Omega xxx (xxxx) xxx

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

A branch-and-price approach for trip sequence planning of high-speed train units☆

Yuan Gao[a,*], Marie Schmidt[b], Lixing Yang[c], Ziyou Gao[c]

[a] School of Management and Economics, Beijing Institute of Technology, Beijing, 100081, China
[b] Rotterdam School of Management, Erasmus University Rotterdam, Rotterdam, 3000 DR, The Netherlands
[c] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, 100044, China

ARTICLE INFO

Article history:
Received 24 February 2019
Accepted 27 October 2019
Available online xxx

Keywords:
Train units scheduling
Trip sequence planning
Maintenance
High-speed railway
Branch-and-price algorithm

ABSTRACT

In high-speed railway operations, a trip sequence plan is made once the timetable is determined, and serves as a reference in the subsequent operations of train units scheduling. In light of the maintenance requirements of train units and periodicity characteristics of trip sequences, we introduce a trip sequence graph to describe the train units' movement and coupling/splitting in a railway network. Based on the trip sequence graph, two integer linear programming models are then formulated, namely a path-based model and an arc-based model. Integrated with the characteristics of the trip sequence graph, a customized branch-and-price algorithm is developed to solve the path-based model. The two models are applied to the high-speed railway network in eastern China, and through numerical experiments, the effectiveness and applicability of the models are discussed.

1. Introduction

Due to the system complexity, the methods of multi-stage planning are widely applied in railway systems, and a typical example is the scheduling of train units in a high-speed railway network. The process of train units scheduling relates to line planning, train timetabling, trip sequence planning and train unit assignment planning. The former three belong to the tactical level, the horizons of which are several months or years, while the last one, which is usually a weekly plan, belongs to the operational level.

Considering network condition and passenger demand, *line planning* determines the origin and destination stations, the route and the stop plan of a trip. Besides, the type of train unit carrying out the trip is also determined in line planning. The output of line planning is the input of *train timetabling*, which gives the specific departure and arrival times at each station of all trips. Next is *trip sequence planning*, which is also referred to as train unit routing. A trip sequence consists of a series of trips satisfying some restrictions, such as shunting, coupling/splitting and maintenance. Instead of a trip, a trip sequence is the smallest task for train units. In the stage of trip sequence planning, a trip sequence plan is made, which determines a series of trip sequences to cover all trips. In *train unit assignment planning*, the trip sequence plan serves as a reference. Specifically, based on a given trip sequence plan and the maintenance state of the train units in the beginning of a week, a train unit assignment plan assigns individual train units to trip sequences on each day of the week or arranges some train units to undergo maintenances on certain days of the week.

Given a timetable, trip sequence planning and train unit assignment planning together determine the movement of train units in a high-speed railway network, including carrying out trips and undergoing maintenances. A main reason that trip sequence planning and train unit assignment planning are separated is due to the complex maintenance regulations. For example, in China, there are five levels of maintenance, two short-term and three long-term ones. A train unit must undergo maintenance of a certain level before the accumulative travel distance or elapsed time since the last maintenance of this level exceeds a predetermined value, whichever comes first. The duration of maintenances of different levels and the locations where they can be executed differ. Since level I maintenance is executed every day or every two days, when making a trip sequence plan, restrictions of level I maintenance are taken into account. That is, the total travel distance and elapsed time of a trip sequence are respectively less than the maximum travel distance and elapsed time between two level I maintenances, and the origin and destination of a trip sequence must be the same station where level I maintenance is performed. As a

☆ This manuscript was processed by Associate Editor Salazar-Gonzalez.

* Corresponding author.
E-mail address: gaoyth@163.com (Y. Gao).

https://doi.org/10.1016/j.omega.2019.102150
0305-0483/

result, it is convenient to use trip sequences instead of trips as the smallest unit in scheduling train units when taking into account different levels of maintenance.

A good trip sequence plan means that the train units assigned to the trip sequences can be fully used before level I maintenance. The more the train units are used, the lower operation cost is. In this paper, the operation cost of a trip sequence plan includes the cost of train units, the cost of level I maintenance and the travel cost. We aim to find a trip sequence plan which leads to the minimal operation cost. Note that in different countries, the regulations on high-speed railway management and train unit maintenance vary. We propose the trip sequence models on the background of the Chinese high-speed railway, the length of which occupies more than two thirds of the total length of the high-speed railway in the world. In other countries with similar maintenance regulations, such as Japan and Korea, similar models can be constructed.

In order to express the movement and coupling/splitting of high-speed train units, we develop a trip sequence graph in this paper. This graph differs from other graph-based models in the literature on train unit scheduling or aircraft routing problems, since they either ignored the difference of shunting times between different platforms, or were not able to express coupling/splitting times. In a busy high-speed railway network, in which the train units circulate rapidly, shunting times and coupling/splitting times significantly affect the construction of a trip sequence. In the trip sequence graph proposed in this paper, shunting times and coupling/splitting times are be explicitly considered.

Based on the trip sequence graph, two ILP models, namely a path-based model and an arc-based model, are formulated for the trip sequence planning problem. If the problem is of medium size, the arc-based model can be solved to optimality by an ILP solver within acceptable time. If the problem is of large size, only the path-based model can be used and a customized branch-and-price algorithm is proposed to solve the model. We propose an efficient branching strategy, which is motivated by the rule of branch-on-follow-ons, to generate left and right child nodes. In order to efficiently solve the pricing problems, a bi-directional label setting algorithm is developed. We also introduce some methods to generate the initial column subset at the non-root nodes in the branch-and-bound tree, which significantly speed up the column generation algorithm.

The reminder of this paper is organized as follows. Section 2 reviews the related research on trip sequence planning problem. In Section 3, we give a detailed introduction of the train units movement in a high-speed railway network. In Section 4, we propose a trip sequence graph to express the movement and coupling/splitting of train units. In Section 5, we construct a path-based model on the trip sequence graph for the trip sequence planning problem, and a customized branch-and-price algorithm is developed to solve the path-based model. In Section 6, we formulate an alternative model, namely an arc-based model. In Section 7, we apply these two models to eight cases from the Chinese high-speed railway network, and the effectiveness and applicability of the models are discussed. In Section 8, we draw some conclusions and outline directions for further research.

2. Related work

The trip sequence planning problem is a special problem of train unit scheduling problem. In the past 20 years, different models for scheduling train units from different perspectives have been proposed. To our best knowledge, Schrijver [29] was one of the earliest researchers paying attention to train units scheduling problems. Schrijver [29] aimed to minimize the number of train units of different types for periodic trips on a line in the Netherlands, under the requirement that the passengers' seat demand must be satisfied. In the paper, an integer linear programming model based on a time-space graph was constructed, without considering maintenance requirements. The framework of this model, namely an ILP model on a directed graph, has been widely used in most subsequent research on train units scheduling problems. From then on, researches on train unit scheduling problem can be divided into two categories.

In the first category, the researchers focus on the allocation and composition of train units on each trip, under the condition that the predecessor and successor trips of each trip are given in advance. Abbink et al. [1] presented an integer programming model to allocate train units to different trips, which minimized the seat shortages during the morning peak hours. The model was solved by CPLEX directly, and tested on several scenarios based on the 2001–2002 timetable of NS Reizigers. Alfieri et al. [2] focused on optimizing the numbers of train units of different types together with their efficient circulation on a railway corridor. The (un)coupling of train units was taken into account, which largely increased the complexity of the problem. A solution approach based on an integer multi-commodity flow model was proposed, and applied to a real-life case of NS Reizigers. Peeters and Kroon [26] described a model to determine an optimal daily allocation and composition of train units on a railway corridor of the Netherlands, in which the solution was evaluated based on service quality, robustness and circulation cost. The changes in train composition at origin and destination stations were taken into account, and a branch-and-price algorithm was developed to solve the model, which outperformed the method of solving an integer programming model directly by commercial solvers in a e real-life case study in the Netherlands. Fioole et al. [15] extended the model in [26] to a railway line with branches. A complex mixed integer linear programming model was constructed, and was solved by CPLEX. Taking the maintenance rules of NS Reizigers in the Netherlands as background, Maróti and Kroon [23,24] proposed two models, namely the transit model and the interchange model, to describe the schedule of the train units that require maintenance in the forthcoming one to three days. Given a set of generated disruption scenarios, Cacchiani et al. [7] proposed a two-stage optimization model to obtain a train units circulation plan with high robustness. The model was solved heuristically using Benders Decomposition, and tested on the data of the railways in the Netherlands.

In the second category, trips are not pre-sequenced in advance, and the researchers pay more attention on how to connect the trips and generate proper routings for the train units. Although coupling and splitting are also considered in the second category, they only happen at the origin or destination station of a trip, while in the first category, coupling and splitting are allowed also in some intermediate stations of a trip. Note that the trip sequence planning problem belongs to this category, and a trip sequence is actually a routing of train units. In the following paragraph, we review some representative literature in this category of train unit scheduling problem.

Based on the maintenance rules of the Korean high-speed railway, Hong et al. [17] described the trip sequence planning problem without coupling or splitting as an Eulerian walk and solved the problem in a two-stage heuristic approach. In the heuristic approach, a solution with relaxed maintenance requirements was first obtained by a min-cost flow algorithm, and then a heuristic was employed to modify the solution to meet the maintenance requirements. Considering a coupling upper bound of two units, Cacchiani et al. [6] formulated a path-based models for the train unit scheduling problem. They proposed a heuristic based on column generation to minimize the number of train units used. In their extended model, maintenance constraints were considered, which required that all train units must undergo at least one mainte-

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

nance during a weekly schedule.Cacchiani et al. [9] described the train unit scheduling problem as a multi-commodity flow model, the objective of which was to minimize the number of train units needed. A heuristic based on the Lagrangian relaxation of a natural formulation of the problem was developed to solve the model, which turned out to be much faster in practice and still provide solutions of good quality. For the same problem as in [6,9], Cacchiani et al. [10] proposed a heuristic algorithm based on the optimal solution of the restricted problem associated with a peak period. The experiments on real-world instances showed that the new algorithm was able to find optimal or near-optimal solutions, and outperformed the previous algorithms both in solution quality and computation time. Lin and Kwan [20] formulated an integer multi-commodity flow model for train unit scheduling problem, which aimed at determining an assignment plan such that all trips in the timetable of one day were appropriately covered. Some requirements in UK railway network were considered in the model, such as train unit type compatibility relations and locations banned for coupling/decoupling. They proposed a customized branch-and-price algorithm with multiple branching rules to solve the model. Borndörfer et al. [5] constructed a mixed-integer programming model based on a hyper-graph for train units scheduling, which considered several requirements, such as train composition, maintenance constraints, infrastructure capacities, and regularity aspects. An arc-based model was formulated, and a heuristic based column generation and local search was developed to solve the model.

The specific conditions and operations of Chinese high-speed railway, such as maintenance requirements, periodicity requirements and coupling/splitting restrictions, make the trip sequence planning problem largely different from existing models on train unit scheduling problem. In [17], the authors did not consider coupling or splitting. In [6,9,20], the operation times of coupling and splitting are ignored in the underly networks and models. The trip sequence graph introduced in the paper explicitly takes the operation times of coupling and splitting into account, which makes it different from existing underlying graphs. Besides, in the trip sequence planning problem, the routing of train units is a cycle with a span of one or two days, which cannot be guaranteed by the model in [20] or [5]. In [20], only one-day trip sequences were considered, and the model did not ensure that a train unit returns to its base for maintenance after carrying out a trip sequence. The latter problem also existed in [5], which was formulated based on arcs.

Besides train units, some literature focused on dealing with the efficient circulation of locomotives. Ziarati et al. [32] proposed an integer linear programming model for the problem of assigning locomotives to trains. The model was solved by a branch-and-cut approach, and was tested on the actual data from the Canadian National railway company. Cordeau et al. [12] described a decomposition method for the simultaneous assignment of locomotives and cars for passenger trips. In a subsequent paper, Cordeau et al. [13] extended their model by adding the maintenance requirement of locomotives. Lingaya et al. [21] focused on the problem of assigning cars to scheduled trains for VIA Rail, Canada, in which typical constraints such as maintenance requirements and minimum connection times of individual cars were taken into account. The problem was solved heuristically by a branch-and-bound method, in which the linear relaxations were solved by a column generation algorithm.

