UNIVERSITY OF LEEDS

This is a repository copy of *A branch-and-price approach for solving the train unit scheduling problem*.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/105182/

Version: Accepted Version

Article:
Lin, Z and Kwan, RSK (2016) A branch-and-price approach for solving the train unit scheduling problem. Transportation Research Part B: Methodological, 94. pp. 97-120. ISSN 0191-2615

https://doi.org/10.1016/j.trb.2016.09.007

© 2016 Elsevier Ltd. All rights reserved. This manuscript version is made available under the CC-BY-NC-ND 4.0 license http://creativecommons.org/licenses/by-nc-nd/4.0/.

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

White Rose
university consortium
Universities of Leeds, Sheffield & York

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk/

A branch-and-price approach for solving the train unit scheduling problem

Zhiyuan Lin *[1] and Raymond S. K. Kwan †[1]

[1]School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom

September 15, 2016

Abstract

We propose a branch-and-price approach for solving the integer multicommodity flow model for the network-level train unit scheduling problem (TUSP). Given a train operator's fixed timetable 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 train units. The TUSP is challenging due to its complex nature. Our branch-and-price approach includes a branching system with multiple branching rules for satisfying real-world requirements that are difficult to realize by linear constraints, such as unit type coupling compatibility relations and locations banned for coupling/decoupling. The approach also benefits from an adaptive node selection method, a column inheritance strategy and a feature of estimated upper bounds with node reservation functions. The branch-and-price solver designed for TUSP is capable of handling instances of up to about 500 train trips. Computational experiments were conducted based on real-world problem instances from First ScotRail. The results are satisfied by rail practitioners and are generally competitive or better than the manual ones.

Keywords: Rolling stock scheduling; Train unit scheduling; Integer multicommodity flow problems; Branch-and-price

1 Introduction

1.1 Train unit scheduling problem

A *train unit* is a self-propelled non-splittable fixed set of train *cars* (carriages), which is the most commonly used passenger rolling stock in the UK and many other European countries. A train unit is able to move in both directions on its own. Train units are classified into types having different characteristics. A train unit can be coupled with other units of the same or compatible types.

Given a rail operator's timetable on one operational day, a fleet of train units of different types and a rail network of routes, stations and infrastructures, the *train unit scheduling problem* (TUSP) (Lin and Kwan, 2014) refers to the planning of how timetabled trains are covered by a single or coupled train units from the fleet. From the perspective of a train unit, scheduling assigns a sequence of trains to it as its daily workload. A route refers to a unique path in a rail network between two locations. The

*Z.Lin@leeds.ac.uk (Corresponding author)

†R.S.Kwan@leeds.ac.uk

TUSP may also include auxiliary activities, e.g. empty-running insertion, coupling/decoupling control, platform assignment, platform/siding/depot capacity control, re-platforming, reverse, shunting movements from/to sidings or depots and unit blockage resolution. Capacity control is needed when there are several train units staying at the same berthing place such as a platform/siding/depot and it is to make sure the capacity of the berthing place is not exceeded. Re-platforming refers to the operation to shunt a train unit from its arrival platform to a different departure platform. Reverse activities usually occur at a dead-end platform such that the train unit will depart in the opposite direction to its arrival. A reverse activity will change the unit permutation if a train is formed by coupled units, where the unit permutation refers to the order of units (e.g. front, middle, rear) in a coupled formation. More details on the train unit operation rules can be found in Lin and Kwan (2014). In some cases, maintenance and unit overnight balance planning (or unit cycles) are also included; however in the UK, they are often achieved at later stages after train unit scheduling. A similar problem in the literature to the TUSP is the *train unit assignment problem* (TUAP) (Cacchiani et al., 2010b, 2012b, 2013b). Another relevant problem is *rolling stock circulation problem* (Schrijver, 1993; Alfieri et al., 2006; Peeters and Kroon, 2008), which however has different problem definitions than TUAP and TUSP.

1.2 A branch-and-price solver for TUSP

This paper addresses the particular issue of designing an efficient branch-and-price ILP solver for the integer multicommodity flow (IMCF) models used for the network-level TUSP. An IMCF model is proposed in Cacchiani et al. (2010b) for the TUAP, where the integer program is solved by an LP-based diving heuristic. A two-phase approach is presented in Lin and Kwan (2014) for the TUSP. The first phase is a fixed-charge IMCF model, where for most tested real-world instances the integer solver only generated columns at the root node in the BB tree, such that the solution qualities are not guaranteed to be exact. Although exact methods are used for the rolling stock circulation problems, the problem definitions are different from Cacchiani et al. (2010b) or Lin and Kwan (2014). For the TUAP/TUSP, there is no exact solution method in literature that is able to handle realistic problem sizes especially in the UK as far as the authors are aware.

A branch-and-price solver proposed in Lin and Kwan (2014) mainly uses columns from root node in the BB tree in most tested instances because of its limited capability. No customized branching or node selection strategy is used in the solver. No method is used to strengthen the weak LP-relaxation. The node-family variables are also not capable in handling all possible combination-specific coupling upper bound requirements. Compared with the aforementioned solver, significant improvements have been made in the proposed branch-and-price approach. It is capable of handling real-world instances of up to around 500 trains. The improvements include: (i) A branching system with multiple branching rules including two customized rules as train-family branching and banned location branching; (ii) An adaptive node selection method; (iii) A column inheritance strategy; (iv) A branch-and-bound (BB) system with estimated upper bounds and tree node reservation, and (v) the use of convex hulls for satisfying difficult real-life constraints.

Computational experiments were carried out based on real-world instances from First ScotRail. The solutions often outperform the manual schedules in many aspects and the practitioners from ScotRail are satisfied. Post-processing using TRACS-RS (Tracsis plc, 2016) to resolve the station-level issues like unit blockage and redundant coupling/decoupling is also shown to be successful for the instances tested.

1.3 Contributions

The main contribution of this paper is the design of a customized branch-and-price solver for solving the TUSP. While the TUAP studied by Cacchiani et al shares similar features of TUSP, the solution approaches for the TUAP are all based on heuristics. For instance, in Cacchiani et al. (2010b), an LP based diving heuristics is used for solving the integer multicommodity flow ILP model. In Cacchiani et al. (2013b), a fast heuristics is applied in conjunction with Lagrangian relaxation. Moreover, compared with the TUAP, additional real-life requirements that are crucial in UK railway operations are considered in the TUSP, such as train unit type compatibility relations, locations banned for coupling/decoupling and combination-specific coupling upper bounds. Customized strategies to accommodate those requirements have been implemented in the branch-and-price solver, such as local convex hulls, branching on train families and banned locations. Finally, some advanced techniques are used in the solver for speeding up the solution process, such as an adaptive node selection method combined with best-first and depth-first, column inheritance and estimated upper bound. Computational experiments have proved their usefulness for the tested real-world instances.

It is worth mentioning that some constraints considered in Cacchiani et al. (2010b), such as maintenance and overnight balance, are not included in our solver. This is because they are conventionally regarded as separate stages in the UK railway industry. Moreover, in Cacchiani et al. (2010b) and Cacchiani et al. (2013b), larger numbers of different unit types (up to 10 for real-world instances and 20 for realistic instances) are experimented. In our work, up to three different types are experimented, which is based on the real-world operational rules of the operator's network.

1.4 Organization of the paper

The remainder of this paper is organized as follows. Section 2 surveys the relevant literature. Section 3 describes the TUSP in more details. Section 4 presents the model and integer linear programming formulation. Section 5 gives the branch-and-price approach for solving the above formulation. Section 6 reports computational experiments based on real-world instances. Finally we conclude this paper in Section 7.

2 Literature review

With respect to the objects being planned, railway planning activities can often be divided into phases as train scheduling (timetabling), rolling stock scheduling and crew scheduling (Huisman et al., 2005), and possibly with rescheduling on the above stages (Cacchiani et al., 2014). Train scheduling or timetabling (Higgins et al., 1996; Carey and Crawford, 2007; Zhou and Zhong, 2007; Kroon et al., 2008; Cacchiani et al., 2010a; Mu and Dessouky, 2011; Barrena et al., 2014) defines the planned arrival and departure times of train trips to/from stations, yards, and sidings in timetables. When a timetable is fixed, rolling stock scheduling (Cordeau et al., 2001a; Lingaya et al., 2002; Alfieri et al., 2006; Cacchiani et al., 2010b) assigns rolling stock to cover the train trips in the timetable. Major kinds of rolling stock include locomotives with cars and train units. When trains and rolling stock schedules are given, the next step would be crew scheduling (Caprara et al., 1999; Kroon and Fischetti, 2001; Kwan, 2010; Shen et al., 2013) assigning duties to transport staff in accordance with train and rolling stock planning. Rescheduling (recovery) on the above three stages is also important for real-time disturbance and disruption management. Examples of railway rescheduling include Adenso-Díaz et al. (1999); Huisman (2007); Tornquist and Persson (2007); Burdett and Kozan (2009); Kroon et al. (2015); Kang et al. (2015); Zhan et al. (2015); Meng and Zhou (2011). See a survey on railway rescheduling in Cacchiani et al. (2014). For surveys on methodological approaches in general railway planning, see Caprara et al. (2007, 2011); Huisman et al. (2005).

2.1 Train unit resource planning

Train unit scheduling is a kind of rolling stock scheduling. Below, we review the researches that are directly pertinent to this field.

2.1.1 The train unit circulation problem

Although sharing a common feature as being a practice to assign train units to a given timetable, problem definitions for the train unit circulation problem are different from the TUAP and TUSP. For a train/trip at Nederlandse Spoorwegen Reizigers (NSR)[1], its unique predecessor and successor are given in advance. By predecessor and successor, 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 (Fioole et al., 2006; Maróti, 2006); 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 associated NSR-train. This is referred to as the "continuity requirement" (Fioole et al., 2006; Maróti, 2006). In the TUAP/TUSP, the connection possibilities among trains are left flexible such that no prescribed connection is imposed. The unit permutation (order) issue is also considered in the research on NSR. In the UK, the train timetables are not regular enough for pre-sequencing, and FIFO is thus not suitable for connecting most trains. In addition, circulation and rostering of train units are often left as a separate process in the UK railway industry.

Schrijver (1993) proposes an integer multicommodity flow model 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 used in most subsequent research on rolling stock circulation by NSR. The objective is to minimize 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. (2006) consider a problem scenario similar to Schrijver (1993). The same flow graph framework is used. The objective is to minimize 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 similar convex hull strengthening as in Schrijver (1993). 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. Issues like cycles, 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.

[1] A "train" at NSR does not have the same meaning as a "train" in the TUAP or TUSP. The unit composition within an NSR-train can be changed and it is trips, as subsections of a train, that cannot have their unit compositions changed.

Peeters and Kroon (2008) 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 unit inventories at stations are used. Dantzig-Wolfe decomposition is used for solving the model, where since the order of the joint flows are considered, the master problem is decomposed with respect to NSR-trains. Branch-and-price is applied to get integer solutions. 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 Alfieri et al. (2006) and Peeters and Kroon (2008) 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. This is called combining/splitting of a train and has been catered for in a customized way by a model proposed in Fioole et al. (2006). 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.

