CASPT 2018

Resolution of Station Level Constraints in Train Unit Scheduling

Li Lei[1] · **Raymond S.K. Kwan**[1] ·
Zhiyuan Lin[2] · **Pedro J Copado-Mendez**[1]

Abstract Train unit scheduling assigns vehicles to cover all trips of a fixed timetable satisfying seat demands at minimum operational costs. Solved as a network flow problem, train stations are simplified to single points where certain station operation details are not considered. For instance, when a trip is covered by coupled train units, the coupling order is left undetermined, and infeasibility may arise because of track and platform layouts, movement directions and timings. This paper investigates the resolution of such station level constraints for a multi-commodity flow based approach; in particular, we shall focus on feasible coupling order as an example. A hybrid-iterative approach is proposed for coupling order assignment with refinement of train unit scheduling by considering conflict-free linkage implementation at the station level.

Keywords Train unit scheduling · Coupling order · Station Constraints · Resolution

1 Introduction

A physical *train unit* has fixed carriages and integral engine(s) that can move in either directions. Train units are commonly used in railway passenger transport replacing traditional locomotive trains. They are classified into many *types* with characteristics such as power source, number of carriages, and number of seats. In the UK, there are many franchised train operators, such as Greater Anglia, Virgin Trains, Northern, First Great Eastern, and TransPennine Express. Each operator manages a certain number of train units constituting a *train unit fleet*. A timetable describes a set of *logical trips* with fixed routes (origin, destination and intermediate stations) and timings (clock time of arrival

[1]School of Computing, University of Leeds, Leeds, UK
E-mail: scll, r.s.kwan, p.j.copado-mendez@leeds.ac.uk
[2]Alliance Manchester Business School, University of Manchester Manchester, UK
E-mail: zhiyuan.lin@manchester.ac.uk

and departure, and duration time of each trip). Physical units are assigned to serve logical trips under the condition of passenger demands, i.e. a trip can be served by a single unit or multiple units defined as *unit blocks* in this research. To satisfy different seat demands, unit blocks may involve *coupling operation* to form longer unit blocks or *decoupling operation* to be decomposed into shorter unit blocks. That gives rise to the issues of unit positions in trips described as the *coupling order problem.* If a trip is served by multiple units of different types, the coupling order is significant, in particular when the unit block serving this trip involves coupling or decoupling operations. Usually, coupling and decoupling operations are executed at corresponding platforms at certain stations. As the movement of unit blocks are strictly restricted by tracks, station layouts and how they connect with other stations are key factors giving rise to the coupling order issues.

Post network flow scheduling conflict resolution is vital to avoid operational blockages at the station level and to determine feasible coupling order, which can bring the network scheduling studied by Lin and Kwan (2014) to be more operable since they consider stations as simple points without structural details. The process of coordinating with the *RS-Opt* (network-scheduling solver developed by Lin and Kwan (2016)) is called *RS-Opt-PT* in this paper. *RS-Opt* optimizes train unit assignment at the network flow level by a branch-and-price approach and tentatively allocates unit blocks to trips and connections among assigned unit blocks. Its solution is incomplete since the coupling order is not determined and the feasibility of the tentative linkages among trips has not been verified.

Stations are the places to implement linkages. Coupling order is normally formed at the corresponding station, however, its influence is not isolated within that station but can be propagated to the entire network because of running trips. If a blockage caused by coupling order arises at a certain station, more operations (for instance reordering shunting) are needed to fix the blockage. This increases operational cost and may disturb the operation of an entire railway network. Hence, coupling order assignment must be considered not only at the station level but also at the network level. In this paper, a hybrid-iterative approach is proposed. In the main loop, a new scheduling solution will be sought if *RS-Opt-PT* has encountered unresolvable coupling order conflicts. *RS-Opt-PT* feeds the critical station constraints detected back to *RS-Opt* to eliminate inoperable factors and make the train unit scheduling solution more complete and operable.

2 Literature review

Huisman et al (2005) survey on the operational research in passenger railway transportation focusing on European cases, where the train unit planning is classified as two levels. The central level covers the entire network operations such as timetabling, train unit scheduling, circulation and etc. The local level focuses on the local-size-impact operations including shunting, platform as-

signment, and routing of train units at station etc. The train unit scheduling problem can be addressed as two levels (Lin and Kwan (2014) and Kwan et al (2017)). The network flow level aims to solve train sequencing and fleet assignment by treating stations as simple points temporarily ignoring station layouts. The coupling and decoupling activities are a distinct feature to satisfy passenger demands at the network flow level. The station level copes with finalizing the operational plan and any train unit shunting within stations/depots.
The network flow level of train unit scheduling finds paths to assign unit type and quantities to each timetabled trip. Once an assignment has been found from the simplified model temporarily ignoring station-level constraints, there are two operational aspects open to be further determined. The first aspect is the unit coupling order in multi-type trips, which has no impact on the network flow but must be finalized to prevent blockages at stations. The second aspect is the linkage implication restricted by station layouts. This research focuses on these two points to make the train unit scheduling solution more operable which has not been scrutinized carefully in other researches.

2.1 Train unit scheduling at the network flow level

Cacchiani et al (2010) consider basic constraints of passenger demands, coupling upper bounds and additional constraints of maintenance and overnight balance. Two inter-convertible ILP formulations (arc formulation and path formulation) are established based on a directed acyclic graph described in Cacchiani et al (2013a). A LP-based heuristic is designed to solve the path model but it is reported as crucial in finding feasible solutions in many tested instances since the problem is regarded as very difficult. Another heuristic method based on Lagrangian relaxation developed by Cacchiani et al (2013b) to increase the performance.
A more comprehensive real-world constraint model to describe the network flow problem in the UK is described by Lin and Kwan (2013) and a hybridized iterative method is proposed by Lin and Kwan (2016) and Copado-Mendez et al (2017) to solve larger problem instances. In practice, most train operators in the UK consider different levels of capacity provisions, such as for peak and off-peak. To achieve this, Lin et al (2017) study the train unit scheduling with bi-level capacity requirements and propose a new integer multi-commodity flow model guided by historic capacity provisions and passenger count surveys supported by computational experiments on real-world data showing the effectiveness of this methodology.

