Submitted to *Transportation Science*
manuscript (Please, provide the manuscript number!)

Disruption management in railway systems by safe place assignment

Anna Livia Croella
Sapienza University of Rome, Rome, Italy, annalivia.croella@uniroma1.it,

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,

Paolo Ventura
Istituto di Analisi dei Sistemi ed Informatica (IASI) del CNR, Rome, Italy, paolo.ventura@iasi.cnr.it,

When major disruptions occur in a rail network, the infrastructure manager and train operating companies may be forced to stop trains until the normal status is recovered. A crucial aspect is to identify, for each train, a location (a *safe place*) where the train can hold during the disruption, avoiding to disconnect the network and allowing a quick recovering of the plan, at restart. We give necessary and sufficient conditions for a safe place assignment to have the desired property. We then translate such conditions into constraints of a suitable binary formulation of the problem. Computational results on a set of instances provided by a class 1 U.S. railroad show how the approach can be used effectively in the real-life setting that motivates the study, by returning optimal assignments in fractions of second.

Key words: Disruption Management, Safe Place Assignment, Train Rescheduling

1. Introduction

When a disruption occurs on a railway network, parts of the network may become unavailable for inbound trains, and decisions must be taken to mitigate the impact on overall traffic. This generally requires train movements to be re-planned in short time, that is, to take appropriate rerouting and

rescheduling decisions. Arguably the most critical issue is to avoid creating *deadlocks*, a situation that arises when a group of trains is positioned in such a way that none can move due to other trains in the group blocking their path. When, given the trains' positions, a deadlock is unavoidable, we say that the trains are *bound-to-deadlock*. Bound-to-deadlock scenarios may vary in terms of number of trains involved and distance between them based on the railway layout. For a more in-depth discussion on deadlocks we refer to Dal Sasso et al. (2021). Deadlocks are more prone to occur during disruptions, as the capacity of the network is often abruptly reduced, and traffic controllers (*dispatchers*) face unfamiliar and critical circumstances. In order to prevent deadlocks and in general to mitigate the negative effects of a disruption, dispatchers are required to implement specific recourse actions, sometimes based on predefined rules. For instance, the Norwegian infrastructure manager Bane NOR issues a list of such rules[1]. These rules define how to move or where to stop the trains in the affected region during the disruption, with some key objectives. Firstly, to leave, if possible, a viable path in the network with no obstructions, so as to allow possible work/maintenance trains to reach the disruption site(s). Secondly, to quickly return to the normal traffic regime once the disruption is over. Finally, to ensure continuity in the service offered to passengers, by taking recourse actions such as minimizing cancellations, adding short-turn trains and substituting missing segments with alternative bus transport, etc.

Indeed, most of the contributions presented in the literature on disruption management deal with (some of) these three issues. We refer the reader to Cacchiani et al. (2014), Corman and Meng (2015), and Ghaemi, Cats, and Goverde (2017b) for an extensive review on the subject. According to the topography of the infrastructure, the strategies used to manage a disruption are classified in short-turning/cancelling trains or stopping trains. The first approach is usually applied to short-line networks, as the ones in the European countries, where trains can be more easily rescheduled (see Ghaemi, Cats, and Goverde (2017a), Louwerse and Huisman (2014), Veelenturf et al. (2016), Schöbel (2007), and Schöbel (2009)). On the other hand, long-distance high-speed railway lines, such as the Chinese and the Japanese ones, usually need to stop trains on tracks or in stations for the time needed to recover from the disruption (see Hirai et al. (2009), Zhan et al. (2015), Zhan et al. (2019)).

Other aspects of the problem are considered in Zhu and Goverde (2019) and Zhu and Goverde (2020) that take into account the uncertainty of the disruption duration. Moreover, the *resilience* of a railway network is introduced in Bešinović (2020), by considering proactive (planning a robust system) and reactive (efficiently recovering from a disruption) issues.

[1] These rules are called *action cards* and can be found in BaneNOR (2021). The text is in Norwegian. The first set of action cards, namely 1A to 1K, concerns the Asker-Drammen line. For instance, action card 1A describes what to do with the trains when Drammen station is closed

Despite quite a large scientific literature on disruption management algorithms, we are not aware of existing real-life implementations. Note that such practical implementations will become indispensable once Autonomous Traffic Management Systems (ATMSs) will be used extensively. ATMSs are software components capable of generating routes and schedules (or *movement plans*) in real-time for the trains in a given region and planning horizon. An ATMS must solve repeatedly, and in short time, a difficult optimization problem, because the returned plan should minimize some measure of the overall train delays. The scientific literature on train scheduling is very extensive and dates back almost half a century (for extensive surveys, see Cacchiani et al. (2014), Corman et al. (2011), Fang, Yang, and Yao (2015), Josyula and Törnquist Krasemann (2017), Lamorgese et al. (2018), Mannino, Lamorgese, and Piacentini (2017), Wen et al. (2019)). Nevertheless, until now, there have been only a handful of real-life implementations (Lamorgese et al. (2018)), and in any case rather small in size or restricted to special cases; the implemented solution approaches are typically limited to simple local heuristics, and most research advances remain on paper or, at best, at the prototype level.

However, the landscape has changed quite rapidly over the past few years. Some infrastructure managers in Europe, such as Bane NOR in Norway (see BaneNOR (2018)) and DSB in Denmark (see DSB (2020)), and others, have issued tenders for the renewal of their TMS systems which include some explicit requirements for autonomous and optimized real-time scheduling functions.

In North-America, class 1 railroads Norfolk Southern and BNSF deployed General Electric's movement planner, which incorporates some optimization capabilities (see Bollapragada et al. (2018)); moreover, BNSF is directly involved in advanced research on dispatch optimization (Borraz-Sánchez, Klabjan, and Uygur (2020)). The General Electric movement planner was recently rolled out also by Aurizon in Australia (see Aurizon (2020)). Far from being exhaustive, the above examples show that the interest in ATMSs is on the rise, and the penetration of such tools, or at least the awareness of their potential, is growing in the industry.

Train scheduling problems in different regions of the world share a common core problem, but may also differ significantly in certain aspects. For example, in Europe rail traffic is largely focused on operating passenger services, with the Infrastructure Manager (IM) being a single, public authority, and train operating companies competing to provide services to passengers and, to a lesser extent, to ship freight. In other markets, such as the North American one, the rail industry is predominantly shaped by freight-hauling companies, with most companies owning and operating their rail network. Freight traffic presents some fairly different challenges from passenger traffic. Firstly, train traffic is often "un-scheduled", that is, no official timetable to adhere to exists. Secondly, railway business rules in America are somewhat more relaxed than in Europe, where trains generally have rigid routes

between stations and certain train movements cannot be automated for (passenger) safety concerns. Finally, and perhaps more importantly, freight trains are typically much longer than passenger ones, which drastically amplifies the challenge described above in dealing with disruptions and their aftermath.

The Safe Place Assignment Problem (SPAP) tackled in this paper follows a protocol (later denoted as *SP-protocol*) adopted by a large class 1 North-American railway to manage disruptions. The problem consists in finding, for each train affected by the disruption, a location on its path where the train can be safely parked during the critical situation, allowing traffic to route around it.[2] A good location for holding a train is called *safe place* (a more formal definition is presented in the following section). This safe place assignment serves the purpose of avoiding that trains drive past the last location where they can be safely parked, which could otherwise lead to further blockages and deadlocks. Currently, the safe place assignment is decided by experienced dispatchers in a greedy fashion. The ILP approach developed in this paper will allow to find an optimal assignment in a fraction of a second even for large instances (see Section 5), and can provide a strong support to dispatchers in their task.

Another application of the SPAP is related to the current usage of ATMSs. Since an ATMS solves instances of a hard optimization problem in very limited computational time, it may in general fail to return a plan. This can happen for different reasons. One possibility is that a solution exists, but the algorithm is not able to find it. Another is that the problem is infeasible, but the algorithm is not able to prove infeasibility. Infeasible instances correspond to so called *bound-to-deadlock* configurations, because if trains proceed from their current positions they will inevitably end up in a deadlock. Depending on the underlying algorithm used for planning, in these cases the ATMS may still be able to provide a *partial plan*, that is, a (feasible) plan that however spans a shorter planning horizon. Indeed, scheduling algorithms based on a "rolling horizon" paradigm, which tackle the problem by solving conflicts in a chronological order, are quite common in these applications (Zhan et al. (2016)). This approach mimics the most common planning process of human dispatchers, that can therefore be seen as a particular rolling horizon algorithm. The routes and schedules of such partial plan become input to the SPAP. Finding a solution to the SPAP for these plans adds an additional level of safety, as dispatchers are provided with the last safe location for a train to drive within the plan's horizon. Trains would not be automatically let past these locations until a complete plan is available, or only when the dispatchers take direct control of the situation.

Despite its significant practical relevance, the SPAP (in any of its variants) is rarely addressed in the scientific literature. We are only aware of two other papers on the subject.

[2] We note that the word "safe" here does not refer to the risk of accidents or damages to the infrastructure or rolling stock, which is responsibility of the safety layer of the signalling system

Zhan et al. (2019) presents a mixed integer program derived from Zhan et al. (2015) for rescheduling and/or canceling and/or assigning to safe places the long-distance and short-distance trains operating in the high-speed railway system connecting Beijing and Shanghai. The Petri-net-based model proposed in Hirai et al. (2009) assigns trains to safe places once the set of cancelled trains has been fixed and the tracks that should be unoccupied have been identified. Moreover, the model provides the sequence of atomic movements that vehicles have to perform to reach the stop location produced as output. Unfortunately, the resulting algorithm does not scale well and cannot solve real-world instances in a reasonable amount of time.

