University of Cincinnati

Date: 4/29/2015

I, Nathan Wessel, hereby submit this original work as part of the requirements for the degree of Master of Arts in Geography.

It is entitled:
Discovering the Space-Time Dimensions of Schedule Padding and Delay from GTFS and Real-time Transit Data

Student's name: Nathan Wessel

This work and its defense approved by:

Committee chair: Michael Widener, Ph.D.

Committee member: Tomasz Stepiński, Ph.D.

Committee member: Paul Zender, M.F.A.
Discovering the Space-Time Dimensions of Schedule Padding and Delay from GTFS and Real-Time Transit Data

Thesis submitted in May of 2015 to the Graduate School of the University of Cincinnati
In partial fulfillment of the requirements for the degree of

Master of Arts

In the Department of Geography
of the College of Arts and Sciences by

Nathan S. Wessel
Bachelor of Urban Planning,
University of Cincinnati

Committee chaired by Dr. Michael Widener
Abstract

Schedule padding is the extra time added to transit schedules due to expected random variability in travel times throughout a route. To this point, methods for applying padding to certain route segments and times have been relatively unsophisticated, largely reacting to observed changes in travel time variability relative to the existing schedule. By comparing schedule data and real-time vehicle locations, we aim to locate the segments of routes that are most affected by this random variability, and thus have the most padding. These segments could most benefit from targeted delay reduction techniques, such as signal prioritization or multi-door boarding. We also outline cartographic methods that could be used to depict such results to lay people and policy-makers. Our approach is relevant to any city with both General Transit Feed Specification (GTFS) data and a real-time vehicle location feed, though we take a single large city as our case study.

For this research, we focus on Toronto, Ontario, and the Toronto Transit Commission. We use real-time transit vehicle locations, obtained from a publicly available API, to establish what we take to be a reasonable maximum speed for each segment of a route, or any set of routes. We then compare this ideal performance to the scheduled performance, derived from GTFS data, where the difference between the two can be interpreted as the amount of schedule padding on a segment at a particular time. Since schedule padding is a response to stochastic delay, this technique should lead us to the spatio-temporal locations of the most significant sources of delay and guide attempts to reduce delay and maintain acceptable on-time performance.

Results suggest that choke-points are observed to crop up in expected places, like downtown at rush-hour, or near major signalized intersections. What is interesting is that we are able to begin to quantify delays in one part of a transit system, and compare them to other delays anywhere else in the same system. This should help to guide infrastructure investments that can minimize the impacts of random delay and to justify such expenses with explicit potential time or money savings.

Transit delay has been discussed extensively in the literature and in the press, but this delay is relative to a schedule which can be fairly arbitrary. This thesis is novel in its emphasis on schedule padding as a signal of avoidable delay below the level of the scheduled expectation.
# Table of Contents

List of Figures, Maps, Tables and Videos.................................................................V
1 Introduction...........................................................................................................1
2 Literature Review.................................................................................................5
   2.1 Data-derived performance measures for industry............................................5
   2.2. The great opening of transit data and the real-time revolution.........................6
   2.3 Performance measures outside of industry.........................................................9
   2.4 What remains to be considered.......................................................................11
3 Outline of the Project..........................................................................................12
4 Passengers as Sources of Delay........................................................................15
   4.1 Stopping.........................................................................................................15
   4.2 Internal congestion.......................................................................................16
5 Data.....................................................................................................................18
   5.1 General Transit Feed Specification.................................................................18
   5.1.1 Simplification, Segmentation, and Summation..............................................18
   5.2 Real-time Vehicle Locations.........................................................................21
   5.2.1 Handling Errors.........................................................................................22
   5.2.2 Distinguishing Tracks................................................................................23
6 Taking Measures................................................................................................25
   6.1 Establishing reasonable minimum speeds on edges........................................27
   6.2 Calculating schedule padding.........................................................................28
7 Cartographic Depiction.......................................................................................31
   7.1 Maps............................................................................................................31
   7.2 Animated Maps.............................................................................................57
8 Interpretation and Discussion..............................................................................61
   8.1 Do the results match basic expectations?.........................................................61
   8.2 Some exemplary segments............................................................................63
9. Concluding remarks............................................................................................67
   9.1 Lessons learnt and directions for future research............................................67
   9.2 In Summary.................................................................................................70
References:..............................................................................................................72
Appendix 1: Code.......................................................................................................76
Appendix 2: Example padding calculation...............................................................84
List of Figures, Maps, Tables and Videos

Figures

5. Map showing unintended effect of stop clustering technique. page 20.
6. Map showing sample polylines generated from NextBus API vehicle locations feed. page 22.
7. Map showing location reporting error in Downtown Toronto. page 23.
8. Taking measurements: matching vehicle tracks to scheduled edges. page 25.
10. Taking measurements: points on line intersections from which the measurements are finally taken. page 26.
11. Cumulative distribution of service hours at observed speeds across all segments. page 28.
12. Scheduled temporal distribution of service hours and the proportion of them which appears to be padding. page 30.

Maps

1A & 1B: Observed mean speeds on network edges. pages 33, 35.
2A & 2B: Scheduled mean speeds on network edges. pages 37, 39.
3A & 3B: Observed fastest speeds (9th decile observations). pages 41, 43.
4A & 4B: Speed of fastest scheduled trips. pages 45, 47.
5A & 5B: Padding per trip per kilometer given 9th decile observations as maximum speed. pages 49, 51.
6A & 6B: Padding per trip per kilometer given fastest scheduled speeds as maximum speed.

7: Padding on Eglinton Avenue East between Laird Drive and Yong Street.

8: Padding on Sheppard Avenue West from Allen Road to Keele Street.

9: Padding on Broadview Avenue and Pape Avenue.

Tables
1. Variables returned by the NextBus API's vehicleLocations command.
2. Correlation between various values on network edges.

Videos
1. Map 5A animated with 15 minute time steps.
2. Map 6A animated with 15 minute time steps.
1 Introduction

Public transportation is a complex phenomenon, fundamental to the growth and maintenance of major cities in the industrial and postindustrial parts of the world. A great deal has been written about public transit in recent decades, particularly as the bloom has worn on modernity's experiments in automotive mass-mobility. Most of this work demonstrates that there is little agreement, outside of the transit industry itself, about what constitutes good or appropriate transit, how much needs to be spent to get it right, or even what the real point of transit is supposed to be in the first place. One thing does seem certain though: where transit does exist, all else being equal, everyone wants it to be faster and more reliable.

In scheduled transit operations, speed and reliability can be thought of as being a function of three things:

1. **Fixed speed limits** of a legal, physical or social variety.
2. **Random delay** which a vehicle actually encounters along its route.
3. The amount of random delay planners **expect** a vehicle to encounter along its route, and for which they account in their schedules: this is called **schedule padding**.

The combination of these three factors determines how fast a scheduled transit service can and does operate and the latter two by how much it should be expected to deviate from its schedule. Random delay, is best understood as something not directly affected by schedules. It is anything that can delay a transit vehicle or a vehicle generally, but which is not strictly predictable. This sort of delay is due to many possible factors: traffic conditions of various sorts, wrecked or double-parked vehicles, the number of red traffic lights encountered, the number and type of passengers needing special attention, the number of tortoises observed to be crossing the street, etc. By contrast, any...
fixed ceiling on a vehicle’s speed is outside of this category; speed limits include things like stop signs, actual legal speed limits, the laws of physics, and the psychological constraints enforced by the need to operate safely around humans. Of course, ‘random delay’ does not imply that randomness is not fairly predictable; this should be an intuitive concept: if someone is heading to a meeting, they often plan to leave a bit earlier than you think you absolutely need to so that if anything ‘unexpected’ happens, they are still likely to get there on time. The ‘unexpected’ is expected. Scheduled transit operations do the same thing on a larger scale, planning to allow some amount of extra time to get to each stop on the line or each ‘timepoint’. If vehicles encounter no delay and arrive early, they can simply linger until their scheduled departure before moving on to the next stop. The extra time built into a schedule for this reason is called schedule padding. (The use of the exact phrase ‘schedule padding’ is surprisingly scarce in the academic literature, given that such a thing so clearly exists and commonly goes by that name. But rare or not, schedule padding has indeed been mentioned in government documents, academic research, and the popular press.) Another way of describing padding is to say that schedule padding is the difference between how fast you could get somewhere in the best case, and how fast you expect to be able to get there in the average or somewhat-worse-than-average case. To be clear, schedule padding is a response to random delay, paying in potential speed to purchase reliable adherence to a prearranged schedule. If there were absolutely no chance of delay, there would be no need for schedule padding. Ideally we could eliminate the potential for delay altogether; things like stop consolidation, transit-only lanes, signal prioritization or multi-door boarding all could have the effect of reducing travel time variability and increasing speed to the point where it bumps into stronger limits. In theory, and with enough resources, we could get much closer to operating a system not subject to significant
random delay. Indeed, this sort of reliability is one of the benefits of something like a metro or subway system: a vehicle that stops at every single stop, and which takes roughly the same time to board one passenger as a hundred, and which never gets caught in highway gridlock should experience relatively little travel time variability.

To say prosaically what Figure 1 describes, there is an interaction between random delay and schedule padding that determines the overall speed and reliability of a trip. If no delay is encountered and no padding scheduled, then everything is running maximally fast and perfectly on schedule. If a great deal of delay is encountered and still there is no padding, then the trip runs very slowly and is also very late. If a great deal of delay is encountered and the padding is commensurate, then things are running very slowly but they are still on time. If a great deal of...
padding is scheduled but no delay encountered then the trip runs much slower than it needed to, but remains on time. There is an obvious balance between the two: padding should be directly proportional to the amount of delay expected, and the relationship, at least in theory, should go in that direction: expected random delay determines a proportionate response in padding from the makers of schedules.

