A real-time particle filter for bus arrival prediction with discrete CDFs for journey planning

Tom Elliott
University of Auckland, Auckland, New Zealand
E-mail: tom.elliott@auckland.ac.nz
Thomas Lumley
University of Auckland, Auckland, New Zealand
E-mail: t.lumley@auckland.ac.nz

Summary. The prediction of transit vehicle arrival time is a complex problem with the added difficulty of its real-time nature. Reliable predictions require the combination of both vehicle and network states, implying that real-time traffic congestion information is incorporated into the predictions. Inevitably, however, there is a lot of uncertainty around arrival times: road speeds fluctuate, particularly around peak time, and multimodality is added with every stop passed (which may or may not cause the bus to stop depending on passenger demand). Therefore, not only do we need to predict arrival time, but also assess and report the uncertainties involved. We present a method of using a particle filter to combine the states of both vehicle and network to obtain arrival time distributions. These are then discretised in such a way that allows real-time calculation of event probabilities for use in journey planning applications.

Keywords: particle filter, transport, real-time, GTFS, transit networks, journey planning

1. Introduction

- the problem of predicting bus arrival - it needs to use real-time traffic information Citation Needed (2020)
- however, many deployed methods are either specific to a provider/city, or don't make use of real-time data (only vehicle position and/or arrival delays, e.g., in Auckland)
- since the only logical source of "traffic data" in this setting is the transit vehicles themselves, makes sense to develop framework that uses them to estimate real-time network state Elliott and Lumley (2020)

- in this example, particle filter is used to obtain a sample of points from the arrival time distribution - many many points, which cannot possibly be distributed or stored efficiently
- we propose a method of reducing this to a simple discrete CDF of arrival time (in minutes)
- we show that these results can be used to answer some common journey planning questions, in real-time, which outperform current "method"

Some of the references include:

- Cathey and Dailey (2003)
- Yu et al. (2006, 2010, 2011)
- Hans et al. (2015)

2. Background

Before describing the process of obtaining arrival time distributions, we must first define the framework with which we obtain vehicle and network state estimates. Elliott and Lumley (2020) present a process for constructing a transit road network from raw GTFS data.

most of this section will be removed/cut-down to just the basic concepts

2.1. GTFS static and real-times

The structure of public transport systems is very similar across the globe, so much so that Google developed GTFS (Google Developers, 2006) to standardise the way transport agencies organise and distribute transit data. GTFS consists of two core components: *static*, which contains pre-defined information, such as routes and stops, and *real-time*, which contains vehicle location and other data collected in real-time.

Static GTFS contains *structural* data, descibing information one might observe on a printed timetable. This includes route information about trips (instances of a route at a specific time), the geographical path the vehicle takes, and the stops it services. This provides the basis of transit data, and plays a key role in developing the aforementioned *transit network*.

Real-time GTFS descibes how live data, such as the location of a bus, or its time of arrival at or departure from a bus stop, is formatted. Vehicle

locations allow users to track the progress of the vehicle along its route, and are often updated every 10–30 seconds. In some implementations, such as in Auckland, they are also triggered by events such as arrival at a bus stop or (major) intersection. Arrival and departure events, on the other hand, are triggered whenever the bus arrives at or departs from a bus stop (or at least thinks it does), and include the time of the event and, if scheduled arrival times are available, the *delay* between scheduled and actual arrival (or departure).

A huge benefit of GTFS being adopted widely around the world is that developers can easily access transit feeds for different providers in different cities and countries using the same structure. Any application developed using (only) GTFS information can, technically, be redeployed to any other system also using GTFS. For this reason, we opted to develop our application framework based solely on GTFS data, as described next.

2.2. A transit road network

One important predictor of bus arrival is travel time (Shalaby and Farhan, 2004). However, previous research has focused on situations where either only a single route is considered, or else different routes and combined by manually locating common road segments. This is, of course, not easy or even possible to automate. Elliott and Lumley (2020) proposed a method of developing a transit road network from only GTFS data by identifying *nodes* (bus stops and intersections) and the *edges* (roads) between them.

Here, we use the simplest version of this method, which places nodes at bus stops. Thus, any two routes that share the same sub-sequence of bus stops are assumed to follow the same path between them (this is almost always the case). By expressing the route as a series of nodes and road segments, we can begin to share information across routes.

