Track Assignment [1]

Sabine Cornelsen

Universität Konstanz, Fachbereich Informatik & Informationswissenschaft,
cornelse @ inf.uni-konstanz.de

Gabriele Di Stefano

Università dell'Aquila, Dipartimento di Ingegneria Elettrica e dell'Informazione,
gabriele @ ing.univaq.it

Abstract

We consider a station in which several trains might stop at the same track at the same time. The trains might enter and leave the station from both sides, but the arrival and departure times and directions are fixed according to a given time table. The problem is to assign tracks to the trains such that they can enter and leave the station on 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 optimal $\mathcal{O}(n \log n)$ coloring algorithm.

Key words: graph algorithms, graph coloring, circular arc containment graphs, railway optimization

1 Introduction

The shunting problem concerns the rolling stock allocation on a railway infrastructure under time, space and operational constraints. It occurs in practical railway optimization problems like, e.g., the storage of trains, trams or buses in a depot outside the rush hours, the rearrangement of railroad cars among

[1] Work partially supported by the Human Potential Program of the EU under contract no. HPRN-CT-1999-00104 (AMORE Project), and by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

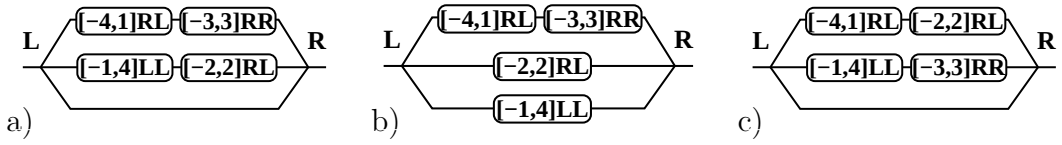

Fig. 1. a) A non-feasible track assignment. b) A feasible track assignment that is achieved with the first-fit algorithm. c) A feasible track assignment that uses the minimum number of tracks.

different trains, the freight car distribution, and also the assignment of trains to platforms in a station [1,4–7,9–11,18,20,21].

However, appropriate mathematical models which take into account all the routing restrictions, capacity limits, and also time and operating constraints that arise in this context are quite complex. Thus, motivated by the purpose of analyzing the abstract background of this problem and of gaining more insight into some of its aspects, we study the following combinatorial problem. Even though it is quite idealized to directly model a real situation, it seems to be somehow natural and it allows to capture important structural properties related to time and capacity constraints. We will formulate the problem in terms of stations and trains, but it can be restated in terms of depots, and trams, buses (or trains), as well.

The problem. We are given a set of trains that has to enter a station consisting of a set of parallel tracks. 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 or to 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 arrival and departure direction d_{arr} and d_{dep}, respectively. For example, a train labeled [14,15]RL is a train that enters the station at 14 from the right-hand side and leaves the station at 15 to the left-hand side. In the meanwhile, it stops on one of several parallel tracks in the station. There might be several trains waiting at the same track. This situation occurs, e.g., with short local trains but also with trams or buses.

The problem is to assign each train to a track such that it can leave the station on 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, which uses the minimum number of tracks. Knowing it, one can decide, whether a time table can be run at all on a given infrastructure.

We consider both linear and cyclic time tables. In cyclic time tables, the situa-

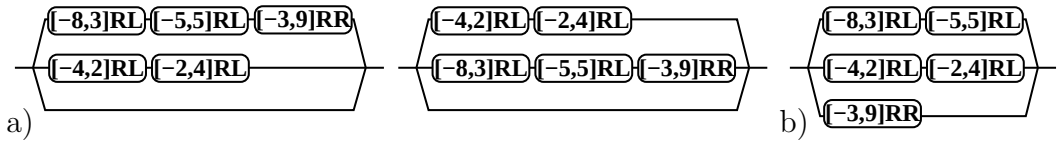

Fig. 2. Cyclic track assignment with time period $T = 16$. a) A track assignment that repeats itself every second time period. b) A cyclic track assignment that repeats itself every time period.

tion repeats itself each time period T, i.e., a label $[t_{\text{arr}}, t_{\text{dep}}]d_{\text{arr}}d_{\text{dep}}$ 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 $t_{\text{dep}} + iT$ from direction d_{dep}. We further assume that no train stays in the station longer than the time period T, i.e., that $t_{\text{dep}} - t_{\text{arr}} < T$. The goal is a track 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 track. E.g., we are not interested in an assignment as indicated in Fig. 2a, but rather in an assignment as indicated in Fig. 2b.

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 the common situation in which all trains enter a shunting depot in the evening and leave it in the morning. It is doubtlessly a reasonable constraint for facing the problem of storing trains or trams in a depot. But the midnight constraint even makes sense in the case of assigning trains to platforms in a station. Consider, e.g., the model called integrated timetables, or ITF (Integraler Taktfahrplan) which is applied in Switzerland and parts of Germany. An ITF is a cyclic time table such that in some linkup stations all vehicles (trains, trams, buses) of all lines periodically meat each other such that passengers can change from any line to any other line. For a good introduction to ITF and a discussion about its advantages and disadvantages see [13].

