Submitted to *Operations Research*
manuscript OPRE-2020-08-535.R2

Easy cases of deadlock detection in train scheduling

Veronica Dal Sasso
OptRail, Rome, Italy
veronica.dalsasso@optrail.com

Leonardo Lamorgese
OptRail, Rome, Italy
leonardo.lamorgese@optrail.com

Carlo Mannino
SINTEF, Oslo, Norway and University of Oslo, Norway
carlo.mannino@sintef.no

Antonio Tancredi
OptRail, Rome, Italy
antonio.tancredi@optrail.com

Paolo Ventura
Institute of System Analysis and Informatics (IASI) of CNR, Rome, Italy
paolo.ventura@iasi.cnr.it

A deadlock occurs when two or more trains are preventing each other from moving forward by occupying the required tracks. Deadlocks are rare but pernicious events in railroad operations and in most cases are caused by human errors. Recovering is a time-consuming and costly operation, producing large delays and often requiring crew rescheduling and complex switching moves. In practice, most deadlocks involve only two long trains missing their last potential meet location. In this paper we prove that, for any network configuration, the identification of two-train deadlocks can be performed in polynomial time. This is the first exact polynomial algorithm for such a practically relevant combinatorial problem. We also develop a pseudo-polynomial but efficient oracle which allows real-time early detection and prevention of any (potential) two-train deadlock in the Union Pacific (a U.S. class 1 rail company) railroad network. A deadlock prevention module based on the work in this paper will be put in place at Union Pacific to prevent all deadlocks of this kind.

Key words: railways, deadlocks, train scheduling

1. Introduction

Rail traffic has shown a steady increase over the last decade, and recent studies predict continued growth in the next years. Indeed, before the COVID-19 pandemic, the global rail supply market was projected to grow at an annual rate of 2.7% over the next few years (UNIFE (2018)). Also, the freight hauling segment, particularly important in such Countries as the United States, China and Russia, was predicted to be the fastest growing segment of the railway system market (M&Ms (2019)). Investments in rail infrastructure and capacity, while also on the rise, are not expected to meet this surge in demand, which will inevitably lead to increased congestion. Rail traffic controllers, also known as *dispatchers*, will experience a mounting pressure, both in workload and complexity of the tasks. At the same time, there is growing awareness among rail companies of the potential of planning technology, and the opportunity it presents in terms of increasing efficiency. Such context sets the stage for a large-scale adoption of "intelligent" decision support tools in the rail industry, which still lags relatively behind compared to other transport modes and industries (see Lamorgese et al. (2018)).

In this paper we focus on a particular problem that arises in rail traffic management, namely the detection and avoidance of *deadlocks* (Pachl (2011)). A group of trains are *in deadlock* if no train can move because another train is blocking the next track along its route. More in general, we say that a group of trains are *bound-to-deadlock* if every combination of feasible trajectories for the involved trains ends up in a deadlock (Arbib et al. (1990)). For an example, see Figure 1. On the right, the trains are blocking each other's next movements (i.e. a deadlock). On the left, the trains are bound-to-deadlock because any subsequent routing choice will lead to a deadlock.

Releasing trains from a deadlock typically requires complex switching moves such as pulling back trains, and, once a deadlock is in place, it may take many hours to resume regular traffic

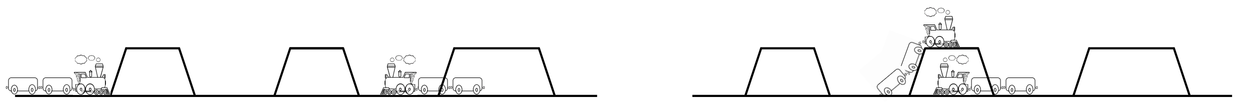


Figure 1 Trains bound-to-deadlock (left) and in a deadlock (right)

flow. This can trigger a number of unanticipated costs, such as crew overtime, re-crewing costs, transporting the crews to/from the location, train fuel, and costs associated with loss of locomotive productivity. Furthermore, there is a risk of missing contractually defined deadlines, a damaging event both financially and in terms of customer satisfaction, and thus taken very seriously by the railroads. The severity of such events is further compounded in parts of the network that cannot be routed around by other incoming trains, which propagates the impact of the deadlock. In other words, a bound-to-deadlock situation is a highly expensive and operationally challenging event to recover from for a railroad, one that must be avoided at all costs. Furthermore, detecting these situations as early as possible is critical for two key reasons. The first is that there is generally some time between when the dispatcher lines a blocking path and when the train actually occupies it. This means that if caught soon enough, the command could still be reversed, thus avoiding the deadlock. The second is that, even should the deadlock be inevitable, trains should in any case be stopped as soon as possible to minimize the cost and complexity of recovering from the deadlock.

It follows from this discussion that preventing deadlocks is arguably the first and foremost responsibility of a dispatcher. This is particularly the case for dispatchers operating predominantly freight traffic (as in North America), which presents a number of specific challenges (see also Pachl (2011)):

- Freight trains are typically ”overlength”, that is, longer than the majority of railroad resources. This largely restricts the number of locations for two such trains to meet, because long stretches of the railroad are single-track and trains can meet and pass each other only when sufficiently long, parallel tracks are available (a current trend in the North American freight market, *precision railroading* is expected to further significantly increase the average size and number of trains operated under these conditions). This means that a meet/hold decision may have to be taken many hours before the trains are projected to meet.

- The goal in operating freight is to transport goods from origin to destination in the shortest time possible, responding to constant changes in supply and demand. This leads to a fairly dynamic planning environment, which lacks the structure and regularity that is instead a feature of passenger services. Indeed, many operators run an "unscheduled" railroad, where typically train composition is known only on the short-term (a day to a week beforehand) and timetables are defined only at a high-level (no conflict resolution). This leaves much of the routing and scheduling decisions in the hands of the dispatchers, which increase their burden compared to dispatchers that operate predominantly passenger services. Indeed, freight traffic conditions may vary significantly from day to day.

In this paper we focus on a specific case of deadlock detection, namely when a deadlock is caused by two trains, which is the most common kind of deadlocks on freight railroads (see Pachl (2011)). This application was introduced to these authors by Union Pacific (UP), a primarily freight-hauling railroad operating 7700 locomotives and thousands of trains over its approximately 52000 km network (covering 23 U.S. states and 7 border crossings). At 37000 employees, over 10000 customers, $34 billion USD capital expenditure over the last decade, UP is by all measures one of the largest transport companies in the world. The information and data provided by UP was essential to study the problem and to set up meaningful computational experiments. Finally, the algorithm presented in this paper will be embedded in a deadlock detection and avoidance tool soon to be deployed at Union Pacific. To the best of our knowledge, UP is the first railroad to systematically approach this serious challenge, that all railroads share, by supporting the development of deadlock detection and avoidance tools based on advanced mathematical techniques. Deploying such technology will lead to eliminating the risk of deadlocks of the kind described in this paper, generating large value for UP's customers and shareholders alike.

1.1. Background and literature.

A railroad can be represented as a network of connected tracks and the trajectory of a train from its origin to destination is an ordered sequence of adjacent tracks (the *route*). In general, there exists a very large number of potential routes for each train (their number can grow exponentially with the number of switches). For each train t in the set of controlled trains T, a *plan* specifies the route of t and the time (*schedule*) t enters each track in its route. Note that a train cannot enter a blocked track, i.e. a track already occupied by another train (a track can actually be blocked even if the other train is occupying some nearby tracks). The plan is feasible if this rule is never violated. In the literature, the problem of finding a feasible, possibly optimal plan (according to some cost function) is often referred to as *Train Scheduling* (with or without routing), or *Train Dispatching*, and may be solved to produce strategic or tactical timetables (Galli and Stiller (2018)), or in real-time to dispatch trains (see Lamorgese et al. (2018)).

Note that, if we have a feasible plan for a set of trains, then no trains are bound-to-deadlock, and if some trains are bound-to-deadlock then we do not have a feasible plan. Given a railroad network and a set of trains, each with an origin, a destination and a specified length, we define the *Bound-to-Deadlock Train Problem (BDTP)*, as the problem of deciding whether the trains in the set cannot all reach their final destination. This is a decision problem: if the answer is YES, then the trains are bound-to-deadlock, if the answer is NO then a feasible plan exists. The BDTP is NP-complete as it generalizes problem (P4) in Arbib et al. (1990). So there is little hope that a polynomial time algorithm exists for the general case.

In mathematical optimization terms, a YES instance of BDTP corresponds to an infeasible instance of train scheduling (with routing). It follows that, at least in principle, we can solve the BDTP by applying any <u>exact</u> method to the train scheduling problem. Indeed, if the algorithm is given enough time to terminate and it does not find a feasible plan, then the answer to BDTP is YES; otherwise a plan is found and the answer is NO. Note that the train scheduling problem generalizes the job-shop scheduling problem, where jobs are trains and machines are tracks or track segments (see Mascis and Pacciarelli (2002)), which, in turn, is known to be NP-hard (see Garey and Johnson (1979)). So there is no (known) polynomial time algorithm for the train scheduling problem. In practice, especially with infeasible input instances, exact methods may require extremely large computing times to terminate, time which typically grows exponentially with the size of the instance. This is why there is a need to develop ad-hoc, exact methods for the BDTP.

On the other hand, one may still resort to heuristic algorithms for solving the train scheduling problem. If the heuristic algorithm terminates with a plan, then the answer is NO and the trains are obviously not bound-to-deadlock. The drawback is that if the algorithm does not terminate with a plan, we cannot conclude that the trains are bound-to-deadlock. A plan may indeed exist but the algorithm is simply not able to find it (*false positive*). There is a very large body of scientific literature devoted to train scheduling, with several exact models and approaches, and even more heuristic approaches. We refer the reader to recent surveys on the topic, such as Corman and Meng (2015), Lamorgese et al. (2018), Wen et al. (2019), among others. Also note that other transport systems face similar scheduling problems, in particular in maritime traffic control (Lübbecke et al. (2019)), on ground and airborne air traffic control (Kjenstad et al. (2013), Mannino et al. (2021)) and automated guided vehicles (Gawrilow et al. (2008)). Indeed, the mathematical models adopted for such problems share many features with the models for train scheduling. However, to the best of our knowledge, deadlock handling is not specifically addressed in the literature on these other transport systems, nor does it seem there is a special need to do so in practice.

Besides this general literature on train and transport scheduling, there is a small set of papers explicitly devoted to handling deadlocks for train scheduling instances. They may be distinguished in two classes: heuristic approaches for the general case, exact approaches for special cases (our paper belongs to the latter class).

Heuristic approaches are based on exploiting the relationship between the train scheduling problem and BDTP. The basic idea is to find a feasible schedule to prove that the answer to BDTP is NO. Note that if one provides the order in which trains occupy the contended resources, then a feasible schedule can be immediately derived (unless the order contains a cycle).

To this first class of approaches belongs the seminal work of Petersen and Taylor (1982), where a train move (from the current position to the next) is performed only if it leaves the possibility for opposite trains to reach their destination. More recently, Pachl (2011) presents a rule based approach to construct an ordering of trains on the resources. The paper is also interesting because it provides a nice introduction to the problem and a classification of actions related to deadlock handling: *deadlock detection*, which amounts to deciding whether a set of trains are bound-to-deadlock; and *deadlock avoidance*, i.e. never taking routing or scheduling decisions which lead to a bound-to-deadlock situation. Banker's algorithm is a simple, greedy algorithm to construct an ordering of the trains on each resource (Haberman (1969)). The algorithm is so basic that it fails to find feasible schedules even in simple situations where such schedules exists. Several authors developed techniques to extend and enhance the Banker's algorithm: among these, Lu et al. (2004), Cui (2010) and Cui et al. (2017). Finally, Li et al. (2014) present an original heuristic approach to train scheduling in single track lines, based on necessary conditions for deadlocks to occur.

