Platform Assignment

Sabine Cornelsen[1]* and Gabriele Di Stefano[2]

[1] Universität Konstanz, Fachbereich Informatik & Informationswissenschaft,
cornelse @ inf.uni-konstanz.de
[2] Università dell'Aquila, Dipartimento di Ingegneria Elettrica,
gabriele @ ing.univaq.it

Abstract. We consider a station in which several trains might stop at the same platform at the same time. The trains might enter and leave the station to both sides, but the arrival and departure times and directions are fixed according to a given time table. The problem is to assign platforms to the trains such that they can enter and leave the station in time without being blocked by any other train. We consider some variation of the problem on linear time tables as well as on cyclic time tables and show how to solve them as a graph coloring problem on special graph classes. One of these classes are the so called circular arc containment graphs for which we give an $\mathcal{O}(n \log n)$ coloring algorithm.

1 Introduction

We consider the following problem. We are given a set of trains that has to enter a station with Marshalling topology. Each train might enter the station either from the left hand side or from the right hand side and might leave the station to the left hand side or to the right hand side. The direction from which it enters the station and in which it leaves the station, however, is fixed. Also the arrival time and departure time is fixed. Hence, each train is labeled with the time interval $[t_{\text{arr}}, t_{\text{dep}}]$ in which it stays in the station, and an entering and leaving direction d_{arr} and d_{dep}, respectively. For example, a train labeled [24,25]RL is a train that enters the station at 24 from the right hand side and leaves the station at 25 to the left hand side. In the meanwhile, it stops on one of several parallel platforms (tracks) in the station. There might be several trains waiting at the same platform. We further assume that the platforms have infinite length. The problem is to assign each train to a platform such that it can leave the station in time without being blocked by other trains. E.g., the assignment in Fig. 1a is not feasible, since train [-2,2]RL would be blocked by train [-1,4]LL. The assignments in Fig. 1b and c are both feasible. However, the interesting assignment is the one in Fig. 1c, that uses the minimum number of platforms.

We consider as well linear time tables as cyclic time tables. In cyclic time tables, the situation repeats itself each time period T, i.e., a label $[t_{\text{arr}}, t_{\text{dep}}]d_{\text{arr}}d_{\text{dep}}$

* Work partially done while the author was visiting the University of L'Aquila, supported by the Human Potential Program of the EU under contract no HPRN-CT-1999-00104 (AMORE Project).

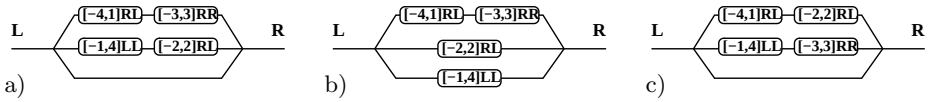


Fig. 1. a) A non-feasible platform assignment. b) A feasible platform assignment that is achieved with the first-fit algorithm. c) A feasible platform assignment that uses the minimum number of platforms.

represents a series of trains $(z_i)_{i\in\mathbb{Z}}$. Train z_i arrives at time $t_{\text{arr}} + iT$ from direction d_{arr} and leaves at time $t_{\text{dep}} + iT$ in direction d_{dep}. We further assume that no train stays longer than the time period T in the station, i.e., that $t_{\text{dep}} - t_{\text{arr}} < T$. The goal is a platform assignment that repeats itself every time period T, i.e., to assign each train of a series belonging to the same label to the same platform. E.g., we are not interested in an assignment as indicated in Fig. 2a, but rather in an assignment as indicated in Fig. 2b.

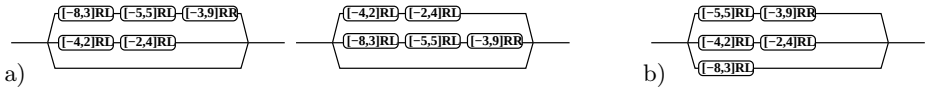


Fig. 2. Cyclic platform assignment with time period $T = 16$. a) A platform assignment that repeats itself every second time period. b) A cyclic platform assignment that repeats itself every time period.

As an additional constraint, we will sometimes consider the so called *midnight constraint*, i.e., the condition that all trains enter the station before the first train leaves the station. The name of this constraint is motivated by shunting problems in which all trains enter a shunting depot in the evening and leave it in the morning. But the midnight constraint even seems to be a reasonable constraint for modeling the situation in some stations where first all trains enter the station such that passengers can change trains and afterwards all trains leave the station. A train is a *turning back train*, if it leaves the station in the direction from where it entered it, i.e., if $d_{\text{arr}} = d_{\text{dep}}$.

