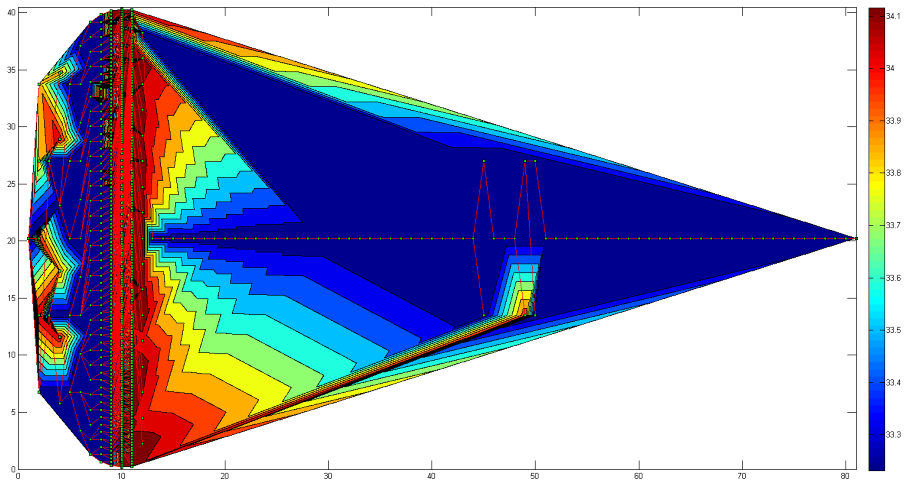


Figure 6.11: BB tree with node values for experiment GB-334R-1a

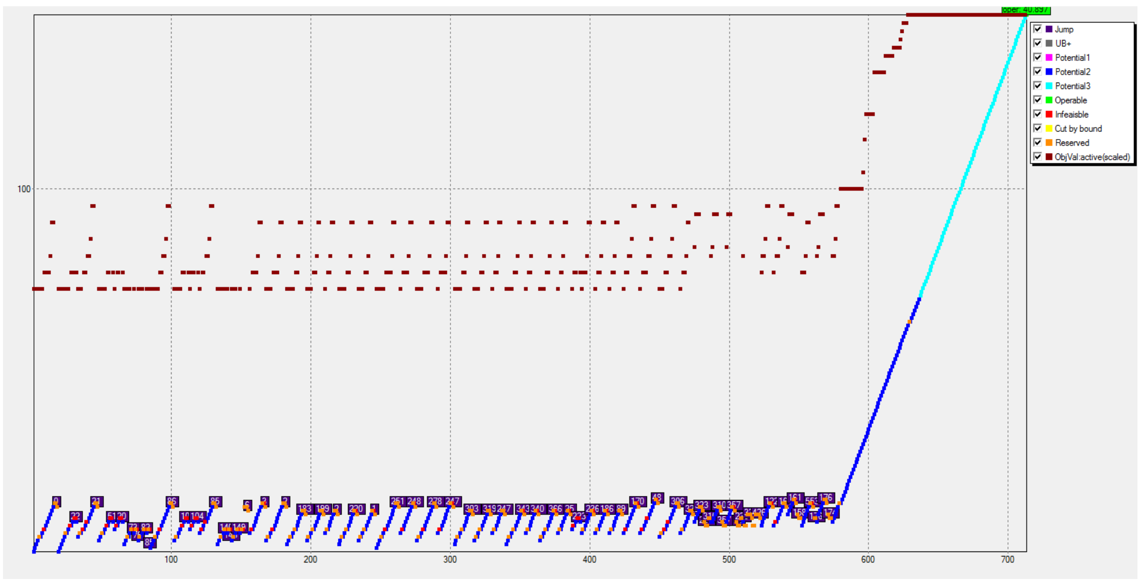


Figure 6.12: Node-depth graph for experiment GB-318(320)R-1a

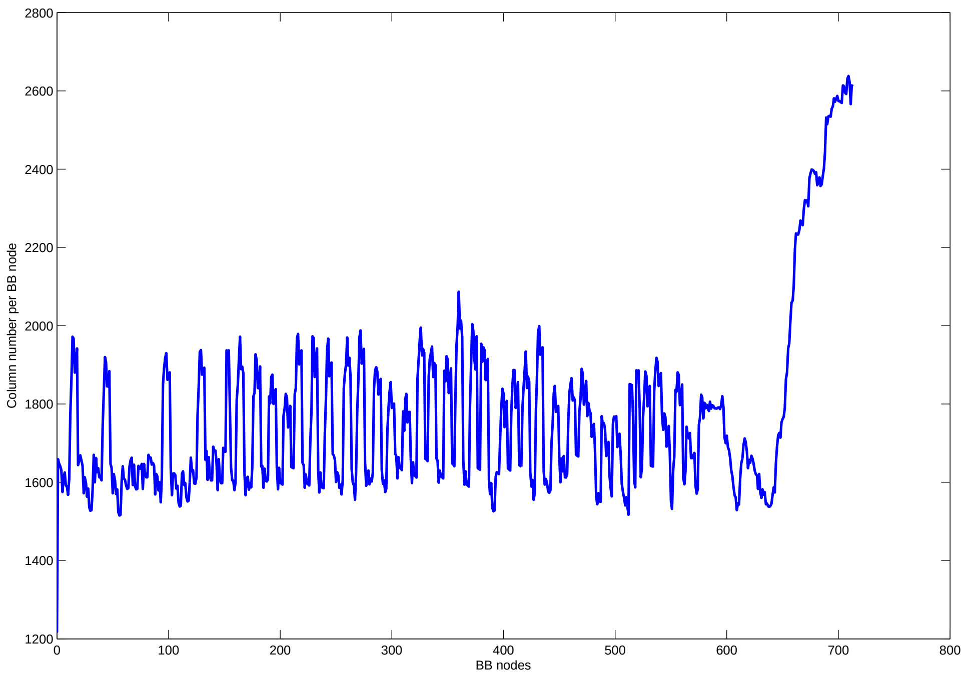


Figure 6.13: Column numbers per tree node for experiment GB-318(320)R-1a

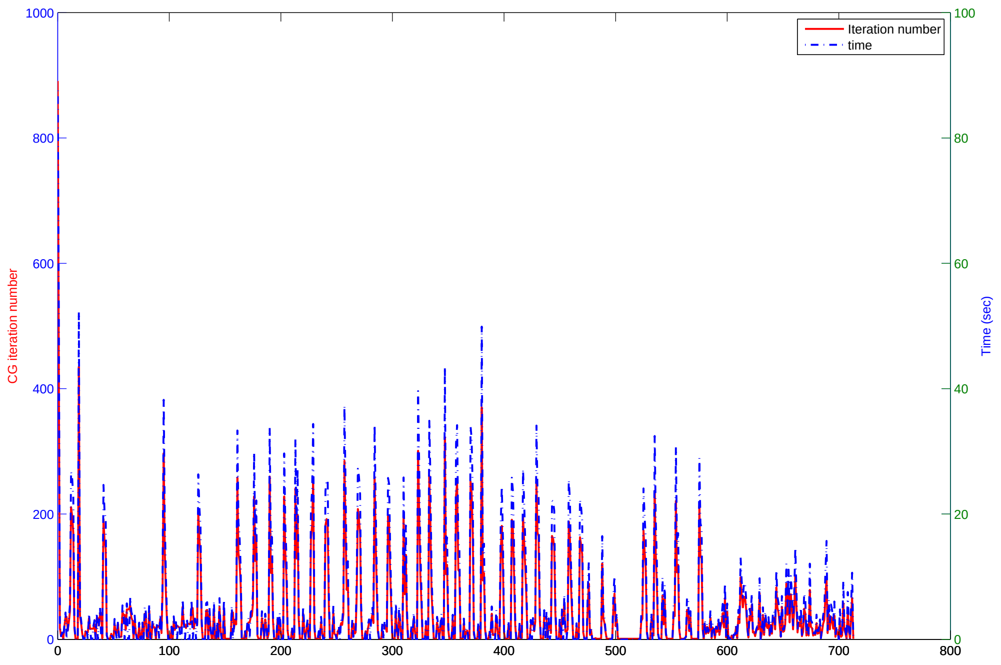


Figure 6.14: CG iteration numbers against CG time per tree node for experiment GB-318(320)R-1a

Table 6.10: Result on experiment GA-314-1a

	unit#	oper-obj	root-obj	root-LB	fit[a]	OP[a]	UP[a]	ECS#	rule%(B/A)	BB#	time
manual	14	–	–	–	248	9	12[b]	2	–	–	–
solver	14	14.406	14.5024	11.0537	253	7	9[b]	1	6/94	419	7766

[a] There were 9 trains whose December 2013 passenger count survey data was unknown. They were mainly newly added trains in the May 2014 timetable.

[b] There were 12 under-provided trains in the manual solution which were all peak hour ones and whose PAX# were more than one unit's capacity but still less than two. Nevertheless, if to let the solver to satisfy all of them, then the number of used units will exceed the fleet size bound. A compromise had been made to let the solver to only satisfy 3 of the 12 trains while there were still 9 under-provided trains.

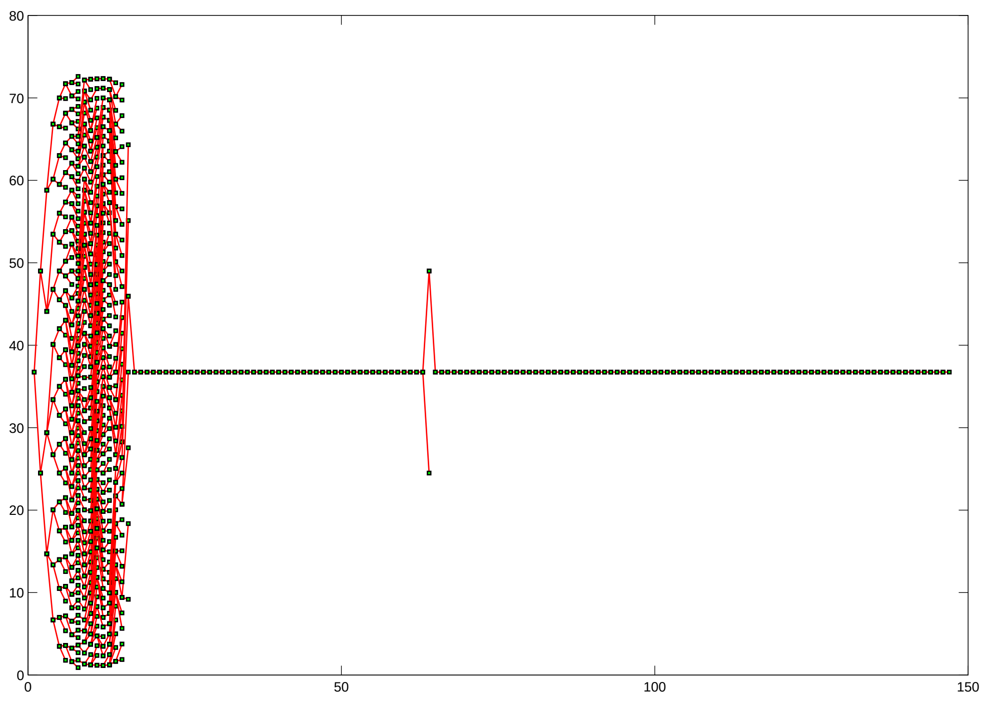


Figure 6.15: BB tree for experiment GB-318(320)R-1a

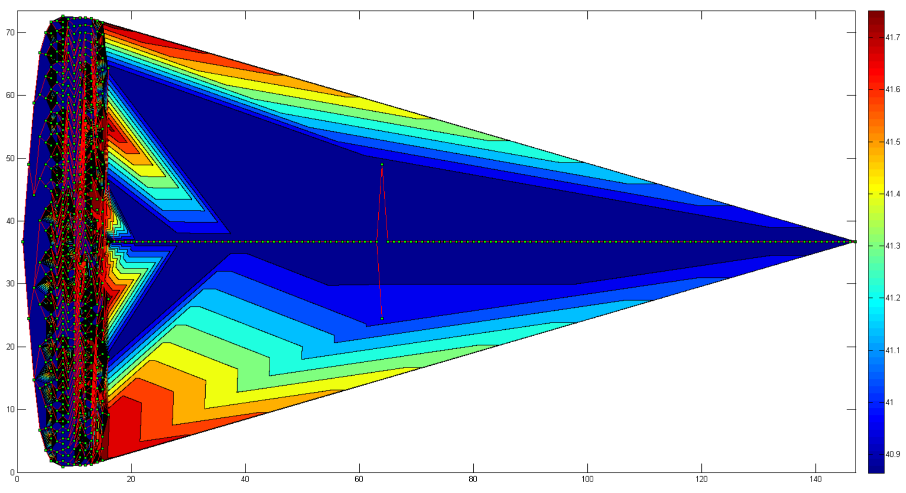


Figure 6.16: BB tree with node values for experiment GB-318(320)R-1a

Table 6.11: Result on experiment GA-380-1a

	unit#	oper-obj	root-obj	root-LB	UP[a]	ECS#	rule%(B/A)	BB#	time
manual	35	–	–	–	14[b]	3	–	–	–
solver	35	35.986	35.7769	35.7285	6[b]	1	2/98	720	12220

[a] There were 17 trains whose December 2013 passenger count survey data was unknown.
[b] There were 14 under-provided trains in the manual solution whose PAX# were theoretically possible to be satisfied. Nevertheless, if to let the solver to satisfy all of them, then the number of used units will exceed the fleet size bound. A compromise had been made to let the solver to only satisfy 8 of the 14 trains while there were still 6 under-provided trains.

6.3.3.3 GA-314-1a

Table 6.10 gives the result on experiment GA-314-1a against the manual solution and Figure 6.18 – 6.19 give the relevant figures. The entire process was quite smooth as a perfect dive almost without any interruptions. Note that the root was not solved to optimality due to the maximum number of CG iterations threshold of 500. The solver uses the same fleet size as the manual diagrams while it has satisfied 3 additional trains whose demand requirement was not meet by the manual solution. The solver also uses one less empty-running movement. The computational time 7766 seconds is quite long considering the relatively small number of 419 tree nodes. This could be due to multiple reasons as listed below.

(i) The timetable pattern of GA area is much denser (i.e. each diagrams contains more trains and the overall turnround time durations at different places are shorter) compared with GB.

(ii) 94 % of the tree nodes used arc variable branching, which is less efficient (more time-consuming and making the ILP more difficult to solve) compared with banned location branching. There were simply not enough banned location targets available to be branched for the instance of GA-314. This was rather different from the instances in the GB area. Figure 6.18 shows that during the branch-and-price the number of columns kept increasing due to the use of arc variable branching.

(iii) There was an abnormal behaviour at the later stage (roughly after node 350) as shown in Figure 6.19, where the CG iteration number and time were not consistent. At the very end part, the computational time had even reached a little less than 200 seconds for completing the column generation for a single BB tree node.

(iv) The root had not been solved to optimality with a relatively large gap to its lower bound (11.0537 vs. 14.5024) since $\varepsilon_{CG} = 500$.

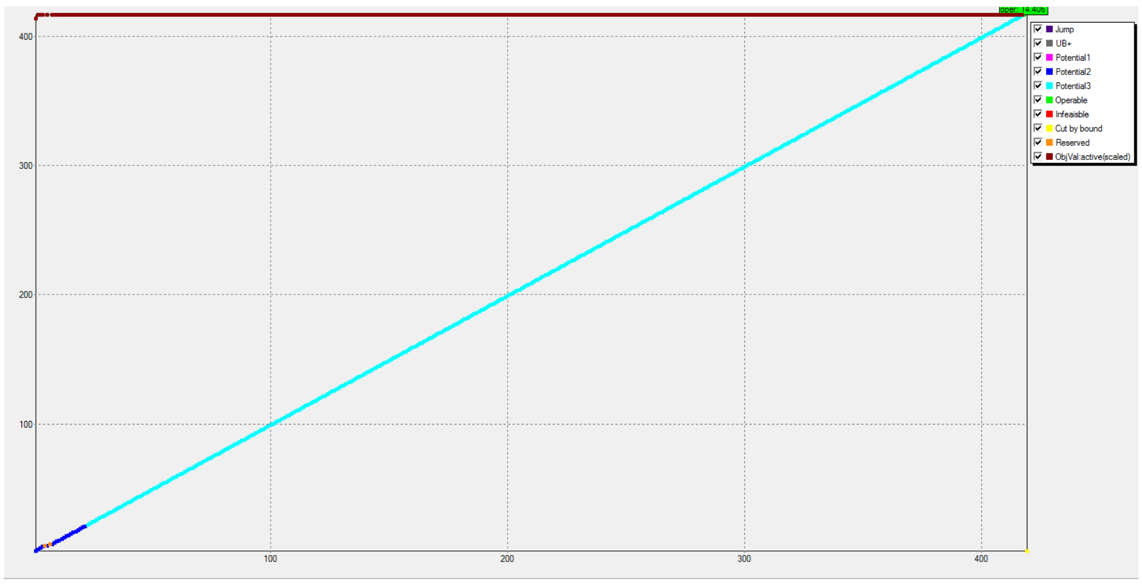


Figure 6.17: Node-depth graph for experiment GA-314-1a

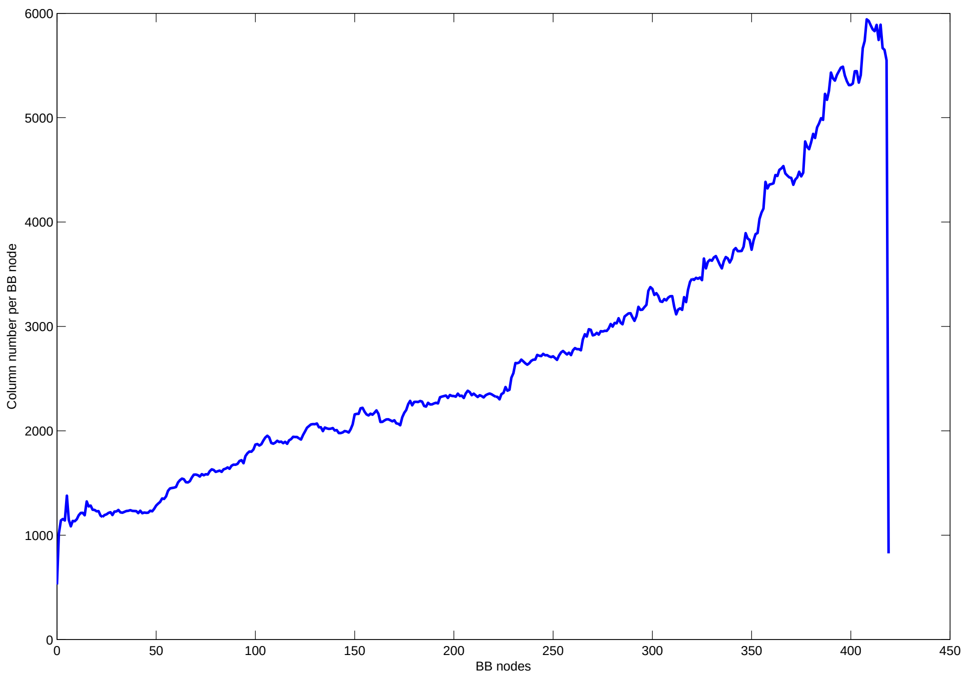


Figure 6.18: Column numbers per tree node for experiment GA-314-1a

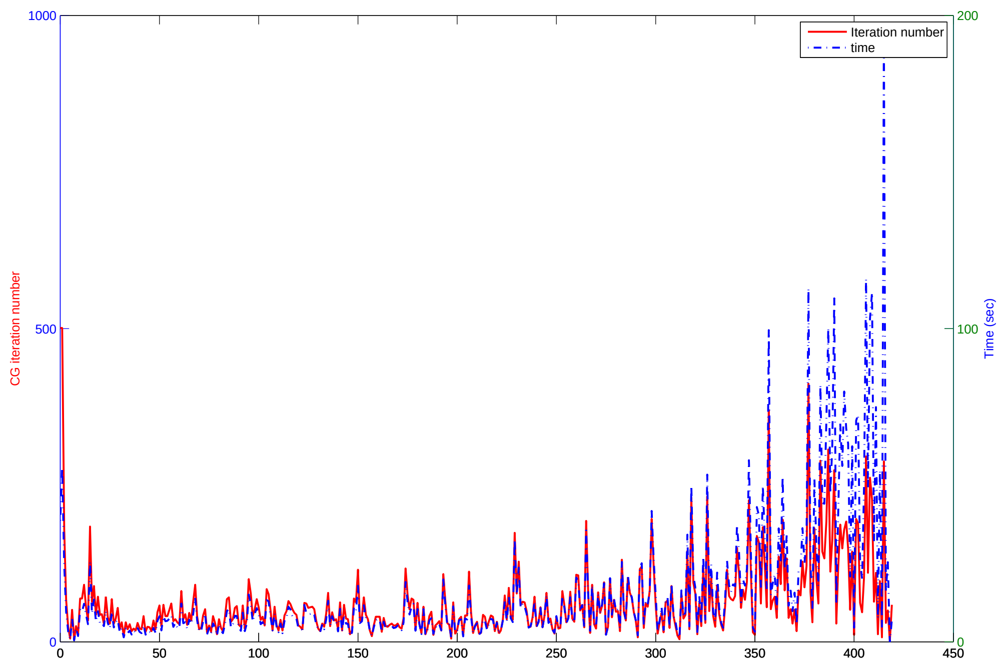


Figure 6.19: CG iteration numbers against CG time per tree node for experiment GA-314-1a

6.3.3.4 GA-380-1a

Table 6.11 gives the result on experiment GA-380-1a against the manual solution and Figure 6.18 – 6.19 give the relevant figures. Since GA-380 is an instance with two types, it is tricky to give a rigorous definition on "over-provision" and "fit" (e.g. when "2×c380/0" is sufficient for a train, would it be appropriate or not to regard "c380/0+c380/1" or "2×c380/1" as over-provided for the train). Therefore only the numbers of under-provided trains are given in Table 6.11.

From Figure 6.18 – 6.19, it can be observed that the behaviour of the solver in GA-380-1a is very similar as in GA-314-1a. Both instances could not provide many banned location branching opportunities such that the arc variable branching was dominant throughout the processes, giving dramatic increase in column numbers and computational time.

Similar as in GA-314-1a, attempting to satisfy the 14 under-provided trains in the manual solutions will lead to infeasible solutions violating the fleet size limit. Therefore compromise had to be made to only satisfy 8 out of the 14 trains in the solver without violating the overall fleet size limit. The solver only used 1 empty-running train from the 3 existing ones.

The four points given in § 6.3.3.3 in analysing the solver's behaviour in GA-314-1a are basically all applicable to GA-380-1a. Clearly the timetable patterns in the GA area

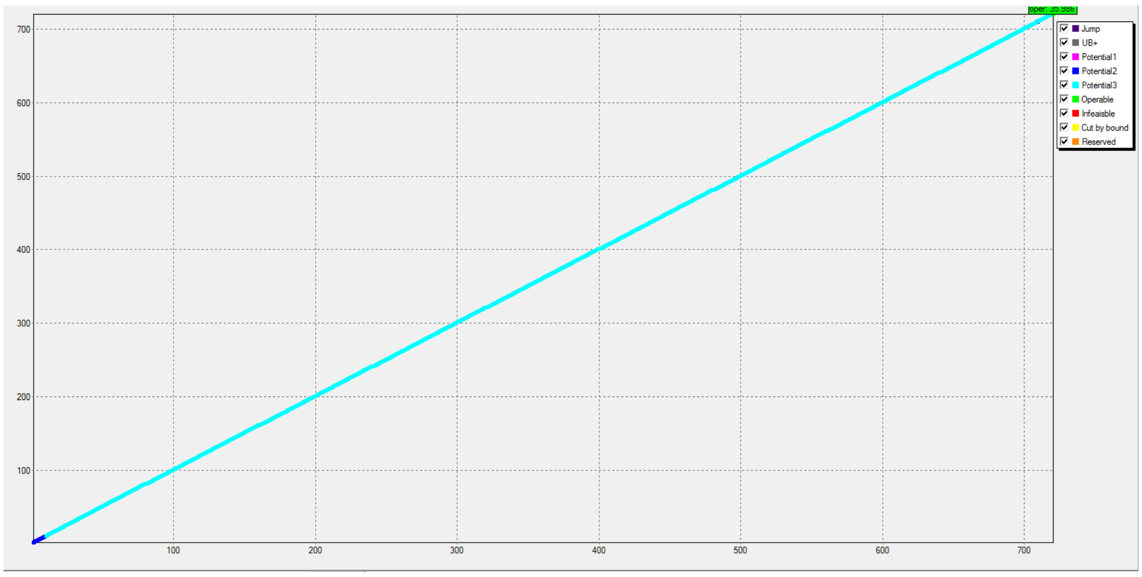


Figure 6.20: Node-depth graph for experiment GA-380-1a

are significantly different from the ones in GB. The denser patterns in the former along with the scarce opportunity for banned location branching make the instances in GA more difficult to solve than GB.

6.3.3.5 Conclusion for the experiments on the May 2014 timetable

Given the actual passenger numbers were available for many trains, the solver is able to provide solutions that are significantly better than the current manual ones and it is possible to give comparisons on the capacity provision between the manual and the solver's results for most trains. For the two experiments on GB, besides similar conclusions as for GB-entire-1a can be drawn regarding the two-level framework's feasibility and the branch-and-price solver's operability, the solution qualities in fleet size and capacity provision have been greatly improved in GB-334R and GB-318(320)R compared with the manual diagrams. This shows the power of an optimisation-based tool in making the best or much better unit resource distribution assignments across the network with respect to actual PAX numbers that a manual approach may find it very difficult to achieve. Moreover, the solver can produce the results within a few hours compared with the manual approach which requires weeks to months. The problem instances in GA are harder for the solver due to reasons like denser timetable patterns, less locations banned for coupling/decoupling etc. In addition, as shown in Table 6.10 and Table 6.11, these instances are also challenging due to the under-provided trains that could not be all satisfied using the limited unit resources. It is an interesting topic to be further investigated that among the under-provided trains how to satisfy as many as possible but not necessarily all while keeping the fleet size within the required limit. Those issues will be left for future research.

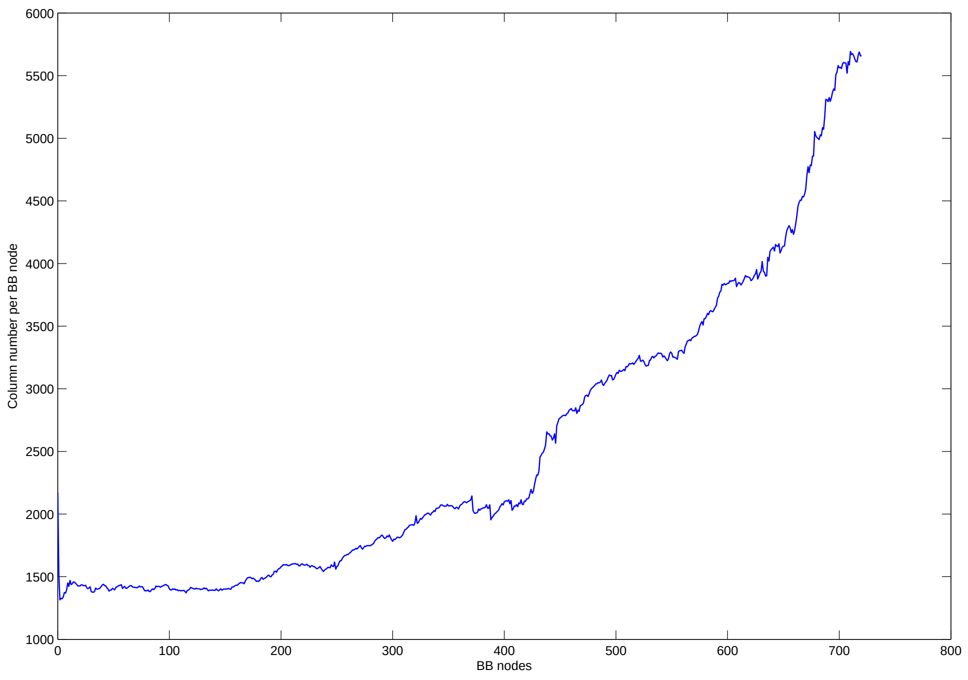


Figure 6.21: Column numbers per tree node for experiment GA-380-1a

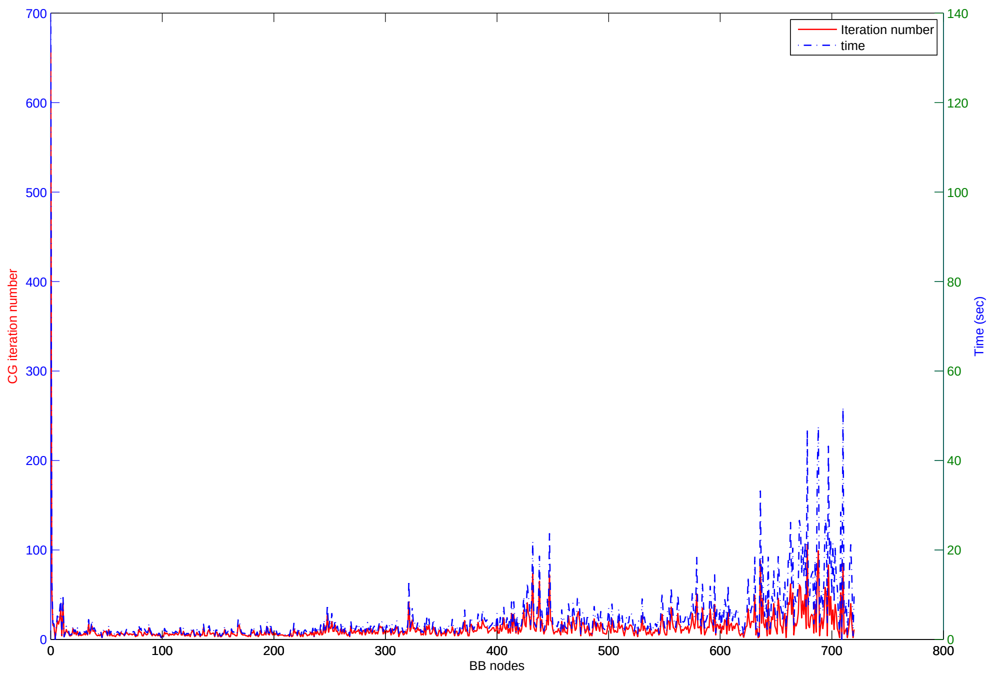


Figure 6.22: CG iteration numbers against CG time per tree node for experiment GA-380-1a

Table 6.12: Non-general parameter setting of the fine-tuning experiments on GB-334R

Experiment	M_{CG}	ε_{CG}	filter/it_F, ρ_{RC}	ρ_B	order	strategy	Δ^0	jump/G	$iniL$
GB-334R-2	varied	0	no/–	0.01	B≻A	switch@new-rule	0.5	yes/3	0
GB-334R-3	450	0	no/–	0.01	B≻A	switch@new-rule	0.5	yes/25	varied
GB-334R-4	800	varied	no/–	0.01	B≻A	switch@new-rule	0.5	yes/25	3
GB-334R-5	800	0.05	varied	0.01	B≻A	switch@new-rule	0.5	yes/25	3
GB-334R-6	800	0.05	yes/20,0.8	0.01	varied	varied	0.5	yes/25	3
GB-334R-7	800	0.05	yes/20,0.8	varied	B≻A	switch@new-rule	0.5	yes/25	3
GB-334R-8i	800	0.05	yes/20,0.8	0.01	B≻A	switch@new-rule	0.5	varied	3
GB-334R-8ii	800	0.05	yes/20,0.8	0.01	B≻A	switch@new-rule	1	varied	0
GB-334R-9i	800	0.05	yes/20,0.8	0.01	B≻A	switch@new-rule	varied	yes/25	0
GB-334R-9ii	800	0.05	yes/20,0.8	0.01	B≻A	switch@new-rule	varied	no	0

6.3.4 Fine-tuning: A series of experiments on GB-334R

6.3.4.1 Experimented parameters

Now we report a series of experiments based on the GB-334R instance by varying different parameters in fine-tuning the branch-and-price solver. The purpose is to have a deeper insight on the branch-and-price solver regarding to parameter settings. The experiments tested were grouped into the following categories: the column generation early stop parameters M_{CG} and ε_{CG} (cf. § 5.4.3), the column filter and relevant parameters it_F and ρ_{RC} (cf. § 5.4.2), the banned location branching parameter ρ_B (cf. § 5.3.1.2), the branching rule order and strategies (cf. § 5.3.1.4), the estimated upper bound initial increment Δ^0 (cf. § 5.4.4), the jump option and the jump gap G (cf. § 5.3.2.2) and the initialisation level $iniL$ (cf. § 5.3.2.1). Table 6.12 gives the names and non-generic parameter settings for the fine-tuning experiments.

6.3.4.2 GB-334R-2: The maximum column generation iteration number M_{CG}

In this group of experiments the maximum column generation iteration number M_{CG} were varied from 600 to 250. To solve each BB tree node to exact optimality may yield too many columns that can slow down the simplex solver. Therefore when the iteration number exceeds M_{CG}, the column generation at the current node will stop and the lower bound given by (5.8) will be used as the node's value. This capping strategy may seem less reasonable than a strategy that measures the gap between the upper and lower bound given by (5.8). However, given the special case that all of our experiments used the column inheritance method such that the column iteration number at the root would be generally much larger than the iteration numbers at the leaves, this cap of M_{CG} was mainly set for limiting the number of column generated at the root. The advantage is that all leaf nodes can be solved to exact optimality by only a few column generation iterations, compared with the case in setting a bound gap parameter as ε_{CG} which may have many leaf nodes solved non-optimally.

Table 6.13 gives the result of Experiment GB-334R-2. It can be observed that the differences among the instances with different M_{CG} are not very notable. Only the root of 2b was solved to optimality and for all the other instances, only 2c used less time than 2b. From Figures A.1 and A.2 in Appendix A, the number of columns had not been notably reduced in the instances with non-optimal roots. And when the root's quality was really poor, i.e. in 2f and 2g, some leaf nodes had to yield extra numbers of iterations at the early stages to do some remedies to obtain sufficient paths.

Table 6.13: Results on experiment GB-334R-2

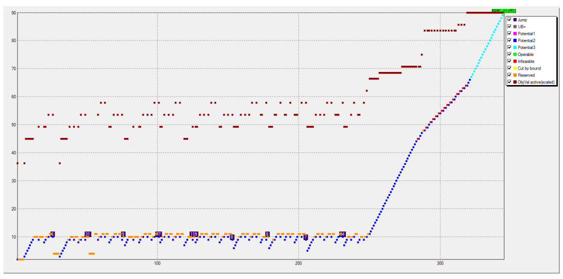
solver	M_{CG}	unit#	oper-obj	root-obj	root-LB	fit	OP	UP	ECS#	rule%(B/A)	BB#	time
2b	600[a]	33	33.2435	33.232	33.232	168	15	1	0	96/4	707	559
2a	500	33	33.245	33.232	33.2287	167	16	1	0	98/2	754	596
2c	450	33	33.245	33.2324	33.224	167	16	1	0	99/1	630	461
2d	400	33	33.2445	33.2331	33.2212	167	16	1	0	98/2	1075	917
2e	350	33	33.245	33.2343	33.2053	167	16	1	0	97/3	927	742
2f	300	33	33.245	33.2362	33.201	167	16	1	0	96/4	838	651
2g	250	33	33.245	33.2378	33.188	167	16	1	0	100/0	843	749

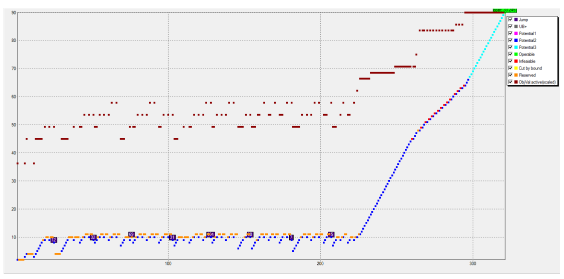
[a] It needed 503 column generation iterations to solve the root to optimality.

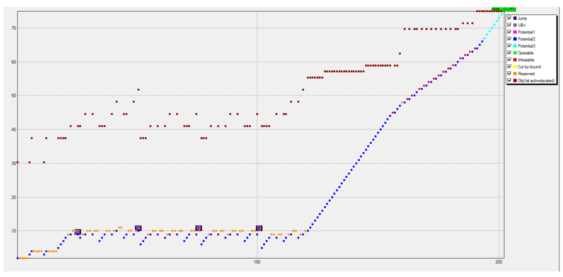
It appears that for the instances tested in GB-334R-2 which represents small scale timetables, reducing M_{CG} cannot significantly improve the performance of the branch-and-price process. Therefore, for most of the remaining experiments we would set a large number of $M_{CG} = 800$ to ensure the root can be solved to optimality. It is unknown what the effect in capping a maximum CG iteration number by M_{CG} will be for medium/large scale instances.

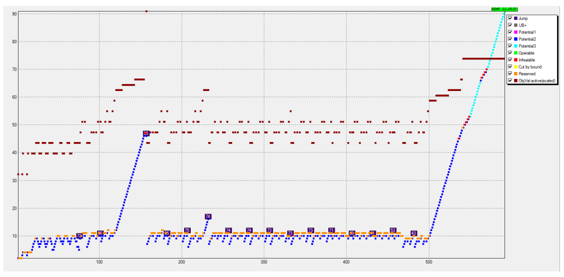
6.3.4.3 GB-334R-3: The initialisation level *iniL*

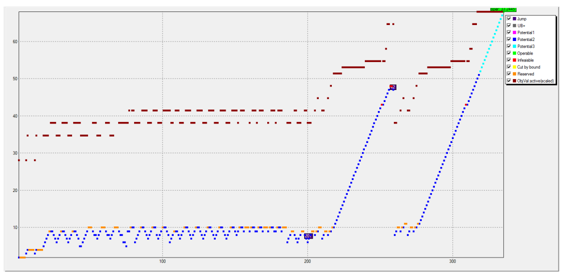
Table 6.14 gives the results on experiments GB-334-3 and Figures 6.23–6.24 show the relevant node-depth graphs and BB trees. Note the root is regarded as level 0 and the node where the initialisation was completed is also marked after BB#. The table shows for the instance of GB-334R and the fixed settings on other parameters, initialising within 3 levels would be the best strategy, especially since the branch-and-price finished within only 202 BB nodes. On the other hand, it may be not a good idea to set *iniL* a large value, especially larger than 9. Also when $iniL \geq 5$, most of the BB nodes are within the initialisation stage where the jumps would not be triggered, as what can be observed from Figure 6.23. Lack of jump actions may significantly prolong the branch-and-price process, as shown in Table 6.14.

