Performance Evaluation of Metro Regulations Using Probabilistic Model-checking

Nathalie Bertrand, Benjamin Bordais, Loïc Hélouët, Thomas Mari, Julie Parreaux, Ocan Sankur

Performance Evaluation of Metro Regulations Using Probabilistic Model-checking

Nathalie Bertrand[1], Benjamin Bordais[2], Loïc Hélouët[1], Thomas Mari[2], Julie Parreaux[2], and Ocan Sankur[1]

[1] Univ. Rennes, Inria, CNRS, IRISA, Rennes, France
[2] ENS Rennes, France
{nathalie.bertrand,loic.helouet}@inria.fr, ocan.sankur@irisa.fr,
{Julie.Parreaux,Benjamin.Bordais,Thomas.Mari}@ens-rennes.fr

Abstract. Metros are subject to unexpected delays due to weather conditions, incidents, passenger misconduct, etc. To recover from delays and avoid their propagation to the whole network, metro operators use regulation algorithms that adapt speeds and departure dates of trains. Regulation algorithms are ad-hoc tools that are tuned to cope with characteristics of tracks, rolling stock, and passengers habits. However, there is no universal optimal regulation adapted in any environment. So, performance of a regulation technique must be evaluated before its integration in the network.
In this work, we use probabilistic model-checking to evaluate the performance of regulation algorithms in simple metro lines. We model the moves of trains and random delays with Markov decision processes, and regulation as a controller that forces a decision depending on its partial knowledge of the state of the system. We then use the probabilistic model checker PRISM to evaluate performance of regulation: We compute the probability to reach a stable situation from an unstable one in less than d time units, letting d vary in a large enough time interval. This approach is applied on a case study, the metro network of Glasgow.

1 Introduction

Urban Train Systems (UTS) play an increasing role in modern cities: they provide connections from work to residential areas, and have become a key element for economical and environmental concerns. Usually, UTS are operated by private or semi-public companies, whose role is to provide services with contractualized performance. A typical demand of local authorities is to guarantee departures with a high pace (for instance one train every two minutes) during peak hours to avoid networks congestion, and then ensure punctual/regular departures at lower pace for the rest of the day. Form a contractual point of view, performance is often specified in terms of Key Performance Indicators [12] (or KPIs for short). KPIs are measures for trains punctuality, passenger comfort, average trip times, etc. They are evaluated a posteriori from weekly or monthly logs recorded during operation of the network. Failing to meet fixed quality objectives may result in financial penalties.

A normal behavior of a train in a metro network is a succession of arrivals at stations and departures, usually scheduled at precise dates, or cadenced according to chosen departure rates at each station. Operators often rely on an a priori schedule called a *timetable*, that fulfills the KPI objectives if realized properly. Now, even during a normal day of operation, departures and arrivals of trains cannot match exactly such a precise schedule: trains are frequently delayed, due to passenger misbehavior, weather conditions, incidents on tracks, etc. Further, incidents are not independent: as trains have to maintain security distances, a primary delay on a train rapidly propagates to the following trains. To recover from delays and eventually meet quality objectives, UTS are equipped with *regulation algorithms*. Regulation algorithms consider the state of the network (train positions, delays w.r.t. a timetable, etc.), and compute advices to adapt trains dwell times or speeds. Several strategies such as trying to recover delays, sticking to existing timetables, or trying to equalize distances between trains.

It is well known that there is no *universal and optimal* regulation algorithm: efficiency of a particular regulation depends on the targeted KPI objective, on the topology of a line, on the number of trains running in the network, and even on passenger behavior. It is hence important to evaluate and compare performance of several algorithms to provide the most adapted solution in a given situation. Evaluation of a regulation scheme is hence often addressed as a performance evaluation question.

In this work, we use the probabilistic model checker PRISM [10] to evaluate the performance of regulation schemes. We model UTS as Markov decision processes (MDPs for short) [5], and regulation algorithms as controllers that make decisions in MDPs (i.e. implement a strategy). The approach is the following: a metro network is seen as a finite number of discrete locations. The behaviors of trains are modeled as processes which maintain discrete variables memorizing their positions, and have probabilistic guarded transitions, that randomly increment the position of a train. Regulation is also specified as a process whose decisions synchronize with the rest of the system to allow or prevent some transitions of trains. The overall behavior of the network is a form of product between processes, with safety constraints (trains shall not collide), which gives a Markov Decision Process. The accuracy of the model is chosen by appropriately discretizing time and space, and choosing probabilities to define models with sensible distributions of trips durations. We then study performance of regulation in a ring network equipped with a simple regulation algorithm with PRISM. More precisely, we initialize the system in a highly perturbed state (some significant delays have occurred), and compute the probability to get back to a *stable* situation (i.e. a situation in which the network has recovered from delay) in less that d time units, letting d vary in a significant time interval. Getting back to a normal situation in less than d time units is encoded with a PCTL formula [6].

The results obtained show that one can obtain the exact values of probabilities for fleets of 4 trains with a reasonable discretization factor. For larger fleets and larger discretization factors, one has to rely on statistical model checking

techniques. Another lesson learned is that without regulation, the probability to recover from a delay by chance is close to zero.

Model checking of railway systems has already been addressed in a Boolean setting, mainly with safety objectives (see for instance [7]). Verification of railway crossing, for instance, is a standard case study for model-checkers (see for instance [4]). Boolean verification mainly addresses safety issues (critical sections must not be violated), but usually cannot address quantitative properties, such as the time needed to recover from a primary delay. These quantitative notions are often addressed using simulation tools dedicated to performance evaluation. To evaluate a regulation policy, one can design a model of an UTS, and simulate a large sample of runs representing operation days with incidents, and derive statistics. Dedicated tools address railway systems modeling at a microscopic or macroscopic level. Macroscopic tools such as NEMO [8] use abstract models (a graph representing the network), and do not consider details such as adherence of trains to tracks, passenger flows. They are mainly used by infrastructure managers. On the other hand, tools working at microscopic level (e.g. Opentrack [11]) consider every detail of rail systems: characteristics of rolling stocks and tracks, weather conditions... Then, simulation steps compute the evolution of the network during a fixed time period (typically, one second). However, micro-steps simulation is time and space consuming. Microscopic and macroscopic approaches used for mainline trains can be adapted for metros, but metro networks have two characteristics that need to be considered: first they embed regulation algorithm, and second, decisions have to be made in a few seconds to avoid delays and their propagation. This makes a big difference, for instance, with macroscopic models of mainlines, where track occupancy schedules can be easily maintained in case of short delay impacting only a few trains. The SimMETRO tool [9] is specialized for simulation of metro systems. It includes regulation schemes, and was used to simulate performance of regulation in the Boston Metro network. The work in [2] uses a macroscopic simulation approach based on a Petri Net variants to model metro system equipped with regulation. The approach presented in this paper is a quantitative and macroscopic one, based on model checking. When the considered model is of reasonable size, the values computed are exact values, and hence provide strong performance guarantees. However, as subway systems are complex, their state space can rapidly exceed the limits of standard model-checkers that compute exact probabilities of properties from an explicit representation of the state space of the system. Even in this case, Statistical Model Checking can be used to obtain these probabilities, but only with a confidence interval.

This paper is organized as follows: Section 2 describes the Glasgow metro network, that will be used as a case study to illustrate our approach. Section 3 defines the formal material used later in the paper, namely Markov Decision Processes and PCTL properties, and shows how to use them to evaluate Performance in a regulated metro network. Section 4 gives experimental results, and comments them these results. Section 5 concludes this work and gives some perspectives.

