Solving a Real-World Train Unit Assignment Problem

Valentina Cacchiani, Alberto Caprara, and Paolo Toth

DEIS, University of Bologna, Viale Risorgimento 2, I-40136 Bologna, Italy,
valentina.cacchiani@unibo.it,acaprara@deis.unibo.it,paolo.toth@unibo.it

Abstract. We face a real-world train unit assignment problem for an operator running trains in a regional area. Given a set of timetabled train trips, each with a required number of passenger seats, and a set of train units, each with a given number of available seats, the problem calls for an assignment of the train units to trips, possibly combining more than one train unit for a given trip, that fulfills the seat requests. With respect to analogous case studies previously faced in the literature, ours is characterized by the fairly large number of distinct train unit types available (in addition to the fairly large number of trips to be covered). As a result, although there is a wide margin of improvement over the solution used by the practitioners (as our results show), even only finding a solution of the same value is challenging in practice. We present a successful approach, based on an ILP formulation in which the seat requirement constraints are stated in a "strengthened" form derived from the description of the convex hull of the variant of the knapsack polytope arising when the sum of the variables is restricted not to exceed two, illustrating computational results on our case study.

1 Introduction

The assignment of locomotives and cars, generally referred to as *rolling stock*, to trains with published timetables is a key problem to be faced by operators of passenger trains, given that the acquisition of rolling stock is an expensive long-term investment, and that fulfilling the passenger requests, namely guaranteeing (within reasonable margins) that each passenger has a seat, is fundamental to ensure customer satisfaction. In this paper, we illustrate how we solved a real-world case of the problem for the trains operated by a passenger train operator operating in a regional area. In this problem, so-called *Train Units* (TUs), rather than locomotives and cars, have to be assigned to trains. A TU is a self-contained train with an engine and passenger seats, and TUs can be combined together to increase the number of available seats.

The large number of TU types, along with the fairly large number of train trips to be covered, namely a few hundred, make our case study very challenging from an optimization viewpoint. In particular, although unavoidably the mathematical programming models that one may consider are analogous to those

ATMOS 2007 (p.79-95)
7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems
http://drops.dagstuhl.de/opus/volltexte/2007/1172

presented in the references mentioned below, the optimal solution of these models appears to be out of reach at the moment. Moreover, even only finding a feasible solution following the classical heuristic approaches, based or not on these models, is far from trivial. On the other hand, we eventually managed to design an effective heuristic procedure based on an appropriate *Integer Linear Programming* (ILP) formulation that allowed us to find solutions significantly better than the "manual" solutions found by practitioners. Based on our previous experience on similar case studies, we found this very strange: there is a wide margin of improvement over the manual solution, but even only finding a feasible solution of the same value as the manual one (which is feasible according to our formal definition of the problem) appears to be challenging.

1.1 Literature review

Given its importance, the problem has been widely studied in the literature on railway optimization; for surveys on the specific problem as well as on the use of combinatorial optimization in railway planning see, e.g., [6, 7, 10, 14, 19].

Most of the approaches in the literature consider the case in which locomotives and cars have to be assigned to trains [5, 11–14, 20, 24]. In particular, Brucker et al. [5] consider the problem of routing railway cars through a railway network, so that seat requirements are satisfied while minimizing a non-linear cost function. The problem is solved through a simulated annealing procedure. In [11], Cordeau et al. present a simultaneous locomotive and car assignment problem, which is formulated as a large ILP and solved by Benders decomposition. Cordeau et al. [13] extend the model by considering real-life aspects, such as maintenance operations, and propose a heuristic branch-and-bound approach based on column generation. Lingaya et al. [20] present a model for operational management of cars, where the order in which cars are combined to cover a train is taken into account. The problem is solved using a Dantzig-Wolfe reformulation.

There are a few references that consider the assignment of TUs: [1, 2, 15, 23, 25]. Most of them consider the case in which there is a very small number of distinct TU types (two in most cases). On the other hand, in most of these cases, the rules for composing TUs for a trip are quite difficult. In [3], Ben-Khedher et al. consider the case in which there is a unique type of TUs. The objective is to maximize the expected profit for the company and the problem is solved by means of stochastic optimization, branch-and-bound and column generation. In [1], Abbink et al. present an ILP formulation with the objective of minimizing the seat shortages during the rush hours. Alfieri et al. [2] propose an ILP model for the case of multiple TU types, aimed at satisfying the seat requests while minimizing the travel distance. The problem is solved by decomposition into subproblems. Schrijver [25] presents a problem where a single-day workload is considered, with the objective of minimizing the number of TUs used. The problem is formulated as an ILP and solved by a general-purpose solver. Peeters and Kroon [23] present a problem in which the *train series* concept is introduced: given two endpoints between which several trains run up and down according to

the timetable, for a train series the available rolling stock consists of the same material type with different subtypes, which differ in number of cars and capacity. The order of the units in a composition is considered. They take into account three evaluation criteria, namely the kilometer-shortages, the number of shunting operations and the carriage-kilometers, and model the problem by using a transition graph, which represents the set of feasible transitions between compositions. They solve the problem by using a Dantzig-Wolfe reformulation and applying a branch-and-price algorithm, being able to find the optimal solution of real-world instances of NSR (the main Dutch Train Operator) in very short computing times. Fioole et al. [15] present a mixed ILP model that can be seen as an extended version of the model described by [25]. They apply several methods to improve the continuous relaxation and manage to solve to near optimality real-world instances by a general-purpose ILP solver.

