Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 310 (2015) 157–177

www.elsevier.com/locate/entcs

Patch-based Modelling of City-centre Bus Movement with Phase-type Distributions

Daniël Reijsbergen, Stephen Gilmore and Jane Hillston[1,2]

Laboratory for Foundations of Computer Science
The University of Edinburgh
Edinburgh, Scotland

Abstract

We propose a methodology for constructing a stochastic performance model of a public transportation network using real-world data. Our main data source consists of Automatic Vehicle Location (AVL) measurements of buses in the Edinburgh region. Although the data has a relatively low frequency, we can use it to parameterise a model in which a bus moves between predefined patches in the city. We fit the probability distributions of the sojourn times in the patches to phase-type distributions using the tool HyperStar. We then translate the output from HyperStar to a model of a complete part of a bus route expressed in the reactive modules language of the PRISM model checker. Finally, we demonstrate how we can use the numerical techniques implemented in PRISM to answer meaningful questions about the performance of the bus network in the context of a case study involving the addition of trams to a busy section of Edinburgh's city centre.

Keywords: Public transportation, phase-type fitting, model checking.

1 Introduction

A well-functioning public transportation network can help a modern city minimise road use, air pollution, fuel consumption and carbon emissions. However, in times of austerity and budget cuts, city governments and bus companies are under pressure to reduce operating costs, possibly to the detriment of customer satisfaction. In an attempt to achieve a reasonable balance between cost and service quality, external regulators are tasked by higher levels of government to evaluate whether service quality remains above a given minimum. The task of formulating sensible service requirements is non-trivial, and is complicated by the fact that the functioning of the public transportation network is subject to random behaviour. Hence, powerful stochastic models are needed both to be able 1) to assess whether current service

[1] This work is supported by the EU project QUANTICOL, 600708.

[2] Corresponding author: dreijsbe@inf.ed.ac.uk

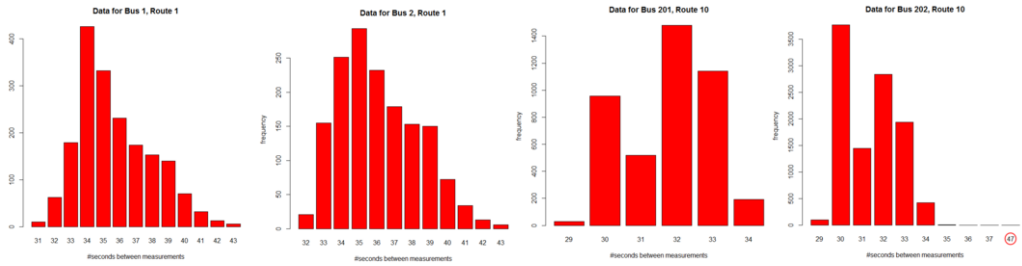
http://dx.doi.org/10.1016/j.entcs.2014.12.017

requirements can reasonably be met and 2) to gauge the impact of proposed changes to the network.

In this paper we present a data-driven methodology for constructing a formal stochastic model that can be used for both purposes. The data is provided to us by the Lothian Buses company, based in Scotland and operating an extensive bus network in Edinburgh. Having had disagreements with regulators in the past [7] over the impact of tram works in the city on timetable adherence, question 1) is of particular interest to them. Furthermore, questions of type 2) arise frequently when studying 'what-if' scenarios involving policy ideas or capacity reallocations. One policy idea suggested that the timing of traffic light phases should be tuned in order to favour late-running buses (by extending the green phase in their favour) and impede early-running buses (by extending the red phase in order to slow them down) [16]. An example of capacity reallocation is the redirection of bus routes to less congested streets during the very busy Edinburgh festival period in August. The focus of this paper is on a case study involving the introduction of trams to Edinburgh's public transportation network [6], and in particular on its impact on the punctuality of buses in Princes Street, a busy shopping street and the main artery connecting the west and east of the city.

The data consists of GPS measurements of bus locations throughout the day, reported to a central server roughly every 30-40 seconds. To cope with the relatively low frequency of the measurements, we divide the part of the route under study into a sequence of patches. We then propose a stochastic model that models the continuous-time movement of buses between the patches. The probability distribution of the time spent in each patch is modelled as a hyper-Erlang distribution. The hyper-Erlang distribution allows for easy parameterisation using the tool HyperStar [13] and gives a good fit to the data when parameterised. Also, the hyper-Erlang distributions of the individual patch sojourn times can be used as building blocks to construct a Continuous-Time Markov Chain (CTMC), which in turn can be concisely expressed using the reactive modules-based language of the PRISM [9] model checker. These models can then be analysed using PRISM's powerful numerical solution engine.

The structure of the paper is as follows. In Section 2, we give a more detailed account of the data, discuss the part of the city that is relevant to the case study and discuss the question of dividing this part into patches. In Section 3 we explain how to fit the empirical distributions of the sojourn times in the patches to a hyper-Erlang distribution using the tool HyperStar. In Section 4 we discuss how to use the results from HyperStar to construct models of a bus on the relevant part of the route, and discuss the formal specification of performance requirements. In Section 5, we carry out the proposed methodology and present the results. In Section 6 we present related work. Section 7 concludes the paper.



(a) (b) (c) (d)

Fig. 1. Distributions of the times between measurements for four different buses. Some distributions are heavy-tailed, for example in (d), a 47-second delay is observed: note the discontinuous axis.

2 Data availability and defining patches

Since our methodology is data-driven, a clear understanding of the available data is vital. An important feature of the available data is the relatively low frequency of measurements, which is best suited for a model in which the spatial representation of bus locations is coarse-grained. Hence we divide the part of the city that is relevant to our case study into *patches*. In this section, we begin with a discussion of the available data, before moving on to a description of the patch structure.

2.1 Automatic Vehicle Location (AVL) data

The data provided to us by Lothian Buses consists of AVL data obtained using periodic GPS location measurements. Each data entry consists of a bus identifier, a location measurement and a measurement timestamp. Our dataset contains information about the full fleet of 745 buses, collected between 28th January 2014 at 11:31:14 and 30th January 2014 at 12:38:31.

The data is gathered through centralised pull requests. Consequently, the measurements generally align within a dataset, i.e., if a measurement exists for a certain point in time for one bus then a measurement should exist at the same point in time for all other buses in the same dataset. The main exception to this rule is that when a bus has not moved between two measurements, the second measurement is not recorded (and similarly for larger sequences). The time between measurements is generally between 30 and 40 seconds. Distribution plots are given in Figure 1.

The data is given in Ordnance Survey for Great Britain (OSGB) eastings and northings rather than latitude/longitude coordinates. Once converted to latitude/longitude coordinates, the measurements have a horizontal (longitudinal) offset of about 0.00143 degrees. The GPS measurements are quite accurate once the offset is removed; we can determine the direction of the bus by the side of the road on which it is driving, illustrated in Figure 2.

The data is not always perfect: in extreme cases, this means that the location data is obviously invalid (e.g., buses appearing in a patch of farmland or the middle of the Firth of Forth). In less obvious cases, buses seem to freeze for extended time periods before suddenly appearing in distant locations. Buses appearing to freeze

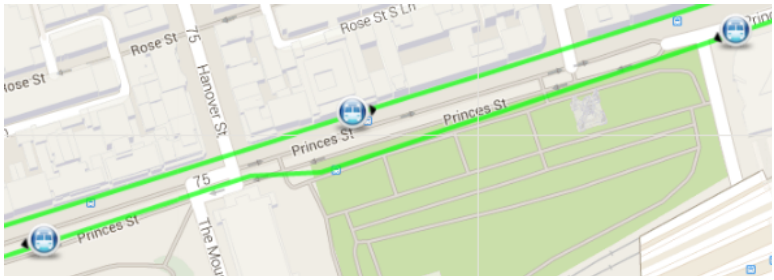


Fig. 2. The data can be accurate enough to confirm the direction of the bus by inspecting visually the side of the road on which it is driving.

in locations in the middle of the city may have a significant impact on journey time analysis. Hence, it is advisable to have a human check the data for obvious errors using a visualisation tool, for example [18]. Self-evidently erroneous observations seem to disproportionately often occur at night; hence, we restrict our data set to only include measurements recorded between 7:30 AM and 7:30 PM.