2 A case study : a metro network in Glasgow

The usual behavior of metros is the following. A metro travels at a given speed between two stations, and then dwells in station for a predetermined duration. Commercial speed of metros usually lies between 30 and 40 km/h. When the dwell time has expired, the doors close, and the train leaves for the next station. This is where some incident may delay a train: doors may not close well, usually when passengers try to alight while doors are closing. This can result in delays of several seconds w.r.t. the expected departure date.

We consider as a case study the metro line of Glasgow [1]. This line is a bi-directional 10.5 kilometer ring with 15 stations, as depicted in Figure 1. Train can travel both clockwise and counterclockwise, using distinct tracks in separate tunnels; therefore, we only study a unidirectional line. The ring has no intersection with other lines. Completing a full round trip takes approximately 24 minutes. Several trains are used to provide optimal service: if 4 trains are in use, then a metro leaves a station every 6 minutes. The planned service is one metro every 4 minute at peak hours and every 6 to 8 minutes otherwise. The average dwell time in stations is around $30s$.

Fig. 1: A schema of the subway line of Glasgow

Ideally, providing a high quality service requires to maintain the network in a *stable* configuration, i.e. a situation where distances between consecutive trains are approximately equal (up to some small deviations appearing when trains stop or suffer delays of a few seconds). Such situations are ideal to enforce arrivals and departures at a regular pace. As in the Glasgow network the expected service is one train every 6 minutes, maintaining such balanced situations is a good way to fulfill the fixed quality objectives. However, networks do not remain in stable configurations without external help. As delays tend to accumulate, one may face the following situation: a train that is delayed arrives late at the next station, which increase the size of the crowd, and results in new door incidents, and penalizes the late train with an additional delay. Usually, as people tend to rush in trains as soon as they arrive in station, fewer passengers will enter the next train alighting at this station. This situation repeats all along the line, causing delays. As trains cannot overtake, accumulation of delays results in a bunching phenomenon, i.e. in a situation where a crowded train is followed closely by almost empty trains. To cope with this situation, metro operators use *regulation techniques*.

A regulation is an algorithm that gives advice to trains: these pieces of advice can be about changing the speed between two stations or the dwell time at a station, depending on the global state of the network. The range of values that can be returned by a regulation algorithm are of course bounded: there is a minimal and maximal running speed for trains, and similarly, a minimal dwell time allowing a sufficient number of passengers to leave trains or alight. In complex line topologies, a standard way to address regulation is to build precomputed timetables, and to try to stick as much as possible to these schedules. Of course, timetables are never realized exactly as specified, they are simply idealized schedules. A standard regulation called *hold-on* technique tries to return to this schedule by reducing dwell times and increasing trains speeds when a delay is measured. In the case of rings such as Glasgow network, the most relevant objective is to maintain a constant duration between arrivals at each station. Considering that characteristics of rolling stocks allow all trains to have the same speed ranges, this pace objective can be addressed as a distance objective, by requiring trains to maintain equal spacing among them. In practice, as distances between stations in metro networks are short, changing train speeds has little impact on delay recovery. We will hence consider regulation policies that change dwell times in stations in order to equilibrate distances among trains.

For the Glasgow network, we will build a stochastic model encompassing train behaviors, the possible perturbations, a regulation algorithm, then study the performance of this algorithm with a probabilistic model checker. To achieve this objective, we will compute the probability to reach a stable situation from an unstable one in less than d minute, letting d vary in interval $[0; 250]$.

3 Models

Unpredictable external events can affect the durations of dwelling and of trips from one station to the next one: it is natural to model them using probabilities. These events delay trains, and as explained in Section 2, regulation policies are then used to recover from primary delays and avoid their propagation to the whole network. Given the current state of the system, an appropriate regulation decision is chosen from a set of possible options and given as instructions to the trains. To represent a metro system, we thus need a model that combines probabilities (for the unpredictable events) and non-determinism (for the choice of regulation decisions), hence we choose Markov decision processes (MDP). In this section, we first define the mathematical model of MDP. Then we provide a model of a generic ring metro line with several trains as an MDP with parameters. We explain how to tune the values of the parameters for the Glasgow case study, to reflect the number of trains in the network and the average trip durations. Finally, we define properties on this instantiated MDP model, that are of particular interest to evaluate the performances of regulation policies.

3.1 Markov Decision Processes

Definition 1. *A* Markov decision process *(MDP) is a tuple* $\mathcal{M} = (S, s_0, \mathsf{Act}, \delta, \mathsf{AP}, \ell)$, *where* S *is a set of states,* s_0 *is the initial state,* Act *is a finite set of* actions, $\delta : S \times \mathsf{Act} \times S \to [0,1]$ *is the probabilistic transition function such that for every* $s \in S$ *and* $\alpha \in \mathsf{Act}$, $\sum_{s' \in S} \delta(s, \alpha, s') \in \{0, 1\}$, AP *is a set of atomic propositions, and* $\ell : S \to 2^{AP}$ *is the labeling function.*

An action $\alpha \in \mathsf{Act}$ is *enabled in state* s if $\sum_{s' \in S} \delta(s, \alpha, s') = 1$. The semantics of a Markov decision process $\mathcal{M} = (S, s_0, \mathsf{Act}, \delta, \mathsf{AP}, \ell)$ operates in discrete time as follows: from some state $s \in S$, when an enabled action α is chosen, the probability to be in s' at the next time instant is $\delta(s, \alpha, s')$. Actions in $\mathcal{M}$ thus model the possible choices one has to guide the system. A *path* in an MDP is a finite or infinite alternating sequence of the form $s_0.\alpha_0.s_1\alpha_1.s_2 \ldots$ such that, for every $i \geq 0$, $\delta(s_i, \alpha_i, s_{i+1}) > 0$, that is, the probability to reach s_{i+1} from s_i when choosing action α_i is positive. We denote by $\mathcal{P}ath^{\mathcal{M}}$ (resp. $\mathcal{P}ath^{\mathcal{M}}_{\mathsf{fin}}$) the set of all paths (resp. all finite paths) of $\mathcal{M}$. Figure 2 is an example MDP, with $S = \{s_a, s_1, s_2, s_{\mathsf{goal}}\}$, $s_0 = s_a$, $\mathsf{Act} = \{\alpha, \beta\}$, and $AP = \{a, goal\}, \ell$) with $\ell(s_a) = \{a\}$, $\ell(s_1) = \ell(s_2) = \emptyset$ and $\ell(s_{\mathsf{goal}}) = \{goal\}$. The transition relation of this MDP is given by: $\delta(s_a, \alpha, s_1) = 1$, $\delta(s_1, \alpha, s_2) = 0.3$, $\delta(s_1, \alpha, s_a) = 0.7$, $\delta(s_2, \alpha, s_a) = 0.5$, $\delta(s_2, \alpha, s_2) = 0.5$, $\delta(s_1, \beta, s_1) = 0.5$, $\delta(s_1, \beta, s_2) = 0.1$, $\delta(s_1, \beta, s_{\mathsf{goal}}) = 0.4$, $\delta(s_2, \beta, s_a) = 0.9$, $\delta(s_2, \beta, s_{\mathsf{goal}}) = 0.1$, $\delta(s_{\mathsf{goal}}, \beta, s_{\mathsf{goal}}) = 1$ and $\delta(s, \gamma, s') = 0$ for all other states s, s' and action $\gamma \in \mathsf{Act}$.

