Modeling Subway Networks and Passenger Flows

Antoine Thébault
Univ. Rennes, IRISA, CNRS & INRIA, Rennes, France
Alstom Transport, Saint-Ouen, France

Loïc Hélouët
Univ. Rennes, IRISA, CNRS & INRIA, Rennes, France

Kenza Saiah
Alstom Transport, Saint-Ouen, France

Abstract

Simulation of urban rail networks provides useful information to optimize traffic management strategies w.r.t. goals such as satisfaction of passenger demands, adherence to schedules or energy saving. Many network models are too precise for the analysis needs, and do not exploit concurrency. This results in an explosion in the size of models, and long simulation times. This paper presents an extension of Petri nets that handles trajectories of trains, passenger flows, and scenarios for passenger arrivals. We then define a fast event-based simulation scheme. We test our model on a real case study, the Metro of Montreal, and show that full days of train operations with passengers can be simulated in a few seconds, allowing analysis of quantitative properties.

2012 ACM Subject Classification Computing methodologies → Modeling and simulation

Keywords and phrases Subways, Passenger Flows, Modelization, Petri-Nets, Trajectory-Nets

Digital Object Identifier 10.4230/OASIcs.ATMOS.2024.16

Funding Work supported by ANRT CIFRE Grant No 2022-0444.

1 Introduction

Development of urban transport networks such as metros is a key issue in the development of cities. Beyond infrastructure, efficient traffic management algorithms are needed to plan operations in the network, and take the most appropriate decisions to optimize its performance. Traffic management can be seen as a combination of planning (one needs a priori schedules to provide a transport offer to clients, and plan composition of train fleets in metro lines), control (to take decisions online to handle delays caused by incidents, and avoid their propagation), and optimization of key performance indicators (KPIs).

Several quality criteria are usually addressed at the same time. An obvious one is passengers satisfaction, which is highly correlated with waiting times in stations. This means that to satisfy passengers demand, traffic management has to consider issues raised by crowds in stations, and by delays. Other quality criteria address running cost of a network, energy consumption, adherence to a determined schedule, etc. To answer all the needs of these complex systems, it is natural to consider automated techniques. However, models for metros are usually huge: even for a small network with a single line, a few stops and a small size fleet of metros, the size of models that can capture the dynamics of the network exceed several millions of states [3]. This exceeds the limits of most verification and optimization tools that rely on an explicit state representation. This calls for the use of efficient simulation techniques and statistical approaches to address real-size case studies.

Related Work. Several commercial tools propose realistic models to simulate metro networks. OpenTrack [20] is often used to measure the KPIs achieved under a certain traffic management policy. OpenTrack models are precise, and almost digital twins of the running systems. As a

consequence, the simulation of a day of operation requires important computing resources. The Railsys suite [7] is another tool addressing train operations at a microscopic level. The simulation scheme of [11] starts from timetables. It considers how primary delays propagate in a schedule, and uses symbolic descriptions of train trips and reduction rules to represent states in a compact way and speed up the simulation process. However, this simulation framework does not consider passengers. Models to simulate metro with their payloads have been developed. Railnet is a simulation model to help planning shared occupation of tracks between freight and passengers transport [19]. The network model is a graph and the vehicles moves are depicted as sequences of arrival and departure dates. Railnet is used to find a solution for insertion of new freight trips that do not harm passengers trains. The model proposed in [1] is an origin-destination model for passengers interconnection to forecast passenger movements and help propose new layouts for interconnections. Another simulation method proposed in [9] evaluates passenger flows in case of service interruption. The simulation is mainly a calculus of paths duration, involving access time, waiting time, and transport duration. Many other tools exist, addressing modeling and simulation at low (microscopic) or high (macroscopic) level, with a formal background inspired from graphs, queuing theory, etc. We refer to [10, 15] for surveys of the domain.

Challenges. Most of the models proposed are either too low level and require important computing resources to simulate a network, or describe metros at a higher level, but ignore passengers. An ideal model should have a semantics close enough to the behavior of the running system, but yet allow fast techniques to perform large simulation campaigns and produce sound statistics on the performance of the system. The basic ingredients of the model must take care of the network topology, of varying composition of vehicle fleets, timetables, and passengers. Of course, an appropriate model for a metro network is a timed model (performance of traffic management is often measured w.r.t. delays or end-to-end trip durations) but also a stochastic model, as trips and dwell durations are subject to random perturbations: unexpected delays can be imposed to a vehicle by a passenger blocking a door, or by a choice of a driver to extend the dwell time in case of important crowds on platforms. As already mentioned, state space explosion is an issue for tools and models relying on explicit states representation. Further, as train movements are independent as long as the distance between the two vehicles is sufficient, concurrency models such as Petri nets are natural candidates to design metros.

Contributions. In this paper, we propose an efficient simulation scheme for metros operated with a moving block policy. Our model includes train movements, passenger flows in stations, platforms and vehicles. Important situations such as delayed trains are difficult to address with standard models such as timed automata [2] or Time Petri nets [18], which led to proposing a new timed variant of Petri nets in [13]. We extend this work, and propose *trajectory nets*, a variant of Time Petri that includes passenger flows. Roughly speaking, a trajectory net is a Petri net in which some places represent track portions and contain forecast train trajectories instead of tokens. Passenger flows in a network are modeled by assigning an integral vector indexed by destinations to trains and platforms, and by queues representing connections between lines. Last, scenarios to describe passenger arrivals in the network are modeled as sequences of Origin-Destination matrices indexed by time periods. Despite these extensions, the model still benefits from fast simulation techniques that were developed for Time Petri nets. We show the efficiency of our simulation scheme on a real case study: we evaluate the effects of train insertions on passengers waiting times during peak hours in the metro of Montreal.

2 A subway network model

A subway network is mainly a graph, and it is natural to see a day of operation as a description of how vehicles move on this graph. A metro that follows a path in the network graph travels at a given speed between two stations, i.e. the train moves for a period of time between two events: a departure and an arrival. The other events that can be considered are incidents, or decision taken by an operator. Last, vehicles have independent moves if they are distant enough in the network, but close trains may have to adapt their speed to maintain safety distances. This calls for the use of concurrent models, and Time Petri nets [18] seem tailored for this task: the flow relation of a net can represent the network, tokens can be used to represent trains, and the timing constraints attached to transitions can be used to enforce trips durations or dwell times. It was shown however that Time Petri nets are not expressive enough to model simple situations where a train has to wait for a departure order even when its dwell time has expired. A new model called waiting nets was proposed to solve this issue in [13]. Further, speed adaptation for close trains cannot be captured by tokens and clocks. In the rest of this section, we describe *trajectory nets*, a model that extends Time Petri nets to handle safety distances and varying speeds of trains. Roughly speaking, a trajectory net is a Petri net in which some places can contain diagrams representing several train trajectories from a station to the next one instead of tokens. Trajectories are discretizations of space-time diagrams, a representation of trains moves frequently used to give a feedback on realized and planned train movements in a network. In the rest of the paper, configurations of a network will be mainly collections of trajectories.

Before introducing the elements of our new model, let us recall the basics of Petri nets. A Petri net is a bipartite graph composed of a set of places, denoted P that represent resources, of a set of transitions, denoted T representing actions. Places and transitions are connected via a *flow relation*, i.e. a subset of $P \times T \cup T \times P$. The set ${}^\bullet(t)$ denotes the resources (set of places) required for transition t to occur, and the set $(t)^\bullet$ represent the resources produced by execution of transition t. Petri nets are often used to model systems with a lot of concurrency and analyze their behaviors. The state of a Petri net is called a *marking*, and is a map that associates a number of *tokens* to each place. The state of a Petri net evolves by firing actions, i.e. consuming one token in each place representing a resource required by a transition t and producing one token in each place in the postset of t. Behaviors of Petri nets can be infinite, and the state state space reachable by a Petri net from an initial marking can also be infinite. Despite this expressive power, many properties such as reachability are decidable for Petri nets [17]. Figure 1 shows an example of Petri nets with places $P = \{p_1, p_2, p_3\}$ and transitions $T = \{t_1, t_2\}$. Repeating sequence of transitions $t_1.t_2$ allows to fill place p_3 with any number of tokens. Due to their graphical nature that can easily simulate networks, to their clear semantics, and to their decidability, Petri nets or their extensions are often used to model railway systems.

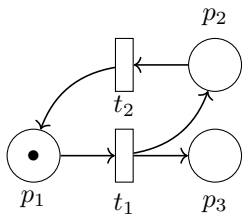


Figure 1 An example Petri net.