The trip sequence planning problem in high-speed railway is similar to the problem of aircraft maintenance routing in American airlines. The aircraft maintenance routing problem does not assign individual aircrafts to flights explicitly, but identifies a generic aircraft routing, which serves as a reference for the subsequent operations of aircraft rotation. Barnhart et al. [4] proposed a string-based model for aircraft fleeting and maintenance routing problem, which was solved by a branch-and-price algorithm. Note that a string is a sequence of flights, which is similar to a trip sequence in this paper. Liang et al. [18] presented a new compact network representation of the aircraft maintenance routing problem, based on which a new mixed-integer linear programming model was formulated. Since the size of the model was polynomial in theory and relatively small even for the large real-life test cases, the model was solved by CPLEX in a short time. Other examples include [8,16,19,25]. Since coupling or splitting does not exist in aircraft maintenance routing problems, these models cannot be used in trip sequence planning problems. However, some methods proposed in the research on aircraft maintenance routing problem can be be used for reference.

In very recent, some researchers integrated train unit scheduling or aircraft routing with other related optimization problems. Salazar-González [28] proposed an integrated model involving fleet assignment, aircraft routing and crew paring problems in a single day. The integrated model was heuristically decomposed into a series of small models, which were tracked by general-purpose solvers. Wang et al. [31] integrated train unit scheduling problem and train scheduling problem on an urban rail line, and a multi-objective mixed integer nonlinear programming model was proposed, which were solved by three approaches. Similar problem was study in [22], and a heuristic algorithm based on Lagrangian relaxation was proposed to solve the problem. Cacchiani and Salazar-González [11] integrated flight retiming, fleet assignment, aircraft routing and crew pairing, which led to a complex mixed integer linear programming model. Four heuristic algorithms, based on column generation, were proposed and compared. Tönissen et al. [30] proposed a two-stage stochastic programming model and a two-stage robust optimization model to integrate maintenance facility location problem and train unit scheduling problem under uncertain conditions. In the first stage, they considered the maintenance facility location problem, and in the second stage, they considered train unit scheduling problem.

3. Background and problem description

In this section, the detailed background requirements on the movement of train units in the Chinese high speed railway is presented.

3.1. Train unit and maintenances

In the Chinese high-speed railway, a train unit is composed of 8 or 16 self-propelled carriages. Two train units with 8 carriages can be coupled to carry out a trip (see Fig. 1), while train units with 16 carriages cannot. With the development of control technology, the times for coupling and splitting are reduced to less than 10 minutes, and even in small stations, coupling and splitting of train units can be performed. As a result, two train units with 8 carriages are much more flexible than one train unit with 16 carriages.

Note that the type and the number of the train units carrying out a trip are predetermined. In this paper, we assume that there is only one type of train units with 8 carriages. Besides, for a trip, its train unit combination is fixed, which means that coupling and/or splitting is only allowed at its origin and destination stations.

Recall that there are five levels of train unit maintenances in the Chinese high-speed railway, two short-term and three long-term. Not every station is equipped with maintenance facilities. Currently, the short-term maintenances, namely level I and II maintenances, can be performed in one of 22 high-speed *train bases*, which are located near big stations. In this paper, a station equipped with a train base is referred to as *major station*.

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

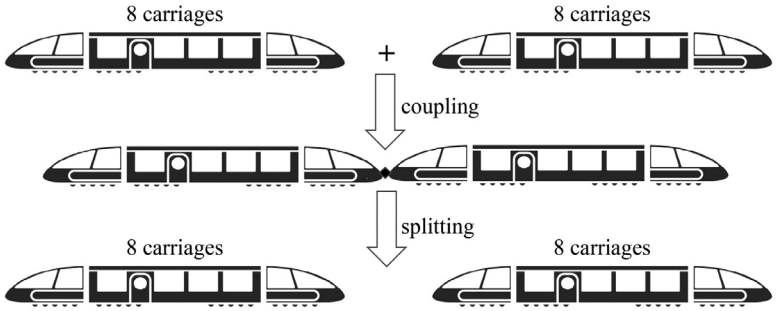

Fig. 1. Train units coupling and splitting.

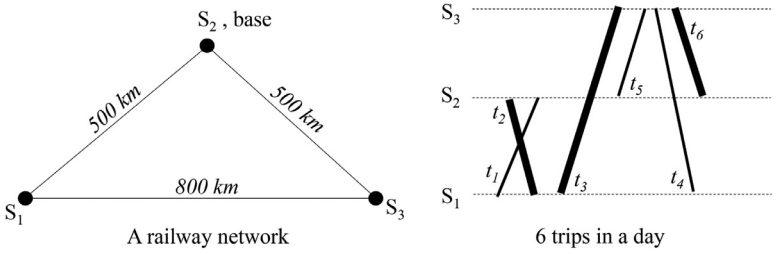

Fig. 2. A small railway network with 6 trips.

Every train unit is affiliated to one and only one major station, in which short-term maintenances are performed. For long-term maintenances (level III, IV and V), however, a train unit must be sent to a special maintenance center or to its producer.

There is no trip on high-speed railway in the night, since there is a maintenance window for railway infrastructure, such as tracks, electrification and communication systems. In China and Japan, train units also undergo level I maintenance in this time window. As a result, level I maintenance does not affect the operation of train units in daytime. For almost all types of high-speed train units, level I maintenance must be carried out after every 4,000 km or 48 h, whichever comes first. We refer to 4,000 km as the *maximum travel distance* between two consecutive level I maintenances, and 48 h as the *maximum elapsed time*. Moreover, a train unit must return to its train base for level I maintenance after carrying out a sequence of trips. For this reason, a trip sequence starts and ends at the same major station.

3.2. Trip sequence

The underlying timetable we consider in this paper is periodic with a period length of one day. Once the timetable of the trips is determined, a corresponding trip sequence plan is made. Generally, the trip sequence plan will not be changed until a new timetable is used.

Note that a trip sequence plan does not assign individual train units to the trip sequences yet. Only in the next step, train unit assignment planning, dispatchers assign individual train units to trip sequences and arrange level II-V maintenances of train units for the following week, based on the state of train units and the trip sequence plan. Fig. 2 presents a small railway network with 6 trips to illustrate the concept of a trip sequence.

In Fig. 2, S_1, S_2 and S_3 are origin and destination stations for the trips we consider, and the intermediate stations are all ignored. In the network, S_2 is the only station equipped with a train base, which serves as the start and end of all trip sequences. Trip t_2, t_3 and t_6 are carried out by two coupled train units, which are depicted by thick lines in Fig. 2; trip t_1, t_4 and t_5 are carried out by one train unit, which are depicted by thin lines.

Fig. 3 gives a possible trip sequence plan for the trips in Fig. 2, which consists of two trip sequences, namely, $R_1 = t_2 \to t_3 \to t_4 \to t_1 \to t_5 \to t_6$ and $R_2 = t_2 \to t_3 \to t_6 \to t_2 \to t_3 \to t_6$. Fig. 3 illustrates these trip sequences, in which the start and end of them are represented by two solid dots respectively. In the figure, if trip sequence R_1 starts on odd days, it is depicted in red color; if R_1 starts on even days, it is depicted in blue color; trip sequence R_2

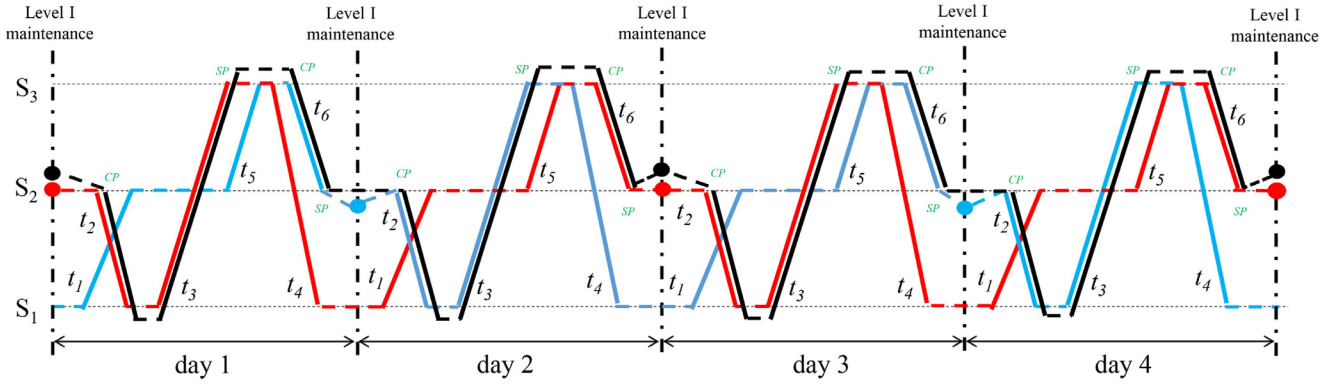

Fig. 3. A possible trip sequence plan.

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

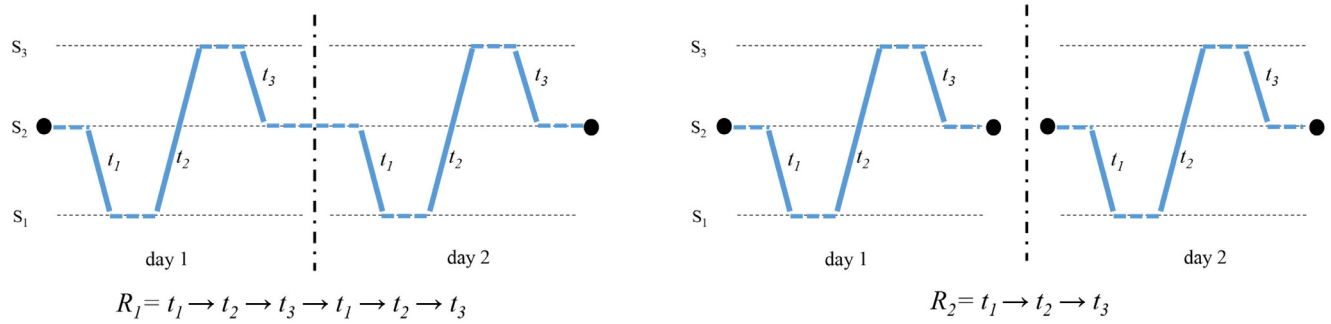

Fig. 4. Illustration of periodic restriction (ii).

starts on odd days only, and is depicted in black color. In Fig. 3, '*CP*" means that there is a coupling operation, and "*SP*" means that there is a splitting operation.

In the morning of odd days, station S_2 assigns two train units that have just been maintained to trip sequence R_1 (in red) and trip sequence R_2 (in black), respectively. These two trains are coupled to carry out trips t_2 and t_3 on odd days, and they are decoupled after trip t_3, and they are coupled again to carry out trip t_6 on even days. In the evening of the even days, these two train units finish the trip sequences and return to S_2 for level I maintenance.

In the morning of even days, station S_2 assigns one train unit that has just been maintained to trip sequence R_1 (in blue). This train unit and the train unit of the black trip sequence are coupled to carry out trip t_6 on odd days, and are coupled to carry out trips t_2 and t_3 on even days. In the evening of the second day, the train unit of the blue trip sequence returns to S_2 for level I maintenance.

It is easy to see that any solution to cover the trips shown in Fig. 3 requires three train units. In the night of odd days, there is one train unit undergoing level I maintenance; in the night of even days, there are two train units undergoing level I maintenance.

Note that the length of a feasible trip sequence must be less than the maximum travel distance. Furthermore, to construct a trip sequence, the time interval between the arrival of a trip and the departure of its successor trip should be wide enough. Take two trips i and j as an example. Without considering coupling and splitting, to ensure a feasible connection between trip i and trip j, the time interval between the arrival of trip i and the departure of trip j should be wide enough for (1) passengers' alighting and boarding, and (2) the shunting time from trip i's platform to trip j's platform.

If coupling and splitting are involved, the time interval should further contain the extra coupling/splitting times. In a few words, a feasible connection depends on passengers' alighting and boarding times, the platforms, and the coupling/splitting operations.

3.3. Periodic restriction and empty trip

Recall that the timetable is periodic with a period length of one day. We make a trip sequence plan which is also periodic with the period length of one day, in the sense that each day, the same trip is covered by the same trip sequence. Since the elapsed time between two maintenances cannot be longer than 48 h and maintenance is executed at night, we allow trip sequences which start and end in the same day (referred to as *one-day trip sequences*) and trip sequences spanning two days (referred to as *two-day trip sequences*). Note that the trip sequences in Fig. 3 are all two-day trip sequences.

Assume that R is a two-day trip sequence, *i.e.*, $R = R^{(1)} + R^{(2)}$, where $R^{(1)}$ is the trip set on "day1", and $R^{(2)}$ is the trip set on "day2". In practice, it is required that $R^{(1)}$ and $R^{(2)}$ must satisfy either (i) $R^{(1)} \cap R^{(2)} = \emptyset$ or (ii) $R^{(1)} = R^{(2)}$, which is referred to as *periodic restriction*. In Fig. 3, we see that trip sequence R_1 satisfies periodic restriction (i), and trip sequence R_2 satisfies restriction (ii).

