Assessing Transit Accessibility Unreliability and Social Equity Impact with Space-time Prisms

Luyu Liu, Adam Porr, Harvey Miller

Uncertainties in public transit systems’ travel time have been a major obstacle to make transit more accessible and reliable for commuters. Due to delays caused by traffic and road conditions, the actual accessibility derived from real-time data can be very different from the one promised by the schedule. However, very few prior accessibility studies addressed this discrepancy explicitly with time geography methods. In this paper, we use a well-established time geography method, space-time prism (STP), to measure the *accessibility reliability*. Accessibility reliability is defined as the difference between STP derived from retrospective real-time data and STP derived from schedule data. The methods will use two mobility datasets of large volumes: General Transit Feed Specification (GTFS) real-time data, which produce retrospective real-time STP, and GTFS schedule data, which produce scheduled STP. We will also investigate the reliability measure’s connections to social equity factors, such as different social, demographic, and economic factors. The paper will conduct a case study in the Central Ohio Transit Authority (COTA) bus system, a public transit agency in Columbus Ohio. The analysis will focus on the spatial and temporal patterns of the reliability measure from 2018 – 2021 across the city of Columbus, especially the changes before and during the COVID-19 pandemic. This can provide insights about possible impacts of the pandemic on the reliability of the transit accessibility in different communities. All the analyses and results will be visualized in a public web-based platform. This paper provides a scalable time-geography approach to gauge the reliability of transit accessibility with very large datasets; some results can also reveal new empirical patterns of transit accessibility’s impact on social equity.

1. Introduction

As transit planning’s focus shifting from mobility to accessibility (Banister, 2008), accessibility becomes a crucial determinant to promote transit policies and foster transit use. Accessibility measures the ease of reaching destinations: a transit system with higher accessibility can have better usability and user experience for transit users; therefore, authorities aim to increase accessibility as one of their priorities. As public schedule dataset like General Transit Feed Specification (GTFS) becomes more available, researchers and transit planners start to calculate accessibility with static scheduled data. Standard data formats introduce a convenient way to access the transit system’s network information, including stops as nodes, trips as links, and scheduled travel time as travel cost. This has greatly helped us understand the accessibility of transit system.

However, accessibility of transit systems is highly dynamic and time-dependent due to traffic condition and delay that the actual performance can be very different from the schedule (Park, Mount, Liu, Xiao, & Miller, 2020). Many prior studies have examined the possible problems of using scheduled data for modelling and benchmarking (Bills & Carrel, 2021; Wessel, Allen, & Farber, 2017; Wessel & Farber, 2019): actual accessibility can have significant deviations from the accessibility calculated from schedule data. There are several factors that contribute to this deviation: first, road conditions and congestions can slow the vehicles down and human operators can leave the stops earlier than the scheduled time. Second, only travel time at timepoint stops is explicitly defined in the official timetables of many transit systems; travel time at non-timepoint stops is derived from interpolation, which is not strictly followed in practice.

This paper introduces a time geography approach based on space-time prism (STP) to understand *accessibility reliability*, retrospective accessibility’s deviation from scheduled accessibility. This measure represents the difference between the actual physical accessible space and the expected accessible space calculated from schedule data during the same time budget; the aggregated version of this measure can also show consistency and quality of the transit service, which is vital for administrative and planning purposes. We use GTFS schedule and GTFS real-time data, then produce retrospective real-time STP and scheduled STP, and finally derive the difference between the two STPs. The origin-level deviation measure has been reported to be non-uniform (Wessel & Farber, 2019) and non-random (Wessel et al., 2017); however, few studies examined its connections to social equity factors, such as different social, demographic, and economic factors. This paper fills in this blank by using geostatistics model to investigate the potential correlations.

We conduct a case study in the Central Ohio Transit Authority (COTA) bus system, a public transit agency in Columbus Ohio. The analysis focus on the spatial and temporal patterns in different levels from 2018 – 2021 across Columbus, especially during the COVID-19 pandemic. This paper provides a scalable time geography approach to measure the reliability of transit accessibility with very large datasets and investigate its implications on social equity.

1. Background
2. Time-independent network based on static data
3. Time-dependent network based on static data. unreliability research
4. Time-dependent network based on dynamic data
5. Method

We introduce the method to calculate the accessibility unreliability in this section. We first introduce the two main transit datasets we use in this paper. Then, we demonstrate the time-dependent Dijkstra algorithm to calculate the two versions of space-time prisms. Last, we propose the data source and geostatistic model to investigate the connection between accessibility reliability and other social and demographic factors.

* 1. Data source

We use General Transit Feed Specification (GTFS) data as the main data source for time geography analyses in this paper. GTFS is a data standard that let transit authorities publish transit data and developers/researchers consume the data (Google Developers, 2020). GTFS includes two parts: GTFS static and GTFS real-time data.