The goal is to model vehicles as they travel along their respective routes, estimating their average speed along each road segment. Having traversed a road segment, the data is passed to the network to update the real-time traffic state. This can in turn be used, optionally with a forecasting function, to predict a future travel time of a bus along the road, which is used in arrival time prediction.

2.3. Particle filters estimating vehicle state

Estimation of vehicle state in real-time commonly uses *recursive Bayesian estimation*, in which the posterior becomes the prior for the next iteration.

Particle filters are a powerful method of estimating successive states, as they make few assumptions and, most importantly, cover a large range of possible trajectories, even when there is a high degree of multimodality as is the case with transit data (e.g., around bus stops). Here, I give a brief overview of the particle filter which was used by Elliott and Lumley (2020) to estimate vehicle state.

In a recurisive Bayesian model, the underlying state we are interested in is denoted $\boldsymbol{x} = \boldsymbol{x}_{0:k} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k]$. It can rarely be observed directly, and instead we obtain measurements $\boldsymbol{y} = \boldsymbol{y}_{1:k} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k]$. In a particle filter (Gordon et al., 1993), this state is represented by a weighted sample of *points*, and makes use of the delta measure δ,

$$p(\boldsymbol{x}_{k-1}|\boldsymbol{y}_{0:k-1}) \approx \sum_{i=1}^{N} w_{k-1}^{(i)} \delta_{\boldsymbol{x}_{k-1}^{(i)}}(\boldsymbol{x}_{k-1}) .$$

Elliott and Lumley (2020) describe the particle filter in detail, with discussion of its use for modelling transit vehicles in real-time. However, here we only discuss the merits of the particle filter, and need no more details than given already.

The main advantage of the particle filter is, as mentioned, its ability to cope with a wide range of plausible, discontinuous trajectories. This often happens in transit data as the bus passes bus stops at which it may or may not stop. It's main downside is the computational demand, since we need 1000s of particles in memory for each vehicle, and perform transitions and updates to each of them. However, Elliott and Lumley (2020) have shown that, so long as it is programmed efficiently, the particle filter can run fast enough to be feasible in a real-time setting, at least for updating vehicle states.

In the next section, we make use of the flexible nature of the particle filter in that each particle is transitioned independently, making it possible to account for a wide variety of complex behaviours. These most notably include bus stops, which introduce a lot of uncertainty, but also layovers, which can result in a wide temporal gap between possible arrival times.

The particle filter allows us to estimate the state of a vehicle and, most importantly, it's average speed along road segments:

$$p(b_\ell|\boldsymbol{y}_{1:k}) \approx \sum_{i=1}^{N} w_k^{(i)} \delta_{b_\ell^{(i)}}(b_\ell) .$$

2.4. Estimating network state

Having obtained a estimate of the speed of each vehicle along a road, we can proceed to updating the state of the road network itself. This uses a Kalman filter to update the state, $\boldsymbol{\beta}_c$, using the estimated vehicle speeds.

This all provides a flexible method for modelling real-time traffic conditions in any network using only GTFS, and not rely on other data sources (for example taxis or video cameras).

We use a hierarchical model on vehicle travel times, as individual vehicles have different average speeds, even if they travel at the same time, due to different driver behavior and other factors. The between-vehicle noise parameter $\boldsymbol{\psi}_\ell$ describes this, and is estimated from historical data.

The observations are also uncertain, as they are estimated from the particle filter. We can assume between-vehicle and measurement uncertainties are independent, so the total variance of observations is

$$\text{Var}\left[b_\ell | \boldsymbol{y}_{1:k}\right] = e_\ell^2 + \boldsymbol{\psi}_\ell^2.$$

The system noise q is also estimated from historical data. Together we can quickly update the network state using the information variance of the Kalman filter (since this allows data from multiple sources simultaneously, which can happen when two or more buses are travelling one behind the other).

The result is an estimate of the state of the network at time t_c,

$$p(\beta_c | \cup_m \cup_\ell b_{\ell,c,m}) \sim \mathcal{N}\left(\boldsymbol{\beta}_\ell, \mathbf{P}^2\right).$$

This can be used to predict the travel time of buses.

If a forecast function is available, we could also forecast future travel time based on current state and historical trends, and this would be incorporated.