The problem has some similarities with the multiple-depot vehicle scheduling problem (see, e.g., [18, 8]), which however has two remarkable differences with respect to our problem. First, each vehicle must depart from a depot and go back to the same depot at the end of the day, which makes the problem hard, whereas in our case each TU (or locomotive/car) goes back to its original depot only after a certain number of days, generally not specified in advance. Second, each trip has to be covered by one vehicle only, of any type, so the complicating seat requirement constraint, which may lead to TU combinations to cover a trip, is not imposed.

1.2 Outline of the paper

As will be discussed next, the key constraints of our problem concern the minimum number of passenger seats that have to be assigned to every trip. In ILP models, this is naturally formulated as a "knapsack-type" constraint in "$\geq$" form. The numerical nature of this constraint makes it very "weak" when the *Linear Programming* (LP) relaxation of the problem is considered, as already observed in [25]. In particular, none of the approaches we tried, among those based on ILP models and LP relaxation, managed to find a feasible solution as long as we stuck to these constraints. On the other hand, taking into account the fact that in our case at most two TUs can be combined to cover a trip, we replaced the "weak" constraints above by the inequalities obtained from a complete description of the knapsack polytope for the special case in which the sum of the variables cannot exceed two. This is similar to what was done in [25], with the difference that in that case the description was found numerically, case by case, for polytopes with two variables, whereas in our case the upper bound of two on the variable sum allows a formal description that is valid for any number of variables. Our final heuristic method, based on the ILP model with these new inequalities, yields the results mentioned above.

The paper is organized as follows. In Sect. 2 we formally define the problem considered, whose computational complexity is analyzed in Sect. 2.1. ILP models are illustrated in Sect. 3, strengthened as outlined above in Sect. 4, and used to drive our heuristic method, presented in Sect. 5. In Sect. 6 we define additional

maintenance constraints for the problem and discuss how to deal with them. Finally, Sect. 7 presents the computational results on our case study.

2 Problem Description

Given a set of timetabled trips to be performed every day, and a set of TUs of different types, the *TU Assignment Problem* (TUAP) calls for the specification of the TUs to be used, and, for each of these TUs, of the associated trips. The sequence of trips associated with a TU corresponds to a possible daily workload for the TU, and must satisfy a set of sequencing constraints. For instance, in our case study, for each pair of consecutive trips in the sequence, the time elapsing between the arrival of the first one and the departure of the second one must be large enough to allow the TU to travel from the arrival station of the first one to the departure station of the second one (this is a deadhead in case the two stations do not coincide).

Given that there is an overnight break of a few hours, it is not necessarily the case that every TU used performs the same set of trips every day. Indeed, after having performed a sequence of trips on one day, a TU can perform on the following day a sequence of trips assigned to another TU of the same type (possibly performing a deadhead transfer within the night break). In other words, the, say, q trip sequences assigned to TUs of a given type can be numbered as $1, \ldots, q$ in an arbitrary way, and can be performed by q TUs of that type, all performing a different sequence on each day, and each one performing the q sequences in the cyclic order $1, \ldots, q$ over a period of q days. This is important when maintenance constraints, illustrated in Sect. 6, are introduced in the problem.

TUs can be assigned to the same trip in order to guarantee that the number of passenger seats required by the trip is reached. As our problem concerns a suburban area, there is no distinction between first and second class seats, as in most references above. At the end of the trip, the TUs assigned to the trip can be uncoupled and assigned to different trips following the rules outlined above. In particular, the feasibility of a sequence of trips for a TU does not depend on the other TUs assigned to the trips, which is a notable simplification with respect to other cases of the problem addressed in the literature, see, e.g., [23]. This is related with the fact that in our case at most two TUs can be combined assigned to a trip, in order to keep coupling and uncoupling operations simple, so these operations take relatively short.

Although there are many factors contributing to the cost of a solution, such as deadheading or coupling/uncoupling operations, the dominant cost in the case we consider is related with the use of a TU, and in this paper we will restrict ourselves to this cost. In fact, although we will formulate our model with a generic cost associated with the use of a TU of a given type, as is the case in [25], in our experiments our objective will be to minimize the overall number of TUs used.

Formally, the problem input specifies a set of n train *trips* and a set of p TU types. Each trip $j \in \{1, \ldots, n\}$ is defined by a required number r_j of passenger

seats, and a maximum number u_j of TUs that can be assigned to the trip. (Additionally, each trip is characterized by an arrival time and station and a departure time and station, and by a subset of TU types that can perform it, but this information is implicitly encoded in the graph illustrated below.) Each TU type $k \in \{1, \ldots, p\}$ is defined by a number d^k of available TUs, a cost c^k for each such TU used, and an associated capacity s^k (number of available seats). We say that a trip j is *covered* if the overall capacity of the TUs assigned to the trip is at least r_j. Finally, as is customary, the sequencing constraints are represented by a directed multigraph $G = (V, A)$, where each node corresponds to a trip, and in addition there are a dummy start node 0 and a dummy end node $n + 1$, i.e., $V = \{0, \ldots, n + 1\}$, and arc set A is partitioned into p subsets $A^1, \ldots, A^p$, where A^k is associated with TUs of type k for $k = 1, \ldots, p$. Given two distinct trips $i, j \in V \setminus \{0, n+1\}$, arc $(i, j)^k \in A^k$ exists if and only if a TU of type k can be assigned to i and then to j within the same day. (Specifically, arc $(i, j)^k$ exists whenever both trips i and j can be assigned to a TU of type k, and the time between the arrival of trip i and the departure of trip j allows a TU of such type to travel from the arrival station of trip i to the departure station of trip j.) Moreover, the dummy nodes are connected with all other nodes, namely $(0, i)^k, (i, n+1)^k \in A^k$ for $i = 1, \ldots, n$ and $k = 1, \ldots, p$. Note that each subgraph (V, A^k) is simple and transitive. Given a node $i \in V$, we will let $\delta^k_-(i)$ and $\delta^k_+(i)$ denote, respectively, the set of arcs entering and leaving node i.