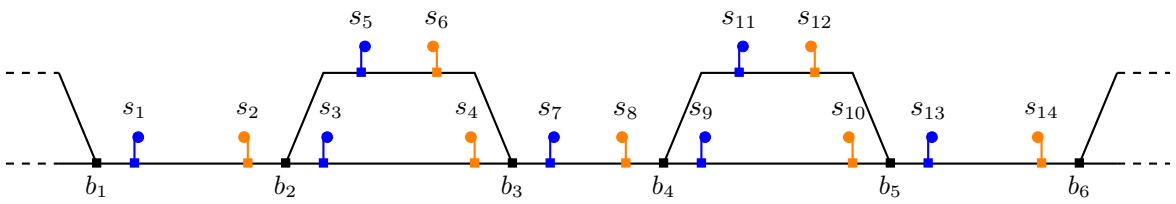
In the next sections, we present a purely binary programming model to solve the SPAP, adhering as much as possible to a given plan and fulfilling some residual capacity requirement of the network. Basic definitions and properties are given in Section 2 and Section 3. The resulting mathematical optimization model is described in Section 4. Finally, computational results over a set of realistic instances are presented in Section 5.

2. Networks, trains and schedules

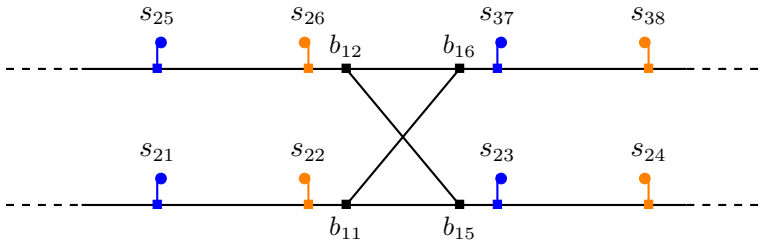
In this section we give a formal definition of the key elements of the SPAP, namely the railway network, trains and their movements.

The railway can be decomposed in sub-networks called *lines*. Lines can be traversed by trains in two directions, conventionally called *east-bound* and *west-bound*. A train running west-bound and a train running east-bound can share some tracks of the network. Trains running in opposite directions will traverse shared tracks in opposite order.

Rail line network. A schematic representation of a portion of rail line is shown in Figure 1. Here, solid segments represent tracks, while the small squares at the extreme of each track denote different objects: orange and blue squares correspond to *signal points*, locations where a signal is present and trains will stop (assuming a red light). The black squares represent *bifurcation points*, where a track splits into two (or two tracks converge). Notice that bifurcations actually identify track portions of different length for the different railway branches, while we model them as points. In fact, the length difference of using a reverse switch or not will be reflected on the length of occupied track portions, without the need of knowing the exact position of the ideal bifurcation point. Signal points are partitioned into east-bound signal points (where east-bound trains can stop) and west-bound signal points (where west-bound trains can stop). The track between two successive points p, q (either signal or bifurcation) is called *segment* and denoted as the unordered pair of extremes $\{p, q\}$. In Figure 1, the orange squares are the east-bound signal points, (denoted by s_i, with i odd), while the blue squares (s_i, with i even) are the west-bound signal points. The black squares (denoted by b_i, $i \in \mathbb{N}_+$) are the bifurcation points. $\{b_4, s_8\}, \{s_1, s_2\}, \{b_4, s_9\}, \ldots$ are segments. Following the planned



(a) Single-track network



(b) Multiple-track network

Figure 1 **Schematic railway networks. Figure 1a shows a single track area, where stations are linked by only one track, while Figure 1b shows a portion of multiple track network, where more than one track enter/exit a station or a crossover.**

movement of trains through the line, two "opposite" partial orders can be defined over the points and segments of a line network: an east-bound order, corresponding to the order in which tracks are traversed by an east-bound train, and a west-bound order. Note that if p_1, p_2 (s_1, s_2) are two points (segments), then $p_1 \prec p_2$ $(s_1 \prec s_2)$ in the east-bound order if and only if $p_2 \prec p_1$ $(s_2 \prec s_1)$ in the west-bound order.

Definition 2.1. A *rail path* is an alternating sequence of points and segments of the line that is ordered either west-bound or east-bound.

In Figure 1, $P_W = (s_5, \{s_5, s_6\}, s_6, \{s_6, b_3\}, b_3, \{b_3, s_7\}, s_7, \{s_7, s_8\}, s_8)$ is a (west-bound) rail path and $P_E = (s_{14}, \{s_{14}, s_{13}\}, s_{13}, \{s_{13}, b_5\}, b_5, \{b_5, s_{10}\}, s_{10})$ is an east-bound rail path. Note that, according to the definition, the alternating sequence $(s_5, \{s_5, s_6\}, s_6, \{s_6, b_3\}, b_3, \{b_3, s_4\}, s_4)$ is not a rail path, because it is ordered neither west-bound nor east-bound. Observe that the initial position of an X-bound train, from its head to its tail, defines an X-bound rail path. We say that two rail paths *intersect* if they share at least a segment.

Train movements Let t be an X-bound train, with $X \in \{east, west\}$. The movement of t across the network is identified by the movement of its head. The *route* R_t of a train $t \in T$ is the rail path obtained by concatenating the segments occupied by the train in its initial position, with the X-path followed by the head of t from origin to destination across the line. For our purpose we can decompose such movement into a discrete sequence of elementary movements, or *steps*, from an X-bound signal to the next X-bound signal on R_t, and we can assume that, at the end of each

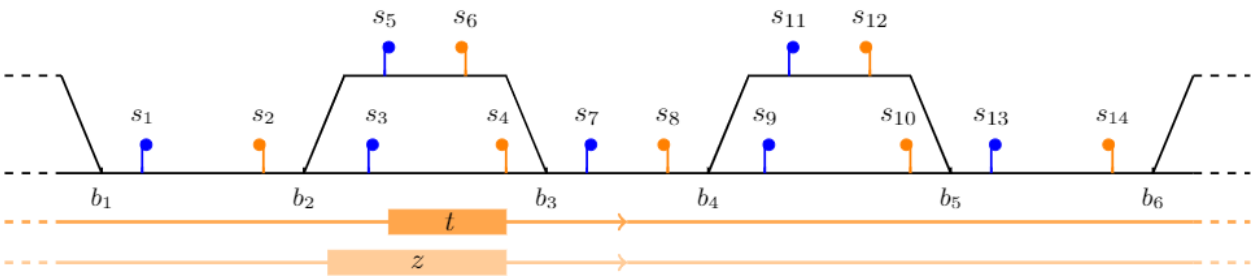


Figure 2 **Examples of blocking paths occupied by trains. Trains t and z are east-bound trains having their heads at signal point s_4. We have that $P(t, s_4) = (\{s_4, s_3\})$ and $P(z, s_4) = (s_4, \{s_4, s_3\}, s_3, \{s_3, b_2\}, b_2, \{b_2, s_2\}, s_2)$.**

elementary movement, the head of the train is always facing a signal (point) in such direction. After each elementary step, the train can stop at the signal point or proceed with the next step. Depending on its length from head to tail, a train t will then be occupying a portion of the line which cannot be occupied by other trains. This portion starts at the signal point faced by the head of t and includes all the segments physically occupied by the train from head to tail. However, for safety reason, the portion of line forbidden to other trains (*blocked* segments) is extended "backward" on R_t from the segment containing the tail of t until a segment containing a signal point (in any direction) is met. This motivates the following definition:

Definition 2.2. Let t be a X-bound train, with $X \in \{\text{east, west}\}$, let R_t be the route of t and assume that the head of t is at the X-bound signal point s. Then we associate with t, s a *blocking path* $P(t, s)$ as follows:

1. $P(t, s)$ is a sub-path of R_t with head-point at s
2. The tail-point of $P(t, s)$ is a signal point
3. $P(t, s)$ contains all segments physically occupied by t
4. $P(t, s)$ is minimal

. Condition 4 implies that any rail path satisfying conditions 1, 2 and 3 contains $P(t, s)$. To better grasp this definition we can look again at Figure 2. Suppose, the tracks $\{s_4, s_3\}$, $\{s_3, b_2\}$, and $\{b_2, s_2\}$, have length, respectively, 2100, 900 and 400 m and assume that an east-bound train t has its head at signal s_4. Now, if the length of a train t is ≤ 2100 m, then t occupies only the segment $(\{s_4, s_3\})$. On the other hand, if a train z has length 2450 m then z blocks the entire blocking path $(s_4, \{s_4, s_3\}, s_3, \{s_3, b_2\}, b_2, \{b_2, s_2\}, s_2)$.

Note that the route R_t of t can also be viewed as an ordered sequence of blocking paths, which in turn corresponds to the ordered sequence of elementary movements of t. Also observe that two blocking paths of the same train can overlap on some segments. We let $\mathcal{P}(t) = \{P(t, s) : s \in R_t\}$ be the set of all blocking paths associated with a train t.

Blocking paths and train scheduling. Summarising, when a train t is at a signal point s, it is actually blocking, or virtually occupying, all segments in the blocking path $P(t,s)$. Blocking paths play a central role when coordinating trains through the network because two trains can never occupy overlapping blocking paths at the same time. A *train plan* specifies the route for each train t and the schedule of t, i.e. the time train t enters each blocking path in its route.

If P_t is a blocking path on the route R_t of t and P_z is a blocking path on the route R_z of another train z, and P_t and P_z intersect, we say that there is a potential *conflict* between t and z, and either t moves through P_t before z moves through P_z, or vice versa. Any feasible schedule for the trains in the network establishes which train goes first for any potential conflict.