The platform assignment problem is closely related to the shunting problem some variation of which have been studied in the following work [1, 3, 5, 6, 8, 13]. We will consider the platform assignment problem as a graph coloring problem. Given a platform assignment problem, we define the following *constraint graph*. The vertices of the graph are the trains (or the series of trains if cyclic time tables are considered). Two trains are adjacent if they cannot be put on the same platform. The platform assignment problem then corresponds to coloring the constraint graph. For some variation of the shunting problem, this approach has been studied, e.g., in [4, 12].

In this paper, we show the following results for the platform assignment problem. In the cases of linear time tables with midnight constraint or without

turning back trains, we show that the constraint graph is a permutation graph and hence it can be colored in $\mathcal{O}(n \log n)$ time. However, the algorithm does not consider the trains in the order in which they enter the station. We further give a 2-competitive online-algorithm to solve this problem. This algorithm also runs in $\mathcal{O}(n \log n)$ time.

In the case of cyclic time tables with midnight constraint, the constraint graph is not necessarily perfect. For this problem, we give a 3-approximation algorithm that runs in quadratic time. Cyclic time tables without turning back trains yield a constraint graph that is a comparability graph and, hence, can be colored in $\mathcal{O}(n^2)$ time. If we further restrict the input to trains all entering from the right hand side and leaving to the lefthand side the constraint graph is a circular arc containment graph. We show how to color such graphs optimally in $\mathcal{O}(n \log n)$ time. The results are summarized in the following table.

	linear time table	cyclic time table
with midnight constraint	$\mathcal{O}(n \log n)$	3-approx within $\mathcal{O}(n^2)$
without midnight constraint without turning back trains	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$
without midnight constraint only type RL		$\mathcal{O}(n \log n)$

The distribution of this paper is as follows. In Sect. 2, we give a formal definition of the platform assignment problem and show that it is equivalent to coloring the conflict graph. In Sect. 3, we consider some variations of the linear platform assignment problem and in Sect. 4 variations of the cyclic platform assignment problem. As a byproduct, we give our algorithm for coloring circular arc containment graphs in Sect. 4.2. Finally, in Sect. 5, we discuss some open problems.

2 The Conflict Graph

In this section, we give a formal definition of the platform assignment problem with linear or cyclic timetable, define the conflict graph and show that the platform assignment problem is equivalent to the coloring problem on the conflict graph.

Linear Platform Assignment Problem (LPA) Given a set

$$Z = \{[t^{(1)}_{\text{arr}}, t^{(1)}_{\text{dep}}]d^{(1)}_{\text{arr}}d^{(1)}_{\text{dep}}, \ldots, [t^{(n)}_{\text{arr}}, t^{(n)}_{\text{dep}}]d^{(n)}_{\text{arr}}d^{(n)}_{\text{dep}}\}$$

of train labels $(t^{(i)}_{\text{arr}}, t^{(i)}_{\text{dep}} \in \mathbb{R}, t^{(i)}_{\text{arr}} < t^{(i)}_{\text{dep}}; d^{(i)}_{\text{arr}}, d^{(i)}_{\text{dep}} \in \{R, L\}\})$ with the property that all arrival and departure times are distinct and given a number k, is there an assignment $p : Z \to \{1, \ldots, k\}$ such that the following *train operation procedure* does not return false?

1. Sort arrival and departure times increasingly.

2. For all arrival and departure events in the sorted list
   (a) If the next event is an arriving train z. If z arrives from the right, append z to the righthand side of list $L_{p(z)}$ else append z to the lefthand side of $L_{p(z)}$.
   (b) If the next event is a departing train z. If z departs to the righthand side and is on the righthand side of list $L_{p(z)}$, or if z departs to the lefthand side and is on the lefthand side of list $L_{p(z)}$ then delete z from $L_{p(z)}$, else return false.

For a train label $z = [t_{\text{arr}}, t_{\text{dep}}]d_{\text{arr}}d_{\text{dep}}$, an integer i and a time period T set $z + iT = [t_{\text{arr}} + iT, t_{\text{dep}} + iT]d_{\text{arr}}d_{\text{dep}}$.

Cyclic Platform Assignment Problem (CPA) Given a set Z of train labels (with the property that all arrival and departure times are distinct), a time period T (such that $t_{\text{dep}} - t_{\text{arr}} < T$ for all $[t_{\text{arr}}, t_{\text{dep}}]d_{\text{arr}}d_{\text{dep}} \in Z$) , and a number k, is there an assignment $p : Z \to \{1, \ldots, k\}$ such that the above train operation procedure applied to $\{z + iT;\ z \in Z, i \in \mathbb{Z}\}$ (with $p(z + iT) = p(z)$) would never return false?

