Roadef Challenge 2014: A Modeling Approach, Rolling stock unit management on railway sites

Nicolas Catusse, Hadrien Cambazard

ROADEF CHALLENGE 2014: A MODELING APPROACH

Rolling stock unit management on railway sites

Hadrien Cambazard and Nicolas Catusse

Laboratoire G-SCOP, Grenoble INP, France
{hadrien.cambazard|nicolas.catusse}@grenoble-inp.fr

1 Introduction

We report here our analysis of the problem presented in [5] and propose a methodology strongly based on modeling with MIP and CP technologies. Due to the complexity of the problem formulation, we believe that a robust engineering is easier to achieve by relying on models than dedicated code. Two core modeling ideas are presented for relating the **daily maintenances limit** and the **linked departures** to an assignment based MIP model. Additionnaly, a variant of the maximum matching problem lying at the heart of the problem is shown to be NP-Complete. Intermediate experimental results are given along the way to support the ideas reported.

Section 2 gives an overview of the proposed methodology. The two main components of this approach relying on a MIP model and a CP model are discussed in section 3 and 4 respectively.

2 Overview of the proposed method

We propose to decompose the problem into two natural stages of decisions.

Firstly, We suggest to handle the **assignment** of the arrival's trains to the departures as well a the choice of platforms for all events with a MIP solver. The rationale is twofolds: this first problem has a strong matching component for which MIP technologies are efficient and the objective function can be nearly entirely expressed on this first part. Two key modeling ideas are presented to integrate **daily maintenances limits and linked-arrivals** naturally in the matching based MIP model.

Secondly, we believe that **the routing** of trains through the station can be computed without the need of really questionning the assignment first step. This was the case on the first problem formulation. From an engineering point of view as well as efficiency, we believe that CP is the best technology to implement this part and adress the routing under time-windows constraints.

We were not able to fully implement the proposed methodology but will give preliminary experimental results that support our analysis of the problem.

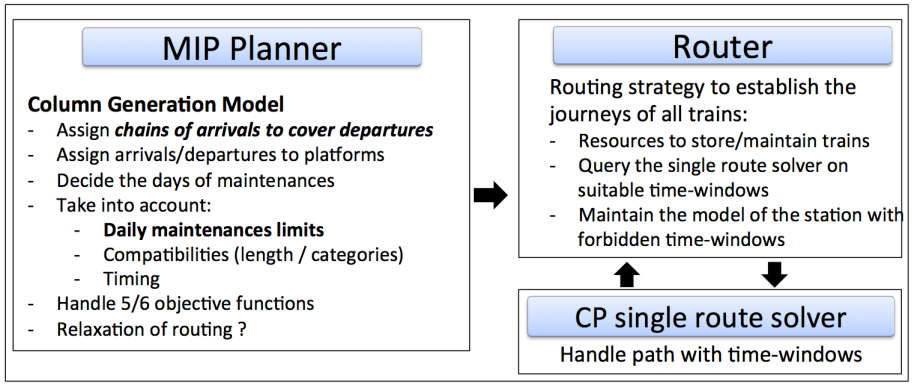

Fig. 1. Overall view of the proposed solution approach.

3 Assignment of trains, platforms and maintenances

We chose to give the most important lines of the MIP model used and skip the details that we believe do not prevent to understand the approach (and reproduce it) but would significantly burden the readibility due the complexity of the problem.

3.1 Notations

We use the notations introduced in [5] regarding the problem definition. Additionally, a bipartite graph is denoted $G(A, B, E)$ and the set of edges incident to a vertex $x \in A \cup B$ is denoted $\Gamma(x)$. If G is oriented the notation Γ is extended so that $\Gamma^+(x)$ and $\Gamma^-(x)$ are the set of vertices that are incident to arcs respectively leaving and entering vertex x: $\Gamma^+(x) = \{y \mid (x, y) \in E\}$ and $\Gamma^-(x) = \{y \mid (y, x) \in E\}$. This notation is generalized to sets and for $X \subseteq A$, we denote by $\Gamma^+(X) = \cup_{x \in X} \Gamma^+(x)$. Finally for $Y \subseteq B$ we denote $\Gamma^{-1}(Y) = \{x \in A \mid \Gamma^+(x) \subseteq Y\}$ the set of vertices of X that have **all** their neighours in Y.

3.2 Model restricted to *unlinked* arrivals

In this section we design a model under the assumption that all arrivals are known and do not depend on decisions made for earlier departures *i.e* **we assume that $\mathcal{A}$ does not contain any linked-arrivals**. Under this assumption, the category and all associated features of an arrival are fully known. We then show how the model presented for unlinked arrivals can be generalized to linked arrivals.

1. **Assignment of arrivals to departures.**

   We first present the variables encoding the assignment of arrivals to departures:

$y_j \in \{0,1\}$: indicates if departure $j \in \mathcal{D}$ is **covered** (assigned to an arrival).
$w_i \in \{0,1\}$: indicates if arrival $i \in \mathcal{A}$ is **covered** (if yes it must go on a platform).
$x_{ij} \in \{0,1\}$: indicates whether arrival $i \in \mathcal{A} \cup \mathcal{T}$ is **assigned to departure** $j \in \mathcal{D}$.
$u_{gj} \in \{0,1\}$: indicates if departure $j \in \mathcal{LD}$ is **of group category** g.