▶ **Definition 1.** *A* segment *is a pair of points* $\langle (x,y)(x',y')\rangle$ *where* x, x' *are real values that represent time and* y, y' *are real values that represent distances. A* trajectory *is a pair* (tr, b)*, where* b *is a boolean indicating if the trajectory is blocked, and* $tr = \langle (x_0,y_0)(x_1,y_1)\rangle \cdots \langle (x_{k-1},y_{k-1})(x_k,y_k)\rangle$ *is a finite sequence of consecutive segments such that* $x_0 = 0$*, and* $\forall i, x_i > x_{i-1}$ *and* $y_i \leq y_{i-1}$*. A trajectory is* complete *if* $y_k = 0$*.*

In a point (x, y) of a segment, x represents a duration, and y a remaining distance to a destination. So, trajectories define how distance to arrival decreases over time. For instance a trajectory $tr = \langle (0,200)(10,100)\rangle.\langle (10,100)(30,0)\rangle$ describes a train at distance 200 m from its goal, that arrives 30 seconds later. It travels at $10m/s$ during 10 seconds, and then slows down at $5m/s$. A train approaching a close predecessor will have to adapt its trajectory to preserve headways, and elapsing δ time units will simply consist in shifting trajectory segments by δ units to the left. For a trajectory $tr = \langle (x_0,y_0)(x_1,y_1)\rangle \cdots \langle (x_{k-1},y_{k-1})(x_k,y_k)\rangle$, and a date $d \leq x_k$ we will denote by $tr(d)$ the coordinate of a train at date d. For $x_i \leq d \leq x_{i+1}$ we have $tr(d) = y_i + (d - x_i).\frac{(y_{i+1}-y_i)}{(x_{i+1}-x_i)}$. We will say that two trajectories tr, tr' *respect a safety headway* h if and only if $\forall d, |tr(d) - tr'(d)| \geq h$, and that tr' is above tr if $\forall d, tr'(d) - tr(d) \geq 0$. Trajectories are defined for track portions of bounded length. For a length H we assume that for every trajectory, $y_0 \leq H$. So, we can have at most H/h trajectories respecting a safety headway h in a track segment of length H.

Trajectory nets defined hereafter will use *sequence of trajectories* of the form $TS = (tr_1, b_1)\cdots(tr_k, b_k)$ to represent trips for several trains in a given space of length H. We forbid ill-formed sequences of trajectories where trains overtake or violate the safety headway.

▶ **Definition 2.** *A sequence of trajectories* $TS = (tr_1, b_1)\cdots(tr_k, b_k)$ *is* consistent *(w.r.t. headway* h*) iff every pair of trajectories in* TS *respects safety headway* h*, and* $b_i = true$ *implies that* $b_j = true$ *for every* $j < i$*.*

For a sequence of trajectories $TS = (tr_1, b_1)\cdots(tr_k, b_k)$, we can denote by x_j^k (resp. y_j^k) the j^{th} value of x (resp. y) in trajectory k. With this notation, $y_0^k = tr_k(0)$ is the current distance to next stop of train k in sequence TS. Let $CTS(H, h)$ denote the set of consistent sequences of trajectories (w.r.t. headway h) in a space of length H. Given a consistent sequence of trajectories $TS = (tr_1, b_1)\cdots(tr_k, b_k)$ such that $tr_k(0) \leq H - h$, we can sample an additional trajectory tr_{k+1} such that $TS.(tr_{k+1}, false)$ is consistent and $y_0^{k+1} = H$. Let $I = [\alpha, \beta]$ be a time interval depicting a possible trip duration (these values depend on train speeds allowed by the network, the rolling stock specifications, and the policies given by operators), $d \in I$ be a rational value, and $\widehat{tr_{k+1}} = \langle (0,H)(d,0)\rangle$. Trajectory $\widehat{tr_{k+1}}$ depicts a space time diagram for a train traveling at average speed $\frac{H}{d}$ when no other train is on the track. Now, to preserve consistency of a place contents one cannot always add directly $\widehat{tr_{k+1}}$, to an existing sequence TS : the wished trajectory may have to be adapted (this amounts to reducing the speed of a train) to stay at distance h from preceding trains. That is, one has to compute a trajectory $tr_{k+1} = \uparrow(\widehat{tr_{k+1}}, TS, h)$ such that, for every time value t, $\uparrow(\widehat{tr_{k+1}}, TS, h)(t) = \max(\widehat{tr_{k+1}}(t), tr_k(t) + h)$.

Computing $\uparrow(\widehat{tr_{k+1}}, TS, h)$ is a simple geometrical calculus, defined formally in Appendix A. One can notice that by construction, $tr_{k+1} = \uparrow(\widehat{tr_{k+1}}, TS, h)$ is the only trajectory for a train running at speed H/d whenever possible such that $TS.(tr_{k+1}, false)$ is consistent. For a given sequence TS and time interval $I = [\alpha, \beta]$, we denote by $\text{SAMPLE}(TS, H, h, I) = \{\uparrow(\widehat{tr_{|TS|+1}}, TS, h) \mid \exists d \in I \wedge \widehat{tr_{|TS|+1}} = \langle (0,H)(d,0)\rangle\}$ the set of additional trajectories depicting trains with trips of length H, that can be added consistently to TS while respecting headway h with an initial speed in interval $[\frac{H}{\beta}, \frac{H}{\alpha}]$. This way of sampling trajectories allows the introduction of random perturbations w.r.t. a chosen speed

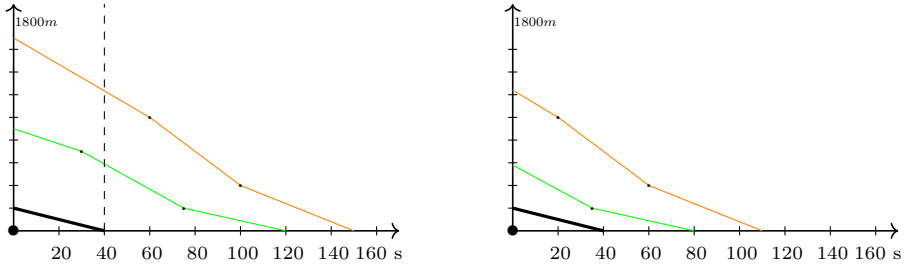


Figure 2 A sequence of trajectories (left) and its left shift by $\delta = 40s$ (right).

profile with an appropriate probability distribution on I. The sampled trajectories can be simple segments respecting headways (such as the black trajectory in Figure 2), or sequences of consecutive segments at distance $\geq h$ from preceding train (such as the orange and green trajectories in Figure 2).

As one can see on Figure 2, trajectories have an intuitive graphical representation, showing, after x time units the distance from a train to the next station. A trajectory that is blocked represents a train that cannot move because it does not have a sufficient headway ahead. Let us now give a graphical intuition of how trajectories evolve when time progress. We will call the *left shift* of a sequence of trajectories by a real value δ the sequence of trajectories obtained by letting trajectories of blocked trains unchanged, and translating all other trajectories by a value $-\delta$. Figure 2 shows an example of left shift. In the original configurations, two trajectories are blocked : a train at distance 0 from arrival (represented by a dot) and its successor, that has to stop to preserve a headway, represented by a thick black line. The next two trains are at a sufficient distance to continue their planned trip during 20 seconds. The left shift of the initial positions of trains by 20 seconds simply consists in a translation of 20 units to the left of unblocked trajectories (erasing the parts of shifted trajectories with negative absiscaes). As for trajectory creation, left shift of a trajectory tr by δ time units is a simple geometrical operation, denoted by $LS(tr, \delta)$, that we define formally in Appendix B. We now have all elements to define our simulation model for metro networks.

▶ **Definition 3.** *A* trajectory net *is a tuple* $\mathcal{N} = (P, T, F, H, I)$ *where* $P = P_C \cup P_T$ *is a set of places partitioned into control places* P_C *and trajectory places* P_T. T *is a set of transitions, and* $F \subseteq P \times T \cup T \times P$. *Maps* $H : P_T \to \mathbb{Q}$ *associates a length, and* $I : P_T \to \mathbb{Q}^2$ *a time interval with each trajectory place of* P_T.

