
Article:
Lin, Z and Kwan, RSK (2014) A two-phase approach for real-world train unit scheduling. Public Transport, 6 (1-2). 35 - 65. ISSN: 1866-749X

https://doi.org/10.1007/s12469-013-0073-9

Noname manuscript No.
(will be inserted by the editor)

A Two-phase Approach for Real-world Train Unit Scheduling

Zhiyuan Lin · **Raymond S. K. Kwan**

Abstract A two-phase approach for the train unit scheduling problem is proposed. The first phase assigns and sequences train trips to train units temporarily ignoring some station infrastructure details. Real-world scenarios such as compatibility among traction types and banned/restricted locations and time allowances for coupling/decoupling are considered. Its solutions would be near-operable. The second phase focuses on satisfying the remaining station detail requirements, such that the solutions would be fully operable.

The first phase is modeled as an integer fixed-charge multicommodity flow (FCMF) problem. A branch-and-price approach is proposed to solve it. Experiments have shown that it is only capable of handling problem instances within about 500 train trips. The train company collaborating in this research operates over 2400 train trips on a typical weekday. Hence, a heuristics has been designed for compacting the problem instance to a much smaller size before the branch-and-price solver is applied. The process is iterative with evolving compaction based on the results from the previous iteration thereby converging to near-optimal results.

The second phase is modeled as a multidimensional matching problem with a mixed integer linear programming (MILP) formulation. A column-and-dependent-row generation method for it is under development.

Keywords train unit scheduling · railway shunting · fixed-charge multicommodity flow · branch-and-price · hybridization method

JEL Code C60 · C61 · R40

Z. Lin
School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom
Tel.: +44 (0)113 343 5430, Fax: +44 (0)113 343 5468, E-mail: tszl@leeds.ac.uk

R. S. K. Kwan
School of Computing, University of Leeds, Leeds, LS2 9JT, United Kingdom
Tel.: +44 (0)113 343 5430, Fax: +44 (0)113 343 5468, E-mail: r.s.kwan@leeds.ac.uk

1 Introduction

A train multiple unit, or *train unit*, is a set of train carriages with its own built-in engine(s). Without a locomotive, it is able to move in both directions on its own. A train unit can also be coupled with other units of the same or similar types. *Train unit scheduling* refers to the planning of how timetabled train trips (or simply *trains*) in one operational day are sequenced to be operated by each train unit in the company's fleet. Usually performed after the timetable has been fixed, and followed by a subsequent crew scheduling stage, train unit scheduling is also called *train unit diagramming* in the UK, where a *unit diagram* is a documentation of the sequence of trains and other auxiliary activities, e.g. coupling/decoupling and shunting, an individual train unit will serve on a specified day of operation.

Optimization in train unit scheduling is very important because of the high costs associated with leasing, operating and maintaining a fleet. It would also impact upon the subsequent planning of crew resources. For some heavily used commuter networks, e.g. in the North and South of England, train unit scheduling is also important for best distributing limited rolling stock resources, by coupling/decoupling train units and/or running them empty, to meet passenger demands. In real-world practice, there are often thousands of timetabled trains to be scheduled and complex rail infrastructure layouts, especially at large train stations, making train unit scheduling optimization a very hard problem to solve.

In this paper, a two-phase approach is proposed. The first phase is mainly a *train sequencing and fleet assignment problem*, leaving the resolution of how the train units would feasibly flow through the precise station layouts to the second phase. An integer fixed-charge multicommodity flow (FCMF) model similar to Cacchiani et al (2010), but more comprehensive in modeling the real-world conditions and constraints typical of UK rail operations, is used in the first phase. Fleet size and type constraints, linkage time allowance validity and coupling/decoupling possibilities are included in the model. The objective function minimizes a total weighted cost based on fleet size, operational costs, empty movement mileage, and route preferences.

The second phase, as a *railway shunting problem*, finally determines the finer operational plan details. For example, precise track and platform layout constraints are considered. Prior to unit scheduling the timetable is already fixed, and the planning of which would have ensured that the infrastructure capacity at each station would be able to cope with the demand. In Phase I, platform length and coupling upper bound constraints further ensure that unit coupling would not create infeasible additional demands on stations. The shunting (empty movements between platforms, sidings and depots) problems are modeled as a multidimensional matching problem, which eliminates "crossing" shunting movements and minimizes operational costs. A column-and-dependent-row (Muter et al (2012)) approach with pre-generation will be used.

Traditionally human schedulers are compiling train unit schedules based on individual train stations. Because of the input timetable, all train arrivals and departures are predetermined and a human scheduler would try to make feasible links between them at each station. This manual process is analogous to our proposed second phase. However, the consideration of empty running across stations, coupling and decou-

pling possibilities and passenger demands would force the scheduler to switch their thought process between stations on a trial-and-error basis; our proposed first phase takes a holistic network wide approach instead. Our rationale for the proposed two-phase approach is partly due to the observation on the manual process that once the cross-station flows are satisfactorily or nearly sorted, the final logistics at the station level is usually resolvable. This observation has been further reaffirmed by our experiments in post-processing the first phase results using Tracs-RS (Tracsis Plc (2013)), a piece of interactive software which aims at facilitating human schedulers' manual process by visualizing and resolving blockage and shunting plans at the station level. Results of the first phase results have been uploaded into the Tracs-RS system and operable final results can be easily obtained by some straightforward modification in all such experiments. The second phase model basically realizes the same tasks as Tracs-RS, except that it will be a totally automatic process.

The remainder of this paper is organized as follows. Section 2 introduces the train unit scheduling problem and highlights some aspects and features typical of the UK railway industry. Section 3 gives a brief overview on relevant literature. Section 4 and 5 present the two-phase approach models. Section 6 describes the methods for solving the proposed models. Section 7 presents the results from our experiments. Finally Section 8 draws conclusions and remarks on the ongoing work and further research.

2 Problem description

Given a railway operator's timetable on a particular day of the week, and a fleet of train units of different types, the train unit scheduling problem aims at determining an assignment plan such that each train is appropriately covered by a single or coupled units, with certain objectives achieved and certain constraints respected. From the perspective of a train unit, the scheduling process assigns a sequence of trains to it as its daily workload (a unit diagram). Maintenance provision can often be achieved either within the slacks in the diagrams or by swapping physical units assigned to the unit diagrams (Maróti and Kroon (2005)), thus maintenance planning is conventionally ignored at the stage of train unit scheduling.

The main objectives are to minimize the number of units used and/or the operational costs. It is also a common objective to meet the passenger capacity demands.

2.1 Scheduling constraints

The basic hard constraints for the schedule to be operable are:

(i) Each train should be covered by one or more units whose total capacity satisfies the passenger demand expected for the train.
(ii) In between serving a sequence of passenger carrying trains, the gaps must be time feasible for the unit. When auxiliary activities, e.g. empty running and unit coupling/decoupling, have to be inserted in such gaps, sufficient time must be allowed for.

(iii) Coupling/decoupling activities may be banned or restricted at some locations.
(iv) The sequence of trains assigned to a unit must be compatible in terms of the unit traction types and the routes traversed.
(v) The unit diagrams must not be in temporal or spatial conflict with each other, causing blockage on the tracks/platforms within a station.

Some soft constraints are:

(i) Some auxiliary operations (empty-running, coupling/decoupling, shunting) would be minimized.
(ii) It is desirable to achieve some long gaps of appropriate time lengths between trains during certain times. Such long gaps in the unit diagrams would ease the subsequent task of maintenance planning.
(iii) Beyond mandatory compatibility between the trains sequenced together, there may be preferences for specific unit types and routes for individual trains

2.1.1 Train unit types

There could be many types of train units in a railway operator's fleet. Different unit types will have different features like number of seats, number of cars, unit length, permitted routes to run, permitted home-depots, etc. We use *type-route compatibility* for the relation that certain types can only run on certain routes in the rail network. Notice there can be *preferences* for permitted types used for a route, as well as the choice of home-depots.

2.1.2 Train unit coupling and decoupling

A challenging issue of train unit scheduling is that more than one unit can be attached to serve the same train, known as *coupling*; the reversal activity is called *decoupling*. Units of the same type can be coupled, and a certain number of different types may also couple with each other. We call this relation as *type-type compatibility*. The maximum number of coupled units or cars for a train may depend on many factors, such as routes, platform lengths and unit types.

With respect to type-type compatibility relation, we introduce the concept of *train unit family*. Simply speaking, all unit types that are coupling compatible belong to the same family. The permitted coupling combinations can be expressed in terms of upper bounds on the total number of units and total number of cars. When the permitted combinations have to be further restricted, e.g. for certain routes, sub-families using the same unit types with different upper bounds are introduced. This is illustrated in Table 1 for a problem instance of Southern Railway.

For any unit (sub-)families used by a train, both upper bounds have to be satisfied. For example, using family VI, the combination of one "377/1,2,4 [4 car]" and three "377/3 [3 car]" would be within the upper bound of 4 units but exceeds the upper bound of 12 cars and is therefore ruled out.

Coupling/decoupling activities are often essential for satisfying passenger demands. However, there are restrictions on where coupling/decoupling activities are permitted to take place (usually at big stations or hubs). The time allowances required

(typically 2 to 5 minutes) for accomplishing such activities may also hinder the overall schedule efficiency to some extent, and add to the complexity of the scheduling problem. Moreover, after having provided the required passenger carrying capacity, some coupled units would inevitably have been displaced to parts of the network that become in excess in terms of capacity provision, and they would have to be redistributed to other locations by coupling, serving trains that do not need the extra capacity, or by empty running.

Coupling/decoupling sometimes is not restricted to take place at the beginning/end of a train trip. That is, train units may be joined or split en route. These scenarios are often related to the topological structure of the railway network, e.g. hubs and junctions. This sort of en route coupling/decoupling operation is very rare in the UK, and might cause confusion and inconvenience to passengers. Maróti (2006) refers to these special cases of coupling/decoupling as "joining/splitting". We do not consider the operation of coupling/decoupling en route in our work.

For a set of coupled units (which we call a *unit block*), both its *composition* and *permutation* are important. The reason is that sometimes coupling/decoupling can only be performed in certain order, and corrective repositioning may not be possible within time constraints.