The *conflict graph* for a platform assignment problem (Z, k) or (Z, T, k), respectively, is defined as follows. The vertices are the train labels. Two vertices z_1 and z_2 are adjacent if and only if the platform assignment problem $(\{z_1, z_2\}, 1)$ or $(\{z_1, z_2\}, T, 1)$ is a false-instance.

Graph Coloring Problem Given a graph $G = (V, E)$ and a number k. Is there an assignment $c : V \to \{1 \ldots, k\}$ such that $c(v) \neq c(u)$ for $\{u, v\} \in E$?

Theorem 1. *Let G be the conflict graph of a platform assignment problem (Z, k) or (Z, T, k), respectively. Then the solutions of the platform assignment problem and the solutions of the graph coloring problem correspond.*

Proof. Let $p : Z \to \{1, \ldots, k\}$ be a solution of the platform assignment problem. Let $p(z_1) = p(z_2)$. Assume without loss of generality that z_1 has to leave the station before z_2 and that z_1 departs to the right hand side. If $\{z_1, z_2\}$ where an edge of G then, by definition of the conflict graph, z_2 would be on the right hand side of z_1 at the time that z_1 had to depart. But then p would not have been a valid assignment. Hence setting $c(z) = p(z), z \in Z$ yields a solution for the graph coloring problem.

Now, let $c : Z \to \{1, \ldots, k\}$ be a solution of the graph coloring problem. Set $c(z) = p(z), z \in Z$. Suppose that at some point the train operation procedure returns false. So suppose without loss of generality that at some point z_1 has to leave to the right hand side, but that there is another train z_2 in the list $L_{p(i)}$ on the right hand side of z_1. But then $\{z_1, z_2\}$ would be an edge in G – contradicting that c is a valid coloring. □

A *transitive orientation* of an undirected graph $G = (V, E)$ is an orientation of its edges with the following property.

$$(u, v) \in E \text{ and } (v, w) \in E \Longrightarrow (u, w) \in E$$

A *comparability graph* is a graph that has a transitive orientation. A *permutation graph* is a graph G that is associated with a permutation π of the set $\{1, \ldots, n\}$. The set of vertices of G is $\{1, \ldots, n\}$ and two vertices i and j with $i < j$ are adjacent if and only if $\pi(i) > \pi(j)$. A graph G is a permutation graph if and only if G and its complement are comparability graphs (see e.g. [7, p. 158]). The *chromatic number* of a graph G is the minimum number k such that (G, k) is a yes-instance of the graph coloring problem. A graph G is *perfect* if for all induced subgraphs H of G the size of a maximum clique of H equals the chromatic number of H. Comparability graphs and, hence, permutation graphs are perfect [7, p. 133].

The *first-fit algorithm* is a heuristic to solve the graph coloring problem. It works as follows. Start with an ordering $v_1, \ldots, v_n$ on the vertices. Color the vertices in this order. Assign vertex v_i the first color that was not assigned to an adjacent vertex of v_i. Chvátal [2] characterized all orderings that are such that the first-fit algorithm solves the graph coloring problem on the graph and all its induced subgraphs optimally. This is for example true, if the vertices are ordered according to a transitive orientation. In general, the first-fit algorithm runs in $\mathcal{O}(n^2)$ time. For permutation graphs, it can be implemented to run in $\mathcal{O}(n \log n)$ time (see e.g. [7, p. 168]). Note however that for arbitrary vertex-orderings the first-fit algorithm can behave arbitrarily bad even on permutation graphs [10].

