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 infrastructure details at train stations. The solutions of the first phase would be near-operable. The second phase focuses on satisfying the remaining constraints with respect to detailed station layouts. Real-world scenarios such as compatibility among traction types and time allowances for coupling/decoupling activities are considered, so that the solutions would be fully operable.

The first phase is modeled as a large scale integer multi-commodity network flow (MCNF) problem. A branch-and-price integer linear programming approach is proposed. Preliminary experiments have shown that the approach would not yield solutions in reasonable time for problem instances exceeding about 1000 train trips. The train company collaborating in this research operates over 2400 train trips on a typical weekday. Hence, a heuristic has been designed for compacting the problem instance to a much smaller size before the ILP 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.

Keywords train unit scheduling · railway shunting · large-scale integer linear programming · column generation

1 Introduction

A train multiple unit, or *train unit* for short, is a set of train carriages with its own built-in engine(s), i.e. it is not pulled by a separate locomotive, and being 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 to increase the passenger carrying capacity or to redistribute the units across the rail network. *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* where a *unit diagram* is a standardized documentation of the sequence of trains and other auxiliary activities, e.g. coupling and decoupling, an individual train unit will serve on a specified day of operation.

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

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 multi-commodity network flow (MCNF) 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, route preferences and a penalty measure of the extent to which the train capacity achieved is not meeting targets.

The second phase, as a *railway shunting subproblem*, finally determines the finer operational plan details. For example, precise track and platform layout constraints are considered. The shunting (empty movements between platforms, sidings and depots) subproblems are modeled as a multidimensional matching problem, which eliminates "crossing" shunting movements and minimizes operational costs. The branch-and-price (Barnhart et al (1998)) approach, which has been successfully applied in solving large-scale integer linear programming (ILP) problems, will be used. In applying branch-and-price, the alternatives of column generation by means of either solving a sub-problem dynamically or selecting from a vast pre-generated set are compared.

The remainder of this paper is organized as follows. Section 2 introduces the train unit scheduling problem and highlights some aspects and features that are essential or important in 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. Finally Section 7 draws conclusions and remarks on the ongoing testing, with collaboration with Southern Railway, UK, and further research.

2 Problem Description

Given a set of fixed trains in a railway operator's daily timetable, 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 (forming 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, which are often higher than the minimum capacity of individual units in the fleet, as much as possible.

2.1 Scheduling Constraints

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

(i) Each train must be served by at least one unit.
(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) The sequence of trains assigned to a unit must be compatible in terms of the unit traction types and the routes traversed
(iv) The unit diagrams must not be in temporal or spatial conflict with each other. Such conflicts often arise from track and platform layouts at the train stations.

Some soft constraints are:

(i) Each train should be covered by a *train set* of one or more units whose total capacity satisfies the passenger demand predicted for the train.
(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 Coupling and Decoupling Activities

A challenging aspect of train unit scheduling is that sometimes more than one units can be assigned to cover the same train, known as *coupling* (or attaching); the reversal activity is called *decoupling* (or detaching). Units of the same traction type (disregarding the number of carriages, which might be referred to as sub-types) can be coupled together, although a certain small number of different types may also be treated as compatible for coupling. The maximum number of units that can be coupled depends on many factors such as platform lengths and traction types. There are also restrictions on where coupling/decoupling activities are permitted to take place.

Coupling/decoupling activities are often essential to satisfying passenger demands. However, the time allowances required (typically 2 to 5 minutes) for accomplishing such activities may 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. Passengers would have to pay attention as to which carriages are associated with which destinations. Maróti (2006) refers to these special cases of coupling/decoupling as "joining/splitting".

For a set of coupled units (a *unit block*), not only its *composition* is important, but also its *permutation/order* of units. The reason is that sometimes coupling/decoupling can only be performed in certain order; it may also cause issues like blockage in shunting movements.

Although appropriate coupling/decoupling activities may contribute to an optimal schedule, they would also consume resources like tracks, shunting operations, crew 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.

2.1.2 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 any 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 to ensure that it is not in conflict 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.

Empty running may be planned on non-passenger routes. This may either shortcut 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.

Table 1 Example trains at the same platform

Train No.	Origin – Destination	Departure time	Arrival time
1	York – Leeds	09:30	09:50
2	Sheffield – Leeds	09:20	09:55
3	Leeds – Brighton	10:10	14:30
4	Leeds – London	10:20	13:30

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 turn out to be infeasible.

2.1.3 Shunting Movements and 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 (i.e. without infrastructure information on track layouts etc.), consider a simple example. Table 1 gives a timetable of four trains. Suppose at Leeds Station all the four trains have been assigned to the same platform, where the tracks are *bidirectional* – i.e. both "up" and "down" movements are allowed. 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:

(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, illustrating how train directions and infrastructure can seriously 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

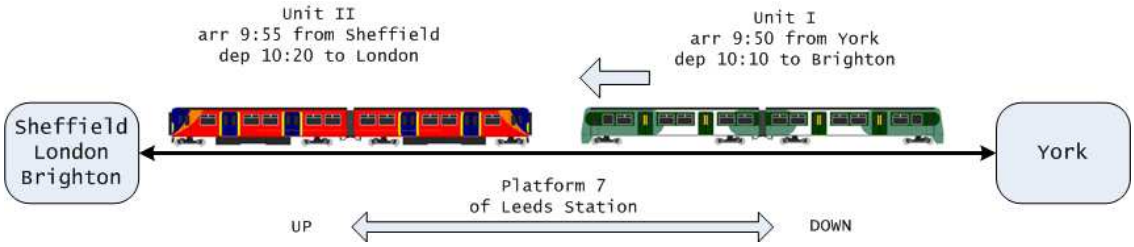

Fig. 1 Crossing problem with the FIFO solution (i)

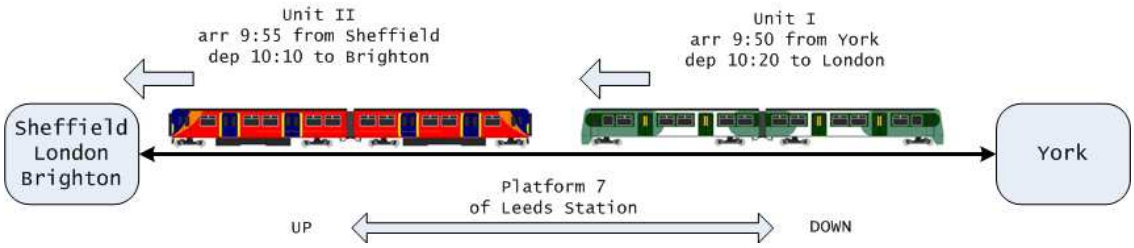

Fig. 2 A feasible solution (ii)

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 algorithm will lead to an extremely huge-sized optimization problem instance 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

Our discussion will focus on each of the two phases of the proposed approach as well as solving large-scale integer linear programs.