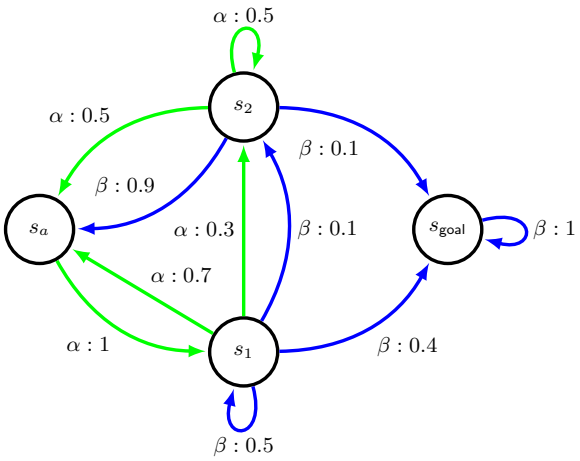

Fig. 2: An example MDP

Given an MDP, a policy resolves the non-determinism by choosing an enabled action after each history of states and actions seen so far, that is for every finite path of $\mathcal{M}$. Formally,

Definition 2. *A* policy *for the MDP* $\mathcal{M} = (S, s_0, \mathsf{Act}, \delta, \mathsf{AP}, \ell)$ *is a function* $\sigma : \mathcal{P}ath^{\mathcal{M}}_{fin} \to \mathsf{Act}$.

Given an MDP $\mathcal{M}$, each policy σ defines a probability measure on infinite paths of $\mathcal{M}$ originating from initial state s_0, that we write $\mathbb{P}^{\sigma}_{\mathcal{M}}$. Reasonable sets of paths A are measurable: one can write $\mathbb{P}^{\sigma}_{\mathcal{M}}(A)$ for the probability that a sampled path of $\mathcal{M}$ starting from s_0 and following σ belongs to A.

3.2 An MDP model for the Glasgow network

As argued earlier, MDPs are well suited to model random events occurring in a metro, as well as non-determinism for the choice of the regulation decision. However, MDPs are discrete models, whereas metro networks are continuous systems, both in space and time. We thus propose to approximate their behavior with a fine enough discretization of space and time. In this discrete model, each state of the MDP will indicate the trains positions, and if some trains are stopped their remaining dwell time. Each discrete step in the MDP model corresponds to changes (in the positions and remaining dwell times) occurring during a fixed delay. To obtain a relevant model, we associate a position to each station, but we also consider intermediate positions between two stations, i.e. we discretize the distance between two stations by adding k intermediate positions between two consecutive stations of the network. The number of intermediate positions is one of the parameters of the model that need to be fixed a priori. We will call *location* either a station or one of these intermediate positions.

As for regulation policies, the interesting elements to consider are arrival and departure dates, and train positions. The behavior of each train T_i traveling in the network can then be seen as an individual MDP $\mathcal{M}_i$, in which states are locations of the network, and actions are regulation decisions. In this MDP, train T_i applies an action $\alpha \in \mathsf{Act}_{\mathcal{M}_i}$ (that represents a regulation decision), and moves to the next location in the network with some probability p_α, or stays at its current position with probability $1 - p_\alpha$.

Figure 3 represents the MDP model of the individual behavior of a train between two stations S_0 and S_1, with two intermediate points. At each location (that correspond to states of the MDP), there are three possible actions, representing the three different target speeds. Action α, labeling green transitions represents the behavior of the trains when running at their standard speed, action β (blue transitions) symbolizes the reduced speed mode, and action γ (red transitions) the stopped mode.

In the following, we will restrict to train models with the three possible running modes α, β, γ illustrated in Figure 3: one mode in which the train is running at its standard commercial speed, one mode in which it travels at reduced speed, and a last mode where the train is stopped. Intuitively, the intermediary mode will be used if some other train is too close ahead, and the stopped mode will be used to avoid collisions. However, the model easily extends to an arbitrary number of target speeds, but at the cost of an increased the number of actions and transitions in the model.

The overall behavior of the metro network is also an MDP, obtained as a product $\bigotimes \mathcal{M}_i \otimes \mathcal{M}_R$ of all individual MDP models for each train and of regulation, with safety constraints. An important constraint is that the product

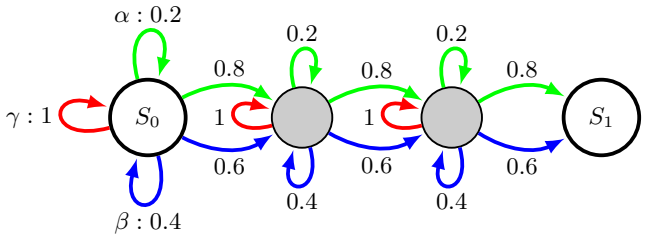

Fig. 3: Part of the MDP model for a train, representing the journey between stations S_0 and S_1 and with two intermediate locations.

forbids two trains (or more) to be at the same location at the same moment. This safety requirement can be easily be implemented as a constraint on the enabled transitions in each state of the product MDP. For example, if a train T_i is in location x and at the same instant another train T_j is in the next location, i.e. $x+1$, then all transitions that send T_i in location $x+1$ are forbidden. This restriction guarantees that trains never collide.

In our MDP model for each train, probabilities attached to transitions allow one to reflect the time a train will take to travel from a station S_i to the next station S_{i+1}. For each action α, β, γ, at each step, a train can stay at its current location with probabilities $p_\alpha, p_\beta, p_\gamma$, respectively, or go to the next location with probabilities $1-p_\alpha, 1-p_\beta, 1-p_\gamma$. The parameters $p_\alpha, p_\beta, p_\gamma$ must be adjusted to reflect the speed instructions (normal, reduced, 0). To simplify our model, we will assume that the distance between two consecutive stations is the same in the whole network. The distance Δ_d between two consecutive locations is hence also uniform, and depends on the number k of intermediate positions. Each step in the execution of a MDP symbolizes elapsing of a fixed duration Δ_t. We will explain how to choose Δ_t once the probability parameters are fixed.

First of all, let us consider the speed induced by the choice of a particular action by regulation. In our model, we want the speed induced by action α to be the usual speed of trains, and the speed corresponding to action β to be a lower speed occurring when trains slow down for safety reasons (if the next train is too close), finally γ corresponds to stopping the train. Start for example with action α. The probability of going to the next location in one step under action α is set to $p_\alpha = 0.8$. In every location of the network, the probability to stay in there under action α during m steps is thus $(1-p_\alpha)^m$, and the probability to move to the next location exactly at the m^{th} step is $(1-p_\alpha)^{m-1} \times p_\alpha$. More generally, the number of steps needed to move to the next location follows a geometric law, whose expected value is $\mathbb{E}_\alpha = \dfrac{1}{p_\alpha}$. With the parameters of Figure 3, $\mathbb{E}_\alpha = \dfrac{1}{0.8} = 1.25$, so that in average, it takes 1.25 steps to go from a location to the next one, assuming the regulation decision is always α.

The standard speed v_α of a train traveling at speed given by regulation instruction α is obtained by dividing the distance Δ_d between two successive

locations by the average time needed to go from a location to the next one under action α. The first parameter Δ_d can be easily computed as soon as k, the number of intermediary locations, is fixed. In the Glasgow line, the total length of the network is 10.5 km, and there are 15 stations. So the distance between two consecutive stations is $\Delta_d = 10500/15 \cdot k$. Similarly, for a geometric law of parameter p_α, the expected number of discrete steps to move from one station to the next one is $\mathbb{E}_\alpha \cdot k$. We know that the total duration of a round trip in the Glasgow line is 1440 seconds, including a dwell time of 30 seconds at each station. Hence the total running time in seconds is $t_{tot} = 1440 - (30 \times 15) = 990$. The time needed to move from a station to another, in seconds, is thus $t_{stat} = 990/15 = 66$. Fixing parameter k, we should have $\mathbb{E}_\alpha.k.\Delta_t = 66$. If we choose to have $k = 5$ locations between two stations, we obtain $\Delta_d = 140$ meters and $\Delta_t = 10.56$ seconds. We finally obtain the commercial speed $v_\alpha = \dfrac{\Delta_d}{E_\alpha \times \Delta_t} = p_\alpha.\dfrac{\Delta_d}{\Delta_t} \approx 10.6$ m·s^{-1} ≈ 38.2 km·h^{-1}.

