Exact Methods for Solving the Train Departure Matching Problem

Haahr, Jørgen Thorlund; Bull, Simon Henry

Publication date:
2015

Exact Methods for Solving the Train Departure Matching Problem

Jørgen Thorlund Haahr and Simon Henry Bull
Department of Engineering Management
Technical University of Denmark

Abstract

In this paper we consider the train departure matching problem which is an important subproblem of the *Rolling Stock Unit Management on Railway Sites* problem introduced in the ROADEF/EURO Challenge 2014. The subproblem entails matching arriving train units to scheduled departing trains at a railway site while respecting multiple physical and operational constraints. In this paper we formally define that subproblem, prove its NP-hardness, and present two exact method approaches for solving the problem. First, we present a compact *Mixed Integer Program* formulation which we solve using a MIP solver. Second, we present a formulation with an exponential number of variables which we solve using column generation. Our results show that both approaches have difficulties solving the ROADEF problem instances to optimality. The column generation approach is however able to generate good quality solutions within a few minutes in a heuristic setting.

1 Introduction

Many railway planning problems have been studied in the literature for the last two decades. These range from long term high level planning problems, such as line planing, to short term rolling stock and crew scheduling problems. At train stations, planning problems include platform assignment, routing, and shunting. These problems often have direct or indirect dependencies but due to the high complexity, or company organizational structure, they are often solved in isolation and in sequential order. The *ROADEF/EURO Challenge 2014* presents a problem where the goal is to use a holistic approach to the planning problems at railway stations. The problem combines several planning aspects that must be handled between arrivals and departures at terminal stations such as matching available trains to departures, routing trains in the station infrastructure (without any two trains occupying the same infrastructure without sufficient time between them), determining whether and when to perform maintenance, and when and where to do the couplings and decouplings of train convoys.

In this paper we present our contribution for the Departure Matching Problem (DMP) in this challenge. The DMP can be considered a pure subproblem of the problem presented in the *ROADEF/EURO Challenge 2014* document [11], which we will refer to as the Rolling Stock Unit Management on Railway Sites (RSUM) problem.

The RSUM problem entails many different aspects but the performance of any solution approach will be greatly affected by how trains are matched to departures. The considered subproblem (DMP) is the problem of finding a good and feasible matching of trains to departures while respecting train compatibility and maintenance constraints. In contrast to the RSUM problem the DMP does not consider how to route, couple and de-couple train units in the station. For the purpose of this paper, we will assume that routing is done in a subsequent step. Importantly, many routing decisions are motivated by the matching of trains to departures; whether an arriving train should visit a maintenance facility or be parked in a yard depends on the train's subsequent departure.

In this paper we propose and benchmark two distinct optimal solution methods for solving the DMP. We will consider finding solutions that are either optimal or proven to be some percentage from the optimum. The proposed solution methods are however flexible and can easily be adjusted to find solutions in a heuristic manner. We investigate the potential of the methods in a heuristic setting.

1.1 Our Contributions

We present a definition for the DMP; a distinct, self contained sub-problem of the RSUM, and present two optimization-based approaches for solving instances of the DMP. Firstly, we introduce a Mixed Integer Program (MIP) mathematical model and present results produced using a commercial MIP solver. Secondly, we propose an alternative but equivalent MIP mathematical model that is solved using column generation. Due to time limitations and the hardness of the instances we only solve the pricing problem at the root node. The results are presented in the benchmark section.

2 Problem Definition

A full solution to the RSUM problem requires routing trains in the station infrastructure, respecting routing restrictions, headways, capacities, and many other constraints. Solutions are ranked using a weighted sum of multiple objective measures such as making preferred matchings, allocating arrivals and departures to preferred platforms, avoiding unnecessary platform dwell times etc. Solving the entire problem as a single optimization problem is intractable considering the given strict run-time requirements of the competition. We decompose the RSUM problem into several subproblems. The first subproblem, namely the DMP, is the scope of this paper.

A solution to the matching subproblem (DMP) can be used to build a solution to the overall RSUM problem, taking into account its other constraints and objectives. However the fixed matching provided from the DMP may lead to suboptimal solutions to the entire problem. Our approach for solving instances of the RSUM is to first solve a DMP instance, and then use an optimal approach to assigning platforms, before heuristically finding train routes. Our solution method is described in [3]. The DMP contains components of the RSUM sufficiently unrelated to routing, and those that we have determined through experimentation provide a sufficiently detailed problem, capturing related RSUM components to create a detailed but tractable subproblem.

The DMP is a matching problem between arrivals and departures at some terminal station, with complicating dependencies between potential matches. Given a fixed planning horizon at the station, there is a set $\mathcal{A}$ of trains arriving and a set of pre-specified departures $\mathcal{D}$ that must be assigned some compatible train unit. Every arrival $a \in \mathcal{A}$ has an arrival time $arrTime_a$, and every departure $d \in \mathcal{D}$ has a departure time $depTime_d$. One arrival corresponds to a single train unit, and if several trains arrive together as a convoy then they are represented by multiple arrivals with the same arrival time. Similarly one departure exists for every train that departs in a convoy sharing the same departure time. A set $\mathcal{I}$ of initial trains may reside in the station infrastructure at the start of the planning horizon. These are all available from the start of the planning horizon h_0. The total set of trains is denoted by $\mathcal{T} = \mathcal{I} \cup \mathcal{A}$. We define the availability time of a train $t \in \mathcal{T}$ as $startTime_t$ where $startTime_t = arrTime_t$ if t is part of an arrival and $startTime_t = h_0$ if t is an initial train (i.e. in $\mathcal{I}$).

Some departures are the beginning of a tour that returns the train units to the terminal station as arrivals within the planning horizon. If an arrival is linked to an earlier departure we call it a *linked* arrival. Likewise we call the earlier departure a *linked* departure. In order to avoid confusion, we note that the "linked" concept does *not* refer to physical train units that are coupled together to form a convoy. A linked arrival states that the arriving train is the same train that was assigned to the corresponding (earlier) linked departure. In contrast to a non-linked arrival, this means that the arriving train is the same physical train which was assigned to the linked departure. If the linked departure is cancelled, then a new replacement train arrives instead. The linking between arrivals and departures is important because it means that the properties of (linked) arrivals are not known, without knowing what trains (if any) are matched to those arrivals' linked departures.

We define the set $\mathcal{L} \subseteq \mathcal{A}$ as the set of arrivals which have an earlier linked departure, and define $\sigma(a)$ as the linked departure of the arrival train $a \in \mathcal{L}$.

Every train $t \in \mathcal{T}$ belongs to some category $cat_t \in \mathcal{C}$ that defines common characteristics such as train length, capacity and maintenance durations. Two trains of the same category are considered interchangeable when assigning trains to departures – with one exception. All trains in the system are subject to two types of maintenance constraints: Distance Before Maintenance (DBM) and Time Before Maintenance (TBM). Each train t has some initial *remaining DBM* ($remDBM_t$) and *remaining TBM* ($remTBM_t$). Every departure $d \in \mathcal{D}$ has a *required DBM* ($remDBM_d$) and a *required TBM* ($remTBM_d$) to finish the task. If a train is to be matched to departure d, then it must have sufficient DBM and TBM.

