Online Train Shunting

Vianney Bœuf and Frédéric Meunier

Université Paris Est, CERMICS
6-8 avenue Blaise Pascal, Cité Descartes, 77455 Marne-la-Vallée, Cedex 2, France
vianney.boeuf@polytechnique.org, frederic.meunier@enpc.fr

Abstract

At the occasion of ATMOS 2012, Tim Nonner and Alexander Souza defined a new train shunting problem that can roughly be described as follows. We are given a train visiting stations in a given order and cars located at some source stations. Each car has a target station. During the trip of the train, the cars are added to the train at their source stations and removed from it at their target stations. An addition or a removal of a car in the strict interior of the train incurs a cost higher than when the operation is performed at the end of the train. The problem consists in minimizing the total cost, and thus, at each source station of a car, the position the car takes in the train must be carefully decided.

Among other results, Nonner and Souza showed that this problem is polynomially solvable by reducing the problem to the computation of a minimum independent set in a bipartite graph. They worked in the offline setting, i.e. the sources and the targets of all cars are known before the trip of the train starts. We study the online version of the problem, in which cars become known at their source stations. We derive a 2-competitive algorithm and prove than no better ratios are achievable. Other related questions are also addressed.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.1.6 Optimization, G.2.1 Combinatorics

Keywords and phrases Bipartite graph, competitive analysis, online algorithm, train shunting problem, vertex cover

Digital Object Identifier 10.4230/OASIcs.ATMOS.2014.34

1 Introduction

1.1 Context

The Train Shunting Problem, defined by Tim Nonner and Alexander Souza at the occasion of ATMOS 2012 [11], was motivated by concrete problems met by Deutsche Bahn AG. The problem goes as follows. We are given a set of cars and a set of stations. Each car has a *source* station and a *target* station. A locomotive visits the stations according to a predefined order. Once the locomotive passes the source station of a car, this latter is added to the train, and once the locomotive passes its target station, it is removed from the train. Adding or removing a car at the end of the train incurs a cost assumed to be smaller than the cost of adding or removing a car in the interior of the train. Hence, once a car has to be added to the train, a decision must be taken regarding the position it will take in the train. The objective of the Train Shunting Problem consists in minimizing the total cost. Nonner and Souza proved that this problem is polynomially solvable by a reduction to the problem of finding a maximum-weight independent set in a bipartite graph. They also propose some extensions they prove to be polynomially solvable as well, with the help of dynamic programming.

The main assumption they made is that the number of cars, their sources, and their targets are known before solving the problem. However, due to random events and to new demands that can occur during the trip of the locomotive, we expect to face a dynamic part in concrete applications, requiring online decisions. This paper aims to make a step in this direction by defining and studying an online version of the Train Shunting Problem.

1.2 Model

The stations are numbered $1, 2, \ldots$ and visited in this order. We denote the set of cars by $J = [n]$ (throughout the paper, the set $\{1, 2, \ldots, a\}$ is denoted $[a]$). The source station of a car j is denoted s_j and its target station is denoted t_j. If a car j is added to or removed from the exact end of the train, then an *outer* operation of cost c_j is performed. If a car j is added to or removed from the true interior of the train, then an *inner* operation of cost c'_j is performed, with $c'_j > c_j$. An *event* is a source or a target station. Nonner and Souza proved that we can assume that at each station exactly one operation is performed (Lemma 5 in their paper): when several operations must be performed at the same station, we can split the station into as many copies as there are operations to perform and easily order them in a way minimizing the total cost (and which does not depend on future cars). We make the same assumption throughout the paper.

A *train configuration* is a sequence of distinct cars, corresponding to the sequence of cars in the train. A sequence $(C_i)_{i=1,\ldots,m}$ of train configurations is *feasible* if for each station i the train configuration C_i is a sequence of cars involving only cars j such that $s_j \leq i < t_j$ and the common cars of C_i and C_{i+1} occur in the same order in both configurations. Such a sequence encodes a feasible solution of the Train Shunting Problem: under the assumption made above, C_i and C_{i+1} differs only by one car, and the corresponding operation is completely determined. The cost of a sequence $(C_i)_{i=1,\ldots,m}$ is the sum of the costs of these operations.

The problem can be formalized as follows.

Train Shunting Problem

Input. A number m of stations; a set $J = [n]$ of cars; for each car j, two costs $c_j < c'_j$ and a source-target pair of stations (s_j, t_j) with $1 \leq s_j < t_j \leq m$.

Output. A feasible sequence of train configurations $(C_i)_{i=1,\ldots,m}$.

Measure. The cost of $(C_i)_{i=1,\ldots,m}$.

Nonner and Souza proved that a solution of minimal cost can be computed in polynomial time and explained its relation with independence set problems in bipartite graphs. While they worked in the more traditional offline framework, we focus in this work on online algorithms.

An *online algorithm* for this problem is an algorithm which computes C_i without taking into account the cars j such that $s_j > i$. However, at station i, the algorithm can use the information regarding the target station of a car j when $s_j \leq i$, even if $t_j > i$. We require moreover that the online algorithms do not know the number of cars in advance.

Let $\mathcal{A}$ be an online algorithm. Denote by $SOL(I)$ the value of the solution it returns when applied on an input I, and denote by $OPT(I)$ the optimal value of the instance. $\mathcal{A}$ is *c-competitive* for $c \geq 1$ if for some real number b, we have

$$SOL(I) \leq c \cdot OPT(I) + b$$

for all instances I.