As for the class of exact approaches, Li et al. (2014) present an exact MILP formulation for train scheduling in single track lines (but resorts to the discussed heuristic approach to solve it). The work in Simon et al. (2014) is also devoted to single track lines, with the additional (very special) layout feature that stations have exactly two sidings. The authors present a polynomial time solution algorithm for this special case. In Mazzanti et al. (2014), a model checking approach is presented in which train routes are assumed fixed in advance. Finally, a more general deadlock-related discussion can be found in job-shop scheduling and store-and-forward network literature. In Mascis and Pacciarelli (2002) it was observed that train scheduling - at least when trains are short - reduces to an instance of multi-machine, blocking, no-wait job-shop scheduling (see Pinedo (2012), Queyranne and Schulz (1994)). Arbib et al. (1990) provides a proof that the BDTP is NP-complete for generic transport networks, but can be solved in polynomial time if the network has a tree structure.

1.2. Our contribution

In this paper, we focus on the special case of BDTP between two trains of any length. We consider a general network that can model any track layout, including multiple lines and complex stations. We refer to this problem as the Bound-to-Deadlock 2 Trains Problem (BD2TP). Our key contributions are two:

- Practical: in Section 4 we develop an algorithm which solves the BT2DP in a general railroad network. The algorithm performs very well in all instances based on the UP network. Moreover, it is embedded in a deadlock detection and avoidance tool that will be deployed on the UP network in the near future.
- Theoretical: in Section 3.2 we show that the BT2DP can be solved in polynomial time. The proof is constructive, namely we develop an algorithm for the problem and prove that the algorithm terminates in polynomial time with the input size. To the best of our knowledge, there are very few interesting cases of polynomially solvable deadlock detection problems in job-shop scheduling, and only one devoted to train scheduling (Simon et al. (2014)). Note that the latter, however, requires an ideal network configuration that is very unlikely to occur in practice; moreover, trains must be short (i.e. they can occupy only one railroad resource at the time).

In Section 2 we formally introduce the notation and define the problem, then give some preliminary results. We present our main theoretical contribution in Section 3, where we prove that the BD2TP can be solved in polynomial time. The proof is based on a characterization of bound-to-deadlock situations for two trains (1). The theorem provides also the basis for the pseudo-polynomial algorithm of Section 4 which turned out to be very effective in practice. Computational results are presented in Section 5, showing the efficacy of such algorithm over a set of realistic instances derived from the UP network and traffic, along with some additional information regarding the data and application. Section 6 concludes the paper.

2. Modelling rail network and train movement

We consider trains that run in a railroad network. Trains have different lengths, from head to tail. In practice, the railroad is decomposed in sub-networks called *lines*. Informally, lines can be traversed in two directions by trains, conventionally called *eastbound* and *westbound*. A train running westbound and a train running eastbound can share parts of the lines, always traversing sequences of shared resources (like tracks and stations) in reverse order.

t^E (t^W)	eastbound (westbound) train
H^E (H^W)	signals for eastbound(westbound) trains
B	bifurcation points
$G = (H \cup B, \mathcal{S})$, $H = H^E \cup H^W$	segment graph
$G^E = (H \cup B, \mathcal{S}^E)$ $(G^W = (H \cup B, \mathcal{S}^W))$	eastward (westward)-oriented segment graph
$\delta^-(u)$ $(\delta^+(u))$	incoming (outgoing) arcs of $u \in H \cup B$
$P^X = (\sigma^X_1, \dots, \sigma^X_r)$, $X \in \{E, W\}$	X-bound path on G
$S(P)$ $(S^X(P), X \in \{E, W\})$	set of non-oriented (oriented) segments that belong to P
$\prec_E$ $(\prec_W)$	partial order on the segments of $\mathcal{S}^E$ $(\mathcal{S}^W)$
$\mathcal{P}^X, X \in \{E, W\}$	set of blocking paths associated with $t^X, X \in \{E, W\}$
$P^X_o, X \in \{E, W\}$	path blocked by the train in its initial position
$P^X_d, X \in \{E, W\}$	path blocked by the train at destination
$P^X_1 \to P^X_2$ (in G^X)	P^X_2 is reachable from P^X_1 in G^X
$R^E(P^E, P^W)$	set of blocking paths in $\mathcal{P}^E$ reachable from P^E while P^W is occupied
P^E is P^W-green (-red)	P^W_d is (not) reachable from P^W while P^E is occupied

Table 1 Notation

A schematic example of a portion of rail line is shown in Figure 2. Here, the orange and the blue dots indicate the points where trains can stop, which in turn correspond to signals: an eastbound (westbound) signal faces eastbound (westbound) trains and force them to stop when on red. In particular, the orange dots are the stopping points (denoted by H^E) for the *eastbound* trains T^E (running from left to right in the picture), while the blue dots (H^W) are the stopping points for the *westbound* trains (T^W, that go from right to left). Dots are joined by (pieces of) tracks. Observe that, in rail jargon, the portion of track between two successive signals in a given direction is called *interlocking route*. However, we can do without this term and the associated concept in this paper, also in order to avoid confusion with the standard meaning of "route" in graph theory. Besides the signals $H = H^E \cup H^W$, other relevant points in the rail network are the *bifurcation points* B, where tracks split. Such splitting corresponds to a physical switch which allows a train to select the next track among exactly two tracks, or joins two tracks into a single one.

The movement of a train across the network is identified by the movement of its head. To our purpose we can decompose such a movement into a discrete sequence of elementary movements, or *steps*, from a stopping point to the next stopping point, and we can assume that, at the end of the elementary movement, the head of the train is always at a stopping point.

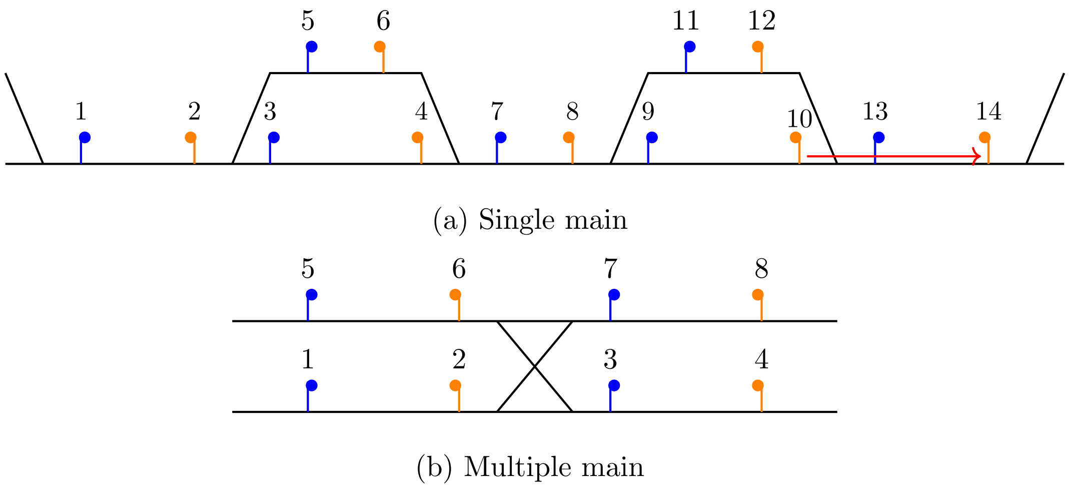
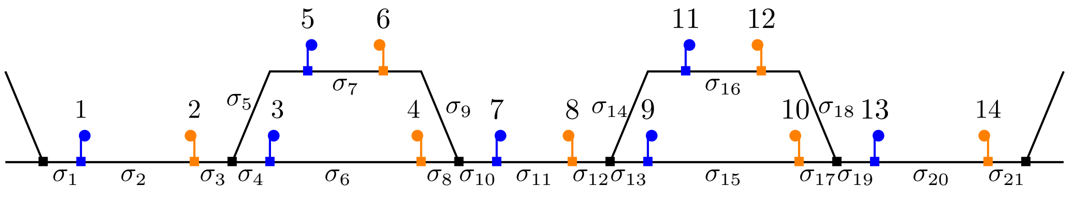
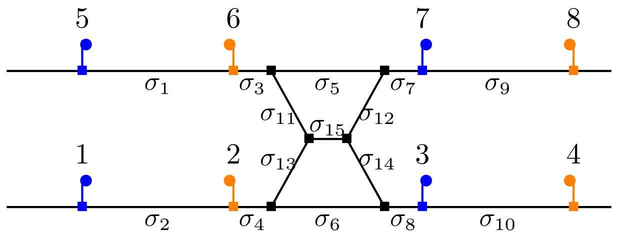


Figure 2 Schematic railway network

The segments graph. In general, trains may use different paths in their movement across the rail network. To represent all potential movements, we introduce the undirected *segment graph* $G = (H \cup B, \mathcal{S})$. Each node corresponds to a relevant point of the railroad, namely it is either a stopping point or a bifurcation point. Each edge of $\mathcal{S}$ corresponds to the segment of track between two adjacent points. Points (nodes of G) and segments (edges of G) of the rail sections of Figure 2 are shown in Figure 3 (nodes are represented as solid squares, black for bifurcations and blue or orange for respectively westward or eastward signals). Because the edges of G correspond to segments, we will use indifferently both terms. Similarly, we may refer to the nodes as "points" or "signals" for the elements of H. Notice that, for the physical switch of Figure 2b, we introduce a fictitious segment σ_{15} to represent the crossing point of the switch tracks. This extra segment has 0-length and will only be used in the proofs.


5
6
11
12
1
2
3
4
7
8
9
10
13
14
σ_1 σ_2 σ_3 σ_4 σ_5 σ_6 σ_7 σ_8 σ_9 σ_{10} σ_{11} σ_{12} σ_{13} σ_{14} σ_{15} σ_{16} σ_{17} σ_{18} σ_{19} σ_{20} σ_{21}

(a) Single main


5
6
7
8
1
2
3
4
σ_1 σ_2 σ_3 σ_4 σ_5 σ_6 σ_7 σ_8 σ_9 σ_{10} σ_{11} σ_{12} σ_{13} σ_{14} σ_{15}

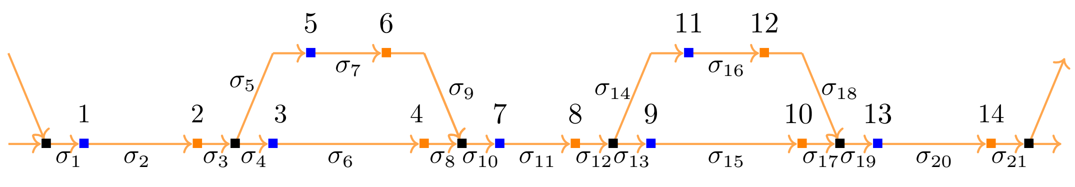
(b) Multiple main

Figure 3 Segmentation of the network in Figure 2. The squares are the extreme points of the segments

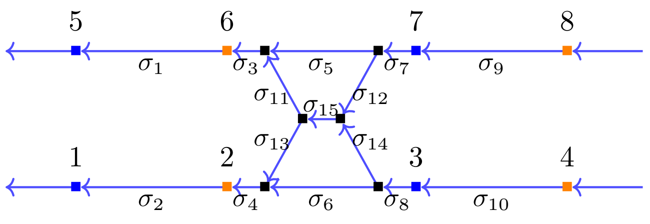
Orienting segments. A rail segment $\sigma \in \mathcal{S}$ can be traversed by trains in both directions. For instance, a train running eastbound will traverse from left to right the segment σ_2, with endpoints signal 1 and signal 2. A train running westbound will traverse σ_2 from right to left. Therefore, we consider two orientations of the undirected graph G, namely the directed graph $G^E = (H \cup B, \mathcal{S}^E)$ where edges are oriented eastbound, and the directed graph $G^W = (H \cup B, \mathcal{S}^W)$ where edges are oriented westbound. So, an undirected edge (or segment) $\sigma = \{p, q\} \in \mathcal{S}$, corresponds to directed edges $\sigma^E = (p, q) \in \mathcal{S}^E$ and $\sigma^W = (q, p) \in \mathcal{S}^W$. An eastbound (westbound) train travelling from its origin to its destination traverses an ordered sequence of segments, which corresponds to a directed path in G^E (in G^W) from the origin to destination. Notice that some particular short sequences may be forbidden for a certain train (for instance the sequence $\sigma_{12} - \sigma_{15} - \sigma_{11}$ for a westbound train in Figure 3b). For sake of simplicity, here we neglect this case, as it can be easily handled by slightly adapting the algorithm described in the sequel.