Although the data entries include identifiers which uniquely identify a bus, it is non-trivial to match a bus identifier to a bus route because the assignment of buses to routes may vary between days (and even sometimes hours). Currently, we do not identify the route to which a bus belongs, so sojourn time measurements for each patch (discussed in more detail in Section 2.2) are calculated for all buses that pass through. We would wish to be able to automatically match buses to routes in the future but this is dependent on the availability and release of a look-up table mapping buses to routes at a sufficiently detailed granularity.

A positive aspect of AVL data is that it is amenable to *interpolation*. This helps to ameliorate the problems of low frequency data by allowing us to insert additional points between reported data. Using linear interpolation, we insert a mid-point between adjacent points to generate (approximate) 18-second resolution data from (genuine) 36-second resolution data. Repeating this process, we generate (approximate) 9-second resolution data from (approximate) 18-second resolution data. This interpolation is reasonable because the latitude and longitude position of a bus changes incrementally in continuous space. Thus, we supplement our reported data in this way where necessary in the belief that we will not be introducing significantly erroneous values through our use of linear interpolation. An example of this interpolation is shown in Figure 3.

Once buses are mapped to routes, we will be able to implement a better interpolation in which corners are never cut short. Another possibility for future research would be to implement the more refined interpolation technique of [17], in which the interpolation is based on a second dataset of historical bus location measurements.

2.2 Patch selection

The main idea behind our formal model of bus movements is to abstract the behaviour of the buses by dividing their routes into patches. Our model then describes the movement of buses between patches. Specifically, we are interested in the time

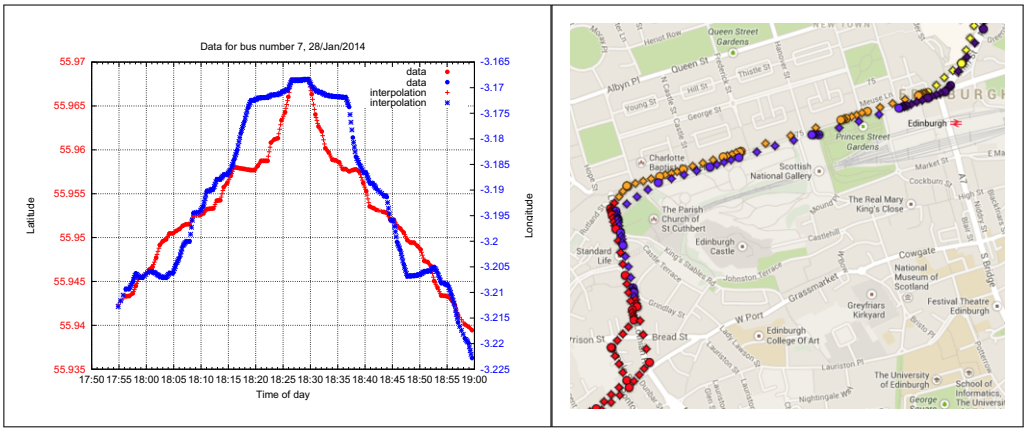


Fig. 3. The same interpolated AVL data set is interpreted semantically on a graph as latitude and longitude time series (left) and rendered visually on a map (right, detail). In the visualisation, the true data points are represented by circles, and the interpolated points are represented by diamonds. The passing of time is represented by the spectrum so that the data cycles through red, orange, yellow, green, blue, indigo and violet, before returning to red again. This shows that the bus first passes along Princes Street from west-to-east and then returns back, travelling east-to-west. The colours represent the passage of time in an abstract sense, and explain the symmetry in the graph by showing that the bus is returning along the route travelled in the outgoing part of the journey.

spent by a bus in each of the patches. An example of a division of a bus route into patches is displayed in Figure 4. Currently, the patches are manually selected.

In this paper, we focus on a specific portion of the city, namely Princes Street in the Edinburgh city centre. Alongside being a busy shopping street, it is the main artery connecting the west and east of the city and nearly all buses that cross the Edinburgh city centre follow a route that takes them along at least a part of Princes Street. The advantage of focusing on a single street is that the patches can be ordered, in this case starting from the south-west.

The choice of patches on Princes Street is non-trivial; e.g., one has the choice between a regular or an irregular grid. The patch division of Figure 4 is irregular, and is based on the location of bus stops and traffic lights along the route. Specifically, a patch represents either (i) the area directly around a junction, or (ii) a large area between patches of type (i). Bus stops are always contained in patches of the second category. This choice is motivated by the fact that we are not yet able to automatically map a bus to a route, as discussed in Section 2. Even without knowledge of the exact route that is being serviced by a bus, we know that when it enters a patch of type (ii), it always has to complete the entire patch. This enables us to fit the distribution of the sojourn time to a distribution that is as simple as possible. Also, in the area of the city which we are considering here, buses stop at at most one bus stop inside a single patch. Not all buses stop in every patch with bus stops, but most do. There are exceptions to this rule-of-thumb, such as the number 36 bus which runs from Ocean Terminal to Holyrood.

3 Sojourn time distributions

In this section, we describe how we fit hyper-Erlang distributions to empirical sojourn time distributions. We begin by formally defining the hyper-Erlang distribution, after which we discuss the tool HyperStar and the parameter fitting results.

3.1 Hyper-Erlang distributions

Although we assume that the reader is familiar with the concept of CTMCs, we begin with the very basics, namely the exponential distribution. A random variable X is said to be *exponentially distributed* with rate λ, $\lambda > 0$, if

$$\mathbb{P}(X < x) = \begin{cases} 1 - e^{-\lambda x} & \text{if } x \geq 0, \\ 0 & \text{if } x < 0. \end{cases}$$

If $X_1, \ldots, X_k$ are all pairwise independent and exponentially distributed with rate λ and $k \in \mathbb{N}$, then $S = \sum_{i=1}^{k} X_i$ is *Erlang distributed* with rate λ and shape k. The probability density function (pdf) f_S of S is given by

$$f_S(x) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}. \tag{1}$$

A hyper-Erlang distributed random variable is constructed in the following way: given some $m \in \mathbb{N}$, let $\alpha_1, \ldots, \alpha_m$ be a probability distribution over $\{1, \ldots, m\}$, i.e., $\forall i,\ \alpha_i \in [0,1]$ and $\sum_{i=1}^{m} \alpha_i = 1$. Furthermore, let S_i, $i \in \{1, \ldots, m\}$, be Erlang distributed with rate λ_i and shape k_i — these random variables are called the *branches* of the hyper-Erlang distribution. We then say that a random variable Y is *hyper-Erlang distributed* with rates $\lambda_1, \ldots, \lambda_m$, shapes $k_1, \ldots, k_m$ and branch