There is a one-to-one correspondence between trips assigned to a TU of type k and a path in G formed by arcs in A^k. The problem calls for the determination, for each TU type $k \in \{1, \ldots, p\}$, of up to d^k paths from 0 to $n + 1$ formed by arcs in A^k, each path having cost c^k and capacity s^k, such that each trip $j \in \{1, \ldots, n\}$ is visited by at most u_j paths whose overall capacity is at least r_j, with the objective of minimizing the overall cost of the paths. In the following we will use the acronym TUAP to denote the problem just described.

In the specific application that we will consider, we have $u_j = 2$ for $j = 1, \ldots, n$, i.e., each trip can be assigned to at most two TUs. For this specific case, we will discuss how to write the constraints on the required number of seats in a way that is much stronger than the trivial one.

2.1 Complexity

In this section we discuss the complexity of TUAP, proving in particular that the specific version considered in our case study is strongly NP-hard. The first result shows that the real difficulty of the problem is due to the presence of distinct TU types.

Observation 1 *TUAP is solvable efficiently in case $p = 1$, i.e., if there is a unique TU type.*

Proof. In this case, one can replace each trip j by $\lceil r_j/s^1 \rceil$ trips with the same timetable and request s^1: the associated problem calls for the determination of the minimum number of paths to cover all the vertices in a transitive directed acyclic graph, which is polynomially solvable by flow techniques (see, e.g., [17]).

If distinct TU types are present, the problem is already difficult if each trip must be covered by one TU only, and the minimum connection time between two trips does not depend on the trips nor on the TU type (e.g., it is 0, as in the statement of the proposition below). This problem has already been considered in the literature as it arises in other applications, e.g., in the assignment of classrooms to timetabled classes, with the constraint that each class receives a classroom having a number of seats at least equal to the number of students attending the class. The following proposition is due to [4].

Proposition 1. *TUAP is strongly NP-hard in the special case in which* $u_j = 1$ *for* $j = 1, \ldots, n$*, and* $(i, j) \in A^k$ *if and only if the departure time of trip* j *is not smaller than the arrival time of trip* i *for* $i, j = 1, \ldots, n$ *and* $k = 1, \ldots, p$*.*

Moreover, the following simpler result shows that, when $u_j = 2$ for $j = 1, \ldots, n$, the problem is strongly NP-hard even if all trips are simultaneous, due to its numerical nature. The proof is omitted for space reasons and will be given in the full paper.

Proposition 2. *TUAP is strongly NP-hard in the special case in which* $u_j = 2$ *for* $j = 1, \ldots, n$ *and* $A^k = \emptyset$ *for* $k = 1, \ldots, p$*.*

3 ILP Formulations

The two ILP formulations that we use for our problem, one with variables associated with arcs of G and the other with variables associated with paths in G, are standard, being analogous to others that have been widely used both in the context of TU assignment and for other optimization problems in transportation, see, e.g., the survey in [14].

3.1 Arc formulation

Let us introduce an integer variable x_a, for each $k = 1, \ldots, p$ and $a = (i, j)^k \in A^k$, that indicates the number of arcs $a \in A^k$ selected in the solution, i.e., the number of TUs of type k that execute trip i before trip j in the associated sequence. The ILP model is the following:

$$\min \sum_{k=1}^{p} \sum_{a \in \delta_+^k(0)} c^k x_a, \tag{1}$$

$$\sum_{a \in \delta_-^k(j)} x_a = \sum_{a \in \delta_+^k(j)} x_a, \;\; k = 1, \ldots, p, \;\; j = 1, \ldots, n, \tag{2}$$

$$\sum_{a \in \delta_+^k(0)} x_a \leq d^k, \;\; k = 1, \ldots, p, \tag{3}$$

$$\sum_{k=1}^{p} \sum_{a \in \delta_-^k(j)} s^k x_a \geq r_j, \;\; j = 1, \ldots, n, \tag{4}$$

$$\sum_{k=1}^{p} \sum_{a \in \delta^k_-(j)} x_a \leq u_j, \ j = 1, \ldots, n, \tag{5}$$

$$x_a \geq 0, \text{ integer }, \ k = 1, \ldots, p, \ a \in A^k. \tag{6}$$

Flow conservation constraints (2) guarantee that the solution contains a number of paths in (V, A^k) from 0 to $n+1$ equal to the number of arcs in A^k leaving node 0. Accordingly, constraints (3) ensure that the solution contains at most d^k such paths, i.e., no more than d^k TUs of type k are used. Moreover, as each of these paths has cost c^k, the objective function (1) calls for the minimization of the total cost of the paths. Finally, constraints (4) and (5) guarantee that each trip j is visited by at most u_j paths, having overall capacity at least r_j.

In the general context of multicommodity flow, it is well known that the ILP formulation based on path variables, illustrated later, is to be preferred to the one above when approaches based on the solution of the LP relaxation are used, see, e.g., [9]. This will also be shown by the experiments performed for our case study.

On the other hand, given the relatively large size of the ILP in our case study, it is natural to consider the Lagrangian relaxation of the above formulation, obtained by relaxing constraints (4) and (5) in a Lagrangian way. The resulting Lagrangian relaxed problem is easy to solve, recalling also Observation 1, as it amounts to finding optimal paths in graphs (V, A^k) for $k = 1, \ldots, p$. However, despite completely analogous approaches are the best ones in practice in many similar cases, our implementation of a customary heuristic method based on this Lagrangian relaxation performed extremely poorly in practice for our case study, in terms of both lower bound produced and solution found (in fact, it was never able to find a solution respecting all constraints (5), always requiring more TUs than those available). Given that the results were so poor, we will not even present these results in the experimental section.