The key assignment variables are therefore the x_{ij} variables. The possible assignment of a train (an initial or an arriving train) to a departure is a-priori limited by three elements (chronology, maximum maintenance capacities and categories). For a given pair of train $i \in \mathcal{A} \cup \mathcal{T}$ and departure $j \in \mathcal{D}$, we directly state $x_{ij} = 0$ (infeasible assignment) if one of the three conditions hold:

– **Chronology**: The time between the arrival and departure is not large enough to accomodate for any maintenance and junction/disjunction needed as well as routing through the station. The routing is represented by a unique constant b_1 representing a time **buffer** that we estimate initially.

$$\begin{gathered} arrTime_i + \\ \mathbb{1}_{remDBM_{t_i} < reqDBM_j} maintTimeD_{cat_{t_i}} + \\ \mathbb{1}_{remDBM_{t_i} < reqDBM_j} maintTimeT_{cat_{t_i}} + \\ \mathbb{1}_{i \in \mathcal{J}_{arr}} disjTime \times (|jaList_i| - 1) + \\ \mathbb{1}_{j \in \mathcal{J}_{dep}} (junTime \times (|jdList_j| - 1) + minAsbTime) + \\ b_1 > depTime_j \end{gathered}$$

Note that the full disjunction/junction time is included whenever i/j are respectively part of a joint arrival/departure. This can be refined (or removed in particular to use the MIP for a lower bound) when the assignmengt i to j can potentially be done without these operations (their neighbhoring rolling units are potentially compatible in the corresponding convoys).

– **Maximum maintenance capacity**: The trains don't have enough maintenance capacity even if maintenance is done.

$$maxDBM_{cat_{t_i}} < reqDBM_j \qquad \text{or} \qquad maxTBM_{cat_{t_i}} < reqTBM_j$$

– **Categories**: Categories are not compatible.

$$cat_{t_i} \notin compCatDep_j$$

Constraints (1)-(2) channels x, w and y variables and also enforce an arrival to be assigned to at most one departure; (2) channels u and y variables enforcing exactly one **group** category for a covered departure belonging to a joint; (3) channels x and u making sure that the arrivals assigned to a joint departure have the same corresponding category.

$$\begin{array}{lll} (1) & \sum_{j \in \mathcal{D}} x_{ij} \leq w_i & \forall i \in \mathcal{A} \cup \mathcal{T} \\ (2) & \sum_{i \in \mathcal{A} \cup \mathcal{T}} x_{ij} = y_j & \forall j \in \mathcal{D} \\ (3) & \sum_{g \in \mathcal{G}} u_{gj} = y_j & \forall j \in \mathcal{J}_{dep} \\ (4) & \sum_{i \in \mathcal{A} | catGroup_{t_i} = g} x_{ij} \leq |jdList_j| u_{gj} & \forall j \in \mathcal{J}_{dep}, \forall g \in \mathcal{G} \end{array}$$

2. **Assignment of arrivals/departures to platforms.**

We turn our attention to the assignment of platforms. We use another time buffer b_2 corresponding to a minimum time that the rolling unit should be able to stay on the platform without causing a crash. The following decision variables are added:

$pl_{ik} \in \{0,1\}$: indicates if **platform** $k \in \mathcal{P}$ **is assigned to arrival** $i \in \mathcal{A}$.
$pl_{jk} \in \{0,1\}$: indicates if **platform** $k \in \mathcal{P}$ **is assigned to departure** $j \in \mathcal{D}$.
$l_j \geq 0$: **total length of the convoy** made for departure $j \in \mathcal{D}$.
$ct_{jc} \in \{0,1\}$: indicates if **category** $c \in \mathcal{C}$ **is assigned to departure** $j \in \mathcal{D}$.

The assignment of a platform $k \in \mathcal{P}$ to an arrival $i \in \mathcal{A}$ is a-priori restricted by four elements. We state $pl_{ik} = 0$ if:
- **Length**: $length_k < length_{t_i}$
- **Categories**: $cat_{t_i} \notin compCatRes_k$
- **Imposed consumption**: $l \in \mathcal{I}$ and $[beg_l, end_l - 1] \cap [arrTime_i, arrTime_i + minResTime + b_2] = \emptyset$
- **Arrival sequence**: the sequence of i ($arrSeq_i$) is not connected to platform k.

Similarly **Imposed consumption** and **Departure sequence** can be used to prefix pl_{jk} to 0. The model is now shown below. Constraints (5) states that a single category is assigned to a covered departure; (6) channel departures category with the arrival assigned; (7) and (8) enforces a single platform to be chosen for the a covered arrival/departure; (9) channels the length of the train of a departure and to the one of the chosen arrival; (10) ensures the departure's platform is long enough; (11) ensures that a departure's platform is compatible with its category; (12) make sure that all arrivals in the same joint arrival are assigned to the same platform.