A train may visit a maintenance facility at the station between arriving and departing, which in the RSUM problem takes a fixed amount of time and resets both the DBM and TBM to their maximum value for the train, and we include the decision of whether or not to perform maintenance in the DMP. The constants $maxDBM_t$ and $maxTBM_t$ indicate the level of DBM and TBM that is obtained if a maintenance operation is performed on train t. For trains of the same category ($i \in \mathcal{C}$) the constants have identical values, and we can therefore use the notation $maxDBM_i$ and $maxTBM_i$ without ambiguity. Due to a limited amount of manpower at the maintenance facilities the total number of maintenance operations per day is limited to a constant of $maxMaint \in \mathbf{Z}^+$. The imposed limit means that certain combinations of matches can not all be made: if there are $n > maxMaint$ otherwise independent matches that would all occur on the same day and all require maintenance, at most $maxMaint$ of them can be made. The remaining $(n - maxMaint)$ trains could not be maintained and would therefore not have sufficient DBM and TBM to be matched to the $(n - maxMaint)$ departures.

For those arrivals that have a linked departure, category, DBM and TBM are dependent on any matching made to that linked departure. An arrival $a \in \mathcal{L}$ is linked to a previous departure $d \in \mathcal{D}$, i.e., the train $t \in \mathcal{T}$ assigned to departure d is the same train arriving later in a. The train in a therefore inherits the remaining DBM and TBM and category of train t. In the case where no train is assigned to d then another new train arrives with its own specified remaining DBM, TBM, and category.

Every train $t \in \mathcal{T}$ can only be matched with a limited set of departures. We define $CompDep(t)$ as the set of departures that are compatible with t. Likewise we define $CompTr(d)$ as the set of trains that are compatible with departure $d \in \mathcal{D}$. These two sets are considered as parameters to the problem, and it is up to the end-user to specify which factors to consider. These factors could include for example a minimum routing time, a maximum time between arrival and departure, or a maximum number of potential matchings for any given train. For the sake of simplicity, we define only a few simple rules for possible matchings. Given a departure $d \in \mathcal{D}$ and a non-linked arrival train or initial train $t \in \mathcal{T}$ a matching is possible if the following conditions are satisfied:

$$\begin{gathered}
cat_t \in compCatDep_d \\
startTime_t + maintenance_t > depTime_d \\
\max\{remDBM_t, allowDBM_t \cdot maxDBM_t\} \geq reqDBM_d \\
\max\{remTBM_t, allowTBM_t \cdot maxTBM_t\} \geq reqTBM_d
\end{gathered}$$

where the binary parameters $allowDBM_t$ and $allowTBM_t$ indicate whether DBM and TBM are allowed to be performed. In some cases there is only time for one of the two operations but not both. The constant $maintenance_t$ indicates the time needed to perform the necessary maintenance operations for train t, or zero if no maintenance is required.

Given a departure $d \in \mathcal{D}$ and a linked arrival $t \in \mathcal{L}$ it is harder to limit the options beforehand. In the preprocessing it can only be restricted by $startTime_t > depTime_d$ as the category, remaining DBM and remaining TBM of the linked arrival t are unknown.

In practice it is in some cases expected that certain arrivals are matched with specific departures. If such a match is successful we call it a *train reuse*, otherwise it is a missed train reuse. We denote $\mathcal{U}$ as the set of train uses, where tr_u and dep_u define the train $t \in \mathcal{T}$ and departure $d \in \mathcal{D}$ for a reuse $u \in \mathcal{U}$.

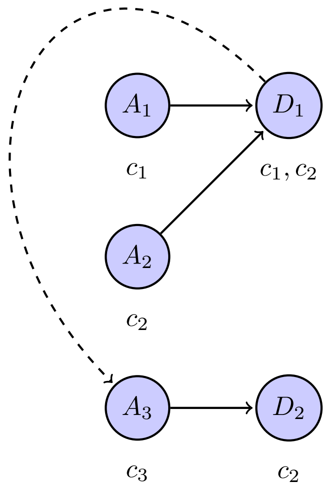

Figure 1: An illustration of the matching problem showing potential matches between three arrival trains (A_1, A_2, A_3) and two departures (D_1, D_2). Each arrival has a train category (c_1, c_2, c_3) and each departure has one or more acceptable train categories. Departure D_1 and arrival A_3 are linked; arrival A_3 has replacement category c_3 only if departure D_1 is unmatched, but instead has category c_1 or c_2 depending on which arrival is matched to D_1. The only matching of cardinality 2 is $\{(A_2, D_1), (A_3, D_2)\}$; matching A_1 with D_1 precludes matching A_3 with D_2 as the category of arrival A_1 is incompatible with departure D_2.

Definition 2.1 (The Departure Matching Problem Definition)**.** We define the DMP as the problem of finding a feasible matching between trains and departures that respects the departure maintenance requirements, the departure train category compatibility, the time required to perform the needed maintenance, and the total number of maintenance operations per day. The objective of the DMP is to minimize the number of uncovered departures and to maximize the number of train reuses. In every instance of the problem there is a fixed penalty for every missed train reuse and a fixed penalty for every uncovered departure.

Figure 1 shows a small example with three arrival trains (A_1, A_2, A_3) and two departure trains (D_1, D_2), considering three different train categories having no maintenance requirements or other restrictions. In this example departure D_1 and arrival A_3 form a *linked arrival* pair: whatever is matched to departure D_1 returns as arrival A_3. Arrival train A_3 is in $\mathcal{L}$, and its linked departure is D_1; that is, $\sigma(A_3) = D_1$. In the figure, arrival A_3 has category $c3$ marked as its *replacement train* category. That is, it *only* has category c_3 if no match is made to departure D_1, but if instead some match is made, the category of A_3 is inherited from that match. The match between arrival A_3 and departure D_2 is not independent of other matchings made; it is only feasible if a match is also made between arrival A_2 and departure D_1. If D_1 is unmatched, arrival A_3 will have the category of its replacement train, incompatible with departure D_2. Similarly, if D_1 is matched to arrival A_1 then arrival A_3 will inherit the category of arrival A_1, also incompatible with departure D_2.

3 Related Problems

The RSUM as defined for the ROADEF competition is based on real station infrastructure and problems, with certain simplifications to make it appropriate for the competition, such as simplifications of the switching and yard infrastructure. Real train stations face similar problems, though those problems differ in specific details. A matching problem similar to the DMP subproblem (that we have identified) could also exist as a subproblem at stations, though it may not necessarily be treated as a self contained sub-problem.