Although appropriate coupling/decoupling activities may contribute to an optimal schedule, they would also consume resources like tracks, shunting operations, crews and time. It is therefore important to avoid *unnecessary* coupling/decoupling activities. Sometimes, it may be preferable to keep a unit block together for longer rather than reversing the coupling/decoupling actions between the units in the block several times.

2.1.3 Empty-running

An advantage of relocating a train unit by means of coupling onto another unit serving a timetabled train trip is that there would not be conflicts in terms of track usage, and therefore eases the scheduling process. On the other hand, running a unit empty, i.e. without passengers, may be more flexible in terms of its timing. The disadvantages are that the path and timing of an empty train has to be checked and approved such that no conflict would occur with other track users; and that the empty running is not directly contributing to passenger carrying capacity. Also, empty-runnings have significant additional operational costs.

Table 1 Unit families of the fleet of Southern Railway, UK

Families	Types	Upper bound on unit #	Upper bound on car #
I.a	171/7 [2 car], 171/8 [4 car]	2	8
I.b	171/7 [2 car], 171/8 [4 car]	1	4
II	455/8 [4 car], 456/0 [2 car]	3	8
III	313/1 [3 car]	1	3
IV	460/0 [8 car]	1	8
V	442/1 [5 car]	2	10
VI	377/1,2,4 [4 car], 377/3 [3 car]	4	12

Table 2 Example trains at the same platform

Train No.	Origin – Destination	Departure time	Arrival time
1	B – A	*	09:50
2	C – A	*	09:55
3	A – C	10:10	*
4	A – C	10:20	*

Empty-running may be planned on non-passenger routes. This may either short-cut the journey, or to allow the unit to reach locations that passengers normally would not reach, e.g. depots, shunting tracks, cleaning and refueling locations.

In an automatic scheduling process, empty-running is usually planned using a predefined collection of empty-running journey time allowances between location pairs. There may also be time zones when empty-running is prohibited. At the end of the scheduling process, empty-running trains are planned in detail and ensured to be conflict free. At that stage, it is possible that some adjustments to the schedule are required because some empty-runnings may turn out to be infeasible.

2.1.4 Shunting movements and station infrastructure

The linkage between an arriving train and a departing train sometimes may only be accomplished by certain shunting movements. Railway *shunting* generally refers to empty-running movements between *platforms*, *sidings* and *depots* during a linkage. Shunting plans enable units to be delivered to the right location at the right time without causing blockage to each other. The length of empty running required varies between different types of shunting movement. Whereas the arrival and departure platforms for trains are usually assigned at the timetable planning stage, the assignment of which sidings or depots the units will be shunted to are only planned at the unit diagramming stage. Since depot shunting usually occurs before or after an idle period for the unit, it is generally less constraining on unit scheduling.

A distinctive feature that makes rolling stock scheduling different from road (e.g. bus) vehicle scheduling is that movements of rolling stocks are strictly restricted by tracks and other railway infrastructures like platforms and sidings, giving additional problems such as unit blockage or hitting platform or siding capacity restrictions. In some literature, blockage is also known as *crossing* (Freling et al (2005), Kroon et al (2008)). Other issues may also be relevant to shunting, for example, compatibility between traction type and platform/siding/depot.

To illustrate how crossing can invalidate a unit diagram derived solely based on timetable and fleet information without station infrastructure detail, consider a simple example. Table 2 gives a timetable of four trains. Suppose at station A all the four trains have been assigned to the same long platform, where the tracks are bidirectional. There are two train units available each with sufficient capacity for any of the trains. Suppose Unit I can only serve Trains 1, 3 and 4, while Unit II can serve Trains 2, 3 and 4.

Applying the FIFO rule matching arrivals to departures at the platform would give the following solution:

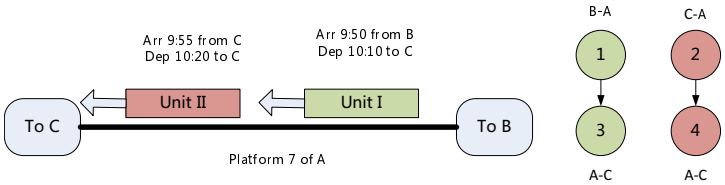

Fig. 1 Station blockage with the FIFO solution (i)

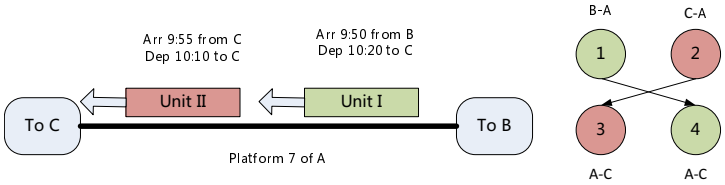

Fig. 2 A feasible solution (ii)

(i) Unit I: Train 1 $\longrightarrow$ Train 3, Unit II: Train 2 $\longrightarrow$ Train 4;

But solution (i) is infeasible because when departing at 10:10, Unit I will be obstructed by Unit II whose departure time is 10:20. It would be easy to see that an alternative feasible solutions exists:

(ii) Unit I: Train 1 $\longrightarrow$ Train 4, Unit II: Train 2 $\longrightarrow$ Train 3.

The two solutions are illustrated in Figure 1 and 2, showing how train directions and station infrastructure can adversely affect the results if the scheduling process does not take them into consideration.

Within the train unit scheduling process, train sequencing and fleet assignment based on timetable, fleet and route knowledge can be regarded as network wide high-level planning. On the other hand, shunting movements (including null shunting) based on detailed infrastructure and train direction knowledge belong to a lower level planning involving platforms, sidings and depots. High-level planning alone may result in problems such as crossing, which would hinder the solutions from being operable; but then, incorporating every detailed infrastructure information into the scheduling method will lead to an extremely huge-sized optimization problem intractable to solve. Therefore, a two-phase approach is preferred as a compromise, which is to be discussed in Section 4 and 5.

3 Literature review

3.1 Train sequencing and fleet assignment problem

Schrijver (1993) proposes a multicommodity circulation flow integer linear programming (ILP) model concerning the rolling stock scheduling problem for a single line and a single day for NS Reizigers with about 100 train trips; the objective is to minimize the total number of train units used. As only two types of unit are used for

coupling, the weak knapsack constraints for passenger demand satisfaction are dealt with by explicitly finding relevant convex hulls in 2-dimensional spaces.

Alfieri et al (2006) consider a similar scenario as Schrijver (1993) for NS Reizigers; the objective is to minimize the total number of train units used plus carriage-kilometers. Additional considerations are given in unit permutation and shunting time buffering. Some detailed infrastructure-related problems such as shunt crossing are ignored at this planning stage. The transition graph concept is introduced.

Fioole et al (2006) describe a similar model and instance as Alfieri et al (2006) with additional considerations on train splitting and joining. Peeters and Kroon (2008) describe a similar model and instance as Alfieri et al (2006) with a specialized solver employing branch-and-price.

Maróti (2006) proposes generalized models of the above works in the Netherlands in his PhD thesis.

Cacchiani et al (2010) propose an integer multicommodity flow ILP model for some local train unit scheduling problems. The LP-relaxation is solved by column generation and a diving heuristics is used for finding integer solutions and speeding-up the process. Taking advantage of the local instances where no more than two units are coupled (from many potential types), the weak knapsack constraints per train for demand satisfaction are overcome by computing a dominant set of a polytope defined over another space. This model is applied for real-world instances of up to 600 trains. Location restrictions and time allowances for coupling/decoupling, type-type compatibility and shunting resolutions are not considered.

3.2 Railway shunting problem

Freling et al (2005) propose a two-phase approach to solve the railway shunting problem at a station within a 24-hour horizon. The method is based on a set-partitioning ILP model with each column corresponding to a feasible assignment plan for a siding. Both single (FILO) and double (free) sidings are considered. It prevents overcapacity and crossing at each siding, and also minimizes unnecessary coupling/decoupling activities. The instances tested have up to about 80 train units to be parked at 19 sidings. This two-phase approach gives near-optimal solutions, with small gaps to global optimality.

Kroon et al (2008) later propose another approach for the same railway shunting problem as in Freling et al (2005), where an integrated model (and three of its variants) is devised such that matching and assigning is processed together, hence global optimality is guaranteed. To avoid a possible huge number of crossing constraints, maximal cliques and comparability graph techniques are used.

3.3 Generate-and-select and PowerSolver

Dynamic column generation needs a fast non-enumerative pricing algorithm. However for real-world applications, the pricing subproblems are often computationally expensive for even moderately large problem instances. Kwan (2004) presents a

Generate-and-Select (GaS) approach in crew scheduling, in which the column generation process only brings in new columns within a pre-generated very large collection of columns. Apart from the advantage of reducing computation associated with solving pricing subproblems, a reasonably sized selection of the pre-generated columns could also be made for the branch-and-bound process. Kwan and Kwan (2007) further propose a hybrid method called *PowerSolver* to solve very-large-scale train crew scheduling problems. PowerSolver uses an iterative heuristics to control the size of the problem instance in each execution of its core algorithm (GaS with set-covering ILP) such that the ILP solver works comfortably due to the reduced problem size. Initially, the controlled problem instance is a crude simplification of the original problem instance; it is then refined and converges to containing the crucial features in a compact form over a number of iterations. A mechanism to ensure that new solutions would not be worse than the current best is embedded, thus the results would converge to a near-optimal one within a relatively short time. PowerSolver is now part of the TrainTRACS system (Fores et al (2002); Wren et al (2003)) and is proving successful by many UK transport operators who are now routinely using it. A similar approach to PowerSolver could be applied in train unit scheduling and will be discussed in later sections.

4 A fixed-charge multicommodity flow model for the first phase

Considering that the unit scheduling problem would be huge and intractable if all detailed infrastructure constraints are to be considered at once, a two-phase approach is proposed.

The first phase for train sequencing and fleet assignment uses an integer FCMF model similar to Cacchiani et al (2010). But the model is more comprehensive in modeling real-world conditions and constraints typical of UK railway operations, especially the coupling/decoupling location restriction and time allowances, and type-type compatibility enforcement. The aim is to produce solutions that are near to being fully operable.