2.2 Train unit shunting at the station level

A timetable defines arrival/departure times and platforms for each trip, and a train unit schedule assigns unit blocks to serve each trip. Shunting schedule guarantees the assigned unit blocks are operable at the fixed timings and

platforms at stations.
Tomii et al (1999) consider the station shunting problems as a resource-constrained project scheduling problem divided into two sub-problems: resource allocation and shunting time decision. A two-stage-search algorithm is proposed to solve this problem combining with probabilistic local search and PERT (programming evaluation and review technique).
Freling et al (2005) and Kroon et al (2008) consider the train unit shunting problem at a station as two sub-problems: matching problem and parking problem. Freling et al (2005) describe a set-partitioning integer linear programming model to solve these two sub-problems separately, which prevents capacity overflow and unit blockage at all sidings minimizing the number of coupling/decoupling operations. This model is solved by root-only column generation combined with a branch-and-bound strategy. Since the matching and parking problems are connected to each other, Kroon et al (2008) propose an integrated approach with four models considering both dead-end and through types of siding, thus, the global optimality is guaranteed. The idea of virtual tracks is introduced to reduce the problem size. Computational experiments on two stations of the NSR network have been done.

2.3 Bridging between network flow and station plan

The train unit scheduling at the network flow level and station level are not isolated but connected by running trips on the railway network. Lin and Kwan (2014) propose models for both the network flow level and the station level, and the two levels can communicate through arc variables. Conceptually, the re-matched linkages at the station-level model are encouraged if they are also chosen by the network flow level. Ideally, the good quality and operable solution at both levels are achieved through this communication. Based on their work, Lei et al (2017) analyze the potential station shunting issues caused by a solution at the network flow level without determination of coupling order. In addition, a specific branch-and-cut algorithm of connecting flow level and depot shunting together is proposed by Haahr and Lusby (2017) with simplified assumption that tracks are all dead-end.

3 Problem description

Train unit scheduling at the network flow level concerns train unit assignment to satisfy all the trips fixed by a timetable with respect to a series of constraints such as passenger demands, coupling and decoupling, and compatibility etc. This problem can be modeled as an integer multi-commodity flow problem based on a DAG (Directed Acyclic Graph) where the nodes and arcs represent trips and possible connections among trips respectively. In Lin and Kwan (2016), limited train units are assigned to fixed trips in which the matches between arrival and departure unit blocks have been tentatively decided but the coupling order in trips served by coupled units and the feasibility of the matching between arrivals and departures considering station level details are left open to be further determined. This may cause blockages while implementing the network-flow-level solution at the station level because of factors such as railway connections among stations, directions of approaching to and leaving from a station, platform type, shunting operations etc.

3.1 Resolution constraints

The basic hard constraints to get a conflict-free schedule at the station level are:
(1) If some extra station shunting movements are added to some unit blocks, there must be sufficient time available.
(2) There may be flexibility in the timings of re-platforming, depot/siding shunting, and empty running unit blocks. Their movements must not cause any blockage.
(3) Station layouts and how stations connect to each other.
(4) If a linkage is to be feasible, the unit blocks linked have to be at the right time and at the right platform.
(5) The unit block accumulation at each platform is restricted by the platform usable length.
Some factors helpful for determining coupling order and easing station-level conflicts:
(1) Some platform types may force upon some specific coupling orders so as to make feasible certain coupling and decoupling operations.
(2) Interchangeability of the same type of units.
(3) Introducing extra essential shunting movements within time allowance.
(4) Adjusting the unit blocks with flexible timings related to re-platforming, siding/depot shunting, empty running etc.
(5) Capacity over-provision on some trips.

3.2 Inoperable scenarios in a given network-flow-level schedule

The linkages of matching the arrival to departure unit blocks are tentatively given by the network flow level. The linkages imply a set of shunting operations at stations to transit arrivals to departures, such as coupling, decoupling, station empty running for re-platforming or depot/siding shunting etc. Linkage implementation is based on station infrastructure which has been ignored by the network flow level, such that a given network-flow-level schedule may be inoperable during the implementation process at the station level. The blockage is sometimes known as *crossing* in (Kroon et al (2008) and Freling et al (2005)).

Crossing matching Rolling stocks can only run on rail tracks, which is a distinct feature from road (e.g. bus) vehicles. The movements and operations

Table 1: Example of 4 trips at same platform

Trip No.	Origin-Destination	Departure time	Arrival time
T1	A-B	07:00	10:30
T2	C-B	09:00	10:40
T3	B-C	10:50	12:30
T4	B-D	10:55	12:00

of transition from arrival to departure unit blocks are restricted by tracks. To illustrate how crossing matching invalidate a schedule derived solely based on timetable and fleet information (i.e. without infrastructure information on track layouts etc.), consider a simple example. Table 1 is the timetable of 4 trips and suppose all the four trips have been assigned to the same through platform of station B. Suppose Unit I can serve T1, T3 and T4, while Unit II can serve T2, T3 and T4. There may be two possible solutions: Solution (1) Unit I (T1 $\rightarrow$ T3), Unit II (T2 $\rightarrow$ T4); Solution (2) Unit I (T1 $\rightarrow$ T4), Unit II (T2 $\rightarrow$ T3). The FIFO (first in first out) principle is encouraged at the network flow level, hence, the most likely schedule given by the network flow level is Solution (1). At the station platform level, the positions of unit blocks are presented in Fig. 1. It is inoperable as this assignment causes a blockage

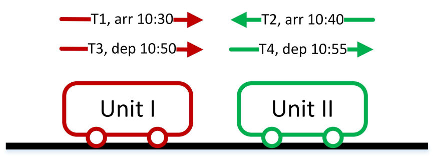


Fig. 1: Cross matching of network flow level

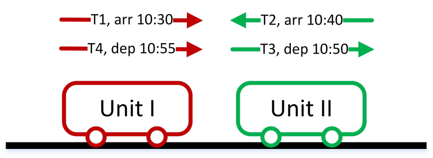


Fig. 2: Feasible matching at station level

between unit I and unit II. There is an alternative feasible Solution (2), shown in Fig. 2. This example illustrates how train directions and station layouts can seriously affect the feasibility of a network-flow-level schedule.

Unit block accumulation on platform At the network flow level, platform length constraints are implicitly considered via coupling car upper bound for every trip. However, the unit block accumulation with time at each platform is not considered. This may result in an invalid network-flow-level solution.