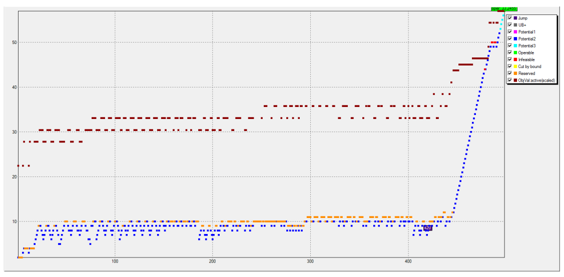

(a) $iniL = 1$

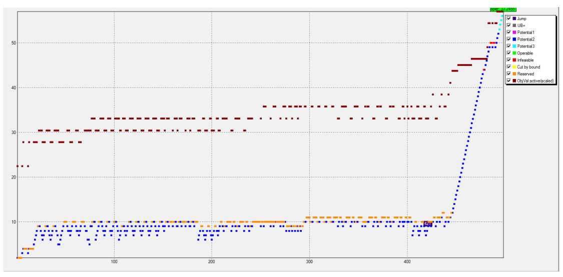

(b) $iniL = 2$

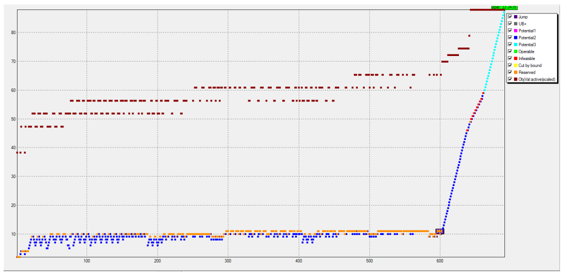

(c) $iniL = 3$

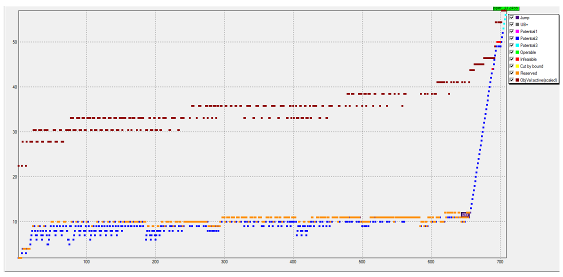

(d) $iniL = 4$


(e) $iniL = 5$


(f) $iniL = 6$


(g) $iniL = 7$


(h) $iniL = 8$


(i) $iniL = 9$

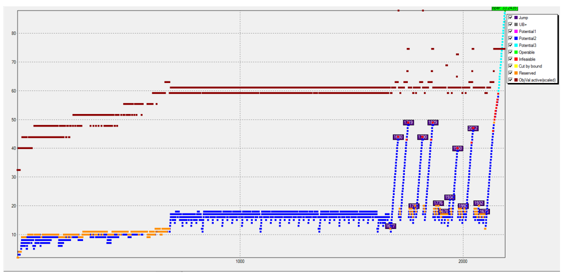
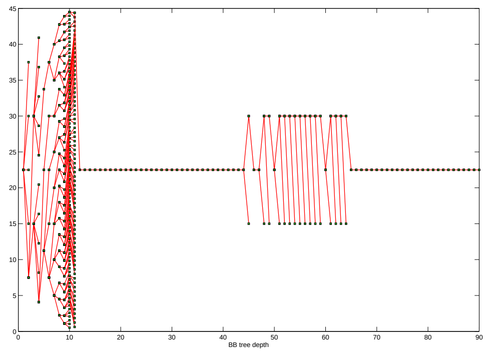
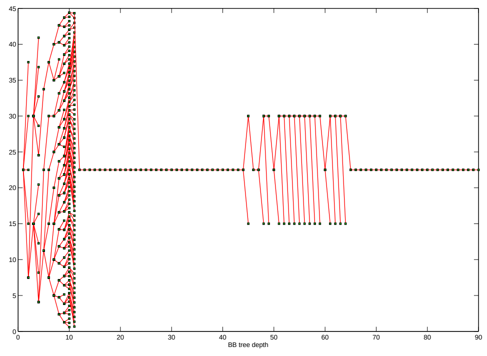

(j) $iniL = 10$

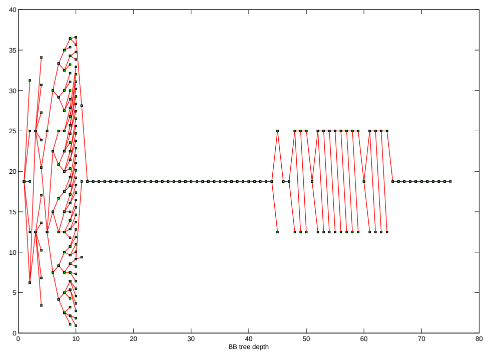
Figure 6.23: Node-depth graphs from experiments GB-334R-3



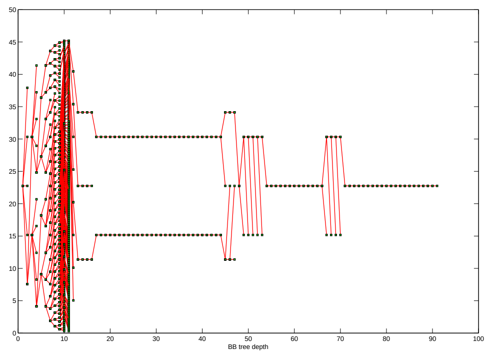
(a) $iniL = 1$



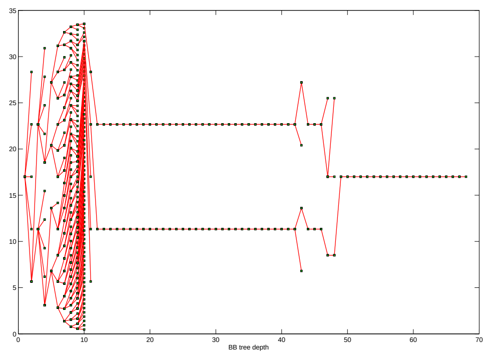
(b) $iniL = 2$



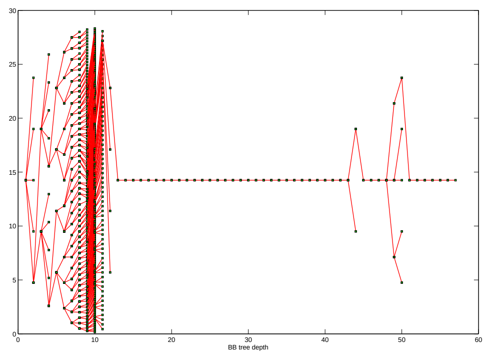
(c) $iniL = 3$



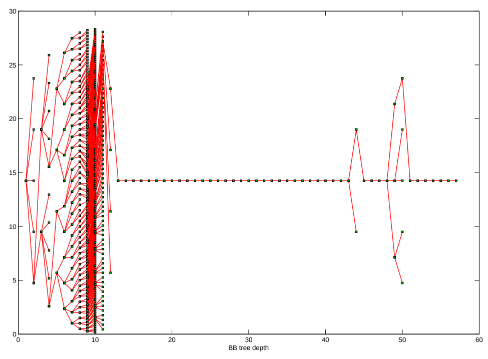
(d) $iniL = 4$



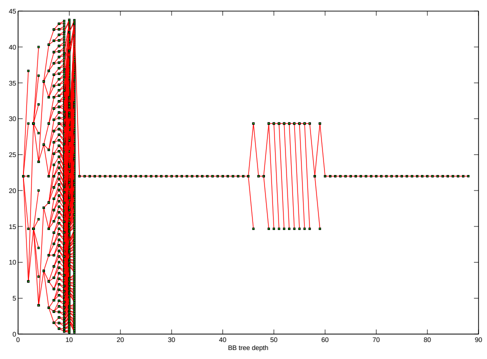
(e) $iniL = 5$



(f) $iniL = 6$



(g) $iniL = 7$

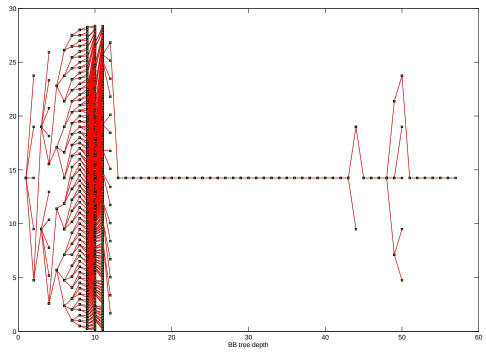


(h) $iniL = 8$

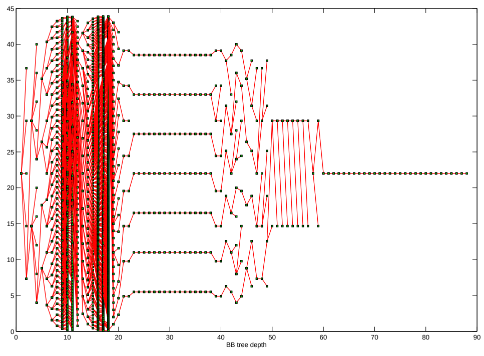
Table 6.14: Results on experiment GB-334R-3

solver	*iniL*	unit#	oper-obj[a]	fit	OP	UP	ECS#	rule%(B/A)	BB# (ini. done)	time
3a	0	33	33.245	167	16	1	0	93/7	346 (0)	243
3b	1	33	33.245	167	16	1	0	93/7	345 (6)	327
3c	2	33	33.245	167	16	1	0	92/8	321 (13)	307
3d	3	33	33.245	167	16	1	0	96/4	202 (18)	216
3e	4	33	33.2435	168	15	1	0	94/6	592 (76)	552
3f	5	33	33.2445	167	16	1	0	95/5	335 (201)	370
3g	6	33	33.2455	166	17	1	0	99/1	498 (420)	439
3h	7	33	33.2455	166	17	1	0	99/1	498 (421)	446
3i	8	33	33.2435	168	15	1	0	96/4	691 (596)	549
3j	9	33	33.2455	166	17	1	0	99/1	709 (650)	573
3k	10	33	33.2435	168	15	1	0	99/1	2188 (1677)	2921

[a] root-obj = 33.2324 and root-LB = 33.224 for all instances.



(i) $iniL = 9$



(j) $iniL = 10$

Figure 6.24: BB trees from experiments GB-334R-3

Table 6.15: Results on experiment GB-334R-4

solver	ε_{CG}	unit#	oper-obj	root-obj	root-LB	fit	OP	UP	ECS#	rule%(B/A)	BB#	time
4a	0.01	33	33.245	33.2328	33.2231	167	16	1	0	98/2	474	338
4b	0.025	33	33.2475	33.2342	33.2093	166	17	1	0	73/27	217	160
4c	0.05	33	33.247	33.2379	33.188	168	15	1	0	50/50	216	153
4d	0.075	33	33.249	33.2385	33.1642	166	17	1	0	97/3	2301	1631
4e[a]	0.1	–	–	33.2403	33.1411	–	–	–	–	–	>4617	>5000

[a] Unfinished within the time limit.

6.3.4.4 GB-334-4: Column generation stop gap tolerance ε_{CG}

This group of experiments was conducted by varying the gap tolerance ε_{CG}. The absolute form $UB - LB \leq \varepsilon_{CG}$ was used. It was expected that there should be a value for ε_{CG} which gives the most efficient performance for the BB tree. Table 6.15 gives the results on experiments GB-334-4, which shows gradually varied ε_{CG} can lead to solution processes that are quite different and the expectation on the value has been roughly confirmed by the overall trend of the results. For instance, the solver with $\varepsilon_{CG} = 0.05$ uses only 216 BB nodes and 153 seconds while the solver with $\varepsilon = 0.75$ uses 2301 BB nodes and 1631 seconds. When the tolerance was set too wide, such as $\varepsilon_{CG} = 0.1$, the solver could not find any operable solution within time limit.

As for the instance GB-334R, the gap tolerance ε_{CG} appears to be more sensitive and will result in more drastic changes (either good or bad) compared with setting a maximum column generation iteration number as M_{CG}. Understandably, different early stop settings by varying M_{CG} would mainly affect the columns generation process at the root while the ones by varying ε_{CG} could both affect the root and the leaves.

6.3.4.5 GB-334-5: Column filter, it_F and ρ_{RC}

During the column inheritance procedure, when the column filter is enabled, if the column generation iteration number at the parent node exceeds it_F, then all columns which, (i) contain unused arcs in the LP optimal solution at the parent node and (ii) have a reduced cost larger than ρ_{RC}, will not be passed down to their children nodes. Table 6.16 gives the results on Experiment GB-334-5. No significant pattern strongly connected with it_F and ρ_{RC} can be identified from the results although the performances could be quite different from solver to solver. Even when the strategies were very radical, like deleting all affected paths with positive reduced costs as in 5a–5e with $\rho_{RC} = 0$, or applying the filter at every node irrespective of the detection of excessive column generation iterations as in 5w–5af with $it_F = 1$, optimal operable solutions could always be found. Nevertheless, some local patterns could be found. The solvers' performance seemed to be stable when it_F was set around 20. Radically setting $it_F = 1$ could often result in more BB tree nodes and longer solution time, unless ρ_{RC} was also more restrictively set a higher value. It is unclear whether these patterns are only the characteristics from the individual case of instance GB-334R.

On the other hand, as for the effects in reducing the number of columns by enabling the column filter, Figure A.3 in Appendix A shows that evident drops in the number of columns at leaf nodes compared with the root can be observed. This is generally in contrast with the cases where the column filter was disabled such that the number of columns were generally in a rising or oscillation state unless the search jumps back to the top of the tree. Note that at a later stage of a dive with the arc variable branching in use, the number of columns often increases drastically.

The experiments show that the column filter can effectively reduce the number of columns at leaf nodes on instance GB-334R.

6.3.4.6 GB-334R-6: Branching rule order and strategies

This group of experiments involves two variable settings. One is order of the two branching rules available for GB-334R (banned location branching and arc variable branching). Generally the banned location branching is considered to be more preferred than the arc variable branching, since it forms branches by only deleting columns in the RMP and nodes/arcs in subproblems. The other setting is the strategy in controlling the multiple branching rules. Three strategies are available. "Static" keeps the same order of rule priority during the entire branch-and-price process; "switch@new-rule" changes the head rule from the original one to its next rule in the initial list when the branching targets associated with the original rule has been temporarily used up; "switch@stagnation" will switch the head rule from the original one to its next rule in the initial list when the process has been "trapped" in the current rule more than M_{UG} search rounds.

Table 6.17 gives the results on Experiment GB-334R-6. Note that as long as the

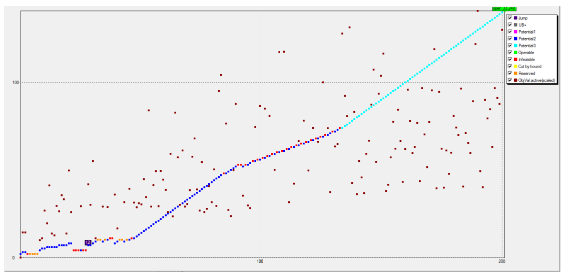
Table 6.16: Results on experiments GB-334R-5

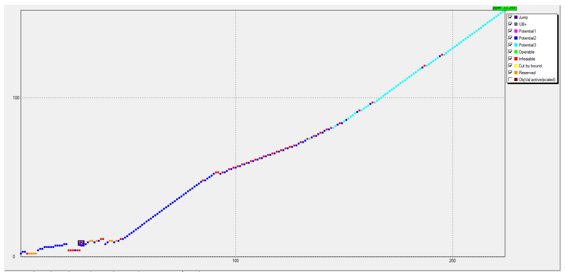
solver	it_F	ρ_{RC}	unit#	oper-obj[a]	fit	OP	UP	ECS#	rule%(B/A)	BB#	time
5a	20	0	33	33.254	161	22	1	0	75/22	207	381
5b	25	0	33	33.249	166	17	1	0	68/32	215	197
5c	30	0	33	33.2545	161	22	1	0	93/7	669	437
5d	35	0	33	33.2495	166	17	1	0	57/43	259	150
5e	40	0	33	33.2495	166	17	1	0	57/43	259	154
5f	20	0.1	33	33.2379	169	14	1	0	91/9	590	372
5g	25	0.1	33	33.2379	169	14	1	0	91/9	601	382
5h	30	0.1	33	33.2455	169	14	1	0	91/9	572	363
5i	35	0.1	33	33.2455	169	14	1	0	91/9	572	375
5j	10	0.2	33	33.247	168	17	1	0	51/49	205	140
5k	20	0.2	33	33.248	167	16	1	0	60/40	201	128
5l	25	0.2	33	33.248	167	16	1	0	60/40	201	126
5m	10	0.3	33	33.2465	168	15	1	0	52/48	198	141
5n	20	0.3	33	33.248	167	16	1	0	60/40	201	128
5o	25	0.3	33	33.248	167	16	1	0	60/40	201	127
5p	20	0.4	33	33.248	167	16	1	0	60/40	201	114
5q	20	0.5	33	33.248	167	16	1	0	60/40	201	134
5r	20	0.6	33	33.248	167	16	1	0	60/40	201	134
5s	20	0.7	33	33.248	167	16	1	0	60/40	201	120
5t	20	0.8	33	33.248	167	16	1	0	61/39	201	104
5u	20	0.9	33	33.247	168	15	1	0	72/28	1678	1121
5v	20	1.0	33	33.2505	165	18	1	0	75/25	362	247
5w	1	0.1	33	33.2495	166	17	1	0	90/10	665	540
5x	1	0.2	33	33.2465	168	15	1	0	65/35	285	207
5y	1	0.3	33	33.2485	167	16	1	0	71/29	973	682
5z	1	0.4	33	33.2485	167	16	1	0	71/29	973	685
5aa	1	0.5	33	33.2485	167	16	1	0	71/29	973	691
5ab	1	0.6	33	33.2485	167	16	1	0	71/29	973	680
5ac	1	0.7	33	33.2485	167	16	1	0	71/29	973	700
5ad	1	0.8	33	33.249	166	17	1	0	73/27	186	106
5ae	1	0.9	33	33.249	167	16	1	0	71/29	349	204
5af	1	1.0	33	33.2475	167	16	1	0	58/42	222	145

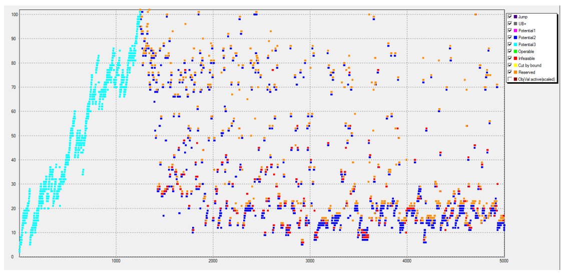
[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.

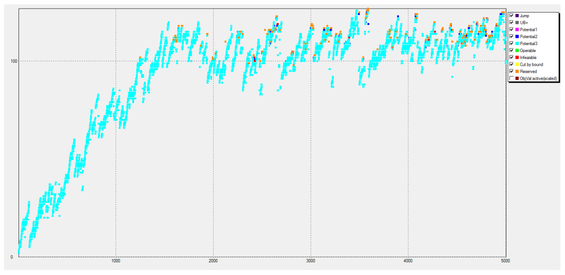
order was set B$\succ$A, meaning the banned location branching has a higher priority than arc variable branching, optimal solutions could always be found by the solver, as shown in 6a,6b,6c–6g. On the other hand, none of 6d,6e,6h–6j which used A$\succ$B, could find any operable solution within the 5000 tree nodes limit, no matter what strategies were in use. The above experiments show the importance of setting the correct rule order (here in GB-334R-6, B$\succ$A) for many train unit scheduling problem instances, which can yield completely different results. Another interesting issue is the fact that it seems starting the branch-and-price process with arc variable branching could seriously undermine the solver's performance over the BB tree, even the arc variable branching was in use for only a small number of rounds. This particular issue was found in Experiments 6k–6m where the strategy was modified such that only the first switch@stagnation action was allowed and then the process would be exactly the same as switch@new-rule. The M_{CG} was set to be relatively small (5,10,50) such that the initial stage starting with arc variable branching would not last a very long time and then banned location branching would be dominant for a long time until it was used up. 6k–6m show that although the solver could find optimal solutions, the efficiency was rather poor compared with 6a,6b,6c–6g. In 6m, even only the first five nodes were using the arc variable branching and from the sixth node the entire process would be exactly the same as a B$\succ$A+switch@new-rule, the numbers of BB nodes used would be almost 4–5 times larger. Figure 6.25 give the node-depth graphs from Experiment GB-334R-6.

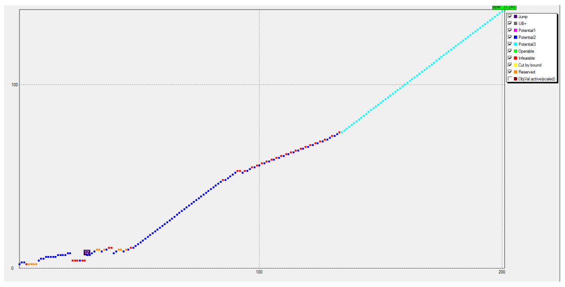
In conclusion, the importance of correctly setting the order of the branching rules in a multi-rule branching system cannot be over-emphasised. The relations can be more complicated in a three-rule or four-rule system that will be developed in the future. For instance, given arc variable branching is probably less desirable than banned location branching and train-family branching, it is also important to investigate what kind of priority relation exists between the two latter rules that both only realise the branches by removing columns and subproblem components.

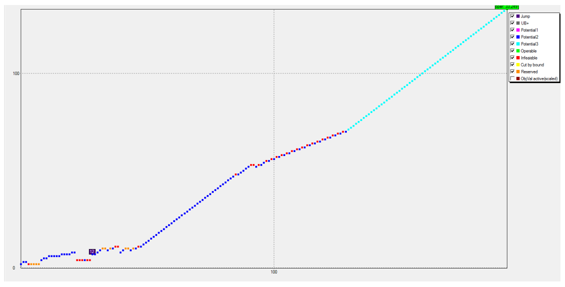

(a) 6a: B≻A, switch@new-rule

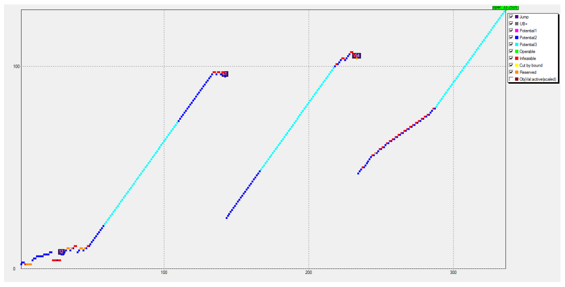

(b) 6b: B≻A, static


(c) 6d: A≻B, switch@new-rule


(d) 6e: A≻B, static


(e) 6c: B≻A, switch@stagnation, $M_{UG} = 200$


(f) 6f: B≻A, switch@stagnation, $M_{UG} = 100$


(g) 6g: B≻A, switch@stagnation, $M_{UG} = 50$

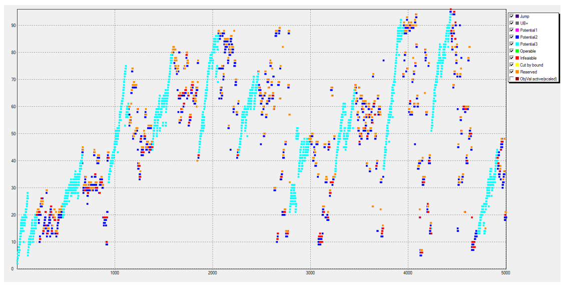
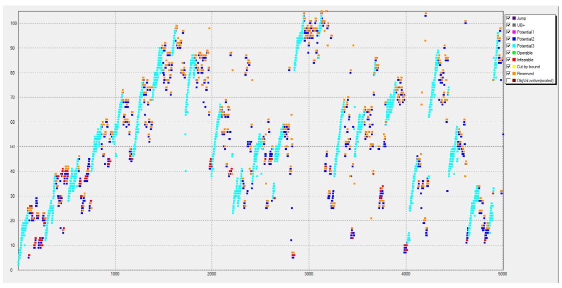
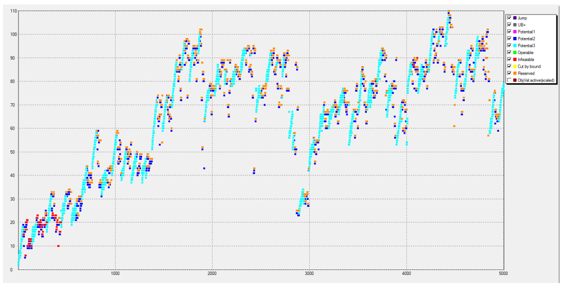

(h) 6h: A≻B, switch@stagnation, $M_{UG} = 200$

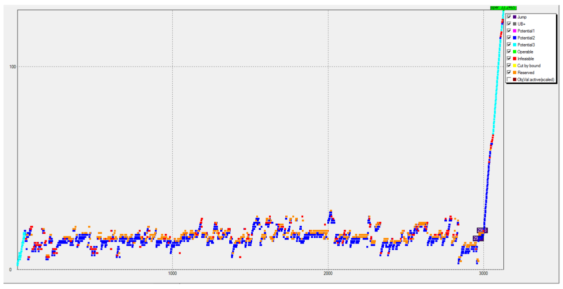
Table 6.17: Results on experiments GB-334R-6

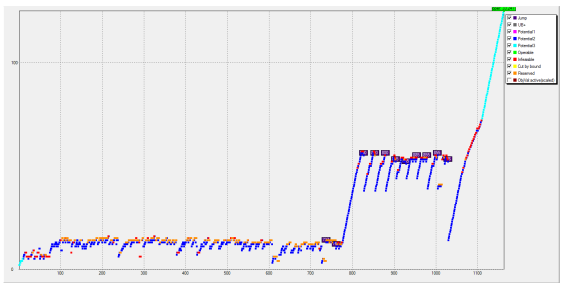
solver	order	strategy	unit#	oper-obj[a]	fit	OP	UP	ECS#	rule%(B/A)	BB#	time
6a	B≻A	switch@new-rule	33	33.248	167	16	1	0	61/39	201	131
6b	B≻A	static	33	33.249	166	17	1	0	63/37	224	114
6d[b]	A≻B	switch@new-rule	–	–	–	–	–	–	71/29	>5000	>5805
6e[b]	A≻B	static	–	–	–	–	–	–	6/94	>5000	>31720
6c	B≻A	switch@stagnation, $M_{UG} = 200$	33	33.248	167	16	1	0	61/39	201	111
6f	B≻A	switch@stagnation, $M_{UG} = 100$	33	33.249	167	16	1	0	62/38	191	106
6g	B≻A	switch@stagnation, $M_{UG} = 50$	33	33.2505	165	18	1	0	49/51	336	203
6h[b]	A≻B	switch@stagnation, $M_{UG} = 200$	–	–	–	–	–	–	49/51	>5000	>7965
6i[b]	A≻B	switch@stagnation, $M_{UG} = 100$	–	–	–	–	–	–	52/48	>5000	>8424
6j[b]	A≻B	switch@stagnation, $M_{UG} = 50$	–	–	–	–	–	–	53/47	>5000	>10311
6k	A≻B	switch@stagnation,[c] $M_{UG} = 50$	33	33.2465	168	15	1	0	96/4	3127	3179
6l	A≻B	switch@stagnation,[c] $M_{UG} = 10$	33	33.247	167	16	1	0	94/6	1163	844
6m	A≻B	switch@stagnation,[c] $M_{UG} = 5$	33	33.2485	167	16	1	0	94/6	931	668

[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.
[b] Unfinished within the time limit.
[c] Switch@stagnation for once only.


(i) 6i: A≻B, switch@stagnation, $M_{UG} = 100$


(j) 6j: A≻B, switch@stagnation, $M_{UG} = 50$


(k) 6n: A≻B, switch@stagnation once only, $M_{UG} = 50$


(l) 6l: A≻B, switch@stagnation once only, $M_{UG} = 10$

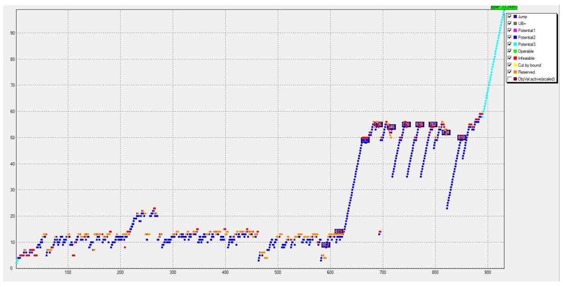

(m) 6m: A≻B, switch@stagnation once only, $M_{UG} = 5$

Figure 6.25: Node-depth graphs from experiments GB-334R-6

6.3.4.7 GB-334R-7: Banned location branching parameter ρ_B

Table 6.18 shows the results of experiment GB-334-7 by varying the banned location parameter ρ_B. At a banned location for coupling/decoupling, if multiple arcs are identified to be in use at an LP optimal solution, then for an arc having a weight proportion larger than ρ_B among all the flowed arcs at that node, a branch will be created retaining only this arc. Surprisingly, for instance GB-334R with the given parameter setting, considering almost all arcs with non-negative flows ("flowed" arcs) at banned locations ($\rho_B = 0.0001$) yielded the best result. As long as sufficient "flowed" arcs were used in forming branches, the solvers could find optimal operable solutions very efficiently, as shown in 7x–7e. On the other hand, only considering an inadequate subset of the flowed arcs might either give unfinished or inoperable solutions as shown in 7f–7i.

The relevant figures for experiments GB-334R-7 are given in Figures A.4–A.6 in Ap-

Table 6.18: Results on experiment GB-334R-7

solver	ρ_B	unit#	oper-obj[a]	Fit	OP	UP	ECS#	rule%(B/A)	BB#	time
7x	0.0001	33	33.248	167	16	1	0	61/39	203	111
7a	0.01	33	33.248	167	16	1	0	61/39	201	133
7b	0.02	33	33.248	167	16	1	0	60/40	200	130
7c	0.05	33	33.2475	167	16	1	0	84/16	540	327
7d	0.07	33	33.2475	167	16	1	0	81/19	465	273
7e	0.1	33	33.248	166	17	1	0	86/14	276	178
7f[b]	0.2	–	–	–	–	–	–	100/0	>3300	>3231
7g[c]	0.3	–	–	–	–	–	–	100/0	181	82
7h[c]	0.4	–	–	–	–	–	–	100/0	130	66
7i[c]	0.5	–	–	–	–	–	–	100/0	299	101

[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.
[b] Unfinished within the time limit.
[c] Problem inoperable.

pendix A. Note that when $\rho_B = 0.3, 0.4, 0.5$ that there were already many flowed arcs being ignored in banned location branching, the trees' structure could be still quite complex—nodes with more than 2 branches were commonly seen and yet these were not sufficient. Figure A.6(g) shows the branch-and-price process was stuck at the nodes with bad values (green coloured), probably because the nodes with more promising values had not been given the chance to appear due to the radical branching threshold $\rho_B = 0.2$. From Figure A.4(g), one can see many estimated upper bound updates (the grey boxes) at the early stages and the frequent divings and jumps at later stages. When this threshold gets even more radical, as in (h), (i) and (j), the tree could only provide nodes with very poor values (red coloured) at deep levels, thus the infeasibility was declared very quickly. From Figure A.4(h)(i)(j), one can see several rounds of node reservation and estimated upper bound updates, until the last round where all reserved nodes were exploited and the search was declared a failure.

It can be concluded that for instance GB-334R, the banned location parameter ρ_B should be set a small value like 0.0001 to let the branch-and-price explore all possibilities in using flowed arcs at a node in $N_B^{\pm}$. Setting a large value to ρ_B may make the problem unfinished or inoperable.

6.3.4.8 GB-334-8: Jump gap G

Two groups of experiments were designed with respect to the jump gap G. The first group GB-334R-8i used a smaller initial estimated upper bound $\varDelta^0 = 0.5$ while the second group GB-334R-8ii used a larger one $\varDelta^0 = 1$. The jump gap G has been varied from 2 to some sufficiently larger numbers within 100. An experiment where the jump action was disabled was also conducted for comparison.

Table 6.19 shows the results on experiments GB-334R-8i. When the initial estimated upper bound $\varDelta^0$ was set smaller to 0.5, the solvers could solve all instances conveniently with small numbers of BB tree nodes and short time. The jump gap G had almost no influence with respect to the solvers' performances. This was probably because the small estimated upper bound had effectively narrowed down the search space, making the subsequent BB tree search much easier.

Table 6.20 shows the results on experiment GB-334R-8ii. When the initial estimated upper bound Δ^0 was set a larger value as 1, such that the difficulty for BB tree search had been significantly increased due to the much widened search space, the influence of jump gap G had become noticeable. From Table 6.20, the general pattern is that the solvers' performance became worse (in number of BB nodes and solution time) as the jump gap was set larger, and when the jump action was disabled, the solver could only find sub-optimal operable solutions within the 10000 tree nodes limit. When the jump gap was relatively small, as in the range between 2–10, the solver's performance was good and stable. Moreover, the overall performance from GB-334R-8ii with $\Delta^0 = 1$ was much worse than GB-334R-8i with $\Delta^0 = 0.5$.

Figure 6.26 shows the node-depth graphs of experiments GB-334R-8ii, where any solvers with the jump action enabled could perform many dives. The solvers with a smaller jump gap found an optimal operable solution very quickly with a small number of nodes. On the other hand, when jump was disabled, the solver using a pure depth-first strategy did not give too many dives (only 6) and could only find three sub-optimal solutions all with a fleet size of 34 (the green boxes in (8t2)). Note that before the first operable solution was found, many nodes were reserved as marked in orange, while after the first operable solution, many nodes were truly cut-off by bound with the yellow marks.

Figures A.7 and A.8 in Appendix A show the BB tree without/with the node values of experiments GB-334R-8ii. The trees' structures turned out to be more complex as the jump gap increased. The BB trees with node values show that the frequent jumps could effectively lead the search to nodes with good values, as indicated by the blue colours, and there were more nodes with undesirable values in the trees with larger jump gaps. As for the solver whose jump action was disabled (8t2), the entire tree was almost covered with nodes with less desirable values.

It seems at least for instance GB-334R, enabling frequent jumps is a good idea. This conclusion is also very likely to be applicable for many other occasions. In fact one may consider to *always* allow jumping at every tree node, which will give a strategy as a combination of best-first and depth-first (if the jump action is set to be jumping to the node with the smallest value). In such a case best-first will be always performed, except when a dive is detected such that the search has to follow the diving action rather than seeking for the next best node. The strategy in establishing an estimated upper bound $\Delta^k, k = 1, 2, \ldots$ is also proved to be effective for instance GB-334R, especially in the fact that an appropriately smaller initial value Δ^0 can be significantly more advantageous than a larger one.

Table 6.19: Results on experiment GB-334R-8i, $\varDelta^0 = 0.5$

solver	G	unit#	oper-obj[a]	Fit	OP	UP	ECS#	rule%(B/A)	BB#	time
8a0	2	33	33.2455	168	15	1	0	71/29	227	139
8aa	3	33	33.2455	168	15	1	0	71/29	227	153
8b	5	33	33.2495	167	16	1	0	55/45	215	158
8c	7	33	33.2495	167	16	1	0	56/44	225	163
8d	10	33	33.2495	167	16	1	0	53/47	214	162
8e	15	33	33.2545	161	22	1	0	84/16	155	109
8f	20	33	33.248	167	16	1	0	61/39	201	106
8g	25	33	33.248	167	16	1	0	61/39	201	127
8h	30	33	33.248	167	16	1	0	64/36	217	149
8i	35	33	33.248	167	16	1	0	64/36	217	131
8j	40	33	33.248	167	16	1	0	64/36	217	115
8k	45	33	33.2545	161	22	1	0	86/14	187	113
8l	50	33	33.2542	161	22	1	0	87/13	208	138
8m	60	33	33.256	160	13	1	0	85/15	233	148
8n[b]	70	33	33.248	167	16	1	0	64/36	217	131
8o[b]	80	33	33.248	167	16	1	0	64/36	217	131
8s[b]	dis.jump	33	33.248	167	16	1	0	64/36	217	131

[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.

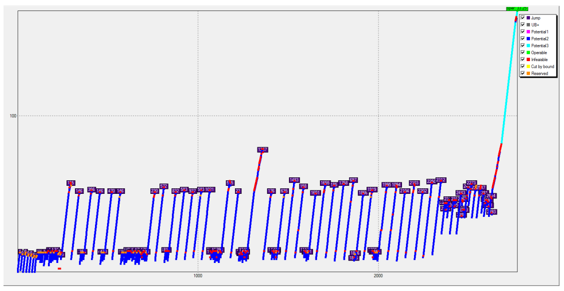
[b] The initialisation stages were not finished in experiments 8n and 8o such that no jump could ever be triggered there. Therefore the results of 8n–8s are identical.

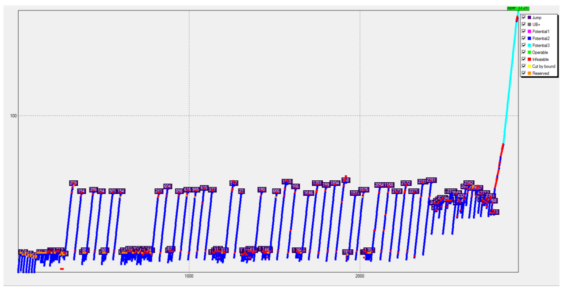
Table 6.20: Results on experiment GB-334R-8ii, $\varDelta^0 = 1$

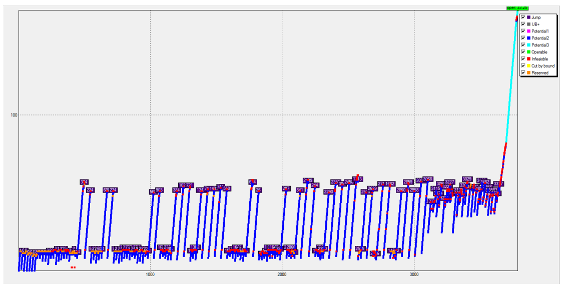
solver	G	unit#	oper-obj[a]	Fit	OP	UP	ECS#	rule%(B/A)	BB#	time
8a2	2	33	33.25	165	18	1	0	97/3	2770	2601
8i2	3	33	33.25	165	18	1	0	97/3	2772	2614
8j2	4	33	33.25	165	18	1	0	97/3	2798	2657
8k2	5	33	33.25	165	18	1	0	97/3	2801	2624
8l2	6	33	33.25	162	18	1	0	97/3	2814	2646
8m2	7	33	33.25	165	18	1	0	97/3	2772	2613
8n2	8	33	33.25	165	18	1	0	97/3	2811	2677
8o2	9	33	33.25	165	18	1	0	97/3	2870	2776
8b2	10	33	33.25	165	18	1	0	97/3	2925	2773
8c2	20	33	33.25	165	18	1	0	97/3	3784	3974
8d2	30	33	33.25	165	18	1	0	95/5	4788	5465
8e2=	40	33	33.2495	166	17	1	0	93/7	5239	6606
8f2=	50	33	33.2485	166	17	1	0	99/1	5249	6105
8g2=	60	33	33.2495	166	17	1	0	89/11	6048	8170
8h2=	70	33	33.2555	160	23	1	0	94/6	6670	10011
8t2[b]	dis.jump	34	34.2485	168	15	1	0	97/3	>10000	>17650

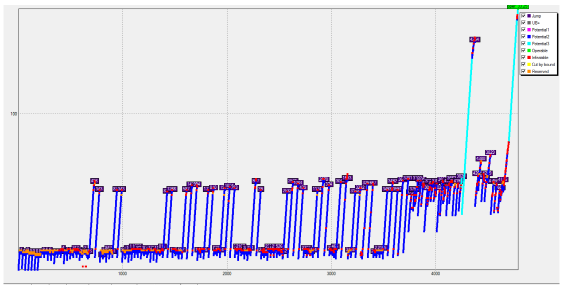
[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.

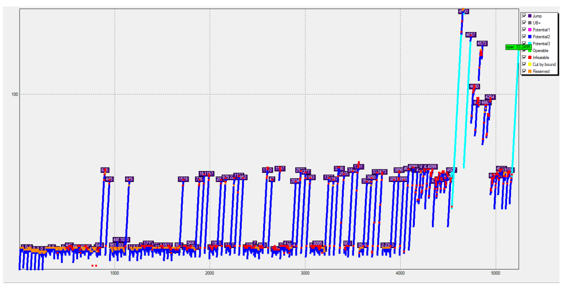
[b] Within 10000 BB tree nodes, experiment 8t2 could only find sub-optimal operable solutions all with fleet sizes of 34.

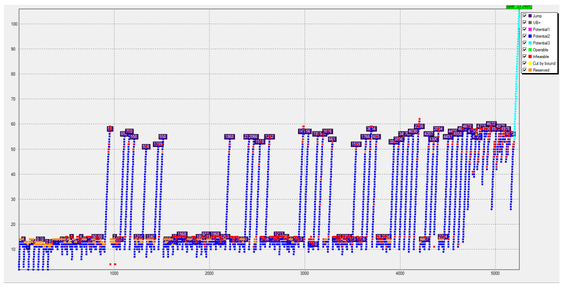

(a) 8a2: $G = 2$

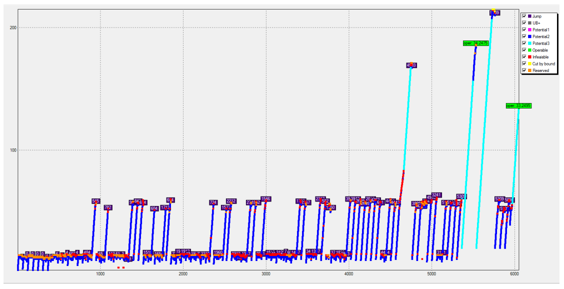

(b) 8b2: $G = 10$

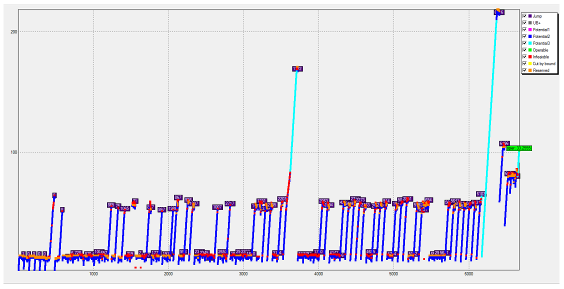

(c) 8c2: $G = 20$


(d) 8d2: $G = 30$


(e) 8e2=: $G = 40$


(f) 8f2=: $G = 50$


(g) 8g2=: $G = 60$


(h) 8h2=: $G = 70$

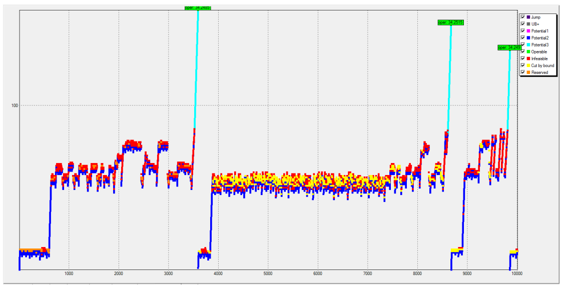

(i) 8t2: jump disabled

Figure 6.26: Node-depth graphs for experiments GB-334R-8ii

Table 6.21: Results on experiment GB-334R-9i, $G = 25$

solver	$\mathit{\Delta}^0$	unit#	oper-obj[a]	Fit	OP	UP	ECS#	rule%(B/A)	BB#	time
9n	0.1	33	33.249	166	17	1	0	73/27	353	192
9a	0.2	33	33.249	166	17	1	0	73/27	353	249
9b	0.3	33	33.249	166	17	1	0	73/27	353	193
9c	0.4	33	33.249	166	17	1	0	73/27	353	181
9d	0.5	33	33.249	166	17	1	0	73/27	353	185
9e	0.6	33	33.249	166	17	1	0	73/27	353	185
9f	0.7	33	33.249	166	17	1	0	73/27	353	189
9g	0.8	33	33.249	166	17	1	0	73/27	353	193
9h	0.9	33	33.249	166	17	1	0	73/27	353	212
9i	1	33	33.249	166	17	1	0	95/5	2371	1822
9j	2	33	33.2485	167	16	1	0	98/2	3975	4067
9k	3	33	33.2485	167	16	1	0	98/2	3972	4139
9l	99999	33	33.2485	167	16	1	0	98/2	3972	4128

[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.

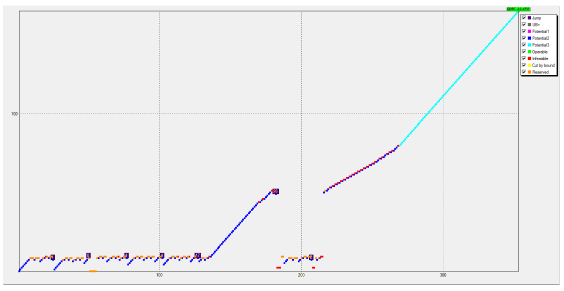
6.3.4.9 GB-334R-9: Initial estimated upper bound increment $\mathit{\Delta}^0$

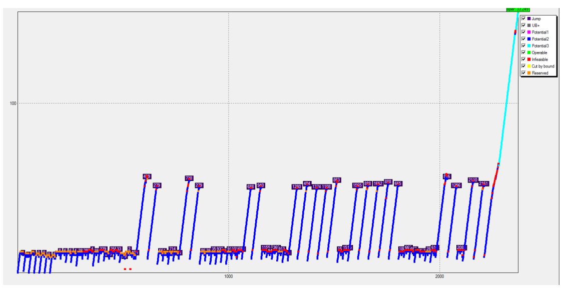
Two groups of experiments were designed with respect to the initial estimated upper bound increment $\mathit{\Delta}^0$. The jump action had been enabled in first group GB-334R-9i with a jump gap $G = 25$ while the second group GB-334R-8ii had its jump action disabled. The initial estimated upper bound increment $\mathit{\Delta}^0$ has been varied from a small number 0.1 to a fairly large number 3. An experiment where this increment was almost disabled ($\mathit{\Delta}^0 = 99999$) had also been added for comparison.

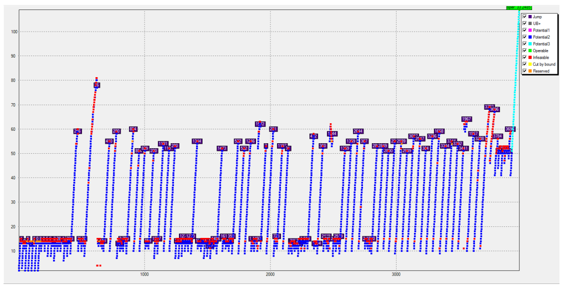
Table 6.21 give the results for experiments GB-334R-9i. The solvers gave identical BB tree and solutions for all instances with $\mathit{\Delta}^0 = 0.1$–0.9 with 353 BB nodes. When $\mathit{\Delta} = 1, 2, 3$, the number of BB nodes and solution time suddenly increased into another magnitude with several thousands of BB nodes. To emphasise the importance of using the jump strategy, even when the upper bound increment was set to 99999 as in solver 9l, the optimal operable solution could still be found. The above results show that for instance GB-334R, setting a smaller, reasonable initial estimated upper bound would be helpful in finding optimal operable solutions efficiently. Note that theoretically it is even possible to set an estimated upper bound that is smaller than the optimal operable solution's objective value, because upper bounds can be updated with the node reservation strategy when a previous one fails.