If $\sigma^X = (u, v) \in S^X$ is a directed segment in S^X for some $X \in \{E, W\}$, we let $head(\sigma^X) = v$ and $tail(\sigma^X) = u$; by $\delta^-(u)(\delta^+(u)) \subseteq S^X$ we denote the incoming (outgoing) star of u, namely the set of segments $\sigma^X \in \mathcal{S}^X$ such that $head(\sigma^X)$ ($tail(\sigma^X)$, resp.) $= u$, for each $u \in H \cup B$.

A path in G^E (G^W) is called an E-bound (W-bound) path. If $P^X = (\sigma_1^X, \ldots, \sigma_r^X)$ is a X-bound directed path, we let $Head(P^X) = \sigma_r^X$ be its *head* segment and $Tail(P^X) = \sigma_1^X$ be its tail segment.



(a) Single main, eastward orientation



(b) Multiple main, westward orientation

Figure 4 The eastward/westward orientation of the segment graphs of Figure 3

In the rail networks we consider, trains will never travel back or cycle, and we may assume the following:

Assumption 1. The graphs G^E and G^W do not contain directed cycles.

A directed graph with no directed cycles (as G^E and G^W) is also called a *DAG*. Moreover, we can also take for granted the following assumption, as it descends directly from the way segments are connected to each other in the rail network. In particular, recall that physical switches either join two entering tracks into a leaving one, or split one entering track into two leaving ones.

Assumption 2. The graph G^X is such that $\min\{|\delta^+(u)|, |\delta^-(u)|\} \leq 1$ and $\max\{|\delta^+(u)|, |\delta^-(u)|\} \leq 2$, for all $u \in H \cup B$ and $X \in \{E, W\}$.

As a consequence of Assumption 1, G^E induces a partial order $\prec_E$ on the segments of $\mathcal{S}^E$. With some abuse of notation, we can extend such an order to $\mathcal{S}$. In particular, for $\sigma, \theta \in \mathcal{S}$, we let $\sigma \prec_E \theta$ if and only if there is a directed path from σ^E to θ^E in G^E (with σ^E and θ^E being the oriented segments corresponding to σ and θ, resp.). Similarly, also G^W induces a partial order $\prec_W$ on the segments $\mathcal{S}$.

The next observation corresponds to the fact that two trains in opposite directions will traverse any pair of shared segments in reverse order.

Observation 1. Let $\sigma, \tau \in \mathcal{S}$ be two segments, and let $\sigma^E, \tau^E \in \mathcal{S}^E$, $\sigma^W, \tau^W \in \mathcal{S}^W$ be the corresponding directed segments in the eastbound and westbound orientation, respectively. Then, $\sigma^E \prec_E \tau^E$ implies $\tau^W \prec_W \sigma^W$.

Finally, observe that any potential physical path across the rail network of an eastbound (westbound) train corresponds to a directed path in G^E (G^W). In practice, when considering a specific train t, say eastbound, not all existing physical paths from the origin to the destination of t are actually feasible for t. Indeed, some segments may be forbidden for various reasons, or train t may be required to pass through specific points (called *way-points*, or platforms in stations, etc.). So, the graph representing the feasible potential movements of t will in general be a subgraph G_t^E of G^E. Also, since we are interested only in segments contained in some feasible paths from origin to destination, G_t^E is a single-source, single-sink acyclic digraph. In the sequel, for sake of simplicity, we will assume that $G_t^E = G^E$ ($G_t^W = G^W$).

We are now ready to define the Bound-to-Deadlock 2 Trains problem. We will look at cases in which only two trains are present in the network. Because two trains running in the same direction (*trailing trains*) cannot end up in a deadlock, we will focus only on two trains running in opposite directions (*crossing trains*). In short, we want to determine whether a pair of opposite trains t^E and t^W are bound-to-deadlock or not.

2.1. Blocked segments and blocking paths

As mentioned, we can ideally decompose the movement of a train $t^E \in T^E$ in a discrete sequence of elementary steps. At each step, t^E moves from the current signal until its head reaches the next signal along its trajectory. Suppose that, at some point in time, t^E has its head in a segment $\sigma_n^E = (q_n^E, p_n^E) \in \mathcal{S}^E$, where $p_n^E \in H^E$ is an eastbound signal. Let $P = (\sigma_1^E, \ldots, \sigma_l^E, \ldots, \sigma_n^E)$ be the path travelled by t^E from the origin to p_n^E, where σ_l^E is the segment containing the tail of t^E. Because of its length, train t^E may occupy several segments from its head to its tail, i.e. we may have $l < n$. The occupied segments $\{\sigma_l^E, \ldots, \sigma_n^E\}$ are blocked for any other train. If another train is approaching, it will have to wait at a previous signal until t^E has released the segments. Moreover, for safety reasons, the set of blocked segments may also include segments which are not physically occupied by train t^E. Namely, all the segments preceding the tail of t^E on the path P backward to the first segment hosting a signal (either eastbound or westbound). For instance, with reference to Figure 5, suppose that segments σ_6, σ_{15} have length 2100 m, σ_4, σ_{13} have length 750 m and σ_3, σ_{12} have length 600 m. Assume that an eastbound train t^1 is less than (or equal to) 2100 m long and has its head at signal 4. Then t^1 occupies only segment σ_6 (in correspondence with the westbound signal 3). On the other hand, if the eastbound train t^2 is 2800 m long and its head is at signal 10, then t^2 blocks the entire path between stopping points 10 and 8, i.e. the three segments σ_{15}, σ_{13}, and σ_{12}.

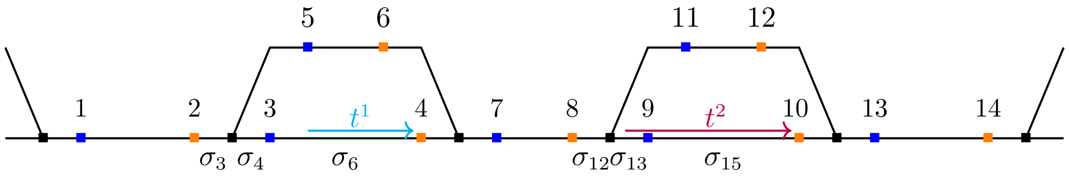


Figure 5 Examples of blocked segments for trains of different length. Train t_1 is short and occupies and blocks only one segment. Train t_2 is longer than the first segment, so it blocks all the segments from its tail backward to the first signal.

So, when the head of t^E is in σ^E_n, and the tail is in σ^E_l, the blocked segments are those belonging to a sub-path $(\sigma^E_i, \dots, \sigma^E_l, \dots, \sigma^E_n)$ of P, with $i \le l \le n$. We call such directed sub-path a *blocking path*. More formally:

Definition 1. For an X-bound train $t \in T^X$, with $X \in \{E, W\}$, a direct path $\bar{P} = (\sigma^X_i, \dots, \sigma^X_l, \dots, \sigma^X_n)$ of G^X, with head segment σ^X_n and tail segment σ^X_i, is a **blocking path** if and only if:

1. the head (node) of the head segment of $\bar{P}$ is an X-bound signal, i.e. $head(\sigma^X_n) \in H^X$.
2. the tail (node) of the tail segment of $\bar{P}$ is a signal, i.e. $tail(\sigma^X_i) \in H$.
3. $\bar{P}$ is long enough to accommodate the length of train t but any proper sub-path of $\bar{P}$ satisfying 1. and 2. is not.

For a train t^X and directed segment graph $G^X = (H \cup B, \mathcal{S}^X)$, with $X \in \{E, W\}$, we let $\mathcal{P}^X$ be the family of distinct blocking paths associated with all possible positions of t^X. In principle, the size of $\mathcal{P}^X$ can grow exponentially with the the number of segments $\mathcal{S}^X$. In real-life instances, however, because the length of the train is limited (typically no more than a dozen segments) and because the degree of the nodes of G^X is small and G^X is sparse, the number of elements of $\mathcal{P}^X$ is reasonably small even for large real-life instances (see Section 5).

Summarizing, at any point in time a train t^X, from head to tail, physically or "logically" occupies all the segments of a certain blocking path, blocking them for other trains: no other train can occupy or block at the same time such blocked segments. The overall movement of the train along its trajectory can be represented as a sequence of blocking paths that move forward. Since a train can stop only in proximity of a signal (indeed, the train MUST stop if the signal is red), the head segment $Head(P^X) = (p_n^X, q_n^X)$ of any blocking path $P^X \in \mathcal{P}^X$ is such that $q_n^X \in H^X$. When performing a step forward a train will travel through one or more new segments, i.e., all segments which separate the current H^X signal from the next H^X signal on the path followed by t^X.

Now, let P_1^X and P_2^X be two paths of G^X. If there is a path $\overline{P}^X$ in G^X that contains both P_1^X and P_2^X as sub-paths and such that $Head(\overline{P}^X) = Head(P_2^X)$ and $Tail(\overline{P}^X) = Tail(P_1^X)$, we say that P_2^X is reachable from P_1^X in G^X and we write $P_1^X \rightarrow P_2^X$ (in G^X). We also say that $\bar{P}^X$ *connects* P_1^X to P_2^X. Note that if P_1^X and P_2^X are blocking paths of t^X and $P_1^X \rightarrow P_2^X$, then t^X can move from P_1^X to P_2^X. For example, in Figure 4a, $P_1^E = (\sigma_2, \sigma_3, \sigma_4) \rightarrow P_2^E = (\sigma_{11}, \sigma_{12}, \sigma_{14})$. Observe that P_1^X and P_2^X may also overlap (again, in Figure 4a, $P_1^E = (\sigma_2, \sigma_3, \sigma_4, \sigma_6, \sigma_9, \sigma_{10}) \rightarrow P_2^E = (\sigma_6, \sigma_9, \sigma_{10}, \sigma_{11}, \sigma_{12}, \sigma_{14})$).

Observe also that the binary relation $\rightarrow$ induces a partial order on the paths of a DAG, with $P_1 \prec P_2$ if and only if $P_1 \rightarrow P_2$. Then, if we consider two elementary paths $P_1 = (\sigma_1)$ and $P_2 = (\sigma_2)$ of G^X, with $\sigma_1, \sigma_2 \in \mathcal{S}^X$, then $P_1 \rightarrow P_2$ if and only if $\sigma_1 \prec_X \sigma_2$.

For $P \in \mathcal{P}^X$, we let $S(P)$ ($S^X(P)$) be the set of non oriented (oriented, resp.) segments that belong to P. Then, observe that, if $P_1^X \rightarrow P_2^X$ in G^X, then no segment in $S(P_2^X)$ can precede all the segments in $S(P_1^X)$.