3.1 Train Sequencing and Fleet Assignment Problem

Bodin et al (1983) give a general survey on vehicle scheduling. Maróti (2006) and Cacchiani et al (2010) have reviewed relevant railway scheduling problems. Researches specifically on train unit scheduling are relatively few. In this section, we shall focus on the sequencing/circulation aspects, shunting and infrastructure related research will be discussed in the next section.

Schrijver (1993) proposes a multi-commodity circulation flow model concerning the rolling stock scheduling problem for a single line and a single day for NS Reizigers with about a hundred train trips; the objective is to minimize the total number of train units used. Passenger capacity demands, maximum number of units coupled, multiple types of train unit and cycling/maintenance are considered. Other coupling/decoupling and shunting/infrastructure restrictions are ignored. Some polyhedral techniques are used to speed up the solution process.

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. Some train trips in this research have been partially pre-sequenced into "trains" on a single line. Swapping trials for improvements is done locally in a station-based manner.

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.

More recently, Cacchiani et al (2010) propose an integer MCNF model solved by column generation with shortest path subproblems including some heuristic methods to improve efficiency. This model takes considerations of coupling/decoupling and vehicle-route compatibility, and is reported to have been applied for real-world instances of up to 600 train trips. Coupling/decoupling time allowances, traction type compatibility and shunting restrictions do not seem to have been considered in the model.

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 first phase consists of matching arrival units to departure units such that blocks of linked units are generated; the second phase assigns the unit blocks to appropriate sidings. 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, and the gap from global optimality seems to be small as reported.

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. This model uses a 3-dimensional matching formulation, requiring a reasonable amount of computation time to obtain high quality results such that unnecessary coupling/decoupling is minimized and units of the same type are encouraged to be parked at the same siding. To avoid a possible huge number of crossing constraints, maximal cliques and comparability graph techniques are used. Some heuristics are also employed, for example, taking advantage of homogeneous sidings to reduce crossing constraints.

Cardonha and Borndörfer (2009) propose a novel set-partitioning model and an associated column generation approach for a generalized vehicle shunting problem. The model provides a tight linear description of the problem and can, in particular, produce non-trivial lower bounds. The pricing problem, and the LP relaxation itself, can be solved in polynomial time, for some versions of the problem.

3.3 Solving Large-scale Integer Linear Programs

3.3.1 Branch-and-price and Integer Multi-commodity Network Flow Problem

branch-and-price is a successful approach for solving large-scale ILP problems which was used in vehicle routing and scheduling problems (see Desrosiers et al (1995), Vanderbeck and Wolsey (1996) and Savelsbergh (1997)). Barnhart et al (1998) give an important generalization of branch-and-price. However, for most real-world ILP problems, customised branch-and-price methods have to be designed. Barnhart et al (2000) describe a branch-and-price-and-cut algorithm for an integer origin-destination MCNF problem; Holmberg and Yuan (2003) present an MCNF problem with side constraints solved by column generation. Alvelos (2005) discusses the overall issues. Generally a path formulation (extensive formulation) is to be used as the master problem while branching can be performed on the corresponding arc formulation (compact formulation). Vanderbeck (2000) andVilleneuve et al (2003) give some theoretical analyses on the relation between compact and extensive formulations where branch-and-price is used.

3.3.2 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 seeking integer solutions. 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 Multi-commodity 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 MCNF model similar to Cacchiani et al (2010). But the model is more comprehensive in modeling real-world conditions and constraints typical of railway operations, especially the coupling/decoupling time allowance constraints and type compatibility constraints. The aim is to produce solutions that are near to being fully operable. The objective function minimizes a total weighted cost based on fleet size, operational costs, empty movement mileage, route preferences, idle time and a penalty measure of the extent to which the train capacity achieved is not meeting targets.

The second phase (railway 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 MCNF model is based on a framework of *directed acyclic graph* (DAG) representing trains and their linkage relations. A generic DAG $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ 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*. In addition, a *source node* 0 and a *sink node* ∞ are added as usual. There are three kinds of arcs in the DAG, namely *sign-in arcs*, *sign-off arcs* and *linkage arcs*. A sign-in arc starts from the source node and terminates at a train node; a sign-off arc starts from a train node and terminates at the sink node. Generally all train nodes have a sign-in arc coming into it and a sign-off arc leaving from it. A linkage arc 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 block with permitted type(s) can continue to serve train j as its next task. A *path* 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 the daily workload (a diagram) for a unit.

Table 2 describes the symbols representing entities in the DAG.

In order to deal with type-route compatibility, type-graphs $\mathcal{G}^t$ (are also DAGs) representing routes each unit type $t \in \mathcal{T}$ is able to cover are constructed based on the above complete DAG $\mathcal{G}$.

Table 2 DAG entities

Symbols	Entities in the DAG
0	the source node
∞	the sink node
$i, j, ...$	nodes
$\mathcal{N}$	set of all nodes
$\mathcal{N}^\circ = \{0, \infty\}$	set of source and sink nodes
$\widetilde{\mathcal{N}}$	set of train nodes
a or (i, j)	an arc
$\mathcal{A}$	set of arcs
$\delta_-(j)$	set of in-flow arcs of node j
$\delta_+(j)$	set of out-flow arcs of node j
$\mathcal{A}^\circ$	set of sign in and off arcs
$\widetilde{\mathcal{A}}$	set of linkage arcs
$v_-(a)$	start vertex (node) of arc a
$v_+(a)$	end vertex (node) of arc a
$\mathcal{E} \subset \widetilde{\mathcal{A}}$	set of all empty-running trains (represented by the arc containing it)
$\mathcal{E}^\dagger(a) \subset \widetilde{\mathcal{A}}$	set of all conflicting empty-running trains of empty-running train inside arc a
p	a path
$\mathcal{P}$	set of paths
$\mathcal{P}_j$	set of paths passing through node j
$\mathcal{P}_a$	set of paths passing through arc a
$\mathcal{J}_p$	set of *train* nodes in path p
t	a unit type
$\mathcal{T}$	set of all unit types
f	a unit family (see Section 4.2)
$\mathcal{F}$	set of unit families (see Section 4.2)