Coupling order reversal en-route of multi-unit blocks Usually, no coupling or decoupling operation happens at intermediate stations, but one important feature of coupling order captured while considering moving directions is front-rear-reversal. If the arrival and departure directions of a coupled unit block are opposite at some platforms of intermediate stations, the front and rear of the unit block of the corresponding trip must be reversed; if they are the same, its coupling order keeps the same. Fig. 3 illustrates three scenarios of coupling order reversal en-route of a coupled unit block based on platform structure and trip directions. i.e. the coupling order of a trip may be mutative during its journey.

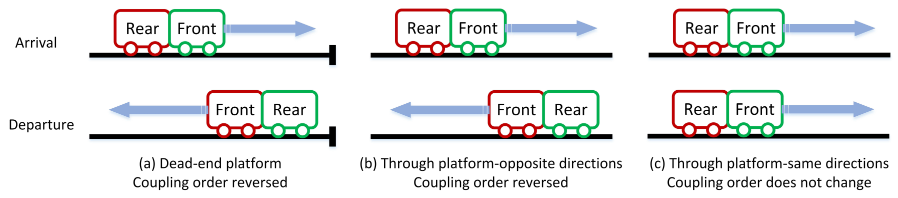

Arrival
Departure
Rear
Front
Front
Rear
(a) Dead-end platform
Coupling order reversed
(b) Through platform-opposite directions
Coupling order reversed
(c) Through platform-same directions
Coupling order does not change

Fig. 3: Coupling order scenarios en-route

Coupling and decoupling operations The aim of coupling and decoupling operations is to redistribute units to serve fixed trips and balance unit resources on the network. Units of compatible types can be coupled together to form longer unit blocks and multi-unit blocks can be decoupled into shorter unit blocks. For a multi-type unit block, its coupling order may be important as some coupling/decoupling operations can only be performed in a certain order. Improper coupling order causes blockages at stations which may impact the entire network operating. To avoid blockages while coupling/decoupling operations is a big step to bring the schedule more operable.

Coupling order propagation on the network Railway network contains stations and tracks connecting stations together. Running trips concatenate the operations of generating and terminating trips at stations together based on a rough time sequence. Thus, the operational decisions especially coupling order at stations would be passed around the network via running trips. For instance, station A proposes a certain coupling order for a unit block based on its current shunting environment, however, it may be invalid to other shunting environment at other stations or even at station A since the shunting environment is dynamically changing by time. Hence, extra operations must be adopted to fix this invalidity. Generally speaking, if the network-flow-level solution contains the structure in Fig. 4, the coupling order propagation on the network must be considered carefully.

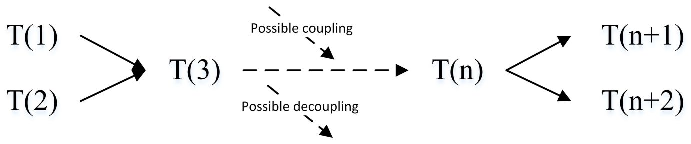


Fig. 4: A critical unit diagram structure

4 A hybrid-iterative approach

Train unit scheduling at the network flow level aims at assigning limited train units to cover trips with satisfied passenger demands. The results from the *RS-Opt* contains incomplete unit diagrams with unfixed coupling order and unverified linkages. To illustrate coupling order clearly, let us define a

Table 2: Notations used in this approach

G	directed acyclic graph
G^*	directed acyclic graph refined from G
y_a	arc-selection binary variables
τ	unit type
x_a^τ	arc-type-flow integer variables
$x_a^{\tau q}$	binary variables indicating whether the x_a^τ is positive or not
0	source of G^*
∞	sink of G^*
S_c	station set related to conflict c, $c \in CP$ or $c \in CN$
H	platform set
N	trip node set
N_h^a/N_h^d	time sorted arrival/departure node list of platform h, $h \in H$
N_c	node set related to conflict c, $c \in CP$ or $c \in CN$
A	arc set of G^*
A_0	sign-on arc set of A, $A_0 \subset A$
A_∞	sign-off arc set of A, $A_\infty \subset A$
A_N	trip-to-trip arc set of A, $A_N \cup A_0 \cup A_\infty = A$
A_+^i	entering arc set of trip i, $i \in N$
A_-^i	leaving arc set of trip i, $i \in N$
A_d^i	same platform trip-to-trip arc set of trip node i, $i \in N$
A_r^i	re-platforming arc set of trip node i, $i \in N$
A^*	arc set supposed to be implemented at the stations in S_c
A_c^+	arc set causing conflict c, $c \in CP$ or $c \in CN$
A_c^-	complementary arc set of A_c^+ out of A^*
Γ	timetable input
θ	geography input
$c_e(a), c_e'(a)$	additional cost of arc a, $a \in A_N$
α, β, γ	weight for different types of additional cost
ts_a	slack time of arc a, $a \in A_N$
tr_a	time consumed by station shunting operation of arc a, $a \in A_N$
tp_h	operational time consumed at platform h, $h \in H$
U	set of dummy trip
$\tilde{u}_{(i,j)}$	dummy trip interpreted by a re-platforming arc (i,j), $(i,j) \in A_N$
$ta_i/ta_{\tilde{u}}$	arrival time of trip i/dummy trip $\tilde{u}$, $i \in N$, $\tilde{u} \in U$
$td_i/td_{\tilde{u}}$	departure time of trip j/dummy trip $\tilde{u}$, $i \in N$, $\tilde{u} \in U$
s, s', s''	solution at different stages
CP	conflict list at the platform stage
CN	conflict list at the network stage

unit-composition sequence $[u_1...u_m]$ as the fixed coupling order, and a unit-composition multi-set $< u_1...u_m >$ as unfixed coupling order, and a reverse function $rev(couplingOrder, number)$ to express the coupling order reversal en-route. Note that odd *number* gets reversed coupling order and even *number*

gets the same coupling order. A hybrid-iterative approach of finalizing implementable linkages and assigning coupling order with refinement of train unit scheduling at the station level is proposed.
This approach contains four main parts: DAG refinement, platform-based coupling order assignment, network-based coupling order assignment and conflicts resolving stage. The DAG refinement is based on introducing station-level configuration such as layouts to get a smaller graph which is helpful to reduce the computational time of the network-level *RS-Opt*. The platform-based coupling order assignment tentatively determine the coupling order for some trips according to critical station operations for example coupling, decoupling, re-platforming etc. The network-based coupling order assignment finally decide the coupling order considering the structure of the railway network. The conflicts are collected during these two assignment processes, which are going to be added to *RS-Opt* as dynamic constraints. Table 2 describes the notations used in this approach.