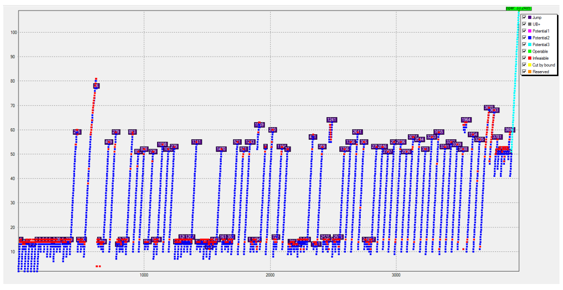
Figure 6.27 shows the node-depth graphs and Figures A.9–A.10 in Appendix A show the BB trees regarding experiments GB-334R-9i. Notably even when the estimated upper bound was very large, frequent jumps in conjunction with dives could always help the search process to find an optimal operable solution, and when $\mathit{\Delta}^0$ was very small, the tree's structure was much simpler.

Table 6.22 gives the results for experiments GB-334R-9ii where the jump option was disabled. In this condition, the solvers were still capable of finding optimal operable solutions quickly as long as $\mathit{\Delta}^0$ was small. However, when $\mathit{\Delta}^0 \geq 1$, the solvers' performance


(a) 9n: $\varDelta^0 = 0.1$


(b) 9i: $\varDelta^0 = 1$


(c) 9j: $\varDelta^0 = 2$


(d) 9k: $\varDelta^0 = 3$

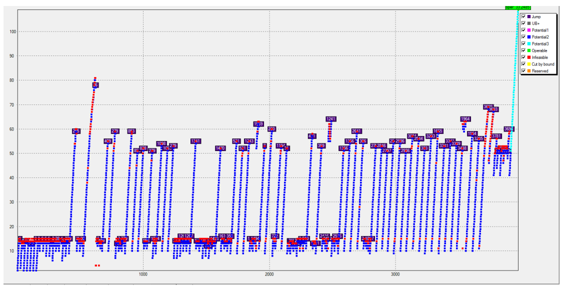

(e) 9l: $\varDelta^0 = 99999$

Figure 6.27: Node-depth graphs for GB-334R-9i, $G = 25$

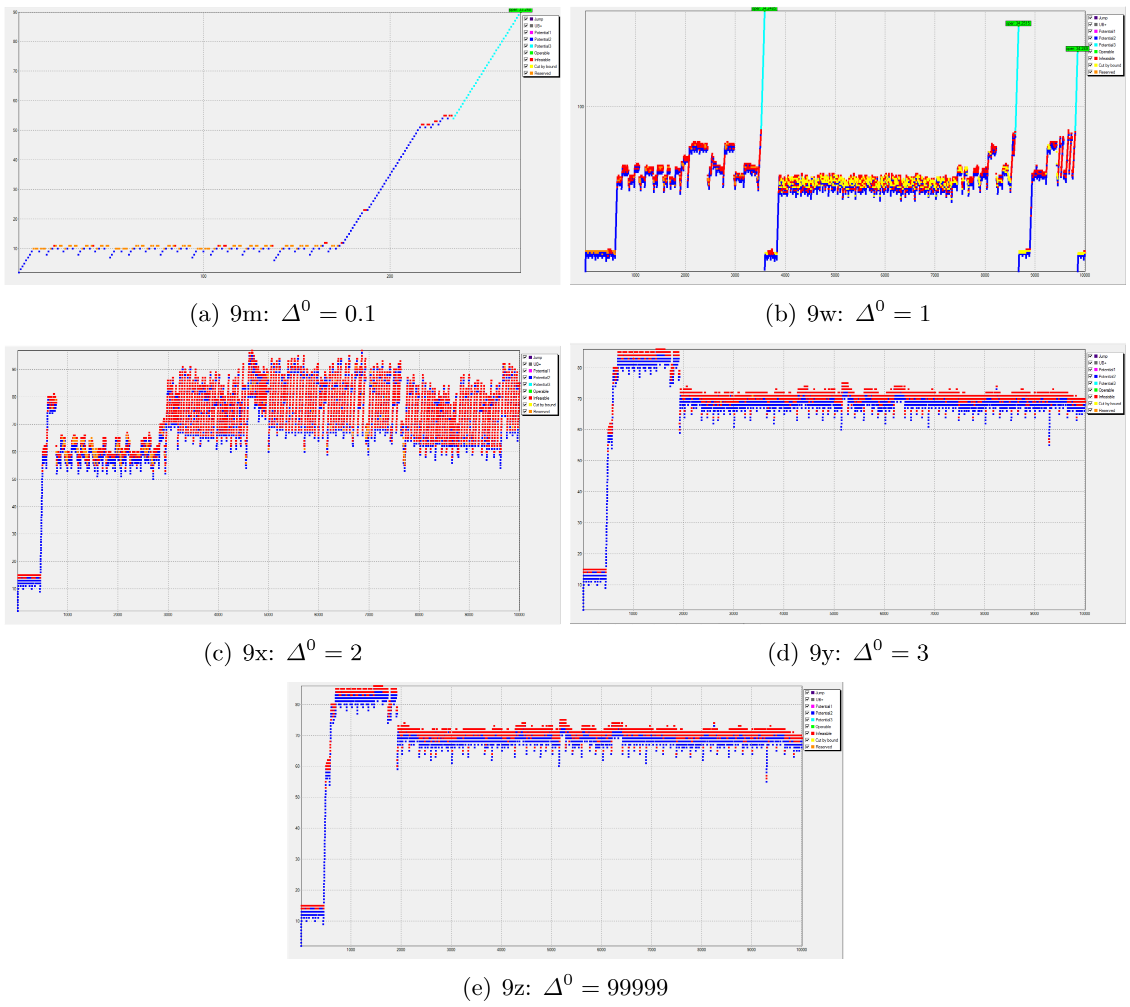
Table 6.22: Results on experiment GB-334R-9ii, jump disabled

solver	$\varDelta^0$	unit#	oper-obj[a]	Fit	OP	UP	ECS#	rule%(B/A)	BB#	time
9m	0.1	33	33.248	166	17	1	0	0/85/15	270	171
9o	0.2	33	33.248	166	17	1	0	0/85/15	270	174
9p	0.3	33	33.248	166	17	1	0	0/85/15	270	154
9q	0.4	33	33.248	166	17	1	0	0/85/15	270	186
9r	0.5	33	33.248	166	17	1	0	0/85/15	270	156
9s	0.6	33	33.248	166	17	1	0	0/85/15	270	157
9t	0.7	33	33.248	166	17	1	0	0/85/15	270	160
9u	0.8	33	33.248	166	17	1	0	0/85/15	270	165
9v	0.9	33	33.248	166	17	1	0	0/85/15	270	196
9w[b]	1	34	34.2485	168	15	1	0	0/97/3	10000	18027
9x[c]	2	–	–	–	–	–	–	0/100/0	>10000	>9049
9y[c]	3	–	–	–	–	–	–	0/100/0	>10000	>17688
9z[c]	99999	–	–	–	–	–	–	0/100/0	>10000	>17641

[a] root-obj = 33.2379 and root-LB = 33.188 for all instances.
[b] Sub-optimal solutions found within 10000 BB tree nodes.
[c] Unfinished within the time limit.

suddenly became extremely poor, most of which could not find any operable solution at all within 10000 tree nodes. This gives a sharp contrast compared with the first group in Figure 6.27, and again shows the importance of using the jump strategy—such jumps can effectively trigger many dives basically only from which operable solutions can be found. Moreover, it shows how the search process could benefit from setting a smaller initial estimated upper bound.

Figure 6.28 shows the node-depth graphs and Figures A.11–A.12 in Appendix A show the BB trees regarding to experiments GB-334R-9i. From the node-depth graphs, one can see without frequent jumps, diving could hardly be triggered automatically and the search would very likely to be trapped in the middle levels of the tree without any useful findings. The BB trees in Figure A.10 show that without restricting the search scope to the nodes with more promising values, the BB tree had been searching nodes with less desirable objective values until reaching the maximum node limit, and no jump/dive was available to change this situation.


(a) 9m: $\varDelta^0 = 0.1$

(b) 9w: $\varDelta^0 = 1$

(c) 9x: $\varDelta^0 = 2$

(d) 9y: $\varDelta^0 = 3$

(e) 9z: $\varDelta^0 = 99999$

Figure 6.28: Node-depth graphs for GB-334R-9ii, jump disabled

Experiments GB-334R-9i and GB-334R-9ii show that employing sophisticated node selection methods in using frequent jumps and appropriate estimated upper bounds is very important in ensuring the problem instance to be solved efficiently. It is very likely that this conclusion could also be applicable to many other problem instances sharing similar features as GB-334-R.

6.4 Ideal demand estimation

As discussed in § 3.1.4, specifying the passenger demand is tricky and even the manual schedulers may not have an accurate estimation of all the trains' demands due to the complexity of manual scheduling and the lack of documentation. Therefore, over-provision and under-provision are commonly seen in the solutions provided by the manual schedulers by comparing them with the PAX surveys. In this section, we give some insights into this problem with an example for over-provision from our experiments on the ScotRail datasets.

6.4.1 Over-provided trains: c334 December 2011

A series of experiments regarding the over-provided trains from the 2011 December manual train unit diagrams on type c334 was carried out, after the passenger counts at the same period were available to us. Instead of using the PAX# directly as the demand input, as has been done for the May 2014 timetable, a method referred to as "OP-X/UP-X" was used which focused on the over-provided trains from the manual diagrams in order to make an estimation of the "ideal" demands. The two approaches (using the solver to schedule directly based on the PAX# and applying the OP-X/UP-X indirectly) each has its own merits—the former is more suitable for the cases where sufficient units in the fleet are available and/or there is no special requirement on retaining the original pattern of schedules as much as possible; the latter on the other hand is more suitable for the cases where satisfying all trains' PAX# will result in a fleet size that is greater than the current inventory and/or the route cannot endure too much change in the schedule's pattern.

Among the 156 trains served by c334 units in the December 2011 timetable, 64 of them are over-provided (OP). It should be mentioned that some of the OP trains had strengthened capacities due to the purpose of redistributing unit resources across the network (like an empty-running movement but coupled to an existing passenger train) to minimise the fleet size. For such trains, even reducing the target demand from the manually scheduled to the PAX# would not change the resulting capacity in the solver's solutions. On the other hand, the capacity strengthening over some other trains were not contributing to minimising the fleet size hence they were truly over-provided. Reducing the target demand of them to the PAX# may lead to a reduction in the fleet size.

The above assumption had been verified. One test has shown that setting the demand input as the manually diagrammed capacities would give a fleet size the same as the

Table 6.23: Results on experiment OP-X

instance	fleet size	time (sec)	ECS#	instance	fleet size	time (sec)	ECS#
OP64	29	56	1	OP46	32	66	2
OP63	29	59	1	OP45	32	75	2
OP62	30	145	0	OP44	33	88	2
OP61	30	60	1	OP43	33	90	2
OP60	30	51	1	OP42	33	59	2
OP59	30	71	1	OP41	33	111	2
OP58	31	71	1	OP40	33	63	2
OP57	31	72	1	OP39	33	72	2
OP56	31	65	1	OP38	33	69	2
OP55	31	75	1	OP37	33	73	2
OP54	31	67	1	OP36	33	75	2
OP53	31	62	1	OP35	33	76	2
OP52	31	56	1	OP34	33	83	2
OP51	31	74	1	OP33	33	88	2
OP50	32	78	1	OP32	33	60	2
OP49	32	66	2	⋮[a]	⋮	⋮	⋮
OP48	32	104	2	OP0	33	72	2
OP47	32	63	2				

[a] The same fleet size and ECS# values from OP33 to OP0.

manual solution (33). On the other hand, setting the demand input as the PAX# for all trains would give a fleet size of only 29 units. However, there may be some reasons why the practitioner may not be totally satisfied with serving the 156 trains with 29 units, e.g. they may want some better service quality provision; the PAX# may not be the same in the following year; the route cannot be subject to a drastic change in the service pattern. Therefore, the ideal demand the practitioner actually wants lies somewhat in the solutions with a fleet size between 29 and 33 units. The solver can compute the 156 train instance very efficiently within 1–2 minutes. This allows the practitioner to use the solver to do what-if analysis many times to determine the best plan in their mind.

To aid the manual scheduler to perform the what-if analysis, a series of tests were conducted. First the 64 OP trains were sorted descendingly into a so-called OP-X list according to their PAX#. We will omit the detailed list only mentioning that the train at the top had a PAX# of 178 and the train at the end had a PAX# of 10. One unit of c334 has a capacity of 183 seats. Then the top members of the OP-X list (with the current largest PAX#) were removed one by one from the list and only the remaining $X-1$ trains in the list would have their demands reduced. This process varied from OP-64 (all the 64 trains to reduce their demand) to OP-0 (none of them to reduce) and the results are given in Table 6.23.

As expected, the fleet size gradually increased to 33 units as the instances varied from OP64 to OP0 and the number of empty-running movements increased from 1 to 2 (OP62 was an exception and was a sub-optimal solution). The practitioners found the above results helpful in estimating the minimum fleet size with respect to different demand level requirements.

6.5 Solution quality analysis

§ 3.2.4 has discussed the issue of preferences regarding to solution qualities. Here some examples from the computational experiments on this particular topic will be given.

6.5.1 Desired and undesired gaps

The desirable long gaps during the daytime have been given a lower weight compared with an ordinary arc since the very early stage of this research and the solver will most often give diagrams containing sufficient long gaps for maintenance. On the other hand, the problem about undesired gaps in the evening or at night had not occurred until the recent close interactions with the railway practitioners. Figure 6.29 gives a Gantt chart of the result from one experiment on the GB-334-entire instance where 72 units are used. In such a Gantt chart, the rows are the train units (diagrams) and the horizontal scale shows the time horizon. Within each diagram, a blue bar shows a passenger train and a red bar shows an empty-running train. There are quite a lot of evening/night gaps in Figure 6.29. As a comparison, the Gantt chart from the manual results of the same set of trains are given in Figure 6.30. Note how the midday long gaps for maintenance are preserved in the manual diagrams and how each unit ends its daily workload mostly without an evening/night gap. Therefore, in a new round of experiments, connection-arcs containing potential evening/night gaps were identified and were given slightly higher weights than ordinary arcs. Defining such arcs is often tricky and problem-specific where the practitioners' experience can give a lot of help. Figure 6.31 gives the Gantt chart of a run after the weights of undesired gaps were changed as described above. One can see that those evening/night gaps have been mostly removed in the new results.

6.5.2 Short diagrams

In many solutions given by the solver, the so-called "short diagrams"—diagrams with very few trains can be found. For instance, the sixth row in Figure 6.29 and the tenth row from the bottom in Figure 6.31 have only one train in each diagram. Those short diagrams can even be found in some manual diagrams, e.g. the eighteenth row from the bottom in Figure 6.30. Regarding to those short diagrams, § 3.2.4 has already given some insights into this topic. The dilemma here is: having a unit to only serve one train seems to be not a perfect plan but it is also a waste of resource to let this unit to serve other trains that have already been adequately satisfied by the current schedule. This issue needs more research in the future with the cooperation from railway practitioners. Several possible remedies are discussed here.

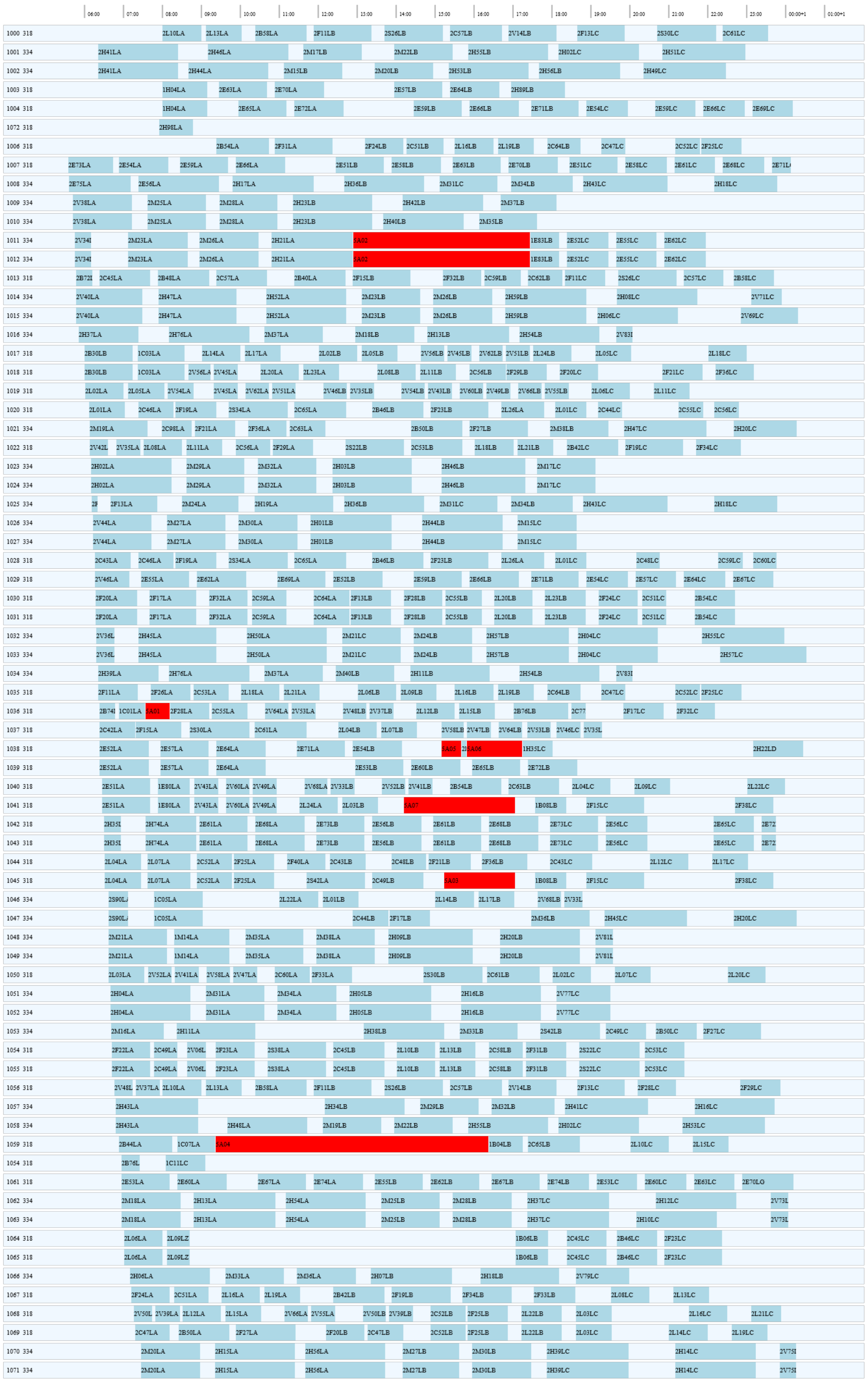


Figure 6.29: Diagrams from solver's scheduling with undesired evening/night gaps

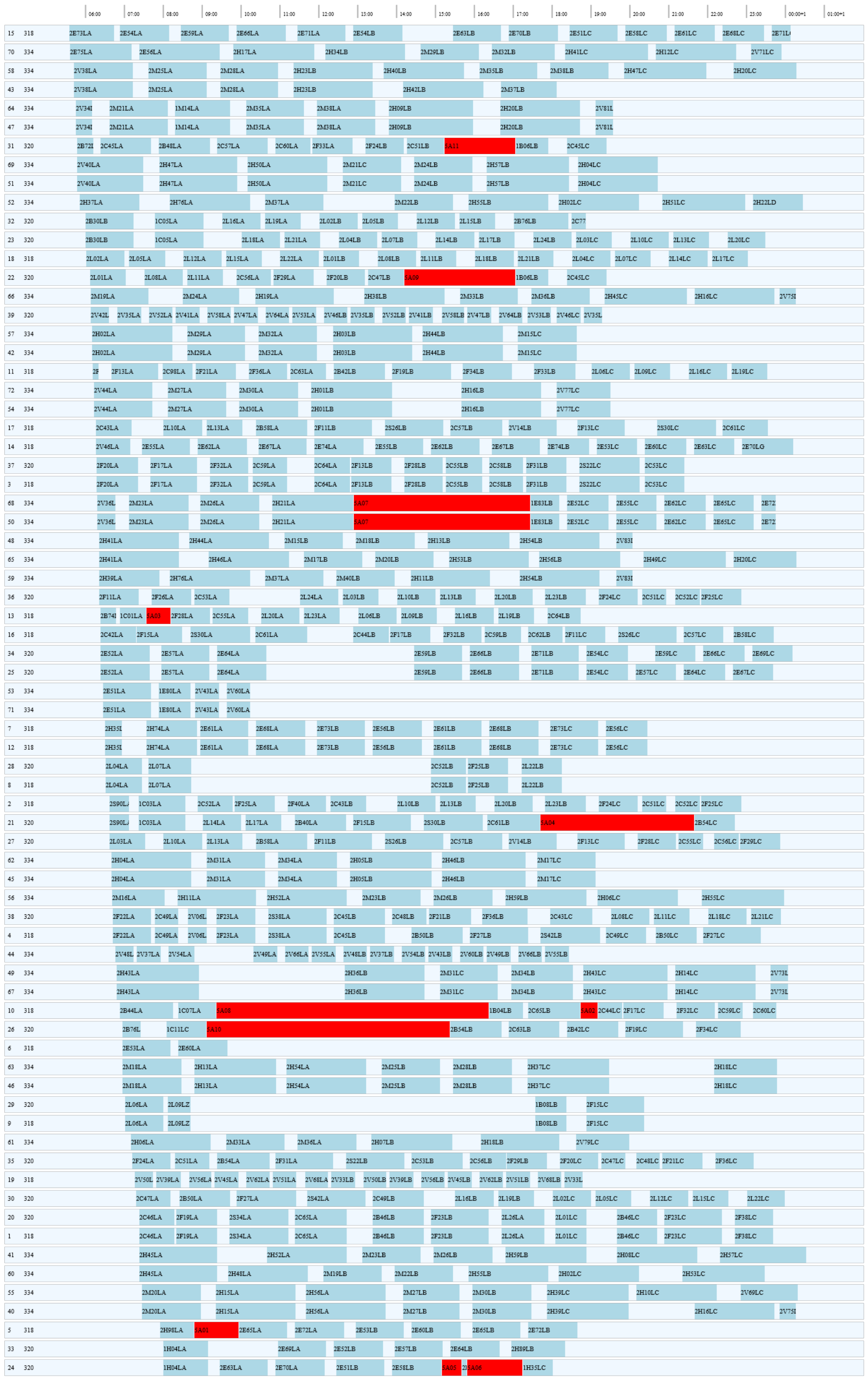


Figure 6.30: Diagrams from manual scheduling

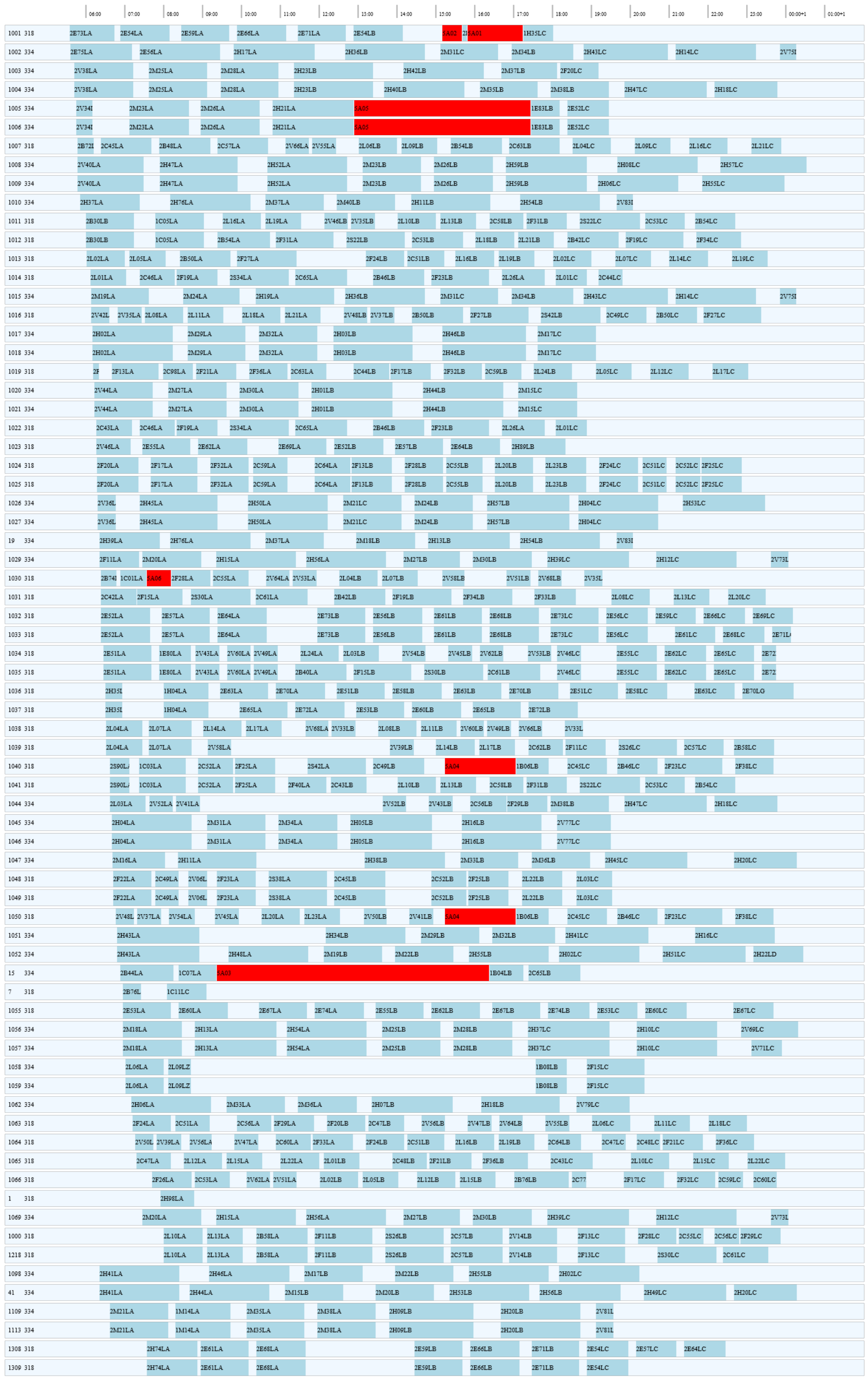


Figure 6.31: Diagrams from solver's scheduling with undesired evening/night gaps removed

6.5.2.1 Two-level demands

For each train, it is possible to set two demands with it such that there is a hard demand that must be satisfied and a soft demand that will be satisfied as much as possible. Generally the hard demand should be lower than or be the same as the soft demand, and it can be decided directly from the passenger count survey (PAX#). On the other hand, it may require more data analysis and decision-making for deciding the soft demands. A simple way would be to add a constant value to the PAX#, e.g. 30. Taking unit type c334 for an example, one unit of it (with 183 seats) will satisfy a train i with a PAX# of 20 passengers and another train j with a PAX# of 180 passengers. Although the two trains are all "satisfied" by one unit of c334, one may expect to have 2 units for j if possible due to its higher PAX# while it is not necessary to do the same for i. Then for train j, its soft demand would be 210, requiring two of c334 and for train i, its soft demand would be 50, not requiring an extra unit.

The hard demands will be transformed as constraints in the ILP model as before while the soft constraints can be put in the objective with appropriate weights. By doing so, one may expect the short diagrams can cover more trains that are at least worth being covered due to their higher PAX# such that unit resources will not be wasted on trains with very low PAX# such as 20 passengers.

The trade-off in having the extra soft demands has to be carefully considered, as realising them in the objective function may bring much burden in solving the new model. Define r'_j as the soft demands for train j, then a term

$$\sum_{j\in N}\left|\sum_{k\in K,p\in P_j^k} q_k x_p - r'_j\right| \tag{6.1}$$

will be added in the objective with some weight to make the capacity provision achieved at train j (which equals to $\sum_{k\in K,p\in P_j^k} q_k x_p$) as close as possible to r'_j for all $j \in N$. The linearisation on the absolute value in the objective will further require $|N|$ constraints and $2|N|$ decision variables following a similar strategy as in the overnight balance constraints discussed in § 3.6.2.

6.5.2.2 Change the timetable

Regarding the short diagrams, the practitioners from ScotRail have made some comments that there may be good evidence to indicate that in fact the single trains in those short diagrams should be removed from the timetable, if possible. This shows that, apart from its default ability in unit scheduling based on a given timetable as input, the solver may have some extra potential in helping the practitioners to do some decision-making process given their valuable experiences and understanding about the problem and the solver's ability to schedule relatively quickly many times subject to different inputs and parameters to provide many what-if solutions such that the practitioners can perform many analyses that were not possible before.

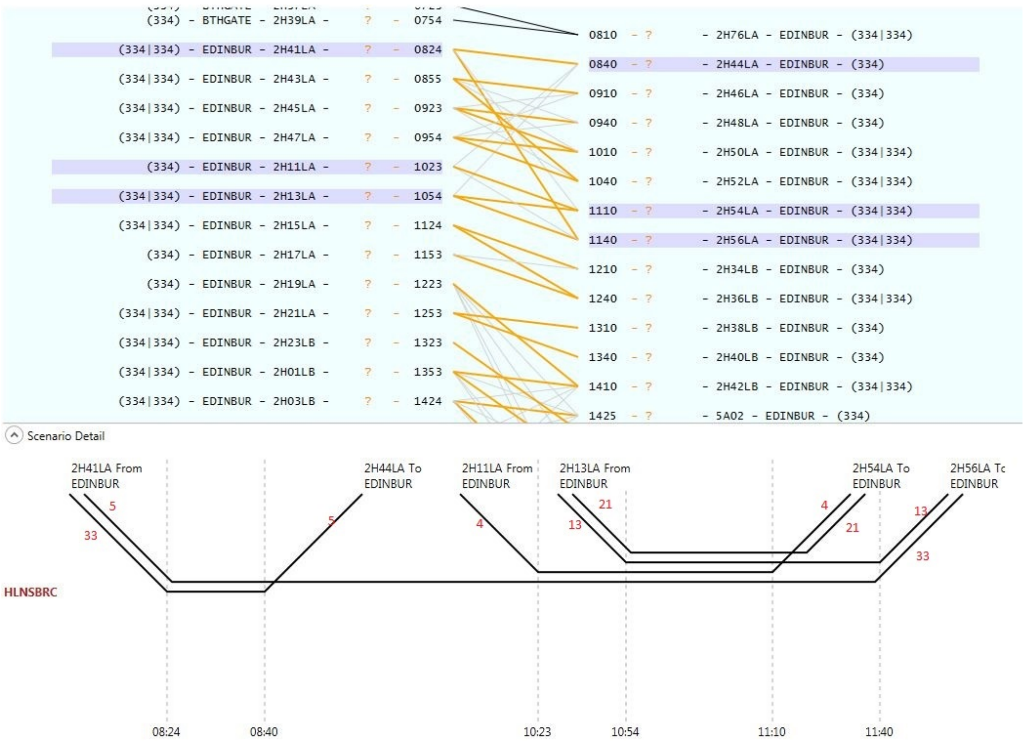
6.6 Post-processing

As mentioned in § 3.1.7, the network-level framework alone without consideration of station-level details may often yield the unit blockage problem. Moreover, when a reduced formulation (e.g. (AF_3), (PF_3)) is used, the requirements of the time allowance for coupling/decoupling and the elimination of excessive/redundant coupling/decoupling have to be realised locally at each station. This shows the need for a post-processing phase taking the results of the integer multicommodity flow models as its input and giving output such that no unit blockage, time allowance violation or excessive coupling/decoupling occurs. As already explained at the beginning of Chapter 3, the rationale of such a post-processing phase is partly from the observation on the manual process that once the overall network flow of units has been fixed, the station-level logistical details are often easily manually solvable. Discussions with some manual schedulers from some rail companies have also confirmed this point. It will be shown that the above rationale will be further reaffirmed by the experiment using the interactive software TRACS-RS to do the post-processing tasks.

The post-processing problem can be formulated as an optimisation problem such that it would be realised automatically. The designing, modelling and experimenting of such an automatic post-processing system will be left to future research. Alternatively, to achieve the same or similar effect on post-processing, we have used an analogous approach. TRACS-RS [73] produced by Tracsis plc, a piece of interactive software for facilitating human schedulers' manual process by visualizing and resolving blockage and shunting plans at the station level, has been used for post-processing at this stage of research. Below two experiments in using TRACS-RS for post-processing the schedules from the c334 units will be presented.

6.6.1 Experiments on c334 units

Results of the network-level ILP solving two instances with 158 trains covered by c334 units from ScotRail's December 2011 and May 2014 timetables were uploaded into the TRACS-RS system. The former took both the "manually diagrammed" and the "required" passenger demands in the network-level ILP while the latter took the passenger count PAX#. Large numbers of unit blockage were observed at many stations. Moreover, excessive/redundant coupling/decoupling operations were found especially at big stations such as Helensburgh Central. For example, Figure 6.32 shows some cases combined with unit blockage and excessive coupling/decoupling at Helensburgh Central mainly with trains from/to Edinburgh Waverley. The left side shows the arrival trains at this station



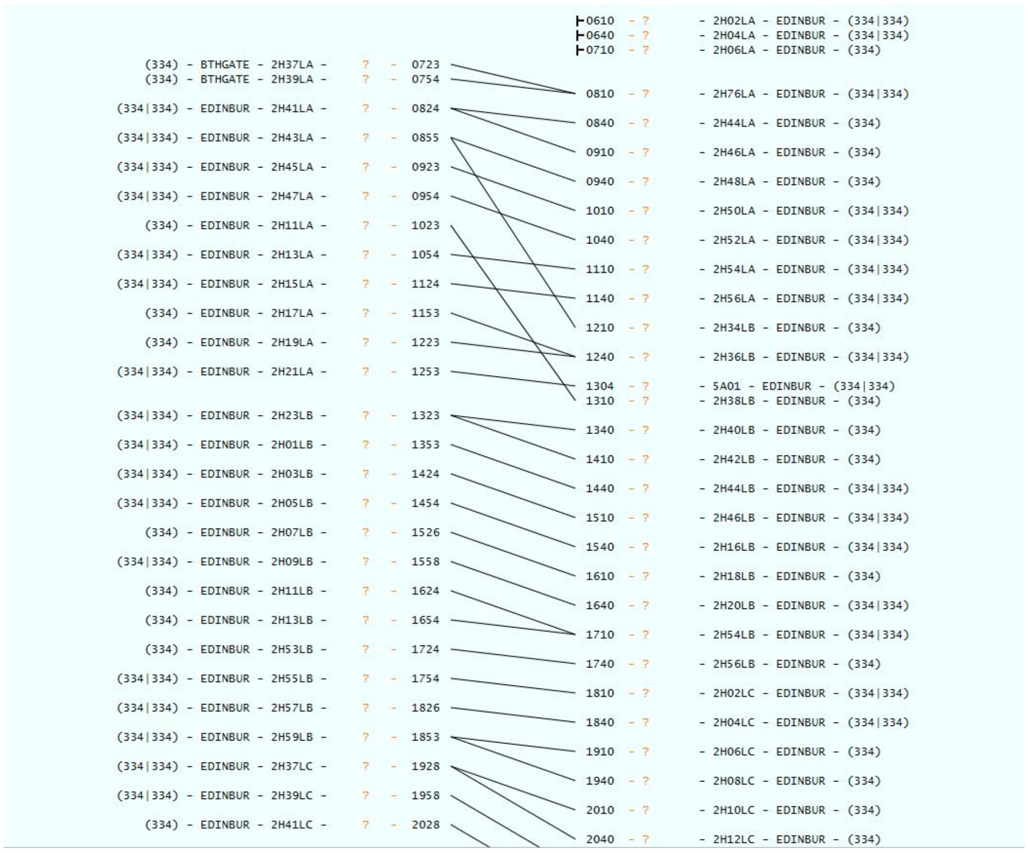
```
        (334) - BTHGATE - 2H39LA -  ?  -  0754
                                                 0810  - ?    - 2H76LA - EDINBUR - (334|334)
    (334|334) - EDINBUR - 2H41LA -  ?  -  0824
                                                 0840  - ?    - 2H44LA - EDINBUR - (334)
    (334|334) - EDINBUR - 2H43LA -  ?  -  0855
                                                 0910  - ?    - 2H46LA - EDINBUR - (334)
    (334|334) - EDINBUR - 2H45LA -  ?  -  0923
                                                 0940  - ?    - 2H48LA - EDINBUR - (334)
    (334|334) - EDINBUR - 2H47LA -  ?  -  0954
                                                 1010  - ?    - 2H50LA - EDINBUR - (334|334)
        (334) - EDINBUR - 2H11LA -  ?  -  1023
                                                 1040  - ?    - 2H52LA - EDINBUR - (334|334)
    (334|334) - EDINBUR - 2H13LA -  ?  -  1054
                                                 1110  - ?    - 2H54LA - EDINBUR - (334|334)
    (334|334) - EDINBUR - 2H15LA -  ?  -  1124
                                                 1140  - ?    - 2H56LA - EDINBUR - (334|334)
        (334) - EDINBUR - 2H17LA -  ?  -  1153
                                                 1210  - ?    - 2H34LB - EDINBUR - (334)
        (334) - EDINBUR - 2H19LA -  ?  -  1223
                                                 1240  - ?    - 2H36LB - EDINBUR - (334|334)
    (334|334) - EDINBUR - 2H21LA -  ?  -  1253
                                                 1310  - ?    - 2H38LB - EDINBUR - (334)
    (334|334) - EDINBUR - 2H23LB -  ?  -  1323
                                                 1340  - ?    - 2H40LB - EDINBUR - (334)
    (334|334) - EDINBUR - 2H01LB -  ?  -  1353
                                                 1410  - ?    - 2H42LB - EDINBUR - (334|334)
    (334|334) - EDINBUR - 2H03LB -  ?  -  1424
                                                 1425  - ?    - 5A02 - EDINBUR - (334)
```

Figure 6.32: Blockage and excessive coupling/decoupling from network flow model results at Helensburgh Central Station (from 07:54 to 14:25)

and the right shows the departure trains. Each link from an arrival train to a departure train corresponds to a block arc in $\mathcal{G}$. The inappropriate coupling/decoupling can be identified in the used links marked in yellow colour. The bottom shows the specific scenario of the purple-shaded trains above. Many blockage instances can be identified. For example, unit 4 comes from train 2H11LA, but it cannot couple with unit 21 to serve train 2H54LA later as unit 13 decoupled from unit 21 is in the middle of unit 4 and unit 21. A better alternative plan would be to simply let the coupled unit 13+21 from 2H13LA serve the later train 2H54LA or 2H56LA both requiring 2 units, rather than the schedule giving excessive coupling/decoupling as in the Figure 6.32.

Although the network-level framework may often provide inoperable raw solutions as illustrated in Figure 6.32, as far as for the ScotRail instances we have processed, fully workable final results could always be easily obtained by some straightforward manual modifications by TRACS-RS on the raw solutions given by the network-level framework. Figure 6.33 gives an example of how the above problem at Helensburgh Central Station has been fully resolved by TRACS-RS. The validity of the new results has been confirmed by experts from ScotRail. In fact all of our post-processing experiments were able to successfully resolve the problems left by the network-level framework ILP, which makes us believe the rationale of the post-processing phase is well-established.

Theoretically the locally based post-processing may increase the number of used units,



(334) - BTHGATE - 2H37LA - ? - 0723
(334) - BTHGATE - 2H39LA - ? - 0754
(334|334) - EDINBUR - 2H41LA - ? - 0824
(334|334) - EDINBUR - 2H43LA - ? - 0855
(334|334) - EDINBUR - 2H45LA - ? - 0923
(334|334) - EDINBUR - 2H47LA - ? - 0954
(334) - EDINBUR - 2H11LA - ? - 1023
(334|334) - EDINBUR - 2H13LA - ? - 1054
(334|334) - EDINBUR - 2H15LA - ? - 1124
(334) - EDINBUR - 2H17LA - ? - 1153
(334) - EDINBUR - 2H19LA - ? - 1223
(334|334) - EDINBUR - 2H21LA - ? - 1253
(334|334) - EDINBUR - 2H23LB - ? - 1323
(334|334) - EDINBUR - 2H01LB - ? - 1353
(334|334) - EDINBUR - 2H03LB - ? - 1424
(334|334) - EDINBUR - 2H05LB - ? - 1454
(334) - EDINBUR - 2H07LB - ? - 1526
(334|334) - EDINBUR - 2H09LB - ? - 1558
(334) - EDINBUR - 2H11LB - ? - 1624
(334) - EDINBUR - 2H13LB - ? - 1654
(334) - EDINBUR - 2H53LB - ? - 1724
(334|334) - EDINBUR - 2H55LB - ? - 1754
(334|334) - EDINBUR - 2H57LB - ? - 1826
(334|334) - EDINBUR - 2H59LB - ? - 1853
(334|334) - EDINBUR - 2H37LC - ? - 1928
(334|334) - EDINBUR - 2H39LC - ? - 1958
(334) - EDINBUR - 2H41LC - ? - 2028
⊢0610 - ? - 2H02LA - EDINBUR - (334|334)
⊢0640 - ? - 2H04LA - EDINBUR - (334|334)
⊢0710 - ? - 2H06LA - EDINBUR - (334)
0810 - ? - 2H76LA - EDINBUR - (334|334)
0840 - ? - 2H44LA - EDINBUR - (334)
0910 - ? - 2H46LA - EDINBUR - (334)
0940 - ? - 2H48LA - EDINBUR - (334)
1010 - ? - 2H50LA - EDINBUR - (334|334)
1040 - ? - 2H52LA - EDINBUR - (334|334)
1110 - ? - 2H54LA - EDINBUR - (334|334)
1140 - ? - 2H56LA - EDINBUR - (334|334)
1210 - ? - 2H34LB - EDINBUR - (334)
1240 - ? - 2H36LB - EDINBUR - (334|334)
1304 - ? - 5A01 - EDINBUR - (334|334)
1310 - ? - 2H38LB - EDINBUR - (334)
1340 - ? - 2H40LB - EDINBUR - (334)
1410 - ? - 2H42LB - EDINBUR - (334)
1440 - ? - 2H44LB - EDINBUR - (334|334)
1510 - ? - 2H46LB - EDINBUR - (334|334)
1540 - ? - 2H16LB - EDINBUR - (334|334)
1610 - ? - 2H18LB - EDINBUR - (334)
1640 - ? - 2H20LB - EDINBUR - (334|334)
1710 - ? - 2H54LB - EDINBUR - (334|334)
1740 - ? - 2H56LB - EDINBUR - (334)
1810 - ? - 2H02LC - EDINBUR - (334|334)
1840 - ? - 2H04LC - EDINBUR - (334|334)
1910 - ? - 2H06LC - EDINBUR - (334)
1940 - ? - 2H08LC - EDINBUR - (334)
2010 - ? - 2H10LC - EDINBUR - (334)
2040 - ? - 2H12LC - EDINBUR - (334)

Figure 6.33: Blockage and excessive coupling/decoupling resolved by post-processing at Helensburgh Central Station (from 07:23 to 17:40)

because more constraints have to be satisfied there. However, in all our current experiments with TRACS-RS, the fleet sizes remain the same as the raw results given by the network-level ILP, except for one instance from the new May 2014 GB area instance where in fact one unit has been saved due to the insertion of 4 new empty-running movements in the post-processing (cf. Table 6.9).