The second phase for station shunting finally determines the finer operational plan details in a station-by-station manner. For example, precise track and platform layout constraints are considered for determining the best operable shunting movement plans. It will be described in Section 5.

4.1 Model description

The integer FCMF model is based on a framework of *directed acyclic graph* (DAG) representing trains and their relations. A generic DAG $G = (N,A)$ consists of nodes and directed arcs such that no cycle exists.

In this framework, the majority of nodes represent the trains in the timetable, thus are called *train nodes*, denoted by $\widetilde{N}$. In addition, a *source node* 0 and a *sink node* ∞ are added as usual. Hence $N = \widetilde{N} \cup \{0,\infty\}$. We also use $N_B^- \subseteq \widetilde{N}$ to denote the set of all trains whose departure locations are banned for coupling/decoupling, and

$N_B^+ \subseteq \widetilde{N}$ for arrival locations. There are three kinds of arcs in the DAG, namely *sign-on arcs*, *sign-off arcs* and *linkage arcs*. A sign-on arc starts from the source node and ends at a train node; a sign-off arc starts from a train node and ends at the sink node. Generally all train nodes have a sign-on arc and a sign-off arc. We denote the set of all sign-on/off arcs by A°. A linkage arc $a \in \widetilde{A}$ links two train nodes i and j ($a = (i, j)$), representing a potential link, with a sufficient time gap, such that after serving train i a unit can continue to serve train j as its next task. Note that $A = A^\circ \cup \widetilde{A}$. We use $\delta_-(j)$ to denote all arcs that terminate at node j, and $\delta_+(j)$ for all arcs that originate from node j; and use E to represent the set of all arcs implying empty-running. A *path* $p \in P$ in the DAG is defined as a sequence of nodes starting from the source to the sink such that from each of its nodes there is an arc to the next node in the sequence. A path represents a daily workload (a diagram) for a unit. In addition, P_j and P_a denote the sets of paths passing through node j and arc a respectively; J_p and A_p represent the sets of nodes and arcs in path p respectively.

For the fleet, we denote T the set of all unit types. In order to deal with type-route compatibility, type-graphs G^t (also DAGs) representing routes each unit type $t \in T$ is able to serve are constructed based on the above complete DAG G. All entities in the type-graphs G^t can be symbolized by adding a t superscript in the corresponding notations of the complete DAG G, e.g. P^t refers to the set of all paths in type-graph G^t. We use F to represent the set of all unit families, and F_j the set of families that can serve train j.

There are three kinds of decision variables. *Path variable* $x_p \in \mathbb{Z}_+, \forall p \in P^t, \forall t \in T$ is used to indicate the number of units used in path p from G^t. For each train and the families serving it, we set *train-family variable* $y_j^f \in \{0,1\}, \forall j \in \widetilde{N}, \forall f \in F_j$, to indicate whether a train j is served by any units from family f. *Blockflow variable* $z_a \in \{0,1\}$ is used to indicate whether an arc $a = (i, j) \in A$ is used. It corresponds to a block of units that remains coupled together during two consecutive trains i and j. z_a is used to forbid coupling/decoupling at banned locations, calculate time consumed by coupling/decoupling operations, as well as eliminate unnecessary coupling/decoupling activities.

Finally, Table 3 describes the parameters used in the model.

4.2 Path formulation

If column generation is to be used for a multicommodity flow problem, generally a path formulation would be preferred than an arc formulation as the master problem, although the bound given by LP-relaxations of both are the same (Geoffrion (1974); Barnhart et al (1998); Cook et al (1998); Cacchiani et al (2010)). We choose an LP-relaxation rather than a Lagrangian relaxation as nowadays a state-of-art simplex solver can be more efficient than a subgradient method.

The path formulation:

Table 3 Parameters in the model

Symbols	Meaning
$W_i,\ i=1,\cdots,7$	weights of terms in the objective
μ_j^t	mileage of train j used by type t (not including empty-running trains)
γ_a^t	weight for the preference showing whether arc a is a long gap for type t
π_j^t	weight for the preference of type-train pair (t,j)
π_a^t	weight for the preference of type-sign-in/off arc pair $(t,a), a \in A^\circ$
b^t	upper bound of fleet size of unit type t
κ^t	unit capacity of type t
r_j	passengers' demand for train j
u^f	coupling upper bound in number of units for family $f \in F_j$
v^f	coupling upper bound in number of cars for family $f \in F_j$
n^t	number of cars of unit type t
m_a	the maximum number of units coupled in arc a
l^t	length of unit type t
L_j	the minimum length of platforms at all the stations passed by train j
$\tau^C_{\mathrm{dep}(j)}$	single coupling time at the departure platform of train j
$\tau^D_{\mathrm{arr}(j)}$	single decoupling time at the arrival platform of train j
e_{ij}	slack time in linkage arc $(i,j) \in \widetilde{A}$

$$\begin{aligned}\min_{x,z}\quad & W_1\sum_{t\in T}\sum_{p\in P^t}x_p + W_2\sum_{t\in T}\sum_{j\in\widetilde{N}^t}\sum_{p\in P_j^t}\mu_j^t x_p + W_3\sum_{t\in T}\sum_{a\in E^t}\sum_{p\in P_a^t}x_p \\ & - W_4\sum_{t\in T}\sum_{a\in\widetilde{A}^t}\sum_{p\in P_a^t}\gamma_a^t x_p - W_5\sum_{t\in T}\sum_{j\in\widetilde{N}^t}\sum_{p\in P_j^t}\pi_j^t x_p \\ & - W_6\sum_{t\in T}\sum_{a\in(A^\circ)^t}\sum_{p\in P_a^t}\pi_a^t x_p + W_7\sum_{a\in A}z_a\end{aligned} \tag{1}$$

subject to

$$\sum_{p\in P^t}x_p \le b^t, \quad \forall t\in T; \tag{2}$$

$$\sum_{t\in T}\sum_{p\in P_j^t}\kappa^t x_p \ge r_j, \quad \forall j\in\widetilde{N}; \tag{3}$$

$$\sum_{t\in f}\sum_{p\in P_j^t}x_p \le u^f y_j^f, \quad \forall j\in\widetilde{N},\forall f\in F_j; \tag{4}$$

$$\sum_{t\in f}\sum_{p\in P_j^t}n^t x_p \le v^f y_j^f, \quad \forall j\in\widetilde{N},\forall f\in F_j; \tag{5}$$

$$\sum_{f\in F_j}y_j^f = 1, \quad \forall j\in\widetilde{N}; \tag{6}$$

$$\sum_{t\in T}\sum_{p\in P_j^t}l^t x_p \le L_j, \quad \forall j\in\widetilde{N}; \tag{7}$$

$$\sum_{t\in T}\sum_{p\in P_a^t} x_p \le m_a z_a, \quad \forall a \in A; \tag{8}$$

$$\tau^D_{\mathrm{arr}(i)}\left(\sum_{a\in\delta_+(i)} z_a - 1\right) + \tau^C_{\mathrm{dep}(j)}\left(\sum_{a\in\delta_-(j)} z_a - 1\right) \le e_{ij}, \quad \forall (i,j)\in\widetilde{A} : i \notin N_B^+, j \notin N_B^-; \tag{9}$$

$$\sum_{a\in\delta_-(j)} z_a = 1, \quad \forall j \in N_B^-, \tag{10a}$$

$$\sum_{a\in\delta_+(j)} z_a = 1, \quad \forall j \in N_B^+; \tag{10b}$$

$$x_p \in \mathbb{Z}_+, \quad \forall p \in P^t, \forall t \in T, \tag{11a}$$

$$y_j^f \in \{0,1\}, \quad \forall j \in \widetilde{N}, \forall f \in F_j, \tag{11b}$$

$$z_a \in \{0,1\}, \quad \forall a \in A. \tag{11c}$$

4.2.1 Objective function

The objective (1) is formed by seven terms. The first term minimizes the total number of units used. The second term minimizes the total passenger train unit-mileage, where μ_j^t is the mileage of train $j \in \widetilde{N}$ used by single unit of type t. The third term minimizes the total number of empty-running trains. The fourth term encourages insertion of long gaps in unit diagrams such that maintenance can be realized whereby. Parameter $\gamma_a^t = 0$ if arc a does not imply a long gap or this gap is not suitable for type t; $\gamma_a^t > 0$ if arc a contains a long gap that is suitable for type t. There are also preferences among positive values of γ_a^t as some long gaps are more suitable for type t than others. The fifth term encourages type-train (route) preferences, where $\pi_j^t \ge 0$ is a preference weight for a type-train pair (t, j): the higher the more suitable. The sixth term encourages sign-on/off arc preferences (usually related to home depots) for each type of units, where π_a^t stands for a preference parameter for a type/sign arc pair (t,a): the higher the more suitable. Notice if a certain type t is not suitable for a sign-on/off arc, this arc will not have been inserted into G^t in the first place. The seventh term minimizes the total number of blocks, leading to minimization of the total number of coupling/decoupling activities; it also partly drives z_a to desired binary values. Notice the appearance of z_a in the objective categorizes this model into a fixed-charge multicommodity flow type.

Although the objective is combined with seven terms, in practice, it is possible to only select a subset of them according to the user's wish.

4.2.2 Constraints

Constraints (2) ensure deployed number of units for each type within its upper bound. Constraints (3) guarantee satisfaction of capacity demand for each passenger train. Constraints (4) and (5) calculate each family-train indicator variable as well as restricting the possible coupling combinations in terms of both the total number of units and the total number of cars. Constraints (6) allow only one family to serve a train. Notice (4), (5) and (6) together will ensure that $y_j^f = \begin{cases} 1, \sum_{t\in f}\sum_{p\in P_j^t} x_p > 0 \\ 0, \sum_{t\in f}\sum_{p\in P_j^t} x_p = 0 \end{cases}, \forall j \in \widetilde{N}, \forall f \in F_j$. Constraints (7) are platform length constraints. Constraints (8) calculate blockflow variables. Notice as the sum of blockflow variables is minimized in the objective, the relation $z_a = \begin{cases} 1, \sum_{t\in T}\sum_{p\in P_a^t} x_p > 0 \\ 0, \sum_{t\in T}\sum_{p\in P_a^t} x_p = 0 \end{cases}, \forall a \in A$ is guaranteed. Constraints (9) are to ensure time allowance validity for coupling/decoupling if it is allowed. Constraints (10) are to forbid coupling/decoupling at banned locations. Finally (11) gives the variable domain.