3. Predicting arrival time

Accurate, reliable prediction of bus arrival time requires knowledge of both vehicle and network states, represented by $\boldsymbol{x}_k$ and β_c, respectively. Since there is a high degree in the uncertainty of these, particularly when forecasting future network states around peak times, it is crucial to incorporate this uncertainty into the arrival time distribution and, ultimately, decision making processes.

The particle filter presents a robust method of sampling all of the possible trajectories the vehicle might take (Hans et al., 2015).

- initial state incoporates uncertainty (shape, multimodality) of vehicle state
- trajectory of each particle = one possible path the bus might take
- accounts for uncertainty in (forecasted) road speed, correlations, etc
- result makes no assumptions about shape of distribution

The main downside of the particle filter is the computational demand of it, often requiring 5000-10000 particles per bus.

- Elliott and Lumley (2020) showed the pf is feasible in real-time for modelling
- need to also show predicions can be done quickly and usefully
- reduce number of particles where possible

Our implementation is described in **??**. To examine the effectiveness of our approach, we run our method on a full day and compared the predictions with the actual arrivals, as well as the currently used "GTFS" predictions. These are discussed in **??**.

3.1. Particle filter ETAs

In our application, the vehicle state is already represented by a set of N particles,

$$p(\boldsymbol{x}_{k|k} \,|\, \boldsymbol{y}_k) \approx \sum_{i=1}^{N} w_k^{(i)} \delta_{\boldsymbol{x}_k^{(i)}}(\boldsymbol{x}_k)$$

which we use directly. However, were vehicle states available in some other form, one would simply take a sample from the posterior state estimate $p(\boldsymbol{x}_{k|k} \,|\, \boldsymbol{y}_k)$. This gives us a sample of plausible bus states for which we can predict individual arrival times at upcoming stops.

As with the vehicle model described by Elliott and Lumley (2020), we can iteratively forecast each particle's arrival at all upcoming stops. This involves incorporating network state (vehicle speeds along roads) as well as bus stopping behaviour. These two aspects are described individually below, and each particle simply iterates between them until it reaches the end of the route.

To begin, we define the state of an active trip at time t_k in which the vehicle is at or last visited stop s_k, together with an indicator d_k of whether or not the bus has departed the stop, the index of the current road segment ℓ_k, and the progress along that segment $\hat{p}_k$:

$$\mathcal{T}_k = [t_k, s_k, d_k, \ell_k, \hat{p}_k]^\top.$$

Next, we approximate the distribution of trip state by generating a sample of particles from $p(\mathcal{T}_k|\boldsymbol{y}_{1:k})$. In our case, we already have a weighted sample of particles, but the sample could be taken from any posterior distribution, for example if vehicle state is estimated using a Kalman filter the particles can be sampled from the Gaussian state estimate.

We wish to take a sample of size $\tilde{N} \leq N$, which can depend on the number of remaining stops and the level of accuracy desired. However, in the next section we approximate the predictive distributions using a discrete CDF, so accuracy only needs to be at about the 30 second level. Thus, we pull sample with replacement from $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k})$, yielding the trip state estimate

$$p(\mathcal{T}_k|\boldsymbol{y}_{1:k}) \approx \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \delta_{\mathcal{T}_k^{(i)}}(\mathcal{T}_k)$$

where the components of $\mathcal{T}_k^{(i)}$ are computed from the vehicle state particles. This doesn't make sense. Gosh darnit.

Assuming I can rewrite the above to make sense ...

Now we iterate each trip particle to the end of the route, storing the arrival times at all stops along the way. This is a five-step process.

Note: need to add current speed to trip state ???

Step 1: complete the current segment

Each particle has completed $100p_\ell^{(i)}\%$ of the segment. If the vehicle is nearing the end of the segment (less than 200 meters remaining), we keep the particle's initial speed $\dot{x}_k^{(i)}$. Otherwise, we simulate a speed from the network state with mean β_ℓ, uncertainty $\mathbf{P}_\ell$, and between-vehicle variability $\boldsymbol{\psi}_\ell$, as follows:

$$v_\ell{}^{(i)\star} = \begin{cases} \dot{x}_k^{(i)} & (1-\hat{p}_k)\mathcal{L}_\ell < 200, \\ v_\ell{}^{(i)} \sim \mathcal{N}_T\left(\beta_\ell,\, \zeta_\ell^2 + \boldsymbol{\psi}_\ell^2,\, 0,\, \mathbb{V}_\ell\right), & \text{otherwise.} \end{cases}$$

Now the travel time to the end of the current segment can simply be obtained as

$$\tilde{z}^{(i)} = \frac{(1-\hat{p}_k^{(i)})\mathcal{L}_\ell}{\dot{x}_\ell{}^{(i)\star}}.$$

We now store an iterative variable *travel-time-so-far* with the time taken to reach the end of the current segment, $\eta^{(i)} = \tilde{z}^{(i)}$.