6.7 Conclusions

This chapter presents the experiments using real-world problem instances in the UK rail industry and the relevant results that were all checked by experts from train operators.

At the early stage of the research, problem instances from Southern Railway were tested. Since the ILP solver was still immature, we could only solve problem instances within 500 trains to optimal or sub-optimal solutions. Moreover, since the idea of the two-level framework had not been developed yet, there were significant station blockage problems identified by the practitioners.

The major part of the research has benefited from the collaboration with ScotRail where various sets of experiments on the GA and GB areas of the ScotRail network were tested. The branch-and-price solver was more powerful with the local convex hull strengthening and sophisticated branching strategies. Moreover, the two-level framework had been developed where the interactive software TRACS-RS was used for post-processing.

The first set of experiments consisted of two groups scheduling GA and GB based on the December 2012 and May 2014 timetables respectively. The customised branch-and-price solver successfully solved the above real-world instances to optimality, either giving the same or superior results compared with the manual diagrams and the practitioners from ScotRail were satisfied. There are also several interesting topics to further investigate such as the peak hour provision subject to limited fleet resources in the GA timetable. The second set of experiments is a series of experiments based on GB-334R by varying different parameters in fine-tuning the branch-and-price solver with the purpose of gaining a deeper insight into the solver. These experiments have been very useful in best utilising the solver. Results and analyses of this series of experiments have been reported in detail. The third set of experiments are for ideal demand estimation focusing on the over-provided trains from the December 2011 timetable has been given. The practitioners found the results helpful in estimating the minimum fleet size with respect to different demand levels. The fourth set of experiments relate to the issue of solution qualities, including desired/undesired gaps and short diagrams. Finally some test examples on the station-level post-processing by TRACS-RS were given based on c334 units. The unit blockage and excessive/redundant coupling/decoupling could always be eliminated from all the instances we have dealt with without undermining the solution qualities. This reaffirmed the rationale and feasibility of the two-level framework.

Chapter 7

Local convex hulls

Based on previous work in rolling stock scheduling problems, we generalise a local convex hull method for the corresponding integer multicommodity flow problems, and discuss its feasibility range in high dimensional cases. It will be shown that if certain conditions are met, a local convex hull can be divided into a main hull, an up hull and a down hull. Preliminary research shows that the main hull can only have at most two nonzero facets, and further investigation will be continued to verify this result in the future. The numbers of points in the up and down hull will also be explored mainly on an empirical basis. The above properties of local convex hulls will give a slightly modified QuickHull algorithm ("2-facet" QuickHull) based on the original version in [6]. As for the feasibility in applying this method to rolling stock scheduling, our empirical experiments show that for the problem instances of ScotRail and Southern Railway, even in the most difficult real-world or artificial conditions (e.g. with a space of dimension within 11), the standard QuickHull in [6] can easily compute relevant convex hulls from various instances. For some even more difficult artificial instances that may fall outside the scope of rolling stock scheduling (e.g. with a space of dimension higher than 11), the "2-facet" QuickHull shows some advantages over the standard QuickHull.

The contents of this chapter are supposed to be supplementary directions in this research which may be useful for future researchers in relevant fields to carry out further studies. In particular, the theoretical investigations are preliminary attempts arguably outside the main scope of the thesis and hence are presented for completeness only. The empirical experiments provide useful results on the feasibility range of the local convex hull method within the context of rolling stock unit scheduling.

7.1 Introduction

7.1.1 A special class of multicommodity flow problems

We generalise the scenario found in the train unit scheduling problem to a broader range of integer multicommodity flow problems. Consider an integer multicommodity flow problem defined over a directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $j \in \mathcal{N}$ and $a \in \mathcal{A}$ are the nodes and arcs, and $k \in K$ the commodities to be flowed in integral amounts from origins to destinations that are specific for each k. Each commodity's total amount may be fixed or bounded by a constant b_0^k. Also there are compatibility relations among the commodities and nodes/arcs such that for a certain node/arc not all commodities may be allowed to flow through it. For the cases of nodes, we use $K_j \subseteq K$ to denote the set of commodities that are allowed at node j.

Apart from that, how the commodities should flow is restricted by two kinds of local constraints associated with each node and/or arc. Here we assume that the local constraints are all based on nodes while similar situations on arcs can be derived by analogy. Let $w_k^j \geq 0$ be the flow amount of commodity k passing thorough node j. There are commonly two kinds of local constraints associated with each node in multocommodity flow problems. The first kind is the *demand requirement*, which requires the total "demand" or "profit" achieved at j from used commodities to be at least at a required level, as

$$\sum_{k \in K_j} q_k w_k^j \geq r_j, \quad \forall j \in \mathcal{N}, \tag{7.1}$$

where q_k is the provision or profit contributed by a unit flow amount of commodity k and r_j is the required total demand or profit at j. The second kind is the *bounding restriction*, which caps the total "resource" taken up by used commodities at j to be no more than an upper bound, as

$$\sum_{k \in K_j} v_k w_k^j \leq u_j, \quad \forall j \in \mathcal{N}, \tag{7.2}$$

where v_k is the resource taken up by a unit flow amount of commodity k and u_j is the total resource upper bound at j. We assume, as in many real-world problems, that q_k, r_j, v_k, u_j are integer coefficients.

(7.1) and (7.2) are the most commonly seen constraints in an integer multicommodity flow problem. To satisfy their requirements, it is sufficient to simply insert them into the associated ILP after some variable conversion, e.g. $w_k^j = \sum_{p \in \mathcal{P}_j^k} x_p$ for a path formulation, where $\mathcal{P}_j^k$ is the set of commodity k path passing through j and x_p is the path variable on path p. However, there are disadvantages or incapabilities in directly using (7.1) and (7.2).

(i) *Weak LP relaxation* As often observed, the two kinds of constraints are very likely to yield weak LP-relaxation due to their knapsack nature. The train unit/locomotive scheduling (assignment) problems are typical examples, where, e.g. q_k is the number of seats in a train unit of type k and r_j is the required passenger demand measured in number of passengers for train j.

(ii) *Combination-specific upper bounds* The bounding restriction may be combination-specific, i.e. the upper bound will vary according to different combinations in terms of which commodities are used. The complex upper bound restrictions for coupled train units in the train unit scheduling problem is such an example. If different constraints with different bounds are formed, *disjunctive* relations among these constraints are required. However coexisting constraints in an LP are *conjunctive* thus will not realise the target. One remedy would be to introduce extra binary variables and constraints to represent the use of different combinations. This however may often slow down the solution process.

(iii) *Commodity compatibility* Sometimes there can be compatibility relations among the commodities used at a node where only certain subsets of them can coexist. For instance, when five commodities $k_1, \ldots, k_5$ are allowed at a node, they should only flow it such that either members from k_1 and k_2 coexist or members from k_3, k_4 and k_5 coexist but not otherwise. The creation of train unit families based on the coupling compatibility introduced in previous chapters is such an example.

To deal with the weak LP relaxation problem above, a class of similar methods has been proposed in several papers [3, 15, 70, 84], which happen to be all in railway rolling stock assignment/scheduling problems. Also in dealing with all the three points listed above, a method for directly computing the train convex hulls has been given in § 4.1.2 of this thesis. We will refer to this class of method as the *local convex hull method* and will generalise it in the subsequent sections. This method is not universal for all integer multicommodity flow problems. To successfully apply it, a problem should possess certain features. It is difficult to give a precise and rigorous standard in defining these features. Simply speaking however, as long as the local convex hulls can be explicitly computed within a reasonable time in practice, the problem would be regarded as being suitable. The observation below gives a rough idea on those features.

Observation 7.1. *For most of the nodes $j \in \mathcal{N}$, the number of commodities suitable for node j, i.e. $|K_j|$, should not be large, although the total number of commodities over the network may still be large. For the commodities suitable at node j, the number and values of valid commodity combinations, i.e. $(w_k^j)_{k \in K_j}$, should also be not large.*

These features are commonly seen in a large class of integer multicommodity problems, typically arising in transportation scheduling problems as train unit or locomotive scheduling, where the nodes are train services to be covered by rolling stock, the commodities are different types of rolling stock and w_k^j are the number of rolling stock type k used for train j. Despite the above features, those scheduling problems are generally very difficult to solve by optimisation methods as the number of trains can be from hundreds to several thousands with a very high density of arcs.

7.1.2 Local convex hull method

A generalisation on the train convex hull method proposed in § 4.1.2 from train unit scheduling to a generic integer multicommodity flow problem is given here. For each node j, a commodity combination set W_j is defined as

$$W_j = \left\{ w^j \in \mathbb{Z}_+^{K_j} \middle| \forall w^j : \text{a valid commodity combination for node } j \right\}, \forall j \in \mathcal{N}, \tag{7.3}$$

where $w^j = (w_1^j, \ldots, w_{|K_j|}^j)^T$ is a vector representing a combination. We assume that due to Observation 7.1 the number of combinations are small enough such that W_j can be simply obtained by enumeration. For problem instances with the demand and bounding restrictions exactly given by (7.1) and (7.2), we also have

$$W_j = \left\{ w^j \in \mathbb{Z}_+^{K_j} \middle| \sum_{k \in K_j} q_k w_k^j \geq r_j, \sum_{k \in K_j} v_k w_k^j \leq u_j \right\}, \forall j \in \mathcal{N}. \tag{7.4}$$

However for cases with combination-specific upper bounds the combination set may only be obtained from (7.3) by enumeration. Next for each node the *local convex hull* $\operatorname{conv}(W_j)$ of the above combination set is computed explicitly before the optimisation process, given that the number of points $|W_j|$ is not too large and the dimension $|K_j|$ is appropriately small:

$$\operatorname{conv}(W_j) = \left\{ w^j \in \mathbb{R}_+^{K_j} \middle| H^j w^j \leq h^j \right\}, \forall j \in \mathcal{N}. \tag{7.5}$$

In a path formulation for example, after replacing w-variables by x-variables through $w_k^j = \sum_{p \in \mathcal{P}_j^k} x_p$, it is sufficient to use (7.5) to satisfy the demand and bounding restrictions, as well as the bounding restrictions involving combination-specific requirement. Usually a much tightened LP relaxation will also be obtained compared with solely using (7.1) and (7.2). This pre-processing on convex hull computation is carried out before the ILP optimisation rather than during it.

Within the realm of railway rolling stock scheduling/assignment/circulation, the method of using explicitly computed local convex hulls to strengthen LP relaxation first appears in Schrijver [70] for a train unit scheduling problem, where since at most two commodities are involved, the convex hull computation is done in $\mathbb{R}_+^2$. Ziarati et al. [84] propose a similar method to generate cuts for a locomotive assignment problem. For $|K_j| = 2$ they have established a relationship for their problem instance that the maximum number of nonzero facets of the lower envelope of $\operatorname{conv}(W_j)$ is $2m$ when $\max w_k^j = \frac{m^2+5m}{2}$ and $2m+1$ when $\max w_k^j = \frac{m^2+5m}{2} + 1$, where $m \in \mathbb{N}_+$. They also show that this facet number will not exceed 4 when $|K_j| = 3$ and $\max w_k^j \leq 6$. Cacchiani et al. [15] give a local convex hull method for the train unit assignment problem to tighten the LP relaxation. Taking advantage of the problem's feature that $u_j = 2$, $v_k = 1, \forall k \in K_j, \forall j \in \mathcal{N}$ and based on the implicit combination sets in the form of (7.4), they find the explicit description of the dominants of the local convex hulls and apply this method to real-world instances where $|K_j| = 10$. Also the characteristics of relevant integer polytopes have been studied [14]. In § 4.1.2 of this thesis, local convex hulls are explicitly computed to deal with combination-specific upper bounds and commodity compatibility, as well as to sharpen the LP relaxation.

In all the instances above utilising local convex hulls, it is either the case that the dimension number $|K_j|$ and the point number $|W_j|$ are rather small such that generic convex hull algorithms (e.g. QuickHull [6]) are capable or the problem has special features to allow a generalised description on the relevant dominants (e.g. [15]). However, for the former, the generic convex hull algorithms are not guaranteed to cater for all general cases with higher dimension numbers and a moderate or large number of points, and for the latter, the special features of $u_j = 2, v_k = 1$ are unlikely to exist in the majority of real-world problem instances.

We will give a further exploration on the feasibility of the local convex hull method subject to instances whose combination sets W_j have higher dimensions (e.g. 5–10) and a larger number of points (e.g. hundreds to more than one thousand), as well as present a tailored convex hull computation algorithm based on QuickHull. It is certain that the computational feasibility of this method must have a limit when the number of points and space dimensions are getting large. Therefore empirical experiments will also be conducted to explore the feasibility range of this method within the context of rolling stock scheduling.

7.2 The structure of local convex hulls

In this section we assume that everything is based on the n-dimensional Euclidean space $\mathbb{R}^n$ (and its subset $\mathbb{R}^n_+$), associated with a commodity combination set $W \subset \mathbb{Z}^n_+$ containing $|W|$ finite points representing all possible combinations from n available commodities indexed by $i = 1, \ldots, n$. The node name j will be omitted. Let r be the required demand or profit and let u be the shared resource upper bound for all combinations when W can be also represented by (7.4). A point in $\mathbb{R}^n$ is written as $w = (w_1, \ldots, w_n)^T$. We are interested in explicitly computing $\mathcal{H} = \text{conv}(W) = \{w \in \mathbb{R}^n_+ | Hw \leq h\}$, the convex hull of W.

Among all the combinations in W, consider those where only *one* commodity is used. In $\mathbb{R}^n_+$ the points representing them should lie precisely on the axes each associated with a single commodity and there will be only one nonzero entry in each of these points. A set containing all such points on axes is defined as

$$W' = \{w \in W | w \text{ is on an axis of } \mathbb{R}^n_+\}. \tag{7.6}$$

Moreover, denote $W'_i = \{w \in W' | w \text{ is on axis } i\}, \forall i = 1, \ldots, n$, i.e. the set of points on axis i. We assume that for all commodities $i = 1, \ldots, n$, $W'_i \neq \emptyset$. Taking the TUSP for example, it is uncommon for a train to have an available unit type which can only be coupled with other types but is not allowed to run on its own. For a combination set W that can be represented by (7.4), this assumption means $\lceil \frac{r}{q_i} \rceil < \lfloor \frac{u}{v_i} \rfloor, \forall i = 1, \ldots, n$. When this "single-commodity-presence" assumption cannot be met, it is not possible to construct the main hull to be introduced later in this section and the customised convex hull algorithm to be introduced in § 7.3 will thus be inapplicable. However it is still possible to compute $\mathcal{H}$ using the standard convex hull algorithm, which, as will be shown in the experiments later, is still very efficient for the tested instances.

Now consider a combination set W that satisfies the above "single-commodity-presence" assumption. Within each W'_i, let

$$a_i = \min_{w \in W'_i} w_i, \quad \forall i = 1, \ldots, n, \tag{7.7}$$

$$b_i = \max_{w \in W'_i} w_i, \quad \forall i = 1, \ldots, n. \tag{7.8}$$

Then for each commodity (axis) $i = 1, \ldots, n$, a_i and b_i are the minimum and maximum flow amounts achieved by single commodity i. For W that can be represented by (7.4), $a_i = \lceil \frac{r}{q_i} \rceil$ and $b_i = \lfloor \frac{u}{v_i} \rfloor$, $\forall i = 1, \ldots, n$. The axis points having the entries of a_i and b_i are referred to as *end axis points*, denoted by

$$\underline{w}^i = \{w \in W'_i | w_i = a_i\} = a_i e_i, \quad \forall i = 1, \ldots, n, \tag{7.9}$$

$$\overline{w}^i = \{w \in W'_i | w_i = b_i\} = b_i e_i, \quad \forall i = 1, \ldots, n, \tag{7.10}$$

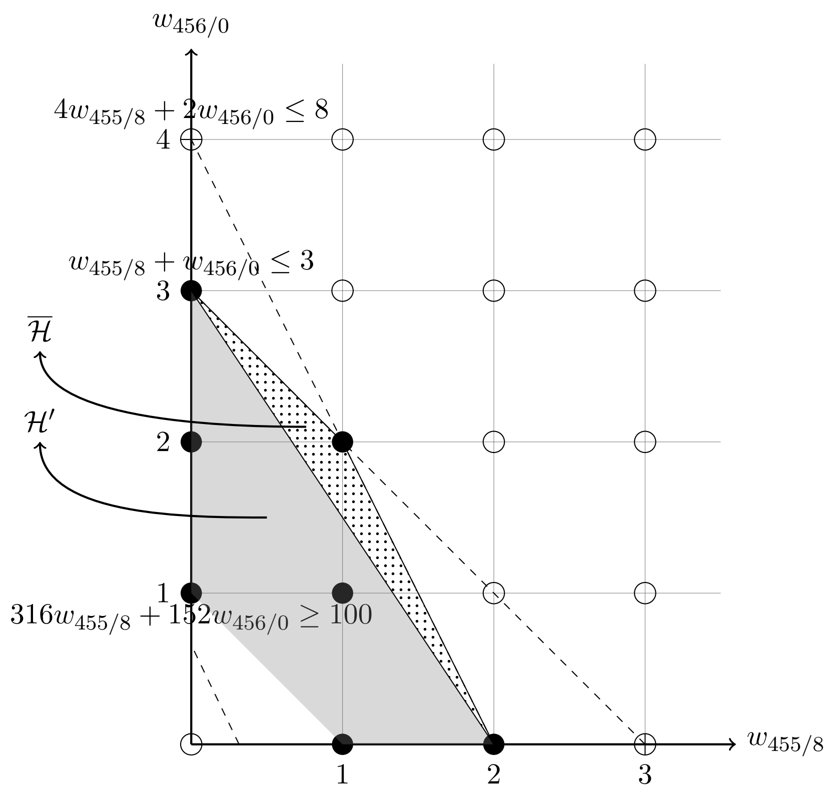
where $e_i \in \mathbb{R}^n$ is the unit vector with a 1 in the i-th entry and 0's in the other entries. Let $V' = \{\underline{w}^1, \overline{w}^1, \ldots, \underline{w}^n, \overline{w}^n\}$ be the set of all end axis points. Note that it is possible for a commodity i to have the case of $a_i = b_i$ such that $\underline{w}^i = \overline{w}^i$. Therefore $n \leq |V'| \leq 2n$.

We then define a polytope $\mathcal{H}'$ called the *main hull* as the convex hull of all end axis points, i.e. $\mathcal{H}' = \text{conv}(V')$. Since $V' \subseteq W$, then $\mathcal{H}' \subseteq \mathcal{H}$. Figure 7.1 shows an example of the main hull in the convex hull from Example 4.1. For V' and $\mathcal{H}'$ we also have the following result.

Proposition 7.1. *V' is the set of vertices of $\mathcal{H}' = \text{conv}(V')$.*

Proof. Let V'' be the set of vertices of $\mathcal{H}'$. First it is true that $V'' \subseteq V'$ [79]. Second any point in V' cannot be expressed as a convex combination of any other points in V', since they are the end axis points of the axes. Thus all points in V' are vertices of $\mathcal{H}'$, or $V' \subseteq V''$. Therefore $V' = V''$. □

The importance of $\mathcal{H}'$ lies in two aspects: First it can only have at most only two nonzero facets; second it often contains a large proportion of the points in W for many problem instances in practice. The subsequent sections will elaborate the above two aspects.



$w_{456/0}$
$4w_{455/8} + 2w_{456/0} \leq 8$
$w_{455/8} + w_{456/0} \leq 3$
$\overline{\mathcal{H}}$
$\mathcal{H}'$
$316w_{455/8} + 152w_{456/0} \geq 100$
$w_{455/8}$
1
2
3
4

Figure 7.1: The main hull $\mathcal{H}'$, the up hull $\overline{\mathcal{H}}$ and the down hull (empty) from Example 4.1

7.2.1 Nonzero facets of main hull $\mathcal{H}'$

7.2.1.1 Preliminaries

We will first show that $\mathcal{H}'$ may have no more than two nonzero facets—facets that are not zero facets, where the theoretical analyses are to be further investigated in the future. For an optimisation problem defined in $\mathbb{R}^n_+$, a zero facet represents a constraint in the form of $w_i \geq 0$ which is often satisfied implicitly. Here we briefly give some results in polyhedral combinatorics that will be used in deriving the above conclusion. For details of the results, see Nemhauser and Wolsey [65], Webster [79] and Mahjoub [60].

In $\mathbb{R}^n$, a set of r points $w^{(1)}, \dots, w^{(r)} \in \mathbb{R}^{n\times 1}$ is affinely independent if the unique solution of the system of r variables $\sum_{i=1}^r \lambda_i w^{(i)} = \mathbf{0}$, $\sum_{i=1}^r \lambda_i = 0$ is $\lambda_i = 0, \forall i = 1, \dots, r$. Equivalently, letting $A_{\{w^{(1)},\dots,w^{(r)}\}} = \left(\begin{smallmatrix} w^{(1)}, & \dots, & w^{(r)} \\ 1, & \dots, & 1 \end{smallmatrix}\right) \in \mathbb{R}^{(n+1)\times r}$ be the matrix associated with the system, the points are affinely independent if $\operatorname{rank}\big(A_{\{w^{(1)},\dots,w^{(r)}\}}\big) = r$. r affinely independent points will uniquely define an $(r-1)$-flat (aka an $(r-1)$-dimensional affine set) which equals their affine hull. In particular, an $(n-1)$-flat is also called a hyperplane in $\mathbb{R}^n$. A set $\varPi$ in $\mathbb{R}^n$ is a hyperplane if and only if there exist scalars $\pi_0, \pi_1, \dots, \pi_n$, being not all zero, such that

$$\varPi = \{w \in \mathbb{R}^n | \pi_1 w_1 + \cdots + \pi_n w_n = \pi_0\} := \{w \in \mathbb{R}^n | \pi^T w = \pi_0\}. \tag{7.11}$$

Each r-flat ($r = -1, 0, \dots, n$) can be expressed as the intersection of $n - r$ hyperplanes, and so is the solution set of some system of $n - r$ linear equations.

Let S' be a subset of a finite set S in $\mathbb{R}^n$ such that $\text{aff}(S') \cap \text{conv}(S \setminus S') = \emptyset$. Then $\text{conv}(S')$ is a face of the polytope $\text{conv}(S)$. If F is a nonempty $(r-1)$-face of $\text{conv}(S)$, then there are r affinely independent points in $S \cap F$. A face F of $\text{conv}(S)$ is a facet of $\text{conv}(S)$ if $\dim(F) = \dim(\text{conv}(S)) - 1$.

A polyhedron $P \in \mathbb{R}^n$ is full-dimensional if $\dim(P) = n$. It has a unique minimal representation

$$P = \{w \in \mathbb{R}^n | (\pi^i)^T w \le \pi_0^i, \forall i = 1 \dots, t\} \tag{7.12}$$

as a finite set of t linear inequalities each representing a facet of P. For a polyhedron $P \in \mathbb{R}^n$ that is not full-dimensional with $\dim(P) = n - k, k > 0$, its minimal representation is

$$P = \left\{ w \in \mathbb{R}^n \middle| \begin{array}{l} (\pi^i)^T w = \pi_0^i, \forall i = 1, \dots, k; \\ (\pi^i)^T w \le \pi_0^i, \forall i = k+1, \dots, k+t. \end{array} \right\}, \tag{7.13}$$

where each inequality of $i = k+1, \dots, k+t$ is from the equivalence class of inequalities representing a facet of P.

7.2.1.2 Number of nonzero facets of main hull

The following Theorem 7.1 gives the number of nonzero facets of $\mathcal{H}' = \text{conv}(V')$.

Theorem 7.1. *When $a_i \neq b_i$ for at least one commodity i, then the main hull $\mathcal{H}' = \text{conv}(V')$ is full-dimensional and has precisely 2 nonzero facets represented by*

$$\frac{w_1}{a_1} + \dots + \frac{w_n}{a_n} \ge 1, \tag{7.14}$$

$$\frac{w_1}{b_1} + \dots + \frac{w_n}{b_n} \le 1, \tag{7.15}$$

and has at most n zero facets represented by

$$w_i \ge 0, \quad \forall i = 1, \dots, n. \tag{7.16}$$

When $a_i = b_i, \forall i = 1, \dots, n$, then $\mathcal{H}'$ degenerates into an $(n-1)$-face per se represented by a single nonzero hyperplane

$$\frac{w_1}{a_1} + \dots + \frac{w_n}{a_n} = 1, \tag{7.17}$$

and n zero facets each corresponding to one of the inequalities as in (7.16).

Proof. To get the very details of $\mathcal{H}$, we use a straightforward way in proving based on the points in W.

(1) First consider the case $a_i \neq b_i$ for at least one commodity i.

Let $I = \{1, \ldots, n\}$. Divide the commodities into two groups as $I^{\neq} = \{i \in I | a_i \neq b_i\}$ and $I^{=} = \{i \in I | a_i = b_i\}$ such that $I = I^{\neq} \cup I^{=}$, $1 \leq |I^{\neq}| \leq n$ and $0 \leq |I^{=}| \leq n-1$. Since $|I^{\neq}| \geq 1$, without loss of generality, $\exists k \in I^{\neq}$ such that we can find $n+1$ points from $\mathcal{H}'$ as $\underline{w}^1, \ldots, \underline{w}^k, \overline{w}^k, \ldots, \underline{w}^n$. Then we have

$$\operatorname{rank}(A_{\{\underline{w}^1, \ldots, \underline{w}^k, \overline{w}^k, \ldots, \underline{w}^n\}}) = \operatorname{rank} \begin{pmatrix} \underline{w}_1^1 & & & & & & \\ & \underline{w}_2^2 & & & & & \\ & & \ddots & & & & \\ & & & \underline{w}_k^k & \overline{w}_k^k & & \\ & & & & & \ddots & \\ & & & & & & \underline{w}_n^n \\ 1 & 1 & \ldots & 1 & 1 & \ldots & 1 \end{pmatrix} = n+1, \tag{7.18}$$

which shows the $n+1$ points are affinely independent. Therefore $\mathcal{H}'$ is full-dimensional with $\dim(\mathcal{H}') = n$. Its minimal representation is a finite set of inequalities each corresponds to a facet of $\mathcal{H}'$.

Based on the $n < |V'| \leq 2n$ points in V', we can find all possible facets of $\operatorname{conv}(V')$ by enumerating the $\binom{|V'|}{n}$ combinations and checking their validity. Two cases are identified and will be discussed separately.

Case 1: Collect the n points by taking one and only one from each of the n axis as $w^{(1)}, \ldots, w^{(n)}$ such that $w^{(i)} \in \{\underline{w}^i, \overline{w}^i\}, \forall i \in I$. The n points are affinely independent since it can be verified that $\operatorname{rank}(A_{\{w^{(1)}, \ldots, w^{(n)}\}}) = n$. Suppose the hyperplane formed by them is $\{w \in \mathbb{R}^n | \pi^T w = \pi_0\}, (\pi_0, \pi) \neq \mathbf{0}$. Then the solution of the system $\pi^T w^{(i)} = \pi_0, \forall i \in I$ is $(\pi_0, \pi) = c\Big(1, \frac{1}{w_1^{(1)}}, \ldots, \frac{1}{w_n^{(n)}}\Big)^T, \forall c \in \mathbb{R} \setminus \{0\}$. Letting $c = 1$ gives a convenient expression of this nonzero hyperplane:

$$\frac{w_1}{w_1^{(1)}} + \cdots + \frac{w_n}{w_n^{(n)}} = 1. \tag{7.19}$$

If $w^{(i)} = \underline{w}^i, \forall i \in I$, then (7.19) becomes $\frac{w_1}{a_1} + \cdots + \frac{w_n}{a_n} = 1$, or in short as $(a^{-1})^T w = 1$, which supports $\operatorname{conv}(V')$ since (i) $(a^{-1})^T w \geq 1, \forall w \in V'$; (ii) There are n affinely independent points $\underline{w}^1, \ldots, \underline{w}^n \in V'$ such that $(a^{-1})^T \underline{w}^i = 1, \forall i \in I$. Therefore, $(a^{-1})^T w \geq 1, w \in \mathbb{R}^n$ defines a nonzero facet of $\operatorname{conv}(V')$ as given by (7.14). By similar reasoning, it can be concluded that if $w^{(i)} = \overline{w}^i, \forall i \in I$, then (7.19) will give another nonzero facet of $\operatorname{conv}(V')$ as given by (7.15).

If neither $w^{(i)} = \underline{w}^i$, $\forall i \in I$ nor $w^{(i)} = \overline{w}^i$, $\forall i \in I$ (which can only happen when $|I^{\neq}| \geq 2$, since when $|I^{\neq}| = 1$, the only two possible combinations are still $w^{(i)} = \underline{w}^i$ and $w^{(i)} = \overline{w}^i$, $\forall i \in I$), the hyperplane given by (7.19) cannot yield any valid inequality. To see this, suppose $|I^{\neq}| \geq 2$ and not all commodities in $I^{\neq}$ are from the same group of $\{\underline{w}^i\}_{\forall i \in I^{\neq}}$ or $\{\overline{w}^i\}_{\forall i \in I^{\neq}}$. Divide the commodities in $I^{\neq}$ into two nonempty subsets as $I^{\neq} = \underline{I}^{\neq} \cup \overline{I}^{\neq}$ such that $i \in \underline{I}^{\neq}$ if $w^{(i)} = \underline{w}^i$ and $i \in \overline{I}^{\neq}$ if $w^{(i)} = \overline{w}^i$. Then the hyperplane given by (7.19) would be

$$h(w) = \sum_{i \in \underline{I}^{\neq}} \frac{w_i}{a_i} + \sum_{i \in \overline{I}^{\neq}} \frac{w_i}{b_i} + \sum_{i \in I^{=}} \frac{w_i}{a_i} - 1 = 0. \tag{7.20}$$

Now we can always find at least two points in V' as $\underline{w}^p = a_p e_p, p \in \overline{I}^{\neq}$ and $\overline{w}^q = b_q e_q, q \in \underline{I}^{\neq}$, such that $h(\underline{w}^p) = \frac{a_p}{b_p} - 1 < 0$ and $h(\overline{w}^q) = \frac{b_q}{a_q} - 1 > 0$. Therefore (7.20) cannot yield any valid inequality and is not facet-defining for $\operatorname{conv}(V')$.

Case 2: Collect the n end axis points such that the points from k of the axes will not be present. Note since each axis only has at most two distinct points and there are n axes, then points from an axis $p \in I$ are absent if and only if another axis $q \in I^{\neq}$ has both $\underline{w}^q$ and $\overline{w}^q$ collected. Thus for a given set of collected points $1 \leq k \leq |I^{\neq}|$, i.e. there will be no more than $|I^{\neq}|$ absent axes or double-collected axes.

Suppose $p_1, \ldots, p_k \in I$ are absent and $q_1, \ldots, q_k \in I^{\neq}$ are correspondingly double-collected and let the n points be $w^{(1)}, \ldots, w^{(n)}$ such that $w^{(i)} \in \{\underline{w}^i, \overline{w}^i\}, \forall i \in I \setminus \{p_1, \ldots, p_k\}$, $w^{(p_j)} \in \{\underline{w}^{q_j}, \overline{w}^{q_j}\}$ and $w^{(p_j)} \neq w^{(q_j)}$, $\forall j = 1, \ldots, k$. Then by a similar reasoning as in (7.18) and noticing there are k rows of all zeros in $A_{\{w^{(1)}, \ldots, w^{(n)}\}}$ corresponding to the k missing axes, we have $\operatorname{rank}\big(A_{\{w^{(1)}, \ldots, w^{(n)}\}}\big) = n - k + 1$.

When $k = 1$, the above rank is n showing the n points are still affinely independent. Suppose they form a hyperplane $\{w \in \mathbb{R}^n | \pi^T w = \pi_0\}, (\pi_0, \pi) \neq \mathbf{0}$. Then the solution of the system $\pi^T w^{(i)} = \pi_0, \forall i \in I$ is $\pi_{p_1} = c, \forall c \in \mathbb{R} \setminus \{0\}, \pi_i = 0, \forall i \neq p_1$, which however leads to a zero facet represented by $w_{p_1} \geq 0$.

When $1 < k \leq |I^{\neq}|$, the above rank is less than n, showing the n points are no longer affinely independent. They can be disregarded in the search for the facets of $\mathcal{H}'$. We will show that none of the facets can be derived from them. Suppose $\operatorname{aff}\{w^{(i)}\}_{i=1}^n \cap \operatorname{conv}(V' \setminus \{w^{(i)}\}_{i=1}^n) = \emptyset$ such that $\operatorname{conv}\{w^{(i)}\}_{i=1}^n$ defines a face F of $\mathcal{H}'$, then we have $\dim(F) = \dim(\operatorname{conv}\{w^{(i)}\}_{i=1}^n) = \dim(\operatorname{aff}(\operatorname{conv}\{w^{(i)}\}_{i=1}^n)) = \dim(\operatorname{aff}\{w^{(i)}\}_{i=1}^n) < n - 1$. In fact here the points can only yield some zero faces of dimensions less than $n - 1$ as the intersections of some zero facets.

Now we can have an exact description of the zero facets in $\mathcal{H}'$. Since the absence of points from one and only one axis p leads to a zero facet $w_p \geq 0$, the zero facets are solely determined by the axes that are absent in all possible point combinations with $k = 1$. We only focus on the cases when $n > 1$ as the condition $n = 1$ is trivial. If $|I^{\neq}| > 1$, when an axis $p \in I$ is absent there is always at least a $q \in I^{\neq}$ available to be double-collected, including those $p \in I^{\neq}$ with a $q \in I^{\neq} \setminus \{p\}$. So there are n zero facets $w_i \geq 0, i \in I$. If $|I^{\neq}| = 1$, however, there is no axis to be double-collected if the only axis $p \in I^{\neq}$ is absent. Thus there are $n - 1$ zero facets $w_i \geq 0, \forall i \in I^{=}$.

(2) Second consider the case $a_i = b_i$, $\forall i = 1, \ldots, n$, such that $|V'| = n$ and $|I^{\neq}| = 0$.

Let the vertex set $V' = \{w^{(1)}, \ldots, w^{(n)}\}$, which contains n affinely independent points. We have $\dim(\mathcal{H}') = \dim(\operatorname{conv}\{w^{(i)}\}_{i=1}^n) = \dim(\operatorname{aff}(\operatorname{conv}\{w^{(i)}\}_{i=1}^n)) = \dim(\operatorname{aff}\{w^{(i)}\}_{i=1}^n) =$

$n-1$. Therefore $\mathcal{H}'$ is not a full-dimensional polytope. Since $\dim(\mathcal{H}') = n-1$, its minimal representation consists of (i) a finite set of inequalities each corresponds to a facet of $\mathcal{H}'$ and (ii) an equality that is attained by all points in $\mathcal{H}'$.

The equality is just the hyperplane $\Pi_0 = \operatorname{aff}\{w^{(1)}, \ldots, w^{(n)}\} = \{w \in \mathbb{R} | \sum_{k=1}^{n} \frac{w_k}{a_k} = 1\}$ as given by (7.19).

Facets of $\mathcal{H}'$ are $(n-2)$-faces each being a convex hull formed by $n-1$ affinely independent points in V'. Moreover, each facet is also associated with a supporting $(n-2)$-flat which is the affine hull of the same $n-1$ points that forms the facet. The number of facets will not exceed n as at most n such point combinations can be from V' by each time removing a point in axis p, $\forall p \in I$. Now consider the n flats of dimension $n-2$: $\Phi_p = \operatorname{aff}\{w^{(i)}\}_{\forall i \in I \setminus \{p\}}, \forall p \in I$ as intersections of two non-parallel hyperplanes Π_0 and Π_p

$$\Phi_p : \begin{cases} \Pi_0 : \frac{w_1}{a_1} + \cdots + \frac{w_n}{a_n} = 1, \\ \Pi_p : w_p = 0. \end{cases} \quad \forall p \in I. \tag{7.21}$$

Let $h_0(w) = \frac{w_1}{a_1} + \cdots + \frac{w_n}{a_n} - 1$ and $h_p(w) = w_p$, $\forall p \in I$. Then Φ_p supports $\mathcal{H}'$, $\forall p \in I$ since (i) $h_0(w^{(i)}) = 0$, $h_p(w^{(i)}) \geq 0$, $\forall i \in I$ and (ii) there are $n-1$ affinely independent points $\{\underline{w}^j\}_{j \in I \setminus \{p\}} \in \mathcal{H}'$ such that $h_0(\underline{w}^j) = 0$ and $h_p(\underline{w}^j) = 0$. Therefore, apart from the redundant valid inequality from Π_0, the remaining valid inequalities representing facets of $\mathcal{H}'$ are just $w_p \geq 0, \forall p \in I$, as given by (7.16). □

Now we have the exact description of the main hull, as (it is also valid for degenerated $\mathcal{H}'$)

$$\mathcal{H}' = \left\{ w \in \mathbb{R}^n_+ \middle| \frac{w_1}{a_1} + \cdots + \frac{w_n}{a_n} \geq 1, \frac{w_1}{b_1} + \cdots + \frac{w_n}{b_n} \leq 1 \right\}. \tag{7.22}$$

It can be verified that this is actually a frustum of a simplex formed by the intersection of an n-simplex $\left\{ w \in \mathbb{R}^n_+ \middle| (b^{-1})^T w \leq 1 \right\}$ and a halfspace $\left\{ w \in \mathbb{R}^n \middle| (a^{-1})^T w \geq 1 \right\}$. See [76] for details on a frustum of a simplex.

7.2.2 Number of outside points

Now consider the points in W outside the main hull $\mathcal{H}'$ which we refer to as *outside points* as the remaining points in $W \setminus W'$. Since $\mathcal{H}'$ has only at most two nonzero facets, for an outside point $w \in W \setminus W'$, it is either in $\overline{W} = \{w \in W | (b^{-1})^T w > 1\}$ or in $\underline{W} = \{w \in W | (a^{-1})^T w < 1\}$. Thus we define an *up hull* and a *down hull* as

$$\overline{\mathcal{H}} = \left\{ w \in \mathcal{H} \middle| \frac{w_1}{b_1} + \cdots + \frac{w_n}{b_n} > 1 \right\}, \tag{7.23}$$

$$\underline{\mathcal{H}} = \left\{ w \in \mathcal{H} \middle| \frac{w_1}{a_1} + \cdots + \frac{w_n}{a_n} < 1 \right\}. \tag{7.24}$$

such that $W = W' \cup \overline{W} \cup \underline{W}$ and $\mathcal{H} = \mathcal{H}' \cup \overline{\mathcal{H}} \cup \underline{\mathcal{H}}$. Note that $\overline{\mathcal{H}}$ and $\underline{\mathcal{H}}$ are convex sets each formed by a polytope without one of its facet. It is also not difficult to verify that $\overline{W} = \overline{\mathcal{H}} \cap \mathbb{Z}^n_+$ and $\underline{W} = \underline{\mathcal{H}} \cap \mathbb{Z}^n_+$. Figure 7.1 gives an example of the upper hull and the down hull (empty) in the convex hull from Example 4.1.

If the number of outside points is of moderate size, and the dimension n is appropriately small, then the entire convex hull $\mathcal{H}$ can be computed based on $\mathcal{H}'$, $\overline{W}$ and $\underline{W}$ by some convex hull algorithms. This might be more efficient and less intractable than computing $\mathcal{H}$ directly starting with the given points in W. In this part we will briefly explore the number of outside points $\overline{W}$ and $\underline{W}$ on and leave the discussion of this convex hull algorithm to § 7.3.

7.2.2.1 Two special conditions

The number of outside points can be analytically determined under two special conditions, i.e. when all commodities are incompatible and when u (or r) as given in (7.4) is a multiple of all elements in v (or q).

For an instance with commodity compatibility relations as mentioned in § 7.1.1, the combination point enumeration and outside point counting can be decomposed into subsets of compatible commodities. Let $I_1, \dots, I_S \subset I = \{1, \dots, n\}$ be the subsets of commodities each containing compatible commodities such that $I_{s_1} \cap I_{s_2} = \emptyset, \forall s_1 \neq s_2$. With respect to each subset $s = 1, \dots, S$, we have the combination set W_s such that $\bigcup_{s=1}^S W_s = W$, and the outside points $\overline{W}_s, \underline{W}_s$ defined by $\sum_{i \in I_s} \frac{w_i}{b_i} > 1$ or $\sum_{i \in I_s} \frac{w_i}{a_i} < 1$ such that $\bigcup_{s=1}^S \overline{W}_s = \overline{W}$ and $\bigcup_{s=1}^S \underline{W}_s = \underline{W}$. In fact in this case $\mathcal{H} = \text{conv}(W)$ can be constructed by "wrapping the projections" of the sub-hulls of each subset. Moreover, when commodities are all incompatible with each other (which can be found in real-world instances in train unit scheduling), the following Proposition 7.2 states that the convex hull can be given directly by the main hull since $\underline{W}$ and $\overline{W}$ are both empty.

Proposition 7.2. *For a combination set W where all commodities $k \in K$ are incompatible with each other, W will only have axis points such that $W = W'$ and $\mathcal{H} = \mathcal{H}'$.*

In addition, consider a combination set $W = \{w \in \mathbb{Z}^n_+ | q^T w \geq r, v^T w \leq u\}$ that can be implicitly defined by (7.4). Two simplices can be found such that their difference contains the up hull as $\overline{\mathcal{H}} \subseteq \overline{P} = S_{vu} \setminus S_b$, where $S_{vu} = \{w \in \mathbb{R}^n_+ | v^T w \leq u\}, S_b = \{w \in \mathbb{R}^n_+ | (b^{-1})^T w \leq 1\}$ and $b_i = \lfloor \frac{u}{v_i} \rfloor, \forall i = 1, \dots, n$. The situation is slightly more complicated for the down hull where $\underline{\mathcal{H}} \subseteq \underline{P} = S_a \setminus S_{qr} \cup \Pi_{qr} \setminus \Pi_a$, where $S_{qr} = \{w \in \mathbb{R}^n_+ | q^T w \leq r\}$, $S_a = \{w \in \mathbb{R}^n_+ | (a^{-1})^T w \leq 1$, $\Pi_{qr} = \{w \in \mathbb{R}^n_+ | q^T w = r\}$, $\Pi_a = \{w \in \mathbb{R}^n_+ | (a^{-1})^T w = 1\}$, and $a_i = \lceil \frac{r}{q_i} \rceil, \forall i = 1, \dots, n$.