3.2 Path formulation

Let $\mathcal{P}^k$ denote the collection of paths from 0 to $n+1$ in (V, A^k), and introduce an integer variable x_P, for each $k = 1, \ldots, p$ and $P \in \mathcal{P}^k$, that indicates the number of times that path P is selected in the solution, i.e., the number of TUs of type k that execute the trips sequence corresponding to P. Moreover, for each $k = 1, \ldots, p$ and $j = 1, \ldots, n$, let $\mathcal{P}^k_j \subseteq \mathcal{P}^k$ denote the subcollection of paths in $\mathcal{P}^k$ that visit trip j. The ILP model is the following:

$$\min \sum_{k=1}^{p} \sum_{P \in \mathcal{P}^k} c^k x_P, \tag{7}$$

$$\sum_{P \in \mathcal{P}^k} x_P \leq d^k, \ k = 1, \ldots, p, \tag{8}$$

$$\sum_{k=1}^{p} \sum_{P \in \mathcal{P}^k_j} s^k x_P \geq r_j, \ j = 1, \ldots, n, \tag{9}$$

$$\sum_{k=1}^{p} \sum_{P \in \mathcal{P}_j^k} x_P \leq u_j, \ j = 1, \ldots, n, \tag{10}$$

$$x_P \geq 0, \text{ integer }, \ k = 1, \ldots, p, \ P \in \mathcal{P}^k. \tag{11}$$

The interpretation and verification of correctness of the model is analogous (and in fact simpler) than the one of model (1)–(6). The fact that the LP relaxations of the two models presented are equivalent is a well known fact; see, e.g., [9].

Observation 2 *To each solution of the LP relaxation of (1)–(6) there corresponds a solution of the LP relaxation of (7)–(11) of the same value, and viceversa.*

Although model (7)–(11) has, in general, an exponential number of variables, as opposed to model (1)–(6), the LP relaxation of the former is faster to solve in practice by column generation techniques. Letting $J_P \subseteq \{1, \ldots, n\}$ be the set of trips visited by a path $P \in \mathcal{P}^k$, the dual of the LP relaxation of model (7)–(11) reads:

$$\begin{aligned} \max & -\sum_{k=1}^{p} d^k \alpha^k + \sum_{j=1}^{n} r_j \beta_j - \sum_{j=1}^{n} u_j \gamma_j, \\ & -\alpha^k + \sum_{j \in J_P} s^k \beta_j - \sum_{j \in J_P} \gamma_j \leq c^k, \ k = 1, \ldots, p, \ P \in \mathcal{P}^k, \\ & \alpha^k, \beta_j, \gamma_j \geq 0, \ k = 1, \ldots, p, \ j = 1, \ldots, n, \end{aligned} \tag{12}$$

and hence the column generation problem, which is the separation problem for constraints (12), given a dual solution $\overline{\alpha}, \overline{\beta}, \overline{\gamma}$ calls for $k \in \{1, \ldots, p\}$ and $P \in \mathcal{P}^k$ such that

$$\sum_{j \in J_P} (s^k \overline{\beta}_j - \overline{\gamma}_j) > c^k + \overline{\alpha}^k,$$

and can be solved as a maximum-profit path from 0 to $n+1$ in (V, A^k) with node profits $s^k \overline{\beta}_j - \overline{\gamma}_j$ for each $j \in V \setminus \{0, n+1\}$.

Not only the LP relaxation of (7)–(11) is much faster to solve in practice by column generation techniques than the LP relaxation of (1)–(6), but also heuristic methods based on this LP relaxation, that proceed by fixing variables x_P, i.e., entire sequences for TUs in the solution, tend to perform better in practice. However, as already mentioned, in order to get useful results for our case study we had to replace constraints (9) by stronger constraints, as illustrated in the next section.

4 Strengthening the Capacity Constraints for the Case Study

In all natural ILP models for the problem, including those of the previous section, letting w_j^k be an integer variable indicating the number of TUs of type k assigned

to a trip j ($k = 1, \ldots, p,\ j = 1, \ldots, n$), the following constraints are imposed:

$$\sum_{k=1}^{p} s^k w_j^k \geq r_j, \quad j = 1, ..., n, \tag{13}$$

$$\sum_{k=1}^{p} w_j^k \leq u_j, \quad j = 1, ..., n. \tag{14}$$

(In particular, variables w_j^k would be defined by equations $w_j^k = \sum_{a \in \delta_-^k(j)} x_a$ in model (1)–(6), and by equations $w_j^k = \sum_{P \in \mathcal{P}_j^k} x_P$ in model (7)–(11).)

It is well known that the constraints (13) can be very weak for the LP relaxation. Moreover, since in our case study we have $u_j = 2$ for $j = 1, \ldots, n$, the dominant of the convex hull of the nonnegative integer vectors satisfying (13) and (14) is defined by $O(p)$ simple constraints, that we will use to replace (13) in our models. In order to simplify the notation, we will remove the index j and study the following polytope:

$$P := \text{conv}\left\{ w \in \mathbb{Z}_+^p : \sum_{k=1}^{p} s^k w^k \geq r, \sum_{k=1}^{p} w^k \leq 2 \right\}, \tag{15}$$

assuming $s^1 \geq s^2 \geq \ldots \geq s^p$. Its *dominant* $\overline{P}$ is defined as follows:

$$\overline{P} := \{ w \in \mathbb{R}^p : \text{ there exists } \overline{w} \in P \text{ such that } w \geq \overline{w} \}. \tag{16}$$

All the inequalities in "$\geq$" form with nonnegative coefficients that are valid for P are also valid for $\overline{P}$ and viceversa, so the description of $\overline{P}$ yields a set of stronger inequalities to replace the "weak" inequality $\sum_{k=1}^{p} s^k w^k \geq r$.