If the linking between some departures and later arrivals is ignored or not present, and performing maintenance is ignored, then whether or not a match is possible would be pre-determinable. If the problem was just that of minimizing the number of uncovered departures, then it would be a

$$\max \sum_{i\in\{1,2,3\}} p_1 x_{i1} + p_2 x_{i2} + p_3 x_{i3} + p_4 x_{i4}$$

$$\begin{aligned} w_1 x_{11} + w_2 x_{12} + w_3 x_{13} + w_4 x_{14} &\leq c_1 \\ w_1 x_{21} + w_2 x_{22} + w_3 x_{23} + w_4 x_{24} &\leq c_2 \\ w_1 x_{31} + w_2 x_{32} + w_3 x_{33} + w_4 x_{34} &\leq c_3 \end{aligned}$$

$$\sum_{i\in\{1,2,3\}} x_{ij} \leq 1 \quad \forall j \in \{1,2,3,4\}$$

$$x_{ij} \in \{0,1\}$$

Figure 2: Multiple Knapsack Problem instance with 4 (coloured) items and three knapsacks.

relatively simple maximal bipartite matching problem, solvable in polynomial time [4]. If minimizing a weighted sum of uncovered departures and missed reuses, the problem could be formulated as an assignment problem, also solvable in polynomial time [9].

The presence of linked arrivals inherently changes the structure of the problem. The matchings themselves are not independent because whether or not some train can be matched to some departure can depend on what *other* train is matched to some *other* departure. Similarly, the ability to perform maintenance changes which matches are possible, and the restriction of the maximum number of maintenance operations per day makes matches non-independent. The DMP combines the matching with components of the RSUM that we have identified as being closely related to and significantly interacting with the matching, without including so many aspects of the RSUM to make the problem intractable. For example, in the RSUM problem the restriction of the maximum number of maintenance operations per day means that some sets of potential matches can not all be included, and maintenance decisions should be included with the matching subproblem. If however there was no restriction on the maximum number of maintenance operations per day then perhaps a subproblem that ignores maintenance could be sufficient to provide feasible or optimal solutions, relying on it always being possible to perform maintenance if necessary. For some other similar station arrival problem, a similar but distinct problem to the DMP might instead be identified as a subproblem.

Freling et al. identify the subproblem of matching departures and arrivals as part of a shunting problem [2]. The authors formulate a matching subproblem that considers the unattractiveness of producing matches that require breaking up trains into units matching different departures. In contrast, in the DMP we do not include any cost or penalty for matches which require coupling or decoupling. The authors do not describe anything similar to linked arrivals or maintenance decisions and daily restrictions, which are the features of our problem that make matches interdependent.

Kroon et al. identify a matching sub-problem as part of a larger station shunting problem [8]. However there is no analogue to the linked arrivals of the DMP. The authors do not solve the matching problem in isolation but as part of a larger formulation that includes shunting features that are not part of the DMP or even necessarily part of the RSUM.

In the shunting literature there are many problems that share similarities with the RSUM problem and potentially have a subproblem that is very similar to the DMP. However shunting is not necessarily an important component of the ROADEF challenge because the station infrastructure has large and simplified “yard” resources. These are abstractions with only a maximum capacity, but train units can be parked in or removed from the yards without considering their ordering.

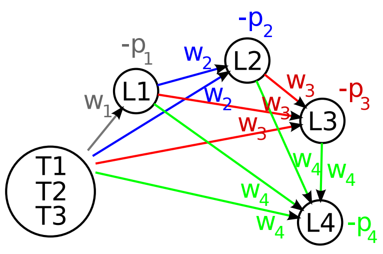

Figure 3: The constructed Train Matching Problem instance from the Multiple Knapsack Problem instance shown in Figure 2. Each train $(T1, T2, T3)$ corresponds to a single knapsack, and each linked departure $(L1, L2, L3, L4)$ corresponds to one item. For the sake of clarity, initial trains have been grouped together since the graph is identical for each train.

4 NP-hardness

It is relatively simple to prove that the DMP is NP-hard by reduction from the Knapsack Problem. However, since the classic version of the Knapsack Problem is NP-hard in the weak sense, we prove the same by reduction from the 0-1 Multiple Knapsack Problem (MKP). We adopt the definition of the MKP of [7]:

Definition 1. Given a list of items to pack, each with a profit p_i and weight w_i, and one or more knapsacks of capacity c_j. The 0-1 Multiple Knapsack Problem is the problem of choosing a list of items for every knapsack such that the profit of the selected items is maximized while respecting the weight-restriction of every knapsack.

Note that the 0-1 variant only allows each item to be packed at most once. We show NP-completeness by reduction from a MKP instance to a DMP instance with a simple mapping from knapsacks to trains and items to departures. All physical train units have a DBM constraint that must be respected, this will be the map to the capacity of a knapsack. All departures assigned to a train consume some level of DBM and a profit/penalty is achieved/given, this corresponds to putting an item into the knapsack. The weight corresponds to the required level of DBM.

An illustrative example of the transformation is shown in Figure 2. The knapsack example contains four items and three knapsacks with individual knapsack capacities where every item can be packed at most once. The resulting DMP instance is shown in Figure 3.

Consider a MKP instance with N items and M knapsacks, let the weights and profits respectively be defined as w_i and p_i for every item i, and finally the capacities as c_j for every knapsack j.

We construct a DMP instance with M initial (or arrival) trains and N linked departures. Each individual train has remaining DBM corresponding to one of the capacities c_j of a knapsack. We set the remaining TBM to some high value such that this constraint is never binding. The departure and arrival times are set such that any of the four trains could be matched to any of the four linked departures, provided that it has sufficient remaining DBM. Any train can also visit any subset of the departures. The cost of matching every respectable (linked) departure is set to the negative profit of the item corresponding to each departure in the MKP instance. The cost of cancelling a departure is set to zero, and w.l.o.g. no maintenance is allowed. We can ensure that no maintenance will be performed by enforcing zero allowed maintenance operations, or by setting a high cost on maintenance, or by setting the replenished value to zero after maintenance. No train-reuse costs are defined. The initial trains are of the same single category, and all departures (of the linked departures) are compatible with only that train type. The replacement train of the linked arrivals have a different (incompatible) train type, and therefore do not influence the solution or quality in any way. Thus it is only possible for the initial trains to be matched to any of the linked departures. The optimal solution of the DMP instance will therefore only match the linked departures with the physical trains starting as the specified initial trains. Further, the matching-sequences are restricted by the given capacities of the MKP instance, where every matching/item consumes the specified knapsack weights. Due to the departure constraint, every linked departure can be matched at most once, thus every item is at most assigned to one knapsack. If any train is matched to linked departure i, $-p_i$ is added to the objective and inflicts a consumption ($reqDBM_i$) of w_i to the DBM of the train. As no costs, but the profits of the MKP instances are present, the optimal solution value will correspond to the optimal value of the MKP instance, only with a negative sign. Alternatively, in order to more closely follow the DMP objective function we adopt in later sections, the profit can be reformulated in terms of minimizing the cost of cancellations. A item of the MKP instance is only picked in a particular knapsack iff the corresponding linked departure is matched in the DMP instance to a particular train (initially, or as a returning linked arrival of that train). In conclusion, by transforming (in polynomial time and resources) an instance of the MKP is solved by solving the constructed DMP instance. The transformation graph is polynomial in the number of vertices and edges. The number of vertices is $1 + M$. The graph is acyclic and every node only connects forward in time (with respect to arrival and departure times) which results in a total of $\sum_{k=1}^{N} k = {}^{1}\!/_{2}(N^2 + N)$ edges.