$$
\begin{array}{lll}
(5) & \sum_{c \in C} ct_{jc} = y_j & \forall j \in \mathcal{D} \\
(6) & ct_{jc} \geq x_{ij} & \forall j \in \mathcal{D}, \forall c \in \mathcal{C}, \forall i \in \mathcal{A} | cat_{t_i} = c, \\
(7) & \sum_{k \in \mathcal{P}} pl_{ik} = w_i & \forall i \in \mathcal{A} \\
(8) & \sum_{k \in \mathcal{P}} pl_{jk} = y_j & \forall j \in \mathcal{D} \\
(9) & l_j \geq \sum_{i \in \mathcal{A} \cup \mathcal{T}} length_{t_i} x_{ij} & \forall j \in \mathcal{D} \\
(10) & length_k pl_{jk} + L(1 - pl_{jk}) \geq l_j & \forall j \in \mathcal{D},\ \forall k \in \mathcal{P} \\
(11) & pl_{jk} \leq 1 - ct_{jc} & \forall j \in \mathcal{D},\ \forall k \in \mathcal{P}, \forall c \in \mathcal{C} | c \notin compCatRes_k \\
(12) & pl_{j_1 k} = pl_{j_2 k} & \forall j \in \mathcal{J}_{arr}, \forall (j_1, j_2) \in jaList_j,\ \forall k \in \mathcal{P}
\end{array}
$$

Finally two platforms can be shared by two **events** (an event is either an arrival or a departure) only if the time gap inbetween is large enough to free the platform. A platform k can not be shared by two *events* (a,b) (ordered chronologically) if :

- $a \in \mathcal{D}, b \in \mathcal{D}$: $depTime_b - depTime_a < idealDwell_b + b_2$
- $a \in \mathcal{A}, b \in \mathcal{A}$: $arrivTime_b - arrivTime_a < idealDwell_a + b_2$
- $a \in \mathcal{A}, b \in \mathcal{D}$: $depTime_b - arrivTime_a < idealDwell_a + idealDwell_b + b_2$
- $a \in \mathcal{D}, b \in \mathcal{A}$: $arrivTime_b - depTime_a < b_2$

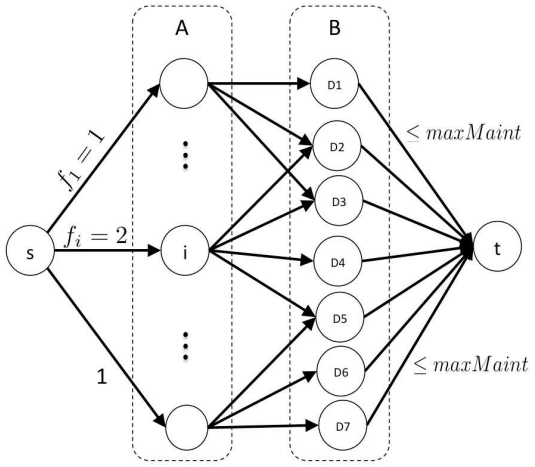

Fig. 2. An exemple a maximum flow problem encoding the choice of days for maintenances. On this example arrival i is performing two maintenances (time and distance).

So finally $\forall k \in \mathcal{P}, \forall (a,b) \in \mathcal{D} \cup \mathcal{A} \times \mathcal{D} \cup \mathcal{A}$, such that event a can not share a platform with event b, we simply state:

$$pl_{ak} + pl_{bk} \leq 1$$

The LP formulation can be strenghtened by stating these constraints over the maximum cliques of the incompatibility graph (but cplex does that automatically).

3. **Modeling daily maintenances limit.** We know if an arrival will require maintenance operations when it is assigned to a given departure, and we don't need to know the exact day it is performed to state the daily limits. This idea is also proposed by [1] and stated similarly with constraints (13). Let's denote by $M(d_a, d_b)$ the set of pairs (arrival, departure) (i,j) that require $o_{ij} \in \{1,2\}$ maintenance operations (distance and/or time) **and that takes place in a time-window** of days $[d_a, d_b]$. A pair (i,j) belongs to $M(d_a, d_b)$ if $arrTime_i$ and $depTime_j$ belong to the interval of days $[d_a, d_b]$. We enforce the daily limits on all possible sliding-timewindows of all sizes (in $\{1, \ldots, nbDays\}$) over the horizon:

$$(13) \qquad \forall d_a \in \{1, \ldots, nbDays\},\ \forall d_b \in \{d_a, \ldots, nbDays\},$$
$$\sum_{(i,j) \in M(d_a, d_b)} o_{ij} x_{ij} \leq (d_b - d_a + 1) \times maxMaint$$

We claim that the days of maintenances can be found afterwards by solving a maximum flow problem in a bipartite graph where arrivals are on the left side (an arrival i has an incoming flow o_{ij}) and possible days are on the right, each with a capacity of $maxMaint$. Figure 2 shows an example. We now give a proof of this claim.

Let's consider a *bipartite flow graph* $G(s,t,A,B,E)$ (figure 2 shows an example of such a graph) defined by a bipartite graph $G(A,B,E^{'})$ (edges are oriented from A to B) as well as two additionnal vertices, a source s and a sink t, the additionnal edges (s,i), $\forall\, i \in A$ and (j,t), $\forall\, j \in B$. Moreover a single capacity $Capa$ is added on all edges leading to the sink *i.e* all $(j,t),\ \forall\ j \in B$. We recall from 3.1 that for $Y \subseteq B$, $\Gamma^{-1}(Y) = \{x \in A \mid \Gamma^{+}(x) \subseteq Y\}$.