Disjunctive Graph. Assuming train routes are given, the problem of finding a feasible (or optimal) train schedule can be regarded as a job-shop scheduling problem with blocking and no-wait constraints (Mascis and Pacciarelli (2002)). We can associate with any train scheduling instance a *disjunctive graph*. Disjunctive graphs were introduced in the context of production scheduling (Balas (1969)), but they can be exploited to represent any problem in which we need to schedule a set of events. A disjunctive graph is a triple $G=(N,E,D)$. The nodes N are in correspondence with the events to be scheduled (e.g. start of operations, landing times of airplanes, project milestones, etc.). A schedule is a vector $\tau \in \mathbb{R}_+^N$ which associates a time with every event in N. N contains a special node o, called *origin*, which represents the event *start schedule*. All other events are scheduled after o and, in our context, we may assume $\tau_o = 0$. The set E of directed edges (or arcs) represents time precedence relations between events. In particular, a directed edge $(u,v) \in E$ of length $l_{uv} \in \mathbb{R}$ represents a constraint of the form $\tau_v - \tau_u \geq l_{uv}$. Also, since no events can start before the origin o, we have $\tau_v - \tau_o \geq 0$, for $v \in N \setminus \{o\}$, and edge $(o,v) \in E$, for $v \in N \setminus \{o\}$, with $l_{ov} = 0$. Finally, each element $d \in D$, called *disjunctive arc* (see Figure 3), is an unordered pair of directed edges $d = \{(u,v),(q,r)\}$ which represents the alternative sequencing of four (not necessarily distinct) events, namely any feasible schedule must satisfy either $\tau_v - \tau_u \geq l_{uv}$, or $\tau_r - \tau_q \geq l_{rq}$, with $l_{uv}, l_{qr} \in \mathbb{R}$.

So, given an instance defined by a disjunctive graph $G=(V,E,D)$ together with a distance vector l, in order to find a feasible schedule of the corresponding scheduling problem, we must decide, for each disjunctive arc, which of the two directed edges has to be "satisfied". A *selection* is a set S of directed edges which intersects exactly once all disjunctive arcs. Then a schedule is feasible if, (i) for each edge $(u,v) \in E$, we have that $\tau_v \geq \tau_u + l_{uv}$ and (ii) for each disjunctive arc $d = \{(u,v),(q,r)\} \in D$, we have $\tau_v - \tau_u \geq l_{uv}$, or $\tau_r - \tau_q \geq l_{rq}$, with $l_{uv}, l_{qr} \in \mathbb{R}$. The following is a key result in job-shop scheduling (Pinedo (2009)):

Theorem 2.3. An instance of scheduling problem with associated disjunctive graph $G = (N,E,D)$ admits a feasible schedule if and only if there exists a selection S such that the graph $G(S) = (N, E \cup S)$ does not contain a strictly positive directed cycle.

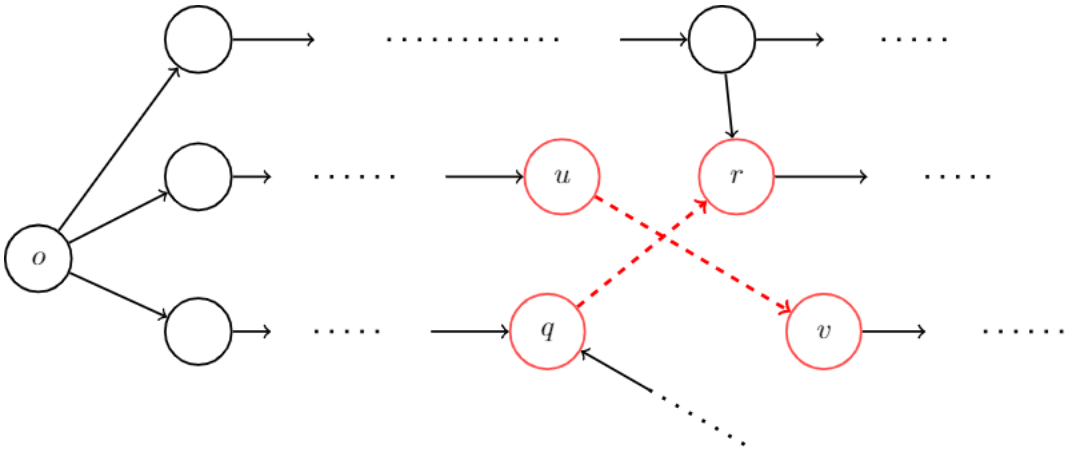


Figure 3 **A disjunctive graph. The dashed edges form a disjunctive arc.**

In particular, given a feasible schedule $\tau \in \mathbb{R}_+^N$, we can immediately derive a selection $S(\tau)$ such that $G(S)$ does not contain directed cycles, by letting, for each $\{(u,v),(q,r)\} \in D$, $(u,v) \in S$ if $\tau_u \geq \tau_v + l_{uv}$, and $(q,r) \in S$ otherwise. On the other hand, given a selection S such that $G(S)$ does not contain a strictly positive directed cycle, one can easily derive a feasible schedule $\tau(S)$ by applying a longest path tree algorithm to the digraph $G(S) = (N, E \cup S)$ with edge lengths l (see Pinedo (2009)).

The following observations are useful for the sequel of the paper.

1. Determining the feasibility of a generic instance of job-shop scheduling problem is NP-complete (see Pinedo (2009)), and this holds also for job-shop scheduling problems with blocking and no-wait constraints (Mascis and Pacciarelli (2002)).
2. Let $c : \mathbb{R}_+^N \to \mathbb{R}$ be a cost function. Finding the feasible schedule τ such that $c(\tau)$ is minimized is computationally a difficult problem. If c is monotone non decreasing, let τ^* be the minimum cost schedule, and let $S^* = S(\tau^*)$ be the selection associated with τ^*. Then the optimal schedule τ_u^*, for $u \in N$, equals the length of a longest path from the origin o to $u \in N$ in $G(S^*)$. On the other hand, given a selection S, finding the associated feasible schedule $\tau^*(S)$ which minimizes a non-decreasing cost function is computationally easy, as it corresponds to computing a longest path tree in $G(S)$.

It follows from the above discussion that we can immediately associate a disjunctive graph $G = (N, E, D)$ and a distance vector l with an instance of the train scheduling problem in normal traffic regime (i.e. with no ongoing disruption). In particular, we let the set of nodes contain all of the blocking paths, i.e. $N = \{o\} \cup \{P \in \mathcal{P}(t) : t \in T\}$. Also, for $t \in T$ and $P \in \mathcal{P}(t)$, we let $\sigma(P)$ be the blocking path next to P on the route R_t of train t. Then edge $(P, \sigma(P)) \in E$ and its length $l_{P,\sigma(P)} > 0$ is the minimum time required by the head of t to travel from the head point of P to the head point of $\sigma(P)$. Finally, let P_t and P_z be two intersecting blocking paths of the route of

train t and z, respectively. Since t cannot be in P_t while z is in P_z, then either (i) t enters the blocking path $\sigma(P_t)$ following P_t before z moves through P_z, or (ii) z enters the blocking path $\sigma(P_z)$ following P_z before t moves through P_t. Condition (i) is represented by the directed edge $(\sigma(P_t), P_z)$ with length $l_{\sigma(P_t),P_z} = \epsilon$, whereas condition (ii) is represented by the directed edge $(\sigma(P_z), P_t)$ with length $l_{\sigma(P_z),P_t} = \epsilon$.[3] Therefore, the disjunctive decision can be represented by a disjunctive arc $\{(\sigma(P_t), P_z), (\sigma(P_z), P_t)\} \in D$. It is not difficult to see that, due to the particular form of the length vector l, if $G = (N, E, D)$ is the disjunctive graph associated with an instance of train scheduling and S is a selection, then any cycle in $G(S)$ is strictly positive.

So, the condition of Theorem 2.3 can be stated simply as $G(S)$ is acyclic (i.e. with no directed cycles). By extension, we say that S is an *acyclic selection* of G.

Note that the main task of train dispatchers is to solve "conflicts" between trains, that is, deciding the order in which the trains will travel through the conflicting blocking paths. From a mathematical standpoint, solving a conflict amounts to selecting an edge in a disjunctive arc. Now, from time to time it may happen that decisions taken by dispatchers lead to a *bound-to-deadlock* configuration, i.e. a situation in which a feasible schedule no longer exists. Theorem 2.3 tells us that the corresponding selection induces a directed cycle in a certain disjunctive graph associated with the train scheduling instance. In the next section we will show how disjunctive graph and selection can be exploited in order to find a suitable set of safe places for trains during disruptions.

3. The Safe Place Assignment Problem

As mentioned in the introduction, a major goal of disruption management is to avoid deadlocks when trains resume their trip. Disruptions reduce the capacity of the network and may force dispatchers to hold a subset of trains, as their final destination is no longer reachable. As these disruptions may last for a long time, it is crucial to avoid that trains "pile" around the disruption, as trains should be able to reach their destination once the disruption is lifted. Moreover, depending on the type of disruption, it may be required that emergency vehicles reach the disruption location. These requirements introduce the idea of assigning “safe” places to the trains, that are locations where the trains can stop without impeding entirely the traffic flow.

