

From Routes to Resources: A Model for Real-Time Train Dispatching with No-Wait Constraints

Luka Stärk ✉ 

Zuse Institute Berlin, Germany
Freie Universität Berlin, Germany

Ralf Borndörfer ✉ 

Zuse Institute Berlin, Germany
Freie Universität Berlin, Germany

Niels Lindner ✉ 

Zuse Institute Berlin, Germany
Freie Universität Berlin, Germany

Abstract

Optimization for train dispatching promises to reduce delays. Yet scalable applications are challenging. We present an efficient MIP model, based on generalized disjunctive graphs, for the train dispatching problem with no-wait constraints. For each train, we model its possible movements with microscopic detail as a DAG of operations. We select one train path per DAG and associate it with reservation and release times for the exclusive infrastructure it uses, such that the resulting schedule is conflict-free. When the connecting velocity between consecutive operations is positive, no-wait constraints apply. Additionally, we give a procedure to transform realistic infrastructure constraints based on route exclusions into an equivalent representation on resources by means of edge clique covers and incorporation of overlap exclusions. This makes resource-based dispatching models applicable to real-life infrastructure. We embed the MIP into an iterative algorithm, which we evaluate on the DISPLIB benchmark set and on a real-life dataset from German dispatching regions that we publish together with this paper. We prove optimality for 70 of the 112 DISPLIB instances within short running times, and we find 27 new best known solutions within a 10-minute time limit.

2012 ACM Subject Classification Applied computing → Transportation; Mathematics of computing → Combinatorial optimization; Theory of computation → Mixed discrete-continuous optimization

Keywords and phrases Train Dispatching, No-Wait Constraints, Disjunctive Graph, Edge Clique Cover, Microscopic Railway Timetabling

Digital Object Identifier 10.4230/OASICS.ATMOS.2026.6

Supplementary Material *Dataset:* <https://doi.org/10.5281/zenodo.21775303> [17]

Funding The work for this article has been conducted in the Research Campus MODAL funded by the German Federal Ministry of Research, Technology and Space (BMFTR), fund number 05M25KEM.

Luka Stärk: Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – The Berlin Mathematics Research Center MATH+ (EXC-2046/2, project ID: 390685689).

Acknowledgements We thank our industrial partner DB InfraGO AG, in particular the ADA-PMB team, for the kind permission to publish our benchmarking set based on real data. We also thank the three anonymous reviewers, whose comments improved the presentation of this paper.

1 Introduction

Real-time train dispatching is the continuous task of resolving conflicts and minimizing delays in an ever-evolving traffic situation where decisions must be made within seconds. This challenge arises when the carefully planned conflict-free timetable meets the unpredictable



© Luka Stärk, Ralf Borndörfer, and Niels Lindner;
licensed under Creative Commons License CC-BY 4.0

26th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2026).

Editors: Valentina Cacchiani and Stefan Funke; Article No. 6; pp. 6:1–6:20



OpenAccess Series in Informatics

OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

reality of delays and disruptions. Dispatchers and signalers are then responsible for making decisions to keep operations running and to prevent delays from spreading to other trains. Most railway networks are separated into dispatching regions where a group of dispatchers supervises the traffic in their region. The primary objective of dispatchers is to minimize delays by reordering and rerouting trains. Optimization-based decision-support tools for train dispatching have the potential to improve this process significantly by increasing the speed and quality of dispatching decisions and by considering solutions and their consequences over longer time periods where delays may propagate.

We consider the train dispatching problem with microscopic feasibility, velocity profiles (acceleration/deceleration curves), and infrastructure use that depends on the position and velocity of a train [18]. For each train, we build a directed acyclic graph of operations. If such an operation reflects a profile ending with positive velocity, no-wait constraints apply. To provide a physically feasible schedule, these constraints are crucial: a stopping train may not only change the release time of the last released resources, but also increase the running time during its acceleration for the next operation after stopping. Furthermore, the set of required resources of an operation depends on the velocity profile for that operation, as the train reserves the next route earlier when driving at higher velocity.

In this work we present three contributions. First, we show how to transform infrastructural exclusions of routes and overlaps into an abstract resource representation. Second, we define TDP-NW, the *Train Dispatching Problem with No-Wait Constraints*, which we model as a MIP based on generalized disjunctive graphs. The core is to select one path of operations per train and the timings along this path such that the selection is conflict-free w.r.t. the resources. In particular, we allow for paths that reflect re-routing of trains. Our resource representation of the route exclusions allows for a simple conflict-detection oracle, by checking for overlapping time intervals per resource. Introducing dedicated reservation and release nodes, whose timing is determined by level-aggregated resource timing constraints, we develop a MIP formulation and an iterative algorithm that is capable of effectively solving realistic instances at microscopic detail. Third, we evaluate our methods on the DISPLIB instances [9] and on a curated dataset of TDP-NW instances based on three German dispatching regions that we make publicly available [17].

We discuss some related work in Section 2. Our transformation from route-exclusion matrices to resources is described in Section 3. The central problem TDP-NW is defined in Section 4, followed by its MIP formulation in Section 5. Our benchmark set and computational results are illustrated in Section 6, before we conclude the paper in Section 7.

2 Related Work

While significant research exists on train dispatching [2], only recently have mathematical optimization algorithms been deployed in real-life applications, such as those of [18, 3]. For the few real-time dispatching algorithms we can see in the industry, their major challenge is scalability in time and space, while ensuring a short optimization time. The spatial dimension refers to the size of the infrastructure segment under consideration.

Definitions of train dispatching problems with and without rerouting in the literature mostly apply formulations of Job-Shop Scheduling Problems with blocking constraints [14]. Different optimization techniques have been explored for solving this problem. The first custom branch-and-bound algorithm restricted to reordering was proposed by D’Ariano et al. [4]. Many approaches use MIP formulations that rely on the big- M trick for disjunctive constraints. To avoid big- M constraints, a non-compact MIP formulation has been developed

that iteratively adds constraints for paths and cycles [10]. The literature becomes sparser when it comes to rerouting. Only recent works have investigated the problem with routing alternatives [11, 9, 16, 1, 5]. Rerouting alternatives are crucial for real-time train dispatching optimization. However, they may lead to increased complexity. One way to reduce this complexity in practice is to use a macro-micro decomposition, where a subnetwork, such as a station, is collapsed into a macroscopic resource that can accommodate more than one train at a time [13].

3 From Routes to Resources

As in related work based on job-shop scheduling, our train dispatching model in Section 4 relies on an infrastructure model, where a train reserves and releases exclusive *resources* along its path. Automatic train protection systems such as ETCS (European Train Control System) implement exclusive infrastructure usage defined for *routes*. A route, also called *block* on the free track, is a directed path between two main signals. A train reserves a route, e.g., before it reaches the EBI (emergency brake intervention point) towards the main signal, and releases or partially releases a route once the train passes a release contact with its last axle. The information which routes cannot be used simultaneously is often given by *route-exclusion matrices*, defining infrastructure resources only implicitly.

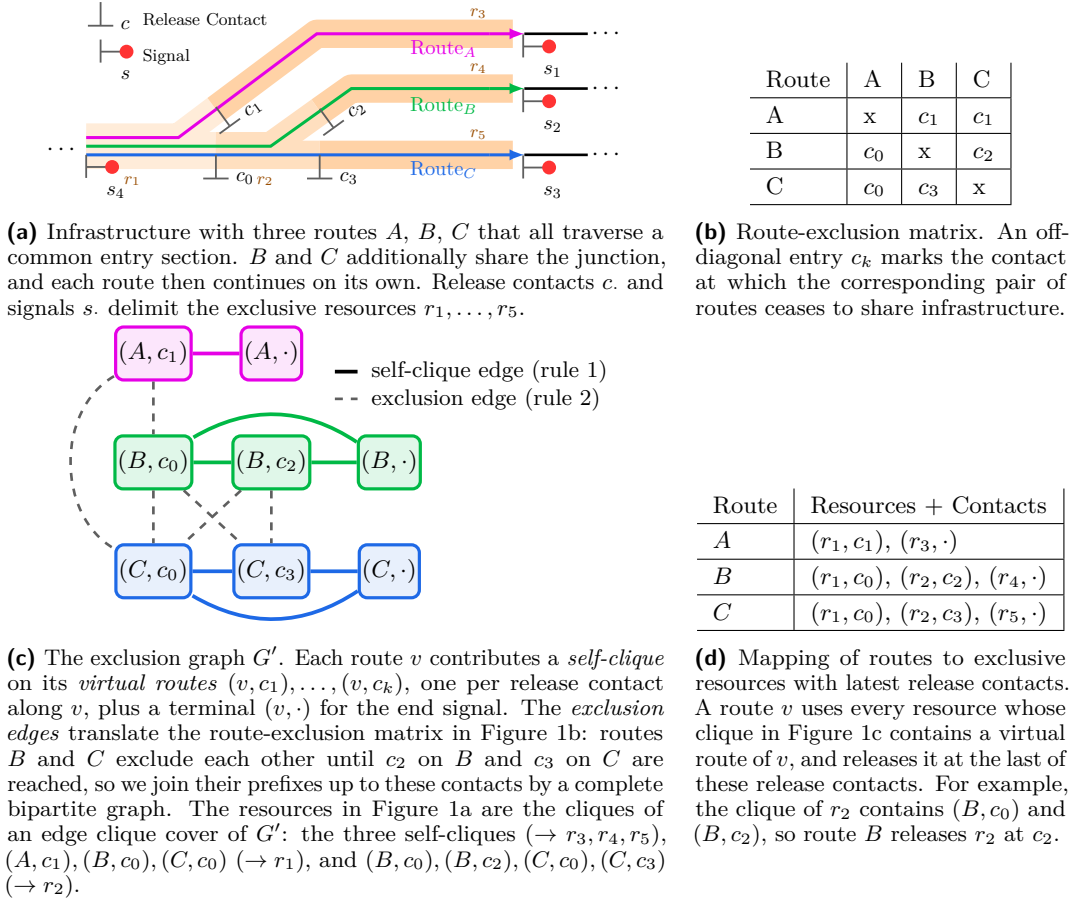
In this section, we describe a procedure to convert route-exclusion matrices to *resources*, hence offering a flat representation of exclusive infrastructure, in which every resource may be used by at most one train at a time. This not only enables us to encapsulate a microscopic level, but also simplifies the check for conflicting resource allocations. Instead of evaluating the route-exclusion matrix for every pair of excluding routes, we collect all overlapping time intervals per resource.