Theorem 1. *The DMP is NP-hard*

Proof. It is straightforward to verify whether a solution to the DMP is feasible or not. This is verified by checking that the DBM and TBM constraints are valid, that the train types are compatible, and that the matched arrival and departures times are respected by the train units. Likewise, it is simple to calculate the objective cost of any given feasible solution. The number of operations and resources used to construct the DMP instance is polynomial. The number of vertices and edges created are polynomial in the MKP instance size. By reduction from the MKP, as described in the example, we have argued that there is a one-to-one correspondence between the solution of the constructed DMP instance and the MKP instance. Any feasible train matching solution to the DMP instance is a feasible item selection for the MKP instance, and vice versa. □

5 Mixed Integer Program Model

In this section we present a MIP mathematical model for the DMP. The size of the proposed MIP model for the DMP is polynomial in the number of input trains and departures, and can be solved using a commercial solver.

The model contains six types of variables. A set of binary variables m_t^d determine whether train $t \in \mathcal{T}$ is matched to departure $d \in \mathcal{D}$. Matches that are not present in the compatibility set ($CompDep(t)$ or $CompTr(d)$) are omitted or fixed to zero. We also introduce a set of binary variables cat_t^i that indicate if train $t \in \mathcal{T}$ is of category $i \in \mathcal{C}$. For all initial trains and non-linked arrival trains the category is known and the corresponding variable can be fixed (or omitted). The continuous variables dbm_t and tbm_t determine the DBM and TBM of train $t \in \mathcal{T}$ at the time before departure, which is the initially available DBM/TBM or $maxDBM_t/maxTBM_t$ if maintenance is performed on the train. Finally we introduce the binary variables f_t^d and g_t^d that determine whether a train $t \in \mathcal{T}$ matched to $d \in \mathcal{D}$ is being maintained on DBM or TBM. In the following we will assume that maintenance of DBM and TBM can only be done on one specific known day, for a particular matching. We introduce a binary parameter $\omega_{t,day}^d$ that is 1 if a match between t and d would perform maintenance on day (if it performs it at all), and 0 otherwise. The objective is formulated as a minimization of the number of unmatched departures and the cost of missed train reuse:

$$\begin{aligned} \min \sum_{d \in D} & cancellationCost_d \cdot c_d \\ & - \sum_{u \in \mathcal{U}} reuseCost \cdot m_{tr_u}^{dep_u} \\ & + reuseCost \cdot |\mathcal{U}| \end{aligned}$$

We minimize the number of cancelled trains and maximize the number of reuses. Note that the final term is constant and can be left out. The constraints of the model are the following:

$$\sum_{t\in\mathcal{T}} m_t^d \geq 1 - c_d \qquad d\in\mathcal{D} \tag{1}$$

$$\sum_{t\in\mathcal{T}} m_t^d \leq 1 \qquad d\in\mathcal{D} \tag{2}$$

$$\sum_{d\in\mathcal{D}} m_t^d \leq 1 \qquad t\in\mathcal{T} \tag{3}$$

$$\sum_{t\in\mathcal{T}}\sum_{d\in\mathcal{D}} \left(\omega_{t,day}^d f_t^d + \omega_{t,day}^d g_t^d\right) \leq maxMaint \qquad day\in\mathcal{H} \tag{4}$$

$$f_t^d + g_t^d \leq 2m_t^d \qquad t\in\mathcal{T}, d\in\mathcal{D} \tag{5}$$

Constraints (1) ensure that every departure is assigned (to some train), unless there is a cancellation. Constraints (2) ensure that at most one train is assigned to every departure. Constraints (3) ensure that each train is assigned at most once. Constraints (4) ensure that the total number of maintenance operations (every day) is respected. Constraints (5) prohibit maintenance usage on a match if that match is not made.

$$\sum_{i\in\mathcal{C}} cat_t^i = 1 \qquad t\in T \tag{6}$$

$$m_t^d \leq \sum_{i\in compCatDep_d} cat_t^i \qquad t\in\mathcal{T}, d\in\mathcal{D} \tag{7}$$

$$cat_t^i \geq cat_{t'}^i + m_{t'}^{\sigma(t)} - 1 \qquad t\in\mathcal{T}, t'\in\mathcal{T}\setminus\{t\}, i\in\mathcal{C} \tag{8}$$

$$cat_t^{i_t} \geq c_{\sigma(t)} \qquad t\in\mathcal{T} \tag{9}$$

Constraints (6) ensure that every train is assigned exactly one category. Again, the constraint for initial trains and non-linked arrival trains can be omitted, if the correct value is fixed. Constraints (7) ensure that trains cannot be assigned to departures where the category is not compatible. Constraints (8) ensure that if train t' is matched to the linked departure $\sigma(t)$ of train t, then train t inherits the category i of train t'. Finally, constraints (9) ensure that, if the linked departure $\sigma(t)$ of train t is not covered, then train t has the category i_t of its replacement train.

$$\begin{aligned} dbm_t \leq & \sum_{t'\in\mathcal{T}\setminus\{t\}} \left(dbm_{t'}^{\sigma(t)} - reqDBM_{\sigma(t)}\right)\cdot m_{t'}^{\sigma(t)} \qquad t\in\mathcal{T} \\ & + c_{\sigma(t)}\cdot remDBM_t \\ & + \sum_{d\in\mathcal{D}} f_t^d \cdot M \end{aligned} \tag{10}$$

$$dbm_t \leq \sum_{i\in\mathcal{C}} maxDBM_i \cdot cat_t^i \qquad t\in\mathcal{T} \tag{11}$$

$$0 \leq dbm_t - m_t^d \cdot reqDBM_d \qquad t\in\mathcal{T}, d\in\mathcal{D} \tag{12}$$

Constraints (10) ensure that the available DBM for a train t is correct. The right hand side consists of three terms. The first term counts the contribution from trains that are linked to t. If the train is cancelled the term sum is zero, and the second term will contribute with $remDBM_t$ which is the *new* train inserted in case of a cancellation of departure $\sigma(t)$. The third term makes the constraint non-binding if maintenance is performed; the constant M is a big number that is no less than the maximum of $maxDBM_i$ for all $i\in\mathcal{C}$. For the sake of clarity the constraints have been presented using a non-linearity in the first term. In order to maintain a linear model we replace the constraints with linear constraints described in section 5.1. Since the train category i of a train t might be unknown we further constraint the DBM of the train to respect the $maxDBM_i$ in

Constraints (11). Constraints (12) make sure that enough DBM is available for a matching. For all matchings that are not active the constraint is just requiring dbm_t to be non-negative.

We will not present the corresponding constraints for TBM since they are analogous to Constraints (10), (11) and (12).