We further assume that the tracks have infinite length. Even though this is an unrealistic assumption, it immediately leads to a 2-approximate solution of the same problem when a maximum number k of trains per track is imposed: Divide the set of trains that is assigned to one track into some sets of k trains and one set of at most k trains. In the conclusion we discuss the computational complexity of solving the track assignment problem with finite tracks optimally.

Results. Given a track assignment problem, we define the following *conflict 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 track. The track assignment problem then corresponds to coloring the conflict graph, where the colors represent the different tracks.

We say that a train is a *turning back train*, if it leaves the station in the direction from where it entered it, i.e., if $d_{\text{arr}} = d_{\text{dep}}$. In the cases of linear time tables with midnight constraint or without turning back trains, we show that the conflict graph is a permutation graph and hence it can be colored in $\mathcal{O}(n \log n)$ time. However, the transformation of the track assignment problem to a graph coloring problem requires the knowledge of the whole sequence of trains. When an assignment of an incoming train must be performed without any information about the remaining trains in the sequence, we have the online version of the problem. To solve it we give a 2-competitive online-algorithm running 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 2-approximation algorithm that runs in $\mathcal{O}(n \log n)$ time. Cyclic time tables without turning back trains yield a conflict 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 left-hand side the conflict 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	$\mathcal{O}(n \log n)$ 2-competitive within $\mathcal{O}(n \log n)$	2-approx within $\mathcal{O}(n \log n)$
without midnight constraint without turning back trains	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$
without midnight constraint only type RL	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$

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

A preliminary version of this paper was previously presented at the 4th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004) and will be published in the corresponding conference proceedings [3].

2 The Conflict Graph

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

The expression $[t_{\text{arr}}, t_{\text{dep}}]d_{\text{arr}}d_{\text{dep}}$ is a train label if $t_{\text{arr}}, t_{\text{dep}} \in \mathbb{R}$, $t_{\text{arr}} < t_{\text{dep}}$, and $d_{\text{arr}}, d_{\text{dep}} \in \{R, L\}$. A set Z of train labels is *assignable to a track* (or simply *assignable*) if the elements of Z can be ordered in a sequence $S = (z_1, z_2, \ldots, z_n)$, $z_i = [t^{(i)}_{\text{arr}}, t^{(i)}_{\text{dep}}]d^{(i)}_{\text{arr}}d^{(i)}_{\text{dep}}$ such that:

input constraint: for each train $z_i \in S$ such that $d^{(i)}_{\text{arr}} = L$ ($d^{(i)}_{\text{arr}} = R$), any train $z_j \in S$, $j < i$ ($j > i$), is such that $d^{(j)}_{\text{arr}} = L$ ($d^{(j)}_{\text{arr}} = R$) and $t^{(j)}_{\text{arr}} > t^{(i)}_{\text{arr}}$.

output constraint: for each train $z_i \in S$ such that $d^{(i)}_{\text{dep}} = L$ ($d^{(i)}_{\text{dep}} = R$), any train $z_j \in S$, $j < i$ ($j > i$), is such that $t^{(j)}_{\text{dep}} < t^{(i)}_{\text{dep}}$ or $t^{(j)}_{\text{arr}} > t^{(i)}_{\text{dep}}$.

The meaning of the input constraint is that S can be divided into a sequence $(z_1, z_2, \ldots, z_{t-1})$ of trains that arrive from the left and are in decreasing order of their arrival time, and the remaining trains $(z_t, z_{t+1}, \ldots, z_n)$, that come from the right and are in increasing order of their arrival time. Note that the input constraint uniquely determines the ordering of the train labels in the sequence S: It is the ordering in which the arriving trains are put on one track without shunting operations. Trains arriving from the same direction at the same time may not be put on the same track. The output constraint says that a departing train z_i must not be obstructed by other trains: They either depart before or arrive after the time $t^{(i)}_{\text{dep}}$.

Linear Track Assignment Problem (LTA) Given a finite set Z of train labels and a positive integer k, is there a map $p : Z \to \{1, \ldots, k\}$ such that each set $L_j = \{z;\ p(z) = j, z \in Z\}$ is assignable?

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 Track Assignment Problem (CTA) Given a finite set Z of train labels, a time period T with $t_{\text{dep}} - t_{\text{arr}} < T$ for all $[t_{\text{arr}}, t_{\text{dep}}]d_{\text{arr}}d_{\text{dep}} \in Z$, and two positive numbers k and N, is there a map $p : Z \to \{1, \ldots, k\}$ such that each set $L_j = \{z + iT;\ p(z) = j,\ z \in Z,\ i \in [0, \ldots, N]\}$ is assignable?