1.3 Results

Our main results are the existence of a 2-competitive online algorithm (Theorem 11) and the proof that there is no better competitive ratio (Proposition 12). The core of our 2-competitive algorithm consists in providing a 2-competitive algorithm for the VERTEX COVER PROBLEM in some special-purpose bipartite graph. While it is known that there is no competitive algorithms with fixed ratio for the general VERTEX COVER PROBLEM in bipartite graphs, see [6], our study provides a family of restricted but not artificial instances for which there is such an algorithm. The precise statement of these results, their proofs, and some related results are given in Section 3. They are based on some properties of vertex covers in bipartite graphs presented and proved in Section 2.

Section 4 is devoted to a slight relaxation of the problem. Suppose that we are now allowed to postpone inner operations, by letting cars at the end of the train for some while before moving them to the interior of the train. Since such an inner operation is decided when more information is available, we can expect to have in this case a better competitive ratio. We prove that actually no online algorithms of this type can achieve a ratio smaller than 4/3. We leave as an open question the existence of an online algorithm achieving this ratio.

1.4 Related works

Many papers are already devoted to shunting for freight trains. To the best of our knowledge, except the one introduced by Nonner and Souza, all shunting problems consider the case when the cars are collected by a train, and then lead to a shunting yard where they are rearranged in one or several trains. This yard plays the role of a hub from which the cars starts their final trip to their destinations. Overviews of problems and practices can be found in [3, 9]. Problems and methods aiming at direct applications are proposed in [2, 4, 10, 12]. When there are only two incoming tracks, the system is often based on a hump. Some papers have considered this special case, which provides nice combinatorial problems, see [1, 5, 7].

Other related references can be found in the corresponding section in the paper by Nonner and Souza.

2 Vertex covers in bipartite graphs with positive weights

Let $G = (V, E)$ be a bipartite graph with colour classes S and T. A *vertex cover* of G is a subset $K \subseteq V$ such that any edge in E has at least one endpoint in K. A vertex cover is *minimal* if it is minimal for inclusion.

Dulmage and Mendelsohn [8] proved several properties on minimal-cardinality vertex covers in bipartite graphs, especially that they form a lattice. We extend some of their results to the weighted case. We assume from now on that a weight-function $w : V \to \mathbb{Q}_+$ is given with $w(v) > 0$ for all $v \in V$. As often in combinatorial optimization, given $X \subseteq V$, we use $w(X)$ to denote $\sum_{v \in X} w(v)$.

A vertex cover is *minimum* if it is of minimal weight. Note that since all weights are positive, a minimum vertex cover is minimal.

▶ **Proposition 1.** *Two minimum vertex covers having the same intersection with S are equal.*

Proof. Let K and K' be two such vertex covers. If $T \cap K = \emptyset$, then $K = K'$. Suppose that $T \cap K \neq \emptyset$ and let $v \in T \cap K$. Since K is minimal, there exists $u \in S \setminus K$ such that $uv \in E$. We have $S \setminus K = S \setminus K'$. Since K' is a vertex cover, the edge uv requires v to be in $T \cap K'$. Thus $T \cap K \subseteq T \cap K'$. The reverse inclusion is obtained by exchanging K and K'. ◀

In our 2-competitive algorithm described in Section 3, some minimum vertex covers play a special role.

▶ **Proposition 2.** *There exists a unique minimum vertex cover $\overline{K}$ such that any other minimum vertex cover K satisfies $S \cap K \subseteq S \cap \overline{K}$. Moreover, this vertex cover can be computed in polynomial time.*

Proof. Let K and K' be two minimum vertex covers. Denote by X (resp. X') the subset $S \cap K$ (resp. $S \cap K'$) and by Y (resp. Y') the subset $T \cap K$ (resp. $T \cap K'$). We claim that $(X \cup X') \cup (Y \cap Y')$ is also a minimum vertex cover.

Indeed, first note that $(X \cap X') \cup (Y \cup Y')$ is a vertex cover. Thus $w(X \cap X') + w(Y \cup Y') \geq w(X) + w(Y)$, which implies that $w(Y' \setminus Y) \geq w(X \setminus X')$. Second, note that $(X \cup X') \cup (Y \cap Y')$ is a vertex cover. Its weight is $w(X \cup X') + w(Y \cap Y') = w(X') + w(Y') + w(X \setminus X') - w(Y' \setminus Y)$. Using the inequality that has just been proved, we get $w(X \cup X') + w(Y \cap Y') \leq w(X') + w(Y')$, which means that $(X \cup X') \cup (Y \cap Y')$ is a minimum vertex cover.

Thus the sets $S \cap K$ where K is a minimum vertex cover are stable by union, which leads to the existence of $\overline{K}$. Proposition 1 ensures then the uniqueness of $\overline{K}$.

It remains to prove the statement about the polynomiality of the computation. $\overline{K}$ is the minimum vertex cover that has the largest number of vertices in S. By simply subtracting a small quantity δ to all weights in S, we reduce the problem of finding $\overline{K}$ to a minimum vertex cover problem in a bipartite graph, which is polynomially solvable (see [13] for instance). Any δ smaller than $\frac{1}{|V|M}$ is suitable, where M is such that $Mw(v) \in \mathbb{Z}_+$ for all $v \in V$ (such an M is polynomially computable). ◀

Such a vertex cover $\overline{K}$ is *source-optimal* (it is our terminology). Note that without the condition $w(v) > 0$ for all v, the proposition would not hold. The next proposition shows that while the source-optimal vertex cover is maximal on the source side, it is minimal on the target side.

