University of California Transportation Center
UCTC-FR-2011-14

Mobile Transit Trip Planning with Real–Time Data

Jariyasunant, Jerald, Daniel B. Work,
Branko Kerkez, Raja Sengupta,
Steven Glaser and Alexandre Bayen
University of California, Berkeley
September 2011

Mobile Transit Trip Planning with Real–Time Data

Jerald Jariyasunant, Corresponding Author, jjariyas@berkeley.edu
Department of Civil Engineering, University of California, Berkeley
621 Sutardja Dai Hall Berkeley, CA 94720
Tel: (510) 642-2468

Daniel B. Work, dbwork@berkeley.edu
Department of Civil Engineering, University of California, Berkeley
621 Sutardja Dai Hall Berkeley, CA 94720
Tel: (510) 642-2468

Branko Kerkez, bkerkez@berkeley.edu
Department of Civil Engineering, University of California, Berkeley
621 Sutardja Dai Hall Berkeley, CA 94720
Tel: (510) 642-2468

Raja Sengupta, sengupta@ce.berkeley.edu
Department of Civil Engineering, University of California, Berkeley
640 Sutardja Dai Hall Berkeley, CA 94720
Tel: (510) 642-9540 Fax: (510) 643-8919

Steven Glaser, glaser@berkeley.edu
Department of Civil Engineering, University of California, Berkeley
151 Hearst Memorial Mining Buiilding Berkeley, CA 94720
Tel: (510) 642-1264 Fax: (510) 642-7476

Alexandre Bayen, bayen@ce.berkeley.edu
Department of Civil Engineering, University of California, Berkeley
642 Sutardja Dai Hall Berkeley, CA 94720
Tel: (510) 642-2468 Fax: (510) 643-5264

Abstract

In this article, we describe the development of a *transit trip planner* (TTP) for mobile devices called Transitr, and evaluate its performance. The system predicts the shortest paths between any two points in the transit network using real-time information provided by a third party bus arrival prediction system, relying on GPS equipped transit vehicles. Users submit their origin and destination through a map-based iPhone application, or through a JavaScript enabled web browser. A server implementing a dynamic K-shortest paths algorithm with predicted link travel times returns personalized route directions for the user, displayed on a map. To assess the optimality and accuracy of the predicted shortest paths, an a posteriori comparison with a schedule-based transit trip planner and the GPS traces of the transit vehicles is performed on six-hundred origin destination pairs in San Francisco. The results show that routing using the predicted bus arrivals marginally increases the accuracy of the total travel time and the optimality of the route. Suggestions to improve the accuracy and optimality using real-time information are proposed.

1 INTRODUCTION

Mobile phones equipped with GPS and Internet access are promising platforms upon which future transportation information will be shared and collected. With these devices, real-time monitoring of the transportation system will become not only feasible, but ubiquitous. Numerous emerging traffic monitoring applications use vehicle probe data collected from on-board GPS devices to reconstruct the state of traffic (for example, velocity or density maps). This information is used to predict travel time of vehicles in the transportation network. Some emerging examples include *Mobile Millennium* [29, 16], *CarTel* [10], *JamBayes* [17], *TrafficSense* [22], and systems for surface street estimation [31].

The Internet has expanded these capabilities significantly, with various mapping providers (i.e. Google and Navteq) merging static and real-time traffic data to help drivers make the most of their commute. Yet, relatively little attention has been paid to public transportation and real-time data applications. Services such as 511.org and Google Transit allow users to plan public transit trips by generating routes based on static scheduling data. A multi-modal trip planning system was developed by Rehrl et al. [26] to address the increased complexity and lack of information required by travelers. Their platform includes directions for public transit based on published schedules, as well as transfer directions between different transportation modes. An open-source multi-modal trip planner, Graphserver, has also been developed by the online transit developer community to address the same issues [13]. However, buses and trains do not always run on time. Some transit agencies, TriMet in Portland, Chicago Transit Authority, Bay Area Rapid Transit, and King County in Seattle have responded by releasing real-time bus arrival feeds online, allowing application developers to use this data in new, novel ways.

Transitr is a real-time public transit trip planning system accessible on mobile devices designed to use this information. It combines a user's geographic location with real-time transit information provided by transit agencies to determine the fastest route to a desired destination. It fuses real-time data feeds with the existing technology of schedule-based *transit trip planners* (TTPs) currently available online. To the best of our knowledge, the research is the first instantiation and evaluation of a real-time TTP which uses the predicted bus arrival feeds to route users on mobile devices.

This article is organized as follows. In Section 2 the challenges and previous work with real-time information in transit. In Section 3, we describe the system architecture of Transitr, which has been deployed to serve transit networks in two metropolitan areas. We describe the dynamic K-shortest paths algorithm with predicted link travel times in Section 4. Using data from hundreds of trips planned in San Francisco, we evaluate the performance of Transitr in comparison with a schedule-based TTP in Section 5. From analysis of the data, we describe areas of research to improve the accuracy and optimality of the system.