The *conflict graph* for a track assignment problem (Z, k) or (Z, T, k, N), respectively, is defined as follows. Z is the set of vertices. Two vertices z_1 and z_2

are adjacent if and only if $\{z_1, z_2\}$ or $\{z_1 + i_1T, z_2 + i_2T;\ i_1, i_2 \in [0, \ldots, N]\}$, respectively, is not assignable.

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 $\{u, v\} \in E$?

The next theorem states in particular that a set of train labels is assignable if and only if each pair of train labels is assignable. Note that for other shunting problems, e.g., if the arrival or departure directions are not fixed, this is no longer true (see, e.g., [5]).

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

PROOF. Let (Z, k) or (Z, T, k, N), respectively, be a track assignment problem and let G be its conflict graph. In the linear case let $T = N = 0$. First, let $p : Z \to \{1, \ldots, k\}$ be a solution of the track assignment problem. Let $p(z_1) = p(z_2) = j$. Then $\{z_1 + i_1T, z_2 + i_2T;\ i_1, i_2 \in [0, \ldots, N]\}$ is a subset of the assignable set $L(j)$ and, hence, assignable. Thus, z_1 and z_2 are not adjacent in the conflict graph G. Hence p is a solution for the graph coloring problem of G.

Now, let $c : Z \to \{1, \ldots, k\}$ be a solution of the graph coloring problem of G. Consider the set $L_j = \{z + iT;\ c(z) = j,\ z \in Z,\ i \in [0, \ldots, N]\}$ for a $j \in \{1, \ldots, k\}$. By the definition of the constraint graph and the graph coloring problem, any pair of train labels in L_j is assignable. Especially, there are no two train labels in L_j that have the same arrival time and direction. Hence, let S be the ordered sequence of L_j as it was determined by the input constraint. Suppose that L_j is not assignable. Then there are two train labels $z_1 + i_1T, z_2 + i_2T \in L_j$ such that the output constraint is not fulfilled. But then $\{z_1 + i_1T, z_2 + i_2T\}$ would be an edge in G – contradicting that c is a valid coloring. Hence, c is a solution for the track assignment problem. □

For an easier discussion, we assume in the following that there are no two trains arriving from the same direction at the same time or departing to the same direction at the same time. Note that without this requirement the conflict graphs have to be modified by adding edges between such pairs of trains. However, the resulting graph classes remain the same.

A *transitive orientation* of an undirected graph $G = (V, E)$ is an orientation of its edges such that: $(u, v) \in E$ 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. [17,8, p. 158]). In that case, a representation of the graph G as a permutation π can be obtained as follows.

First the vertices are labeled according to the ordering that results from the transitive orientation of the conflict graph and its complement. That means, the vertices are assigned distinct labels among $\{1, \ldots, n\}$ such that vertex v has a lower label than vertex w if and only if edge $\{v, w\}$ is directed from v to w. Then the trains are permuted according to the reversed transitive orientation of the conflict graph and the original transitive orientation of its complement. That means, the permutation π is such that $\pi(i) < \pi(j)$ if and only if there is an edge in G directed from the vertex labeled j to the vertex labeled i or there is an edge in the complement of G directed from the vertex labeled i to the vertex labeled j.

The *chromatic number* of a graph G is the minimum number χ such that (G, χ) 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 [8, 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. Given the representation as a permutation π, the first-fit algorithm can be implemented to run in $\mathcal{O}(n \log n)$ time for permutation graphs (see e.g. [8, p. 168]). Note however that for arbitrary vertex-orderings the first-fit algorithm can behave arbitrarily bad even on permutation graphs [15].

A circular arc containment graph (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. A maximum clique of a CACG with n vertices and m edges 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 [14]. Nirkhe [16] showed that a coloring for a CACG can be found in $\mathcal{O}(n \log n + m)$ time.

3 Linear Timetables

In this section, we consider three variations of the LTA. In Sect. 3.1, we assume that the midnight constraint is given. Without this assumption, 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 [19]. In Sect. 3.2, we consider an online algorithm for the LTA with midnight constraint. Finally, in Sect. 3.3, we do not assume that the midnight constraint holds. However, we require that there are no turning back trains. In the following let n be the number of trains in an instance of an LTA.

3.1 With Midnight Assumption

In this section, we assume that we are given an input of the LTA with the midnight constraint, i.e., that all the time intervals intersect in at least one point. For example, in Fig. 1, time 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. A horizontal line within one rectangle indicates the time interval. I.e., 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. For example, indicates that the edge between a train z_1 and a train z_2 with the property that z_2 arrives before z_1 and leaves before z_1 is pointing from z_1 to z_2. indicates that the edge between a train z_1 and a train z_2 with the property that z_1 arrives before z_2 and that z_2 leaves before z_1 is pointing from z_1 to z_2.