▶ **Proposition 3.** *Let $\overline{K}$ be the source-optimal vertex cover. Any other minimum vertex cover K satisfies $T \cap \overline{K} \subseteq T \cap K$.*

Proof. If $T \cap \overline{K} = \emptyset$, then the inclusion is obviously satisfied. Suppose that $T \cap \overline{K} \neq \emptyset$ and let $v \in T \cap \overline{K}$. Let K be any minimum vertex cover. Since $\overline{K}$ is minimal, there exists $u \in S \setminus \overline{K}$ such that $uv \in E$. Since $\overline{K}$ is source-optimal, we have $S \setminus \overline{K} \subseteq S \setminus K$. Since K is a vertex cover, the edge uv requires v to be in $T \cap K$. ◀

3 Competitive algorithms

3.1 Preliminaries

A pair of cars (k, ℓ) is *overlapping* if $s_k < s_\ell < t_k < t_\ell$. It is *non-overlapping* otherwise. Nonner and Souza introduced the *constraint graph* $G = (V, E)$, which encodes the overlaps of an instance. It is defined as follows. Its vertex set is $\bigcup_{j \in J} \{s_j, t_j\}$. The edges are the $s_\ell t_k$ with (k, ℓ) being overlapping. The graph G is bipartite with the set of sources $S = \{s_j : j \in J\}$ as one of its colour class and the set of targets $T = \{t_j : j \in J\}$ as the other colour class.

▶ **Proposition 4** (Nonner and Souza [11])**.** *In a feasible solution, the events having inner operations form a vertex cover of G.*

▶ **Proposition 5** (Nonner and Souza [11])**.** *Let K be a minimal vertex cover in G. Then there exists a feasible solution whose inner operations are performed precisely on the events in K. Moreover, K being given, this solution can be computed in $O(n^2)$.*

Nonner and Souza actually formulated and proved these propositions with outer operations instead of inner operations and independent sets instead of vertex covers, but since they are complement of each others, it is an equivalent point of view.

Defining $w(s_j) = w(t_j) = c'_j - c_j$, the total cost of a feasible solution is $w(K) + 2\sum_{j\in J} c_j$, where K is the vertex cover provided by Proposition 4. The total cost is thus minimum when $w(K)$ is minimum. For positive weights on the vertices, a minimum vertex cover is minimal. Since a minimum vertex cover in a bipartite graph can be computed in polynomial time, the two propositions show that the optimal solution of the TRAIN SHUNTING PROBLEM can be computed in polynomial time in the offline setting.

Let us see how we can adapt these considerations in an online context. To ease the discussion, we assume without loss of generality that $s_j < s_k$ if $j < k$: the cars are ordered by their source stations (recall that we have assumed that at each station exactly one operation is performed).

We define $G_j = (V_j, E_j)$ to be the constraint graph limited to the cars $k \in [j]$:

$$V_j = \bigcup_{k\in[j]} \{s_k, t_k\} \quad \text{and} \quad E_j = \{s_\ell t_k : k, \ell \in [j] \text{ and } (k,\ell) \text{ is overlapping}\}.$$

Note that G_j is a bipartite graph and that $G_n = G$, where G is still the constraint graph of the full input. Moreover, G_j is an induced subgraph of G_{j+1}: we have $V_{j+1} = V_j \cup \{s_{j+1}, t_{j+1}\}$ and $E_{j+1} = E_j \cup \delta(s_{j+1})$, where $\delta(s_{j+1})$ is the set of edges incident to s_{j+1} in G. Using Proposition 4, we can see that a feasible solution induces a chain $K_1 \subseteq K_2 \subseteq \cdots \subseteq K_n$ where K_j is a vertex cover of G_j. Indeed, we can for instance set K_j to be the events subject to inner operations up to station t_j.

A counterpart of Proposition 5 is also true, see Proposition 6 below: a chain $K_1 \subseteq \cdots \subseteq K_n$, where K_j is a vertex cover of G_j satisfying some condition to be detailed below, provides the inner operations of some feasible solution. However, this is not a direct consequence of Proposition 5 – the inner operations programmed up to station s_j must be compatible with the inner operations programmed up to station s_{j-1} – and deserves a proof. By $N(s_j)$, we denote the set of *neighbours* of s_j, i.e. the set of vertices v of G such that $s_j v$ is an edge of G. Note that it is also the set of neighbours of s_j in G_j. By $N[s_j]$, we denote the *closed neighbourhood* of s_j, i.e. the set $N(s_j) \cup \{s_j\}$.

▶ **Proposition 6.** *Let $K_1 \subseteq \cdots \subseteq K_n$ be such that each K_j is a vertex cover of G_j satisfying $N[s_j] \setminus K_j \neq \emptyset$. Then there exists a feasible solution such that*
- *the sources s_j subject to inner operations are exactly those s_j such that $s_j \in K_j$, and*
- *the targets t_j subject to inner operations are such that $t_j \in K_n$.*

Moreover, we can decide in polynomial time the position each car j must take in the train using $K_1, \ldots, K_j$.

In Proposition 6, we have a stronger statement for the sources than for the targets. Anyway, the proposition ensures that we can build a feasible solution online, and allows to bound from above its cost: $w(K_n) + 2\sum_{j\in J} c_j$ is an upper bound on the cost of this feasible solution.

To prove Proposition 6, we mimic the proof of Theorem 2 in [11] but several difficulties related to the online aspect arise. We assume given a chain of vertex covers $K_1 \subseteq \cdots \subseteq K_n$ such that each K_j is a vertex cover of G_j satisfying $N[s_j] \setminus K_j \neq \emptyset$.

For each $j \in J$, we define a directed graph $H_j = ([j], A_j)$, whose vertices are the integers from 1 to j (the cars up to j). The arcs are defined as follows. For a car $k \leq j$ and an event e such that $s_k < e < t_k$, with $e \in \{s_\ell, t_\ell\}$ and $\ell \leq j$, the arc (k, ℓ) is in A_j if $e \notin K_{\max(k,\ell)}$.