In the rest of this section we describe an optimization model to support the SP-protocol for disruption management developed by a Class 1 North-American railway operator. As mentioned in the introduction, when a major disruption occurs dispatchers may not be able to stick to the (real-time) plan they initially established. The SP-protocol enforces that affected trains hold at

[3] Where $\epsilon > 0$ represents the generic clearing time of the resource.

some location along their planned routes, until the disruption is over and regular traffic is restored. Now, an X-bound train t can only stop at an X-bound signal, and therefore the set of potential holding places is in correspondence to the X-bound signal points in R_t. Moreover, because t will indeed virtually occupy an entire blocking path, we can say that the set of (potential) safe places for t is the set $\mathcal{P}(t)$ of its blocking paths. Thus, the Safe Place Assignment Problem consists in assigning a safe place $\overline{P}_t \in \mathcal{P}(t)$ to each train t. In finding the best possible holding locations, i.e. the safe places, the SP-protocol pursues three distinct and somehow conflicting objectives:

- O1 The primary goal is to avoid that when the disruption is lifted trains are bound-to-deadlock.
- O2 Secondly, to leave some paths free of trains so that worker trains can reach the disrupted area.
- O3 Finally, to "push" trains as far along their routes as possible, so as to mitigate the impact of the disruption in terms of train delays.

Objectives O1 and O2 are conditions imposed on the feasible assignments and will be translated into constraints of the ILP formulation in Section 4. In contrast O3 only affects the objective function of the formulation. Notice that this SP-protocol has been defined according to the requirements of one railway operator. The particular type of trains and layout of the railway network considered may affect the definition of the preferred, or even of the feasible safe places assignments. However, the formulation presented in this paper is flexible and can be adapted to different requirements. As an example, if the preferred policy is not to push all the trains as far as possible, but there is a special focus on not stopping passengers trains outside of a station, this could be easily integrated in the model by modifying the objective function.

Implementing goal O1. We start the discussion with how to pursue goal O1. We assume that, before the disruption occurs, trains run through the network according to a feasible plan which specifies the routing R and the schedule τ for all trains. As described in Section 2, we can associate with R a disjunctive graph $G = (N, E, D)$ with arc lengths l, which represents our instance at the moment the disruption strikes. Note that the planned schedule τ induces a selection $\overline{S} = S(\tau)$, and $G(\overline{S})$ is acyclic. However, as a consequence of the disruption, trains will have to hold somewhere on their routes, the scheduling instance changes and (in general) the schedule τ is no longer feasible.

Therefore, let ϕ be the function that, when the disruption occurs and according to the SP-protocol, assigns each train t a safe place $\overline{P}_t \in \mathcal{P}(t)$. Trains will arrive at their assigned safe place and will hold there until the disruption is lifted. Now, to represent the new scheduling instance associated with the disruption event and the additional constraints imposed on the movement of trains by the SP-protocol, we derive from the original instance $G = (N, E, D)$ a new disjunctive graph $G'(\phi) = (N', E', D')$ and distance function l'. The node set $N' = N \cup \{w\}$ contains a new element w that represents the end of the disruption. Next, the set E' properly contains the original set $E \subset E'$, with their original lengths. Moreover, since we assume trains arrive at their assigned

safe places before the disruption ends, we have $(\overline{P}_t, w) \in E'$, with $l'_{\overline{P}_t,w} = 0$, for $t \in T$. Also, since the trains can only move away from their safe places after the disruption ends, we have $(w, \sigma(\overline{P}_t)) \in E'$, with $l'_{w,\sigma(\overline{P}_t)} = 0$, for $t \in T$. Finally, because train routes have not changed and there are no new blocking paths, we have $D' = D$, and the edges in each disjunction maintain their original lengths.

Summarizing, the disjunctive graph $G'(\phi) = (N', E', D')$ with edge lengths l', associated with an assignment of safe places $\phi = \{\overline{P}_t \in \mathcal{P}(t) : t \in T\}$, is defined as follows:

- Node set $N' = N \cup \{w\}$. Node w represents the event *end of disruption*.
- Edge set $E' = E \cup E^{in}_w \cup E^{out}_w$, where $E^{in}_w = \{(\overline{P}_t, w) : t \in T\}$, representing the fact that trains enter their safe place before the end of the disruption, and $E^{out}_w = \{(w, \sigma(\overline{P}_t)) : t \in T\}$, representing the fact that trains exit their safe place after the end of the disruption. Moreover, $l'_e = l_e$ if $e \in E$, and $l'_e = 0$ otherwise.
- Disjunctive arcs $D' = D$ and $l'_e = l_e$ for $e \in d$, $d \in D'$.

Note that, since $D' = D$, the selection $\overline{S} = S(\tau)$ of G associated with the original schedule τ is also a (not necessarily acyclic) selection of G', although τ is not a feasible schedule for G'. Observe now that, in order to achieve objective O1, we need to find an assignment of safe places ϕ and a selection S such that $G'(\phi)(S)$ is acyclic. The next theorem provides a way to find such an assignment and a selection in a natural and effective way by exploiting $\bar{S}$. The theorem provides a necessary and sufficient condition on ϕ so that $G'(\phi)(\overline{S})$ is acyclic, i.e. so that the selection induced by the original schedule τ is still feasible after the disruption. This result is the basis for our ILP formulation in Section 4. In the following, for any $t \in T$ and $P, Q \in \mathcal{P}(t)$, we let $P \preceq Q$ ($P \prec Q$) if Q does not precede P (Q strictly follows P) on the route R_t of train t. In the proof of the next Theorem 3.1 we make use of a well known result in graph theory: Given a directed graph $H = (V, A)$ with edge set A, a topological ordering is a function $f : V \to Z_+$ which associates with every node a non-negative integer such that $f(v) > f(u)$ for all $(u, v) \in A$. Then a graph H admits a topological ordering if and only if G does not contain a directed cycle (Cook et al. (1998)).

Theorem 3.1. Let $G(\overline{S}) = (N, E \cup \bar{S})$ be acyclic and let $\overline{\phi}$ assign each train t a safe place $\overline{P}_t \in \mathcal{P}(t)$. Then the following claims are equivalent

i) $G'(\overline{\phi})(\overline{S}) = (N', E' \cup \bar{S})$ is acyclic;

ii) for each $(\sigma(P_t), P_z) \in \overline{S}$ with $\overline{P}_t \preceq P_t$, we have that $\overline{P}_z \prec P_z$.

Proof. We first show that i) implies ii). In particular, we prove that if $\overline{S}$ contains an arc $(\sigma(P_t), P_z)$ such that $\overline{P}_t \preceq P_t$ with $\overline{P}_z \succeq P_z$, then $G'(\phi)(\overline{S})$ admits a cycle C. Indeed, since $\overline{P}_t \preceq P_t$, then $\sigma(\overline{P}_t) \preceq \sigma(P_t)$. Therefore, $\sigma(P_t)$ follows $\sigma(\overline{P}_t)$ on the route of train t and there is a path in G' from $\sigma(\overline{P}_t)$ to $\sigma(P_t)$, say Q_t (see Figure 4). Similarly, as $P_z \preceq \overline{P}_z$, G' also contains a directed path, say Q_z, from P_z to $\overline{P}_z$. Now, we get C by concatenating $(w, \sigma(\overline{P}_t))$, Q_t, $(\sigma(P_t), P_z)$, Q_z, and $(\overline{P}_z, w)$.

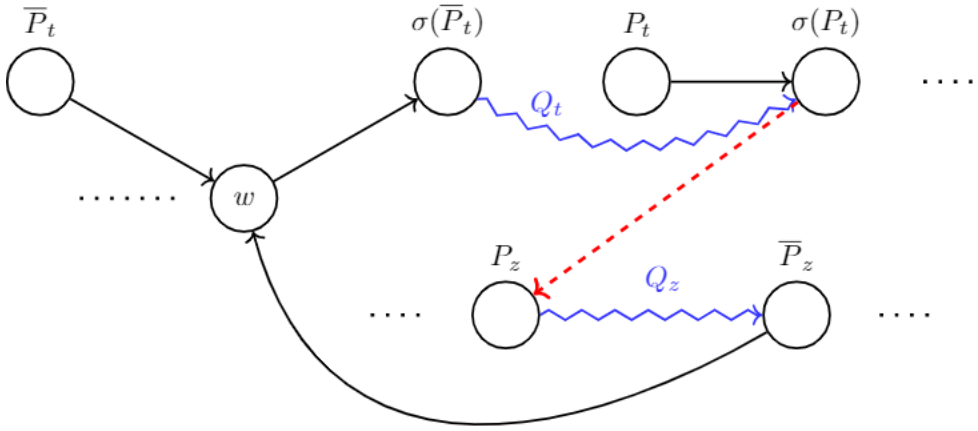


Figure 4 **Proof of Theorem 3.1 i) -> ii).**

Now we prove that ii) implies i). First observe that, since $G(S)$ is acyclic, there is a topological ordering $f : N \to \{1, \ldots, n\}$ of its nodes. Now, let $B = \{P_t \succ \overline{P}_t : t \in T\}$, and let $A = N \setminus B$. Then $N' = A \cup B \cup \{w\}$. Define $f' : N' \to \mathbb{Z}_+$ as follows: $f'(u) = f(u)$, for $u \in A$, $f'(u) = f(u) + n + 1$, if $u \in B$, and let $f'(w) = n + 1$. We show that f' is a topological ordering of G'.

Suppose not, then there is an edge $(u, v) \in E' \cup \overline{S}$ such that $f'(u) > f'(v)$.

- If $u = w$, then $v = \sigma(\overline{P}_z)$ for some $z \in T$. But then $\sigma(\overline{P}_z) \in B$, $f'(\sigma(\overline{P}_z)) \geq n + 2$ and we have $f'(u) = f'(w) = n + 1 < f'(\sigma(\overline{P}_{\bar{t}})) = f'(v)$, a contradiction.
- If $v = w$, then $u = \overline{P}_z$ for some $z \in T$. But then $\overline{P}_z \in A$, $f'(\overline{P}_z) \leq n$ and we have $f'(u) = f'(\overline{P}_z) \leq n < n + 1 = f'(w) = f'(v)$, a contradiction.
- Finally, consider the case with $u, v \neq w$. Then arc (u, v) belongs also to G, implying $f(u) < f(v)$. Let $u = \sigma(P_t) \in \mathcal{P}(t)$, for a $t \in T$ and $v = P_z \in \mathcal{P}(z)$, for a $z \in T$. Since $f'(u) > f'(v)$, but $f(u) < f(v)$, we must have $u = \sigma(P_t) \in B$ and $v = P_z \in A$. But this implies that $t \neq z$, and $(\sigma(P_t), P_z) \in \overline{S}$. Then we have $\overline{P}_t \prec \sigma(P_t)$ and $P_z \preceq \overline{P}_z$, a contradiction.