Note that a two-day trip sequence with periodic restriction (ii) can be regarded as a repeated one-day trip sequence,which is illustrated in Fig. 4, *i.e.,* R_1 and R_2. The difference is that if R_1 is used, the train unit undergoes maintenance every two days, while if R_2 is used, the train unit undergoes maintenance every day. In this paper, it is required that a trip sequence is accompanied by a maintenance. From the perspective of covering trips, there is no difference between R_1 and R_2. However, considering the maintenance cost, R_1 is better than R_2. The introduce of two-day trip sequence like R_1 is to avoid generating one-day trip sequences like R_2, the total travel distance of which is too short.

In some situations, there are *empty trips*, which means that a train unit may travel from station s_1 to station s_2 without passengers, just in order to carry out a trip whose origin station is s_2 or to undergo maintenance in station s_2. In some publications, empty trips are also referred to as *deadheading*. In this paper, a trip that transports passengers is referred to as a *normal trip*, which is in the publicly released timetable. In a busy railway network, empty trips are usually in the early morning or in the late night, in order to reduce the effect on normal trips. In this paper, for the sake of simplicity we assume that empty trips can only take place in the morning a major station to a nearby station without train base, or in the evening, vice versa, from a station without train base to a major station. In particular, there is no empty trip between two stations, if neither of them is equipped with a train base.

3.4. Assumptions

As mentioned above, a trip sequence plan is the base of train units movement. This paper aims to generate trip sequence plans for the Chinese high-speed railway network, which reduce the weighted sum of train units cost, empty trips cost and level I maintenance cost. To construct models for making a trip sequence plan, the following assumptions are considered in this paper:

(1) Each normal trip must be covered by the required number of trip sequences. That is, if a trip is required to be carried out by n train units, the number of trip sequences covering the trip must be equal to n.
(2) The timetable and the trip sequence plan are both periodic with a period of one day. There is only one type of train unit with 8 carriages, and two train units can be coupled together.
(3) Each trip sequence starts and ends at the same station equipped with a train base, and spans one or two days. The maintenances on train units are performed in the night.
(4) Empty trips are permitted, but an empty trip can only take place from a major station with a train base to a nearby station without train base in the morning, or vice versa in the evening.

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

It should be pointed out that in a similar maintenance framework, our models and methods are still valid after minor modifications. For example, if the maximum elapsed time for level I maintenance is 3 days, such as in Korea, we may introduce three-day trip sequences, and the pricing problem in Section 5.2.2 will be more complicated, since three parts are considered. If maintenance can be performed in any time, however, our model will be invalid. Fortunately, in most countries, high-speed train units undergo level I maintenance in the night.

4. Underlying graph

In this section, we propose a *trip-sequence graph* to describe the movement and coupling/splitting of train units in the Chinese high-speed railway network.

4.1. Expression of trips and connections

Note that some trips need two train units, and some trips need one train unit. The former situation is much more complicated. If the two train units are from different earlier trips, coupling and splitting are needed; if the two train units are from the same earlier trip which needs two train units, there is no splitting or coupling. In a trip-sequence graph, the trips and the connections between trips are constructed as in Fig. 5, where we only depict the possible connections between the first four trips and trip t_5. In this graph, each trip is represented by an arc or by a set of arcs. We depict trips that need two units with a double-arrow and trips that need only one unit with a single-arrow. Note that in Fig. 5 all trips except trip t_2 need two train units.

In Fig. 5, the arc of trip t_2 is a single-arrow, while the arcs of other trips are double-arrows. Note that trip t_5 is represented by three arcs, namely, (e, m), (f, m) and (g, m). We refer to arc (g, m) as the *main arc* of trip t_5. In a feasible trip sequence plan, one and only one arc from (e, m), (f, m) and (g, m) is selected for trip t_5. If the time interval between the arrival of a preceding trip and the departure of trip t_5 is wide enough for passengers' alighting and boarding, shunting between platforms and coupling/splitting time, we connect the arrival of the earlier trip and the departure of trip t_5, for example arcs (a, g) and (b, g).

If the time interval between the arrival of a preceding trip needing two train units and the departure of trip t_5 is wide enough for passengers' alighting and boarding and shunting, but not wide enough to allow for splitting and coupling, there still exists a connection between this trip and trip t_5. However, it is required that the two train units carrying out trip t_5 are both from this trip, so coupling or splitting is not needed. For example, trip t_4 is such a trip in Fig. 5. In order to expression the connection between trip t_4 and t_5, we first construct a *special arc* (e, m) for trip t_5, and then construct connection arc (d, e). It should be pointed out that arc (d, e) is the only incoming arc of node e, and the selection of arcs (d, e) and (e, m) ensures that the two train units of trip t_4 are both used to carry out trip t_5. Similarly, we construct another special arc (f, m) for trip t_5, and arc (c, f) is the only incoming arc of node f. In order to distinguish from main arcs, special arcs (e, m) and (f, m) are depicted by double-arrows with solid rhombus in the middle. Obviously, every trip needing two train units has a main arc. Since the coupling and splitting times are quite short, the number of special arcs is actually small.

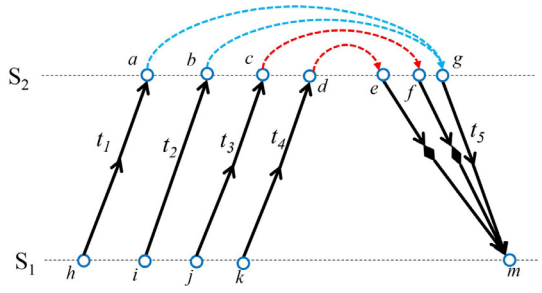

Fig. 5. Constructing arcs for trips needing two train units.

The connection between a trip needing one train unit and a trip needing two train units is also affected by coupling/splitting. Fig. 6 gives an illustration, in which trip t_1 and trip t_2 arrive at platform 1 of station S_2, and trip t_3 and trip t_4 depart from platform 2 of station S_2. Note that in Fig. 6(a), only trip t_1 needs two train units, while in Fig. 6(b), only trip t_4 needs two train units.

In Fig. 6(a), the time interval $[b, c]$ is wide enough for passengers' alighting and boarding and train unit's shunting from platform 1 to platform 2, so we connect node b and node c, which indicates that a train unit can carry out trip t_3 after carrying out trip t_2. Although the time interval $[a, c]$ is wider than time interval $[b, c]$, we cannot connect node a and c. This is because splitting is needed from trip t_1 to trip t_3, and time interval $[a, c]$ is not wide enough for passengers' alighting and boarding, train unit's shunting and splitting. Similar things happen in Fig. 6(b), where interval $[f, h]$ is not wide enough for passengers' alighting and boarding, train unit's shunting and coupling.

4.2. Construction of trip-sequence graph

In the Chinese high-speed railway network, many small stations only serve as intermediate stations, and no shunting or coupling/splitting of train units happens at these stations. So, we neglect these small stations in the trip-sequence graph, and only consider the stations that serve as origin or destination stations of trips. There is a *depot* in each involved station, which serves as the place that train units stay in the night.

A trip sequence graph is denoted by $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the node set and $\mathcal{A}$ is the arc set. In a trip sequence graph, the node set consists of: (i) Departure nodes from origin stations, (ii) arrival nodes at destination stations, (iii) depot nodes, and (iv) train base nodes. The arcs in the graph are (1) normal trip arcs, (2) empty trip arcs, (3) depot-base arcs, and (4) station arcs. Note that depot-base arcs connect depot node and train base node in the same major station, and station arcs connect departure nodes and arrival nodes in the same station if the corresponding time intervals are wide enough. Accordingly, arc set $\mathcal{A}$ is the union of four disjoint sets, *i.e.*, $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$.

Fig. 7 presents the corresponding trip sequence graph for two days for the railway network and the daily trips in Fig. 2. Since the timetable and trip sequence plan are both periodic with a period of one day, in the two-day trip sequence graph, the "day1" part is the same as the "day2" part. In a trip sequence graph, the train base in major station s is modeled by nodes $B_d^+(s)$ and $B_d^-(s)$, which represent entering and leaving the base on "day d" respectively. The same applies to a depot in station s, namely $D_d^+(s)$ and $D_d^-(s)$. It is also assumed that after leaving node $B_d^-(s)$, a train unit must first enter node $D_d^+(s)$, from where the train unit can go to other nodes. Similarly, before entering node $B_d^+(s)$, a train unit must first enter node $D_d^-(s)$. If a two-day trip sequence is carried out by a train unit in major station s on "day1", then $D_1^-(s)$ is the start node of the corresponding path and $D_2^+(s)$ is the end node.

In Fig. 7, the arrival and departure nodes are represented by small blue circles, the train base nodes are black dots, and the depot nodes are represented by shadowed circles. Arcs in $\mathcal{A}_1$ (normal trips) are depicted in the way that was stated in Section 4.1, arcs in $\mathcal{A}_2$ (empty trips) are depicted by green dashed arrows, arcs in $\mathcal{A}_3$ (depot-base arcs) are depicted by blue solid arrows, and arcs in $\mathcal{A}_4$ (station arcs) are depicted by blue dashed arrows. Note that

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

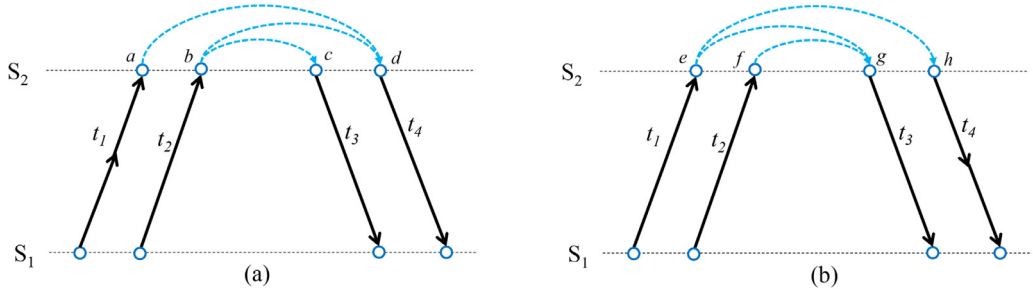

Fig. 6. Connection between a trip needing one train unit and a trip needing two train units.

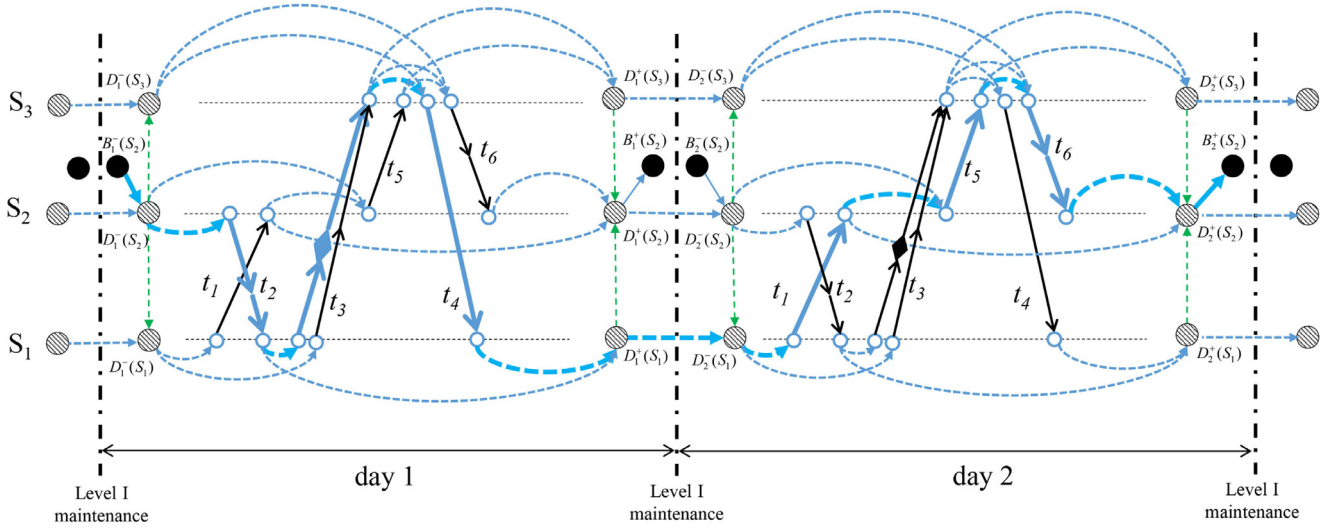

Fig. 7. Trip sequence graph.

trip t_3 is represented by two arcs, namely, a main arc and a special arc. In Fig. 7, the entire path of the two-day trip sequence $t_2 \to t_3 \to t_4 \to t_1 \to t_5 \to t_6$ is illustrated in bold.