2.1.2 The train unit assignment problem and train unit scheduling problem

The *train unit assignment problem* (TUAP) shares similar definitions and settings with the TUSP, especially in the sense that no trains are pre-sequenced in advance and no coupling/decoupling is allowed en route. Cacchiani et al. (2010b) present an integer multicommodity flow model for the TUAP. For each train, there is a passenger demand in the number of seats and a coupling upper bound of two units. The model can also include additional constraints for maintenance and overnight balance. The model is based on a directed acyclic graph (DAG) 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 used there. A similar DAG framework is also used in this work and in Lin and Kwan (2014). The path formulation is reported to be more efficient in solving the TUAP. Taking advantage that no more than two units can be coupled, the LP-relaxation is strengthened by replacing relevant knapsack constraints by their dominants (see Cacchiani et al. (2013a)). An LP-based heuristic is designed for finding the integer solutions without using a branch-and-bound tree. This heuristic is reported to be crucial in finding feasible solutions in many of the tested instances. The above model and the heuristic method 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 able to find solutions 10–20% better than the manual solutions in practice. Experimented on a PC with Pentium 4, 3.2GHz, 2GB RAM, and an LP-solver of ILOG-CPLEX 9.0, the solution time ranges from 1878–1932 seconds without the additional maintenance and overnight balance constraints, and 5897–14105 seconds with those constraints. Compared with TUSP, TUAP does not consider locations banned for coupling/decoupling, unit type compatibility relations or combination-specific coupling upper bound. The objective function in the reported experiments is set to only minimize the number of used units, while in TUSP, mileage, number of empty-movements and other preferences are also included in the objective.

Cacchiani et al. (2013b) later develop a fast and effective heuristic method based on Lagrangian relaxation for the same problem as in Cacchiani et al. (2010b). 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. Computational experiments have been conducted for the same instances as in Cacchiani et al. (2010b) and for some larger instances of up to around 1000 trains. Comparisons with the method given in Cacchiani et al. (2010b) have been made. The same authors have proposed another fast heuristic for the same train unit assignment problem in Cacchiani et al. (2012b). 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. Real-world instances were tested and comparisons with the approaches in Cacchiani et al. (2010b, 2013b) were made. See Cacchiani (2007, 2009); Cacchiani et al. (2012a) for more of the research on the TUAP.

The train unit scheduling problem (TUSP) described in Lin and Kwan (2014) is similar to the TUAP (Cacchiani et al., 2010b), except that additional real-world requirements are considered, such as unit type coupling compatibility, locations banned for coupling/decoupling, combination-specific coupling upper bounds, and station-level unit blockage issues. A two-phase approach is presented since including station-level details into the main model would give an intractable huge-sized problem. This two-level decomposition framework is justified from railway operation practice and station-based simulations. In the reported experiments, the integer solver only generated columns at the root node of the BB trees for most of the tested instances such that the exactness of the solutions cannot be guaranteed. Moreover, the integer solver reported did not use customized branching strategies and the node selection method was fixed to best-first. No attempt was made at strengthening the lower bound given by LP-relaxation. The train-family variable method for ensuring coupling compatibility and combination-specific upper bounds also had limitations such that not all possible combination-specific upper bound scenarios can be handled.

While rolling stock scheduling usually assume unique and well-defined train capacity requirements, in practice rail operators may consider different levels of capacity provisions. In Lin et al. (2016), the problem of train unit scheduling with bi-level capacity requirements is studied and an integer multicommodity flow model is proposed based on previous research.

2.1.3 Other research in train unit planning

Jiang et al. (2014) propose a model for scheduling additional train unit services in a double parallel rail transit line such that timetable scheduling and train unit scheduling are integrated into the model with two objectives: minimizing travel times of additional trains and minimizing shifts of initial trains. Computational experiments on real rail transit line 16 in Shanghai were tested yielding a reasonable new timetable.

Lu et al. (2016) present a solution approach for scheduling train units under the condition of their utilizations on one sector or within several interacting sectors. The method consists of two stages: a greedy stage to construct a feasible circulation plan fragment, and a stochastic disturbance to generate a whole feasible solutions or get a new feasible solution. Computational experiments on a railway corridor representing the basic feature of railway networks were reported.

2.2 Locomotive scheduling

Being a different kind of rolling stock, the requirements and problem settings for locomotive scheduling are quite dissimilar from train unit circulation/assignment/scheduling. For instance, the carriages and locomotives can be scheduled separately with different operating rules. Locomotive scheduling and train unit scheduling both can be formulated as integer multicommodity flow problems, where often different commodities represent rolling stock types and nodes/arcs represent trips to be covered by rolling stock. Locomotive hauled train carriages are now rarely used for passenger rail transport in the UK and many other countries.

Wolfenden and Wren (1966) propose a heuristic method such that it can resize the problem scale small enough for the Hungarian algorithm to repeatedly solve updated instances until the converged 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 reported to be the world's first computer-produced working schedule (Wren, 2004).

Ziarati et al. (1999) 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 with 2000 trains and 26 locomotive types.

Cordeau et al. (2000, 2001a,b) 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. They considered 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. (2002) 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.

3 Problem description

We briefly list the requirements/restrictions and objectives for the TUSP. For a more detailed description, see Lin and Kwan (2014).

3.1 Requirements and restrictions

(1) Fleet size limit: Each train operator has a fleet of units of a limited number per type. A schedule whose used unit number exceeding the fleet size limit is invalid.

(2) Type-route compatibility: There are compatibility relations among unit types and routes. As each train belongs to a route, this turns out to be a compatibility relation among unit types and trains.

(3) Time allowances: A prerequisite for two trains to be consecutively connected by the same unit is that the arrival time of the previous train should be earlier than the departure time of the next train. Extra buffering time also has to be imposed on this time gap, as well as empty-running and shunting movements if relevant.

(4) Passenger demand: Each train in the timetable should be covered by unit(s) whose total capacity satisfies a passenger demand expected for the train, which is often measured in number of seats.

(5) Empty-running: Generating an empty-running train without passengers when necessary may make contribution to the overall solution quality. However, track paths and timing have to be checked and approved by authorities such that no conflict with other track users may occur.

(6) Unit blockage: Movements of rolling stock vehicles are strictly restricted by tracks and other rail infrastructures, which will give various blockage problems. Detecting and eliminating potential blockage requires the collection of comprehensive station-level infrastructure knowledge and operation details.

(7) Coupling/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.

(8) Type-type compatibility: Train units of the same type are permitted to be coupled and some different types are also allowed to be coupled. This relation is referred to as 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 in ScotRail and Southern Railway, train unit families partition the fleet into mutually exclusive subsets. Table 1 shows the unit type and family information of Southern Railway and ScotRail. The unit types are rewritten as "Type in paper" to be referred to in this paper for convenience. Note that this requirement will increase the difficulty of the problem as it is proved in Lin and Kwan (2016a) that the multicommodity flow problem with incompatible commodities is NP-hard even when the flows are allowed to be fractional (A general fractional multicommodity flow problem can be solved in polynomial time (Ahuja et al., 1993).)

(9) Complex coupling upper bounds: When train units are coupled into a unit block, the total number of cars is restricted by an upper bound. This upper bound is often combination-specific that it is difficult to describe the relevant rules in a unified form such as linear constraints. Table 2 gives the coupling upper bound as a result of different unit combinations.

(10) Locations banned/restricted for coupling/decoupling: At some locations coupling/decoupling operations are either forbidden or restricted under certain conditions. The proportion of banned/restricted locations can be large. Taking the ScotRail network for example, among the 75 locations that can be the end points of trains, coupling/decoupling is totally banned at 49 of them. Generally, locations banned for coupling/decoupling will make the overall problem more difficult to solve. We conjecture that the problem is NP-hard even for single unit type cases if the banned locations are imposed due to the fact that the single commodity confluent flow problem is NP-hard (Chen, 2005). Note that in Cacchiani et al. (2010b), it is proved that the TUAP is polynomial solvable for one type of unit.

Table 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	Real Type	Type in paper	Capacity (seats)	# of cars
ScotRail	A	c156	A1	145	2
	B	c158	B1	136	2
		c170	B2	189	3
		c170S	B3	198	3
	C	c314	C1	212	3
	D	c318	D1	219	3
		c320	D2	230	3
	E	c334	E1	183	3
	F	c380/0	F1	208	3
		c380/1	F2	282	4
Southern Railway	G	c171/7	G1	107	2
		c171/8	G2	241	4
	H	c455/8	H1	316	4
		c456/0	H2	152	2
	I	c313/1	I1	194	3
	J	c460/0	J1	366	8
	K	c442/1	K1	320	5
	L	c377/1	L1	223	4
		c377/2	L2	223	4
		c377/3	L3	160	3
		c377/4	L4	243	4

Table 2: Coupling upper bounds in number of cars associated with type combinations, regardless of other factors such as routes

Operator	Family	Combination	Upper bound (in # of cars)
ScotRail	A	A1	6
	B	any combinations by B1, B2, B3	6
	C	C1	6
	D	any combinations by D1, D2	6
	E	E1	8
	F	any combinations by F1,F2	7
Southern Railway	G	G1 only	4
		G2 only	8
		G1 and G2	6
	H	H1 only	8
		H2 only	6
		H1 and H2	8
	I	I1	3
	J	J1	8
	K	K1	10
	L	any combinations by L1,L2,L3,L4	12

3.2 Objectives and solution qualities

(1) Fleet size: Because of the high costs associated with leasing and maintaining a train unit, minimizing the number of used units is the most important objective, given the basic fleet limit requirement has been satisfied.

(2) Operational costs: Common operational costs include carriage-kilometer (mileage), empty-running movements, shunting movements, unnecessary coupling/decoupling, and so on. Carriage-kilometer 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, over-provision may be used to relocate unit resources across the network such that a smaller number of used units and/or less empty-running can be achieved. In this paper, shunting and unnecessary coupling/decoupling are processed in Phase-2 as a separate stage from the network flow model (see § 4.1).

(3) Preferences: Various preferences can be found in forming a real-world train unit schedule. For instance, one type of unit is more preferable than another over the same route. A practical schedule also needs plenty of long gaps during the off-peak time for unit maintenance. On the other hand, medium and long gaps may not be desirable at some evening times, which are regarded as irregular patterns.

4 Model description

4.1 Network-level and station-level

The entire train unit scheduling problem would be too huge-sized to be tractable if all detailed station-level restrictions such as unit blockage resolution are considered in a "once-for-all" process. Therefore, a two-phase strategy is proposed in Lin and Kwan (2014) to allow some of the station-level requirements to be omitted from a main network-level phase, whose aim is to globally assign train units to trains temporarily ignoring some station layout details such as unit permutation and the elimination of unit blockage. The results of the network-level phase (Phase-1) would be a set of sequenced train trips as the daily workload for a train unit in the fleet. For trains covered by multiple units, no unit permutation information is given and unit blockage may occur at the station level. The network-level phase can be regarded as an extended version of Cacchiani et al. (2010b) with more real-world constraints such as locations banned for coupling/decoupling, unit type coupling compatibility, combination-specific coupling upper bounds and so on. Phase-2 will take the raw results from Phase-1 in a station-by-station manner to further complete local shunting plans, remove unit blockage and assign unit permutations for coupled trains. This paper will only consider the network-level phase. In Lin and Kwan (2014), a mixed integer matching model is proposed to solve Phase-2. In practice, it is sufficient to use a visual interactive software, e.g. TRACS-RS (Tracsis plc, 2016), to handel the basic tasks realized by the above model. The results of Phase-1 are uploaded to the software and train connections at each station will be displayed in detail. A user of the software thus can adjust the schedules if unit blockage exists or assign correct unit permutations for trains with coupled units.

At the early stage of the research, Lin and Kwan (2014) has attempted the incorporation of an integer fixed-charge multicommodity flow formulation, which led to a difficult integer program that cannot be solved to exact optimality within practical time. Nonetheless, experiences and real-world data based experiments suggest that the solution quality often will not be compromised if the "fixed-charge" variables and constraints in Lin and Kwan (2014) are dropped since extra tasks realized by them such as connection time allowances involving coupling/decoupling and penalizing redundant coupling/decoupling can be effectively achieved in Phase-2 by TRACS-RS. Therefore, here a standard integer multicommodity flow problem omitting the "fixed-charge" variables is considered.