Algorithm 1 A hybrid-iterative approach for train unit scheduling

```
Require: $\Gamma$, $\theta$
Ensure: $s''$
 1: $CP := emptyList$
 2: $CN := emptyList$
 3: $G^* := DAG\_Refinement(\Gamma, \theta)$
 4: $s := RSOpt(G^*)$
 5: repeat
 6:     for all $h$ in $H$ do
 7:         $< s', CP > := PlatformAssignment(s, h)$
 8:     end for
 9:     $< s'', CN > := NetworkAssignment(s', \theta)$
10:     $s := RSOpt(CP, CN, G^*)$
11: until ($CP.isEmpty()$ and $CN.isEmpty()$)
12: End algorithm
```

Algorithm 1 describes how this hybrid-iterative approach works, where the input Γ represents a given timetable and θ denotes corresponding physical structure information of stations and railway network which is a new feature for enhancing the basic model in Lin and Kwan (2014), and the output s'' is the full train unit scheduling results with finalized coupling order assignment and linkages. CP and CN are conflict lists and initialized to be empty at the beginning of the algorithm. A refined graph G^* is to be generated by introducing θ for the use of *RS-Opt* to give an incomplete scheduling solution s. Next step is to pass s through two stages of coupling order assignment, platform-based assignment and network-based assignment respectively. During coupling order assignment, the station-level conflicts are to be detected and saved in the lists of CP and CN. The collected station-level conflicts are converted as dynamic constraints to be added back to *RS-Opt* re-optimizing a new solution. This loop will be executed until the algorithm reaches the stop conditions such as no conflict detected, iteration time bound etc.

4.1 *DAG* Refinement

A directed acyclic graph (DAG, $G = (N, A)$) consists of a set of nodes and directed arcs such that no cycle exists. *RS-Opt* is based on the original graph G where nodes are the trips in a given timetable and directed arc a between nodes i and j ($a = (i, j)$) is representing a potential linkage and x_a^τ are the unit type flow on arcs. Source and sink nodes are added as usual. The arcs start at the source node (0) are called *sign-on arcs* and the arcs end at the sink node (∞) are called *sign-off arcs* and the other arcs are defined as *trip-to-trip arcs*, denoted by A_0, A_∞ and A_N respectively. A path p is defined as a serving sequence of trip nodes starting at the source node 0 and ending at the sink node ∞, which is a *diagram* of daily workload for a unit. While generating the original G, a set of constraints are implemented for instance minimum turnaround time, location banning for coupling/decoupling operations, permitted unit type for each trip, etc. Since each station is considered as a single point and linkages are basically restricted by rough minimum turnaround times. G does not include the station-level constraints. On the other hand, a great large number of possibilities of linkage combinations among all trips in a scheduling solution increase the difficulty and complexity of the train unit scheduling problem. Two measures are taken to enhance the original graph G, which is significantly helpful to narrow down the searching space of *RS-Opt*.
(1) Updated minimum turnaround time to reduce the size of G: usually, the arrival and departure platforms are fixed for each trip during the timetabling stage. Regarding the station geographical structure and connections of railway network, some platform pairs at a station are incompatible for re-platforming. This feature cannot be realized as G does not contain any station structure. However, it is significant to slim the size of G. The linkage arcs between incompatible platform pairs can be crossed out or assigned a very large turnaround time. Once the railway structure is considered, some simple station-level shunting movements can be easily detected such as re-platforming at a station. The station shunting could be time consuming and the turnaround time must be modified according to the activated platform information. If some station shunting movements are needed for implementing a linkage, some essential operational time must be considered. Hence, the linkages with slack time less than the updated turnaround time can be eliminated from G.
(2) Updated arc cost to increase the quality of G: in the daily railway operation, normally, a unit block should not park at a platform for a long time. It should serve another trip as soon as possible or be shunted to the depot/siding if there is enough time. Thus, the first-in-first-out principle is implied as an additional cost for each arc in A_N, shown in equation 1, which can encourage the connection in Figure 5 (a) rather than (b). In addition, the linkage preference can be set at a few levels. Each trip node $i \in N$ has a group of entering arcs (A_+^i) and leaving arcs (A_-^i). The arc-costs can be set precisely such that the longer the time gap, and the more operations, the higher the cost. Consider the leaving linkages of each trip node i as two groups: direct linkages (A_d^i) without any location change, indirect linkages (A_r^i) which may

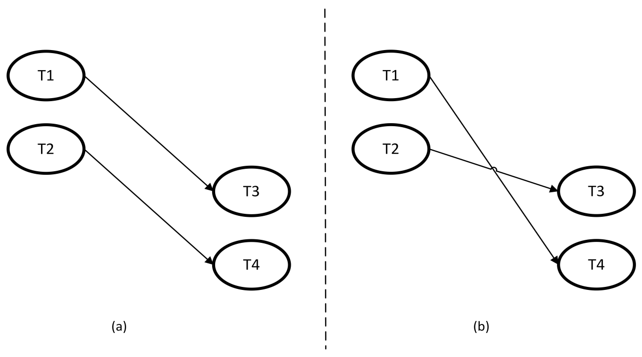


Fig. 5: Linkages encouraged by FIFO principle

involve locations changes at corresponding stations for example re-platforming linkages, depot-return linkage, etc. Therefore, a hierarchical additional linkage cost strategy can be designed for both groups of linkages based on the practical knowledge, shown in equations 2 and 3.

$$c_e(a) = \alpha * (t_s^a)^2,\ \forall a \in A_N \tag{1}$$

$$c'_e(a) = \beta * (ts_a)/\sum_{a \in A_d^i} (ts_a),\ \forall a \in A_d^i,\ \forall i \in N \tag{2}$$

$$c'_e(a) = \gamma * (ts_a + (tr_a)^2)/\sum_{a \in A_r^i} (ts_a),\ \forall a \in A_r^i,\ \forall i \in N \tag{3}$$