3. A polynomial time algorithm for BD2TP

In this section we present the main theoretical contribution of the paper, proving that the Bound-to-Deadlock 2 Train Problem can be solved in polynomial time. In order to achieve this result, we first reduce BD2TP to the problem of finding two arc disjoint paths in a suitable graph, and then show that this can be solved with a polynomial time algorithm.

To this end, observe that the movement of a train, for example an eastbound train t^E, from origin to destination can be represented as a sequence $P_o^E, P_1^E, \ldots, P_d^E$ of blocking paths of G^E, where P_o^E is the path blocked by the train in its initial position, whereas P_d^E is the path blocked by the train at destination. Now, when t^E is in a certain blocking path P_j^E of its sequence, it may impede an opposite (westbound) train t^W to access some segments of network. Moreover, if these prohibited segments disconnect (in G^W) the current position P_k^W of t^W from the destination P_d^W of t^W, then t^W cannot reach its destination while t^E holds in P_j^E. In this case, we will say that P_j^E is *red* for P_k^W, otherwise it is *green* for P_k^W (we will give formal definitions in the next section). Clearly, if t^E moves forward from P_j^E, then the situation may change. But it is also possible that, no matter where t^E or t^W move next, the situation does not change and t^W will always be disconnected from its destination. Then the two trains are bound-to-deadlock. Starting from this simple observation, the next discussions will give necessary and sufficient conditions for when the trains are bound-to-deadlock.

3.1. Green and red paths

We consider two opposite trains, $t^E \in T^E$ and $t^W \in T^W$, and let $G^E = (H \cup B, \mathcal{S}^E)$ and $G^W = (H \cup B, \mathcal{S}^W)$.

From now on the roles of trains t^E and t^W can be interchanged. Observe that, when train t^E is in a (position of the network corresponding to a) blocking path $P^E \in \mathcal{P}^E$, then t^E is blocking all the segments in $S(P^E)$, that is all the corresponding westbound oriented segments $S^W(P^E)$ cannot be accessed by the other train t^W. Then, we let $G^W(P^E) = (H \cup B, \mathcal{S}^W \setminus S^W(P^E))$ be the graph obtained from the westbound graph by deleting the blocked edges, that is the segments of P^E.

Denote now by $P_o^E \in \mathcal{P}^E$ and $P_d^E \in \mathcal{P}^E$ ($P_o^W, P_d^W \in \mathcal{P}^W$) the initial position and final destination of train t^E (train t^W, resp.). In particular, let $\sigma_o^E = Head(P_o^E)$ and $\sigma_d^E = Head(P_d^E)$ ($\sigma_o^W = Head(P_o^W)$ and $\sigma_d^W = Head(P_d^W)$) be the segments occupied by the head of t^E (t^W) in its initial and final positions, respectively.

Then, for any $P^E \in \mathcal{P}^E$ and $P^W \in \mathcal{P}^W$, we denote by $R^E(P^E, P^W)$ the set of blocking paths $\bar{P}^E \in \mathcal{P}^E$ such that $P^E \to \bar{P}^E$ in $G^E(P^W)$. Analogously, let $R^W(P^W, P^E)$ be the sets of blocking path that t^W can reach from P^W when t^E is in P^E. The sets $R^E(P_o^E, P_o^W)$ and $R^W(P_o^W, P_o^E)$ play a crucial role, because they represent the set of positions that trains t^E and t^W, respectively, can reach from their origin while the other train holds in its own origin. Since the initial position of each train is not blocked by the other train, we assume that $P_o^E \in R^E(P_o^E, P_o^W)$ and $P_o^W \in R^W(P_o^W, P_o^E)$ (incidentally, observe that the origins can be simultaneously blocked only if the trains are virtually crashing!).

Now, also following the discussion in the introduction of the section, it should be apparent that if two trains are not bound-to-deadlock there must exist two sequences of non blocked paths (one for t^E and one for t^W) which bring the trains from origin to destination. This can be formalized in the next

Definition 2. Two trains t^E, t^W are **not** bound-to-deadlock if, for $\{X, Y\} = \{E, W\}$ there exist two sequences of blocking paths $(P_0^X, ..., P_k^X)$ and $(P_0^Y, ..., P_k^Y)$ such that

- $P_0^X = P_o^X$, $P_0^Y = P_o^Y$, $P_k^X = P_d^X$, and $P_k^Y = P_d^Y$;
- $P_i^X \to P_{i+1}^X$ in $G^X(P_i^Y)$, for each $i = 0, ..., k-1$;
- $P_i^Y \to P_{i+1}^Y$ in $G^Y(P_{i+1}^X)$, for each $i = 0, ..., k-1$.

In other words, the two trains are not bound-to-deadlock if they can both reach their destination through a sequence of feasible alternate moves (at each iteration i, first t^X moves from P_i^X to P_{i+1}^X while t^Y holds in P_i^Y and then t^Y moves from P_i^Y to P_{i+1}^Y while t^X is in P_{i+1}^Y). Observe that could be the case that $P_i^X = P_{i+1}^X$ or $P_i^Y = P_{i+1}^Y$ for some $i = 0, ..., k-1$.

In the rest of this sub-section, we will prove that such a couple of sequences exists if and only if there exists a couple of feasible sequences with $k = 2$.

In order to better explain such result, we need to introduce some further notation. In particular, we say that P^E is P^W-green if train t^W can reach its destination when t^E holds in P^E, i.e. $P^W_d \in R^W(P^W, P^E)$; otherwise we say that P^E is P^W-red. The same for P^W, that is said to be P^E-green if $P^E_d \in R^E(P^E, P^W)$, and P^E-red, otherwise (see Figure 6).

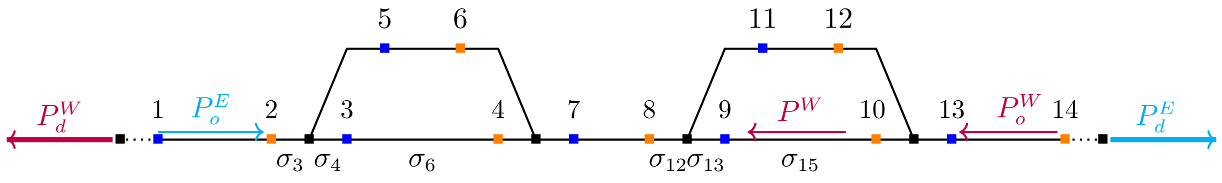


Figure 6 Example of connectivity: P^W_o is P^E_o-red, P^E_o is P^W_o-red, while P^W is P^E_o-green

Note that if P^E_o is P^W_o-green, then we are not bound to deadlock. Indeed, since P^W_d can be reached from P^W_o in $G^W(P^E_o)$, then train t^W can travel to its destination while t^E holds in P^E_o. Similarly when P^W_o is P^E_o-green. Therefore, for a bound to deadlock situation, we need both P^E_o to be P^W_o-red and P^W_o to be P^E_o-red and in the following we will use this assumption. Indeed, we have the following necessary condition for deadlocks, whose proof is straightforward.

Lemma 1. If t^E and t^W are bound to deadlock then all (blocking) paths in $R^E(P^E_o, P^W_o)$ are P^W_o-red and all paths in $R^W(P^W_o, P^E_o)$ are P^E_o-red.

In the following, we will show now that the condition of the above Lemma is also sufficient for two opposite trains to be bound to deadlock. This is good news, because, as we will see in Section 3.2, the condition of the lemma can be tested efficiently.

Suppose the condition of Lemma 1 holds. We already know that if one of the two trains, say t^E, is holding in P^E_o or in any other node in $R^E(P^E_o, P^W_o)$, then t^W cannot reach its destination *from its origin*, because P^W_o cannot reach P^W_d. However, at least in principle, there could be a sequence of moves which "clears" one of the trains. For instance, t^E could move in $P^E_1 \in R^E(P^E_o, P^W_o)$, then t^W could move in $P^W_1 \in R^W(P^W_o, P^E_1)$, then t^E could move in $P^E_2 \in R^E(P^E_1, P^W_1)$, and so forth until the destination becomes now reachable for one of the two trains. We will show that this cannot happen, and the condition of Lemma 1 is also sufficient for the trains to be bound to deadlock. In particular, we will show that when one of the trains move forward from P_o^X to one of its reachable paths P^X, the set of paths which are reachable for the other train can only “shrink”. To this end, we need first a few intermediate results.

Lemma 2. [Shrinking lemma] Let $\bar{P}^E \in \mathcal{P}^E$ and $\bar{P}^W \in \mathcal{P}^W$. Then for all $P^E \in R^E(\bar{P}^E, \bar{P}^W)$ we have that

1. $R^E(P^E, \bar{P}^W) \subseteq R^E(\bar{P}^E, \bar{P}^W)$.

Moreover, if P^E is $\bar{P}^W$-red, then

2. $R^W(\bar{P}^W, P^E) \subseteq R^W(\bar{P}^W, \bar{P}^E)$.

Proof. Claim 1. $\tilde{P}^E \in R^E(P^E, \bar{P}^W)$ implies (i) $P^E \to \tilde{P}^E$ in $G^E(\bar{P}^W)$; since $P^E \in R^E(\bar{P}^E, \bar{P}^W)$ then (ii) $\bar{P}^E \to P^E$ in $G^E(\bar{P}^W)$. Finally (i) and (ii) imply $\bar{P}^E \to \tilde{P}^E$ in $G^E(\bar{P}^W)$, and then $\tilde{P}^E \in R^E(\bar{P}^E, \bar{P}^W)$.

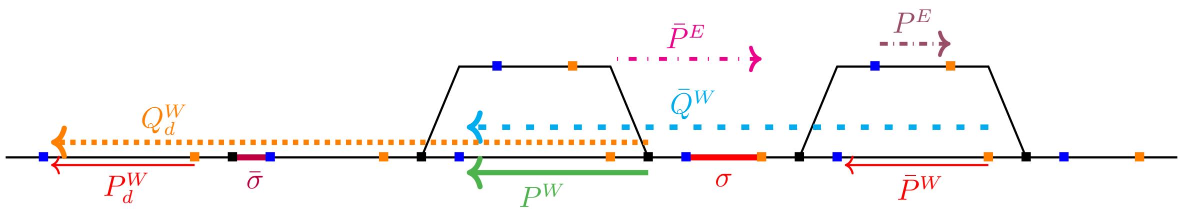


Figure 7 Proof of Lemma 2 (Claim 2.)

Claim 2. (See Figure 7) Suppose, by contradiction, that there exists $P^W \in R^W(\bar{P}^W, P^E)$ such that $P^W \notin R^W(\bar{P}^W, \bar{P}^E)$. Hence $\bar{P}^W \to P^W$ in $G^W(P^E)$ but not in $G^W(\bar{P}^E)$. Then let $\bar{Q}^W$ be a path that connects $\bar{P}^W$ to P^W in $G^W(P^E)$. Then there exists $\sigma \in S(\bar{Q}^W) \cap S(\bar{P}^E)$ (otherwise $\bar{Q}^W$ would connect $\bar{P}^W$ to P^W in $G^W(\bar{P}^E)$, a contradiction). Observe that, as σ is a segment of $\bar{Q}^W$, then $\sigma \notin S(P^E)$. Moreover, there exists a path Q_d^W that connects P^W to P_d^W in G^W (such a path exists because, by construction, the destination P_d^W can be reached from every blocking path in G^W). Since, by hypothesis, P^E is $\bar{P}^W$-red, then $P_d^W \notin R^W(\bar{P}^W, P^E)$. Therefore, there exists $\bar{\sigma} \in S(Q_d^W) \cap S(P^E)$. Note that, since $P^W \in R^W(\bar{P}^W, P^E)$, then $\bar{\sigma} \notin S(P^W)$, and thus $\bar{\sigma}$ strictly follows P^W on Q_d^W. It follows that $\sigma \prec_W \bar{\sigma}$ and therefore $\bar{\sigma} \prec_E \sigma$. But then we have two paths $P^E, \bar{P}^E \in \mathcal{P}^E$ with $\bar{P}^E \to P^E$ in G^E and two segments $\sigma, \bar{\sigma} \in \mathcal{S}$ with $\bar{\sigma} \prec_E \sigma$ such that $\sigma \in S(\bar{P}^E)$ and $\bar{\sigma} \in S(P^E)$. This implies that $\sigma \in S(P^E)$. Contradiction. Q.E.D.