4.2 A network-level framework based on DAG representation

A network-level DAG representation (Cacchiani et al. (2010b), Lin and Kwan (2014)) is summarized here for completeness. For the DAG $\mathscr{G} = (\mathscr{N}, \mathscr{A})$, we define a node set $\mathscr{N} = N \cup \{s, t\}$, where N is the set of *train nodes* representing trains in the timetable, and s and t are the *source* and *sink* node. An arc set is defined as $\mathscr{A} = A \cup A_0$, 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 connection such that after serving train i a unit can continue to serve train j as its next task. To be specific, if the time gap between two trains are greater than or equal to the minimum turnaround time, then a connection arc will be added between the two nodes; otherwise, no arc between them will be created. When extra time such as empty-running or shunting is involved, it should be also considered in validating the relevant time slacks. 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. $\delta_-(j)$ and $\delta_+(j)$ denote all arcs that terminate at / originate from node j respectively. Finally, an *s-t* path $p \in P$ in $\mathscr{G}$ 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 model. As for type-route compatibility, type-graphs $\mathscr{G}^k$ representing routes each type $k \in K$ can serve are constructed based on $\mathscr{G}$, as well as the components of $\mathscr{G}$, e.g. P^k and $\mathscr{A}^k$ refer to the set of all paths and arcs in type-graph $\mathscr{G}^k$ respectively. We call an arc in a type-graph $\mathscr{G}^k, k \in K$ a *type-arc*; on the other hand, we call an arc in the original graph $\mathscr{G}$ a *block-arc*.

4.3 Arc formulation and train convex hulls

First, an arc formulation based on the network-level framework is given as,

$$(AF) \qquad \min \qquad \sum_{k \in K} \sum_{a \in \mathscr{A}^k} c_a x_a \tag{1}$$

$$\text{s. t.} \qquad \sum_{a \in \delta_+^k(s)} x_a \le b_0^k, \qquad \forall k \in K; \tag{2}$$

$$\sum_{a \in \delta_-^k(j)} x_a - \sum_{a \in \delta_+^k(j)} x_a = 0, \qquad \forall j \in N, \forall k \in K; \tag{3}$$

$$\sum_{k \in K_j} \sum_{a \in \delta_-^k(j)} H_{f,k}^j x_a \le h_f^j, \qquad \forall f \in F_j, \forall j \in N; \tag{4}$$

$$x_a \in \mathbb{Z}_+, \quad \forall a \in \mathscr{A}^k, \forall k \in K;. \tag{5}$$

In the above arc formulation (AF) (1)–(5), the Objective (1) is defined according to the models described in Lin and Kwan (2014) where fleet size costs, adjusted arc costs (consisting of carriage-kilometers and preferences) and empty-running costs are the three major costs and they have different weights in the objective. The carriage-kilometer of a train (node) is associated with an arc in the following way: The incoming arcs of each train node (including the sign-on arcs) will be given a cost reflecting the carriage-kilometer of that node. This also applies to the other train-related costs to be reflected in arcs. Preference costs (associated with arcs) are used to reflect preferences among different connection possibilities. For instance, long gaps during off-peak hours as mentioned in § 3.2 will have their preference costs multiplied by a sub-weight of 0.5. Carriage-kilometers and preferences are combined into adjusted arc costs for the arcs. Constraints (2) ensure the number of deployed units for each type $k \in K$ is within its fleet size limit b_0^k. Constraints (3) are to force the flow conservation balance at each train node. Constraints (4) are the "train convex hull" constraints to be described in detail later. They are used to satisfy the passengers' demands and the combination-specific coupling upper bounds. Finally Constraints (5) give the variable domain where $x_a \in \mathbb{Z}_+, \forall a \in \mathscr{A}^k, \forall k \in K$ indicates the flow amount (number of units) at type-arc $a \in \mathscr{A}^k$.

In order to handle the complex combination-specific coupling upper bounds, which are commonly seen in the fleets of Southern Railway and ScotRail, we use a method presented in Lin and Kwan (2016b) based on the computation of "local convex hulls" per train. In the rolling stock scheduling literature, similar ideas in constructing local convex hulls can be found in Cacchiani et al. (2010b, 2013a); Schrijver (1993); Fioole et al. (2006); Alfieri et al. (2006); Ziarati et al. (1999), where local convex hulls are solely used to strengthen LP relaxation though.

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. A *unit combination set* is defined as:

$$W_j := \left\{ w^j \in \mathbb{Z}_+^{K_j} \middle| \forall w^j : \text{a valid unit combination for train } j \right\}, \forall j \in N, \tag{6}$$

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.

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 (Barber et al., 1996). For computational feasibility in computing local convex hulls for problems with higher dimensions and larger number of points, see Lin and Kwan (2016b). Such convex hulls associated with each train are referred to as *train convex hulls* as

$$\text{conv}(W_j) = \left\{ w^j \in \mathbb{R}_+^{K_j} \middle| H^j w^j \leq h^j \right\}, \quad \forall j \in N, \tag{7}$$

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 have the following train convex hull constraints:

$$\sum_{k \in K_j} \sum_{a \in \delta_-^k(j)} H_{f,k}^j x_a \leq h_f^j, \quad \forall f \in F_j, \forall j \in N, \tag{8}$$

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

4.4 Path formulation

Let $Mx \leq b$ be the matrix form of Constraints (2)–(3), and let H and h be the coefficient matrix and right-hand side. Also let $X^k = \{x \in \mathbb{Z}_+^{\mathscr{A}^k} | M^k x^k \leq b^k\}$ and c^k, M^k, H^k, x^k the corresponding components with respect to type k. Assume that the coefficients in M, b, H, h are all integers. We rewrite (AF) as:

$$(AF') \quad \min \quad \sum_{k \in K} c^{kT} x^k \tag{9}$$

$$\text{s. t.} \quad \sum_{k \in K} H^k x^k \leq h; \tag{10}$$

$$x^k \in X^k, \forall k \in K. \tag{11}$$

It is well-known that $M^k, k \in K$ are totally unimodular (Cook et al., 1998). Therefore $Q^k = \{x \in \mathbb{R}_+^{\mathscr{A}^k} | M^k x \leq b^k\}, \forall k \in K$ are integral and the corresponding decomposed subproblems can be solved efficiently. Moreover, the following Theorem 1 can be derived from Gallo and Sodini (1978) and Vohra (2011):

Theorem 1. *An extreme point of $Q^k = \{x \in \mathbb{R}_+^{\mathscr{A}^k} | M^k x \leq b^k\}, b^k = (b_0^k, 0, \ldots, 0)^T$ is either the null point $\mathbf{0}$ or a feasible solution corresponding to a single path with a magnitude of b_0^k. A feasible solution corresponding to a single path with a magnitude of b_0^k is an extreme point of Q^k.*

Let $\{x^p\}_{p \in P^k}$ be the nonzero extreme points of $\text{conv}(X^k)$, and let $\hat{x}^p = \frac{1}{b_0^k} x^p$. By applying Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960), the following "extensive" formulation based on paths can be derived, where y_p represents the flow amount used on path p:

$$(PF) \quad \min \quad \sum_{k \in K} \sum_{p \in P^k} \left(c^{kT} \hat{x}^p\right) y_p \tag{12}$$

$$\text{s. t.} \quad \sum_{k \in K} \sum_{p \in P^k} \left(H^k \hat{x}^p\right) y_p \leq h; \tag{13}$$

$$\sum_{p \in P^k} y_p \leq b_0^k, \quad \forall k \in K; \tag{14}$$

$$y_p \in \mathbb{Z}_+, \quad \forall p \in P^k, \forall k \in K. \tag{15}$$

In fact (PF) can be re-expressed into a more "intuitive" formulation that is self-explanatory based on the meaning of y_p, which reads:

$$(PF') \quad \min \quad \sum_{k \in K} \sum_{p \in P^k} c_p y_p \tag{16}$$

$$\text{s. t.} \quad \sum_{p \in P^k} y_p \leq b_0^k, \quad \forall k \in K; \tag{17}$$

$$\sum_{k \in K_j} \sum_{p \in P_j^k} H_{f,k}^j y_p \leq h_f^j, \quad \forall f \in F_j, \forall j \in N; \tag{18}$$

$$y_p \in \mathbb{Z}_+, \quad \forall p \in P^k, \forall k \in K. \tag{19}$$

We leave the proof of the above equivalence between (PF) to (PF') to Appendix A.

It is worth mentioning that the lower bound given by the LP relaxation of the path formulation (PF') is the same as the one given the arc formulation (AF), since the decomposed parts $X^k, k \in K$ are integral. Due to the same reason, the lower bound given by the Lagrangian relaxation on (AF) by relaxing Constraints (4) is the same as the above two bounds.

Despite the same lower bound given by LP or Lagrangian relaxation, (PF') is usually more preferable to (AF), mainly because the former can be solved more efficiently by column generation based methods. This is also reported in Cacchiani et al. (2010b).

5 A branch-and-price solver

In this section, a branch-and-price solver for solving the path formulation (PF) (and (PF')) is presented, from which many of the techniques used can also be applied to the path formulations of the "fixed-charge" version in Lin and Kwan (2014), either directly or with certain modifications.

5.1 Branch-and-price and column generation

Branch-and-price (Barnhart et al., 1998) is a high-level framework for solving large-scale ILPs by combining branch-and-bound (BB) (Wolsey, 1998) and column generation (Lübbecke and Desrosiers, 2002; Desrosiers and Lübbecke, 2005). In a generic branch-and-bound framework, if only 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 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. Branch-and-price is used in Peeters and Kroon (2008) for solving the rolling stock circulation problem.

5.1.1 Restricted master problem and subproblems

Consider the path formulation (PF), whose LP relaxation is denoted as $(\overline{PF})$. Let $\widetilde{P}^k \subseteq P^k$ be the set of paths in the restricted master problem (RMP) and 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 (14) and (13) of the RMP of $(\overline{PF})$. We have the dual problem of the RMP of $(\overline{PF})$ as:

$$\max \quad \sum_{k \in K} b_0^k \phi_k + h^T \psi \tag{20}$$

$$\text{s. t.} \quad \phi_k + (H^k \hat{x}^p)^T \psi \leq c_p \quad \forall p \in \widetilde{P}^k, \forall k \in K; \tag{21}$$

$$\phi, \psi \leq \mathbf{0}, . \tag{22}$$

There are $|K|$ separation problems from the dual in finding constraints associated with $p \in P^k$ in each $\mathscr{G}^k$ 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

$$\overline{c}_k^*(\phi, \psi) := \min_{p \in P^k} \{c_p - \phi_k - (H^k \hat{x}^p)^T \psi\}. \tag{23}$$

In the TUSP, the cost of path p can be defined in the form of $c_p = \sum_{j \in N_p} c_j + \sum_{a \in A_p} c_a + c_p^0$ (where the first term is node-related, the second term is arc-related and the third term is path-related). The subproblem (23) can be regarded as a shortest path problem with a node weight $c_j - \sum_{f \in F_j} H_{fk}^j \psi_{fj}$ at each j, an arc weight c_a at each a plus a source-sink weight $c_p^0 - \phi_k$. To be more specific, we have

$$\begin{aligned} \overline{c}_p &= c_p - \sum_{f \in F_j, j \in N} (H^k \hat{x}^p)_{fj} \psi_{fj} - \phi_k \\ &= c_p^0 - \phi_k + \sum_{j \in N_p} \left(c_j - \sum_{f \in F_j} H_{fk}^j \psi_{fj} \right) + \sum_{a \in A_p} c_a. \end{aligned}$$

The above expression on reduced cost can also be directly obtained from (PF'). There is an equivalence relation between the ILP under the network embedded in X^k and a shortest path problem over the same network, since (23) can be understood as

$$\begin{aligned}
\overline{c}_k^*(\phi,\psi) &= \min_{p\in P^k} \left\{ c^{kT}\hat{x}^p - (H^k\hat{x}^p)^T\psi - \phi_k \right\} \\
&= \min_{x\in X^k} \left\{ \frac{1}{b_0^k}\left[c^{kT}x - (H^kx)^T\psi \right] - \phi_k \right\} \\
&= \min_{x\in X^k} \left\{ \frac{1}{b_0^k}\left(\sum_{a\in\mathscr{A}^k} c_a x_a - \sum_{j\in N, f\in F_j} \sum_{a\in\delta_-^k(j)} H_{fk}^j \psi_{fj} x_a \right) - \phi_k \right\},
\end{aligned} \tag{24}$$

i.e. a shortest path problem for type k with modified costs.