4.2 Platform-based coupling order assignment

Day-time coupling and decoupling operations are mostly done at platforms in the UK, which can be interpreted by the solution graph of *RS-Opt*. The linkages can be roughly described as three fundamental scenarios: direct scenario, coupling scenario and decoupling scenario. The multi-coupling/decoupling operations can be regarded as the combinations of these basic scenarios.
(1) Direct scenario: for a trip node i, if $\sum_{a \in A_+^i} y_a = 1$ and $\sum_{a \in A_-^i} y_a = 1$, the unit block serving trip i does not involve neither coupling nor decoupling operations. For this scenario, the coupling order is not constrained.
(2) Coupling scenario: for a trip node j, if $\sum_{a \in A_+^j} y_{(i,j)} \geq 2$, at least one coupling operation must be implemented to form trip j, which may impose a critical coupling order for trip j at its origin.
(3) Decoupling scenario: for a trip node i, if $\sum_{a \in A_-^i} y_{(i,j)} \geq 2$, at least one decoupling operation must be executed for the unit block of node i to serve other different trips, where a specific coupling order of trip i at its destination may be needed.

Normally, two types of operation are critical to the coupling order determination: coupling operation forms a certain coupling order; decoupling operation needs a certain coupling order for a multi-type unit block. Coupling and decoupling operations highly depend on station geographical layouts, arrival and departure times of trips at corresponding stations, and approaching directions to platforms. Thus, the coupling order can be tentatively decided by coupling or decoupling operations combined with timings, directions and geographical station layouts. Besides, the cross matching blockage and unit block accumulation at each platform can be verified and the conflicts caused by these two factors can also be collected at this stage.

Algorithm 2 takes each platform (h) as the input and obtains train unit schedule (s') with partial coupling order assigned by coupling and decoupling operations at each platform and collects conflicts (CP) to be further resolved. For each platform, there are three lists from the *RS-Opt* solution s: arrival list, departure list and linkage list connecting the arrivals to departures. This algorithm starts with the following initialization: dL and aL are time sorted departure and arrival trip list respectively; *dtrip* takes the first departure trip and its departure time is assigned to *time*. If an arrival unit block u needs re-platforming shunting, one dummy trip ($\tilde{u}_{(i,j)}$) will be generated to express that u leaves the destination platform of trip i to the departure platform of trip j. Its arrival and departure times are flexible and how to verify those flexible times will be discussed in section 4.4.

unitStore is a data structure for holding the unit blocks at each platform. For the dead-end platform, an element (unit block) can only get in or get out from one end of the *unitStore*; but the element can be pushed in or popped out from either end of the *unitStore* for a through platform. Each element pushed into *unitStore* is delivered by arrival trips (*atrip*) and each element popped out from *unitStore* is going to serve departure trips (*dtrip*), including newly generated dummy trips for shunting movements. Since coupling and decoupling operations are required sometimes, each element may be split into multiple elements and contiguous elements may be glued together as one element. The elements get in/out *unitStore* by a certain sequence sorted by timings, coupling/decoupling operations, moving directions and station layouts.

The *unitStore.length* is for limiting the unit accumulation within the maximum unit number/length (max) for each platform as the platform length is limited. An arrival unit block can be pushed into the *unitStore* if it has enough space, otherwise, the method $unitShuntAway()$ will be applied to check if some units can be shunted away to give enough space to push it into *unitStore*, which is based on the time available in the corresponding linkages. If $unitShuntAway() = true$, corresponding dummy trips for the moved-away unit blocks will be generated. The unit-shunted-away operations (sh) of these shunting movements will be merged into the solution s', and the arrival unit block will be pushed into *unitStore*; if $unitShuntAway() = false$, this case will be recorded as an over accumulating conflict to be saved in the conflict list CP. The unit blocks (u_c) causing this conflict will be removed from *unitStore*. For each departure trip, the method of checking if its related linkages can be

Algorithm 2 Platform assignment algorithm (for each platform)

```
Require: $h, h \in H$
Ensure: $s'$ and $CP$
 1: $dL := sortedDepartureTripList$
 2: $aL := sortedArrivalTripList$
 3: $dtrip := dL.firstrip()$
 4: $time := dtrip.depTime$
 5: repeat
 6:     for all $atrip$ in $(aL \mid atrip.arrTime < time)$ do
 7:         if $(unitStore.length < max)$ then
 8:             $unitStore.push(atrip.composition)$
 9:         else if $(unitStore.length > max$ and $unitShuntAway() = true)$ then
10:             $sh := shuntingInformation$
11:             $s' := union(s, sh)$
12:             update $element\ sequence$
13:             $unitStore.push(atrip.composition)$
14:         else if $(unitStore.length > max$ and $unitShuntAway() = false)$ then
15:             $c := newConflict$
16:             $CP.add(c)$
17:             $unitStore.remove(u_c)$
18:         end if
19:     end for
20:     $link := dtrip.linkages$
21:     if $(canImpli(link) = true)$ then
22:         $s' := union(s', couplingOrder)$
23:         $unitStore.pop(dtrip.composition)$
24:     else
25:         $c := newConflict$
26:         $CP.add(c)$
27:         $unitStore.remove(u_c)$
28:     end if
29:     $dtrip := dL.next()$
30:     $time := dtrip.depTime$
31:     update $aL$ and $dL$
32: until $(dL.isEmpty())$
33: End algorithm
```

implemented at the station level is called $canImpli(link)$. If yes, the coupling order *couplingOrder* will be merged into the solution s' and the unit block serving the departure trip will be popped out of *unitStore*; if not, a new conflict will be saved into CP and the unit blocks related to this conflict will be removed out of *unitStore*. Then *dtrip* and *time* will be renewed as the next departure trip and its departure time respectively and aL and dL will be also updated. The main loop of this algorithm is executed for each departure trip until dL is empty.