5.1 Reformulating the Nonlinear Constraints

The first term on the right hand side of Constraints (10) is non-linear and can be remodelled as a linear constraint by adding one more group of continuous variables and additional constraints. We introduce one new auxiliary variable κ_t^{dbm} for each train t that measures the DBM contribution of the train linked to departure $\sigma(t)$. We replace Constraints (10) with the following set of constraints:

$$dbm_t \leq \kappa_t^{dbm} + c_{\sigma(t)} \cdot remDBM_t + \sum_{d \in \mathcal{D}} f_t^d \cdot M \qquad t \in \mathcal{T} \qquad (13)$$

$$\kappa_t^{dbm} \leq dbm_{t'} - m_{t'}^{\sigma(t)} \cdot reqDBM_{\sigma(t)} + \sum_{d \in \mathcal{D} \setminus \sigma(t)} m_{t'}^d \cdot M + \sum_{t'' \in \mathcal{T} \setminus t'} m_{t''}^{\sigma(t)} \cdot M \qquad t \in T, t' \in \mathcal{T} \setminus \{t\} \qquad (14)$$

$$\kappa_t^{dbm} \leq M \cdot (1 - c_{\sigma(t)}) \qquad t \in T \qquad (15)$$

The three terms of Constraints (13) are analogous to the three terms of Constraints (10), except the first non-linear term is replaced with the new contribution term κ_t^{dbm}. Constraints (14) have four terms defining an upper bound on the linked contribution for train t. The first two terms are relevant if train t' is matched to the linked departure $\sigma(t)$ and ensure the contribution is no greater than the difference between the DBM for t' and the required DBM for departure $\sigma(t)$. The third term loosens any bound on κ_t^{dbm} related to t' if t' is matched to some departure other than $\sigma(t)$. The fourth term loosens the bound on κ_t^{dbm} related to t' if some other train t'' is matched to $\sigma(t)$. Finally, Constraints (15) make sure that the contribution is zero if departure $\sigma(t)$ is cancelled.

5.2 Reducing the Model

In order to simplify (and streamline) the model description we treated every train as a linked arrival train. However, some arrival trains are not linked, and some trains are initial trains already in the system. For both types of trains the constraints can be simplified, and for an instance with few linked arrivals the model may be reduced substantially. The variables cat_t^i decide the category for a train t. For initial trains and non-linked arrivals the category is set as a problem parameter, and so for such trains Constraints (6) are not necessary. Constraints (7) can be removed and replaced by setting the upper bound to zero for any departure for which the known category of t is incompatible. Constraints (8)-(9) are unnecessary.

A non-linked arrival or an initial train t has no DBM or TBM contribution from any previous train. Constraints (14) can have the κ_t^{dbm} term removed, and we can set $c_{\sigma(t)} = 1$ (because instead train t will always take its own remaining DBM). Constraints (14)-(15) are unnecessary in such cases.

6 Column Generation Model

The formulation presented in section 5 models every initial and arrival train individually even if some of them constitute a sequence of linked arrivals and departures and therefore essentially represents a single physical train unit. In this section we present a formulation that models each physical train unit using a single variable. The number of possible variables grows more than exponentially by the number of present linked arrivals. Initially every train can be matched to any linked departure (i.e. departure with a linked arrival), and continue to any other linked departure, before it finally reaches its final departure. The number of potential paths is bounded by $O(|T| \cdot |L|! \cdot |D|)$, where $|L|$ is the number of linked arrivals/departures. We therefore propose a column generation approach for solving the model. Column generation is a well-described technique used with success for solving MIP problems, e.g. the Vehicle Routing Problem with Time Windows (VRPTW) [1, 6, 10]. It is assumed that the reader is familiar with column generation solution methods.

The model contains two types of variables. For every departure $d \in \mathcal{D}$ we introduce a binary variable c_d that indicates whether d is cancelled or not. For every possible train unit pattern $p \in \mathcal{P}$ we have a binary variable λ_p that indicates whether pattern p is used or not. A pattern represents a sequence of linked arrivals and departures. The objective is formulated as a minimization of the number of unmatched departures and cost of missed train reuses :

$$\begin{aligned}
\min \sum_{d \in D} & \mathit{cancellationCost} \cdot c_d && && (16)\\
& - \sum_{u \in \mathcal{U}} \sum_{p \in \mathcal{P}} \mathit{reuseCost} \cdot \alpha_p^{dep_u} \cdot \beta_p^{arr_u} \cdot \lambda_p \\
& + \mathit{reuseCost} \cdot |\mathcal{U}|
\end{aligned}$$

$$\sum_{p \in \mathcal{P}} \alpha_p^d \lambda_p \geq 1 - c_d \qquad d \in \mathcal{D} \qquad (17)$$

$$\sum_{p \in \mathcal{P}} \alpha_p^d \lambda_p + \sum_{p \in \mathcal{P}} \phi_p^d \lambda_p \leq 1 \qquad d \in \mathcal{D} \qquad (18)$$

$$\sum_{p \in \mathcal{P}} \beta_p^t \lambda_p + \sum_{p \in \mathcal{P}} \varphi_p^t \lambda_p \leq 1 \qquad t \in \mathcal{T} \qquad (19)$$

$$\sum_{p \in \mathcal{P}} \mathit{maint}_p^{day} \cdot \lambda_p \leq \mathit{maxMaint} \qquad \mathit{day} \in \mathcal{H} \qquad (20)$$

$$\lambda_p \in \{0, 1\} \qquad p \in \mathcal{P} \qquad (21)$$

$$c_d \in \{0, 1\} \qquad d \in \mathcal{D} \qquad (22)$$

The α_p^d is a binary coefficient that indicates whether departure d is covered by pattern p. The β_p^t is a binary coefficient that indicates whether train t is used by patten p. Therefore $\alpha_p^{dep_u}$ indicates whether departure $dep_u \in \mathcal{D}$ is covered by pattern p, and $\beta_p^{arr_u}$ whether train $dep_u \in \mathcal{T}$ is used by p. Finally, ϕ_p^d and φ_p^t are binary coefficients that indicate whether a departure d or train t is blocked as a result of pattern p. A train is blocked by a pattern if the final matching of the pattern ends (or terminates) on a departure that is linked to some arrival. No other pattern may use this arrival-train, as it would mean that two patterns are using the same physical train without keeping proper score on train type, DBM and TBM. A departure is likewise blocked by a pattern if it starts using a train of a linked arrival. This corresponds to using one of the replacement trains which assumes (or requires) that the linked departures was cancelled. Essentially this means that the departure must be blocks if such a pattern is used. Finally, the coefficient $\mathit{maint}_p^{day} \in \mathbf{Z}^+$ indicates the number of maintenance operations performed on day *day* using pattern p.

Constraints (17) ensure that every departure is assigned (to some train), unless there is a cancellation. Constraints (18) ensure that at most one train is assigned to every departure, and also blocks departures for trains that assume that the departure is cancelled. Constraints (19) ensure that each train is assigned at most once. Trains are blocked by patterns if they use the corresponding linked departure. Constraints(20) ensure that the total number of maintenance operations (every day) is respected. For comparability it is here assumed that, given a matching (t,d), the day that a maintenance operation is performed is fixed. The day of maintenance operations can however be made a choice in the subproblem without difficulty. Finally, Constraints (21) and (22) show the variable domains. Note that Constraints (22) can be relaxed as Constraint (17) ensures that the variables are *naturally* binary in all feasible solutions.