As usual, we denote by ${}^\bullet(t) = \{p \in P \mid (p,t) \in F\}$ the preset of a transition $t \in T$ and by $(t)^\bullet = \{p \in P \mid (t,p) \in F\}$ its postset. A *marking* M of P_C is a map that associates an integral number of tokens with every control place in P_C. We say that M enables a transition $t \in T$ iff, for every place $p \in P_C \cap {}^\bullet(t)$, $M(p) > 0$. The effect of a firing of a transition t on a marking M that enables it is to produce a new marking M' obtained by removing a token from each place in ${}^\bullet(t) \cap P_C$ and then adding a token in each place of $(t)^\bullet \cap P_C$. With a slight abuse of notation, we will write $M' = M - {}^\bullet(t) + (t)^\bullet$. Let $H_{max} = \max_{p \in P_T} H(p)$, and assume a fixed headway h for the whole network. A *trajectory marking* (or TMarking for short) is a map $\mu : P_T \to CTS(H_{max}, h)$ that associates a sequence of trajectories with each place of P_T. We will say that μ enables t if, for every place $p \in {}^\bullet(t) \cap P_T$, $\mu(p)$ is not empty and contains a trajectory of the form $(tr, true)$, with $tr = \langle (0,0)(0,0) \rangle$ depicting a train arrived at the end of its trip. A *configuration* of a net $\mathcal{N}$ is a pair $C = (M, \mu)$ where M is a marking, and μ a TMarking. Figure 3 is a simple trajectory net with two control places p_3, p_4, two trajectory places p_1, p_2 and a transition t. The TMarking of place p_1 is a sequence of two trajectories, where the green trajectory was adapted to preserve safety headways.

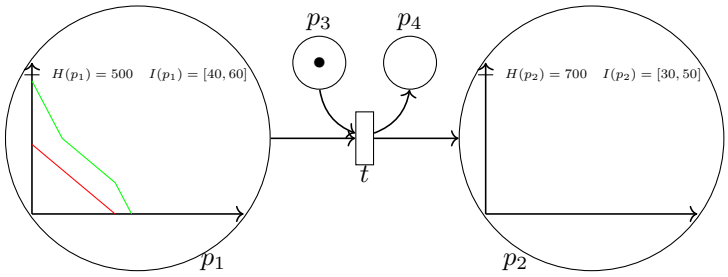


Figure 3 Basic elements of a trajectory net: places, transitions, trajectories.

The semantics of trajectory nets is given in terms of discrete and timed moves from a configuration to the next one. Discrete moves depict arrivals and departures of trains, trajectory changes imposed to preserve safety headways, and timed moves depict how trajectories evolve when time elapses. One can simulate a train waiting in a station S with a trajectory place p_S representing the station, such that $H(p_S) < h$. In this setting, a TMarking of place p_S can contain at most one train, which waiting duration lays in $I(p_S)$. Adding a trajectory in place p_S models an arrival in station, removing a trajectory a departure.

Given a non-empty TMarking $\mu(p) = (tr_0, b_0) \cdots (tr_k, b_k)$ one can easily find the remaining time $\delta_{arr}(p)$ before arrival of a train depicted by trajectories in $\mu(p)$ if tr_0 is not blocked, or $\delta_{block}(p)$ before a trajectory in $\mu(p)$ has to be blocked to preserve consistency of $\mu(p)$. For a sequence $TS = (tr_0, b_0) \cdots (tr_k, b_k)$, $\text{UNBLOCK}(TS) = (tr_0, false) \cdots (tr_k, false)$ is the sequence obtained by unblocking all trajectories in TS.

Consider semantics rule $R1$ below, depicting an arrival or a departure. Roughly speaking, the rule consists in "consuming" a complete trajectory representing a train arrived in station, creating new ones in the places[1] of $(t)^\bullet$, and unblocking trajectories of trains that were blocked until this arrival. The second semantics rule $R2$ below describes how trains can get blocked when there is not enough space ahead to move.

$$(R1)\quad \frac{\begin{array}{c}\text{M enables } t,\ \mu \text{ enables } t,\ M' = M - {}^\bullet(t) + (t)^\bullet \\ \forall p \in P_T \cap {}^\bullet(t), \mu(p) = (\langle (0,0)(0,0) \rangle, true).W \wedge \mu'(p) = \text{UNBLOCK}(W) \\ \forall p' \in P_T \cap (t)^\bullet, \mu'(p') = \mu(p') \cdot (tr, false),\ \text{where } tr \in \text{SAMPLE}(\mu(p'), H, h, I(p))\end{array}}{C = (M, \mu) \xrightarrow{t} C' = (M', \mu')}$$

$$(R2)\quad \frac{\begin{array}{c}\mu(p) = (tr_1, b_1) \cdot (tr_2, b_2) \cdots (tr_i, b_i) \cdots (tr_k, b_k) \\ b_i = false \wedge \big((b_{i-1} = true \wedge tr_i(0) - tr_{i-1}(0) = h) \vee tr_i = \langle (0,0)(0,0) \rangle\big) \\ \mu'(p) = \{(tr_1, b_1), (tr_2, b_2) \cdots (tr_i, b_i' = true) \cdots (tr_k, b_k)\}\end{array}}{C = (M, \mu) \xrightarrow{blockp,i} C' = (M, \mu')}$$

Timed moves just let time elapse. We adopt an urgent semantics: a timed move of duration δ is allowed from configuration C if and only if no discrete move is allowed in C. Unsurprisingly, letting time pass is modeled by a left shift of TMarkings. However, blocked trajectories represent trains that cannot move without violating a safety headway. Hence, the left shift of a TMarking $\mu(p) = \{(tr_1, b_1) \cdots (tr_k, b_k)\}$ by a duration δ is a new TMarking $\{(tr_1', b_1) \cdots (tr_k', b_k)\}$ where $tr_i' = LS(tr_i, \delta)$ if $b_i = false$ and $tr_i = tr_i'$ otherwise.

[1] To represent metro networks, one only needs $|(t)^\bullet| = 1$ but this restriction is not needed in the semantics.

$$(R3)\quad \frac{\begin{array}{c}\delta \in \mathbb{R}^{>0} \\ \forall t \in T, M \text{ does not enable } t \vee \mu \text{ does not enable } t \\ \forall p \in P_T \text{ such that } \mu(p) \text{ is defined } \delta \leq min(\delta_{arr}(p), \delta_{block}(p)) \\ \forall p \in P_T, \mu'(p) = LS(\mu(p), \delta)\end{array}}{C = (M, \mu) \xrightarrow{\delta} C' = (M', \mu')}$$

With these semantic rules, we can simulate the behavior of a metro network represented by a trajectory net. Arrivals and departure are symbolized by transitions firings, and result in the sampling of a new trajectory that is consistent with the possible speeds of trains and with the TMarking of the new track entered. An interesting feature of Rule 3 is that one can directly elapse time for a duration $\delta = \min\limits_{p \in P_T} \min(\delta_{arr}(p), \delta_{block}(p))$, i.e. progress time up to the date of the next discrete event : arrival/departure in station, or blocking of a trajectory. With this approach, one does not need to sample a discrete clock and compute what happened at each time step. This approach is called *event-based simulation*, and considerably speeds up simulation of metro networks. One can also see from these rules the advantages of using Petri nets variants: in each discrete rule, the effect of an event changes the contents of a single place, or the preset/postset of a single transition.

We can now define runs of a trajectory net. A *run* is a sequence $\rho = C_0 \xrightarrow{\delta_0} C_1 \xrightarrow{e_0} C_1 \cdots$ where each C_i is a configuration, $C_i \xrightarrow{\delta_i} C_{i+1}$ is a legal time move, and $C_i \xrightarrow{e_i} C_{i+1}$ is a legal discrete move (e_i is either a transition firing, or the blocking of a trajectory in a place). Note that the behavior of a net starting from a configuration C_0 is not deterministic, as transitions firings sample random durations for newly created trajectories.

Despite their apparent simplicity, trajectory nets allow for the design of complex topologies of metro networks involving forks, joins, reversal or garage zones for trains, ... Using control places filled at the appropriate moment, one can guide a train reaching a fork to enter the next track segment on its trip. We refer interested reader to Appendix F for examples.