Step 2: compute arrival time at next stop

If there are any remaining segments between the current segment $\ell_k^{(i)}$ and the next stop $j_k^{(i)}+1$, their travel times are sampled from the network state with variance increased by network system noise to account for forecasting changes.

$$v_j^{(i)} \sim \mathcal{N}_T\left(\hat{\beta}_j,\, \left[(\zeta_j + \eta^{(i)}q)^2 + \boldsymbol{\psi}_j^2\right] \wedge P_{\max},\, 0,\, \mathbb{V}_j\right),$$

allowing uncertainty to increase up to a maximum $P_{\max}$ which is the overall variance of all speeds along all segments estimated from historical data. The travel time along each segment is added to the travel-time-so-far,

$$\eta^{(i)} = \eta^{(i)} + \frac{\mathcal{L}_j}{v_j^{(i)}}.$$

Now, arrival time at stop j is the sum of travel time along all segments up to the next stop. In the iterative process, this is simply $\alpha_j^{(i)} = t_k + \eta^{(i)}$.

Step 3: compute stop dwell time

Once at a stop, a bus either stops and waits while passengers board and disembark, or travels past without stopping. This *dwell time* needs to be modelled (Citation Needed, 2020), but adds multimodality to the arrival time distribution.

The dwell time model is ... [from chapter 3], as used by Hans et al. (2015); Elliott and Lumley (2020). This now gets added to the travel-time-so-far,

$$\eta^{(i)} = \eta^{(i)} + d_j^{(i)}$$

unless the stop is a layover. In that case, we allow the particle—with some probability—to remain at the stop until the end of the dwell time, or the scheduled departure time, whichever is later.

Step 4: repeat steps 2–3

We now repeat each of the steps, sampling travel times for intermediary road segments as the particle arrives, and storing stop arrival times, until the particle reaches the end of the route.

Step 5: obtain arrival time distributions

Since each particle stores its arrival at all stops, we can very easily obtain the predictive distribution of arrival time for the bus using the delta measure,

$$p(\alpha_j|\boldsymbol{x}_k, \boldsymbol{\beta}_k) \approx \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} \delta_{\alpha_j^{(i)}}(\alpha_j)\,.$$

Table 1. Results for particle filter and other.

	RMSE (s)	MAE (s)	MAPE (%)	PICP (%)
Particle filter	232	146	19	77
Schedule-delay	238	164	27	

We may obtain any summary statistics from this distribution, including the mean and prediction intervals.

3.2. Results

To assess the feasibility and reliability of this approach, we ran the full model on one day of data from DATE. This used the particle filter and network model described by Elliott and Lumley (2020), and the arrival time predictions described above. At the end of the day, we could compare the predicted with the actual arrival times. In addition, we compare to the GTFS-based predictions which are simply the scheduled arrival time plus the delay at the most recent stop.

To assess, we use RMSE, MAE, MAPE, and PICP. RMSE and MAE provide an assessment of the absolute prediction reliability, while MAPE scales predictions by the time-until-arrival; that is, it is more affected by short-term predictions when the bus is near, giving us a comparative measure of short-term reliability (which is useful when deciding between walking, jogging, or sprinting to the bus stop). PICP (which is only available for the particle filter estimates) assess how well the uncertainty is captured (we would expect a 95% prediction interval to capture the true value in 95% of cases).

The overall values are shown in table 1. The particle filter predictions are slightly more accurate than GTFS on average, and have about 20% error versus GTFS's 27%. However, only 77% of prediction intervals contain the actual arrival time, indicating that uncertainties are being under-predicted (the nominal coverage is 85%).