We show that the LTA with midnight constraint can be solved in $\mathcal{O}(n \log n)$ time by demonstrating that $G_{\text{lin}}^{\text{mid}}$ is the permutation graph for the following permutation $\pi_{\text{lin}}^{\text{mid}}$. 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 track such that they can leave on 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

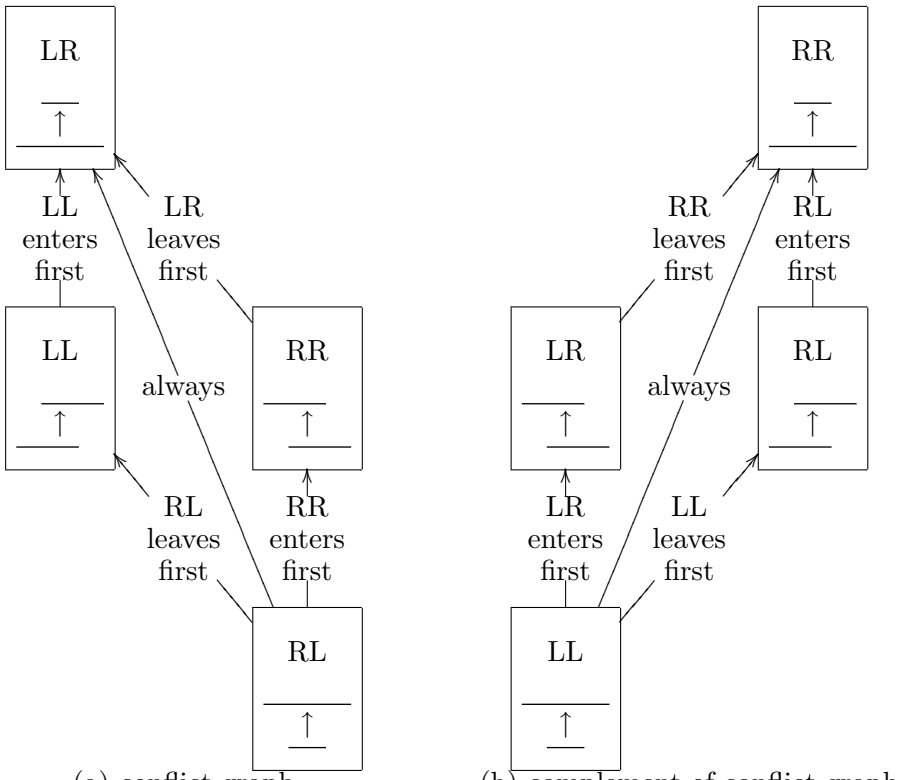

Fig. 3. A transitive orientation of the conflict graph and its complement. Note that there is an edge in the complement of the conflict graph if and only if it is incident to two trains that are assignable to one track.

which the trains would be positioned at midnight if only one track was given.

Before we give the proof, we first illustrate this representation as permutation for the conflict graph of the LTA in Fig. 1. First the trains are labeled according to their departure direction and time.

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

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

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

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

A direct proof that $G_{\mathrm{lin}}^{\mathrm{mid}}$ is indeed the permutation graph of the permutation $\pi_{\mathrm{lin}}^{\mathrm{mid}}$ would yield a nasty case distinction. Instead, we show that $\pi_{\mathrm{lin}}^{\mathrm{mid}}$ can be constructed from a transitive orientation of $G_{\mathrm{lin}}^{\mathrm{mid}}$ and its complement as indicated on page 7.

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

PROOF. A case distinction reveals that the orientations of the conflict graph $G_{\text{lin}}^{\text{mid}}$ and its complement as defined in Fig. 3 are transitive. Hence, $G_{\text{lin}}^{\text{mid}}$ is a permutation graph. Consider now the ordering that results from the transitive orientation of the conflict graph and its complement. From the schematic representation of the transitive orientations in Fig. 3, it is easy to see that the trains departing to the left are ordered in increasing order of their departure time followed by the trains departing to the right in decreasing order of their departure time. Further, it is also easy to see that the reversed transitive orientation of the conflict graph and the original transitive orientation of its complement imply that 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. Hence, $G_{\text{lin}}^{\text{mid}}$ is the permutation graph of $\pi_{\text{lin}}^{\text{mid}}$.

Since ordering can be done in $\mathcal{O}(n \log n)$ time, the representation $\pi_{\text{lin}}^{\text{mid}}$ of $G_{\text{lin}}^{\text{mid}}$ as a 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 on the basis of their underlying permutation, the LTA with midnight constraint can be solved in the same asymptotic running time.

Note that we do not have to construct the conflict graph to find its representation as a permutation. □

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 LTA 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 track in the order in which they enter the station. In fact, Fig. 1b showed that the first-fit algorithm that assigns tracks to trains in the order they enter the station would not necessarily yield the minimum number of tracks. The next theorem states that there is an algorithm that assigns tracks to trains in the order in which they enter the station which uses at most twice the number of tracks that the optimal offline-algorithm in Sect. 3.1 would use.