Notice some intuitive simplification will apply, e.g., trains whose platform lengths are long enough do not need constraints (7), and the linkage arcs whose slack time is long enough and relevant locations allow coupling/decoupling, constraints (9) can also be omitted. For those trains only allowing compatible types to serve, train-family variables y_j^f and constraints (6) associated with them can be omitted, and relevant coupling upper bound constraints (4) (5) can also be simplified. And so on.

5 A multidimensional matching model for the second phase

5.1 Station element representation and model description

5.1.1 Arrival and departure units

The first phase provides a tentative schedule based on which the second phase operates within the scope of each individual station $\sigma \in \Sigma$. The first phase fixes the unit-type assignment to each arrival and departure train, e.g., "train 1E08 is served by one British Rail Class 171/7 and one British Rail Class 171/8 units". We can thus define a set of *arrival units* and a set of *departure units* for a day's operation at each station σ, as $u_1, u_2, \cdots, u_n \in U^\sigma$ and $v_1, v_2, \cdots, v_n \in V^\sigma$ respectively. The first phase also results in a tentative assignment of a next departure unit for every arrival unit, and these assignments may have to be modified by the second phase because of operational conflicts at the station. Here we assume empty running trains have been inserted by the first phase and $|U^\sigma| = |V^\sigma| = n$. Sign-on/off trains from/to a home depot can be dealt with by slight adaptation. Exceptions, such as over-night units remaining at the station cannot invalidate this model as sign-on/off arcs regard this station as the source/sink node. For simplicity, we will omit the superscript σ in later part of this section as everything is clearly based on a single station, except where explicitly stating the station is necessary.

The attributes of each arrival and departure unit include:

- The train the unit belongs to: $T(u), T(v)$. Notice an arrival unit only belongs to one arrival train and a departure unit only belongs to one departure train.
- The attributes due to its train: the arrival platform of an arrival unit $P(u)$ and the departure platform of a departure unit $P(v)$; the arrival time of an arrival unit $\tau(u)$ and the departure time of a departure unit $\tau(v)$.
- The type of that unit: $t(u), t(v)$.

A *matching* between an arrival unit and a departure unit is used to represent a linkage relation. A matching between two *conceptual* arrival and departure units u and v indicates they will be *materialized* by being assigned to be the same unit. The first-step rule of how an arrival unit u can be matched with a departure unit v is: (a) Same type: $t(u) = t(v)$; (b) the arrival time of u is appropriately earlier than the departure time of v: $\tau(u) \prec \tau(v)$.

Other linkage validity rules regarding shunting/coupling/decoupling times will be considered respectively in other ways. It is important to note that a *valid* matching between a pair of arrival and departure units $(u, v),\ u \in U, v \in V$ will correspond to one and only one linkage arc $(T(u), T(v)) \in \widetilde{A}$ in the DAG defined in the first phase. This linkage arc is called a *dominant arc* of the matching pair.

5.1.2 Unit position

If a train is a coupled one, the *positions* of the units in the coupled formation are important information. Since the first phase has not provided anything on unit permutation for coupled trains, it has to be decided in the second phase. Let $p \in P_u, q \in Q_v$ denote positions and their possible choice sets for arrival unit u and departure unit v respectively. For a unit u (or v) from a non-coupled train, its position is unique($|P_u| = 1$ or $|Q_v| = 1$); for a coupled single-type (homogenous) train, unit positions can be pre-assigned based on the first phase result in an arbitrary order, leading to $|P_u| = 1$ or $|Q_v| = 1$ as well; for a coupled multi-type (heterogenous) train where different permutations can have essentially different results, a decision has to be made from $p \in P_u, q \in Q_v, |P_u| > 1, |Q_v| > 1$. Therefore, the above matching has to be further modified within a *positioned* pair of (u, p) and (v, q) such that a plan $(u, p, v, q), u \in U, p \in P_u, v \in V, q \in Q_v$ has to be made.

Unit permutation can significantly affect a schedule without infrastructure details. Figure 3 shows two arrival units u_1, u_2 coupled in a train arriving at a dead-end platform, also two departure units v_3, v_4 coupled in a later departure train at the same platform suitable to be linked with the arrival train. Suppose all the units have the same traction type and their positions have been pre-assigned as indicated in the figure. Without unit position and platform layout knowledge, it seems that any arrival units $u_i, i = 1, 2$ can be matched with any departure units $v_j, j = 3, 4$. But in fact u_1 can only be matched with v_3 and u_2 with v_4. Figure 4 gives another example concerning arrival units u_1, u_2 and departure units v_3, v_4, suppose they all have the same traction type and are at the same FIFO platform. Suppose their positions have been pre-assigned as indicated in the figure. Moreover, $T(u_1) \neq T(u_2)$ but $T(v_3) = T(v_4)$ and $\tau(u_2) \prec \tau(u_1) \prec \tau(v_3) = \tau(v_4)$ such that the two arrival units can be coupled to serve the departure train. Here as the arrival train $T(u_2)$ comes earlier than $T(u_1)$ in a

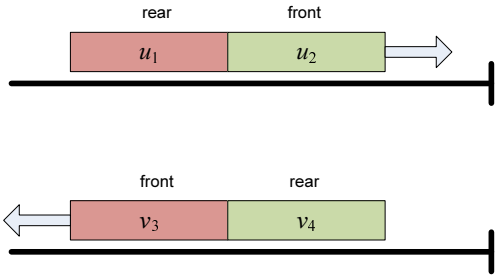

Fig. 3 A matching in a dead-end platform scenario

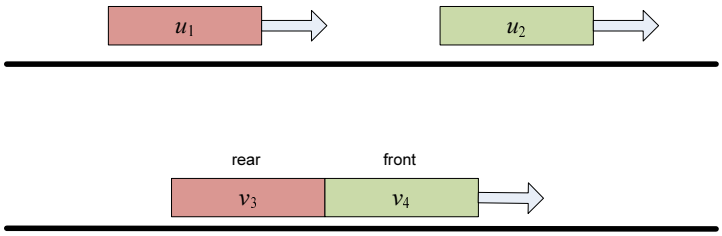

Fig. 4 A matching with coupling in a FIFO scenario

FIFO track, then u_1 can only be coupled at the rear of u_2, which indicates u_1 can only be matched with v_3 and u_2 with v_4. Cases in coupled multi-type trains will be more complicated.

5.1.3 Parking berths

Generally speaking, there are three possible kinds of shunting plans at each linkage arc: to let the unit continue to serve the next train within the platform area (short time gap linkage), to put the unit to a siding temporarily until it is needed for the next train (medium time gap linkage) and to put the unit to a nearby depot until it is needed for the next train (long time gap linkage). Here we denote *parking berths* as $b \in B := \{0\} \cup S \cup D$ where 0 refers to a dummy parking berth representing platform stay, S the siding set and D the depot set.

For a positioned pair of arrival and departure units $(u, p, v, q), u \in U, p \in P_u, v \in V, q \in Q_v$, the possible choice for parking berth types (dummy, siding or depot) is dependent on the relevant time gap length of its dominant linkage arc $(T(u), T(v))$. A dummy berth is only imaginary, i.e. the unit stays put. However, if the parking berth turns out to be a siding or a depot, usually there are a number of sidings or depots available for a unit to be shunted to. We denote the possible parking berths for a pair of positioned units (u, p, v, q) as $B(u, p, v, q)$.

5.1.4 Parking methods

In order to avoid crossings, the way a unit comes to/leaves a platform or a siding should be taken into account. It is usually not important how a unit comes to/leaves

a depot after an empty-running trip. This is because such movements are usually infrequent with sufficient time for resolving any conflicts that might arise.

Here we define the parking method $m \in M(u,p,v,q,b)$ for a pair of positioned units (u,p,v,q) with its parking berth $b \in B(u,p,v,q)$ as possible ways of how this unit arrives at and leaves its arrival platform $P(u)$, comes to and leaves its parking berth b, and then comes to and departs from the departure platform $P(v)$. For a pair yielding a dummy or a depot parking berth $b \in \{0\} \cup D$, the ways coming to and leaving b is omitted as a default "00".

Here is an example of such a parking method with $b = 0$ and a null-shunting: $m = [LR, 00, LR]$, which means arriving from the left and departing from the right, at the same platform without any shunting. Another example of a parking method involving a double (free) siding shunting with re-platforming: $m = [LR, RL, LL]$ which means:

- at the arrival platform: arriving from the left and leaving from the right (to a siding);
- at the siding: coming from the right and leaving from the left (to the departure platform);
- at the departure platform: coming from the left and departing from the left.

Parking methods are determined by various factors, e.g. train directions, platform, unit position, siding, depot and overall layout relations, plus possible engineering and operational regulations. Usually when (u,p,v,q,b) is fixed, the resulting $M(u,p,v,q,b)$ would have a very small number of candidate methods. Especially when the parking berth is a dummy one, a single (FILO) siding or a depot, the possible parking method is very likely to be unique.