The proposed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ conveniently transforms a trip sequence to a path between two base nodes, in which the feasibility of connection between trips is expressed by node connectivity. Besides, the coupling and splitting are expressed by the selection of arcs that represent a trip needing two train units. Note that there exists a mapping from trip sequences to paths from node $D_d^-(s)$ of major station s to $D_d^+(s)$ in graph $\mathcal{G}$, which are referred to as *valid paths*. Without loss of generality, "trip sequence" and "valid path" are interchangeable in this paper. In the next two sections, we formulate two models for the trip sequence planning problem, namely the path-based model and the arc-based model, both of which are based on the trip sequence graph presented in this section. Also, the pricing problem of our column generation algorithm for the path-based model is described on the trip sequence graph.

Although it is intuitive to describe a two-day trip sequence in a two-day trip sequence graph (as in Fig. 7), we use a one-day trip sequence graph when constructing the mathematical programming models. This is because the "day2" part of a two-day trip sequence graph is just a copy of the "day1" part, and the trip sequence plan is periodic with a period of one day.

5. Path-based model

According to assumption (1) in Section 3.4, the generation of the trip sequence plan is derived based on the set covering/partition problem. There are two main formulation approaches for the set covering/partition problem: (i) A column generation formulation or "path-based model", and (ii) a conventional formulation or "arc-based model". Generally, if the problem scale is small, the latter may find an optimal solution in an acceptable computation time. However, if the problem scale is large, the approach of the conventional formulation cannot even find a feasible solution, and we have to resort to a column-generation-based method to find a good solution.

In this section, we propose a path-based model for the trip sequence planning problem, and develop a customized branch-and-price algorithm to solve the problem. Besides, in Section 6, an arc-based model is formulated as an alternative approach for the problem, which also serves as basis for comparison for the path-based model in numerical experiments.

Some basic notations used in the two models are summarized in Table 1.

5.1. Path-based model formulation

Note that for a major station $s \in \mathcal{S}_0$, trip sequence set $\mathcal{R}_s$ can be divided into three disjoint subsets, *i.e.*, $\mathcal{R}_s = \mathcal{R}_s^{(1)} \cup \mathcal{R}_s^{(2)} \cup \mathcal{R}_s^{(3)}$. Specifically, $\mathcal{R}_s^{(1)}$ is the set of one-day trip sequences, $\mathcal{R}_s^{(2)}$ is the set of two-day trip sequences with periodic restriction (i), and $\mathcal{R}_s^{(3)}$ is the set of two-day trip sequences with periodic restriction (ii).

For trip sequence $r \in \mathcal{R}_s, s \in \mathcal{S}_0$, we define an integer decision variable X_r as follows,

X_r the number of train units assigned to trip sequence r.

In our paper, $X_r \in \{0, 1, 2\}$, since at most two train units are coupled to carry out a trip. Recall that a trip sequence $r \in \mathcal{R}_s$ corresponds to a valid path in the two-day trip sequence graph. Without loss of generality, trip sequence set $\mathcal{R}_s$ is also treated as valid

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

Table 1
Notations.

Notation	Definition
$\mathcal{S}$	set of stations
$\mathcal{S}_0$	set of major stations, which are equipped with a train base, *i.e.*, $\mathcal{S}_0 \subset \mathcal{S}$
$\mathcal{T}$	set of trips
$\mathcal{R}$	set of trip sequences
$\mathcal{G} = (\mathcal{N}, \mathcal{A})$	trip sequence graph with node set $\mathcal{N}$ and arc set $\mathcal{A}$, which is presented in Section 4
s	index of stations, *i.e.*, $s \in \mathcal{S}$
t	index of trips, *i.e.*, $t \in \mathcal{T}$
r	index of trip sequences, *i.e.*, $r \in \mathcal{R}$
u, v	index of nodes, *i.e.*, $u, v \in \mathcal{N}$
w	index of arcs, *i.e.*, $w \in \mathcal{A}$
$q_{r,w}$	=1, if arc w is contained in trip sequence r; = 0, otherwise, *i.e.*, $r \in \mathcal{R}$, $w \in \mathcal{A}$
$\mathcal{R}_s$	set of trip sequences in major station s, *i.e.*, $s \in \mathcal{S}_0$
N_w	number of train units needed on arc w, *i.e.*, $w \in \mathcal{A}$
M_s	maximum number of train units that can be maintained in major station s, *i.e.*, $s \in \mathcal{S}_0$
L	maximum travel distance between two consecutive level I maintenances
l_t/l_w	travel distance of trip t, *i.e.*, $t \in \mathcal{T}$; length of arc w, *i.e.*, $w \in \mathcal{A}$
$\epsilon_{LP}/\epsilon_{BB}$	predetermined gaps used in the branch-and-price algorithm

path set in major station s. We associate a set of parameters $q_{r,w}$ for every arc $w \in \mathcal{A}$, which is equal to 1 if arc w is contained in valid path r, and 0 otherwise. Obviously, $q_{r,w}$ describes the arc information of the valid path.

As stated in Section 4.1, a trip that needs two train units may be represented by more than one arc in the trip sequence graph, which correspond to different coupling/splitting types. For trip $t \in \mathcal{T}$, we define $\mathcal{A}(t)$ as the set of arcs that represent trip t. When generating a trip sequence plan, we must select one and only one arc from set $\mathcal{A}(t)$. For this reason, we define binary decision variables Y_w as follows,

$$Y_w \quad = 1 \text{ if arc } w \in \mathcal{A}(t), t \in \mathcal{T} \text{ is selected; } = 0, \text{ otherwise.}$$

We aim to minimize the total cost during the considered time horizon, including the cost of train units, level I maintenance cost and travel cost. The cost of train units is simply expressed by the number of train units that are needed, which can be formulated as

$$F_1 = \sum_{s \in \mathcal{S}_0} \sum_{r \in \mathcal{R}_s} a_r \cdot X_r, \tag{1}$$

where a_r is defined as follows

$$a_r = \begin{cases} 1, & \text{if } r \in \mathcal{R}_s^{(1)} \cup \mathcal{R}_s^{(3)} \\ 2, & \text{if } r \in \mathcal{R}_s^{(2)}. \end{cases}$$

Assume that r is a two-day trip sequence with periodic restriction (i), *i.e.*, $r \in \mathcal{R}_s^{(2)}$. Recall that trip sequence r covers different trips in "day1" and "day2". In order to ensure the periodicity of the trip sequence plan, every day we need $2X_r$ train units to carry out the trips covered by trip sequence r.

The sum of maintenance cost and travel cost is calculated by formula (2),

$$F_2 = \sum_{s \in \mathcal{S}_0} \sum_{r \in \mathcal{R}_s} c_r \cdot m \cdot X_r + \sum_{s \in \mathcal{S}_0} \sum_{r \in \mathcal{R}_s} \Big(\sum_{w \in \mathcal{A}_1 \cup \mathcal{A}_2} c_r \cdot c_t \cdot l_w \cdot q_{r,w} \Big) X_r. \tag{2}$$

In the first term, m is the cost of one maintenance, and in the second term, c_t is the travel cost of per kilometer, l_w is the length of arc w, and c_r is defined as follows,

$$c_r = \begin{cases} 0.5, & \text{if } r \in \mathcal{R}_s^{(3)} \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Recall that trip sequence $r \in \mathcal{R}_s^{(3)}$ covers the same trips on "day1" and "day2". We may balance the maintenance numbers on "day1" and "day2" via choosing the start day of trip sequence $r \in \mathcal{R}_s^{(3)}$. For example, trip sequence $r \in \mathcal{R}_s^{(3)}$ and $X_r = 2$, which means that we assign two train units to trip sequence r on "day1", and we do not assign train unit to trip sequence r on "day2". As a result, these two train units are only maintained in the night of "day2". If we assign one train unit to trip sequence r on "day1" and one train unit on "day2". Then, the former is maintained in the night of "day2" (even days), and the latter is maintained in the night of "day1" (odd days). For this reason, we define coefficient c_r as expression (3).

Then, the total cost can be formulated as follows

$$\begin{aligned} F &= F_1 + \beta \cdot F_2 \\ &= \sum_{s \in \mathcal{S}_0} \sum_{r \in \mathcal{R}_s} \left(a_r + \beta \cdot c_r \cdot m + \beta \cdot \sum_{w \in \mathcal{A}_1 \cup \mathcal{A}_2} c_r \cdot c_t \cdot l_w \cdot q_{r,w} \right) X_r. \end{aligned} \tag{4}$$

Compared with the price of train units, the cost of maintenance and travel cost is quite small, which indicates that coefficient β is a small value. Besides, β depends on the time horizon. In this paper, we simply set $\beta = H/P_{tu}$, where P_{tu} is the price of a train unit and H is the time horizon.

For convenience sake, we define

$$u_r = a_r + \beta \cdot c_r \cdot m + \beta \cdot \sum_{w \in \mathcal{A}_1 \cup \mathcal{A}_2} c_r \cdot c_t \cdot l_w \cdot q_{r,w}, \forall s \in \mathcal{S}_0, r \in \mathcal{R}_s, \tag{5}$$

as the cost parameter associated with trip sequence r.

According to the definition of $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ and variables X_r and Y_w, the path-based model for the trip sequence planning problem can be formulated as follows:

$$\min_X \quad F = \sum_{s \in \mathcal{S}_0} \sum_{r \in \mathcal{R}_s} X_r u_r \tag{6}$$

$$s.t. \sum_{s \in \mathcal{S}_0} \sum_{r \in \mathcal{R}_s} q_{r,w} X_r = Y_w N_w, \quad \forall w \in \mathcal{A}(t), t \in \mathcal{T} \tag{7}$$

$$\sum_{w \in \mathcal{A}(t)} Y_w = 1, \quad \forall t \in \mathcal{T} \tag{8}$$

$$\sum_{r \in \mathcal{R}_s} X_r c_r \le M_s, \quad \forall s \in \mathcal{S}_0 \tag{9}$$

$$X_r \in \{0, 1, 2\}, \quad \forall r \in \mathcal{R}_s, s \in \mathcal{S}_0 \tag{10}$$

$$Y_w \in \{0, 1\}, \quad \forall w \in \mathcal{A}(t), t \in \mathcal{T} \tag{11}$$

where N_w represents the number of train units needed on arc w.

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

Constraints (7) are coverage constraints, which state that if arc $w \in \mathcal{A}(t), t \in \mathcal{T}$ is selected, it must be covered N_w times. Constraints (8) express that one and only one arc in set $\mathcal{A}(t), t \in \mathcal{T}$ must be selected, which determines the type of coupling/splitting of trip t. Constraints (9) ensure that the number of maintenance in train base s is not greater than its capacity. Constraints (10) and (11) are the integer and binary constraints of decision variables X_r and Y_w, respectively.

Model (6)–(11) is derived based on set covering/partition problems. In standard covering/partition problems, there is no variable Y_w. In model (6)–(11), all further restrictions on trip sequences are implicitly expressed by set $\mathcal{R}_s$. It is possible to list all the valid paths in R_s in a trip sequence graph via some search algorithms, however, there is no need to do this. Finding all paths would only lead to extremely many variables, and it is difficult to solve a complicated integer programming with millions of variables. To solve such a set covering/partition problem, a customized branch-and-price algorithm is usually used. Branch-and-price ([3]) is branch-and-bound with a linear programming (LP) relaxation solved by column generation algorithm at each node of the branch-and-bound tree.

5.2. Column generation

Column generation is a commonly used iterative method to solve LPs with a tremendous number of variables, which are inefficient to enumerate explicitly. The algorithm starts with constructing a restricted master LP problem with a subset of columns. After solving the restricted master problem, the dual solutions are used to implicitly find the nonbasic variables with minimum reduced costs. This procedure is referred to as *pricing problem*, which is the most crucial part in designing an efficient column generation algorithm. If the minimum reduced cost is negative, the variable and column corresponding to the minimum reduced cost should be added to the restricted master problem, and the next iteration starts; otherwise, the optimal solution of the original LP has been found in this iteration. The readers may refer to [14] for a detailed description.

We do not, in general, solve the LP relaxation of model (6)–(11) to optimality, but stop the column generation algorithm if the objective value changes little after some iterations. In this paper, after solving the restricted master problem in iteration n, we calculate the average values of the objectives from iteration $(n-p+1)$ to iteration n and from iteration $(n-2p+1)$ to iteration $(n-p)$, which are denoted by $AObj_1$ and $AObj_2$ respectively. If $|AObj_1 - AObj_2|/AObj_2 < \epsilon_{LP}$, then terminate the column generation algorithm. Note that p is an integer larger than 1, and ϵ_{LP} is a predetermined gap. In each iteration, the restricted master problem can be solved by using LP solvers, such as CPLEX and Gurobi.