In a similar way, we can compute the average speed of trains when their behavior is dictated by regulation instruction β. The expected number of steps needed to go from a location to the next one is now $\mathbb{E}_\beta = 1/p_\beta$. We hence have $v_\beta = p_\beta \cdot \dfrac{\Delta d}{\Delta t}$. If $p_\beta = 0.6$ as in the model of Figure 3, we obtain a reduced speed $v_\beta \approx 7.95$ m·s^{-1} ≈ 28.6 km·h^{-1}. Finally, for action γ that represents the instruction not to move, the speed is obviously $v_\gamma = 0$. All in all, using three types of actions, associated each with a probability to move one location forward in the next step, we were able to model three regulation instructions for the speed of trains.

Notice that as soon as the discretization constant k is known, the value of Δ_d follows from the length of the line. Similarly, considering that the dwell time in stations, the total duration of a round trip are known, it suffices to choose the value of p_α to obtain the value of Δ_t. As we had no data on round trip durations in Glasgow, we have chosen a value for p_α such that the distribution of trip durations in our model matches statistics recorded for a track portion in another line with similar characteristics. We detail in Appendix A how this fitting was performed, and how value $p_\alpha = 0.8$ was chosen to model duration of trips from a station to the next one at standard commercial speed.

3.3 Integrating a regulation policy in the model

Regulation policies decide which instruction to give to the trains, given their positions (and possibly other useful data). In this work, we assume that a *signaling system* is used to determine the safe speed of running trains at every step. We also consider that trains run at their usual commercial speed whenever this speed is allowed. We thus consider regulation policies that only choose the dwell times at stations, that is, determining whether a given train should leave the station early or continue waiting.

The signaling system works as follows. Between two stations, trains travel at their commercial speed (following α-transitions) if there is no train too close

ahead, at reduced speed (β-transitions) if the train ahead is close, and stop moving to prevent collision or if they have to stay longer in a station.

A regulation policy aims at avoiding delays or recovering from them. In a train networks, delay can be interpreted as a difference between an arrival/departure date and the expected date of realization of this event in a pre-computed timetable. However, in metros like Glasgow, emphasis is put on regularity of departures, not on precise schedules. To formalize delays, we compute, for each state q of the MDP representing the whole network, a function $bal_q : 1..\mathsf{nb_{trains}} \to \mathbb{R}$ that measures whether time intervals between trains are similar or not in state q. In a state q, let us call $pos(q, i)$ a number between 0 and $k.15 - 1$ denoting the position of train i in sate q. Then, for a particular train i, we define: $bal_q(i) = \dfrac{d_q(i, i+1)}{d_q(i-1, i) + d_q(i, i+1)}$ where $d_q(i, j)$ measures the difference between the position of trains T_i and T_j in sate q. It has to be noted that bal_q cannot be equal to 0 nor 1 since two trains cannot be in the same location.

An ideal situation is when $\alpha_q(i) = 0.5$ for every train T_i in the network, which means that each train is positioned precisely in the middle of the space delimited by its predecessor and its successor. Maintaining this equilibrium is a way to guarantee regularity of service. However, it cannot be achieved unless the movements of trains are quasi synchronous, which is too much requiring, as distances between trains always vary due to dwell in stations. We will say that the position of train T_i is *balanced* in state q if $\alpha_q(i) \in [0.4, 0.6]$, and unbalanced otherwise. We say that the current state of the network is balanced if all train positions are balanced. As this definition only depends on the current state, one can attach an atomic proposition $\mathsf{balanced}$ to every balanced state of the global MDP. Similarly, we can associate a tag $\mathsf{collision}$ to a state if $pos(q, i) = pos(q, j)$ for some $i \neq j \in 1..\mathsf{nb_{trains}}$, that is to represent that a collision occurs.

We can now define our regulation policy as a linear function $\mathsf{Dwell} : [0, 1] \to [20, 40]$ applies from the value $bal_q(i)$ for every train T_i. In each state of the system, when a train T_i arrives in station, the regulation algorithm imposes a dwell time of $\mathsf{Dwell}(bal_q(i))$ to train T_i. If $bal_q(i)$ is close to 1, the train has to leave the station early, and if it is close to 0, it has to dwell in station for a longer duration than the nominal dwell time. Due to the discretization of time, the dwell duration in station depends on the interval to which $bal_q(i)$ belongs and not the exact value of $bal_q(i)$. For instance, with $\Delta_t \simeq 5\ s$, the dwell time was set as follows:

$$\mathsf{Dwell}(v) = \begin{cases} 4 \times \Delta_t & \text{if } v \in [0.875, 1] \\ 5 \times \Delta_t & \text{if } v \in [0.625, 875] \\ 6 \times \Delta_t & \text{if } v \in [0.375, 0.625] \\ 7 \times \Delta_t & \text{if } v \in [0.125, 375] \\ 8 \times \Delta_t & \text{if } v \in [0, 0.125] \end{cases} \tag{1}$$

Similarly, if $\Delta_t = 10s$, we have set $\mathsf{Dwell}(v) = 2 \times \Delta_t$ if $v \in [0.667, 1]$, $\mathsf{Dwell}(v) = 3 \times \Delta_t$ if $v \in [0.334, 0.666]$ and $\mathsf{Dwell}(v) = 4 \times \Delta_t$ if $v \in [0, 0.333]$.

3.4 Logical properties for MDP

The overall objective of our experiment is to assess the performances of regulation policies from a metro model. To do so, we use relevant quantitative properties. For example, we evaluate the probability of a regulation policy to recover from a delay in less than 30 minutes. This can be done by writing "from initial state, the MDP reaches a balanced state within 30 minutes" as a property φ in logics, and then computing the probability of this event in the MDP given the policy σ corresponding to the regulation at hand. One can also evaluate the maximum probability for this event when σ ranges over all possible regulation policies. Such an optimal policy σ can be computed: we have $\mathbb{P}^{\sigma}_{\mathcal{M}}(\varphi) = \max_{\tau} \mathbb{P}^{\tau}_{\mathcal{M}}(\varphi)$ where τ ranges over a finite number of policies (see appendix B for details). Typically, we will denote this value by $\mathbb{P}^{\max}_{\mathcal{M}}(\varphi)$.

The mentioned property φ is usually written using a formula $F^{\leq 30}\mathsf{balanced}$ in linear temporal logics [13]. Such a property φ is a *bounded reachability property*: the aim is to reach a given set of states (here the balanced ones) within a given number of steps (or seconds). We are also interested in (unbounded) *reachability properties*, such as $F\,\mathsf{collision}$ that expresses that eventually a collision occurs, yet with no time bound. For our metro system, we aim at proving that under all regulation policies the probability of $F\,\mathsf{collision}$ is 0, showing that collisions are impossible. This constraint can be expressed as $\max_{\tau} \mathbb{P}^{\tau}_{\mathcal{M}}(F\,\mathsf{collision}) = 0$, or in a more compact way $\mathbb{P}^{\max}_{\mathcal{M}}(F\,\mathsf{collision}) = 0$.