3 Passenger flows

The model of Section 2 does not consider passengers, but only possible moves of trains. In this section, we model passenger flows as a side quantitative information evolving with runs of a trajectory net, i.e. as quantities representing crowds, that are updated at each timed and discrete move. First of all, passenger flows are not constant over time, and depend on particular scenarios. In most cities, during a working day, one can observe peak hours during which commuters move from their home to their workplace, and the way back in the evening. Scenarios for passenger flows during weekends differ w.r.t. dates and duration of peaks, number of moving passengers or destinations. To control efficiently a transport network, i.e. provide the expected transport offer but nevertheless use the right amount of resources, operators use different policies for each scenario. In this section, we first explain how scenarios for passenger flows are usually represented in metro networks (using Origin-Destination matrices), how connections between parts of a network (corridors or access to quays) are specified. We use these representations to decorate in a consistent way runs of our metro model with quantities depicting passengers that are moving towards a platform, waiting on a platform, or traveling in a train.

Roughly speaking, passenger behaviors (arrivals and final destination) are encoded as sequences of origin/destination matrices, connections as queues, and train payloads as vectors containing the number of passengers alighting at each destination in the network. To avoid overloading our model, we consider that all passengers move at identical constant speed

from a point to the next one. This modeling choice may result in slight modification in the duration of transit times. Indeed all passengers do not move at the same speed, corridors may contain bottlenecks, and paths from a platform to another one on a different line may include stairs, etc. However, there is no clear consensus on how passenger flows should be modeled. It seems that for corridors with bottlenecks, the throughput depends mainly on the width of the bottleneck [21]. We will hence model passengers moving in a station by queues of integers. In a similar way, there is no canonical model to represent passengers entering or leaving a trains, and the time needed to load and unload may depend on characteristics of the rolling stock (e.g. the number of doors, their width and their position) [16]. Here again, we will adopt a simple model, and consider the exchanges that occur depending on the dwell time, passengers on platform, and available space in trains.

▶ **Definition 4.** *Let $\mathcal{S} = \{S_1, \cdots S_m\}$ be a list of stations in a metro network, and let us fix a duration Δ. An* Origin Destination Matrix *(ODM for short) is an integer matrix OD indexed by $\mathcal{S}$. Each entry $OD[s_i, s_j]$ represents a number of passengers willing to travel from station s_i to station s_j in Δ time units. A* scenario *is a sequence $OD_1, OD_2, ..., OD_n$ of origin destination matrices. The* duration *of a scenario is $n \cdot \Delta$*

An ODM depicts the number of passengers traveling from one origin station (represented by a row in the matrix), to a destination station (represented by a column) during a duration of Δ time units. The origin and destination stations need not be located on the same subway line. This means that passengers may have to use interconnection at stations shared by several lines, and move from one platform to another one to continue their journey. Hence, the path followed by passengers will have an impact on passenger flows in the transit areas at connection points. A metro network can be seen as a graph depicting connections from a station to the next one along each line and interconnections between two lines. We assume that this graph is consistent with the flow relation of the trajectory Petri net modeling our network. We also assume that passengers always follow the shortest path (in terms of distance or in terms of number of stations) in this graph when traveling from an origin station S_o to a destination station S_d. With this assumption one can compute the shortest path for every pair (S_o, S_d) of origin/ destination, using for instance the well known Floyd-Warshall algorithm (see for instance [5]). For completeness, we provide this algorithm in Appendix D. Let us assume that we have obtained a matrix M_D giving the shortest distance between any pair of connected stations. Then, we can compute a matrix M_{NS} such that $M_{NS}(S_i, S_j) = k$ if station S_k is the next station to visit on an optimal path going from S_i to S_j. When one knows the final destination of passengers, this matrix M_{NS} allows to decide, when a train stops at a station, how many passengers leave the train, and how many passengers continue their journey to the next station on the same line. The calculus of M_{NS} can be done with a simple extension of the Floyd-Warshall algorithm, given in Appendix E. Notice that this way of modeling passengers choices is a way to keep the model simple. We could make passenger choices more complex, e.g. allow choices of alternative routes to a destination, or choice of a particular line to go from an origin to a destination. Note that line information is not needed when lines do not share common track portions (as in the metro of Montreal).

Each ODM is used to depict passenger arrivals and destinations in the network for a duration Δ. Scenarios allow changes in the arrival dates at each station, at each period of the day. Let $d \leq \Delta$. Assuming that an ODM OD represents faithfully passenger trips, the number of passengers entered in the network at station S_i willing to go to station S_j is $\lfloor OD[i, j] \cdot (d/\Delta) \rfloor$. Remembering the time t elapsed since the beginning of a simulation, one can compute how many passengers have entered the network and their destination at time $t + \delta$. Note that this calculus may involve contents of several OD matrices.

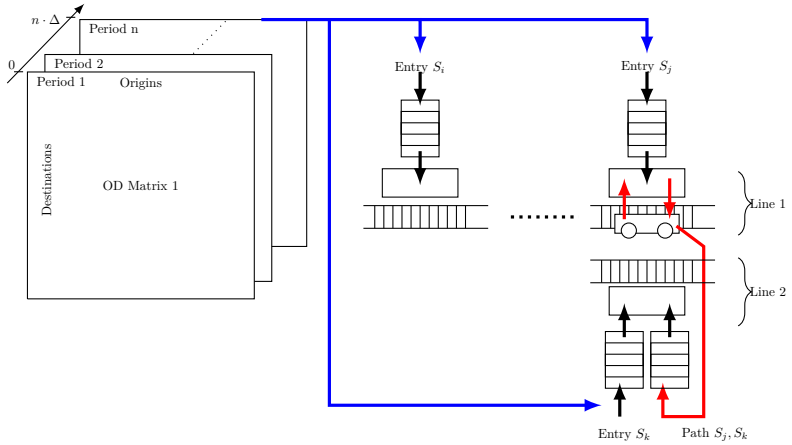


Figure 4 Integration of scenarios and passenger flows in the model.

At each instant, one wants to remember the number of passengers for a destination in a train, on platforms, in corridors connecting one entry to a platform, or on platform serving a line to another platform serving a different line in the same station. We define a *payload* as a vector of integers indexed by station, that is vectors of the form $PL = [v_1, \cdots, v_n]$. The interpretation is the following: if PL describes the passengers of a train, then $PL[i]$ indicates the number of passengers willing to reach station s_i in that train. Obviously, when a train alights at station S_i, then at least $PL[i]$ passengers leave the train. Further, if S_i is the last station on the same line on a shortest path to station S_k, then $PL[k]$ passengers leave the train too, and move towards a corridor connecting the reached platform S_i to the platform $MN[S_i, S_k]$, that is the next step on the journey to S_k. We assume that passengers do not make mistakes, i.e. they all follow the shortest path to their destination. Payloads are also used to represent awaiting passengers on a platform: $PL[i]$ is then the number of passengers that did not yet board a metro and are waiting for a train to go to station S_i.

Let us now add corridors to the model. Corridors connect entries of the network to platforms, and bridge lines to allow journeys involving several metro lines. As for trains and platforms, one has to remember the number of passengers moving in a corridor and their final destination. However, passenger moves in corridors take time, and cannot be represented with a payload. We use FIFO queues to represent passenger flows.

▶ **Definition 5.** *A* passenger queue *is an tuple $Q = (l_Q, sp_Q)$ where l_Q is a length (in meters), sp_Q is an average speed. A state of a passenger queue Q is a vector of payloads M_Q indexed by $1..l$. We will denote by $M_Q[k]$ the passengers at distance k from the beginning of the queue, and $M_Q[k][j] = n$ means that at distance k, n passengers are willing to go to station S_j.*