In particular, when $\bar{P}^E = P_o^E$ and $\bar{P}^W = P_o^W$, Lemma 2 shows that, if we move train t^E from P_o^E to any path $P^E \in R^E(P_o^E, P_o^W)$ that is P_o^W-red, the set of paths reachable for train t^W shrinks from $R^W(P_o^W, P_o^E)$ to $R^W(P_o^W, P^E) \subseteq R^W(P_o^W, P_o^E)$. Next, we will prove that the paths in $R^W(P_o^W, P^E)$ are all P_o^E-red with respect to the new position P^E of train t^E. Intuitively, this is all we need to prove, because, after each move, the cardinality of reachable paths will decrease and no green paths will ever appear.

Lemma 3. Suppose all paths in $R^E(P_o^E, P_o^W)$ are P_o^W-red and all paths $R^W(P_o^W, P_o^E)$ are P_o^E-red. Then, if $P_u^E \in R^E(P_o^E, P_o^W)$, we have that

Claim 1. all paths in $R^E(P_u^E, P_o^W)$ are P_o^W-red;

Claim 2. all paths in $R^W(P_o^W, P_u^E)$ are P_u^E-red.

Proof.

Claim 1. Follows from $R^E(P_u^E, P_o^W) \subseteq R^E(P_o^E, P_o^W)$ (Claim 1 of Lemma 2) and the hypothesis that all paths in $R^E(P_o^E, P_o^W)$ are P_o^W-red.

Claim 2. Suppose not and let $P_v^W \in R^W(P_o^W, P_u^E)$ be P_u^E-green. Then there is a path Q_{ud}^E in $G^E(P_v^W)$ that connects P_u^E to P_d^E. Moreover, let Q_{ou}^E be a path that connects P_o^E to P_u^E in G^E (such a path always exists by construction of G^E). Then, there exists $\sigma_1 \in S(Q_{ou}^E) \cap S(P_v^W)$, otherwise t^E could use the path that connects Q_{ou}^E to Q_{ud}^E to go from P_o^E to P_d^E in $G^E(P_v^W)$ (i.e. while t^W holds in P_v^W), so contradicting the assumption that P_v^W is P_o^E-red. Note that $\sigma_1 \notin S(P_u^E)$, since $P_u^E \to P_d^E$ in $G^E(P_v^W)$ (see Figure 8a).

Similarly, let Q_{ov}^W be the path of $G^W(P_o^E)$ that connects P_o^W to P_v^W (recall that $P_v^W \in R^W(P_o^W, P_o^E)$) and let Q_{vd}^W be the path of G^W that connects P_v^W to P_d^W. Then there exists $\sigma_2 \in S(Q_{vd}^W) \cap S(P_u^E)$, otherwise t^W could use the path that connects Q_{ov}^W to Q_{vd}^W to go from P_o^W to P_d^W, while t^E is in P_u^E, contradicting the hypothesis that P_u^E is P_o^W-red. Note that $\sigma_2 \notin S(P_v^W)$, since $P_o^W \to P_v^W$ in $G^W(P_u^E)$ (see Figure 8b).

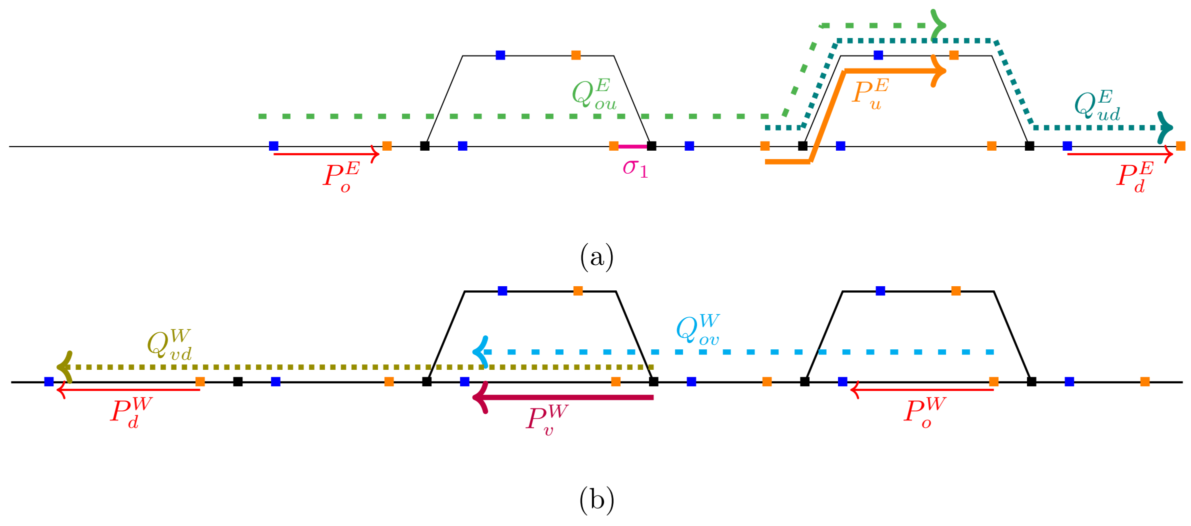


Figure 8 Proof of Lemma 3

Then we have: (i) $\sigma_1 \in S(P^W_v)$ and $\sigma_2 \in S(Q^W_{vd})$, with $P^W_v \to Q^W_{vd}$ in G^W and $\sigma_2 \notin S(P^W_v)$; (ii) $\sigma_1 \in S(Q^E_{ou})$ and $\sigma_2 \in S(P^E_u)$ with $Q^E_{ou} \to P^E_u$ in G^E and $\sigma_1 \notin S(P^E_u)$. Therefore, (i) implies $\sigma_1 \prec_W \sigma_2$ and (ii) implies $\sigma_1 \prec_E \sigma_2$, a contradiction (Observation 1).

Q.E.D.

We are now ready to prove the main result of the section.

Theorem 1. Let trains t^E and t^W have starting positions in $P^E_o \in \mathcal{P}^E$ and $P^W_o \in \mathcal{P}^W$, respectively. Then t^E and t^W are not bound-to-deadlock if and only if there exists at least one path $P^E \in R^E(P^E_o, P^W_o)$ that is P^W_o-green or one path $P^W \in R^W(P^W_o, P^E_o)$ that is P^E_o-green.

Proof. Sufficiency was proven earlier (Lemma 1).

Necessity. Suppose, by contradiction, that all paths in $R^E(P^E_o, P^W_o)$ are P^W_o-red and all paths in $R^W(P^W_o, P^E_o)$ are P^E_o-red, but the trains are not bound-to-deadlock. Then there exists a sequence of alternating moves for t^E and t^W

$$P^E_o, P^W_o, P^E_1, P^W_1, \ldots, P^E_{k+1} = P^E_d, P^W_{k+1} = P^W_d$$

such that $P^E_{i-1} \to P^E_i$ in $G^E(P^W_{i-1})$ and $P^W_{i-1} \to P^W_i$ in $G^W(P^E_i)$ for each i. Observe that, for some $i \in [1, \ldots, k]$, we may have $P^E_{i-1} = P^E_i$ or $P^W_{i-1} = P^W_i$. Let this sequence be defined in such a way that k is the minimum index from which t^E can reach its destination (i.e., $P_d^E \in R^E(P_k^E, P_k^W)$ but $P_d^E \notin R^E(P_i^E, P_i^W)$ for any $i < k$). Hence P_k^W is P_k^E-green.

By Lemma 2, $R^W(P_k^W, P_k^E) \subseteq R^W(P_o^W, P_k^E) \subseteq R^W(P_o^W, P_o^E)$ and, by Lemma 3, each $P^W \in R^W(P_k^W, P_k^E)$ is P_k^E-red. In particular, as $P_k^W \in R^W(P_k^W, P_k^E)$, it follows that P_k^W is P_k^E-red, contradicting the statement above. Q.E.D.

We are now ready to prove that the Bound-to-Deadlock 2 Train Problem can be solved in polynomial time.

3.2. BD2TP can be solved in polynomial time

Because of Theorem 1, the Bound-to-Deadlock 2 Train Problem can be solved by enumerating all possible blocking paths of a train and, for each of them, check if the other train can reach its final destination. Unfortunately, this procedure (although very effective in practice - as explained in detail in the next two sections) is only pseudo-polynomial, as the number of blocking paths can grow exponentially with the length of the train. In the following, we will present a strongly polynomial algorithm for the problem. Indeed, we will show that, given the initial position of the trains, the condition of Theorem 1, can be checked without actually generating the blocking paths. As usual, we let $G = (H \cup B, \mathcal{S})$ be the rail network and $G^E = (H \cup B, \mathcal{S}^E)$, $G^W = (H \cup B, \mathcal{S}^W)$ be the eastbound and westbound orientation of G, respectively. Recall that G^E and G^W are both acyclic (Assumption 1). Then let t^E and t^W be the two opposite trains we are considering and let l_E be the length of train t_E. Moreover, for each $X \in \{E; W\}$, let P_o^X and P_d^X be the origin and destination (respectively) positions of train t^X. Now, G^E and G^W share the same rail resources. A P_o^W-green blocking path P^E for t^E corresponds to a sequence of (one or more) segments of G^E of total length at least l_E which, when occupied/blocked by train t^E, leave a path P^W in G^W allowing train t^W to move from P_o^W to its destination. Note that the two paths P^E and P^W live in different graphs, namely G^E and G^W. Without loss of generality, for each $X \in \{E, W\}$, we assume that G^X contains all and only the feasible trajectories of t^X.

So, if a P_o^W-green path is reachable for t^E given the current position P_o^W of t^W, then the two trains are not bound-to-deadlock. Viceversa, Theorem 1 tells us that if no such green path exists either for t^E of for t^W, then the two trains are bound-to-deadlock. In this section we will show that checking whether such green path exists for any of the two trains can be performed in polynomial time, which in turn implies that problem $BD2TP$ is polynomially solvable. To this end, we will need a technical result (Lemma 4), which adapts to our case a result presented in Li et al. (1992). The lemma allows to check in polynomial time whether there exist two node disjoint paths (with given origins and destinations) in an acyclic directed graph, one of which is at least of a prescribed length.

In our problem, we do seek for two paths, but in different graphs. So, the first step will be to reduce our original problem to the problem of searching (suitable) two disjoint paths in a suitable directed graph.

Constructing auxiliary graph G^{XY}. Now, let P^X be a blocking path for t^X that we want to test if P_o^Y-green for $Y \in \{E, W\}, Y \neq X$. Denote by $l(P^X)$ the length of P^X and by $o(P^X) \in H$ and $d(P^X) \in H^X$ its first and last node. Recall that the last node of a X-bound blocking path (where t^X has its head) is always a X-signal, and thus belongs to H^X, whereas the first node can be any signal in $H = H^X \cup H^Y$, respectively. Then, let o_Y denote the head node of P_o^Y (i.e. $o_Y = o(P_o^Y) = head(Head(P_o^Y)))$ and d_Y the tail node of P_d^Y (i.e. $d_Y = d(P_d^Y) = tail(Tail(P_d^Y)))$. Moreover, let $\overline{G}^Y$ be the X-bound oriented graph obtained from G^Y by reversing the direction of its arcs. Therefore, any o_Y-d_Y in G^Y, corresponds to a d_Y-o_Y path in $\overline{G}^Y$.