Here we have the following properties for the emptiness of $\overline{P}$ and $\underline{P}$, due to the fact that $a_i = \lceil \frac{r}{q_i} \rceil, b_i = \lfloor \frac{u}{v_i} \rfloor, \forall i = 1, \dots, n$.

Proposition 7.3. *For a combination set W that can be defined by* (7.4)*, if u is a multiple of all $v_i, i = 1, \dots, n$, then $\overline{P} = S_{vu} \setminus S_b = \emptyset$ such that $\overline{\mathcal{H}} = \emptyset$; if r is a multiple of all $q_i, i = 1, \dots, n$, then $\underline{P} = S_a \setminus S_{qr} \cup \Pi_{qr} \setminus \Pi_a = \emptyset$ such that $\underline{\mathcal{H}} = \emptyset$.*

The above condition can often happen in real-world instances. Taking the train unit scheduling problem for example, where the upper bound u is measured in number of cars and v_i are the number of cars of unit type i, there can be many trains with u as the multiple of all types' car numbers in the instances both from Southern Railways and ScotRail. If u is measured in number of units, as in [15], then this condition for the upper hull will always hold. Also note that when $\overline{P} \neq \emptyset$ it is still possible that $\overline{\mathcal{H}} = \emptyset$. On the other hand, r, as the demand measured in passenger numbers, can hardly be a multiple of all q_i, which are the numbers of seats of unit types $i = 1, \dots, n$. Although $\underline{P}$ can hardly be empty, the size of $|\underline{W}|$ tends to be very small, even often be zero, since a non-empty $\underline{P}$ does not necessarily imply a non-empty $\underline{W}$. Figure 7.1 is a good example. This fact can also be observed in the experiments to be reported in § 7.4.

7.2.2.2 Empirical experiments

Pragmatically for a given set W with points w, it is sufficient to determine the number of outside points simply by checking the values of $h_a(w) = (a^{-1})^T w - 1$ and $h_b(w) = (b^{-1})^T w - 1$. If $h_a(w) < 0$ then $w \in \underline{W}$ and if $h_b(w) > 0$ then $w \in \overline{W}$. A series of computational experiments were conducted to show the characteristics on the number of outside points under different circumstances. They will be reported in § 7.4 in detail.

7.3 A customised QuickHull algorithm to compute local convex hulls

In this section we will describe how to use a slightly modified QuickHull algorithm adapted from [6] to exactly compute the convex hull $\mathcal{H} = \text{conv}(W)$ based on the main hull $\mathcal{H}'$ and the outside points in $\underline{W}, \overline{W}$. QuickHull is an algorithm that is theoretically able to compute the convex hull of a finite set of points in $\mathbb{R}^n$. The capability of the QuickHull algorithm is always problem-specific, although it is reported that a generic QuickHull is suitable for medium/large-sized inputs for $n \leq 8$ while not suitable for medium-sized inputs for $n \geq 9$ [66]. Its rationale is based on the following simplified Grünbaum's Beneath-Beyond Theorem [6, 38].

Theorem 7.2 (Grünbaum). *Let $\mathcal{H}$ be a convex hull in $\mathbb{R}^n$, and let w be a point in $\mathbb{R}^n \setminus \mathcal{H}$. Then F is a facet of* conv$(w \cup \mathcal{H})$ *if and only if*

(i) *F is a facet of $\mathcal{H}$, and w is below F; or*

(ii) *F is not a facet of $\mathcal{H}$, and its vertices are w and the vertices of a ridge (i.e. an $(n-2)$-face) of $\mathcal{H}$ with one incident facet below w and the other incident facet above w.*

A point's position as being above or below a hyperplane/facet is defined by giving the hyperplane/facet an orientation as their outer normal's direction and if the signed distance of a point to the hyperplane/facet is positive (negative), then the point is said to be above (below) the hyperplane/facet. A facet is said to be visible to a point if the point is above it.

For a given set of points, QuickHull first selects a nondegenerated subset of them to form an initial simplex as their convex hull. If possible, this initial simplex will be selected such that it will cover as many points as possible by choosing the points with either a maximum or minimum coordinate. Each point outside the initial simplex will be assigned to one and only one of its visible facet(s) of this simplex. Then recursive processes will be done for each facet of the updated hull with its associated outside points. Within each facet, one of its associated point (generally the "furthest" one) will be selected. New facets will be made by joining this point and all horizonal ridges that enclose all visible facets of this point. A process called partitioning will either reallocate each outside points associated with one visible facet to a new facet or include that point into the hull. Then the point's visible facets will be discarded. This will be repeated for the hull with updated facets until all facets have empty outside point sets. See [6] for details of the QuickHull algorithm.

From the view of the QuickHull algorithm, it can be seen that for the case of commodity combination set W, the "up" and "down" points in $\overline{W}$ and $\underline{W}$ should be above the hyperplane $(b^{-1})^T w = 1$ and $(a^{-1})^T w = 1$ respectively. Moreover, the main hull $\mathcal{H}'$, as a frustum of a simplex, should be an ideal alternative for the initial simplex in the QuickHull's first stage (by Theorem 7.2, any full-dimensional convex polytope inside $\mathcal{H}$ would do the job). Moreover, when $\mathcal{H}'$ is chosen as the initial "simplex", there is initially only two visible facets—the two nonzero facets—to be further processed possibly in parallel with the "up" and "down" points. If the sizes of $|\overline{W}|$ and $|\underline{W}|$ are within an appropriate range and the dimension n is not too large, then QuickHull is very likely to compute the convex hull $\mathcal{H}$ within reasonable time. A customised "2-facet" QuickHull algorithm is given in Algorithm 6, which can be regarded as a tailored version of [6] only differing in how to construct the "initial simplex".

In practice, implementing Algorithm 6 from scratch may require a considerable amount of work. Pragmatically having the free and highly efficient QuickHull program available from its official site [66], one may consider the following alternative Algorithm 7 by directly using the official QuickHull program from [66].

Algorithm 6 A 2-facet QuickHull for computing $\mathcal{H}$

```
given: $W$
create the main hull $\mathcal{H}'$
generate two nonzero facets: $\underline{F} = \{w \in \mathbb{R}^n_+ | (a^{-1})^T w = 1\}$, $\overline{F} = \{w \in \mathbb{R}^n_+ | (b^{-1})^T w = 1\}$
generate the outside points $\underline{W}, \overline{W}$
initialise the nonzero facet set $\mathcal{F} \leftarrow \{\underline{F}, \overline{F}\}$
initialise the outside point set $OP(\underline{F}) \leftarrow \underline{W}$, $OP(\overline{F}) \leftarrow \overline{W}$,
for all $F \in \mathcal{F} | OP(F) \neq \emptyset$ do
  select the furthest point $w$ in $OP(F)$
  initialize the visible set $\mathcal{V}(w) \leftarrow \{F\}$
  for all unvisited neighbor facets $F_N$ of facets in $\mathcal{V}(w)$ do
    if $w$ is above $F_N$ then
      $\mathcal{V}(w) \leftarrow \mathcal{V}(w) + \{F_N\}$
    end if
  end for
  $\mathcal{V}(w)$'s boundary $\partial\mathcal{V}(w)$ is the set of horizon ridges
  initialize the new facet set of $w$ as $\mathcal{F}^+(w) \leftarrow \emptyset$
  for all $R \in \partial\mathcal{V}(w)$ do
    create a new facet $F'(R, w)$ from $R$ and $w$
    $\mathcal{F}^+(w) \leftarrow \mathcal{F}^+(w) + \{F'(R, w)\}$
  end for
  for all $F' \in \mathcal{F}^+(w)$, $F'' \in \mathcal{V}(w)$ do
    for all unassigned $v \in OP(F'')$ do
      if $v$ is above $F'$ then
        $OP(F') \leftarrow OP(F') + \{v\}$
      end if
    end for
  end for
  $\mathcal{F} \leftarrow \mathcal{F} + \mathcal{F}^+(w)$, $\mathcal{F} \leftarrow \mathcal{F} - \mathcal{V}(w)$
end for
```

A special case in computing local convex hulls for instances with incompatible commodities QuickHull is very sensitive to increasing dimensionality. When the node is to be served by commodities divided into subset $I_1, \ldots, I_S \in I = \{1, \ldots, n\}$ of compatible ones, it is possible to first compute the convex hulls $\mathcal{H}_s$ (either by the customised Algorithm 6 or by a generic QuickHull) within each subset s in lower dimensional spaces $\mathbb{R}^{|I_s|}_+$ and merge the above "sub-hulls" for all subsets into the desired as $\mathcal{H} = \bigcup_{s=1}^{S} \mathcal{H}_s$. One way of performing such merging is to construct $\mathcal{H}$ based on the vertices of all sub-hulls in $\mathbb{R}^n$. There is a nice property such that vertices from different sub-hulls are orthogonal. It is not apparent if this method can be competitive with Algorithm 6 subject to different n. We will leave this alternative method to future research.

7.4 Computational experiments on local convex hulls

7.4.1 Feasibility range of local convex hull method

The feasibility range of the local convex hull method is an interesting topic, which asks to what limit in terms of the input parameters will it become impractical or intractable to compute $\mathcal{H}$ explicitly based on points in W or indirectly based on $\mathcal{H}'$, $\overline{W}$ and $\underline{W}$ by some

Algorithm 7 An alternative 2-facet QuickHull $\mathcal{H}$ directly using the standard QuickHull

```
given: $W$
obtain the up and down end axis points $\overline{V'} = \{\overline{w}^1, \dots, \overline{w}^n\}$ and $\underline{V'} = \{\underline{w}^1, \dots, \underline{w}^n\}$
generate two nonzero facets: $\underline{F} = \{w \in \mathbb{R}^n_+ | (a^{-1})^T w = 1\}$, $\overline{F} = \{w \in \mathbb{R}^n_+ | (b^{-1})^T w = 1\}$
generate the outside points $\underline{W}, \overline{W}$
initialise the nonzero facet set $\mathcal{F} \leftarrow \emptyset$
**for** $\overline{W}$ and $\underline{W}$ in parallel **do**
  **if** $\overline{W} \neq \emptyset$ **then**
    compute the up hull $\overline{\mathcal{H}} = \text{conv}\{\overline{V'}, \overline{W}\}$ by standard QuickHull
    obtain the set of nonzero facets of the up hull $\overline{\mathcal{F}}$
    $\mathcal{F} \leftarrow \mathcal{F} + \overline{\mathcal{F}} \setminus \{\overline{F}\}$
  **else**
    $\mathcal{F} \leftarrow \mathcal{F} + \overline{F}$
  **end if**
  **if** $\underline{W} \neq \emptyset$ **then**
    compute the down hull $\underline{\mathcal{H}} = \text{conv}\{\underline{V'}, \underline{W}\}$ by standard QuickHull
    obtain the set of nonzero facets of the down hull $\underline{\mathcal{F}}$
    $\mathcal{F} \leftarrow \mathcal{F} + \underline{\mathcal{F}} \setminus \{\underline{F}\}$
  **else**
    $\mathcal{F} \leftarrow \mathcal{F} + \underline{F}$
  **end if**
**end for**
```

convex hull algorithms. This would be very useful to guide the modelling and solving for some upgraded or new problem instances. For example, in § 4.1.2 the local convex hulls are easily computed by QuickHull since $|K_j| \leq 4, |W_j| \leq 9, \forall j \in \mathcal{N}$. However, if now it is that $|K_j| = 10$ for trains j as a result of the electrification upgrades in some routes such that electric units can also run on the routes that were previously only covered by diesel units, then it is important to see if the original methods are still practical.

However the problem of determining the feasibility of convex hull algorithms is complicated itself. Generally the difficulty of convex hull computation grows drastically when n becomes large, as a result of the complex components and structural sophistication of high dimensional polytopes. Anyway if the number of outside points are still within a reasonable range, like in many practical problems, the computation may still be possible for some not too large n. At this stage, we may not be able to give a definite or even precise answer to this question. Nevertheless, within possibly the most difficult artificial conditions (with space dimensions no greater than 11) in the context of train unit scheduling, a series of computational experiments have been conducted giving some positive results on the feasibility range of the local convex hull method. Investigations on to what extent local convex hulls are still computable for a broader range of problem instances arising in various integer multicommodity flow problems are outside the scope of this thesis. Nevertheless, to check the performance of the 2-facet QuickHull under certain circumstances, we will report a series of preliminary experiments on the artificial instances based on the fleets of ScotRail and Southern Railway by increasing the space dimensions from 2 to 21, using both the standard and the "2-facet" QuickHull.

7.4.2 Number of points in the three hulls: empirical tests based on real-world and artificial instances

In this part, datasets from ScotRail will be tested with respect to the number of points in $W', \underline{W}$ and $\overline{W}$. Originally, although ScotRail has a train unit fleet with 10 unit types, what makes the scenario quite simple is the fact that for each train no more than 4 unit types (commodities) are permitted to cover it. Moreover, there are type compatibility relations that divide the type set into 6 families. This makes computing convex hulls for all trains a trivial matter in spaces of dimensions no more than 4 with the number of points less than 9. The fleet information of ScotRail can be found in Table 3.1 and 3.2. Now we would like to see the potential of the local convex hull method regarding the number of points in the main, up and down hulls on an artificial train with modified type-route and type-type relations that are rendered more complex and difficult. Two kinds of conditions will be tested separately, as:

(i) Assume each train can be served by all the 10 types, but retain the original compatibility relations among unit types and the combination-specific upper bounds.

(ii) Assume each train can be served by all the 10 types, where all unit types are compatible and have the same coupling upper bound.

The first condition makes the convex hull computation into $\mathbb{R}^{10}_{+}$ with a slightly increased point number. The second condition will give a greatly increased point number in $\mathbb{R}^{10}_{+}$, where as there are many new combinations that do not exist in practice, unified uppers bound u will be used for all combinations varying from 4 to 8 cars which sufficiently covers the real-world cases in ScotRail.

Table 7.1 gives the exact number of points in the upper ($\overline{W}$), main (W') and down ($\underline{W}$) unit combination sets as triplets ($|\overline{W}|, |W'|, |\underline{W}|$) for condition (i). When the main set W' cannot be defined when the "single-commodity-presence" condition is not satisfied (usually due to the demand is too high), the number of all points in W will be reported as a singleton $|W|$. The passenger demand r varies from 25 to 500, which represents the range commonly seen in the ScotRail datasets. The results show that when the type-type compatibility relation and combination-specific coupling upper bound have been retained, and all types are supposed to be allowed at a train, most of the points still lie inside the main hull, in which case there is no need to explicitly compute the train convex hull and it is sufficient to use $(a^{-1})^T w \geq 1$ and $(b^{-1})^T w \leq 1$. Moreover, the number of points for all cases are still very small. Since the upper bounds in practice are multiples of the car numbers of the relevant unit, in all cases the upper hulls are empty, as expected from Proposition 7.3. In one occasion, there are 2 points in the down hull when $r = 300$. When the demand gets larger, the main hull cannot be defined as the last three entries show.

Table 7.2 shows the result based on Condition (ii). In these artificially created scenarios, when the coupling upper bounds are too small and the demands are too large, no

Table 7.1: Condition (i) in $\mathbb{R}_{+}^{10}$, passenger demands r varies from 25 to 500

r	25	50	100	150	200	250	300	350	400	450	500
$(\lvert\overline{W}\rvert, \lvert W'\rvert, \lvert\underline{W}\rvert)$ or $\lvert W\rvert$	(0,27,0)	(0,27,0)	(0,27,0)	(0,25,0)	(0,22,0)	(0,18,0)	(0,13,2)	(0,13,0)	9	3	1

feasible combination can be made, as shown by those cells in the bottom-left corner of the table. Since the restrictions have been further relaxed compared with Condition (i), especially due to the modification that *all* the 10 types are compatible, one can see an increase in the number of combination points. Moreover, the numbers of the points inside the up hull are considerable, since the upper bounds are no longer always multiples of the numbers of cars of all types. Sometimes, there can be more points in the up hulls than in the main hulls.

Experiments on the number of points in $\overline{W}, W', \underline{W}$ were also performed based on instances from other available sources. Table 7.3 gives the results on the number of points in W taken the Instance A from Cacchiani et al. [15] with 8 train unit types. If no such three parts as $\overline{W}, W', \underline{W}$ can be divided, the total number of points $|W|$ will be reported. 50 random number from a uniform distribution on the range of $[360, 1404]$ (as given in [15] as the passenger demand range) were used as the passenger demands. Since in [15] the coupling upper bound is measured in the number of units and for all trains this upper bound is 2 units, it was set that $v_k = 1$ and $u = 2$ for all the 8 types. Moreover, the 8 types are all compatible. Table 7.3 shows that the number of points in the combination set is still quite small from the simulated instance based on the real-world dataset from [15]. Moreover, within these samples, as long as the three parts in W can be divided, the up parts $\overline{W}$ and down parts $\underline{W}$ are all empty, which means the convex hulls can be directly obtained by $(a^{-1})^T w \geq 1$ and $(b^{-1})^T w \leq 1$ without any further computation.

From the above experiments, for the instances within the context of train unit scheduling, generally the numbers of points in the train combination sets are very small and are thus within the capability of a standard convex hull algorithm. The next section will further justify this conclusion empirically.

7.4.3 Computing local convex hulls using standard QuickHull

In this section, a series of experiments on computing local convex hulls are reported. The convex hull computation tool used was an official version of the QuickHull algorithm available from [66]. All experiments were performed on a Dell workstation with 8G RAM and an Intel Xeon E31225 CPU.

The first experiment was based on the three instances (A,B,C) from Cacchiani et al. [15]. The passenger demand ranges for the three instances are given in r_A, r_B, r_C respectively. In Instance A, there are 8 compatible types with their capacities given in q_A. In Instances B and C, there are 10 compatible types respectively with their capacities given

Table 7.2: Numbers of point as "(up, main, down)" or "total" under condition (ii) in $\mathbb{R}^{10}_{+}$, unified upper bounds $u = 4, \dots, 8$, demand $r = 25, \dots, 500$.

r \ u	4	5	6	7	8
25	(0,13,0)	(14,13,0)	(2,59,0)	(30,59,0)	(56,98,0)
50	(0,13,0)	(14,13,0)	(2,59,0)	(30,59,0)	(56,98,0)
100	(0,13,0)	(14,13,0)	(2,59,0)	(30,59,0)	(56,98,0)
150	(0,11,0)	(14,11,0)	(2,57,0)	(30,57,0)	(56,96,0)
200	8	22	(2,54,0)	(30,54,0)	(56,93,0)
250	4	18	(2,50,0)	(30,50,0)	(56,89,0)
300	–	–	48	76	(56,69,0)
350	–	–	40	68	(56,69,0)
400	–	–	25	53	118
450	–	–	1	29	94
500	–	–	–	–	73

Table 7.3: Numbers of points as "(up, main, down)" or "total" in $\mathbb{R}^{8}_{+}$ based on "Instance A" from Cacchiani et al. [15] *

r	1133	625	815	1296	794	1286	1047	1017	395	936
$\lvert W\rvert$	26	(0,40,0)	37	20	37	20	29	32	(0,43,0)	34
r	1017	1047	1390	746	804	828	809	774	943	546
$\lvert W\rvert$	29	29	18	38	37	37	37	38	34	(0,40,0)
r	1344	389	964	1137	786	389	1065	1285	1085	1360
$\lvert W\rvert$	18	(0,43,0)	34	26	38	(0,43,0)	27	20	27	18
r	1056	1105	1373	965	1295	852	1169	621	641	575
$\lvert W\rvert$	29	29	18	34	20	37	24	(0,40,0)	(0,40,0)	(0,40,0)
r	1097	424	663	383	1031	837	421	427	659	709
$\lvert W\rvert$	26	(0,43,0)	(0,40,0)	(0,43,0)	32	37	(0,43,0)	(0,43,0)	(0,40,0)	(0,39,0)

* Input specifications: $r \in \mathcal{U}[360, 1404]$, $q \in \{1150, 1044, 786, 702, 543, 516, 495, 360\}$.

Table 7.4: Number of combination points based on “Instance A,B,C” from Cacchiani et al. [15] *

r_A	*360*[a]	410	460	510	560	610	660	710	760	810	860	910	960	1010	1160	1210	1260	1310	1360	1410
$\|W\|$	44	43	43	42	40	40	40	39	38	37	36	34	34	33	28	26	24	23	21	19

r_B	*590*[b]	640	690	740	790	840	890	940	990	1040	1090	1140	1190	1240	1290	1340	1390	1440	1490	1530
$\|W\|$	64	64	64	64	64	61	59	59	58	58	58	57	56	56	56	56	56	52	50	50

r_C	*590*[c]	640	690	740	790	840	890	940	990	1040	1090	1140	1190	1240	1290	1340	1390	1140	1490	1540
$\|W\|$	64	64	64	64	64	61	59	59	59	59	59	58	57	57	57	57	57	53	51	51

* Input specifications: $r_A \in [360, 1404], q_A \in \{1150, 1044, 786, 702, 543, 516, 495, 360\}$, $r_B \in [588, 1534], q_B \in \{1534, 1473, 1128, 980, 887, 840, 834, 824, 805, 588\}$, $r_C \in [588, 1610], q_C \in \{1644, 1624, 1473, 1128, 887, 840, 834, 824, 805, 588\}$

[a] For Instance A, the time to compute the convex hull of the 44 points from $r = 360$ by standard QuickHull is displayed as 0 second.

[b] For Instance B, the time to compute the convex hull of the 64 points from $r = 590$ by standard QuickHull is displayed as 0 second.

[a] For Instance C, the time to compute the convex hull of the 64 points from $r = 590$ by standard QuickHull is displayed as 0 second.

in q_B and q_C. Note that the coupling upper bound is measured in number of units and is 2 for all trains. The experiments were carried out by increasing the passenger number evenly within the given range and calculating the number of points in the corresponding train unit combinations sets, as shown in Table 7.4. The samples with a passenger demand in italic were the ones with the largest number of points in their unit combination sets and had their local convex hull computed by QuickHull by directly taking the points in W. Notably as for the three samples from Table 7.4, since the number of points were too small, the computation time were all displayed as 0's.

The second experiment was based on the ScotRail fleet as described before. Originally there were strict type compatibility relation among the 10 types and for each train no more than 4 types can be used. This real-world scenario has no challenge for testing the workability of the local convex hull method. Even if we allow all the 10 types to serve a train and retain or disable their compatibility relations, the number of points in the unit combinations sets are still too small to challenge the QuickHull algorithm (as shown in Table 7.1 and Table 7.2). Therefore, an artificial scenario is designed to increase the difficulty in computing the local convex hulls. We assume that all the 10 types are compatible, and for the train to be tested, all the 10 types can be used to serve it. Moreover, we have widened the range of coupling upper bounds to be $[4, 12]$ and the range of passenger demand numbers to be $[25, 900]$. This significantly increases the number of points in the resulting train unit combination sets. Such an artificial manipulation gives us the opportunity to explore the potential of the local convex hull method empirically. Theoretically similar but more difficult conditions may occur in integer multicommodity problems where the number of points in the local convex hulls may be much larger than in the cases of train unit scheduling. Practically other train unit operators rather than ScotRail may have more difficult instances and the advancements of railway infrastructure and engineering technology may propose new conditions that will be much harder than

Table 7.5: Number of points for the artificial instance of ScotRail in $\mathbb{R}_+^{10}$

$u \setminus r$	25	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900
4	13	13	13	11	8	4	–	–	–	–	–	–	–	–	–	–	–	–	–
5	27	27	27	25	22	18	–	–	–	–	–	–	–	–	–	–	–	–	–
6	61	61	61	59	56	52	48	40	25	1	–	–	–	–	–	–	–	–	–
7	89	89	89	87	84	80	76	68	53	29	–	–	–	–	–	–	–	–	–
8	154	154	154	152	149	145	141	133	118	94	73	38	–	–	–	–	–	–	–
9	280	280	280	278	275	271	267	259	244	220	199	163	100	16	–	–	–	–	–
10	404	404	404	402	399	395	391	383	368	344	323	287	224	136	51	–	–	–	–
11	635	635	635	633	630	626	622	614	599	575	554	518	455	367	278	–	–	–	–
12	1029	1029	*1029*[a]	1027	1024	1020	1016	1008	993	969	948	912	849	761	672	545	335	99	3

[a] The time to compute the convex hull of the 1029 points from $r = 100, u = 12$ by standard QuickHull is 0.006 seconds.

the current status. Bearing in mind that the local convex hull method cannot be used universally for all kinds of integer multicommodity flow problems, it is useful to have a preliminary idea on its practical range subject to different scenarios.

As for the above artificially created instances, Table 7.5 gives the results. Note that when the passenger demand is too high and the coupling upper bound is too small, there will be no feasible combination (marked as "–"), which mainly appear at the top-right corner of the table. These r, u pairs causing the infeasible scenarios nevertheless will be unlikely to happen in practice. The maximum number of points in W is 1029 in three scenarios, which is still too easy for QuickHull to compute, by merely 0.006 seconds.

In practice, there are frequent type-type compatibility relations among the Southern Railway unit types, and there will be no more than 7 types available for any trains. Those real-world conditions are not challenging for the standard QuickHull algorithm. Therefore, a similar series of experiments were carried out for the 11 unit types in the Southern Railway fleet, assuming that they were all compatible and were all allowed to serve any trains. The ranges of passenger demand r and coupling upper bound u were also widened.

Table 7.6 gives the results on the experiments for the artificial scenarios based on the Southern fleet. It has some similar patterns as the experiments for ScotRail in Table 7.5. The convex hull of one of the u, r pairs yielding the maximum number of combination points, $r = 25, u = 12$ with 505 points was computed by standard QuickHull, giving a computation time of 0.015 seconds.

7.4.4 Computing local convex hulls using 2-facet QuickHull for more difficult cases

Computing local convex hulls in spaces with even higher dimensions (e.g. $\mathbb{R}_+^n$ with $n > 11$) is outside the scope of this thesis. Therefore this part is merely presented for completeness. Moreover, a comparison between standard QuickHull and "2-facet" QuickHull can be made under these more difficult conditions because as shown in § 7.4.3, the instances with $n \leq 11$

Table 7.6: Number of points for the artificial instance of Southern Railway in $\mathbb{R}_+^{11}$

$u \setminus r$	25	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900
4	12	12	12	11	8	3	2	–	–	–	–	–	–	–	–	–	–	–	–
5	17	17	17	16	13	8	6	–	–	–	–	–	–	–	–	–	–	–	–
6	34	34	34	33	30	25	23	12	4	2	–	–	–	–	–	–	–	–	–
7	52	52	52	51	48	43	41	30	19	8	–	–	–	–	–	–	–	–	–
8	96	96	96	95	92	87	85	74	62	44	21	7	3	–	–	–	–	–	–
9	136	136	136	135	132	127	125	114	102	84	57	28	11	–	–	–	–	–	–
10	223	223	223	222	219	214	212	201	189	171	143	107	63	27	9	3	–	–	–
11	320	320	320	319	316	311	309	298	286	268	240	203	154	92	42	15	–	–	–
12	*505*[a]	505	505	504	501	496	494	483	471	453	425	388	336	268	178	97	38	15	4

[a] The time to compute the convex hull of the 505 points from $r = 25, u = 12$ by standard QuickHull is 0.015 seconds.

are not challenging enough.

A series of experiments on artificial problem instances with dimensions varying from 2 to 21 will be briefly reported here. However, future research regarding train unit scheduling will not further focus on this particular issue unless necessary.

This series of experiments was conducted in the same machine as described in § 7.4.3. The instances are artificially created based on the fleets of ScotRail (10 types) and Southern Railway (11 types) by gradually increasing the number of types (assuming all compatible) in the merged artificial fleet from 2 to 21 in a way that the types of ScotRail will be first included and the types of Southern Railway will be added one by one after the entire number of types is over 10. The fixed demand $r = 107$ and shared coupling upper bound $u = 9$ is used. Since 107 is the smallest unit capacity among all unit types, this may give different unit combinations as many as possible. Moreover, since any type can handle the demand 107 on its own, $a_i = 1, \forall i = 1, \ldots, n$, and this implies $\underline{W}_i = \emptyset, \forall i = 1, \ldots, n$ such that there is no need to compute any down hulls.

Table 7.7 gives the results of the above experiments. The second column gives the number of input points ($|W|$ for standard and $|\overline{V}'| + |\overline{W}|$ for 2-facet), the third column gives the computational time for standard QuickHull in seconds and the fourth column for 2-facet QuickHull. When $n \leq 13$, both methods can compute the local convex hulls very quickly. The time consumptions begin to increase drastically for both methods when $n \geq 14$ and the standard QuickHull failed for $n = 16$–21 while the 2-facet QuickHull failed for $n = 19$–21. The above results show that the 2-facet QuickHull based on a modification of the standard QuickHull can outperform the latter under some certain circumstances such as the instances $n = 16, 17$ as in Table 7.7.

7.5 Conclusions on local convex hull method

The number of outside points in $\overline{W}$ and $\underline{W}$ can be either small or large compared with the main part W'. However, in the real-world example "Instance A" from Cacchiani et al. [15] and the artificial condition (i) where unit type compatibility is retained, which are the

Table 7.7: Results in computing $\mathcal{H}$ by standard and 2-facet QuickHull in $\mathbb{R}^n_+$

n	input point # ($\lvert W\rvert, \lvert\overline{V}'\rvert + \lvert\overline{W}\rvert$)	time (standard)	time (2-facet)
2	14, –[a]	0	–[a]
3	28, 7	0[b]	0[b]
4	47, 12	0[b]	0[b]
5	72, 17	0[b]	0[b]
6	104, 22	0	0[b]
7	144, 27	0	0[b]
8	193, 32	0	0
9	252, 37	0	0
10	280, 42	0.031	0
11	353, 59	0.031	0.015
12	385, 76	0.171	0.124
13	418, 93	1.123	0.811
14	590, 166	43.76	39.19
15	703, 186	165.3	282.9
16	704, 187	980	300.5
17	726, 209	F[c]	1226
18	777, 238	F[c]	15270
19	829, 267	F[c]	F[c]
20	965, 294	F[c]	F[c]
21	1022, 326	F[c]	F[c]

[a] $\overline{W} = \emptyset$, so $\mathcal{H}$ is given by $(a^{-1})^T w = 1$ and $(b^{-1})^T w = 1$ directly.
[b] The option "joggled input" (QJ) [66] was used to avoid precision errors.
[c] Failed due to various reasons. The most common one was reportedly as "QH6082 qhull error (qh_memalloc)" meaning insufficient memory to allocate some data.

two most realistic instances, there are almost no outside point in the simulated scenarios. It is also commonly seen that no such a three-hull division can be made since some types cannot be used alone, especially when the passenger demands get large while the upper bound is set small.

In conclusion, as long as the context is set within train unit scheduling with the parameter settings either the same or modified to increase the computational difficulty, in all the experiments, the local convex hulls can be easily computed by the standard QuickHull algorithm within a very short time. Note that in the UK, and other countries where train units are commonly used in passenger railway, it is very rare for a train operator to have a route or train service that can be served by more than 10 types of unit being all compatible with each other. Due to this reason, we find the standard QuickHull sufficient for computing local convex hulls arising in train unit scheduling problems in most possible cases.

Pragmatically, using Algorithm 6 or Algorithm 7 to speed up the computation process is less necessary for train unit scheduling problems. Nevertheless, there might be other fields employing integer multicommodity flow models where the local convex hull method can also be applied. It is possible that some of them have a far larger number of commodities and combination points at a node or an arc such that the standard QuickHull algorithm may fail or may be less efficient. In those cases, the proposed Algorithm 6 and

Algorithm 7 may be helpful in improving the computational efficiency or the computability. § 7.4.4 gives some examples on the above point. Since this thesis solely deals with train unit scheduling problem, further experiments and discussions on utilising the above algorithms for other possible cases in integer multicommodity flow problems will not be presented here.

Chapter 8

Other investigations

Although the main focus of this thesis is on the connection-arc graph representation based formulations, the associated branch-and-price solver and the relevant computational experiments, in this chapter another two lines of investigations as supplementary directions outside the main line of the thesis are presented for completeness only, including a combination-arc based ILP formulation as an alternative of the connection-arc based formulation and a multidimensional matching model for the post-processing phase.

8.1 Combination-arc graph representation

In § 3.5.3 the idea of a novel modelling strategy called combination-arc graph representation for the network-level framework is given. Here we describe the details of this strategy and compare it with the connection-arc graph representation.

Although in this thesis the modelling strategy and the mathematical formulation based on combination-arc graph representation are going to be formally proposed and described, this representation is only served as an alternative and tentative approach to be further explored in the future. Although we have observed that the ILP formulations based on connection-arc graph representation currently can serve quite well in solving the problem instances that have been tested so far, this new model might be an ideal substitute when the current formulations fail in solving some harder instances. We will also give preliminary discussions on how to solve the ILP formulations based on the combination-arc graph representation.

Since the focus of this thesis has been put onto the connection-arc based formulations and the their solution methods applied to a large number of real-world instances, and also

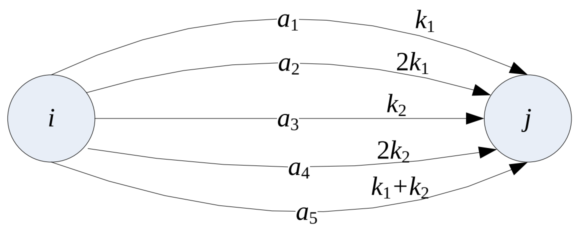


Figure 8.1: A link and its combination arcs

due to the time limit for this research, no computational experiment based on this new modelling strategy will be given. We will leave this to future work.

8.1.1 Modelling of combination-arc graphs

A combination-arc graph is different from a connection-arc graph. This paragraph describes how a combination-arc graph is constructed. First we define between each pair of train nodes i, j, as well as from the source s to all train nodes and from all train nodes to the sink t, a link $l = (i, j)$ such that potential connection relation exists for a unit to serve the two trains consecutively or a unit starts/ends its daily workload. Simply speaking, a link in a combination-arc graph corresponds to an arc in an connection-arc graph. However a link is only conceptual—it does not physically appear in a combination-arc graph and what exist there are the *combination-arcs*. Let $K_{ij} = K_i \cap K_j = \{k_1, k_2, \ldots, k_{|K_{ij}|}\}$ be the unit types available for link $l = (i, j)$. Associated with each link, there is a set of combination-arcs $\{a_1, a_2, \ldots, a_n\}$ each of them representing a block of possible unit combination moving from i to j. A unit combination represented by combination-arc a is given by an integer vector $n_a = (n_a^{k_1}, n_a^{k_2}, \ldots, n_a^{k_{|K_{ij}|}})$, where n_a^k is the number of type k unit in this combination. Note that for train unit scheduling all possible combinations at each link can be easily enumerated in advance since their number is not large.

Example For a link $l = (i, j)$, suppose between train i and j there are two types available as k_1 and k_2, and there are five possible unit combinations flowing from i to j as: $k_1, 2k_1, k_2, 2k_2, k_1 + k_2$ which corresponds to five arcs of l as $\{a_1, \ldots, a_5\}$ such that $n_{a_1} = (1, 0), n_{a_2} = (2, 0), n_{a_3} = (0, 1), n_{a_4} = (0, 2), n_{a_5} = (1, 1)$. See Figure 8.1 for an illustration.

8.1.2 ILP formulation based on combination-arc graph representation

Let $\mathcal{A}$ be the set of all combination-arcs and L be the set of all links. Binary variables $x_a \in \{0, 1\}, \forall a \in l, \forall l \in L$ are used to indicate if a combination-arc a is used or not. There is no need to further distinguish type-graphs and type-arcs, unlike the case in the connection-graph representation. For a link that is used, among all its combination-arcs, one and only one of them can be used. If a link l is not used, then all its arcs will remain $x_a = 0, \forall a \in l$.

The train combination sets and train convex hulls described in § 4.1.2 are irrelevant to how arcs are defined. Therefore the train convex hull method aforementioned for connection-graphs can also be used in combination-graphs by the following simple conversion. For each train j, its convex hull is given by

$$\text{conv}(W_j) = \left\{ w^j \in \mathbb{R}_+^{K_j} \middle| H^j w^j \leq h^j \right\},$$

or written as constraints

$$\sum_{k \in K_j} H^j_{f,k} w^j_k \leq h^j_f, \quad \forall f \in F_j.$$

Let $\delta^k_-(j), \delta^k_+(j) \subset \mathcal{A}$ be the incoming and outgoing combination-arcs associated with type k at node j respectively. The number of units of type k at j can be expressed by

$$w^j_k = \sum_{a \in \delta^k_-(j)} n^k_a x_a = \sum_{a \in \delta^k_+(j)} n^k_a x_a.$$

Then replacing w^j_k by n^k_a and x_a gives the equivalent convex hull constraints for a combination-arc formulation as

$$\sum_{k \in K_j} \sum_{a \in \delta^k_-(j)} H^j_{f,k} n^k_a x_a \leq h^j_f, \quad \forall f \in F_j.$$

The other techniques in satisfying the requirements in the mathematical formulation are similar to those in the connection-graph representation. Therefore, we have the combination-arc ILP formulation (CAF) (8.1) – (8.9) as below. It is possible to also add the overnight balance variables and constraints similar as in the ones from the formulations based on connection-arc graph representation. However in formulation (CAF) they are omitted for ease of presentation.

$$(CAF) \qquad \min \quad C_1 \sum_{k \in K} \sum_{a \in \mathcal{A}^k} n^k_a x_a + C_2 \sum_{a \in \mathcal{A}} x_a; \tag{8.1}$$

$$\text{s. t.} \quad \sum_{a \in \delta^k_-(j)} n^k_a x_a - \sum_{a \in \delta^k_+(j)} n^k_a x_a = 0, \qquad \forall k \in K_j, \forall j \in N; \tag{8.2}$$

$$\sum_{a \in \delta^k_+(s)} n^k_a x_a \leq b^k_0, \qquad \forall k \in K; \tag{8.3}$$

$$\sum_{k \in K_j} \sum_{a \in \delta^k_-(j)} H^j_{f,k} n^k_a x_a \leq h^j_f, \qquad \forall f \in F_j, \forall j \in N; \tag{8.4}$$

$$\sum_{a \in l} x_a \leq 1, \qquad \forall l \in L; \tag{8.5}$$

$$\sum_{a \in \delta_-(j)} x_a = 1, \qquad \forall j \in N_B^-; \tag{8.6}$$

$$\sum_{a \in \delta_+(j)} x_a = 1, \qquad \forall j \in N_B^+; \tag{8.7}$$

$$\tau^D_{\mathrm{arr}(i)} \left(\sum_{a \in \delta_+(i)} x_a - 1 \right) + \quad \tau^C_{\mathrm{dep}(j)} \left(\sum_{a \in \delta_-(j)} x_a - 1 \right) \le e_{ij}, \quad \forall (i,j) \in L^*; \tag{8.8}$$

$$x_a \in \{0,1\}, \forall a \in \mathcal{A}. \tag{8.9}$$

Constraints (8.2) ensure the flow conservation for each type at each train node. Constraints (8.3) restrain the number of units used for each type within the fleet size bound. Constraints (8.4) are the train convex hulls per train. Let $\delta_-(j), \delta_+(j)$ be the incoming and outgoing combination arcs at node j respectively. Constraints (8.5) ensure at most only one arc can be used within a link. Constraints (8.6)–(8.7) forbid coupling/decoupling operations at banned locations, although this can also be realised by banned location branching without these constraints. Constraints (8.8) ensure time allowance is not violated by coupling/decoupling at some links.

A significant feature of (CAF) is that it is no longer an ordinary network flow problem due to the irregular “flow conservation” constraints in (8.2) taking into account that x_a’s are not the real flow amount at a combination arc a but an indicator whether arc a is used or not. This brings new challenges in how to design appropriate solution approaches for it. Noticeably, decomposed appropriately by Dantzig-Wolfe decomposition, the subproblems become the so-called generalized network flow problems. We will give some tentative discussions on possible solution method regarding to (CAF) in § 8.1.4.

8.1.3 Comparisons between connection-arc graph representation and combination-arc graph representation

If one converts the same problem instance to a connection-arc graph and a combination-arc graph, the number of arcs in the former will be less than the latter. However, the latter may have the following advantages over the former for some problem instances.

- If the network-level framework includes abilities to ensure turnround time allowances involving coupling/decoupling and to minimize the number of coupling / decoupling operations (e.g. as in (AF_0), (AF_1) and (AF_2)), then the ILP based on a connection-graph has to be formulated as an integer fixed-charge multicommodity flow problem with a kind of general integer variables and another kind of binary variables, which is known to be difficult to solve. On the other hand, formulation (CAF) is able to realise the same requirements solely by a single type of binary variables.

- If one would like to solve the two relevant arc formulations by Dantzig-Wolfe decomposition, the LP relaxation of the extended formulation given by formulation (CAF)

will be at least the same as or tighter than the LP relaxation from the extend formulation of the equivalent ILP based on connection-arc graph representation. This is because the node-arc incidence matrix of (CAF) is not totally unimodular, such that the integrality property of the subproblems is no longer kept.

Arguably for the second point above, a price for having tighter LP relaxation in the extended formulation is harder subproblems to solve. A network flow problem with a "standard" node-arc incidence matrix that is totally unimodular as the cases in (AF_1) – (AF_3) will generally yield subproblems as shortest path problems or ILPs with integer extreme points in their LP relaxation problems, which are not difficult to solve. As for (CAF), it yields subproblems with "weighted" flow conservation relations or ILPs without the integrality property, which are expected to be more difficult. We will elaborate this theoretical difference in § 8.1.4

Admittedly, the above discussion is very preliminary. It is difficult to make deeper and more insightful comparisons between the two alternatives without further theoretical analysis and computational experiments based on real-world datasets. These will be part of future work.

8.1.4 Solving the LP relaxation of the combination-arc formulation (CAF)

The problem structure of the combination-arc formulation (CAF) significantly differs from the connection-arc based formulations if to be solved by Dantzig-Wolfe decomposition as in § 4.3. This is mainly because the network node-arc incidence matrix is no longer totally unimodular. Since formulation (CAF) has the unique advantage that can realise the requirements on time allowances involving coupling/decoupling and the minimization over the excessive coupling/decoupling with only one set of binary variables, it is worth some exploration on how to solve it. Nevertheless, the solution approach exploration given in this section is only tentative and preliminary and no conclusive result will be given at this stage.

8.1.4.1 Dantzig-Wolfe decomposition on the combination-arc formulation

Here a tentative scheme in applying Dantzig-Wolfe decomposition in solving (CAF) is briefly described. This may not be the best approach for this special problem and investigations on alternative methods such as Langrangian relaxation will be left to the future work.

We will rewrite the formulation (CAF) (8.1) – (8.9) into a more concise form. In order to use Dantzig-Wolfe decomposition, two issues have to be dealt with beforehand.