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

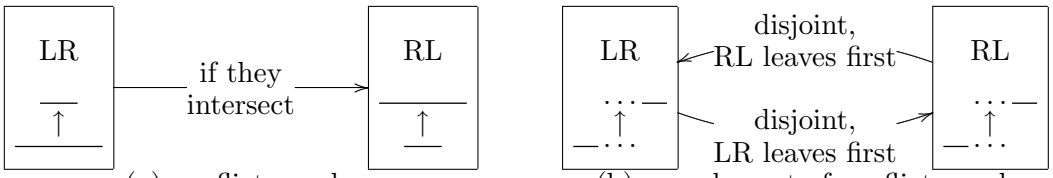

Fig. 4. Transitive orientations of the conflict graph and its complement for the linear track assignment problem without turning back trains.

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. Consider for the trains entering from the right-hand side the permutation as it was given 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 and use the representation as a permutation obtained from this orientation. In either case the permutation is in increasing ordering of the arrival time. Hence, in either case, the first fit algorithm is performed on a permutation graph on the basis of an underlying permutation, which means that the two subproblems are solved optimally in $\mathcal{O}(n \log n)$ time. Note that we do not need to compute the whole graph nor the labeling to apply the coloring algorithm for permutation graphs. It suffices to know for each pair of trains which one has the lower label. □

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 track 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. An easy case distinction reveals that the orientations indicated in Fig. 4 are transitive orientations of the conflict graph and its complement. $\_\dot{\uparrow}\dot{\_}$ indicates that there is an edge between two trains if neither of the time intervals is contained in the other. Hence, also in this case the conflict graph is a permutation graph and we can conclude the following theorem.

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

4 Cyclic Time Tables

In this section, let T denote the time period of a CTA 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 CTA. In Sect. 4.1, we give a 2-approximation algorithm for the CTA with midnight constraint. In Sect. 4.2, we first briefly consider the CTA 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 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 there is a conflict within different time periods only if there is a conflict within one time period. Hence cyclic time tables can be handled in the same way as linear time tables. This changes, however, if there are trains that differ in either the arrival direction or the departure direction (but not both) such that the intersection of their 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, such two trains can never be put on the same track. The conflict graph $G^{\text{mid}}_{\text{cyc}}$ of the CTA with midnight constraint is illustrated in Fig. 5.

If there are only trains of type LL and RR involved, $G^{\text{mid}}_{\text{cyc}}$ is a subgraph of $G^{\text{mid}}_{\text{lin}}$ and, hence, it is always a permutation graph. The case in which the input consists only of trains of type LR and RL will be considered more generally (without midnight constraint) in the next subsection. In general, $G^{\text{mid}}_{\text{cyc}}$ need not to 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 computational complexity of the CTA with midnight constraint in these cases is sofar open.

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

PROOF. Consider the partition of $G^{\text{mid}}_{\text{cyc}}$ into two subgraphs G_1 and G_2: the one induced by trains of type RL and LR on one hand and the one induced by trains of type RR and LL on the other hand. G_1 and G_2 are both permutation

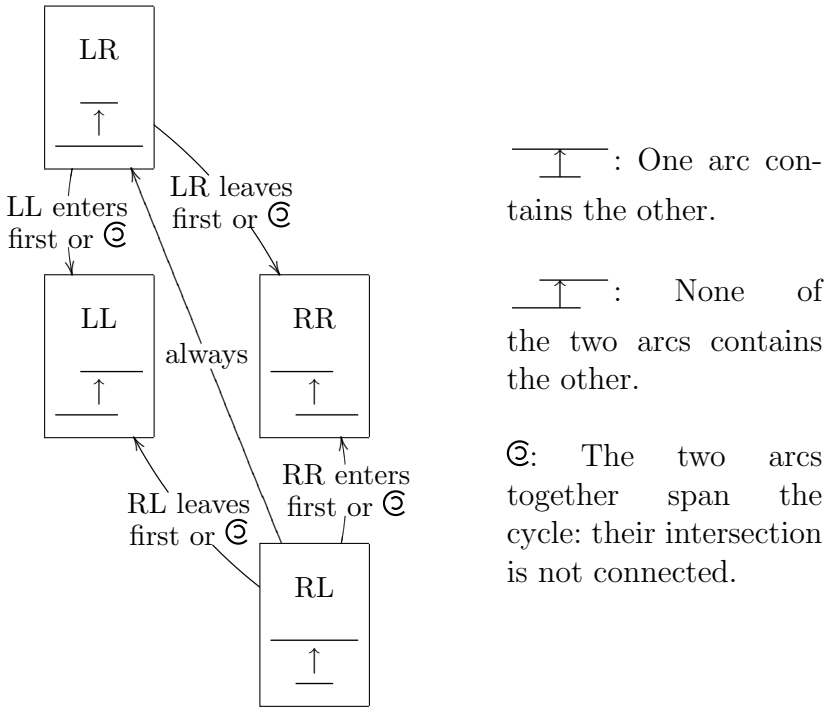

Fig. 5. The conflict graph for the cyclic track assignment problem with midnight constraint.