5.2.1. Initial column subset

Note that we need a feasible solution and a corresponding initial subset of columns to start the column generation algorithm. However, finding a feasible solution to the LP relaxation of model (6)–(11) is difficult, especially when maintenance capacity is considered. In order to overcome this problem, we modify constraints (7) by introducing a dummy variable $\tilde{X}$ (which takes value between 0 and 1) with proper coefficient on the left side, and at the same time, we penalize $\tilde{X}$ in the objective function by a sufficiently large coefficient.

It is easy to find that by setting dummy variable $\tilde{X}$ to be 1 and all X_r to be 0, we obtain a feasible solution to the LP relaxation of model (6)–(11). If a feasible solution to the original model is found, dummy variable $\tilde{X}$ is equal to 0. In other words, we start with a column set which only contains the column of dummy variable $\tilde{X}$, and after inserting more and more newly generated columns and finding a feasible solution, the value of $\tilde{X}$ becomes 0.

The solution of the LP relaxation model is not necessarily integer, so the column generation algorithm is integrated into a branch-and-bound tree, which is referred to as branch-and-price algorithm. At each node of the branch-and-bound tree, a column generation algorithm is performed. However, the initial column subset at each non-root node contains more columns than the initial column set at the root node.

In Section 5.3, a detailed branching strategy is presented, and the trip sequence graph at a child node is constructed by modifying the trip sequence graph at its parent node. As a result, some trip sequences (paths) generated at the parent node become invalid at the child node. Since the modification is slight, most trip sequences are still valid at the child node. When constructing the initial column subset at the child node, the columns associated with these valid trip sequences can be inherited.

Recall that a column corresponds to a trip sequence, or a path. Denote the final column subset at the parent node as $\tilde{R}_0$, and the construction process of initial column subset at the child node is presented as follows.

- Step 1. Construct an initial column subset $\tilde{R}_1$, which only contains the column of dummy variable $\tilde{X}$. Set a threshold value $\rho \geq 0$.
- Step 2. If $\tilde{R}_0 = \emptyset$, stop. Otherwise, select column $r \in \tilde{R}_0$, set $\tilde{R}_0 = \tilde{R}_0 \setminus r$. If $\tilde{X}_r \leq \rho$, go to Step 2; otherwise, go to Step 3.
- Step 3. If column r is valid at the child node, set $\tilde{R}_1 = \tilde{R}_1 \cup r$; otherwise, go to Step 2.

In Step 2, $\tilde{X}_r$ is the solution of the variable corresponding to column r at the parent node. Since the information of the solution at the parent node is partly preserved, it is hopeful that the column generation algorithm at the child node uses less iterations and less time to find a good solution. In Step 3, the validity of a column at the child node means that the trip sequence corresponding to the column can be expressed by a path in the trip sequence graph at the child node.

5.2.2. Pricing problems

For the LP relaxation of model (6)–(11), denote the dual variables associated with constraints (7) respectively by π_w, where $w \in \mathcal{A}(t), t \in \mathcal{T}$, and the dual variables associated with constraints (9) by λ_s, where $s \in \mathcal{S}_0$. Recall that for $s \in \mathcal{S}_0$, $\mathcal{R}_s = \mathcal{R}_s^{(1)} \cup \mathcal{R}_s^{(2)} \cup \mathcal{R}_s^{(3)}$, the pricing problems can be expressed by

$$r_{s,1}^* = \arg\min_{r \in \mathcal{R}_s^{(1)}} \left(u_r - \sum_{t \in \mathcal{T}} \sum_{w \in \mathcal{A}(t)} \pi_w \cdot q_{r,w} - \lambda_s \right), \tag{12}$$

$$r_{s,2}^* = \arg\min_{r \in \mathcal{R}_s^{(2)}} \left(u_r - \sum_{t \in \mathcal{T}} \sum_{w \in \mathcal{A}(t)} \pi_w \cdot q_{r,w} - \lambda_s \right), \tag{13}$$

$$r_{s,3}^* = \arg\min_{r \in \mathcal{R}_s^{(3)}} \left(u_r - \sum_{t \in \mathcal{T}} \sum_{w \in \mathcal{A}(t)} \pi_w \cdot q_{r,w} - \lambda_s/2 \right), \tag{14}$$

where $r_{s,1}^*$ represents the trip sequence in set $\mathcal{R}_s^{(1)}$ with minimum reduced cost. The same applies to $r_{s,2}^*$ and $r_{s,3}^*$. If all the reduced costs are nonnegative, then the optimal solution of the LP relaxation of model (6)–(11) has been found in the last iteration of the column generation algorithm; otherwise, insert the columns with minimum reduced cost into the restricted master problem.

Substitute expression (5) into pricing problem (12). Fixing major station s and removing the constants, pricing problem (12) can be reformulated as:

$$r_{s,1}^* = \arg\min_{r \in \mathcal{R}_s^{(1)}} \left(\sum_{w \in \mathcal{A}_1} (\beta \cdot c_t \cdot l_w - \pi_w) \cdot q_{r,w} + \sum_{w \in \mathcal{A}_2} \beta \cdot c_t \cdot l_w \cdot q_{r,w} \right). \tag{15}$$

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

Note that problem (15) is indeed a shortest path problem from node $D^-(s)$ to node $D^+(s)$ in trip sequence graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $s \in \mathcal{S}_0$. In trip sequence graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, the weight of arc $w \in \mathcal{A}_1$ is $(\beta \cdot c_t \cdot l_w - \pi_w)$, the weight of arc $w \in \mathcal{A}_2$ is $\beta \cdot c_e \cdot l_w$ and the weights of other arcs are all 0. Since $r^*_{s,2}$ and $r^*_{s,3}$ are two-day paths, we discuss them in a two-day trip sequence graph as Fig. 7. As a result, $r^*_{s,2}$ is the shortest path from node $D_1^-(s)$ to node $D_2^+(s)$ satisfying periodic restriction (i), and $r^*_{s,3}$ is the shortest path from node $D_1^-(s)$ to node $D_2^+(s)$ satisfying periodic restriction (ii). It should be pointed out that the length of these paths must be less that the maximum travel distance. In one words, the pricing problems are shortest path problems with resource constraints in the trip sequence graph.

In this paper, we use label setting algorithm in the trip sequence graph to find the solutions to the pricing problems. A label at node u corresponds to a path from the source node to node u, which is characterized by two values, namely reduced cost and travel distance. At node $u \in \mathcal{N}$, we define $Label_u$ as the label set, $label_{u,k}$ as the k-th label in set $Label_u$, $path_{u,k}$ as the path corresponding to $label_{u,k}$, and $cost_{u,k}$ and $dist_{u,k}$ as the reduced cost and travel distance of $path_{u,k}$, respectively. We further define an order on the label set of each node. Specifically, $label_{u,k_2}$ dominates $label_{u,k_1}$, if

$$cost_{u,k_2} \le cost_{u,k_1}, \quad dist_{u,k_2} \le dist_{u,k_1},$$

and at least one of the inequalities is strict. Before performing the label setting algorithm, the nodes in the trip sequence graph are topologically sorted. The extension of the label setting algorithm follows the order of nodes after topological sorting.

In order to reduce the computation time of pricing problem (13), we simultaneously perform the label setting algorithm in two directions. Specifically, the forward extension only searches the nodes of the trip sequence graph in "day1", while the backward extension only searches nodes in "day2" in a reverse direction. We choose the best solution via combining the candidate paths obtained by the forward and backward extensions. This method is referred to as a *bidirectional label setting algorithm*.

It should be pointed out that the label setting algorithm can always find the optimal paths for pricing problem (12) and (14). Although we cannot prove that the path obtained by the bidirectional label setting algorithm is the optimal one of pricing problem (13), it is at least a good one. In Section 7.4.2, we will check the gap between the path obtained by the bidirectional label setting algorithm and the optimal path obtained by solving an ILP problem, and find that in most experiments the gap is zero or close to zero.

5.3. Branching strategy

In this paper, the branching strategy rule is motivated by the rule of *branch-on-follow-ons*, which was proposed by Ryan and Foster [27]. According to the rule of branch-on-follow-ons, two fractional paths are found, *i.e.,* r_1 and r_2, which contain a common trip, *i.e.,* t_1. Assume that in path r_1, trip t_1 is immediately followed by trip t_2, while in path r_2 it is not. Then, on the left branch, it is required that if trip t_1 is contained in a path, it must be immediately followed by trip t_2; otherwise, neither t_1 nor t_2 is contained in the path. On the right branch, it is forbidden that trip t_1 is immediately followed by t_2.

In our branching strategy, the trip sequence graph on the right branch is constructed in the same way of the rule of branch-on-follow-ons, namely removing the arc that connecting arcs of trip t_1 and arcs of trip t_2. However, due to the existence of trips needing two train units, the construction of the trip sequence graph on the left branch is different. We require that *at least one* of the train units carrying out trip t_1, immediately carries out trip t_2. For arrival/departure node u, define $n(u)$ as the number of train units needed on node u. The set of incoming arcs of node u is denoted by $\mathcal{A}^+(u)$, and the set of outgoing arcs of node u is denoted by $\mathcal{A}^-(u)$. Assume that the arrival node of trip t_1 is node u, and the departure node of trip t_2 is node v. The construction of the trip sequence graph breaks down into the following cases.

Case 1: $n(u) = 1$ and $n(v) = 1$. This is the simplest case, which is the same as the one in the rule of branch-on-follow-ons. To construct the trip sequence graph, we just remove all the outgoing arcs of node u and all the incoming of node v, and then connect node u and v directly.

Case 2: $n(u) = 2$ and $n(v) = 1$. In this case, we just remove all the incoming arcs of node v except arc (u, v), which are illustrated in Fig. 8(a).

Case 3: $n(u) = 1$ and $n(v) = 2$. In this case, we remove (1) all the incoming arcs of node p, where $p \in \mathcal{N}_d(t) \setminus \{v\}$, (2) all the outgoing arcs of node u except arc (u, v), which are illustrated in Fig. 8(b).

It is easy to find that in **Case 3**, when removing all the incoming arcs of node p, we actually determines the coupling/splitting type of trip t_2, which means that the values of decision variables Y_w related to trip t_2 are determined.

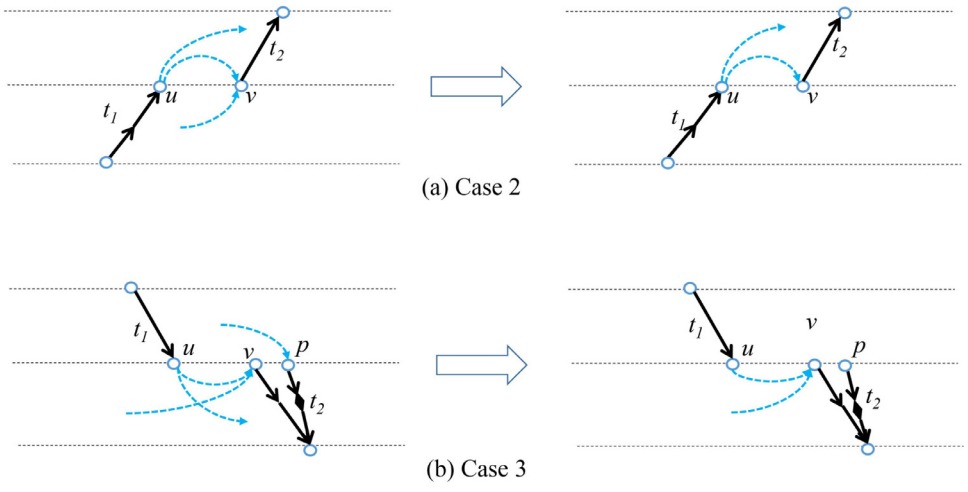

Fig. 8. Construction of trip sequence graph in Case 2 and Case 3.