Example. Before providing the formal details, we will first illustrate the conversion of routes to resources in a small example, see Figure 1.

Overlaps. Route exclusions can mostly be derived from physical track intersections. However, there are special cases such as exclusions due to flank protection and overlap exclusions [15]. A train that will stop in front of a main signal additionally claims an *overlap*: a safety margin of track beyond the signal that must be free before the train is allowed to reserve this route, so that it can come to a safe halt even if it overruns the signal. The overlap is released once the train comes to a halt. Figure 2 shows an example situation of two disjoint parallel routes *A* and *B*, e.g., two station tracks. The sections from the signals through the switch up to the release contact c_0 are the *overlaps* of routes *A* and *B*: they are the safety overlap a train of route *A* or *B* claims when it intends to stop at s_1 or s_2 , and at the same time the shared first part of routes *C* and *D*. These two overlaps will become an *overlap resource* r_o in the resource representation for the overlap routes of *A* and *B*. Unlike an ordinary resource usage, an overlap reservation blocks other trains from using r_o as part of their *route*, but it does not prevent another train from claiming the same section as its *overlap*.

3.1 Minimum Resource Representation via Edge Clique Cover

We formalize the problem of computing a minimum set of resources from a route-exclusion matrix as follows. We are given a finite set of release contacts C and a symmetric directed graph $G = (V, A)$, where the nodes V are the routes and the arcs A encode the exclusion relations. The graph is symmetric in the sense that $(u, v) \in A \iff (v, u) \in A$. An arc



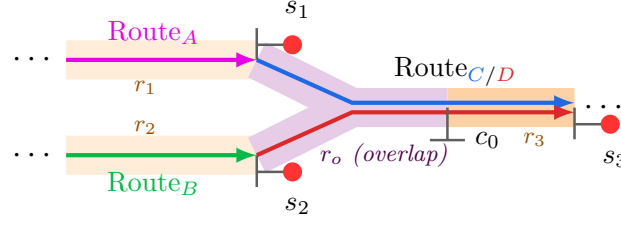
■ **Figure 1** Route to resources conversion for three routes A , B , C sharing track sections.

labeling $\rho: A \rightarrow C \cup \{\cdot\}$ gives, for an arc (u, v) , the release contact $\rho(u, v)$ at which a train on route u (partially) releases its exclusion with route v ; the special symbol $\cdot$ denotes a part of a route that is held until its end signal (more accurately until a release contact after the main signal is passed by the train, which is an implementation detail). Note that the graph is symmetric in structure but the release contacts of the arcs are not, i.e., in general $\rho(u, v) \neq \rho(v, u)$. For each route v we are further given the sequence $\sigma(v) = (c_1, \dots, c_k, \cdot)$ of its partial release contacts, ordered along the direction of travel, and where we append a terminal element “ $\cdot$ ”. We write $c \preceq_v c'$ for the induced order. Finally, let $V_o \subseteq V$ be the set of overlap routes and $V_r = V \setminus V_o$ be the set of regular routes, where each overlap route is associated to exactly one regular route.

Virtual routes. We construct an auxiliary undirected graph $G' = (V', A')$. For every regular route $v \in V_r$ we create a *virtual route* $(v, c) \in V'$ for each $c \in \sigma(v)$. For convenience we write $\text{pre}(v, c) := \{(v, c') \in V' : c' \preceq_v c\}$ for the *prefix* of virtual routes of v up to and including c .

Translating exclusions. The edges A' are derived from A by two rules:

1. (*self*) For every regular route $v \in V_r$, all of its virtual routes $\text{pre}(v, \cdot)$ form a clique in G' .
2. (*exclusion*) For every exclusion pair $\{(u, v), (v, u)\} \subseteq A$ between regular routes, with release contacts $c_u = \rho(u, v)$ and $c_v = \rho(v, u)$, we add all edges of the complete bipartite graph between $\text{pre}(u, c_u)$ and $\text{pre}(v, c_v)$.



■ **Figure 2** Overlap routes and resources. Routes A and B end at signals s_1 and s_2 ; the continuation routes C and D run from s_1 and s_2 through the switch to the common exit signal s_3 . The release contact c_0 lies behind the switch. Each incoming route is associated with a resource (r_1, r_2) ; the section from the signals to c_0 is an overlap resource r_o for A and B ; the outgoing track from c_0 to s_3 is r_3 .

Because each prefix is already a clique (rule 1), rule 2 turns the entire shared region $\text{pre}(u, c_u) \cup \text{pre}(v, c_v)$ into a clique, representing the whole piece of infrastructure two routes share. Note that we can include the prefixes, as the full route must be reserved, and then it can be partially released at the release contacts. To preserve all exclusions, every edge of G' has to be covered by at least one resource, which is an *edge clique cover* of G' : a set of cliques such that every edge is contained in at least one of them. To obtain a representation with a minimum number of resources, we are therefore interested in finding an edge clique cover of G' of minimum size.

Realizing overlap exclusions. An overlap exclusion is an extension beyond the end signal of the regular route, excluding routes to which the regular route has no exclusive relationship. Overlap exclusions apply when a train plans to stop at the end of the route. To cover the overlap exclusions between routes, we create a set S_{v_o} of excluded virtual routes for each overlap route $v_o \in V_o$. If there is an exclusion with route u , we add the virtual routes $\text{pre}(u, c)$ with $c = \rho(u, v_o)$ to S_{v_o} . However, an overlap route usually has several overlap exclusions, so that an overlap route may be connected to several cliques in an edge clique cover, but not necessarily to all virtual routes in any such clique. If the overlap exclusions of v_o do not exclude the whole clique, we cannot assign the clique (resource) to the overlap route; instead we need to search for a clique that is a subset of the excluded virtual routes, or we create a new sub-clique. As an overlap route has no virtual routes of its own, it is never a member of a clique, but only receives cliques as overlap resources. Since there are no overlap-overlap exclusions, we do not need to worry about accidentally creating a new exclusion between overlap routes when assigning cliques as overlap resources. Later, when checking for resource conflicts between trains by comparing the reservation intervals, we simply ignore intersecting intervals where both usages are overlap usages. The problem of finding a minimum representation of overlap resources on top of an edge clique cover can hence be described as follows, where each overlap route forms a *target set* of its excluded virtual routes:

► **Definition 1** (Minimum Sub-Clique Cover Extension). *Let $G' = (V', A')$ be the exclusion graph with an edge clique cover $\mathcal{C}$, and let $S = \{S_1, \dots, S_m\}$ be a family of target sets $S_i \subseteq V'$. For each S_i find a set H_i of cliques of G' such that*

- *every clique is internal to its target set, i.e. $C \subseteq S_i$ for all $C \in H_i$.*
 - *the union of cliques covers the target set, $\bigcup_{C \in H_i} C = S_i$, i.e. $\forall v \in S_i \exists C \in H_i : v \in C$.*
- Cliques already present in $\mathcal{C}$ may be reused at no cost. Minimize the number of newly created sub-cliques, counted globally across all targets, $|\bigcup_i H_i \setminus \mathcal{C}|$.*

3.2 Algorithm From Routes To Resources

Our algorithm from routes to resources is a two-stage optimization with the objective to minimize the number of resources to represent the given route exclusions. We first compute a small edge clique cover of G' ; in practice our instances are solved completely (and optimally) using only the data-reduction rules 1 and 2 of [6]. Rule 1 removes isolated vertices and vertices with only covered edges, and rule 2 applies if, for an uncovered edge (u, v) , the common neighborhood of u and v induces a clique C ; that clique C is then added to the cover. As in Figure 1c, each clique becomes one exclusive resource, and a route v receives a resource for every clique that contains one of its virtual routes. The release contact recorded for that resource on v is the latest contact c with (v, c) in the clique, cf. Figure 1d. Then, we solve the Minimum Sub-Clique Cover Extension problem greedily by covering each target set S , corresponding to an overlap route v_o , with cliques, either reusing an existing clique that is contained in S , or otherwise creating a new sub-clique from the intersection of S with an existing clique that covers the most still-uncovered routes of S , until v_o shares a common clique with every excluded virtual route in S . The cliques (resources) referenced this way are labeled as overlap resources for v_o .