To compute such optimal probabilities, we use the probabilistic model checker PRISM [10]. We describe our metro model as processes in the PRISM language: each train is a process, whose state encodes the position of the train in the network. Regulation is also a process. The underlying semantics of these processes is the MDP depicted in section 3.2. A first sanity check for our PRISM model is thus to verify that the safety requirement is met, i.e. that the resulting MDP $\mathcal{M}$ for the network behavior (obtained as the product of processes for trains) satisfies the formula $\mathbb{P}^{\max}_{\mathcal{M}}(F\,\mathsf{collision}) = 0$. Then to evaluate the efficiency of a regulation algorithm, we will compute values such a $\mathbb{P}^{\max}_{\mathcal{M}}(F^{\leq q}\,\mathsf{balanced})$ for q a given number of steps. This value is the maximal probability, among all regulation policies, i.e. considering choices of actions that are not decided by regulation, that the system recovers a balanced situation within q steps (or equivalently within $q \times \Delta_t$ seconds). We will also measure $\mathbb{P}^{\sigma}_{\mathcal{M}}(F^{\leq q}\,\mathsf{balanced})$ to evaluate efficiency of a policy σ. Letting the system start from an unbalanced state, we can compute this probability for different values of q and obtain performance measures for a regulation algorithm.

4 Experimental Results

We performed several experiments with PRISM on the MDP models we constructed for the Glasgow metro. The objective was to evaluate the value of quantitative formulas given in Section 3.4 to study performances of the simple regulation algorithm (balancing of trains positions on the ring) defined in section 3.3. Usually, PRISM explicitly builds the MDP resulting from the processes

description, and then computes the values for the properties to check by value iteration (see [5] for a description of the algorithms). As an alternative to the explicit MDP construction, in case the model is too large (in terms of state space, and transition table) to be stored, PRISM can perform *statistical model checking* (SMC), i.e., generate on the fly a set of sample runs, to approximate, with a given confidence, the values one aims at computing.

For our case-study, the size of the MDP depends on the discretization factor k, and on the number of trains running in the system. The tests we performed were conducted for two discretization values ($k = 5$ and $k = 10$) and two possible number of trains ($\mathsf{nb_{trains}} = 4$ and $\mathsf{nb_{trains}} = 6$). As described in section 3.2, all processes representing trains were designed with three distinct possible speeds α, β, γ, corresponding respectively to standard commercial speed, reduced speed, and train stopped. These speeds were modeled using intermediary locations as in Figure 3, with probabilities $p_\alpha = 0.8, p_\beta = 0.6, p_\gamma = 0$.

As initial state of our MDP, we chose a very unbalanced configuration, in which trains occupy consecutive stations, with no free location between them. This configuration is certainly the worst for the network. We then computed three probabilities, for increasing values of q, the bound on the number of steps to recover a balanced configuration.

We first consider the policy σ_{bal} defined in Section 3.3, which chooses dwell times for each train in order to move the train to the middle of the previous and the next train. This probability is thus denoted $\mathbb{P}^{\sigma_{\mathsf{bal}}}_{\mathcal{M}}(F^{\leq q}\ \mathsf{balanced})$.

Second, , we computed the maximum probability when the policy ranges over all possible ones. This is written as $\mathbb{P}^{\max}_{\mathcal{M}}(F^{\leq q}\ \mathsf{balanced})$. Note that when computing this probability, PRISM also determines *how to optimize this value*, and returns a policy.

Third, we defined a fixed regulation policy σ_{fix} which always picks the dwell time of 30 seconds regardless of the current state. This probability is thus denoted $\mathbb{P}^{\sigma_{\mathsf{fix}}}_{\mathcal{M}}(F^{\leq q}\ \mathsf{balanced})$. This choice corresponds to an absence of regulation policy. We expect that σ_{fix} should perform worse than σ_{bal} since the latter makes clever choices in order to bring the system to a balanced situation.

Obviously, the size of the models increases with the discretization factor k and with the number of trains. With the smallest values $k = 5$ and $\mathsf{nb_{trains}} = 4$, PRISM was not able to build the complete state space of the model, and hence could not compute that optimal policy. A standard technique to overcome this limit is to use abstraction. Abstraction allows to consider some states as equivalent, and then perform model checking on a quotient (w.r.t. equivalence classes) of the original MDP (which is then smaller). We have used a sound abstraction of states up to a rotation of positions in the network. Indeed, in the Glasgow ring, the dwell time decided for a train only depends on the distance to the predecessor and successor, and not on the visited station. This abstraction allowed to reduce the state space of the original MDP, while preserving values of the formulas (as behaviors of trains and the balanced property of states are equivalent up to rotation). We do not detail here the formal definition of abstraction *up to rotation*, and refer interested readers to Appendix C for details. The table 1

below shows the effect of abstraction of the size of the MDP model computed by PRISM, for a discretization factor $k = 5$, with and without abstraction. One can see that abstraction up to rotation reduces the number of states by a factor 1000 for three trains, and allows PRISM to compute explicitly and MDP for 4 trains, which was not possible without abstraction. However, even with abstraction, taking as discretization factor $k = 10$ or a fleet of 6 trains exceeded the size of models tractable for which PRISM can return an exact vaule for a simple property. With these parameters, one necessarily have to rely on SMC.

Model Number of trains	Before abstraction	After abstraction
Three trains	2.1×10^8 states 3.4×10^9 transitions	3.5×10^5 states 8.3×10^5 transitions
Four trains	Not built in PRISM	2.0×10^7 states 5.7×10^7 transitions

Table 1: Size of the MDP models in terms of number of states and transitions ($k = 5$)

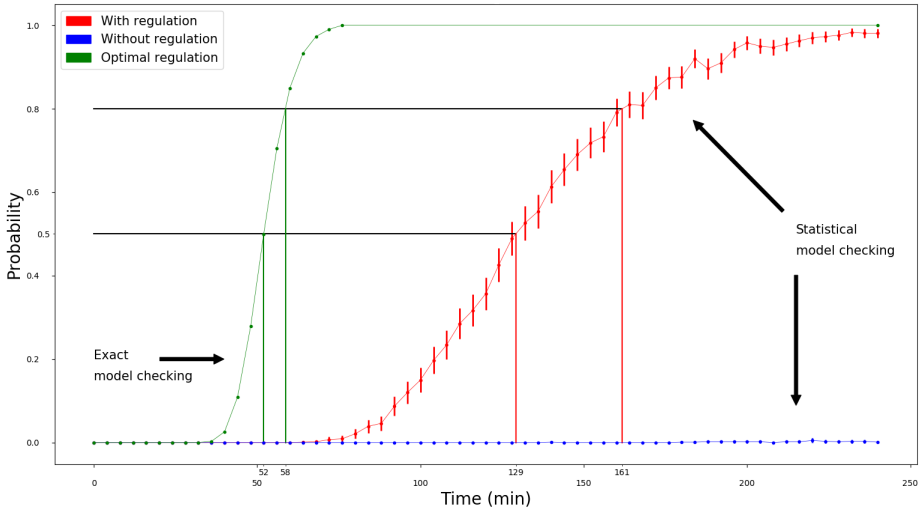

Fig. 4: Probability to recover from a delay with $\mathsf{nb}_{\mathsf{trains}} = 4$ and $k = 5$

We can now show the results obtained from our experiment. Figure 4 shows the probability to return to a balanced state in x minutes when starting from a very unbalanced situation. Absiscae represents time elapsed, and ordinates the probability. The green curve is the exact value of the probability computed from the MDP (an MDP - quotiented by the rotation abstraction- was explicitly built by PRISM) when an optimal policy is used to regulate trains. The red curve is the result obtained with our simple regulation policy σ_{bal}, and the blue curve the results obtained when constant time regulation σ_{fix} is used. For the red and blue series of measures, statistical model checking had to be used. Indeed, introducing a particular regulation scheme may require the use of additional information in states and increases the size of the underlying MDP. One can notice on the figure

