A Simulation-Optimization Framework for Traffic Disturbance Recovery in Metro Systems

M.L. Tessitore[1], M. Samà[1*], A. D'Ariano[1], L. Hélouet[2], D. Pacciarelli[1]

October 5, 2022

[1]Roma Tre University, Department of Engineering, Via della Vasca Navale 79, 00146 Rome, Italy
[2]Inria Rennes - Bretagne Atlantique Campus Universitaire de Beaulieu, Avenue du Général Leclerc, 35042 Rennes, France
* Corresponding author email: marcella.sama@uniroma3.it

Abstract

This paper analyses how train delays propagate in a metro network due to disturbances and disruptions when different recovery strategies are implemented. Metro regulators use traffic management policies to recover from delays as fast as possible, return to a predefined schedule, or achieve an expected regularity of train arrivals and departures. We use as a metro traffic simulator SIMSTORS, which is based on a Stochastic Petri Net variant and simulates a physical system controlled by traffic management algorithms. To model existing metro networks, SIMSTORS has been mainly used with rule-based traffic management algorithms. In this work, we enhance traffic management strategies. We integrate SIMSTORS and the AGLIBRARY optimization solver in a closed-loop framework. AGLIBRARY is a deterministic solver for managing complex scheduling and routing problems. We formulate the real-time train rescheduling problem by means of alternative graphs, and use the decision procedures of AGLIBRARY to obtain rescheduling solutions. Several operational issues have been investigated throughout the use of the proposed simulation-optimization framework, among which how to design suitable periodic or event-based rescheduling strategies, how to setup the traffic prediction horizon, how to decide the frequency and the length of the optimization process. The Santiago Metro Line 1, in Chile, is used as a practical case study. Experiments with this framework in various settings show that integrating the optimization algorithms provided by AGLIBRARY to the rule-based traffic management embedded in SIMSTORS optimizes performance of the network, both in terms of train delay minimization and service regularity.

Keywords: Advanced Public Transport Systems, Alternative Graph, Closed-Loop Framework, Train Scheduling, Heuristic algorithms.

1 Introduction

It has been estimated by the United Nations (2018) that today more than half of the world population lives in urban areas. Therefore, cities deal with an increasing number of challenges that affect the environment (e.g., air pollution, emissions of CO_2, etc.), and subsequently the quality of life of their citizens (e.g., traffic congestion, high motorization rate, etc.). Having a good and reliable public transportation system is one of the key factors to make urban areas more livable. Public transport systems are often composed of several mixed transportation means, including city buses, trolleybuses, light and urban rails. Metro and subway lines are usually the backbone of these transport networks.

Train service frequency is high in metro systems, due to the short distance between stations and the passenger demand, especially during peak hours. Unfortunately, such a crowded system leads to a high number of disturbances in everyday operations that may cause delays and disruptions with many passengers stranded at platforms (Cadarso et al., 2013; Wang et al., 2018; Yin et al., 2021). Customers may thus perceive the quality of this public service as insufficient. Therefore, a better quality of service has to be provided in terms of frequency, comfort, regularity of services, along with information availability about traveling times and routing alternatives.

City stakeholders and metro operators have the goal to maintain certain Key Performance Indicators (KPIs) above a given threshold level. The most commonly used performance indicators studied in the literature are standardized by UITP (2011). Among these, we focus on measurements regarding the punctuality of trains and the regularity of services, seen respectively as the ability of the system to adhere to its schedule and to guarantee feasible time intervals between consecutive train departures. To keep KPIs high and to meet the customers demand, metros are equipped with traffic management systems that mitigate disturbances, and help the system to recover from an initial delayed situation through the use of ad-hoc rescheduling strategies.

Several works deal with the real-time rescheduling problem for urban rail transit systems, focusing on punctuality and/or regularity. Among the most recent, Gao et al. (2016) propose a retiming model for disturbances that minimizes the gap between the timetable and the actual data on traffic flows. By skipping stations in the recovery period, they are able to speed up the circulation of trains and quickly reduce the number of waiting passengers. Huang et al. (2020) propose a two-stage methodology aiming to minimize the number of waiting passengers and to guarantee a certain level of train regularity. They solve several MILP problems by using a hybrid formulation that minimizes the total number of non-boarding passengers and aims to regain nominal train regularity. Kang et al. (2015) study the rescheduling problem at the end of an operational day, minimizing running, dwell and transfer time, while maintaining important passengers connections, considering that the last train of the day is delayed. Xu et al. (2016) look at incidents occurring in the network. A discrete event model that allows for train crossovers is built and solved by using

a custom made algorithm that decides the order in which trains from different directions use the shared capacity, based on a check of the capacity of the infrastructure itself. Yin et al. (2016) solve the rescheduling problem by considering uncertain passengers demands and energy consumption, using adaptive dynamic programming, maintaining the order in which trains travel the network. Zhang et al. (2019) also focus on train regulation through the adjustment of dwell and running time considering passenger flow, although they solve the rescheduling problem with a model predictive control algorithm.

Still, it is very difficult to meet the required trade-off between the computational complexity and the optimality of the returned solution. Metro systems are complex timed systems, and computing an optimal decision for such systems is computationally hard. The usually implemented rescheduling decisions are not necessarily the optimal ones, but those that can be implemented in the shortest time. In fact, in metro systems, time is a key aspect since the network status and forecast traffic conditions need to be constantly updated, especially during peak hours. Therefore, robust control feedback mechanism becomes necessary to get good KPI values during operations.

Traffic management algorithms are proprietary algorithms that are tested in laboratory environments before the actual physical realization of the signaling system. For obvious economical reasons, it is not possible to preliminary evaluate a traffic management solution on a real system. Working with a realistic environment forces to consider the benefits of an algorithm on a limited set of scenarios. An effective testing of train rescheduling models can be done by replacing the metro system with digital computer simulations that reproduce its main features, while allowing an easy modification of the parameters. Traffic operators test the effects of traffic management solutions on simulators, which are faithful enough to draw conclusions on the reliability of an algorithmic approach.

Several simulators have been proposed for metro systems (Nash & Huerlimann, 2004; Radtke & Hauptmann, 2004; Takagi et al., 2006; Grube et al., 2011; Wang & Cheng, 2012). Some of them are abstract descriptions based on graphs, automata, or queuing networks, often too abstract to integrate accurate decisions of traffic management algorithms. On the other hand, commercial simulation tools, such as Railsys or OpenTrack, provide very complete and low-level descriptions of the rail networks, with the drawback of time consuming simulations. For this reason, most of the existing simulation methods for rail transit systems focus on specific issues, such as the simulation of train movements (Allègre et al., 2010), the calculation of energy-efficient driving strategies (D'Acierno et al., 2018), or the generation of train time schedules (Albrecht, 2009), and represent on a microscopic level specific characteristics of the system.

Most of the existing works on real-time train rescheduling assessed their performance on single calls to the optimizer, aiming to analyze the quality of their algorithms. However, optimization approaches are developed in the literature with the scope to be converted into tools to be used in

real-world applications. While few approaches have been applied in practice (Altazin et al., 2020; Borndörfer et al., 2017), they represent more the exception than the norm. To reduce the gap between literature and practice, the impact of optimal real-time rescheduling solution approaches on railway traffic management systems has to be effectively studied. Train rescheduling strategies and optimization algorithms should thus be evaluated by using advanced rescheduling schemes, such as open and closed-loop ones.

In the open-loop scheme, control actions are optimized and implemented only once at the beginning of the control period. Data is considered as perfectly known over the entire train traffic period to be optimized. Since unexpected events are considered as known in advance, open-loop schemes are not suitable for evaluating how the implementation of train rescheduling solutions would affect the railway system, since these schemes do not evaluate how the system would react to unforeseen traffic changes. A more effective and reliable evaluation can be obtained through the use of closed-loop schemes. In this case, the information is not perfect and the traffic state is not known in advance. Train rescheduling measures need to be repeatedly computed on the basis of the current traffic state to manage train delays and/or dwell time perturbations. In the open-loop scheme, no feedback is provided on the train rescheduling decisions implemented at the beginning of the control period. Differently, in the closed-loop scheme, the train rescheduling decisions taken in an optimization step are propagated over time and applied to regularly update the current traffic state. This also means that, in case of new unexpected events, these decisions can be changed, due to new information collected at the current iteration, when still possible (i.e., on operations not executed). Therefore, the closed-loop scheme allows for a better understanding and implementation of the train rescheduling strategies, since all the train rescheduling decisions taken during each optimization call are known and continuously traced during the simulation.

The European project ON-TIME focused on developing an integrated framework for timetabling and automatic real-time management of railway traffic perturbations. The aim of the ON-TIME project was to study the impact on real systems of replanning tools and their effectiveness when disturbances and disruptions occur (Quaglietta et al., 2016). The framework proposed for this project mainly consists of a closed-loop system that integrates the HERMES simulation environment (HERMES simulator, 2014) with decisions provided by the train rescheduling algorithms ROMA (D'Ariano & Pranzo, 2009) and RECIFE (Pellegrini et al., 2014, 2015). Computational results show that the closed-loop framework is able to manage traffic perturbations and to reduce delays, independently from the network topology, the traffic pattern, the perturbation considered, and the solution algorithm. In the same stream of research, Corman & Quaglietta (2013) propose a closed-loop framework by integrating the rescheduling tool ROMA with the stochastic microscopic railway simulation model EGTRAIN (Quaglietta & Punzo, 2013). They analyze how train schedules vary over time, against train service deviations due to stochastic disturbances, and test

the impact of their framework based on different settings for the rescheduling tool. Computational results obtained in Corman & Quaglietta (2015), where several kinds of train rescheduling schemes were compared, confirm that closing the loop in real-time rescheduling problems delivers superior results compared to single open-loop schemes. While these works begin to highlight to practitioners the benefit of embedding rescheduling tools into traffic management systems, they still leave several research points unanswered. The main open questions, for both researchers and practitioners, that this paper aims to address, are: when and how often automatic train rescheduling tools should be used in a dynamic traffic situation, how much ahead a train rescheduling tool should look from when it has been called, but also which is the best train rescheduling strategy to apply to optimize traffic flows in metro systems. Therefore, our work investigates the performance of various rescheduling strategies for detecting and solving potential train conflicts arising in a metro system. Specifically, we want to verify which kinds of train rescheduling actions are better suited to recover delays, if recovery strategies should be analysed on a time or event basis, and how large should be the optimization time window to analyse, i.e. for how long future traffic states should be predicted and optimized each time train rescheduling is called.

We build a close-loop framework, in which a metro system simulator is interfaced with an optimization solver. We apply the simulation model to evaluate the correlation between the daily operations and the stochasticity of the studied problem. Then, we consider the disturbed part of the schedule as a deterministic problem and solve it, optimizing a chosen time period and returning the solution found to be applied in the simulated context. In this work, we investigate how to best obtain the potential reduction of delays, when studying deterministic optimization techniques in a stochastic environment, aimed to reproduce the real-world context, in which such optimization is expected to be used. We achieve this by building a framework that combines the optimization model with the simulator. We evaluate the performance of the simulation-optimization framework through the aforementioned KPIs, assessing the service quality in terms of train frequency and punctuality. In this paper, we use the mesoscopic simulator SIMSTORS (Kecir et al., 2017; Kecir, 2019), highly specialized in the management of metro networks. It was created during the joint ALSTOM-INRIA P22 Project (2015), developed and maintained by the SUMO Team at the INRIA Research Centre in Rennes, France. Here, delays on arrival and departure times are represented by probability density functions, and are built based on assumptions (e.g., delays occur more likely than advances, most incidents occur as doors close, etc.), and real-world data. The traffic management component is represented by simple regulation algorithms that detect and solve train conflicts automatically during the simulation. As optimizer, we use the state-of-the-art solver AGLIBRARY (D'Ariano et al., 2007), created at Roma Tre University, in Rome, Italy, which represents the optimization core of the ROMA software. We test the rescheduling framework on real-world data, from the Santiago Metro Line 1 (Chile), by using a series of optimization

scenarios, in which updated information is utilized to optimize short-term traffic predictions and to propose suitable train rescheduling actions, including train retiming when entering and exiting metro stations and train reordering at merging points in the metro network, such as junction points between lines, interconnecting loops, and entrance/exit to/from the depot(s).

The paper is organized as follows: Section 2 introduces the problem, while Section 3 describes the models used both for the simulation and optimization tools. Section 4 discusses the proposed closed-loop infrastructure. Experiments with real-world data from Santiago Metro Line 1 (Chile) are reported in Section 5. Section 6 summarizes the main paper findings.

2 Problem Description

Metro systems are usually composed of several lines and characterized by block sections, platforms at stations, junctions, turnarounds, and depots, where trains are assigned at the end of their service or wait to start their first operation. Lines topology can vary from classically straight with single or multiple turnarounds to star-shaped topologies or grids. Some lines can partially share tracks, or be completely independent from each others. Hence, a metro system can be divided in infrastructurally independent closed networks, either used by one line or (even partially) shared between several ones. Networks usually have double tracks, each used for a single service direction.

Each train is associated with a daily schedule, known as *timetable*, a succession of trips within the network during a day. Trips can be divided into *passenger services*, from one terminus to the other, and *maneuvering services*, such as the entrance of a train from the depot, the looping of the train from one direction to the other, or the return in the depot. At the beginning of daily operations, trains move from a depot to the commercial lines to start their passenger services. Pairs of successive trips in opposite travel directions are linked with turnaround maneuvers. At the end of their services, trains are brought back to depots or stopped at parking areas.

For safety reasons, a train is not allowed to depart from a station if the subsequent one is still occupied. Furthermore, traffic regulations impose a minimum distance separation among the trains, which translates into a minimum headway time, called *safety headway*, between two consecutive trains on the same network resource. To ensure that safety headways are respected, metro systems rely on signaling systems, imposing that each network resource is assigned to at most one train at any time. In this work, we divide the network into two types of resources: *station platforms* and groups of block sections between stations, henceforth referred as *interstations*.

The passing of a train through an interstation is called *running time* and starts when its head enters the first block section of the group and finishes when its head reaches the end of the last block section of that same group. At a platform, the scheduled waiting time of each train for the boarding and alighting of passengers is called *dwell time*. Running times on interstation resources and dwell

times at stations are described through three different time values: a *nominal* value, indicating the running/dwell time expected in the timetable; a *minimal* value, representing the minimum time required by the train on that network resource, either related to the maximum allowed train speed in an interstation or indicating the minimum time to let passengers board and alight at a station; a *maximum* value that, if reached, indicates a severe problem, not solvable by train rescheduling actions. The difference between the train minimum and nominal running/dwell time on a resource is called *recovery* time. As long as a train is not delayed, it traverses the network by using its nominal running/dwell times. In case of unexpected events, such as the delayed departure of a train from a station, using recovery times allows the train delay to be (eventually) absorbed without changing the train schedule, guaranteeing some degree of light timetable robustness.