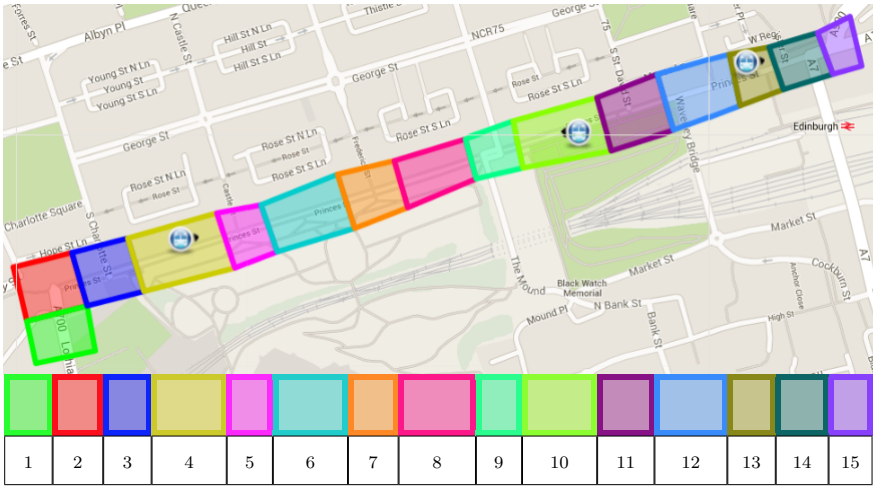


Fig. 4. Buses in two- and one-dimensional space; part of the route, including patches. Note that, in 1D, we assume for both directions that they cross the same amount of space in the leftmost red patch, although it is obvious from the 2D map that this is not the case.

probabilities $\alpha_1, \ldots, \alpha_m$ if

$$\mathbb{P}(Y < y) = \sum_{i=1}^{m} \alpha_i \mathbb{P}(S_i < y) \qquad \text{and} \qquad f_Y(y) = \sum_{i=1}^{m} \alpha_i \frac{\lambda^{k_i} x^{k_i - 1} e^{-\lambda_i x}}{(k_i - 1)!}, \tag{2}$$

where f_Y denotes the pdf of Y. Realisations of Y are easily drawn as follows: choose an integer i from $\{1, \ldots, m\}$ with probability α_i, draw k_i realisations from the exponential distribution with rate λ_i and take the sum.

The hyper-Erlang distribution is a *phase-type distribution*: it can be described in terms of the time until absorption in a CTMC. An example of such a CTMC representing a hyper-Erlang with three branches is given in Figure 5.

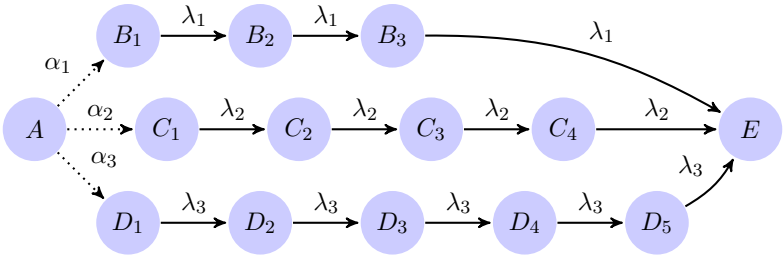


Fig. 5. An example three-branch (3/4/5)-stage hyper-Erlang distribution represented as a CTMC. Dotted arrows are used to indicate instantaneous probabilistic transitions. Solid arrows are used to indicate transitions with exponentially distributed durations.

An important motivation for the use of the hyper-Erlang distribution is that the Erlang and hyper-Erlang distributions exhibit behaviour that is commonly observed in the analysis of transportation systems. The distribution that is used in [17] and recommended by the Traffic Engineering Handbook [10] is a three-parameter gamma distribution that is related to the Erlang distribution in the following way. The standard (two-parameter) gamma distribution is a generalisation of the Erlang distribution in the sense that its pdf is equivalent to (1) but with a gamma function replacing the factorial, hence allowing the parameter k to be any positive real rather than just a positive integer. The aforementioned three-parameter gamma distribution is a standard gamma distribution plus a constant. This constant compensates for the fact that for large values of k (which often arise when fitting gamma distributions to sojourns in large patches), the gamma distribution resembles the normal distribution. The normal distribution is symmetric, while patch sojourn time distributions tend to be skewed towards the right (i.e., towards large delays).

We do not use the three-parameter gamma distribution in this paper for the following reasons. First, both the generalisation from discrete to continuous values of k and the addition of the deterministic constant prohibit the model from being expressed in the form of a CTMC, and thus from using the powerful numerical techniques implemented in PRISM. Furthermore, because the patches under consideration are relatively small, the impact of the additional constant in our setting is not very great. Third, using hyper-Erlang distributions allows us to include a skewness to the right by mixing two Erlang distributions with different peaks.

3.2 HyperStar

The software tool HyperStar [13] combines an efficient parameter fitting algorithm for phase-type distributions with a user-friendly interface. After choosing the dataset, the user is asked to visually identify peaks in the data, which would then roughly correspond to the means of the individual Erlang branches. The HyperStar algorithm respects the number of peaks identified by the user and will return a phase-type distribution with that number of branches. Given m peaks, there are $3 \times m$ parameters to optimise, namely λ_i, k_i and α_i for each $i \in \{1, \ldots, m\}$.

The criterion over which we optimise is given by the *log-likelihood*: the log-likelihood of a sojourn time measurement t is equal to $\log f_Y(t)$, where f_Y is as defined in the second equation of (2). The log-likelihood of the entire sample is then the sum of all the individual log-likelihoods. The higher the log-likelihood, the better. HyperStar allows the user to choose one among several phase-type fitting algorithms; for the analysis in Section 5, we use the standard procedure, namely cluster-based fitting as described in [12].

3.3 HyperStar results

In this section we discuss the results of the parameter fitting carried out using HyperStar. The structure of the section is as follows: we first introduce the three means of presenting the data, namely Tables 1 and 2 and Figure 6. We then move on to interpretations of the results, particularly those that will be relevant to the case study of Section 5.

In Figure 6, the pdf and cumulative distribution function (cdf) of the hyper-Erlang distribution fitted to each patch are graphically depicted alongside their empirical equivalents. Figure 6 can be used to visually measure the goodness-of-fit of the hyper-Erlang distribution, and to investigate properties of the empirical distributions such as tail behaviour and the existence of multiple peaks.

In Table 1, the exact values of the hyper-Erlang parameters are given together with other detailed information. We only consider hyper-Erlang distributions with two branches, as we explain later. Alongside the standard hyper-Erlang parameters, three other metrics that give important information about the expected sojourn time are presented. The first two are μ_1 and μ_2 which equal k_1/λ_1 and k_2/λ_2 respectively; these values correspond to the means of the two individual Erlang branches. The third metric is $\mu = \alpha_1\mu_1 + \alpha_2\mu_2$, which corresponds to the mean of the hyper-Erlang distribution as a whole. The branches are ordered by the branch means; i.e., for each patch α_1, k_1 and λ_1 correspond to the parameters of the Erlang branch with the shortest time until completion. Finally, we indicate for each patch whether or not it contains a bus stop, as this has a major impact on the expected sojourn time.