To put our approach into perspective, Table 1 shows, for three dispatching regions, the size of the exclusion graph G' and the number of cliques (resources) found. The edge clique cover is solved completely by the reduction rules alone, and the overlap exclusions add a moderate number of extra sub-cliques on top of it. This comes as no surprise, since the exclusions are local relations, and G' typically consists of nearly disconnected components, representing, e.g., stations.

■ **Table 1** Size of the exclusion graph G' and number of exclusive resources (cliques) for three dispatching regions, showing the number of routes, overlap routes, and virtual routes $|V'|$. The edge clique cover (ECC) is computed by the data-reduction rules of [6]. Covering the overlap exclusions adds the sub-cliques, yielding the total number of exclusive resources. Note that these numbers do not reflect a whole dispatching region but only the routes used in a specific instance. The instances used as examples are `fbzm_large_dh_90_0`, `mbzs_large_dh_60_0`, `rbzs_dh_90_0` [17].

Region	Routes	Overlap routes	$ V' $	ECC cliques	Sub-cliques	Total
FBZM	1160	1342	4547	946	525	1471
MBZS	336	248	934	289	156	445
RBZS	841	851	3330	723	388	1111

4 Problem Definition

We consider the *Train Dispatching Problem with No-Wait Constraints* (TDP-NW), which can be seen as a variation of the DISPLIB problem defined in [9] and to some degree as a generalization. An instance consists of a set of trains I that share a common set of resources $U = \{0, \dots, |U| - 1\}$. Each train must be routed and scheduled through its own graph of operations, and the schedules of different trains interact only through their competition for the exclusive resources. The task is to choose, for every train, a path P^i together with a timing $t_v \in \mathbb{R}$ of its operations $v \in P^i$ that avoids resource conflicts while minimizing delay and operation costs.

Operations. The possible movements of a train i are modeled by a directed acyclic graph $G^i = (V^i, A^i)$ whose nodes $v \in V^i$ are the *operations* of the train and whose arcs $(v, u) \in A^i$ encode the successor relation. Each train has a single source operation v_0^i and target operation $v_{n_i}^i$, and a train path is a $v_0^i - v_{n_i}^i$ path in G^i . An operation v has the following attributes:

- a *duration* $\delta_v \in \mathbb{R}_{\geq 0}$, the running time of the operation;
- optional *start-time bounds* $[\alpha_v, \beta_v]$ on the time $t_v \in \mathbb{R}$ at which the operation v starts;
- a *penalty* $\lambda_v \in \mathbb{N}_0$ incurred whenever v lies on the chosen path;
- a *no-wait* flag: if set, every selected successor u must start exactly when v ends, $t_v + \delta_v = t_u$; otherwise waiting is allowed, $t_v + \delta_v \leq t_u$;
- a set of *resource usages*.

Reservations, releases, resource usages. An operation v interacts with the exclusive resources through *resource usages*. A resource usage refers to a resource $r_x \in U$ and may carry a *reservation* offset $\delta_v^{\text{rv}} \in \mathbb{R}$ and/or a *release* offset $\delta_v^{\text{rl}} \in \mathbb{R}_{\geq 0}$, given relative to the start time t_v of the operation: at $t_v + \delta_v^{\text{rv}}$ the train reserves r_x , and at $t_v + \delta_v^{\text{rl}}$ it releases it again. We assume that, along any route, every reservation of a resource has a matching release downstream (or the resource stays reserved until the target operation, modeling indefinite occupation), and every release has a matching reservation upstream. A resource that is never reserved on the chosen path is unused. After a release it may happen that the same resource is reserved again, for instance when a train reverses direction. A usage may additionally be flagged as an *overlap* (see Section 3).

Objective components. Each train has a set $\mathcal{D}^i$ of *objective components*. An objective component is defined for a set of mutually alternative operations, i.e., operations forming a source-to-target node-cut of G^i , such that exactly one of them lies on any train path, or the source or target itself. Each objective component c has a *threshold* $\theta_c \in \mathbb{R}$ and a *coefficient* $\gamma_c \in \mathbb{R}_{\geq 0}$. Let v be the operation of the component on the chosen train path. The component's cost is then

$$\text{cost}_c := \gamma_c \cdot \max(0, t_v + \delta_v - \theta_c). \quad (1)$$

Note that the delay is measured at the arrival of v and not the departure, which is a difference compared to the DISPLIB definition [9]. The reason is that we have different durations for the arrival operations but a single threshold per objective component, e.g., the trains' planned arrival time. We cannot measure the delay at the successor operations, since a train may wait longer than the operations already account for, e.g., longer than the minimum dwell time at a commercial stop. Alternatively, one could introduce an operation for the stop with the duration of the minimum dwell time.

► **Definition 2** (Train Dispatching Problem with No-Wait Constraints (TDP-NW)). *Select, for each train, a path from source to target in the DAG and assign finite start times to the selected operations such that:*

1. **Time windows** – start time bounds $\alpha_v \leq t_v \leq \beta_v$ are respected.
2. **Successor constraints** – for no-wait arcs: $t_v + \delta_v = t_u$; otherwise: $t_v + \delta_v \leq t_u$.
3. **Exclusive resource usage** – each resource $r_x \in U$ is used by at most one train at a time.
4. **Optimality** – the sum of the costs of all objective components and all operation penalties is minimized.

Even the decision variant of TDP-NW is NP-complete: restricted to a single path per train, it is job-shop scheduling with blocking and no-wait constraints, for which deciding whether a feasible solution exists is NP-complete [14].

5 Methodology

In the following we describe how we model the TDP-NW as a compact MIP using the terminology of *generalized disjunctive graphs* [12]. We consider a generalized disjunctive graph $D = (V \cup V_{rv} \cup V_{rl}, A \cup A_{lb} \cup A_{rv} \cup A_{rl} \cup A_d)$, where V are the operations, V_{rv} and V_{rl} the reserving and releasing nodes, respectively. Moreover, A are the successor arcs, A_{lb} the lower-bound arcs, and A_{rv} and A_{rl} the reserving and releasing arcs, and finally A_d are the disjunctive or alternative arcs with both endpoints in $V_{rv} \cup V_{rl}$. The disjunctive graph is generalized in the sense that we select a source-to-target path for each train in addition to the disjunctive arcs. Given such a selection, the maximum bound applies among the incoming arcs of each node, as in the standard disjunctive graph.

Rather than tracking exclusive resource usage for each operation, we reason over each *resource conflict* through dedicated reservation and release time variables for each $v^{\text{in}} \in V_{rv}$ and $v^{\text{out}} \in V_{rl}$, see Figure 3. This construction was inspired by [12], where auxiliary variables model how resource conflicts depend on routing decisions in a disjunctive program, but this yields one variable per alternative operation. The same idea of auxiliary variables for a resource's reservation and release has been developed by [5], using the notion of chunks of operations with the same resource. For our MIP model we anchor start times at the *level* (topological depth) of each train's DAG rather than at individual operations, see Figure 4. Both choices substantially reduce the number of continuous variables, especially since the instances under study contain many alternative operations with very similar resource usages.

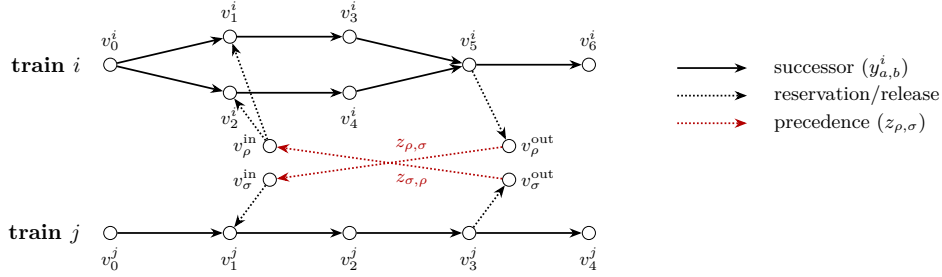


Figure 3 Resource reservation (v_ρ^{in}) and release (v_ρ^{out}) nodes for two conflicting resource usages ρ, σ of trains i, j .

Additional notation. We reuse the notation of Section 4.

- For a vertex v of a directed graph, $N^+(v) = \{u : (v, u) \in A\}$ denotes its *out-neighborhood* and $N^-(v) = \{u : (u, v) \in A\}$ its *in-neighborhood*.
- $S_a^i = \{v_b^i : (v_a^i, v_b^i) \in A^i\}$, $P_a^i = \{v_b^i : (v_b^i, v_a^i) \in A^i\}$ – the successors/predecessors of v_a^i .
- $A^{i,0} \subseteq A^i$ – the *waiting* arcs (e.g., leaving an operation with zero velocity); the remaining arcs $A^i \setminus A^{i,0}$ are *no-wait* arcs and enforce $t_b^i = t_a^i + \delta_a^i$.
- $\ell(v_a^i) \in \{0, \dots, L^i\}$ – the *level* of v_a^i , its topological depth in G^i ; all operations of the same commercial stop share a level. $V_l^i = \{v_a^i \in V^i : \ell(v_a^i) = l\}$ is the set of operations at level l , and $V_l^{i,nw} \subseteq V_l^i$ those with a no-wait out-arc. A level l is *optional* if some arc skips it, see also Figure 4.
- A *resource usage* of train i is a triple $\rho = (r, o, k)$, where $r(\rho) \in U$ is the resource, $o(\rho) \in \{0, 1\}$ its overlap flag, and $k(\rho) \in \mathbb{N}$ its *ordinal*: among all source-to- v -paths reaching a reserving operation v of the pair (r, o) , the ordinal is the maximum number of

re-reservations on any path. The ordinal distinguishes repeated reservations, e.g., a track section released and reserved again. We write R^i for the set of resource usages of train i , $\mathcal{R} = \bigsqcup_{i \in I} R^i$ for all usages across trains, and $i(\rho)$ for the train of a usage $\rho \in \mathcal{R}$.