As the backbone of urban rail public transport, the frequency of metro services is very high, especially during peak hours. Despite train departure and arrival times are planned in the timetable, they are rarely perfectly met, due to unexpected events to which the system is exposed, that may occur in everyday operations, i.e. technical problems, small running speed variations on tracks, dwell time variations due to heavy passenger load on and off trains, etc. In saturated metro networks, even these small disturbances can have strong consequences, causing severe disruptions and delays, with many passengers stranded at platforms. These problems (caused, e.g., by a delay of a single train) are particularly hard to be managed, and easily propagate to other trains, spreading over the network as a domino effect. We can distinguish between light traffic perturbations, caused by a set of delayed trains, and heavy traffic disruptions, such as infrastructure failures, accidents, etc.

A potential train conflict arises when two or more trains request either the same station platform or the same interstation within a time interval smaller than the safety headway among them. Real-time traffic management copes with potential train conflicts by adjusting the timetable, in terms of retiming and/or reordering trains at junctions and crossing points. Train retiming actions are easier to implement, since they initially affect only a delayed train. These actions involve speeding up the delayed train to absorb small disturbances through the recovery times scheduled in the timetable, or holding the following train to wait for the delayed one. However, train retiming actions risk to propagate the initial train delay to the overall timetable, if the delays are not quickly recovered. Train reordering actions can only be taken where possible (i.e. at merging points in the metro network) and require the coordination between at least two trains, that needs to be carefully evaluated, allowing to recover train delays which would be otherwise impossible to mitigate through retiming actions. Real-time traffic management performance in metro systems is evaluated through several Key Performance Indicators (KPIs), e.g., by assessing the service quality in terms of frequency and punctuality. Service frequency measures how often the service is provided. The more frequent is the service, the shorter is the passenger waiting time, and

the greater is the possibility that customers can choose their preferred transport mode. Service punctuality in public transport systems is another important criteria, representing the ability of the metro system to adhere to its schedule.

In the literature, the Conflict Detection and Resolution (CDR) problem consists of computing a conflict-free schedule for each train, such that safety headway times between trains are satisfied, no train enters the network before its scheduled entrance time, and train delays are minimized.

Given the complexity of the CDR problem, this work focuses on two different aspects of this problem: the need to represent the stochastic characteristics of metro systems, and the need to optimize traffic operations while maintaining a satisfactory level of service. Simulation and optimization approaches are thus considered for the CDR and then linked together.

We use the simulator to get a detailed representation of the network, trains, station platforms, interstations and schedules while keeping track of traffic flows and various sources of stochasticity during operations. When applied, rule-based algorithms recompute a new train schedule by providing ad-hoc local decisions, based on the single disruption or disturbance occurred.

In the CDR problem, optimization methods are often applied to find conflict-free schedules, which satisfy operational objectives by looking at the overall traffic situation, in which disruptions or disturbances occur. Each time the optimization is called, we consider the current position of trains and their past services. In this work, we take a given demand-driven timetable as input, with a high frequency of service which varies over time, depending on the hour of the day, affected by disturbances and disruptions. To guarantee a satisfactory regularity and punctuality, a train is considered late if arriving at a station platform after its scheduled arrival time in the timetable, allowing the use of recovery times. The optimization, however, does not look at the overall timetable, but only at a specific time window. Therefore, to achieve a more compact schedule that would influence as little as possible subsequent parts of the timetable and would preserve a high frequency on the overall timetable and not just on its optimized part, we consider as objective function the minimization of the maximum consecutive train delay, i.e. the delay introduced by rescheduling decisions.

3 Problem Formulation

This section describes the mathematical formulations of the CDR problem used in the simulator and the optimizer, which can be respectively found in Sections 3.1 and 3.2.

3.1 Simulation model

Assuming that a reference timetable is provided, indicating scheduled train departure and arrival times, we model the metro system stochasticity via a variant of Stochastic Time Petri Net (STPN)

with blocking semantics, as defined in Helouet et al. (2018).

Formally, a Petri Net is a tuple $< P, T, R, W, M_0 >$ where P is a set of *places*, T a set of *transitions*, $R \subseteq (P \times T) \cup (T \times P)$ a set of arcs, $W : R \to \mathbb{N} > 0$ an arc weighting function, while $M_0 : P \to \mathbb{N}$ contains the initial *marking* of the net. The Petri Net can be depicted as a directed bipartite graph with two types of nodes, representing the places and the transitions, usually depicted, respectively, by circles and rectangles. Arcs in R are either from a place to a transition or vice versa. A place $p \in P$ is *marked* if its corresponding circle contains a number of black dots, called *tokens*. Any distribution of tokens represents a configuration M, called *marking*. A transition $t \in T$ is *enabled* if all its input places contain at least one token. When a transition t is enabled by a marking m, firing this transition consists of consuming tokens from the *preset* $\bullet t$ (i.e. the set of its input places) and producing tokens in its *postset* $t\bullet$ (i.e. the set of its output places). In Time Petri Nets, time constraints are attached to transitions, and enforce the firing of a transition to occur within a rational time interval after that transition has been enabled. In Stochastic Time Petri Nets, cumulative distribution functions are associated with firing times of transitions and represent the probability that a transition will fire before (or after) its firing time.

In order to model the stochasticity of traffic disturbance, a natural approach is to consider dwell and running times as random variables by following a probability law. Since experience shows that delay probability is substantially higher than the probability of advances, truncated expolynomial functions are chosen in this work to model dwell and running times and their perturbations. The curves drawn for these functions are asymmetrical truncated bell-shaped curves, with several local maximal probabilities, depicting, for instance, the most probable dwell time at each station, and the second most probable dwell time when an incident occurs.

STPNs have been widely used to model and describe the dynamics of trains (Gaied et al., 2019; Ghazel, 2010; Giglio & Sacco, 2016). In our problem, places represent either stations, interstations, or boolean conditions allowing train departures. Boolean conditions are associated to *control places*, that are used to model train departure instructions, guaranteeing that each train does not leave any station before its scheduled departure time. A control place is created for each station and a token in a control place allows the train departure from the corresponding station. Transitions represent *actions*, such as the arrivals or departures of trains. Each transition has a time interval expressing the possible range of dwell and running times and their corresponding probability distribution. When a transition t, representing the arrival of a train at a station, is enabled by a marking m, firing this transition implies the train arrival at that station. If t represents a train departure, the firing of t implies that the train is leaving the station.

Standard semantics of Petri nets are defined in terms of sequences of discrete and timed moves. A discrete move consists in firing a transition t among the set of enabled transitions. A time move consists in simply letting time elapse, hence decreasing the remaining time before a departure or

an arrival, until any transition is enabled. To preserve the safety headways, Helouet et al. (2018) implemented a fixed block policy by representing the train network at mesoscopic level, (i.e. each place represents either a station platform or an interstation, which can be used by at most one train at a time), and adapt the semantics of STPNs, in such a way that a transition cannot fire if places in its postset are not empty. A Time To Fire (TTF) variable τ_t is thus attached to each transition t. When transition t is enabled, τ_t is set with the time that must elapse before firing t. The value of τ_t decreases as time elapses and when it reaches 0, if $t\bullet$ is empty, t fires, otherwise t is blocked until the postset of t becomes empty, signifying the availability of the corresponding network resource.

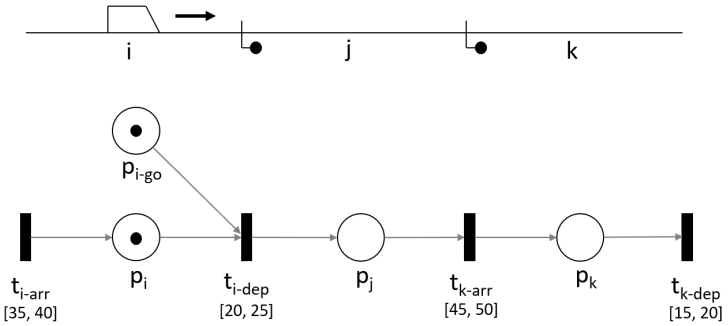

Figure 1: Example of STPN with blocking semantics for a simple network with one train

Let us consider the network of Figure 1. The upper side of Figure 1 depicts a network topology with two stations i and k, one interstation j, and one train starting from station i. The bottom part of Figure 1 shows a STPN with blocking semantics for the network above. Places p_i, p_k and p_j are associated, respectively, to stations i and k, and interstation j. The train, stopped at station i, is modelled by a token located in p_i. $p_{i\text{-}go}$ represents the control place corresponding to the departure instruction from station i. Transitions $t_{i\text{-}dep}/t_{i\text{-}arr}$, and $t_{k\text{-}dep}$ $/t_{k\text{-}arr}$ are associated, respectively, with the train departure and arrival from/to stations i and k. In this example, the blocking semantics require for place p_j (p_k) to be empty when firing transition $t_{i\text{-}dep}$ ($t_{k\text{-}arr}$). An interval representing the possible ranges of dwell and running times is associated to each transition, i.e. intervals on $t_{i\text{-}dep}$ and $t_{k\text{-}arr}$ model the dwell time range for station i and the running time range for interstation j. Cumulative distribution functions are defined on these intervals, providing probabilities for dwell/running times distributed around their nominal value. A TTF variable for each transition is sampled from the corresponding cumulative distribution function. For example, the TTF distribution of transition $t_{i\text{-}dep}$ represents the time needed to leave station i when the scheduled departure time is reached. When transition $t_{i\text{-}dep}$ is enabled, a TTF variable $\tau_{t_{i\text{-}dep}}$ is set for this transition, representing the time that must elapse before firing $t_{i\text{-}dep}$. This TTF decreases

as time elapses and when it reaches 0, since its postset p_j is empty, transition $t_{i\text{-}dep}$ is able to fire. In the example, the train can leave station i after a dwell time sampled between 20 and 25 time units. When $t_{i\text{-}dep}$ fires, the train starts moving from station i, and place p_j is marked, representing the train entrance in interstation j. The running time on j is sampled between 45 and 50 time units. The train enters station k after the firing of transition $t_{k\text{-}arr}$.

3.2 Optimization model

As optimization model for the CDR problem, we use the alternative graph formulation of Mascis & Pacciarelli (2002). The CDR is considered as a job shop scheduling problem, where *resources* represent depots, turnarounds, station platforms and interstations, while each *job* is a train. The term *operation* refers to the occupation of a resource by a job and is here used to indicate the passing of a train through an interstation or its waiting at platforms.

The alternative graph is a triple $G_A = (N, F, A)$. $N = \{0, 1, ..., n-1, n\}$ is a set of $n+1$ nodes: nodes 0 and n represent two fictitious operations, while each node $o \in \{1, ..., n-1\}$ is associated with the start of an operation. A head value h_o is associated to each node $o \in N$, representing the start time of the related operation, i.e. the longest path between nodes 0 and o in G_A. For example, when looking at a train in a station, the start time h_o is the entrance time of the related train in that station. For nodes 0 and n, h_0 and h_n represent, respectively, the start and end times of the considered optimization time window. F is the set of *fixed* arcs that model running times, dwell times, minimum arrival and departure times of the related trains. The *processing time* of each operation, which is the resource occupation time, is associated to the dwell/running time on the related station/interstation. Fixed arcs can represent *release* and *deadline* times linked to an operation, referring respectively to the minimum and maximum times in which that operation must start to be the processed. A is the set of pairs of *alternative* arcs that model the train ordering decisions. Each alternative pair is composed of two alternative directed arcs $((l, o), (q, r)) \in A$ that model the ordering decisions between pairs of trains requiring to use the same resource. The times associated to the alternative arcs represent the minimum headway between the pair of operations involved in each alternative pair.

$$\min \ h_n \tag{1a}$$

$s.t.$

$$h_{\sigma(o)} - h_o \geq w_{o\_\sigma(o)} \qquad \forall\ (o, \sigma(o)) \in F \tag{1b}$$

$$h_o - h_{\sigma(o)} \geq w_{\sigma(o)\_o} \qquad \forall\ (\sigma(o), o) \in F \tag{1c}$$

$$h_o - h_0 \geq w_{0\_o} \qquad \forall\ (0, o) \in F \tag{1d}$$

$$h_0 - h_o \geq w_{o\_0} \qquad \forall\ (o, 0) \in F \tag{1e}$$

$$h_n - h_o \geq w_{o\_n} \qquad \forall\ (o, n) \in F \tag{1f}$$

$$(h_o - h_l \geq w_{l\_o}) \quad \vee \quad (h_r - h_q \geq w_{q\_r}) \qquad \forall\ (l, o), (q, r)) \in A \tag{1g}$$

We consider as objective function, described in (1a), the minimization of the maximum delay introduced by the train scheduling decisions.

Let o be an operation performed by a train and $\sigma(o)$ its subsequent one. Constraints (1b) model the fixed arcs $(o, \sigma(o)) \in F$ representing the minimum processing time of operation o. If o is associated to an operation performed by a train on a station resource, its weight $w_{o\_\sigma(o)}$ represents the minimum dwell time at that station; otherwise, if o is associated to an operation of an interstation, $w_{o\_\sigma(o)}$ is its minimum running time. Symmetrically, constraints (1c) model the fixed arcs $(\sigma(o), o) \in F$ representing the maximum processing time of operation o. If o is associated to an operation performed by a train on a station resource, its weight $w_{\sigma(o)\_o}$ represents, in absolute value, the maximum dwell time at that station, otherwise, if o is associated to an operation on an interstation, $w_{\sigma(o)\_o}$ is its maximum running time.

Constraints (1d) model the fixed arc $(0, o) \in F$ representing the release time constraint related to operation o, i.e. the minimum time at which operation o can start. Symmetrically, constraints (1e) model fixed arc $(o, 0) \in F$ representing the deadline time constraint related to operation o, i.e. the maximum time at which o must start without incurring into an infeasibility. Constraints (1f) model the fixed arc $(o, n) \in F$ representing the due date time constraint related to operation o, i.e. the maximum time at which o can start without deteriorating the objective function value. Due date arcs are here used to model train delays. Since we consider the minimization of the maximum train delay, we fix w_{o_n} as the maximum between the train arrival time in the timetable and in the actual train schedule. We can assign to a single operation each of these time constraints on its own or together with the others. For example, when operation o has both a deadline time constraint and a release time constraint, we generate a time window in which this operation has to start. If the width of this time window is zero, the release and deadline constraints fix the start time of operation o to the same value. Furthermore, when operation o has both a deadline time constraint and a due date time constraint, we enforce a maximum feasible delay for the related operation, represented by the difference between the deadline and the due

date, which, if exceeded, translates into an infeasible schedule, since the deadline would generate a positive weight cycle in the graph.