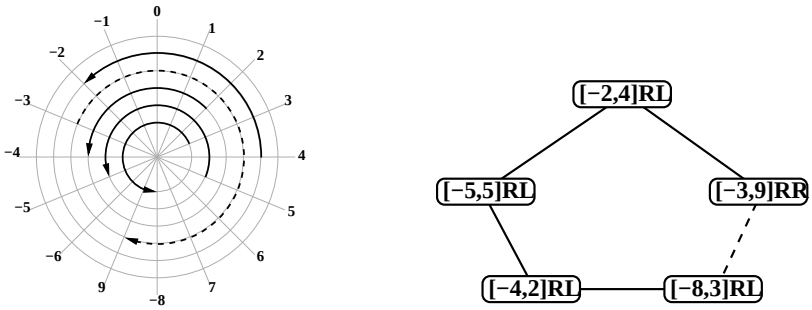

Fig. 6. The cyclic time table considered in Fig. 2 and its conflict graph which is a C_5 and hence not perfect. Clockwise arcs indicate trains leaving to the right and counter clockwise arcs indicate trains leaving to the left. Dashed arrows indicate turning-back trains. The dashed edge in the conflict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

graphs, hence we can color them optimally in $\mathcal{O}(n \log n)$ time. Using different colors for G_1 and G_2 this yields a feasible coloring of $G_{\text{cyc}}^{\text{mid}}$ with at most twice the colors necessary. □

A coloring that is at least as good as the one in the proof of Theorem 5, but might be better in practice can be obtained as follows. We consider the acyclic orientation on the graph that is indicated in Fig. 5. The *height function*

is defined recursively by setting

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

Theorem 6 *The height function is a 2-approximation algorithm for the cyclic track assignment problem with midnight constraint that works in $\mathcal{O}(n^2)$ time.*

PROOF. The height function is a proper coloring of $G_{\text{cyc}}^{\text{mid}}$ and the number of colors $\chi(h)$ that are used is equal to the number of vertices in a longest directed path P in $G_{\text{cyc}}^{\text{mid}}$ [8, p. 132]. The height function can be computed in time linear in the size of $G_{\text{cyc}}^{\text{mid}}$ by a DFS-algorithm [8, p. 46] and hence in $\mathcal{O}(n^2)$ time.

Like in the proof of Theorem 5, let us consider the partition of $G_{\text{cyc}}^{\text{mid}}$ into two subgraphs G_1 and G_2: the one induced by trains of type RL and LR on one hand and the one induced by trains of type RR and LL on the other hand. Note that the induced orientation on G_1 and G_2 is transitive. Hence, any directed path within G_1 or G_2 induces a clique. In any directed path in $G_{\text{cyc}}^{\text{mid}}$ no train of G_2 is followed by a train of G_1. Hence, the subgraph induced by P can be covered by two cliques – one for G_1 and one for G_2. Thus, $\chi(h)$ is lower or equal to the sum of the chromatic numbers of G_1 and G_2. Hence, the height function yields a coloring that is at least as good as the one in Theorem 5. □

In practice, the height function might be better than the solution indicated in the proof of Theorem 5, however, in the worst case, it is not: Consider the following set of $n = 2k, k > 1$ trains, k trains of type LL and k trains of type RL.

$$\begin{array}{llll} [& -\frac{i}{8k}T, & \frac{i+3k-1}{8k}T &] \quad \text{RL}, \quad i = 1, \ldots, k-1, \\ [& -\frac{1}{8}T, & \frac{4k+1}{8k}T &] \quad \text{RL}, \\ [& -\frac{4k-i}{8k}T, & \frac{i}{8k}T &] \quad \text{LL}, \quad i = 2, \ldots, k, \text{ and} \\ [& -\frac{4k+1}{8k}T, & \frac{1}{8k}T &] \quad \text{LL} \end{array}$$

Then the conflict graph $G_{\text{cyc}}^{\text{mid}}$ consists of two cliques of size k that are connected by a single edge. Hence it can be colored with k colors. The height function, however, would yield a coloring with $2k$ colors.