Lemma 1: Given a *bipartite flow graph* and an entering flow f_i for each edge (s,i) for all $i \in A$, the flow is feasible **if and only if** $\sum_{i\in\Gamma^{-1}(Y)} f_i \leq Capa \times |Y|$ for all $Y \subseteq B$.

Proof: The Hall's theorem states that a perfect matching (matching saturating A) exists **if and only if** for all $X \subseteq A$ we have $|X| \leq |\Gamma^{+}(X)|$. This can be generalized to flows. Typically the flow (defined by the f_i) is feasible in the bipartite flow graph if and only if $\forall\ X \subseteq A$, we have $\sum_{i\in X} f_i \leq Capa \times |\Gamma^{+}(X)|$. To prove it, we apply Hall's theorem in the bipartite graph where each vertex of $i \in A$ is replicated f_i times and each vertex $j \in B$ is replicated $Capa$ times. A feasible flow gives a maximum matching in the graph with replicated vertices by dividing the flow in units. Reversely by combining units, we can obtain the flows.
Now for any $X \subseteq A$ we can state $Y = \Gamma^{+}(X)$ (and so $\Gamma^{-1}(Y) = X$) and by assumption we have $\sum_{i\in X} f_i = \sum_{i\in\Gamma^{-1}(Y)} f_i \leq Capa \times |Y| = Capa \times |\Gamma^{+}(X)|$. Therefore there is a feasible flow since Hall's theorem is verified. Reversely, if we assume a feasible flow we obtain the condition from Hall's theorem.

A *convex bipartite flow graph* is a bipartite flow graph such that the underlying bipartite graph is *convex* over the set of vertices A. This means that there is an ordering of the vertices of A such that all neigbhors $\Gamma^{+}(i)$ of a vertex $i \in A$ *are consecutives* (figure 2 shows an example of such a convex bipartite flow graph).

Lemma 2: Given a *convex bipartite flow graph* and an entering flow f_i for each edge $(s,i), \forall i \in A$, the flow is feasible **if and only if** $\sum_{i\in\Gamma^{-1}(W)} f_i \leq Capa \times |W|$ for all consecutive subset W of vertices of B.

Proof: Consider any subset Y of B, Y is made of a number k of subsets of consecutives vertices so that $Y = Y_1 \cup Y_2 \cup \ldots \cup Y_k$. For each Y_l ($l \leq k$) we have $\sum_{i\in\Gamma^{-1}(Y_l)} f_i \leq Capa \times |Y_l|$ by assumption. Since the graph is convex, the neigbhors of any two parts of Y are disjoint so that $\Gamma^{-1}(Y_a) \cap \Gamma^{-1}(Y_b) = \emptyset$ for all pairs (Y_a, Y_b) of distinct parts of Y ($a \leq k, b \leq k$ and $a \neq b$). As a result we have $\sum_{i\in\Gamma^{-1}(Y)} f_i = \sum_{l=1}^{k} \sum_{i\in\Gamma^{-1}(Y_l)} f_i$ and we can state:

$$\sum_{i\in\Gamma^{-1}(Y)} f_i = \sum_{l=1}^{k} \sum_{i\in\Gamma^{-1}(Y_l)} f_i \leq \sum_{l=1}^{k} Capa \times |Y_l| = Capa \times |Y|$$

So by Lemma 1, the flow is feasible. Reversely, if the flow is feasible, the condition (on all subsets Y) of Lemma 1 holds and consequently the condition (on all con-

secutive subsets W) of Lemma 2.

Back to our problem, the bipartite graph of the arrivals and days is a convex bipartite flow graph (the set of possible maintenance days for an arrival is made of **consecutive days** without any *holes* and we set $Capa$ to $maxMaint$). The conditions stated in the MIP is the one of Lemma 2 and is enough to make sure feasible maintenance days are always found afterwards by solving the corresponding flow.

4. **Modeling the objective function.** We can model all elements of the objective function to the exception of the *platform usage costs*. The *uncovered arrivals/departures and unused initial trains costs* are easily modeled based on the y and w variables. The *non-satisfied preferred platform assignment cost* as well as the *non-satisfied train reuse cost* are directly encoded based on the pl and x variables. Similarly the *over-maintenance cost* of an arrival/departure pair (i, j) that require a maintenance is known and counted when the corresponding x_{ij} is set to 1. Modeling the *train junction and disjunction operation costs* requires more effort and intermediate variables must be introduced to represent each pair of consecutive train in a convoy. We do not give the full details here. We mention however that our model remains a relaxation of this cost as it does not take into account the need for junctions/disjunctions when maintenances have to be performed.

5. The MIP formulation presented here has a very strong relaxation in practice. The reason is that the core problem is an assignment problem and that the side constraints seem relatively easy to take into account for this benchmark. Despite the size of the formulation, the results[1] shown in Table 1 prove that the MIP model without linked arrivals can be solved to optimality (which is not really needed in practice) for the B instances. Half of the instances are solved without branching.