5.1.5 Shunting plans and conflicts between a pair of shunting plans

The second phase shunting problem is defined as

At each station $\sigma \in \Sigma$ of the rail network, give each arrival unit $u \in U$ a position $p \in P_u$ and assign it to a departure unit $v \in V$ with position $q \in Q_v$, via a parking berth $b \in B(u,p,v,q)$ and with a parking method $m \in M(u,p,v,q,b)$ such that

1. *Each matching pair (u,v) implies a dominant linkage arc relation $(T(u),T(v)) \in \widetilde{A}$;*
2. *Each positioned pair (u,p,v,q) is operationally feasible for connecting $T(u)$ and $T(v)$;*
3. *No crossing occur at platforms or sidings;*
4. *No overcapacity occur at any sidings or depots;*
5. *No breaking on any other temporal, spatial, engineering or operational rules that makes the overall plan inoperable.*

A *shunting plan* can be expressed as a 6-tuple (u,p,v,q,b,m), $u \in U$, $v \in V$, $(T(u),T(v)) \in \widetilde{A}$, $p \in P_u$, $q \in Q_v$, $b \in B(u,p,v,q)$, $m \in M(u,p,v,q,b)$ representing a *self-consistent* shunting plan. For convenience, a shunting plan (u,p,v,q,b,m) is also written as $\pi \in \Pi$ when its details can be omitted, where we denote the set of all possible shunting plans at a station as Π. The set of shunting plans all involving a

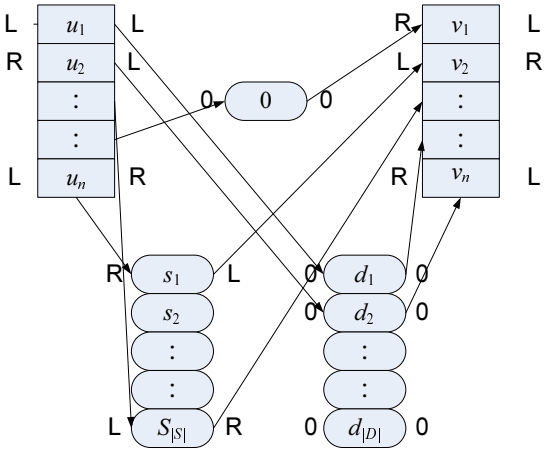

Fig. 5 An illustration of shunting plans

specific arrival unit u can be written as Π_u, the same for Π_v for a specific departure unit v. Fig.5 gives an illustration of the concept of shunting plans (unit positioning is not depicted).

All the possible shunting plans can be pre-computed based on input data of station information and the first phase results before the second phase begins. With a shunting plan (u,p,v,q,b,m) timings can be determined for the unit leaving the arrival platform, coming to the parking berth, leaving the parking berth and coming to the departure platform. Also, the following parameters are known from station layout knowledge. $\Delta\tau_u$: time duration at the arrival platform $P(u)$; $\Delta\tau_{ub}$: shunting/empty-running time from platform $P(u)$ to parking berth b; $\Delta\tau_{bv}$: shunting/empty-running time from parking berth b to platform $P(v)$; $\Delta\tau_v$: time duration at the departure platform $P(v)$. Then, the attributes of a shunting plan (u,p,v,q,b,m) include:

- All attributes inherited from its arrival and departure units, including traction types, trains, positions and platforms;
- At the arrival platform $P(u)$: arrival time $\tau(u)$, leaving time $\lambda(u) := \tau(u) + \Delta\tau_u$, which forms a *time window* at the arrival platform: $W_u(u,p,v,q,b,m) := [\tau(u),\lambda(u)]$;
- At the departure platform $P(v)$: coming time: $\chi(v) := \tau(v) - \Delta\tau_v$, departure time $\tau(v)$, which forms a time window at the departure platform: $W_v(u,p,v,q,b,m) := [\chi(v),\tau(v)]$;
- At parking berth b: coming time $\chi(b) := \lambda(u) + \Delta\tau_{ub}$, leaving time $\lambda(b) := \chi(v) - \Delta\tau_{bv}$, which forms a time window at the parking berth: $W_b(u,p,v,q,b,m) := [\chi(b),\lambda(b)]$.

Although a shunting plan may be self-consistent in operation, the interaction between shunting plans may still make the overall schedule inoperable. We denote a *conflict* relation between two shunting plans π and π' as $\pi\dagger\pi'$. Due to the deterministic feature, conflict relations between every pair of fixed shunting plans can also be

pre-computed as input data for the second phase. We use

$$\Pi^{\dagger}(\pi) = \left\{\pi' | \pi' \dagger \pi, \forall \pi' \in \Pi\right\}$$

to denote the set of all shunting plans that have a conflict relation with shunting plan π and

$$C = \left\{(\pi, \pi') | \pi \dagger \pi', \forall \pi, \pi' \in \Pi\right\}$$

the set of all conflicting shunting plan pairs.

5.2 Model formulation

If the unit-type contents of each train derived from the first phase result remain unchanged, the fleet size and most other terms in the objective function (1) will remain unchanged even if the second phase modifies the first phase results by resetting some matching pairs. The second phase can be regarded as rematching arrival units with departure units at each station, or from a diagram's point of view, *swapping* trains between the diagrams derived from the first phase. Therefore the global optimality given by the first phase will be mostly preserved.

We denote the set of all dominant arcs pertaining to station σ as A^{σ}. As for each valid shunting plan (u, p, v, q, b, m), $(T(u), T(v))$ corresponds to a unique dominant arc $a \in A$, we thus denote the arc corresponding to shunting plan π as $a(\pi)$, i.e. if $\pi = (u, p, v, q, b, m)$, then $a(\pi) = (T(u), T(v)) \in A^{\sigma}$. Then, after the path variables $x_p, p \in P^t, t \in T$ from the first phase results have been transformed into arc variables by $x_a = \sum_{p \in P^t_a} x_p, \forall a \in A^t, t \in T$, it is possible to assign costs c_π to each shunting plan $\pi \in \Pi$ according to the following rule:

(i) If first phase result $x_{a(\pi)} > 0$, then $c_\pi = 0$;
(ii) If first phase result $x_{a(\pi)} = 0$, then $c_\pi > 0$ and can be set according to some preferences (e.g. traction type / parking berth).

By setting the above shunting plan costs c_π and minimizing $\sum_{\pi \in \Pi} c_\pi x_\pi$, where $x_\pi \in \{0, 1\}$ indicates whether to use a shunting plan $\pi \in \Pi$, it ensures that if the part of first phase result related with station σ does not cause any conflicts at all, it will be exactly retained after the second phase.

Similar as in Kroon et al (2008), it is desirable to have each siding serving as few types of unit as possible, yielding more flexibility and robustness. There is no such a need for platforms or depots. We denote $y^t_b, b \in S, t \in T$ as a binary variable indicating whether at least one unit of type t is parked at siding b, and try to minimize $\sum_{t \in T} \sum_{b \in S} y^t_b$.

Similar as in the first phase, it is possible to introduce binary blockflow variables $z_a, a \in A^{\sigma}$ indicating whether an arc a is used, which are used for forbidding coupling/decoupling at banned locations, calculating coupling/decoupling time durations at permitted locations and eliminating unnecessary such operations. We denote the set of all shunting plans pertaining to arc $e \in A^{\sigma}$ as Π_e, i.e. $\Pi_e := \{\pi | a(\pi) = e\}$. Then $z_a = \begin{cases} 1, \text{ if } \sum_{\pi \in \Pi_a} x_\pi > 0 \\ 0, \text{ if } \sum_{\pi \in \Pi_a} x_\pi = 0 \end{cases}, \forall a \in A^{\sigma}$.

Thus, the second phase model:

$$\min_{x,y,z} \quad W_1 \sum_{\pi\in\Pi} c_\pi x_\pi + W_2 \sum_{t\in T}\sum_{b\in S} y_b^t + W_3 \sum_{a\in A^\sigma} z_a \tag{12}$$

subject to

$$\sum_{\pi\in\Pi_u} x_\pi = 1, \quad \forall u \in U; \tag{13}$$

$$\sum_{\pi\in\Pi_v} x_\pi = 1, \quad \forall v \in V; \tag{14}$$

$$x_\pi \le y_{b(\pi)}^{t(\pi)}, \quad \forall \pi \in \Pi : b(\pi) \in S; \tag{15}$$

$$x_\pi \le z_{a(\pi)}, \quad \forall \pi \in \Pi; \tag{16}$$

$$\sum_{a\in\delta_-(j)} z_a = 1, \quad \forall j \in \widetilde{N}^\sigma; \tag{17a}$$

$$\sum_{a\in\delta_+(j)} z_a = 1, \quad \forall j \in \widetilde{N}^\sigma; \tag{17b}$$

$$\tau^D_{\mathrm{arr}(i)} \left(\sum_{a\in\delta^\sigma_+(i)} z_a - 1 \right) + \tau^C_{\mathrm{dep}(j)} \left(\sum_{a\in\delta^\sigma_-(j)} z_a - 1 \right) \le e_{ij}, \quad \forall (i,j) \in \widetilde{A}^\sigma; \tag{18}$$

$$x_\pi + \frac{1}{|\Pi^\dagger(\pi)|} \sum_{\pi'\in\Pi^\dagger(\pi)} x_{\pi'} \le 1, \quad \forall \pi \in \Pi : \Pi^\dagger(\pi) \neq \varnothing; \tag{19}$$

$$l_{t(\pi)} x_\pi + \sum_{\pi' : W_b(\pi')\cap W_b(\pi)\neq\varnothing} l_{t(\pi')} x_{\pi'} \le L_{b(\pi)}, \quad \forall \pi \in \Pi : b(\pi) \in S \cup D; \tag{20}$$

$$x_\pi \in \{0,1\}, \quad \forall \pi \in \Pi; \tag{21a}$$

$$y_b^t \in \mathbb{R}_+, \quad \forall b \in S, \forall t \in T; \tag{21b}$$

$$z_a \in \mathbb{R}_+, \quad \forall a \in A^\sigma. \tag{21c}$$

The first term in the objective (12) minimizes the cost of shunting plans selected; the second term encourages units of the same type to be grouped together as much as possible; the third term minimizes total number of unit blocks such that unnecessary coupling/decoupling activities are also reduced. Generally, the first term should be given a higher weight (W_1) than others (W_2 and W_3).

Constraints (13)–(14) match each arrival unit to a unique departure unit via a unique parking berth by a unique parking method, and the same for each departure unit. Similar concept for ensuring each position in coupled multi-type trains to be occupied by one and only one unit has been encapsulated implicitly by conflict sets.

Constraints (15) calculate the siding-type variables, where $t(\pi)$ is the traction type implied by shunting plan π and $b(\pi)$ is the parking berth of plan π. Notice here we use a disaggregated formulation with many constraints ($= |\Pi_{b(\pi)\in S}|$) rather than an aggregated formulation with less constraints, e.g. a multiple variable lower bound (MVLB) (Ciriani et al (2003)) like

$$\frac{1}{|\Pi_b^t|}\sum_{\pi\in\Pi_b^t} x_\pi \le y_b^t, \quad \forall b\in S, \forall t\in T, \tag{22}$$

where Π_b^t is the set of shunting plans implying a type t to be shunted to a berth b. The stronger disaggregated formulation (15) will generally give a much tighter LP-relaxation than (22), and also let y_b^t be only continuous while functioning binary. The side effect from excessive number of constraints in (15) can be neutralized if a column-and-dependent-row generation approach is used.