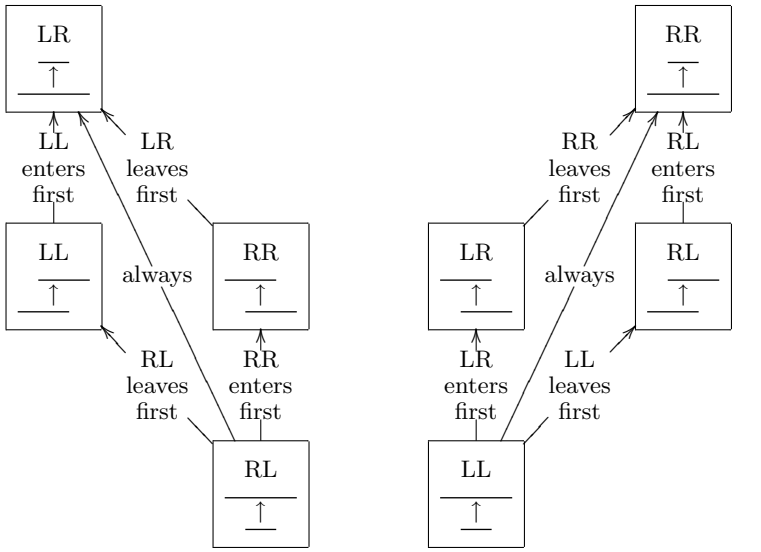
3 Linear Timetables

In this section, we consider three variations of the LPA. In Sect. 3.1, we assume that the midnight constraint is given. Else even the restricted problem in which every train enters from and leaves to the right hand side is $\mathcal{NP}$-complete: it corresponds to coloring circle graphs (see, e.g., [4] for a review). In Sect. 3.2, we consider an online solution for the LPA with midnight constraint. Finally, in Sect. 3.3, the midnight constraint need not be true, but there may not be any turning back trains. In the following let n be the number of trains in an instance of an LPA.

3.1 With Midnight Assumption

In this section, we assume that we are given an input of the LPA with the midnight constraint, i.e., that the time intervals intersect in at least one point. For example, in Fig. 1, 0 is a common point in all time intervals. Let $G_{\text{lin}}^{\text{mid}}$ be the conflict graph for this problem. A schematic image of $G_{\text{lin}}^{\text{mid}}$ is indicated in Fig. 3. Each rectangle represents all trains that have the same arrival and departure direction. E.g., the rectangle labeled RL represents all trains with $d_{\text{arr}} = R$ and $d_{\text{dep}} = L$. We will refer to this as the *type* of a train. Within one rectangle,  and  indicate that there is an edge between two trains if the time intervals overlap, but neither is contained in the other.  and  indicate that there is an edge between two trains if the time interval of one train is contained in the time interval of the other train. The edges between

the rectangles indicate when there is an edge between two trains of different type. An orientation of the edges is visualized by upward pointing arrows.



(a) conflict graph (b) complement of conflict graph

Fig. 3. A transitive orientation of the conflict graph and its complement.

Theorem 2. *The linear platform assignment problem with midnight constraint can be solved in* $\mathcal{O}(n \log n)$ *time.*

Proof. We show that $G_{\text{lin}}^{\text{mid}}$ is a permutation graph and that a representation as permutation can be found in $\mathcal{O}(n \log n)$ time. Hence, since permutation graphs can be colored in $\mathcal{O}(n \log n)$ time, the LPA with midnight constraint can be solved in the same asymptotic running time.

To show that $G_{\text{lin}}^{\text{mid}}$ is a permutation graph, we have to show that it is a comparability graph and that its complement is also a comparability graph. A transitive orientation of the conflict graph is indicated in Fig. 3. Exchanging LL with RL and LR with RR, one can see that the complement of the conflict graph has the same structure as the conflict graph itself. Hence it is also a comparability graph.

According to the general rule [7, p. 158], the permutation that corresponds to the conflict graph can be constructed as follows. First the trains are labeled according to the ordering that results from the transitive orientation of the conflict graph and its complement. Then the trains are permuted according to

the reversed transitive orientation of the conflict graph and the original transitive orientation of its complement. Hence, from the schematic representation in Fig. 3, we can immediately see, that for two trains z_1, z_2 and for each of the two orderings, we can decide in $\mathcal{O}(1)$ if z_1 is before z_2. Hence, since ordering can be done in $\mathcal{O}(n \log n)$ time, the representation of $G_{\text{lin}}^{\text{mid}}$ as a permutation can found in $\mathcal{O}(n \log n)$ time. □

The two orderings constructed in the proof of Theorem 2 for constructing the representation as a permutation for $G_{\text{lin}}^{\text{mid}}$ have a quite intuitive meaning. The trains departing to the left are labeled in increasing order of their departure time followed by the trains departing to the right in decreasing order of their departure time. This is exactly the order in which the trains can be positioned on one platform such that they can leave in time. For the permutation, the trains arriving from the left are ordered in decreasing order of their arrival time followed by the trains arriving from the right in increasing order of their arrival time. This corresponds exactly to the order in which the trains would be positioned at midnight if only one platform was given. E.g., the representation as permutation for the conflict graph of the LPA in Fig. 1 is constructed as follows. First the trains are labeled according to there departure direction and time.

$$1 : [-4,1]\text{RL},\ 2 : [-2,2]\text{RL},\ 3 : [-1,4]\text{LL},\ 4 : [-3,3]\text{RR}$$

Then they are permuted according to there arrival direction and time.

$$3 : [-1,4]\text{LL},\ 1 : [-4,1]\text{RL},\ 4 : [-3,3]\text{RR},\ 2 : [-2,2]\text{RL}$$

Hence the resulting permutation is $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \end{pmatrix}$.