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

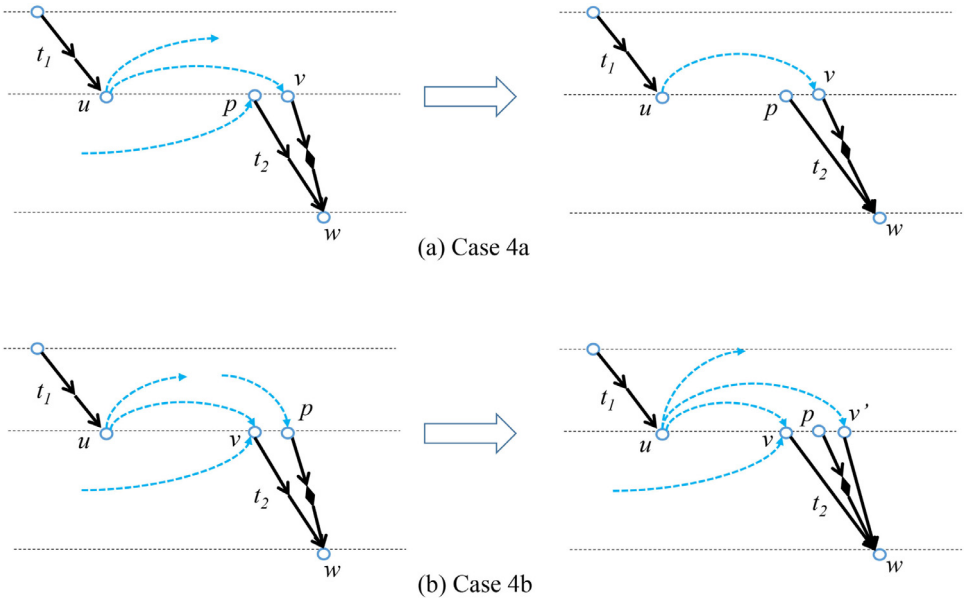

Fig. 9. Construction of trip sequence graph in Case 4.

Case 4: $n(u) = 2$ and $n(v) = 2$. Due to the existence of main arc and special arcs, this case further breaks down into the following two subcases.

(i)**Case 4a**: node v is the departure node of a special arc of trip t_2. The construction is illustrated in Fig. 9(a).

- Step 1. Remove the incoming arcs of node p, where $p \in \mathcal{N}_d(t) \setminus \{v\}$.
- Step 2. Remove the outgoing arcs of node u.
- Step 3. Add arc (u, v). Stop.

(ii)**Case 4b**: node v is the departure node of the main arc of trip t_2. The construction is illustrated in Fig. 9(b).

- Step 1. Remove the incoming arcs of node p, where $p \in \mathcal{N}_d(t) \setminus \{v\}$.
- Step 2. Add a new node v' to the network, and add arcs (v', w) and (u, v').
- Step 3. Set $n(v) = 1$ and $n(v') = 1$. Stop.

Note that in **Case 4b**, a new node and two new arcs are added to the trip sequence graph, so we must perform topological sorting again on the new node set for the pricing problems. As in **Case 3**, when removing all the incoming arcs of node p, we determines the coupling/splitting type of trip t_2, and accordingly the values of decision variables Y_w related to trip t_2.

5.4. Node search strategy

A depth-first search on the branch-and-bound tree is used. In most experiments, we find that once an integer solution is found, the corresponding objective value is very close to the optimal objective value of the LP relaxation model at the root node. In other words, the lower bound provided by the LP relaxation model at the root node is quite tight.

A recursive structure is used when implementing the branch-and-bound algorithm. The recursive function is named "BRANCH-AND-BOUND", which returns a node index of the tree. The pseudo-code of function "BRANCH-AND-BOUND" is presented in Algorithm 1. In line 1 of Algorithm 1, "X_{Best}" represents the current best integer solution, "F_{Best}" represents the corresponding objective value, "*ParentGraph*" means the trip sequence graph at the parent node, and "X_{Pnt}" represents the optimal solution at the parent node, which is fractional and corresponds to objective value "F_{Pnt}".

In line 3 of Algorithm 1, the trip sequence graph at the left node, namely "*LeftGraph*", is constructed according to Construc-

Algorithm 1 A branch-and-bound algorithm with rollback.

```
1: function BRANCH-AND-BOUND(X_Best, F_Best, ParentGraph, X_Pnt)
2:     // Left branch
3:     LeftGraph = Construct_Graph(ParentGraph, X_Pnt, Left);
4:     [X_Left, F_Left] = Column_Generation(LeftGraph);
5:     LeftNode = Construct_Node(LeftGraph, X_Left, F_Left);
6:     if X_Left if not integer then
7:         if |F_Left − F_Best|/F_Best ≥ ε_BB then // gap check
8:             Index =BRANCH-AND-BOUND(X_Best,F_Best,LeftGraph,X_Left);
   // recurrence
9:             if Index ≠ LeftNode_Index then
10:                return Index;
11:            end if
12:        end if
13:    else // the solution is integer
14:        if F_Left < F_Best then // a better integer solution is found
15:            X_Best ← X_Left;
16:            F_Best ← F_Left; // update
17:
18:            Pnt ← LeftNode.parent_Index;
19:            Index ← LeftNode_Index;
20:            while |F_Left − F_Pnt|/F_Pnt < ε_BB do // rollback
21:                Index ← Pnt;
22:                Pnt ← Pnt.parent_Index;
23:            end while
24:            if Index ≠ LeftNode_Index then
25:                return Index;
26:            end if
27:        end if
28:    end if
29:
30:    // Right branch
31:    RightGraph = Construct_Graph(ParentGraph, X_Pnt, Right);
32:    ...
33: end function
```

tion 1. In line 4, the column generation algorithm is implemented on the left trip sequence graph, and the corresponding solution is "X_{Left}" with objective value "F_{Left}". From line 6 to line 12, we deal with the situation of a fractional solution. Note that only when the gap between "F_{Left}" and "F_{Best}" is large enough, we implement further branching on "X_{Left}". From line 13 to line 28, we deal with the situation that a better integer solution than the currently best one is found. In order to speed up the algorithm, a rollback procedure is used. We go upward along the tree until the gap between "F_{Left}" and the objective value at a predecessor node is not smaller than ϵ_{BB}. This procedure is referred to as *rollback procedure* and is presented from line 18 to line 26. Obviously, the rollback procedure may prune a considerable numbers of nodes in the tree.

Note that in Algorithm 1, only the left branch is presented in detail. On the right branch, the same procedure is performed on the right trip sequence graph, namely, "*RightGraph*".

6. Arc-based model

In this section, we formulate an arc-based model for the trip sequence planning problem, which is also on the trip sequence graph. When the problem is of a small scale, the arc-based model can provide an optimal solution in an acceptable computation time.

Assume that $\mathcal{R}_s$ is the set of train units in major station $s \in \mathcal{S}_0$. For $r \in \mathcal{R}_s$, we define decision variables $x_r^{(i)}$ as follows

$x_r^{(i)}$ = 1, if train unit r is used to carry out trip sequence of type i; =0, otherwise.

Note that type $i \in \{1, 2, 3\}$, which respectively represent one-day trip sequence, two-day trip sequence with periodic restriction (i) and two-day trip sequence with periodic restriction (ii).

For type $i \in \{1, 3\}$, we further define decision variables $z_{r,w}^{(i)}$ to explicitly characterise the path (trip sequence) of train unit r as follows

$z_{r,w}^{(i)}$ = 1, if arc $w \in \mathcal{A}$ is contained in the path associated with $x_r^{(i)}$;
= 0, otherwise.

For type $i = 2$, we define decision variables $z_{r,w}^{(2,j)}$,

$z_{r,w}^{(2,j)}$ = 1, if arc $w \in \mathcal{A}$ is contained in the "day j" part of the path associated with $x_r^{(2)}$;
= 0, otherwise.

where $j \in \{1, 2\}$ representing "day1" and "day2" parts of two-day trip sequence r with periodic restriction (i).

Obviously, in the arc-based model, $x_r^{(i)}$ only serves as an indicator, while $z_{r,w}^{(i)}$ and $z_{r,w}^{(2,j)}$ are the more fundamental part. Without loss of generality, in this paper, we assume that $|\mathcal{R}_s| \le M_s$, where M_s is the maintenance capacity in station s.

As in the path-base model, we define binary decision variables Y_w as follows,

Y_w = 1 if arc $w \in \mathcal{A}(t), t \in \mathcal{T}$ is selected; = 0, otherwise.

The objective function is the same to that in the path-based model, namely expression (4), except that $q_{r,w}$ is replaced by $z_{r,w}^{(i)}$ and $z_{r,w}^{(2,j)}$. The arc-based model can be formulated as follows:

$$\min_X \quad F$$

$$s.t.$$

$$\sum_{s\in\mathcal{S}_0}\sum_{r\in\mathcal{R}_s}\left(z_{r,w}^{(1)} + (z_{r,w}^{(2,1)} + z_{r,w}^{(2,2)}) + z_{r,w}^{(3)}\right) = Y_w N_w, \quad \forall w \in \mathcal{A}(t), t \in \mathcal{T}, \tag{16}$$

$$\sum_{w\in\mathcal{A}(t)} Y_w = 1, \quad \forall t \in \mathcal{T} \tag{17}$$

$$\sum_{r\in\mathcal{R}_s} (x_r^{(1)} + x_r^{(2)} + x_r^{(3)}/2) \le M_s, \quad \forall s \in \mathcal{S}_0 \tag{18}$$

$$\sum_{w\in\mathcal{A}_1\cup\mathcal{A}_2} z_{r,w}^{(1)} \cdot l_w \le x_r^{(1)} \cdot L, \quad \forall s \in \mathcal{S}_0, r \in \mathcal{R}_s, \tag{19}$$

$$\sum_{w\in\mathcal{A}_1\cup\mathcal{A}_2} (z_{r,w}^{(2,1)} + z_{r,w}^{(2,2)}) \cdot l_w \le x_r^{(2)} \cdot L, \quad \forall s \in \mathcal{S}_0, r \in \mathcal{R}_s, \tag{20}$$

$$\sum_{w\in\mathcal{A}_1\cup\mathcal{A}_2} z_{r,w}^{(3)} \cdot l_w \le x_r^{(3)} \cdot L/2, \quad \forall s \in \mathcal{S}_0, r \in \mathcal{R}_s, \tag{21}$$

Flow conservation constratins (22)

$$z_{r,w}^{(2,1)} + z_{r,w}^{(2,2)} \le 1, \quad \forall s \in \mathcal{S}_0, r \in \mathcal{R}_s, t \in \mathcal{T} \tag{23}$$

$$x_r^{(1)} + x_r^{(2)} + x_r^{(3)} \le 1, \quad \forall s \in \mathcal{S}_0, r \in \mathcal{R}_s \tag{24}$$

All decision variables are binary. (25)

The above constraints are all linear, which means that we formulate an integer linear programming (ILP) model. Constraints (16)–(18) play the same roles as constraints (7)–(9) in the path-based model, namely coverage, selection and capacity constraints. Constraints (19)–(21) ensure that the total travel distance of a trip sequence cannot be greater than the maximum travel distance L. These constraints also imply that if $x_r^{(i)} = 0$, then $z_{r,w}^{(i)} = 0$ or $z_{r,w}^{(i,j)} = 0$.

Constraints (22) are the flow conservation constraints. Constraints (23) ensure the periodicity restriction (i) of trip sequences of type 2. Constraints (24) state that trip sequence r can be at most one of the three types of trip sequence.

7. Case study

In this section, we test our models and methods on real data from the Chinese high-speed railway. The numerical experiments are all performed on a computer with Intel(R) Core(TM) i5-4570S 2.90 GHz CPU and 8.00GB RAM, running on a Microsoft Windows 7(64bit) platform, and all the programming is implemented in C++ language on Microsoft Visual Studio 2015. The LP models are solved to optimality by Gurobi 6.0.5, which is a state of the art optimization solver.

7.1. Case descriptions

We consider three subnetworks from the eastern Chinese high-speed railway network, which connects two most developed regions in China, namely the Jingjinji region and the Yangtze Delta region. Ever since being opened to the public, the eastern network has been the busiest high-speed railway network in the world.

The sizes of the three subnetworks are increasing. Network I, which is the Shanghai-Hangzhou-Ningbo (SHN) corridor, consists of 15 stations and 2 train bases; Network II consists of 67 stations and 5 train bases; Network III consists of 83 stations and 7 train bases. Based on these networks, we construct 8 cases, *i.e.*, 2 medium cases ("MC") and 6 large cases ("LC"). The high-speed trips in these cases are all selected from the actual timetable. Note that some trips travel out of the considered network. When constructing the cases, we cut these trips at boundary stations, and treat the boundary stations as origin and/or destination stations of the trips.

The characteristics of the cases are listed in Table 2. In the table, "Network" lists the physical railway networks that the corresponding case is constructed on, "Length" is the total length of the high-speed railway in the network, "$|\mathcal{S}|$", "$|\mathcal{S}_0|$" and "$|\mathcal{T}|$" respectively represent the number of all stations, the number of major stations and the number of trips. Define $\mathcal{T}_2$ as the set of trips that need two train units, and accordingly "$|\mathcal{T}_2|$" is the number of trips

Table 2
Characteristics of the cases.