In Table 2, we compare the goodness-of-fit of the Erlang distribution to that of the two-branch hyper-Erlang distribution for each of the 15 patches. We compare the goodness-of-fit in terms of *cdf-diff*, a measure related to the difference between the empirical and fitted cdfs. Low values of *cdf-diff* correspond to a good fit. The optimisation procedure implemented in HyperStar seeks to maximise the

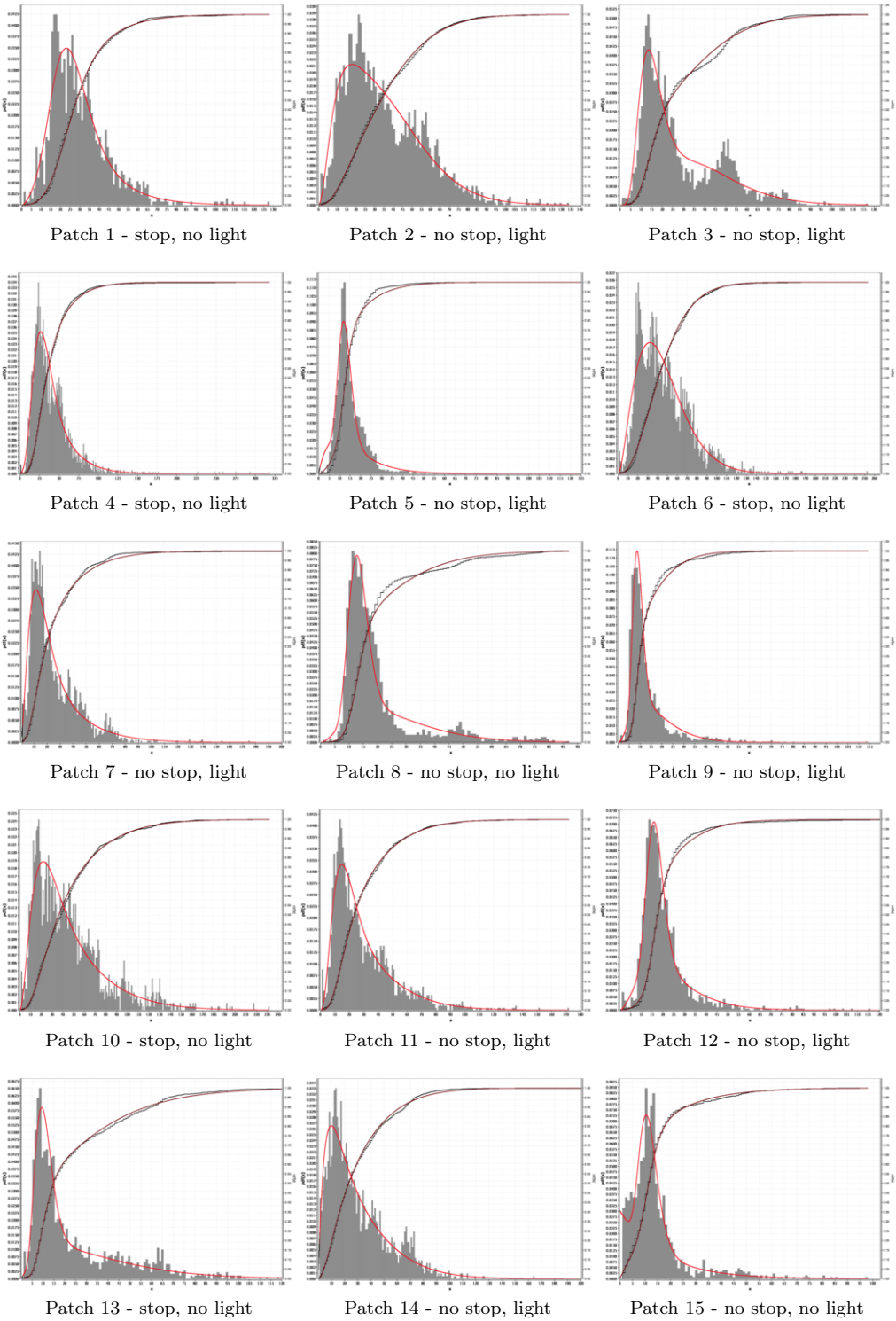


Fig. 6. Empirical sojourn time distributions and corresponding hyper-Erlang fits for each of the 15 patches. The labels on the axes are readable in the digital version of this paper; the unit of measurement on the x-axes is seconds. Note that the scale of both the x-axis and the y-axis may vary between subfigures.

Patch	(α_1, α_2)	(k_1, k_2)	(λ_1, λ_2)	(μ_1, μ_2)	μ	Stop?
1	(0.4938, 0.5062)	(8, 3)	(0.3041, 0.0899)	(26.31, 33.38)	29.89	yes
2	(0.4970, 0.5030)	(3, 5)	(0.1374, 0.1083)	(21.83, 46.17)	34.07	no
3	(0.5298, 0.4702)	(7, 5)	(0.4536, 0.1178)	(15.43, 42.44)	28.13	no
4	(0.5043, 0.4957)	(6, 3)	(0.1949, 0.0618)	(30.78, 48.57)	39.60	yes
5	(0.5320, 0.4680)	(16, 2)	(1.2626, 0.1144)	(12.67, 17.48)	14.92	no
6	(0.5065, 0.4935)	(4, 6)	(0.1309, 0.0983)	(30.56, 61.02)	45.59	yes
7	(0.4939, 0.5061)	(3, 2)	(0.1857, 0.0572)	(16.15, 34.98)	25.68	no
8	(0.5568, 0.4432)	(17, 3)	(1.2685, 0.1108)	(13.40, 27.07)	19.46	no
9	(0.5239, 0.4761)	(16, 3)	(1.7485, 0.1660)	(9.15, 18.07)	13.40	no
10	(0.5146, 0.4854)	(3, 3)	(0.1025, 0.0488)	(29.28, 61.43)	44.89	yes
11	(0.5098, 0.4902)	(4, 3)	(0.2206, 0.0736)	(18.13, 40.77)	29.23	no
12	(0.5378, 0.4622)	(17, 3)	(1.0080, 0.1230)	(16.87, 24.39)	20.34	no
13	(0.4800, 0.5200)	(7, 2)	(0.6398, 0.0519)	(10.94, 38.50)	25.27	yes*
14	(0.4883, 0.5117)	(2, 3)	(0.1229, 0.0664)	(16.27, 45.18)	31.07	no
15	(0.4958, 0.5042)	(10, 1)	(0.8518, 0.0616)	(11.74, 16.23)	14.00	no

Table 1
HyperStar results for each of the patches.
*: there is a bus stop in Patch 13, but it is only used by two routes (22 and 25).

	number of branches			
Patch	1	2	bus stop	junction
1	0.9647	0.6900	yes	no
2	1.5643	0.9805	no	yes
3	3.3444	1.2408	no	yes
4	2.3328	1.4514	yes	no
5	1.9218	1.1134	no	yes
6	0.9710	0.9320	yes	no
7	2.0532	1.7146	no	yes
8	3.5191	1.3365	no	no
9	2.1952	0.7391	no	yes
10	1.3143	1.2243	yes	no
11	1.3341	0.8073	no	yes
12	2.8960	0.9251	no	yes
13	4.6573	1.2144	yes*	no
14	2.3440	1.1864	no	yes
15	2.0293	0.5990	no	yes

Table 2
In this table, we compare for a given patch the *cdf-diff* statistic for several choices of the number of Erlang branches. A lower value of *cdf-diff* means a better fit. We do not consider hyper-Erlang distributions with more than two branches, as we have observed that this often leads to over-fitting (e.g., two or more Erlang branches that are close to the mode to just slightly improve the fit for a large number of samples).
*: there is a bus stop in Patch 13, but it is only used by two routes (22 and 25).

log-likelihood, so a low value for cdf-diff is merely a by-product. We do not consider hyper-Erlang distributions with more than two branches in Table 2. We do this for two reasons. First, the optimisation procedure used by HyperStar uses randomisation to move away from local optima, and for higher numbers of Erlang branches the results start to vary considerably between runs. Second, we have observed that using more than two branches often leads to over-fitting, mostly witnessed by mixtures of Erlang distributions with two very similar means around a large peak.