Since the underlying graph $\mathscr{G}^k$ is a DAG, the shortest path problem can be solved efficiently by topological sorting in the linear rate of $O(|N^k|+|\mathscr{A}^k|)$ (Ahuja et al., 1993).

5.1.2 Upper and lower bounds

When solving (PF) by column generation, during each iteration a pair of upper and lower bounds on the master problem's objective value z_{MP}^* can be obtained. The advantage of taking records of the bounds is that it is possible to terminate the column generation process before it is solved to optimality when the gap between the upper and lower bound is regarded as accepted. Then the lower bound can be used as the BB tree's node relaxation value.

It is by default that the objective value of the current RMP before optimality gives an upper bound on z_{MP}^*. Theorem 2 shows that a lower bound can be obtained by considering the value of the subproblem results $\overline{c}_k^*, k\in K$ and assuming an upper bound on the sum of the optimal master problem variables can be found (Desrosiers and Lübbecke, 2005; Lübbecke and Desrosiers, 2002).

Theorem 2. *For each $k\in K$, let $\kappa_k \geq \sum_{p\in P^k} y_p^*$, where $y_p^*, p\in P^k$ are the elements of type k in the optimal solution of the master problem $(\overline{PF})$. Then*

$$z_{\text{RMP}}^*(\phi,\psi) + \sum_{k\in K} \kappa_k \overline{c}_k^*(\phi,\psi) \leq \sum_{k\in K, p\in P^k} c_p y_p^* = z_{\text{MP}}^*. \tag{25}$$

Proof. Noting that $\sum_{p\in P^k} y_p^* \leq b_0^k, \sum_{k\in K, p\in P_j^k} H_{fk}^j y_p^* \leq h_f^j$ and $\overline{c}_k^*, \phi, \psi \leq 0$, we have

$$\begin{aligned}
z_{\text{MP}}^* - z_{\text{RMP}}^*(\phi,\psi) &= \sum_{k\in K, p\in P^k} c_p y_p^* - \left(\sum_{k\in K} b_0^k\phi_k + \sum_{f\in F_j, j\in N} h_f^j \psi_{fj} \right) \\
&\geq \sum_{k\in K, p\in P^k} c_p y_p^* - \left(\sum_{k\in K, p\in P^k} y_p^*\phi_k + \sum_{f\in F_j, j\in N, k\in K, p\in P_j^k} H_{fk}^j y_p^* \psi_{fj} \right) \\
&= \sum_{k\in K, p\in P^k} c_p y_p^* - \left(\sum_{k\in K, p\in P^k} y_p^*\phi_k + \sum_{f\in F_j, j\in N_p, k\in K, p\in P^k} H_{fk}^j y_p^* \psi_{fj} \right) \\
&= \sum_{k\in K, p\in P^k} y_p^* \left(c_p - \phi_k - \sum_{f\in F_j, j\in N_p} H_{fk}^j \psi_{fj} \right) \\
&\geq \sum_{k\in K, p\in P^k} y_p^* \overline{c}_k^*(\phi,\psi) \\
&\geq \sum_{k\in K} \kappa_k \overline{c}_k^*(\phi,\psi),
\end{aligned}$$

which finishes the proof. □

Corollary 1. *At iteration i in the column generation process in solving $(\overline{PF})$, let ϕ^i, ψ^i be the corresponding optimal dual variables. A pair of bounds on the optimal objective value of the master problem $(\overline{PF})$ can be obtained as*

$$z^*_{\text{RMP}}(\phi^i, \psi^i) + \sum_{k\in K} \kappa_k \bar{c}^*_k(\phi^i, \psi^i) \leq z^*_{\text{MP}} \leq z^*_{\text{RMP}}(\phi^i, \psi^i) \tag{26}$$

Since the lower bound may not be monotonically increasing, a tighter pair of bounds at iteration i can be used in practice:

$$\max_{l=1,\dots,i} \left\{ z^*_{\text{RMP}}(\phi^l, \psi^l) + \sum_{k\in K} \kappa_k \bar{c}^*_k(\phi^l, \psi^l) \right\} \leq z^*_{\text{MP}} \leq z^*_{\text{RMP}}(\phi^i, \psi^i). \tag{27}$$

As for choosing appropriate values for parameters κ_k, two strategies are discussed. First, it is possible to set $\kappa_k = b^k_0$. The lower bound becomes $z^*_{\text{RMP}}(\phi, \psi) + \sum_{k\in K} b^k_0 \bar{c}^*_k(\phi, \psi)$. This is the same lower bound if the original arc formulation (AF') is solved by Lagrangian relaxation with the same dual variables ψ associated with the penalized Constraints (10). Let $L(\psi)$ be the Lagrangian lower bound and $\mathscr{L} = \max_{\psi\leq 0} L(\psi)$ be the optimal solution of the Lagrangian dual problem. We have

$$\begin{aligned} L(\psi) &= \min_{x\in X} \left\{ \sum_{k\in K} c^{kT} x^k - \psi^T \left(\sum_{k\in K} H^k x^k - h \right) \right\} \\ &= \sum_{k\in K} b^k_0 \min_{x^k\in X^k} \left\{ \frac{1}{b^k_0} \left[c^{kT} x^k - \psi^T (H^k x^k) \right] \right\} + \psi^T h \\ &= \sum_{k\in K} b^k_0 \min_{p\in P^k} \left\{ c^{kT} \hat{x}^p - \psi^T (H^k \hat{x}^p) \right\} + \psi^T h. \end{aligned} \tag{28}$$

The resulting Lagrangian dual in finding $\mathscr{L}$ 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, such that

$$\mathscr{L} = \max \quad \sum_{k\in K} b^k_0 \phi_k + \psi^T h \tag{29}$$

$$\text{s. t.} \qquad \phi_k \leq c^{kT} \hat{x}^p - \psi^T (H^k \hat{x}^p), \quad \forall p \in P^k, \forall k \in K; \tag{30}$$

$$\phi, \psi \leq \mathbf{0}. \tag{31}$$

One can verify the Lagrangian dual problem (29)–(31) is exactly the dual problem of the path formulation (PF). 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 (25) with $\kappa_k = b^k_0$, i.e.,

$$\begin{aligned} L(\psi) &= \sum_{k\in K} b^k_0 \min_{p\in P^k} \left\{ c^{kT} \hat{x}^p - \psi^T (H^k \hat{x}^p) - \phi_k \right\} + \psi^T h + \sum_{k\in K} b^k_0 \phi_k \\ &= \sum_{k\in K} b^k_0 \bar{c}^*_k(\psi, \phi) + \psi^T h + \sum_{k\in K} b^k_0 \phi_k. \end{aligned} \tag{32}$$

Alternatively, let $\tilde{y}^*(\phi, \psi)$ be the optimal solution of the RMP with dual variables ϕ, ψ. It is true that $\sum_{p\in P^k} y^*_p \leq \sum_{p\in P^k} \tilde{y}^*_p(\phi, \psi) \leq b^k_0$ for each $k \in K$. Therefore, by setting $\kappa_k = \sum_{p\in P^k} \tilde{y}^*(\phi, \psi)$ promptly in each round of column generation LP, a tighter lower bound on z^*_{MP} can be obtained as $z^*_{\text{RMP}}(\phi, \psi) + \sum_{k\in K} \sum_{p\in P^k} \tilde{y}^*_p(\phi, \psi) \bar{c}^*_k(\phi, \psi)$.

5.2 Branching strategies

Here we propose a system of branching strategies based on BB. A major difference between this system and an ordinary BB method 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 being "operable" means: (i) all (path or arc) variables are integral, (ii) all coupled trains are served by units of compatible types, and (iii) there is no coupling/decoupling at banned/restricted locations. Moreover, the term "feasible" will be solely used in the context for showing whether an LP (relaxation) problem is feasible.

In developing an efficient BB method, two issues have to be concerned problem-specifically, i.e. branch design and node selection. 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. We will discuss the details of them in the following sections.

5.2.1 Branch design

The essence of branching is to partition the solution space by imposing mutually exclusive restrictions in each branch without losing any potential operable ones. The imposed restrictions could be implemented by explicitly discarding the offending solutions instead of additional LP constraints. In a traditional BB framework, branching is generally employed to make fractional variables integral. However, in our work, branching is also used as a method to realize the requirements that are difficult to satisfy by constraints. Moreover, if used appropriately, such branches can both satisfy those tough requirements as well as effectively divide the solution space, making the "fractional-to-integer" process more efficient.

A multi-rule branching system is thus designed, where three branching rules are involved: train-family branching, banned location branching, and arc variable branching. The first two rules are compatible with the structure of the sub-problems in a branch-and-price framework, since corresponding modifications on the subproblem networks will take place to be consistent with the branching.

5.2.1.1 Train-family branching

In forming the train unit combination set W^j in (6), the combinations with incompatible types are not included in this discrete set. However as discussed in applying train convex hulls (see § 4.3), some invalid combinations can be still inside the continuous set $\mathrm{conv}(W^j)$ and thus may appear in a relevant LP relaxation solution.

Train-family branching is thus used to eliminate the remaining type-incompatible combinations. It is more advantageous than the train-family variables described in Lin and Kwan (2014) since it does not require additional binary variables and, in conjunction with train convex hulls, it is suitable for a broader range of combination-specific coupling upper bounds compared with the method in Lin and Kwan (2014). To form such branches, the solution of the relevant LP relaxation at a BB tree node will be checked and the earliest departure train covered by more than one family will be identified. Due to the nature of DAG, the flow type covering an earlier train is likely to have immediate impact on its subsequent trains. Computational experiments also suggest there is empirical evidence for the advantage of this prioritization on train's departure times. Suppose such a train j is detected with families $\varphi_1, \ldots, \varphi_n$ (n is usually not a large number) serving it and let Φ_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 $1, \ldots, n$, say at the i-th branch where $i \in \{1, \ldots, n\}$, only family φ_i is allowed to serve train j. To achieve this, in the RMP all paths indicating any family in $\Phi_j \setminus \{\varphi\}$ serving 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 $\Phi_j \setminus \{\varphi_1, \ldots, \varphi_n\} \neq \emptyset$, then families $\varphi_1, \ldots, \varphi_n$ will be forbidden to serve j, which can be realized by similar path/node deleting schemes as described above; if $\Phi_j \setminus \{\varphi_1, \ldots, \varphi_n\} = \emptyset$, then the $(n+1)$-th branch is no longer needed.

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.

5.2.1.2 Banned location branching

The banned location branching rule aims at ensuring no coupling/decoupling operation takes place at locations banned/restricted for coupling/decoupling. Let N_B^- (N_B^+) be the set of train nodes where its departure (arrival) station is banned for coupling/decoupling. Then for any $j \in N_B^-$ (N_B^+), there should be one and only one used incoming (outgoing) arc among $\delta_-(j)$ ($\delta_+(j)$) in a valid solution.