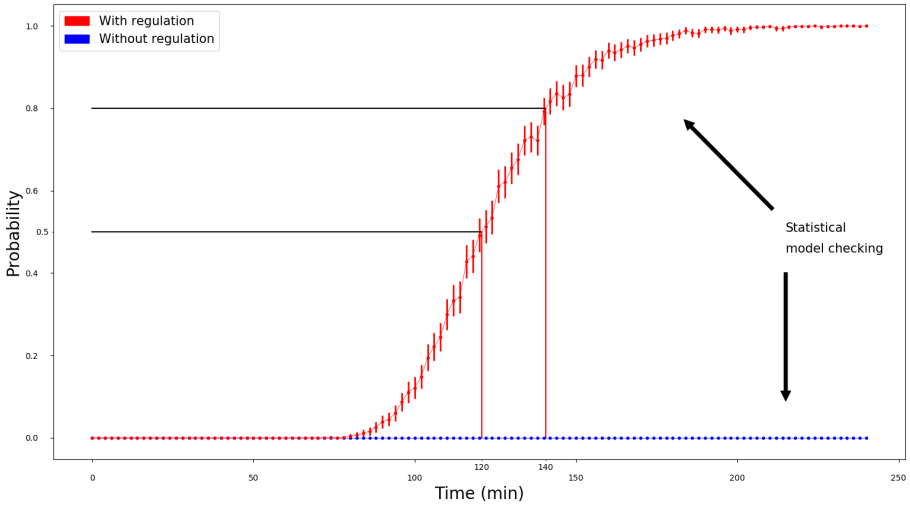

Fig. 5: Probability to recover from a delay with $\mathsf{nb}_{\mathsf{trains}} = 4$ and $k = 10$

that as the results are obtained with statistical model checking, the probabilities in the red and blue curse are not given as a single point but as an interval. On this figure, one can notice that the best possible choices when doing regulation cannot do better than returning to a balanced state with probability 0.5 in 52 minutes, and with probability 0.8 in 58 minutes. The regulation policy σ_{bal} needs respectively 129 minutes and 161 minutes to reach probabilities 0.5 and 0.8. One can however notice that this regulation improves the performance of the metro network, as the regulation σ_{fix} that sticks to standard dwell times of 30s in stations has a probability to return to a balanced state that stays close to 0.

Let us now compare these results with the curves obtained for $k = 10$ and $\mathsf{nb}_{\mathsf{trains}} = 4$ (Figure 5) and for $k = 10$ and $\mathsf{nb}_{\mathsf{trains}} = 6$ (Figure 6). As explained before, these values do not allow PRISM to compute and store the whole state space of the MDP and hence to obtain the green curve representing results achieved with an optimal policy. However, measures for regulation σ_{bal} or σ_{fix} can still be obtained with statistical model checking. We can now discuss the effect of discretization by comparing the red curve in Figure 4 and Figure 5. One can notice that the global shape of the curve is the same, but that with a discretization factor of 10, it takes 120 minutes (instead of 129) to return to a balanced state with probability 0.5. Similarly, it takes a slightly smaller time (140 minutes instead of 161) to get back to a balanced state with probability 0.8. This can be explained by several facts. First, when choosing a rough discretization, one gives regulation the ability to take fewer choices, and starting from less precise information than with a finer discretization. A second aspect is that trains reduce their speed when they approach their predecessor (i.e. the number of locations between the two trains is low), and stop when moving to the next location would cause a collision. So, with a coarse discretization, trains will slow down and stop more frequently, which will delay the date of recovery.

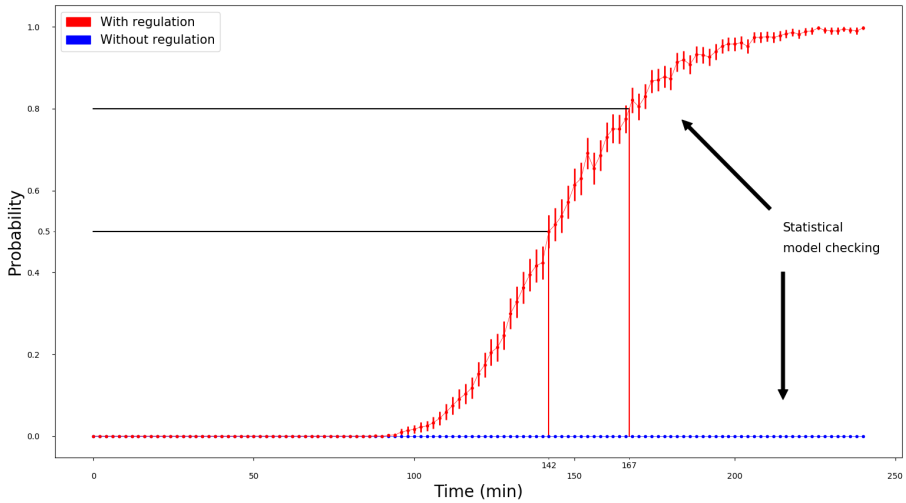

Fig. 6: Probability to recover from a delay with $\mathsf{nb}_{\mathsf{trains}} = 6$ and $k = 10$

Similarly, we can compare the curves of Figures 5 and 6, respectively obtained with values $(k = 10, \mathsf{nb}_{\mathsf{trains}} = 4)$ and $(k = 10, \mathsf{nb}_{\mathsf{trains}} = 6)$. It take slightly longer to return to a balanced situation with 6 trains than with 4: the network needs 142 minutes instead of 120 to get back to a balanced state with probability 0.5, and 167 minutes instead of 140 to return to a balanced state with probability 0.8. One can easily understand why: each train introduces randomness in the system, and increasing the number of trains increases the probability to move to an unbalanced state.

Let us now address the time needed by PRISM to compute the curves. We recall that the green curve in Figure 4 is the exact probability to reach a stable configuration in x steps achievable with an optimal regulation scheme, for $k = 5$ and $nb_{trains} = 4$. Each point in this curve took up to 5 hours to compute on an average laptop. For Statistical Model Checking, with a confidence level of 99%, the time needed to compute each interval in the red curves of Figures 4, 5 and 6 (i.e., the probability to return to an equilibrate situation in x minutes with our regulation algorithm) is less than 2 minutes per interval.

5 Conclusions

We have proposed a quantitative evaluation scheme for metro networks with simple ring topologies. The steps are the following : first choose a discretization factor, and the appropriate probabilities needed to fit the known speeds of the trains in the network. Then, generate a MDP model as a product of train movements and regulation. Last, given regulation policy σ, for a large enough set of steps, compute $\mathbb{P}^{\mathrm{pmax}}_{\mathcal{M}}(F^{\leq q}\, balanced)$ to draw an estimate of the cumulative distribution function for the optimal time needed to recover from an unbalanced situation, and $\mathbb{P}^{\sigma}_{\mathcal{M}}(F^{\leq q}\, balanced)$ for the time needed with regulation σ.

This technique provides interesting results. First of all, for a coarse discretization and a small number of trains, it allows to compute the exact value of probabilities under any finite memory policy. The size of the MDP rapidly exceeds the model-checker's limits, but statistical model checking still works for finer discretization and larger number of trains. The parameters used to model the network of Glasgow are bounded in terms of discretization ($k \leq 10$) and in terms of trains. Regarding the number of trains, as far as the Glasgow network is concerned, this value is sufficient to model the activity in the network at peak hours. For discretization, it has to be noted that the state space of the MDP is greater than $(K \times k)^{\mathsf{nb}_{\mathsf{trains}}}$ where K is the number of stations and k the discretization factor. Overall, appropriately chosen parameters allow to obtain a fair estimation of the distribution of time needed to return to normal situations. The durations obtained may seem rather high (on the average 150 minutes), but the initial state of simulation is the worst possible situation for the network. A natural extension of this work is to consider the time needed to return to a balanced state under less severe perturbations (e.g. with a single train delayed)

One advantage when working with explicitly built MDP models in PRISM, is that computing the optimal value for a formula also gives a finite memory strategy to reach this optimal. However, these strategies are state-based, and cannot be interpreted immediately as a regulation algorithm. An interesting issue is hence to understand better the output of quantitative model checkers to be able to synthesize efficient strategies in terms of user-understandable rules. Another possible extension for this work is to consider more complex topologies, and more complex regulation techniques, for instance network topologies with forks and joins where trains complete distinct trips, networks with parking locations allowing to remove a train from the network or insert it at the most appropriate moment to improve performance of the network...