Speed and reliability are, in scheduled transit services, a good thing; we want to go as fast as possible and we want to eliminate any discord between the schedule and reality. If speed and reliability are indeed a function of random delay and schedule padding, and schedule padding is itself a function of random delay, then the whole root of the matter could be unearthed by observing the incidence of random delay in space and time. This however is difficult, even conceptually, because vehicles are always simultaneously operating under the constraints of both the delay itself and expectations about that delay: schedule padding. Once padding is introduced into a schedule, it becomes difficult to tell whether a vehicle is slowing down for a delay or to adhere to its schedule or both. Therefore, instead of looking for the delay directly, we will look for schedule padding, which we hypothesize to be a direct function of actual travel time variability. Once we can quantify where and when schedule padding exists, we should also be able to quantify the local impacts of random delay. With quantifiable measures of the financial and time costs of delay, efforts to reduce delay and improve speed and reliability can be better targeted to the times and places where they can have the most impact. As civil governments decide what to do with their inevitably limited resources, quantitative measures of the potential for improvement should help to guide their efforts.
2 Literature Review

There are three broad areas of the literature that will be of interest here. In the first, we will explore the research, starting in the 1990's, which discusses the use of new technologies and the data they produced to create performance measures for transit agencies themselves. In the second area, we look at research done by academics and others outside of the transit industry once the same types of data became available to the general public. This research focuses on the way transit users use these data to navigate and on efforts to optimize these systems for actual use by passengers. In the third area of the literature, researchers, still generally outside of the transit industry, look to see what can be learned from taking a wider view of these data, applying them for purposes not originally intended.

2.1 Data-derived performance measures for industry

For as long as the technology has been available, transit agencies have been using computerized scheduling systems and, more recently, real-time vehicle position sensors and other types of onboard sensors to manage and assess their daily operations. Such tools have provided an abundance of data which has helped researchers and professionals develop techniques and performance measures to improve the efficiency of transit services. One of the most tangible examples of these are the real-time schedule displays now shown to many bus drivers throughout the course of their duties. The displays relate the vehicle's position to a clock and a digital schedule to give the driver an estimate of how far behind or ahead of schedule they are, allowing them to adjust their speed appropriately. Another innovation, this one affecting the planning more than the operation of transit lines, is the automated passenger counter or APC. APCs are sensors that hang over the doors of a vehicle and count the number of passengers boarding and alighting at each stop. These passenger counts can be highly disaggregate, allowing transit planners to make informed decisions.
decisions about even fairly fine-grained details like where a new stop should be placed or an existing one removed.

There are many other and more esoteric examples, some of which are listed here. In 1989, Bookbinder and Ahlin looked at the interaction effects of routes which diverge from a shared segment while attempting to maintain consistent headways on that segment. They find that once headways on the segment go below a certain threshold they may as well be random, allowing operational savings on certain very frequent routes.

In 1999, Dessouky et al. wrote about using GPS bus tracking technology to better manage dispatching near coordinated transfer points. In 2002, Chien et al. discussed the use and benefits of several different types of artificial neural networks to improve arrival time predictions. In 2003, Bertini and El-Geneidy demonstrated the possibility for a wide range of simple performance measures derived from archived spatio-temporal data in Portland. They break the measures into system-, route-, segment- and point-level metrics ranging from average dwell-time at a particular stop to average passenger loads across the system.

In 2004, Zolfaghari et al. proposed a model to optimize stop dwell times using real-time location information. In 2006, Zhao et al. proposed a mathematical model for optimizing schedule padding in simple scheduled services under the assumption that travel times are exponentially distributed.

Transit agencies find themselves working with more data, sensors, and digital systems than ever before. The trajectory toward more digitally and algorithmically managed operations seems likely to continue in the transit industry, and this sort of research into potential industry applications is ongoing.

2.2. The great opening of transit data and the real-time revolution
transit industry until very recently, when the open-data movement and its early fruits began to lure transit agencies out of their institutional shells and into collaboration with the private sector and eventually the general public. In 2005, Portland's TriMet Transit Authority collaborated with Google to produce what became known as the General Transit Feed Specification or GTFS. The goal was to integrate transit directions with the Google Maps trip planner and TriMet's schedules did indeed go live in the new 'Google Transit' with data supplied to Google in the new format. Fatefully, TriMet decided that their schedule data was public information and that should not only be available to one corporation; they decided to make the same data they were giving Google available to the general public on the same terms. Little else happened with the new TriMet data for a while, but other American transit agencies quickly began to see the benefit of having a well-functioning trip planner already built into a popular navigation tool. TriMet did not stop there and in April of 2006, they were publishing real-time vehicle locations and arrival predictions. By now though, other transit agencies were catching up. In May of 2007 Bay Area Rapid Transit released their schedule information in GTFS format, followed about a year later by real-time data. 2009 saw the start of an avalanche of schedule and real-time data, including releases by San Francisco's MUNI, the Chicago CTA, the Washington's WMATA, LA's Metro, Boston's MBTA and others. By now, hundreds of agencies all over the world have released their own GTFS feeds and many also have real-time data published for at least a subset of their services. Following TriMet's example, most of the transit agencies publishing GTFS to Google for use in Google Transit also published their data on the web, available at least in theory for the general public to consume. From the beginning, and with increasing speed, quite a large number of non-Google user interfaces, mostly cellular phone 'apps', sprung up to help transit customers use this data more easily. Most of these apps were designed primarily to reproduce in a localized way what Google had already done, but some also surpassed it or adapted the basic idea to specialized customer use-cases like making...
the data accessible to blind or cognitively impaired users.

Unlike the GTFS, which quickly became the standard for data on scheduled operations, real-time data has not yet converged on one standard format. At least a few agencies developed and manage their own application programming interface (API) suited to their own local and technological idiosyncrasies. Some proposed standards have emerged, but they have yet to dominate the market. The first is 'GTFS-realtime', proposed by Google; the standard is well documented but we have not been able to find more than a few agencies actually using this format. Another emerging standard is that implicit in the NextBus API, a service provided to transit agencies by a private company; about 65 “agencies” are using this service as of March of 2015.

GTFS was explicitly designed to facilitate customer-facing trip planning applications (like Google Transit), and most relevant research to date has focused on ways to produce and employ such applications or on the measurable or perceived effects these tools have on customer utility and satisfaction. The same is largely true for real-time data APIs. In general, the literature shows that Google Transit and other trip-planners based on GTFS and real-time data, have been a success as far as passengers are concerned. This is perhaps because agencies have only increased the amount and types of information available to customers, without cutting out older forms like paper schedules and system maps. One study in Seattle found that when customers had access to real-time information, they were much less likely to consult published schedules, preferring instead the more up-to-date source. The authors drew some conclusions about the relative importance of regular headways for such customers as opposed to on-time performance. Another study, also in Seattle, showed that people with access to mobile real-time information feel like they spend less time waiting for transit, and in fact that they actually do spend slightly less time waiting.
cases, this is the result of passengers delaying their walk to a stop until the vehicle is actually nearby, or by using some extra time in a productive or social way such as by running errands at a nearby store.

Other research has focused on ways GTFS and real-time information can be presented visually. Since the GTFS encodes so much spatio-temporal information, it should in theory be possible to algorithmically construct a decent transit map. It is in this vein that Alan Joyce in 2011 proposed a generalizable, stationary “spider” type map that ingests GTFS and real-time updates and produces a digital display showing incoming trains and their destinations. Similarly, Wang and Chi describe an algorithm for presenting a localized train map on a small screen. Neither of these projects would have been conceivable without the widespread availability of GTFS data.

Still other research in this general area has proposed methods for approximating agency-provided real-time information in its absence. With the spread of location aware smartphones, and as volunteered geographic information had begun to demonstrate its value, Cuong, in 2013, proposed an algorithm for incorporating user-volunteered real-time updates into a central routing application for other users. Similarly, in 2010, Arvind et al. proposed a method to essentially bootstrap real-time data into existence by using people’s cell-phones, determining when they are on transit by the way they are moving and accelerating, and relaying to the cloud their anonymized positions and thus the positions of the buses.

2.3 Performance measures outside of industry

The third broad category explores the use of publicly available schedule and position data to learn something new about transit, cities or transportation as such. These create performance measures of a sort, but they differ from the first category in that they are more exploratory. They try to answer the question: what is transit like? What is our city like?
To start with a fairly simple case, Patrick Brosi, a computer scientist, proposed in 2014 a system for integrating various real-time vehicle location feeds into a single web map. The challenge may be a more technical than theoretical one, but the interest is general, and the question implicit: what would it look like if we could see on a map all of the transit vehicles in New York City, or in Europe?

Van Oort et al. in 2013 describe their approach to the analysis of several months of real-time stop-arrival information from the Netherlands. They look primarily at several measures of schedule adherence and illustrate examples where the data could be used to better match the schedule to reality or vice-versa. The most interesting of these measures described schedule adherence using mean performance and 15th and 85th percentile bounds to show average performance and variability through the length of a single route. Their analysis focused on particular lines and stopped short of showing significant patterns at the system level.

Gasparini et al. describe a more comprehensive citizen-facing application in Dublin, that ingests real-time transit data and other traffic data sources to make a much broader range of real-time statistics and predictions: a 'dashboard'. This system was designed for highway administrators (who presumably have the immediate power to interact with traffic) as well as citizen-users who can use the system for live trip-planning. Though in this way it seems to propose a predictable hierarchy of access, the platform is designed to be extensible and we see no reason why the detailed analytics available to administrators and planners would not also be available to lay people in the style of other dashboards like the Dublin Dashboard. In fact, since this paper was written in 2011, the Dublin Dashboard is almost certainly the (currently) ultimate fate of the system the paper describes.

This area of research may tackle problems that are of interest to people working in transit,
but it also begins to answer the questions of citizens, politicians and curious transit users who do
ultimately have an enormous impact on the way transit is planned and operated in any given city.