The first one lies in the Objective (8.1), where the second term $C_2 \sum_{a \in A} x_a$ gives a certain difficulty in decomposing the objective to cater for a decomposition scheme based on unit types. The remedy is to impose an order among all traction types $k_1 \succ k_2 \succ \cdots \succ$

$k_{|K|}$ and for each combination arc a associated with multiple types K_a, its corresponding $C_2 x_a$ in the second term will be combined into the first term with arc a and type k^* where k^* is the most prior one among the types in K_a, i.e. to rewrite $C_2 \sum_{a\in A} x_a$ into $C_2 \sum_{k\in K} \sum_{a\in A(k)} x_a$ where $A(k)$ is the set of arcs whose most prior type is k and combine the two terms into

$$\sum_{k\in K}\left(C_1 \sum_{a\in\mathcal{A}^k} n_a^k x_a + C_2 \sum_{a\in A(k)} x_a\right). \tag{8.10}$$

The second issue is the fact that if the problem is to be decomposed based on types $k \in K$, subproblems will be coupled in having overlapping variables where a combination arc is formed by different types. For instance, for the link shown in Figure 8.1, arc variable on a_5 will appear in the subproblems of both k_1 and k_2. This makes the decomposition invalid. There is research on Dantzig-Wolfe decomposition with overlapped subproblems where the original problem is "staircase-structured", e.g. [46]. However the kind of overlapped subproblems here are more complicated in their non-staircase structures and the methods such as in [46] will likely be unworkable.

Possible remedies for the second issue need further exploration in the future and the following gives two potential candidates:

(i) *Clone variables*

For each arc variable x_a representing a unit combination based on an arc a with more than one types $k_1, \dots, k_n$, replace it with a set of "clone variables" $x_a^{k_1}, \dots, x_a^{k_n}$ with respect to each type $k_1, \dots, k_n$. Then add the constraints (given by a single label below):

$$x_a^{k_1} = \cdots = x_a^{k_n}. \tag{8.11}$$

to the original problem to force all of them to be equal. The above constraints will be regarded as part of the linking constraints in the master problem and their duals passed to the subproblems will force $x_a^{k_1}, \dots, x_a^{k_n}$ to have the same value there. Each clone variable will be grouped into its associated type in the later decomposition stage. Therefore it is possible that the subproblems will no longer share variables.

Some modifications have to be made in (CAF) to accommodate the above scheme, apart from adding the equalisation constraints (8.11). First, in the Objective (8.10), for those variables replaced by the clone variables, only the cloned ones with the most prior type (i.e. k^* given by $k_1 \succ k_2 \succ \cdots \succ k_{|K|}$) will be added to the second term $C_2 \sum_{a\in A(k)} x_a$ where they will be grouped to the very types associated with themselves. Second, in Constraints (8.2)–(8.4), each clone variable will only be present at the block of its own type. Third, in Constraints (8.5)–(8.8) where no type is concerned, for an arc with clone variables, only the one $x_a^{k^*}$ with the most prior type k^* will be present.

(ii) *Sequential subproblems*

Solve the subproblems independently only at the level of unit families. Within each family where variable overlapping occurs, solve the subproblems sequentially by fixing affected variables based on the solutions of previously solved subproblems. Note that the instance feature of ScotRail and Southern Railways shows the fleet will generally be partitioned into a large number of families with a small number of compatible types within each family.

The validity and practicality of this scheme is unknown yet as it cannot guarantee the exact optimality of the subproblems with extra constrained variables given by previous subproblems. Nevertheless, it is not necessary to solve each subproblem to strict optimality at the final iterations of a column generation process as a valid lower bound may still be available from the values of the master problem and the subproblems.

We will only give some possible solution steps with respect to the first point above, i.e. adding clone variables and equalisation constraints. Consider a new concise form of the combination-arc formulation as below with all variables on coupled arcs replaced by their corresponding clone variables (and redefine the set $\mathcal{A}^k, \forall k \in K$ accordingly):

$$\min \quad \sum_{k\in K} c^{kT} x^k \tag{8.12}$$

$$\text{s. t.} \quad \sum_{k\in K} H^k x^k \leq h; \tag{8.13}$$

$$x^k \in X^k, \forall k \in K. \tag{8.14}$$

where (8.12) corresponds to the Objective (8.10), (8.13) corresponds to the linking constraints (8.4) – (8.8) plus the equalisation constraints (8.11) in the modified formulation, and the subsystems X^k are defined for all $k \in K$ as

$$X^k = \left\{ x \in \{0,1\}^{\mathcal{A}^k} \middle| \begin{array}{l} \sum_{a\in\delta^k_-(j)} n^k_a x_a - \sum_{a\in\delta^k_+(j)} n^k_a x_a = 0, \forall j \in \mathcal{N}^k \\ \sum_{a\in\delta^k_+} n^k_a x_a \leq b^k_0 \end{array} \right\}. \tag{8.15}$$

Note the special structure of the network node-arc incidence matrix presented in (8.15), where the flow “conservation” at each node is based on an equation with the sums of weighted incoming and outgoing arcs. This form of network flow problem with the “weighted” flow conservation structure falls into the category of the so-called generalized network flow problem [2] or flows through network with gains [47].

Let M^k be the “weighted” node-arc incidence matrix with respect to X^k such that $X^k = \{x \in \{0,1\}^{A^k} | M^k x \leq b^k_0\}$. Indeed M^k is not totally unimodular which indicates if the decomposition is successful the lower bound given by LP relaxation on the extensive formulation might be tighter than the compact formulation (8.12) – (8.14). Similar to the cases in the connection-arc graph based formulations, X^k is bounded and contains no ray. Let $\{x^p\}_{p\in P^k}$ be the set of feasible points in X^k such that each of them can be expressed in a way as

$$x^k = \sum_{p \in P^k} \lambda_p x^p, \\ \sum_{p \in P^k} \lambda_p = 1, \lambda_p \in \{0,1\}, \forall p \in P^k. \tag{8.16}$$

Technically the above decomposition method, focusing on all the points in the discrete set X^k rather than its convex hull $\text{conv}(X^k)$, is referred to as discretization as opposite to convexification [74]. Nevertheless, since here the variables are all binary, which means the extreme points of $\text{conv}(X^k)$ are the points of X^k per se, the difference between the two methods disappears [8, 74].

Then the subsequent procedure would be very similar to the Dantzig-Wolfe decomposition given in § 4.3. First the original variables x are replaced by the new variables λ via (8.16) into the extended formulation as

$$\min \quad \sum_{k \in K} \sum_{p \in P^k} \left(c^{kT} x^p \right) \lambda_p \tag{8.17}$$

$$\text{s. t.} \quad \sum_{k \in K} \sum_{p \in P^k} (H^k x^p) \lambda_p \leq h; \tag{8.18}$$

$$\sum_{p \in P^k} \lambda_p = 1, \qquad \forall k \in K; \tag{8.19}$$

$$\lambda_p \in \{0,1\}^{\mathcal{A}^k}, \quad \forall k \in K. \tag{8.20}$$

Note that by decomposing via discretization, the integrality requirement on the extensive formulation can be directly realised by (8.20) [74].

Let ϕ, ψ be the dual variable vectors associated with constraints (8.18) and (8.19) respectively. For each $k \in K$, the k-th subproblem in yielding promising columns will be defined as

$$\bar{c}_k^*(\phi, \psi) = \min_{p \in P^k} \{ c^{kT} x^p - \phi^T (H^k x^p) - \psi \} \tag{8.21}$$

$$= \min_{x \in X^k} \{ c^{kT} x - \phi^T (H^k x) - \psi \}. \tag{8.22}$$

8.1.4.2 Solve the subproblem

The subproblem given in (8.21) is an ILP as

$$\min \quad c^{kT} x - \phi^T (H^k x) - \psi \tag{8.23}$$

$$\text{s. t.} \quad \sum_{a \in \delta_+^k} n_a^k x_a \leq b_0^k; \tag{8.24}$$

$$\sum_{a \in \delta^k_-(j)} n^k_a x_a - \sum_{a \in \delta^k_+(j)} n^k_a x_a = 0 \qquad \forall j \in \mathcal{N}^k; \tag{8.25}$$

$$x_a \in \{0,1\}, \quad \forall a \in \mathcal{A}^k. \tag{8.26}$$

The subproblem (8.23) – (8.26) is strongly related to the generalized flow problems. A detailed introduction on generalized flow problems can be found in [2] with extensive literature resources. No detailed solution approach on the subproblem will discussed in this thesis.

8.1.4.3 Some discussions on (CAF)

The uncommon network structure and problem definition make (CAF) a unique network flow problem requiring special solution approaches. To avoid overlapped subproblems, two remedies have been given on a tentative and exploratory basis. As mentioned before, the use of the clone variables may avoid the possible variable overlapping in the subproblems, but its workability is currently unknown without computational experiments. This is because the numbers of additional variables and constraints to be added can be huge, which are proportional to the number of links. Nevertheless, theoretically it will guarantee the optimality of the global solution when no subproblem can give promising candidate column to the RMP. The second remedy by solving the subproblems sequentially within families, does not increase the problem size but it cannot guarantee the optimality of all subproblems and requires efficient solution approaches for the subproblems, which can be difficult to design.

Anyway, all of the above analyses on the combination-arc formulation (CAF) are very preliminary and need subsequent investigations in the future research. As the focus of the current research is the formulations based on the connection-arc graph representation, as well as the limited timescale and research resources, further theoretical and computational exploration on the combination-arc graph representation and the relevant development on solution algorithms will be left to future work.

8.2 A matching model for post-processing

In this part a matching model for automated post-processing developed at the early stage of the research will be presented. As the study on train unit scheduling proceeds, it is realised that this model may be over-sophisticated in dealing with the local issues at stations which can be tackled in other simpler methods. Nevertheless, for completeness we present this model which has no chance to be experimented with real-world datasets.

For simplicity, in this part the network-level framework is referred to as Phase-1 and the station-level post-processing is referred to as Phase-2.

8.2.1 Station element representation and model description

8.2.1.1 Arrival and departure units

Phase-1 provides a tentative schedule based on which Phase-2 operates within the scope of each individual station $\sigma \in \Sigma$, where the main contribution for Phase-1 would be to fix unit combinations for each arrival and departure train in terms of how many units of which type would be assigned to it. For example, "train 1E08 is served by one 171/7 unit and one 171/8 unit". We can thus define a set of *arrival units* and a set of *departure units* for a day's operation at each station σ, as $u_1, u_2, \dots, u_n \in U^\sigma$ and $v_1, v_2, \dots, v_n \in V^\sigma$ respectively. The Phase-1 framework also results in a tentative matching from an arrival unit to a departure unit as its next task. The purpose of Phase-2 is to modify these matching assignments given by Phase-1 due to operational conflicts at the station. Here we assume empty-running trains have been inserted by Phase-1 and $|U^\sigma| = |V^\sigma| = n$. Sign-on/off trains from/to a home depot can be dealt with by slight adaptation. Exceptions, such as overnight units remaining at the station cannot invalidate this model as sign-on/off arcs regard this station as the source/sink node. For simplicity, we will omit the superscript σ in later part of this section as everything is clearly based on a single station, except where explicitly stating the station is necessary.

The attributes of each arrival (departure) unit u (v) include:

- The train the unit belongs to: $T(u), T(v)$. Notice that an arrival unit only belongs to one arrival train and a departure unit only belongs to one departure train.
- The attributes due to its train: the arrival platform (if known) $P(u)$ and the departure platform (if known) $P(v)$; the arrival time $\tau(u)$ and the departure time $\tau(v)$.
- The unit type: $k(u), k(v)$.

A *matching* between an arrival unit and a departure unit is used to represent a connection relation as in the DAG. A matching between a pair of arrival and departure units u and v indicates the two trains they belong to will contain the same unit. The first-step rule of how arrival unit u can be matched with departure unit v is: (a) Same type: $k(u) = k(v)$; (b) There is a connection arc between $T(u)$ and $T(v)$ in Phase-1 DAG, i.e. $(T(u), T(v)) \in A$. This connection arc is called a *dominant arc* of the matching pair. Other connection validity rules regarding shunting/coupling/decoupling times will be considered respectively in other ways.

8.2.1.2 Unit position

As aforementioned, if a train is formed by coupled units, the positions of the units should be considered. Since Phase-1 has not provided anything on unit permutation for coupled

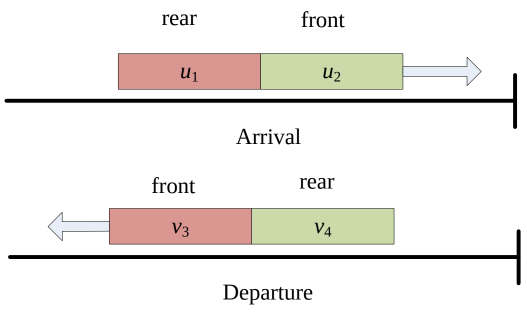


Figure 8.2: A matching in a dead-end platform scenario

trains, it has to be decided in Phase-2. Let $p \in P_u, q \in Q_v$ be the positions and their possible choice sets for arrival unit u and departure unit v respectively. For a unit u (or v) from a non-coupled train, its position is trivially unique ($|P_u| = 1$ or $|Q_v| = 1$); for a coupled single-type train, unit positions are flexible due to the interchangeability, meaning the position specification is also trivial (it can be assumed that $|P_u| = 1$ or $|Q_v| = 1$); for a coupled multi-type train, a specification on p or q has to be made from P_u or Q_v which contain multiple candidates. In conclusion, the above matching is now further modified into between a *positioned* pair of (u, p) and (v, q) such that a plan $(u, p, v, q), u \in U, p \in P_u, v \in V, q \in Q_v$ has to be made.

Unit permutation can significantly affect a schedule without infrastructure details. Figure 8.2 shows two arrival units u_1, u_2 coupled in a train arriving at a dead-end platform, also two departure units v_3, v_4 coupled in a later departure train at the same platform suitable to be linked with the arrival train. Suppose all the units have the same traction type and their positions have been pre-assigned as indicated in the figure. Without unit position and platform layout knowledge, it seems that any arrival units $u_i, i = 1, 2$ can be matched with any departure units $v_j, j = 3, 4$. But in fact u_1 can only be matched with v_3 and u_2 with v_4. Figure 8.3 gives another example concerning arrival units u_1, u_2 and departure units v_3, v_4, suppose they all have the same traction type and are at the same FIFO platform. Suppose their positions have been pre-assigned as indicated in the figure. Moreover, the train $T(u_1) \neq T(u_2)$ but $T(v_3) = T(v_4)$ and the arrival and departure times are appropriate such that the two arrival units can be coupled to serve the departure train. If the arrival train $T(u_2)$ comes earlier than $T(u_1)$ in a FIFO track, then u_1 can only be coupled at the rear of u_2, which indicates u_1 can only be matched with v_3 and u_2 with v_4. Cases in coupled multi-type trains will be more complicated.

8.2.1.3 Parking berths

Generally speaking, there are three possible kinds of shunting plans at each connection arc: to let the unit continue to serve the next train within the platform area (short time gap connection), to put the unit to a siding temporarily until it is needed for the next

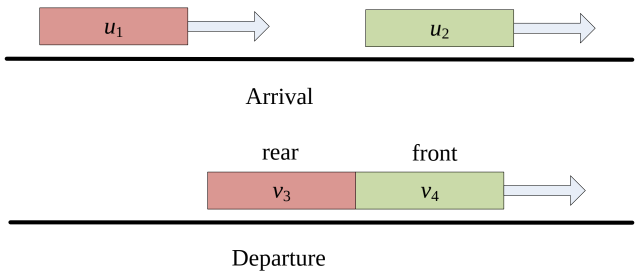


Figure 8.3: A matching with coupling in a FIFO scenario

train (medium time gap connection) and to put the unit to a nearby depot until it is needed for the next train (long time gap connection). Here we denote *parking berths* as $b \in B := \{0\} \cup S \cup D$ where 0 refers to a dummy parking berth representing platform stay, S the siding set and D the depot set.

For a positioned pair of arrival and departure units $(u, p, v, q), u \in U, p \in P_u, v \in V, q \in Q_v$, the possible choice for parking berth types (dummy, siding or depot) is dependent on the relevant time gap length of its dominant connection arc $(T(u), T(v))$. A dummy berth is only imaginary, i.e. the unit stays put. However, if the parking berth turns out to be a siding or a depot, usually there are a number of sidings or depots available for a unit to be shunted to. We denote the possible parking berths for a pair of positioned units (u, p, v, q) as $B(u, p, v, q)$.

8.2.1.4 Parking methods

In order to avoid blockage, the ways a unit comes to/leaves a platform or a siding should be taken into account. It is usually not important how a unit comes to/leaves a depot after an empty-running trip. This is because such movements are usually infrequent with sufficient time for resolving any conflicts that might arise.

Here we define the parking method $m \in M(u, p, v, q, b)$ for a pair of positioned units (u, p, v, q) with its parking berth $b \in B(u, p, v, q)$ as possible ways of how this unit arrives at and leaves its arrival platform $P(u)$, comes to and leaves its parking berth b, and then comes to and departs from the departure platform $P(v)$. For a pair yielding a dummy or a depot parking berth $b \in \{0\} \cup D$, the ways coming to and leaving b is omitted as a default "00".

Here is an example of such a parking method with $b = 0$ and a null-shunting: $m = [LR, 00, LR]$, which means arriving from the left and departing from the right, at the same platform without any shunting. Another example of a parking method involving a double (free) siding shunting with re-platforming: $m = [LR, RL, LL]$ which means:

- at the arrival platform: arriving from the left and leaving from the right (to a siding);
- at the siding: coming from the right and leaving from the left (to the departure

platform);

- at the departure platform: coming from the left and departing from the left.

Parking methods are determined by various factors, e.g. train directions, platforms, unit positions, sidings, depots and overall layout relations, plus possible engineering and operational regulations. Usually when (u, p, v, q, b) is fixed, the resulting $M(u, p, v, q, b)$ would have a very small number of candidate methods. Especially when the parking berth is a dummy one, a single (FILO) siding or a depot, the possible parking method is very likely to be unique. Note that in a realistic schedule, most of the units will use a short time gap between two trains, meaning the parking berth is almost certain to be a dummy one with a unique parking method as the approaching directions on platforms are also almost uniquely determined by the trains' directions.

8.2.1.5 Shunting plans and conflicts between a pair of shunting plans

The Phase-2 shunting problem is defined as

At each station $\sigma \in \Sigma$ of the rail network, give each arrival unit $u \in U$ with position $p \in P_u$ and assign it to a departure unit $v \in V$ with position $q \in Q_v$, via a parking berth $b \in B(u, p, v, q)$ and with a parking method $m \in M(u, p, v, q, b)$ such that

(i) Each matching pair (u, v) implies a dominant connection arc relation $(T(u), T(v)) \in A$;

(ii) Each positioned pair (u, p, v, q) is operationally feasible for connecting $T(u)$ and $T(v)$;

(iii) No unit blockage occurs at platforms or sidings;

(iv) No overcapacity occur at sidings or depots;

(v) No breaking on any other temporal, spatial, engineering or operational rules that make the overall plan inoperable.

A *shunting plan* can be expressed as a 6-tuple (u, p, v, q, b, m), $u \in U$, $v \in V$, $(T(u), T(v)) \in A$, $p \in P_u$, $q \in Q_v$, $b \in B(u, p, v, q)$, $m \in M(u, p, v, q, b)$ representing a *self-consistent* shunting plan. For convenience, a shunting plan (u, p, v, q, b, m) is also written as $\pi \in \Pi$ when its details can be omitted, where we denote the set of all possible shunting plans at a station as Π. The set of shunting plans all involving a specific arrival unit u can be written as Π_u, the same for Π_v for a specific departure unit v. Figure 8.4 gives an illustration of the concept of shunting plans (unit positioning is not depicted). For instance, the arrival unit u_1 does not require a shunting movement to a siding or depot, therefore it stays at the platform (a null shunting 0) where it arrives from the left (L), and it departs at the same platform from the left as the departure unit v_1.

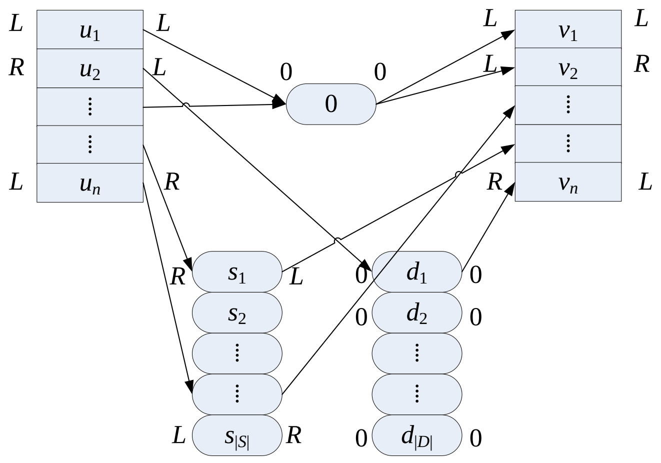


Figure 8.4: An illustration of shunting plans

All the possible shunting plans can be pre-computed based on input data of station information and the first phase results before the second phase begins. With a shunting plan (u, p, v, q, b, m) timings can be determined for the unit leaving the arrival platform, coming to the parking berth, leaving the parking berth and coming to the departure platform. Also, the following parameters are known from station layout knowledge. $\Delta\tau_u$: time duration at the arrival platform $P(u)$; $\Delta\tau_{ub}$: shunting/empty-running time from platform $P(u)$ to parking berth b; $\Delta\tau_{bv}$: shunting/empty-running time from parking berth b to platform $P(v)$; $\Delta\tau_v$: time duration at the departure platform $P(v)$. Then, the attributes of a shunting plan (u, p, v, q, b, m) include:

- All attributes inherited from its arrival and departure units, including traction types, trains, positions and platforms;
- At the arrival platform $P(u)$: arrival time $\tau(u)$, leaving time $\lambda(u) := \tau(u) + \Delta\tau_u$, which forms a time window at the arrival platform: $W_u(u, p, v, q, b, m) := [\tau(u), \lambda(u)]$;
- At the departure platform $P(v)$: coming time: $\chi(v) := \tau(v) - \Delta\tau_v$, departure time $\tau(v)$, which forms a time window at the departure platform: $W_v(u, p, v, q, b, m) := [\chi(v), \tau(v)]$;
- At parking berth b: coming time $\chi(b) := \lambda(u) + \Delta\tau_{ub}$, leaving time $\lambda(b) := \chi(v) - \Delta\tau_{bv}$, which forms a time window at the parking berth: $W_b(u, p, v, q, b, m) := [\chi(b), \lambda(b)]$.

Although a shunting plan may be self-consistent in operation, the interaction between shunting plans may still make the overall schedule inoperable. We denote a *conflict* relation between two shunting plans π and π' as $\pi \dagger \pi'$. Due to the deterministic feature, conflict relations between every pair of fixed shunting plans can also be pre-computed as input data for the second phase. We use

$$\varPi^{\dagger}(\pi) = \{\pi' | \pi' \dagger \pi, \forall \pi' \in \varPi\}$$

to denote the set of all shunting plans that have a conflict relation with shunting plan π and

$$C = \{(\pi, \pi') | \pi \dagger \pi', \forall \pi, \pi' \in \varPi\}$$

the set of all conflicting shunting plan pairs.

8.2.2 Model formulation

In this part we assume that the path formulation (PF_3') is used as the Phase-1 ILP model. If the unit combination of each train derived from the Phase-1 result remains unchanged, the fleet size and some other terms in the objective function (4.32) will remain unchanged even if Phase-2 modifies the Phase-1 results by resetting some matching pairs. Phase-2 thus can be regarded as re-matching arrival units with departure units at each station, or from a diagram's point of view, swapping trains between the diagrams derived from the first phase. Therefore the global optimality given by the first phase will be partly preserved.

We denote the set of all dominant arcs pertaining to station σ as A^{σ}. As for each valid shunting plan (u, p, v, q, b, m), $(T(u), T(v))$ corresponds to a unique dominant arc $a \in A$, we thus denote the arc corresponding to shunting plan π as $a(\pi)$, i.e. if $\pi = (u, p, v, q, b, m)$, then $a(\pi) = (T(u), T(v)) \in A^{\sigma}$. According to which formulation is used, the parameter settings for Phase-2 will vary.

If a formulation containing the block-arc variables like (AF_1) is employed in Phase-1, then we can assume that most of the arcs used are appropriate matchings from the point of view of Phase-2, especially in minimizing used block-arcs and ensuring time allowances from coupling/decoupling. Then for the arc variables given by the Phase-1 results $x_a = \sum_{p \in P_a^k} x_p, \forall a \in A^k, k \in K$, it is possible to assign costs c_{π} to each shunting plan $\pi \in \varPi$ according to the following rule:

(i) If first phase result $x_{a(\pi)} > 0$, then $c_{\pi} = 0$;

(ii) If first phase result $x_{a(\pi)} = 0$, then $c_{\pi} > 0$ and can be set according to some preferences (e.g. traction type / parking berth).

By setting the above shunting plan costs c_{π} and minimizing $\sum_{\pi \in \varPi} c_{\pi} x_{\pi}$, where $x_{\pi} \in \{0, 1\}$ indicates whether to use a shunting plan $\pi \in \varPi$, it ensures that if the part of the first phase result related to station σ does not cause any conflicts at all, it will be exactly retained after the second phase.

On the other hand, if a reduced version like (PF_3') is used in Phase-1, since excessive coupling/decoupling has not been eliminated and there may also be violated time allowances due to coupling/decoupling, then it is very likely that lots of the arcs used are matching pairs of bad quality in the perspective of Phase-2, and in this case the contribution of Phase-1 will be only in providing a globally optimized unit resource distribution in terms of unit combination assignments for all trains. Therefore, the costs of each shunting plan c_π will be set regardless of the Phase-1 arc solution values. The costs can be set, for instance, by the dwell time during the connection (the shorter the better) as well as some other preferences like in encouraging specific long gap insertion for maintenance.

Similar to [49], it is desirable to have each siding serving as few types of unit as possible, yielding more flexibility and robustness. There is no such need for platforms or depots. We denote $y_b^k, b \in S, k \in K$ as a binary variable indicating whether at least one unit of type k is parked at siding b, and try to minimize $\sum_{k\in K}\sum_{b\in S} y_b^k$.

Similar to the first phase, binary blockflow variables $z_a, a \in A^\sigma$ indicating whether arc a is used is introduced, which are used for forbidding coupling/decoupling at banned locations, calculating coupling/decoupling time durations at permitted locations and eliminating unnecessary attaching/detaching operations. We denote the set of all shunting plans pertaining to arc $e \in A^\sigma$ as Π_e, i.e. $\Pi_e := \{\pi | a(\pi) = e\}$. Then it is expected that $z_a = \begin{cases} 1, & \text{if } \sum_{\pi\in\Pi_a} x_\pi > 0 \\ 0, & \text{if } \sum_{\pi\in\Pi_a} x_\pi = 0 \end{cases}, \forall a \in A^\sigma.$

The second phase model reads

$$\min_{x,y,z} \quad W_1 \sum_{\pi\in\Pi} c_\pi x_\pi + W_2 \sum_{k\in K}\sum_{b\in S} y_b^k + W_3 \sum_{a\in A^\sigma} z_a \tag{8.27}$$

subject to

$$\sum_{\pi\in\Pi_u} x_\pi = 1, \quad \forall u \in U; \tag{8.28}$$

$$\sum_{\pi\in\Pi_v} x_\pi = 1, \quad \forall v \in V; \tag{8.29}$$

$$x_\pi \le y_{b(\pi)}^{k(\pi)}, \quad \forall \pi \in \Pi : b(\pi) \in S; \tag{8.30}$$

$$x_\pi \le z_{a(\pi)}, \quad \forall \pi \in \Pi; \tag{8.31}$$

$$\sum_{a\in\delta_-(j)} z_a = 1, \quad \forall j \in N^\sigma; \tag{8.32a}$$

$$\sum_{a\in\delta_+(j)} z_a = 1, \quad \forall j \in N^\sigma; \tag{8.32b}$$

$$\tau^D_{\mathrm{arr}(i)}\left(\sum_{a\in\delta^\sigma_+(i)} z_a - 1\right) + \tau^C_{\mathrm{dep}(j)}\left(\sum_{a\in\delta^\sigma_-(j)} z_a - 1\right) \le e_{ij}, \quad \forall (i,j)\in A^\sigma; \tag{8.33}$$

$$x_\pi + \frac{1}{|\varPi^\dagger(\pi)|}\sum_{\pi'\in\varPi^\dagger(\pi)} x_{\pi'} \le 1, \quad \forall \pi\in\varPi : \varPi^\dagger(\pi)\neq\emptyset; \tag{8.34}$$

$$l_{k(\pi)}x_\pi + \sum_{\pi' : W_b(\pi')\cap W_b(\pi)\neq\emptyset} l_{k(\pi')}x_{\pi'} \le L_{b(\pi)}, \quad \forall\pi\in\varPi : b(\pi)\in S\cup D; \tag{8.35}$$

$$x_\pi \in\{0,1\}, \quad \forall\pi\in\varPi; \tag{8.36a}$$

$$y^k_b \in\mathbb{R}_+, \quad \forall b\in S, \forall k\in K; \tag{8.36b}$$

$$z_a\in\mathbb{R}_+, \quad \forall a\in A^\sigma. \tag{8.36c}$$

The first term in the objective (8.27) minimizes the cost of the shunting plans selected; the second term encourages units of the same type to be grouped together as much as possible; the third term minimizes the total number of unit blocks such that unnecessary coupling/decoupling activities are also reduced. Generally, the first term should be given a higher weight (W_1) than others (W_2 and W_3).

Constraints (8.28)–(8.29) match each arrival unit to a unique departure unit via a unique parking berth by a unique parking method, and the same for each departure unit. A similar concept for ensuring each position in coupled multi-type trains to be occupied by one and only one unit has been encapsulated implicitly by conflict sets.

Constraints (8.30) calculate the siding-type variables, where $k(\pi)$ is the traction type implied by shunting plan π and $b(\pi)$ is the parking berth of plan π. Notice here we use a disaggregated formulation with many constraints ($= |\varPi_{b(\pi)\in S}|$) rather than an aggregated formulation with less constraints, e.g. a multiple variable lower bound (MVLB) ([23]) like

$$\frac{1}{|\varPi^k_b|}\sum_{\pi\in\varPi^k_b} x_\pi \le y^k_b, \quad \forall b\in S, \forall k\in K, \tag{8.37}$$

where $\varPi^k_b$ is the set of shunting plans implying a type k to be shunted to a berth b. The stronger disaggregated formulation (8.30) will generally give a much tighter LP-relaxation than (8.37), and also let y^k_b be only continuous while functioning as binary. The side effect from excessive number of constraints in (8.30) can be neutralised if a column-and-dependent-row generation approach is used.

Constraints (8.31) calculate the blockflow variables for each dominant arc pertaining to station σ. Similar to (8.30), we use a strong formulation rather than an MVLB formulation.

Only one from Constraints (8.32) and (8.33) will appear for a specific station σ, depending on whether coupling/decoupling is banned at σ (use (8.32)) or allowed (use (8.33)). N^{σ} is the set of train nodes related with station σ.

Constraints (8.34) prohibit conflicts between each pair of shunting plans. Note that a generic conflict (or crossing) exclusion formulation such as

$$x_{\pi} + x_{\pi'} \leq 1, \quad \forall(\pi, \pi') \in C \tag{8.38}$$

will give a huge number of constraints leading to computational difficulty (e.g. Model 1 in [49]). Some successful remedy methods, for instance, taking advantage of clique inequalities or comparability graphs (e.g. Model 2 and 2a in [49]) are not suitable for the current problem, where the concept of conflict is based on a more complex context concerning positions, platforms, sidings and combined parking methods. Here we propose another remedy method as in (8.34). Compared with (8.38), the number of constraints has been significantly reduced, as usually $|C| \gg |\mathit{\Pi}|$. Notice the number of constraints will be further reduced if a column-and-dependent-row generation approach is used.

Constraints (8.35) prohibit overcapacity for each siding and depot berth, where $l_{k(\pi)}$ is the capacity consumed by type $k(\pi)$ implied from shunting plan π, and $L_{b(\pi)}$ is the total capacity of parking berth b implied from plan π. From observations of the manual scheduling process, feasible berthing plans can always be found in a station-by-station manner. Moreover, berth capacity would have been satisfied at the preceding timetabling stage, i.e. the input timetable determines the main unit flows on stations at certain times and places, as train unit scheduling only assigns units to predetermined trains.

Finally (8.36) gives variable domains.

Chapter 9

Conclusions and future work

9.1 Conclusions

In this thesis we have studied the optimisation methods for the train unit scheduling problem (TUSP) particularly for the UK railway industry. To the best of the author's knowledge, the studies on the TUSP have been very scarce in the literature. There are researches from the NSR-related group in the Netherlands on a related train unit circulation problems which however has different requirements and problem settings from the TUSP in this thesis. The train unit assignment problem studied by the DEIS group in Italy has some similarities as the TUSP in this thesis, but some major operational requirements either commonly seen in general railway operations or typical in the UK railway industry have not been considered in their work. Moreover, all their solution approaches are based on heuristics. Before this thesis, there is no study on the TUSP especially for solving the problem instances arising in the UK railway industry. As for the application aspect, there is no known successful commercial software for automatically optimising train unit scheduling in the world as far as the author is aware, in contrast with bus/train crew scheduling and flight scheduling.

The research has been carried out with close collaboration with First ScotRail and Southern Railway, two of the major train operators in the UK, from whom valuable real-world datasets and experience of decades have been obtained and benefited the research. Tracsis plc, a University of Leeds spin-out company specialising in railway-related software products has also provided important station infrastructure datasets of the ScotRail network and the interactive unit diagramming software TRACS-RS that has been used for post-processing.

The main contributions of this research are summarised in the following.

9.1.1 Problem description

Since no previous literature is available for the specific problem in the UK, detailed description on the TUSP typical in the UK is given in this thesis. Some real-world problem features that the methods proposed in previous literature have not considered are highlighted and tackled by this thesis, including time allowance for coupling/decoupling operations, unit type-type compatibility relations, combination-specific coupling upper bounds, locations banned or restricted for coupling/decoupling operations, excessive/redundant coupling/decoupling and unit blockage at stations. Some more complicated issues such as overnight balance, maintenance, objectives and solution qualities are also discussed.

9.1.2 Modelling

Due to the complexity and sophistication of the TUSP, it is realised that the solution approach is better to be divided into two levels: a network-level model that can capture the essence of the rail network and allocate the optimum amount of train unit resources to each train globally to ensure the overall optimality, and a station-level process (post-processing) that can resolve the remaining local issues such as unit blockage. Some of the tasks can be either solved centrally in the network-level model or locally by the station-level process, such as time allowance for coupling/decoupling and removal of excessive/redundant coupling/decoupling operations.

An integer fixed-charge multicommodity flow model for the TUSP is proposed, which is novel and can handle all the real-world requirements in the UK except unit blockage. Since the model's intuitive knapsack constraints have some disadvantages and may not handle combination-specific coupling upper bound requirements, a local convex hull method is used such that for each train possible unit combinations are enumerated into a set and its convex hull is computed directly. The above model becomes very difficult to solve when the problem size is medium or large, due to its fixed-charge nature. Realising that the banned location requirement can be solely satisfied by branching, as well as in most cases the tasks of removing excessive/redundant coupling/decoupling and ensuring time allowance for coupling/decoupling can be sufficiently handled locally in the station-level process, the integer fixed-charge multicommodity flow model is then reformulated into a simpler integer multicommodity flow model that is much easier to solve. Sometimes the practitioner may need unit overnight balance to be ensured after diagrams are formed and the ability to flexibly consider overnight balancing is added to the network-level model. An alternative combination-arc model is also proposed. This arc-based novel model can realise all requirements as the integer fixed-charge multicommodity flow model does but with only one set of binary variables. Its network incidence matrix is no longer unimodular, which brings some interesting theoretical issues to be further explored in future.

9.1.3 Solution approaches

The network-level integer multicommodity flow model is solved exactly given sufficient computational time by branch-and-price. Note that the closest models with instances of similar sizes (at most 500–600 trains) in previous literatures are all solved by heuristics [15].

Theoretical analysis on applying Dantzig-Wolfe decomposition to the model is given in detail, which is not commonly seen in previous researches. A customised branch-and-price ILP solver is developed including the following original features:

- A branching system with multiple branching rules. Especially two customised branching rules as train-family branching and banned location branching are proposed. They not only can be used to satisfy the type-type compatibility and banned location requirements, but also often help in improving the efficiency in the branch-and-price process.

- The order of the branching rules and its controlling strategies as static or dynamic (including "switch@new-rule" and "switch@stagnation").

- An efficient and flexible node selection method. Features include initialisation, jumps, dives, extended-best node determination and so on.

- Column inheritance from a parent node to a child node in conjunction with naive columns and a column filter option.

- Estimated upper bounds and node reservation.

9.1.4 Experiments

Comprehensive experiments have been carried out mainly based on real-world instances on the GA and GB areas from the ScotRail network. The solutions often outperform the manual diagrams in many aspects and the practitioners from ScotRail are satisfied. In particular, some experiment results have already been involved in the decision-making process of train unit diagramming for the new May 2014 timetable at ScotRail.

Moreover, regarding the parameter settings of the branch-and-price solver, a series of experiments on fine-tuning the solver have been conducted and the results are reported. Issues like solution qualities and ideal demand estimation have also been experimented with and discussed.

Post-processing using TRACS-RS to resolve the station-level issues like unit blockage and excessive/redundant coupling/decoupling is also proved to be successful for the instances tested.

9.1.5 Local convex hulls

Some studies on the local convex hull method have been conducted. As a method for a class of integer multicommodity flow problems with not too large a number and amount of commodities available for each node, the local convex hull method is generalised and discussed as a generic method for the first time to the best of the author's knowledge. The method has been used in several researches in rolling stock scheduling since 1990s. Its major characteristic is to use enumeration to capture all the possible combinations at a node with different commodities and the convex hull of such combinations will be computed explicitly if possible. With the nonzero facets of the local convex hulls, a conversion makes the combination variables (based on flow amount per node per commodity) to the original network flow variables (based paths or arcs) and thus the facets will take their effects as valid inequalities for the original problem. This method can be used to strengthen the LP relaxation as well as satisfying complex and difficult restrictions such as combination-specific coupling upper bounds.

One contribution of this thesis is a preliminary exploration of the structure of local convex hulls for the kind of instances where every commodity can be solely used in satisfying the node, making the relevant convex hull divided into a main hull, an up hull and a down hull. The research shows that the number of nonzero facets of a main hull may never be greater than 2. Issues like numbers of outside points and feasibility range in computing local convex hulls directly by the standard QuickHull algorithm subject to different dimensions and commodity parameters have also been studied. It is found that, by the empirical experiments based on real-world and artificial instances, the number of points in a combination set are generally within the range of QuickHull's capability in the context of train unit scheduling without dividing the convex hull. Finally, the "2-facet" QuichHull can be more advantageous than the standard QuickHull under certain circumstances as shown by the experiments.

9.2 Future work

Although the work of this thesis has achieved some results that are novel and can solve some real-world problems, the research on the optimisation on train unit scheduling in the UK is still far from being mature. A number of issues are worth exploring.

This PhD research on train unit scheduling optimisation will be further investigated in a new project supported by EPSRC EP/M007243/1 in collaboration with ScotRail and Tracsis and possibly other train operating companies. The project will focus on two aspects in developing a powerful automatic train unit scheduling solver, i.e. a core ILP solver and a hybridisation scheme "Size Limited Iterative Method" (SLIM) (cf. § 9.2.2).

The major targets of the new project on train unit scheduling optimisation are highlighted in the following.

9.2.1 Enhancing the power of the branch-and-price solver

The ILP branch-and-price solver now can solve problem instances with 200–500 trains, representing small to medium sized cases in the UK, from minutes to hours. In the future, we would like the solver to solve a problem size of around 500 trains in less than 30 minutes.

Ideally, the integer fixed-charge multicommodity flow model will be solved, although now it appears to be only sufficient to solve the simpler integer multicommodity flow model in the network-level process, leaving the ignored requirements to the local process. The overall optimality may be further improved by solving the fixed-charge model compared with the reduced versions as (AF_3).

Although the local convex hulls are computed explicitly, the LP relaxation may be further strengthened by other kinds of valid inequalities, which will give a branch-and-price-and-cut method. Dantzig-Wolfe decomposition with LP relaxation is used in this thesis to solve the network-level ILP. As an alternative approach, Lagrangian relaxation may be worth trying. When it is sufficient to only use a single type to cover some services, a minimum-cost flow method may be employed, which would be much more efficient than the current ILP. Note that although the path variables can be automatically integral due to the polynomial-time minimum-cost flow method, a BB tree is still needed for banned location branching. The minimum-cost flow can be used as a complementary component when it is detected that only a single type is being used.

Customised branching strategies taking advantage of problem features would be key to successfully finding integer/operable solutions within reasonable time. The current research has achieved some success in using the type-type compatibility relations and banned locations for coupling/decoupling. More features could be explored in the future. One such feature is the possibility of branching on the routes and/or “obvious” connections.

The future work on the local convex hulls will be focused on the following aspects. First, more rigorous theoretical investigation on the polyhedral combinatorics side will be conducted. More computational experiments will be carried out on the customised QuickHull algorithms. Finally we would like to further design an alternative convex hull computation method by computing sub-hulls with respect to subsets of compatible commodities and then using a final wrapping over all sub-hulls.

Due to the time limit, the combination-arc model has not been fully investigated. As an arc-based formulation, this formulation is unique in its non-unimodular node-arc incidence matrix, which has the advantage of possibly tighter LP relaxations but the disadvantage of harder subproblems.

9.2.2 Hybridisation

A step-change in transforming the outcomes of this thesis would be having the ability to solve problem instances with a huge number of trains. ScotRail has a daily timetable of around 2200 trains but they are divided into three relatively independent areas. Southern Railway, on the other hand, has a daily timetable of around 2400 trains that are heavily overlapped in routes and unit types. This scale of such huge instances may be far beyond an exact ILP solver's ability to handle. It is commonly regarded that combining the exact ILP solver with some heuristics would be a better alternative.

Based on the successful experiences from the development of PowerSolver for huge-scale crew scheduling [52, 53], the rough idea of a possible hybridisation method has already been outlined, which is provisionally named SLIM ("Size Limited Iterative Method"). SLIM will consist of two main components—(i) an ILP solver for solving the network-level multicommodity flow model and (ii) an iterative heuristic controller.

The ILP solver needs to be mature enough such that it can solve problem instances of around 500 trains within less than half an hour comfortably and robustly. The controller is iterative such that in each iteration the original huge problem will be reduced to a suitably small size such that the ILP solver can be applied. The reduction will preserve the essence and characteristics of the original problem while allowing the reduced instance to be solved easily by the ILP solver. The specific compression method is not trivial and will need some major research effort. For instance, for some connection links that are obviously reasonable to use, the controller will pre-sequence them such that the number of "trains" can be significantly reduced. The controller should also abandon some pre-sequenced connections in previous iterations when those links are regarded to be not contributing to optimality convergence. After each iteration some connections will be pre-sequenced and some abandoned and a mechanism is enforced to make sure the current solution will never be worse than the previous one. Ideally the overall compressed instance will thus converge to a state that is very close to the optimality at the final iterations.