Regarding the interpretation of the resulting values, we first observe that for each of the patches, the Erlang branch means μ_1 and μ_2 are noticeably different, and that the branch probabilities α_1 and α_2 are all relatively close to $(\frac{1}{2}, \frac{1}{2})$. For the patches without bus stops, an interpretation of the two branches is that the branch with the lowest mean (i.e., μ_1) represents the probability distribution of the time spent in a patch if the bus is allowed to cross the patch largely unhindered by traffic lights. For the patches with bus stops, the time spent at the bus stops is noticeable even if the Erlang branch with the lowest mean is taken. An illustration of this is that μ_1 is over twice as high for Patch 6 as for Patch 8, despite the fact that the size of these patches is roughly the same. The patches with bus stops tend to be better approximated using an Erlang distribution than the ones without. This is witnessed by the fact that the *cdf-diff* results for the Erlang distribution (i.e., the hyper-Erlang distribution with one branch) for Patches 1, 6 and 10 in Table 2 gives a relatively good fit; the exception is Patch 4, which has a very drawn-out tail.

The impact of the traffic lights is very noticeable in some patches. For example, the distribution of the time spent in Patch 3 has more than one noticeable peak, as can be seen in Figure 6. Also, the sojourn time distributions of Patches 8 and 13 have very drawn-out tails which cannot be modelled accurately by an Erlang distribution. This can be seen in Table 2: the Erlang distribution has a relatively poor fit for Patches 3, 8 and 13. One very notable difference caused by the traffic light behaviour is between Patches 7 and 9, which have a similar size and similar traffic patterns but, as can be seen in Table 1, a large difference in terms of μ. The main difference between the two patches is that traffic is not allowed to turn in Patch 9, so that the traffic lights only need three phases: one for west-east traffic, one for north-south traffic and one for pedestrians. The traffic lights in Patch 7 need more phases, leading to a significant increase in the expected sojourn time. We will use this observation about the difference in number of traffic light phases for our analysis of the PRISM models in Section 4.1.

Traffic intensity varies along the route. A very noticeable example of this is in the part of Princes Street that is traffic-restricted (i.e., no cars on the road), namely the area that roughly corresponds to Patches 4, 5 and 6. We observe that Patch 5 has a much lower expected sojourn time than Patches 3 and 13 than would be expected if only the difference in size were considered. The effect of the pedestrianisation does not seem to extend to Patch 7, as there is not much difference compared to Patch 3, which is on the other side of the pedestrianised area.

Concerning current timetable adherence, note that if we sum the values of μ in Table 1 for each of the patches we find that the total time needed to cross the entire

set of 15 patches equals approximately 415 seconds (almost seven minutes). It is not immediately obvious what the timetable says for these stops, as the online timetables give arrival times only for several main stops along the route. However, we can use the journey planner on the website of Lothian Buses to obtain an estimate for the timetabled duration of the entire route. To do this, we focus on the Routes 1 and 34, which stop both at the stop named 'Caledonian Hotel' (bus stop code 36232692) in Patch 1 and at the stop named 'Leith Street' (bus stop code 36242687) which is immediately after Patch 15. The procedure is then as follows: we use the journey planner to get the arrival time of a bus 1 or 34 at Caledonian Hotel roughly 30 minutes in the future, and then compare this to the nearest time after this at Leith Street. The difference is then an estimate of the time to complete the route as specified by the timetable. Although the measurements vary depending on the time of day, we have generally observed that the projections correspond to a journey time of roughly six minutes between the two stops.

4 Modelling

Having obtained the hyper-Erlang parameters, we use them to construct a model that can answer questions about the case study involving trams in Edinburgh. We first describe the model in a formal and concise manner using PRISM. We then discuss the performance measures used to evaluate the impact of the trams.

4.1 The PRISM model

PRISM works with models expressed in its own language, based on the formalism of reactive modules [1]. In the following we present a brief description of the PRISM language that is not fully general but which covers the scope of this paper. For a full description, see [11].

As the name suggests, the idea behind the reactive modules formalism is to decompose a complex system model into a combination of interacting *modules*. A typical module includes a set of local variables that together represent the state of the module; the global state of the model can then be derived using the states of all the individual modules. The transition structure of the model is described using *commands*, where each command is linked to a specific module. The commands

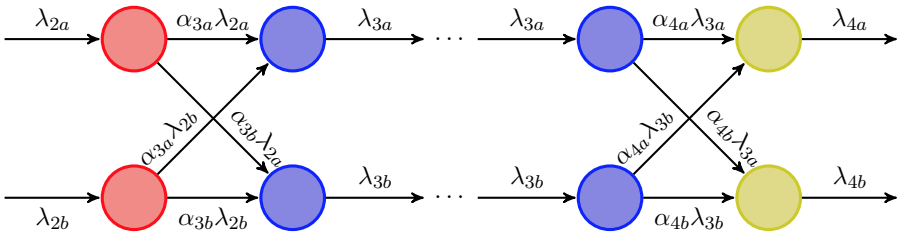


Fig. 7. CTMC representation of part of the PRISM model. The red states correspond to the bus being in Patch 2, the blue states correspond to the bus being in Patch 3 and the yellow states correspond to the bus being in Patch 4. When a bus moves between patches, a probabilistic choice is made between the two Erlang branches of the next patch.

```
module Patch3

BusesInP3a : [0..1] init buses3a;
BusesInP3b : [0..1] init buses3b;
Phases3a : [1..k3a] init k3a;
Phases3b : [1..k3b] init k3b;

//buses coming into Patch 3
[p2aTo3a] (BusesInP3a = 0) → 1 : (BusesInP3a′ = BusesInP3a + 1);
[p2aTo3b] (BusesInP3b = 0) → 1 : (BusesInP3b′ = BusesInP3b + 1);
[p2bTo3a] (BusesInP3a = 0) → 1 : (BusesInP3a′ = BusesInP3a + 1);
[p2bTo3b] (BusesInP3b = 0) → 1 : (BusesInP3b′ = BusesInP3b + 1);

//internal transitions
[phCom3a] (Phases3a > 1 & BusesInP3a = 1) → lambda3a : (Phases3a′ = Phases3a − 1);
[phCom3b] (Phases3b > 1 & BusesInP3b = 1) → lambda3b : (Phases3b′ = Phases3b − 1);

//buses leaving Patch 3
[p3aTo4a] (Phases3a = 1 & BusesInP3a = 1) → lambda3a ∗ alpha4a :
  (BusesInP3a′ = BusesInP3a − 1) & (Phases3a′ = k3a);
[p3aTo4b] (Phases3a = 1 & BusesInP3a = 1) → lambda3a ∗ alpha4b :
  (BusesInP3a′ = BusesInP3a − 1) & (Phases3a′ = k3a);
[p3bTo4a] (Phases3b = 1 & BusesInP3b = 1) → lambda3b ∗ alpha4a :
  (BusesInP3b′ = BusesInP3b − 1) & (Phases3b′ = k3b);
[p3bTo4b] (Phases3b = 1 & BusesInP3b = 1) → lambda3b ∗ alpha4b :
  (BusesInP3b′ = BusesInP3b − 1) & (Phases3b′ = k3b);

endmodule
```

Fig. 8. PRISM language description of a typical patch (Patch 3) in the model.

that we use consist of an *action label*, a *guard*, a *rate* and an *update*. A *guard* is a condition imposed on the local variables of the module; for a transition with a given *action label* to be enabled, the guards of all commands in the system that share this action label must be satisfied. Each individual command has an associated *rate*: the total rate of the transition is the product of the individual rates. In each step of the execution of the system model, a transition is chosen based on the rates of the enabled transitions. Specifically, when a transition is chosen the local variables in the modules with the commands that correspond to the transition are changed in a way that is specified using their *updates*.