The definition of the graph H_j resembles the definition of the graph H of the original proof, but is not completely identical. Note that the sequence of graphs H_j is increasing, $A_j \subseteq A_{j+1}$ for all $1 \leq j \leq n-1$, and that all arcs in $A_{j+1} \setminus A_j$ are incident to $j+1$.

▶ **Lemma 7.** *The graph H_j is acyclic.*

Proof. Suppose for a contradiction that there is a directed cycle $C = (k_1, \ldots, k_r)$ in H_j. We choose C with the minimum number of arcs. Note that we have anyway $r \geq 2$. Without loss of generality, we assume that k_1 is the smallest integer on C.

The arc (k_r, k_1) exists in H_j, thus $s_{k_r} < t_{k_1} < t_{k_r}$ and $t_{k_1} \notin K_{k_r}$. As $s_{k_1} < s_{k_r}$, the pair (k_1, k_r) is overlapping and G_{k_r} contains the edge $s_{k_r} t_{k_1}$. Necessarily, $s_{k_r} \in K_{k_r}$. Consider now the arc (k_{r-1}, k_r). We prove that $s_{k_{r-1}} \in K_{k_{r-1}}$, that $t_{k_r} \notin K_{k_r}$, and that (k_r, k_{r-1}) is overlapping.

Suppose first that $s_{k_{r-1}} < s_{k_r}$. We necessarily have $s_{k_{r-1}} < t_{k_r} < t_{k_{r-1}}$ and $t_{k_r} \notin K_{k_r}$ because $s_{k_r} \in K_{k_r}$. (Note that it implies that $r \geq 3$.) Thus $s_{k_{r-1}} < t_{k_1} < t_{k_{r-1}}$, and there should be an arc (k_{r-1}, k_1) in H_j since $t_{k_1} \notin K_{k_{r-1}}$ (otherwise we would have $t_{k_1} \in K_{k_r}$, in this case k_r being larger than k_{r-1}). Such an arc would contradict the minimality of C. Hence $s_{k_{r-1}} > s_{k_r}$ and $K_{k_r} \subseteq K_{k_{r-1}}$. We have $s_{k_{r-1}} < t_{k_r} < t_{k_{r-1}}$ and $t_{k_r} \notin K_{k_{r-1}}$ and the pair (k_r, k_{r-1}) is overlapping. There is therefore an edge $s_{k_{r-1}} t_{k_r}$ in $G_{k_{r-1}}$, which implies that $s_{k_{r-1}} \in K_{k_{r-1}}$ as required. We also have $t_{k_r} \notin K_{k_r}$ since in this case $K_{k_r} \subseteq K_{k_{r-1}}$.

Repeating the argument along the same lines, we get then that $s_{k_{r-i}} \in K_{k_{r-i}}$, that $t_{k_{r-i+1}} \notin K_{k_{r-i+1}}$, and that (k_{r-i+1}, k_{r-i}) is overlapping for all $i \in [r-1]$. In particular, for $i = r-1$, we get that $s_{k_2} < s_{k_1}$, which is a contradiction. ◀

Since H_j is acyclic, we can define a partial order on $[j]$: we set $k \preceq_j \ell$ if there is a directed path from k to ℓ in H_j. Since the sequence (A_j) is increasing, $k \preceq_j \ell$ implies $k \preceq_{j'} \ell$ for all $j' \geq j$. The converse is actually true.

▶ **Lemma 8.** *Let k and ℓ be two integers in $[j]$. If k and ℓ are incomparable for $\preceq_j$, they are incomparable for all $\preceq_{j'}$ with $j' \geq j$.*

Proof. Assume for sake of a contradiction that k and ℓ are incomparable for $\preceq_j$ but not for some $\preceq_{j'}$ with $j' > j$. We choose j' as small as possible with this property. Without loss of generality, we assume that $k \preceq_{j'} \ell$. It means that there is an elementary path from k to ℓ in $H_{j'}$ that goes through j' (because of the minimality of j'). Moreover, it means also that the two neighbours of j' on this path are incomparable in $H_{j'-1}$: if there were a path between these two neighbours, it would either contradict the acyclicity of $H_{j'}$ (Lemma 7), or the minimality of j' (the integers k and ℓ would already have been comparable for $H_{j'-1}$), depending on the direction of the path. The two neighbours of j' are thus incomparable for $\preceq_{j'-1}$ and comparable for $\preceq_{j'}$, and they would also contradict the statement of the lemma we want to prove. We can thus assume without loss of generality that k and ℓ are the two neighbours of j' and that the arcs (k, j') and (j', ℓ) exist in $A_{j'}$.

By definition of $A_{j'}$, we have $s_k < e < t_k$, with $e \in \{s_{j'}, t_{j'}\}$ and $e \notin K_{j'}$, and $s_{j'} < f < t_{j'}$, with $f \in \{s_\ell, t_\ell\}$ and $f \notin K_{j'}$. Since $s_\ell < s_{j'}$, we necessarily have $f = t_\ell$, and (ℓ, j') is overlapping. It implies that $s_{j'} \in K_{j'}$, and thus $e = t_{j'}$. Therefore, we have $s_k < t_\ell < t_k$ with $t_\ell \notin K_{j'}$, which implies that the arc (k, ℓ) exists in $A_{j'}$, and thus already in A_j. It is in contradiction with the fact that k and ℓ are incomparable. ◀

We are now in position to prove Proposition 6.