As a consequence of Theorem 3.1 we have that, if $G(\overline{S})$ is acyclic, then we can always assign all trains the safe places associated with their starting positions. Indeed, the following holds true.

Corollary 3.2. Let $G(\overline{S})$ be acyclic and let $\overline{\phi}$ assign each train t a safe place $\bar{P}^t$ that corresponds to its starting position P_0^t. Then $G'(\overline{\phi})(\overline{S})$ is acyclic.

Proof. At the beginning of the planning horizon, every train $t \in T$ is in its starting position P_0^t implying $\bar{P}^t = P_0^t \preceq P^t$, for all $P^t \in \mathcal{P}(t)$. Suppose now $(\sigma(P^t), P^z) \in \overline{S}$ for some $\sigma(P^t) \in \mathcal{P}(t), P^z \in \mathcal{P}(z)$, with $t \neq z$. Since no train can enter (a segment of) P_0^z before z, we must have $P^z \neq P_0^z$, and so $P_0^z \prec P^z$, thus verifying condition ii) of Theorem 3.1.

In our ILP model (Section 4), we assume that a newly computed schedule that factors in the safe place assignment will also agree with the selection $\overline{S}$ associated with the original schedule τ. Then,

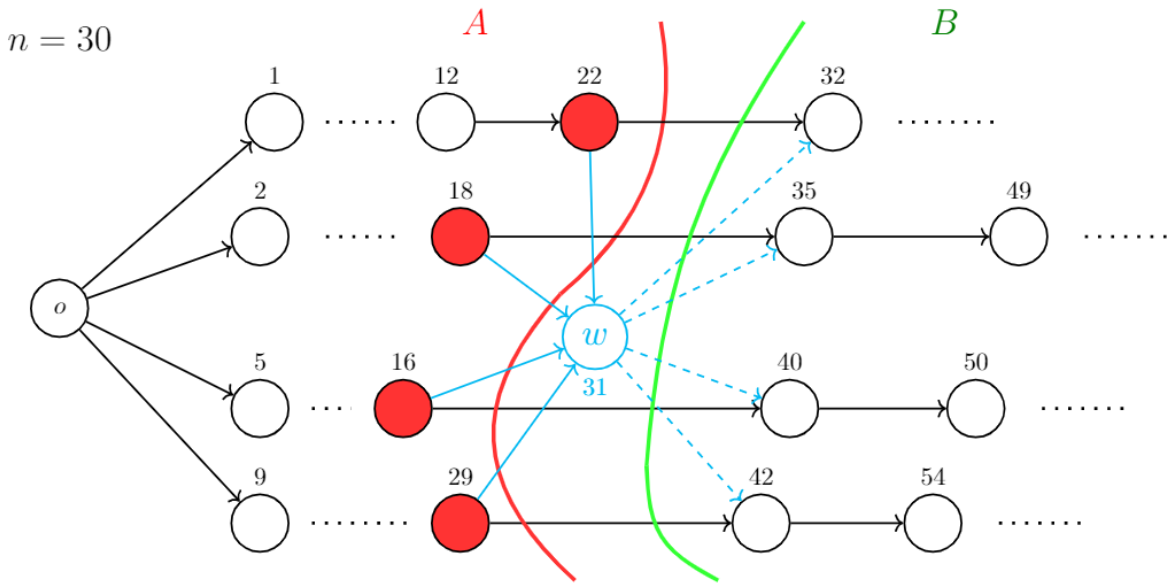


Figure 5 **Proof of Theorem 3.1 ii) -> i). Filled nodes represent a chosen *safe place* assignment.**

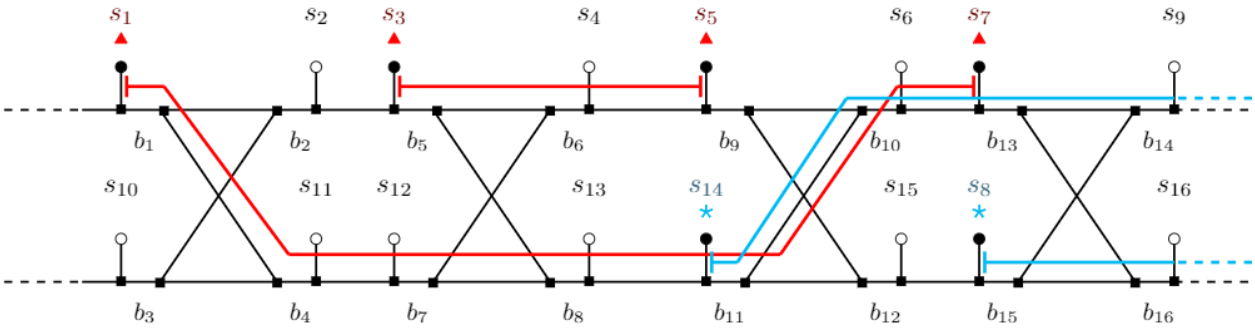


Figure 6 **Example of paths that can be used to bypass a train parked in a safe place. Signals with the same token are endpoints signals for the paths that allow to bypass the safe place segment marked with the same color.**

we have constraints imposing that the safe place assignment $\overline{\phi}$ satisfies the condition ii) of Theorem 3.1.

Implementing goal O2. The purpose of goal O2 is to allow worker trains to reach the disruption, bypassing trains that are parked in safe place. In order to model this requirement, we need first to give a formal definition for a rail path Q to bypass a blocking path P.

Definition 3.3. Let P be a blocking path and Q be a rail path with endpoints q_1 and q_2. Then we say that Q *bypasses* P if

(i) Q does not contain any segment of P

(ii) there exists a rail path between q_1 and q_2 that contains (all the segments of) P.

Informally, condition (ii) states that P "lies" between q_1 and q_2 and condition (i) ensures that there is a path (Q) between q_1 and q_2 which avoids P. As an example, in Figure 6, path $P = (s_3, \{s_3, b_5\}, b_5, \{b_5, b_6\}, b_6, \{b_6, s_4\}, s_4, \{s_4, s_5\}, s_5)$ is bypassed by path $Q = (s_1, \{s_1, b_1\}, b_1, \{b_1, b_4\},$

b_4, $\{b_4, s_{11}\}$, s_{11}, $\{s_{11}, s_{12}\}$, s_{12}, $\{s_{12}, b_7\}$, b_7, $\{b_7, b_8\}$, b_8, $\{b_8, s_{13}\}$, s_{13}, $\{s_{13}, s_{14}\}$, s_{14}, $\{s_{14}, b_{11}\}$, b_{11}, $\{b_{11}, b_{10}\}$, b_{10}, $\{b_{10}, s_6\}$, s_6, $\{s_6, s_7\}, s_7$).

Therefore, goal O2 can be implemented by requiring that each train, once stopped in a safe place (i.e. in a blocking path), has to be bypassed by some "short" rail path that will not be occupied by other trains parked in their safe places. The maximum length of such short rail path is a requirement that can be defined from time to time according to some rules fixed by the rail company. For example, a path can be considered short if it does not exceed a given length or, alternatively, if it is composed by at most a given number of segments. All these alternatives can be easily implemented in the proposed model.

In the SP-protocol considered here, the rail company directly identifies the set $\mathcal{U}$ of paths that can be used to bypass the trains parked in the safe places, which is thus an input to our model. In the following, for each possible safe place $P \in \mathcal{P}(t)$ for some train $t \in T$, we will denote by U^P the set of paths of $\mathcal{U}$ that bypass P. Such sets U^P can be easily calculated off-line for each P, once the family $\mathcal{U}$ is given. Then, in our model condition O2 is satisfied if the following holds: for each train t assigned to a safe place $P \in \mathcal{P}(t)$, at least one path Q in U^P is "free", i.e. it does not intersect with any safe place assigned to some train. We say that, in this case, Q is *reserved* (because of P).

Implementing goal O3. The last goal O3 does not require further discussion, as it only affect the objective function on the ILP model.

Summarizing, the purpose of the SPAP is to find an assignment of blocking paths to trains and a conflict-free schedule pursuing objective O3 while conditions O1 and O2 are satisfied.

A case that must be considered is when a feasible assignment does not exist. In this situation, the SP-protocol allows any train $t \in T$ to hold in the first blocking path P_t^0 of its route while violating the constraints associated with goal O2. We define such trains as *halted*. Note that the solution in which all trains are halted is conflict-free (as trains are physically occupying these positions in the instance) and satisfies the conditions imposed by goal O1 (thanks to Corollary 3.2). Clearly, one wants to minimize the number of halted trains, which will thus be taken into account in the objective function. It must be noted that when the number of halted trains is larger than one, some conditions must be verified before the safe place assignment can be applied automatically in the case of partial plan. Namely, the assignment can be carried out if, for every pair of halted trains, these are either trailing, i.e. travelling in the same direction, or have no "interaction", that is they do not risk having conflicts on any resource potentially reachable within the planning horizon. These conditions are often satisfied in North-American freight railways, where traffic is relatively sparse in lower capacity areas in which trains are more prone to being "halted".[4] In those rare cases of partial

[4] In denser, typically multiple track areas instead, the higher capacity available will generally work towards having less/no halted trains altogether

plans in which these conditions do not apply, the SP-protocol requires the dispatcher overseeing operations to take control of the involved trains and "manually" assign trains to their desired safe place (if any).