Part of the coupling order and necessary shunting movements for some unit blocks can be determined and this is the basis of the network-based coupling order assignment process. Meanwhile, this stage collects some conflicts caused by a set of linkages used in a solution, called *arc-only conflicts*. The resolution of this type of conflict is going to be discussed in section 4.5

4.3 Network-based coupling order assignment

Unit block coupling order may have been fixed for some trips by the platform-based coupling order assignment. Nevertheless, there may be some trips left to be further determined, gathered in the unfixed-trip list uL and sorted by their departure times. Since station operations are connected by running trips, the coupling order of some trips in uL can be eventually determined with respect to the order-fixed unit blocks and railway-network structure. Two particular issues for the network assignment processes: the en-route reversal of a unit block; flexible timings for some empty shunting movements (to be discussed in section 4.4).
The propagation of coupling order on the network can be addressed from the solution graph of *RS-Opt* and network conflicts can be captured during the propagation process. A simple example is shown in Fig 6, in which three unit (of types X and Y) paths serving seven trips are used. Suppose that those three

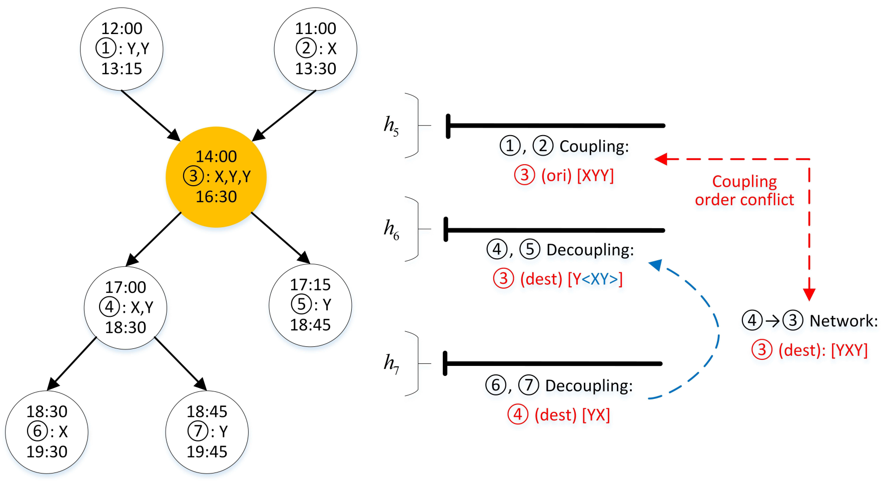


Fig. 6: Coupling order propagation on a network

platforms are dead-end and all the trips have no en-route reversal operation. After the platform-based coupling order assignment stage, the coupling order of trip 3 at its destination is partially undermined, which can be further determined by searching forward/backward along with relevant unit paths to find some other coupling-order determined trips. Ultimately, the coupling order of trip 3 ends up with a coupling order conflict.
A node $i, i \in uL$ is covered by unit diagram path(s) P_i connecting other nodes on this network. Set the trip node i as the divider. The trip nodes on the paths in P_i can be split into two time-sorted lists. One includes the trips earlier than i, called *backward trip list* and denoted by bL_i; the other contains the trips

later than i, called *forward trip list* and denoted by fL_i.
The main loop of Algorithm 3 is to search in bL_i and fL_i for each trip i in uL, shown in Figure 7. The searching process follows the paths in s. Each search step traces through trips on a path till fixed coupling order is found or the coupling order of node i does not propagate in the searching space anymore. Note that only the arcs involving coupling, decoupling operations and the arcs

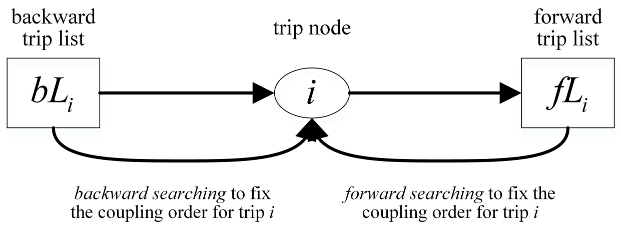

backward trip list
trip node
forward trip list
bL_i
i
fL_i
backward searching to fix the coupling order for trip *i*
forward searching to fix the coupling order for trip *i*

Fig. 7: Backward and forward searching at the network level

with multi-type unit flow pass the propagation of coupling order. The trip nodes in bL_i which can fix the coupling order of its divider trip i are called *backward valid trip nodes*. Similarly for *forward valid trip nodes*. The coupling order of node i can be further determined by the valid nodes found from either bL_i or fL_i. A coupling order conflict is to be captured if the coupling orders of node i decided by bL_i and fL_i are different. If there is no any valid node found, the coupling order will be left undetermined.
Some trips may not be assigned a fixed coupling order after these two-stage coupling order assignment. For example in the case of $i_b = i_f = undetermined$, the coupling order does not cause any potential operational blockage, and it is not critical to a coupled unit block on its path. Coupling order conflict arises only if $backwardCouplingOrder \neq forwardCouplingOrder$ and the coupling order for other cases can be addressed out.
The coupling order conflict is not confined to a set of arcs but also the unit-type-quantity flow on those arcs, which is defined as an *arc-flow conflict*. For instance, Fig. 8 shows an example of 19 trips with assigned unit compositions and tentative linkages, but the set of red arcs and the set of purple arcs are reported as two arc-flow conflicts to be added to the list of CN considering the station layouts. These two conflicts are to be fed back to *RS-Opt* to be re-optimized and will be eliminated in a new solution.

4.4 Flexible timings

The time available for unit blocks transiting from arrivals to departures are limited, which is rigidly constrained by the timetable. Within this limited time space, a series of time-consuming operations must be accomplished at the corresponding stations including coupling, decoupling, re-platforming, shunting

Algorithm 3 Network assignment algorithm

```
Require: $s'$, $\theta$
Ensure: $s''$ and $CN$
 1: $uL := sortedUnfixedTripList(s')$
 2: $i := uL.firstrip()$
 3: $fL_i := sortedForwardTripListforTrip\ i$
 4: $bL_i := sortedBackwardTripListforTrip\ i$
 5: repeat
 6:     $backwardsearching(bL_i)$
 7:     if $validtripisfound\ in\ bL_i$ then
 8:         $i_b := backwardCouplingOrder$
 9:     else
10:         $i_b := undetermined$
11:     end if
12:     $forwardsearching(fL_i)$
13:     if $validtripisfound\ in\ fL_i$ then
14:         $i_f := forwardCouplingOrder$
15:     else
16:         $i_f := undetermined$
17:     end if
18:     if $i_b = i_f$ and $i_b \neq undetermined$ then
19:         $s'' := union(s', i_b)$
20:     else if $i_b \neq i_f$ and $i_b := backwardCouplingOrder$ then
21:         $s'' := union(s', i_b)$
22:     else if $i_b \neq i_f$ and $i_f := forwardCouplingOrder$ then
23:         $s'' := union(s', i_f)$
24:     else if $i_b = i_f$ and $i_b = undetermined$ then
25:         $s'' := union(s', i_b)$
26:     else
27:         $c := newConflict$
28:         $CN.add(c)$
29:     end if
30:     $i := uL.next()$
31:     update $uL$
32: until $(uL.isempty())$
33: End algorithm
```