Each entry $M_Q[k]$ in a passenger queue represents a section of 1 meter on a path from an origin (a platform or a station entry) to a destination (a platform). We take some simplifying assumptions : the speed of passengers is constant in the whole space represented by Q and is the same for all passengers. Passengers do not interact. With these assumptions, we can simply remember with $M_Q[k][j]$ how many passengers willing to go to station S_j are at distance k from the next platform. So, using passenger queues, one need not assign a behavior to every passenger in the network (which would be too costly to simulate). More complex models for crowd behaviors have been proposed [12, 16], but we are mainly interested in durations of transfers and in destinations of passengers, not in individual movements. Further, modeling flows with queues allow for an easy calculus of the evolution of crowds. Let M_Q be the current state of a queue $Q = (l, sp)$ from an entry at station S_i to a platform, d be the

current date. Assume that the played scenario is $OD_1, OD_2, ..., OD_n$. We can easily compute how the state of Q evolves within δ time units. Passengers in M_Q at distance $x \leq \delta \cdot sp$ have arrived, so the corresponding sum of payloads from 1 to x is added to the payload of the platforms. Passengers in M_Q at distance $y > x + 1$ simply progressed in the queue, i.e. $M_Q[y - \lfloor \delta \cdot v \rfloor] = M_Q[y]$. Last, assuming that d and $d + \delta$ belong to the same period, i.e. arrivals are given by a single ODM OD, we have $OD[i][j] \cdot \frac{\delta}{\Delta}$ new passengers arriving in Q willing to go to station S_j, and spread uniformly on entries $\delta \cdot sp \ldots l$ of the queue. In practice, computing the actualized contents of queues within δ time units may involve several OD matrices, and one has to pay attention to rounding to avoid loosing passengers. To avoid heavy notations, we leave the complete definition of passenger moves to a more formal definition provided in Appendix G, and we will simply write $M'_Q = Update(d, \delta, M_Q)$ to denote that M'_Q is the state of queue Q after δ time units have elapsed since date d. Similar rules apply for queues representing connections between platforms. In a similar way, we can represent how the payload of a train evolves when time elapses. When a train arrives at station S_i, passengers alighting at S_i leave the train, and either leave the system, or move to the queue representing the connections to another platform. Then, passengers on platform board the train up to the maximal capacity CT_{max} of the vehicle. We refer to the model of [12] to compute the time needed by n passengers to leave a train. We set $t_{alighting} = \lceil n \cdot D_{width} \times D_{nb} \rceil \times t_a$ where D_{width} is the width of a door in meter, D_{nb} is the number of doors of the vehicle and t_a is the time for one passenger to alight, estimated in seconds. Note that a train cannot leave if its passengers are still alighting, so when $t_{alighting}$ is greater than the dwell time planned, the train is delayed. When all passengers alighting at s_i have left the train, and the payload of the train is TP then the free space can be used to let $n \leq CT_{max} - \sum TP_k$ passengers enter the train. Boarding is similar to alighting time, i.e. we have $t_{boarding} = \lceil n \cdot D_{width} \times D_{nb} \rceil \times t_b$, where t_b is the average time needed by a passenger to board a train. Here, if $t_{boarding}$ is larger than the remaining dwell time of a train, then our model assumes that doors close, and that some passengers fail to board.

As the remaining space, or the dwell time of trains might not allow all passengers to board, some of them will stay on the platform. Failures to board are an important performance indicator, as this is one of the quantitative aspects that characterizes users' satisfaction. As for queues, we will denote by $TP'_i = Update(d, \delta, TP_i)$ the change of payload in train T_i within δ time units, and $P'_i = Update(d, \delta, P_i)$ the change in platform i's payload. Figure 4 gives an illustration of passenger flows between platforms, corridors, entries and trains.

Let us now reconnect trajectory nets, passenger queues and scenarios. The global state of a simulation is represented by a tuple $\big(C, d, (P_i)_{i\in 1..K_P}, (TP_i)_{i\in 1..K_T}, (M_{Q_i})_{i\in 1..K_Q}\big)$ where C is a configuration of the trajectory net depicting the physical network, d is a date, $(P_i)_{i\in 1..K_P}$ represents platforms crowds, $(TP_i)_{i\in 1..K_T}$ represents train payloads, and $(M_{Q_i})_{i\in 1..K_Q}$ are the queues contents. Let t be an arrival of a train in station s_i after a delay δ. The new state of the system is $\big(C', d + \delta, (P'_i)_{i\in 1..K_P}, (TP'_i)_{i\in 1..K_T}, (M'_{Q_i})_{i\in 1..K_Q}\big)$, where C' is the configuration reached by the trajectory net after elapsing δ time units and firing t, i.e., such that $C \xrightarrow{\delta} C'' \xrightarrow{t} C'$, and $P'_i = Update(d, \delta, P_i)$, $TP'_i = Update(d, \delta, TP_i)$, $M'_{Q_i} = Update(d, \delta, M'_{Q_i})$. Similar rule applies when blocking operations occur. At each use of an operational rule for a train movement, at most one train has its payload updated, but the contents of all queues and platforms change. So, one has to perform a number of updates in $O\left((K_p + K_Q \cdot L_{\max} + 1) \cdot S\right)$, where $L_{\max}$ is the maximal length of a queue. Experimental results in the next section show that this simulation model is efficient, and allows for fast simulation of a complete day of metro operation.

4 A case study: The metro of Montreal

We consider a real-size case study, namely the Metro network of Montreal (see [6] and Appendix H for a complete map). The network is composed of three main lines (orange, green, blue) plus a short yellow side line. The orange line is 31km long, it has 42 trains and 31 stations, which corresponds to 128 places and transitions in our net model. The green line is 22km long, it has 34 trains, which corresponds to 108 places and transitions in our net model. The blue line is 9km long, with 32 trains, which corresponds to 48 places and transitions. In addition to the trajectory net defining the physical behavior of trains, queues have been created for each platform, and set a limit for the capacity of platforms. We choose a default queue distance of 10m for corridors from an entry to a platform, a platform capacity of 1000 awaiting passengers, and a train capacity of 1000 passengers. We represent the 4 interconnections in the network with 8 queues (one per direction) of 50m.

The control part of the trajectory is extracted from a real GTFS file [8] (a standard proposed by Google in 2006 for timetable description). This public data depicting one day of operation on Montreal's network is available on the STM site [22]. From this timetable, we generate a list of events associating a date, a vehicle id and a transition to fire. This timetable allows to feed the control part of our model to trigger train departures. If needed, delays can be inserted to simulate small perturbations (e.g. doors blocking) or a more serious anomalies causing a technical interruption of traffic.

We simulated the trajectory net depicting Montreal's network with the MOCHY tool [14], an Open-GL software developed for the simulation of transport networks described with variants of Petri nets. The experimentation was conducted with a weekday scenario repeating the same OD matrix representing one hour of passenger flows. In this matrix, for every entry $OD[i, j]$, we generated randomly a high number of passengers entering the network at station S_i and willing to go to destination S_j. This number of passengers per hour was generated with a uniform law to sample a value ranging from 0 to 100. Multiplying an average number of 50 passengers per hour by the number of stations in the network rapidly leads to large flows of incoming passengers. With this scenario, we can model a peak period: in all stations, a large number of passengers arrive and travel from their home place to their work. In weekdays scenarios, the payload of all trains increase up to their full capacity if the operators do not inject new trains in the network. The objective of this simulation was to consider the effects of train insertion during peak hours, to verify that this policy influences train payloads. Overall, simulating a full day of operation for the three main lines with 110 trains and their passenger flows takes between 1 and 2 minutes on a standard laptop (Intel core i7). Our first simulation results are represented in Figures 5 and 6. Figure 5 is a space-time diagram drawn from our simulation logs. It represents train moves between station Montmorency and station Cote-Vertu on the Orange line from 5:00 to 8:20. The horizontal axis represents time. The vertical axis represents localization of trains, labeled by station names plus a garage, and ordered according to their position on the line. Each colored line represents a train. This simulation follows exactly the planned timetable, i.e, events have been realized exactly at their planned dates. The timetable was designed to guarantee regularity of train departures at each station, and to increase the frequency of trains to absorb passenger peaks. On the space time diagram, one can observe regularity of service: train trajectories are parallel lines. When a train is inserted, trajectories adapt and get closer, as expected.

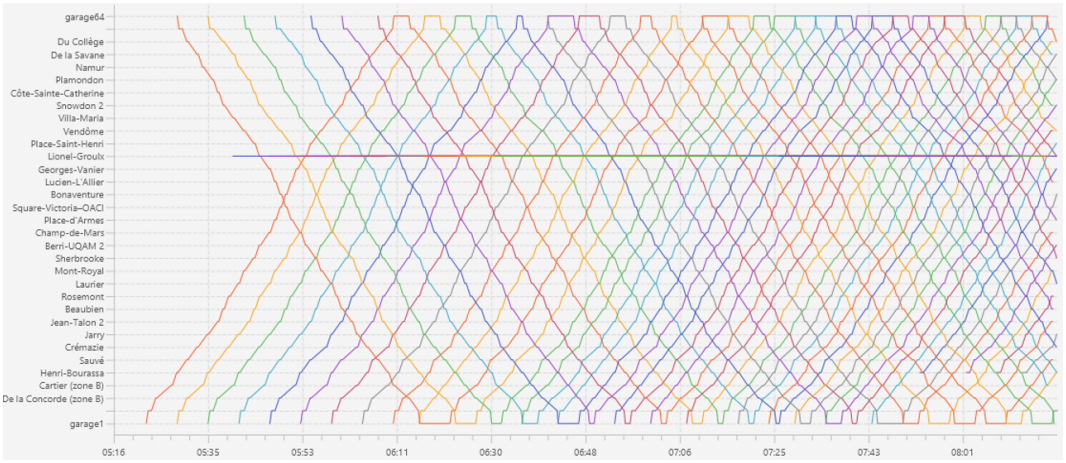


Figure 5 A Space-Time diagram for the Orange line.