4. A binary linear programming formulation for SPAP

Let $T = T^E \cup T^W$ be a finite set of trains, where T^E indicates *east*-bound trains and T^W *west*-bound ones.

For each train $t \in T$, let $\mathcal{P}(t)$ be the set of its candidate safe places, ordered from the origin to the destination. We denote with P_0^t the first blocking path on the route of t (starting position), that is the only assignable blocking path for a halted train.

Next, we introduce the following set of binary variables:

$$x_P^t = \begin{cases} 1 & \text{if train } t \text{ is assigned to safe place } P \\ 0 & \text{otherwise} \end{cases} \qquad t \in T,\ P \in \mathcal{P}(t)$$

$$h^t = \begin{cases} 1 & \text{if train } t \text{ is halted in its starting position } P_0^t \\ 0 & \text{otherwise} \end{cases} \qquad t \in T$$

The first set of linear constraints implies that each train is either halted in its initial position or assigned to a safe place.

$$h^t + \sum_{P \in \mathcal{P}(t)} x_P^t = 1 \qquad t \in T \tag{1}$$

Note that the two conditions $h^t = 1$ and $x_{P_0^t}^t = 1$, although train t is physically parked in the same blocking path (the first), actually represent very different situations. Indeed, a halted train does not need to satisfy condition O2. Precisely to distinguish these cases, we define $P \in \mathcal{P}(t)$ to be a safe place for t if and only if $x_P^t = 1$.

Clearly, if a train is parked (either halted or in safe place) in a blocking path P, then no other train can be parked in a blocking path P' sharing a segment $s \in P \cap P'$. This is guaranteed by the following two sets of constraints.

$$h^t + \sum_{z \in T} \sum_{P' \in \mathcal{P}(z): s \in P'} x_{P'}^z \leq 1 \qquad t \in T,\ s \in P_0^t; \tag{2}$$

$$x_P^t + \sum_{z \in T \setminus \{t\}} \sum_{P' \in \mathcal{P}(z): s \in P'} x_{P'}^z \leq 1 \qquad t \in T,\ P \in \mathcal{P}(t),\ s \in P. \tag{3}$$

Once again, note that, since the input plan is feasible, we have that $P_0^t \cap P_0^z = \emptyset$ for any pair of distinct trains t, z.

Now, in order to fulfill objective O1, we want the solution to satisfy condition ii) of Theorem 3.1. Therefore, for each oriented arc $(\sigma(P_t), P_z)$ in the current selection $\overline{S}$, we forbid that the conditions $\overline{P}_t \preceq P_t$ and $\overline{P}_z \succeq P_z$ both hold true.

$$h^t + \sum_{P \in \mathcal{P}(t) | P \preceq P_t} x_P^t + \sum_{P \in \mathcal{P}(z) | P \succeq P_z} x_P^z \leq 1 \qquad (\sigma(P_t), P_z) \in \overline{S}. \tag{4}$$

In order to implement condition O2, we need to introduce the binary variables that define the activation of the paths in $\mathcal{U}$.

$$y^Q = \begin{cases} 1 & \text{if path } Q \text{ is active} \\ 0 & \text{otherwise} \end{cases} \qquad Q \in \mathcal{U}$$

The following constraints imply that, whenever a train is assigned to safe place P, then at least one path in U^P must be reserved.

$$\sum_{Q \in U^P} y^Q - x_P^t \geq 0 \qquad t \in T,\, P \in \mathcal{P}(t); \tag{5}$$

Then, we need to ensure that all reserved paths do not intersect the safe place assigned to some train.

$$y^Q + x_P^t \leq 1 \qquad Q \in \mathcal{U},\, t \in T,\, P \in \mathcal{P}(t) : P \cap Q \neq \emptyset;$$

Since two assigned safe places cannot share a common segment, we can strengthen the previous constraints as follows

$$y^Q + \sum_{t \in T} \sum_{P \in \mathcal{P}(t) : s \in Q} x_P^t \leq 1 \qquad Q \in \mathcal{U},\, s \in Q. \tag{6}$$

Note that there is no analogous of constraints (5) and (6) involving the variables h instead of variables x, as halted trains do not need to satisfy condition O2.

The set of constrains (1)-(6), plus the binary stipulation on the variables x, h, y, defines the feasible set of solutions to the SPAP-ILP model. Observe that this set is non-empty since the solution $h^t = 1$ for all $t \in T$ (*all halted*) is always feasible.

As for the cost of a feasible solution, we consider two components. First recall that we want the trains to be parked in the farthest possible safe place. To this end, for train t, and each potential safe place $P \in \mathcal{P}(t)$ we associated a cost c_P^t that is proportional to the distance of P from the final destination of t. In other words, $c_{P'}^t < c_P^t$ for any $P' \in \mathcal{P}_{>P}(t)$. Next, we have a cost w^t for every halted train t. Since we want to minimize the number of halted trains, we choose $w^t >> c_P^t$ for any $P \in \mathcal{P}(t)$. Then, the objective function of the SPAP-ILP writes as the following cost minimization:

$$\min \sum_{t \in T} w^t\, h^t + \sum_{t \in T} \sum_{P \in \mathcal{P}(t)} c_P^t\, x_P^t. \tag{7}$$

Note that modelling the cost of a solution in this way allows us to achieve condition O3.

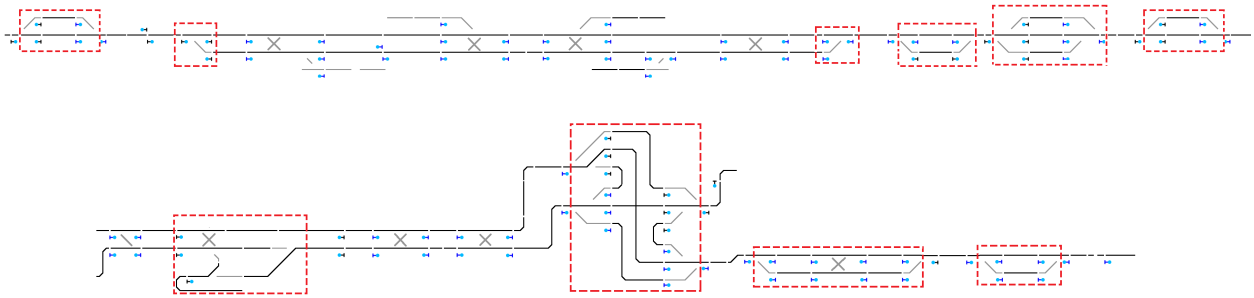


Figure 7 **Sample portions of railway network. Stations (within dotted lines) can be connected by a single or multiple (bidirectional) tracks. Stations can have a different number of tracks. There can be junctions between different lines .**

As anticipated while defining the three objectives, this formulation can be modified to incorporate different requirements. For example, we may discourage to stop a passengers train on a potential safe place $\bar{P}$ where there is not an available platform by increasing costs $c^t_{\bar{P}}$.

In the next section we discuss our computational results over a set of instances of realistic size and complexity. Although MILPs are in general hard to solve in practice and in theory (see Garey and Johnson (1979)), all instances can be solved in very short computing time by a commercial solver. This can be partly due to the special structure of our MILP. Indeed, all the (family of) constraints but one are shared with the maximal independent set formulation (also called *node packing formulation*, see Nemhauser and Wolsey (1999)). The associated integer polytope has been much investigated in the literature, and most commercial solvers can exploit the special structure of the constraint matrix.

5. Computational experiments

In this section, we present the results obtained by applying the $SPAP$ formulation on sixty realistic instances based on data provided by a class 1 North-American railroad. The underlying network alternates single track and multiple track stretches. Tracks are bidirectional. The number of routes within one station may vary: most stations have two bidirectional tracks, but some stations may have three or more of them. Since the data comes from a wide area of the railway network, which comprises almost 300 stations[5], in Figure 7 we show only a couple of sample portions. In particular, note that the network also includes junctions between different lines. Distance between adjacent stations may vary considerably, from 6000 to 100000 ft in single track areas, while in multiple track areas they are usually shorter (mainly spanning from 2000 to 20000 ft). The trains in the network are mostly freight trains.

[5] although not all stations appear in every instances, as some stations are not traversed by any train during the time interval considered

Instances. Our test-bed allows us to highlight the proactive and reactive aspect of our formulation. As discussed in the introduction, we encounter the SPAP mainly in two application contexts:

(i) *disruption management*, when a major blockage occurs in the network;

(ii) *partial plan recovering*, when an ATMS fails to produce a solution for the entire planning horizon.

Accordingly, we generated two sets of instances. The first set $\mathcal{S}_1 = \{I_1, ..., I_{10}\}$ of instances is related to case (i). Each instance in $\mathcal{S}_1$ is relative to a small set of trains approaching an area affected by a disruption, in a limited time frame of two hours. The second set $\mathcal{S}_2 = \{I_{11}, ..., I_{60}\}$ refers to case (ii), i.e. includes instances for which the ATMS only provides a plan with a time horizon that is smaller than the default planning horizon.