Proof of Proposition 6. We build a sequence of total orders $(\preceq_j^{tot})_{j\in J}$, the order $\preceq_j^{tot}$ being defined on $[j]$ and being compatible with the partial order $\preceq_j$ defined above. We build this sequence so that if $k \preceq_j^{tot} \ell$ for $k,\ell \in [j]$, then $k \preceq_{j'}^{tot} \ell$ for all $j' \geq j$.

When $j=1$, the definition is trivial. Suppose that $\preceq_j^{tot}$ is defined for some j. We explain how to build $\preceq_{j+1}^{tot}$. We consider the tournament induced by $\preceq_j^{tot}$ on $[j]$. (Recall that a tournament in graph theory is obtained by giving an orientation to each edge of a complete graph). The tournament is acyclic. We add a vertex $j+1$ to this tournament, as well as all arcs $(k, j+1)$ with $k \preceq_{j+1} j+1$ and all arcs $(j+1, k)$ with $j+1 \preceq_{j+1} k$. Let D'_{j+1} be this new graph. We claim that D'_{j+1} is acyclic. Indeed, suppose for a contradiction that it contains a directed cycle. It necessary goes through $j+1$. The two neighbours of $j+1$ on this cycle are comparable according to $\preceq_{j+1}$. According to Lemma 8, they are already comparable for $\preceq_j$. As it has been noticed right before the statement of Lemma 8, these two neighbours should then be ordered in a same way by $\preceq_j$ and by $\preceq_{j+1}$, which is in contradiction with the acyclicity of the tournament. We can thus complete D'_{j+1} into an acyclic tournament, which provides the total order $\preceq_{j+1}^{tot}$. To conclude this part of the proof, note that $\preceq_{j+1}^{tot}$ is compatible with $\preceq_{j+1}$: it is compatible with $\preceq_j$ and thus with $\preceq_{j+1}$ (Lemma 8) for the elements in $[j]$; since all arcs $(k, j+1)$ with $k \preceq_{j+1} j+1$ and all arcs $(j+1, k)$ with $j+1 \preceq_{j+1} k$ are present in D'_{j+1}, the order $\preceq_{j+1}^{tot}$ is compatible with $\preceq_{j+1}$ on all elements of $[j+1]$.

Note that this construction is polynomially computable.

We say that a car j is *active* at station i if $s_j \leq i < t_j$. Now, we define the following sequence $(\widetilde{C}_i)$ of train configurations: $\widetilde{C}_i$ is the sequence of active cars at station i ordered from right to left according to $\preceq_{j(i)}^{tot}$, where $j(i) = \max\{j : s_j \leq i\}$. We assume that the end of the train is at the left-most position, the right-most position being the one of the locomotive. Note that in particular we have the maximal element for the total order at the end of the train and that the first car after the locomotive is the minimal element for the total order.

The sequence $(\widetilde{C}_i)$ is feasible: the common cars in $\widetilde{C}_i$ and $\widetilde{C}_{i+1}$ occur in the same order because $j(i+1) \in \{j(i), j(i)+1\}$ and in any case $\preceq_{j(i)}^{tot}$ and $\preceq_{j(i+1)}^{tot}$ are compatible. Note that the operation to perform at station i, and in particular the exact position the car must take in the train in case i is a source station, can be done in polynomial time using $\preceq_{j(i)}^{tot}$.

We prove now that $s_j \in K_j$ if and only if s_j is subject to an inner operation in the sequence $(\widetilde{C}_i)$. Suppose first that $s_j \in K_j$. Since, $N[s_j] \setminus K_j \neq \emptyset$, there is a car $k < j$ such that (k, j) is overlapping and $t_k \notin K_j$. We have thus an arc (j, k) in H_j. Therefore, j precedes k in $\preceq_j^{tot}$, which means that j cannot be at the end of the train when the train leaves station s_j. Suppose now that $s_j \notin K_j$ and let k be any active car at station s_j distinct from j. We have $s_k < s_j < t_k$ and thus the arc (k, j) exists in H_j. Since it holds for any such k, the car j is the maximal element for $\preceq_j^{tot}$ on the subset of active cars and is at the end of the train when the train leaves s_j: the car j has incurred an outer operation.

Finally, we prove that if t_j is subject to an inner operation, then $t_j \in K_n$. Suppose that $t_j \notin K_n$. For any active car k at station t_j, i.e. any car such that $s_k < t_j < t_k$, we have $t_j \notin K_{\max(k,j)}$. There is thus an arc (k, j) in $H_{\max(k,j)}$. Thus at t_j, the car j is located at the end of the train and is subject to an outer operation. ◀

We end the section with a lemma that will be useful in the next section. It explains how the source-optimal vertex covers of the sequence of graphs (G_j) are related. For each j, we denote by $\overline{K}_j$ the source-optimal vertex cover of G_j.

▶ **Lemma 9.** *For each $j \geq 2$, we have $T \cap \overline{K}_{j-1} \subseteq T \cap \overline{K}_j$ and exactly one of the following relations is satisfied:*

- $\overline{K}_j = \overline{K}_{j-1} \cup \{s_j\}$.
- $S \cap \overline{K}_{j-1} \supsetneq S \cap \overline{K}_j$.