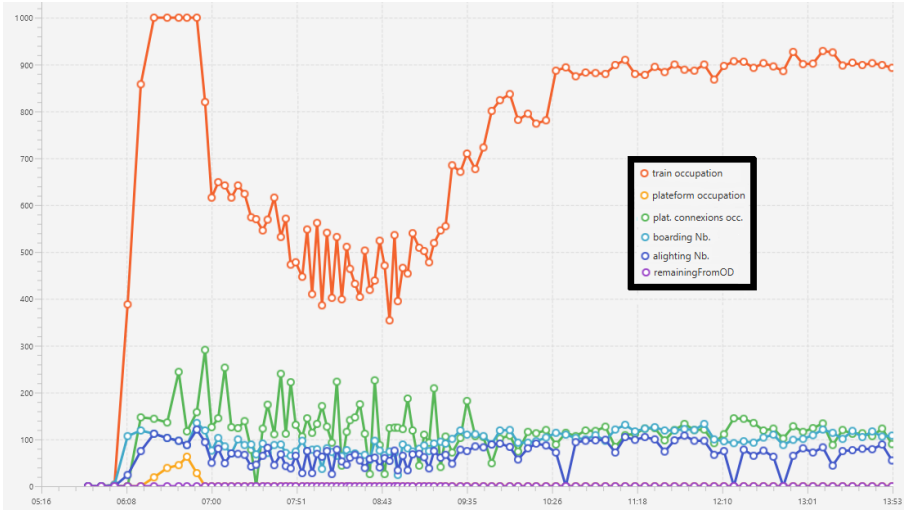


Figure 6 Passenger flows at Berri-Uqam Orange line.

Figure 6 gives information on passenger flows at station Berri-UQAM (BUQ), more precisely for the platform located on the orange line. This station is an important node in the network, because it is an interconnection between the orange and green lines. The horizontal axis represents time, and the vertical axis a number of passengers. Each point on each curve was recorded at the date of closure of train doors, i.e. when a train is about to leave the station. The orange curve describes occupation of trains that stop at the considered platform. As one can see the trains rapidly fill at their maximum capacity between 6:00 and 7:00. The green curve represents passengers who alighted at BUQ, and will move to the green line in the next coming seconds, the light blue curve represents the number of passengers that boarded the currently stopped train. The dark blue curve represents the number of passengers who alighted at that station but do not connect to the green line (they will exit the station). Last, the yellow curve represents passengers who failed to board because the train was full. This is an important indicator, that impacts user satisfaction. The pink curve depicts the number of passengers who had to wait before entering the station (this value remains low during the depicted simulation because platforms are never full).

The simulation is done with heavy passenger flows conditions from early morning until 12 o'clock. The consequence is that trains are almost full on the whole time period shown in Figures 5 and 6 except for the period 07:00 - 10:00, when frequency of trains is increased sufficiently to reduce platforms occupation and hence avoid failures to board. Then, from 7:00 to 9:30AM, train occupation is low to 50-60% and no passenger fails to baord. After 10:00AM, the number of trains on the network is reduced. As the incoming passenger flows remain high on all lines, trains occupation increases. However frequency of stops remains sufficient to keep trains occupation at 90% of their maximal capacity, and the number of boarding and alighting passengers remains rather stable after 10:00 during the simulation.

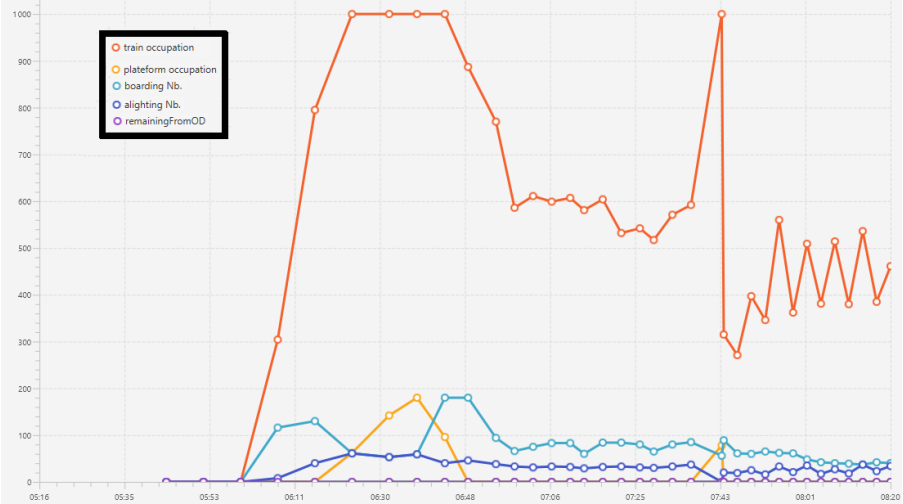


Figure 7 A delay at station Sauve creates a passengers peak at Beaubien.

In a second experiment, we studied the consequence of a 10min delay at 7:30AM at station Sauvé (the 5^{th} station on the Orange line) with the same weekday scenario. For space reasons, the space-time diagram of this scenario appears in Appendix I. The networks takes around 30 minutes to recover from this primary delay. Figure 7 shows the consequences of this perturbation on passengers at station Beaubien (located 4 stations after Sauvé on the orange line). A peak in trains occupation arises at 7:43AM. It is due to the increased time gap between train departures, but also to a larger number of passengers boarding at the previous stations for the same reason.

5 Conclusion

We have proposed a model for metro networks with their passenger flows. This extension of Petri nets includes time, positions of moving objects, and queues to represent passenger moves between platforms and metros. The proposed semantics allows fast simulation, by computing the state of the system at each discrete event occurrence. At each event, the time elapsed is known, which is sufficient to update trajectories and passenger flows. Simulation of a complete day of operation is fast enough to handle real-size cases such as the Metro of Montreal, and study the effects of operational choices on passenger flows.

A first direction to extend this work is to consider other values than the number of passengers. For instance, considering the energy consumed by a metro network is a crucial need. Another challenging task is to address traffic management as a controller synthesis problem, building controllers that optimize a quantitative criterion (energy, nb. of failures to

board...). Standard approaches of control (e.g. à la [4]) will not work for metros, due to the number of states to consider. Solutions to build effectively optimized traffic management algorithms may come from abstraction, approximation, and possibly learning techniques. Last, even if this model is tailored for metros, the concurrent nature of the model should be useful in a setting where objects are most of the time independent, and interact only in local delimited areas, such as road sections or automated plants.

References

1 Y. Ahn, T. Kowada, H. Tsukaguchi, and U. Vandebona. Estimation of passenger flow for planning and management of railway stations. *Transportation Research Procedia*, 25:315–330, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016. doi:10.1016/j.trpro.2017.05.408.

2 R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

3 N. Bertrand, B. Bordais, L. Hélouët, T. Mari, J. Parreaux, and O. Sankur. Performance evaluation of metro regulations using probabilistic model-checking. In *Proc. of RSSRail 2019*, volume 11495 of *Lecture Notes in Computer Science*, pages 59–76, 2019.

4 C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems, Third Edition*. Springer New York, 2021. doi:10.1007/978-0-387-68612-7.

5 T. H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

6 Société des Transports de Montréal. GTFS Static Overview. https://www.stm.info/sites/default/files/media/Stminfo/images/plan-metro-blanc.pdf, 2024.

7 Rail Management Consultants International GmbH. Railsys suite. https://www.rmcon-int.de/home-en/.

8 Google. GTFS Static Overview. https://developers.google.com/transit/gtfs, 2022.

9 S. Guanghui, S. Bingfeng, Z. Kun, Z. Ben, and Z. Xuanchuan. Simulation-based method for the calculation of passenger flow distribution in an urban rail transit network under interruption. *Urban Rail Transit*, 9:110–126, 2023. doi:10.1007/s40864-023-00188-z.

10 R. Haehn. *Optimisation and Analysis of Railway Timetables under Consideration of Uncertainties*. PhD thesis, Aachen University, 2022.