Table 3 Parameters

Symbols	Meaning
$W_i,\ i = 1, \cdots, 8$	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	a binary indicator showing whether arc a is a long gap for type t
π_j^t	weight for the preference of type-train pair (t, j): the higher, the more suitable
π_a^t	weight for the preference of type-sign-in/off arc pair $(t, a), a \in \mathcal{A}^\circ$: the higher, the more suitable
$\overline{F}^t$	upper bound of fleet size of unit type t
$\underline{F}^t$	lower bound of fleet size of unit type t
κ^t	unit capacity of type t
$\underline{r}_j$	hard (lower) passengers' demand for train j
$\overline{r}_j$	soft (upper) passengers' demand for train j
U_j^f	maximum number (in cars or units) of family $f \in \mathcal{F}$ coupled at train j
ν^t	number of cars of unit type t
l^t	length of unit type t
$L_j := \min\{L_{\mathrm{dep}(j)}, L_{\mathrm{arr}(j)}\}$	the minimum length between departure platform and arrival platform of 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}	spare time in linkage arc $(i, j) \in \widetilde{\mathcal{A}}$, used for time allowance validity for coupling/decoupling activities
M_a	the possibly largest number of coupled units at arc a

All entities in the type-graphs $\mathcal{G}^t$ can be symbolized by adding a t superscript in the corresponding notations of the complete DAG $\mathcal{G}$, e.g. $\mathcal{P}^t$ refers to set of all paths in type-graph $\mathcal{G}^t$.

Table 3 describes the parameters used in the model.

A path p in a type-graph $\mathcal{G}^t$ of type t (denoted as $p \in \mathcal{P}^t$) represents a sequence of trains to be covered by a unit or units of type t (if the latter case, it means the units are coupled together throughout the day). Also, if more than one flow/selected path share a common train node, it means this train is covered by a unit block coupled by the units from those paths. A variable $x_p \in \mathbb{Z}_+$ is used to indicate the number of unit(s) of type t used to serve the trains in path $p \in \mathcal{P}^t$. Moreover, family-train indicator variable $y_j^f \in \{0, 1\}$ is used to indicate whether train

$j \in \widetilde{\mathcal{N}}$ is served by a type from family f (see Section 4.2); it can be used to calculate the existence of specific types in a coupled block such that incompatible types cannot be coupled. Blockflow variable $z_a \in \{0,1\}$ is used to indicate whether an arc $a = (i,j)$ is used. It is a block of units that remains coupled together during two consecutive trains i and j. z_a is used to calculate the time consumed by coupling/decoupling operations; it can also be used to eliminate unnecessary coupling/decoupling activities at each station.

Table 4 describes the decision variables used in the MCNF ILP model.

Table 4 Decision variables

Symbols	Decision variables for
$x_p \in \mathbb{Z}_+$	*path flow variable*: to indicate the number of units of type $t \in \mathcal{T}$ used in path $p \in \mathcal{P}^t$
$y_j^f \in \{0,1\}$	*family-train indicator variable*: to indicate whether train $j \in \widetilde{\mathcal{N}}$ is served by family $f \in \mathcal{F}$
$z_a \in \{0,1\}$	*blockflow variable*: to indicate whether arc $a \in \mathcal{A}$ is used

4.2 Some Issues on Coupling/Decoupling

4.2.1 Restricted Locations for Coupling/decoupling

This can be achieved by setting the single coupling/decoupling operation time of that location (usually a platform) into a very big number, e.g. $\tau^C_{\mathrm{dep}(j)} := M,\ \tau^D_{\mathrm{arr}(i)} := M$, then such operations are prohibited at those locations (cf. Eq. (1)).

4.2.2 Time Consumed by Coupling/decoupling in a Linkage

Associated with an arc joining two train nodes, it is not possible to predetermine the amount of time required for coupling/decoupling activities because how many units flowing through the nodes are variable. The remedy is to use the block flow variable $z_a,\ a \in \widetilde{\mathcal{A}}$ at each linkage arc a with the constraints below:

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

4.2.3 Platform Lengths

There is an upper bound on the number of units that can be coupled in serving a train trip due to limited platform lengths, which can be achieved by the following constraints on each train node:

$$\sum_{t \in \mathcal{T}} \sum_{p \in \mathcal{P}^t_j} l^t x_p \leq L_j := \min\{L_{\mathrm{dep}(j)}, L_{\mathrm{arr}(j)}\}, \quad \forall j \in \widetilde{\mathcal{N}}; \tag{2}$$

4.2.4 Type-type Compatibility

Coupling upper bound with respect to unit type is complicated by type-type compatibilities. In order to let these compatibilities be consistent with their varied coupling upper bounds, we introduce the concept of *train unit family*. Table 5 shows all possible combinations of unit types that can be coupled from the entire Southern Railway fleet based on different coupling upper bounds.

We refer to the types that can be coupled and share some forms of common coupling number upper bound as of the same *family* (denoted as $f \in \mathcal{F}$). The italics in the last column refer to those common bounds (except Family II). From the above table, it can be seen that there are eight unit families in the entire fleet, as

- Family I = {‘171/7’, ‘171/8’}
- Family II = {‘455/8’, ‘456/0’}
- Family III = {‘313/1’}
- Family IV = {‘460/0’}
- Family V = {‘442/1‘}
- Family VI = {‘377/1’, ‘377/2’, ‘377/3’, ‘377/4’}

Notice members in Family II actually do not share a common upper bound. Nevertheless, this can be remedied by adding extra constraints limiting the number of coupled 456/0 units to be no greater than 6. For each family, we set binary decision variables $y_j^f \in \{0,1\}, \forall f \in \mathcal{F}$ to indicate whether a train j is served by any units from family f. Then, incompatible types are excluded by the following constraints forcing only one family in $\mathcal{F}$ to serve a train:

$$\sum_{f\in\mathcal{F}} y_j^f = 1, \quad \forall j \in \widetilde{\mathcal{N}}. \tag{3}$$