We first use GTFS static data to acquire the transit schedule data. GTFS static data has been the *de facto* standard for transit schedule and associate geographic data (Google Developers, 2020). The data are organized into several separate relational tables, which define a transit system’s stops, trips, routes, arrival and departure time, and other schedule information (Google Developers, 2020). GTFS static data are very convenient for researchers and developers to obtain complete, systematic, and official schedule data directly from the transit authorities.

We collect both GTFS static and real-time trip update data from the official application programming interface (API) of the Central Ohio Transit Authority (COTA) from February 2018 to July 2020 (Central Ohio Transit Authority, 2021). We save the GTFS static data whenever there are any changes in the dataset. Besides the major schedule adjustments such as the route redesign in 2018 (Lee & Miller, 2018; Schmitt, 2018) and COVID-19-related adjustments in 2020 (Liu, Miller, & Scheff, 2020) as well as the three seasonal schedule adjustments in every January, May, and September, we also record every minor change that happen day by day.

We use GTFS real-time data to obtain the real-time or retrospective time in the system. GTFS real-time extension is a feed specification for transit authorities to provide real-time updates about their fleet to the public (Google, 2021). GTFS real-time data include two main datasets: *trip update*, which contains the expected arrival/departure time of each trip at each stop in the transit system, and *vehicle position*, which is similar to automatic vehicle location (AVL) data and shows the location of active vehicle in the system. GTFS real-time data are broadcasted by transit authorities at a regular interval from 10 second to 90 seconds (Liu & Miller, 2020). We derived the actual arrival time of each trip at each stop from trip update data for: 1) it is very convenient and has relatively high accuracy; 2) most trip planning apps, which are the source of transit information for many passengers, are using trip update as the source for arrival time.

We also collect GTFS real-time trip update data from COTA’s official API. We collect trip update feeds at the interval of 60 seconds, which is a common frequency for US transit systems (Liu & Miller, 2020). The timespan covers February 2018 to July 2021 and the total data volume exceeds 1 terabyte. It is challenging to maintain such a huge dataset and conduct scientific calculations in traditional relational database environment; therefore, we use noSQL and MongoDB database to adapt to these difficulties.

* 1. Space-time prism and travel time

As we introduced in the last section, space-time prism is a well-established time geography method to measure accessibility in public transit systems (Miller, 1991; Wu & Miller, 2001). In practice, we calculate the shortest travel time between the origin stop to other stops to produce the contour lines of different travel time at an equal interval and the STP.

* + 1. Time-dependent routing problem

It is hard to obtain accurate travel times in a transit network, even with a complete archive of retrospective arrival times. A major reason is because transit networks are *discontinuous* and *time-dependent* (Gendreau, Ghiani, & Guerriero, 2015; Wang, Yuan, Ma, & Wang, 2019). Unlike private vehicle or pedestrian network, a user cannot move in the network unless there is an available vehicle, which is scheduled to arrive at only some specific time points. Therefore, the network costs of a transit work can vary depending on the passenger’s arrival time at the origin node of a transit link. This variation also applies to components of the travel time, including wait time and in-vehicle time.

The time-dependent routing problem can be generally categorized into two models: deterministic and stochastic model (Gendreau et al., 2015). Stochastic models use a stochastic factor to measure or predict the time-varying travel time. They are more useful to capture the randomness caused by congestion, weather, accidents, and road maintenance (Gendreau et al., 2015); however, due to the random nature of these models, the results include many uncertainties, even with deterministic and retrospective travel time records. Because we collected all the arrival times at all the stops and aim for more accurate travel time, we use a deterministic approach to address the time-dependent routing problem.

We use a modified Dijkstra algorithm with dynamic costs to solve the problem. Dijkstra algorithm is a classic and efficient algorithm to solve the shortest path routing problem (Golden, 1976). It uses a greedy heuristic search strategy to find the shortest path from the origin node to every other nodes (Xie, Zhu, Yan, Yuan, & Zhang, 2012), which is very useful and efficient to calculate the STP.

However, traditional Dijkstra algorithm’s correctness is based on non-negative static costs, which time-dependent transit networks do not satisfy. Due to *passing*, when a later start time may result in an earlier arrival time (Gendreau et al., 2015), the results generated by Dijkstra algorithm with dynamic costs may not be the global optimal answer. Therefore, many prior studies introduced *no-passing* or *first-in-first-out (FIFO)* rule to make Dijkstra algorithm compatible with the time-dependent requirement (Ahn & Shin, 1991; Ichoua, Gendreau, & Potvin, 2003). FIFO rule assumes a vehicle leaving the origin stop earlier will never arrive later at the destination stop than another vehicle. FIFO rule is a prerequisite to use Dijkstra to calculate routing problem in a transit system.

Therefore, we test if COTA system satisfies the FIFO rule. We calculate whether a subsequent bus in a same route

Traditional methods focus on simulation with stochastic process to simplify the process to relax the very high calculation complexity. For example,