to depot/siding, cleaning, equipment inspection etc. If a unit block u arrives with trip i at platform h and finishes all the necessary operations consuming time of tp_h and later leaves from h to another platform h' to serve trip j. The dummy trip $\tilde{u}_{(i,j)}$ from h to h' is generated and its departure time $td_{\tilde{u}}$ and arrival time $ta_{\tilde{u}}$ are flexible. However, the duration time of dummy trip u is restricted by the clock time of timetable such that time boundaries for the departure and arrival times of dummy trips must be considered to ensure no blockage caused at inappropriate time. Normally, $\tilde{u}$ can be moved away from h after the necessary operations are done but it must be shunted away before the next arriving trip, and its departure time range is shown in constraint 4. Besides, $\tilde{u}$ must arrive at h' earlier than the departure time of trip j minus necessary departure operations and later than the last departure trip at h', and its arrival time range is shown in constraint 5.

$$ta_i + tp_h < td_{\tilde{u}} < ta_{i+1},\ \forall \tilde{u} \in U \tag{4}$$

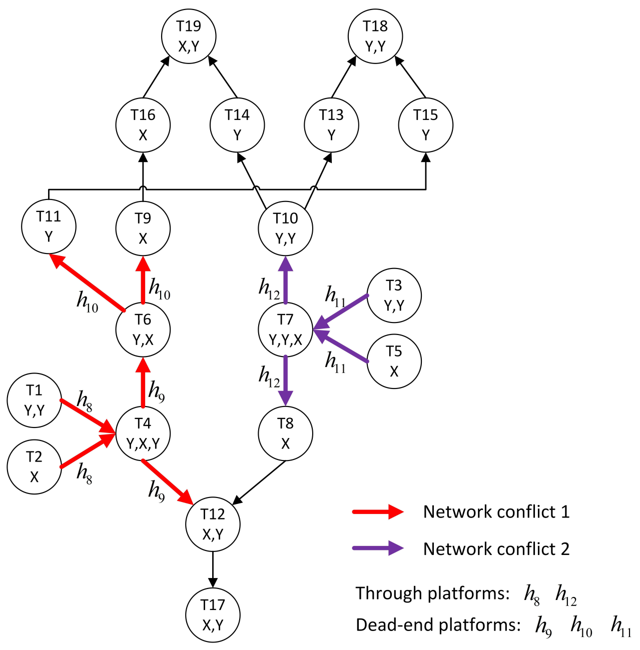
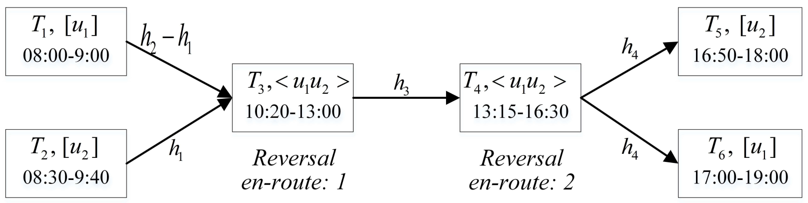


Fig. 8: Network conflicts to be added to CN

$$td_{j-1} < td_{\tilde{u}} < ta_j - tp_{h'},\ \forall \tilde{u} \in U \tag{5}$$

Figure 9 shows a simple example of avoiding blockage by manipulating flexible time boundaries together with coupling order decision. Suppose that h_1 and h_4 are dead-end platforms and h_3 is a through platform and the directions of T_3 and T_4 are the same. The following two procedures explain what time boundary is feasible.



T_1, $[u_1]$ 08:00-9:00
T_2, $[u_2]$ 08:30-9:40
$h_2 - h_1$
h_1
$T_3, <u_1u_2>$ 10:20-13:00
Reversal en-route: 1
h_3
$T_4, <u_1u_2>$ 13:15-16:30
Reversal en-route: 2
h_4
h_4
T_5, $[u_2]$ 16:50-18:00
T_6, $[u_1]$ 17:00-19:00

Fig. 9: Avoid blockage by manipulating the flexible timings boundaries

Procedure 1. $ta_2 < ta_{\tilde{u}_1} < td_3 - tp_{h_1} \rightarrow T_3[u_1u_2]_{ori} \rightarrow rev([u_1u_2], 1) \rightarrow T_3[u_2u_1]_{dest} \rightarrow T_4[u_2u_1]_{ori} \rightarrow rev([u_2u_1], 2) \rightarrow T_4[u_2u_1]_{dest} \rightarrow u_1$ blocks the

departure of $u_2 \rightarrow$ infeasible time boundaries
Procedure 2. $ta_{\tilde{u}_1} < ta_2 \rightarrow T_3[u_2u_1]_{ori} \rightarrow rev([u_2u_1], 1) \rightarrow T_3[u_1u_2]_{dest} \rightarrow T_4[u_1u_2]_{ori} \rightarrow rev([u_1u_2], 2) \rightarrow T_4[u_1u_2]_{dest} \rightarrow T_5[u_2]_{ori}$ and $T_6[u_1]_{ori} \rightarrow$ feasible time boundaries

4.5 Resolving remaining coupling order conflicts

The conflicts detected at both the platform-based and network-based stages must be resolved to obtain a fully operable solution. A naive method is to introduce extra shunting movements (such as re-ordering, side shunting etc.) to make it be operable if the relevant linkages have sufficient slacks for such shunting operations (Lei et al (2017)). For the conflicts which cannot be resolved by that naive method, new cut will be generated for each conflict accordingly and fed back to *RS-Opt* as dynamic constraints to be re-optimized. A set of arcs in a solution causing either arc-only or arc-flow conflict is denoted by A_c^+; $y_a = 1$, $\forall a \in A_c^+$.
(1) The resolution of arc-only conflict (CP): if a network flow solution contains all the arcs in A_c^+, the conflict c will happen. To avoid the conflict c, the arcs selected in a new solution must not include all the arcs in A_c^+, which can be realized by constraints 6. By dynamically adding these constraints at each iteration of Algorithm 1, solutions including arc-only conflicts will be eliminated.