Similar to train-family branching, banned location branching will check the LP relaxation solution at a BB tree node and select a train j among N_B^- or N_B^+ that has multiple used block-arcs in $\delta_-(j)$ or $\delta_+(j)$. Also due to the DAG's nature, the checking order is set to follow the departure times of trains with banned locations. 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 $A_j \setminus \{a_1, \ldots, a_n\} = \emptyset$, 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. However unlike train-family branching where often only a very small number of families are involved for each train, the number of flowed arcs linked with a banned location train can be much larger and the number of unused arcs corresponding to the last $(n+1)$-th branch can be huge. Therefore, heuristics may be needed in practice that a parameter $0 < \rho_B < 1$ is set to only allow the arcs with a flow proportion greater or equal to ρ_B to be considered in forming branches, i.e. only block-arcs a with $\frac{\sum_{k \in K} \sum_{p \in P_a^k} y_p}{\sum_{k \in K} \sum_{p \in P_j^k} y_p} \geq \rho_B, a \in \delta_-(j)$ or $a \in \delta_+(j)$, will be used.

5.2.1.3 Arc variable branching

Since the flow variables are required to be integers, the traditional "fractional-to-integral" branching is needed, which will be processed based on individual type-arc variables $x_a = \sum_{p \in P_a^k} y_p$. Experience suggests that branching on the "most fractional" arc whose decimal part is nearest to 0.5 often gives the best performance for tested instances. When there is a tie, the arc associated with an earlier departure time will be chosen, and if there is a second tie, the arc with the tightest connection time will be chosen. In forming branches, we use the method given in Alvelos (2005), where a fractional arc $a \in \mathscr{A}^k$ is branched by two explicit constraints as

$$\sum_{p\in P_a^k} y_p \leq \lfloor x_a \rfloor, \quad \sum_{p\in P_a^k} y_p \geq \lceil x_a \rceil. \tag{33}$$

Constraints (33) will be used except the case of $\lfloor x_a \rfloor = 0$, where arc a can be deleted by removing the associated paths in the RMP and the associated arc in the subproblem. Constraints (33) are added in the RMP. Let $\mu_a^- \leq 0$ and $\mu_a^+ \geq 0$ be the dual variables associated with the left and right part of Constraints (33) respectively for arc a. Then the reduced cost for path p will be modified as $\bar{c}_p = c_p^0 - \phi_k + \sum_{j\in N_p}\left(c_j - \sum_{f\in F_j} H_{fk}^j \psi_{fj}\right) + \sum_{a\in A_p}(c_a - \mu_a^+ - \mu_a^-)$.

5.2.1.4 The order of the branching rules

Employing several branching rules, it is necessary to control their order (or priority) used during the BB process. For instance, if train-family branching has a higher priority than arc variable branching, when both of them can be used, the system will use the former. In this case there is an order that "train-family branching $\succ$ arc variable branching". If no branching object is available for the current rule used, the next inferior rule will be used, and so on. When no objects are available for any rules, the BB process is accomplished.

The order on branching rules can either be unchanged (static) or updated according to the current status (dynamic). Suppose there are n branching rules $r_1, \ldots, r_n$. In either case, an initial list L_0 representing an order among them at the beginning is set as

$$L_0 = \langle r_{\pi_1}, r_{\pi_2}, \ldots, r_{\pi_n} \rangle, \text{where } \{\pi_1, \ldots, \pi_n\} = \{1, 2, \ldots, n\}, \tag{34}$$

which states that $r_{\pi_1} \succ r_{\pi_2} \succ \cdots \succ r_{\pi_n}$.
(i) *Static rule ordering* The order is static if it is always the same as the initial list L_0, which is specified in Algorithm 1. In this algorithm, `objectAvailable`(r) is a boolean valued function to check whether there are available objects for rule r in the LP relaxation solution.

Algorithm 1 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 $n_{\mathrm{BB}}$
Result: the chosen branching rule $r^*$ stored in $n_{\mathrm{BB}}$ for later use
Begin:
initialise $r^* \leftarrow null$
for all $r \in L_0$ do
  if objectAvailable$(r)$ = TRUE then
    $r^* = r$
    break
  end if
end for
if $r^* = null$ then
  an operable solution found at $n_{\mathrm{BB}}$
end if
```

(ii) *Dynamic rule ordering*

The dynamic strategy requires the order list to be updated according to the current status. Once updated, the new checking order will be locally passed down to descendants of the current node being checked. The dynamic strategy is given in Algorithm 2.

Algorithm 2 Dynamic branching rule ordering

```
Given:
  a list of ordered branching rules $L_j = \langle r_1, \ldots, r_n \rangle$
  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 objectAvailable$(r) = \mathrm{TRUE}$ 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
```

5.2.2 Node selection method

Node selection in a branch-and-bound process refers to the decision making on how to select a next node to solve from the active queue when the BB process needs to be continued. An adaptive node selection method combining both best- and depth-first is used for the branch-and-price solver. The goal is to achieve as many as possible "dives" towards operable solutions, while keeping the solution qualities and avoiding being trapped at the middle-levels of the tree. It can be summarized as

(i) Do depth-first.

(ii) During depth-first, if a *jump condition* is triggered, the search will jump to an active node with the best value. The active nodes before the selected best node in the active queue will be repositioned to after the current tail in their original order.

(iii) After a jump, continue with depth-first.

5.2.2.1 Jump condition

The BB search starts with depth-first. The qualities of the early nodes yielded by depth-first are often not satisfactory, as their objective values are not considered for node selection. Moreover, two cases should be noted. First, although diving is expected, depth-first search may experience oscillations trapped at the middle levels on the tree without any effective dive at all. Second, intuitively though, when the current node yields an operable solution, the probability of finding other operable solutions in its neighborhood is likely to be low. In the above two cases, a "jump" will be triggered such that depth-first is stopped temporarily and the next node to be solved will be a best active node with the smallest objective. To be specific, a jump will be triggered at a current node n if the below is TRUE:

$$\left[(noJump(n) > G) \bigwedge \mathtt{notInDiving}(n) \bigwedge \frac{d_{\mathrm{BB}} - d(n)}{d_{\mathrm{BB}}} > \varepsilon_{BB} \right] \bigvee \mathtt{oprSol}(n). \tag{35}$$

In (35), $noJump(n)$ is a counter recording how many nodes have been searched at tree node n since the last jump action. G is a "jump gap" that controls the possibly minimum distance in number of traversed nodes between two jumps. When $G = 0$, the searching is similar to best-first (assuming not in diving) and when $G = \infty$, it is close to depth-first. `notInDiving`(n) detects whether current searching is "not in diving" (returning TRUE or FALSE). Two parameters are set in determining the diving status: a backward check range R and a rising action limit L. To check whether depth-first searching is in diving, R times of comparisons will be made among the recently searched nodes starting from node n to node $n-R$, i.e. the R-th earlier searched node before n. For $i = n \dots n-R$, each time `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 than its former node $i-1$, then a "rise" is identified. If within R times of comparisons, more than L times of rises are detected, `notInDiving`(n) will return TRUE. Note that if $R > G$, then R should be reset to G.

Figure 1 and 2 give an example of how `notInDiving`(n) determines the status at node $n = 11$. The numbers marked inside the nodes indicate the order of how they are searched. Suppose from node 1 to node 11 depth-first is in use, the current inoperable node $n = 11$ while in (35) the other conditions in the first disjunction term except `notInDiving`(n) are already satisfied. Also assume that $R = 8, L = 2$. As shown in Figure 1, there are two rises within the checking range, i.e. between node 6 and 7, and between node 10 and 11. They are both caused by the fact that the children of node 4 and node 8 are both cut-off thus the diving cannot continue (see Figure 2). Therefore, at node 11, a jump is triggered to relocate the search to node 12 which is currently the best node in the active queue.

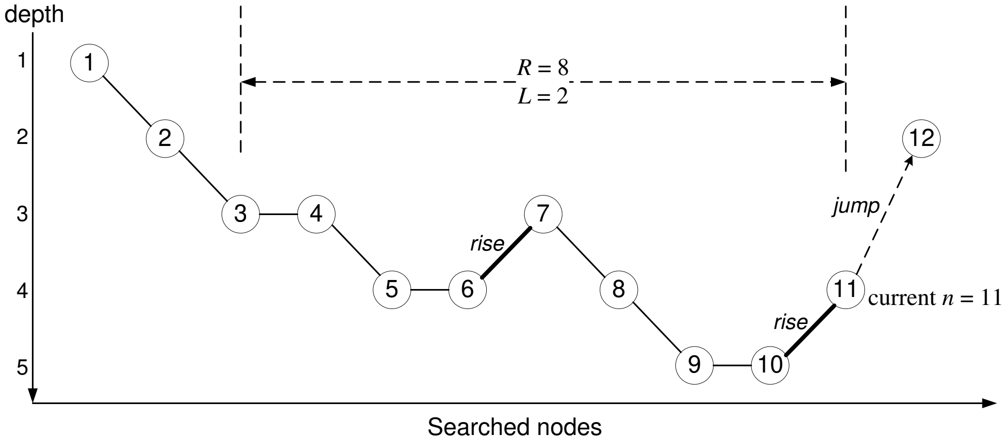

Figure 1: Searched nodes and their depths for the `notInDiving`(n) example

The third term $\frac{d_{\text{BB}} - d(n)}{d_{\text{BB}}} > \varepsilon_{\text{BB}}$ ensures jumps will not take place when n is within an ε_{BB}-percent distance from the BB tree's deepest places, where the possibility of finding operable solutions is generally higher. $d(n)$ and d_{BB} are the depth of current node n and the entire BB tree respectively.

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

5.3 Other technical issues of the branch-and-price solver

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

A primal heuristic based on a greedy algorithm is used to construct an initial feasible solution to trigger the column generation. The basic idea is to select the tightest connection arc (with respect to

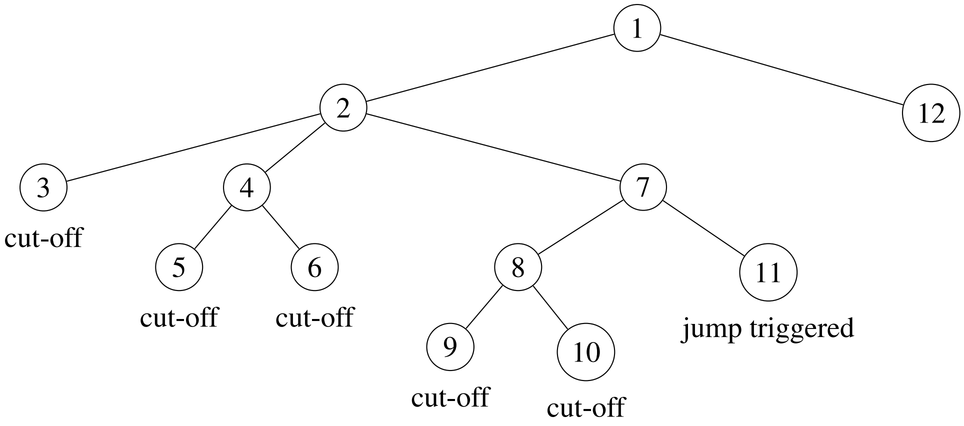

Figure 2: The BB tree for the `notInDiving`(n) example

turnround time) for connecting the next train at each train node. The resulting solutions are integer flows satisfying passenger demands, coupling upper bounds and coupling/decoupling time allowance while type-type compatibility and banned location restrictions are ignored. For some instances, it 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 types 171/7 and 171/8, the greedy method uses 15 units while the fleet size in the manual schedule is 13 units. Note that initial feasible solutions generated by the primal heuristic often cannot provide a global upper bound at the start point for branch-and-price since requirements on type-type compatibility and banned locations are not included.

To keep the initial solution "LP-feasible" even if the number of used units exceeds b_0^k for some type k, artificial variables $\beta^k \geq 0$ associated with each k for Constraints (17) are added with big-M costs in the objective. To be specific, we have the following constraints

$$\sum_{p \in P^k} y_p \leq b_0^k + \beta^k, \quad \forall k \in K \tag{36}$$

to replace (17) and the term $M \sum_{k \in K} \beta^k$ is added in the objective.

Finally when the primal heuristic cannot yield an "LP-feasible" solution, an alternative process will be invoked. It simply constructs a set of $\sum_{j \in N} |K_j|$ paths each containing one and only one train $j \in N$ served by one type of unit. These paths are called "naive columns" and they can always give an initial feasible solution as long as the problem is "LP-feasible". Column management strategies are also designed to remove some or all of such naive columns beyond the root. It is often observed that naive columns yield no worse subsequent performance than the aforementioned greedy method.

5.3.2 Initial feasible solutions at leaves: column inheritance

The above primal heuristic is not used at leaf nodes, mainly because it cannot consider accumulated branching requirements. 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 branching and also prevents inheritance of columns from other inappropriate nodes—e.g. its parent's siblings, where the branching is opposite. Another advantage in using inherited columns is that subsequent column generation often becomes very easy to solve, usually requiring only a very small number of column generation iterations, as shown by experiments. Moreover, it is observed that the "tail-off" effect (Lübbecke and Desrosiers, 2002; Desrosiers and Lübbecke, 2005) is hardly found at leaf nodes if column inheritance is used.

When the inherited columns fail in constructing an initial feasible solution, it does not necessarily indicate the infeasibility of a current leaf node. When this happens, naive columns similar as those described in § 5.3.1 will be generated to construct a feasible solution. If naive columns also fail, then the node will be claimed infeasible.

5.3.3 Column generation early termination

There is an option in the branch-and-price solver to stop column generation early before it is fully solved to optimality, as suggested by Barnhart et al. (1998). This is especially useful when column generation is triggered by naive columns rather than inherited columns, as the tail-off effect is frequently encountered when the former are used. At the end of each CG iteration, a pair of upper and lower bounds UB, LB on the optimal objective of the master problem can be obtained from (27). If their gap is smaller than a tolerance ε_{CG}, the LP can be regarded as sufficiently optimal and the lower bound will become the node's LP relaxation value. Options are available in whether the gap is measured absolutely $UB - LB \leq \varepsilon_{CG}$, or relatively $\frac{UB-LB}{UB+1} \leq \varepsilon_{CG}$.

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

5.3.4 Estimated global upper bounds and node reservation

In order to speed up the BB process by temporarily ignoring tree nodes with high values, an artificial estimated global upper bound UB^0 is set up as $UB^0 = z^*(0) + \Delta z^0$, where $z^*(0)$ is the root's objective value and Δz^0 is the initial increment value usually set based on experience. Let UB be the real global upper bound either initialized as $+\infty$ or given by operable solutions found in the BB search in later stages.BB search is then carried out with an estimated global upper bound UB^0 in conjunction with UB. As long as $UB^0 < UB$, the following rule will be used for a BB node n that is LP-feasible but non-operable with an objective value $z(n)$:

(i) If $z(n) \leq UB^0$, then put n into the current active queue;

(ii) If $UB^0 < z(n) < UB$, then put n into the reservation queue;

(iii) If $z(n) \geq UB$, then cut-off n by bound.

Note that nodes in the reservation queue will neither be part of the current active queue nor be abandoned as long as $UB^0 < UB$. Moreover, an operable node n with $UB^0 < z(n) < UB$ will update the real upper bound by $UB := z(n)$.

If UB^0 fails in finding optimal operable solutions, it implies a temporarily empty active queue bounded by UB^0. Then the estimated upper bound will be updated to $UB^1 = UB^0(1 + \Delta^0)$, where Δ^0 is a percentage increment value for the first update. Accordingly, the nodes in the reservation queue having objective values between UB^0 and UB^1 will become the new active queue. If the new active queue becomes empty, analogous process will take place for subsequent $k = 1, 2, \ldots$ with $UB^{k+1} = UB^k(1 + \Delta^k), k = 1, 2, \ldots$, where the parameter Δ^k can be set adaptively according to the current status or as a constant $\Delta^k = \Delta', k = 1, 2, \ldots$.

If an operable solution can be found within the active nodes bounded by a UB^k (i.e. an operable solution at n with $z(n) \leq UB^k$) for the first time, it implies that $UB^k \geq UB = z(n)$ and the reservation queue will be cleared since the reserved nodes will never be used. The estimated upper bound is no longer needed and the BB will continue only using the newest UB as its global upper bound.

The exactness of branch-and-bound is preserved under this scheme. The algorithmic description of this scheme is given in Algorithm 3.

Algorithm 3 Estimated upper bound for BB