The following theorem provides a simple description of $\overline{P}$ by $O(p)$ linear inequalities. The proof is omitted for space reasons and will be given in the full paper.

Theorem 1. *If $2s^1 < r$, then $\overline{P} = \emptyset$. Otherwise, letting g be such that $s^g \geq r$ and $s^{g+1} < r$ (with $g := 0$ if $s^1 < r$ and $g := p$ if $s^p \geq r$), t be such that $2s^t \geq r$ and $2s^{t+1} < r$ (with $t := p$ if $2s^p \geq r$), and, for each $k = g+1, \ldots, t$, $f(k)$ be such that $s^k + s^{f(k)} \geq r$ and $s^k + s^{f(k)+1} < r$ (with $f(k) := p$ if $s^k + s^p \geq r$ and $f(t+1) := t$):*

$$\overline{P} = \left\{ w \in \mathbb{R}_+^p : \sum_{\ell=1}^{k-1} 2w^\ell + \sum_{\ell=k}^{f(k)} w^\ell \geq 2, \quad k = g+1, \ldots, t+1 \right\}. \tag{17}$$

Example 1. In order to illustrate the above result, let us consider the numerical example, taken from our case study, in which $p = 8$, $r = 1302$ and $s = (1150, 1044, 786, 702, 543, 516, 495, 360)$. In this case we have $g = 0$, $t = 4$,

$f(1) = f(2) = 8$, $f(3) = 6$, $f(4) = 4$, leading to the following constraints:

$$\begin{aligned} w_j^1 + w_j^2 + w_j^3 + w_j^4 + w_j^5 + w_j^6 + w_j^7 + w_j^8 &\geq 2 \\ 2w_j^1 + w_j^2 + w_j^3 + w_j^4 + w_j^5 + w_j^6 + w_j^7 + w_j^8 &\geq 2 \\ 2w_j^1 + 2w_j^2 + w_j^3 + w_j^4 + w_j^5 + w_j^6 &\geq 2 \\ 2w_j^1 + 2w_j^2 + 2w_j^3 + w_j^4 &\geq 2 \\ 2w_j^1 + 2w_j^2 + 2w_j^3 + 2w_j^4 &\geq 2 \end{aligned}$$

out of which the second is dominated by the first and the last is dominated by the last but one.

According to the above discussion, the two ILP models of the previous section can be strengthened by letting $g_j, t_j, f_j(\cdot)$ be defined from r_j as $g, t, f(\cdot)$ were defined from r in the statement of Theorem 1, and replace (13) by the following constraints:

$$\sum_{\ell=1}^{k-1} 2w_j^\ell + \sum_{\ell=k}^{f_j(k)} w_j^\ell \geq 2, \quad j = 1, ..., n, \ k = g_j + 1, \ldots, t_j + 1, \tag{18}$$

noting that some of the constraints in the list may be dominated by others and therefore not imposed in practice.

Without explicitly introducing the variables w_j^k, in model (1)–(6) constraints (4) can be replaced by:

$$\sum_{\ell=1}^{k-1} \sum_{a \in \delta_-^k(j)} 2x_a + \sum_{\ell=k}^{f_j(k)} \sum_{a \in \delta_-^k(j)} x_a \geq 2, \quad j = 1, ..., n, \ k = g_j + 1, \ldots, t_j + 1, \tag{19}$$

and in model (7)–(11) constraints (9) can be replaced by:

$$\sum_{\ell=1}^{k-1} \sum_{P \in \mathcal{P}_j^\ell} 2x_P + \sum_{\ell=k}^{f_j(k)} \sum_{P \in \mathcal{P}_j^\ell} x_P \geq 2, \quad j = 1, ..., n, \ k = g_j + 1, \ldots, t_j + 1, \tag{20}$$

observing that this latter replacement does not affect the structure of the column generation problem discussed in the previous section.

5 An LP-Based Heuristic Method

We next illustrate the heuristic method, based on the LP relaxation of model (7)–(11) with (9) replaced by (20), that eventually allowed us to improve the practitioners' solution for our case study. Besides the (customary) column-generation based procedure to solve the LP relaxation, the heuristic method has three main components: (1) a diving rule to fix the value of some of the variables following the current LP optimal solution, reoptimizing the LP after the addition of these

TUAP_heur
begin
initialize the current LP as a reduced version of LP (7)–(11), with (9) replaced by (20), with only a subset of the variables;
repeat
solve the current LP by a general-purpose LP solver, letting $\overline{x}$ be the optimal primal solution, $(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$ the optimal dual solution, and $\overline{z}$ the corresponding value;
apply the constructive heuristic procedure based on $(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$;
refine the solution found by the constructive heuristic procedure, possibly updating the incumbent solution;
if there are dual constraints violated by $(\overline{\alpha}, \overline{\beta}, \overline{\gamma})$ **then**
add some of the corresponding primal variables to the current LP;
else
fix the value of some of the primal variables by changing the associated bounds;
until the current LP is infeasible **or** $\overline{z} \geq$ value of the incumbent solution;
end.

Fig. 1. General structure of the LP-based heuristic method.

fixing constraints, (2) a simple constructive heuristic procedure based on the current dual LP solution that is applied at each iteration of the column-generation based procedure, and (3) a refinement procedure that is applied to improve each solution produced by the constructive heuristic procedure in (2). The general structure of the method is outlined in Fig. 1.