2.4 What remains to be considered

The literature that has just been discussed is interested primarily in reliability, measured as
adherence to a schedule published in advance, and about managing expectations about that
reliability. Do transfers work as we expect them to when there are delays? Do headways on a
schedule relate well to headways in reality? Is it even worth looking at a schedule when service is
supposed to be frequent? Or when real-time data is available? They have implicitly explored the
potential for improvement in schedule adherence, but they have done much less to look at the other
side of that balance: the potential for speed increases. Recalling that speed is traded for reliability
(and vice-versa) through the addition or removal of schedule padding, it becomes clear that we
cannot focus on reliability without bringing speed into the equation. With so much focus on
schedule adherence, it is a wonder that schedules have not slowed transit to a crawl under the
pressure to make conservative estimates of the required travel time.
3 Outline of the Project

As previously noted, we should expect schedule padding to be an adaptive response to systematic random delay. The distribution of this random delay is something we might reasonably expect transit planners to know somewhat intimately; they are constantly tweaking schedules to keep things from running, on average, too late or too slow. Each such adjustment to a schedule is an implicit acknowledgement of some newly recognized change in the distribution of systematic random delay. But this is not to say that transit planners should be expected to have an implicit knowledge of the whole structure of delay. Rather, it is to say that the schedule itself might be expected to evolve step by step toward encoding a response to this distribution as it incrementally attempts to better match reality. It could be that transit planners sculpt the surface of an iceberg without ever knowing its shape beneath the water [figure 2]. Even with well adapted schedule padding, transit vehicles still run late and early relative to their schedule just as well-adapted animals in a balanced ecosystem still die with a certain frustrating regularity. If we were to look at the space-time distribution of variance from the schedule in an ideally padded system, we should expect to see that its distribution over scheduled services is more or less uniformly random. This is because any systematic variation (such as a particular route always being late at 7pm every Tuesday) should have been accounted for (such as by giving that route a bit more padding on its 7pm run).

The balance between speed and reliability is one that can be difficult to reassess once it has been established. For one thing, transit passengers are generally much more concerned about schedule adherence than about potential speed improvements; written complaints about early or late

2 Or a change in tolerance for variance from the schedule, while the actual underlying distribution remains the same.

Figure 2: A metaphor
arrivals seem to be the second most common type of complaint that agencies receive, right after complaints about unpleasant or unhelpful drivers. Transit agencies almost never hear that schedules are not as fast as they could be. This bias is probably due to the fact that schedule padding is generally less visible to transit users than late-running buses. Padding takes many forms. Bus drivers, dwelling longer than necessary at a stop, for example, are not easily interpreted as trying to ‘waste time’ if they use that time to allow people to get to their seats before they accelerate. Drivers going more slowly than they need to are easily interpreted as driving cautiously. Sometimes the excess padding can be used up by trying to catch red lights; again, in this case the passenger’s most straightforward interpretation will be that catching red lights is simply the result of bad luck rather than of any conscious effort expended by the driver. Late buses on the other hand are easy to see, particularly if the passenger has checked or memorized the relevant parts of the schedule ahead of time. Anyone can mentally calculate the number of minutes a bus is late.

But padding is also difficult to see for another reason. Since drivers slow down to exhaust their padding, and may be scheduled for heavily padded routes, it may be that we only rarely see a driver attempting to go as fast as reasonably possible. Anyone getting on a very late bus may notice for instance that the driver will most certainly not wait for them to pay their fare before taking off, and that what might have been a red light is often transformed magically into a yellow. This is not to condone aggressive driving of course, but to admit that there is a difference between driving leisurely and driving quickly in order to make up time. Comedian Jerry Seinfeld once noted the frustration of going more slowly than necessary, in this case, on an airplane, “Pilot says, ‘We’re going to be making up some time’. They just ‘make up time’. That’s why you have to reset your watch. Of course when they say they’re making up time, obviously they’re increasing the speed of the aircraft. Now my question is, if you can go faster, why don’t you just go as fast as you can all the time? Come on. There’s no cops up here. Nail it. Give it some gas. We’re flying!”
The job of the comedian is to comment on things that generally are not reflected upon by the audience, drawing attention to the things we take for granted; Seinfeld's comment here may serve to indicate again that potential speed is not something that most people think about regularly. If it were, the routine would not still be funny.

Despite these tendencies to misunderstanding, if we seek to improve the overall speed and reliability of scheduled transit services, we should follow schedule padding to the source of random delay. This paper proposes a simple model to measure and visualize the spatio-temporal structure of random delay in scheduled transit services. First, we simplify the scheduled operations of a transit system into a graph with stops as nodes and trips between stops as edges representing streets or sets of streets. Next we observe actual transit vehicles traversing these edges and try to establish from these a reasonable minimum time required to traverse each edge. That is, we are trying to find the floor of the time required to operate on any given segment of a route. Finally, we compare this theoretical best time to the average and scheduled times for each segment. By subtracting the best times from the padded, we should be able to see where the most time is being lost to random delay and schedule padding. In so doing, we hope to raise interesting questions about how we might fruitfully intervene to reduce delay and what the time costs are for failing to. We take the Toronto Transit Commission (TTC) as our case study and use publicly available GTFS and real-time vehicle location data for the analysis.
4 Passengers as Sources of Delay

Some sources of variable delay should already be obvious to the attentive reader and obviously undesirable in any situation; red lights and general traffic congestion come to mind. But others may prove more ambiguous. Foremost among these is the simple act of stopping to pick up or drop off a passenger.

4.1 Stopping

Many, perhaps most scheduled transit services do not always stop at all of their scheduled stops. They only stop when someone wants to get off there or when someone is waiting to get on. Further variability is introduced when the number of passengers is varied, since many of these same vehicles are only capable of boarding passengers in $\Theta(n)$ time: a single file queue for the till. First, it must be stated plainly that passengers are not a bad thing that slows down transit service; they are the point of transit service and nothing should be done to diminish their absolute numbers. There are however ways of reducing the time it takes to board passengers and more importantly here, the total variability in the time it takes to board them. There are three basic approaches which may be divided into policies, technologies and service patterns. An example of each will be given.

One policy that can reduce boarding times is a method of collecting fares: what might be called the honor system. In these arrangements, which are not at all uncommon, fare is paid not during boarding, but at some other time, most likely before boarding the vehicle from a vending machine at the stop. Proof of payment is held by the passengers and occasional spot checks by enforcement officers keep the honor of the system from degenerating. In this system, everyone can use all of the vehicles doors as quickly as their legs will take them across the threshold. An example of a technology that has exactly the same effect is the turnstile-walled platform. These are familiar, if not by that name, from almost every subway or ‘metro’ system. Everyone queues to pay to get on...
the platform, and everyone on the platform can board the vehicle all at once when it arrives. This leads us to the final and perhaps most important example: service patterns. If stops can be spaced far enough apart that it is almost a sure thing that at least one person will always be waiting at each, then every vehicle is almost guaranteed to stop at each. Stopping and starting can be made to take a fairly predictable amount of time, so stopping as such does not cause unexpected delay. Not knowing in advance whether a stop will be made is what can result in variable travel times and thus padding. The close spacing of bus stops in many cities means that the number of stops that must actually be made on a given trip should be expected to be highly variable if there are not enough people to keep all of the stops active all of the time.

Changes to service patterns are worth dwelling on for a minute longer. It seems like an arrangement of stops that kept all of the stops busy during rush hour would possibly fail to do so at midnight. What happens then? Service becomes less frequent and passengers are allowed more time to accumulate to the desired densities. This may be hard to imagine for a bus service as they typically exist these days, but think of a subway system and imagine how many stations are ever totally empty when a car pulls up. We will not contest that all transit lines could or should be operated with stops spaced as far apart as the average subway, but it is worth acknowledging that different stop spacing patterns should be expected to result in stops that are more or less full on average, and that stops that are only sometimes occupied will result in more travel time variability.

4.2 Internal congestion

Another source of delay is the time spent shuffling people around after the stop is made. Standing while a vehicle is accelerating is difficult, particularly for the infirm or over-laden, and many people wait until the vehicle is at rest before making for the door. In a typical sparsely filled vehicle, this adds little if any time since the passengers may alight from a second door while fares
are paid at the other. In congested vehicles though, even multi-door traincars, when people are standing in the narrow walkways, this internal reshuffling can take some time. At least two solutions to this problem exist. First, the interior of the vehicle can be decompartmentalized, allowing passengers to spread out and decongest themselves before they select a seat. This can be seen in practice in many new train designs currently coming into use that join the cars with flexible and passable segments. Another approach is to maximize the openings through which an exit can be made. The San Francisco cable cars exemplify this technique with their unusually outward-facing benches and their famously out-of-doors hangers on. This sort of vehicle design is quite unusual in transit of the 21st century, but there is no reason it should not make a comeback and even be extended and improved by new technologies.

Imagine for instance a typical bus or trolley, the walls of which are almost completely made of doors. Passengers would less often need to shuffle from their seats into the aisle—they could jump right off, and others on.
To actually measure the size and location of schedule padding and delay we need some data. There are two distinct data sources used for the analysis that follows and we will describe them one at a time before getting into the ways they are combined. Both data sources are available for the TTC, a large agency with a wide variety of transit services. The goal though is that the methods described here will be easily extensible, and applicable in particular to other cities with similarly structured data.

5.1 General Transit Feed Specification
A schedule published in GTFS is a set of CSV text files that form a relational database which explicitly describes the entirety of an agency's scheduled operations. Each route has a set of trips and each trip is given a "shape" (a polyline) and a set of stops (points along that polyline) with arrival and departure times at each stop. A "trip" is defined loosely as one instance of a vehicle going in one direction on one route, something such as "#17 eastbound, from 5:30". Many routes have several types of trips, one for each possible deviation and direction. We will be using GTFS data from the TTC, issued for the period from January 4th, 2015 through February 14th, 2015.