Proof. Suppose first that $s_j \in \overline{K}_j$. The set $\overline{K}_j \setminus \{s_j\}$ is a vertex cover of G_{j-1}, and thus $w(\overline{K}_j) - w(s_j) \geq w(\overline{K}_{j-1})$. The set $\overline{K}_{j-1} \cup \{s_j\}$ is a vertex cover of G_j, and thus $w(\overline{K}_j) - w(s_j) \leq w(\overline{K}_{j-1})$. Combining both inequalities shows that $\overline{K}_j \setminus \{s_j\}$ is a minimum vertex cover of G_{j-1} and that $\overline{K}_{j-1} \cup \{s_j\}$ is a minimum vertex cover of G_j. The vertex cover $\overline{K}_{j-1}$ being source-optimal, we have $S \cap \overline{K}_{j-1} \supseteq S \cap (\overline{K}_j \setminus \{s_j\})$, which implies $S \cap (\overline{K}_{j-1} \cup \{s_j\}) \supseteq S \cap \overline{K}_j$. The vertex cover $\overline{K}_j$ being source-optimal, we have $\overline{K}_j = \overline{K}_{j-1} \cup \{s_j\}$ by uniqueness of the source-optimal vertex cover, and we have $T \cap \overline{K}_{j-1} \subseteq T \cap \overline{K}_j$.

Suppose then that $s_j \notin \overline{K}_j$. Let $X_k = S \cap \overline{K}_k$ and $Y_k = T \cap \overline{K}_k$. The set $(X_{j-1} \cap X_j) \cup (Y_{j-1} \cup Y_j)$ is a vertex cover of G_j. Indeed, an edge $s_k t_\ell$ in E_j with $t_\ell \notin Y_{j-1} \cup Y_j$ is such that $s_k \in X_j$ because $\overline{K}_j$ is a vertex cover of G_j, and also such that $k \neq j$ because we supposed $s_j \notin \overline{K}_j$; it implies that $s_k t_\ell$ is in E_{j-1} as well and that $s_k \in X_{j-1}$. Thus, $w(X_{j-1} \cap X_j) + w(Y_{j-1} \cup Y_j) \geq w(\overline{K}_j)$, which implies that $w(X_j \setminus X_{j-1}) \leq w(Y_{j-1} \setminus Y_j)$. Since $w(X_{j-1} \cup X_j) + w(Y_{j-1} \cap Y_j) = w(\overline{K}_{j-1}) - w(Y_{j-1} \setminus Y_j) + w(X_j \setminus X_{j-1})$, the latter inequality shows that $w(X_{j-1} \cup X_j) + w(Y_{j-1} \cap Y_j) \leq w(\overline{K}_{j-1})$. The set $(X_{j-1} \cup X_j) \cup (Y_{j-1} \cap Y_j)$ is a vertex cover of G_{j-1}, and thus is a minimum vertex cover of G_{j-1}. The set $\overline{K}_{j-1}$ being source-optimal, we get $X_{j-1} \cup X_j \subseteq X_{j-1}$, which implies $S \cap \overline{K}_{j-1} \supsetneq S \cap \overline{K}_j$. Moreover, Proposition 3 implies that $Y_{j-1} \subseteq Y_{j-1} \cap Y_j$, i.e. $T \cap \overline{K}_{j-1} \subseteq T \cap \overline{K}_j$. ◀

3.2 A 2-competitive algorithm

We present in this section an online algorithm with a 2-competitive ratio. Roughly speaking, the algorithm goes as follows. At each source station, it checks with the help of a computation of a source-optimal vertex cover whether there is an optimal schedule for the whole known instance in which this source station is subject to an inner operation. If it is the case, the car is added at an inner position determined with the help of Proposition 6. Otherwise, the car is added to the end of the train. In a sense, the algorithm tries to make the inner operations as soon as possible without worsen the quality of the solution.

The online algorithm goes precisely as follows.

Start with an empty graph G_0 and an empty set $\widetilde{K}_0$; when the train arrives at station s_j, build G_j as described in Section 3.1, compute a source-optimal vertex cover $\overline{K}_j$ of G_j for the weight function w, define $\widetilde{K}_j = \widetilde{K}_{j-1} \cup \overline{K}_j$.

In other words, the set $\widetilde{K}_j$ is equal to $\bigcup_{k=1}^{j} \overline{K}_k$. We are going to prove that the sequence of the $\widetilde{K}_j$ satisfies the condition of Proposition 6 and thus the algorithm computes a feasible solution performing an inner operation at station s_j if and only if $s_j \in \overline{K}_j$.

▶ **Proposition 10.** *Each $\widetilde{K}_j$ is a vertex cover of G_j satisfying $N[s_j] \setminus \widetilde{K}_j \neq \emptyset$ and we have the following chain: $\widetilde{K}_1 \subseteq \cdots \subseteq \widetilde{K}_n$.*

Proof. The fact that $\widetilde{K}_j$ is a vertex cover and the inclusion $\widetilde{K}_{j-1} \subseteq \widetilde{K}_j$ are obvious.

Suppose that $N(s_j) \subseteq \widetilde{K}_j$. Then necessarily, the elements in $N(s_j)$ belong to the union of some $T \cap \overline{K}_k$ with $k \leq j$. Lemma 9 implies that actually $N(s_j) \subseteq \overline{K}_j$. Since $\overline{K}_j$ is minimal, we have $s_j \notin \overline{K}_j$, and thus $s_j \notin \widetilde{K}_j$. ◀

Proposition 6 and Proposition 10 show that the online algorithm described above computes a feasible solution to the Train Shunting Problem. It is polynomial according to

Proposition 2. We have thus the following theorem, the calculation of the competitive ratio being done in the proof.

▶ **Theorem 11.** *There is a polynomial* 2*-competitive online algorithm for the* Train Shunting Problem.