3.2 Online Solution

The first-fit algorithm for coloring permutation graphs, colors the vertices in the order in which they occur in the permutation. I.e., in the solution of the LPA with midnight constraint given in Sect. 3.1, we first have to decide how to color the trains that enter last from the left hand side. Hence, it is not an *online-algorithm*, i.e., the trains are not assigned a platform in the order in which they enter the station. In fact, Fig. 1a showed that the first-fit algorithm that assigns platforms to trains in the order they enter the station would not necessarily yield the minimum number of platforms. The next theorem states that there is an algorithm that assigns platforms to trains in the order in which they enter the station which uses at most twice the number of platforms that the optimal offline-algorithm in Sect. 3.1 would use.

Theorem 3. *There is a 2-competitive online-algorithm for the linear platform assignment problem which runs in $\mathcal{O}(n \log n)$ time.*

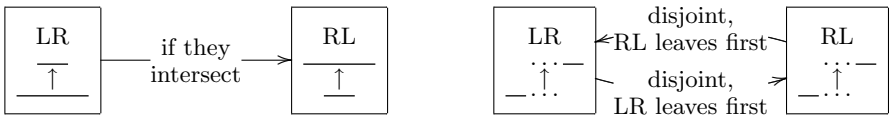
Proof. First, we divide the input into two subproblems – the trains entering from the left hand side and the trains entering from the right hand side. The conflict

graph for both subproblems is a permutation graph. For the trains entering from the right hand side, we use the representation as a permutation as it was computed in Sect. 3.1. For the trains entering from the left hand side, we reverse the orientation of the conflict graph and its complement given in Fig. 3. Then we apply the standard procedure described in the the proof of Theorem 2 on these two transitive orientations to obtain a representation as a permutation. Representing the conflict graphs of the two subproblems like that, the first-fit algorithm for coloring them is an online algorithm to solve the two subproblems optimally in $\mathcal{O}(n \log n)$ time. Hence, applying the first-fit algorithm with the additional constraint that a train that enters form the left hand side and a train that enters from the right hand side may not be put on the same platform yields a 2-competitive online-algorithm for solving the whole problem in $\mathcal{O}(n \log n)$ time. □

3.3 Without Turning Back Trains

In this section, we do not require the midnight constraint, but we assume that we have an instance of the linear platform assignment problem without turning back trains. Hence, we consider the quite typical case that a train coming from one side will continue its trip to the other side of the station. Fig. 4 shows that also in this case the conflict graph is a permutation graph. $\_.\overset{\cdot}{\uparrow}\overset{\cdot}{.}\overline{\_}$ indicates that there is an edge between two trains if neither of the time intervals is contained in the other. Hence, we can conclude the following theorem.

Theorem 4. *The linear platform assignment problem without turning back trains can be solved in $\mathcal{O}(n \log n)$ time.*



(a) conflict graph (b) complement of conflict graph

Fig. 4. Transitive orientations of the conflict graph and its complement for the linear platform assignment problem without turning back trains.

4 Cyclic Time Tables

In this section, let T denote the time period of a CPA and let n be the number of series of trains in the input. We represent an interval $[t_{\text{arr}}, t_{\text{dep}}]$ in the input as a circular arc $[t_{\text{arr}} \frac{2\pi}{T}, t_{\text{dep}} \frac{2\pi}{T}]$. The thus represented input for the example in Fig. 2

is shown in Fig. 6. In this section, we consider three subproblems of the CPA. In Sect. 4.1, we give a 3-approximation algorithm for, the CPA with midnight constraint. In Sect. 4.2, we first briefly consider the CPA without turning back trains in general. Then we solve the special case that there are only trains of type RL or only trains of type LR by giving an $\mathcal{O}(n \log n)$ algorithm for coloring the class of graphs called circular arc containment graphs.

4.1 With Midnight Assumption

In this section, we assume again that the midnight-assumption holds. Hence, without loss of generality, we may assume that 0 is contained in the intersection of all intervals of the input. If there is only one type of trains (among LL, LR, RL, RR) the midnight-assumption implies that cyclic time tables can be handled in the same way as simple intervals. This changes, however, if there are trains that differ in either the incoming direction or the outgoing direction (but not both) such that the intersection of its time arcs is not connected, i.e. if the difference between the departure time of the train that leaves last and the arrival time of the train that arrives first is greater or equal T. In fact, in a solution that repeats every time period, two such trains can never be put on the same platform. The conflict graph $G_{\text{cyc}}^{\text{mid}}$ of the CPA with midnight constraint is illustrated in Fig. 5.