- For each $\rho \in \mathcal{R}$, we dynamically create a single *reserving node* $v_\rho^{\text{in}} \in V_{rv}$ and *releasing node* $v_\rho^{\text{out}} \in V_{rl}$. The *reserving* and *releasing* operations, which anchor the reservation and release time of ρ , are denoted $\mathcal{R}_\rho^{\text{in}} = N^+(v_\rho^{\text{in}})$ and $\mathcal{R}_\rho^{\text{out}} = N^-(v_\rho^{\text{out}})$, respectively.
- $\mathcal{K}$ – the set of resource conflicts. A conflict $\{\rho, \sigma\} \in \mathcal{K}$ couples two usages $\rho, \sigma \in \mathcal{R}$ of different trains, $i(\rho) \neq i(\sigma)$, on the same resource $r(\rho) = r(\sigma)$ that are not both overlaps, i.e. $\neg(o(\rho) \wedge o(\sigma))$.
- $\mathcal{D} = \bigsqcup_{i \in I} \mathcal{D}^i$ – the objective components of Section 4 across all trains, where each component c belongs to a train $i(c)$. We write component $c \in \mathcal{D}$ as $(\ell_c, \mathcal{V}_c, \theta_c, \gamma_c)$: its anchor level ℓ_c , the operation cut $\mathcal{V}_c \subseteq V_{\ell_c}^{i(c)}$, the threshold θ_c , and the unit delay cost $\gamma_c \geq 0$.

5.1 MIP Model

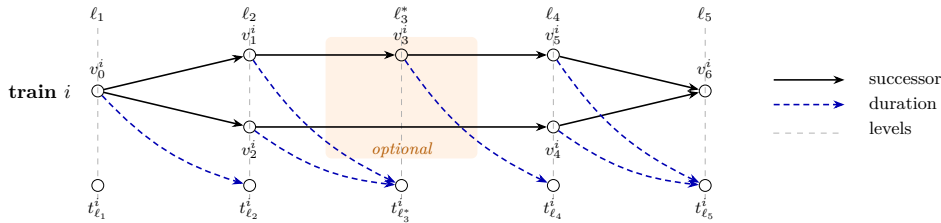
We express the disjunctive graph and the timings of the solution using the following variables:

- $t_l^i \in \mathbb{R}$, $\alpha_l^i \leq t_l^i \leq \beta_l^i$ – start time of level l of train i , where $\alpha_l^i := \min_{v \in V_l^i} \alpha_v$ and $\beta_l^i := \max_{v \in V_l^i} \beta_v$ are the level start-time bounds.
- $y_{a,b}^i \in \{0, 1\}$ – equals 1 iff arc $(v_a^i, v_b^i) \in A^i$ is used on its selected train path.
- $t_\rho^{\text{in}}, t_\rho^{\text{out}} \in \mathbb{R}$ – the reservation/release time of usage ρ , for each ρ appearing in a conflict.
- $x_\rho \in \{0, 1\}$ – equals 1 iff train $i(\rho)$ realizes usage ρ on its chosen path; introduced only when ρ is *optional* (otherwise $x_\rho \equiv 1$).
- $z_{\rho, \sigma} \in \{0, 1\}$ – equals 1 iff ρ is released before σ is reserved, for each conflict $\{\rho, \sigma\} \in \mathcal{K}$ (one per ordered pair).
- $w_c \in \mathbb{R}_{\geq 0}$ – the delay incurred by component $c \in \mathcal{D}$.

Unlike the classical disjunctive formulation, we do not introduce an explicit operation-selection variable x_a^i . As the flow constraints below force exactly one out-arc for any selected non-target operation, we write the selection of v_a^i as

$$x_a^i := \sum_{(v_a^i, v_b^i) \in A^i} y_{a,b}^i \in \{0, 1\}, \quad (2)$$

which equals 1 iff v_a^i is selected (the target $v_{n_i}^i$ is always selected). This eliminates the variables x_a^i and their linking constraints from the model.



■ **Figure 4** Level structure and duration constraints for train i . Level ℓ_3^* (shaded) is optional.

Path selection and level duration constraints. The flow constraints select one source-to-target path per train:

$$\sum_{v_b^i \in S_0^i} y_{0,b}^i = 1 \quad i \in I, \quad (3)$$

$$\sum_{v_a^i \in P_{n_i}^i} y_{a,n_i}^i = 1 \quad i \in I, \quad (4)$$

$$\sum_{v_a^i \in P_c^i} y_{a,c}^i = \sum_{v_b^i \in S_c^i} y_{c,b}^i \quad v_c^i \in V^i \setminus \{v_0^i, v_{n_i}^i\}, \quad i \in I. \quad (5)$$

For the duration constraint, we apply ideas of [5] on maximal and non-maximal clusters, which correspond to our definition of levels and optional levels. For each consecutive level pair $(l, l+1)$ of train i , the level start times respect the durations of the selected operations, allow waiting at waiting arcs, and are tight on no-wait arcs. Writing x_a^i as in (2):

$$t_{l+1}^i - t_l^i \geq \sum_{v_a^i \in V_l^i} \delta_a^i x_a^i \quad l \in \{0, \dots, L^i - 1\}, \quad i \in I, \quad (6)$$

$$t_{l+1}^i - t_l^i \leq \sum_{v_a^i \in V_l^{i, \text{nw}}} \delta_a^i x_a^i + M \left(1 - \sum_{v_a^i \in V_l^{i, \text{nw}}} x_a^i \right) \quad l \in \{0, \dots, L^i - 1\}, \quad i \in I, \quad (7)$$

$$t_{l+1}^i - t_l^i \leq M \sum_{v_a^i \in V_l^i} x_a^i \quad l \text{ optional}, \quad i \in I. \quad (8)$$

Constraint (7) activates the no-wait equality $t_{l+1}^i = t_l^i + \delta_a^i$ exactly for the selected no-wait operation, and (8) collapses an unused optional level onto its predecessor ($t_{l+1}^i \leq t_l^i$), keeping the no-wait chain transitive across skipped levels. A level start time t_l^i carries only the loosest lower bound among the operations sharing level l . When a selected operation v_a^i has a tighter bound $\alpha_a^i > \alpha_l^i$, we enforce it conditionally with the exact gap $M_a^i = \alpha_a^i - \alpha_l^i$, so the constraint is inactive unless v_a^i is selected:

$$t_{\ell(a)}^i \geq \alpha_a^i - M_a^i (1 - x_a^i) \quad v_a^i \in V^i : \alpha_a^i > \alpha_{\ell(a)}^i. \quad (9)$$

For the instances under study no symmetric constraint is needed for the upper bound, since upper bounds are only used for the source node v_0^i , so that $\beta_a^i = \beta_{\ell(a)}^i$. In general, the analogous constraint can be added for operations with $\beta_a^i < \beta_{\ell(a)}^i$.

Resource timing constraints. Consider a usage $\rho \in \mathcal{R}$ that appears in a conflict, and let $\mathcal{A}_\rho^{\text{res}} \subset A^{i(\rho)}$ be a $v_0^i - v_\rho^{\text{in}}$ -cut, the *indicator arcs* whose selection leads to the reservation of ρ . The resource indicator variable is then

$$x_\rho = \sum_{(v_a^i, v_b^i) \in \mathcal{A}_\rho^{\text{res}}} y_{a,b}^i \quad \rho \in \mathcal{R}, \quad (10)$$

for each $\rho \in \mathcal{R}$ whose $\mathcal{A}_\rho^{\text{res}}$ is not a $v_0^i - v_{n_i}^i$ -cut; otherwise we set $x_\rho \equiv 1$. The reservation and release times of a usage ρ are relative to the start times of the operations that reserve it first and release it last, and both the reserving operations $\mathcal{R}_\rho^{\text{in}}$ and the releasing operations $\mathcal{R}_\rho^{\text{out}}$ may lie on different levels of G^i . Let $L_\rho^{\text{rv}} = \{\ell(v_a^i) : v_a^i \in \mathcal{R}_\rho^{\text{in}}\}$ and $L_\rho^{\text{rl}} = \{\ell(v_a^i) : v_a^i \in \mathcal{R}_\rho^{\text{out}}\}$ be the levels with reserving and releasing operations. We anchor the timing per level: for every releasing level $l \in L_\rho^{\text{rl}}$ the release is bounded below by the start of level l plus the release offset of the releasing operation selected at that level, and symmetrically for every

reserving level $l \in L_\rho^{\text{rv}}$ the reservation is bounded above. A big- M term deactivates each constraint whenever no operation of ρ is selected at its level:

$$t_\rho^{\text{out}} \geq t_l^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{out}} \cap V_l^i} \delta_a^{\text{rl},i} x_a^i - M \left(1 - \sum_{v_a^i \in \mathcal{R}_\rho^{\text{out}} \cap V_l^i} x_a^i \right) \quad l \in L_\rho^{\text{rl}}, \rho \in \mathcal{R}, \quad (11)$$

$$t_\rho^{\text{in}} \leq t_l^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{in}} \cap V_l^i} \delta_a^{\text{rv},i} x_a^i + M \left(1 - \sum_{v_a^i \in \mathcal{R}_\rho^{\text{in}} \cap V_l^i} x_a^i \right) \quad l \in L_\rho^{\text{rv}}, \rho \in \mathcal{R}, \quad (12)$$

where $i = i(\rho)$. If $|L_\rho^{\text{rv}}| = 1$, then no big- M term is needed in (12). In Section 5.3 we show how to replace these by-level constraints with a single constraint for most usages ρ .