4. A discretized CDF approximation for journey planning

One of the main problems with particle filter results is that we have a large sample of points, rather than the description of a posterior distribution (as would be the case with a Kalman filter, for example). Trying to do any real-time processing, for example computing event probabilities, would be very computationally intensive, and probably not feasible on a user's

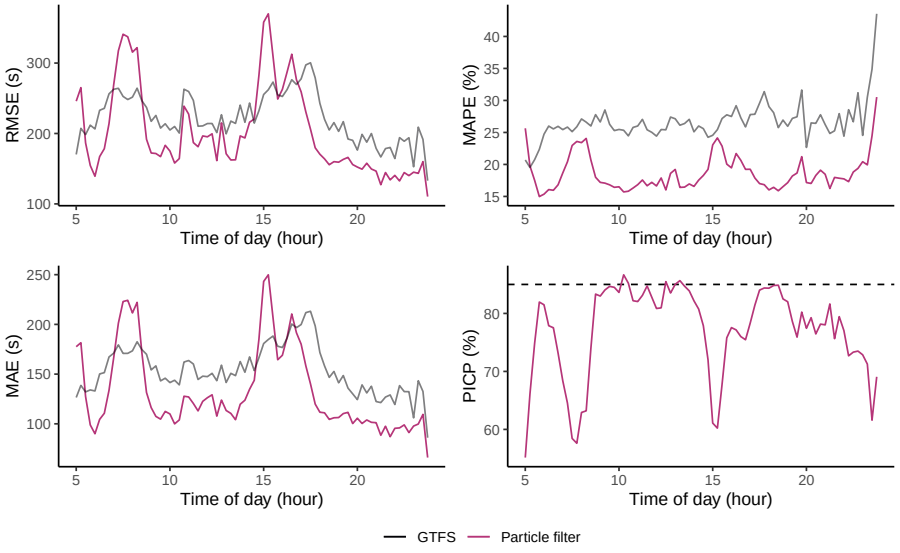

Fig. 1. Results by time of day.

phone, which would require the server to not only generate the arrival time distributions, but to process individual journey planning queries too.

To consider a simpler approximation, we start by noting that arrival times are generally displayed to the user in one-minute accuracy. Therefore, we can consider the concept of discretizing arrival times at the one-minute level and obtaining a CDF which for the probability that the bus arrives within a minutes.

Computing a CDF approximation using particles is in fact a trivial exercise, and can be done in real-time without any negative affect on performance. This is because the method described below *does not* require sorting of the particles, which we would need to do to compute, for example, quantiles (median, or a prediction interval). This essentially consists of counting the number of particles in each one-minute interval and expressing it as a probability. Here, the particle arrival time $\alpha^{(i)}$ is in seconds, is transformed to minutes and rounded down using the floor operator, while the discrete CDF is reported in a minutes.

$$\mathbb{P}\left(A \in [a, a+1]\right) = \frac{1}{\tilde{N}} \sum_{i=1}^{\tilde{N}} I\left\{ \lfloor \frac{\alpha^{(i)}}{60} \rfloor = a \right\}, \tag{1}$$

where the indicator $I\{a = b\}$ is 1 if a equals b, and zero otherwise.

Using eq. (1) the CDF for bus arrival at a given stop is

$$P(A < a) = \sum_{x=0}^{x=a-1} \mathbb{P}\left(A \in [x, x+1]\right),$$

and is demonsrated graphically in fig. 2.

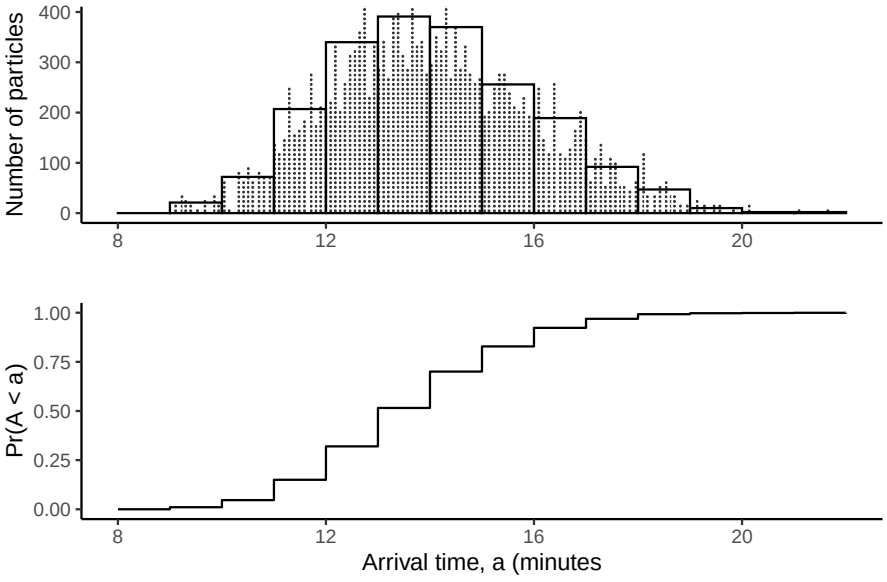

Fig. 2. Particle filter estimates of arrival time and the process to obtain a discretized CDF approximation.