Table 1. Performances for solving to optimality (precision 10^{-3}) the B instances when ignoring the linked arrivals. Buffers (in seconds) were set to $b_1 = 2 \times revTime + 400$ and $b_2 = 30$.

instance	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
Total time (s)	36	37	308	90	386	329	6	3	156	4	306	7
Search space (nodes)	0	0	2249	0	0	1016	0	0	0	2	2481	0

3.3 Model generalized to *linked* arrivals

Generalizing the matching.

Consider the matching problem modelled with the x_{ij} variables and lying at the heart of the previous MIP formulation. We denote the underlying bipartite graph by $G =$

[1] Experiments were performed on a MacBook, running MacOSX 10.5.8, running on a 2GHz Intel Core 2 Duo CPU, with 2 GB 667 MHz DDR2 SDRAM.

$(\mathcal{A} \cup \mathcal{I}, \mathcal{D}, E)$. An edge (i, j) belongs to E if the aforementionned criteria (chronology, maximum maintenance capacities and categories) are satisfied for the pair (i, j).

The presence of linked-arrivals ($\mathcal{LA} \subseteq A$) alter the problem by introducing dependencies. Typically, the possible departures $\Gamma(i)$ that can be assigned to an arrival i can now depend on the assignment made on an earlier departure j referred here as a *linked departure*. We formalize the problem as follow:

The Dependent Matching Problem (DMP): Consider a bipartite graph $G(A, B, E)$ and let $D = \{x_1, \ldots, x_k\} \subseteq A$ and $C = \{y_1, \ldots, y_k\} \subseteq B$ be two subsets of respectively A and B so that $|C| = |D|$. D stands for Dependencies and C for Conditions. For each vertex x_i of D, we define $\tilde{\Gamma}(x_i) = \{\Gamma^{(a,y_i)}(x_i) \mid a \in \Gamma(y_i)\}$ so that each $\Gamma^{(a,y_i)}(x_i) \subseteq \Gamma(x_i)$ represents the possible neigbhors of x_i if a is matched to y_i. THE DEPENDENT MATCHING PROBLEM (DMP) is to determine if there is a complete matching M in G so that for all $i = 1, \ldots, k$:

- If x_i is matched to b of B and y_i is matched to a of A, we have $b \in \Gamma^{(a,y_i)}(x_i)$.
- If y_i is unmatched and x_i is matched to b, we have $b \in \Gamma(x_i)$.

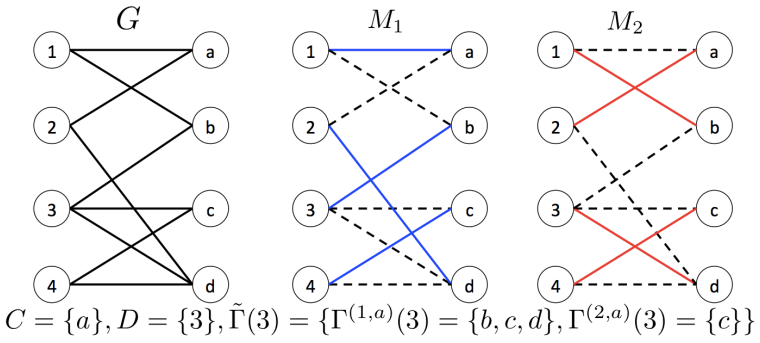

Fig. 3. An instance of DMP (left), a complete matching M_1 solution (middle), a complete matching M_2 which do not satisfy the dependencies.

Theorem: *DMP is NP-Complete*

Proof: It is easy to see that DMP belongs to the class NP.

We show a reduction from the SAT [2] problem to DMP. Let $\psi = C_1 ... C_p$ be a boolean expression in conjunctive normal form and $(x_1, ..., x_q)$ the variables of ψ. We define the bipartite graph as follows:

- For each literal x_i and each clause C_j wherein x_i appears, we create a gadget, named $x_i C_j$, consisting of a $K_{2,2}$ graph. Notice that this gadget has exactly two possible complete matchings: two parallel or two crossing edges. In the following, we will code the value x_i true with two parallel edges and $\overline{x_i}$ true with two crossing edges (see Fig.4).

– For each clause C_j with n literals, we create a gadget, named C_j, consisting of a $K_{n,1}$ graph, such as vertices on the left represent every literals in C_j.

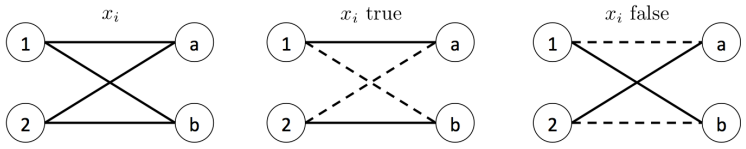

Fig. 4. Gadget for the literal x_i (left), set to true (middle) and false (right).