Now, let $G^{XY} = (H \cup B, A)$ be the X-bound oriented graph whose set of arcs A contains all the arcs of G^X and $\overline{G}^Y$ (the two sets in general do overlap) minus the arcs that corresponds to the segments of P_o^Y. Then, P^X is a $o(P^X)$-$d(P^X)$ path of G^{XY} of length $l(P^X) \geq l_X$ for some $o(P^X) \in H$ and $d(P^X) \in H^X$. Furthermore, P^X is P_o^Y-green if and only if there exists a d_Y-o_Y path P^Y of G^{XY} such that: i) P^Y uses only arcs from G^Y (in their X-bound orientation); ii) P^X and P^Y are arc disjoint.

Therefore, Theorem 1 can be rephrased as follows.

Corollary 1. Let trains t^E and t^W have starting positions in $P_o^E \in \mathcal{P}^E$ and $P_o^W \in \mathcal{P}^W$, respectively. Then t^E and t^W are not bound-to-deadlock if and only if the graph G^{XY} contains two node disjoint paths P^X and P^Y such that:

- P^X is a o_X-d_X path of length at least l_X, for some $o_X \in H$ and $d_X \in H^X$. It uses only arcs from G^X;
- P^Y is a d_Y-o_Y path that uses only arcs from $\overline{G}^Y$, with $o_Y = head(Head(P_o^Y))$ and $d_Y = tail(Tail(P_d^Y))$;

for some $X, Y \in \{E, W\}$ with $X \neq Y$.

Hence, we reduced BD2TP to the problem of checking the existence of the two paths P^X and P^Y that satisfy the conditions of Corollary 1.

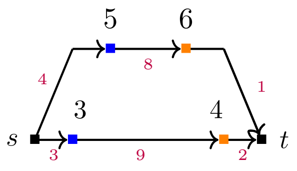
In Li et al. (1992), the authors present an $O(|A|^k)$ algorithm that solves the k arc disjoint path problem with non-uniform costs in an acyclic graph $G = (V, A)$. Here, we adapt their proof of Lemma 8 to get the following result.

Lemma 4. Let $G = (V, A)$ be an acyclic oriented graph and let A_1 and A_2 be two (possibly overlapping) sets of arcs such that $A_1 \cup A_2 = A$. Moreover, let $s, t \in V$, $l_1 : A_1 \to \mathbb{R}_+$ be a cost function on the arcs of A_1 and $L \in \mathbb{R}_+$ a parameter. Then the problem of finding whether there exist two node disjoint s-t paths $P_1 \subseteq A_1$ and $P_2 \subseteq A_2$ such that $l(P_1) = \sum_{\sigma \in P_1} l_1(\sigma) \geq L$, can be solved in polynomial time.

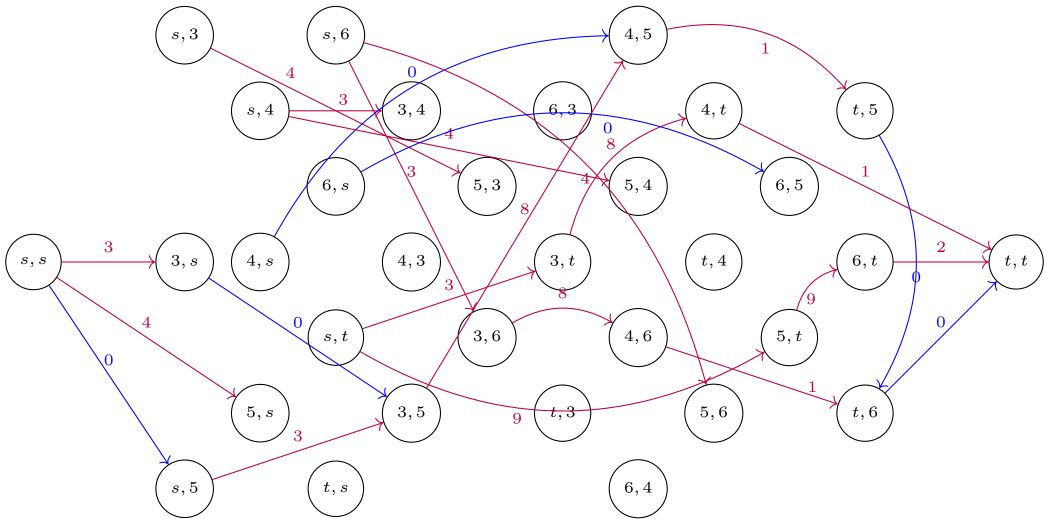
Proof. As the graph G is acyclic, we can assume, w.l.o.g., that its nodes are ordered in such a way that $(u, v) \in A$ implies $u < v$. Then build a 2-dimensional graph $\bar{G} = (\bar{V}, \bar{A})$ as follows.

$$\begin{aligned}
\bar{V} = &\quad \{(u,v) | u, v \in V \text{ and } (u \neq v \text{ if } u, v \notin \{s,t\})\} \\
\bar{A} = \bar{A}_1 \cup \bar{A}_2, &\quad \text{with } \{\bar{A}_1 = ((u,v),(w,v)) | (u,w) \in A_1 \text{ and } u \leq v\} \\
&\quad \text{and } \bar{A}_2 = \{((u,v),(u,w)) | (v,w) \in A_2 \text{ and } v \leq u\}
\end{aligned}$$

Notice that both (s, s) and (t, t) belong to $\bar{V}$. Moreover, since $(u, v) \in A$ implies $u < v$, then the arcs of $\bar{A}$ define a partial order on the nodes of $\bar{V}$. Hence, also $\bar{G}$ is acyclic, $(x, y) \in \bar{A}$ implies $x < y$.



(a) $(s-t)$-Network



(b) 2-dimensional network

Figure 9 Example of how to build the 2-dim network

An arc of type $((u,v),(w,v))$ (resp. $((u,v),(u,w))$) is called a 1st-(2nd-)dimensional arc. We give the cost $c_a = l_1(u,w)$ to any arc $a = ((u,v),(w,v)) \in \bar{A}_1$, while we set $c_a = 0$ for all 2nd-dimensional arcs $a \in \bar{A}_2$. For an example of how these 2-dimensional networks look, see Figure 9.

We now show that G admits two s-t node-disjoint paths $P_1 \subseteq A_1$ and $P_2 \subseteq A_2$ if and only if $\bar{G}$ contains an (s,s)-(t,t) path $\bar{P}$ of cost $l(P_1)$.

Necessity Suppose $\bar{G}$ admits an (s,s)-(t,t) path $\bar{P}$ and let $(u^1,v^1),\dots,(u^p,v^p)$ be the sequence of nodes defined by $\bar{P}$, with $u^1 = v^1 = s$ and $u^p = v^p = t$. Then there is an s-t path in G which uses only arcs of A_1. Indeed, suppose $\bar{P}$ contains at least one arc of $\bar{A}_1$.

Now, the sequences $\{s = u^1, u^2, \dots, u^p = t\}$ and $\{s = v^1, v^2, \dots, v^p\}$ (with possible repeated nodes skipped) define two distinct s-t paths P_1 and P_2 of G, P_1 only using arcs in A_1 and P_2 only in A_2. Node indices in both sequences are non-decreasing. Moreover, for $i = 2,\dots,p-1$, if $u^i < v^i$, then $((u^i,v^i),(u^{i+1},v^{i+1})) \in \bar{A}_1$, $v^i = v^{i+1}$ and $(u^i,u^{i+1}) \in A_1$ (informally, P_1 moves forward to u^{i+1}

while P_2 stays in $v^{i+1} = v^i$). Otherwise, if $v^i < u^i$, then $((u^i, v^i), (u^{i+1}, v^{i+1})) \in \bar{A}_2$, $u^i = u^{i+1}$ and $(v^i, v^{i+1}) \in A_2$ (i.e. P_2 stretches to v^{i+1} while P_1 remains in $u^{i+1} = u^i$). Observe that $u^i = v^i$ implies $u^i = v^i = s$ or $u^i = v^i = t$; in the first case $((u^i, v^i), (u^{i+1}, v^{i+1}))$ can be either in $\bar{A}_1$ or in $\bar{A}_2$; in the second case, $i = p$ and we are done.

We show now that P_1 and P_2 are node disjoint (except for s and t). Suppose not. As $u^i = v^i$ only for $i = 1$ or $i = p$, then there must be two distinct nodes $(u^i, v^i), (u^j, v^j)$ of $\bar{P}$, with $1 < i < j < p$, such that $u^i = v^j$ or $v^i = u^j$. Suppose $u^i = v^j$ (a symmetric argument holds for the other case). Since $u^j \neq u^i$ (otherwise $u^j = u^i = v^j$), in the subpath $\bar{P}^{ij}$ of $\bar{P}$ from (u^i, v^i) to (u^j, v^j) we move away from u^i and thus $\bar{P}^{ij}$ contains (at least) an arc from $\bar{A}_1$. It follows that there is an index $i \leq k < j$ such that $u^k < v^k$ and thus $u^i \leq u^k < v^k \leq v^j = u^i$, a contradiction.

Sufficiency. Let $P_1 = (a_1^1, \dots, a_1^k)$, $P_2 = (a_2^1, \dots, a_2^h)$ be two node-disjoint s-t paths of G. We arrange the arcs of these paths in ascending order of their tails. Let this sequence be $\sigma^1, \sigma^2, \sigma^3, \dots, \sigma^d$, with $d = k + h$. As $s = tail(a_1^1) = tail(a_2^1)$, we can assume, w.l.o.g. that $\sigma^1 = a_1^1$ and $\sigma^2 = a_2^1$. Now, let $\bar{P} = (\alpha^1, \dots, \alpha^d)$ be defined as follows. First, set $\alpha^1 = ((s, s), (head(\sigma^1), s))$ and $\alpha^2 = ((head(\sigma^1), s), (head(\sigma^1), head(\sigma^2)))$. For each $i = 3, ..., d$, let $\sigma^{i-1} = (u, v)$. Then, $\sigma^i \in P_1$ implies $u < v$; in this case, set $\alpha^i = ((u, v), (head(\sigma^i), v))$. Otherwise, if $\sigma^i \in P_2$, then set $\alpha^i = ((u, v), (u, head(\sigma^i)))$. It is easy to check that all α^i are arcs of $\bar{G}$ and that $\bar{P}$ is an (s, s)-$(t - t)$ path of cost $l(P_1)$.

As a consequence, in order to solve our problem, it suffices to answer whether the longest path on $\bar{G}$ from (s, s) to (t, t) has cost not smaller than L. The computational complexity of finding the longest path on an acyclic graph is $O(|\bar{A}|)$ (Ahujia et al. (1993)). Notice that $|\bar{A}| \leq 2|V||A| \leq 2|H \cup B||\mathcal{S}|$, hence the complexity of our problem is polynomial and it is $O(|H \cup B||\mathcal{S}|)$. Q.E.D.

Finally, we can prove our main result.

Theorem 2. The Bound-to-Deadlock 2 Trains Problem can be solved in polynomial time.