The number of variables in the model is exponential by the number of linked departures, however compared to the MIP model present earlier the number of constraints is reduced to

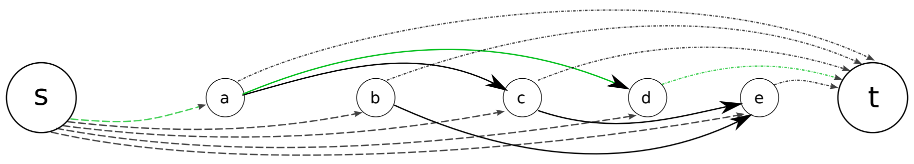

Figure 4: An illustration of the underlying graph for the matching subproblem. Every arc represents a match between a train and a departure. A feasible path (or linked sequence pattern) starts in the source s and ends in the sink/target t traversing a set of arcs. The feasible path (s, a, d, t) is shown in green. A linked sequence can start (or terminate) with many different trains (or departures) thus the dashed arcs illustrate where multiple arcs exist with the same origin and destination vertex.

$O(|\mathcal{D}| + |\mathcal{T}| + |\mathcal{H}|)$.

For obtaining the optimal *integer* solution to this problem (using column generation) a Branch and Price (B&P) framework can be adopted. In our solution method we will however only add columns in the root node and use Branch and Bound (B&B) for finding the best solution using the generated columns. In general this produces solutions of high quality, but no guarantee of optimality can be given in general. The gap between the found solution and the LP solution (of the root node) gives a bound on the optimality gap. It is left as future research to develop a full B&P framework.

6.1 Column Generation Subproblems

We distinguish between generating two families of columns. The first family of variables consists of patterns containing only one train-to-departure matching. It is relatively easy to enumerate all choices in this family and produce columns with the most negative reduced costs. In our implementation we pre-generate all columns of this family. Therefore, for the next family of variables, assume that the following patterns consist of at least two train-to-departure matches.

The subproblem can be split into one subproblem per train category. This transformation will both simplify and reduce each individual problem as the number of compatible departures is lower. The complexity of a labeling algorithm is also reduced since it is no longer needed to keep track of the train category when extending arcs. An additional advantage is that all subproblems can then be solved in parallel.

The subproblem consists of solving a Resource Constrained Shortest Path Problem (RSCPP). The underlying graph consists of one node per linked arrival in addition to one source and one sink node, see Figure 4. The arcs constitute matching choices. Three types of arcs are added. First, arcs originating from the source to every node in the graph represent compatible trains that are matched to the linked departure of the node. Second, arcs are added between nodes that represent compatible linked continuations, i.e., linked arrival/departures that connect to another linked arrival/departure. An example: in the (s, a, d, t) path the departure of the initial train matching (represented by arc (s, a)) is linked to a arrival (node a). The departure of the next matching (arc (a, d)) is connected to another linked arrival (node d). The departure of the last matching is not linked to any arrival, and thus the sequence ends. As described earlier, some patterns (represented by the path) may block other trains or departures.

More formally, we construct a directed non-cyclic graph $G(V, A)$ for every category $c \in \mathcal{C}$. Let $v_\pi \in V$ denote the *source* vertex, and $v_\omega \in V$ denote the *sink* vertex. Finally, we construct one vertex $v_{la} \in V$ per compatible linked arrival $la \in \{a \in \mathcal{L} | c \in \mathit{compCatDep}_a\}$. The arcs in the graph represents a train and departure match. For every train t of category c ($\{t \in \mathcal{T} | cat_t = c\}$) an arc $a_{(\pi,la)} \in A$ connects v_π and to v_{la} if the distance between availability time of t and departure time of departure $\sigma(la)$ is sufficient[1]. Likewise for every departure d compatible with category

[1]This could depend on multiple factor such as expected/minimum routing time, maintenance time and dwell time. For the sake of simplicity and comparability reasons we only require that the train availability time is less (or

c $(\{d \in \mathcal{D} | c \in compCatDep_d\})$ an arc $a_{(ld,\omega)} \in A$ connects vertex v_{la} with v_ω, again only if the necessary time is available. Finally for every pair of linked arrivals (la_1, la_2) an arc $a_{(la_1,la_2)}$ connects the train of la_1 to departure $\sigma(la_2)$ if it respects the required time. No edge directly connects the v_π to v_ω.

Observation 1. *Any path originating in π and terminating in ω represents a feasible matching.*

Proof. Every path consists of an initial arc and an terminating arc, and optionally multi intermediate arcs. All arcs represent matchings that are possible to make. Since we only have one category per subproblem, the construction of the graph inherently ensures that the train category is compatible with the assigned departures. Since we assume that maintenance can be performed on any matching, the required DBM and TBM can be fulfilled. □

Observation 1. *All possible matchings of at least two departures are valid paths in the constructed graph, originating in π and terminating in ω.*

Proof. All feasible matchings, w.r.t. category and time, are present in the graph. All initial matchings are represented as arcs originating in the source π. All intermediate matchings are present as arcs connecting the non-terminal nodes. Finally, all possible terminations of linked sequences are represented as arcs terminating in the sink ω. □

The subproblem can now be defined as the problem of finding the minimum cost path originating from v_π and to v_ω, given some edge costs, maintenance costs and restrictions. Every edge has a primal cost corresponding to the objective in the master problem, i.e., zero if no train reuses are satisfied or $-(n \cdot reuseCost)$ cost if n reuses have been satisfied by the path. A dual cost also appears on every edge that depends on the dual values given after solving the master problem in each iteration. The dual value of (17) and (18) are added to all arcs that include the corresponding departure. The dual of (19) is added to arcs including the corresponding train. The maintenance restrictions relate to the DBM and TBM restrictions. These values must be positive at all times, and all edges either increase or decrease these values. A path starts with values of 0 for DBM and TBM. All arcs originating from the source increase the values as indicated by the $remDBM$ and $remTBM$ on the train (of the arc). Other edges only decrease the values as indicated by $reqDBM$ and $reqTBM$ of the departures. Before extending an arc maintenance can be performed, which replenishes the DBM and/or TBM levels, but this comes at a cost of the corresponding dual of (20).

The problem is a RSCPP due to the maintenance constraints. In addition to the objective coefficients, the duals from Constraints (17) and (18) are added to arcs of the corresponding departure, and the duals from Constraint (19) are added to arcs of the corresponding trains. The appropriate dual from Constraints (20) is added every time a maintenance operation is scheduled.