Proof. The preceding discussion shows that the online algorithm described above is polynomial and computes a feasible solution. It remains to evaluate its competitive ratio. The cost of the solution computed by the online algorithm is bounded from above by $w(\widetilde{K}_n) + 2\sum_{j\in J} c_j$ because of Proposition 6. The set $\overline{K}_n$ is a minimum vertex cover of $G = G_n$. According to Nonner-Souza's result (see discussion in Section 3.1), the optimum of the Train Shunting Problem is $w(\overline{K}_n) + 2\sum_{j\in J} c_j$. The proof strategy consists in bounding $w(\widetilde{K}_n)$ from above using $w(\overline{K}_n)$. We set $\overline{K}_0 = \emptyset$.

$\widetilde{K}_n$ can also be written $\overline{K}_n \cup \left(\bigcup_{j\in J} \overline{K}_{j-1} \setminus \overline{K}_j\right)$, and hence

$$w(S \cap \widetilde{K}_n) \le w(S \cap \overline{K}_n) + \sum_{j\in J} w(S \cap (\overline{K}_{j-1} \setminus \overline{K}_j)).$$

Since $\overline{K}_j$ contains a vertex cover of G_{j-1}, we have $w(\overline{K}_{j-1}) \le w(\overline{K}_j)$. According to Lemma 9, we know that $T \cap \overline{K}_{j-1} \subseteq T \cap \overline{K}_j$ and that if $S \cap (\overline{K}_{j-1} \setminus \overline{K}_j) \neq \emptyset$, then $S \cap \overline{K}_{j-1} \supseteq S \cap \overline{K}_j$. Therefore $w(S \cap (\overline{K}_{j-1} \setminus \overline{K}_j)) \le w(T \cap \overline{K}_j) - w(T \cap \overline{K}_{j-1})$. Hence

$$w(S \cap \widetilde{K}_n) \le w(S \cap \overline{K}_n) + \sum_{j\in J} (w(T \cap \overline{K}_j) - w(T \cap \overline{K}_{j-1})).$$

Therefore $w(S \cap \widetilde{K}_n) \le w(S \cap \overline{K}_n) + w(T \cap \overline{K}_n)$ and $w(S \cap \widetilde{K}_n) \le w(\overline{K}_n)$.

On the other hand, we have $w(T \cap \widetilde{K}_n) = w(T \cap \overline{K}_n)$ again because of Lemma 9. Thus $w(T \cap \widetilde{K}_n) \le w(\overline{K}_n)$.

The two inequalities lead to $w(\widetilde{K}_n) \le 2w(\overline{K}_n)$. Our algorithm provides thus a solution of cost bounded from above by $w(\widetilde{K}_n) + 2\sum_{j\in J} c_j \le 2(w(\overline{K}_n) + 2\sum_{j\in J} c_j)$. ◀

▶ Remark. No algorithms computing $\overline{K}_j$ in linear time are known up to now. However, if $c_j = 0$ and $c'_j = 1$ for all j, it is possible to compute $\overline{K}_j$ in $O(|E_j|)$ by maintaining a maximum-cardinality matching of G_j along the algorithm. Nevertheless, we do not know whether a similar idea can be extended to the case with general costs.

3.3 Lower bound on the competitive ratio

▶ **Proposition 12.** *No online algorithms computing a solution to the* Train Shunting Problem *can have a competitive ratio smaller than* 2.

Proof. Let $\mathcal{A}$ be an online algorithm computing a solution to the Train Shunting Problem. The proof consists in describing for any integer q, an instance with at most $3q$ cars for which $SOL \ge (2 - 1/q) \cdot OPT$, where SOL is the value of the solution computed by $\mathcal{A}$, and OPT is the optimum. The instance is built dynamically as follows, taking into account the decisions of $\mathcal{A}$.

All costs c_j are set to 0 and all costs c'_j are set to 1. For $j = 1, \ldots, q$, define $s_j = j$ and $t_j = 4q - j + 1$. Set $s_{q+1} = q + 1$ and $t_{q+1} = 6q$. Then, from $j = q + 1$, we repeat the following loop:

If the operation performed by $\mathcal{A}$ at station j is an outer operation or if $j = 3q$, then stop. Otherwise, set $j \leftarrow j + 1$; define $s_j = j$ and $t_j = 7q - j + 1$.

Denote by r the number of times the loop has been repeated. We have

$$SOL \geq \begin{cases} r+q-1 & \text{if } r \leq 2q-1 \\ 2q & \text{if } r = 2q. \end{cases}$$

Indeed, if $r \leq 2q-1$, the r repetitions of the loop correspond to $r-1$ inner operations. The car $q+r$ is added to the end of the train and implies q inner operations to remove from the train the cars indexed from 1 to q. If $r=2q$, no cars between q and $3q-1$ are added to the end of the train and their addition to the train provides $2q$ inner operations.

We have $OPT = \min(q,r)$. This can be seen by considering the constraint graph of the instance and by computing a minimum vertex cover of it, see Section 3.1.

If $r=2q$, we have $SOL/OPT \geq 2$. If $q \leq r \leq 2q-1$, we have $SOL/OPT \geq 2-1/q$. If $r \leq q-1$, we have $SOL/OPT \geq 2$. ◀

There are two natural algorithms we can also think of. Unfortunately, they do not even enjoy a fixed competitive ratio.

The first consists in always introducing the cars at the end of the train. In this case, the competitive ratio can be arbitrarily large, as shown by the following example. Consider the instance with $s_j = j$ and $t_j = 2n-j$ for $j=1,\ldots,n-1$, and $s_n = n$ and $t_n = 2n$. Take as costs $c_j = 0$ and $c'_j = 1$ for all j. It is easy to check that the total cost is then $n-1$ when that algorithm is applied, while the optimal cost is 1.

The second algorithm consists in building a sequence of vertex covers, similarly as for the algorithm of Section 3.2.