Constraints (1g) model the alternative pair $((l, o), (q, r)) \in A$. Each alternative pair represents the two possible processing orders between the corresponding pair of trains on the common resource. The weights w_{l_o} and w_{q_r} represent the minimum headway times between those trains. If o and r are the operations associated with the entrance of the trains A and B in the same resource, when o precedes r, the safety headway $w_{q\_r} = w_{\sigma(o)\_r}$ requires that train A must leave the resource, i.e. start its processing of $\sigma(o)$, at least w_{q_r} time units before train B can start processing r. Vice versa, when r precedes o, train B must leave the resource, i.e. start its processing of $\sigma(r)$, at least $w_{l\_o} = w_{\sigma(r)\_o}$ time units before train A can start processing o.

To ensure that each train cannot enter any station unless the previous train occupying that station is at least in the next one, stations are modelled in our problem by means of *blocking* resources, and interstations as *dummy* resources. A blocking resource is a resource that cannot be occupied by more than one train at any time. When performing an operation on a dummy resource, the previous resource remains occupied.

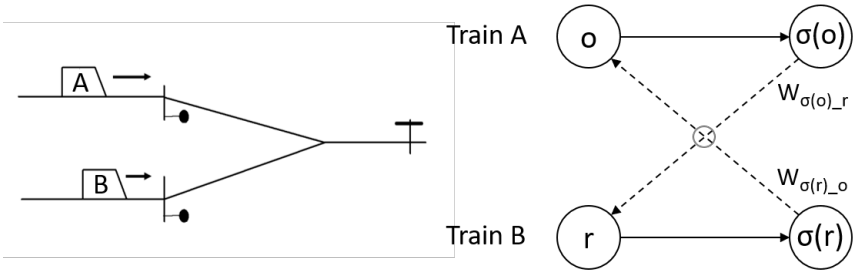

Figure 2: Alternative arcs for a blocking resource

The example of Figure 2 presents an alternative pair when a blocking resource is considered. Figure 3 shows how to model a dummy resource when it follows a blocking one. Let us suppose that operations o, $\sigma(o)$, $q = \sigma(\sigma(o))$, and operations r, $\sigma(r)$, $l = \sigma(\sigma(r))$ in Figure 3 represent, respectively, three consecutive operations processed by trains A and B. Operations o and r require the use of the same blocking resource, in this case a platform at a station. Instead, operations $\sigma(o)$ and $\sigma(r)$ take places on a dummy resource, which represents the traversing of an interstation. This means that, when a train enters the dummy resource, this train will continue to occupy the previous blocking one as well, preventing another train from entering it. So, alternative arcs that, in case of a blocking resource not followed by a dummy one, would have started from nodes $\sigma(o)$ and $\sigma(r)$, as in Figure 2, are replaced by alternative arcs starting from their subsequent nodes $\sigma(\sigma(r))$ and $\sigma(\sigma(o))$, i.e. $((\sigma(\sigma(r)), o), (\sigma(\sigma(o)), r)) \in A$.

A feasible schedule is a digraph $G_A = (N, F, S)$, containing no positive weight cycles, where S

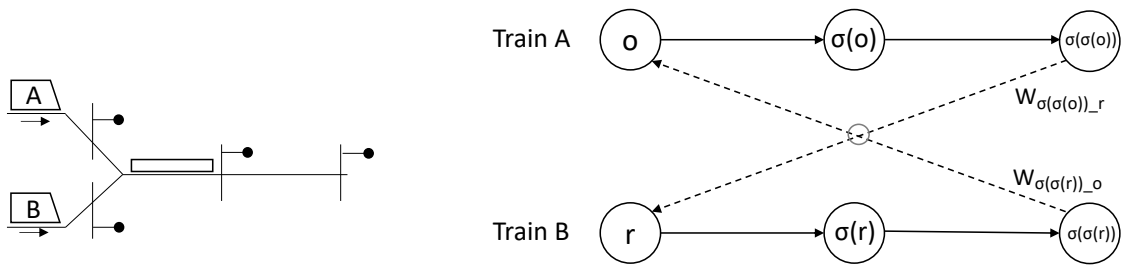

Figure 3: Alternative arcs with a dummy and a blocking resource

is a (feasible) selection of pairs in A, where exactly one alternative arc is chosen for each pair in A.

4 The Framework

To evaluate how to obtain punctual and regular train services, we propose the closed-loop scheme of Figure 4, a framework where train rescheduling measures are repeatedly computed over a specific time period and traffic flow predictions are frequently updated on the basis of the current trains and infrastructure status. We interface AGLIBRARY, a tool for optimal train rescheduling, with the metro traffic simulation environment SIMSTORS. The resulting closed-loop scheme alternates the monitoring of the current infrastructure and trains status in the simulator and the optimization of a given time period of train traffic flow prediction in the metro system.

In general, SIMSTORS simulates a day of operations following a reference timetable. The train schedule is continuously updated, keeping track of traffic flows, disturbances and disruptions. These are managed by regulation algorithms that detect and solve conflicts automatically. The SIMSTORS-AGLIBRARY interaction is implemented by performing successive optimization calls: at periodic or event-based time intervals, conflict-free train schedules are computed based on the current traffic information, such as train speeds and positions, and implemented in the simulator. An optimization call may depend on a triggering condition, i.e. the deviation over a certain threshold of the expected and actual values of an operation start time, or be performed at predefined periodic time intervals. When the optimization is called, the current schedule data and train positions are obtained from the simulator and sent to AGLIBRARY, which predicts potential train conflicts and computes a new conflict-free plan that prescribes a new train schedule and arrival/departure times. The computed train schedule is transferred to the simulator. This newly computed schedule is implemented in the simulator such that trains will follow the generated train arrival and departure times. The next call to the optimizer will (again) depend either on the triggering condition or on the periodic time interval considered for traffic flow prediction and optimization.

The closed-loop rescheduling framework of Figure 4 is carefully illustrated in detail in the

remaining of this section, as follows: Section 4.1 describes the traffic simulator used in this work, while Section 4.2 the optimization solver. Section 4.3 presents the proposed framework for the closed-loop train rescheduling and the interactions between the two tools.

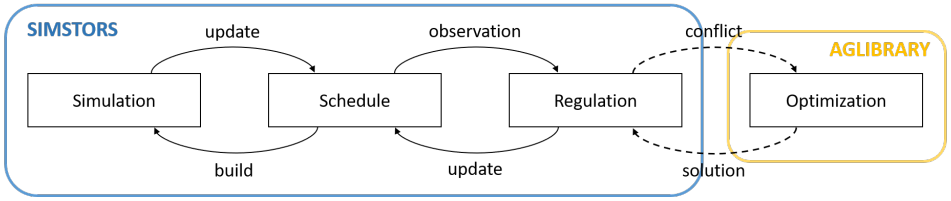

Figure 4: The proposed closed-loop rescheduling framework

4.1 SIMSTORS

SIMSTORS is a simulator for stochastic and concurrent timed systems, that is developed and maintained by the SUMO Team at the INRIA Research Centre in Rennes, France (Kecir et al., 2017; Kecir, 2019). SIMSTORS was created during a joint project between ALSTOM and INRIA P22 Project (2015). As shown in Adeline et al. (2017), SIMSTORS is able to simulate metro systems and disturbed train traffic flows, and to compute statistics for key performance indicators (KPIs).

The design of this simulator, as shown in Figure 4, is modular. The *Simulation* module implements a model of metros, by using the STPN variant described in Section 3.1. The *Schedule* module is used to ensure that train departures follow the given timetable (i.e. an a priori schedule of train movements). This train schedule is used to constrain departure times in the physical model, but is also used to compare the actual operation start times with the planned ones, and hence to compute the KPIs. The *Regulation* module implements a train regulation strategy. Its effect is to collect train delays with regard to the planned train schedule, and to use traffic management rules, while computing a new conflict-free train schedule. The train timetable is modelled by a direct graph $G_T = (V, E, \lambda)$, where V is the set of nodes representing train operations, such as train arrivals and departures at/from stations, $E \subseteq V \times \mathbb{R} \times V$ is the set of arcs indicating precedences and other constraints between train operations. Arc (u, val_{uv}, v) means that event u (an arrival in a station, for instance) precedes event v (a departure from the same station). The value val_{uv} indicates that at least val_{uv} time units must elapse between u and v. The dating function $\lambda : V \to \mathbb{R}$ associates a realization date to each node of G_T. To satisfy the problem constraints, for every arc (u, val_{uv}, v), we have $\lambda(u) + val_{uv} \leq \lambda(v)$.

Let us consider the toy example of Figure 5. The top plot of Figure 5 depicts the network topology, which consists of a depot d, two stations (i and k), and two interstations (j and l). We consider 2 trains running on this network, identified by colours green (G) and red (R). The bottom

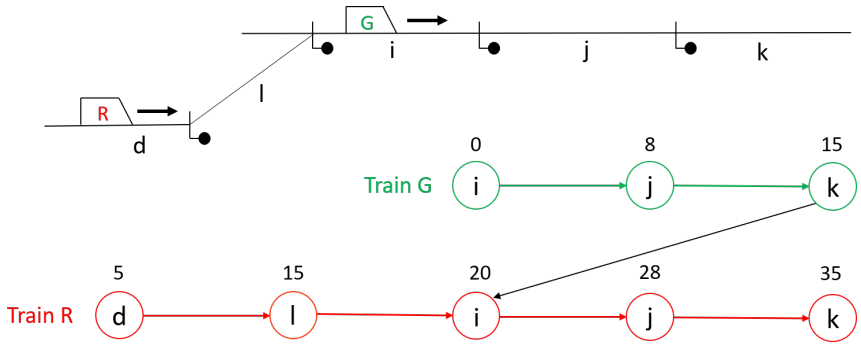

Figure 5: Example of a network with two trains and the corresponding graph $G_T = (V, E, \lambda)$

plot of Figure 5 shows the timetable graph representation for the given network and train traffic flow. Nodes i, j and k represent, respectively, the arrival of train G at station i, its departure from i and its concurrent entrance in interstation j, and its arrival at station k. The arrival of train R at depot d, its departure from d and its arrival at station i are depicted by the red nodes d, l and i, respectively. Scheduled start times are associated to each node, e.g., the departure of train G from station i is scheduled at time 8, its arrival at station k is scheduled at time 15. Each coloured arc depicts a precedence constraint between two operations performed by the same train. Black arcs represent precedence constraints between operations to be performed on the same resources by different trains, i.e. the timetable order between each pair of trains on the shared resources. To ensure that train safety headways are satisfied, each train cannot enter any station if the following interstation is still occupied by a previous train. In the example of Figure 5, black arc (k, i) is used to model that train R can enter station i only after train G arrives at station k.

During the simulation, departure and arrival times of trains are built based on timetable data (*build* arrow in Figure 4 from *Schedule* to *Simulation*). However, according to the timetable, train arrivals and departures to/from a station cannot occur earlier than scheduled, but these events could be delayed. When a train arrives/departs late at/from a station, and its actual arrival and departure times differ from the scheduled one, the train schedule needs to be updated (*update* arrow from *Simulation* to *Schedule*).

SIMSTORS recovers delays by using simple train regulation techniques based on local decision rules: the primary delay is propagated to compute the earliest realization dates of the next event, by considering the imposed time elapsed between each pair of events. This corresponds to a *hold-on policy* that tries to realize the timetable by planning the earliest possible dates for the upcoming events. SIMSTORS *Regulation* algorithm is implemented as a method that, when triggered (*observation* arrow from *Schedule* to *Regulation*), computes new train arrival and departure times and updates the schedule based on the hold-on policy, by changing departure and arrival times of the involved trains, aiming to remain as close as possible to the original timetable (*update* arrow

from *Regulation* to *Schedule*). The decisions allowed by these local rules are thus train retiming decisions based on minimum dwell/running times and train reordering decisions at junctions.

4.2 AGLIBRARY

AGLIBRARY is a deterministic solver developed by the AUT.OR.I (AUTomation and Operations Research in Industry) research team at the Engineering Department of Roma Tre University, for the management of complex scheduling and routing problems, ranging from production scheduling (Pranzo & Pacciarelli, 2016) to air traffic control (Samà et al., 2013), or rail traffic management (Samà et al., 2017). AGLIBRARY has been applied to optimize train traffic flows of several conventional and high-speed railway networks. AGLIBRARY solves job shop scheduling problems with special constraints formulated as alternative graphs, including the CDR problem of Section 3.2.

Both heuristic and exact train scheduling algorithms are implemented in the solver (Meloni et al., 2004; D'Ariano et al., 2007; Corman et al., 2014). Fast heuristics in AGLIBRARY can be divided into two families of iterative heuristics: either time-based or arc-based greedy heuristics. Specifically, the time-based heuristics choose how to select the (unselected) alternative arc from each pair based on the start times of the nodes in the graph, e.g., First In First Out heuristic rule gives precedence to the operation with the earliest start time at each iteration, thus looking at head value h_o associated to each node o and selecting the node, not yet evaluated, with the minimum head value. Arc-based heuristics choose how to select the alternative arcs based on some key information regarding each alternative pair, e.g., AMCC (avoid most critical completion time) works as follows: at each iteration, the alternative arc of an unselected pair that would lead the worst deterioration of the objective function value is first found; the other arc in the corresponding unselected alternative pair is then included in the selection S. In this way, the most critical decision is taken at each iteration of this greedy heuristic, by avoiding the worst possible decision. In both time-based and arc-based heuristics, the selection S is iteratively enlarged by focusing either on a node of the graph, thus selecting among its incident alternative unselected arcs, or on an alternative pair, thus selecting one arc from this pair.

To solve the CDR problem in this paper, we employ a truncated version of the branch & bound algorithm of D'Ariano et al. (2007), able to provide proven optimal or near-optimal solutions within the short computation time limits of real-time applications, i.e. this exact algorithm is truncated when a given maximum computation time is reached (or earlier if the proven optimum is found) and returns the best-known solution plus its optimality gap. For a more detailed description of the train scheduling algorithms implemented in AGLIBRARY (and used in this paper), we refer the interested reader to Meloni et al. (2004); D'Ariano et al. (2007); Corman et al. (2014).