In our model, we use a module for each individual patch. A module for a typical patch (in this case, `Patch3`) is depicted in Figure 8. The local variables correspond to whether a bus is inside the patch and how many exponential phases have to be completed before it leaves. One of the two Erlang branches is chosen when the bus enters the patch. The choice of branch is encoded in the variables: since there are two Erlang branches per patch, each module has two variables for the buses (`BusesInP3a` and `BusesInP3b` in Figure 8). Since we are interested in the behaviour of a single bus, the maximum value for both of these variables is one. Furthermore, we encode the phases to be completed in each of the two branches separately, which results in two variables for the phase counters (`Phases3a` and `Phases3b` in Figure 8).

Each module has two internal transitions, namely the completion of phases in each of the two Erlang branches (represented by the commands with action labels

phCom3a and phCom3b in Figure 8). The modules synchronise over the transitions that take a bus from one patch to another. For each patch module, there is synchronisation with at most two other modules: the one corresponding to the patch before it (i.e., the one from which buses arrive) and the one corresponding to the patch after it. Since a bus moving to a patch can cause the bus location variable for two different Erlang branches to be reset, and because a probabilistic choice has to be made between the two branches of the next patch, adjacent modules have four action labels in common. For Patch3 (depicted in Figure 8), which has two adjacent patches, there are eight commands. Each command has an action label that is shared with one other command in a module corresponding to an adjacent patch: for example, there is also a command with action label p2aTo3a in Patch2. Note that in Patch3, the command with action label p2aTo3a is *passive*; i.e., it has rate 1 (which we have made explicit — PRISM allows us to omit the rate specification in this setting). The actual rate of this transition is given in Patch2 (recall that the total rate is the product of the individual rates). In general, for each module the action corresponding to a bus entering the patch is made passive.

There are two patches whose behaviour differs slightly from Patch3. First, the final module (Patch15) has a '*sink*' variable which is set to one when the bus has completed the patch (and, consequently, the entire route). There are two actions corresponding to a bus entering the '*sink*': one for buses leaving branch a of Patch 15 (with action label p15aToSink) and the other corresponding to branch b. Second, there is no phase species for branch a of Patch 15 because this branch only has one phase, as can be seen in Table 1. Instead, the action with label p15aToSink can be taken directly when BusesInP15a becomes greater than zero. Also, the distribution of the sojourn time in Patch 1 is given by an Erlang rather than a 2-branch hyper-Erlang distribution. The reason is technical: to be able to analyse the model using PRISM, a unique initial state is required, while if the sojourn time distribution in Patch 0 had a 2-branch hyper-Erlang distribution we would like to start in branches $c \in \{a, b\}$ with probability α_{0c}. This can be overcome if we instead use the Erlang approximation for this patch. Another possible remedy would be create a dummy patch from which one would jump to Patch 1 with a very high rate, but this creates stiffness problems for the numerical CTMC analysis so we prefer not to do this.

4.2 Performance measures

The main performance metric that we consider in this paper is *on-time performance* [5]. Specifically, we are interested in whether a service is *punctual*, which is defined by the Scottish government [14] as an arrival that occurs within a so-called *window of tolerance* of between one minute early and five minutes late. Note that government regulators only measure whether a service is punctual at pre-specified timing points, and the portion of the city that we consider does not contain such points. However, since the punctuality on a route depends on the punctuality on the portions that make up the route, this is still an interesting metric, even though the probability of being over five minutes late on such a short part of the route is fairly low.

We consider the punctuality of a bus moving from the stop 'Caledonian Hotel'

Fig. 9. The route of the tram along Princes Street, represented by the red line. There is a single tram stop on the route, indicated by the red marker near the Royal Scottish Academy. The buses that we consider drive from the blue dot in the bottom left to the orange dot in the top right.

in Patch 1 to the stop 'Leith Street' which is just after Patch 15, assuming that this bus is exactly on time when it arrives in Patch 1. As we discussed in the previous section, the relevant part of the route is completed when a bus enters the *'sink'* of Module 15. If `BusesInSink` represents the number of buses in the sink, then the probability of being over one minute early can be expressed in PRISM's property specification language (based on CSL [2,3]) as

$$\texttt{P} =?\ [\texttt{true U} < 300\ \texttt{BusesInSink} = 1],$$

where the number 300 corresponds to 300 seconds or five minutes; we assume that this portion of the route is scheduled to take six minutes as discussed in Section 3.3. Similarly, we express the probability of being over five minutes late as

$$\texttt{P} =?\ [\texttt{BusesInSink} = 0\ \texttt{U} > 660\ \texttt{BusesInSink} = 1].$$

The CTMC for our case study consists of 164 states and 191 transitions, which (comfortably) enables us to apply numerical techniques to evaluate these formulae.

5 Case study: Edinburgh trams

In this section we discuss the results obtained using PRISM. As discussed earlier, we focus on a case study involving the addition of trams to the public transportation network, and in particular on its impact on the on-time performance of the buses (as discussed in Section 4.2), assuming that the bus timetables remain unchanged. We begin with a detailed description of the case study and several associated what-if scenarios, before moving on to the numerical analysis.

5.1 Description of the case study

Trams in Edinburgh will service a single line running from the airport, located to the west of the city, to York Place, which is to the north east of the city centre. The part of the tram route that overlaps with Princes Street is depicted in Figure 9.

A single tram stop is located on Princes Street, namely in Patch 8 (to see this, compare Figures 4 and 9). A bus travelling through the patches from the west to the east (the only direction that we consider) will cross the tram line twice: in Patches 2 and 12.

Although the trams do not contend with the buses for access to stops, we investigate three possible other ways in which the introduction of trams may impact the bus network. The first is the impact of additional traffic light phases in Patches 2 and 12. The second is the impact of the addition of traffic lights to Patch 8, which is a consequence of the fact that pedestrians need to cross part of the road to reach the tram stop in the middle. The third is the impact of passengers taking the tram instead of the bus on the time spent at the bus stops of Patches 6 and 10.

5.2 Impact of trams crossing the bus route

The fact that trams cross the bus route in Patches 2 and 12 means that an additional traffic light phase is needed to accommodate this, or that an existing phase needs to be lengthened. We cannot directly observe the effect of this is because we do not have data about the traffic lights to match the AVL data, so we cannot measure exactly when a bus is waiting in front of a light. However, we can approximate the effect of this by comparing patches in the relevant part of the city of which we know that they differ in terms of traffic light behaviour; we then change the sojourn time distributions in Patches 2 and 12 such that the average behaviour changes accordingly. As mentioned in Section 3.3, the main difference between Patches 7 and 9 is the number of traffic light phases. The effect of this is clearly noticeable in the parameters of the hyper-Erlang distributions: both μ_1 and μ_2 are higher for Patch 9. In order to incorporate this effect in Patches 2 and 11, we could alter their hyper-Erlang parameters such that μ_1 and μ_2 are increased.

In Figure 10, this is investigated. We divide the rates λ_1 and λ_2 for

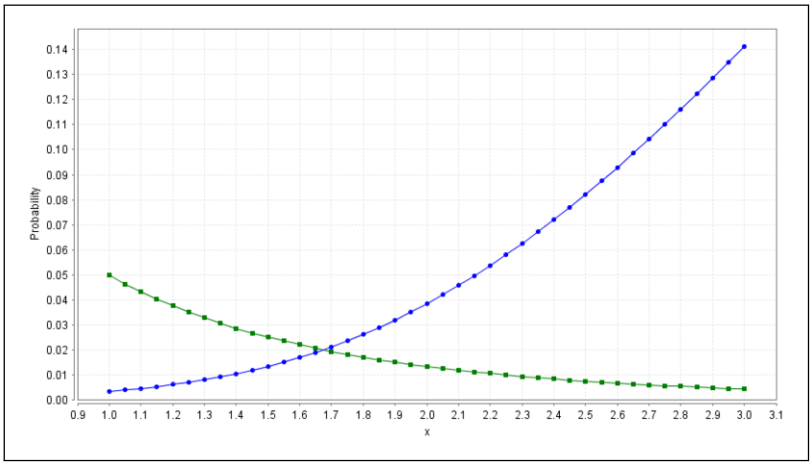


Fig. 10. Probability of not being on time as a function of x, where x is such that we divide the rates λ_1 and λ_2 for Patches 2 and 12 by this constant. The green (decreasing) line represents the probability of being more than a minute early compared to the current timetable. The blue line (increasing) represents the probability of being more than five minutes late compared to the current timetable.