Start with an empty graph G_0 and an empty set $\widetilde{K}_0$; when the train arrives at station s_j, build G_j as described in Section 3.1, compute a vertex cover $\widetilde{K}_j$ of G_j of minimal cost such that $\widetilde{K}_{j-1} \subseteq \widetilde{K}_j$.

This algorithm can be considered as natural since computing $\widetilde{K}_j$ amounts to choose $\widetilde{K}_j$ among $\widetilde{K}_{j-1} \cup \{s_j\}$ and $\widetilde{K}_{j-1} \cup N(s_j)$, the solution being the one of minimal cost. Proposition 6 shows that we build in this way a feasible solution. It means that we always choose an operation that is locally the best solution.

The following example shows that this algorithm can also have an arbitrarily large competitive ratio. Consider the instance with $s_j = j$ for $j=1,\ldots,n$, $t_1 = n+2$, $t_2 = n+1$, and $t_j = 2n-j+3$ otherwise. Set the costs to be $c_j = 0$ and $c'_j = 1$ for all j. It is easy to check that the total cost is then $n-3$ when this algorithm is applied, while the optimal cost is 2.

4 Postponing inner operations

Suppose that we modify the Train Shunting Problem in the following sense: at any station, a car at the end of the train can be moved to the interior and such an operation can be repeated several times at a same station. The cost of such an operation is assumed to remain the same, namely c'_j for car j.

It does not change the optimal solution of an instance. Indeed, suppose that we have an optimal solution such that a car j is added to the train at the station s_j, and moved to the interior from the end of the train at some station $i \geq s_j$. Then the solution consisting in inserting the car j directly at some inner position so that the train configuration will be the same at station i will not be of larger cost.

Hence, from an offline point of view, this new possibility does not reduce the best cost that can be achieved. However, we do not know whether the conclusion is identical in the online setting. We were however able to prove the following result, which leaves some hope for a better ratio.

▶ **Proposition 13.** *No online algorithms computing a solution to the* Train Shunting Problem *in this modified setting can achieve a competitive ratio smaller than* 4/3.

Proof. Let $\mathcal{A}$ be an online algorithm computing a solution to the Train Shunting Problem with this additional possibility. Consider the instance where the first five cars are such that

$$(s_1, t_1) = (1, 11),\ (s_2, t_2) = (2, 10),\ (s_3, t_3) = (3, 6),\ (s_4, t_4) = (4, 16),\ (s_5, t_5) = (5, 15).$$

Then, if $\mathcal{A}$ has chosen an inner removal for car 3, then stop. Otherwise, three cars 6, 7, and 8 are added to the instance with

$$(s_6, t_6) = (7, 14),\ (s_7, t_7) = (8, 13),\ (s_8, t_8) = (9, 12).$$

If the car 3 is in the interior of the train when it leaves station 5, then the total cost achieved by the algorithm is 3 at best: the car 3 will be subject to an inner operation, and the instance reduced to cars 1, 2, 4, and 5 has an optimal cost of 2.

If the car 3 is at the end the train when it leaves station 5, then the cars 4 and 5 have been added or moved to the interior of the train, and the instance reduced to the cars 1, 2, 3, 6, 7, and 8 has an optimal cost of 2, which gives in total a cost of 4. So, the total cost achieved by the algorithm is 4 at best, while the optimum is 3. ◀

References

1 Katharina Beygang, Florian Dahms, and Sven O. Krumke. Train marshalling problem: Algorithms and bounds. Technical report, 2010.

2 Markus Bohlin, Florian Dahms, Holger Flier, and Sara Gestrelius. Optimal freight train classification using column generation. In *Proceedings of the 12th workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'12)*, volume 25, pages 10–22, 2012.

3 Nils Boysen, Malte Fliedner, Florian Jaehn, and Erwin Pesch. Shunting yard operations: Theoretical aspects and applications. *European Journal of Operational Research*, 220:1–14, 2012.

4 Alberto Ceselli, Michael Gatto, Marco E. Lübbecke, Marc Nunkesser, and Heiko Schilling. Optimizing the cargo express service of Swiss federal railways. *Transportation Science*, 42:450–465, 2008.

5 Elias Dahlhaus, Peter Horák, Mirka Miller, and Joseph F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103:41–54, 2000.

6 Marc Demange and Vangelis T. Paschos. On-line vertex-covering. *Theoretical Computer Science*, 332:83–108, 2005.

7 Gabriele Di Stefano and Magnus Love Koci. A graph theoretical approach to the shunting problem. *Electronic Notes in Theoretical Computer Science*, 92:16–33, 2004.

8 Andrew L. Dulmage and Nathan S. Mendelsohn. Coverings of bipartite graphs. *Canadian Journal of Mathematics*, 10:517–534, 1958.

9 Michael Gatto, Jens Maue, Matús Mihalák, and Peter Widmayer. *Robust and Online Large-Scale Optimization*, chapter Shunting for dummies: An introductory algorithmic survey, pages 310–337. Springer, 2009.

10 Riko Jacob, Peter Marton, Jens Maue, and Marc Nunkesser. Multistage methods for freight train classification. *Networks*, 57:87–105, 2011.

11 Tim Nonner and Alexander Souza. Optimal algorithms for train shunting and relaxed list update problems. In *Proceedings of the 12th workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'12)*, volume 25, pages 97–107, 2012.

12 Marc Nunkesser, Michael Gatto, and Riko Jacob. Optimization of a railway hub-and-spoke system: routing and shunting. In *Proceedings of WEA 2005*, 2005.

13 Alexandrer Schrijver. *Combinatorial Optimization.* Springer, 2003.