5.1.1 Simplification, Segmentation, and Summation
Our analysis takes place at the level of distinct, scheduled route segments. That is, any set of two stops such that a vehicle on any route is scheduled, to go from stop A directly to stop B. These segments can be conceptualized as edges of a directed graph, with stops as the nodes. The GTFS data from the TTC contains 183 routes, 10,749 stops, and 12,969 distinct segments (as segments have just been defined). In the course of a week, these segments are scheduled to be traversed (a vehicle is scheduled to go from one stop to another) a total of 9,538,851 times.
times. Computationally, these numbers present little difficulty, but a visual inspection shows that there appears to be significant redundancy in the data, particularly near bus terminals and major intersections where multiple stops may be used to represent what is more easily thought of as one large station [Figure 3]. There are also at least several hundred places where stops are paired on opposite sides of a street or clustered tightly around an intersection. Because one of the goals of this paper is to visualize the results for a general audience, and since we want to get the largest possible number of observations on each segment, we will cluster these stops and re-segment the data. To do this, we draw 30 meter buffers around all stops and derive the union of the geometries where there is any overlap. This new unified buffer geometry will form the boundary for clusters of stops, where all stops within the bound are in the same cluster. The centroid of the stops in a cluster is established as the new node for the purposes of visualization. The unified buffer geometry will be used later in the analysis. A visual inspection shows that for the most part this clustering works about as expected [Figure 4]. In most instances, stops are left alone, paired with another directly across a street, or joined as part of a larger intersection or station cluster. There are however a few places where the clustering appears to be somewhat less successful in simplifying things [Figure 5]. In these cases, stops that obviously form distinct clusters to the human eye were merged by the algorithm. In a number of cases, this pulled the centroid of a cluster decidedly out of an important intersection as in the right side of Figure 5. These errors were relatively few and were balanced by occasional poor performance in the other direction: stops failing to cluster when they
definitely should have. In any case, establishing a good clustering distance was difficult, but after trying several we believe we have converged on a good solution in the general case, at least for Toronto. Other agencies will have different data quality standards and stop distributions. To emphasize again though, this step is primarily to simplify the visualization of the results of the analysis.

Because the transit system is being represented as a graph, it can be seen that this method may have done more than innocently combine unconnected nodes: it may also have collapsed edges. In fact it did do just that; at this level of clustering, approximately 2% of all scheduled weekly vehicle hours (~3,000 of ~145,000 hours) operate on segments that now exist completely inside of clusters. This is assumed to be a tolerable payment for the graphic simplicity afforded by this...
These inter-cluster trip segments will be disregarded in the rest of the analysis. After clustering, we observe in our dataset 5,261 stop clusters (reduced from 10,749 stops), 141,598 weekly vehicle hours scheduled on segments between clusters, 3,319 undirected (or bidirected) edges, and 3,710 directed edges. These simplified edges will become the basic unit of analysis.

5.2 Real-time Vehicle Locations

The schedule must be compared to some measure of reality and it is for this purpose that the NextBus API [19] is used. NextBus is a company whose business it is to publish, on behalf of transit agencies, useful information about where their fleet is right now (in ‘real-time’). They maintain a public web API which, for example, makes predictions about when the next vehicle will be arriving at any given stop. The part of this API relevant to this paper reports the current locations of all vehicles in the agency’s fleet along with several other variables [Table 1].

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique identifier for actual vehicle</td>
</tr>
<tr>
<td>routeTag</td>
<td>Public route number</td>
</tr>
<tr>
<td>dirTag</td>
<td>Some combination of the routeTag and a 0 or a 1 for inbound or outbound</td>
</tr>
<tr>
<td>lat</td>
<td>Latitude to seven digits after the decimal</td>
</tr>
<tr>
<td>lon</td>
<td>Longitude to seven digits after the decimal</td>
</tr>
<tr>
<td>secs</td>
<td>Integer number of seconds since last update from vehicle to server</td>
</tr>
<tr>
<td>predictable</td>
<td>Vehicle is on-route and arrival time predictions can be made</td>
</tr>
<tr>
<td>heading</td>
<td>Integer bearing in degrees between 0 and 360</td>
</tr>
<tr>
<td>time</td>
<td>Epoch in milliseconds that the server generated the report</td>
</tr>
</tbody>
</table>

Table 1: Description of variables reported in by the NextBus API

The API serves up XML in response to HTTP GET requests and a Python script was used to request this data.
updated vehicle locations once every several seconds. These recordings were gathered on and off for several weeks, for several hours or days at a stretch during the period between January 4th and February 14th of 2015. This was not a systematic effort because of the difficulty of finding a truly stable Internet connection, but at least one recording was gathered through each distinct part of the service period, including a weekday and both days of the weekend. These records were stored in a PostGIS database alongside the GTFS data discussed in the last section. Just over 33 million vehicle locations were recorded. Unfortunately, the API does not report the position of any of the TTC operated subways, presumably because of a reliance on satellite-based GPS devices. It does however report for all other lines, 179 in total, including frequent urban buses, infrequent suburban buses, streetcars, night routes and lines operating in transit-only rights of way.

### 5.2.1 Handling Errors

The data, while generally of a high quality [Figure 6], had a somewhat confounding number of seemingly erroneous location reports. This was most notable in downtown Toronto, presumably where tall buildings interrupted GPS signals. A map of downtown [Figure 7] in the same style as Figure 6 serves to show the obvious positional error in the data for the city's core. Outside of downtown though, locations would occasionally seem to be running along fine just before jumping.

---

4. Several different amounts of delay between requests were tried, mostly between 5 and 10 seconds, but they seemed to make little difference. The vehicles were not updating their locations fast enough for it to matter. Some amount of randomness was introduced to all request times anyway by the network delay between the computer and the NextBus servers, which, while mostly in the millisecond range, got as high as 5-10 seconds.

5. The 'period' of transit schedules generally, as in the amount of time before the repeat of a cycle, is one week. Weekday service is all the same, but the weekend days are distinct from the weekdays and from each other.
a block or more and then returning to their previous course. The worst of these sorts of errors tended to occur near bus depots; it seemed as though when a vehicle left service it would sometimes, instead of ceasing to report its location, report a location as much as 50km outside of the city as though to place itself off of a map. Handling these errors was the easiest; points that were clearly not on any route, such as those in Lake Ontario, were manually deleted after all the data had been collected. This included something on the order of 10,000 points clearly outside the bounds of any route. Many smaller errors remained though, such as those in downtown, and these were tougher to detect. For these, we measured the speed of every reported segment (Euclidean distance over time from one reported location to the next) and dropped the later location when the speed of the segment exceeded 120kmph (~75mph). Again, only ~5,000 points were removed but the data visibly improved.

5.2.2 Distinguishing Tracks

For the analysis, these (point) location reports had to be strung together into polylines representing the trajectories of discrete transit services. These polylines, based on observed vehicle locations, will be referred to as 'tracks' (as in 'GPS tracks') throughout this paper. Without getting into the details of implementation, it will suffice to say that a new track was started in any of the following cases:

1. A vehicle appears for the first time in more than 60 seconds.

Figure 7: Map showing erratic GPS positions in downtown Toronto, most apparent here on Bay Street, King Street and Queen Street.
2. A vehicle reports operating on a new route, different from the last one, if any, that it reported
3. A vehicle reports operating in a different direction the same route as the last one it reported
These tracks simply continue until the vehicle creating them meets one of those criteria again, at
which point a new track is started and the old track ended at the previous point. Excluding tracks of
5 or less points, and tracks shorter than 300 meters, the original ~33 million points generated
~540,000 tracks with a mean length of 6.3 kilometers, median length of 4.92 kilometers and an
average of 61 points per track. Track length was naturally somewhat variable, but a visual
inspection showed that almost all tracks appeared to extend the whole length of the route to which
the vehicle was assigned. After removing outliers and constructing tracks, the average temporal
delay between location reports (inside of all the tracks) was about 20 seconds.
In order to get observations at the level of network edges, it was necessary to match tracks to the edges on which they seemed to be operating. Essentially, the goal was to locate two points on each track where that track passes nearest to the stops at the ends of an edge and to take the measurements from between those points. There are two measures of interest: the distance between the stops and the time between the stops. Both of these measures, one for each matched track, were stored with each edge for later analysis.

Each vehicle track has a route assignment and each edge has a set of routes that potentially operate on it, making it easy to match an edge to a set of potential tracks: if the edge serves routes 3 & 4, then select all tracks from routes 3 & 4. Next, a simple spatial intersect operation filters out any tracks that do not pass within 30 meters of any stop in the clusters at both ends of a segment [Figure 8]. In order to match, a track must pass within 30m of stops in both clusters. This leaves only the tracks that are potentially scheduled for the edge and that also come very close to both ends of it. From here, the portion of the tracks that intersects the cluster buffer geometry is isolated [Figure 9]. For each track, the center of this intersection is the same 30 meter buffer geometry that was used earlier to cluster the stops.
chosen as the point from which measurements will be taken

These points will either be coincident with a point on the track (an actual location report), in which case the time for that point can be found easily, or they will be between two such points, in which case simple linear interpolation gives us an estimate of the time. Only tracks going in the correct direction are matched to a segment. Bi-directed segments divide the tracks into a group for each direction and store the measurements accordingly.

Since the distance is measured from the center of the track's intersection with the cluster, the distance is measured not from the nearest points to the individual stops, which are all in slightly different positions, but from an approximate middle of the cluster relative to a direction of travel. Since we wish to assume that an edge represents a distinct (possibly bi-directed) part of a roadway, measuring from the centroid of a cluster has the benefit of selecting a point somewhere between staggered stops which would not be the case if we measured from the nearest points.
Otherwise overlap. Using this method, the data gives ~11 million matches between the 10,085 directed edges and the ~540,000 tracks. This is an average of around 1,100 trips matched to each directed segment.

Significantly, some edges had less than 10 matched tracks, and these were discarded from the rest of the analysis. In almost every case, such edges appeared in one of two conditions. Either the edge had only a few scheduled trips and there were accordingly not enough tracks, or the edge happened to be right where many vehicles declared new routes or directions and those broke into a new and separate track.

6.1 Establishing reasonable minimum speeds on edges

From the observed speeds must be found something like a reasonable best case travel time – undelayed but neither erratic nor breaking any speed limits. A naive approach would be to select the fastest observed trip and be done with it but the presence of occasional erratic positional jumps in the data makes this approach unreasonable even after the removal of obvious outliers. Instead, we assume that the speeds observed on each edge will be distributed with outliers at both ends, slow and fast. To get a reasonable but not extreme maximum speed, the observations on each edge are sorted according to their durations (the time between the intersection with the first stop cluster and that with the second), and the observation at the boundary of the first decile is used [Figure 11]. To say it another way, the observation at the 90th percentile of observed speeds is used as the theoretical maximum. It would be interesting to investigate more advanced algorithms for establishing a theoretical maximum speed for network edges, but this metric will be sufficient for our purposes in elaborating the basic method we set out to discuss.
6.2 Calculating schedule padding

One interesting measure of the reasonableness of a choice of minimum duration (maximum speed), is to multiply the new time by the number of trips traversing that edge for a given period [Equation 1].

\[
\text{MinimumServiceHours} = \sum (\text{MinimumDuration} \cdot \text{NumTrips})
\]

[Equation 1]

Doing this for the average scheduled times on each edge and summing the result [Formula 2], one would get the total number of service hours that the agency is actually scheduled to operate [Figure 10].

\[
\text{ServiceHoursScheduled} = \sum (\text{AverageDuration} \cdot \text{NumTrips})
\]

[Equation 2]

We take the week as the service period, and observe for the TTC ~131,500 service hours at the average scheduled speed and ~74,500 service hours at the new first-decile minimum speed on the same number of trips. Now clearly this should not be read too easily to say that the difference, about 28,000 hours, should be interpreted as a reduction in the need for service hours.