4.3 Closed-loop train rescheduling system

As introduced in Section 4.1, SIMSTORS simulates train operations, both in terms of normal traffic situations and delayed/disrupted ones. When train departure/arrival times differ from the schedule, the *Regulation* module adopts train rescheduling strategies, based on local decision rules, to recover the original timetable, or at least, to mitigate the impact of timetable perturbations, by changing departure and arrival times such that the new ones are compatible with the current train positions. When local decisions are not enough to avoid train delay propagation on the network, the train conflicting operations (*conflict* arrow of Figure 4) are identified and solved through the *Optimization* module (i.e. AGLIBRARY in this paper). Once a new conflict-free train schedule is computed by AGLIBRARY, the *solution* is sent back to the simulator, in the form of operation start times and dwell/running times. These values are then used by the simulator to replace the train arrival and departure times and update the current train schedule.

4.3.1 Framework outline and configuration parameters

The identification of potential train conflicts is accomplished through time-based or event-based rescheduling strategies. Both types of strategies have been implemented in the proposed closed-loop framework. For the time-based control strategies, optimization is called at periodic time intervals during the simulation. For event-based strategies, optimization takes place when a pre-defined triggering condition occurs, such as a threshold deviation between the expected and actual values for at least an operation start time. When considering time-based control strategies, we investigate the combination among different values of *Periodic Interval* (PI), that is the time interval between two consecutive calls to the optimizer, and the length of *Time Window* (TW), indicating for how long train trips are predicted. Figure 6a shows an illustrative example of the time-based configuration with three consecutive PIs. Three different lengths of TW are used to represent the possibility of choosing TW equal (TW1), larger (TW2) or smaller (TW3) than PI. As for the event-based configuration, SIMSTORS controls the deviation of each operation start time from its target value. The optimization process is performed, on a given TW, when the positive deviation between the expected start time of an operation in the last updated schedule and the actual one exceeds the given threshold, i.e. when a Triggering Event (TE) is detected. Figure 6b gives an illustrative example of the event-based configuration. The triggering event and the time window length do not change during each simulation, but multiple simulations can have different values of TE and TW. Choosing a suitable length for PI and TW plus a careful fitting of TE are critical issue to improve the performance of the closed-loop system. The use of short PIs or small TEs allows for frequent calls to the optimizer, but this can be useless if the traffic conditions are undisturbed, due to the risk of rescheduling the same train operations multiple times. On the other hand, long PIs or big TEs might have the opposite problem, being not adequate to be adapted to

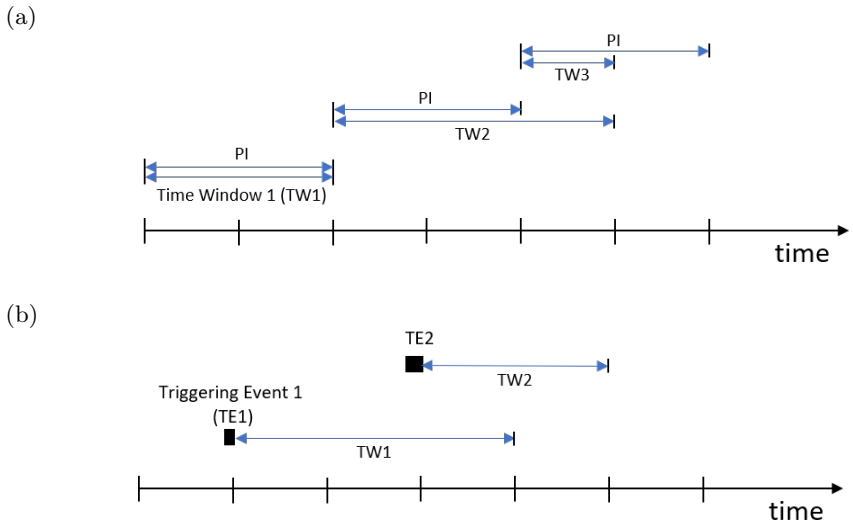

Figure 6: Illustrative examples of (a) time-based and (b) event-based configurations

the evolution of the traffic conditions. While the adoption of short TWs may provide a myopic view on future traffic flows, long TWs may dilute the optimization effort and generate less accurate traffic flow predictions.

The pseudocode of Algorithm 1 illustrates the interactions between the simulator and the optimizer in our closed-loop rescheduling framework, starting from the setting of the chosen configuration parameters. The rest of this subsection focuses on explaining the pseudocode and on giving an example of its application. The proposed framework configuration is chosen at the start of each simulation: PI and TW values need to be set for the time-based configuration, while TE and TW values for the event-based configuration.

In the *Simulation* module introduced in Figure 4, a time-space diagram is used to visualize the train traffic flows. This type of diagram is a common representation of train schedules, where the abscissa reports time, while the ordinate the platforms at each station, the depots, and the network junctions. Each point in the diagram is part of a train trip and depicts the train position in a station, depot, or junction at a specific time; each line connects two points and describes the movement of a specific train on an interstation.

For clarification purposes, we next introduce and discuss the reference example of Figure 7. On the left side of Figure 7, a single loop metro network is depicted with 6 platforms ($k1$, $i1$, $u1$, $k2$, $i2$, and $u2$), 2 depots (d and z), 2 turnaround points (f and w) and 10 interstations between them (e.g. $i1$-$u1$). We consider 4 trains running on this metro network, identified by different colours (letters): blue (B), red (R), green (G) and yellow (Y). Each train can stop at any platform of the stations (e.g., $u1$ or $u2$), depending on their traveling direction. Trains cross the network counterclockwise. Each train performs a daily schedule, i.e. a succession of trips

Algorithm 1:

Input: Set the configuration parameters: PI/TE and TW

Begin simulation

while *simulation not completed* **do**

  if *an operation start time differs from the scheduled one* **then**

    Apply train retiming actions by using the *Regulation* module;

  end

  if *a TE is detected or the previous PI is completed* **then**

    Select the optimization TW through the given configuration parameters;

    Identify the trains in the TW;

    Select the train operations in the optimization instance associated to the TW;

    Model the optimization instance through alternative graph $G_A = (N, F, A)$;

    Detect the potential train conflicts;

    while *a conflict-free schedule is not found or the maximum computation time is not reached* **do**

      Solve the CDR problem by using AGLIBRARY;

    end

    Update the current timetable in the *Schedule* module;

  end

end

within the network, representing either passenger services (bold lines), or maneuvering services (dotted lines). Trains operate passenger services between stations $k1$ and $u1$, or between $u2$ and $k2$, respectively. For example, once reached station $u1$, each train starts a maneuvering service to reach station $u2$, passing through resource f. When a train arrives at station $k2$, it can move toward station $k1$ or enter depot z.

We recall from Section 4.1 that the train schedule is updated during the simulation when the actual train departure/arrival times differ from the timetable ones. In the example of Figure 7, at time 12, a delay is detected for train B, while this train is performing a turnaround maneuver. In undisturbed situations, train B would take 8 time units to travel from w to $k1$, while (in the proposed example) we assume that this operation will take 38 time units. Figure 8 shows the direct graph $G_T = (V, E, \lambda)$ representing the train schedule of Figure 7 at time 12, as computed in the *Schedule* module in Figure 4. In the Figure, each train service is modelled as a sequence of nodes and directed arcs. Each node (i.e. operation) is identified by the corresponding station or interstation, and by the colour (letter) used to identify the train performing that specific operation. The start time of each operation is characterized by a label identifying the scheduled entrance time of the corresponding train in the related resource. For example, the arrival of train B at

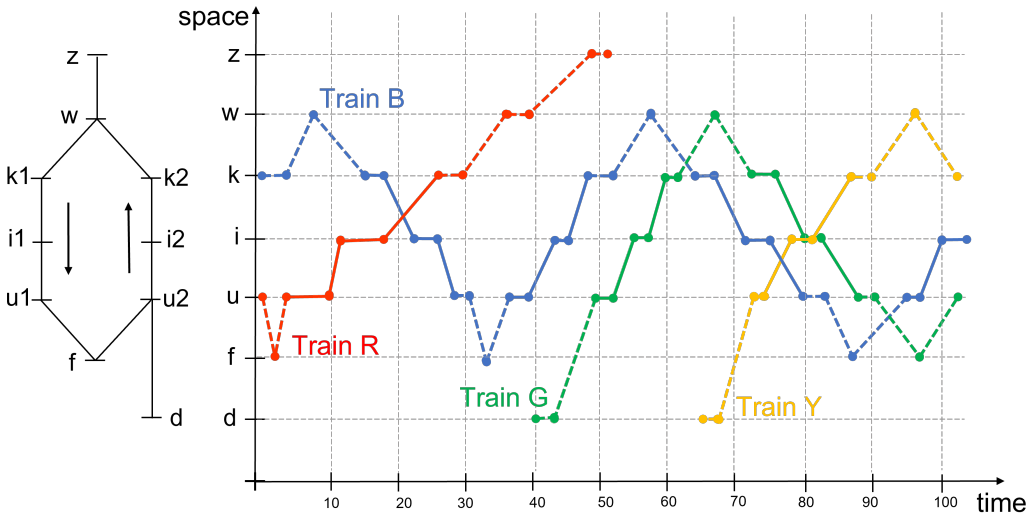

Figure 7: Example of train traffic flow prediction on a time-space diagram

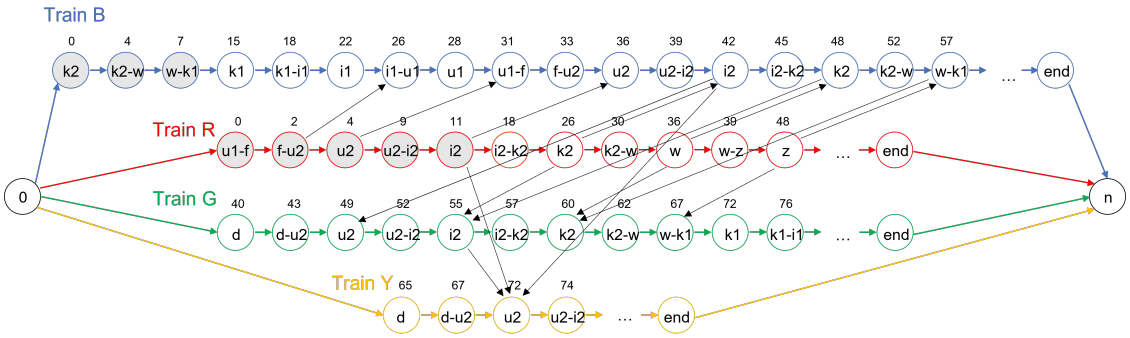

Figure 8: Graph $G_T = (V, E, \lambda)$ for the timetable of the reference example

station $k1$ is scheduled at time 15. The *end* nodes represent the last train operations of the day. Since trains cannot wait at turnaround points during maneuvering services, we do not include the corresponding nodes, with processing time 0, in graph G_T, for easy-to-visualize reasons. Each coloured arc depicts a precedence constraint between two operations performed by the same train. For each of these arcs, its weight corresponds to the nominal dwell/running time, varying from 2 to 10 time units. To lighten Figure 8, we do not add this time information on the coloured arcs, since this can easily be obtained as the difference between the start times of consecutive operations. For simplicity, we consider (for all train operations in the example) a minimum dwell/running time of 2 time units and a maximum one of 50 time units. Since each train cannot enter any station if the following interstation is still occupied, black arcs are used to represent precedence constraints between consecutive operations to be performed on the same resources by different trains, i.e. the timetable order between each pair of trains on the shared resources. For simplicity's sake, for each of these arcs, its weight (corresponding to the minimum headway time between two

operations) is set to 0 time units. To lighten Figure 8, we do not add this type of information on the black arcs. For instance, train G can enter station $u2$ after train B has left station $i2$ plus their minimum separation/headway time. White nodes represent operations that are not yet started at time 12, e.g. the arrival of train B at station $k1$ or the departure of G from d. Grey nodes represent operations that have already started at time 12, either ended or still on-going, such as the maneuvering operation of train B from junction w to station $k1$ or the arrival of train R at $i2$.

4.3.2 Applying train retiming actions by using the *Regulation* module

Let us consider, in Figure 9, an updated schedule for the example of Figure 7, where only train retiming actions are considered in the *Regulation* module. When a delayed train is detected during the simulation, its minimum dwell and running times are adopted to recover from this delay, aiming to remain as close as possible to the current train schedule. In this example, train B arrives late at station $k1$, its dwell time is thus reduced to its minimum value (2 time units) to recover from this initial delay.

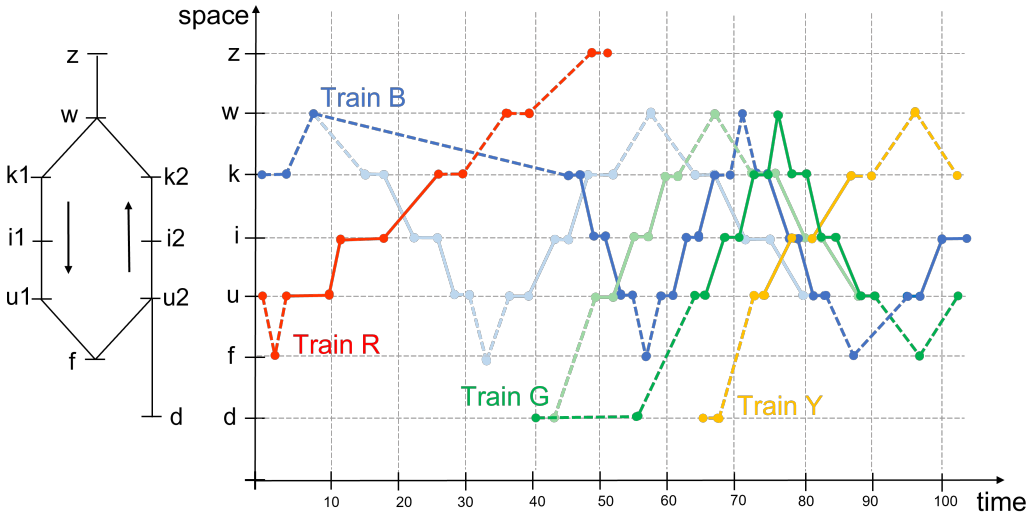

Figure 9: Space-time diagram for the example of Figure 7 when train retiming decisions are applied

If the minimum processing time is not enough to recover the delayed operation, either train retiming actions are applied to successive stations and interstations (and eventually to other trains as well), or local train reordering decisions are possibly taken, until the train delay is recovered or reduced below a given threshold.