Constraints (16) calculate the blockflow variables for each dominant arc pertaining to station σ. Similar as in (15), we use a strong formulation rather than an MVLB formulation.

Only one from Constraints (17) and (18) will appear for a specific station σ, depending on whether coupling/decoupling is banned at σ (use (17)) or allowed (use (18)). $\widetilde{N}^\sigma$ is the set of train nodes related with station σ.

Constraints (19) prohibit conflicts between each pair of shunting plans. Note that a generic conflict (or crossing) exclusion formulation such as

$$x_\pi + x_{\pi'} \le 1, \quad \forall(\pi,\pi')\in C \tag{23}$$

will give a huge number of constraints leading to computational difficulty (e.g. Model 1 in Kroon et al (2008)). Some successful remedy methods, for instance, taking advantage of clique inequalities or comparability graphs (e.g. Model 2 and 2a in Kroon et al (2008)) are not suitable for the current problem, where the concept of conflict is based on a more complex context concerning positions, platforms, sidings and combined parking methods. Here we propose another remedy method as in (19). Compared with (23), the number of constraints has been significantly reduced, as usually $|C| \gg |\Pi|$. Notice the number of constraints will be further reduced if a column-and-dependent-row generation approach is used.

Constraints (20) prohibit overcapacity for each siding and depot berth, where $l_{t(\pi)}$ is the capacity consumed by type $t(\pi)$ implied from shunting plan π, and $L_{b(\pi)}$ is the total capacity of parking berth b implied from plan π. From observations of the manual scheduling process, feasible berthing plans can always be found in a station-by-station manner. Moreover, berth capacity would have been satisfied at the preceding timetabling stage, i.e. the input timetable determines the main unit flows on stations at certain times and places, as train unit scheduling only assigns units to predetermined trains.

Finally (21) gives variable domains.

6 Solution approaches

6.1 Pre/dynamic column generation

We first give a short discussion on ways columns are generated in a generic column generation process, which seems has not been given much attention in the literature.

Generally, if a generic column generation approach is to be used, except the initial columns providing a feasible solution to trigger the subsequent process, all new columns to be added to the restricted master problem (RMP) have to be generated from some methods. There are usually two ways for generating them. The first one is to generate dynamically from subproblems until no good candidate is available and theoretically the solution will be guaranteed global optimal. However, this approach has two major disadvantages, both due to the dynamic nature how new columns are generated. One is the difficulty in satisfying sophisticated rules imposed on each column; the other, arising in integer programs solved by branch-and-bound (BB), is that dynamically generated columns may have some properties inconsistent with the current branch status (Barnhart et al (1998)). These lead to the need for designing very complicated subproblems or ad-hoc branching strategies to exclude invalid candidates. Constrained shortest path subproblems for crew scheduling (Irnich and Desaulniers (2005)), generalized assignment problems (Savelsbergh (1997)) and single-path multicommodity flow problems (Barnhart et al (2000)) are well-known examples.

Alternatively, it is possible to perform a pre-generation beforehand trying to extract the essence of all possible candidates into a pot, and then carry on subsequent column generation process only based on this pot. Candidates are priced out by merely calculating and comparing their reduced costs, and unlike in dynamic-generation, it is convenient and flexible to add *several* candidates to the RMP from the same decomposed subproblem. Moreover, the above two disadvantages from dynamic-generation are unlikely to occur in this approach, making it more preferable for solving very large-scale complex problems, especially when sophisticated rules are imposed on individual columns. Its disadvantage is theoretically it cannot always guarantee global optimality, as the solution only converges to a local optimum with respect to the pre-generated pot. Examples of this style include Fores (1996) for bus driver scheduling, Kwan (2004) for large-scale crew scheduling and Hennig et al (2012) for maritime transport routing.

6.2 A hybrid approach for the first phase

Integer FCMF problems are generally believed to be very difficult to solve. The first difficulty in solving our specific model in Section 4.2 is the weak knapsack constraints corresponding to passenger demand satisfaction (constraints (3)) and platform length (constraints (7)). As already observed by Schrijver (1993), Ziarati et al (1999), and Cacchiani et al (2010), they will make the LP-relaxation give very poor lower bounds and always fractional solutions. Unfortunately, the Southern Railway instance we are dealing with does not provide sufficient characteristics for simplify-

ing those knapsack constraints by similar methods as in the above literature, and we have to keep them when solving the model.

We find that, as also observed in Cacchiani et al (2010), exact methods would be incapable of handling very large problem instances. Here we propose a Size Limited Iterative Method (SLIM) based on hybridization of an exact method (a branch-and-price integer linear programming (ILP) solver) and a heuristic framework.

6.2.1 A branch-and-price approach for the exact solver

A branch-and-price (Barnhart et al (1998)) approach is used for the exact ILP solver. Unlike in crew scheduling, the rules imposed upon each unit diagram are not very complex. Thus we use dynamic column generation where the subproblem is a generic shortest path problem. Although there are three kinds of variables, as the last two are solely determined by the path variables, we only price out path variables.

Let $\beta^t \le 0, \rho_j \ge 0, \phi_j^f \le 0, \psi_j^f \le 0, \lambda_j \le 0, \zeta_a \le 0$ be the dual variables from constraints (2), (3), (4), (5), (7), (8), $f(p)$ be the family of path $p \in P^t, t \in T$, and $c_p = c_p^0 + \sum_{j \in J_p} c_j + \sum_{a \in A_p} c_a$ be the generalized cost of p in the objective (1) consisting of three parts (relevant with nodes, relevant with arcs and irrelevant with any of them). The reduced cost of p from G^t is,

$$\bar{c}_p = c_p^0 - \beta^t + \sum_{j \in J_p} \left(c_j - \kappa^t \rho_j - \phi_j^{f(p)} - n^t \psi_j^{f(p)} - l^t \lambda_j \right) + \sum_{a \in A_p} (c_a - \zeta_a), \quad (24)$$

finding the smallest value of which can be regarded as a shortest path problem with train node $j \in \widetilde{N}$ the weight $c_j - (\kappa^t \rho_j + \sum_{f \in F_j} \phi_j^f + n^t \sum_{f \in F_j} \psi_j^f + l^t \lambda_j)$, arc $a \in A$ the weight $c_a - \zeta_a$, plus a source-sink weight $c_p^0 - \beta^t$. There are $|T|$ subproblems from Dantzig-Wolfe decomposition.

As for the branching strategy, it is a usual way to branch on the corresponding compact formulation while working on the extensive formulation (Villeneuve et al (2005)). This means we can branch on arc variables while performing the column generation with path variables. However, we find it difficult to design an efficient branching method with respect to arc variables by only deleting certain columns without adding explicit branching constraints, which is to be preferred. This is because the arc variables are integral rather than binary; moreover, each type can use multiple paths in the final integer solution, unlike the case in Barnhart et al (2000) where each commodity takes one and only one path. We here propose a mixed branching strategy consisted of two branching rules, namely *train-family branching* and *arc variable branching*.

Train-family branching The main idea of train-family branching is to check the LP relaxation solution and select a train j that is covered by more than one family, say families $f_1, f_2, \cdots, f_k$ (k is usually not a large number). Then we form $k+1$ branches with respect to families $f_1, f_2, \cdots, f_k$.

- For the first k branches $1,\cdots,k$, say at a branch $i \in \{1,\cdots,k\}$, only the family f_i is allowed to serve train j. To achieve this, in the RMP, all paths indicating any families in $F_j \setminus \{f_i\}$ serving j are deleted. Moreover, to prevent subproblem regeneration in a branch-and-price framework, in the shortest path problem of type t whose family is not f_i, node j is deleted from the shortest path network. Variables y_j^f and constraint (6) for train j can then be removed from the RMP, and constraint (4) (5) can also be modified according to which family serves that train.
- In the last $(k+1)$th branch, if $|F_j \setminus \{f_1,\cdots,f_k\}| \geq 1$, then we forbid families $f_1,\cdots,f_k$ to serve train j, which can be realized by similar path/node deleting ways as described above, and there is no removal/modification of related variables/constraints unless $|F_j \setminus \{f_1,\cdots,f_k\}| = 1$; if $F_j = \{f_1,\cdots,f_k\}$, then the $(k+1)$th branch is no longer needed.

Notice this branching rule does not add any extra constraints to the RMP. It actually reduces the number of columns in the RMP and the network scale of the shortest path subproblems, which effectively divides the search space and forces the train-family variables to be integral.

Arc variable branching The train-family branching is the first choice in the branch-and-bound tree. However, when all trains are served by single families, this branching rule cannot be used any more. At this moment, the second branching rule, arc variable branching, will be invoked. This rule detects fractional arc variables from the result of the RMP LP relaxation by $x_a^t = \sum_{p \in P_a^t} x_p$, and branches over this arc by explicit branching constraints. We use a constraint branching rule similar as the one for integer multicommodity flow in Alvelos (2005), i.e. on one branch, add constraint $\sum_{p \in P_a^t} x_p \leq \lfloor x_a^t \rfloor$, and on the other branch, add constraint $\sum_{p \in P_a^t} x_p \geq \lceil x_a^t \rceil$. As all those constraints are associated with arcs, the structure of shortest path subproblem can be kept by adding arc weights from dual variables of those constraints; invalid path generation is also prevented straightforwardly. We currently first select on the arcs with most fractional values to branch but find it is not very efficient. It is also a good idea to first branch on fractional arcs at peak times, as is used in Schrijver (1993). For node selection, a best-first strategy is used, which will always provide us a global lower bound.

The second rule will only be triggered when the first rule cannot be used. Notice after the second rule is used, there may be some trains served by multiple families again, and thus the first rule will be called again until all trains are again covered by single families, and the switch to the second rule, and so on.