Figure 11: Cumulative distribution of potential total service hours at a range of observed speeds relative to the observations on each segment. Service hours are calculated as the time taken for a given trip times the number of such trips scheduled. If, to invent a limiting example, all edges could be traversed at 120 kmph on each trip, TTC would only need to operate for ~21,900 hours each week to make the same number of trips on the same routes.
57,000 service hours, is due purely and simply to the confounding forces of delay and padding, though it does seem to imply that. A more cautious way to approach this might be to also compare all scheduled times on the same edges, selecting the minimum scheduled time and multiplying that by the number of trips scheduled [Equation 3].

\[
\text{MinimumServiceHours} = \sum (\text{MinimumDurationScheduled} \cdot \text{NumTripsScheduled})
\] [Equation 3]

To get some idea of what a reasonable range might be, we did this for two agencies, the TTC and also the Southwest Ohio Regional Transit Authority (SORTA) a smaller transit agency in Cincinnati, Ohio. For the TTC, we found a padding proportion of 29.4%, and for SORTA, 20.7%.

This analysis is simple to implement with plain GTFS data, but of course it only looks at the schedules and not at any measure of how well those schedules reflect reality. We can use this measure though to give an idea how realistic any estimate of minimum speed may be.

Figure 10 shows the relation of these two numbers, TTC’s actual scheduled service hours [Equation 2] compared to the summed scheduled floor of the segments for the same trips [Equation 3], as well as the range of observed trips on those segments.

Ultimately, this sort of analysis may raise an empirical question interesting in its own right: what portion of an agency’s schedule is, just is, padding? Naturally, the amount would vary widely between agencies and there is probably a normal amount of padding that depends on the urban and political context.

Also of interest is the temporal distribution of scheduled service and schedule padding [Figure 12]. The highest curve on the chart represents the total number of service hours per hour, or to put it more intuitively: the number of vehicles in operation. The lowest curve is the total portion of those service hours that can be identified by the schedule alone as due to padding, or the...
The middle curve is the same as for the lowest but using the observed maximum speeds [Equation 1]. Plotting a distribution of service and padding through time shows that first, and obviously, there has to be service for there to be padding. Second, padding, and a lack of padding, show up right where we should expect them to. Padding is thickest in the weekday rush-hours, even when accounting for overall service levels, and thinnest in the early mornings and late evenings. Similarly, weekdays have relatively more padding overall than either Saturday or Sunday. The most important lesson to draw though is that total padding is dominated by the overall level of service.

Figure 12: Proportion of padding to service.
7 Cartographic Depiction
7.1 Maps

There are many dimensions of this data that call out to be mapped, and mapped at a significantly greater scale than this format allows. In the following pages, we have provided two maps of each of a number of variables along with a prose summary of each. One map of the pair overviews the entire TTC system, covering an area of several hundreds of square miles, and the other depicts a more manageable subset of that area, allowing an analysis at the level of streets and edges. All maps use the same basic cartographic method:

1. The width of edges is set according to the number of trips scheduled to traverse that edge in a one-week schedule period. This has the effect of giving the edges a visual weight roughly equal to their share in the number of scheduled vehicle miles traveled.

2. Edges are offset to their right, making directionality visible even on bidirected edges.

3. Edges that were discarded from the final dataset for any reason are depicted as a narrow dashed line.

4. Color is used to depict the variable of interest, which is always divided by a measure of distance. This means that something like the amount of padding on a segment becomes the amount of padding per kilometer. This controls for the length of the edges and prevents long edges from dominating the map.

Maps 1 & 2 compare scheduled and observed mean speeds. Maps 3 & 4 depict maximum speeds, both observed and scheduled. Maps 5 & 6 derive a measure of schedule padding.

A more advanced but slightly more obscure technique might try to control for the difference in length between edges and actual paths by multiplying the number of trips by a simple factor derived as the edge length over the actual length.
It is worth noting briefly that a weakness of this approach to representing a transit network as a graph of stop-to-stop edges is that express lines, lines that do not stop at all the stops that local lines do, can cause problems by overlapping the local segments. This also means that levels of service are occasionally improperly depicted (by line thickness) because the trips do not compound to create thicker lines when they are on separate edges. This problem is visible particularly in the subset maps that follow, but addressing it seriously is beyond the scope of this paper.
Map 1A: Observed Mean Speeds (full system, 1:250,000)
Mean speed is calculated as the sum of the observed lengths of matched tracks divided by the sum of the observed durations [Equation 4].

For the most part, this map shows that the observed speeds are spatially distributed as should be expected. On the whole, speeds are lower in the center of the city and somewhat higher in the suburbs. Limited access highways, those long red edges on the west side of the map, have the highest average speeds of all. In many of the outer neighborhoods, where the regular grid pattern of the network establishes itself most firmly, trips become slower on segments adjacent to the major intersections and faster on segments not adjacent to these. The next map will show this effect in more detail.

The numerical distribution of average observed speeds across edges appears to be more or less normal, though it should be noted that there is a slight tendency for edges with fewer trips to move faster than those with more.
Map 1B: Observed Mean Speeds (subset, 1:40,000)
To establish the pattern for the next several pages, it should be reiterated that this map is a subset of the previous one, at 4/25 of its scale (1:250,000 vs. 1:40,000). The subset is drawn from the northwest quadrant of the city.

Again, we see the effect of trips appearing slower near intersections and faster on edges that do not adjoin intersections.
While similar to map 1A at the highest level, particularly in the central slow-down, scheduled speeds are clearly different at the edge level in a number of ways. Most notably, it seems that there is a much stronger network autocorrelation. Contiguous edges have close or identical speeds in the schedule. Breaks in speed seem to occur at, rather than near, major intersections. This would seem to indicate that planners may be writing schedules for larger blocks of edges and interpolating across the edges that we take here as the unit of analysis. If true, this would have important implications for the calculation of schedule padding. These concerns will be addressed in more detail later.
Map 2B: Scheduled Mean Speeds (subset, 1:40,000)
Again, we see that the schedule does not seem to depict the change in speed near intersections to the same extent that was so visible in map 1B. Routes have a much more continuous speed, and it seems as though speeds may be set between major intersections, but not at the level of stop-to-stop segments.
Map 3a: Fastest (reasonable) Observed Trips (full system, 1:250,000)

Recall that what is shown here is the ninth decile of the observed speeds for a segment. Clearly faster than the average speeds, this is more than just a subtraction—it is a definite change in relative distribution. Speeds appear to be slightly more continuous than those in Map 1A, indicating that perhaps something like a ceiling/floor for many edges has indeed been reached. This effect is due in part of course to our having exhausted the range of the color scheme, however the speeds do appear more continuous even where red does not predominate.

42
Map 3B: Fastest Reasonable Observed Traversals (subset, 1:40,000)
It is difficult to interpret this map because of the predominance of values in the highest, and unlimited, category. One thing that does seem clear is that intersections still seem to have their slowing effect. Whether this is due to more passengers boarding at major transfer points, the timing of traffic lights, or with something else entirely, there is no imminent way of knowing.
Map 4A: Fastest Scheduled Trips (full system, 1:250,000)
This map is much more continuous than maps 3A & 3B, indicating again that the schedulers may be writing schedules for whole blocks of what we take as our basic unit and interpolating across stops. Similar to what the earlier, and non-spatial distribution showed [Figures 11 & 12], this map is on the whole slower than 3A and 3B. This is because we selected a set of maximum speed observations based on the expectation that the schedule would be padded throughout and that the floor was on average lower.
Map 4B: Fastest Scheduled Trips (subset, 1:40,000)
Map 4B: Fastest Scheduled Traversals (subset, 1:40,000)

Speeds on map 4B are still more continuous than the observed speeds [Map 3B]. It also becomes clear that there are big differences in speeds scheduled for overlapping segments on Finch Avenue East, the main street crossing the right/north side of the map. This street is served by the 39 Finch East and the 199 Finch Rocket.

Sometimes called a 'limited' service.

As the name implies, the rocket is an express bus, stopping only at major intersections. The Finch East runs local, stopping three or four times as often as the Rocket.
Map 5A: Padding per kilometer at observed minimum (full system, 1:250,000)
This map shows the average number of seconds of padding scheduled per trip, per kilometer. That is, if a vehicle traverses a 1 kilometer edge, we expect to observe this many seconds of padding on that portion of the trip. Or to put it a different way, this is the difference between the time needed to cover a kilometer in the (presumedly) undelayed case and the average time scheduled to do so. Note that since observations are being compared to the schedule, there are some negative values. These indicate that the schedule is probably setting unrealistic expectations in those places since less than 10% of trips were observed to be going that fast.

A large amount of padding appears in downtown as expected, but several other corridors also stand out sharply. Several of these will be explored in more detail in section 8.2. Also of interest are the areas that seem to have little padding: much of the far west side of the city for example.
Map 5B: Padding per kilometer at observed minimum (subset, 1:40,000)
Two corridors in this subset appear to have quite a bit of padding, both of them on Sheppard Avenue East:

1. Between Warden Avenue and Victoria Park Avenue (upper left / southwest of map): This is a five and six lane roadway with a sixty kilometer per hour speed limit and a mix of strip-commercial and residential uses fronting.

2. Between Brimley Road and McCowan Road (bottom/east of map): This is a four-lane roadway with a sixty kilometer per hour speed limit and a mix of mostly big-box and industrial/warehousing uses fronting.