Resource conflict constraints. For each conflict $\{\rho, \sigma\} \in \mathcal{K}$, the orientation variables $z_{\rho, \sigma}$ and $z_{\sigma, \rho}$ select an order only if both usages are realized, as proposed before in [9]:

$$x_\rho \geq z_{\rho, \sigma} + z_{\sigma, \rho}, \quad x_\sigma \geq z_{\rho, \sigma} + z_{\sigma, \rho}, \quad (13)$$

$$x_\rho + x_\sigma - 1 \leq z_{\rho, \sigma} + z_{\sigma, \rho} \leq 1, \quad (14)$$

where a mandatory usage substitutes $x \equiv 1$. In particular, if both usages are mandatory, then (14) forces $z_{\rho, \sigma} + z_{\sigma, \rho} = 1$. The chosen order separates the conflicting reservations, enforced by big- M constraints that are non-binding when the orientation is inactive:

$$t_\sigma^{\text{in}} - t_\rho^{\text{out}} \geq -M(1 - z_{\rho, \sigma}) \quad \{\rho, \sigma\} \in \mathcal{K}, \quad (15)$$

$$t_\rho^{\text{in}} - t_\sigma^{\text{out}} \geq -M(1 - z_{\sigma, \rho}) \quad \{\rho, \sigma\} \in \mathcal{K}. \quad (16)$$

Objective. Each objective component lower-bounds its delay variable by the arrival time beyond the threshold:

$$w_c \geq t_{\ell_c}^{i(c)} + \sum_{v_a^{i(c)} \in \mathcal{V}_c} \delta_a^{i(c)} x_a^{i(c)} - \theta_c \quad c \in \mathcal{D}. \quad (17)$$

The objective minimizes the fixed operation penalties and the weighted delays:

$$\min \sum_{i \in I} \sum_{v_a^i \in V^i} \lambda_a^i x_a^i + \sum_{c \in \mathcal{D}} \gamma_c w_c. \quad (18)$$

5.2 Algorithm

■ **Algorithm 1** Conflict-driven decomposition for TDP-NW.

```

1: Initialize the disjunctive graph  $D$  from the train DAGs  $\{G^i\}_{i \in I}$ 
2:  $\mathcal{C} \leftarrow \emptyset$  ▷ active conflict set
3:  $\mathcal{S} \leftarrow \{\{i\} : i \in I\}$  ▷ one subproblem per train
4: while true do
5:   for  $S \in \mathcal{S}$  do ▷ in parallel
6:      $\text{SOLVEMIP}(S \mid \mathcal{C}, D)$ 
7:   end for
8:    $\mathcal{C}_{\text{new}} \leftarrow \text{CONFLICTDETECTION}(D)$  ▷  $D$  optimal w.r.t.  $\mathcal{C}$ 
9:   if  $\mathcal{C}_{\text{new}} = \emptyset$  then
10:    return current solution
11:   end if
12:    $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}_{\text{new}}$ 
13:   merge the subproblems in  $\mathcal{S}$  that are newly linked by  $\mathcal{C}_{\text{new}}$ 
14: end while

```

The orientation and separation constraints (13)–(16) couple two trains through a resource conflict $\{\rho, \sigma\} \in \mathcal{K}$, and in practice most conflicts are not relevant for solving the optimization problem. We solve the model iteratively, and in a decomposed fashion: the trains are grouped into independent subproblems, each subproblem is the compact model of Section 5.1 restricted to its own trains, and the constraints of a conflict are only added once that conflict is actually observed in an intermediate infeasible solution.

Conflict Detection. We implemented an oracle that, given the selected and timed operations for each train, returns for each resource $r \in U$ all pairs (ρ, σ) of conflicting usages, using algorithmic ideas of maximal cliques in interval graphs as done before in [8]. Due to potential sub-cliques introduced to cover overlap exclusions, see Section 3, a conflict between the same two trains may appear for two equivalent resources from the perspective of the two trains. For the conflict discovery and the given usage intervals, we test for domination among conflicts between the same two trains. A conflict $C' = \{\rho_i, \rho_j\}$ is dominated by another conflict $C'' = \{\sigma_i, \sigma_j\}$ of the same train pair if each train occupies its resource of C' within the time it occupies its resource of C'' , that is $[t_{\rho_i}^{\text{in}}, t_{\rho_i}^{\text{out}}] \subseteq [t_{\sigma_i}^{\text{in}}, t_{\sigma_i}^{\text{out}}]$ and $[t_{\rho_j}^{\text{in}}, t_{\rho_j}^{\text{out}}] \subseteq [t_{\sigma_j}^{\text{in}}, t_{\sigma_j}^{\text{out}}]$. The first processed conflict wins in tie-breaking situations. We collect the conflicts for each train pair by resource $r \in U$ in ascending order, so that resources from sub-cliques are processed last and we only create a conflict for a sub-clique resource if one of the participating trains uses that resource as an overlap.

Starting from one subproblem per train and an empty conflict set, Algorithm 1 repeatedly solves all subproblems and inspects the combined solution for conflicts $\{\rho, \sigma\} \in \mathcal{K}$ that are violated and not yet represented. Each such conflict is added to $\mathcal{C}$ and the subproblems of the respective trains are potentially merged. The algorithm terminates when no new conflict is found, at which point the combined solution is feasible and optimal for the full model.

5.3 Strengthened Model

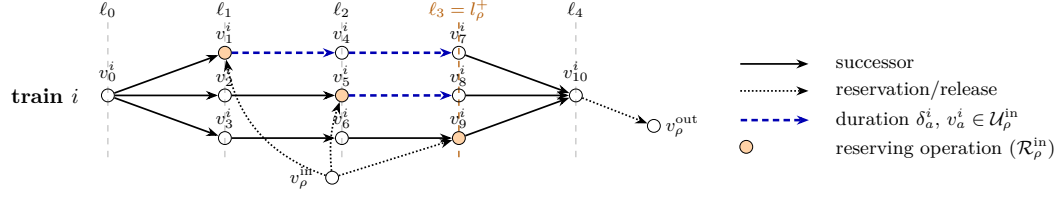
In this section, we discuss several performance improvements. Furthermore, for the DISPLIB instances we require some so-called no-swap constraints. Finally, we detail the implemented heuristic configurations to support faster convergence of the iterative algorithm.

Level-aggregated resource timing constraints. Let l_ρ^- and l_ρ^+ be the levels of the earliest releasing and latest reserving operation of usage $\rho \in \mathcal{R}$, which we call the *anchor levels* of ρ . The release time is bounded below by the start of its earliest releasing level l_ρ^- plus the durations of the selected operations traversed up to the actual release, plus the release time of the selected releasing operation. The reservation time is bounded above by the start of its latest reserving level l_ρ^+ plus the reservation times of the selected reserving operations, minus the durations of the intermediate operations selected before l_ρ^+ . Here $\mathcal{U}_\rho^{\text{in}} \subseteq V^i$ are the intermediate operations on the path strictly before l_ρ^+ , and $\mathcal{U}_\rho^{\text{out}} \subseteq V^i$ are the intermediate operations from l_ρ^- until (and excluding) the actual release. They lie on a path between the anchor level of ρ and its actual reservation or release, respectively. Figure 5 illustrates the constraint for a train with three alternative paths reserving ρ on different levels.

$$t_\rho^{\text{out}} \geq t_{l_\rho^-}^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{out}}} \delta_a^{\text{rl}, i} x_a^i + \sum_{v_a^i \in \mathcal{U}_\rho^{\text{out}}} \delta_a^i x_a^i \quad \rho \in \mathcal{R}, \quad (19)$$

$$t_\rho^{\text{in}} \leq t_{l_\rho^+}^i + \sum_{v_a^i \in \mathcal{R}_\rho^{\text{in}}} \delta_a^{\text{rv}, i} x_a^i - \sum_{v_a^i \in \mathcal{U}_\rho^{\text{in}}} \delta_a^i x_a^i \quad \rho \in \mathcal{R}, \quad (20)$$

where $i = i(\rho)$.



■ **Figure 5** Level-aggregated resource constraint for a resource usage ρ with three alternative paths that reserve ρ at different levels. The latest reserving level is $\ell_\rho^+ = \ell_3$, with the associated start-time variable $t_{\ell_3}^i$. The dashed arcs are the durations δ_a^i of the intermediate operations $\mathcal{U}_\rho^{\text{in}} = \{v_1^i, v_4^i, v_5^i\}$, which conditionally push the reservation time back from the anchor level to the actual reservation.