5.1 Fixing phase

Each time we have obtained the optimal solution $\overline{x}$ of the current LP with the fixing constraints, i.e., there are no dual constraints violated, we change the bounds of the variables as follows. We consider all variables x_P such that $\overline{x}_P$ is integer, setting the associated lower bound to $\overline{x}_P$, i.e., imposing at least $\overline{x}_P$ paths P in the solution. Moreover, we consider the variable x_P whose value $\overline{x}_P$ is the largest among the fractional ones, and set the associated lower bound to $\lceil \overline{x}_P \rceil$. Note that, in this way, we may, e.g., fix the lower bound of a variable to 1, and then find values of these variables that are strictly larger than 1 in subsequent LP solutions.

We observed that, after the fixing phase, it may happen that the current LP becomes infeasible, and then become feasible again after some iterations of the column generation procedure. In order to avoid dealing with LPs that are infeasible due to the fact that we are only considering a subset of the variables, we introduce explicit slack variables for constraints (20), adding them to the objective function with a high penalty. This simplifies also the initialization of the current LP at the beginning of the procedure. Note that the "the current

LP is infeasible" condition to be checked at the end is then equivalent to having some of the slack variables strictly positive in the solution.

5.2 Constructive heuristic procedure

The constructive heuristic procedure that we apply at each iteration considers the TU types one at the time, according to increasing values of c^k/s^k. For each TU type k, we define up to d^k paths to be added to the solution. In addition to the paths that possibly were already fixed in the solution by the fixing phase, the remaining paths are found by computing maximum-profit paths in (V, A^k), analogously to the column generation procedure, with node profits defined in a more complex way. For the trips that are not covered by the previously-defined paths, the profit takes into account (a) the associated dual variables, and (b) how well the capacity of the current TU type matches the residual request of the trip, i.e., by assigning a TU of this type to the trip, will it be possible to satisfy *at equality* the trip request? Moreover, we assign in any case a small positive profit to the trips already covered.

One of the main ideas is to try to follow the dual profits for the trips that still have to be covered, but also to try to satisfy at equality the request of these trips and to *over-cover* trips that have already been covered, in the hope of being able to achieve larger improvements with the subsequent refinement procedure. To this aim, we do not consider explicitly constraints (10) on the maximum number of TUs that can be assigned to a trip in the construction.

The constructive procedure terminates either when we have used all the available TUs, or when the paths constructed so far cover all the trips. Note that in the latter case we have saved some TUs of the last type (largest c^k/s^k ratio), and, in case all of them were saved, some TUs of the last but one type, and so on. On the other hand, in the former case, some of the trips are not covered. Moreover, in both cases we have that constraints (10) may be violated. The following refinement procedure tries to take care of these infeasibilities.

Concerning the fact that we are trying to satisfy at equality the trip requests, note that the input instance can always be preprocessed so that this is possible, by redefining the request r_j of each trip $j \in \{1, \ldots, n\}$ as:

$$r_j := \min \left\{ \sum_{k=1}^{p} s^k w_j^k : \sum_{k=1}^{p} s^k w_j^k \geq r_j, \sum_{k=1}^{p} w_j^k \leq u_j, w_j^k \in \{0, \ldots, d^k\},\ (k = 1, \ldots, p) \right\}$$

The associated optimization problem, which is a cardinality constrained bounded subset sum problem [21], can easily be solved by enumeration given the small values of p in practical cases.

5.3 Refinement

This is a key step in our framework. We consider the solution produced by the constructive heuristic procedure by taking into account only the information about the number of times $\overline{w}_j^k$ that each trip $j \in \{1, \ldots, n\}$ is assigned to a

TU of type $k \in \{1, \ldots, p\}$, *without* considering the specific sequences (paths) defined. In other words, we take care only of the information that would be given by variables w_j^k as defined in Sect. 4.

In order to find the "best" solution that takes into account this trip assignment information, we use a variant of ILP model (1)–(6) with (4) replaced by (19), in which, for each trip j that is (over-)covered, we impose that the number of times that the trip is assigned to a TU of type k does not exceed $\overline{w}_j^k$. More precisely, for all trips j that are covered but not over-covered by the solution, i.e., for which $\sum_{k=1}^{p} s^k \overline{w}_j^k \geq r_j$, $\sum_{k=1}^{p} \overline{w}_j^k \leq u_j$, and $\sum_{k=1}^{p} s^k \overline{\overline{w}}_j^k < r_j$ for each vector $(\overline{\overline{w}}_j^1, \ldots, \overline{\overline{w}}_j^p) \lneq (\overline{w}_j^1, \ldots, \overline{w}_j^p)$, we impose the additional constraints:

$$\sum_{a \in \delta_-^k(j)} x_a = \overline{w}_j^k, \;\; k = 1, \ldots, p,$$

removing constraints (5) and (19) associated with j. For all trips j that are over-covered by the solution, we impose the additional constraints:

$$\sum_{a \in \delta_-^k(j)} x_a \leq \overline{w}_j^k, \;\; k = 1, \ldots, p.$$

In this case, the constraints (19) associated with j are modified (strengthened) taking into account that not all TU types can be used to cover the trip, changing $g_j, t_j, f_j(\cdot)$ accordingly. Finally, for all trips that are not covered by the solution, we do not impose any additional constraint. The resulting "reduced" ILP is solved by a general-purpose ILP solver to optimality.

6 Maintenance Constraints

A key constraint that is imposed in our case study, and that we did not discuss in detail so far to keep the presentation simple, is the one imposing that each TU of type k ($k = 1, \ldots, p$) has to undergo a maintenance operation every m^k days. Generally speaking, this operation requires a transfer to a maintenance point (by deadheading), a certain amount of time at the maintenance point, and then a transfer from the maintenance point.