11 R. Haehn, E. Ábrahám, and N. Kotowski. Acceleration techniques for symbolic simulation of railway timetables. In *Proc. of RSSRail'22*, volume 13294 of *Lecture Notes in Computer Science*, pages 46–62, 2022.

12 N. Harris. Train boarding and alighting rates at high passenger loads. *Journal of Advanced Transportation*, 40:249–263, June 2006. doi:10.1002/atr.5670400302.

13 L. Hélouët and P. Agrawal. Waiting nets. In *Proc. of PETRI NETS'22*, volume 13288 of *Lecture Notes in Computer Science*, pages 67–89. Springer, 2022.

14 L. Hélouët and A. Thébault. Mochy: A tool for the modeling of concurrent hybrid systems. In *Application and Theory of Petri Nets and Concurrency*, pages 205–216, 2023.

15 K. Kecir. *Performance evaluation of urban rail traffic management techniques*. PhD thesis, Universite de Rennes 1, 2019.

16 N. Luangboriboon, S. Seriani, and T. Fujiyama. The influence of the density inside a train carriage on passenger boarding rate. *International Journal of Rail Transportation*, 9(5):445–460, 2021.

17 E.W. Mayr. An algorithm for the general petri net reachability problem. *SIAM Journal on Computing*, 13(3):441–460, 1984.

18 P. M. Merlin. *A Study of the Recoverability of Computing Systems*. PhD thesis, University of California, Irvine, CA, USA, 1974.

19 G. Michal, N. Huynh, N. Shukla, A. Munoz, and J. Barthelemy. Railnet: A simulation model for operational planning of rail freight. *Transportation Research Procedia*, 25:461–473, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016. doi:10.1016/j.trpro.2017.05.426.

20 A. Nash and D. Huerlimann. Railroad simulation using OpenTrack. In *Computers in Railways IX*, pages 45–54, 2004.

21 A. Seyfried, T. Rupprecht, O. Passon, B. Steffen, W. Klingsch, and M. Boltes. New insights into pedestrian flow through bottlenecks. *Transportation Science*, 43:395–406, March 2007. doi:10.1287/trsc.1090.0263.

22 STM. STM Developers section. https://www.stm.info/en/about/developers, 2023.

A Computing $\uparrow(tr_{k+1}, TS, h)$

Let $tr_k = \langle (x_0, y_0)(x_1, y_1)\rangle \cdots \langle (x_{n-1}, y_{n-1})(x_n, y_n)\rangle$, and let $d_h \in [x_m, x_{m+1}]$ be the date where $\overbrace{tr_{k+1}}(d_h) = tr_k(d_h) + h$. Then

$$\begin{aligned}\uparrow(tr_{k+1}, TS, h) = \quad & \langle (0,H).(d_h, tr_k(d_h)+h)\rangle \cdot \langle (d_h, tr_k(d_h)+h)(x_{m+1}, y_{m+1}+h)\rangle \cdots \\ & \cdots \langle (x_{n-1}, y_{n-1}+h)(x_n, y_n+h)\rangle \cdot \langle (x_n, y_n+h)(x_n + h\cdot\tfrac{d}{H}, 0)\rangle\end{aligned}$$

B Formal definition of Left Shift

▶ **Definition 6.** *The* left shift *of a segment $s = \langle (x_{i-1}, y_{i-1})(x_i, y_i)\rangle$ is defined if $x_i \geq \delta$, and is a segment $LS(s,\delta) = \langle (x'_{i-1}, y'_{i-1})(x'_i, y'_i)\rangle$, where : $x'_{i-1} = \max(0, x_{i-1} - \delta)$, $x'_i = x_i - \delta$, $y'_{i-1} = y_{i-1} + \delta \cdot \frac{y_i - y_{i-1}}{x_i - x_{i-1}}$, $y'_i = y_i$.*

Let (tr, b) be a trajectory with $tr = s_1 \cdot s_2 \cdots s_k$, and let $\delta \leq x_k$. The left shift of (tr, b) by duration δ, is denoted $LS(tr, b)$. If b is true, then $LS(tr, b) = (tr, b)$. If b is false, then $LS(tr, b) = LS(S_{i_\delta}, \delta) \cdots LS(s_k, \delta)$, where i_δ the index of the first segment such that $LS(s_i, \delta)$ is defined.

Notice that as soon as $\delta \leq x_k$, index i_δ exists.

C Computing $\delta_{arr}, \delta_{block}$

Let us now detail, for a given configuration $C = (M, \mu)$ and a given place p how to compute the value of $\delta_{arr}(p)$ and $\delta_b lock(p)$ when $\mu(p) = tr_1 \cdot tr_2 \cdots tr_k$. First, the arrival date to consider is the date of the first unblocked trajectory. Let i be the index of this trajectory, and let $tr_i = s_1 \dots s_q$ with $s_q = \langle (x_{q-1}, y_{q-1})(x_q, 0)\rangle$. Then, the train represented by trajectory tr_i can only arrive in station within x_q time units, so $\delta_{arr}(p) = x_q$. In the same setting, one can compute the remaining time before train represented by tr_i has to brake to maintain a safety headway h. Letting $tr_i^{-1}(y)$ denote the date x at which $traj_i(x) = y$, then the train has to adapt its speed at date $\delta_{block}(p) = tr_i^{-1}\big((n-1)\cdot h\big)$.

D Algorithm to compute passenger paths

Algorithm 1 Floyd-Warshall Distance Matrix.

parameter N : the set of ids of the places
the cells of the matrix are initially null
M_D : the distance square matrix from an origin to a destination
for all $(id_1, id_2) \in A$ **do** ▷ *Matrix initialization*
    $M_D(id_1, id_2) = 1$ ▷ *arcs weights are set to 1*
end for
for all $k \in N$ **do** ▷ *Distance Matrix Building*
    for all $i \in N$ **do**
        for all $j \in N$ **do**
            $v \leftarrow null$
            if $M_D(i,j)! = null$ **then**
                if $M_D(i,k)! = null$ AND $M_D(k,j)! = null$ **then**
                    $v \leftarrow min(M_D(i,j), M_D(i,k) + M_D(k,j))$
                else
                    $v \leftarrow M_D(i,j)$
                end if
            else if $M_D(i,k)! = null$ AND $M_D(k,j)! = null$ **then**
                $v \leftarrow M_D(i,k) + M_D(k,j)$
            end if
            $M_D(i,j) \leftarrow v$
        end for
    end for
end for

E Algorithm to compute the next station on a trip

Algorithm 2 Floyd-Warshall Next step Matrix.

parameter M_D : from Floyd-Warshall Distance Matrix algorithm
parameter N : the set of ids of the places
the cells of the matrix are initially null
M_{NS} : the next step squared matrix from an origin to a destination
for all $(id_1, id_2) \in A$ **do** ▷ *Matrix initialization*
  $M_{NS}(id_1, id_2) = id_2$
end for
for all $k \in N$ **do** ▷ *Next Step Matrix Building*
  for all $i \in N$ **do**
    for all $j \in N$ **do**
      $v \leftarrow null$
      if $i! = j$ **then**
        if $M_D(i,k) == null$ OR $M_D(j,k) == null$ **then**
          $v \leftarrow M_{NS}(i,j)$
        else if $M_D(i,j) == null$ **then**
          $v \leftarrow M_{NS}(i,k)$
        else if $M_D(i,j) \leq M_D(i,k) + M_D(k,j)$ **then**
          $v \leftarrow M_{NS}(i,j)$
        else
          $v \leftarrow M_{NS}(i,k)$
        end if
      end if
      $M_{NS}(i,j) \leftarrow v$
    end for
  end for
end for

F Complex network patterns with Trajectory nets

Consider for instance Figure 8-a). This piece of trajectory net represent a typical pattern to design forks. In this Figure, a train represented by a segment in place $P_{T,A}$ can be guided to a track U (represented by place $p_{T,U}$) if place $p_{c,U}$ is filled or to track D (represented by place $p_{T,D}$) if place $p_{c,D}$ is filled. Figure 8-b) represents another typical pattern appearing in networks, namely the en of track and the turn back procedure. Form these two examples, one can see that filling control places at the right moment is a way to control arrivals, departures and directions of trains. If timetable is provided, one can even implement it with a controller that fills the appropriate places at the planned departure dates. A pattern of the form of the net given in Figure 9 can also be used to represent a garage, initially filled with a sequence of trajectories representing the available fleet of vehicles.