In order to diffuse the value of a literal in every clauses, we will build a chain with conditions between every gadgets on the same literal. We add the following condition on the lower right vertex (vertex b in Fig. 5) of the literal gadget: if the edge connected with this vertex is the crossing one, we add a dependance with the lower left vertex (vertex 4 in Fig. 5) of the next literal gadget (corresponding to the next occurence of this literal in a clause) in the chain to remove the parallel edge from its neighborhood ($\tilde{\Gamma}(4) = \{\Gamma^{(1,b)}(4) = \{c\}, \Gamma^{(2,b)}(4) = \{d\}\}$ in Fig. 5). We also create another dependance in the same way for parallel edges. Notice that with these conditions, if one literal gadget is set to true or false, every other literal gadget on the same literal must be set to the same value to obtain a complete matching. Then in order to avoid the selection of literal

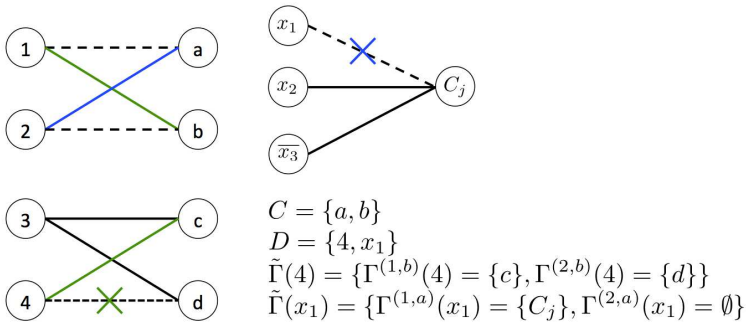

Fig. 5. Literal gadget set to false (upper left), diffusion to its clause gadget C_j (upper right), diffusion to the next literal gadget (lower left).

without the proper value in the clause, we add a condition on the higher right vertex (vertex a in Fig. 5) of the corresponding literal gadget: if the clause C_j has a positive literal x_i (resp. a negative literal $\overline{x_i}$) and the edge connected with this vertex is the crossing one (resp. the parallel one), we remove the edge between x_i (resp. $\overline{x_i}$) and C_j in the clause gadget ($\tilde{\Gamma}(x_1) = \{\Gamma^{(1,a)}(x_1) = \{C_j\}, \Gamma^{(2,a)}(x_1) = \emptyset\}$ in Fig. 5).

We now show that the DMP problem B with the conditions C has a solution if and only if ψ is satisfiable.

(i) Assume that ψ is satisfiable. For each clause C_j choose a single literal y ($y = x_i$ or $y = \overline{x_i}$) such that y is true. If $y = x_i$ (resp. $y = \overline{x_i}$) then include the edge $(C_j, (x_i, j))$ (resp. $(C_j, (\overline{x_i}, j))$) in the matching M. For each literal gadgets where y appears, set two parallel edges if x_i is true, and two crossing edges if $\overline{x_i}$ is true. Finally, if some literal gadgets aren't already set, we can choose to set them to the value true with two parallel edges. Thus, M is a complete matching. For each literal y, every gadgets with y is either set with two parallel edges or two crossing edges, but not both, depending of the value of y in ψ. And since we set one value for each clause gadget and this value stays the same on literal gadget, all conditions are respected.

(ii) Let M be a solution to the DMP problem. The truth value of x_i is assigned as follows: if literal gadgets of x_i are set to parallel edges (resp. crossing edges), x_i is given the value true (resp. false). The conditions imply that the value of x_i is well defined, x_i and $\overline{x_i}$ cannot both be true since all gadgets for x_i must be set to the same type of complete matching. Since M is a complete matching, every vertex C_j from clause gadget is matched with some vertex y. From the construction, the literal y is true and the clause C_j is satisfied. Thus ψ is satisfiable.

Solution 1: An extended formulation.

A solution of the original problem can involve *chains of events* alternating arrivals and departures: $i_1, j_1, i_2, j_2, \ldots i_c, j_c$ using the same rolling stock unit. We denote by Ω the set of all such possible chains and suggest to replace the x_{ij} variables of the previous MIP model by variables $c_k \in \{0, 1\}$ indicating if the k-th chain is used in the matching. A similar idea was also investigated by [6, 3]. The key idea is therefore that **one can see DMP as a matching problem in a hyper-graph** where all possible chains are represented (as edges). This would allow to preserve the matching structure of the previous MIP and give the rationale for simply replacing x_{ij} by c_k.

Now in our problem, maintenances can be performed at different positions along the chain and this information must be represented. The k-th chain is denoted $chain_k = (i_1, md_1, mt_1, j_1, i_2, md_2, mt_2, j_2, \ldots, i_c, md_c, mt_c, j_c)$ where i_x are arrival nodes, j_x are departures nodes and md_x (resp. mt_x) are 0/1 values denoting if a distance (resp. time) maintenance is performed between the two events i_x and j_x. All features (such as category, group category, length, remaining DBM/TBM at any stage) of a given chain are known. Note that the previous model was simply limited to chains (i, j) of size 2 represented by the x_{ij} variables. The MIP model is modified by replacing the x_{ij} variables with the appropriate c_k variables in all constraints. Typically constraints (1), (2), (4), (6), (9) now becomes:

$$
\begin{array}{lll}
(1') & \sum_{k \in \Omega | i \in chain_k} c_k \leq w_i & \forall i \in \mathcal{A} \cup \mathcal{T} \\
(2') & \sum_{k \in \Omega | j \in chain_k} c_k = y_j & \forall j \in \mathcal{D} \\
(4') & \sum_{k \in \Omega | catGroup_{chain_k} = g} c_k \leq |jdList_j| u_{gj} & \forall j \in \mathcal{JD}, \forall g \in \mathcal{G} \\
(6') & ct_{jc} \geq c_k & \forall c \in \mathcal{C}, \forall k \in \Omega | cat_{chain_k} = c, \\
(9') & l_j \geq \sum_{k \in \Omega | j \in chain_k} length_{chain_k} c_k & \forall j \in \mathcal{D}
\end{array}
$$