In the example of Figure 9, using the minimum dwell time for train B at station $k1$ does not eliminate the initial delay accumulated by this train. The departure of train B from station $k1$, originally scheduled at 18, is rescheduled at time 47 (the arrival at $k1$ plus its minimum dwell time, i.e. $45 + 2$). It is thus necessary to reduce the dwell and running times of train B at

several successive stations and interstations, until its delay is sufficiently reduced. In Figure 9, the minimum dwell and running times are used for train B until its arrival at station $u1$, i.e. until time 79, when the delay of this train is fully recovered.

Due to the precedence constraints between trains, the delay accumulated by each train in any resource of the network may affect the following trains. Since the delayed operations (nodes) of train B are connected in Figure 8 with the nodes of train G by the black arcs, this delay propagates to the latter train. Minimum dwell and running times are thus used for train G, to reduce the wide-spreading of train delays.

4.3.3 Identify the trains in the TW

Depending on the framework configuration, the optimizer is called when either a triggering event is detected or the last considered periodic interval is completely processed. The instance for the optimizer is defined on the basis of the trains (and operations) that need to be processed in the considered TW. Furthermore, the information regarding the current position of each train in the network, its speed, its minimum/maximum dwell/running time is taken from the simulator.

Each optimization call can deal with various traffic situations: trains that are going to depart from the depot in the time window, trains that are already running in the network, and trains that will go to depots during the time window. Trains, whose service has already ended or whose departure from their depot is later than the completion of the optimization time window, are not taken into consideration during the optimization, since either their influence is in the past or it is deemed too limited in the time window to optimize. Let us suppose to choose, in our example of Figure 7, an event-based configuration with a triggering event of 5 time units and a time window of 50 time units. The new optimization call will start at time 12, when the delay of train B on resource w-$k1$ (TE) is detected. At time 12, train B is performing its maneuvering operation from junction w to station $k1$, while train R is dwelling at station $i2$. Train G is going to start its first service at time 40, during the considered optimization time window. We do not take train Y into consideration in this train rescheduling call, because Y will leave its depot after the end of this time window.

4.3.4 Select the train operations in the optimization instance associated to the TW

Since the optimizer needs to know the train traffic flows starting from a time instant significant for each train in the time window, the operations considered in the current optimization instance do not only include the operations scheduled in this time window (TW), but also the operations already-started or scheduled after the TW. If a train is already running in the network when the optimizer is called, we include in the current optimization instance its first operation started, but not yet ended. On the other hand, if the train is still in the depot when the optimizer is called, this

train will be included in the instance, starting from its departure from the depot. Among the start times of the first operations, the earliest one indicates the time from which the optimizer needs to correctly depict the initial occupancy situation of the corresponding train in the network. In our example, the investigated rescheduling call starts while train B is still running on interstation w-$k1$; hence, AGLIBRARY considers this train starting from its actual departure time from turnaround point w. Trains R and G are considered, respectively, from their arrival at station $i2$ and their departure from depot d. Let us suppose that train B has left station w at time 9, and train R has a scheduled departure time from $u2$ at 11. Among the start times of these two operations, the smallest one (9) is chosen as the optimization instance start time to fully depict the availability of the network resources. All train operations started from that time (9) will be included in the train scheduling instance, along with the departure of train R from station $u2$.

All train services, that in the simulation are scheduled to start in the time window to be optimized, are included in the corresponding alternative graph until their completion, i.e. until they reach their terminal station or complete their maneuvering service, even if they exceed the end of the time window. This ensure that each train is entirely considered by the optimizer, and that the corresponding train retiming and reordering decisions are propagated by the *Regulation* module to the timetable, in such a way that no infeasibility will arise in the future. When considering the example of Figure 7 and the event-based configuration with TE = 5 and TW = 50, the optimization (starting at time 12) ends at time 62. The corresponding alternative graph includes trains B and G until the end of their turnaround maneuvering services from station $k2$ to station $k1$, even if these are scheduled in the simulation at times 64 and 73, respectively. Train R enters depot z before the end of the optimization time window. The latter train is thus included in the alternative graph until the end of its last service.

4.3.5 Model the optimization instance through alternative graph G_A

In the alternative graph G_A, a release time constraint is associated to each operation included in the optimization instance, to ensure that either the related train does not enter the network before its scheduled departure time, or to model its nominal dwell and running times, thus maintaining the initial timetable as guideline in the optimization. Furthermore, a release time together with a deadline time are associated to each operation that is already started before the begin of the given time window, both corresponding to its actual start time in the simulator. These constraints ensure that the involved operation start times coincide with their actual values in the simulation, guaranteeing the feasibility of the new solution to past events, i.e. the rescheduling strategies take into account the current traffic state, since past events (such as the current position of each train in the network) cannot be altered by the optimizer.

Figure 10 represents the first trip of train R in the considered time window. To avoid misunderstanding, we refer to an arc in the alternative graph with the following abbreviation: $(Ru2\text{-}i2, 0)$,

where the capital letter (R) before the resource name ($u2$-$i2$) indicates the train performing that operation. The operations of train R on $u2$-$i2$ and $i2$ have started before the optimization call at time 12, thus their start times cannot be changed. To fix their values to their actual start times (respectively, 9 and 11), we include one release arc for each operation (i.e. for $(0, Ru2\text{-}i2)$ and $(0, Ru2)$, we have $w_{0\_Ru2\text{-}i2} = 9$ and $w_{0\_Ri2} = 11$), and one deadline arc for each operation (i.e. for $(Ru2\text{-}i2, 0)$ and $(Ru2, 0)$, we have $w_{Ru2\text{-}i2\_0} = -w_{0\_Ru2\text{-}i2} = -9$ and $w_{Ri2\_0} = -w_{0\_Ri2} = -11$). The operation of train R on $i2$-$k2$ is still not started when the optimization is called, thus its start time can still be changed. In the latter case, only a release arc is associated to this operation, representing its minimum start time in the last updated train schedule.

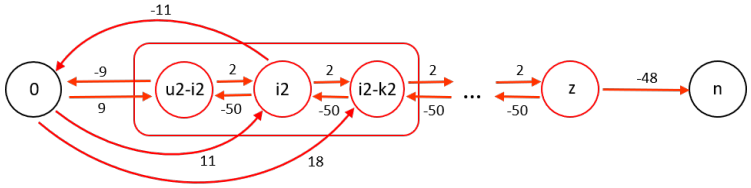

Figure 10: Example of release and deadline constraints in $G_A = (N, F, A)$

In the alternative graph G_A, each passenger or maneuvering service has a given due date time constraint on its last operation in the optimization instance, which is used to minimize the maximum consecutive train delay of the related trip. Additional due date time constraints are included on passenger services. Specifically, due date arcs are incident on the nodes representing operations on interstations following network junctions, to model the traffic flows coming from different directions and to weight the corresponding train ordering decisions. In our example, we have two junction points ($u2$ and w), where the tracks from the depots d and z are connected with the network loop. In Figure 11, due date arcs of train G's passenger service from $u2$ to $k2$ (in the green rectangle) are shown. Specifically, we have due date arcs on the nodes representing interstations $u2$-$i2$ and w-$k1$ to model the traffic flows coming, respectively, from depot d and platform f, and from depot z and platform $k2$. We also have a due date arc constraint on node $i2$-$k2$, since this is the last operation of the related passenger service.

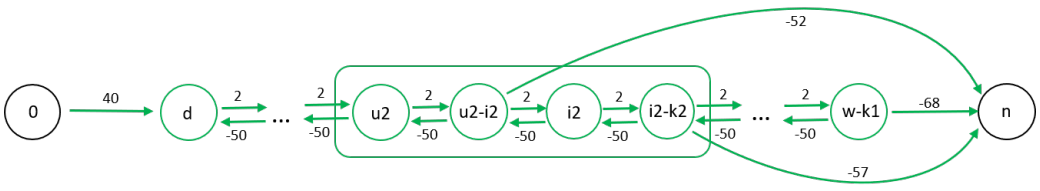

Figure 11: Example of due date constraints in $G_A = (N, F, A)$

When two feasible train ordering decisions exist, a pair of alternative arcs for each blocking

resource models the two possible train ordering decisions. If a train has already started its passenger service and there are no junctions or turnaround points before its arrival in the depot (where train reordering decisions can be applied), one train ordering decision is only feasible. In the latter case, one fixed arc is directly added into the alternative graph, instead of modeling the corresponding alternative pair. In this traffic situation, the train ordering decision is thus pre-defined in G_A.

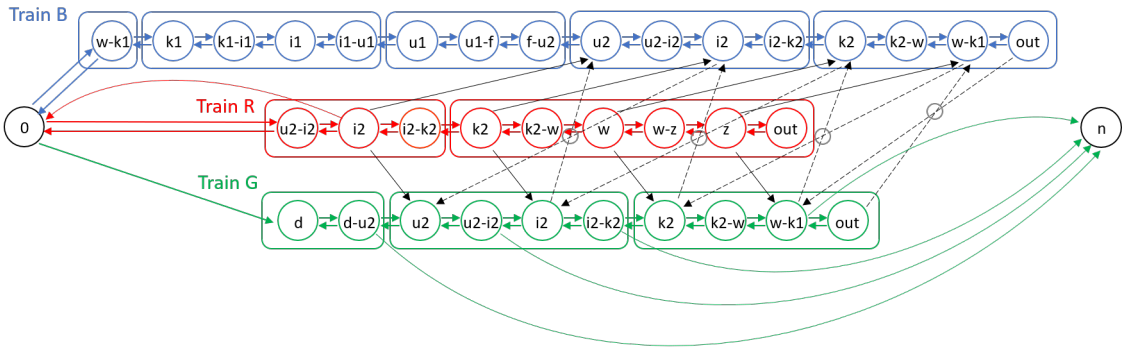

Figure 12: Graph $G_A = (N, F, A)$ for the optimization instance with time window [12,62]

The alternative graph G_A, for the optimization instance of time window $[12, 62]$ in our example, is provided in Figure 12. Each fixed arc is identified with a solid arrow and the colour of the corresponding train. The *out* node at the end of each train in Figure 12 represents the last train operation considered in the considered optimization instance. A release time constraint is associated to each operation, however, for easy-to-visualize reasons, Figure 12 only shows the release arc for the first operation of each train. Nominal dwell/running times of the operations are included in the corresponding release time constraints. Minimum dwell and running times are represented by minimum processing arcs. For example, the weight $w_{Bw\text{-}k1\_k1}$ of blue arc $(Bw\text{-}k1, k1)$ describes the minimum running time of train B on interstation w-$k1$. The maximum running time of train B in resource w-$k1$ is associated to arc $(Bk1, w\text{-}k1)$. Deadline arcs are associated to the operations that are already started at time 12, either ended or still on-going in the current time window: the departure of train B from junction w and the operations of train R, as shown in Figure 10. Due date constraints are used to measure the train delay , as visualized in Figure 11. To lighten Figure 12, due date arcs are included for train G only. The nodes of the alternative graph of Figure 12 are grouped into passenger and maneuvering services, as highlighted by the use of rectangles (e.g., blue nodes $u1$, $u1$-f, and f-$u2$ are part of the turnaround maneuver of train B from station $u1$ to $u2$).

4.3.6 Detect the potential train conflicts

Resources $u2$, $i2$, $k2$, and w are shared by trains B, R and G. Dotted black arcs in Figure 12 denote the required alternative pair for each shared resource, where there are two possible train

ordering decisions. For example, both trains B and G require resource $u2$. Alternative pair $((Bi2, Gu2), (Gi2, Bu2))$ models the two possible ordering decisions between trains B and G on $u2$. Specifically, arc $(Bi2, Gu2)$ allows train B to occupy resource $i2$ before train G, while arc $(Gi2, Bu2)$ gives priority to train G. Between trains B and R, or trains G and R, only one potential train ordering decisions is feasible. These decisions are depicted in Figure 12 by using fixed black arcs. In this case, train R has already started its last passenger service from $u2$ to $k2$, before its return to depot z, and it is thus not possible to alter the scheduled train ordering decisions between trains B and R, or trains G and R.

4.3.7 Solve the CDR problem and update the timetable

Each instance of the CDR problem is solved by using the algorithms in AGLIBRARY (as described in Section 4.2), with a given maximum computation time for each call to the optimizer. Once a CDR solution is found by the optimizer for the current optimization call, this is sent to the simulator and implemented as the new operation start times and dwell/running times, together with the defined precedence constraints. In the simulator, this new information is thus used to update the current timetable in the *Schedule* module. The feasibility of these rescheduling actions with the the rest of the train schedule is guaranteed by the *Regulation* module, which propagates the train rescheduling decisions forward in time. For example, if the order between two trains has been changed by the optimizer, these trains will maintain the new order until either the end of the simulation or another call to the optimizer.

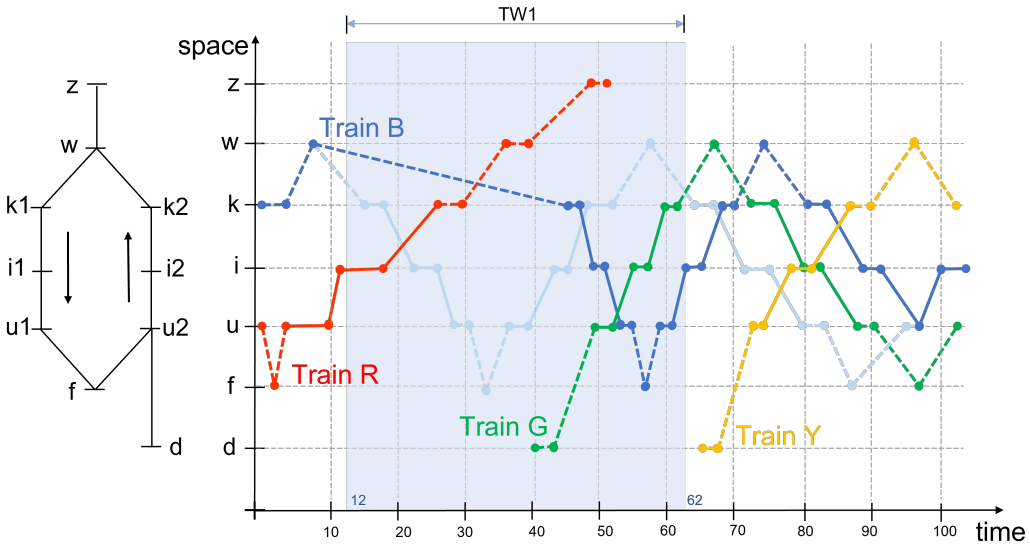

Figure 13: Space-time diagram for the example of Figure 7 when the train schedule is updated at time unit 12 and both train retiming and reordering decisions are applied

Figure 13 shows an optimal solution for the CDR instance with time window $[12, 62]$, when

both train retiming and reordering decisions are applied. The highlighted light blue square in Figure 13, representing the time window $[12, 62]$, shows the operations scheduled in the timetable (lines in transparency) and the ones actually executed in the simulation (bold coloured lines) for each train in the considered time period. In the CDR solution of Figure 13, train G is scheduled as in the timetable, while train B is rescheduled to allow the overtaking of train G, before the arrival of train B on station $u2$. Comparing Figure 13 with Figure 9, it is evident that the start times of the operations related to train G are no longer delayed by train B, as a result of the proposed train reordering decisions.