Proof. For both the cases $X, Y \in \{E, W\}$ with $X \neq Y$, we can verify the conditions of Corollary 1 by iteratively applying Lemma 4 for all possible $o_X \in H$ and $d_X \in H^X$. Let A_X and A_Y denote the set of arcs of G^X and $\overline{G}^Y$, respectively. Then, at each iteration, we set $G = (V, A)$, A_1, A_2, l_1 and L as follows

- $V = H \cup B \cup \{s, t\}$;
- $A_1 = A_X \cup \{(s, o_X), (d_X, t)\}$ and $A_2 = A_Y \cup \{(s, head(Head(P_o^Y))), (tail(Tail(P_d^Y)), t)\}$;
- $A = A_1 \cup A_2$;
- $l_1(a) = l(a)$, for all $a \in A_X$; $l_1(s, o_X) = l_1(d_X, t) = 0$;
- $L = l_X$.

It is important here to observe that Lemma 4 can be properly applied because, as G^{XY} is a subgraph of G^X, then G is acyclic. As the number of $s_1 - t_1$ pairs is bounded by $|H \cup B|(|H \cup B| - 1)/2$, we have a total complexity of $O(|H \cup B|^3 ||\mathcal{S}|)$. Q.E.D.

4. The Path Coloring algorithm (PCA) for BD2TP

In the previous section we have shown that $BD2TP$ is solvable in polynomial time. However, the computational complexity grows as the cubic of the cardinality of nodes times the number of segments of the rail network, so, roughly speaking, as the number of nodes to the power of 4 (because in our representation rail networks are sparse). Based on the results of Section 3.1, we give here an alternative algorithm, which we named Path Coloring algorithm (PCA), whose complexity is proportional to the number of blocking paths in the eastbound and westbound orientations of G. Even if in principle this leads to a potentially exponential time algorithm, in practice, because rail networks are sparse and trains have bounded (and small in terms of number of segments) lengths, the resulting algorithm behaves very well on the real-life instances (see Section 5).

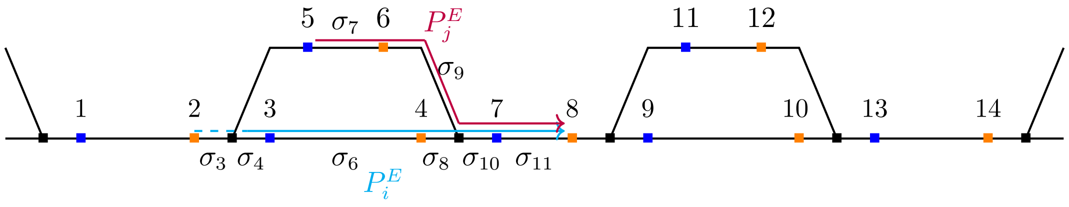


Figure 10 Different blocking paths for t with same head signal.

Building the blocking paths. Given an eastbound train t^E and the orientation $G^E = (H \cup B, \mathcal{S}^E)$, we give a recursive algorithm to construct all the blocking paths of $\mathcal{P}^E$ (the westbound case is symmetric). Some examples of how these paths are built can be seen in Figure 10 and Figure 11. In particular, Figure 10 shows two different blocking paths P_i^E and P_j^E which correspond to two sequences of segments ending in the same signal. Figure 11 shows how different train lengths induce different blocking paths. Notice that, in the second example, segment σ_{11} is enough to contain the whole train, hence the blocking path P_2^E ends with a blue (westbound) signal. In the third example, instead, the train fits into σ_{10} and σ_{11} but, as σ_{10} does not end with a signal, we extend the blocking path to the previous signal (σ_8). The forth example shows that two successive blocking paths may overlap. Notice that, in all four cases, the path P^E from point 2 to point 8 connects P_1^E to P_2^E.

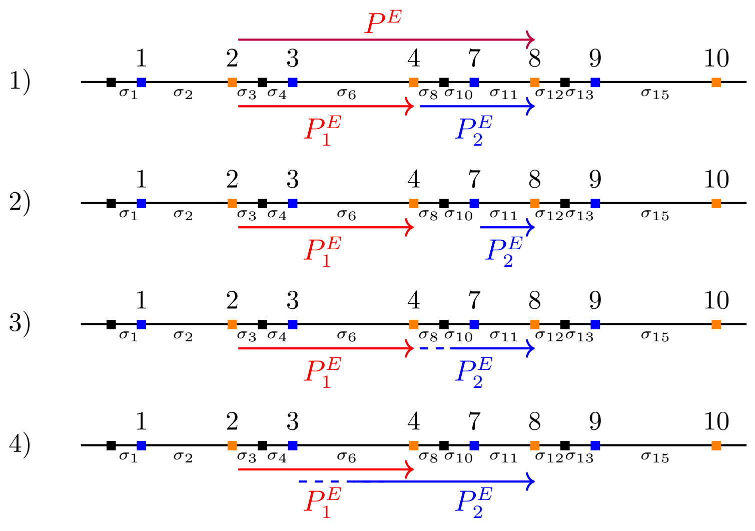


Figure 11 In all the four cases, $P_1^E \to P_2^E$ and P^E connects P_1^E to P_2^E.

Blocking paths are constructed recursively by Procedure Extend described in Algorithm 1. According to Definition 1, the head segment σ^1 of a blocking path $(\sigma^1, \sigma^2, \dots)$ for an eastbound train must host an eastbound signal at its head point. So, for every segment σ such that $head(\sigma) \in H^E$, we build all paths having σ as head segment, and satisfying the other two conditions of Definition 1.

The recursive procedure Extend(P, N_σ) receives in input a non-empty path P, satisfying condition 1 of Definition 1 (but not necessarily also conditions 2 and 3) and extends it (backwards) to all blocking paths having P as a head sub-path. As usual, if σ is a directed segment and P a directed path, with $head(\sigma) = tail(Tail(P))$, we let $\sigma \circ P$ be the path obtained concatenating σ with P.

Algorithm 1 Extends a path P backward to all blocking paths with P as head sub-path

```
1: procedure EXTEND($P, N_\sigma$)
2:     if ($P$ enough for $t^E$) AND ($tail(Tail(P)) \in H$) then        ▷ $P$ fulfills all conditions 1
3:         $N_\sigma = N_\sigma \cup \{P\}$                                  ▷ i.e. $P$ is a blocking path
4:     else
5:         for $\gamma \in \delta^-(Tail(P))$ do
6:             Extend ($\gamma \circ P, N_\sigma$)                 ▷ Extends $P$ backwards in all possible ways
7:         end for
8:     end if
9:     return
10: end procedure
```

Procedure Extend is initially invoked (first recursive call) for every segment $\sigma_P \in \mathcal{S}^E$ hosting an eastbound signal, by initializing $P = (\sigma_P)$ and $N_\sigma = \emptyset$. Eventually, $\mathcal{P}^E = \cup_{\sigma \in \mathcal{S}^E} N_\sigma$. As mentioned in Section 2, some paths in the graph satisfying the conditions of Definition 1 may not correspond to available blocking paths. It is straightforward to modify Procedure Extend and avoid returning such paths.

Coloring the blocking paths. Once the blocking paths are built for both t^E and t^W, with initial positions respectively P_o^E and P_o^W, we compute the sets $R^E(P_o^E, P_o^W)$ and $R^W(P_o^W, P_o^E)$. Then, for each $P^E \in R^E(P_o^E, P_o^W)$, we decide if P^E is P_o^W-red or P_o^W-green. Similarly, for each $P^W \in R^W(P_o^W, P_o^E)$ we decide if P^W is P_o^E-red or P_o^E-green. All these calculations can be done with any efficient algorithm for exploring graphs and checking connectivity (Ahujia et al. (1993)). Then, we check the condition of Theorem 1 to determine whether the trains are bound-to-deadlock.

Finally, algorithms searching for feasible plans may exploit the information about for which pairs of initial positions a pair of trains are not-bound-to-deadlock. To this end, for any pair of crossing trains t^X, t^Y, starting from their initial positions (*origins*) P_o^X, P_o^Y, we repeat the PCA for each pair of potential "initial" positions between the two origins. Namely, for each pair P^X, P^Y of potential initial positions, with $P^X \in R^X(P_o^X, P_o^Y)$ and $P^Y \in R^Y(P_o^Y, P_o^X)$, we establish whether the two trains are bound-to-deadlock. Of course, if t^E and t^W are bound-to-deadlock while in positions P^E and P^W respectively, it is not necessary to test any pair of positions belonging to $R^E(P^E, P^W)$ and $R^W(P^W, P^E)$, as the trains will still be bound-to deadlock. If this is not the case, we can apply some shortcuts whenever the color of a blocking path can be derived from what we have already computed. In particular:

1. $P^X \in R^X(P_o^X, P_o^Y)$ is P_o^Y-green, if P^X does not share any segment with any $P^Y \in R^Y(P_o^Y, P_o^X)$,
2. if $P^X \in R^X(P_o^X, P_o^Y)$ is P_o^Y-red, then P^X is P^Y-red for each $P^Y \in R^Y(P_o^Y, P_o^X)$;
3. if P^X is P_o^Y-green for $P^X \in R^X(P_o^X, P_o^Y)$, then P^X will be P_o^Y-green also when investigating a different initial position P_u^X such that $P_u^X \in R^X(P_o^X, P_o^Y)$ and $P^X \in R^X(P_u^X, P_o^Y)$.

5. Practical experience

In this section we show the relevance in the dispatching practice of the graph-theoretical results presented in the previous sections. We present two sets of experiments over a family of realistic instances, with features similar to the real-life instances from the Union Pacific network. Data set and executable code are both available here: http://www.optrail.com/en/downloads/.

A railroad the size of UP may have several hundreds, or even more than a thousand, trains on the network at any given time, some of which operating long trips through the network (up to days at the time). In general, however, each train interacts only with a subset of all the other trains and, for the purpose of real-time deadlock prevention, our focus can be limited to those that interact within a certain frame of time (we fix this time to 5 hours). This allows us to considerably reduce the number of trains pairs to be tested. The benchmark used in this paper is made of 10 instances, each one containing a set of trains (18 on average, with a minimum of 14 and a maximum of

22 trains) travelling on a network with both single main and multiple main areas. Experiments on larger instances were not included. Indeed, the computational times increase linearly with the number of pairs of interacting trains, and each pair can be processed independently. From each instance we randomly select a single pair of trains, making sure that in total we identify 5 pairs that interact exclusively in a single main scenario, that is a region of the network where adjacent stations are connected only by one track, which must be shared by trains running in both directions (in Europe, these are called single track regions). Because in this case trains can only meet in stations, single-track regions are typically the most prone to the risk of deadlocks. In the remaining 5 instances, the considered couples of trains span mixed regions, with both single and multiple tracks connecting adjacent stations. These pairs will be used to give more computational details.

As mentioned, we carried out two sets of experiments.

• The primary goal of our experiments is to test the efficacy of the PCA in checking whether pairs of trains are bound-to-deadlock. Recall that this test must be performed in real-time after a dispatcher has taken a specific decision about a train, typically preparing the outbound route from a station. The dispatcher's decision is enforced by the traffic management system by activating necessary signals and switches in the field. Therefore, the feasibility test must be very fast, to immediately notify the dispatcher of the potential risk and suspend the procedure before the action is taken. As we will show, the PCA only needs a few milliseconds to compute feasibility for a pair of trains. Since, as reported in Table 3, only a few other opposite trains interact with a given train in the planning horizon (as already mentioned, in our experiments as well as in operations, we considered on average 5 hours), this means that less than a hundredth of a second is required to perform the check.