```
$UB := +\infty$, $k := 0$
$noOperUB^k :=$ true // true if no operable solution found within $UB^k$, false otherwise
$UB^k := z^*(0) + \Delta z^0$
while $activeQueue \neq \emptyset$ do
    if $noOperUB^k$ then
        solve the next active node $n$ from $activeQueue$
        if $n$ is LP-infeasible then
            cut-off by infeasibility
        else
            if $n$ is operable then
                if $z(n) \leq UB^k$ then
                    $noOperUB^k :=$ false
                else if $UB^k < z(n) < UB$ then
                    $UB := z(n)$
                end if
            else
                if $z(n) \leq UB^k$ then
                    $activeQueue+ = \{n\}$
                else if $UB^k < z(n) < UB$ then
                    $reservationQueue+ = \{n\}$
                else
                    cut-off by bound
                end if
            end if
        end if
        if $activeQueue = \emptyset$ then
            $k := k + 1$
            $UB^k = UB^{k-1}(1 + \Delta^{k-1})$
            $activeQueue := \{n : n \in reservationQueue$ and $UB^{k-1} < z(n) \leq UB^k\}$
        end if
    else
        conduct standard BB with current $activeQueue$ and $UB$
    end if
end while
```

6 Computational experiments

This section reports computational experiments for the TUSP using the customozed branch-and-price solver. They are based on real-world instances from First ScotRail, the major passenger train operator in Scotland when the research was conducted. The experiments were 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. The solver only utilized the simplex solver of Xpress-MP to solve LP relaxations during the column generation processes without employing the default integer programming solver of Xpress-MP. Most of the post-processing tasks were carried out on TRACS-RS 1.0.177.0.

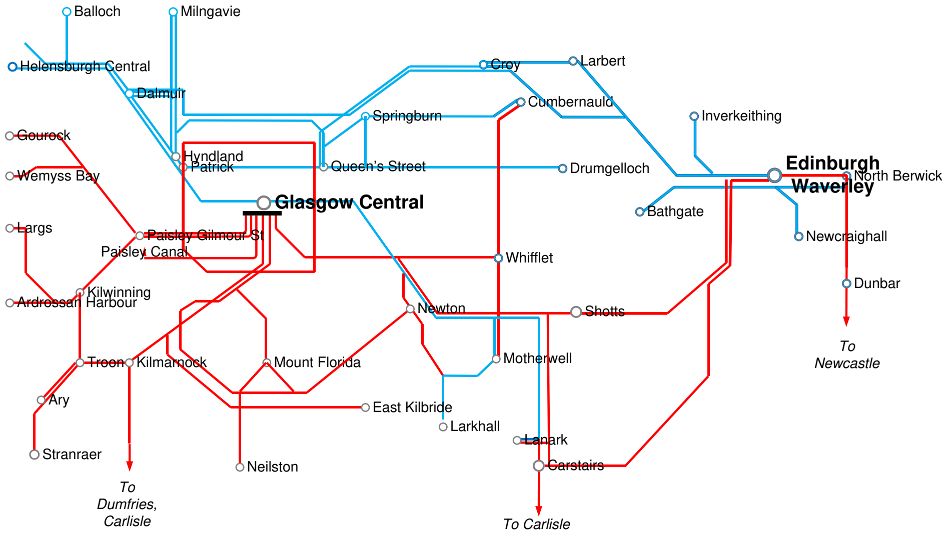

Figure 3: The GA (red) and GB (blue) areas of the ScotRail network

6.1 First ScotRail dataset

6.1.1 Routes and services of ScotRail

First ScotRail has a daily timetable of around 2250 trains. The more than two thousand services have been traditionally divided into three relatively independent areas by the manual schedulers, namely the GA, GB and GC areas. GA and 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 3 gives an illustrative network map of the central belt. The GA and GB areas were tested in this research.

GA Area. As suggested by the practitioners from ScotRail, only the GA routes served by electric units (known as "GA Electrics") were involved in the experiments. This excludes the trains served by the diesel units of A1 and B1 which are relatively independent. In the May 2014 timetable, there are 703 timetabled trains in GA Electrics. The manual solution uses 14 units of C1, 20 units of F1 and 15 units of F2 to cover the 703 trains. F1 and F2 are coupling-compatible while C1 can only be coupled with its own type. Note that the services in GA Electrics generally have very dense patterns since on average one unit covers about 14.3 trains per day in the manual schedules.

GB Area. In the May 2014 timetable, there are 510 timetabled services in the GB area, to be served by 19 units of D1, 20 units of D2 and 36 units of E1 in the proposed manual schedules. D1 and D2 can be coupled with each other while E1 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 schedules.

6.1.2 Tested instances

There were four problem instances tested in the computational experiments, as shown in Table 3. "Train#" gives the number of trains in the instance. "Multi-family train#" gives the number of trains that have more than one family of unit available to serve. $N_B^{\pm}$# gives the number of banned location nodes, i.e. $|N_B^-| + |N_B^+|$. Block-arc# and Type-arc# are the number of block-arcs and type-arcs in

Table 3: Instances from ScotRail tested in the numerical experiments

Instance	GB-entire	GB-334R	GA-314	GA-380
Timetable	Dec 2011	May 2014	May 2014	May 2014
Unit types	D1, D2, E1	E1	C1	F1, F2
Location #	21	15	7	16
Train#	483	184	278	427
Multi-family train#	352	–	–	–
$N_B^{\pm}$#	366	160	218	142
Block-arc#	10641	3252	3483	5242
Type-arc#	17779	3253	3483	10457
Demand input	manual	PAX	PAX	PAX

the DAG. For the Demand input, “manual” means the demands were set as the manually scheduled, and “PAX” means they were set as the surveyed actual passenger number counts. For the May 2014 timetable, the PAX data were collected from the December 2013 survey.

GB-entire includes the entire GB services in the December 2011 timetable with all the three permitted types D1, D2 and E1. GB-334R is an instance of rerunning the GB services based on the May 2014 new timetable; it contains the 184 trains that were served by E1 in the May 2014 manual schedules. GA-314 and GA-380 are decomposed instances for the GA area based on families. This decomposition according to unit families was suggested by practitioners from ScotRail.

6.2 Experiment results

Two groups of experiments are reported. The first group is based on GB-entire from the December 2011 timetable with the passenger demands set as the manually scheduled unit capacities. The second group was formed by three experiments based on GB-334R, GA-314 and GA-380 respectively from the May 2014 timetable with the passenger demands set as the actual passenger counts (PAX).

The parameter settings remain unchanged for all the experiments are described here unless otherwise stated. The minimum and maximum connection time durations were set to be 5 min and 720 min respectively. The objective weights on number of units, adjusted arc costs and empty-running costs were set to be 1, 0.001 and 0.1. Only the existing empty-running movements in the manual schedules were allowed. This is mainly required by ScotRail as creating new empty-running trains may give potential problems such as track path conflict with trains from other train operators. It was set that $M_{CG} = 890, \rho_B = 0.01, \Delta z^0 = 0.1, \Delta' = 0.1$. The “jump triggering” parameters were set as $G = 3, R = 10, L = 1$ and $\varepsilon_{\mathrm{BB}} = 0.02$. For CG early termination, only the maximum number of CG iterations (M_{CG}) was used. A maximum total number of 5000 BB tree nodes and 24 hours of running time were set. The branch-and-price was set to terminate as soon as the gap between the BB tree’s global lower bound and global upper bound 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 of ScotRail. Other non-general parameter settings will be specified on individual cases.

6.2.1 Scheduling GB-entire

Two subgroups of experiments were conducted on GB-entire. The first one (GB-entire-A) compares different branching strategies due to the multi-rule system while the second one (GB-entire-B) compares the “customized” setting with other settings in the solver.

For branching variable selection, the chosen object will be selected as described in § 5.2.1. Table 4 gives the results on GB-entire-A, including the manually produced schedule for comparison at the top.

The numbers of units used (unit#), operable objective values (oper-obj), numbers of empty-running movements (ECS#), percentages of the three branching rules (rule% (T/B/A), i.e. train-family, banned location and arc variable respectively), numbers of nodes in the BB tree (BB #) and computational time in seconds (time) are listed. It can be observed that the initial priority setting for the three rules can be crucial. The best strategy is to give arc variable branching the lowest priority (as shown in 1–4, all solved to optimality in relatively short time), and the worst is to give it the highest priority (shown in 9–12, all failed). Letting arc variable branching be ranked in the middle (5–8) is not a good option as two of them failed and the other two spent much longer time than 1–4. Figure 4 shows the relation between CG iteration numbers and the corresponding time consumption per CG iterations at each BB tree node for Instance 1 in GB-entire-A. The other instance that did not fail demonstrated similar patterns. Due to the use of column inheritance, both the iteration number and the time consumption at the leaf nodes are much smaller compared with the root, avoiding the tail-off effect.

Table 4: GB-entire-A: different multi-rule strategies

instance	priority	switch strategy	unit#	oper-obj [a]	ECS#	rule%(T/B/A)	BB#	time
manual	–	–	72	73.921	12	–	–	–
1	T≻B≻A	dynamic	72	73.537	8	42/26/32	198	2558
2	T≻B≻A	static	72	73.537	8	46/25/29	189	2001
3	B≻T≻A	dynamic	72	73.543	8	32/34/35	173	1952
4	B≻T≻A	static	72	73.543	8	36/37/27	163	2019
5	T≻A≻B	dynamic	fail	–	–	44/6/50	2842	86400
6	T≻A≻B	static	fail	–	–	42/4/54	2469	86400
7	B≻A≻T	dynamic	72	73.5365	8	18/18/64	2602	72286
8	B≻A≻T	static	72	73.535	8	5/20/75	1344	47600
9	A≻B≻T	dynamic	fail	–	–	4/18/78	2799	86400
10	A≻B≻T	static	fail	–	–	0/8/92	1929	86400
11	A≻T≻B	dynamic	fail	–	–	30/0/70	2262	86400
12	A≻T≻B	static	fail	–	–	6/1/93	2030	86400

[a] For all runs by the solver, BB tree root objective value = 73.5262, BB tree root lower bound = 73.4386.

Table 5: GB-entire-B: Comparison with other settings

settings	unit#	root-obj	root-LB	oper-obj	ECS#	oper sol#	rule%(T/B/A)	BB#	time
GB-entire-A:1	72	73.5262	73.4386	73.543	8	1	32/34/35	173	1952
random T&B	102	73.5262	73.4386	103.161	9	3	28/61/11	2836	86400