The CDF can easily be distributed to users' phones, since the data structure is simple and consists of only a few values. Now we assess the validity of the simplification by predicting event probabilities and comparing those to the truth, as well as the GTFS binary estimate of the event outcomes.

Of course, we can obtain simple summary statistics for the distribution, such as the median or other quantiles. For a quantile $q \in (0, 1)$, we calculate

$$\hat{A}_q = \max\{a \in \{0, 1, \ldots\} : \mathbb{P}\left(A < a\right) \leq q\}$$

which provides the smallest arrival time, in minutes, with a probability of no more than q of the bus arriving before that time.

Another question, however, is to compute probabilities of events. The simplest is, of course, the probability that the bus arrives before or after a specific time, $\mathbb{P}\left(A < a\right)$ or $\mathbb{P}\left(A \geq a\right)$. This is useful for answer questions such as

- what is the probability of catching the bus if I arrive by 9:30am?
- what is the probability the bus will arrive before 1pm for my meeting?

Calculating these events is straighforward given the CDF, and can be performed on a user's phone or quickly on the server. We performed a journey planning exercise to assess the reliability of these results by computing the probablity of a bus arriving before a specific time and assessing the outcome. Due to restrains on the current implementation, we only have arrival time distributions for buses which have begun the route. The journey planning problem is this: *given two alternative bus stop options requiring walking in opposite directions, which is the best to take?* We account for the walking time W_1 and W_2 to both stops R_1 and R_2, respectively, and compute the probabilities of catching the next few buses. The probability of catching R_j, $j = 1, 2$, if leaving now at time T, is given by

$$\mathbb{P}\left(\text{catch}|\text{choose } R_j\right) = \mathbb{P}\left(A_j \geq T + W_j\right) = 1 - \mathbb{P}\left(A_j < T + W_j\right)$$

which is easily obtained from the CDF. To simulate, we chose a sequence of times from 9 am til 3 pm in 15 minute intervals, and for each calculated the probability of catching upcoming buses at the two alternative stops. This was compared to the outcome, and the GTFS binary prediction. The results are shown in fig. 3.

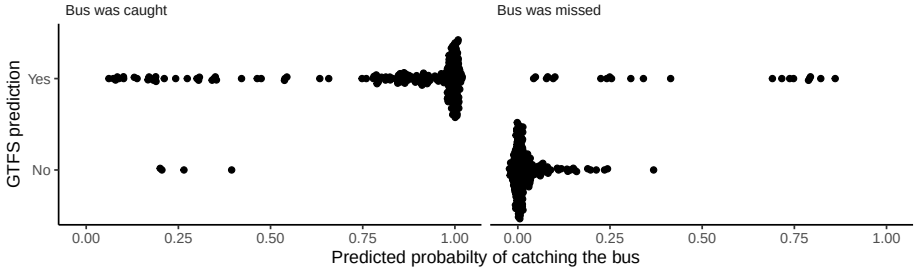

Fig. 3. Predicted outcomes of journey planning process.

Discussion of results!!

- we can adjust decision to fit the situation - coffee with friend vs job interview
- doesn't account for *wait time* (i.e., if you miss the bus, how long till the next one)

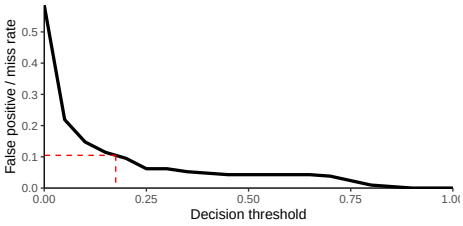

Fig. 4. False positive rate for varying decision level. Increasing the threshold for predicting a positive outcome reduces the *miss rate*, at the cost of not trying to catch potential buses.