SLIM will also be suitable for parallelisation, based on the experience from PowerSolver. Moreover, traditional integer linear programming solvers may suffer from run-time failures due to numerical instability or difficulties. Since SLIM can run in many iterations in conjunction with parallelisation in solving many different compressed instances, it can afford to skip some iterations that have encountered computational difficulties.

9.2.3 Automatic post-processing

Current post-processing is manual with the help of the interactive rolling stock scheduling software TRACS-RS. There is a need to automate this process, given that the problem scales may get larger and it may not be realistic to handle it manually.

At the initial stage of this research a multidimensional matching mixed-integer linear programming model has been briefly explored to resolve local issues on a station-by-station basis, as given in § 8.2. This matching model directly takes the results from the network-level multicommodity model and tries to resolve local problems such as unit blockage and excessive/redundant coupling/decoupling. This two-phase approach would have the advantage of not over-complicating and computationally over-burdening the network-level model. However, collecting relevant datasets for all stations of ScotRail is a massive amount of work that would not be feasible to undertake within the timescale of this research. Therefore no experiment has been carried out for this matching model. Moreover, as the study on local-level processing progressed, it was realised that the above matching model may be over-sophisticated in dealing with the local issues which are in fact much simpler than it was thought to be. Regarding automating the post-processing phase, a complicated optimisation model may not be actually needed if some simple heuristics can handle it.

For a comprehensive train unit scheduling system, the ILP solver plus SLIM would be the core engines, the automatic post-processing component would be a secondary complement, and an interactive subsystem such as TRACS-RS to aid the planners to rectify any other local issues would ensure that the results can be implemented in real practice.

Bibliography

[1] Erwin Abbink, Bianca van den Berg, Leo Kroon, and Marc Salomon. Allocation of railway rolling stock for passenger trains. *Transportation Science*, 38(1):33–41, 2004.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms and applications.* Prentice Hall, Englewoods Cliffs, USA, 1993.

[3] Arianna Alfieri, Rutger Groot, Leo G. Kroon, and Alexander Schrijver. Efficient circulation of railway rolling stock. *Transportation Science*, 40(3):378–391, 2006.

[4] Filipe Alvelos. *Branch-and-Price and Multicommodity Flows.* PhD thesis, Escola de Engenharia, Universidade do Minho, Portugal, 2005.

[5] A. A. Assad. Multicommodity network flows—a survey. *Networks*, 8(1):37–91, 1978.

[6] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

[7] Cynthia Barnhart, Christopher A. Hane, and Pamela H. Vance. Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, 48(2):318–326, 2000.

[8] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[9] N. Ben-Khedher, J. Kintanar, C. Queille, and W. Stripling. Schedule optimization at SNCF: From conception to day of departure. *Interfaces*, 28:6–23, 1998.

[10] Lawrence Bodin and Bruce Golden. Classification in vehicle routing and scheduling. *Networks*, 11(2):97–108, 1981.

[11] Peter Brucker, Johann Hurink, and Thomas Rolfes. Routing of railway carriages. *Journal of Global Optimization*, 27(2-3):313–332, 2003.

[12] Valentina Cacchiani. *Models and Algorithms for Combinatorial Optimization Problems arising in Railway Applications.* PhD thesis, University of Bologna, Italy, 2007.

[13] Valentina Cacchiani. Models and algorithms for combinatorial optimization problems arising in railway applications. *4OR, A Quarterly Journal of Operations Research*, 7(1):109–112, 2009.

[14] Valentina Cacchiani, Alberto Caprara, Gábor Maróti, and Paolo Toth. On integer polytopes with few nonzero vertices. *Operations Research Letters*, 41(1):74–77, 2013.

[15] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Solving a real-world train-unit assignment problem. *Mathematical Programming Series B*, 124(1–2):207–231, 2010.

[16] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. A Fast Heuristic Algorithm for the Train Unit Assignment Problem. In Daniel Delling and Leo Liberti, editors, *12th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 25 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–9, Dagstuhl, Germany, 2012. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[17] Valentina Cacchiani, Alberto Caprara, and Paolo Toth. Models and algorithms for the train unit assignment problem. In *Combinatorial Optimization*, volume 7422 of *Lecture Notes in Computer Science*, pages 24–35. Springer Berlin Heidelberg, 2012.

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

[19] A. Caprara, L.G. Kroon, M. Monaci, M. Peeters, and P. Toth. Passenger railway optimization. In C. Barnhart and G. Laporte, editors, *Handbooks in Operations Research and Management Science*, volume 14, chapter 3, pages 129–187. Elsevier, 2007.

[20] A. Caprara, L.G. Kroon, and P. Toth. Optimization problems in passenger railway systems. In *Wiley Encyclopedia of Operations Research and Management Science*, volume 6, pages 3896–3905. Wiley, 2011.

[21] Alberto Caprara. Almost 20 years of combinatorial optimization for railway planning: from lagrangian relaxation to column generation. In Thomas Erlebach and Marco Lübbecke, editors, *Proceedings of the 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, volume 14 of *OpenAccess Series in Informatics (OASIcs)*, pages 1–12, Dagstuhl, Germany, 2010. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[22] Mark Chamberlain and Anthony Wren. Developments and recent experience with the BUSMAN and BUSMAN II systems. In M. Desrocher and J.-M. Rousseau, editors, *Computer-aided transit scheduling.* Springer, Berlin, 1992.

[23] Tito A. Ciriani, Yves Colombani, and Susanne Heipcke. Embedding optimisation algorithms with mosel. *4OR*, 1(2):155–167, 2003.

[24] Jean-François Cordeau, Guy Desaulniers, Norbert Lingaya, François Soumis, and Jacques Desrosiers. Simultaneous locomotive and car assignment at via rail canada. *Transportation Research Part B: Methodological*, 35(8):767–787, 2001.

[25] Jean-François Cordeau, François Soumis, and Jacques Desrosiers. A benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, 34(2):133–149, 2000.

[26] Jean-François Cordeau, François Soumis, and Jacques Desrosiers. Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research*, 49(4):531–548, 2001.

[27] George B. Dantzig and Philip Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, 1960.

[28] Jacques Desrosiers and Marco E. Lübbecke. A primer in column generation. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 1–32. Springer US, New York, USA, 2005.

[29] Pieter-Jan Fioole, Leo Kroon, Gábor Maróti, and Alexander Schrijver. A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174(2):1281–1297, 2006.

[30] L. R. Ford and D. R. Fulkerson. A suggested computation for maximal multicommodity network flows. *Management Science*, 5(1):97–101, 1958.

[31] Sarah Fores, Les Proll, and Anthony Wren. Experiences with a flexible driver scheduler. In Stefan Voß and R. Daduna, Joachim, editors, *Computer-Aided Scheduling of Public Transport*, volume 505 of *Lecture Notes in Economics and Mathematical Systems*, pages 137–152. Springer Berlin Heidelberg, 2001.

[32] Sarah Fores, Les Proll, and Anthony Wren. TRACS II: A hybrid IP/heuristic driver scheduling system for public transport. *The Journal of the Operational Research Society*, 53(10):1093–1100, 2002.

[33] Richard Freling, Ramon M. Lentink, Leo G. Kroon, and Dennis Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261–272, 2005.

[34] G. Gallo and C. Sodini. Extreme points and adjacency relationship in the flow polytope. *CALCOLO*, 15(3):277–288, 1978.

[35] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[36] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[37] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. North-Holland Publishing Co., Amsterdam, The Netherlands, 2004.

[38] Branko Grünbaum. Measures of symmetry for convex sets1. In *Convexity: Proceedings of the Seventh Symposium in Pure Mathematics of the American Mathematical Society*, volume 7, page 233. American Mathematical Soc., 1963.

[39] F. Hennig, B. Nygreen, M. Christiansen, K. Fagerholt, K. C. Furman, J. Song, G. R. Kocis, and P. H. Warrick. Maritime crude oil transportation — a split pickup and split delivery problem. *European Journal of Operational Research*, 218:764–774, 2012.

[40] Mike Hewitt, George Nemhauser, and Martin WP Savelsbergh. Branch-and-price guided search for integer programs with an application to the multicommodity fixed charge network flow problem. *INFORMS Journal on Computing*, 25(2):302–316, 2013.

[41] Mike Hewitt, George L. Nemhauser, and Martin W. P. Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22(2):314–325, 2010.

[42] A.J. Hoffman and J.B. Kruskal. Integral boundary points of convex polyhedral. In H.W. Kuhn and A.W. Tucker, editors, *Linear Inequalities and Related Systems*, chapter 13. 1956.

[43] T. C. Hu. Multi-commodity network flows. *Operations Research*, 11(3):344–360, 1963.

[44] Dennis Huisman, Leo G. Kroon, Ramon M. Lentink, and Michiel J. C. M. Vromans. Operations research in passenger railway transportation. *Statistica Neerlandica*, 59(4):467–497, 2005.

[45] Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon, editors, *Column Generation*, pages 33–65. Springer US, New York, 2005.

[46] Peter L. Jackson and David F. Lynch. Revised dantzig-wolfe decomposition for staircase-structured linear programs. *Mathematical Programming*, 39(2):157–179, 1987.

[47] William S Jewell. New methods in mathematical programming-optimal flow through networks with gains. *Operations Research*, 10(4):476–499, 1962.

[48] Jeff L. Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26(2):209–236, 1978.

[49] Leo G. Kroon, Ramon M. Lentink, and Alexander Schrijver. Shunting of passenger train units: An integrated approach. *Transportation Science*, 42(4):436–449, 2008.

[50] Ann S. K. Kwan, Margaret E Parker, Raymond S. K. Kwan, Sarah Fores, Les Proll, and Anthony Wren. Recent advances in TRACS. In *Preprints of the 9th International Conference on Computer-Aided Scheduling of Public Transport*, 2004.

[51] Raymond S. K. Kwan. Bus and train driver scheduling. In J. Y. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 51, pages 51(1)–51(19). CRC Press, Boca Raton, 2004.

[52] Raymond S. K. Kwan. Case studies of successful train crew scheduling optimisation. *Journal of Scheduling*, 14(5):423–434, 2011.

[53] Raymond S. K. Kwan and Ann Kwan. Effective search space control for large and/or complex driver scheduling problems. *Annals of Operations Research*, 155(1):417–435, 2007.

[54] Raymond S. K. Kwan and Mohammad A. Rahin. Object oriented bus vehicle scheduling—the boost system. In N.H.M. Wilson, editor, *Computer-aided transit scheduling.* Springer-Verlag, 1999.

[55] Raymond S. K. Kwan, Anthony Wren, and Ann S. K. Kwan. Hybrid genetic algorithms for scheduling bus and train drivers. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 285–292, 2000.

[56] Eugene L Lawler. *Combinatorial optimization: networks and matroids.* Courier Dover Publications, 1976.

[57] Jeff T Linderoth and Martin WP Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

[58] Norbert Lingaya, Jean-François Cordeau, Guy Desaulniers, Jacques Desrosiers, and François Soumis. Operational car assignment at VIA rail canada. *Transportation Research Part B: Methodological*, 36(9):755 – 778, 2002.

[59] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53:1007–1023, 2002.

[60] A. R. Mahjoub. Polyhedral approaches. In V. T. Paschos, editor, *Concepts of combinatorial optimization, Wiley, Hoboken, USA*, chapter 10, pages 261–320. 2010.

[61] Gábor Maróti. *Operations Research Models for Railway Rolling Stock Planning.* PhD thesis, Eindhoven University of Technology, the Netherlands, 2006.

[62] Gábor Maróti and Leo G. Kroon. Maintenance routing for train units: The transition model. *Transportation Science*, 39(4):518–525, 2005.

[63] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations.* Wiley-Interscience series in discrete mathematics and optimization. J. Wiley & Sons, 1990.

[64] R Kipp Martin. Generating alternative mixed-integer programming models using variable redefinition. *Operations Research*, 35(6):820–831, 1987.

[65] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.

[66] The Geometry Center of the University of Minnesota. The QuickHull's official website. `http://www.qhull.org/`.

[67] Marc Peeters and Leo G. Kroon. Circulation of railway rolling stock: a branch-and-price approach. *Computers & OR*, 35(2):538–556, 2008.

[68] Henri Poincaré. Second complément à l'analysis situs. In *Proceedings London Mathematical Society*, volume 32, pages 277–308. London Mathematical Society. [collected in *Œuvres de Henri Poincaré*, volume 6, pages 338–370], 1900.

[69] M. W. P. Savelsbergh. A Branch-and-Price Algorithm for the Generalized Assignment Problem. *Operations Research*, 45:831–841, 1997.

[70] Alexander Schrijver. Minimum circulation of railway stock. *CWI Quarterly*, 6:205–217, 1993.

[71] Bala Shetty. Multicommodity network flows. In Saul I. Gass and Michael C. Fu, editors, *Encyclopedia of Operations Research and Management Science*, pages 993–996. Springer US, 2013.

[72] Barbara M Smith and Anthony Wren. VAMPIRES and TASC: two successfully applied bus scheduling programs. In Anthony Wren, editor, *Computer scheduling of public transport*, pages 97–124. North-Holland, Amsterdam, 1981.

[73] Tracsis Plc. TRACS-RS—rolling stock planning software, 2013.

[74] François Vanderbeck. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48(1):111–128, 2000.

[75] François Vanderbeck and Martin WP Savelsbergh. A generic view of dantzig–wolfe decomposition in mixed integer programming. *Operations Research Letters*, 34(3):296–306, 2006.

[76] Giulio Varsi. The multidimensional content of the frustrum of the simplex. *Pacific Journal of Mathematics*, 46:303–314, 1973.

[77] Arthur F Veinott, Jr and George B Dantzig. Integral extreme points. *SIAM review*, 10(3):371–372, 1968.

[78] R.V. Vohra. *Mechanism Design: A Linear Programming Approach.* Cambridge University Press, 2011.

[79] Roger Webster. *Convexity.* Oxford University Press, 1994.

[80] K. Wolfenden and A. Wren. Locomotive scheduling by computer. In *Proc. British Joint Computer Conference*, volume 1, pages 31–37, London, UK, 1966. IEE Conference Publication No. 19.

[81] Laurence A Wolsey. *Integer programming.* Wiley, 1998.

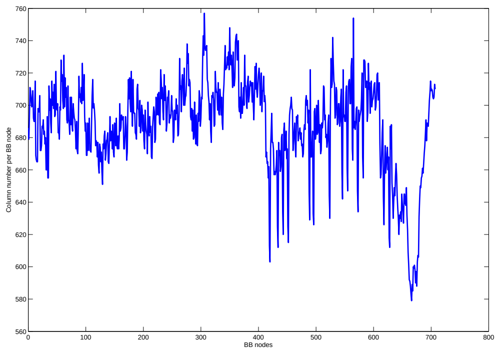
[82] Anthony Wren. Scheduling vehicles and their drivers—forty years' experience. Technical Report 2004.03, School of Computing, University of Leeds, April 2004.

[83] Anthony Wren, Sarah Fores, Ann Kwan, Raymond Kwan, Margaret Parker, and Les Proll. A flexible system for scheduling drivers. *Journal of Scheduling*, 6:437–455, 2003.

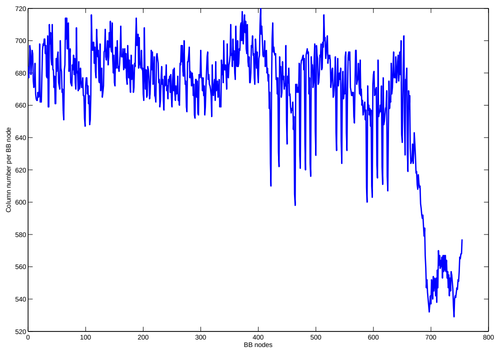
[84] Koorush Ziarati, François Soumis, Jacques Desrosiers, and Marius M. Solomon. A branch-first, cut-second approach for locomotive assignment. *Management Science*, 45:1156–1168, 1999.

Appendix A

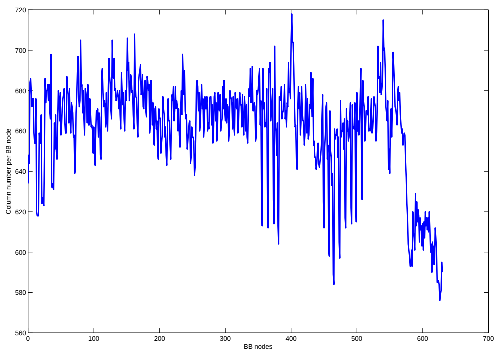
Some figures from experiments



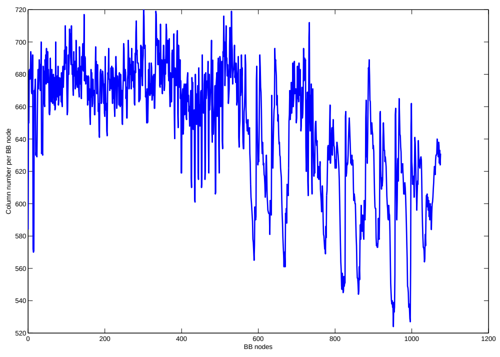
(a) 2b: $M_{CG} = 600$



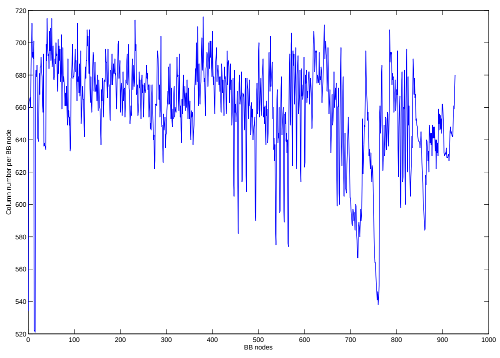
(b) 2a: $M_{CG} = 500$



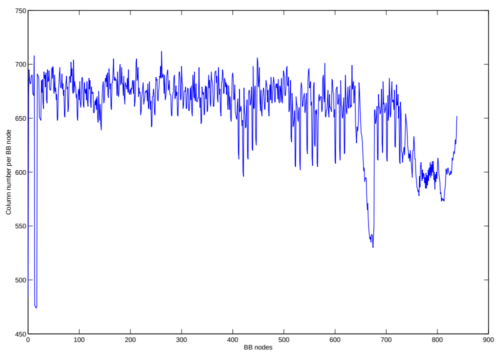
(c) 2c: $M_{CG} = 450$



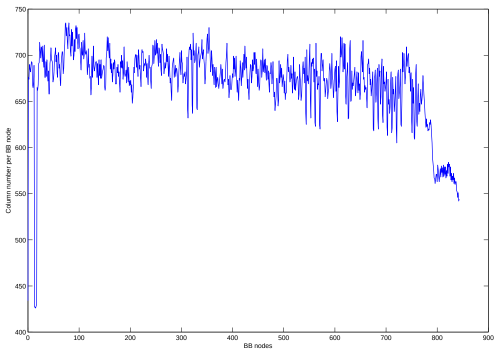
(d) 2d: $M_{CG} = 400$



(e) 2e: $M_{CG} = 350$

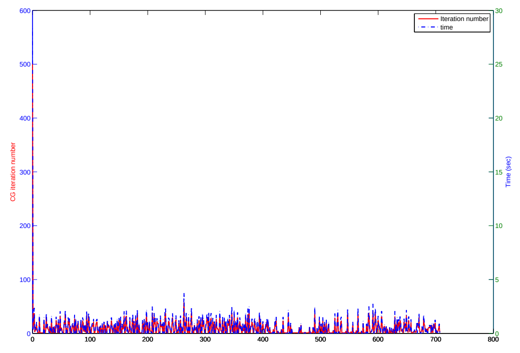


(f) 2f: $M_{CG} = 300$

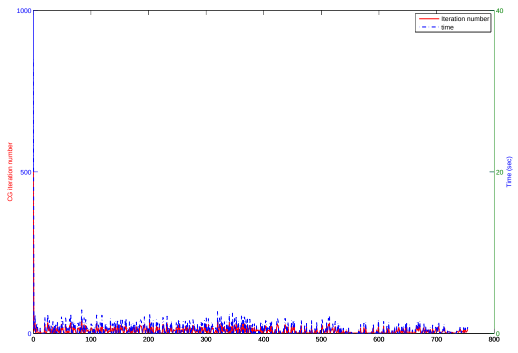


(g) 2g: $M_{CG} = 250$

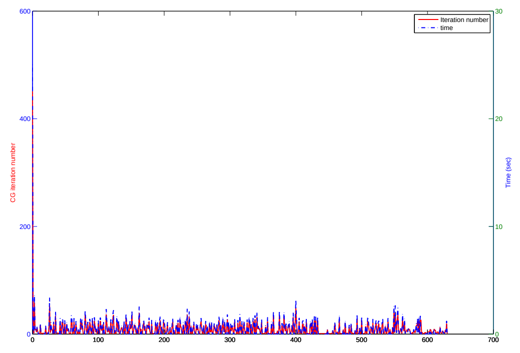
Figure A.1: Number of CG columns for experiments GB-334R-2



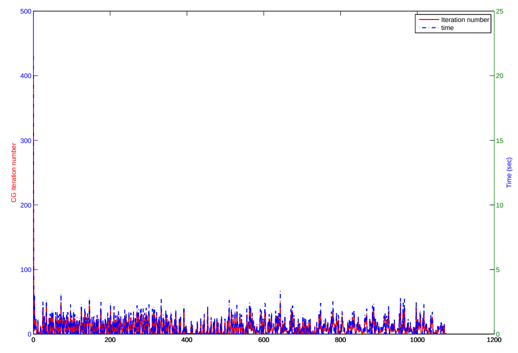
(a) 2b: $M_{CG} = 600$



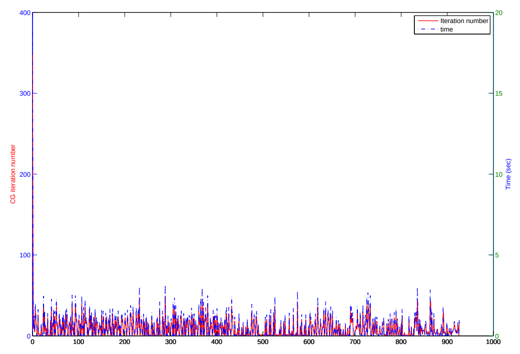
(b) 2a: $M_{CG} = 500$



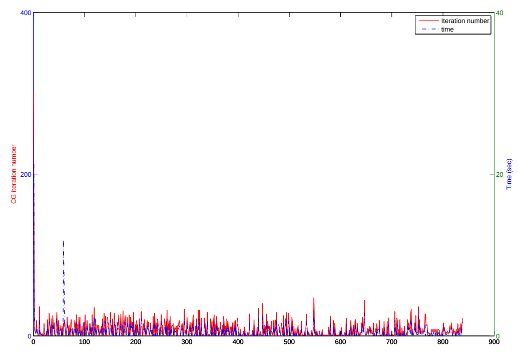
(c) 2c: $M_{CG} = 450$



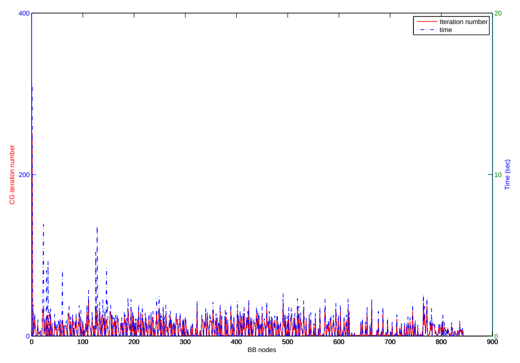
(d) 2d: $M_{CG} = 400$



(e) 2e: $M_{CG} = 350$

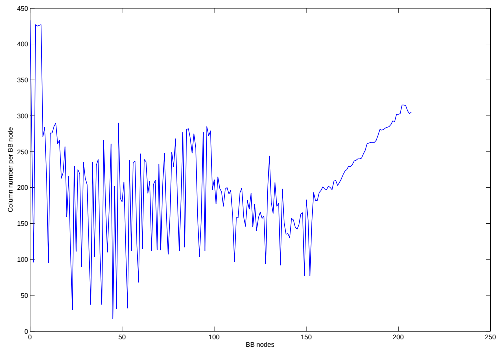


(f) 2f: $M_{CG} = 300$

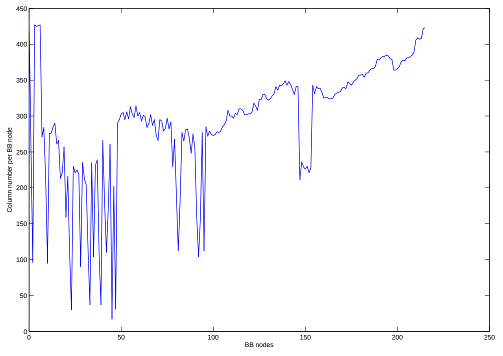


(g) 2g: $M_{CG} = 250$

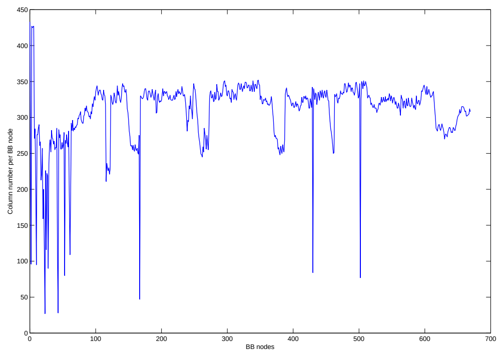
Figure A.2: CG iteratin numbers against CG time per BB node for experiments GB-334R-2



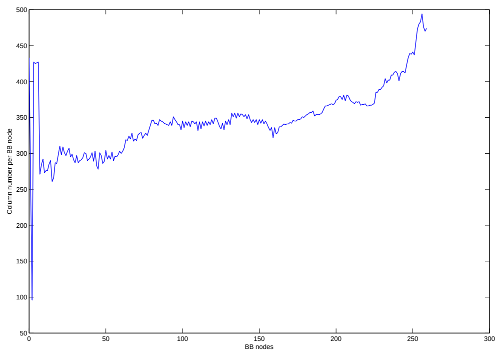
(a) 5a: $it_F = 20, \rho_{RC} = 0$



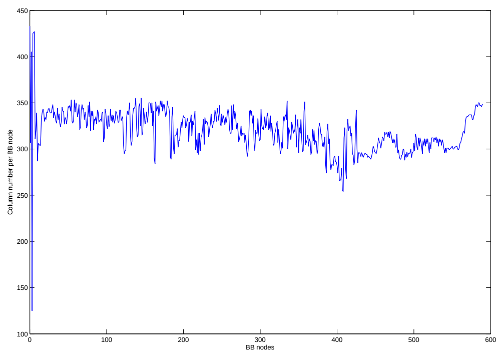
(b) 5b: $it_F = 25, \rho_{RC} = 0$



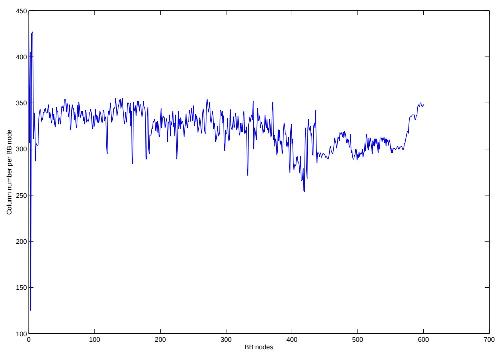
(c) 5c: $it_F = 30, \rho_{RC} = 0$



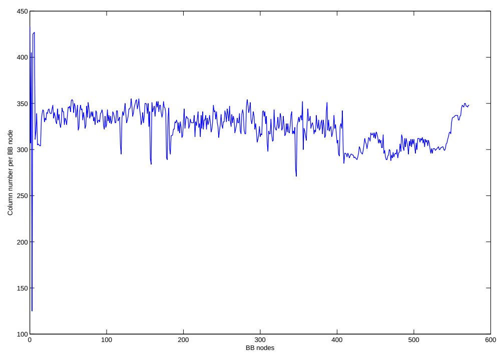
(d) 5d: $it_F = 35, \rho_{RC} = 0$



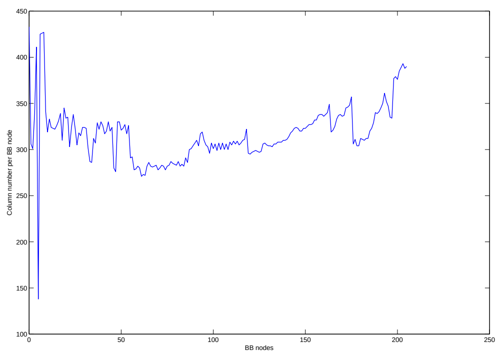
(e) 5f: $it_F = 20, \rho_{RC} = 0.1$



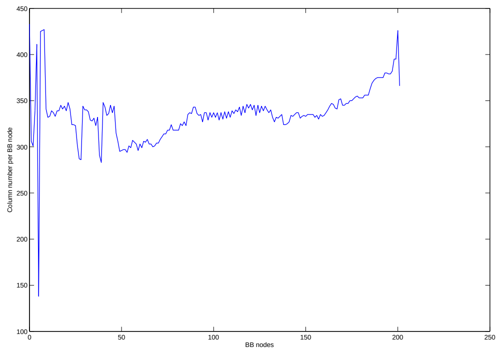
(f) 5g: $it_F = 25, \rho_{RC} = 0.1$



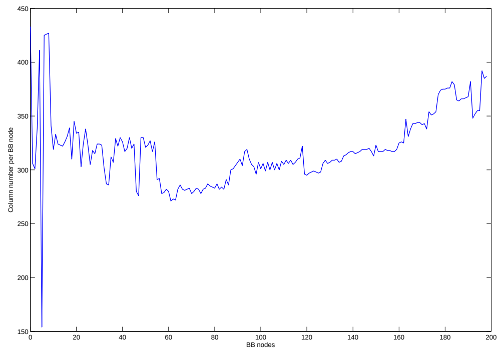
(g) 5h: $it_F = 30, \rho_{RC} = 0.1$



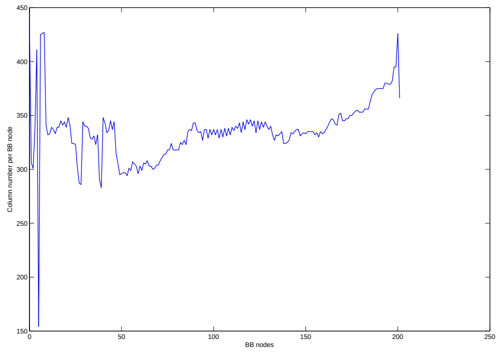
(h) 5j: $it_F = 10, \rho_{RC} = 0.2$



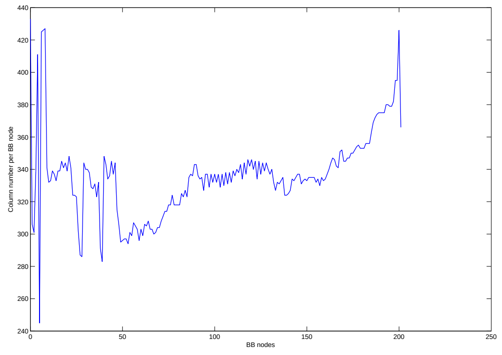
(i) 5k: $it_F = 20, \rho_{RC} = 0.2$



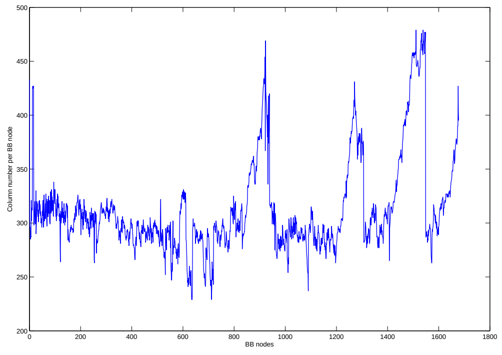
(j) 5m: $it_F = 10, \rho_{RC} = 0.3$



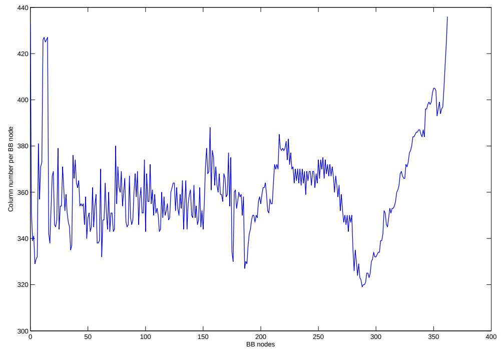
(k) 5n: $it_F = 20, \rho_{RC} = 0.3$



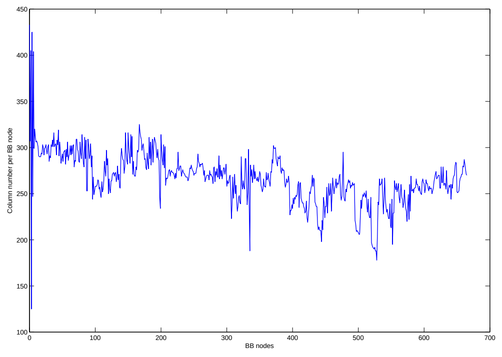
(l) 5t: $it_F = 20, \rho_{RC} = 0.8$



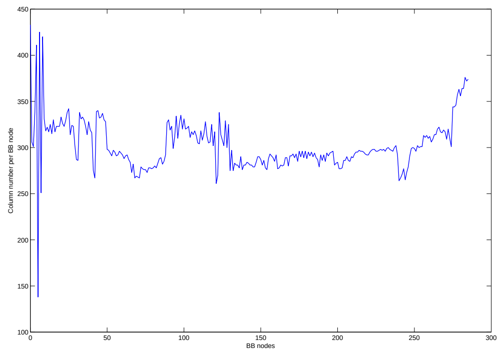
(m) 5u: $it_F = 20, \rho_{RC} = 0.9$



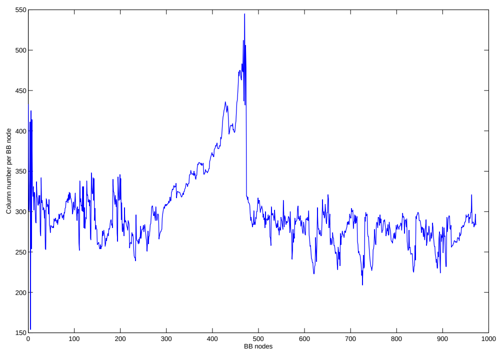
(n) 5v: $it_F = 20, \rho_{RC} = 1.0$



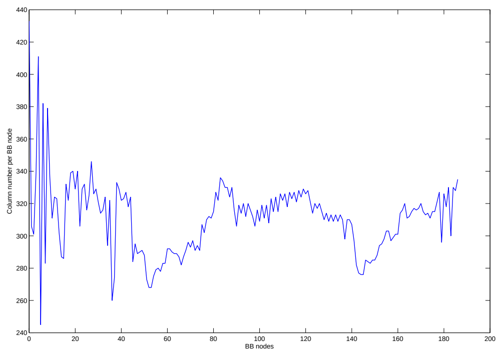
(o) 5w: $it_F = 1, \rho_{RC} = 0.1$



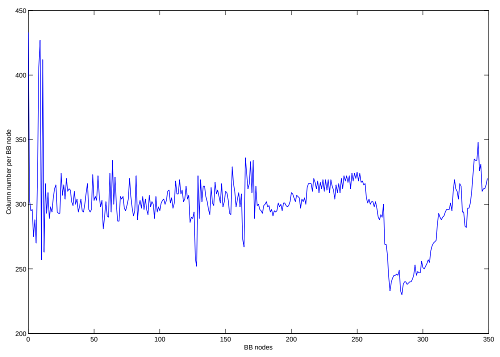
(p) 5x: $it_F = 1, \rho_{RC} = 0.2$



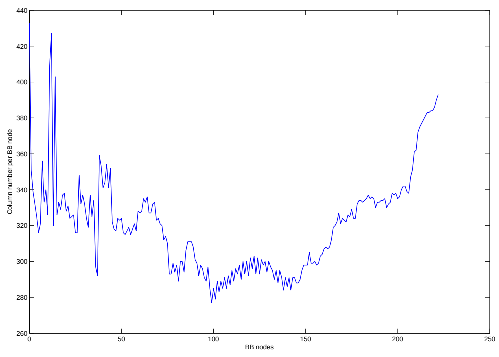
(q) 5y: $it_F = 1, \rho_{RC} = 0.3$



(r) 5ad: $it_F = 1, \rho_{RC} = 0.8$

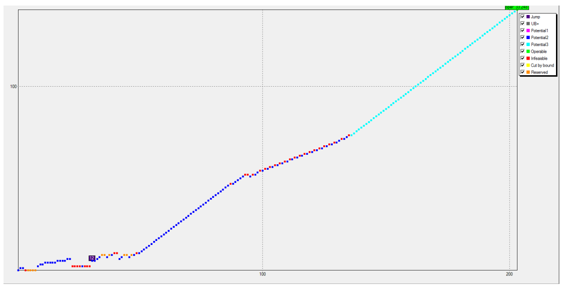


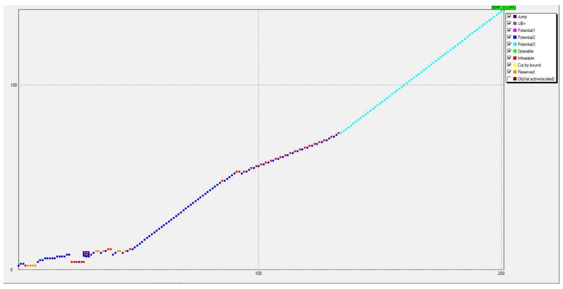
(s) 5ae: $it_F = 1, \rho_{RC} = 0.9$

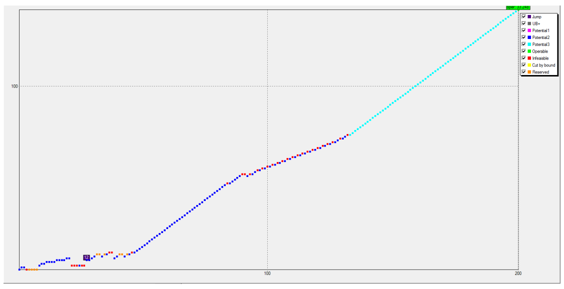


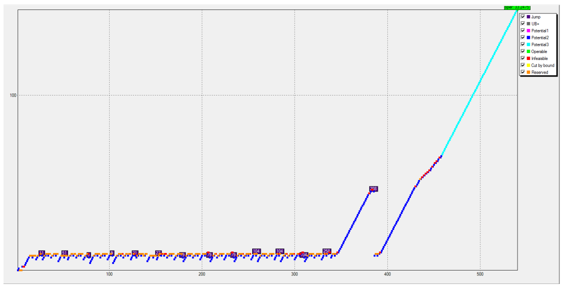
(t) 5af: $it_F = 1, \rho_{RC} = 1.0$

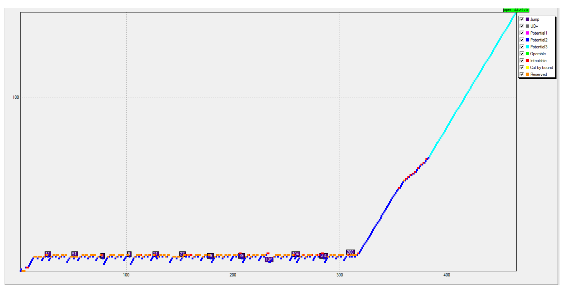
Figure A.3: Number of CG columns for experiments GB-334R-5

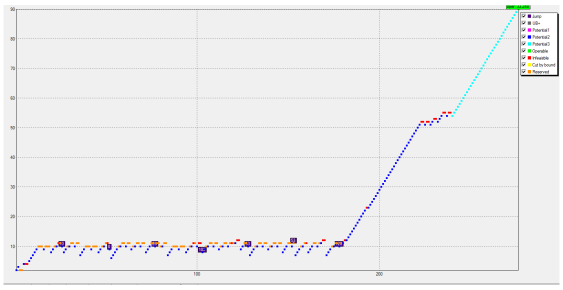

(a) 7x: $\rho_B = 0.0001$

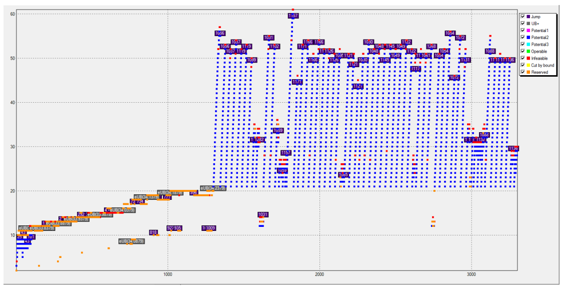

(b) 7a: $\rho_B = 0.01$

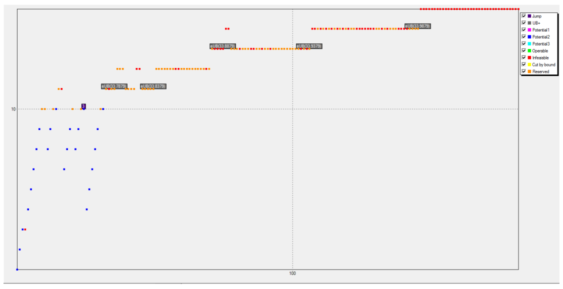

(c) 7b: $\rho_B = 0.02$

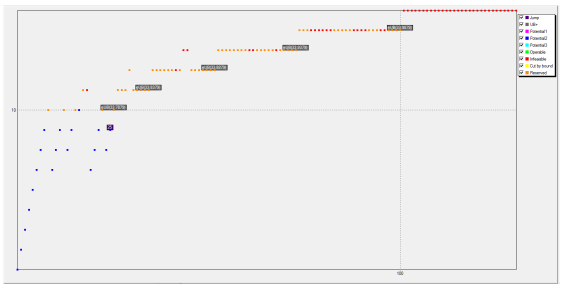

(d) 7c: $\rho_B = 0.05$


(e) 7d: $\rho_B = 0.07$


(f) 7e: $\rho_B = 0.1$


(g) 7f: $\rho_B = 0.2$


(h) 7g: $\rho_B = 0.3$


(i) 7h: $\rho_B = 0.2$

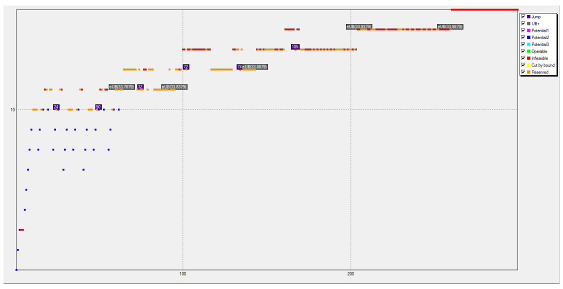
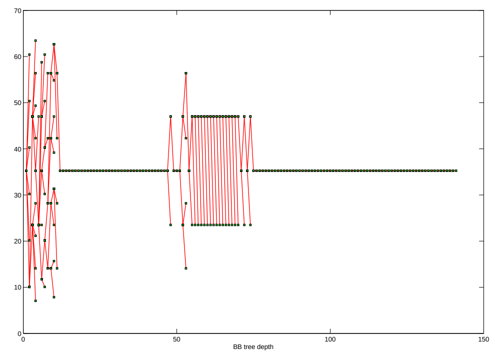
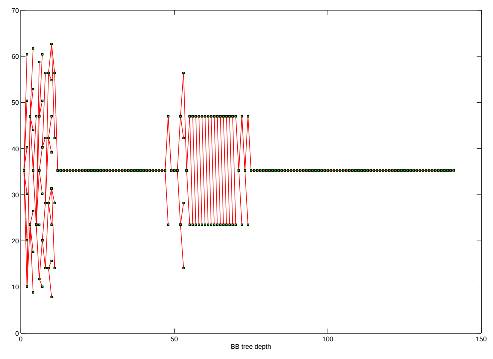