6.2.2 A size limited iterative method

As an exact solver is incapable for large-scaled problems, we further propose a hybridized method which we call *Size Limited Iterative Method* (SLIM). Its rationale is based on PowerSolver proposed by Kwan and Kwan (2007) for crew scheduling, but the train unit scheduling problem is more complex requiring further exploration. The idea is to compact the problem instance such that the embedded exact solver can handle it comfortably. As it proceeds, according to previous solution results plus certain

supplementary methods, the problem instance will be adjusted to derive a different new instance that would not yield a worse result, and to be solved again by the exact solver. This process will be repeated until no significant improvement. The problem instance compaction retains all the train trips, i.e. the method does not subdivide the problem instance into small separate subproblems.

Size reduction on problem instance can be achieved by mainly two methods:

(i) Simplifying permitted types available for trains such that no type incompatibility can occur;
(ii) Pre-sequencing some trains into *super-trains* so that they will each be scheduled as an integral block.

The first method eliminates a large number of variables and constraints used for ensuring type-type compatibility. As this heuristics proceeds, types that have once been deleted from a train will have the chance to be added back based on certain mechanisms to allow solution improvement. The second method significantly reduces the number of "trains" by pre-sequencing them into super-trains. According to the current solution status, super-trains would be reformed between iterations.

Outline of SLIM

1. Associate trains with a subset of permitted types.
2. Form a set of super-trains each replacing the original trains in the sequence.
3. Solve the reduced problem instance derived using the ILP solver. If it is not the first result obtained and there is no improvement on the objective for a predefined number of iterations, GOTO 7.
4. Copy the last problem instance to a new instance and then retain from the current best solution only the linkage arcs used.
5. Extend the search space by decomposing some super-trains and/or forming some new super-trains according to some defined criteria and make sure the resulting solution will be no worse than the current best solution.
6. Heuristically include more potential arcs into the problem instance, ensuring that the problem size is not exceeding a predefined upper limit. Go back to 3.
7. If the full set of permitted types is allowed for all trains, STOP. Otherwise, extend the search space by gradually re-instating the full set of permitted types. Go back to 3.

6.3 A column-and-dependent-row with pre-generation approach for the second phase

Although the second phase is a 6-dimensional matching problem with side constraints, all possible candidate solutions have been encapsulated into shunting plans $\pi \in \Pi$, where components of each (u, p, v, q, b, m) are determined with far less flexibility. The actual size of $|\Pi|$ is expected to be moderate such that a state-of-art MILP solver incorporating BB/branch-and-price would be able to handle it.

Column-and-dependent-row generation A challenge for the second phase model, say if column generation is to be used, is the constraints based on columns (constraints (15), (16), (19) and (20)), which will cause theoretical problems that make the column generation invalid. Simply speaking, if only a subset of columns are present in the RMP, then the rows corresponding to the missing columns will be also absent with their dual variables unknown, leading to incorrect reduced cost calculation, not to mention the primal feasibility is also no longer guaranteed. This is an interesting new research direction of column generation, and very few relevant literature can be found. The only focused research so far, is from Muter et al (2012), which gives a deep insight into it, and also proposes a solution approach for two-variable cases. We believe that it is possible to expand the method in Muter et al (2012) to some three-variable cases, as our second phase model. We will continue to focus on this special kind of column generation problem, and propose a column-and-dependent-row generation approach for our model in the future.

Pre-generation and initial test We use a pre-generation approach for our column-and-dependent-row generation. The complex rules imposed on individual shunting plans make it almost impossible to generate such candidate plans dynamically. All possible candidate shunting plans forming the set of Π will be pre-generated. Another major step beforehand is to construct the conflict sets $\Pi^{\dagger}(\pi)$ for each shunting plan $\pi \in \Pi$. For efficiency reasons, an initial testing phase is used where only a set of columns corresponding to the first phase results (i.e. all the shunting plans with zero costs $c_{\pi} = 0$) is provided for the RMP. If those columns yield a feasible solution for the second phase, then the process is stopped and optimality claimed. Otherwise, extra columns have to be added to construct an initial feasible solution and then the commence of a column generation process.

Heuristics Some heuristics speeding up the solution process can be also used, e.g. the idea of utilizing homogenous sidings to get near-optimal solutions quickly (Kroon et al (2008)). Another direction to be explored is the *order* how stations are processed. Clearly different orders will very likely give different solutions, since solving a station will simultaneously fix some characteristics of arrival/departure units of other related stations, and it is not clear yet how to determine the order to improve the overall solution.

Branching Strategy A mixed branching system is to be used, inclduing: (i) branching on activated dominant arcs $a \in A^{\sigma}$; (ii) branching on shunting plans $\pi \in \Pi$. Activated dominant arcs refer to the arcs whose implied shunting plans have been priced out into the RMP. The former is usually used at early stages of the branching process and the later is more useful at later stages. As pre-generation is used, no subproblem regeneration will occur.

7 Computational experiments

7.1 Southern railway dataset

This research benefits from collaboration with Southern Railway, UK, which is a very large passenger train operator in the South of England. Each year, there are two timetables published. A full set of data for a timetable in December 2011, including their actual unit diagrams used, has been used for developing our models and for testing. The timetable concerned has over 2400 train trips on a typical weekday, and 11 different types of train units are in use. Furthermore, the network covered is extensive with numerous routes, stations and platforms. Hence, the setup and configuration for testing our models is in itself a very substantial ongoing task. Whilst some promising results have been obtained, they still have to be carefully analyzed with Southern Railway. More comprehensive results therefore will be reported in a future paper.

Knapsack constraints The Southern Railway's data has very complicated type-route compatibility relationships, giving large numbers of unit types available for most of the trains (e.g. 7 types can serve the London Bridge – East Croydon route), mainly due to the use of four types of British Rail Class 377, which basically can run on any route. Moreover, Class 377/3 has a coupling upper bound of 4 units, and Class 377/1, 377/2 and 377/4 have a bound of 3 units. If using family generalization, the family Class 377 has an upper bound of 12 cars with various car number coefficients in the constraint. These features forbid us from using the knapsack constraint simplification method developed by Cacchiani et al (2010) (for a coupling upper bound of 2 units) for almost all trains. Thus we have to retain all knapsack constraints at this stage. Since Southern Railway cannot provide us with passenger demand data, all passenger demands per train (r_j) are derived reversely from the Southern Railway December 2011 diagrams. Notice this will not surely alleviate the difficulty of knapsack constraints, as the unit types other than the one used in the real diagrams also appear in the demand satisfaction constraints. Also, not all platform length constraints are present in the experiments since complete data was not available when we did the testing.

Arc length In constructing the DAG network, there is a minimum and maximum time restriction for linkage arc lengths, currently set to be 5 min and 720 min respectively. We are aware that setting the latter a much smaller value (e.g. 360 min) will significantly reduce the number of arcs, which dominates the number of rows in the formulation, while most of the "long" arcs will never be used. However, the need for long gaps for maintenance and off-peak depot berthing requires us to retain them. Typically there is a morning peak and an afternoon peak in the timetable, and between the two peaks it is preferred to have most of the units back to some nearby depots, and some of the units may be under maintenance operations during this idle time.

Empty-running trains We use existing empty-running trains extracted from the actual diagrams in operation, plus some extra ones computed from the timetable and

route network information using a shortest path algorithm. There were originally some constraints for eliminating conflicting empty-running trains in the first phase formulation, but we finally discarded them for not further increasing the problem scale.

7.2 Branch-and-price solver design

The main purpose of our experiments is to verify the validity of the first phase model, rather than to solve the entire problem as a whole or to test our ad-hoc solver's performance, which will be reported in continuing research. For all the experiments reported here, only the first phase is involved, and SLIM is not used. Notice we have to retain all knapsack constraints. We will first shortly describe some practical aspects of our branch-and-price solver used in the experiments.

7.2.1 Primal heuristics

We use a greedy heuristics to generate the initial columns for the column generation at the root node of the BB tree. The main rationale of this primal heuristics is to select the tightest arc (with respect to time) for linking the next train at each train node, always checking other restrictions like type-type compatibility, coupling upper bound, coupling/decoupling time allowance, etc. Ability to self-rescuing/reconstructing is also embedded when the heuristics is trapped into infeasible solutions. This primal heuristics will generally provide a good quality initial solution for the subsequent stage. For instance, it has produced 15 paths for timetable 1 in Table 4, whose optimal final integer solution is 13.

7.2.2 Branch-and-bound tree control

We do not let the LP relaxation problem at each BB node to be always solved to optimality, as is suggested in Barnhart et al (1998). A lower bound is computed from dual variables and the solution of shortest path subproblems. If the gap between the current LP solution objective and this lower bound is less than a tolerance ε and the LP is not optimal yet, we will regard the LP at this node *almost* optimal and use this lower bound as its node value. We find that the tolerance ε is very sensitive in reflecting computation time, and is very problem-specific. It is also a common phenomenon that a rough value (e.g. $\varepsilon = 0.2$) would yield a solution of the same quality as a very small tolerance (e.g. $\varepsilon = 0.001$), with far less time.

Another parameter M_{CG} is also set as the maximum number of LP iterations column generation process is allowed at each BB tree node. The reason is to temporarily abandon the nodes with very difficult LPs. When M_{CG} is hit at a node, we will compare the current LP objective with another tolerance ε', which is always larger than ε. If the current LP objective gap is still less than ε', we remit this node and take the current lower bound as its node value (which is still a valid lower bound anyway); if the gap is greater than ε', then we cut off this node and put it into a waiting-list to be

solved later. As all the nodes in the list contain difficult LPs, powerful solution methods are to be used, involving cut generation. The waiting-list will not be processed until all other "normal" nodes have been visited and global integer solutions are still out of reach. We find that parameters M_{CG} and ε' are also very sensitive in resulting computation time. We currently set $M_{CG} = 50$ and $\varepsilon' = 1$ for most instances.