It seems rather curious that these segments should have a lot of padding, when they are not very different from the rest of Sheppard Avenue. As curious perhaps as that what seems like a suburban thoroughfare should have much at all. One guess, proffered without having actually visited these places, is that they suffer from tremendous traffic congestion. Another is that the traffic lights at four to six lane intersections have particularly long periods, making the effects of catching one more disastrous. A third possibility is that despite the steady bus service, this corridor is beset by lower ridership than other similar areas because of its orientation toward the automobile. Sporadic boardings and alightings could account for the large amount of variability in travel times. These speculations are offered without evidence.
Map 6A: Average number of seconds of padding per kilometer at scheduled minimum speed (full system, 1:250,000)

Fitting with our other observations, this map shows more network autocorrelation than the maps based on observational data. The schedule appears to have been constructed of larger units and that is evident at the level of maximum speeds, average speeds, and here with the combination of the two. Note that there is less overall schedule padding in this map because of the way maximum speeds were selected from the real-time data; they were selected on the assumption that padding would be everywhere in the schedule and that scheduled maximum speeds would be conservative.

Still, many of the same edges pop out with 140+ seconds of padding per km. It seems here that those edges are more likely to be on approaches to the subways. The subways tend to be the ends of routes in Toronto, so it may also be that extra padding is being included at the end of routes to allow them to make timed transfers at the bus stations over the subway entrances.
Map 6B: Padding per kilometer at scheduled minimum (subset, 1:40,000)
Map 6b: Average number of seconds of padding per kilometer at scheduled minimum speed (subset, 1:40,000)
7.2 Animated Maps

The previous section (6.1) has looked at averages across times. Because, as has been
demonstrated already with the GTFS data alone, schedule padding should be expected to vary in
time, it makes good sense to look for that temporal variability in space too. To that end, the two
animated maps that follow use the same cartographic techniques as the maps in the last section, but
apply them to 15 minute periods, each constituting a video frame. Recall that line thickness is
linearly related to the number of trips scheduled for a segment and that the number of trips
scheduled also fluctuates in time. For each map, the histogram should be read as approximating by
its total area the total amount of padding in the system at that moment. The distribution of that
padding around the different values of padding per kilometer give an idea of how spatially
concentrated the sources of that padding are.
Temporal variability in padding per kilometer per trip at observed maximum speed.

Note that edges in green have zero padding or negative padding. This video is also available on YouTube at https://www.youtube.com/watch?v=5GrFkVNcBeI.
Animated map here: SCHEDULED Video 2: Temporal variability in padding per kilometer per trip at scheduled maximum speed. Note that edges in green have exactly zero padding. Green segments define the minimum for an edge. Nothing can be scheduled for less than the minimum, which was not the case for the last animation. This video is also available on YouTube at https://www.youtube.com/watch?v=8Q6yLfygOxo
There is a tremendous amount going on in these animations; in fact, they may seem rather overwhelming. But they are able to draw out a few interesting things that the averaged static maps leave relatively hidden. First, there is a lot of systematic, seemingly explicable, temporal variation across space. Naturally, on the whole, scheduled speeds approach or achieve their maximums in the off-peak and nighttime hours. But beyond that, it seems clear that certain sections of streets operate in different ways. In some cases, a street will appear to be jammed all day, with only a brief respite in the middle of the night. Other streets achieve their maximum speeds for most the day, but show particular distress during the evening rush hour or even on weekends. Some of the minor routes seem to have relatively low padding all the time.

Perhaps an interactive display would be more appropriate. The author was surprised enough to learn that videos could be embedded in PDFs.
8 Interpretation and Discussion

8.1 Do the results match basic expectations?

Before becoming enamored of the compelling power of maps, we should first verify that the observations are plausibly related to some basic expectations. As ever, if they are not, there are probably errors and/or fundamental problems with the theory. The following correlation matrix [Table 2] compares 1) different variables on each network edge, and 2) a variable on one direction of bi-directed edges to the same variable on its partner going the other direction. We expect that:

1. The correlation between any variables on the two directions of a bidirected edge should be quite high. These will typically represent the two sides of the same street or set of streets, so we expect conditions to be roughly the same in both directions.

2. There should be some correspondence between the length of a segment and the average speed achieved over it, with longer segments allowing faster average speeds. Routes with infrequent stops should be expected to go faster.

3. There should be a strong correspondence between the schedule and reality, particularly in the average and maximum speeds in the schedule and average and maximum speeds observed.
Table 2: Correlation matrix showing interrelationship between schedule and observations

To address things in order, the correlations between the two directions of bi-directed edges (red, on the diagonal) directed edges do seem to be uniformly high. The two lowest numbers on the diagonal, scheduled max speed and scheduled mean speed, 0.6014 and 0.7121 respectively, may indicate something interesting about the quality of the schedule data since both comparable measures in the observed data are much more strongly correlated, as was expected. Alternatively, it may be that the way measurements were taken, from the intersections of stop clusters rather than from actual stop locations, made the observed data differ systematically from the schedule data. The high correlation between scheduled and observed lengths (0.9822), would however seem to undermine that possible explanation.
Our second assumption, that longer segments should allow higher average speeds, does not hold up well. The signs are right, but correlations ranging from 0.10 to 0.13 indicate a weak relationship at best.

The third assumption, that the schedule should closely reflect observed reality, does not seem to hold up well either. The correlations between observed and scheduled mean speeds (0.3707) and observed and scheduled max speeds (0.2650) indicate that the observations are fairly different, at least at this level, from what is in the schedule. Again, this does not seem obviously explicable by observational error because of the high correlation between observed and scheduled segment lengths.

One further bit of reassurance lies in the correlation between the number of trips scheduled to traverse an edge and the number actually observed to be doing so. These values, all above 0.9, indicate that the observations do show vehicles operating where they should be in roughly the right proportion.

8.2 Some exemplary segments

Looking at the maps, particularly the animations, a few corridors stand out for having a large amount of both padding and scheduled trips. Three such corridors which stood out are highlighted in maps 7, 8 and 9. Interestingly, they are all streets where several different routes come together in their approach to one of the transit terminals overlaying a subway station. Map 7 shows routes 34, 51, 54, 56, 100, and 103, all surface-running bus lines in mixed traffic, approaching the Eglinton Station bus terminal via Eglinton Avenue East. Map 8 shows routes 84, 101, 106 and 108 approaching the Downsview Station from the west via Sheppard Avenue West. Map 9 shows at 8 different routes on three separate approaches to two different subway stations. Six of these are bus routes and two are streetcar routes, all operating in mixed traffic.
Map 7: Eglinton Avenue East between Laird Drive and Yong Street

Map 8: Sheppard Avenue West from Allen Road to Keele Street
to see the padding in the final segment before or after the station. However, the padding appears to be distributed somewhat evenly across the whole block approaching those stations. These are not only lines approaching subway stations, they are lines approaching their own ends, and it seems equally possible that padding in the final portion of a line might exist for either purpose. It is also the case that development in Toronto is generally much denser around subway stations [Figure 13], so it may be that padding correlates closely with some measure of density.

This is because the data does not model dwell times. If it modeled dwell times we would expect to see it in those instead.

Map 9: Broadview Avenue and Pape Avenue
Figure 13: Satellite image of the section of East Eglinton from Map 7, going from lower left to upper right. Source: Bing.
9. Concluding remarks

The initial results of this research are ambiguous and extremely interesting. Schedule padding is a critical part of all scheduled transit operations, but it is rarely discussed, never defined, and there is no standard way of measuring it. This paper makes an attempt to imagine a transit system without schedule padding, to subtract it from the actual schedule, and to examine the residue for signs. But we do not suppose that this is a sure method of effectively surfacing schedule padding in all situations, or even in the case study. An unpadded system is almost an impossibility, and trying to define one is like trying to define a doctrine of human rights: nice in theory, but never practical when the war is on – or even when the market is open. Still, we should try to have some standards. Further, for the Toronto Transit Commission, the GTFS schedule data does not seem to have been totally adequate to the job asked of it. Our limited knowledge of other city’s GTFS data indicates that many other published schedules may be of a similar quality. Since GTFS data is generally the best source of information on a system’s schedule and the schedule must be (one half of) the final word on schedule padding, the researcher of sub-optimal transit performance is presented with a dilemma.

9.1 Lessons learnt and directions for future research

There are a few distinct weaknesses in this work that are worth noting in the hope that others may improve upon them. Since weaknesses do not often comport themselves well to the exigencies of narrative, we will list them in no particular order.

• Overlapping and redundant edges: One of the weaknesses of GTFS as a data source is that it does not recycle basic street geometry. Stops are reused, but each route has several different “shapes” that bear no explicit relation to each other. These shapes were designed to ease map rendering on the computers and web browsers of transit users rather than to allow people to...
apply the methods of graph theory to an analysis of schedules. To approximate a graph where edges represent shared portions of a common street, we ignored the shapes completely and drew straight edges between clusters of stops. The clustering helped enormously in reducing visual clutter, but there were still many instances of edges that almost totally overlapped one another. This was a particular problem where express and local services share the same street, and even essentially the same route, but serve different sets of stops. A data model for GTFS, or one to which it could be linked, that made explicit and recycled the geometry of the actual streets and that thereby made it clear that vehicles were or were not operating on a street, would have simplified the visualization enormously. One such data model is used by OpenStreetMap, where basic street segments can be organized into 'relations' that define a transit route that operates over many such segments [36].

The method for selecting a minimum time necessary to traverse an edge was unsophisticated. While the spatial resolution of the NextBus data was good when it was working, the temporal resolution did not allow any obvious ways of detecting erroneous location reports when they happened to coincide roughly with the expected route. As we discussed, these positional jumps could result in realistic, but not real, high-speed observations where perhaps no such thing ever existed. A much higher temporal resolution is easily technically possible (for transit agencies to provide) and would allow more ways of detecting errors.