If the chosen path reserves ρ at an operation before its latest reserving level ℓ_ρ^+ and some operations between that reservation and ℓ_ρ^+ permit waiting, then we can satisfy (20) by waiting just before ℓ_ρ^+ , even though the reservation has already occurred. The symmetric situation arises for the release in (19). On both sides the constraint is therefore a relaxation of the true timing: t_ρ^{in} may be set later, and t_ρ^{out} earlier, than the selected operations actually allow. Whenever the solver returns a solution that violates the reservation or release timing, we repair it by adding a constraint anchored at each reserving or releasing level, with (11) and (12). We recognize such a violation when processing a solution, where we propagate the time-constraint bounds in our disjunctive graph and compare them against the solution time variable of each reservation or release node. Ideally, most reservations and releases of a given resource usage ρ are anchored at a common level, keeping the potential number of such repair constraints small.

Implication constraints between conflicts. We generalize the implication rules of [4] to our generalized disjunctive graph. Similar ideas about linking conflicts have been shown by [5]. To this end we define the concept of *overlapping* resource usage of a train between ρ and σ . Let train i visit ρ before σ ; then ρ and σ are overlapping iff for every path between $\mathcal{R}_\rho^{\text{in}}$ and $\mathcal{R}_\sigma^{\text{in}}$, σ is reserved before ρ is released. Furthermore, we say ρ forward implies σ iff ρ and σ are overlapping and every path starting at $\mathcal{R}_\rho^{\text{in}}$ unavoidably visits a node of $\mathcal{R}_\sigma^{\text{in}}$. Conversely, we say σ backward implies ρ iff every path ending at $\mathcal{R}_\sigma^{\text{in}}$ unavoidably visits a node of $\mathcal{R}_\rho^{\text{in}}$. If both directions are true, we say ρ and σ are strictly overlapping.

Now let $C' = \{\rho_i, \rho_j\}$ and $C'' = \{\sigma_i, \sigma_j\}$ be a pair of conflicts between the same two trains (i and j). If both trains have strictly overlapping resource uses, then the decisions for C' and C'' have to be the same. Thus, we can reuse the z -variables for the second conflict, which corresponds to grouping the disjunctive constraints and applying them jointly.

If not both trains have strictly overlapping resources for the conflict pair, but train i forward- or backward-implies from ρ_i to σ_i and likewise train j from ρ_j to σ_j , we can no longer share the z -variables. Still, no overtaking is possible once both trains use both resources of the conflict pair. Therefore we add order agreement cuts, which require that the two orientation decisions coincide whenever both conflicts are active:

$$z_{\rho_i, \rho_j} + z_{\sigma_j, \sigma_i} \leq 1, \quad z_{\rho_j, \rho_i} + z_{\sigma_i, \sigma_j} \leq 1. \quad (21)$$

The first inequality forbids train i from preceding train j at C' while train j precedes train i at C'' , and the second forbids the symmetric mismatch. Together they force the two conflicts to select the same train order on these two resources. For an inactive conflict both its orientation variables are forced to 0 (13), so the cuts stay valid even when a usage is optional and the corresponding resource indicator is not selected.

No-swap constraints. If we want to model operations and resource usages of zero duration, or to model the problem assuming a train length of zero, no-swap constraints become relevant. Swapping appears if all current resources of a train are released at the start of an operation exactly when new resources are reserved; if swapping were allowed, then the train would not hold any resource at this precise moment of the start of the operation, and another train that wants to use these resources in reverse order could match the swap point. The train dispatching problem is usually considered with no-swap constraints, whereas for classic job-shop scheduling, swapping may be reasonable.

A feasible solution is a topological ordering without backward arcs considering the state of the disjunctive graph, which is only possible if there are no cycles. The zero-cost cycles are exactly those that show swapping possibilities. Therefore, for every new conflict, we collect all zero-cost cycles and add corresponding cuts on the disjunctive variables. The number of zero-cost cycles can be quite high. We observed up to 2800 zero-cost cycles in `nor5` instances. Alternatively, we could consider adding only the cycles that incrementally appear in the disjunctive graph for any infeasible solution observed while iterating. Note that we also need to add a special case for the previously explained domination among newly discovered conflicts: a resource usage with an observed zero duration interval cannot be dominated since it may be a swapping pivot.

In Section 6, we will show the performance of our approach on a TDP-NW dataset and the DISPLIB instances. To find optimal and feasible solutions for the DISPLIB instances, exactly these no-swap constraints are needed, as swapping scenarios are common there. In the TDP-NW instances, swapping is impossible since the resource usage of operations is modeled considering the train from head to tail, and no trains of zero length are considered.

Heuristic configurations. We implemented several parameters to configure more heuristic approaches. First, we work with a timeout per iteration. Second, after a given time threshold, we begin to fix the decision variables of conflicts. We only fix confident decisions, i.e., decisions that have remained unchanged for at least the last k iterations, where a longer streak means a more confident decision. Among these candidates, we fix a configurable percentage per iteration. In the experiments in Section 6, we always used a minimum streak of $k = 4$ and a percentage of 5%.

6 Computational Results

First, we present computational results on a challenging curated TDP-NW dataset, created from DB InfraGO instances [18] for three different dispatching regions, and time horizons of 20, 40, 60, and 90 minutes. Second, we show that our algorithm is also competitive for the DISPLIB instances [9], where we find new best known solutions within a time limit of 10 minutes, using the commercial MIP solver Gurobi 13.0.1 [7] and a computer with 10 cores.

Benchmark dataset for TDP-NW. Together with this paper, we publish a dataset of challenging problem instances from dispatching data of a DB InfraGO traffic management system [17]. We selected instances for three dispatching regions of Frankfurt, Munich and Stuttgart, covering morning and afternoon traffic. We select instances with many conflicts and a large connected component in the conflict graph. The threshold values for the objective components are raised to the shortest-path lower bound on the arrival time whenever that lower bound exceeds the threshold, to remove the unavoidable delay from the objective. As in DISPLIB, we keep the instance format simple and general. This first TDP-NW benchmark

is created from graphs of velocity profiles for each train with route reservations and releases given relative to a profile, see [18]. The route exclusions are transformed into exclusive resources, as described in Section 3, and the velocity profiles into operations that reserve and release these resources. In Appendix A, Table 4 reports our results on the curated TDP-NW dataset.

DISPLIB instances. The DISPLIB problem does not explicitly model the kind of microscopic feasibility we address with our work on TDP-NW. Distinguishing features are the overlap resource usages and the no-wait property of operations. For the latter, however, one could subsume paths of no-wait operations into a single operation. The DISPLIB instances come from various sources [9] considering dispatching horizons between 6 and 24 hours. The number of trains per instance ranges from 4 to 505, with a maximum of $\approx 50\,000$ operations for the **nor5** instances. Our approach can prove optimality within 170 seconds for all DISPLIB instances, except **wab** and most **nor5** instances. Table 2 reports the average runtime per instance group for the groups we solve to proven optimality. That includes three new optimality proofs for the **smi_headway** instances compared to [5], and in considerably less running time. For detailed results see Table 5 in Appendix A. In Table 6 we show seven new best known solutions given a time limit of 10 minutes for this benchmark. The solutions themselves are published in the repository with the TDP-NW Benchmark [17]. For the **wab_large** and **wab_small** instances our approach is unable to close the gaps within 10 minutes and we need a heuristic setting to find good solutions. Therefore we run our solver with a timeout of 20 seconds per iteration and start fixing 5% of unfixed conflicts after 50 seconds. Table 7 shows that we find 20 new best known solutions out of 32 instances.

■ **Table 2** DISPLIB instance groups solved to proven optimality, with the average runtime over all instances of each group. The average of **swi** instances is mostly driven by **swi_9**.

Instance group	Count	Avg. time [s]
nor1_critical	10	0.12
nor1_full	10	1.31
nor2	5	1.00
nor3	5	1.62
nor4_small	5	17.22
nor4_large	5	46.18
smi_close	9	0.22
smi_headway	12	0.65
swi	9	22.71

Comparison of TDP-NW and DISPLIB instances. The TDP-NW instances differ structurally from the DISPLIB instances, as Table 3 illustrates. The train graphs observed in the DISPLIB instances reach higher depths but have few operations per level, whereas the TDP-NW instances are shallower but have many more operations per level. This is expected, since the TDP-NW instances cover much shorter dispatching horizons. Moreover, they contain many alternative operations for the same physical track section, differing in start and end velocities and the used route. The ratio of optional levels in the TDP-NW instances is similar to the one in the **wab** instances, which results from rerouting alternatives of different length.

■ **Table 3** Structural comparison of selected instances: average number of levels per train, average number of operations per level, and the ratio of optional levels.