4.3.8 A new optimization call

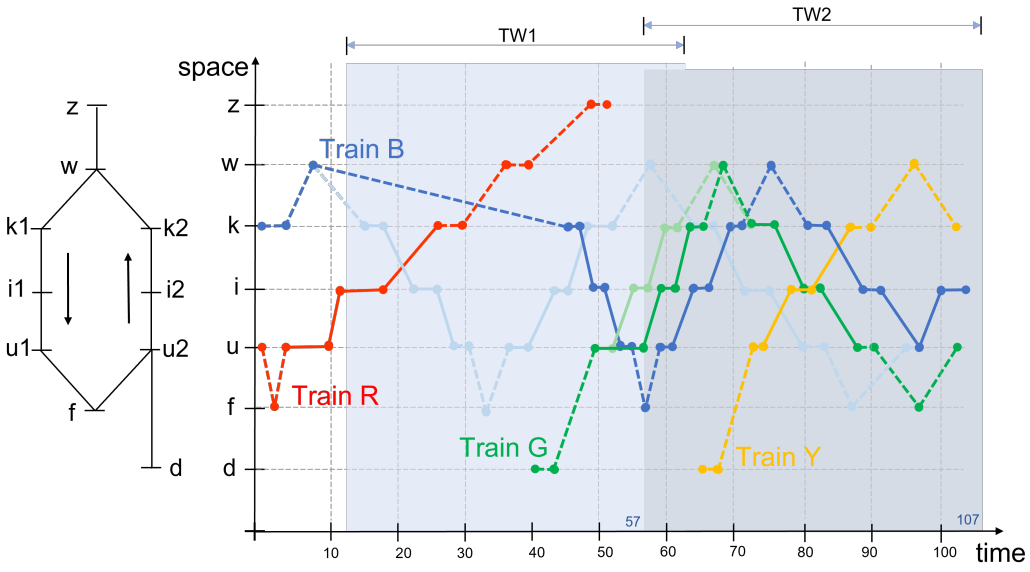

Figure 14: Space-time diagram for the example of Figure 7 when the train schedule is updated at time unit 57

During the simulation, more than one disturb can affect the train traffic flow. Let us suppose that a new train delay is detected for train G at its departure from station $u2$. In undisturbed situations, train G would dwell at station $u2$ for 3 time units before its departure to station $i2$. Due to the delay, we assume that this operation will take 8 time units. A new call to the optimizer and a second update to the train schedule are thus required at time 57. Figure 14 shows the optimal solution for the CDR instance with time window $[57, 107]$, when both train retiming and reordering decisions can be applied. At time 57, decisions taken in the previous optimization call are part of the current traffic state and, if already executed, are unchangeable by the optimizer (e.g., the reordering between trains B and G). However, possible adjustments to the decisions taken but not implemented (i.e., on operations not yet executed) are still possible, if needed, due to the new

information provided during this optimization call. As consequence of the delayed departure of train G, all operations of train G until its arrival at station $k1$ are retimed to allow the recovery of train delays. When train G reaches station $k1$, this delay is completely recovered and the train goes back to follow its initial schedule. Train B can still be scheduled as in the last optimization call ($[12, 63]$), to support the continuity of the train rescheduling solution in different optimization calls (i.e., the propagation over time of each train rescheduling decision). As a result, its timing is not changed by the optimizer. However, if the delay of train G had been greater than the assumed 5 seconds (i.e., departure from $u2$ at 57 instead of 52), this delay would be propagated on train B and new train retiming decisions would have been applied to this train as well.

5 Experimental results

This section addresses the performance evaluation of our closed-loop train rescheduling framework for a metro system. We investigate the potential of the proposed simulation-optimization framework on a practical case study, i.e. the Santiago Metro Line 1 (Chile). Section 5.1 presents the test case and introduces the disturbance scenarios, different framework configurations and parameter combinations, while Section 5.2 provides the computational results on the various problem and framework settings.

5.1 The test case

We consider the Santiago Metro Line 1, a busy line in Chile, as shown in Figure 15. This case study is long 19.3 km and uses a complex ring topology with 27 stations, 2 intertwined rings, and 1 depot of fixed capacity, located nearby the San Pablo (SP) terminal station, which is not included in Figure 15. The trains out-going the depot enter the network at SP2 and, after a turnaround maneuver, start their eastbound passenger services. Turnaround points are located at the terminal stations of SP and Los Dominicos (LD), as well as at the intermediate stations of Pajaritos (PJ), Los Héroes (LH) and Manquehue (MQ). On these intermediate stations, as well as on SP, the merging of different services allows train overtaking actions. Trains are recommended to travel on the metro network by using a nominal speed, when no disturbances affect the normal course of operations. Otherwise, delayed trains can employ minimum running and dwell times, thus using the recovery times allocated in the timetable. Recovery times are, on average, (respectively) 6% and 20% of nominal running and dwell times.

During each weekday, passenger demand is characterized by peak hours (7:00-8:59 and 18:00–19:59) and off-peak hours. The train traffic flow increases during peak hours, and then decreases progressively in off-peak hours. During peak-hours, up to 42 trains can simultaneously travel in the metro network, with services going from LD to SP, PJ and LH (and vice versa), and from MQ to

PJ and SP (and vice versa).

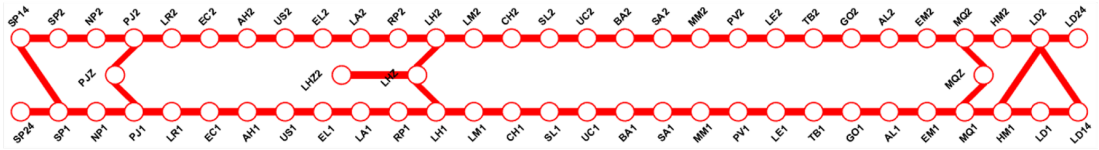

Figure 15: Santiago Metro Line 1

The Petri Net used to model the network is composed of 498 places, of which 147 are control ones, plus 294 transitions. In our experiments, we simulate 5 hours of traffic flow, from the start of the daily service, at around 4:40 AM, to the end of the morning peak-hours, at around 9:40 AM. This simulation range includes the operations related to the entrance/exit of trains in/from the network or the depot. We consider the mentioned peak and off-peak hours plus the two transitions between them, for a total of 500 passenger services, in both traveling directions. We consider a no-conflict-free timetable with 4 traffic scenarios: (i) the nominal metro service; (ii) light (up to 5 seconds) but frequent variations of running and dwell times; (iii) large (up to 30 seconds) but less frequent variations of running and dwell times; (iv) blockage of a train in an intermediate station of the studied metro network for around two minutes.

Different optimization settings are tested for the proposed closed-loop framework. When considering the time-based control strategies, we evaluate 3 periodic time interval values (5, 15 and 30 minutes) and 3 time windows (5, 15 and 30 minutes), and we use AGLIBRARY as the optimizer, with a maximum computation time of 40 seconds. Regarding the event-based configuration, we call AGLIBRARY when the positive deviation between the expected start time of each operation in the current train schedule and its actual start time exceeds a given threshold (TE) of either 0, 1, 3, and 5 minutes. The optimization time windows (used when triggering events are detected) have length of 5 or 15 minutes. The aim of the computational campaign is to test how often the optimization tool should be called and how long future traffic states should be predicted and optimized to consider the effect of train conflict resolution decisions and traffic flow predictions.

5.2 Computational results

This section presents the experimental results obtained for the 5-hour simulations over the 4 traffic scenarios described in Section 5.1. In both time-based and event-based configurations, the results of the first three scenarios show very similar trends, while the fourth scenario has a different one. Consequently, Tables 1 and 2 show, respectively, the average results on the first three (disturbed) scenarios and on the fourth (disrupted) scenario. These two tables are horizontally divided into three parts: Row 1 shows the results obtained when SIMSTORS Local Rules (LR) are used (i.e. when the train rescheduling strategies are only based on the *Regulation* module of SIMSTORS);

Rows 2-10 present the results achieved when AGLIBRARY is called at Periodic Intervals (PI), whose values are specified in the second column; Rows 11-18 report the results obtained when AGLIBRARY is called based on a Triggering Event (TE), whose threshold is specified in the second column.

Table 1: Results obtained for the disturbed scenarios

		TW (min)	# Runs	# Operations	ΔF_m (sec)	ΔF_M (sec)	ΔF_{avg} (sec)	ΔF Var. (sec)	M. Delay (sec)	Avg. M. Delay (sec)	Avg. Delay (sec)
LR	-	-	-	11261	-46.7	152.1	13.7	909.5	1898	1538.8	430.0
PI	5	5	60	11731	-36.9	73.1	7.6	496.2	1146	868.5	237.9
		15	60	11730	-37.9	46.3	**7.4**	432.6	1192	857.8	236.8
		30	60	11732	-36.5	58.5	**7.4**	437.3	1174	850.9	**236.6**
	15	5	20	11719	-38.2	68.9	7.5	462.6	1356	957.8	237.0
		15	20	**11740**	**-36.2**	51.4	**7.4**	425.8	1212	865.1	237.1
		30	20	11723	-38.6	89.8	7.6	483.1	1192	908.6	238.9
	30	5	10	11716	-43.9	86.7	7.6	515.0	1356	959.1	237.9
		15	10	11717	-36.7	72.6	7.5	456.3	1214	880.8	236.9
		30	10	11727	-39.0	89.8	7.6	530.9	1146	884.1	237.7
TE	0	5	49	**11740**	-36.9	**41.8**	**7.4**	**424.0**	1212	858.5	236.7
		15	20	**11740**	**-36.2**	43.3	**7.4**	424.8	**1091**	**812.9**	236.8
	1	5	23	11720	-39.2	102.9	7.5	543.8	1308	918.5	238.9
		15	14	11722	-39.2	112.3	7.6	612.4	1224	931.5	240.0
	3	5	6	11563	-50.2	131.5	9.5	786.1	1492	1186.2	335.5
		15	5	11568	-44.1	108.6	8.2	584.5	1626	1238.2	336.7
	5	5	5	11523	-55.2	146.0	10.1	853.6	1574	1228.3	353.7
		15	3	11542	-48.1	120.0	9.3	720.7	1707	1321.7	355.9

As performance indicators, we consider values linked to train frequency and punctuality. In Tables 1 and 2, columns 1-3 describe the framework configurations used in our experiments, in terms of how and when the optimizer is called (i.e. PI or TE values), plus the width of the optimization time window, expressed in minutes. Columns 4-5 refer to how many times the optimization solver is called and how many train operations (i.e. nodes in G_T) are processed in total in the 5-hour traffic flow of simulation. The remaining columns (6-12) report the average results obtained on the operations processed by all the different framework configurations, to allow a fair comparison between the different settings. Specifically, columns 6-8 report the minimum (m), maximum (M) and average (avg) frequency variations ΔF in all passenger service stations, computed as the difference between the actual and expected frequencies between each pair of consecutive trains in each station platform, while column 9 gives the average variance (Var.) of the frequency variations; all these values are expressed in seconds. Columns 10-12 show the maximum (M.) train delay in the metro line, the average maximum (Avg. M.) train delay in all the stations of the line, and the average (Avg.) train delay in the network, each one expressed in seconds.

Regarding the disturbed scenarios in Table 1, on average, results following similar trends are

Table 2: Results obtained for the disrupted scenario

		TW (min)	# Runs	# Operations	ΔF_m (sec)	ΔF_M (sec)	ΔF_{avg} (sec)	ΔF Var. (sec)	M. Delay (sec)	Avg. M. Delay (sec)	Avg. Delay (sec)
LR	-	-	-	11267	-57.3	159.5	13.3	940.1	1868	1525.1	423.2
PI	5	5	60	11529	-48.2	138.9	9.6	940.3	1508	1162.6	303.9
		15	60	11556	-44.5	94.7	9.4	673.5	1485	1138.5	299.7
		30	60	11732	**-36.6**	58.5	7.4	436.8	1174	850.6	**237.9**
	15	5	20	11719	-39.4	80.8	7.5	496.7	1356	958.4	238.3
		15	20	11541	-47.5	78.3	9.6	661.2	1430	1082.3	313.0
		30	20	11545	-44.3	129.0	9.8	825.4	1430	1232.4	323.3
	30	5	10	11716	-61.5	104.0	7.6	596.6	1356	959.1	239.2
		15	10	11468	-47.1	87.6	10.6	900.2	1496	1186.2	333.2
		30	10	11555	-44.6	121.3	9.5	685.2	1323	1048.2	313.1
TE	0	5	50	**11740**	-42.8	**41.8**	**7.2**	**436.1**	1225	859.5	**237.9**
		15	20	**11740**	-42.1	43.3	**7.2**	437.0	**1091**	**812.3**	238.0
	1	5	24	11720	-45.1	103.3	7.4	557.0	1308	917.5	240.2
		15	15	11722	-45.1	113.1	7.5	627.0	1225	933.0	241.2
	3	5	6	11631	-47.5	114.8	8.5	666.7	1585	1229.5	295.4
		15	5	11640	-48.1	136.9	8.3	699.1	1452	1061.0	295.3
	5	5	5	11527	-62.9	153.1	9.8	858.8	1564	1216.5	349.4
		15	3	11561	-61.5	137.5	9.5	844.8	1697	1311.4	351.8

obtained when using the optimizer periodically or when maintaining, as a triggering event, a small delay tolerance threshold (up to 1 minute). Larger thresholds, due to the high service frequencies and the small train delays involved in the studied traffic flow situations, result in less calls to the optimizer and in more non-optimized train delays. The differences, due to the settings, are more evident when looking at maximum delays and frequencies in Table 1, or in case of disruptions in Table 2. Using TE = 0 allows to mitigate the maximum delay and frequency, but also the deviations from the current train schedule, since the optimizer with this configuration intervenes every time a new positive delay arises in the metro line. When a disruption occurs (Table 2), it becomes more evident that TE = 0 returns a better performance, since periodic calls to the optimizer are less consistent in returning result of average good quality. For both cases, we conclude that event-based configurations with a low delay threshold, as a triggering event, should be used.