Table 2. Prediction results for all journeys, with the displayed values representing the proportion of outcomes in each cell (rows are conditioned by the prediction).

	Observed outcome					
	Schedule-delay		$P > 0.5$		$P > 0.8$	
Predicted outcome	No	Yes	No	Yes	No	Yes
No	0.98	0.02	0.86	0.14	0.81	0.19
Yes	0.09	0.91	0.05	0.95	0.01	0.99

The second journey planning scenario is *arriving before a specified time*, which requires forecasting further into the future. Such a decision might be important for someone, allowing them to either inform their friend they'll be late for that coffee, or catching a taxi to get to the interview. As before, we selected a stop and this time asked if the bus would arrive on-time (where on time was 15–45 minutes in the future). The results, as before, are displayed in, where the successfull event is now that the bus arrived before the specified time A.

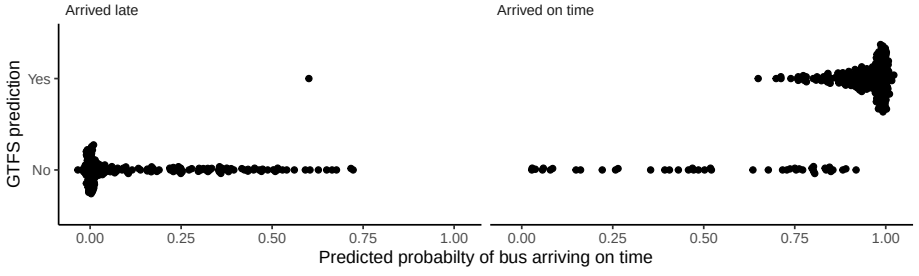

Fig. 5. Predicted outcomes of journey planning process - arrive on time.

The final scenario is for a journey which requires transfering between two

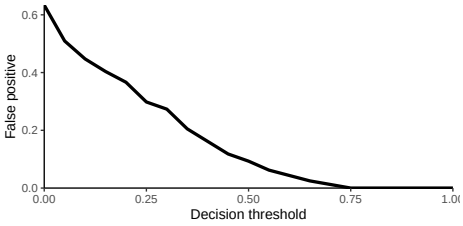

Fig. 6. False positive rate for varying decision level. Increasing the threshold for predicting a positive outcome reduces the arrival late rate, at the cost of not trying to catch potential buses.

Table 3. Prediction results for all journeys 2, with the displayed values representing the proportion of outcomes in each cell (rows are conditioned by the prediction).

	Observed outcome					
	Schedule-delay		P > 0.5		P > 0.8	
Predicted outcome	No	Yes	No	Yes	No	Yes
No	0.78	0.22	0.87	0.13	0.77	0.23
Yes	0.00	1.00	0.06	0.94	0.00	1.00

services, in which case we want to predict the probability that a transfer between two particular trips at the transfer stop will be successful (that is, the first bus arrives before the second). Given two CDFs for the arrival time of the first and second trips, $\mathbb{P}(L_1 < x)$ and $\mathbb{P}(L2 < y)$, respectively, the probability of a successfull transfer between them is

$$\begin{aligned}
\mathbb{P}_{\text{transfer}} = \mathbb{P}(L_1 < L_2) &= \sum_{x=1}^{\infty} \mathbb{P}(L_1 < L_2 | L_2 = x)\, \mathbb{P}(L_2 = x) \\
&= \sum_{x=1}^{\infty} \mathbb{P}(L_1 < x)\, \mathbb{P}(L_2 = x) \\
&= \sum_{x=1}^{\infty} \mathbb{P}(L_1 < x) \left[\mathbb{P}(L_2 < x+1) - \mathbb{P}(L_2 < x)\right].
\end{aligned}$$

This is easily obtained from the CDFs and is a simple enough computation which could be done on a user's device.

- PF methods allows decision making to include risk
- GTFS is a fixed rate; cannot adapt to specifications of the journey planning application (i.e., when it's pouring with rain, want high chance of successful transfer)

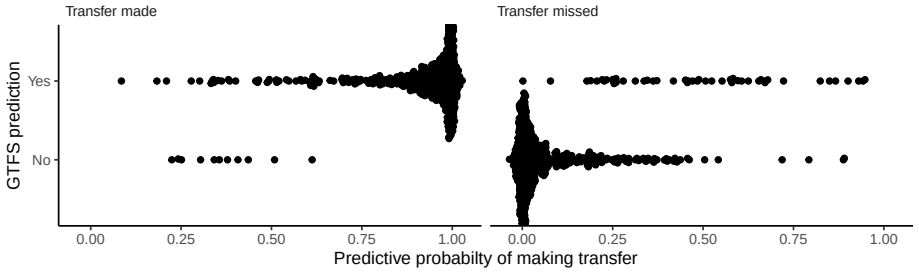

Fig. 7. Predicted outcomes of journey planning process for a transfer between services.