• The second batch of experiments is devoted to compute *how far* a pair of opposite trains can move forward on their routes without risking to end up in a deadlock. This information can be exploited to enhance enumerative algorithms for train dispatching (such as Lamorgese and Mannino (2015)), namely for ranking meeting decisions and for identifying infeasible subtrees in early branchings. Indeed, the number of green blocking paths that a train has on its route is an important indicator of how critical choices are for that train.

Train t^X			Train t^Y			bound-to-	alert	total
l_{t^X}	$\lvert\mathcal{P}^X\rvert$	$\lvert\mathcal{P}^X\rvert/\lvert H^X\rvert$	l_{t^Y}	$\lvert\mathcal{P}^Y\rvert$	$\lvert\mathcal{P}^Y\rvert/\lvert H^Y\rvert$	deadlock	time	time
4789	65	1.41	13897	77	1.85	No	8.77	14.06
10200	133	1.69	11800	162	1.72	Yes	50.06	50.06
6897	55	1.34	9452	77	1.56	No	8.12	15.01
11278	249	1.85	11780	180	1.92	Yes	51.42	51.42
13897	127	1.89	9452	125	1.61	No	20.40	34.97
8300	181	1.79	8889	152	1.82	No	76.49	111.53
9889	143	1.89	8569	157	1.56	No	24.65	39.28
11500	149	1.96	8889	152	1.82	No	31.43	53.08
7700	62	1.41	8300	76	1.52	No	7.68	12.17
5200	63	1.37	9861	89	1.75	No	10.04	17.81

Table 2 Results over pairs of opposite trains. l_t is the train length, $|\mathcal{P}|$ the number of blocking paths and $|\mathcal{P}|/|H|$ the ratio between the number of blocking paths and of stopping points. The time in the last two columns is expressed in milliseconds.

First Batch Results. Here we describe our results for the main application of the PCA, namely determining whether a decision taken by a dispatcher, if executed, would lead a pair of trains to a deadlock.

Table 2 presents results on the individual pairs of trains extrapolated from the 10 instances, which are projected to interact within a given amount of time. Notice that in the upper half of the Table we report results for pairs interacting in single track regions, while in the lower half the entries refer to pairs in multiple tracks regions.

In Table 2, every line corresponds to a different pair of opposite trains and, for each train t^X, t^Y in the pair, we report the corresponding *length* l_{t^X}, l_{t^Y} (in meters), the number $|\mathcal{P}^X|, |\mathcal{P}^Y|$ of associated *blocking paths* and the *ratio* between the number of blocking paths and the number $|H^X|$, $|H^Y|$ of stopping points of the associated graph G^X, G^Y. Note that this ratio never exceeds 2, and so, although in theory the number of blocking paths could be much larger than the the number of stopping points (as it may grow exponentially with train lengths), in practice trains are short enough and tracks long enough to keep this ratio small. Column *bound-to-deadlock* displays whether the PCA detected a bound-to-deadlock situation for the pair, given the initial train positions. If trains are bound to deadlock, a warning should be sent to the dispatcher. Column *alert time* is the time in milliseconds required by the PCA to run and return the response. Note that in these instances deadlocks were found only for pairs in single main scenarios. Nevertheless, deadlocks occur also in multiple main regions, where it may be more difficult for dispatchers to anticipate them. Thus, it is crucial that PCA perform well on all scenarios. The figures reported in column *alert time* show that this is the case, as the maximum time for all instances is well below one tenth of a second.

Finally, column *total time* presents the time (in milliseconds) needed to compute, for each possible initial positions P^X, P^Y, with $P^X \in R^X(P^X_o, P^Y_o)$ and $P^Y \in R^Y(P^Y_o, P^X_o)$, whether the two trains are bound-to-deadlock. Note that when the trains are bound-to-deadlock in their original positions, then they remain bound-to-deadlock for any forward position and the total time coincides with the alert time. As we can see, these values show that the speed-ups described in Section 4 are very effective in reducing the total computing time. Indeed, in all instances, more than half of the time is used by the PCA on the original initial positions (P^X_o, P^Y_o) (alert time). In all 10 instances, less than 50% of the time is needed to extend the information to all the other (potential) initial positions of the two trains.

Second Batch Results. The figures presented in the last column (*total time*) of Table 2 are relevant for the second batch of tests. Indeed, on average, determining, for any possible initial positions of the trains, if a pair of trains is bound-to-deadlock takes just dozens of milliseconds. Such a performance makes it viable to extend the computation to an entire dispatching instance, i.e. for all pairs of interacting trains. As mentioned, this information can be used to guide search in enumerative methods for scheduling trains.

Instance	#trains	#pairs	#inter.	total time	min	max
I1	18	26	3.06	753.67	0.98	97.01
I2	22	44	4.00	1587.83	7.11	77.25
I3	16	22	2.93	1047.43	1.50	85.94
I4	16	34	4.86	1439.97	9.04	108.86
I5	15	21	3.23	391.73	5.96	42.96
I6	15	17	2.83	587.07	3.01	111.53
I7	20	36	3.60	1083.76	2.03	71.95
I8	16	25	3.13	899.18	9.64	85.16
I9	14	17	2.62	688.65	9.38	45.78
I10	20	31	3.65	1353.68	8.03	86.46

Table 3 Results over complete instances. Time is expressed in milliseconds

Table 3 shows aggregate computational times for our benchmark. Column *#trains* shows the total number of trains in each instance. Not all pairs of trains actually will meet/interact in the planning horizon. The number of pairs of interacting trains is reported in column *#pairs*. Column *#inter.* shows, on average, how many interactions a train has with other trains travelling in the opposite direction. As we can see, this number is quite small, with a maximum of almost 5 for instance I4. The total time needed to perform the feasibility test, together with the minimum and maximum time for a single pair (in milliseconds) are recorded respectively on columns *total time, min* and *max*. These results show that the time needed to build all the information for detecting bound-to-deadlock pairs is compatible with the real-life application.

6. Conclusions and future developments

In this paper we introduce the Bound-to-Deadlock 2 Trains Problem (BD2TP). It consists in deciding if two trains in a rail network are bound to end up in a deadlock situation, whatever trajectory they choose to reach their destination. We discuss the practical relevance of the problem and explain why avoiding deadlocks is a very critical issue, particularly for railroads that operate predominantly freight traffic.

We first give an algorithmic proof that BD2TP can be solved in polynomial time, reducing it to the solution of a polynomial number of longest path problems in a suitable acyclic graph. We also introduce an alternative algorithm that, although theoretically not polynomial, works very well in practice. We give evidence of its effectiveness by showing the results of a session of computational experiments conducted on a data set of realistic instances. In particular, we show that the Path Coloring algorithm can detect a possible 2 train bound-to-deadlock occurrence among a set of trains in a realistic size network within a few milliseconds of computing time. A deadlock prevention tool based on the Path Coloring algorithm will be integrated in Union Pacific's traffic management system in the near future.

As future development of the results presented here, we are currently developing approaches for the Bound-to-Deadlock Problem with any given number of trains, exploiting a recently introduced formulation (Lamorgese and Mannino (2019)) for this type of job-shop scheduling problems. Also, we are looking for possible extensions of the Path Coloring Algorithm to the case with three trains.

Acknowledgments

We would like to thank Brian Fish, John Fuller, Michelle Gartner, Shiying Hu and Katie Sanders from Union Pacific for the technical insight and practical support provided during this study.

References

Arbib C and Italiano G and Panconesi A (1988), Predicting deadlock in Store-and-Forward Networks. *Lecture Notes in Computer Science* 338: 123–142.

Arbib C and Italiano G F and Panconesi A (1990), Predicting deadlock in store-and-forward networks, *Networks*, 20(7): 861–881.

Ahujia R K and Magnanti T L and Orlin J B (1993), *Network flows: Theory, algorithms and applications.*

Corman F and Meng L (2015), A review of online dynamic models and algorithms for railway traffic management. *IEEE Transactions on Intelligent Transportation Systems* 16(3): 1274–1284.

Cui Y (2010), Simulation based hybrid model for a partially automatic dispatching of railway operation, *University of Stuttgart*

Cui Y and Martin U and Liang J (2017), Searching feasible resources to reduce false-positive situations for resolving deadlocks with the Banker's algorithm in railway simulation, *Journal of Rail Transport Planning & Management*, 7(1-2): 50–61.

Galli L and Stiller S (2018), Modern challenges in timetabling, *Handbook of Optimization in the Railway Industry*, 117–140.

Garey M R and Johnson D S (1979), *Computers and intractability.*

Gawrilow E and Köhler E and Möhring R H and Stenzel B (2008), Dynamic routing of automated guided vehicles in real-time, *In Mathematics–Key Technology for the Future 2008*: 165–177, Springer, Berlin, Heidelberg.

Kjenstad D and Mannino C and Schittekat P and Smedsrud M (2013), Integrated surface and departure management at airports by optimization, *In 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, 1–5.

Lamorgese L and Mannino C (2015), An exact decomposition approach for the real-time train dispatching problem. *Operations Research* 63(1): 48–64.

Lamorgese L and Mannino C (2019), A Noncompact Formulation for Job-Shop Scheduling Problems in Traffic Management. *Operations Research* 67(6): 1586–1609.

Lamorgese L, Mannino C, Pacciarelli D, Krasemann, JT (2018) Train Dispatching. *Handbook of Optimization in the Railway Industry* 265–283.

Li C L and McCormick T S and Simchi-Levi D (1992), Finding disjoint paths with different path-costs: Complexity and algorith. *Networks* 22(7); 653–667.

Li F and Sheu J B and Gao Z Y (2014), Deadlock analysis, prevention and train optimal travel mechanism in single-track railway system. *Transportation Research Part B: Methodological* 68: 385–414.

Lu Q and Dessouky M and Leachman R C (2004), Modeling train movements through complex rail networks, *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 14(1): 48–75.

Lübbecke E and Lübbecke M E and Möhring R H (2019), Ship traffic optimization for the Kiel Canal, *Operations Research*, 67(3): 791–812.

Habermann A N (1969), Prevention of system deadlocks, *Communications of the ACM*, 12(7):373–ff.

Mannino C and Nakkerud A and Sartor G (2021), Air traffic flow management with layered workload constraints, *Computers & Operations Research*, 127.

Markets & Markets (2019), Railway System Market by System Type, Transit Type, Application, & Region-Global Forecast to 2025.

Mascis A and Pacciarelli D (2002), Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research* 143(3): 498–517.

Mazzanti F and Spagnolo G and Della Longa S and Ferrari A (2014), Deadlock Avoidance in Train Scheduling: A Model Checking Approach. *Formal Methods for Industrial Critical Systems*: 109–123.

Pachl J (2002), Railway Operation and Control. *VTD Rail Publishing, Mountlake, Terrace, WA, USA.*

Pachl J (2007), Avoiding Deadlocks in Synchronous Railway Simulations. *International Seminar on Railway Operations Modelling and Analysis*: 359–369.

Pachl, J (2011), Deadlock avoidance in railroad operations simulations, *In 90th Annual Meeting of the Transportation Research Board*

Petersen E R and Taylor J (1982), A Structured Model for Rail Line Simulation and Optimization. *Transportation Science* 16(2): 192–206.

Pinedo M (2012), Scheduling. *Springer.*

Queyranne M and Schulz A (1994), Polyhedral approaches to machine scheduling. *Tech. Report TU Berlin.*

Simon B and Jaumard B and Le T H (2014), Deadlock Avoidance and Detection in Railway Simulation Systems. *ASME/IEEE Joint Rail Conference.*

Wen C and Huang P and Li Z and Lessan J and Fu L and Jiang C and Xu X (2019), Train Dispatching Management With Data-Driven Approaches: A Comprehensive Review and Appraisal. *IEEE Access* 7: 114547–114571.

World Rail Market Study (2018). *UNIFE.*