Then the upper bound constraint can be written as

$$\sum_{f\in\mathcal{F}_1}\sum_{t\in f}\sum_{p\in\mathcal{P}_j^t} \nu^t x_p + \sum_{f\in\mathcal{F}_2}\sum_{t\in f}\sum_{p\in\mathcal{P}_j^t} x_p \le \sum_{f\in\mathcal{F}_1} U_j^f y_j^f + \sum_{f\in\mathcal{F}_2} U_j^f y_j^f, \quad \forall j \in \widetilde{\mathcal{N}}, \tag{4}$$

where ν^t is the number of cars for type t and U_j^f is the upper bound in number of *units* or *cars* family f can be used to serve train j. The reason why this parameter is devised as U_j^f (rather than U^f) is that a small proportion of this upper bound with respect to families can change on certain routes. For instance, Family I of diesel fleet British Rail Class ‘171/7’ and ‘171/8’ can generally be coupled up to a 2-unit block, however on the route from Ashford to Hastings, this upper bound is reduced to only 1 unit. Finally, $\mathcal{F}_1$ is the set of families sharing common upper bounds in car number, and $\mathcal{F}_2$ is the set of families sharing a common upper bound in unit number. The parameter $U_j^f,\ f \in \mathcal{F}_{1,2}$ will be set with a car number or unit number accordingly.

4.3 Path Formulation

If column generation is to be used for an MCNF problem, generally a path formulation would be preferred than an arc formulation. The LP-relaxations of the two formulations are equivalent (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 5 Unit type combinations and families of Southern Railway, UK

Related families	Types or combined types	Possible combinations (in # of units)	Upper bound property
I	‘171/7’ [2 car]	$1, 2$	max = *2 unit*, 4 car
I	‘171/8’ [4 car]	$1, 2$	max = *2 unit*, 8 car
I	(‘171/7’,‘171/8’)	$(1, 1)$	max = *2 unit*, 6 car
II	‘455/8’ [4 car]	$1, 2$	max = 2 unit, *8 car*
II	‘456/0’ [2 car]	$1, 2, 3$	max = 3 unit, *6 car*
II	(‘455/8’,‘456/0’)	$(1, 1), (1, 2)$	max = N/A, *8 car*
III	‘313/1’ [3 car]	1	max = *1 unit, 3 car*
IV	‘460/0’ [8 car]	1	max = *1 unit, 8 car*
V	‘442/1’ [5 car]	$1, 2$	max = *2 unit, 10 car*
VI	‘377/1,2,4’ [4 car]	$1, 2, 3$	max = 3 unit, *12 car*
VI	‘377/3’ [3 car]	$1, 2, 3, 4$	max = 4 unit, *12 car*
VI	(‘377/1,2,4’,‘377/3’)	$(1, 1), (1, 2), (1, 3), (2, 1)$	max = N/A, *12 car*

$$\begin{aligned}\min_{x,z} \quad & W_1 \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}^t} x_p + W_2 \sum_{t\in\mathcal{T}} \sum_{j\in\widetilde{\mathcal{N}}^t} \sum_{p\in\mathcal{P}_j^t} \mu_j^t x_p + W_3 \sum_{t\in\mathcal{T}} \sum_{a\in\mathcal{E}^t} \sum_{p\in\mathcal{P}_a^t} x_p \\ & - W_4 \sum_{t\in\mathcal{T}} \sum_{a\in\widetilde{\mathcal{A}}^t} \sum_{p\in\mathcal{P}_a^t} \gamma_a^t x_p + W_5 \sum_{j\in\widetilde{\mathcal{N}}} \left| \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_j^t} \kappa^t x_p - \overline{r}_j \right| \\ & - W_6 \sum_{t\in\mathcal{T}} \sum_{j\in\widetilde{\mathcal{N}}^t} \sum_{p\in\mathcal{P}_j^t} \pi_j^t x_p - W_7 \sum_{t\in\mathcal{T}} \sum_{a\in(\mathcal{A}^\circ)^t} \sum_{p\in\mathcal{P}_a^t} \pi_a^t x_p + W_8 \sum_{a\in\mathcal{A}} z_a \end{aligned} \tag{5}$$

subject to

$$\sum_{p\in\mathcal{P}^t} x_p \geq \underline{F}^t, \ \forall t \in \mathcal{T}; \tag{6a}$$

$$\sum_{p\in\mathcal{P}^t} x_p \leq \overline{F}^t, \ \forall t \in \mathcal{T}; \tag{6b}$$

$$\sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_j^t} \kappa^t x_p \geq \underline{r}_j, \quad \forall j \in \widetilde{\mathcal{N}}; \tag{7}$$

$$y_j^f \geq \frac{1}{U_j^f} \sum_{t\in f} \sum_{p\in\mathcal{P}_j^t} x_p, \quad \forall j \in \widetilde{\mathcal{N}}, \forall f \in \mathcal{F}_1; \tag{8a}$$

$$y_j^f \geq \frac{1}{U_j^f} \sum_{t\in f} \sum_{p\in\mathcal{P}_j^t} \nu^t x_p, \quad \forall j \in \widetilde{\mathcal{N}}, \forall f \in \mathcal{F}_2; \tag{8b}$$

$$\sum_{f\in\mathcal{F}} y_j^f = 1, \quad \forall j \in \widetilde{\mathcal{N}}; \tag{9}$$

$$\sum_{f\in\mathcal{F}_1} \sum_{t\in f} \sum_{p\in\mathcal{P}_j^t} \nu^t x_p + \sum_{f\in\mathcal{F}_2} \sum_{t\in f} \sum_{p\in\mathcal{P}_j^t} x_p \leq \sum_{f\in\mathcal{F}_1} U_j^f y_j^f + \sum_{f\in\mathcal{F}_2} U_j^f y_j^f, \quad \forall j \in \widetilde{\mathcal{N}}; \tag{10}$$

$$\sum_{p\in\mathcal{P}_j^t} x_p \leq 3, \quad \forall j \in \widetilde{\mathcal{N}}^t, \ t = \text{`456/0'}; \tag{11}$$

$$\sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_j^t} l^t x_p \leq L_j, \quad \forall j \in \widetilde{\mathcal{N}}; \tag{12}$$