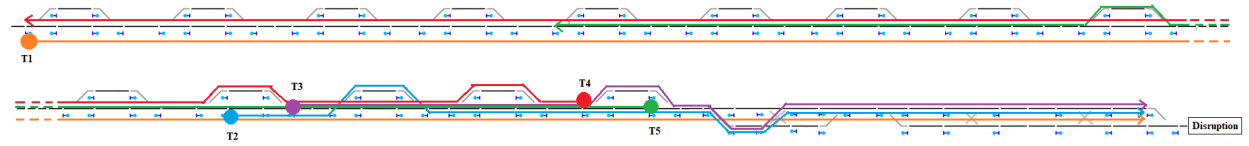
All instances refer to a network with both single and multiple track areas. As already mentioned, here the set $\mathcal{U}$ of paths that can be used to bypass the trains parked in the safe places is given explicitly. Moreover, for each train t, we are given its planned route. Therefore, we can easily calculate the set $\mathcal{P}(t)$ of the potential safe places of t- it is sufficient to enumerate all blocking paths of t along its route. Then, for each $P \in \mathcal{P}(t)$, we construct $U^P \subseteq \mathcal{U}$ by considering all paths of $\mathcal{U}$ that bypass P (see goal O2 in Section 3). Observe that, in a directed graph G, a u-v path Q bypasses P if $G \setminus Q$ admits a u-v path that contains P.

The dataset $\mathcal{S}_1$ consists of ten instances derived by network disruption occurrences, and presents on average 4.8 trains per instance. The average number of potential safe place for each train is 8.6. The dataset $\mathcal{S}_2$ contains instances having from a minimum of 11 to a maximum of 59 trains. Here the number of possible safe place positions per train varies between 2 and 13.

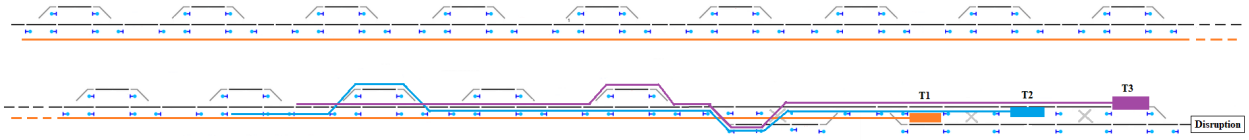
Settings. Since we want to park the trains as close as possible to their final destination (goal O3), we let $c_P^t = K \cdot (K - i_P^t)$, where K is the maximum number of available safe places among all trains and i_P^t indicates the index position of the safe place segment P (the lower the index is, the closer P is to P_0^t). Moreover, as our main objective is to minimize the number of halted trains, in the numerical experiments we set the value $w^t = |T|/2 \cdot K^2$.

The experiments are performed using an x64 MS Windows 10 machine with an Intel (R) Core (TM) i7-10510U CPU and 16 GB of RAM. The algorithm is implemented in Java and uses *IBM ILOG CPLEX v12.10* (all default settings) as MIP optimizer.

Computational results. Figure 8 gives a graphical representation of the input and output of SPAP for instance I_7: as input we have the schedule for each train belonging to the instance (notice that the solid points show the current trains positions). Among the five trains, three of them are obstructed by a disruption and need a safe place, while the remaining trains are travelling in the opposite direction. The optimal safe place assignment (with solid rectangles) is shown in Figure 8b. In Table 1 and Table 2, we report the computational results obtained solving our formulation on



(a) Schedule



(b) Safe places assignment

Figure 8 **Instance I_7: the schedule for each train (above) and the safe places assignment (below) .**

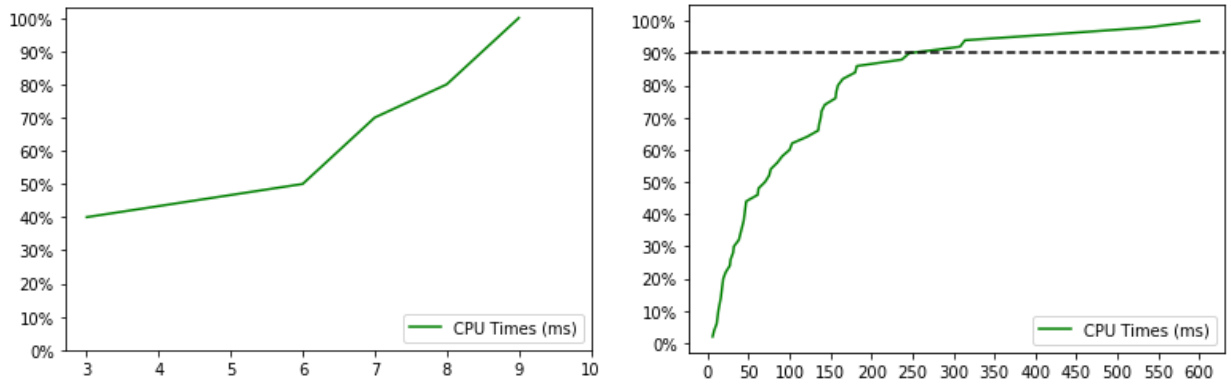


Figure 9 **Computational times cumulative distribution of dataset $\mathcal{S}_1$ instances (on the left) and $\mathcal{S}_2$ instances (on the right).**

the test instances of the set $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. The rows of the tables are associated with the instances of the corresponding dataset. Then, in the first columns, we give some detail about the instance - reporting the number of trains ($\#T$) and the average number of safe places per train ($\#SP$) - and the size of the corresponding LP model, with the number of variables ($\#Var$) and constraints ($\#Con$). Finally, in columns $\#H$ and $Time_{(ms)}$ we report the number of halted trains in the optimal solution and the CPU time needed to solve the formulation (expressed in milliseconds), respectively.

We first observe that, for the instances of $\mathcal{S}_1$, the CPU computational times are negligible. Then, about 90% of the instances of $\mathcal{S}_2$ require around 200 millisecond to be solved and only five need more than 300 ms to produce the final SP assignments. This proves that the proposed method can be very efficiently applied in a real-time setting. The cumulative distribution of the computational times of the instances of $\mathcal{S}_1$ and $\mathcal{S}_2$ are presented in Figure 9.

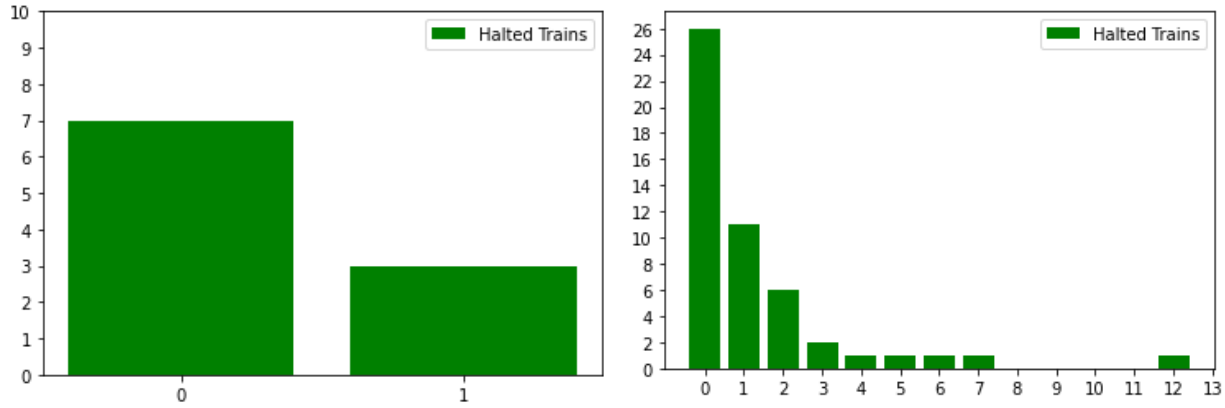


Figure 10 **Distributions of halted trains per instance for dataset $\mathcal{S}_1$ (on the left) and $\mathcal{S}_2$ (on the right).**

We also want to remark that all instances are solved at the root node of the branch-and-bound approach used by the solver. This proves the good quality of the linear relaxation of the MILP model, that is the strength of our formulation.

In Figure 10, we show, for the two groups $\mathcal{S}_1$ and $\mathcal{S}_2$, the cumulative distribution of the instances according to the number of halted trains in the optimal solution. In particular, for dataset $\mathcal{S}_1$, the optimal solutions of about 70% of the instances do not halt any train. This is somehow surprising, since such instances are derived from network disruption situations. Such a percentage is about 50% for the instances of $\mathcal{S}_2$. These numbers could be explained by the fact that the number of involved trains in $\mathcal{S}_1$ is much smaller, which in turn implies: i) that the number of interactions (meets and precedence constraints) is smaller and ii) that the network is less congested, which increases the probability of fulfilling condition O2. Besides, in most of the instances of $\mathcal{S}_2$ (i.e. derived from a partial plan), the trains that are set as halted by the optimal solution have a very small number of available safe places. We also point that, aside from instances I_{47} and I_{48}, in all other cases the halted trains comply with the conditions of either been trailing trains or not having potential interactions with each other (as described in Section 3).