7.2.3 Column management

We do not initialize the column generation at each new leaf-node with new feasible columns (e.g. artificial columns or from subproblem variants) every time. We directly let each new leaf-node inherit all the columns from its parent node. This will provide a good-quality feasible solution to a new leaf-node at the beginning compared with starting from brand new columns, and it can also prevent taking irrelevant columns from other nodes—e.g. to take columns generated from the parent's sibling(s), whose branching is quite opposite. When inheriting parent's node results in an infeasible solution, it does not necessarily indicate infeasibility—it is possible that there exists a feasible solution but the limited existing columns simply cannot construct it, which is a disadvantage of dynamic column generation. At this moment, artificial columns will be generated. If even these artificial columns yield an infeasible solution or they cannot be constructed, then we conclude infeasibility of this leaf-node. We also have tried to let each new leaf-node only inherit the basic or small reduced-cost-valued columns from its parent, but it will frequently give infeasible initial LP, leading to artificial column construction.

The choice between inheriting parent's columns or constructing new feasible columns for the initial LP at leaf-nodes still needs more careful research. The former will make each leaf-node LP many columns but less LP iterations; the latter may have each leaf-node less columns but require more LP iterations. Clearly there is a trade-off between the two and we are interested in how to achieve a balance.

Some other column management techniques are also used, for instance, to locally delete the columns whose reduced costs are larger than the gap derived from the dual information, which are guaranteed to be non-basic in an optimal LP. We also tried to delete columns whose reduced costs are larger than a preset number, or columns which have not been chosen for more than a preset number of iterations, finding that these will generally slow down the process greatly, or even sometimes give infeasible final solutions, however.

7.3 Computational results

We have tested some subsets extracted from the entire Southern Railway December 2012 timetable, each corresponding to trains in real diagrams served by specific type(s). The original aim is to make comparisons with manual diagrams.

Table 4 gives the results from our numerical tests. SLIM is not used, and only the first phase is involved. Sometimes if the computation time is too long, only root columns are taken into the BB tree to get a near-optimal solution. The objective is to only minimize the total number of units used and also to reduce blockflows (i.e.

$W_1 \sum_{t\in T} \sum_{p\in P^t} x_p + W_7 \sum_{a\in A} z_a$). The tests are carried out using a 64 bit Xpress-MP 7.2 on a Dell workstation with Intel Xeon CPU E31225. When the problem has been decomposed into $|T|$ shortest path subproblems, they are processed in parallel with the help of Xpress-MP's synchronization ability between master- and sub-models.

Table 4 Results from numerical tests

Timetable set	No. of trains	No. of units used in practice	Integer solution	Time (seconds)
1	102	13	13	5.065
2	164	17	12	15.652
3	207	22	41	94.979
4	377	49	50* (51.44%)	$>$ 7200
5	496	42	56* (50.48%)	$>$ 7200

Notice due to the constraints associated with arcs in the network, plus extra train-family and blockflow variables/constraints, the problem scale could be very large even for a network with a small number of nodes (trains). With only the exact branch-and-price solver without SLIM, we could not find integer solutions for problem instances more than 500 trains. For the problem instances within 500 trains, integer solutions with various qualities can be found, and if the number of trains is no more than 200, solutions can be found within a relatively short time.

Timetable 1 is actually the trains derived from the diesel set (Class 171/7 and 171/8). We find a solution with the same number of units as the manual one, including many identical diagrams. All of the expected soft and hard requirements for the first phase are also successfully achieved. On the other hand, after an analysis in a station-detailed level, there are diagrams that are infeasible due to train direction and platform blockage, unless extra shunting or swapping actions are taken. For example, we find that 171/8(3) arrives at London Bridge at 14:49 which is after 171/8(1), so 171/8(1) cannot depart at 21:04 as indicated in its diagram because 171/8(3) is in the way. In fact, simply swapping the two will resolve this difficulty, which is exactly what the second phase can do. This clearly illustrates the necessity of the second phase.

Similar phenomena were found in the other experiments in satisfying all first phase requirements but experiencing frequent station blockage problem. For Timetable 2 with 164 trains, we find a much better solution (12 units) than the manual diagrams (17 units). We are only able to find an integer solution (41 units) much worse than the manual one (22) for Timetable 3, which indicates the problem-specific nature such that the solver parameters have to be carefully tuned for different instances. Timetable 4 and 5 are medium sized instances where we only found non-optimal integer solutions (marked with "*" and followed by the gap from root node LP-relaxation) after several hours. Although they are worse than the manual solutions, the LP-relaxation value (although known to be very weak) indicates a great potential for improvement.

We have also obtained from Table 4 a threshold the maximum problem size for which our current exact solver can handle comfortably—around 200 trains. This is an important parameter for designing SLIM, where we would like to sequence trains such that the total number of super-trains will be around this threshold. For the entire Southern Railway timetable with around 2400 trains, this means it is sufficient for

each super-train to contain about 12 (original) trains. Of course, the power of the exact solver itself has to be further improved in the future.

8 Conclusions

This paper has explored a hugely complex real-life problem of train unit scheduling. A two-phase approach is proposed, in which the problem is first tackled from a network wide perspective and then scrutinized at the fine-detailed individual train station level. While mathematical programming models have been developed, hybridization of the first phase solver with an iterative heuristics is also proposed. The use of train-family variables to achieve type-type compatibility and coupling upper bound due to families, and the use of blockflow variables to enforce coupling/decoupling restrictions with respect to location and time allowances, is new in the literature to our knowledge so far. The two-phase approach is also novel in train unit scheduling, especially the concept of "dominant arcs" to link the two phases.

We have carried out experiments on various technical aspects in deriving a practical solver and made some numerical experiments on some small and medium scaled instances from real data of Southern Railway, one of the UK's largest train operators, using the exact branch-and-price solver without SLIM and only involving the first phase. What we have gained from these experiments are two-fold:

- The first phase model's capability of solving the core features of the train unit scheduling problem, especially for type-type compatibility, redistributing unit resources by coupling/decoupling, and forbidding/restricting coupling/decoupling activities with respect to locations/time allowances, have been verified and our conclusion is positive. Moreover, it is reflected from the experiments that without the second phase, the diagrams would very likely to be infeasible due to station blockage. If the research is about to solve real-world instances, the second phase for eliminating station blockage is indispensable.
- We believe that as long as the knapsack constraints are present, the exact solver will perform very poorly. Unfortunately the problem characteristics currently prevent us from developing a simplification method for those knapsack constraints. We have to switch to heuristics, and SLIM is an important future research direction. Moreover, we are also interested in designing simplification methods over those knapsack constraints for instances from other smaller train operators.

We realize the complexity and sophistication the problem we are dealing with. Testing and refinement of the models, especially on the SLIM heuristics, with feedbacks from Southern Railway are ongoing. Future research will include more computationally efficient solution methods, possibly taking advantage of convex hull technique, heuristics, parallel computation, and further refinement of the proposed models taking into account problem instances from more train companies. Further research will also include investigation, theoretical and/or empirical, of error bounds on the solutions under our proposed two-phase approach.

Acknowledgements This research is sponsored by a Dorothy Hodgkin Scholarship funded by the Engineering and Physical Sciences Research Council (EPSRC) and Tracsis Plc. We would also like to thank Southern Railway for their kind and helpful collaboration.

References

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

Alvelos F (2005) Branch-and-price and multicommodity flows. PhD thesis, Escola de Engenharia, Universidade do Minho, Portugal

Barnhart C, Johnson EL, Nemhauser GL, Savelsbergh MWP, Vance PH (1998) Branch-and-price: Column generation for solving huge integer programs. Operations Research 46:316–329

Barnhart C, Hane CA, Vance PH (2000) Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. Operations Research 48(2):318–326

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

Ciriani TA, Colombani Y, Heipcke S (2003) Embedding optimisation algorithms with mosel. 4OR 1(2):155–167

Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1998) Combinatorial Optimization. Wiley, New York

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

Fores S (1996) Column generation approaches to bus driver scheduling. PhD thesis, University of Leeds, the UK

Fores S, Proll L, Wren A (2002) TRACS II: A hybrid IP/heuristic driver scheduling system for public transport. The Journal of the Operational Research Society 53(10):1093–1100

Freling R, Lentink RM, Kroon LG, Huisman D (2005) Shunting of passenger train units in a railway station. Transportation Science 39(2):261–272

Geoffrion A (1974) Lagrangean relaxation for integer programming. Mathematical Programming Study 2:82–114

Hennig F, Nygreen B, Christiansen M, Fagerholt K, Furman KC, Song J, Kocis GR, Warrick PH (2012) Maritime crude oil transportation — a split pickup and split delivery problem. European Journal of Operational Research 218:764–774

Irnich S, Desaulniers G (2005) Shortest Path Problems with Resource Constraints, pp 33–65. In: Guy Desaulniers and Jacques Desrosiers and Marius M. Solomon (eds) Column Generation, Springer US, New York

Kroon LG, Lentink RM, Schrijver A (2008) Shunting of passenger train units: An integrated approach. Transportation Science 42(4):436–449

Kwan RSK (2004) Bus and train driver scheduling, chap 51, pp 51(1)–51(19). In: Leung, J. Y. (ed) Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, Boca Raton

Kwan RSK, Kwan A (2007) Effective search space control for large and/or complex driver scheduling problems. Annals of Operations Research 155(1):417–435

Maróti G (2006) Operations research models for railway rolling stock planning. PhD thesis, Eindhoven University of Technology, the Netherlands

Maróti G, Kroon LG (2005) Maintenance routing for train units: The transition model. Transportation Science 39(4):518–525

Muter I, Birbil SI, Bulbul K (2012) Simultaneous column-and-row generation for large-scale linear programs with column-dependent-rows. Mathethatical Programming Avaiable online at http://dx.doi.org/10.1007/s10107-012-0561-8

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

Savelsbergh MWP (1997) A Branch-and-Price Algorithm for the Generalized Assignment Problem. Operations Research 45:831–841

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

Tracsis Plc (2013) Tracs-RS—rolling stock planning software. URL http://www.tracsis.com/software/tracs-rs

Villeneuve D, Desrosiers J, Lübbecke ME, Soumis F (2005) On compact formulations for integer programs solved by column generation. Annals of Operations Research 139(1):375–388

Wren A, Fores S, Kwan A, Kwan R, Parker M, Proll L (2003) A flexible system for scheduling drivers. Journal of Scheduling 6:437–455

Ziarati K, Soumis F, Desrosiers J, Solomon MM (1999) A branch-first, cut-second approach for locomotive assignment. Manage Sci 45:1156–1168