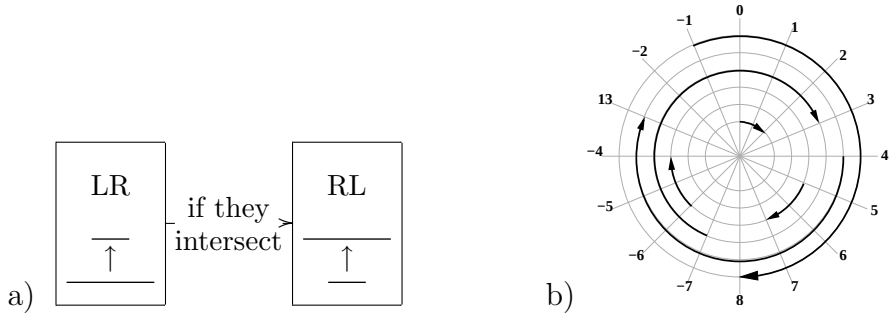

Fig. 7. Trains of type LR and RL only, without requiring the midnight constraint. a) Transitive orientation of the conflict graph b) Example for which the conflict graph is a C_6 and, hence, not a permutation graph – even if only trains of type LR are involved.

4.2 Without Turning Back Trains

In this section, we do not require the midnight constraint. Without loss of generality we may, however, assume that all arrival times are between 0 and the time period T. 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. Note, however, that in this case the conflict graph does not have to be a permutation graph anymore. For example, the conflict graph of the instance in Fig. 7b is a cycle on 6 vertices and, hence, its complement does not have a transitive orientation.

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 [18]. In what follows, let n be the number of vertices of a CACG. We will show a first-fit algorithm to color circular arc containment graphs in $\mathcal{O}(n \log n)$ time.

In a circle of length T, we will refer to a *circular arc* by an interval $[a, b]$ with $0 \leq a < T,\ a < b < a + T$. A circular arc $[a_1, b_1]$ is contained in a circular arc $[a_2, b_2]$ if

(1) $a_2 < a_1$ and $b_1 < b_2$, that is the normal case, or if
(2) $b_2 > b_1 + T$, that could happen when $[a_2, b_2]$ includes T.

In the latter case, that we call special inclusion, $a_2 < a_1$ does not hold. In fact $b_2 > b_1 + T$ implies $a_1 < b_2 - T$, but $a_2 > b_2 - T$, otherwise $[a_2, b_2]$ is longer than T. 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 [16].

We order the arcs in V as follows: $[a_i, b_i] < [a_j, b_j]$ iff $b_i < b_j$. Clearly, this induces a transitive orientation on G and then we can apply the first-fit al-

gorithm, which usually requires $O(n^2)$ time to reach the optimal solution. To obtain the required running time, we apply the following algorithm.

Input: The set V of vertices of a CACG G
Output: A coloring of V
1. Let $\textsc{color}(i) = \phi$ be the list of vertices colored i.
   Let $\mathrm{A}(i) = 0$ for each color i.
   Let $\mathrm{B}(i) = T$ for each color i.
2. Order the elements of V s.t. $[a_i, b_i] < [a_j, b_j]$ iff $b_i < b_j$
3. **for** $j = 1, \ldots, n$
4. Find the smallest k such that $b_j < \mathrm{B}(k) + T$
5. Find the smallest $i \geq k$ such that $a_j > \mathrm{A}(i)$
6. Append $[a_j, b_j]$ to $\textsc{color}(i)$
7. Set $\mathrm{A}(i) = a_j$
8. **if** $\textsc{color}(i)$ contains only $[a_j, b_j]$
   Set $\mathrm{B}(i) = b_j$

After the initialization of the data structures (Step 1), the algorithm orders the arcs in V (Step 2), and starts a cycle to visit all of them (Step 3). Note that, if $[a_i, b_i]$ is contained in $[a_j, b_j]$, $[a_i, b_i]$ is visited before $[a_j, b_j]$, since $b_i < b_j$. At Steps 4–6, the algorithm finds a color for $[a_j, b_j]$. To this end, it first searches for the smallest color not yet assigned to specially contained arcs (Step 4), then, among the remaining colors, it searches for the smallest one not yet assigned to normally contained arcs (Step 5). The remaining steps are used to update the vectors A and B, maintained to speed up the coloring phase of an arc. For an assigned color i, $\mathrm{A}(i)$ contains the value a_j of the last arc colored i, whereas $\mathrm{B}(i)$ contains the values b_j of the first arc colored i.

Lemma 7 *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 and visited according to a transitive orientation. Let $[a_j, b_j]$ be an arc and let k, i be the integers chosen in Steps 4 and 5. We have to show that i is the first color that was not used to color an arc contained in $[a_j, b_j]$. First note that for $\ell < k$ the first arc colored ℓ is specially contained in $[a_j, b_j]$ and that for $k \leq \ell < i$ the last arc sofar colored ℓ is normally contained in $[a_j, b_j]$. Let $[a, b]$ be colored i.

(1) By construction of B (Step 8) and by the order in which the arcs are visited, B is monotonously increasing (i.e. $\mathrm{B}(i) < \mathrm{B}(j)$ for $i < j$) and $b \geq \mathrm{B}(i)$. Hence, $b_j < \mathrm{B}(k) + T < \mathrm{B}(i) + T \leq b + T$. Hence, $[a, b]$ is not

specially contained in $[a_j, b_j]$.