6. Conclusions

In this paper, we present a non-naive formulation for the *Safe Place Assignment Problem*, a problem that arises in the context of railway disturbance and disruption management. Trains must be assigned a location to "safely" park under certain circumstances, such as when a disruption occurs on the network or when the ATMS produces a plan that can only partly be carried out. In Section 5 we showed that realistic instances can be solved in a very small amount of time, giving evidence that the proposed method can be effectively applied also in a practical setting, where (quasi) real-time performance is required. It is perhaps worth noting that the special structure of the model could be among the reasons behind the extremely low computational time. Indeed, as we observed in Section 4, the formulation shares all constraints except one with a maximal independent set problem,

Table 1 **Dataset $\mathcal{S}_1$: metrics and computational times results.**

Instance	#T	#SP	#Var	#Con	#H	Time(ms)
I_1	7	7	278	550	0	9
I_2	4	9	139	288	0	7
I_3	6	8	189	425	1	8
I_4	4	8	156	232	0	6
I_5	5	10	221	367	0	7
I_6	4	10	148	302	1	3
I_7	5	9	236	546	0	9
I_8	4	10	206	390	0	3
I_9	4	9	168	264	0	3
I_{10}	5	6	149	268	1	3

Table 2 **Dataset $\mathcal{S}_2$: metrics and computational times results.**

Instance	#T	#SP	#Var	#Con	# H	Time(ms)	Instance	#T	#SP	#Var	#Con	# H	Time(ms)
I_{11}	54	7	2873	6628	0	429	I_{36}	48	6	1479	2863	0	28
I_{12}	47	12	3261	9020	0	246	I_{37}	40	12	1864	5394	0	47
I_{13}	50	10	3347	8648	0	103	I_{38}	42	10	2075	4933	0	45
I_{14}	45	13	3027	8265	1	136	I_{39}	49	11	2686	7075	1	135
I_{15}	50	8	2835	6683	0	538	I_{40}	40	11	2458	5831	0	77
I_{16}	54	10	3203	8552	1	165	I_{41}	45	11	2467	6469	1	121
I_{17}	46	6	2408	5188	1	44	I_{42}	49	10	2644	7291	0	100
I_{18}	54	10	3176	8637	0	159	I_{43}	59	11	3079	8742	2	139
I_{19}	53	4	1828	3674	0	600	I_{44}	46	11	2610	7128	1	138
I_{20}	53	3	1920	3692	0	42	I_{45}	49	11	2765	7398	1	157
I_{21}	52	11	3946	10596	0	180	I_{46}	50	6	2091	4529	0	61
I_{22}	43	12	3911	10149	0	156	I_{47}	23	2	407	743	7	6
I_{23}	42	12	4071	11170	0	143	I_{48}	29	4	730	1183	5	19
I_{24}	44	11	3221	7904	0	182	I_{49}	11	5	298	621	2	85
I_{25}	59	6	1231	3455	1	314	I_{50}	31	5	997	1588	2	12
I_{26}	51	6	1149	3085	1	308	I_{51}	31	6	1041	1747	2	22
I_{27}	47	6	1112	2732	0	237	I_{52}	29	5	707	1423	6	18
I_{28}	46	11	1987	4229	0	31	I_{53}	26	7	1169	2064	3	16
I_{29}	41	11	2036	4271	0	32	I_{54}	32	6	1073	1807	2	13
I_{30}	57	11	2654	6685	0	70	I_{55}	27	12	2021	6325	1	38
I_{31}	53	10	2174	5105	0	47	I_{56}	44	5	1229	2070	4	27
I_{32}	57	11	2502	6895	0	91	I_{57}	33	7	1039	1847	3	18
I_{33}	43	12	2291	5320	0	42	I_{58}	15	6	559	973	0	16
I_{34}	45	11	1956	5779	1	75	I_{59}	16	4	405	684	2	11
I_{35}	53	9	2147	5777	0	62	I_{60}	27	3	596	983	12	8

that is, it presents a structure that most math programming solvers are able to exploit to improve performances.

As possible future research directions, we consider the case of modeling tracks with a capacity of two or more trains as well as the possibility of partially rerouting some of the trains. The relative closeness to the much studied maximal independent set problem could also inspire alternative solution methods or further work to strengthen the formulation.

References

Aurizon, 2020 *Second ertms pilot line commissioned. Railway Gazette International* URL `https://www.railwaygazette.com/infrastructure/second-ertms-pilot-line-commissioned/55385.article`, last Visited: 7 October, 2020.

Balas E, 1969 *Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. Operations research* 17(6):941–957.

BaneNOR, 2018 *The digital railway – bane nor signs contract with thales.* URL `https://www.banenor.no/Prosjekter/prosjekter/ertms/innhold/2018/the-digital-railway--bane-nor-signs-contract-with-thales/`, last Visited: 20 May, 2021.

BaneNOR, 2021 *Bane nor action cards.* URL `https://orv.banenor.no/orv/doku.php?id=sidebar&id=avvikshandtering:aksjonskort`, last Visited: 20 May, 2021.

Bešinović N, 2020 *Resilience in railway transport systems: a literature review and research agenda. Transport Reviews* 40(4):457–478, URL `http://dx.doi.org/10.1080/01441647.2020.1728419`.

Bollapragada S, Markley R, Morgan H, Telatar E, Wills S, Samuels M, Bieringer J, Garbiras M, Orrigo G, Ehlers F, et al., 2018 *A novel movement planner system for dispatching trains. Interfaces* 48(1):57–69.

Borraz-Sánchez C, Klabjan D, Uygur A, 2020 *An approach for the railway multiterritory dispatching problem. Transportation Science* 54(3):721–739.

Cacchiani V, Huisman D, Kidd M, Kroon L, Toth P, Veelenturf L, Wagenaar J, 2014 *An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological* 63:15–37, URL `http://dx.doi.org/10.1016/j.trb.2014.01.009`.

Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A, 1998 *Combinatorial Optimization* (USA: John Wiley &; Sons, Inc.), ISBN 047155894X.

Corman F, D'Ariano A, Hansen IA, Pacciarelli D, Pranzo M, 2011 *Dispatching trains during seriously disrupted traffic situations. 2011 International Conference on Networking, Sensing and Control*, 323–328, URL `http://dx.doi.org/10.1109/ICNSC.2011.5874901`.

Corman F, 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.

Dal Sasso V, Lamorgese L, Mannino C, Onofri A, Ventura P, 2021 *The tick formulation for deadlock detection and avoidance in railways traffic control. Journal of Rail Transport Planning & Management* 17:100239.

DSB, 2020 *Second ertms pilot line commissioned. Railway Gazette International* URL `https://www.railwaygazette.com/infrastructure/second-ertms-pilot-line-commissioned/55385.article`, last Visited: 7 October, 2020.

Fang W, Yang S, Yao X, 2015 *A survey on problem models and solution approaches to rescheduling in railway networks. IEEE Transactions on Intelligent Transportation Systems* 16(6):2997–3016.

Garey MR, Johnson DS, 1979 *Computers and intractability. A guiged to the theory of NP-completeness* (WH Freman and Company, San Francisco).

Ghaemi N, Cats O, Goverde RM, 2017a *A microscopic model for optimal train short-turnings during complete blockages. Transportation Research Part B: Methodological* 105:423 – 437, URL `http://dx.doi.org/https://doi.org/10.1016/j.trb.2017.10.002`.

Ghaemi N, Cats O, Goverde RMP, 2017b *Railway disruption management challenges and possible solution directions. Public Transport* 9(1):343–364, URL `http://dx.doi.org/10.1007/s12469-017-0157-z`.

Hirai C, Kunimatsu T, Tomii N, Kondou S, Takaba M, 2009 *A train stop deployment planning algorithm using a petri-net-based modelling approach. Quarterly Report of Rtri* 50:8–13, URL `http://dx.doi.org/10.2219/rtriqr.50.8`.

Josyula SP, Törnquist Krasemann J, 2017 *Passenger-oriented railway traffic re-scheduling: A review of alternative strategies utilizing passenger flow data. 7th International Conference on Railway Operations Modelling and Analysis, Lille.*

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

Louwerse I, Huisman D, 2014 *Adjusting a railway timetable in case of partial or complete blockades. European Journal of Operational Research* 235, URL `http://dx.doi.org/10.1016/j.ejor.2013.12.020`.

Mannino C, Lamorgese LC, Piacentini M, 2017 *Integer programming techniques for train dispatching in mass transit and main line. Advances and trends in optimization with engineering applications* .

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

Nemhauser GL, Wolsey LA, 1999 *Integer and combinatorial optimization*, volume 55 (John Wiley & Sons).

Pinedo M, 2009 *Planning and scheduling in manufacturing and services* (Springer).

Schöbel A, 2007 *Integer programming approaches for solving the delay management problem. Algorithmic methods for railway optimization (pp. 145–170). Berlin: Springer* 145–170.

Schöbel A, 2009 *Capacity constraints in delay management. Public Transport* 1:135–154, URL `http://dx.doi.org/10.1007/s12469-009-0010-0`.

Veelenturf L, Kidd M, Cacchiani V, Kroon L, Toth P, 2016 *A railway timetable rescheduling approach for handling large scale disruptions. Transportation Science* 50:841–862, URL `http://dx.doi.org/10.1287/trsc.2015.0618`.

Wen C, Huang P, Li Z, Lessan J, Fu L, Jiang C, Xu X, 2019 *Train dispatching management with data-driven approaches: a comprehensive review and appraisal. IEEE Access* 7:114547–114571.

Zhan S, Kroon LG, Veelenturf LP, Wagenaar JC, 2015 *Real-time high-speed train rescheduling in case of a complete blockage. Transportation Research Part B: Methodological* 78:182 – 201, URL `http://dx.doi.org/https://doi.org/10.1016/j.trb.2015.04.001`.

Zhan S, Kroon LG, Zhao J, Peng Q, 2016 *A rolling horizon approach to the high speed train rescheduling problem in case of a partial segment blockage. Transportation Research Part E: Logistics and Transportation Review* 95:32–61.

Zhan S, Wong SC, Peng Q, Lo SM, 2019 *Train stop deployment planning in the case of complete blockage: An integer linear programing model. Journal of Transportation Safety & Security* 0(0):1–27, URL `http://dx.doi.org/10.1080/19439962.2019.1645773`.

Zhu Y, Goverde R, 2020 *Dynamic and robust timetable rescheduling for uncertain railway disruptions. Journal of Rail Transport Planning & Management* URL `http://dx.doi.org/10.1016/j.jrtpm.2020.100196`.

Zhu Y, Goverde RM, 2019 *Railway timetable rescheduling with flexible stopping and flexible short-turning during disruptions. Transportation Research Part B: Methodological* 123:149 – 181, URL `http://dx.doi.org/https://doi.org/10.1016/j.trb.2019.02.015`.