(j) 7i: $\rho_B = 0.2$

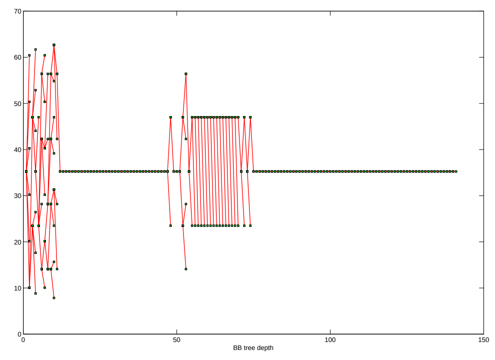
Figure A.4: Node-depth graphs for experiments GB-334R-7



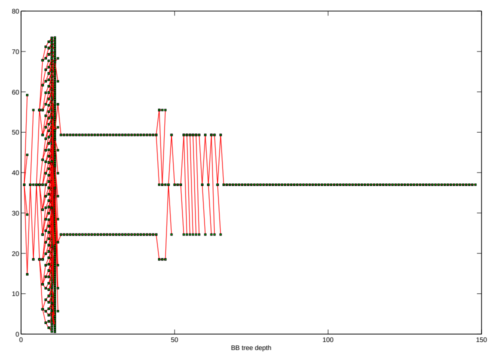
(a) 7x: $\rho_B = 0.0001$



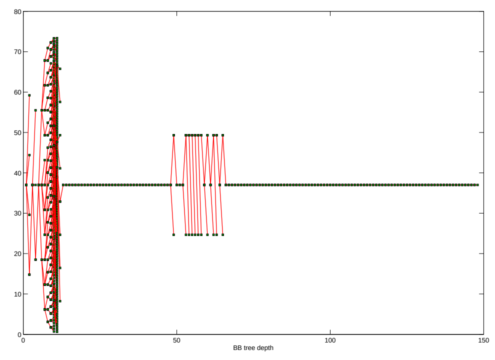
(b) 7a: $\rho_B = 0.01$



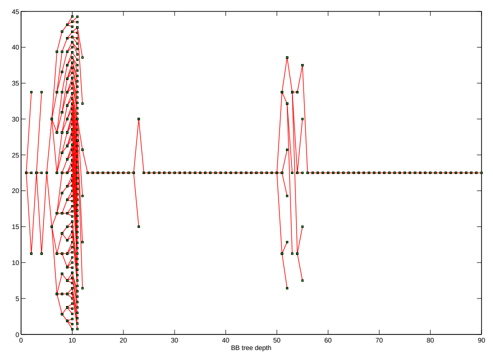
(c) 7b: $\rho_B = 0.02$



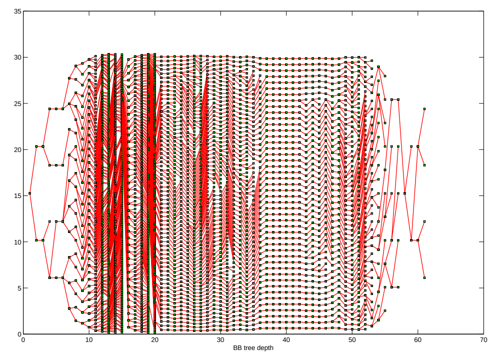
(d) 7c: $\rho_B = 0.05$



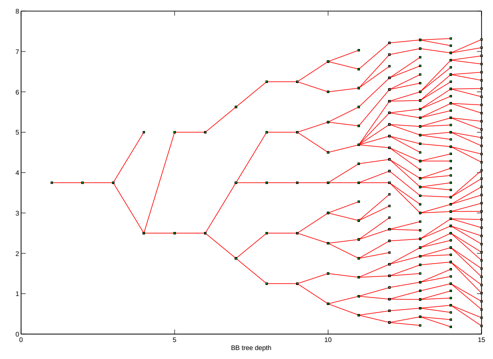
(e) 7d: $\rho_B = 0.07$



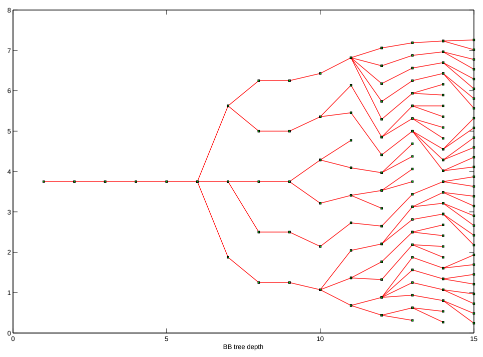
(f) 7e: $\rho_B = 0.1$



(g) 7f: $\rho_B = 0.2$



(h) 7g: $\rho_B = 0.3$


(i) 7h: $\rho_B = 0.2$

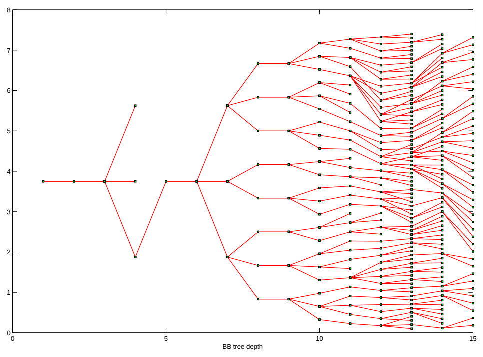
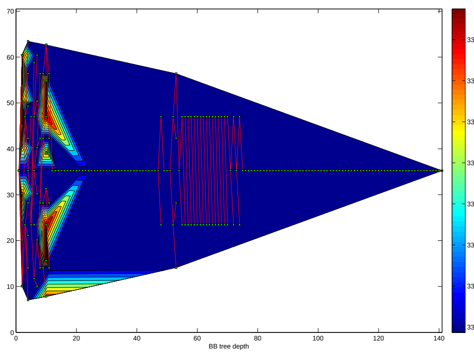
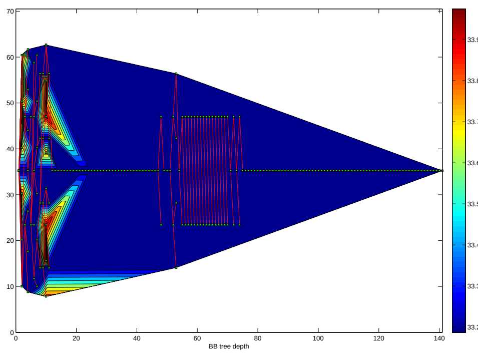
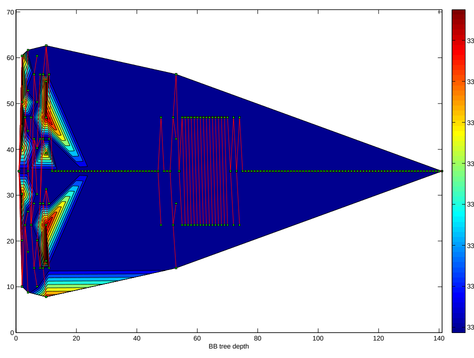
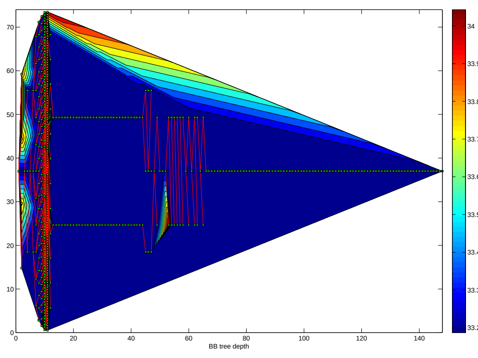

(j) 7i: $\rho_B = 0.2$

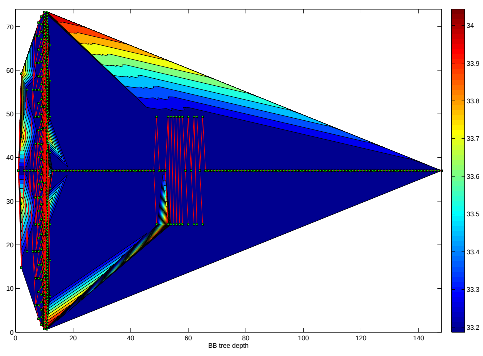
Figure A.5: BB trees for experiments GB-334R-7


(a) 7x: $\rho_B = 0.0001$

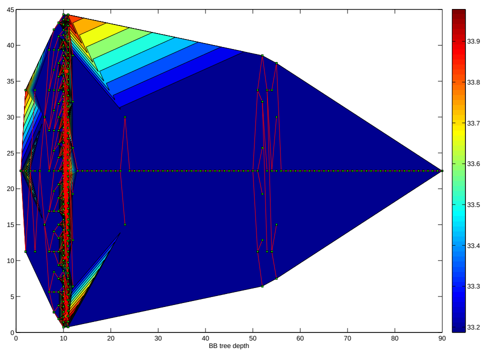

(b) 7a: $\rho_B = 0.01$


(c) 7b: $\rho_B = 0.02$

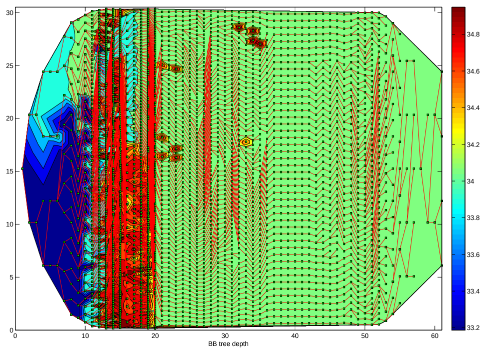

(d) 7c: $\rho_B = 0.05$



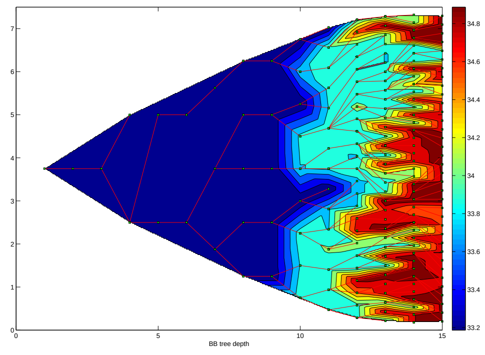
(e) 7d: $\rho_B = 0.07$



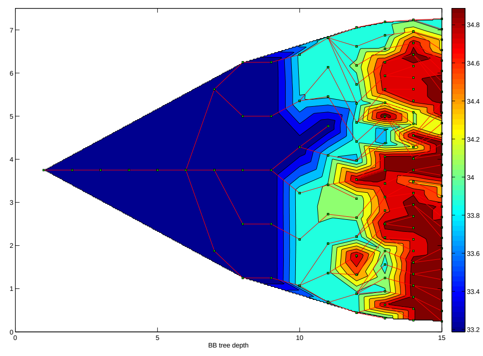
(f) 7e: $\rho_B = 0.1$



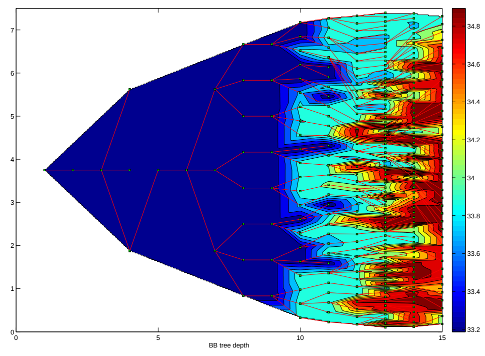
(g) 7f: $\rho_B = 0.2$



(h) 7g: $\rho_B = 0.3$

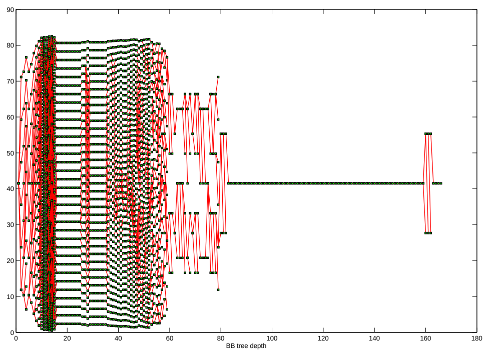


(i) 7h: $\rho_B = 0.2$

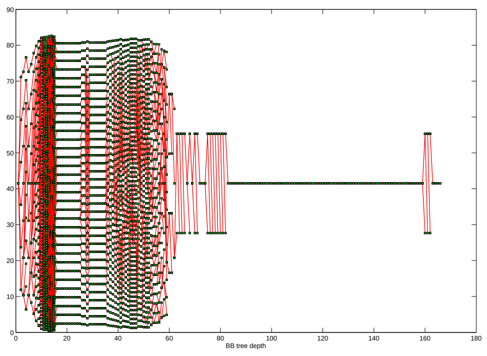


(j) 7i: $\rho_B = 0.2$

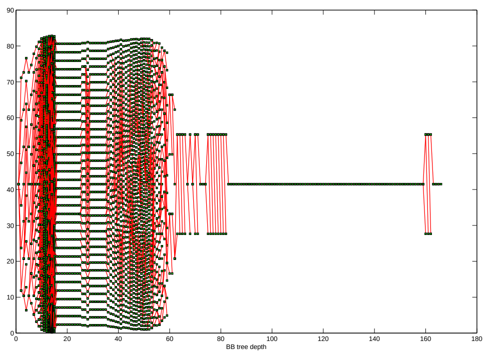
Figure A.6: BB trees with node values for experiments GB-334R-7



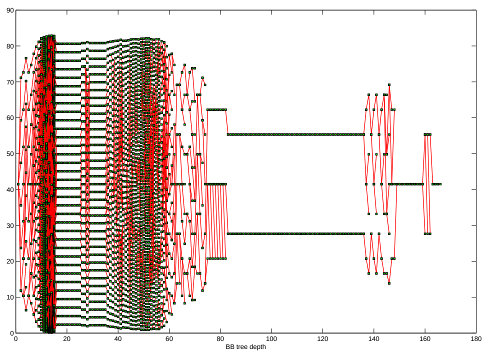
(a) 8a2: $G = 2$



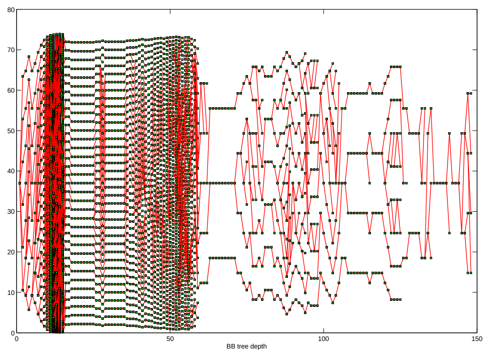
(b) 8b2: $G = 10$



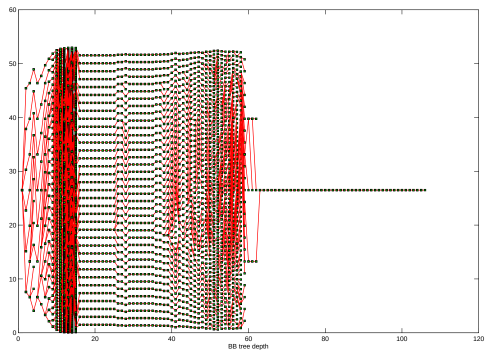
(c) 8c2: $G = 20$



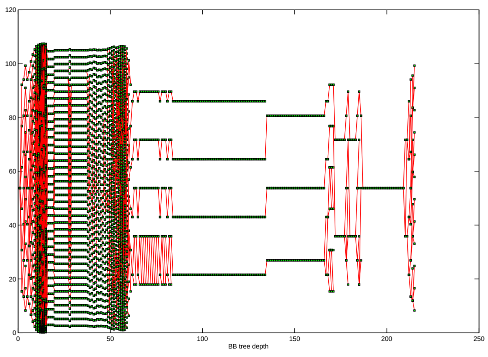
(d) 8d2: $G = 30$



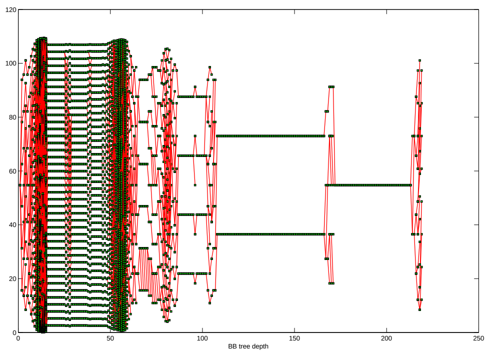
(e) 8e2=: $G = 40$



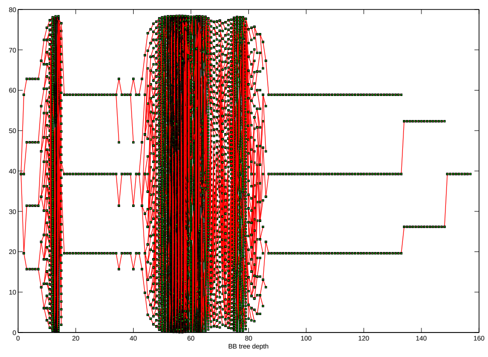
(f) 8f2=: $G = 50$



(g) 8g2=: $G = 60$

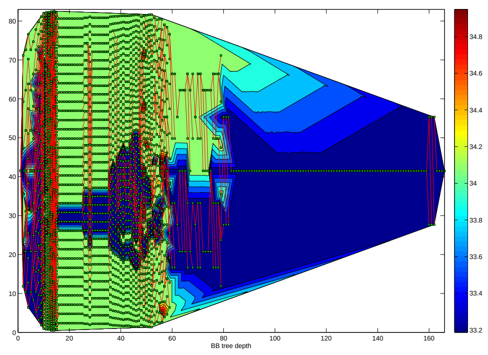


(h) 8h2=: $G = 70$

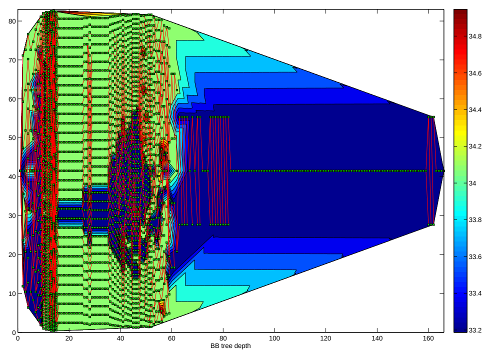


(i) 8t2: jump disabled

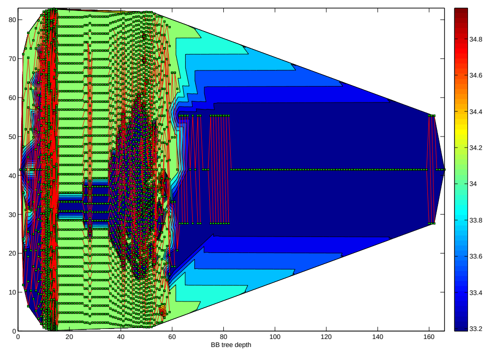
Figure A.7: BB trees for experiments GB-334R-8ii



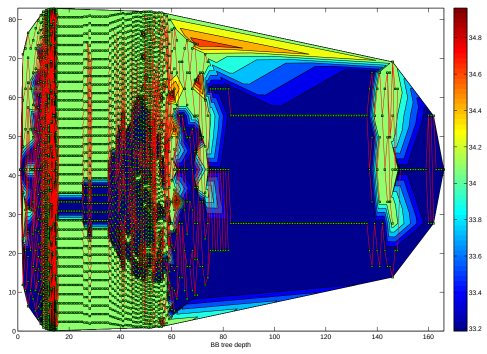
(a) 8a2: $G = 2$



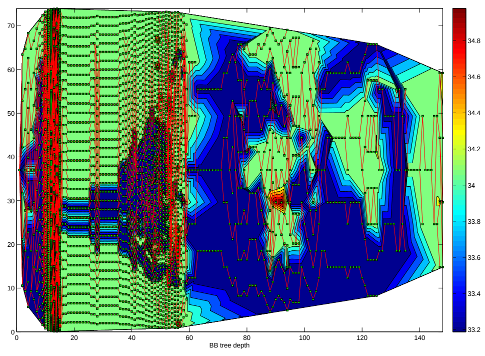
(b) 8b2: $G = 10$



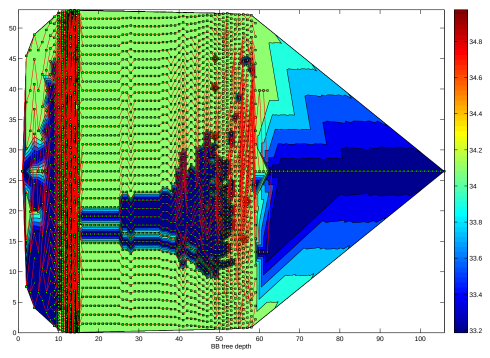
(c) 8c2: $G = 20$

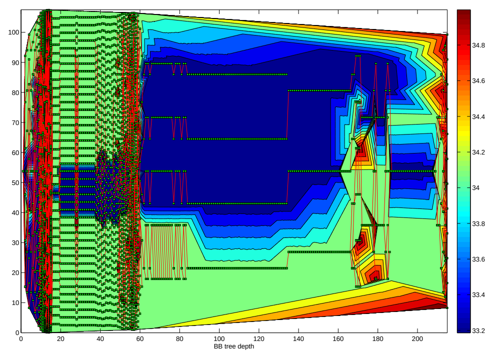


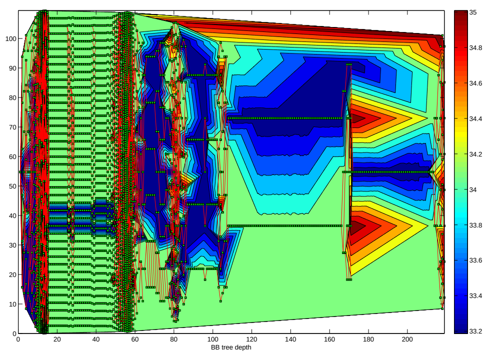
(d) 8d2: $G = 30$



(e) 8e2=: $G = 40$


(f) 8f2=: $G = 50$


(g) 8g2=: $G = 60$


(h) 8h2=: $G = 70$

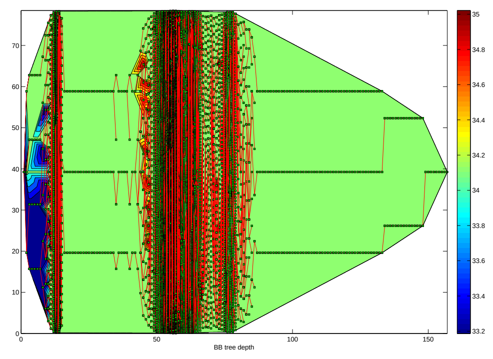
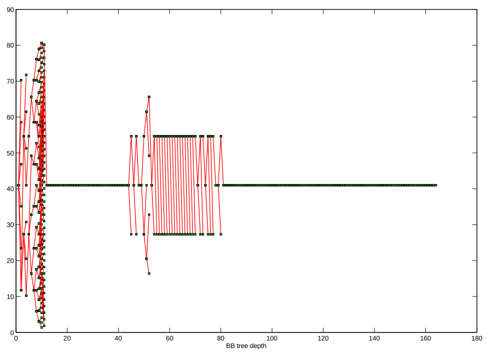
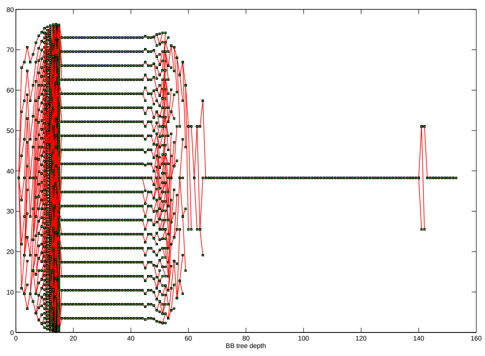

(i) 8t2: jump disabled

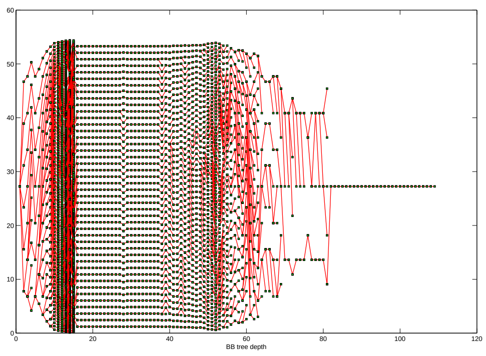
Figure A.8: BB trees with node values for experiments GB-334R-8ii



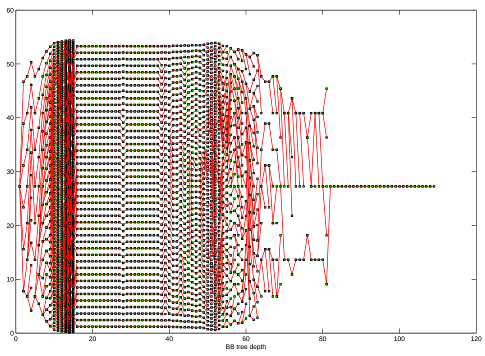
(a) 9n: $\varDelta^0 = 0.1$



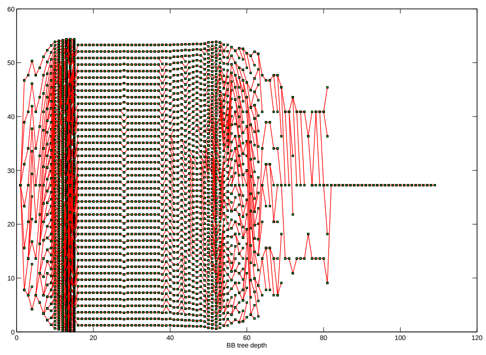
(b) 9i: $\varDelta^0 = 1$



(c) 9j: $\varDelta^0 = 2$

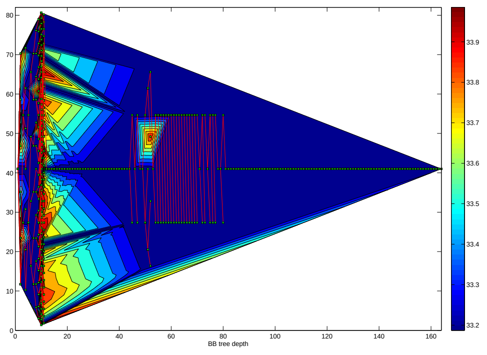


(d) 9k: $\varDelta^0 = 3$

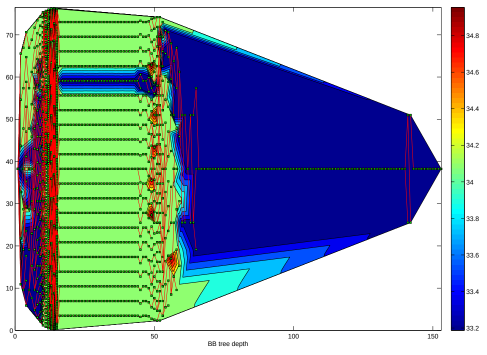


(e) 9l: $\varDelta^0 = 99999$

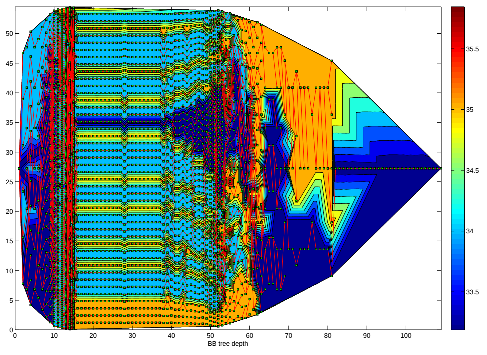
Figure A.9: BB trees for GB-334R-9i, $G = 25$



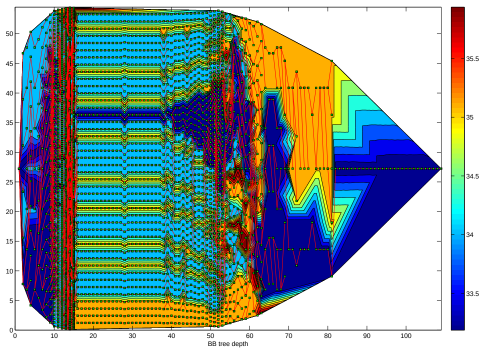
(a) 9n: $\varDelta^0 = 0.1$



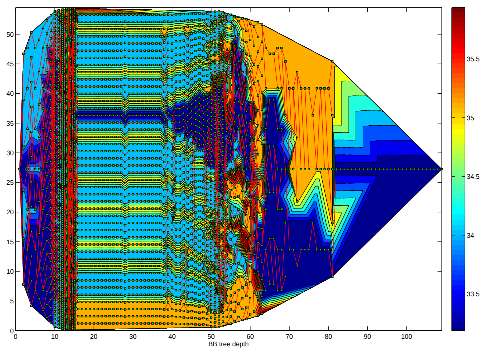
(b) 9i: $\varDelta^0 = 1$



(c) 9j: $\varDelta^0 = 2$

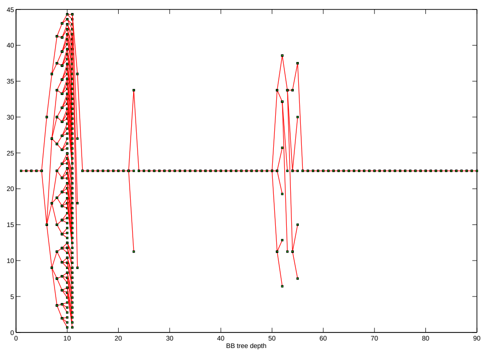


(d) 9k: $\varDelta^0 = 3$

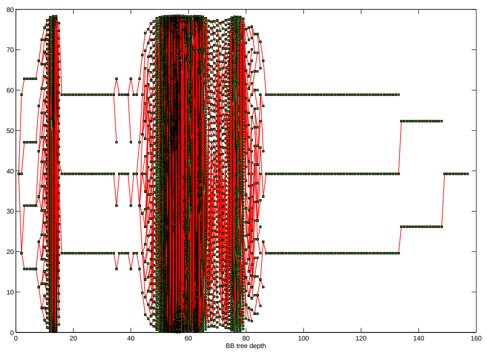


(e) 9l: $\varDelta^0 = 99999$

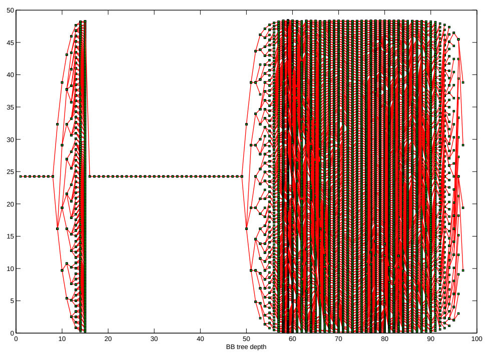
Figure A.10: BB trees with node values for GB-334R-9i, $G = 25$



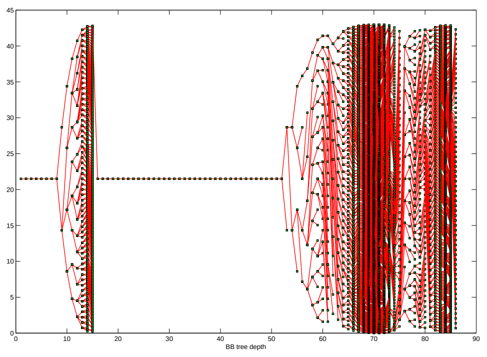
(a) 9m: $\mathit{\Delta}^0 = 0.1$



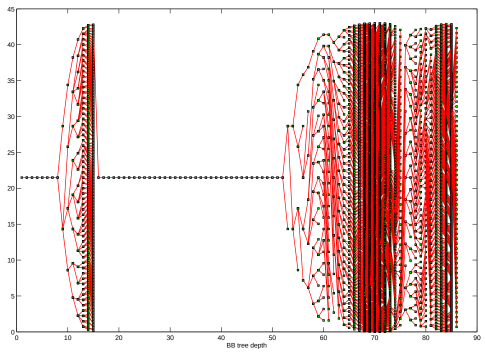
(b) 9w: $\mathit{\Delta}^0 = 1$



(c) 9x: $\mathit{\Delta}^0 = 2$

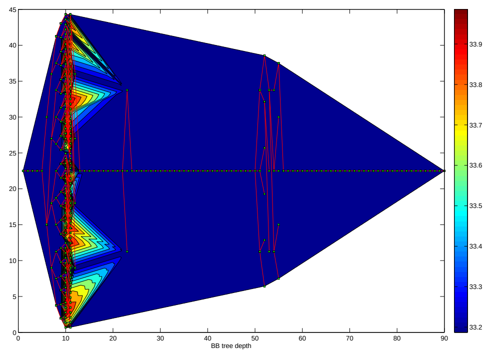


(d) 9y: $\mathit{\Delta}^0 = 3$

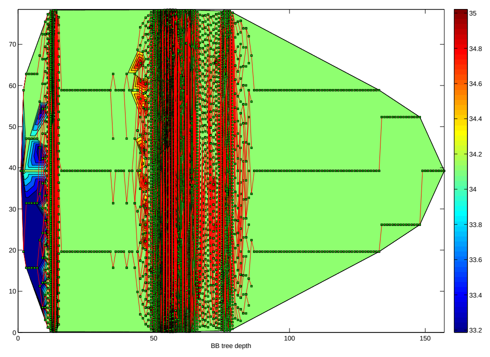


(e) 9z: $\mathit{\Delta}^0 = 99999$

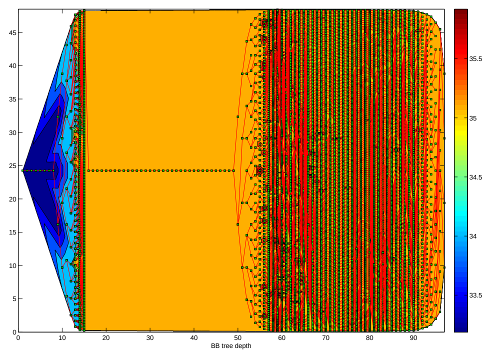
Figure A.11: BB trees for GB-334R-9ii, jump disabled



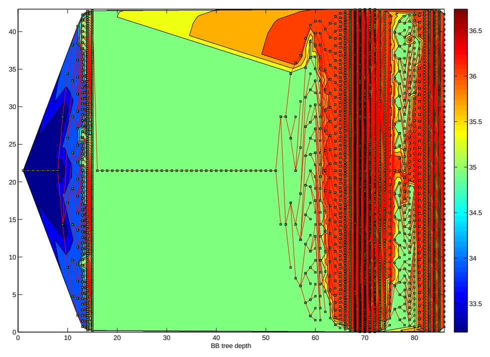
(a) 9m: $\varDelta^0 = 0.1$



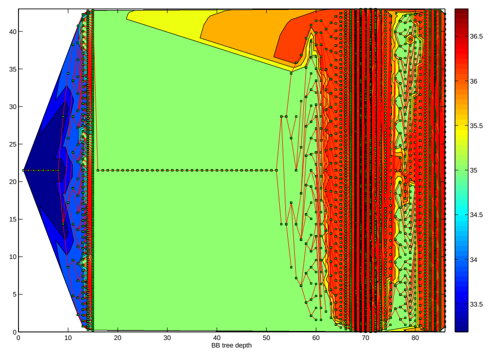
(b) 9w: $\varDelta^0 = 1$



(c) 9x: $\varDelta^0 = 2$



(d) 9y: $\varDelta^0 = 3$



(e) 9z: $\varDelta^0 = 99999$

Figure A.12: BB trees with node values for GB-334R-9ii, jump disabled

Appendix B

Abbreviations

BB	branch-and-bound
CG	column generation
DAG	directed acyclic graph
ECS	empty coaching stock (empty-running)
FCMF	fixed-charge multicommodity flow
FIFO	first-in-first-out
FILO	first-in-last-out
GaS	generate-and-select
ILP	integer linear programming, or integer linear program
LB	lower bound
LP	linear programming, or linear program
NSR	Nederlandse Spoorwegen Reizigers (a Dutch train operating company)
OP	over-provided
PAX	passenger number count (at ScotRail)
RMP	restricted master problem
SLIM	size limited iterative method
SO	Saturday only (in train unit diagrams)
SU (SUN)	Sunday only (in train unit diagrams)
SX	Saturday and Sunday excepted (in train unit diagrams)
TUAP	train unit assignment problem
TUSP	train unit scheduling problem
UB	upper bound
UP	under-provided

Appendix C

Notations

C.1 Notations in Chapter 3

$\mathcal{G}$	directed acyclic graph for the connection-arc representation
$\mathcal{N}$	set of nodes in $\mathcal{G}$
$\mathcal{A}$	set of arcs in $\mathcal{G}$
N	set of train nodes in $\mathcal{G}$
s	origin of $\mathcal{G}$
t	sink of $\mathcal{G}$
A	set of connection arcs in $\mathcal{G}$
A_0	set of sign-on/-off arcs in $\mathcal{G}$
$\delta_-(j)$	set of incoming arcs of node j in $\mathcal{G}$
$\delta_+(j)$	set of outgoing arcs of node j in $\mathcal{G}$
P	set of paths in $\mathcal{G}$
P_a	set of paths passing through arc a in $\mathcal{G}$
P_j	set of paths passing through node j in $\mathcal{G}$
K	set of all train unit types
$\mathcal{G}^k$	type-graph for unit type k
$\mathcal{N}^k$	set of nodes in $\mathcal{G}^k$
$\mathcal{A}^k$	set of arcs in $\mathcal{G}^k$
N^k	set of train nodes in $\mathcal{G}^k$
A^k	set of connection arcs in $\mathcal{G}^k$
A_0^k	set of sign-on/-off arcs in $\mathcal{G}^k$
$\delta_-^k(j)$	set of incoming arcs of node j in $\mathcal{G}^k$
$\delta_+^k(j)$	set of outgoing arcs of node j in $\mathcal{G}^k$
P^k	set of paths in $\mathcal{G}^k$
P_a^k	set of paths passing through arc a in $\mathcal{G}^k$
P_j^k	set of paths passing through node j in $\mathcal{G}^k$
D	set of depots

$x_a \in \mathbb{Z}_+, \forall a \in \mathcal{A}^k, \forall k \in K$	type-arc variables, indicating the flow amount (number of units) in type-arc a of type-graph $\mathcal{G}^k$
$y_a \in \{0,1\}, \forall a \in \mathcal{A}$	block-arc variables, indicating whether block-arc a is used
$z^+_{d,k}, z^-_{d,k} \in \mathbb{R}_+, \forall d \in D, \forall k \in K$	overnight balance variables, calculating the inventory imbalance at depot d for type k
C_1, C_2, C_3	weights in the objective
b_0^k	fleet limit for type k
q_k	unit capacity in number of seats for type k
r_j	passenger demand in number of seats for train j
v^k	number of cars for type k
u_j	(uniform) coupling upper bound for train j
u_a	possible maximum number of coupled units for connection a
$\tau^D_{\mathrm{arr}(i)}$	single decoupling time at the arrival location of train i
$\tau^C_{\mathrm{dep}(j)}$	single coupling time at the departure location of train j
e_{ij}	time slack between train i and j modified with buffering / shunting / empty-running
A^*	set of connection arcs where possible time allowance violation may occur due to coupling/decoupling
N_B^-	set of trains whose departure locations are banned for coupling
N_B^+	set of trains whose arrival locations are banned for decoupling
$(A_d^k)^{\mathrm{on}}$	set of sign-on arcs implied by depot d and type k
$(A_d^k)^{\mathrm{off}}$	set of sign-off arcs implied by depot d and type k
$V_1, \ldots, V_6$	sub-weights for each type-arc cost $c_a, a \in \mathcal{A}^k$
δ_a	fleet size weight modified with arc a
$m^k_{j(a)}$	carriage-kilometre weight for arc a
e_a	empty-running weight for arc a
g_a^k	maintenance gap weight for arc a
$\pi^k_{j(a)}$	type-route preference weight for (k, a) pair
π_a^k	type-depot preference weight for (k, a) pair

C.2 Notations in Chapter 4

$\tilde{u}_j$	uniform coupling upper bound in number of units for train j
N'	set of trains where only a single family of units are available
N''	set of trains where multiple families of units are available
Φ_j	set of families available at train $j \in N''$
u_j^φ	the coupling upper bound in number of cars at train j for family φ
$\tilde{u}_j^\varphi$	the coupling upper bound in number of units at train j for family φ
$\zeta_j^\varphi \in \{0,1\}, \forall \varphi \in \Phi_j, \forall j \in N''$	train-family variable indicating whether family φ covers train j
W_j	unit combination set for train j
w_k^j	number of units of type k used at train j, vector form: w^j
K_j	set of available types at train j
$\mathrm{conv}(W_j)$	the convex hull of W_j
$H^j w^j \leq h^j$	the system describing $\mathrm{conv}(W_j)$
$f \in F_j$	facets of $\mathrm{conv}(W_j)$
$H^j_{f,k}$	entries of H^j corresponding to facet f and type k
h_f^j	entries of h^j corresponding to facet f

X^k	network subsystem with respect to type k
$\{x^p\}_{p\in P^k}$	extreme points of $\mathrm{conv}(X^k)$ except $\mathbf{0}$, corresponding to paths in P^k
λ_p	convex combination variable indicating corresponding to an extreme point x_p
$\xi_p \in \mathbb{Z}_+$ (or $x_p \in \mathbb{Z}_+$)	path variable defined as $\xi_p = b_0^k\lambda_p$, which equals the amount of flow used in path p
c_p	cost of path p

C.3 Notations in Chapter 5 and Chapter 6

ε_{CG}	column generation optimality tolerance
M_{CG}	maximum number threshold of column generation iterations
ρ_T	heuristic parameter for train-family branching
ρ_B	heuristic parameter for banned location branching
$r_1, r_2, \ldots$	branching rules
$L_j, j = 0, \ldots$	list of ordered branching rules
$I_0(r)$	the next inferior rule of r in L_0
$iniL$	initialisation level
G	jump gap
d_{BB}	the depth of the current BB tree
$d(n)$	the depth of node n
L	rising action limit in detecting a dive
R	backward check range in detecting a dive
it_F	maximum number of column generation iterations for leaf filter
ρ_{RC}	reduced cost threshold parameter for leaf filter
$\varDelta^0$	initial estimated upper bound increment
$\varDelta^k$	the k-th estimated upper bound increment
$\varDelta'$	constant estimated upper bound increment
r'_j	soft passenger demand for train j