With TE = 0, the smaller value of time window (TW) achieves a small value of maximum frequency, while the larger value of TW shows a better performance, in terms of train delay minimization. The larger value of TW also requires less calls to the optimizer, although in both smaller and larger time windows the computation time allocated to the optimizer at each run is enough to prove optimality. This is true with very few exceptions: considering all the 1582 calls to the optimizer and for all 4 scenarios, counting all configurations, the average computation time for the optimizer is 1 second, with the best solutions often found in the initial iterations of the B&B algorithm. The optimality of the provided solutions is not obtained (in the given

computation time) for 16 optimization calls only, with an optimality gap that never exceeds 9% as maximum value, while its average value is around 3.9% on all the 16 optimization calls. These 16 optimization instances regard time-based configurations with TW = 30. Therefore, the length of the time window appears to affect the computational effort more than the performance obtained.

In general, we observe that event-based configurations focus the optimization on the time windows that are mostly affected by train delays and in which delay propagation risks to grow unchecked. In this context, our objective function tends to reduce the largest train delay in the studied time window, thus aiming to lessen the most prominent effect of train delay propagation on the overall timetable. With a large time window, a higher number of operations are optimized by the train rescheduler. In the obtained solutions, delayed operations often start as soon as possible if this best solve the existing conflicts, with a positive effect on containing the worsening of the frequency. With a small time window, the train rescheduler optimizes a reduced number of operations, resulting in a smaller effect on the frequency. In this second case, the train reschedule focuses on the given objective function, as more calls are made to the optimizer.

We now evaluate where and how the optimization calls occur during the overall simulation when time-based and event-based framework configurations are applied. Figure 16 shows the calls to the optimizer within the 5-hour traffic simulation in the disrupted scenario, respectively in case of (16a) time-based and (16b) event-based configurations. On the x-axis, the 5-hour simulation is considered. On the y-axis, closed-loop framework configurations are reported, expressed in terms of how and when the optimizer is called, i.e. the PI or TE values, and the related optimization time window.

In the time-based configuration of Figure 16a, calls to the optimizer are placed at regular time intervals, i.e. every 5/15/30 minutes. In the event-based configuration of Figure 16b, the optimizer is called when the positive deviation between the expected and the actual start time of an operation exceeds 0, 1, 3, or 5 minutes. In both these figures, the length of a coloured horizontal bar represents the width of the corresponding time window. The colour intensity is the same for all the horizontal bars. When the optimization time windows of Figure 16a are bigger than the related periodic rescheduling intervals, the TWs of successive calls overlap and the same operations are considered in more than one call to the optimizer. When bars overlap, the resulting color gets more intense to indicate which portion of the simulation has been rescheduled more than once by the optimizer. The darker the color, the higher the number of calls to the optimizer that have considered the related portion of the simulation. For example, when PI = 5 and TW = 15, the first time window to be optimized starts at 4:40 and ends at 4:55 while the second one starts at 4:45 and ends at 5:00. This means that operations in both time windows, i.e. scheduled in the time interval [4:45, 4:55], will be optimized both by the first and second calls to the optimizer. This is made evident by a darker color in the figure. When a new time window starts at 4:50, operations

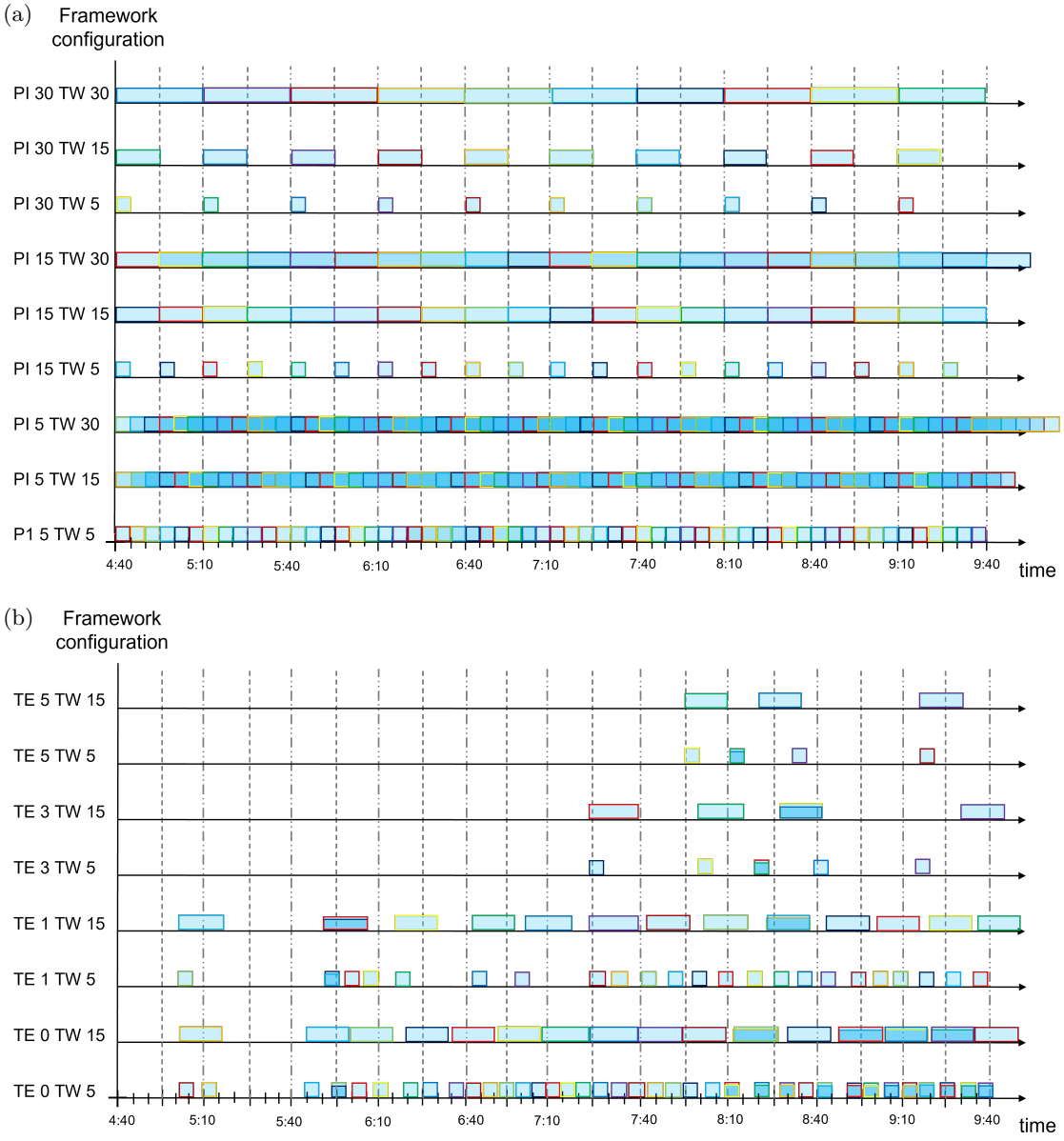

Figure 16: Gantt chart for the disrupted scenario when using (a) time-based or (b) event-based configurations

in [4:50, 4:55] will be optimized for the third time, as shown by an even darker color in the figure.

Comparing Figures 16a and 16b, many of the optimization calls in the time-based configurations turn out to be unnecessary due to the size of the disturbances trains are facing: at the beginning of the simulation, train delays are very small and are immediately re-absorbed. Even if the train traffic flow increases during the simulation and the train delays get larger and larger, the event-based configurations require fewer optimization calls than the time-based ones, thus needing an overall smaller computational effort. On the other hand, when the traffic density increases, in Figure 16b we notice that another call to the optimizer is often needed immediately after some of the optimization calls, to possibly adjust previous train rescheduling decisions, due to the newly available data, or to intervene as soon as new train delays appear. This becomes obvious with TE = 0, i.e. when the optimizer is called each time any new positive deviation arises, and with TW = 5, when the calls to the optimizer are sometimes very close to each other. This increases the risk for the railway systems to become quite nervous, i.e., to have decisions retaken multiple times. However, analyzing in detail the continuity of the train rescheduling solutions, in both time-based and event-based configurations, very few train retiming and reordering decisions are changed between two consecutive calls, and when considering configurations with large time windows only.

Table 3: Train reordering decisions for disturbed and disrupted scenarios

			Disturbed scenarios				Disrupted scenario			
		TW	SP1	PJ1	LD2	MQ2	SP1	PJ1	LD2	MQ2
LR			1	26	2	16	1	27	2	16
PI	5	5	0	1	0	3	0	8	2	4
		15	0	3	1	3	0	7	2	5
		30	0	3	0	5	0	3	0	5
	15	5	0	8	1	6	0	8	1	6
		15	0	3	1	4	0	4	0	4
		30	0	18	0	13	0	10	1	6
	30	5	0	15	1	12	0	15	1	12
		15	0	5	0	5	0	9	1	4
		30	0	3	0	5	0	5	1	4
TE	0	5	0	2	2	3	0	2	2	3
		15	0	1	0	3	0	1	0	3
	1	5	1	6	0	5	1	6	0	5
		15	1	7	1	6	1	7	1	6
	3	5	1	19	2	12	1	8	1	7
		15	1	16	1	9	1	6	2	6
	5	5	1	25	3	15	1	24	3	14
		15	1	20	1	10	1	19	1	10

To better investigate the effects of the optimizer, we next analyze how often train reordering

decisions are taken at the main network junction points, i.e. SP1, PJ1, LD2 and MQ2. As for the previous tables, Table 3 is horizontally divided into three parts based on how decisions are taken in the framework: based on local rules (row 1), by the optimizer periodically (rows 2-10), or by the optimizer when triggering events are detected (rows 11-18). Columns 1-3 describe the framework configurations, in terms of how and when the optimizer is called and the corresponding optimization time window, both expressed in minutes. For the disturbed (disrupted) case, columns 4-7 (8-11) report the number of train reordering decisions detected at the merging points of the two main loops in the studied metro line for the two passenger service directions, i.e. platforms SP1 and PJ1, or platforms LD2 and MQ2.

From the results of Table 3, most train reordering decisions take place in the inner loop (see Figure 15 for the layout of the metro line). This is not a surprising result, since PJ1 and MQ2 are the stations where train traffic flows in each direction merge, while SP1 and LD2 are train entrance points in the network from the depots. However, the best results in Tables 1 and 2 are obtained with a lower number of train reordering decisions, meaning that dwell and running times need to be well calibrated, in such a way that only the necessary train reordering decisions are taken. Otherwise, there is the risk of taking some train ordering decisions, while other actions could have had a similar (or even better) effect in terms of recovering train delays. This can be inferred from Table 3 by the growing number of train reordering decisions taken by AGLIBRARY with the various PI settings for the disrupted case, compared to the disturbed one. Train reordering decisions thus need to be carefully taken, not only by looking at their local effects, but also by evaluating their consequences at a global level on the other trains in the network.

The results in Table 3 are affected by the given configurations of the proposed closed-loop framework. Solving each time window to optimum does not guarantee the avoidance of "unnecessary reordering decisions", since each optimization call focuses on an individual time window and uses the currently available information. For these reasons, train reordering decisions should be reconsidered during successive optimization calls, if they can still be changed. In other words, potential conflicts between trains are solved by the optimizer with the current available information, thus without fully evaluating the consequences of these decisions on the timetable, and the propagation of train delays might be better reduced with multiple calls to the optimizer. This is a clear added-value of the proposed closed-loop framework.

We now further evaluate where and how train delays propagate in the studied metro line. We compare the train delay distribution when Local Rules are used in SIMSTORS and when the best configuration of our framework (i.e. Triggering Event with a threshold of 0 and a TW of 15 minutes) is adopted. Figures 17 and 18 show, respectively, the boxplot of the train delay distributions in case of disturbed and disrupted scenarios. On the x-axis, passenger platforms of the Santiago metro line are considered, by following the counterclockwise sequence, in which trains travel when

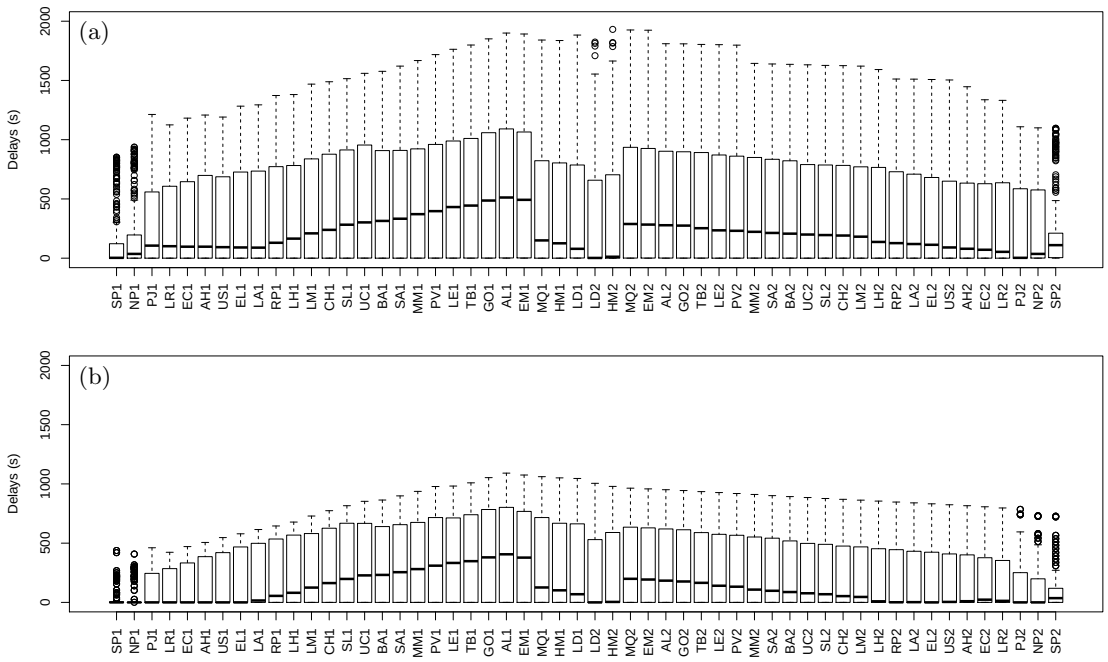

Figure 17: Boxplot for the disturbed scenarios when using (a) Local Rules or (b) AGLIBRARY TE-driven with threshold = 0 and TW = 15 min