Instance	Avg. levels	Avg. level size	Ratio opt. levels
fbzm_large_dh_40	18.19	42.94	0.24
mbzs_dh_20	14.42	34.36	0.19
rbzs_dh_20	15.23	13.14	0.17
wab_large	83.61	1.51	0.20
nor5_large	50.42	1.99	0.00

7 Conclusion and Outlook

The related work on microscopic train scheduling often claims physical feasibility but seldom refers to an explicit enumeration of velocity profiles with precomputed reservation and release times. Moreover, the details of infrastructure exclusions are usually approximated. In this work we have presented a real-life train dispatching dataset which models the physical feasibility to this level. We have established a method to compute an equivalent resource-based representation from route-based infrastructure constraints, including partial route exclusions and overlap route exclusions. This bridges the gap between real-life feasibility and the extensive related work on timetabling and train dispatching using disjunctive graphs. Naturally, no-wait constraints arise in such instances. Moreover, the number of alternative operations becomes very large. While our newly contributed TDP-NW benchmark dataset only covers dispatching horizons of up to 90 minutes and at most 148 trains, its instances are already large in the number of operations. To address that, we developed modeling techniques on disjunctive graphs with dedicated reserving and releasing nodes to express resource conflicts across alternative operations. Combined with big- M -free reservation and release constraints anchored at a single level-time variable for the vast majority of resources, this enables new performance gains. The techniques developed, in particular how to group conflicts by resource usage pair and how conflicts then relate to each other, are MIP-independent contributions that may equally benefit heuristics and custom branch-and-bound methods for train dispatching with rerouting. Table 4 shows that the relative optimality gaps are large for most instances. The absolute gaps, however, stay below 3600 delay seconds for all but four instances, distributed over up to 60 trains of the largest subproblem. This provides some degree of confidence in the quality of the solutions.

While our results are the best known overall for the DISPLIB instances and, for the TDP-NW dataset, meet the standards of our industrial partner DB InfraGO, the high number of operations per level on these instances and the fact that most optimality gaps are far from closed demonstrate the need for different or complementary procedures. Column generation comes to mind, as most operations might be dominated by others given a set of considered conflicts. This has been explored by [16] for the same type of instances, but modeled directly on routes. The transformation and simplification of the instances might enable gains for column generation for TDP-NW. For the same reason data reduction rules could be considered as a preprocessing step, or in an incremental framework depending on the growing set of conflicts. The work of [5] demonstrates data reductions for the DISPLIB instances, contracting sequences of operations. Another stream would be to implement custom branch-and-bound frameworks, where no-wait constraints and fixed operation costs make the longest path subproblem potentially harder. We hope that our TDP-NW dataset will stimulate research for practically efficient and scalable algorithms for railway dispatching.

References

- 1 Maurizio Boccia, Veronica Dal Sasso, Leonardo Lamorgese, Carlo Mannino, and Paolo Ventura. Optimizing train dispatching for the Union Pacific Railroad, March 2025. doi:10.48550/arXiv.2503.19535.
- 2 Valentina Cacchiani, Dennis Huisman, Martin Kidd, Leo Kroon, Paolo Toth, Lucas Veelenturf, and Joris Wagenaar. An overview of recovery models and algorithms for real-time railway rescheduling. *Transportation Research Part B: Methodological*, 63:15–37, May 2014. doi:10.1016/j.trb.2014.01.009.
- 3 Veronica Dal Sasso, Leonardo Lamorgese, Carlo Mannino, Andrea Onofri, and Paolo Ventura. A 0,1 Linear Programming Approach to Deadlock Detection and Management in Railways. *Transportation Science*, 59(1):187–205, January 2025. doi:10.1287/trsc.2024.0521.
- 4 Andrea D’Ariano, Dario Pacciarelli, and Marco Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, December 2007. doi:10.1016/j.ejor.2006.10.034.
- 5 Florian Fuchs, Thomas Dubach, Jan Lordieck, Francesco Corman, and Bernardo Martin-Iradi. Handling conflicts in microscopic railway timetabling at scale: A parallelized approach. *Journal of Rail Transport Planning & Management*, 39:100600, September 2026. doi:10.1016/j.jrtpm.2026.100600.
- 6 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Data reduction and exact algorithms for clique cover. *ACM J. Exp. Algorithmics*, 13:2:2.2–2:2.15, February 2009. doi:10.1145/1412228.1412236.
- 7 Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2026.
- 8 Oddvar Kloster, Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. An Optimization-Based Decision Support Tool for Incremental Train Timetabling. *Operations Research Forum*, 4(3):65, August 2023. doi:10.1007/s43069-023-00243-2.
- 9 Oddvar Kloster, Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. DISPLIB: A library of train dispatching problems, July 2026. doi:10.2139/ssrn.7069663.
- 10 Leonardo Lamorgese and Carlo Mannino. A Noncompact Formulation for Job-Shop Scheduling Problems in Traffic Management. *Operations Research*, 67(6):1586–1609, November 2019. doi:10.1287/opre.2018.1837.
- 11 Florin Leutwiler, Guillem Bonet Filella, and Francesco Corman. Accelerating logic-based Benders decomposition for railway rescheduling by exploiting similarities in delays. *Computers & Operations Research*, 150:106075, February 2023. doi:10.1016/j.cor.2022.106075.
- 12 Florin Leutwiler and Francesco Corman. A logic-based Benders decomposition for microscopic railway timetable planning. *European Journal of Operational Research*, 303(2):525–540, December 2022. doi:10.1016/j.ejor.2022.02.043.
- 13 Bjørnar Luteberget, Carlo Mannino, and Giorgio Sartor. Exploiting Incrementality in Branch and Bound for Train Dispatching, May 2025. doi:10.2139/ssrn.5260286.
- 14 Alessandro Mascis and Dario Pacciarelli. Job-shop scheduling with blocking and no-wait constraints. *European Journal of Operational Research*, 143(3):498–517, December 2002. doi:10.1016/S0377-2217(01)00338-1.
- 15 Jörn Pachl. *Railway Signalling Principles: Edition 2.0*. Universitätsbibliothek Braunschweig, October 2021. doi:10.24355/dbbs.084-202110181429-0.
- 16 Maik Schällicke and Karl Nachtigall. Solving the Real-Time Train Dispatching Problem by Column Generation. *Transportation Science*, 59(3):587–602, May 2025. doi:10.1287/trsc.2023.0215.
- 17 Luka Stärk. TDP-NW: Benchmark Instances for the Train Dispatching Problem with No-Wait Constraints, August 2026. doi:10.5281/zenodo.21775303.
- 18 Stephan Zieger and Hannah Richta. DB InfraGO’s Automated Dispatching Assistant ADA-PMB, May 2025. doi:10.48550/arXiv.2505.10085.

A

 Appendix

■ **Table 4** Our results on the curated TDP-NW dataset, using 10 seconds per iteration and, after 50 seconds, fixing at most 5% of the most confident decisions. Columns: conflicts $|C|$, trains $|I|$, operations $|V|$, largest subset max $|S|$, iterations, objective (UB), lower bound (LB), and time.

Instance	$ C $	$ I $	$ V $	max $ S $	Iter.	UB	LB	Time [s]
fbzm_dh_20_0	117	40	19314	16	5	2172.21	1836.21	58.41
fbzm_dh_20_1	127	48	16003	18	13	2311.17	2311.17	39.69
fbzm_dh_20_2	418	44	18396	29	14	3245.90	2344.65	141.27
fbzm_dh_40_0	80	48	24717	22	10	1177.18	1177.18	34.28
fbzm_dh_40_1	135	49	36616	20	7	1572.11	1327.39	71.27
fbzm_dh_60_0	397	83	52231	45	12	5000.44	3792.61	124.75
fbzm_dh_60_1	365	85	46917	44	14	2602.12	1802.90	152.18
fbzm_dh_60_2	180	77	40889	32	7	1739.01	1572.72	70.32
fbzm_large_dh_20_0	271	56	18710	29	10	2329.97	1863.78	100.56
fbzm_large_dh_20_1	335	56	21945	27	9	3764.98	2486.41	92.54
fbzm_large_dh_20_2	357	49	27794	31	20	2672.22	2450.80	195.20
fbzm_large_dh_20_3	201	48	32261	9	5	2152.21	2152.21	15.62
fbzm_large_dh_20_4	158	39	11057	10	6	1233.10	1170.69	50.17
fbzm_large_dh_40_0	910	86	63760	46	16	3448.25	1877.44	176.60
fbzm_large_dh_40_1	1668	86	70543	55	20	23981.02	2661.29	245.34
fbzm_large_dh_60_0	1695	113	80503	62	33	13442.58	3394.11	369.67
fbzm_large_dh_60_1	1092	107	67923	55	25	6732.99	4482.21	266.61
fbzm_large_dh_90_0	2212	134	118681	83	23	8577.28	4396.40	394.20
mbzs_dh_20_0	135	47	15769	19	5	1346.30	1346.30	4.51
mbzs_dh_20_1	255	41	25314	30	10	3123.87	2960.89	94.69
mbzs_dh_20_2	111	38	10463	8	5	1228.02	1228.02	7.50
mbzs_dh_20_3	274	45	39916	22	10	2334.40	2177.63	103.17
mbzs_dh_20_4	317	46	23593	34	11	2665.67	1773.07	136.36
mbzs_large_dh_40_0	400	63	5778	18	10	5211.96	5209.46	37.90
mbzs_large_dh_60_0	141	46	4718	15	5	2662.33	2662.33	4.59
rbzs_dh_20_0	246	64	12567	33	5	2056.27	2056.27	16.86
rbzs_dh_20_1	378	58	10940	30	7	2724.21	1912.85	74.05
rbzs_dh_20_2	359	60	15629	46	11	3060.58	2419.52	119.58
rbzs_dh_20_3	466	64	12498	40	11	5012.52	2912.90	129.59
rbzs_dh_20_4	348	66	10817	44	9	5340.66	4056.77	87.17
rbzs_dh_60_0	338	120	50457	60	7	3133.21	2818.75	71.42
rbzs_dh_90_0	1208	148	70779	107	17	9290.57	5337.03	197.84

■ **Table 5** Detailed per-instance performance on the DISPLIB instances solved to proven optimality. Columns as in Table 4, with the linking conflicts $|C_{\text{link}}|$ and the order agreement cuts $|C_{\text{agree}}|$.