$$z_a \geq \frac{1}{M_a} \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_a^t} x_p, \quad \forall a \in \widetilde{\mathcal{A}}; \tag{13}$$

$$\tau^C_{\mathrm{dep}(i)} \left(\sum_{a\in\delta_-(i)} z_a - 1 \right) + \tau^D_{\mathrm{arr}(j)} \left(\sum_{a\in\delta_+(j)} z_a - 1 \right) \leq e_{ij}, \quad \forall (i,j) \in \widetilde{\mathcal{A}}; \tag{14}$$

$$z_a + \frac{1}{|\mathcal{E}^\dagger(a)|} \sum_{a'\in\mathcal{E}^\dagger(a)} z_{a'} \leq 1, \quad \forall a \in \mathcal{E} : \mathcal{E}^\dagger(a) \neq \varnothing; \tag{15}$$

$$x_p \in \mathbb{Z}_+, \quad \forall p \in \mathcal{P}^t, \forall t \in \mathcal{T}, \tag{16a}$$

$$y_j^f \in \{0,1\}, \quad \forall j \in \widetilde{\mathcal{N}}, \ \forall f \in \mathcal{F}, \tag{16b}$$

$$z_a \in \{0,1\}, \quad \forall a \in \mathcal{A}. \tag{16c}$$

4.3.1 Objective Function

The objective (5) is formed by a linear combination of eight 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 $j \in \widetilde{\mathcal{N}}$ 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 tries to minimize the summed deviation between actual unit capacity and upper demand per train. The nonlinearity caused by absolute value can be eliminated by adding extra variables $d_j^+, d_j^- \in \mathbb{R}_+, \forall j \in \widetilde{\mathcal{N}}$ with constraints

$$\sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_j^t} \kappa^t x_p - \overline{r}_j = d_j^+ - d_j^-, \quad \forall j \in \widetilde{\mathcal{N}}, \tag{17}$$

and replacing $\left|\sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}_j^t} \kappa^t x_p - \overline{r}_j\right|$ in the objective by $d_j^+ + d_j^-$. The sixth term encourages type-train (route) preferences, where $\pi_j^t \geq 0$ is a preference weight for a type-train pair (t, j): the higher, the more suitable. The seventh term encourages sign-in/off arc preferences (usually related to home depots) for each type of units, where pi_a^t stands for a preference parameter for a type-sign-in/off arc pair (t, a): the higher, the more suitable. Notice if a certain type t is not suitable for a sign-in/off arc, then this arc will have not been not inserted into $\mathcal{G}^t$ in the first place. The eighth 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.

4.3.2 Constraints

Constraints (6) ensure the deployed number of units for each type is within a lower and an upper bound. Constraints (7) guarantee satisfaction of a minimum level of capacity demand for each passenger train. Constraints (8) calculate each family-train indicator variable, and constraints (9) allow only one family to serve a train. Notice (8) and (9) together will ensure that $y_j^f = \begin{cases} 1, \sum_{t\in f} \sum_{p\in\mathcal{P}_j^t} x_p > 0 \\ 0, \sum_{t\in f} \sum_{p\in\mathcal{P}_j^t} x_p = 0 \end{cases}, \forall j \in \widetilde{\mathcal{N}}, \forall f \in \mathcal{F}$. Constraints (10) forbid the number of coupled units exceeding certain upper bounds respect to trains and families. Constraints (11) are to modify (10) for the exceptional case of type 456/0. Constraints (12) are maximum platform length constraints. Constraints (13) calculate blockflow variables. Notice as the sum of blockflow variables is minimized in the objective, the relation $z_a = \begin{cases} 1, \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}^t} x_p > 0 \\ 0, \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}^t} x_p = 0 \end{cases}, \forall a \in \mathcal{A}$ is guaranteed. Constraints (14) are to ensure time allowance validity for coupling/decoupling time. Constraints (15) are to exclude all conflicts among empty-running trains. Finally (16) forces the variables to be integral or binary.

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 $u_1, u_2, \cdots, u_n \in \mathcal{U}^\sigma$ and $v_1, v_2, \cdots, v_n \in \mathcal{V}^\sigma$ 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 $|\mathcal{U}^\sigma| = |\mathcal{V}^\sigma| = n$. Sign-in/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-in/off arcs regard this station as the source/sink node. For simplicity reason, we will omit the superscript σ in later part of this section as everything is clearly based on a 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 an arrival train and a departure unit only belongs to a 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 time validity rules regarding shunting/coupling/decoupling times will be considered respectively in other ways. It is important to note that an *appropriate* matching between a pair of arrival and departure units $(u, v),\ u \in \mathcal{U}, v \in \mathcal{V}$ will correspond to one and only one linkage arc $(T(u), T(v)) \in \widetilde{\mathcal{A}}$ 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 significant. Since the first phase has not provided anything on unit permutation for coupled trains, the position attributes have to be decided in the second phase. Let $p \in \mathcal{P}_u, q \in \mathcal{Q}_v$ 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 has been decided a priori ($|\mathcal{P}_u| = 1$ or $|\mathcal{Q}_v| = 1$); for a coupled single-type (homogenous) train, unit positions can be pre-assigned based on the first phase result in an arbitrary order without losing any generality, leading to $|\mathcal{P}_u| = 1$ or $|\mathcal{Q}_v| = 1$ as well; for a coupled multi-type (heterogenous) train where different permutations can have essentially different results, a decision has to be made from $p \in \mathcal{P}_u, q \in \mathcal{Q}_v, |\mathcal{P}_u| \geq 1, |\mathcal{Q}_v| \geq 1$. Therefore, the above matching has to be further modified within a *positioned* pair of (u, p) and (v, q) such that a plan $(u, p, v, q), u \in \mathcal{U}, p \in \mathcal{P}_u, v \in \mathcal{V}, q \in \mathcal{Q}_v$ 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 $\tau(u_1) \prec \tau(u_2) \prec \tau(v_3) = \tau(v_4)$ 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 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. The cases in coupled multi-type trains will be more complicated.