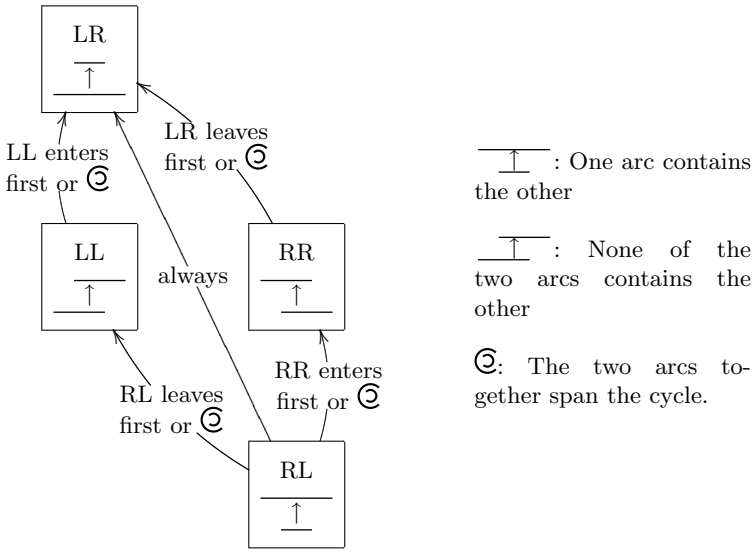


Fig. 5. The conflict graph for the cyclic platform assignment problem with midnight constraint.

If there are only trains of type LL and RR involved, $G_{\text{cyc}}^{\text{mid}}$ is always a permutation graph. The case in which the input consists only of trains of type LR

and RL will be consider more generally (without midnight constraint) in the next subsection. In general, $G^{\text{mid}}_{\text{cyc}}$ need not be perfect, even if only trains of type RR/LR, RR/RL, LR/LL, or RL/LL are involved. See Fig. 6 for an example. The time complexity of the CPA with midnight constraint in these cases is sofar open.

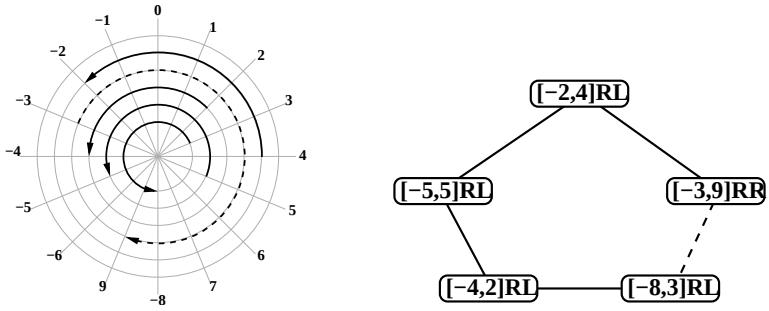


Fig. 6. The cyclic time table considered in Fig. 2 and its conflict graph which is a C_5 and hence not perfect. Clockwise arcs indicate trains leaving to the right and counter clockwise arcs indicate trains leaving to the left. Dashed arrows indicate turning-back trains. The dashed edge in the conflict graph is of type ℭ.

Theorem 5. *There is a 3-approximation algorithm for the cyclic platform assignment problem with midnight constraint that works in $\mathcal{O}(n^2)$ time.*

Proof. We consider the acyclic orientation on the graph that is indicated in Fig. 5. The height function which is defined recursively by setting

$$h(v) = \begin{cases} 1 & \text{if v is a sink,} \\ 1 + \max\{h(w);\ (v,w) \in E\} & \text{otherwise} \end{cases}$$

is a proper coloring of $G^{\text{mid}}_{\text{cyc}}$ and the number of colors $\chi(h)$ that are used is equal to the number of vertices in the longest directed path in $G^{\text{mid}}_{\text{cyc}}$ [7, p. 132]. Note that the orientation within the four types RR, LL, RL, an LR is even transitive. Any directed path P in $G^{\text{mid}}_{\text{cyc}}$ contains trains of at most three of these types and all trains of one type are consecutive in P. Hence, the subgraph induced by P can be covered by three cliques – one for each type of train that is contained in P. Let ω be the size of a maximum clique in $G^{\text{mid}}_{\text{cyc}}$ and let χ be the chromatic number of $G^{\text{mid}}_{\text{cyc}}$. Then $\chi(h) \leq 3\omega \leq 3\chi$. Hence, the height function yields a 3-approximation algorithm.