The subproblem can be formulated as a mathematical problem:

equal) to the departure time.

$$min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{(i,j)\in A} \beta_{ij}y_{ij} + \sum_{(i,j)\in A} \alpha_{ij}z_{ij} \tag{23}$$

$$s.t. \sum_{(i,j)\in\delta^+(\pi)} x_{ij} = 1 \tag{24}$$

$$\sum_{(i,j)\in\delta^-(\omega)} x_{ij} = 1 \tag{25}$$

$$\sum_{(i,j)\in\delta^+(v)} x_{ij} = \sum_{(i,j)\in\delta^-(v)} x_{ij} \qquad \forall v \in V \setminus \{\pi,\omega\} \tag{26}$$

$$\sum_{(i,j)\in\Pi_v} reqDBM_{ij} \cdot x_{ij} + maxDBM \cdot y_{ij} \geq 0 \qquad \forall v \in V \tag{27}$$

$$\sum_{(i,j)\in\Pi_v} reqTBM_{ij} \cdot x_{ij} + maxTBM \cdot z_{ij} \geq 0 \qquad \forall v \in V \tag{28}$$

$$x_{ij} \in \{0,1\} \quad y_{ij} \in \{0,1\} \quad z_{ij} \in \{0,1\}$$

Where $\delta^+(v)$ and $\delta^-(v)$ respectively denote the outgoing and ingoing edges of vertex v. The binary variables x_{ij} define whether flow is used on edge (i,j) in shortest path, and y_{ij} and z_{ij} respectively determine whether DBM and/or TBM is performed on arc (i,j). Constraints (24)-(26) constitute the flow-conservation in a shortest path formulation. Constraints (27) and (28) ensure that the sufficient DBM and TBM is available - these levels can never be negative. Note that the graph is acyclic which means that we know all possible edges that can appear before reaching any vertex v - we denote Π_v as the set of all such edges. The constraints thus ensure that if a matching is made, then the DBM/TBM levels on any edge (leading up to v) must be non-negative.

6.1.1 Labeling Algorithm

As an alternative to solving the mathematical model of the sub-problem directly, we propose a dynamic programming approach for finding the optimal paths for the resource constrained shortest path. We refer to Irnich [5] for a more in-depth description of a this topic.

The labeling algorithm is similar to a shortest path algorithm that uses full enumeration, e.g. using a Breath First Search (BFS) strategy, to find the minimum cost path. In addition, we also need to respect some side-constraints. In our method, a label is a partial path from the source to some intermediate vertex that also keeps track of the total reduced cost, remaining DBM, remaining TBM and performed maintenances. Every arch has a primal and dual cost and a required level of DBM and TBM.

Initially, we generate the empty label at the source. In every iteration of the labeling algorithm, we pick one label at some vertex v and extend it. When extending we generate new labels from every outgoing edges from v. Due to the possibility of performing maintenance, we generate multiple labels for every outgoing edge: one that does not perform any maintenance, one that only performs DBM maintenance, one that only perform TBM maintenance and one that performs both DBM and TBM maintenance. Labels are not extended if either DBM or TBM becomes negative when subtracting the required level of DBM and TBM since these represent prefixes of infeasible paths.

In order to reduce computational time, we introduce dominations rules. A label a is said to dominate another label b, if we can safely remove b without loosing optimality. By removing a label b we omit searching all paths that follow the matching pattern of b.

Domination 1. *A label ($cost_a$, $remDBM_a$, $remTBM_a$) at vertex v dominates ($cost_b$, $remDBM_b$, $remTBM_b$) at vertex v if $cost_a \geq cost_b$ and $remDBM_a \geq remDBM_b$ and $remTBM_a \geq remTBM_b$.*

Proof. The first label has lower cost and more DBM and TBM remaining. The second label cannot have any advantages in terms of future matchings or maintenance costs. Thus, no matter how the path continues from v, the first label will always be at least as good as the second label. □

Instance	Arrivals	Linked	Departures	Reuses
B1	1235	475	1235	804
B2	1235	475	1235	0
B3	1235	0	1235	0
B4	1780	722	1780	1089
B5	2153	720	2153	1089
B6	1780	722	1780	1089
B7	304	144	304	187
B8	304	144	304	187
B9	1967	860	1967	1226
B10	196	89	196	123
B11	1122	486	1122	726
B12	570	263	570	377

Table 1: A description of the tested instances. The columns show instance names, number of train arrivals, number of linked train arrival/departures, number of train departures and number of specified train reuses.

Finding the optimal path in the subproblem can be both time and space consuming. Since it is sufficient to find any one path with negative reduced cost, we will rely on generating heuristic columns initially. Only when no heuristic columns (with negative reduced cost) can be found, we solve using the exact labeling algorithm.

We adopt two variations of the full labeling algorithm to find heuristic negative reduced cost paths. The first variant allows no DBM nor TBM maintenance which efficiently limits the number of possible matchings. The second variant only allows at most one DBM and at most one TBM maintenance. This is motivated by the fact that one maintenance is likely sufficient when considering short planning periods, c.f., the provided data-set.

7 Benchmarks

The main characteristics of the tested instances are shown in Table 1. They correspond to the instances of the final phase of the ROADEF Challenge 2014. All tests are run on a dedicated machine with dual 2.66 GHz Intel Xeon E5345 processors and 24 GB of memory. Both processors have 4 physical cores supporting 2 threads per core. The IBM ILOG CPLEX version 12.5 optimization software is used for solving all Linear Programs (LPs) and MIPs. All solutions are verified to be correct (with respect to feasibility and solution cost) using a modified version of the provided *solution checker* [11]. In the modified version all constraints unrelated to the matching have been omitted.

The timelimit set in the ROADEF Challenge 2014 was 10 minutes. Within this limit the submitted algorithms had to perform both matching and routing within the station. The provided instances cover up to 7 consecutive days of arrivals and departures. The provided timelimit for the challenge seems a bit restricting given a planning horizon of several days. Due to the difficulty of the problem we target a time limit of 10-30 minutes.

An initial benchmark shows that both solution methods are unable to solve all instances to optimality within a few hours. Table 2 shows the details of the instances that were solved to optimality using the Column Generation Method (CGM). A few instances run out of memory before being able to solve the root relaxation to optimality. Given a high time-limit the MIP method is able to solve the same instanced except B12. We discuss details of the column generation later in this section.

Solving the root relaxation of the instances seems to be a hard in general. Given a time-limit of 30 minutes only four instances are solved, see Table 3.

					Runtime in seconds			
Instance	Obj	Generated	Cons	Vars	Sub	LP	MIP	Total
B3	0	0	3 749	229 841	0.0	3.3	21.0	25.0
B7	4 400	15 629	946	19 048	33.4	28.2	20.3	83.0
B8	4 400	17 070	946	19 027	27.4	25.9	21.1	76.0
B10	2 100	3 413	620	7 050	0.7	2.4	0.2	3.0
B12	14 700	25 017	1 753	47 258	22 912.6	81.0	52.3	23 115.0

Table 2: Instances solved using column generation. The columns show instance name, objective of found solution, number of generated column, number of constraints and variables in final program, and finally total runtime of subproblem, LP solving, MIP solving and total. The omitted instances (B1, B2, B4, B5, B6, B9, B11) were not solved.

			Runtime(s)	
Instance	LP Relaxation	Columns	Subproblem	LP
B1	74 278	363 378	255.9	1 484.1
B2	32 596	633 486	214.9	1 532.0
B3	0	0	0.0	3.6
B4	134 582	619 406	264.2	1 485.9
B5	132 582	429 643	191.9	1 567.7
B6	133 095	652 439	270.3	1 477.5
B7	3 400	28 980	10.6	52.4
B8	3 400	30 242	12.0	55.7
B9	222 156	399 799	253.4	1 508.2
B10	1 800	6 168	1.1	3.4
B11	34 735	285 340	393.4	1 368.3
B12	13 503	71 922	1 533.2	173.6

Table 3: Root relaxations results given a time limit of 30 minutes. The columns show the instance name, objective of final relaxation, number of generated columns, and time spent generating columns and solving the LP relaxations. Optimal results were found for B3, B7, B8 and B10.