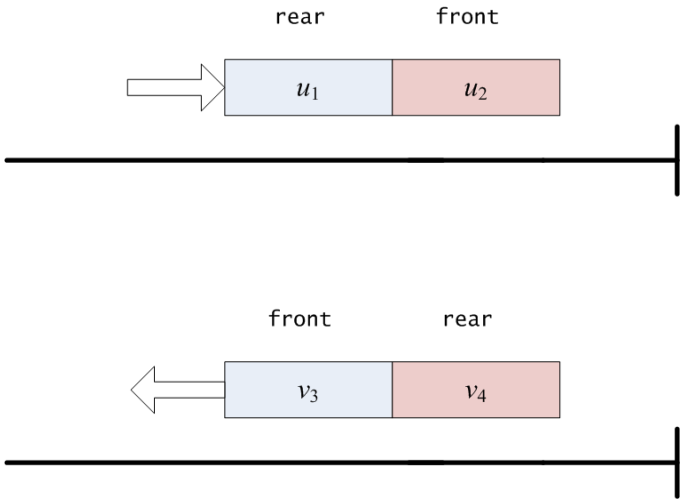

Fig. 3 A matching in a dead-end platform scenario

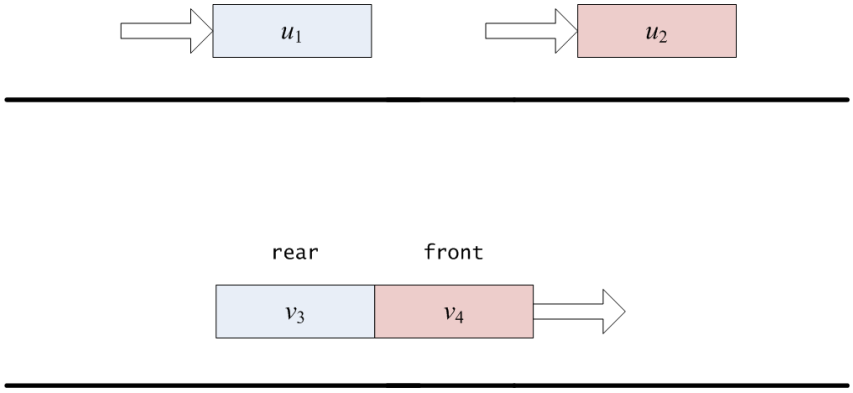

Fig. 4 A matching with coupling in a FIFO scenario

5.1.3 Parking Berths

As discussed in Section 2.1.3, 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 $b \in \mathcal{B} := \{0\} \cup \mathcal{S} \cup \mathcal{D}$ where 0 refers to a dummy parking berth, $\mathcal{S}$ the siding set and $\mathcal{D}$ the depot set.

For a positioned pair of arrival and departure units $(u, p, v, q), u \in \mathcal{U}, p \in \mathcal{P}_u, v \in \mathcal{V}, q \in \mathcal{Q}_v$, the possible choice for parking berth types (dummy, siding or depot) is dependent on the relevant time gap length of its dominant linkage arc. 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 $\mathcal{B}(u, p, v, q)$.

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 much slack time available for resolving any conflicts that might arise.

Here we define the parking method $m \in \mathcal{M}(u, p, v, q, b)$ for a pair of positioned units (u, p, v, q) with its parking berth $b \in \mathcal{B}(u, p, v, q)$ 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 $b \in \{0\} \cup \mathcal{D}$, 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 of 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 $\mathcal{M}(u, p, v, q, b)$ 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 $u \in \mathcal{U}$ a position $p \in \mathcal{P}_u$ and assign it to a departure unit $v \in \mathcal{V}$ with position $q \in \mathcal{Q}_v$, via a parking berth $b \in \mathcal{B}(u, p, v, q)$ and with a parking method $m \in \mathcal{M}(u, p, v, q, b)$ such that

1. *Each matching pair (u, v) implies a dominant linkage arc relation $(T(u), T(v)) \in \widetilde{\mathcal{A}}$;*
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 (already satisfied in the first phase);*
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 $(u, p, v, q, b, m),\ u \in \mathcal{U}, v \in \mathcal{V}, (T(u), T(v)) \in \widetilde{\mathcal{A}}, p \in \mathcal{P}_u, q \in \mathcal{Q}_v, b \in \mathcal{B}(u, p, v, q), m \in \mathcal{M}(u, p, v, q, b)$ representing a *self-consistent* shunting plan. For convenience, a shunting plan (u, p, v, q, b, m) can also be written as $\pi \in \Pi$ when its details can be omitted, where we denote the set of all possible shunting plans at a station as Π. The set of shunting plans all involving a specific arrival unit u can be written as Π_u, the same for Π_v for a specific departure unit v. 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: $\mathcal{W}_u(u, p, v, q, b, m) := [\tau(u), \lambda(u)]$;
- 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: $\mathcal{W}_v(u, p, v, q, b, m) := [\chi(v), \tau(v)]$;
- 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: $\mathcal{W}_b(u, p, v, q, b, m) := [\chi(b), \lambda(b)]$.

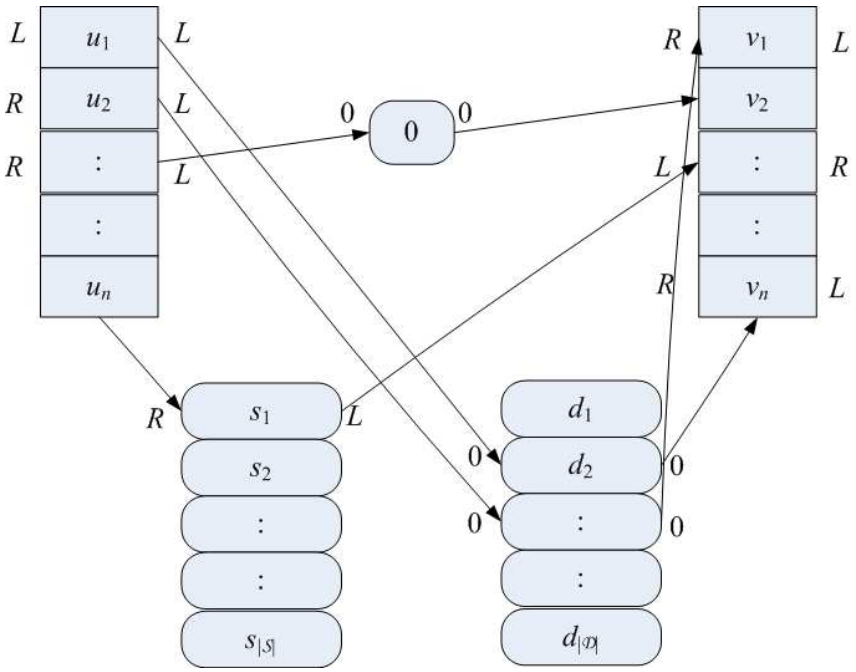

Fig. 5 An illustration of shunting plans

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-processed as input data for the second phase. Hence, we use