This work also helped to discover strengths and weaknesses of generic model-checking techniques. Undoubtedly, generic model checking tools such as PRISM have flexible enough input languages to model complex systems such as regulated metro networks. They also build on solid theory to obtains values for probabilistic properties. However, they also have some obvious weaknesses. First of all, as already mentioned in the paper, the state space of a Metro network, even for a simple topology such as the Glasgow ring is huge. This has an impact on the applicability of exact techniques relying on value iteration techniques, that need to build a complete state space to obtain results. With respect to this drawback, abstraction techniques can help reducing the state space. In this paper, the reduction used a a symmetry, and is an exact abstraction : the values of a property checked on the abstract model is exactly the values of the property on the original model. Exact reductions via symmetries work mainly for ring-like lines, but should be more difficult to obtain, and less efficient for other topologies. Then, one can rely on other techniques that group sets of states into equivalence classes, but at the cost of an approximation in the obtained results.

Discretization is a factor that increases a lot the state space considered by model-checkers. The choice of the discretization levels chosen for the experiment

were mainly guided by the will to model accurately the distributions of transit times between stations. Hence, shapes of network topologies do not impact too much the discretization level of a model. However, the discretization level in an important parameter of our models: when discretization increases, the distribution of transit times that can be obtained approach a Gaussian distribution. Further, if discretization is too coarse, the space between two intermediate locations may cover more than one block. As a consequence, information about block occupation ahead a train is pessimistic, and trains may have to brake more often in the simulated model that in the real network, which decreases the performances of the simulated system. Hence, a rather high level of discretization is preferable to model realistic train movements (at least intermediate locations should not cover several blocks). It should be noted, however, that improving discretization by a factor c results is a blowup of $c^{nb_{trains}}$.

We hence face several generic difficulties that are inherent to model-checking tools (and not only to PRISM) : a very precise model leads to a state space explosion, which disallows computation of exact values for probabilities. To overcome this problem, one has to lowed the discretization level of the model, and obtain pessimistic results in terms of performance of regulation. The other possibility is to find good abstractions, but as long as the abstraction is not exact, this lead to a loss of precision in the results. The Other possibility is to use Statistical Model checking. As shown in this paper, SMC allows to deal with models of larger sizes (in our case 6 trains and a discretization factor of 10, i.e., a system with more than 10^{13} states), but computes a confidence interval. It shall be noted that the precision of the confidence interval can be set in most SMC tool by choosing a confidence level. Of course, the computation time needed to obtain small confidence intervals increase with the required confidence, but the loss of precision when using SMC for performance evaluation from a faithful model can be limited. All these consideration advocate for the use of SMC for evaluation of regulation in Metro networks. Another possible approach is to use SMC with continuous representations of train trajectories instead of discrete positions. This is the approach followed in [2]. The price to pay here is that evolution of the system over time is not discretized and each move performed every each x time units. Instead, one has to compute the next instant at which an event (arrival, departure, braking...) occurs, which can be costly when the size of the model grows.

References

1. Glasgow subway webpage. http://www.spt.co.uk/subway/. 2018.
2. B. Adeline, P. Dersin, E. Fabre, L. Hélouët, and K. Kecir. An efficient evaluation scheme for KPIs in regulated urban train systems. In *Reliability, Safety, and Security of Railway Systems (RSSRAIL'17)*, pages 195–211, 2017.
3. C. Baier and J.-P. Katoen. *Principles of model checking*. MIT Press, 2008.
4. G. Behrmann, A. David, and K.G. Larsen. A tutorial on UPPAAL 4.0, 2006.
5. V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume 6659 of *LNCS*, pages 53–113, 2011.
6. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
7. A.E. Haxthausen and J. Peleska. Formal development and verification of a distributed railway control system. *IEEE Trans. Software Eng.*, 26(8):687–701, 2000.
8. M. Kettner, B. Sewcyk, and C. Eickmann. Integrating microscopic and macroscopic models for railway network evaluation. In *Association for European Transport*, 2003.
9. H.N. Koustopoulos and Z. Wang. Simulation of urban rail operations: model and calibration methodology. In *Transport Simulation, Beyond Traditional Approaches*, pages 153–169, 2009.
10. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591, 2011.
11. A. Nash and D. Huerlimann. Railroad simulation using opentrack. In *Computers in Railways IX*, pages 45–54, 2004.
12. UITP (International Association of Public Transports). Metro service performance indicators, a UITP information sheet. 2011.
13. A. Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science FOCS'77*, pages 46–57. IEEE, 1977.

A Computing a probability for accurate trip durations

The difficult point, to obtain a realistic model for trains movements is to choose a value for p_a. When starting our experiment, no data was available on the distribution of trip durations between two stations in Glasgow. However, distribution of trip durations have typical shapes: they take the form of Weibull functions, i.e. asymmetric bell curves, where the probability of sampling a value greater that the expected value is higher than 1/2. This type of function represents the fact that delays are more probable than advances. To choose our distribution, we have reused data for another line, namely the metro line 1 of Santiago, and considered histograms of trip duration for a track portion of size comparable with the space between locations. A typical distribution is given in figure 7.

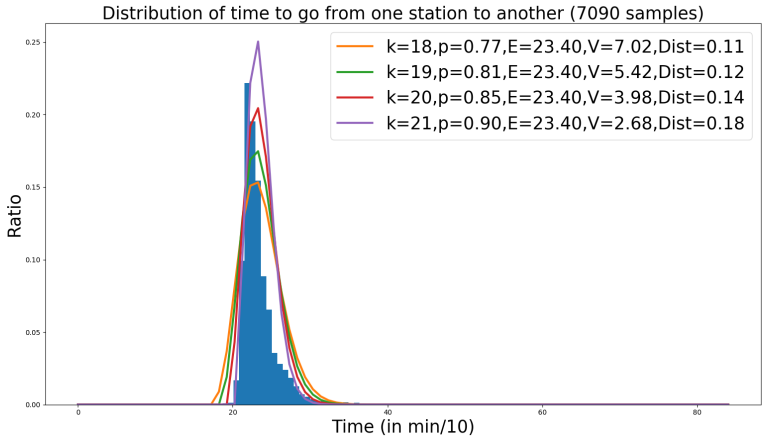

Fig. 7: Histogram (blue) and distributions for the time needed to go from one station the next one.

The blue histogram depicts the actual distribution to go from one station to the next one in the Santiago subway. We can see, as expected, that is more likely to take some delay than to be in advance compared to the nominal time (the mean of the distribution). On this histogram, four curves are drawn which correspond to the probability distribution for the time needed to go from one station to the next one in our model when fixing different values for p and k. These curves represent the sum of k independent geometric law of the same parameter p, with the same mean value E as the blue distribution. Each curve corresponds to the sum of k independent geometric laws of parameter p, for distinct values of p and k. For each curve, value $Dist$ is a distance computed as the sum of squares of differences between the distribution and the curve.