(2) Since at each repetition of the cycle at Step 3 in which color i is assigned to the current arc the values assumed by the variable $\mathrm{A}(i)$ are increasing, $a \leq \mathrm{A}(i)$. It follows $a_j > \mathrm{A}(i) \geq a$. Hence, $[a, b]$ is not normally contained in $[a_j, b_j]$.

For the running time recall that B is monotonously increasing. Further, the integer k chosen at Step 4 is monotonously increasing with each step and, hence, A is monotonously decreasing for $i \geq k$, i.e. $\mathrm{A}(i) \geq \mathrm{A}(j)$ for $k \leq i < j$. Hence, Steps 4 and 5 can both be performed in logarithmic time by binary search. Then the cycle at Step 3 can be performed in $O(n \log n)$ time, as well as the ordering at Step 2. It follows that the over all running time is in $\mathcal{O}(n \log n)$. □

5 Conclusions and Open Problems

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

- Assuming a maximum number of trains per track, the LTA problem with the midnight assumption or without turning back trains is equivalent to the coloring of a permutation graph with a limit on the number of vertices having the same color. Let m be this number. The resulting coloring problem is $\mathcal{NP}$-complete when $m \geq 6$ [12]. It is easily solvable when $m = 2$, being equivalent to a maximum matching problem. Note that when $m = 3, 4$, and 5, perhaps the most interesting cases for the LTA problem applied to tram depots, the computational complexity remains open.
- How bad can the first-fit algorithm behave for solving the linear track 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 [15].
- Is the CTA problem with midnight constraint $\mathcal{NP}$-complete?
- A solution for the cyclic track assignment problem in which trains from a series do not have to be assigned to the same track could use less tracks 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] S. Cornelsen and G. Di Stefano. Platform assignment. In A. Schöbel and F. H. Wagner, editors, *Proceedings of the 4th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2004)*, Lecture Notes in Computer Science. Springer, 2004. To appear.

[4] E. Dahlhaus, P. Horak, M. Miller, and J. F. Ryan. The train marshalling problem. *Discrete Applied Mathematics*, 103(1–3):41–54, 2000.

[5] G. Di Stefano and M. L. Koci. A graph theoretical approach to the shunting problem. In B. Gerards, editor, *Proceedings of the 3rd Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS 2003)*, volume 92 of *Electronic Notes in Theoretical Computer Science*, 2004.

[6] 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.

[7] G. Gallo and F. Di Miele. Dispatching buses in parking depots. *Transportation Science*, 35(3):322–330, 2001.

[8] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Computer Science and Applied Mathematics. Academic Press, 1980.

[9] M. Hamdouni, G. Desaulniers, O. Marcotte, F. Soumis, and M. van Putten. Dispatching buses in a depot using block patterns. Technical Report G-2004-51, Les Cahiers du GERAD, 2005. `http://www.gerad.ca`.

[10] M. Hamdouni, G. Desaulniers, and F. Soumis. Parking buses in a depot using block patterns: A benders decomposition approach for minimizing type mismatches. Technical Report G-2005-70, Les Cahiers du GERAD, 2005. `http://www.gerad.ca`.

[11] 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.

[12] K. Jansen. The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation*, 180:71–81, 2003.

[13] C. Liebchen. Fahrplanoptimierung im Personenverkehr – muss es immer ITF sein? *Eisenbahntechnische Rundschau*, 54(H. 11):689–702, November 2005.

[14] R. D. Lou and M. Sarrafzadeh. Circular permutation graph family with applications. *Discrete Applied Mathematics*, 40:433–457, 1992.

[15] 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.

[16] 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`.

[17] A. Pnueli, A. Lempel, and S. Even. Transitive orientation of graphs and identification of permutation graphs. *Canadian Journal of Mathematics*, 23:160–175, 1971.

[18] A. Rossi. Il problema dell'ordinamento dei treni in un deposito: modellazione e soluzione algoritmica. Master's thesis, Università dell'Aquila, 2003.

[19] W. Unger. On the k-colouring of circle-graphs. In R. Cori and M. Wirsing, editors, *Proceedings of the 5th Symposium on Theoretical Aspects of Computer Science (STACS '88)*, volume 294 of *Lecture Notes in Computer Science*, pages 61–72. Springer, 1988.

[20] T. Winter and U. T. Zimmermann. Real-time dispatch of trams in storage yards. *Annals of Operations Research*, 96:287–315, 2000.

[21] P. J. Zwaneveld, S. Dauzère-Pérès, S. P. M. van Hoesel, L. G. Kroon, H. E. Romeijn, M. Salomon, and H. W. Ambergen. Routing trains through railway stations: model formulation and algorithms. *Transportation Science*, 30:181–194, 1996.