Patches 2 and 12 by a constant x and consider the on-time probabilities as a function of x: we choose to change the rates instead of the number of phases because the rate variables can be set equal to any real value (we currently do not have a real-world interpretation of the difference between the rates and phases so we otherwise have no preference for changing either). The figure depicts both the probability of being more than a minute early (the green, decreasing line) and more than five minutes late (the blue, increasing line). The sum of these two probabilities is the lowest around 1.45, meaning that, according to our model, such a decrease of the phase completion rates would actually help improve on-time performance as both the total probability of being on-time and the probability of not being early (the worst kind of deviation from the timetable) would increase. As μ_1 and μ_2 are about twice as high for Patch 7 as for Patch 9, the impact of this change would be around $x = 2$, meaning that the probability of a bus being late would become higher than the probability of a bus being early; the total on-time probability is similar.

5.3 Impact of a pedestrian phase for the tram stop

The tram stop in Patch 8 is located in the middle of the road, meaning that pedestrians will need a traffic light to enable them to safely reach or leave it. The impact of a single traffic light phase added to the system can be observed in Patch 5. Since the area surrounding Patch 5 is traffic-restricted, and because buses do not turn at Castle Street (see Figure 9), the traffic lights there have only one phase. This is clearly noticeable from the bus sojourn time distributions: the difference in terms of expected sojourn time between Patches 5 and 7 is much larger than one would expect when taking only the size of the patches into consideration. However, while Patch 8 is about twice as big as Patch 5, the expected sojourn time in Patch 8 is more than twice as big as its equivalent in Patch 5. Looking at size only, the value of μ in Patch 5 would be 9.73 seconds if it were half that of Patch 8, compared to 14.92 seconds originally. The delay caused by the traffic lights would hence be

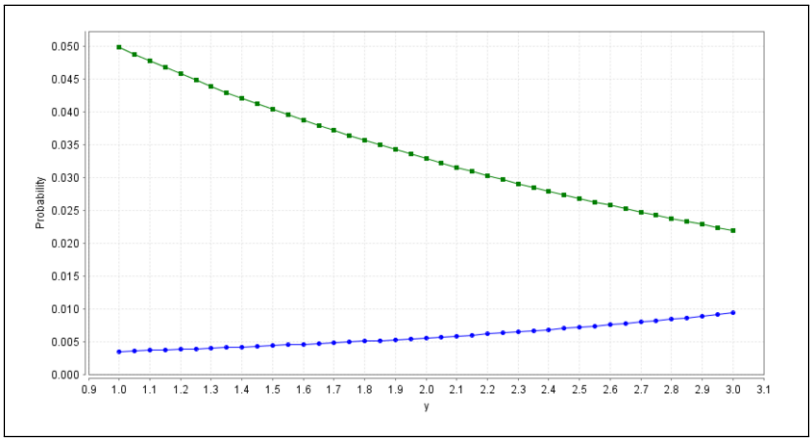


Fig. 11. Probability of not being on time as a function of y, where y is such that we divide the rates λ_1 and λ_2 for Patch 8 by this constant. The green (decreasing) line represents the probability of being a minute early compared to the current timetable. The blue line (increasing) represents the probability of being five minutes late compared to the current timetable.

roughly five seconds *on average*. Figure 11 depicts the impact on the on-time probabilities when both λ_1 and λ_2 are divided by y for Patch 8. Values of y that are close to $\frac{5}{4}$ would correspond to the expected sojourn time in Patch 8 increasing by five seconds. In this setting, the total on-time probability increases.

5.4 Impact of a decrease in the number of bus passengers

In Section 3.3, we noticed that the presence of bus stops in a patch had a much more profound effect on the sojourn distribution in a patch than the presence of a traffic light. This effect is due to the embarking and disembarking of passengers. The number of passengers at a stop may well decrease when they have the choice to travel by tram instead. The bus stops most likely to be affected by the tram stop in Patch 8 are the ones in Patches 6 and 10. In an extreme case, these bus stops would stop being used at all, which would mean that the sojourn time distributions in Patches 6 and 10 would become similar to the current sojourn time distribution of Patch 8. While this is unlikely to happen in practice, it can be helpful to consider this as a worst-case scenario; furthermore, we can use PRISM to get a good idea about the intermediate scenarios.

In the current setting, the expected sojourn time in Patch 8 is 19.46 seconds, while those in Patches 6 and 10 are 45.49 and 44.89 seconds respectively. To be able to calculate the impact of the change in sojourn time distributions for the intermediate scenarios, we introduce the variable z which is such that if $z = 0$, the system behaves as it does currently, while if $z = 1$, the expected sojourn time in Patches 6 and 10 is the same as in Patch 8. To make this concrete, we set λ_1 and λ_2 of Patch 6 equal to

$$\lambda_1 \cdot \left(1 - z \cdot \left(1 - \frac{45.59}{19.46}\right)\right) \quad \text{and} \quad \lambda_2 \cdot \left(1 - z \cdot \left(1 - \frac{45.59}{19.46}\right)\right)$$

respectively. The impact of this for a range values of z between 0 and 1 is given in

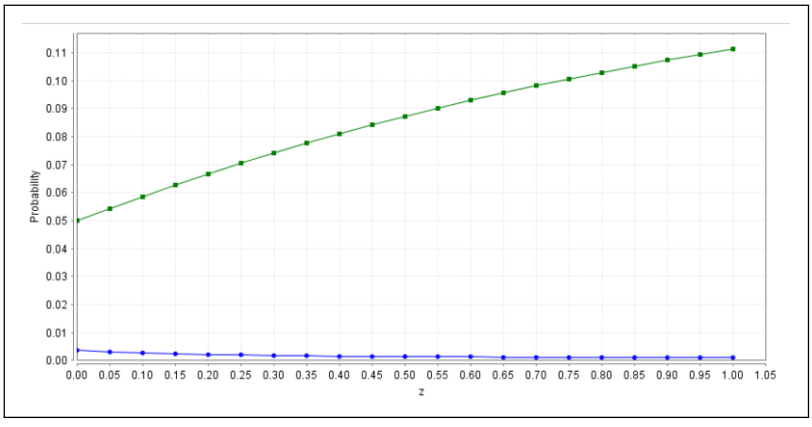


Fig. 12. Probability of not being on time as a function of z, where z is such that if $z = 0$, the system behaves as it does currently, while if $z = 1$, the bus stops in Patches 6 and 10 are never used. The green (increasing) line represents the probability of being a minute early compared to the current timetable. The blue line (decreasing) represents the probability of being five minutes late compared to the current timetable.

Figure 12. The buses were already more likely to be early than being five minutes late on such a short piece of the route, and this is made more extreme due to buses spending less time in the bus stop patches.

6 Related work

In this section we give a short overview of the state of the art in the field of using AVL data for the analysis of public transport performance. We mention a number of important publications and briefly summarise their contents.