$$\sum_{a \in A_c^+} y_a \leq \mid A_c^+ \mid -1, \ \forall c \in CP \tag{6}$$

(2) The resolution of arc-flow conflict (CN): an arc-flow conflict c' arises when a certain structure of arcs $(A_{c'}^+)$ selection and the specific unit-type-quantity flow on those arcs $(x_a^\tau, a \in A_{c'}^+)$ are included in a solution, where x_a^τ can be extracted from the integer path-flow decision variables x_p^τ in *RS-Opt* according to equation 7, in which P_a is the path set covering arc a.

$$x_a^\tau = \sum_{p \in P_a} x_p^\tau, \ \forall a \in A \tag{7}$$

To eliminate the arc-flow conflict from the solutions, the integer cut technique proved by Balas and Jeroslow (1972) can be applied, aiming at adding integer constraints which is infeasible for the integer point ($x_i = 1$, $i \in B$ and $x_i = 0$, $i \in Q$) and feasible for other integer points, where B is the set of variables whose values are equal to 1 and Q is the set of variables whose values are 0. The logical relation of this event is shown in equation 8, which can be converted to equation 9. Since the variable x_i is binary, equation 9 can be restrictively mapped to integer constraints 10 to avoid a certain integer point.

$$\neg[(\wedge x_i, (i \in B)) \wedge (\wedge x_i, (i \in N))] \tag{8}$$

$$(\vee \neg x_i, (i \in B)) \vee (\vee x_i, (i \in N)) \tag{9}$$

$$\sum_{i\in B}(1-x_i)+\sum_{i\in N}(x_i)\geq 1 \tag{10}$$

This technique is only valid for binary variables, but the arc-flows to be avoided are integers. Thus, the binary variable $x_a^{\tau q}$ is introduced, which equals to 1 when x_a^{τ} equals to any value in the set of {1,2,3} as the unit coupling upper bound for a trip is assumed to be three for typical UK operations; otherwise, it equals 0.

$$x_a^{\tau q}=\begin{cases}1, & \text{if } q=(1,2,3).\\ 0, & \text{otherwise.}\end{cases} \tag{11}$$

An arc-flow conflict c' is only part of a solution, which needs to be isolated from the entire solution if we target to eliminate conflict c'. Note that $N_{c'}$ represents the set of trip nodes which cause the arc-flow conflict c'; $S_{c'}$ represents the set of stations related to the conflict c'; A^* represents the set of arcs in which the arcs are supposed to be implemented at the station of $S_{c'}$, $A^*\subset A$ and $A_{c'}^-=A^*\backslash A_{c'}^+$. The integer cuts to be added back to the *RS-Opt* are constraints 12.

$$\sum_{a\in A_{c'}^+}\sum_{\tau}(1-\sum_{q}x_a^{\tau q})+\sum_{a\in A_{c'}^-}\sum_{\tau}\sum_{q}x_a^{\tau q}\geq 1,\ \forall c'\in CN \tag{12}$$

5 Ongoing research and conclusions

Two small artificial datasets are designed for the purpose of testing *RS-Opt-PT*. The first one turned out to have no conflict found in the *RS-Opt* solution and feasible coupling orders can be determined. For the other one, two arc-flow conflicts were present. Investigation of the technique for resolving the conflicts by means of dynamic cuts in section 4.5 is still on going. More datasets are to be tested for covering as many cases as possible. This research also benefits from the collaboration with First Group and real datasets from GWR are being tested. One of the datasets contains 579 trips, 9 unit types from 2 families and 30 origin/destination stations. The setup and configuration for testing and refining our models especially on implementing the conflict constraints in *RS-Opt* are a very substantial ongoing task, which has to be carefully analyzed with artificial and real datasets. More details and results will be reported in a future paper.
The network-flow model has the limitation of ignoring station level constraints, which leads to an incomplete solution. This defect restricts the operability when it is implemented at the station level because of a set of undecided factors, for instance, the coupling order impacted by station layouts, timings, and unit movement directions, etc. This research scrutinizes the potential problems in a solution of *RS-Opt* and the hybrid iterative approach is proposed to offer a more complete and operable solution by systematically analyzing the

station-level conflict constraints. First of all, it can enhance the network-flow solution to fix and validate the coupling orders. Secondly, it can capture the conflict constraints to feed back to the *RS-Opt* to guide the re-optimization process and eliminate potential operational conflicts by narrowing down the search space. In our future research, more features and connections between the network flow level and station level will be investigated and a new mathematical model may also be derived to reform the linkages and to finalize the complete train unit scheduling solution.

Acknowledgements Li Lei is funded by a Ph.D. grant of the School of Computing, University of Leeds. And we would like to thank First Group, and Tracsis Plc for their kind and helpful collaboration.

References

Balas E, Jeroslow R (1972) Canonical Cuts on the Unit Hypercube. SIAM Journal on Applied Mathematics 23(1):61–69

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

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

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

Copado-Mendez P, Lin Z, Kwan R (2017) Size limited iterative method (SLIM) for train unit scheduling. In: Proceedings of the 12th Metaheuristics International Conference, Barcelona, Spain

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

Haahr J, Lusby RM (2017) Integrating rolling stock scheduling with train unit shunting. European Journal of Operational Research 259(2):452–468

Huisman D, Kroon LG, Lentink RM, Vromans MJCM (2005) Operations research in passenger railway transportation. Statistica Neerlandica 59(4):467–497

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

Kwan RSK, Lin Z, Copado-Mendez PJ, Lei L (2017) Multi-commodity flow and station logistics resolution for train unit scheduling. In: Proceedings of the 8th Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA, pp 321–324

Lei L, Kwan R, Lin Z, Copado-Mendez PJ (2017) Station level refinement of train unit network flow schedules. In: 8th International Conference on Computational Logistics, ICCL

Lin Z, Kwan RSK (2013) An integer fixed-charge multicommodity flow (FCMF) model for train unit scheduling. Electronic Notes in Discrete Mathematics 41:165–172

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

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

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

Tomii N, Zhou LJ, Fukumura N (1999) An algorithm for station shunting scheduling problems combining probabilistic local search and PERT. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, pp 788–797