Given the very flexible representation of the sequencing constraints via graph G, we can model the maintenance constraints by specifying, for each $k \in \{1, \ldots, p\}$, a subset of arcs $M^k \subseteq A^k$ corresponding to sequences of two trips that allow a maintenance in between for a TU of type k. Possibly, we have that M^k contains arcs of the form $(0, j)^k$, $(j, n+1)^k$ (e.g., if the maintenance can be performed overnight). Recalling the cyclic nature of the daily assignments to TUs of type k illustrated at the beginning of Sect. 2, letting $e^k \leq d^k$ be the number of paths in (V, A^k) selected in the solution, the maintenance constraints impose that at least $\lceil e^k / m^k \rceil$ of these paths contain at least one arc in M^k.

Within ILP model (7)–(11), letting $\mathcal{Q}^k \subseteq \mathcal{P}^k$ denote the subcollection of paths in $\mathcal{P}^k$ that contain at least one arc in M^k, the maintenance constraints

can be represented by adding the integer variables y^k, indicating the number of paths in $\mathcal{Q}^k$ selected for TUs of type k, along with the constraints:

$$\sum_{P \in \mathcal{Q}^k} x_P \geq y^k, \;\; k = 1, \ldots, p, \tag{21}$$

$$\sum_{P \in \mathcal{P}^k} x_P \leq m^k y^k, \;\; k = 1, \ldots, p, \tag{22}$$

$$y^k \geq 0, \text{ integer }, \;\; k = 1, \ldots, p. \tag{23}$$

The presence of maintenance constraints complicates slightly the column generation procedure, that now calls both for the path of maximum profit in $\mathcal{P}^k$ as well as the path of maximum profit in $\mathcal{Q}^k$. On the other hand, given that the paths have to be found in an acyclic directed graph, their determination simply requires, in the canonical dynamic programming procedure, to store for each node not only the maximum-profit path from 0 to that node, but also the maximum-profit path from 0 to that node containing at least one arc in M^k (if any).

The presence of maintenance constraints must also be carefully taken into account in the heuristic method described in Sect. 5, since these constraints are systematically violated, at least in our case study, if they are not imposed explicitly. In particular, in the fixing phase, when searching for the fractional variable of maximum value to fix, we exclude variables x_P for which the addition of $\lceil \overline{x}_P \rceil$ paths to the other paths in $\mathcal{P}^k$ already imposed by previous fixing phases leads to a collection $\overline{P}^k$ of paths such that $|\overline{P}^k \cap \mathcal{Q}^k| < \lceil |\overline{P}^k|/m^k \rceil$ (in other words, these paths would violate the maintenance constraint for the TUs of type k). The same is done in the constructive heuristic procedure: we do not add a path to those already created for a TU of type k if this violates the maintenance constraint – this simply means that in some cases we add the maximum-profit path in $\mathcal{Q}^k$. Finally, in the refinement ILP, we impose the counterpart of constraints (21)–(23) referred to arc variables.

7 Experimental Results

Our method was implemented in C, the computational tests were executed on a PC Pentium 4, 3.2 GHz, 2 GB Ram, and the LP-solver used was ILOG-CPLEX 9.0. All times reported below are in CPU seconds on this PC.

We considered three different real-world instances provided by an operator running trains in a regional area. In every instance, each trip can be assigned to at most 2 TUs and all TUs have the same cost (normalized to $c^k = 1$ for $k = 1, \ldots, p$), i.e., we wish to minimize the overall number of TUs used. The maintenance constraints require a maintenance every at most $m^k = 5$ days ($k = 1, \ldots, p$), and a maintenance requires a period of at least 6 hours between 5AM and 12AM at a specific maintenance point – the time to travel to and from this maintenance point must be taken into account to establish if a given arc is in M^k.

Table 1. Characteristics of the instances.

inst.	n	r_j	p	(s^k)	(d^k)
A	528	$\in [360, 1404]$	8	(1150,1044,786,702,543,516,495,360)	(2,4,5,18,11,5,24,3)
B	662	$\in [588, 1534]$	10	(1534,1473,1128,980,887,840,834,824,805,588)	(4,3,5,1,18,3,25,5,9,3)
C	660	$\in [588, 1610]$	10	(1644,1625,1473,1128,887,840,834,824,805,588)	(3,1,3,4,18,4,25,5,9,3)

In Table 1 we report the characteristics of these instances, giving their name (inst.), the number of n of trips, the range for the trip requests r_j, the number p of TU types along with the capacity s^k and availability d^k for each type.

Table 2. Comparison of various LP relaxations.

inst.	(1)–(6)		(1)–(6) + (19)		(7)–(11)		(7)–(11) + (20)	
	value	time	value	time	value	time	value	time
A	57	624	62	1201	57	136	62	50
B	41	47242	53	26907	41	174	53	150
C	40	23841	53	27350	40	179	53	177

In Table 2 we compare the results obtained by solving the LP relaxations of the two ILP formulations in Sect. 3 with and without the stronger version of the capacity constraints discussed in Sect. 4 (and without maintenance constraints). The table clearly shows both the bound improvements achieved with the strengthened constraints and the much shorter time required to solve the second LP relaxation (recall that the two LPs are equivalent in the sense of Observation 1).

Table 3. Results for the instances in our case study.

inst.	curr. sol.	LP bound		heur.	
	value	value	time	value	time
A	72	62	130	63	3544
B	76	56	196	59	5471
C	74	55	295	58	8875

Finally, in Table 3 we compare the value of the solutions obtained by the practitioners (curr. sol.) with the lower bound found by solving the LP relaxation of (7)–(11),(20) with the addition of the maintenance constraints (21)–(23) (LP bound) and the value of the heuristic solution found by our method (heur.). The table shows that we can prove that the solutions we found are almost optimal, and that we improve on the practitioners' solution by 10-20%. Although the

latter contains other additional constraints that we did not mention, which makes direct comparison unfair, it seems that these additional constraints have a limited impact on the quality of the solutions found of our method. Evaluating the actual improvements that can be achieved by imposing all real-world constraints in our method is the subject of current research.