	Using Column Generation			No Column Generation	
Instance	Columns	Time	Objective	Time	Objective
B1	278 229	1 207	93 200	1 213	254 600
B2	383 937	1 200	61 000	1 201	228 000
B3	0	26	0	5	0
B4	300 143	1 162	209 300	333	379 000
B5	248 919	1 203	221 100	87	370 900
B6	285 498	1 201	211 800	341	379 000
B7	30 245	78	4 300	81	73 200
B8	27 942	86	4 400	81	73 200
B9	277 485	1 207	252 900	1 215	456 700
B10	5 908	5	2 000	26	42 400
B11	171 118	1 215	80 600	1 210	258 000
B12	71 442	1 206	14 300	1 205	135 600

Table 4: Solutions found using the column generation approach in 20 minutes. Results are compared to using no column generation, i.e., allowing no linked sequences.

Instance	Objective	Constraints	Variables	Time
B3	0	1 151 899	706 138	1199.4
B7	3 400	119 708	42 400	31.3
B8	3 400	119 708	42 400	31.0
B10	1 000	72 160	20 440	56.0
B12	43 900	571 526	131 360	1204.5

Table 5: Solutions found using the MIP approach. This approach was unable to produce any feasible solution to the omitted (B1, B2, B4, B5, B6, B9, B11) instances.

7.1 Heuristic Results

Solving the problem instances using the exact methods proves to be very difficult and we therefore also investigate the performance of the methods in a heuristic context.

The next benchmark shows results obtained using CGM with a 20 minute time limit. We allocate 10 minutes for generating columns, followed by a 10 minute MIP solve (using CPLEX) of the master problem. Table 4 summarizes the results. The results are compared to no column generation, i.e., not allowing any linked sequences. It is observed that the result of disabling column generation seems to have a drastic effect on solution quality. This argues that ignoring linked sequences altogether is undesirable as it will drastically penalize the achievable objective. Solutions are now found for all instances in contrast to the exact approach, were only 4 instances where solved within the same timelimit. It is observed that the time is mostly spent solving LP relaxations, except for one case. Heuristic columns were used to speed up the subproblem process instead of solving the exact optimization problem in every iteration. Whenever no columns with negative reduced cost are found the method solves the exact problem. Preliminary results show that this approach is favourable as solving the exact subproblem is in many cases extremely time-consuming. Many columns are generated during the execution of CGM, and these are gradually increasing the master problem size. In future work, it might be interesting to remove unnecessary columns after a few iterations.

In our final benchmark we run the instances using the MIP Method (MIPM). As before we set a runtime limit of 20 minutes – the results are shown in Table 5. We note that this approach was unable to provide any solution for most instances. However, for instances B7, B8 and B10 MIPM was able to find slightly better results than CGM. A worse objective was found for the B12 instance. MIPM was unable to find solutions to all instances due to insufficient memory.

8 Conclusions

We have described and investigated the Departure Matching Problem which is identified as a crucial subproblem of the RSUM problem in the ROADEF/EURO Challenge 2014. Without explicitly considering the matching problem, too many departures will be uncovered.

We prove in section 4 that the DMP is NP-hard in the strong sense by reduction from the 0-1 Multiple Knapsack Problem.

We have proposed two methods for solving the DMP. We first presented a pure MIP formulation of the problem which could act as the least technical reference point for future studies. The model is however large in terms of variables and constraints and the benchmarks show that this model is unable to solve most of the proposed instances. Memory usage is one significant drawback of this method.

A second solution method based on column generation has also been presented. This model is simple and can without much difficulty be extended even further to handle more constraints. It is show how the subproblem can be split into several independent problems thereby reducing complexity and enabling parallelism. The benchmarks for this approach show that we can find good solutions fast, if the method is used with time limits. However, even solving the root node relaxation is shown to be difficult in multiple cases. We expect that the method could be improved if embedded in a B&P framework with more efficient handing of the columns. Furthermore, the performance of this method could be further improved if a good initial solution can be provided as a *hotstart* to CGM, e.g., the result of a heuristic method.

The considered data instances proved to be surprisingly hard to solve. The final results of the ROADEF challenge winners suggests that it is not possible to obtain a satisfactory solution to the overall problem, RSUM. There may not even exists a good solution, i.e., a solution without cancellations.

For the sake of simplicity it has been assumed that all matches are possible where the departure time occurs after the arrival time. In reality it might be more realistic to remove matching options where the time between arrival and departure is sufficiently high. Trains may arrival late, some time is required to perform routing inside the station and time is required to perform maintenance. Further, it may not be necessary to consider long matchings, e.g., an arrival on day 1 with a departure on day 7. Reducing the number of possible matchings will reduce the number of decision variables and constraints and likely improve the success-rate of solution methods. This may be a viable practical approach to ensure feasibility.

For future work there are a multiple matters worth considering. There are several column generation techniques that can be investigated to improve performance, e.g., dual stabilization, branch-and-price and reduced-cost fixing. The potential of heuristic methods for DMP is unknown and can possibly find good solutions fast, or even be used to speed up an exact approach. It would be interesting to consider new data instances where it is know that there exists at least one feasible solution without any cancellations.

References

[1] Cynthia Barnhart, Ellis L Johnson, George L Nemhauser, Martin WP Savelsbergh, and Pamela H Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[2] Richard Freling, Ramon M Lentink, Leo G Kroon, and Dennis Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261–272, 2005.

[3] Jørgen Haahr and Simon Bull. A Math-Heuristic Framework for the ROADEF/EURO Challenge 2014. Technical report, The Technical University of Denmark, 2014.

[4] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on computing*, 2(4):225–231, 1973.

[5] Stefan Irnich. Resource extension functions: properties, inversion, and generalization to segments. *OR SPECTRUM*, 30(1):113–148, 2008. ISSN 01716468, 14366304. doi: 10.1007/s00291-007-0083-6.

[6] Brian Kallehauge, Jesper Larsen, Oli B.G. Madsen, and Marius M. Solomon. Vehicle routing problem with time windows. In *Column Generation*, pages 67–98. Springer US, 2005. ISBN 978-0-387-25485-2.

[7] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems.* Springer, 2004.

[8] Leo G Kroon, Ramon M Lentink, and Alexander Schrijver. Shunting of passenger train units: an integrated approach. *Transportation Science*, 42(4):436–449, 2008.

[9] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[10] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005. doi: 10.1287/opre.1050.0234. URL `http://dx.doi.org/10.1287/opre.1050.0234`.

[11] François Ramond and Marcos Nicolas. *Trains don't vanish! ROADEF EURO 2014 Challenge Problem Description.* SNCF - Innovation & Research Department, 2014.