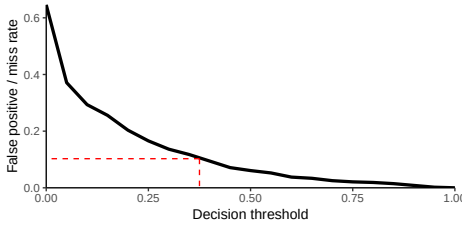

Fig. 8. False positive rate for varying decision level of transfer. Increasing the threshold for ... . The GTFS rate is fixed at about 10%, which corresponds to using a 37.5% decision threshold.

Table 4. Prediction results for all transfer journeys, with the displayed values representing the proportion of outcomes in each cell (rows are conditioned by the prediction).

	Observed outcome					
	Schedule-delay		P > 0.5		P > 0.8	
Predicted outcome	No	Yes	No	Yes	No	Yes
No	0.97	0.03	0.94	0.06	0.85	0.15
Yes	0.10	0.90	0.07	0.93	0.03	0.97

- Of course, the wait time is also important (i.e., how long waiting for the transfer?) which can be calculated as well... ?

$$\begin{aligned}\mathbb{E}_{\text{wait}} &= \mathbb{E}\left[L_2 - L_1 \,\middle|\, L_1 < L_2\right] \\ &= \sum_{y=1}^{\infty}\sum_{x=0}^{y-1}(y-x)\,\mathbb{P}\left(L_1 = x \cap L_2 = y\right) \\ &= \sum_{y=1}^{\infty}\sum_{x=0}^{y-1}(y-x)\,\mathbb{P}\left(L_1 = x\right)\mathbb{P}\left(L_2 = y\right)\end{aligned}$$

where $\mathbb{P}(A = a) = \mathbb{P}(A < a + 1) - \mathbb{P}(A < a)$. This lets us calculate the expected waiting time given the transfer is successful, and accounts for uncertainties in both vehicle's arrival times. A similar formula could be used to calcualte, for example, the variance of waiting time—again, all on a user's phone, which removes compuational demand from the server.

5. Discussion

- what this means
- how this makes JP more accessible

5.1. Future Work

- automated route selection
- improved particle filter
- improved network construction
- improved network state forecasts

5.2. Conclusion

- simple conclusion of the paper

References

Cathey, F. W. and Dailey, D. J. (2003) A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transportation Research Part C: Emerging Technologies*, **11**, 241–264.

Citation Needed (2020) *Missing citations.*

Elliott, T. and Lumley, T. (2020) Modelling the travel time of transit vehicles in real-time through a GTFS-based road network using GPS vehicle locations. *Australian & New Zealand Journal of Statistics*, **62**.

Google Developers (2006) What is GTFS? https://developers.google.com/transit/gtfs/.

Gordon, N. J., Salmond, D. J. and Smith, A. F. M. (1993) Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F Radar and Signal Processing*, **140**, 107–113.

Hans, E., Chiabaut, N., Leclercq, L. and Bertini, R. L. (2015) Real-time bus route state forecasting using particle filter and mesoscopic modeling. *Transportation Research Part C: Emerging Technologies*, **61**, 121–140.

Shalaby, A. and Farhan, A. (2004) Prediction model of bus arrival and departure times using AVL and APC data. *Journal of Public Transportation*, **7**, 41–61.

Yu, B., Lam, W. H. K. and Tam, M. L. (2011) Bus arrival time prediction at bus stop with multiple routes. *Transportation Research Part C: Emerging Technologies*, **19**, 1157–1170.

Yu, B., Yang, Z.-Z., Chen, K. and Yu, B. (2010) Hybrid model for prediction of bus arrival times at next station. *Journal of Advanced Transportation*, **44**, 193–204.

Yu, B., Yang, Z.-Z. and Yao, B. (2006) Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems*, **10**, 151–158.