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

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

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

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 (5) 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 linkage arcs pertaining to station σ as $\widetilde{\mathcal{A}}^{\sigma}$. As for each valid shunting plan (u, p, v, q, b, m), $(T(u), T(v))$ corresponds to a unique dominant linkage arc $a \in \widetilde{\mathcal{A}}$, we thus denote the arc corresponding to shunting plan π as $a(\pi)$, i.e. if $\pi = (u, p, v, q, b, m)$, then $a(\pi) = (T(u), T(v)) \in \widetilde{\mathcal{A}}^{\sigma}$. Then, after the path variables $x_p, p \in \mathcal{P}^t, t \in \mathcal{T}$ from the first phase results have been transformed into arc variables by $x_a = \sum_{p \in \mathcal{P}_a} x_p, \forall a \in \mathcal{A}^t, t \in \mathcal{T}$, 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 preference relations (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 $y_b^t, b \in \mathcal{S}, t \in \mathcal{T}$ as a binary variable indicating whether at least one unit of type t is parked at siding b, and try to minimize $\sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{S}} y_b^t$.

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

Thus, the second phase model:

$$\min_{x,y,z} \quad W_1 \sum_{\pi\in\Pi} c_\pi x_\pi + W_2 \sum_{t\in\mathcal{T}}\sum_{b\in\mathcal{S}} y_b^t + W_3 \sum_{a\in\widetilde{\mathcal{A}}^\sigma} z_a \tag{18}$$

subject to

$$\sum_{\pi\in\Pi_u} x_\pi = 1, \quad \forall u \in \mathcal{U}; \tag{19}$$

$$\sum_{\pi\in\Pi_v} x_\pi = 1, \quad \forall v \in \mathcal{V}; \tag{20}$$

$$x_\pi \le y_{b(\pi)}^{t(\pi)}, \quad \forall \pi \in \Pi : b(\pi) \in \mathcal{S}; \tag{21}$$

$$x_\pi \le z_{a(\pi)}, \quad \forall \pi \in \Pi; \tag{22}$$

$$\tau^D_{\mathrm{arr}(i)} \left(\sum_{a\in\delta^\sigma_+(i)} z_a - 1 \right) + \tau^C_{\mathrm{dep}(j)} \left(\sum_{a\in\delta^\sigma_-(i)} z_a - 1 \right) \le e_{ij}, \quad \forall (i,j) \in \widetilde{\mathcal{A}}^\sigma; \tag{23}$$

$$x_\pi + \frac{1}{|\Pi^\dagger(\pi)|} \sum_{\pi'\in\Pi^\dagger(\pi)} x_{\pi'} \le 1, \quad \forall \pi \in \Pi : \Pi^\dagger(\pi) \neq \varnothing; \tag{24}$$

$$l_{t(\pi)} x_\pi + \sum_{\pi' : \mathcal{W}_b(\pi')\cap\mathcal{W}_b(\pi)\neq\varnothing} l_{t(\pi')} x_{\pi'} \le L_{b(\pi)}, \quad \forall \pi \in \Pi : b(\pi) \in \mathcal{S}\cup\mathcal{D}; \tag{25}$$

$$x_\pi \in \{0,1\}, \quad \forall \pi \in \Pi; \tag{26a}$$

$$y_b^t \in \mathbb{R}_+, \quad \forall b \in \mathcal{S}, \forall t \in \mathcal{T}; \tag{26b}$$

$$z_a \in \mathbb{R}_+, \quad \forall a \in \widetilde{\mathcal{A}}^\sigma. \tag{26c}$$

The first term in the objective (18) minimizes the cost of the set 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 those (W_2 and W_3) for the other terms.

Constraints (19)–(20) 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 (21) 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 strong formulation with many constraints ($= |\Pi|_{b(\pi)\in\mathcal{S}}$) rather than a weak formulation with less constraints, e.g. a multiple variable lower bound (MVLB) (Ciriani et al (2003)) like

$$\frac{1}{|\Pi_b^t|}\sum_{\pi\in\Pi_b^t} x_\pi \le y_b^t, \quad \forall b\in\mathcal{S}, \forall t\in\mathcal{T}, \tag{27}$$

where Π_b^t is the set of shunting plans implying a type t to be shunted to a berth b. The strong formulation (21) will give a much tighter LP-relaxation for y_b^t than (27)—in fact here it is so tight, combined with the second term in the objective, that the y_b^t solution given in an optimal solution automatically equals to a desired binary value. Therefore, even only setting $y_b^t \in \mathbb{R}_+, \forall b\in\mathcal{S}, \forall t\in\mathcal{T}$ is sufficient. This has greatly eased the solution process for the MILP. The side effect from excessive number of constraints in (21) can be neutralized if a column generation approach is used.

Constraints (22) calculate the blockflow variables for each dominant arc pertaining to station σ: $a\in\mathcal{A}^\sigma$. Similar as in (21), we use a strong formulation rather than an MVLB formulation like $\frac{1}{|\Pi_a|}\sum_{\pi\in\Pi_a} x_\pi \le z_a, \forall a\in\widetilde{\mathcal{A}}^\sigma$, and setting $z_a\in\mathbb{R}_+, \forall a\in\widetilde{\mathcal{A}}^\sigma$ is also sufficient.

Constraints (23) guarantee the time consumed by coupling/decoupling activities will not break the linkage time allowance validity, where $\tau^D_{\mathrm{arr}(i)}$ and $\tau^C_{\mathrm{dep}(j)}$ represent single decoupling time at the arrival platform of train i and single coupling time at the departure platform of train j respectively.