w/o eUB & resv	74	73.5262	73.4386	75.5345	8	6	33/40/26	2096	86400
w/o convhull	failed	72.1782	72.1106	–	–	0	0/100/0	2960	86400

Table 5 gives the results on Experiment GB-entire-B. The top row shows Instance 1 from GB-entire-A as in Table 4 where: (i) Chosen branching objects in train-family branching and banned location branching are selected based on trains' departure times in the DAG, (ii) Estimated upper bounds with node reservation are used in the BB tree, and (iii) The train convex hull constraints (18) are used. Four other instances were tested where the first one ("random T&B") chooses branching objects in train-family and banned location branching in a random way, the second one ("w/o eUB & resv") does not use estimated upper bounds and node reservation and the third one ("w/o convhull") solves the instance without the local convex hull constraints but directly by $\sum_{k\in K_j}\sum_{p\in P_j^k} q_k y_p \geq r_j, \forall j \in N$ and $\sum_{k\in K_j}\sum_{p\in P_j^k} y_p \leq u_j, \forall j \in N$. This was possible as for all the routes and unit types in GB, the maximum number of coupled units is a constant of $u_j = 2$ (see Tables 1 and 2). All other settings were kept the same for the four runs. "Oper sol#" gives the number of operable solutions found in the BB trees. It can be observed that selecting branching objects according to trains' departure times and the use of estimated upper bounds can be important for efficiently finding optimal solutions, as the

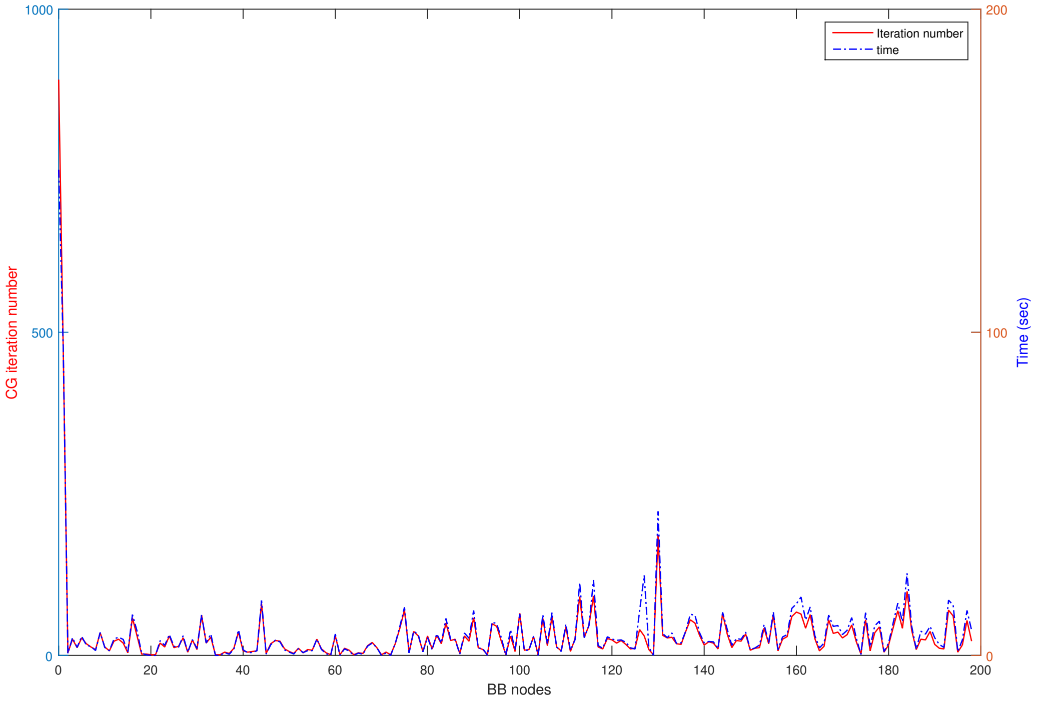

Figure 4: CG iteration numbers against CG time per tree node for experiment GB-entire-A

absence of them can lead to poor performances in the tested instances. Without the train convex hull constraints, the solver failed in finding an operable solution within 24 hours and the branching rule never went beyond banned location branching. The LP-relaxation values at the root were also weak compared with the runs using train convex hull constraints.

6.2.2 Experiments on May 2014 timetable with actual passenger demands

This group of three experiments (GB-334R-1a, GA-314-1a, GA-380-1a) was based on GA and GB in the May 2014 timetable with demands extracted from actual passenger counts in December 2013. It gave the solver an opportunity to provide better solutions than the manual ones. 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. In GB-334R-1a, the minimum and maximum connection time durations were set to 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 discussed in § 5.3.1 was enabled to trigger column generation in experiments on GA.

6.2.2.1 GB-334R-1a

Table 6 gives the results on GB-334R-1a against the manual solution. The solver saves 3 units compared with the manual 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 schedule and the solver with the actual

Table 6: 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	33.2455	33.232	33.232	166	17	1[b]	0	92/8	213	107

[a] There was one train whose December 2013 passenger count survey data was unknown.

[b] There was one train whose PAX#$> 2 \times q_{\mathrm{E1}}$ (unit capacity of E1) thus can never be satisfied. The trains' demand was set as in the manual schedules.

Table 7: 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	14.4041	14.3535	253	7	9[b]	1	5/95	427	5591

[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.

passenger numbers are added as "fit", "OP" and "UP". "Fit" gives the number of trains whose unit capacity provided can satisfy the passenger demand and for a train if the unit number is decreased by one, the new provision will fail to satisfy the demand. "OP" gives the number of trains whose unit capacity are over-provided such that for a train if one decreases the unit number by one unit, the passenger demand can still be satisfied. "UP" gives the number of under-provided trains. From Table 6, it can be seen that although three units are saved by the solver, it still gives better capacity provision than the manual schedule with more fitted trains (111:166), less over-provided trains (68:17) and less under-provided trains (4:1). Moreover, although the manual schedule has 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. The practitioners from ScotRail were satisfied with experiment GB-334R-1a due to its higher qualities in many aspects compared with the manual one.

6.2.2.2 GA-314-1a

Table 7 gives the result on experiment GA-314-1a. The solver uses the same number of units as the manual schedules satisfying 3 additional trains whose demand requirement was not met by the manual solution. The solver also uses one less empty-running movement. The computational time of 7766 seconds is relatively long considering the BB tree of 419 nodes. Note that 94 % of the tree nodes used arc variable branching, which can often make the RMP harder to solve compared with banned location branching since the former accumulatively adds Constraints (40) to the RMP as the branching proceeds while the latter, on the contrary, only deletes columns in the RMP and arcs in the subproblem network. This was rather different from the instances in the GB area.

6.2.2.3 GA-380-1a

Table 8 gives the result on experiment GA-380-1a. 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×F1" is sufficient for a train, would it be appropriate or not to regard "F1+F2" or "2×F2" as over-provided for the train). Therefore only the numbers of under-provided trains are given in Table 8.

Table 8: 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	35.78	35.775	35.775	6[b]	1	1/99	642	14215

[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.

The behavior of the solver in GA-380-1a was very similar as in GA-314-1a. The arc variable branching was dominant throughout the processes, giving increase in computational time. Similar as in GA-314-1a, attempting to satisfy the 14 under-provided trains in the manual solutions would 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.

7 Conclusions

We have proposed a branch-and-price solver for the network-level model of the TUSP. To the best of the author's knowledge, the studies on the TUSP have been very scarce in the literature. There are researches on the related train unit circulation problem (Schrijver, 1993; Alfieri et al., 2006; Peeters and Kroon, 2008) which however have different requirements and problem settings from the TUSP. The train unit assignment problem (Cacchiani et al., 2010b, 2013b) are similar as the network-level TUSP, but some operational requirements in the UK railway industry are not considered and their solution approaches are mainly LP-based heuristics.

A customized branch-and-price solver is presented for solving the network-level model exactly. Its originality includes: (i) A branching system with multiple branching rules; (ii) An adaptive node selection method combining best-first and depth-first. (iii) Column inheritance in the BB tree. (iv) Estimated upper bounds and node reservation, and (v) the use of convex hulls for satisfying combination-specific coupling upper bound requirements.

Experiments were carried out based on real-world instances on the GA and GB areas of the First ScotRail network. The solutions often outperform the manual schedules in many aspects. It is found from the experiments that the initial priority order among the multiple rules can be crucial for efficiently getting high quality solutions. Moreover, the customized settings such as selecting branching objects according to trains' departure times in train-family and banned location branching, the use of estimated upper bounds with node reservation and local convex hulls are also shown to be advantageous in the tested instances. Post-processing using TRACS-RS to resolve the station-level issues like unit blockage and excessive/redundant coupling/decoupling is also shown to be successful for the instances tested. The largest instance solved by the branch-and-price solver (GB-entire) contains 483 trains.

Acknowledgments. This research is supported by an Engineering and Physical Sciences Council (EPSRC) project EP/M007243/1. We would like to also thank First ScotRail and Southern Railway for their kind and helpful collaboration.

Data Statement First ScotRail and Southern Railway have provided relevant data for the research, part of which is commercially sensitive. Where possible, the data that can be made publicly available is deposited in `http://archive.researchdata.leeds.ac.uk/`.

References

Adenso-Díaz, B., González, M. O., and González-Torre, P. On-line timetable re-scheduling in regional train services. *Transportation Research Part B: Methodological*, 33(6):387 – 398, 1999.

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network flows: theory, algorithms and applications*. Prentice Hall, Englewoods Cliffs, USA, 1993.

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

Alvelos, F. *Branch-and-Price and Multicommodity Flows*. PhD thesis, Escola de Engenharia, Universidade do Minho, Portugal, 2005.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.

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

Barrena, E., Canca, D., Coelho, L. C., and Laporte, G. Single-line rail rapid transit timetabling under dynamic passenger demand. *Transportation Research Part B: Methodological*, 70:134–150, 2014.