ID	Network	Length [km]	$\|\mathcal{S}\|$	$\|\mathcal{S}_0\|$	$\|\mathcal{T}\|$	$\|\mathcal{T}_2\|$
MC1	I	350	15	2	90	43
MC2	I	350	15	2	182	81
LC1a	II	2400	67	5	390	0
LC1b	II	2400	67	5	390	170
LC2a	II	2400	67	5	784	0
LC2b	II	2400	67	5	784	318
LC3a	III	3600	83	7	902	0
LC3b	III	3600	83	7	902	368

that need two train units. Note that we cannot obtain the real data of the number of train units that each trip needs. We first obtain a feasible trip sequence plan for each case with $\mathcal{T}_2 = \emptyset$, and then select some trip sequences, based on which we construct set $\mathcal{T}_2$. In cases "LC1a" and "LC2a", all trips need only one train unit, that is, $\mathcal{T}_2 = \emptyset$. In other cases, $\mathcal{T}_2$ is not empty, and the ratio $|\mathcal{T}_2| : |\mathcal{T}|$ is between 40% and 50%.

Throughout the numerical experiments, the extra times of coupling and splitting are both 5 minutes, and the minimum shunting time without coupling or splitting is 40 minutes. That is, we ignore the effect of platform assignment on shunting time, because we do not have these data. It should be pointed out that this simplicity does not affect the experiments. In objective function (4), we set time horizon $H = 200$, train unit price $P_{tu} = 5000$, the cost of one maintenance $m = 1$ and the travel cost per kilometer $c_t = 0.001$. As a result, we have $\beta = H/P_{tu} = 0.04$. In the numerical experiments of the large scale cases, the maximum travel distance between two consecutive Level I maintenances is set to be 4,000 km, *i.e.*, $L = 4000$, while in the medium cases, the value of L varies for different experiments.

7.2. Medium scale case

In this section, we apply the arc-based model and the path-based model to cases "MC1" and "MC2". In the branch-and-price algorithm, we set $\epsilon_{LP} = 0.01\%$ and $\epsilon_{BB} = 1\%$.

Note that in case "MC1", the two major stations are Shanghai and Hangzhou, the maintenance capacities of which are set to be 20 and 10, respectively. For the arc-based model, we set $|\mathcal{R}_s| = 20$ in Shanghai station and $|\mathcal{R}_s| = 10$ in Hangzhou station. Table 3 lists the computational results, when maximum travel distance L takes different values.

In Table 3, the column of "F_a/F_p" lists the objective values obtained by the arc-based/path-based model, "CT" is the computation time, "N" records the number of searched nodes in the branch-and-bound tree, and "F-N" and "B-N" respectively records the index of nodes in the tree where the first feasible solution and the best solution are found. Besides, "Δ_1" is the relative gap between F_p and the objective value of the optimal solution, and "Δ_2" is the relative gap between F_p and the objective value at the root node of the branch-and-bound tree, which is regarded as a lower bound of the optimal solution.

When $L \in \{4000, 2400, 2000\}$, the arc-based model can obtain an optimal solution in an acceptable time, however, the computation time rapidly increases as L decreases. Compared with the arc-based model, the computation time of the path-based model is much shorter and more stable, which is affected little by L. As column "F_p" shows, in all the three experiments, the optimal solution is obtained by the branch-and-price algorithm, which means $\Delta_1 = 0$. We also calculate Δ_2, which measures the relative gap between the objective value of current solution and the lower bound. In all the three experiments, we have $\Delta_2 = 0$. This indicates that the lower bound provided by the LP relaxation model at the root node is very tight. Column "F-N" shows that the first feasible solution can be found quite early, however, it is not so good and the branch-and-price algorithm continues until a good enough solution is found.

In case "MC2", we set the capacities of Shanghai station and Hangzhou station to be 30 and 20 respectively. Table 4 lists the computational results, when $L \in \{4000, 2400, 2000\}$. The time limit of the IP solver is set to be 7200 s. As is shown in Table 4, when $L = 4000$, the arc-based model can be solved to optimality in acceptable time. However, when $L \in \{2000, 2400\}$, the IP solver cannot even find a feasible solution in 2 hours. The branch-and-price algorithm performs much better, which solves the problem to optimality within a gap of 0.10% in short time. Note that when $L \in \{2000, 2400\}$, we do not know the optimal solution, and the value of Δ_1 cannot be calculated. However, since a feasible solution in the last two experiments is always a feasible one in the first experiment, which means that the objective values of the optimal solution in the last two experiments are not smaller than in the first experiment. As a result, we can estimate Δ_1, *i.e.*, $\Delta_1 \leq (F_p - 61.26)/61.26$.

As in Table 4, columns "N", "F-N" and "B-N" show that in each experiment, a feasible solution can be found in a quite early node, however, the algorithm continues until a good enough solution is found. In Table 3, the total computation time decreases as the maximum travel distance L decreases. It is possible because as L decreases, the number of feasible trip sequences decreases, and accordingly the solution space is reduced.

7.3. Large scale case

In the large cases "LC1a/b", "LC2a/b" and "LC3a/b", we set the maintenance capacities of the train bases to be {40, 20, 30, 40, 20}, {60, 30, 50, 60, 30} and {50, 35, 40, 60, 30, 50, 20} respectively. If the arc-based model is applied in case "LC1a", the numbers of constraints and binary decision variables are both more than 300,000. It is impossible to solved the arc-based model of case "LC1a" in acceptable time, not to mention other larger cases. In this section, only the path-based model and the branch-and-price algorithm are used.

Setting $\epsilon_{LP} = 0.01\%$ and $\epsilon_{BB} = 1.0\%$, Table 5 lists the experiment results. Note that the time limit of the branch-and-price algorithm is 14400*s*. Columns "F_p" lists the objective values of the experiments. In columns "CT", the total computation time and the total time of solving pricing problems are respectively listed in columns "Total" and "PP". In Table 5, "N", "F-N", "B-N" and "Δ_2" have the same meaning with Table 3.

We cannot obtain an optimal solution by solving the arc-based model, so Table 5 does not list the value of Δ_1, which measures

Table 3
Comparison between arc-based model and path-based model in case "MC1".

L[km]	Arc-based model		Path-based model						
	F_a	CT[s]	F_p	Δ_1	Δ_2	CT[s]	N	F-N	B-N
4000	37.80	190	37.80	0.00%	0.00%	20	84	80	84
2400	37.80	210	37.80	0.00%	0.00%	23	115	72	115
2000	37.80	510	37.80	0.00%	0.00%	23	111	81	111

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

Table 4
Comparison between arc-based model and path-based model in case "MC2".

L[km]	Arc-based model		Path-based model						
	F_a	CT[s]	F_p	Δ_1	Δ_2	CT[s]	N	F-N	B-N
4000	61.26	6930	61.29	0.05%	0.05%	320	650	192	650
2400	–	7200	61.32	$<0.10\%$	0.05%	340	741	220	741
2000	–	7200	61.34	$<0.13\%$	0.12%	180	419	205	419

Table 5
Results of the branch-and-price algorithm.

ID	F_p	CT[s]		N	F-N	B-N	Δ_2
		Total	PP				
LC1a	131.59	420	55	194	194	194	0.05%
LC1b	191.38	1100	160	605	288	605	0.10%
LC2a	237.69	4950	580	388	388	388	0.15%
LC2b	341.87*	14400	1880	2250	799	2074	1.15%
LC3a	284.45	4210	450	469	469	469	0.16%
LC3b	402.02*	14400	2130	2203	943	2051	1.98%

* The best solution obtained within time limit 4 h

the relative gap between the best obtained solution and the optimal solution. However, the values of Δ_2 indicate that the solutions obtained by the branch-and-price algorithm are of good quality. In cases "LC2b" and "LC3b", the branch-and-price algorithm stops due to the time limit. In the former, the best solution is found at node 2074 with $\Delta_2 = 1.15\%$; in the latter, the best solution is found at node 2051 with $\Delta_2 = 1.98\%$.

In cases "LC1a", "LC2a" and "LC3a", the branch-and-price algorithm stops once the first feasible solution is found. It is interesting to find that in these three cases, each trip needs only one train unit, and the solution quality is much better than in the other two cases. Recall that in path-base model (6)-(11), if each trip needs only one train unit, then there is no coupling or splitting, and the value of Y_w is actually predetermined by constraints (8), *i.e.*, $Y_w = 1$. As a result, the model and the solution space both become much simpler.

It is easy to find that in all experiments, the pricing problem computation time occupies less than 15% of the total computation time, *i.e.*, {13.1%, 14.5%, 11.7%, 13.1%, 10.7%, 14.8%}. This indicates that solving the LP problems occupies most computation time.

Recall that we do not solve the LP relaxed restricted master problems to optimality, but terminate the column generation algorithm following the rule stated in Section 5.2. In the following, we implement the column generation algorithm to solve the restricted master problems at the root node with different ϵ_{LP}, and show that $\epsilon_{LP} = 0.01\%$ is good enough for our cases. The computation results are presented in Table 6, in which ϵ_{LP} is set to be 0.1%, 0.01% and 0.0001%, respectively. In the table, "F" represents the objective value of the LP relaxation model at the root node, "CT" represents computation time, and "#IT" records the number of iteration when the column generation algorithm stops.

As is shown in Table 6, the improvement of the objective values from "$\epsilon_{LP} = 0.1\%$" to "$\epsilon_{LP} = 0.01\%$" is more significant than the improvement from "$\epsilon_{LP} = 0.01\%$" to "$\epsilon_{LP} = 0.0001\%$". Actually, only in case "LC2a", the objective value is improved when $\epsilon_{LP} = 0.0001\%$, and the improvement is very tiny. This indicates that when $\epsilon_{LP} = 0.01\%$, the solution obtained by the column generation algorithm is very close to optimality.

The computation times increase significantly from "$\epsilon_{LP} = 0.1\%$" to "$\epsilon_{LP} = 0.01\%$". This is because as more columns are inserted, the restricted master problem becomes larger, leading to increasing computation time in each iteration. Although the computation times are much shorter when $\epsilon_{LP} = 0.1\%$, the solutions are not of good quality. For example, in case "LC2a", the objective value of the LP solution is $F = 238.22$ when $\epsilon_{LP} = 0.1\%$. However, the objective value of the IP solution found by the branch-and-price algorithm is $F_p = 237.69$ when $\epsilon_{LP} = 0.01\%$ (see Fig. 5). If we set $\epsilon_{LP} = 0.1\%$, it is unlikely to find an IP solution with objective value less than 238.00. So, it is worthy to set $\epsilon_{LP} = 0.01\%$.

In Table 6, it seems that at each node of the branch-and-bound tree, the column generation algorithm costs a long time. However, thanks to the construction method of the initial column subset stated in Section 5.2.1, at the non-root node, the computation times are quite short. Table 7 shows the effectiveness of the construction method of the initial column subset. Recall that at the root node, the initial column subset only contains a column for the dummy variable, while at each non-root node, the initial column subset inherits some columns from its parent node. In Table 7, "CT[s]", "#IT" and "#IF" represents computation time, the number of iteration when the column generation algorithm stops and the number of iteration when the first feasible LP solution is found, respectively. Columns with "avg.non-root" list the average values of "CT[s]", "#IT" and "#IF" at non-root nodes.

Obviously, the column generation algorithm at the non-root nodes performs much more effectively than at the root node. The last column indicates that at the non-root nodes, the column generation algorithm finds the first feasible LP solution after a few iterations. Besides, the first feasible LP solution is of good quality, since the algorithm stops after less than 40 iterations in average. In contrast, at the root node, the algorithm iterates more than 100 times after finding the first feasible LP solution in the first two cases, and more than 200 times in the last two cases.

7.4. Additional experiments

7.4.1. IP problem at the root node

In some column-generation-based algorithms, a good integer solution can be obtained by solving an integer programming (IP)

Table 6
Column generation algorithm at the root node with different ϵ_{LP}.

ID	$\epsilon_{LP} = 0.1\%$			$\epsilon_{LP} = 0.01\%$			$\epsilon_{LP} = 0.0001\%$		
	F	CT[s]	#IT	F	CT[s]	#IT	F	CT[s]	#IT
LC1a	131.70	23	304	131.53	40	344	131.53	46	357
LC1b	190.49	24	299	190.18	45	341	190.18	52	351
LC2a	238.22	150	663	237.33	330	777	237.32	377	805
LC2b	338.77	210	672	337.95	380	775	337.95	407	790
LC3a	284.67	340	685	283.99	460	753	283.92	510	787
LC3b	394.97	370	712	394.20	650	812	394.20	713	817

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

Table 7
Effectiveness of the construction method of the initial column subset.