G Uptading queue, platform and train payloads

Let us detail how passengers waiting on quays, walking, or entered in a train are updated between date d and $d+\delta$. Let $\big(C, d, (P_i)_{i\in 1..K_P}, (TP_i)_{i\in 1..K_T}, (M_{Q_i})_{i\in 1..K_Q}\big)$ be the current configuration, and $Scen = OD_1 \cdots OD_K$ be the scenario used for simulation.

Figure 8 Fork from section A to sections U or D controlled by place $p_{c,U}$ and $p_{c,D}$.

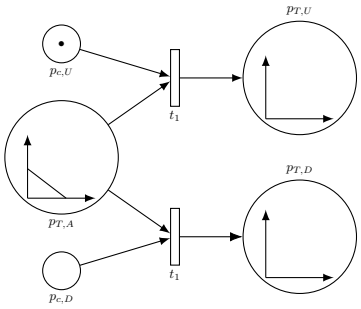
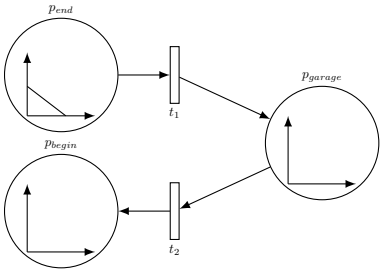


Figure 9 Garage and Reversals.



Let P_i be the contents of a platform at a station S_i, and $Q_i = (l_i, v_i)$ be the queue relating gates at the platform at station S_i. Let M_{Q_i} denote the state of Q_i.

The new state of M_{Q_i} in C' after δ time units is $M'_{Q_i} = M^{\leftarrow\delta}_{Q_i} + Arr_{M_{Q_i}}(Scen, d, d+\delta)$, where $M^{\leftarrow\delta}_{Q_i}$ is the left shift of queue contents, i.e. $M_{Q_i}[\lfloor y - \delta \rfloor \cdot v_i] = M_{Q_i}[y - \delta \cdot v_i]$, and $Arr_{M_{Q_i}}(Scen, d, d+\delta)$ is the payload representing passengers arrived on platform S_i between date d and $d + \delta$.

Formally, let k_1, k_2 be two integers such that $d \in [(k_1 - 1) \cdot \Delta, k_1 \cdot \Delta]$ and $d + \delta \in [(k_2 - 1) \cdot \Delta, k_2 \cdot \Delta]$. Then

$$Arr_{M_{Q_i}}(Scen, d, d+\delta)[x] = \begin{cases} OD[k_2] \cdot \frac{1}{\Delta} \text{ if } x \in [0, d + \delta - k_2] \\ OD[k_2 - i] \cdot \frac{1}{\Delta} \text{ if } x \in [d + \delta - k_2 + 1, d + \delta - k_2 - \Delta \cdot i] \\ OD[k_1] \cdot \frac{1}{\Delta} \text{ if } x \in [(k2 - k1 - 1) \cdot \Delta, \delta] \end{cases}$$

Let us now consider the contents of platform P_i when no train is at station S_i. The payload of the platform is incremented by the number of passengers arrived from the entry gate plus queues arriving on the platform. That is, $P'_i[k] = \sum_{x \in 0..\delta} M_{Q_i}[x][k] \sum_{SQ_{j \to i}} \sum_{x \in 0..\delta} M_{Q_{j,i}}[x][k]$ where $SQ_{j \to i}$ is the set of queues from a platform j to a platform j.

Let us now consider the situation where a train with payload TP_m is stopped at station S_i We consider a simplified model for exchanges, where passengers board a train when all passengers leaving at station S_i have left the train. We consider that leaving a train takes a constant time Δ_{lt} per passenger, and similarly that boarding takes time Δ_{bt}.

If $TP_m[S_i].\Delta_{lt} > \delta$ then $TP'_m[S_j] = TP_m[S_j]$ (i.e., if the time needed to unload the train at station S_i is larger than δ) then for every $S_j \neq S_i$ and $TP'_m[S_i] = TP_m[S_i] - \lfloor \delta / \Delta_{lt} \rfloor$.

If $TC(p)[S_i].\Delta_{lt} \leq \delta$ then $TP'_m[S_i] = 0$, then the allowed number of passengers after all passengers stopping at S_i have exited the train is $maxboard = TC_{max} - \sum_{S_j \neq S_i} TP_m[S_j]$. The number of passengers who will board also depends on the remaining available time, i.e., $\delta_b = \delta - TP_m[S_i] \cdot \Delta_{lt}$.

Then, the number of passengers boarding who will board is: $nbp_{C,\delta} = \max(\frac{\delta_b}{\Delta_{bt}}, maxboard)$

We divide the stations in two groups: $S^{=}$ containing stations S_k such that $P_i[S_k] < nbp_{C,\delta}/|\mathcal{S}|-1$, and S^{+}, containing stations such that $P_i[S_k] \geq nbp_{C,\delta}/|\mathcal{S}|-1$ For every station $S_k \in S^{=}$ we set $P'_i[S_k] = 0$ and $TP'_m[S_k] = TP_m[S_k] + P_i[S_k]$. The remaining number of passengers is $nbpr_{C,\delta} = nbp_{C,\delta} - \sum_{S_k \in S^{=}} P_i[S_k]$ This number of passengers is fairly distributed on other destination, [2], by allowing n_k passengers to board, with $n_k = \lfloor nbpr_{C,\delta}/|S^{+}| \rfloor$ or $n_k = \lfloor nbpr_{C,\delta}/|S^{+}| \rfloor + 1$, i.e. setting $P'_i[S_k] = P_i[S_k] - n_k$ and $TP'_m[S_k] = TP_m[S_k] + n_k$.

H Map of the metro network in Montreal

Figure 10 The STM Montreal Network.

[2] To distribute fairly these passengers, we memorize the last destinations with the most passengers, or choose randomly at each round which destinations receive an additional passenger.

I Appendix: A space-time diagram for a delay occurring at station Sauve

The Space-Time diagram below in Figure 11 depicts a incident causing a 10 minutes delay at 7:30AM at station Sauvé during a weekday scenario. Station Sauvé is the 5^{th} station on the Orange line, after a start in station Montmorency. It is located 4 stations before station Beaubien. Figure 11 shows that this incident impacts 4 vehicles. The networks takes around 30 minutes to recover from this primary delay. This can be observed as a "hole" in the parallel lines representing train trips.

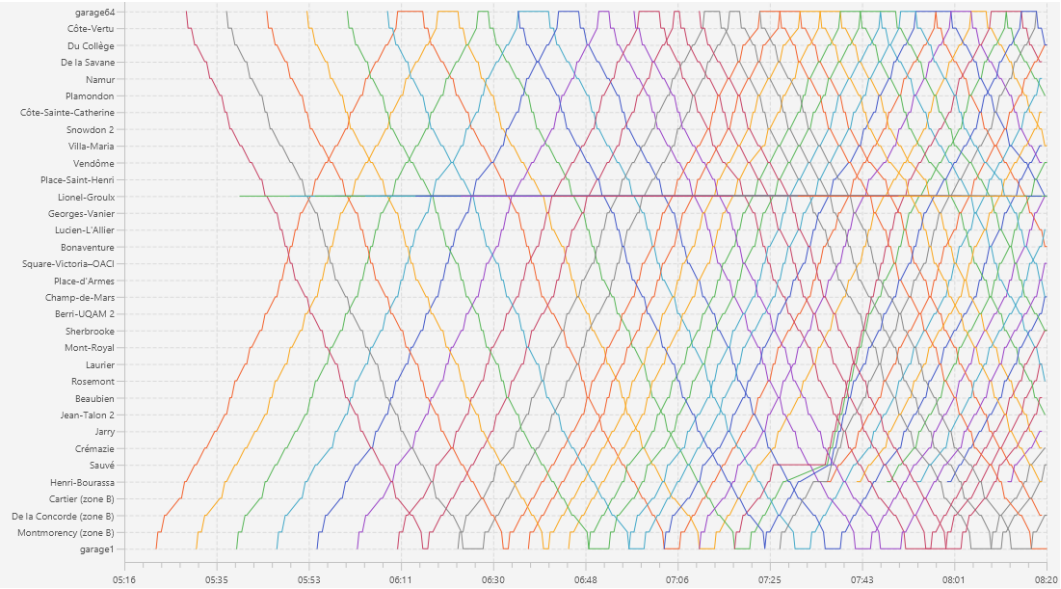


Figure 11 A delay at station Sauve impacts multiple vehicles.