The authors of [5] study the problem of adjusting bus stop timetables such that the percentage of 'on-time' bus arrivals is maximised, assuming that changes in the timetables do not affect the behaviour of the buses. In the paper, a bus is assumed to be 'on-time' if its arrival occurs between X and Y minutes after the scheduled time. The authors assume that the differences between scheduled and actual arrivals of buses to a stop are normally distributed, and based on this (or at least the normal distribution's symmetry) argue that it is optimal to choose the scheduled time of arrival such that the expected time of arrival is $(X+Y)/2$ after the scheduled time. When the on-time interval is $(-2, 5)$, this basically means that buses are scheduled to be 1.5 minutes late on average, which agrees with our observation in Section 5 that slowing buses down can increase average on-time performance. Empirical bus stop arrival times are obtained using AVL. The paper contains a case study with data obtained from Miami-Dade Transit (MDT), the main public transport operator in the area around Miami, Florida.

The authors of [17] study the problem of using low-frequency (in their case, once each 60 seconds) AVL data to estimate the difference between the times of two subsequent arrivals of a bus of a certain route at a bus stop, called the 'headway'. To determine when a bus is at a stop, the 2-dimensional GPS locations are first projected onto a one-dimensional axis (this can be done without problems for 97.6% of all AVL data entries). Since the locations of the bus are only known at snapshots, the locations of the bus between the snapshots are interpolated. Instead of linear interpolation, the authors propose a method that includes a calibration phase (with a corresponding calibration dataset). In the calibration phase, the route is divided into 10-metre zones and the relative proportion of time in the dataset's time period that is spent in each of the 10-metre zones is calculated. The interpolation of all future bus movements is then chosen to match this distribution. This approach could be used to improve on the linear interpolation that we describe in Section 2.1. After determining the bus arrival times using the interpolation, the empirical distribution of the headway at several bus stops is displayed and fitted to the three-parameter gamma distribution discussed in Section 3.1. The authors measure the goodness-of-fit using the Kolmogorov-Smirnov test statistic. A case study is performed using data for a single bus route in Boston.

The authors of [15] propose a model, based on a Kalman filter, that predicts bus arrival and departure times at stops. Particularly, their model separates bus running times from dwell times, and incorporates their interplay. Such interplay

typically leads to alternating late and early buses: if a bus is late, it has to pick up passengers intended for the next bus, which means that the late bus gets further behind schedule and the bus after starts to run early because it has to pick up fewer passengers. This phenomenon is also mentioned in [4]. In our paper, we focus on a single bus so this behaviour is not captured. In [15], a case study is performed involving data provided by the Toronto Transit Commission.

Finally, we mention [8], a comprehensive technical report detailing the state of the art regarding the use of data from AVL systems or Automatic Passenger Counters (APC) to improve transit performance in 2006. Subjects discussed in the report range from a survey of the potential uses of AVL data to practical considerations. A number of case studies are discussed, namely Seattle, Portland, Chicago, New Jersey, and Minneapolis in the USA, Ottawa and Montreal in Canada and The Hague and Eindhoven in the Netherlands. The uses for AVL-APC data discussed by the authors include: analysis of running time, schedule adherence and headway regularity, demand analysis and bus route mapping.

7 Conclusions and future work

In this paper, we have proposed a methodology for constructing PRISM models for bus movements in which the parameters are inferred from historical data through the tool HyperStar. The PRISM model represents the movement of a bus through a sequence of patches, and by altering the parameters in patches to resemble those in other patches we can investigate the impact of external changes. We have demonstrated our methodology by using it to investigate the effect of a real-world case study involving the introduction of trams to the City of Edinburgh. The amount of effort needed to carry out our approach — mainly determined by the parameter fitting and the CTMC analysis — scales linearly with the number of patches.

Development of the methodology is ongoing with specific focus on several extensions. The first is the generalisation of the model to one that incorporates the movement of several buses that contend for shared resources (in this case, access to bus stops). The structure of the PRISM model is such that contention between buses is trivially added. The main challenge is the parameterisation: since the empirical sojourn times already include the effect of contention, the effect of adding a bus to the system is hard to interpret. If we are able to match buses to their routes, we can investigate the difference in sojourn time behaviour between buses of routes that do and that do not stop at a bus stop in a patch. This difference can then be incorporated in the parameters. Another possibility is to partition the sojourn time dataset according to how many other buses are present when a bus enters a patch. If buses that enter patches that already contain other buses have longer sojourn times on average, then this difference can be incorporated into the parameters.

Another research direction is an automated procedure for dividing a portion of a route into patches. Currently, the patches are hand-picked. Thanks to our cooperation with the City of Edinburgh council, we have access to data about the routes that include information about bus stops. The main challenge is to

incorporate information about junctions and traffic lights, which are currently the main criteria on which we base the patch structure. Hence, a different set of criteria for the patch structure may be required.

Acknowledgement

This work is supported by the EU project *QUANTICOL*, 600708. The authors thank Bill Johnston of Lothian Buses and Stuart Lowrie of the City of Edinburgh council for providing access to the data which was used to parameterise the model in the paper. The visualisation tool used to create representations of the bus data on a map view was originated by Shao Yuan during his MSc work at Edinburgh [18].

References

[1] Rajeev Alur and Thomas A Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[2] A. Aziz, K. Sanwal, V. Singhal, and R.K. Brayton. Verifying continuous-time Markov chains. *Lecture Notes in Computer Science*, 1102:269–276, 1996.

[3] C. Baier, B. R. Haverkort, H. Hermanns, and J. P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Computer Aided Verification*, volume 1855, pages 358–372. LNCS Volume 1855, Springer, 2000.

[4] Mathew Berkow, John Chee, Robert L Bertini, and Christopher Monsere. Transit performance measurement and arterial travel time estimation using archived AVL data. *ITE District 6 Annual Meeting*, 2007.

[5] Fabian Cevallos, Xiaobo Wang, Zhenmin Chen, and Albert Gan. Using AVL data to improve transit on-time performance. *Journal of Public Transportation*, 14(3):21–40, 2011.

[6] Alastair Dalton. Edinburgh tram tested on Princes Street. *The Scotsman*, 20 February 2014. Web.

[7] Lothian Buses faced fine for bad service. *Edinburgh Evening News*, 21 June 2010. Web.

[8] Peter G. Furth, Brendon Hemily, Theo H.J. Muller, and James G. Strathman. *Using archived AVL-APC data to improve transit performance and management*, volume 113. Transportation Research Board, 2006.

[9] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification*, pages 585–591. Springer, 2011.

[10] James L. Pline. *Traffic engineering handbook*. Prentice-Hall, 1992.

[11] `http://www.prismmodelchecker.org/manual`.

[12] Philipp Reinecke, Tilman Krauß, and Katinka Wolter. Cluster-based fitting of phase-type distributions to empirical data. *Computers & Mathematics with Applications*, 64(12):3840–3851, 2012.

[13] Philipp Reinecke, Tilman Krauß, and Katinka Wolter. HyperStar: Phase-type fitting made easy. In *Proceedings of the Ninth International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 201–202, 2012.

[14] The Scottish government. Bus Punctuality Improvement Partnerships (BPIPs) guidance. 2009.

[15] Amer Shalaby and Ali Farhan. Prediction model of bus arrival and departure times using AVL and APC data. *Journal of Public Transportation*, 7(1):41–62, 2004.

[16] Plan to boost public transport would let buses beat red lights. *STV News*, 21 January 2014. Web.

[17] Yingxiang Yang, David Gerstle, Peter Widhalm, Dietmar Bauer, and Marta Gonzalez. The potential of low-frequency AVL data for the monitoring and control of bus performance. In *Transportation Research Board 92nd Annual Meeting Compendium of Papers*, 2013.

[18] Shao Yuan. Simulating Edinburgh buses. Master's thesis, The University of Edinburgh, 2013.