We conclude by noting that a few other alternative approaches that we implemented and tested (without mentioning them here) were not even able to find a feasible solution. Moreover, none of the following variants of our method finds a feasible solution, even if maintenance constraints are neglected:

- the one in which constraints (20) are not used;
- the one in which the fixing phase is not used, terminating the procedure when there are no violated dual constraints;
- the one in which the refinement procedure is not used;
- the one in which the constructive heuristic procedure is not used, and refinement is applied only to the final solution found by the fixing phase.

As already mentioned, the fact that there is a wide margin of improvement over the practitioners' solution and that such an improvement is indeed achieved by the best approach we could design is apparently in contrast with the fact that, as soon as any of the parts of this approach are deactivated, no improvement is obtained any more. This is certainly an intriguing aspect of our case study that we plan to investigate further in the future.

Acknowledgments

This work was partially supported by the EU Project ARRIVAL.

References

1. Abbink E.W.J., van den Berg B.W.V., Kroon L.G., and Salomon M.: Allocation of Railway Rolling Stock for Passenger Trains. Transportation Science **38** (2004) 33–41
2. Alfieri A., Groot R., Kroon L.G., and Schrijver A.: Efficient Circulation of Railway Rolling Stock. ERIM Research Report, ERS-2002-110-LIS, Erasmus Universiteit Rotterdam, The Netherlands, (2002)
3. Ben-Khedher N., Kintanar J., Queille C., and Stripling W.: Schedule Optimization at SNCF: From Conception to Day of Departure. Interfaces **28** (1998) 6–23
4. Bonomo F., Durán G., and Marenco J.: Exploring the Complexity Boundary between Coloring and List-Coloring. Electronic Notes in Discrete Mathematics **25** (2006) 41–47
5. Brucker J., Hurink J.L., and Rolfes T.: Routing of Railway Carriages: A Case Study. Osnabrücker Schriften zur Mathematik, Reihe P, Heft **205** (1998)
6. Bussieck M.R., Winter T., and Zimmermann U.T.: Discrete Optimization in Public Rail Transport. Mathematical Programming **79** (1997) 415–444
7. Caprara A., Kroon L., Monaci M., Peeters M., and Toth P.: Passenger Railway Optimization, in C. Barnhart and G. Laporte (eds.). Handbooks in OR & MS **12**, Elsevier Science, (2006)

8. Carpaneto D., Dell'Amico M., Fischetti M. and Toth P.: A branch and bound algorithm for the multiple vehicle scheduling problem. Networks **19** (1989) 531–548
9. Cook W.J., Cunningham W.H., Pulleyblank W.R., and Schrijver A.: Combinatorial Optimization, John Wiley and Sons, (1998)
10. Cordeau J.-F., Toth P., and Vigo D.: A Survey of Optimization Models for Train Routing and Scheduling. Transportation Science **32** (1998) 380–404
11. Cordeau J.-F., Soumis F., and Desrosiers J.: A Benders Decomposition Approach for the Locomotive and Car Assignment Problem. Transportation Science **34** (2000) 133–149
12. Cordeau J.-F., Soumis F., and Desrosiers J.: Simultaneous Assignment of Locomotives and Cars to Passenger Trains. Operations Research **49** (2001) 531–548
13. Cordeau J.-F., Desaulniers G., Lingaya N., Soumis F., and Desrosiers J.: Simultaneous Locomotive and Car Assignment at VIA Rail Canada. Transportation Research **35** (2002) 767–787
14. Desrosiers J., Dumas Y., Solomon M.M., and Soumis F.: Time Constrained Routing and Scheduling, in M.O. Ball et al. (eds.), Handbooks in OR & MS **8**, Elsevier Science, (1995) 35–139
15. Fioole P.-J., Kroon L.G., Maróti G., and Schrijver A.: A Rolling Stock Circulation Model for Combining and Splitting of Passenger Trains. European Journal of Operational Research **174** (2006) 1281–1297
16. Garey M.R. and Johnson D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, (1979)
17. Grötschel M., Lovász L., and Schrijver A.: Geometric Algorithms and Combinatorial Optimization. Springer-Verlag (1988)
18. Hadjar A., Marcotte O. and Soumis F.: A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem. Operations Research **54** (2006) 130–149
19. Huisman D., Kroon L.G., Lentink R.M., and Vromans M.J.C.M.: Operations Research in Passenger Railway Transportation. Statistica Neerlandica **59** (2005) 467–497
20. Lingaya N., Cordeau J.-F., Desaulniers G., Desrosiers J., and Soumis F.: Operational Car Assignment at VIA Rail Canada. Transportation Research **36** (2002) 755–778
21. Martello S. and Toth P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley and Sons (1990)
22. Nemhauser G.L. and Wolsey L.A.: Integer and Combinatorial Optimization. John Wiley and Sons (1988)
23. Peeters M. and Kroon L.G.: Circulation of Railway Rolling Stock: a Branch-and-Price Approach. ERIM Research Report, ERS-2003-055-LIS, Erasmus Universiteit Rotterdam, The Netherlands, (2003)
24. Rouillon S., Desaulniers G., and Soumis F.: An Extended Branch-and-Bound Method for Locomotive Assignment. Transportation Research **40** (2006) 404-423
25. Schrijver A.: Minimum Circulation of Railway Stock. CWI Quarterly **6** (1993) 205–217