The model described here does not account for dwell time, which obviously must exist and "Dwell time" is the time a vehicle spends dwelling, stationary, at a stop. It is usually distinguished from "travel time" or something like that, but the distinction begins to break down when one considers that most transit vehicles need not stop if no one is waiting. What portion of any extra time spent accelerating before and after a stop is allotted to which category of time?
must take up a large proportion of the total time a vehicle spends in service. Interestingly, TTC’s GTFS data itself did not account for dwell time either, though GTFS is designed to do so. This is not completely problematic though, since one source of random delay is the need to stop (or not) for passengers who may (or may not) be waiting to get on or off. Still, to talk of average speeds without acknowledging that much time may be spent without moving at all may miss something important. A model that only measured dwell-time variability might be quite interesting for its presumed implications for schedule padding.

• It may be that transit lines operate in sufficiently different environments at different times of day that something like a rolling ideal transit time might better be established than the fixed maximum speed that we sought to observe. This could be particularly true in cases where there are enough transit vehicles that they cause significant congestion for each other, apart from any other non-transit traffic. We expect though that this would require a longer period of observation to obtain a reasonably large sample for each part of the newly enlarged problem domain.

• This analysis depends enormously on the quality of the schedule data, and that schedule data seems to be imperfect in some important ways, at least as provided by the TTC. First, it appears that scheduled times for any given route were established between major intersections rather than between stops. This is distinctly visible in the scheduled versus observed maps, where the scheduled maps show more discrete speed changes at major intersections rather than near them as in the observed values. This makes sense, and fits with the way planners might be expected to chop up and adjust a schedule before computers were available to do the work. We seem to have incorrectly assumed that since the data was available at the level of stop-to-stop edges, it would be adjusted at that level. It may rather...
the case that the stops between timepoints are ignored completely in the planning process only later to be filled out by something like an automatic linear interpolation. The total neglect of dwell-time expectations would also seem to indicate that this may be the case. If this is indeed the way that the GTFS data are generated, agencies may be hurting themselves and their passengers in subtle ways. If passenger expectations are based on schedules that purport to be accurate to the second, it could be that many people are not able to time their arrivals as accurately as they think they can if the data does not actually resolve to anywhere near that level. This concern is compounded by the fact that any number of measures of on-time performance can be and are derived from GTFS data at the level of individual stops. It is fairly easy to imagine how historic vehicle location data, such as we had to derive from a real-time API, could be used adjust padding and schedule times generally in an automatic or semi-automatic way. Such subtle improvements, if we are anywhere close to correct with our initial assumption, could potentially yield great benefits in on-time performance.

9.2 In Summary

All transit services can be thought of as existing on a spectrum of removal from sources of potential delay. A subway or 'metro' line is much less likely than a streetcar operating in mixed traffic to be delayed in rush hour by heavy traffic or high passenger loads. Such a metro might make every stop on every trip and spend a relatively fixed amount of time both dwelling at stops and traveling between them. The limits to speed that such transit encounters are much firmer than those encountered by the streetcar, which often has unpredictable stop patterns, let alone dwell and travel times. Such predictably unpredictable delays are accounted for in schedules that then, themselves, limit the speed of the vehicles still further. These scheduled delays are difficult to see once they have been put in place because they can be managed discretely by the driver, moment to moment.
and because they help to establish normal travel times which few people see reason to question.

In the last decade, data has begun seeping out of transit agencies around the world. Detailed and standardized schedule data (GTFS), particularly when verified with actual systematic observations (real-time location APIs or other sources) can provide a way for people outside of those agencies to start to assess some of the trade-offs that their public servants are making on their behalf. One such trade-off is the one between how late vehicles generally are and how fast they can go. Another is between the time and money lost every day to random delay, and the money and political capital that would need to be spent to avoid it outright. Technological and political solutions to unpredictable delay exist and their costs are well documented. What is not so well known is the opportunity cost of not implementing them.

The delay caused by traffic and other random events and by the schedule's expectation of them is very real. If our preliminary measurements in one city are any indication, as much as 30% of all scheduled time may be spent preemptively coping with predictable but perhaps mostly avoidable delays. Of course, actually making improvements on that scale is not a realistic goal in the short term in almost any case, but such measures may provide a useful starting point for discussions about how improvement can proceed, and what sorts of interventions can be most effective.
References:


21: Aaron Antrim & Sean J. Barbeau, *The many uses of gtfs data–opening the door to transit and multimodal applications*, Location-Aware Information Systems Laboratory at the University of South Florida, 2013.

information for public transit


sparse and noisy observations.


Appendix 1: Code

Script, in Python, for requesting vehicle locations:

```python
# working entirely with the nextbus feed:
# http://www.nextbus.com/xmlFeedDocs/NextBusXMLFeed.pdf
import xml.etree.ElementTree as ET
# XML parsing
import requests
# HTTP
import psycopg2
# DB interaction
import threading
# timer

# DB connection
conn_string = "host='' dbname='' user='' password=''"
conn = psycopg2.connect(conn_string)

# API business
URL = 'http://webservices.nextbus.com/service/publicXMLFeed'
deflate_header = {'Accept-Encoding':'gzip, deflate'}
request_time = 0

# go ahead and get the starting track id from the table
cursor = conn.cursor()
cursor.execute('SELECT MAX(tid) from nb_vehicles')
current_tid = cursor.fetchone()[0] + 1

# one big function
def get_recent_vehicles():
    global current_tid
    global request_time

    # get updated vehicle locations
    payload = {
        'command': 'vehicleLocations',
        'a': agency,
        't': request_time
    }

    # if not connected, skip the rest of the function and try again after the
    # specified delay
    try:
        vr = requests.get(URL, params=payload, headers=deflate_header, timeout=3)
    except requests.exceptions.ConnectionError:
        print 'not connected to the Internet'
        return
    except:
        print 'unknown error, probably "connection reset by peer"'
        return

    vxml = ET.fromstring(vr.text)
    # get the serve time
    request_time = int(vxml.find('.//lastTime').attrib['time'])
    vehicles = vxml.findall('.//vehicle')
    for v in vehicles:
        # see what variables are set
        vid = v.attrib['id'] if 'id' in v.attrib else None
        lon = v.attrib['lon'] if 'lon' in v.attrib else None
        lat = v.attrib['lat'] if 'lat' in v.attrib else None
        direction = v.attrib['dirTag'] if 'dirTag' in v.attrib else None
        rid = v.attrib['routeTag'] if 'routeTag' in v.attrib else None
        secs_since_report = int(v.attrib['secsSinceReport']) if 'secsSinceReport' in v.attrib else None
```

76
if vid not in tracks:
    # vehicle is new to us
    tracks[vid] = {
        't': current_tid,  # track
        'r': rid,  # route
        'd': direction,  # direction
        'u': request_time  # update time
    }
    current_tid = current_tid + 1
elif (request_time - tracks[vid]['u'] > 60000) or (tracks[vid]['r'] != rid) or (tracks[vid]['d'] != direction):
    # track needs to be re-started for one of three reasons
    tracks[vid] = {
        't': current_tid,  # track
        'r': rid,  # route
        'd': direction,  # direction
        'u': request_time  # update time
    }
    current_tid = current_tid + 1
else:
    # carry on
    tracks[vid]['u'] = request_time
insert each vehicle
    cursor.execute(
        "INSERT INTO nb_vehicles (vid, location, rid, did, report_time, tid) VALUES (%s, ST_SetSRID(ST_MakePoint(%s, %s),4326), %s, %s, TIMESTAMP WITH TIME ZONE 'epoch' + %s * INTERVAL '1 millisecond', %s)",
        (vid, lon, lat, rid, direction, request_time - secs_since_report * 1000, tracks[vid]['t']))
    conn.commit()  # after finished looping, commit and close cursor
print 'i:', len(vehicles), ', d:', delay, ', t:', current_tid
# the timer function!
def tloop():
    # timer reset
    threading.Timer(delay, tloop).start()  # function name
    # timer action
    get_recent_vehicles()
# dictionary of vehicle tracks
tracks = {}
# prompt user input
agency = raw_input('agency tag: ')
delay = int(raw_input('delay (seconds): '))
initial call
tloop()

Python script for comparing real-time traversal of edges:


go through one segment at a time

and each track that matches it

and calculate statistics for the segment

import psycopg2, numpy
import warnings
warnings.simplefilter("error")

# DB connection
conn_string = "host='' dbname='' user='' password=''
conn = psycopg2.connect(conn_string)
c1 = conn.cursor()
c2 = conn.cursor()
c3 = conn.cursor()
c4 = conn.cursor()

# interpolate a time on a segment of a trip
# will be called many times, so should be
# as efficient as possible

def interpolate(time1, time2, m1, m2, i):
    # time1, time2 -- epoch times in seconds
    # m1, m2 -- measures between 0 and 1 along a full track
    # i -- intersection, between or equal to m1, m2
    if i1 == m1:
        return t1

    percent_of_segment = (i - m1) / (m2 - m1)
    additional_time = percent_of_segment * (time2 - time1)
    return t1 + additional_time

# clear the columns we're about to fill
# print "clearing the flag field"
#c1.execute("UPDATE gtfs_ttc_segments SET flag = TRUE")
#conn.commit()

# get the segment ids which we'll query one at a time below
print 'getting a list of segments to use...'
c1.execute("SELECT
 ARRAY_AGG(uid) AS idset,
 edge_id,
 SUM(tracks_used)
 FROM gtfs_ttc_segments
 WHERE flag = TRUE
 GROUP BY edge_id
 ORDER BY SUM(tracks_used) ASC
 LIMIT 20000")
print 'starting on the first one now...'