Since the distance/time maintenances operations are known on a chain, the daily maintenances limit can be stated on the c_k variables. Typically for a pair (i_r, j_r) in $chain_k$, we simply set $o_{i_r j_r} = md_r + mt_r$. So again, we outline the fact that **the exact days of maintenances are not represented in the pricing problem**:

$$(13^{'}) \qquad \forall d_a \in \{1, \ldots, nbDays\},\ \forall d_b \in \{d_a, \ldots, nbDays\},$$
$$\sum_{(i_r,j_r) \in c_k | k \in \Omega \wedge (i_r,j_r) \in M(d_a,d_b)} o_{i_r j_r} c_k \leq (d_b - d_a + 1) \times maxMaint$$

As Ω is of exponential size, this model is meant to be solved using column generation techniques and Branch and Price. Each category leads to an independent path related problem. At most four choices are possible regarding the maintenance between an arrival i_r and a departure j_r: none, time, distance, both time and distance. Each choice can be modeled with a different arc between i_r and j_r (the four arcs are not necessarily present depending of the time available between i_r and j_r). The objective function is expressed on the use of vertices and edges:

- the cost for using an arrival vertex is related to the dual variable of $(1^{'})$
- the cost for using a departure vertex is related to the dual variable of $(2^{'})$
- the cost for using a maintenance arc between a given pair of arrival and departure (i_r, j_r) is related to the dual variables of $(13^{'})$
- a constant cost related to the category itself is due to the dual variables of $(4^{'}), (6^{'}), (9^{'})$

The problem is to find a shortest path in this graph satisfying two ressource constraints on the level of distance and maintenance which are reset depending of the type of arc used (indicating if some maintenance is performed). We believe it can be modeled as a MIP and do not go here into more details.

Solution 2 : A greedy approach.

The currently implemented solution proceeds greedily by forbidding linked-arrivals as long as their corresponding linked-departure is not known. We simply solve the previous MIP model step by step so that chains of linked events have a length of at most $2i$ at the i-th iteration. After each step, potentially knew linked-departures are known (their features are known) and the linked-arrivals can now be included in the model. This is repeated as long as new arrivals are assigned.

The results presented in Table 2 shows that this approach is able to actually provide solutions that involve dependencies but that many departures remain uncovered. We strongly believe that the column generation model given previously has the potential to accurately deal with this issue. The size of the formulation of the column generation model might be actually smaller and require less memory as not all chains of size 2 might be needed to identify near-optimal solutions.

Table 2. Results obtained on the first part when handling greedily (step by step) the linked arrivals.

instance	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12
Cost (s)	169365	103547	42608	150372	182576	164165	73768	73768	253704	79641	322471	261639
Uncovered Arr	0	0	0	1	7	2	0	0	0	14	44	66
Uncovered Dep	86	81	14	52	80	62	55	55	121	51	195	149
Total time (s)	47	54	487	250	1311	1229	26	22	453	10	101	27

4 Routing

4.1 Modeling the station and hardness of routing

The station can be modelled by an oriented graph $G(V, E)$ where vertices are associated to gates, arcs to the transition of a ressource (with a potential transition time $trTime$). Track groups are typically represented by complete bipartite graphs. Some ressources such as *platforms* allow the trains to change direction. If such a move is possible on a ressource between two gates A and B, it is modelled in the graph with two additionnal vertices $A^{'}$ and $B^{'}$ as shown on Figure 6.

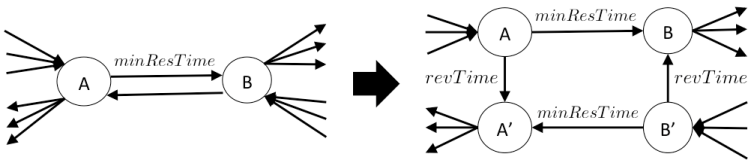

Fig. 6. An exemple structure for performing a reverse move.

Routing in the station is subject to forbidden time-windows expressing the usage of resources by other actors. This is expressed as forbidden time-windows on vertices $i \in V$. We denote by I_i the set of such forbidden time-windows on vertex i : $I_i = \{I_i^1, \ldots, I_i^k, \ldots I_i^{|I_i|}\}$ and H, the overall time-horizon. In particular, track group conflicts can be expressed as forbidden time-windows on intermediate vertices for each crossing of rails in the track group. Additionnal vertices are thus added in V and Figure 7 shows an example in which a train is known to transit on the dashed edges. Note also that track groups now needs to be doubled for the two possible orientations since time-windows differ depending on the orientation.

The problem restricted to the routing of a single train is equivalent to a Resource Constrained Shortest Path Problem (RCSPP) which is known to be NP-Hard. RCSPP is defined with one feasible time-window $[a_i, b_i]$ for each vertex i which can be seen equivalently as two forbidden time-windows $[0, a_i - 1]$ and $[b_i + 1, H]$ over the time-horizon H.