Instance	$ \mathcal{C} $	$ \mathcal{C}_{\text{link}} $	$ \mathcal{C}_{\text{agree}} $	$ I $	$\max S $	Iter.	UB	LB	Time [s]
nor1_critical_0	50	8	27	12	10	9	4133	4133	0.09
nor1_critical_1	45	8	17	8	8	8	2416	2416	0.05
nor1_critical_2	35	10	18	9	7	9	3775	3775	0.05
nor1_critical_3	132	19	70	16	16	15	8016	8016	0.48
nor1_critical_4	9	0	1	4	3	4	1506	1506	0.01
nor1_critical_5	13	4	1	6	5	6	2677	2677	0.02
nor1_critical_6	63	12	32	12	10	14	4491	4491	0.12
nor1_critical_7	41	5	19	10	9	10	4137	4137	0.06
nor1_critical_8	76	16	41	10	9	16	3836	3836	0.17
nor1_critical_9	78	13	41	12	12	13	5488	5488	0.12
nor1_full_0	447	7	67	119	88	17	3299	3299	0.88
nor1_full_1	520	3	62	148	111	22	2486	2486	1.27
nor1_full_2	207	5	84	40	34	16	6046	6046	0.47

continued on next page

■ Table 5 (continued from previous page)

Instance	$ C $	$ C_{\text{link}} $	$ C_{\text{agree}} $	$ I $	$\max S $	Iter.	UB	LB	Time [s]
nor1_full_3	336	21	93	56	47	32	2658	2658	2.55
nor1_full_4	407	4	86	89	73	22	5358	5358	1.16
nor1_full_5	490	3	86	129	103	18	3732	3732	1.06
nor1_full_6	515	4	87	147	106	21	5262	5262	1.05
nor1_full_7	499	0	88	146	108	19	1012	1012	0.99
nor1_full_8	674	21	155	155	121	28	5370	5370	1.88
nor1_full_9	587	8	86	151	121	26	6076	6076	1.79
nor2_1	193	29	105	23	23	15	4937	4937	0.95
nor2_2	230	34	125	23	22	15	4619	4619	1.10
nor2_3	172	25	93	23	19	11	5500	5500	0.85
nor2_4	202	32	102	23	20	17	6186	6186	1.40
nor2_5	170	24	89	23	20	11	5416	5416	0.69
nor3_1	317	45	178	21	20	28	3667	3667	3.21
nor3_2	237	33	139	22	21	22	5740	5740	2.21
nor3_3	193	18	100	22	21	13	5562	5562	0.79
nor3_4	187	25	102	21	21	11	4605	4605	0.51
nor3_5	205	27	119	22	19	21	2923	2923	1.37
nor4_small_1	1587	9	227	165	146	161	8972	8972	23.42
nor4_small_2	1426	23	247	168	146	107	11217	11217	18.21
nor4_small_3	1213	18	178	161	134	107	7924	7924	13.78
nor4_small_4	1149	7	174	157	126	97	8189	8189	12.07
nor4_small_5	1366	18	199	167	148	128	12144	12144	18.64
nor4_large_1	2895	26	255	365	296	148	9551	9551	43.64
nor4_large_2	3085	27	277	347	288	226	8439	8439	61.37
nor4_large_3	1277	24	229	171	138	95	11985	11985	33.53
nor4_large_4	2433	38	347	257	219	140	13260	13260	71.51
nor4_large_5	1669	20	219	204	174	134	8043	8043	20.83
smi_close_0	7	2	2	6	3	1	679	679	0.01
smi_close_1	65	20	24	6	4	4	4316	4316	0.37
smi_close_2	57	12	29	5	5	5	1331	1331	0.81
smi_close_3	7	0	0	13	4	1	1860	1860	0.04
smi_close_4	4	1	0	5	3	2	24225	24225	0.01
smi_close_5	9	1	1	9	7	3	314	314	0.14
smi_close_6	44	8	35	7	7	4	21034	21034	0.15
smi_close_7	14	0	9	6	6	6	582	582	0.30
smi_close_8	40	4	19	9	4	6	3413	3413	0.18
smi_headway_0	14	3	4	6	3	1	1483	1483	0.01
smi_headway_1	66	22	19	6	6	2	5040	5040	0.39
smi_headway_2	124	28	87	5	5	5	2888	2888	1.59
smi_headway_3	24	2	9	13	7	3	3552	3552	0.14
smi_headway_4	4	1	0	5	3	2	24797	24797	0.01
smi_headway_5	24	0	7	9	7	6	869	869	0.96
smi_headway_6	50	11	38	7	7	5	22236	22236	0.19
smi_headway_7	27	3	14	6	6	8	1150	1150	0.61
smi_headway_8	68	11	37	9	6	10	5159	5159	0.56
smi_headway_9	82	20	34	6	6	5	16006	16006	0.52
smi_headway_10	91	15	45	8	7	8	9007	9007	2.10
smi_headway_11	94	19	37	8	7	5	5579	5579	0.73
swi_1	0	0	0	4	1	0	0	0	0.01
swi_2	207	175	11	58	12	8	0	0	0.06
swi_3	839	697	103	143	56	17	0	0	0.81
swi_4	1081	832	200	148	74	23	0	0	1.72
swi_5	1085	835	200	149	74	23	0	0	1.75
swi_6	2270	1809	403	365	86	8	0	0	1.13
swi_7	5108	4161	1018	467	360	36	0	0	23.68
swi_8	2103	1512	872	133	119	32	0	0	5.00
swi_9	14563	8063	11577	287	285	46	0	0	170.20

■ **Table 6** Our results on the **nor5** instances and **nor4_large_4**, obtained with a timeout of five minutes, after which we limit each solve to 20 seconds and start fixing disjunctive variables. Objective values in bold are new best known solutions. Improvement is the gain (0 if we match) over the previous best published on the DISPLIB page and reported in [5] for a time limit of 10 minutes.

Instance	UB	LB	Previous best	Improvement	Time [s]
nor4_large_4	13260	13260	13261	1	72.0
nor5_small_1	57443	57404	57554	111	422.7
nor5_small_2	66751	66348	66751	0	404.5
nor5_small_3	57446	57446	57446	0	333.5
nor5_small_4	57361	57361	58484	1123	339.7
nor5_small_5	62744	62699	62744	0	359.9
nor5_large_1	72713	72248	72799	86	433.5
nor5_large_2	76410	75189	77118	708	510.9
nor5_large_3	69910	69331	70026	116	414.8
nor5_large_4	69519	69051	69519	0	552.3
nor5_large_5	67277	66874	67312	35	431.9

■ **Table 7** Our results on the **wab_large** and **wab_small** instances, using a heuristic configuration: after 50 seconds we iteratively fix at most 5% of the most confident decisions, with 20 seconds per iteration. Objective values in bold are new best known solutions. Improvement is the gain over the previous best published on the DISPLIB page (negative if we do not match it). We report no lower bounds, as they are not competitive in this setting: once we start fixing decisions they no longer apply to the original instance.

Instance	UB	Previous best	Improvement	Time [s]
wab_large_1	141607	151279	9672	480.6
wab_large_2	145791	144017	-1774	475.8
wab_large_3	121948	152642	30694	492.9
wab_large_4	138726	133630	-5096	508.2
wab_large_5	129493	143886	14393	481.7
wab_large_6	127624	134923	7299	516.1
wab_large_7	125161	146501	21340	484.3
wab_large_8	100352	117051	16699	483.9
wab_large_9	113191	123528	10337	477.7
wab_large_10	130166	127116	-3050	481.3
wab_large_11	140696	127862	-12834	507.9
wab_large_12	133456	130225	-3231	495.3
wab_large_13	91603	117875	26272	473.8
wab_large_14	122546	123753	1207	257.3
wab_large_15	124887	129949	5062	501.3
wab_large_16	107823	128082	20259	489.7
wab_small_1	18003	17055	-948	256.4
wab_small_2	16156	16478	322	254.3
wab_small_3	19179	18767	-412	294.1
wab_small_4	21103	20165	-938	301.1
wab_small_5	25298	25255	-43	374.1
wab_small_6	22540	23318	778	296.8
wab_small_7	21274	26486	5212	293.5
wab_small_8	27450	20138	-7312	373.5
wab_small_9	20601	24116	3515	343.6
wab_small_10	26768	28750	1982	357.3
wab_small_11	23676	24189	513	324.1
wab_small_12	24214	22571	-1643	329.4
wab_small_13	19324	19064	-260	315.7
wab_small_14	19626	19938	312	344.4
wab_small_15	20035	21392	1357	286.2
wab_small_16	15742	19015	3273	265.8