Burdett, R. and Kozan, E. Techniques for inserting additional trains into existing timetables. *Transportation Research Part B: Methodological*, 43(8–9):821 – 836, 2009.

Cacchiani, V. *Models and Algorithms for Combinatorial Optimization Problems arising in Railway Applications*. PhD thesis, University of Bologna, Italy, 2007.

Cacchiani, V. Models and algorithms for combinatorial optimization problems arising in railway applications. *4OR, A Quarterly Journal of Operations Research*, 7(1):109–112, 2009.

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

Cacchiani, V., Caprara, A., and Toth, P. Solving a real-world train-unit assignment problem. *Mathematical Programming Series B*, 124(1–2):207–231, 2010b.

Cacchiani, V., Caprara, A., and Toth, P. 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, 2012a.

Cacchiani, V., Caprara, A., and Toth, P. A Fast Heuristic Algorithm for the Train Unit Assignment Problem. In Delling, D. and Liberti, L., 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, 2012b. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

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

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

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., and Wagenaar, J. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15 – 37, 2014.

Caprara, A., Kroon, L., Monaci, M., Peeters, M., and Toth, P. Passenger railway optimization. In Barnhart, C. and Laporte, G., editors, *Handbooks in Operations Research and Management Science*, volume 14, chapter 3, pages 129–187. Elsevier, 2007.

Caprara, A., Kroon, L., and Toth, P. Optimization problems in passenger railway systems. In *Wiley Encyclopedia of Operations Research and Management Science*, volume 6, pages 3896–3905. Wiley, 2011.

Caprara, A., Fischetti, M., Guida, P. L., Toth, P., and Vigo, D. *Computer-Aided Transit Scheduling: Proceedings, Cambridge, MA, USA, August 1997*, chapter Solution of Large-Scale Railway Crew Planning Problems: the Italian Experience, pages 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

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

Chen, J. *Confluent Flows*. PhD thesis, College of Computer and Information Science, Northeastern University, 2005.

Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A. *Combinatorial Optimization*. Wiley, New York, 1998.

Cordeau, J.-F., Soumis, F., and Desrosiers, J. A benders decomposition approach for the locomotive and car assignment problem. *Transportation Science*, 34(2):133–149, 2000.

Cordeau, J.-F., Desaulniers, G., Lingaya, N., Soumis, F., and Desrosiers, J. Simultaneous locomotive and car assignment at via rail canada. *Transportation Research Part B: Methodological*, 35(8): 767–787, 2001a.

Cordeau, J.-F., Soumis, F., and Desrosiers, J. Simultaneous assignment of locomotives and cars to passenger trains. *Operations Research*, 49(4):531–548, 2001b.

Dantzig, G. B. and Wolfe, P. Decomposition principle for linear programs. *Operations Research*, 8 (1):101–111, 1960.

Desrosiers, J. and Lübbecke, M. E. A primer in column generation. In Desaulniers, G., Desrosiers, J., and Solomon, M. M., editors, *Column Generation*, pages 1–32. Springer US, New York, USA, 2005.

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

Gallo, G. and Sodini, C. Extreme points and adjacency relationship in the flow polytope. *CALCOLO*, 15(3):277–288, 1978.

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

Huisman, D. A column generation approach for the rail crew re-scheduling problem. *European Journal of Operational Research*, 180(1):163 – 173, 2007.

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

Jiang, Z., Tan, Y., and Yalcinkaya, O. Scheduling additional train unit services on rail transit lines. *Mathematical Problems in Engineering*, vol. 2014, 2014.

Kang, L., Wu, J., Sun, H., Zhu, X., and Wang, B. A practical model for last train rescheduling with train delay in urban railway transit networks. *Omega*, 50:29 – 42, 2015.

Kroon, L. and Fischetti, M. *Computer-Aided Scheduling of Public Transport*, chapter Crew Scheduling for Netherlands Railways "Destination: Customer", pages 181–201. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

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

Kroon, L. G., Maróti, G., and Nielsen, L. Rescheduling of railway rolling stock with dynamic passenger flows. *Transportation Science*, 49(2):165–184, 2015.

Kwan, R. S. K. Case studies of successful train crew scheduling optimisation. *Journal of Scheduling*, 14(5):423–434, 2010.

Lin, Z. and Kwan, R. S. K. Multicommodity flow problems with commodity compatibility relations. Applied Mathematical Programming and Modelling (APMOD 2016), Brno, June 8–10, 2016., 2016a.

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

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

Lin, Z., Barrena, E., and Kwan, R. S. K. Train unit scheduling guided by historic capacity provisions and passenger count surveys. *Public Transport*, pages 1–18, 2016. doi: 10.1007/s12469-016-0138-7.

Lingaya, N., Cordeau, J.-F., Desaulniers, G., Desrosiers, J., and Soumis, F. Operational car assignment at VIA rail canada. *Transportation Research Part B: Methodological*, 36(9):755 – 778, 2002.

Lu, C., Zhou, L., Yue, Y., and Chen, R. A branch and bound algorithm for the exact solution of the problem of EMU circulation scheduling in railway network. *Mathematical Problems in Engineering*, vol. 2016, 2016.

Lübbecke, M. E. and Desrosiers, J. Selected topics in column generation. *Operations Research*, 53: 1007–1023, 2002.

Maróti, G. *Operations Research Models for Railway Rolling Stock Planning*. PhD thesis, Eindhoven University of Technology, the Netherlands, 2006.

Meng, L. and Zhou, X. Robust single-track train dispatching model under a dynamic and stochastic environment: A scenario-based rolling horizon solution approach. *Transportation Research Part B: Methodological*, 45(7):1080 – 1102, 2011.

Mu, S. and Dessouky, M. Scheduling freight trains traveling on complex networks. *Transportation Research Part B: Methodological*, 45(7):1103 – 1123, 2011.

Peeters, M. and Kroon, L. G. Circulation of railway rolling stock: a branch-and-price approach. *Computers & OR*, 35(2):538–556, 2008.

Schrijver, A. Minimum circulation of railway stock. *CWI Quarterly*, 6:205–217, 1993.

Shen, Y., Peng, K., Chen, K., and Li, J. Evolutionary crew scheduling with adaptive chromosomes. *Transportation Research Part B: Methodological*, 56:174 – 185, 2013.

Tornquist, J. and Persson, J. A. N-tracked railway traffic re-scheduling during disturbances. *Transportation Research Part B: Methodological*, 41(3):342 – 362, 2007.

Tracsis plc. TRACS-RS: rolling stock planning software, 2016. URL `http://www.tracsis.com/software/tracs-rs`.

Vohra, R. *Mechanism Design: A Linear Programming Approach*. Cambridge University Press, 2011.

Wolfenden, K. and Wren, A. Locomotive scheduling by computer. In *Proc. British Joint Computer Conference*, volume 1, pages 31–37, London, UK, 1966. IEE Conference Publication No. 19.

Wolsey, L. A. *Integer programming*. Wiley, 1998.

Wren, A. Scheduling vehicles and their drivers—forty years' experience. Technical Report 2004.03, School of Computing, University of Leeds, April 2004.

Zhan, S., Kroon, L. G., Veelenturf, L. P., and Wagenaar, J. C. Real-time high-speed train rescheduling in case of a complete blockage. *Transportation Research Part B: Methodological*, 78:182 – 201, 2015.

Zhou, X. and Zhong, M. Single-track train timetabling with guaranteed optimality: Branch-and-bound algorithms with enhanced lower bounds. *Transportation Research Part B: Methodological*, 41(3):320 – 341, 2007.

Ziarati, K., Soumis, F., Desrosiers, J., and Solomon, M. M. A branch-first, cut-second approach for locomotive assignment. *Management Science*, 45:1156–1168, 1999.

Appendices

A Proof on the equivalence between (PF) and (PF')

Proof. Let $\delta_a^p = 1$ if path p contains arc a and $\delta_a^p = 0$ otherwise. For (PF), a cost for each path p can be defined as $c^{kT}\hat{x}^p = \sum_{a\in\mathscr{A}^k} c_a\delta_a^p = \sum_{a\in\mathscr{A}_p} c_a = c_p$. Then we can write Constraints (13) in their component-wise form as $\sum_{k\in K_j, p\in P^k}(H^k\hat{x}^p)_{fj}y_p \leq h^j_f, \quad \forall f\in F_j, \forall j\in N$. Consider $\sum_{k\in K_j}(H^k x^k)_{fj} \leq h^j_f$, the (f,j)-th entry of Constraint (10), and compare it with (4). We have

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

Thus

$$\begin{aligned}(H^k\hat{x}^p)_{fj} &= \frac{1}{b^k_0}(H^k x^p)_{fj}\\ &= \frac{1}{b^k_0}\sum_{a\in\delta^k_-(j)} H^j_{fk} x^p_a\\ &= \sum_{a\in\delta^k_-(j)} H^j_{fk}\delta^p_a.\end{aligned}$$

Let P^k_j be the set of paths passing via node j in type-graph $\mathscr{G}^k$, and let $\mathscr{A}^k_p$ be the set of arcs in path p in type-graph $\mathscr{G}^k$. Constraints (13) can be rewritten with the original coefficients in (AF) as,

$$\begin{aligned}\sum_{k\in K_j, p\in P^k}(H^k\hat{x}^p)_{fj}y_p &= \sum_{k\in K_j, p\in P^k}\sum_{a\in\delta^k_-(j)} H^j_{fk}\delta^p_a y_p\\ &= \sum_{k\in K_j, p\in P^k}\left(\sum_{a\in\delta^k_-(j)\setminus\mathscr{A}^k_p} H^j_{fk}\cdot 0 + \sum_{a\in\delta^k_-(j)\cap\mathscr{A}^k_p} H^j_{fk}\cdot 1\right)y_p\\ &= \sum_{k\in K_j, p\in P^k_j} H^j_{fk} y_p \leq h^j_f,\end{aligned}$$

which shows the equivalence between (PF) and (PF'). □