# SEGMENT level
for r in c1:
    uid = r[0][0]
    previous_tracks = r[2]

    print '
','previously', previous_tracks
    78
if len(r[0]) == 1:
# directed segment
print 'id:',uid
directed = True
else:
# bi-directed segment
uid2 = r[0][1]
print 'id:',uid,uid2
directed = False

# create variables to hold the lists
all_times1, all_times2 = [], []
all_speeds1, all_speeds2 = [], []
all_errors1, all_errors2 = 0.0, 0.0
all_tids1, all_tids2 = [], []
all_distances1, all_distances2 = [], []

# get all of the matched tracks on uid

for r2 in c2:
    # first assign variables to decent names
    times = r2[0]
    measure = r2[1]
    km = r2[2]
    i1 = r2[3]
    i2 = r2[4]
    tid = r2[5]
    distance = abs(i2 - i1) * km

    # skip track if going the wrong way on a one-way
    if i1 > i2 and directed:
        continue

    # now loop through each point on the current track looking for the intersections
    first = True
    for m, t in zip(measure, times):
        # loop through pairs
        if first:
            first = False
            # 79
m1, t1 = m, t  # m should always == 0 here
continue
m2, t2 = m, t
# now we're in the proper part of the loop
if m1 <= i1 < m2:
    # first intersection is between these
    atime1 = interpolate(t1, t2, m1, m2, i1)
if m1 <= i2 < m2:
    # second intersection is between these
    atime2 = interpolate(t1, t2, m1, m2, i2)
    # set for the next iteration
m1, t1 = m, t
# now we have our times and need to verify that they are reasonable
try:
    atime2 - atime1
except TypeError:  # in case one or more times is not found
    if i1 < i2:
        all_errors1 += 1
    elif not directed:
        all_errors2 += 1
    continue
span = abs(atime1 - atime2)
# reset these variable for next iteration
atime1 = atime2 = None
# check that the calculated time is plausible before storing it
min_time = distance * (3600/120.0)  # 120kmph
max_time = distance * (3600/0.1)  # 0.1kmph
if min_time < span < max_time:
    speed = distance / (span / 3600)
    if i1 < i2:  # is on uid:
        all_tids1.append(tid)
        all_distances1.append(distance)
        all_times1.append(span)
        all_speeds1.append(speed)
    elif not directed:
        all_tids2.append(tid)
        all_distances2.append(distance)
        all_times2.append(span)
        all_speeds2.append(speed)
else:
    if i1 < i2:
        all_errors1 += 1
    elif not directed:
        all_errors2 += 1
# print at end of segment level loop
all_times1.sort()
all_speeds1.sort()
if not directed:
    all_times2.sort()
    all_speeds2.sort()
print '# tracks: ', c2.rowcount
print '# used1:  ', len(all_tids1)
print '# used2:  ', len(all_tids2)
print 'errors:   ', all_errors1 + all_errors2
if len(all_tids1) > 1:
    print '% error:  ', (all_errors1 + all_errors2) / (len(all_tids1) + len(all_tids2)) * 100.0
    print 'mean span:', numpy.mean(all_times1)
    print 'mean dist:', numpy.mean(all_distances1)
    print 'span std: ', numpy.std(all_times1)
    print 'kmph:     ', numpy.mean(all_speeds1)
if len(all_times1) > 1:
    # insert into DB
    c3.execute("UPDATE gtfs_ttc_segments SET ")
mean_span = %s * interval '1 second',
span_std = %s * interval '1 second',
spans = %s,
tracks_ignored = %s,
speeds_kmph = %s,
mean_kmph = %s,
mean_length = %s,
lengths = %s,
tids = %s,
tracks_used = %s,
flag = %s
WHERE uid = %s;

# only for bidirected do we do this twice
if not directed:
    if len(all_tids2) > 1:
        c3.execute("UPDATE gtfs_ttc_segments SET
mean_span = %s * interval '1 second',
span_std = %s * interval '1 second',
spans = %s,
tracks_ignored = %s,
speeds_kmph = %s,
mean_kmph = %s,
mean_length = %s,
lengths = %s,
tids = %s,
tracks_used = %s,
flag = %s
WHERE uid = %s;")
WITH buffers AS (
    SELECT
        ST_Buffer(the_geom, 30) AS buffer
    FROM gtfs_ttc_stops
    WHERE NOT subway_only
),
    dumper AS (
        SELECT
            ST_Dump(ST_Multi(ST_Union(buffer::geometry))) AS d
        FROM buffers
    )
    SELECT
        (d).geom::geography AS the_geom,
        ST_Centroid((d).geom)::geography AS centroid
    INTO gtfs_ttc_merged_stops
    FROM dumper;

ALTER TABLE gtfs_ttc_merged_stops
ADD COLUMN cluster_id serial,
ADD COLUMN stop_ids varchar[];

-- add stop_id's to cluster
WITH aggs AS (
    SELECT
        tm.cluster_id,
        array_agg(stop_id) AS the_stops
    FROM gtfs_ttc_merged_stops AS tm
    JOIN gtfs_ttc_stops AS s
    ON ST_Intersects(s.the_geom,tm.the_geom)
    WHERE NOT s.subway_only
    GROUP BY tm.cluster_id
)
UPDATE gtfs_ttc_merged_stops AS thm
SET stop_ids = the_stops
FROM aggs
WHERE thm.cluster_id = aggs.cluster_id;

--ALTER TABLE gtfs_ttc_stops ADD COLUMN cluster_id integer;
UPDATE gtfs_ttc_stops
SET cluster_id = NULL;
UPDATE gtfs_ttc_stops AS s
SET cluster_id = cs.cluster_id
FROM gtfs_ttc_merged_stops AS cs
WHERE ARRAY[s.stop_id] <@ cs.stop_ids;

--ALTER TABLE gtfs_ttc_stop_times ADD COLUMN cluster_id integer;
UPDATE gtfs_ttc_stop_times
SET cluster_id = NULL;
UPDATE gtfs_ttc_stop_times AS st
SET cluster_id = s.cluster_id
FROM gtfs_ttc_stops AS s
WHERE s.stop_id = st.stop_id;

SQL for deriving tracks from reported vehicle locations:
--delete records of small tracks
WITH sub AS (
    SELECT
        tid,
        COUNT(DISTINCT location) AS count
    FROM nb_vehicles
    GROUP BY tid
)
DELETE FROM nb_vehicles AS v
USING sub AS s
WHERE s.tid = v.tid AND count < 5;

-- prep the table and do the big select into
DROP TABLE IF EXISTS nb_tracks;
SELECT tid, rid, did, COUNT(*) AS count, array_agg(EXTRACT(EPOCH FROM report_time) ORDER BY report_time ASC) AS etimes, array_agg(report_time ORDER BY report_time ASC) AS times, array_agg(uid ORDER BY report_time ASC) AS stop_ids, array_agg(location ORDER BY report_time ASC) AS stops, ST_MakeLine(location::geometry ORDER BY report_time ASC) AS the_geom, -- leave these here for later... NULL::real AS km, NULL::real[] AS m, NULL::real AS azimuth INTO nb_tracks FROM nb_vehicles GROUP BY tid, rid, did;

-- add length measurement
UPDATE nb_tracks SET km = ST_Length(the_geom::geography, FALSE)/1000;
DELETE FROM nb_tracks WHERE km < 0.3;

-- add length measurements
WITH sub AS (SELECT t.tid, array_agg(ST_LineLocatePoint(t.the_geom::geometry, v.location::geometry) ORDER BY report_time ASC) AS m FROM nb_tracks AS t JOIN nb_vehicles AS v ON t.tid = v.tid GROUP BY t.tid) UPDATE nb_tracks SET m = sub.m FROM sub WHERE nb_tracks.tid = sub.tid;

-- azimuth of start -> end points in degrees
UPDATE nb_tracks SET azimuth = ST_Azimuth(ST_StartPoint(the_geom)::geography, ST_EndPoint(the_geom)::geography) / (2*pi()) * 360;
Appendix 2: Example padding calculation

It will be useful to give a less abstract example of the way schedule padding has been calculated throughout this thesis. To that end, a typical edge from the TTC network graph has been selected at random, and the process of isolating its padding will be described below.

The example edge is pictured on the right and summarily described in the table below. After the table follows a histogram showing the distribution of speeds observed on each segment. From each of these distributions, an observation was taken from the start of the 9th decile (90th percentile) and used as the selected minimum speed. These values are shown in the table.

<table>
<thead>
<tr>
<th>Traveling Northwest</th>
<th>Traveling Southeast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routes in operation</td>
<td>24, 324</td>
</tr>
<tr>
<td>Trips scheduled per week</td>
<td>1,287</td>
</tr>
<tr>
<td>Tracks matched to edge</td>
<td>1,674</td>
</tr>
<tr>
<td>Mean observed length</td>
<td>0.311 km</td>
</tr>
<tr>
<td>Mean observed speed</td>
<td>23.612 kmph</td>
</tr>
<tr>
<td>Mean scheduled speed</td>
<td>25.425 kmph</td>
</tr>
<tr>
<td>Service hours scheduled</td>
<td>13:42:27 (13.708 hours)</td>
</tr>
<tr>
<td>Selected minimum speed</td>
<td>47.05 kmph</td>
</tr>
<tr>
<td>Time required at selected minimum</td>
<td>23.801 seconds</td>
</tr>
<tr>
<td>Padding per km at selected minimum</td>
<td>46.75 seconds</td>
</tr>
<tr>
<td></td>
<td>24, 144, 324</td>
</tr>
<tr>
<td></td>
<td>1,354</td>
</tr>
<tr>
<td></td>
<td>1,710</td>
</tr>
<tr>
<td></td>
<td>0.361 km</td>
</tr>
<tr>
<td></td>
<td>21.807 kmph</td>
</tr>
<tr>
<td></td>
<td>22.878 kmph</td>
</tr>
<tr>
<td></td>
<td>20:48:53 (20.815 hours)</td>
</tr>
<tr>
<td></td>
<td>51.09 kmph</td>
</tr>
<tr>
<td></td>
<td>25.439 seconds</td>
</tr>
<tr>
<td></td>
<td>82.82 seconds</td>
</tr>
</tbody>
</table>

The example edge with travel in two directions
For the northwest edge, we calculate the padding in seconds (per trip, per kilometer) as follows:

\[ 46.75 = \left( 13.708 \cdot 3600 - 1287 \cdot 23.801 \right) / 1287 \]

For the southeast edge, we calculate the padding in seconds (per trip, per kilometer) as follows:

\[ 82.82 = \left( 20.815 \cdot 3600 - 1354 \cdot 25.439 \right) / 1354 \]
I, Nathan Wessel, hereby submit this original work as part of the requirements for the degree of Master of Arts in Geography.

It is entitled:
Discovering the Space-Time Dimensions of Schedule Padding and Delay from GTFS and Real-time Transit Data

Student's name: Nathan Wessel

This work and its defense approved by:

Committee chair: Michael Widener, Ph.D.
Committee member: Tomasz Stepinski, Ph.D.
Committee member: Paul Zender, M.F.A.