Constraints (24) prohibit conflicts between each pair of shunting plans. Note that a generic conflict (or crossing) exclusion formulation such as

$$x_\pi + x_{\pi'} \le 1, \quad \forall(\pi,\pi')\in\mathcal{C} \tag{28}$$

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, to take 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 (24). Compared with (28), the number of constraints has been significantly reduced, as usually $|\mathcal{C}| \gg |\Pi|$. Notice the number of constraints will be further reduced if a column generation approach is used.

Constraints (25) 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.

Finally (26) gives variable domains.

6 Solution Approaches

6.1 A Hybrid Approach for the First Phase Integer MNCF Model

The model given in Section 4.3 corresponds to an extensive (path) formulation, which is suitable for a branch-and-price approach. However, exact methods would be incapable of handling very large problem instances (e.g. with over 2400 trains in Southern Railway) with complex side constraints. Here we propose a Size Limited Iterative Method (SLIM) based on hybridization of an exact method (a branch-and-price ILP solver) and a heuristic framework.

6.1.1 A Branch-and-price Approach with Pre/Dynamic Column Generation

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 have to be generated according to some methods as subproblems. There are usually two ways for generating new

columns to be added to the restricted master problem. The first one is to generate candidates dynamically. Examples of this type include a shortest path subproblem for generic MCNF problems or a knapsack subproblem for the generalized assignment problem. Alternatively, it is possible to perform a pre-generation beforehand trying to extract the essence of all possible candidates into a size-reduced pot, and then carry on subsequent column generation process only based on this pot. Examples of this type include (Kwan and Kwan (2007)) for large-scale crew scheduling and Hennig et al (2012) for maritime transport routing.

If a dynamic-generation approach is used, theoretically the solution will be guaranteed to converge to a global optimum. However, this approach has two major disadvantages, both due to the dynamic nature of how new columns are generated in subproblems. One is the difficulty in satisfying sophisticated rules imposed on each column; the other one, arising in integer programs, is the fact that dynamically generated columns may have some properties inconsistent with the current status of a branch node (Barnhart et al (1998)). These lead to the need for designing very complicated subproblems or branching strategies to exclude invalid candidates. Constrained shortest path subproblems (see a survey by Irnich and Desaulniers (2005)) for crew scheduling is a well-known example. On the other hand, as all columns ever needed have been explicitly pre-generated, the above two disadvantages from dynamic-generation are unlikely to occur in a pre-generation approach, making it more preferable for solving very large-scale complex problems. The only 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.

Our preliminary tests based on branch-and-price with dynamic-generation have shown that this solution approach would not be practical for timetables with more than 1000 trains, even though the subproblem is a standard shortest path in a DAG that can be solved in linear time with $O(|\mathcal{N}| + |\mathcal{A}|)$ by topological sorting. As the real-world instance we are dealing with has a timetable of over 2400 trains, the dynamic-generation approach would not be feasible.

As for the branching strategy, experience from similar problems suggests it is usually an effective choice to branch on the corresponding compact formulation while working on the extensive formulation (Villeneuve et al (2003)), including most integer MCNF problems (Alvelos (2005)). The arc variables to be branched at can also be prioritized according to their LP relaxation solution value, thus speeding up tree searching.

6.1.2 A Size Limited Iterative Method (SLIM)

To further enhance the capability of exact ILP solvers using the above branch-and-price column generation framework for solving very large problem instances, we further propose a hybrid 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 ILP 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 ILP solver. This process will be repeated until no significant improvement can be obtained. The problem instance compaction retains all the train trips, i.e. the method does not subdivide the problem instance into small separate subproblems (by solving subproblems separately, optimality is likely to be lost).

For our first phase integer MCNF model, 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.

6.2 A Solution 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 branch-and-price would be able to handle it.

Pre-generation We use a pre-generation approach with branch-and-price for the second phase. The complex rules imposed on individual shunting plans make it almost impossible to generate such candidate plans dynamically at the pricing stage. All possible candidate shunting plans forming the set of Π will be pre-generated as a pot. Another major step beforehand is to construct the conflict sets $\Pi^{\dagger}(\pi)$ for each shunting plan π. For efficiency reasons, an initial testing phase is used where only a set of columns corresponding to the first phase results (all the shunting plans with zero costs c_π) is provided for the restricted master problem (RMP). If those columns also 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 branch-and-price 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)).

Branching Strategies A branching system with mixed (two) strategies is to be used: (i) branching on activated dominant arcs $a \in \widetilde{\mathcal{A}}^{\sigma}$. Activated dominant arcs refer to those dominating the shunting plans that have been priced out into the RMP, i.e. an arc $(T(u), T(v))$ corresponding to a plan (u, p, v, q, b, m)); (ii) branching on shunting plans $\pi \in \Pi$. The former is usually used at early stages of the branching process and the later is more useful at later stages.

7 Testing and Conclusions

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 a recent year, 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 10 different types of train unit 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 preliminary results have been obtained, which will be presented at the conference, they still have to be carefully analyzed with Southern Railway. More comprehensive results therefore will be reported in a future paper.

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.

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 parallel computation, and further refinement of the proposed models taking into account problem scenarios from many more train companies.

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

Bodin L, Golden B, Assad A, Ball M (1983) Routing and scheduling of vehicles and crews - the state of the art. Computers and Opertions Research, vol 10, pp 63–212

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

Cardonha C, Borndörfer R (2009) A set partitioning approach to shunting. Electronic Notes in Discrete Mathematics 35:359–364

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

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

Desrosiers J, Dumas Y, Solomon MM, Soumis F (1995) Time constrained routing and scheduling. In: Ball M, Magnanti T, Monma C, Nemhauser G (eds) Handbooks in Operations Research and Management Science, vol 8: Network Routing, Elsevier, chap 2, pp 35–139

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, 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

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(3):764–774

Holmberg K, Yuan D (2003) A multicommodity network-flow problem with side constraints on paths solved by column generation. INFORMS J on Computing 15:42–57

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Column Generation, Springer US, pp 33–65

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. In: Leung JY (ed) Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, chap 51, pp 51(1–20)

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

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

Vanderbeck F (2000) On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research 48(1):111–128

Vanderbeck F, Wolsey LA (1996) An exact algorithm for IP Column generation. Operations Research Letters 19:151–159

Villeneuve D, Desrosiers J, Lübbecke ME, Soumis F (2003) On compact formulations for integer programs solved by column generation. In: Les Cahiers du GERAD G-2003-06, HEC, pp 375–388

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