The height function can be computed in time linear in the size of $G^{\text{mid}}_{\text{cyc}}$ by a DFS-algorithm [7, p. 46] and hence in $\mathcal{O}(n^2)$ time. □

4.2 Without Turning Back Trains

In case only trains of type LR and RL are involved, the conflict graph remains a comparability graph, even without the midnight constraint. See Fig. 7a for the transitive orientation. Hence, the conflict graph can be colored in $\mathcal{O}(n^2)$ time. Fig. 7b demonstrates, however, that in that case the conflict graph does not have to be a permutation graph anymore.

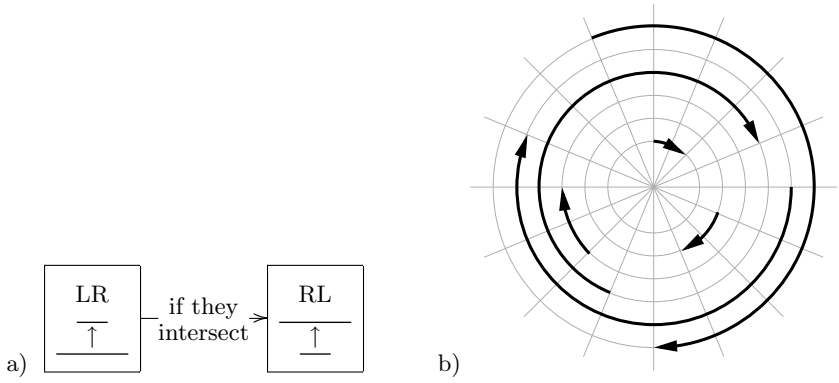


Fig. 7. Trains of type LR and RL only, without requiring the midnight constraint. a) Transitive orientation of the conflict graph b) Example for which the conflict graph is a C_6 and, hence, not a permutation graph – even if only trains of type LR are involved.

In the rest of this section, we now consider the case that there are only trains of type RL or only trains of type LR. In this case, the conflict graph is a circular arc containment graph (CACG) [12]. A CACG is a graph whose vertices are circular arcs and in which two circular arcs are adjacent if and only if one of them is contained in the other. In what follows, let n be the number of vertices and m the number of edges of a CACG. A maximum clique of a CACG and, hence (since CACGs are comparability graphs and, hence, perfect), the chromatic number of a CACG can be determined in $\mathcal{O}(n \log\log n)$ time [9]. Nirkhe [11] showed that a coloring for a CACG can be found in $\mathcal{O}(n \log n + m)$ time. We will show how to modify the first-fit algorithm for permutation graphs such that it can be applied to color circular arc containment graphs in $\mathcal{O}(n \log n)$ time.

We will refer to a *circular arc* by an interval $[a, b]$ with $-2\pi < a < 2\pi, ; 0 < b \leq 2\pi,\ 0 < b - a < 2\pi$. A circular arc $[a, b]$ is contained in a circular arc $[c, d]$ if

1. $c < 0$ and $a > 2\pi + c$ or if
2. $c < a$ and $d > b$.

Let G be the CACG on the set $V = \{[a_1, b_1], \ldots, [a_n, b_n]\}$ of n circular arcs. We may assume that all $a_1, \ldots, a_n, b_1, \ldots, b_n$ are distinct [11].

Since ordering can be done in $\mathcal{O}(n \log n)$ time, we assume that the arcs in V are ordered as follows.

$$i < j \text{ iff } a_i < a_j$$

Then, clearly, this induces a transitive orientation on G. We apply the first-fit algorithm. To obtain the required running time, we apply the algorithm below, maintaining the following arrays.

color: $\textsc{color}(i)$ is the list of vertices that is colored i.
first: $\textsc{first}(i) = 2\pi$ if so far no arc has been colored with color i. Else, let $[a, b]$ be the first arc that is colored with color i. Then $\textsc{first}(i) = a + 2\pi$.
last: $\textsc{last}(i) = 0$ if so far no arc has been colored with color i. Else, let $[a, b]$ be the last arc that is colored with color i. Then $\textsc{last}(i) = b$.