ID	CT[s]		#IT		#IF	
	Root	Avg.non-root	Root	Avg.non-root	Root	Avg.non-root
LC1a	40	1.87	344	36.2	236	8.0
LC1b	45	1.66	341	34.3	226	8.4
LC2a	330	13.67	777	53.1	556	16.1
LC2b	380	7.55	775	53.2	510	16.2
LC3a	460	8.03	753	36.4	589	13.4
LC3b	650	6.35	812	37.8	602	14.7

problem at the root node. Specifically, after solving the restricted master problem at the root node, a large number of newly generated columns are obtained; insert these columns into the original IP model (6)–(11), and then solve this IP problem by an IP solver directly. In other word, the branching procedure presented in Section 5.3 is not needed. We carry out two experiments to verify whether a good integer solution can be obtained at the root node in this way.

The first experiment is on case "MC2" with the same parameters in Section 7.2, except that we set $\epsilon_{LP} = 10^{-9}$. After 326 iterations, the column generation algorithm stops, with 1776 new columns generated. Inserting these columns into model (6)–(11), we obtain an IP problem. We use Gurobi to solve this IP problem, however, this IP model is infeasible.

The second experiment is on case "LC2a" with $\epsilon_{LP} = 10^{-9}$. After 1580 iterations, the column generation algorithm stops, with over 15000 new columns generated. The IP solver cannot find a feasible integer solution for this IP problem in 24 h, even if the maintenance capacity constraints are removed.

The results of these two experiments indicate that it is not a good way to deal with the path-based model by solving an IP problem at the root node. The main reason is that the coverage constraints (7) and (8) in the path-based model are too strict.

7.4.2. Effectiveness of the bidirectional label setting algorithm

As pointed out in Section 5.2.2, the bidirectional label setting algorithm (*Bi-LC algorithm*) does not ensure the optimality of the solutions for pricing problem (13), which correspond to two-day trip sequences with periodicity restriction (i). In this section, we investigate the gaps between the paths obtained by the Bi-LC algorithm and the corresponding optimal ones. To obtain an optimal solution, we first construct an IP model for pricing problem (13), and then solve the model to optimality by Gurobi directly, which is referred to as *IP method*.

We focus on the pricing problems at the root node of case "LC2a". Setting $\epsilon_{LP} = 0.01\%$, the column-generation algorithm stops after 777 iterations at the root node, and we record the solutions and the computation times of the pricing problems in all iterations. Since there are 5 major stations in the eastern network, we find 5 paths for pricing problem (13) in each iteration, which indicates that we have 3,885 samples in total to check the effectiveness of the Bi-LC algorithm.

First, we compare the computation times between the IP method and the Bi-LC algorithm. Note that the number of nodes in the trip sequence graph is 1600, and the number of arcs is 33611. Table 8 presents the computation times of the pricing problems at the five train bases, when the IP method or the Bi-LC algorithm is used. In the table, "Avg", "Min" and "Max" represent the average, minimum and maximum computation times, respectively. Obviously, the Bi-LC algorithm uses much less time to find a solution. For all the 3,885 samples, if the IP method is used, the average computation time of a pricing problem is 0.534s, while if the Bi-LC algorithm is used, the average time is 0.0034s, which is only 0.64% of the former.

Table 8
Comparison of computation times.

Base	IP method			Bi-LC algorithm		
	Avg	Min	Max	Avg	Min	Max
1	0.485	0.286	0.697	0.002	0.001	0.018
2	0.442	0.323	0.861	0.003	0.001	0.016
3	0.583	0.232	1.876	0.004	0.001	0.023
4	0.598	0.262	1.375	0.004	0.001	0.029
5	0.563	0.323	2.008	0.004	0.001	0.018

Table 9
Solution quality of the bidirectional label setting algorithm.

Gap	0	0.01%	0.1%	0.5%	1%	5%	10%
Number	2949	3050	3275	3489	3547	3831	3862
Percent	75.9%	78.5%	84.3%	89.8%	91.3%	98.6%	99.4%

Next, we check the quality of the solutions obtained by the Bi-LC algorithm. Table 9 presents the statistics on the gaps between the Bi-LC algorithm and the IP method. The second row lists the number of solutions whose gaps are less than the corresponding values in the row of "Gap", and the third row lists corresponding percents in the 3,885 pricing problems.

As shown in Table 9, more than 75% solutions obtained by the Bi-LC algorithm are optimal, nearly 90% solutions have a gap less than 0.5%, and more than 99% solutions have a gap less than 10%. This indicates that solutions obtained by the Bi-LC algorithm are of good quality. Considering the computation time discussed above, the Bi-LC algorithm is good enough in both effectiveness and efficiency.

8. Conclusions and future research

Trip sequence planning occupies an important place in the train units scheduling problem of the Chinese high-speed railway network. Taking into account maintenances, we develop two integer programming models for the trip sequence planning problem, the objective of which is to minimize a weighted sum of train units cost, maintenance cost and travel cost. To model the problem, we propose a concept of trip sequence graph, which conveniently expresses the movement and coupling/splitting of train units in high-speed railway network with maintenances. Then, two integer linear programming model, namely a path-based model and an arc-based model, are constructed, based on the trip sequence graph. The arc-based model can be solved by commercial optimization solvers directly, while a customized branch-and-price algorithm is developed to solve the path-based model. A bidirectional label setting algorithm is proposed to efficiently solve the pricing problems, and a rollback procedure is further used to speed up the branch-and-price algorithm.

In order to test the efficiency and applicability, the two models are applied to 8 cases of different sizes in the Chinese high-speed railway network. In the medium-scale cases, the experimental results show that the arc-based model can be solved to optimality within acceptable computation time, when the maintenance constraints are not tight. However, the computation time of the arc-based model increases drastically when the maintenance constraints become tight, while the branch-and-price algorithm performs quite stable. In the large-cases, the arc-based model does not work, while the path-based model can be solved by the branch-and-price algorithm in acceptable time. Finally, some additional experiments are carried out to test the effectiveness of the bidirectional label setting algorithm.

In the future, we may extend the model in the following three aspects. First, in this paper we did not consider the types of train

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150

units. Although in these days, the types of train units are determined before trip sequence plan, it is both of theoretical and practical significance to construct an integrated model to assign train unit type and make trip sequence plan together. To integrate this decision, it may be useful to extend the trip sequence graph, and to consider more details on costs. Second, we may further extend the integrated model by considering the assignment of individual train units. In this integrated model, the state of each train unit and maintenances of long terms should be considered, which makes the model much larger. As a result, heuristics approaches may be needed to solve the model. Third, the study of the robustness of a trip sequence plan is also a challenging research topic. A trip sequence plan with higher robustness may remain feasible after a disturbance, or even a disruption. It is meaningful to find a way to measure the robustness of a trip sequence plan, and then propose a robustness model with efficient solving method.

Acknowledgments

We appreciate the editors and the anonymous reviewers for their valuable questions and suggestions. The research was supported by the National Natural Science Foundation of China (nos. 71771016, 71571018, 71621001 and 71825004), and State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University (no. RCS2019ZT002).

References

[1] Abbink E, Berg van den B, Kroon L, Salomon M. Allocation of train units to passenger trains. Transp Sci 2004;38(1):33–41.
[2] Alfieri A, Groot R, Kroon L, Schrijver A. Efficient circulation of railway rolling stock. Transp Sci 2006;40(3):378–91.
[3] Barnhart C, Johnson E, Nemhauser G, Savelsbergh M, Vance P. Branch-and-price: column generation for solving huge integer programs. Oper Res 1998a;46(3):316–29.
[4] Barnhart C, Boland NL, Clarke LW, Johnson EL, Nemhauser GL, Shenoi RG. Flight string models for aircraft fleeting and routing. Transp Sci 1998b;32(3):208–20.
[5] Borndörfer R, Reuther M, Schlechte T, Waas K, Weider S. Integrated optimization of rolling stock rotations for intercity railways. Transp Sci 2016;50(3):863–77.
[6] Cacchiani V, Caprara A, Toth P. Solving a real-world train-unit assignment problem. Math Program 2010;124(1–2):207–31.
[7] Cacchiani V, Caprara A, Galli L, Kroon L, Maróti G, Toth P. Railway rolling stock planning: robustness against large disruptions. Transp Sci 2012;46(2):217–32.
[8] Clarke LW, Johnson EL, Nemhauser GL, Zhu Z. The aircraft rotation problem. Ann Oper Res 1997;69(0):33–46.
[9] Cacchiani V, Caprara A, Toth P. A lagrangian heuristic for a train-unit assignment problem. Discrete Appl Math 2013;161:1707–18.
[10] Cacchiani V, Caprara A, Toth P. An effective peak period heuristic for railway rolling stock planning. In: Transportation Science; 2019.
[11] Cacchiani V, Salazar-González JJ. Heuristic approaches for flight retiming in an integrated airline scheduling problem of a regional carrier. Omega 2019.
[12] Cordeau JF, Soumis F, Desrosiers J. A benders decomposition approach for the locomotive and car assignment problem. Transp Sci 2000;34:133–49.
[13] Cordeau JF, Soumis F, Desrosiers J. Simultaneous assignment of locomotives and cars to passenger trains. Oper Res 2001;49(4):531–48.
[14] Desaulniers G, Desrosiers J, Solomon MM. Column generation. Boston, MA: Springer; 2005.
[15] Fioole PJ, Kroon L, Maróti G, Schrijver A. A rolling stock circulation model for combining and splitting of passenger trains. Eur J Oper Res 2006;174:1281–97.
[16] Haouari M, Aissaoui N, Mansour FZ. Network flow-based approaches for integrated aircraft fleeting and routing. Eur J Oper Res 2009;193(2):591–9.
[17] Hong SP, Kim KM, Lee K, Park BH. Apragmatic algorithm for the train-set routing: the case of korea high-speed railway. Omega: Int J Manage Sci 2009;37:637–45.
[18] Liang Z, Chaovalitwongse WA, Huang HC, Johnson EL. On a new rotation tour network model for aircraft maintenance routing problem. Transp Sci 2011;45(1):109–20.
[19] Liang Z, Chaovalitwongse WA. A network-based model for the integrated weekly aircraft maintenance routing and fleet assignment problem. Transp Sci 2013;47(4):493–507.
[20] Lin Z, Kwan RSK. A branch-and-price approach for solving the train unit scheduling problem. Transp Res Part B 2016;94:97–120.
[21] Lingaya N, Cordeau JF, Desaulniers G, Desrosiers J, Soumis F. Operational car assignment at VIA rail canada. Transp Res Part B 2002;36:755–78.
[22] Liu R, Li S, Yang L. Collaborative optimization for metro train scheduling and train connections combined with passenger flow control strategy. Omega 2019.
[23] Maróti G, Kroon L. Maintenance routing for train units: the transition model. Transp Sci 2005;39(4):518–25.
[24] Maróti G, Kroon L. Maintenance routing for train units: the interchange model. Comput Oper Res 2007;34:1121–40.
[25] Mercier A, Soumis F. An integrated aircraft routing, crew scheduling and flight retiming model. Comput Oper Res 2007;34(8):2251–65.
[26] Peeters M, Kroon L. Circulation of railway rolling stock: a branch-and-price approach. Comput Oper Res 2008;35:538–56.
[27] Ryan D, Foster B. An integer programming approach to scheduling. In: Wren A, editor. Computer Scheduling of Public Transport: Urban Passenger Vehicle and Crew Scheduling; 1981. North Holland, Amsterdam
[28] Salazar-González JJ. Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. Omega 2014;43:71–82.
[29] Schrijver A. Minimum circulation of railway stock. CWI Q 1993;6(3):205–17.
[30] Tönissen D, Arts J, Shen ZJ. Maintenance location routing for rolling stock under line and fleet planning uncertainty. In: Transportation Science, accepted; 2018.
[31] Wang Y, D'Ariano A, Yin J, Meng L, Tang T, Ning B. Passenger demand oriented train scheduling and rolling stock circulation planning for an urban rail transit line. In: Transportation Research Part B: Methodological, vol. 118; 2018. p. 193–227.
[32] Ziarati K, Soumis F, Desrosiers J, Solomon MM. A branch-first, cut-second approach for locomotive assignment. Manage Sci 1999;45:1156–68.

Please cite this article as: Y. Gao, M. Schmidt and L. Yang et al., A branch-and-price approach for trip sequence planning of high-speed train units, Omega, https://doi.org/10.1016/j.omega.2019.102150