4.2 Modeling the problem restricted to one route bewteen two vertices a and b

Model. We now present a very simple Constraint Programming model to compute a feasible route (satisfying the time-windows) between two given vertices a and b. n de-

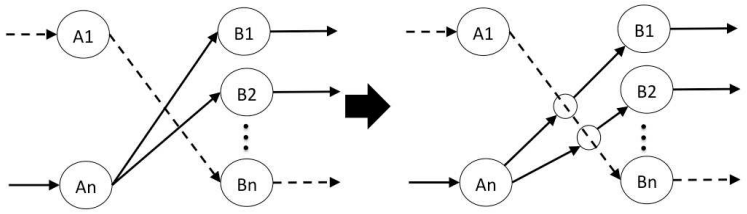

Fig. 7. Additionnal vertices expressing crossing issues in track groups.

notes the number of vertices ($n = |V|$) and the integer d_{ij} refer to the distance between the pair i and j of vertices (we also have $d_{ii} = 0$).

The following variables are used:

- $t_i \in \{0, \ldots, H\} - \cup_{k=1}^{|I_i|} I_i^k$: the time at which the train is entering vertex (gate) i.
- $s_i \in \{k \,|\, (i,k) \in E\} \cup \{i\}$: successor of i in the path ($s_i = i$ if i **is not** in the path).

The model[2] is as follows:

$$
\begin{array}{lll}
(1) & t_{s_i} = t_i + d_{is_i} & \forall i \in V - \{b\} \\
(2) & \textsc{AllDifferent}(s_1, \ldots, s_n) & \forall i \in V \\
(3) & \textsc{NoCycle}(s_1, \ldots, s_n) & \forall i \in V \\
(4) & t_i \in \{0, \ldots, H\} - \cup_{k=1}^{|I_i|} I_i^k & \forall i \in V
\end{array}
$$

The constraint $t_{s_i} = t_i + d_{is_i}$ can be stated with two intermediate variables y_i and z_i representing respectively the time to reach the successor of i ($y_i = t_{s_i}$) and the distance to the successor of i ($z_i = d_{is_i}$). $t_{s_i} = t_i + d_{is_i}$ is stated using two ELEMENT constraints (on a table of variables and on a table of constants i.e the distances):

$$
\begin{array}{l}
\textsc{Element}(y_i, [t_1, \ldots t_n], s_i) \\
\textsc{Element}(z_i, [d_{i1}, \ldots d_{in}], s_i) \\
y_i = t_i + z_i
\end{array}
$$

Strengthening Propagation. Since distances can be null ($d_{ii} = 0$), the shortest path from the source a to all nodes is not actually propagated leading to poor filtering regarding the time-windows. Alternative models using the predecessor variables, min/max constraints to inform about the shortest path from a to all vertices and to any vertex to p can be added and help filtering vertices that can not be part of a feasible path.

Branching. The route is built in a constructive manner selecting vertices from a along the shortest (in number of edges) route leading to b, propagation at each step helps getting rid of invalid successors.

[2] We assume that NOCYCLE allows single loops of type $s_i = i$.

4.3 Overall routing strategy.

Our routing component simply computes routes one by one from the earliest to latest arrival and cancels the arrival if the CP solver is unable to identify a feasible route for the train. A train might require several routes if intermediate resources need to be reached (typically for maintenances or simply for waiting the proper time to be sent to the departure platform). Many heuristics can be added here for the selection of trains such as the ones proposed by [4, 1].

5 Conclusion and future work

A natural problem-decomposition was presented and require limited dedicated algorithms. The router part (see Figure 1) is the only part where a specific strategy has to be designed. We showed that a large part of the problem (larger than expected) related to assignment of arrivals/departures, platforms and in particular the daily maintenances limits can be efficiently handled with MIP. A key idea for integrating efficiently linked-arrivals in such an approach is suggested using a column generation framework. This remains to be evaluated experimentaly but the column generation model can turn out to be smaller in size and should handle linked-arrivals very accurately.

The proposed methodology only holds if the routing can be considered separatly from the assignment. This was the case on the first problem formulation but remains an open question now that track group conflicts are considered as hard constraints. In any case a stronger relaxation of the routing should be included in the assignment part of the model (the two buffers b_1 and b_2 included are a very crude relaxation).

References

1. Mirsad Buljubašić, Haris Gavranović, Said Hanafi, and Michel Vasquez. Team s18. In *Challenge ROADEF/EURO 2014*. IFORS, 2014.
2. Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.
3. Lucas Létocart, Marco Casazza, Antoine Rozenknop, Emiliano Traversi, and Roberto Wolfler-Calvo. Team s20. In *Challenge ROADEF/EURO 2014*. IFORS, 2014.
4. Geiger Martin Josef, Huber Sandra, Sebastian Langton, Marius Leschik, Christian Lindorf, and Ulrich Tshaus. Team s1. In *Challenge ROADEF/EURO 2014*. IFORS, 2014.
5. François Ramond and Nicolas Marcos. Roadef/euro 2014 challenge, trains don't vanish ! - final phase, rolling stock unit management on railway sites. In *Challenge ROADEF/EURO 2014*. SNCF - Innovation & Research Department, 2014.
6. Jørgen Thorlund Haahr and Henry Bull. Team j10. In *Challenge ROADEF/EURO 2014*. IFORS, 2014.