1. **for** $j = 1, \ldots, n$
2. Find smallest k such that $\textsc{first}(k) > a_j$
3. Find smallest $i \geq k$ such that $\textsc{last}(i) < b_j$
4. **if** $\textsc{color}(i)$ is empty
5. Set $\textsc{first}(i) = a_j + 2\pi$
6. Append $[a_j, b_j]$ to $\textsc{color}(i)$
7. Set $\textsc{last}(i) = b_j$

Lemma 1. *The above algorithm solves the graph coloring problem on the class of circular arc containment graphs in $\mathcal{O}(n \log n)$ time.*

Proof. First, we show that the algorithm is the first-fit algorithm and, hence, solves the problem optimally. (Recall that the arcs are ordered according to a transitive orientation.) Note that no not-yet colored arc contains an already colored arc. There are the following two cases in which arc $[a_j, b_j]$ is contained in the arc $[a, b]$ colored by color i.

1. $a_j > a + 2\pi$. By construction, $a + 2\pi \geq \textsc{first}(i)$. Hence, $a_j > \textsc{first}(i)$.
2. $b_j < b$. Hence, since by construction $b \leq \textsc{last}(i)$, it follows $b_j < \textsc{last}(i)$.

Hence, Line 2 and 3 of the algorithm guarantee that $[a_j, b_j]$ is colored with the first color that was not used to color an adjacent arc of $[a_j, b_j]$.

For the running time observe the following. $\textsc{first}(i)$ is monotonely increasing, i.e. $\textsc{first}(i) \leq \textsc{first}(j)$ for $i < j$. The value k that is chosen in Line 2 of the algorithm is monotonely increasing with every step. $\textsc{last}(i)$ is monotonely decreasing for $i \geq k$, i.e. $\textsc{last}(i) \geq \textsc{last}(j)$ for $k \leq i < j$. Hence, Line 2 and 3 can both be performed in logarithmic time by binary search. It follows that the over all running time is in $\mathcal{O}(n \log n)$. □

5 Open Problems

We have considered some variations of the platform assignment problem with linear and cyclic time tables. In the following, we list some interesting questions concerning this topic that sofar remained open.

– How bad can the first-fit algorithm behave for solving the linear platform assignment problem with midnight constraint as an online problem? Recall that the first-fit algorithm on arbitrary vertex-orderings of a permutation graph can behave arbitrarily bad [10].
– Is the cyclic platform assignment problem with midnight constraint $\mathcal{NP}$-complete?
– A solution for the cyclic platform assignment problem in which trains from a series do not have to be assigned to the same platform could use less platforms than a solution that has to be the same every time period. See Fig. 2 for an example. How big can the difference be?

References

1. U. Blasum, M. R. Bussieck, W. Hochstättler, C. Moll, H.-H. Scheel, and T. Winter. Scheduling trams in the morning. *Mathematical Methods of Operations Research*, 49(1):137–148, 1999.
2. V. Chvátal. Perfectly ordered graphs. In *Topics on Perfect Graphs*, volume 21 of *Annals of Discrete Mathematics*, pages 63–65. North-Holland, 1984.
3. E. Dahlhaus, P. Horak, M. Miller, and J. F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103(1–3):41–54, 2000.
4. G. Di Stefano and M. L. Koci. A graph theoretical approach to the shunting problem. In *Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003)*, volume 92 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2003.
5. R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman. Shunting of passenger train units in a railway station. Technical Report EI2002-26, Econometric Institute, Erasmus University Rotterdam, `http://www.eur.nl/WebDOC/doc/econometrie/feweco20020917130601.pdf`, 2002. To appear in Transportation Science.
6. G. Gallo and F. Di Miele. Dispatching buses in parking depots. *Transportation Science*, 35(3):322–330, 2001.
7. M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Computer Science and Applied Mathematics. Academic Press, 1980.
8. S. He, R. Song, and S. S. Chaudhry. Fuzzy dispatching model and genetic algorithms for railyards operations. *European Journal of Operational Research*, 124(2):307–331, 2000.
9. R. D. Lou and M. Sarrafzadeh. Circular permutation graph family with applications. *Discrete Applied Mathematics*, 40:433–457, 1992.
10. S. D. Nikolopoulos and C. Papadopoulos. On the performance of the first-fit coloring algorithm on permutation graphs. *Information Processing Letters*, 75:265–273, 2000.
11. M. V. Nirkhe. Efficient algorithms for circular-arc containment graphs. Master's thesis, University of Maryland, 1987. `http://techreports.isr.umd.edu/report/1987/MS_87-11.pdf`.
12. A. Rossi. Il problema dell'ordinamento dei treni in un deposito: modellazione e soluzione algoritmica. Master's thesis, Università dell'Aquila, 2003.
13. T. Winter and U. T. Zimmermann. Real-time dispatch of trams in storage yards. *Annals of Operations Research*, 96:287–315, 2000.