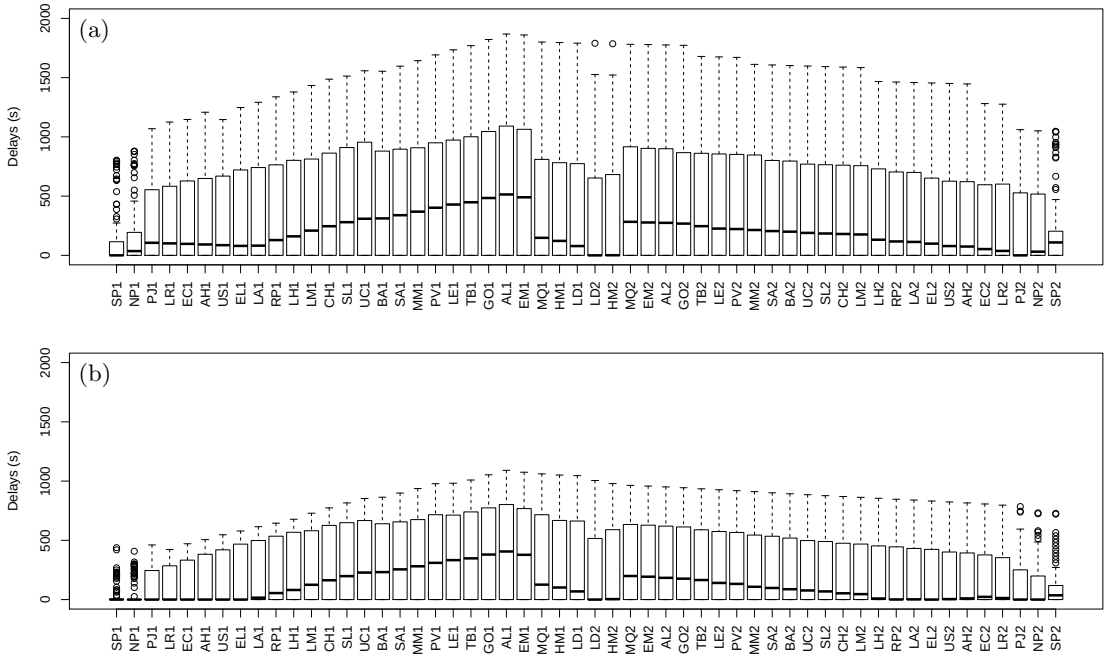

Figure 18: Boxplot for the disrupted scenario when using (a) Local Rules or (b) AGLIBRARY TE-driven with threshold = 0 and TW = 15 min

existing the depot at the beginning of their services. On the y-axis, train delays are reported, expressed in seconds. Each box presents the minimum and maximum values (whiskers), the first and third quartiles (the limit of the box), the median (the black bold line), and the outlier values

(circles) of the train delay distribution at each station. In Figures 17 and 18, the trends of train delay propagation are very similar in terms of the width of boxplots, whiskers and median values, even if these values are smaller in case of TE = 0 with TW = 15 than for LR, as already shown in Tables 1 and 2. Train delays tend to be accumulated during the eastbound passenger services, with a significant increase nearby the merging of junction stations (i.e. in PJ1 and LH1). A noticeable decrease of train delays is visible in MQ1, where trains can either continue their passenger services or start a turnaround maneuver. A similar trend is visible on the westbound traveling direction, in which train delays first increase in the junction station MQ2 and then decrease, as the passenger services approach SP2, due to the trains starting their turnaround maneuvers at LH2 or PJ2.

6 Conclusions

In this paper, we develop a closed-loop train rescheduling framework that combines the metro traffic simulator SIMSTORS, based on a Stochastic Petri Net variant for stochastic and concurrent timed systems, and AGLIBRARY, a state-of-the-art deterministic solver for the management of complex scheduling and routing problems. The aim of this work is to analyse how train delays propagate in a metro network, due to disturbances and disruptions, when different recovery strategies are applied. This paper investigates operational issues regarding how to best use optimization to obtain more punctual and regular metro services. Periodic and event-based calls to the optimizer are tested, together with different lengths for the traffic prediction horizon.

Computational results are obtained for the Santiago Metro Line 1 in Chile. The obtained results show that, when comparing periodic (PI) and event-based (TE) configurations, the latter one is to be preferred in metro systems. Both for the disturbed and the disrupted traffic scenarios, the best performance is obtained when the optimizer is called as soon as a deviation from the current train schedule is detected, i.e. TE = 0. In this case, the solver is able to effectively process a higher number of operations in the 5-hour traffic predictions, as well as maintaining a train service frequency similar to the one designed in the passenger timetable, while better handling the propagation of delays in the network. The time window length instead affects primarily the computational effort, and only secondarily the performance obtained. Larger time windows (i.e., TW = 30) may require additional time to allow the optimizer to provide optimized solutions compared to the smaller ones. Smaller time windows (i.e., TW = 5) may require instead more calls to the optimizer, with the risk of making the system too nervous, due to too many retaken decisions. Since a balance between calling the optimizer a reasonable number of times to avoid a nervous system and performing each call to the optimizer quickly is the reason why a medium-short time window would be preferable, i.e. TW = 15. Furthermore, it has been noticed that in the studied metro system, train reordering decisions may generate a higher train delay propagation

than retiming decisions when these have been myopically taken, i.e. without having evaluated their full effect. The framework configuration in which a reduced but more meaningful number of train reordering decisions is taken has the best overall performance. Train reordering decisions thus need to be carefully evaluated and selected by looking at their (global) effects on subsequent trains in the metro system.

Future research could be dedicated on investigating how the framework can automatically take decisions on the best configuration for each optimization call, depending on the traffic state and the type of disturbances.

References

Adeline, B., Dersin, P., Fabre, É., Hélouët, L., Kecir, K. (2017). An Efficient Evaluation Scheme for KPIs in Regulated Urban Train Systems. *In: Fantechi A., Lecomte T., Romanovsky A. (eds) Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification. RSSRail. Lecture Notes in Computer Science*, **10598**. Springer, Cham.

Albrecht, T. (2009). Automated timetable design for demand-oriented service on suburban railways. *Public Transport,* **1**, 5–20.

Allègre, A. L., Bouscayrol, A., Verhille, J. N., Delarue, P., Chattot, E., El-Fassi, S. (2010). Reduced-scale-power hardware-in-the-loop simulation of an innovative subway. *IEEE Transactions on Industrial Electronics*, **57 (4)**, 1175-1185.

Altazin E., Dauzère-Pérès S., Ramond F., Tréfond S. (2020). A multi-objective optimization-simulation approach for real time rescheduling in dense railway systems. *European Journal of Operational Research,* **286 (2)**, 662–672.

Borndörfer R., Klug T., Lamorgese L., Mannino C., Reuther M., Schlechte T. (2017). Recent success stories on integrated optimization of railway systems. *Transportation Research Part C,* **74 (1)**, 196–211.

Cadarso, L., Marín, A., Gabót, G. (2013). Recovery of disruptions in rapid transit networks. *Transportation Research Part E,* **53**, 15–33.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M. (2014). Dispatching and coordination in multi-area railway traffic management. *Computers and Operations Research* **44 (1)**, 146–160.

Corman, F., Quaglietta, E., Goverde, R.M.P. (2013). Stability analysis of railway dispatching plans in a stochastic and dynamic environment. *Journal of Rail Transport Planning & Management,* **3 (4)** 137–149.

Corman, F., Quaglietta, E. (2015). Closing the loop in railway traffic control: framework design and impacts on operations. *Transportation Research Part C,* **54 (1)**, 15–39.

D'Acierno, L., Botte, M., Gallo, M., Montella, B., (2018). Defining Reserve Times for Metro Systems: An Analytical Approach. *Journal of Advanced Transportation*, vol. 2018, (5983250)

D'Ariano, A., Pacciarelli, D., Pranzo, M. (2007). A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research,* **183 (2)**, 643–657.

D'Ariano, A., Pranzo, M. (2009). An Advanced Real-Time Train Dispatching System for Minimizing the Propagation of Delays in a Dispatching Area Under Severe Disturbances. *Networks and Spatial Economics* **9 (1)**, 63–84.

Gaied, M., Mhalla, A., Lefebvre, D., Othman, K. (2019). Robust control for railway transport networks based on stochastic P-timed Petri net models. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 233. 095965181882358. 10.1177/0959651818823583.

Gao, Y., Kroon, L., Schmidt, M., Yang, L. (2016). Rescheduling a metro line in an over-crowded situation after disruptions. *Transportation Research Part B,* **93, Part A 425-–449**.

Ghazel, M. (2010). Using Stochastic Petri Nets for Level-Crossing Collision Risk Assessment. *IEEE Trans. on Intelligent Transportation Systems,* 668 - 677, 10.1109/TITS.2009.2026310.

Giglio, D., Sacco N. (2016) A Petri net model for analysis, optimisation, and control of railway networks and train schedules. *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, 2016, pp. 2442-2449, doi: 10.1109/ITSC.2016.7795949.

Grube P., Nunez F., Cipriano A. (2011). An event-driven simulation for multi-line metro systems and its application to Santiago de Chile metropolitan rail network. *Simulation Modelling Practice and Theory*, **19** 393-405.

Hélouët, L., Kecir, K. (2018). Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics . In Science of Computer Programming, 157:71-102,2018.

HERMES Simulator, 2014. http://graffica.co.uk/rail-traffic-management/simulation/

Huang, Y., Mannino, C., Yang, L., Tang, T. (2020). Coupling time-indexed and big-M formulations for real-time train scheduling during metroservice disruptions. *Transportation Research Part B,* **133 (1)**, 38–61.

Kang, L., Wu J., Su, H., Zhu, X., Wang, B. (2015). A practical model for last train rescheduling with train delay in urban railway transit networks. *Omega*, **50**, 29-42.

Kecir, K., Hélouet, L., Dersin, P., Adeline, B., D'Ariano, A. (2017). From reactive to predictive regulation in metros. *European Conference on Stochastic Optimization ECSO*.

Kecir K. (2019) Performance evaluation of urban rail traffic management techniques. Cryptography and Security. Université Rennes 1. English. NNT: 2019REN1S026. tel02317224v2.

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

Meloni, C., Pacciarelli, D., Pranzo, M. (2004). A Rollout Metaheuristic for Job Shop Scheduling Problems. *Annals of Operation Research*, **131**, 215–235.

Nash, A., Huerlimann, D. (2004). Railroad simulation using OpenTrack. In J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto, S. Sone (Eds.), *Computers in Railways IX* (p. 45-54). Southampton, UK: WIT Press.

Pellegrini, P., Marlière, G., Rodriguez, J., 2014. Optimal train routing and scheduling for managing traffic perturbations in complex junctions. *Transportation Research Part B,* , 59 (1), 58–80.

Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., 2015. RECIFE-MILP: An effective MILP-based heuristic for the real-time railway traffic management problem. *IEEE Trans. on Intelligent Transportation Systems,* , 16 (5), 2609–2619.

P22 Project Inria-Alstom: http://www.irisa.fr/sumo/P22/. Accessed: 08-02-2021.

Pranzo, M., Pacciarelli, D. (2016). An iterated greedy metaheuristic for the blocking job shop scheduling problem. *Journal of Heuristics*, **22**, 587–611.

Radtke, A., Hauptmann, D. (2004). Automated Planning of Timetables in Large Railway Networks Using a Microscopic Data Basis and Railway Simulation Techniques. In J. Allan, C. A. Brebbia, R. J. Hill, G. Sciutto, S. Sone (Eds.), *Computers in Railways IX* (p. 615-625). Southampton, UK: WIT Press.

Quaglietta, E., Punzo, V. (2013). Supporting the design of railway systems by means of a Sobol variance based sensitivity analysis. *Transportation Research Part C,* **34 (1)**, 38–54.

Quaglietta, E., Pellegrini, P., Goverde, R.M.P., Albrecht, T., Jaekel, B., Marlière, G., Rodriguez, J., Dollevoet, T., Ambrogio, B., Carcasole, D., Giaroli, M., Nicholsonh, G., (2016). The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C,* **63 (1)**, 23–50.

Samà, M., D'Ariano, A., Pacciarelli, D., (2013). Rolling Horizon Approach for Aircraft Scheduling in the Terminal Control Area of Busy Airports. *Transportation Research Part E,* **60 (1)** 140–155.

Samà, M., D'Ariano, A., Corman, F., Pacciarelli D. (2017). A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers and Operations Research* , **78**, 480–499.

Takagi, R., Weston, P. F., Goodman, C. J., Bouch, C., Armstrong, J., Preston, J., Sone, S., (2006). Optimal train control at a junction in the main line rail network using a new object-oriented signalling system model. In J. Allan, C. A. Brebbia, A. F. Rumsey, G. Sciutto, S. Sone, C. J. Goodman (Eds.), *Computers in Railways X* (pp.479–488). Southampton, UK: WIT Press.

UITP, Metro service performance indicators, April 2011.

United Nations, Department of Economic and Social Affairs, Population Division (2018a). World Urbanization Prospects 2018.

Wang, W., Cheng, M. (2012). Simulation of Nanjing Metro Line 1 Using Metro Simulator. *IFAC Proceedings Volumes*, **Volume 45, Issue 21**, 454-459 .

Wang, Y., D'Ariano, A., Yin, J., Meng, L., Tang, T., Ning, B. (2018). Passenger demand-oriented train scheduling and rolling stock circulation planning for an urban rail transit line. *Transportation Research Part B,* **118 (1)**, 193–227.

Xu X., Li K., Yang L. (2016). Rescheduling subway trains by a discrete event model considering service balance performance. *Applied Mathematical Modelling*, **40 (2)**, 1446–1466.

Yin J. T., Tang T., Yang L. X., Gao Z. Y., Ran B. (2016). Energy-efficient metro train rescheduling with uncertain time-variant passenger demands: An approximate dynamic programming approach. *Transportation Research Part B,* **91**, 178—210.

Yin J. T., Yang L. X., Tang T., Gao Z. Y., Ran B. (2017). Dynamic passenger demand oriented metro train scheduling with energy-efficient and waiting time minimization: Mixed-integer linear programming approaches. *Transportation Research Part B,* **97**, 182–213.

Yin J., D'Ariano A., Wang Y., Yang L., Tang T. (2021). Timetable Coordination in a Rail Transit Network with Time-Dependent Passenger Demand. *European Journal of Operational Research,* ISSN: 0377-2217 https://doi.org/10.1016/j.ejor.2021.02.059.

Zhang H., Li S., Yang L. (2019). Real-time optimal train regulation design for metro lines with energy-saving. *Computers & Industrial Engineering*, **107**, 1282—1296.