The choice of $p_a = 0.8$ gave a good approximation of the blue histogram. As shown above, this value corresponds to a realistic average speed of $38km/h$

for standard commercial speed of trains. Once p_a was fixed, the only remaining parameters to fix were the probabilities corresponding to lower speeds (in the above case a single one, corresponding to action b), and the number of trains nb_{train} running on the network to build a product MDP depicting the overall behavior of the whole system. Then a value for $p_b = 0.6$, corresponding to a speed around 30 $km.h^{-1}$ was integrated to the model. We have considered fleets of trains of sizes 4 and 6. According to the Glasgow subway website [1], at off-peak time, there is a train every six minutes, with a complete circuit lasting 24 min, this corresponds to $nb_{train} = 24/6 = 4$. And at peak-time, there is a train every 4 minutes, which corresponds to $nb_{train} = 24/4 = 6$. For discretization, we have consider two sets of parameters, computed according to discretization: $k = 5$, $\Delta t \simeq 10$ s, $\Delta d = 140$ m and $k = 10$, $\Delta t \simeq 5$ s, $\Delta d = 70$ m. The first set of parameters helps keeping the size of the model small enough, the second one gives larger models but with a better accuracy, as trains positions is updated every $\Delta t \simeq 5s$. Note that increasing too much the value of k is not possible with the model used for this experiment : increasing k above 10 leads to distances between locations that are smaller than the size of a train, which does not fit with our location based MDP, where a train can only be in one location at a time.

B Semantics of PCTL

The standard logic to describe properties in PRISM Is PCTL Given a set AP of atomic proposition, a PCTL formula is a logical sentence of the form:

$$\phi ::= true \mid pr \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \mathbb{P}_{\bowtie p}[\psi] \tag{2}$$

where $pr \in AP$ is an atomic proposition, ϕ_1, ϕ_2 are PCTL formulas, $p \in [0, 1]$, $\bowtie \in \{<, \leq, >, \geq\}$, and ψ is a formula of the form

$$\psi ::= X\phi \mid \phi_1 \, U^{\leq k} \, \phi_2 \mid \phi_1 \, U \, \phi_2 \tag{3}$$

where $k \in \mathbb{N}$. Formulas of the form (2) will be called state formulas, and formulas of the form (3) will be called path formulas.

Let us now detail the semantics of PCTL In a nutshell. Let $\mathcal{M}$ be an MDP, s be one of its state. We will write $\mathcal{M}, s \models \phi$ if formula ϕ holds for $\mathcal{M}$ starting at state s. More precisely, we can define truth of a formula inductively. Let Pol be a set of policies, and let $\pi(k)$ denote the k^{th} state of a path π.

- $\mathcal{M}, s \models_{Pol} true$ (this formula is always satisfied)
- $\mathcal{M}, s \models_{Pol} c$ iff $c \in \ell(s)$
- $\mathcal{M}, s \models_{Pol} \phi_1 \wedge \phi_2$ iff $\mathcal{M}, s \models \phi_1$ and $\mathcal{M}, s \models \phi_2$
- $\mathcal{M}, s \models_{Pol} \neg\phi$ iff $\mathcal{M}, s \models \phi$ does not hold
- $\mathcal{M}, s \models_{Pol} \mathbb{P}_{\bowtie p}[\psi$ iff $\mathbb{P}^{\sigma}_{\mathcal{M},s}(\{\pi \in \mathcal{P}ath_{\mathcal{M},s} \mid \pi \models \psi) \bowtie p$ for every policy $\sigma \in Pol$

For path formulas, we define

- $\pi \models_{Pol} X\phi$ if $\pi(1) \models \phi$
- $\pi \models_{Pol} \phi_1 \, U^{\leq k} \, \phi_2$ iff there exists $i \leq k$ such that $\pi(i) \models \phi_2$ and $\pi(j) \models \phi_1$ for every $j < i$.
- $\pi \models_{Pol} \phi_1 \, U \, \phi_2$ iff there exists $k \geq 0$ such that $\pi \models_{Pol} \phi_1 \, U^{\leq k} \, \phi_2$.

An usual shortcut is the operator $F\phi \equiv true\, U\, \varphi$ which says that φ will eventually become true, and its bounded depth counter part $F^{\leq k}\phi \equiv true\, U^{\leq k}\, \varphi$ that says that φ will become true in less than k steps. PCTL formulas do not refer to non-determinism in MDPs due to actions. Indeed, a formula is satisfied with respect to possible policies allowed in a particular set Pol. We will write $\mathbb{P}^{\sigma}_{\mathcal{M},Pol,s}(\psi)$ to denote the probability that a path from s satisfies path formula ψ with policy σ. With this notation, for a fixed set of policies, we can define a minimum and maximum probability to satisfy a path formula with

$$\mathbb{P}^{min}_{\mathcal{M},Pol,s}(\psi) = \inf_{\sigma \in Pol} \mathbb{P}^{\sigma}_{\mathcal{M},Pol,s}(\psi) \text{ and } \mathbb{P}^{max}_{\mathcal{M},Pol,s}(\psi) = \sup_{\sigma \in Pol} \mathbb{P}^{\sigma}_{\mathcal{M},Pol,s}(\psi)$$

It is well known [3] that these minimal and maximal probabilities are achieved with finite memory policies (and even with memoryless policies if the $U^{\leq k}$ operator is not used). Hence, for reachability properties, memoryless policies are sufficient.

To verify $\mathbb{P}_{\leq p}[\psi]$, it is hence sufficient to compute $\mathbb{P}^{max}_{\mathcal{M},Pol,s}(\psi)$ and check that this value is smaller or equal to p. Similarly verify $\mathbb{P}_{\geq p}[\psi]$ amounts to verifying that $\mathbb{P}^{min}_{\mathcal{M},Pol,s}(\psi)$ is greater or equal to p. In the paper, we mainly use formulas of the form $F^{\leq q}pr$ and compute $\mathbb{P}^{max}_{\mathcal{M},Pol,s}(F^{\leq q}pr)$, to obtain the maximal probability to reach a state satisfying $pr \in AP$ in less than q steps.

C Building an abstraction up to rotation

A partition of a set of states S is a set of subsets of S $\mathcal{Q} = \{S_1, S_2, \dots S_k\}$ such that $\bigcup_{i \in 1..k} S_i = S$ and for every $i \neq j$, $S_i \cap S_j = \emptyset$

Definition 3 (Abstract MDP). *Let an MDP $\mathcal{M} = (S, Act, \boldsymbol{P}, s_{init}, AP, L)$ and let $\mathcal{Q} = \{S_1, S_2, \dots S_k\}$ be a partition of the state space S that respects the labeling. The quotient of $\mathcal{M}$ by $\mathcal{Q}$ is given by the abstract MDP $\mathcal{M}/\mathcal{Q} = (\mathcal{Q}, Act', \boldsymbol{P}/\mathcal{Q}, q_{init}, AP, L/\mathcal{Q})$ where*

- *Act' is the set of action to use in the abstract MDP, after renaming the actions in Act;*
- *$\boldsymbol{P}/\mathcal{Q}$ is a transition probability matrix such that $(\boldsymbol{P}/\mathcal{Q})(q, \alpha', q') = \sum_{s \in q'} \boldsymbol{P}(s, \alpha, s')$ (where α' is a renaming of α);*
- *q_{init} is the set of initial state such that $q \in q_{init}$ if there is $s \in q$ such that $s \in s_{init}$;*
- *$L/\mathcal{Q}$ is the labeling function such that $(L/\mathcal{Q})(q) = L(s)$ for all $s \in q$.*

Let s, s' be two states of the MDP associated with a ring network, with K stations and a discretization factor k and $\mathsf{nb}_{\mathsf{trains}}$ trains. This MDP has $K \times k$ locations. We will say that s and s' belong to the same equivalence class iff, there exists an integer $0 \leq a \leq K$ such that for every train $T_i, i \in 1..\mathsf{nb}_{\mathsf{trains}}$ $pos(s', i) = pos(s, i) + a \cdot k \mod (K \times k)$. Note that equivalent states are both balanced or unbalanced, and that from equivalent states, the minimal number of steps before a train arrives in a station is the same.