2 NEED FOR REAL-TIME INFORMATION IN TRANSIT

Annual system reliability reports are published by most transit agencies to address the issue of schedule adherence. Measured at terminals and intermediate points, system reliability of a public transit network is defined as the percentage of vehicles that run on time according to schedule (up to four minutes late and one minute early) [1]. Schedule adherence in the San Francisco area is estimated to be 70% [1]. For the past two years, official *Metropolitan Transportation*

Authority (MTA) numbers for New York, show a system reliability of 80% for subways and 66% for buses. In such cases, real-time information may be used to reduce time wasted waiting for delayed transit vehicles, and it can also enable users to take faster alternative routes. Hickman and Wilson [15] analyzed the benefits of real-time information, limiting the assessment to look at travel time improvements, and concluding that real-time information systems may only lead to marginal travel time benefits from improved path decisions. Their conclusion was supported by the development and subsequent evaluation of a dynamic path choice problem, in which a mathematical optimization reflects the decision of a transit rider to board a transit vehicle based on the availability of real-time information. Mishalani [21] et al. developed an evaluation method for the value of real-time information and noted that the value of information to passengers is affected by the type of available information as well as operations characteristics. The research lead to the development of a piece-wise linear function to model passenger utility, taking into account waiting times at stops as well as projected waiting times given by an arrival time engine. Surveys conducted by Caulfield et al. [24] suggest that users are generally unhappy about the on-time performance of public transit vehicles, and that the bulk of those surveyed would use real-time information if it were available.

The need for real-time data stems from the unreliable nature of bus schedules, which a number of researchers have explored. [27, 23]. One significant problem is the the phenomenon of bus bunching, in which multiple buses on the same line arrive at a stop concurrently, followed by no subsequent buses for a significant amount of time [7, 5]. This can lead to travel delays, as some passengers experience longer wait times than predicted by the published schedules. Various control schemes have been proposed to combat this issue, the most popular of which allots slack time in the routes in a procedure known as holding [30]. By building some wait time into the schedule, a bus can wait at a stop if it is on time or ahead of schedule, or skip the waiting process if behind schedule.

The present paper does not attempt solve the bus bunching issue, but rather aims to enable commuters to make informed decisions based on the inherent variability in the system, and to quantify the discrepancy of travel time as predicted by published schedules and real-time data. This work provides a tool which enables a commuter to use real-time information to find an optimal route in the transit system. Transitr leverages bus arrival estimation engines developed by NextBus, TriMet, and other agencies which provide a real-time feed for bus arrival predictions.

Previous studies have focused on transit vehicle arrival prediction, which convert real-time bus locations to predictions for arrivals at downstream transit stops. However, none of the studies have consider applying the research into a transit trip planning tool. Previous work by Jula et al. [14] used historical and real-time data to show that travel times can be estimated confidently along arcs of dynamic stochastic networks. Ignoring correlation between adjacent arcs on a network, the authors employed a technique based on a predictor-corrector form of the Kalman filter, in which historical data were used to predict travel time, and real-time measurements were used to correct the predictions at each time instant. Shalaby and Farhan [28]used two Kalman filter algorithms for the prediction of running times and dwell times alternately in an integrated framework. Separating the bus dwell time prediction from bus running time prediction in this modeling framework captures the effects of lateness or earliness of bus arrivals at stops on the bus dwell time at those stops, and hence the bus departure from such stops. Other models were developed by Abdelfattah [2] and Chien [4] which used a large number of parameters such as live traffic volumes, speeds, and passenger counts. In the previous two papers, a very comprehensive models were developed, al-

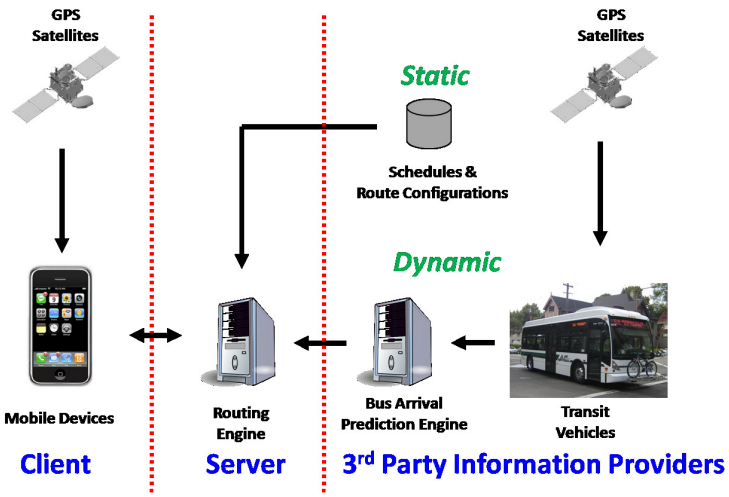

Figure 1: Architecture and system implementation of Transitr.

though many of those parameters featured in these models may be difficult to measure in real-time and consolidate for a user-application for transit trip planning. The paper does not propose new estimation algorithms; it uses the information as provided from the online real-time data feeds.

3 SYSTEM ARCHITECTURE

This section presents the architecture for Transitr, a real–time transit trip-planning system developed for mobile devices which incorporates data from a variety of sources. Unlike traditional schedule-based services, Transitr integrates a user's origin obtained by GPS destination select on a map with real-time transit vehicle arrival time estimates, allowing transit riders to plan trips from their current location. The prototype system relies on three components, described in the following sections and illustrated in Figure 1:

1. Clients: Travelers using location aware mobile devices (such as GPS or cell tower based location technology).
2. Server: A routing engine which determines the K-shortest paths between an origin and destination
3. Third Party static and dynamic information providers:

- A transit vehicle arrival prediction system generating estimates from GPS equipped mass transit vehicles
- A set of static schedules from which to build the transit network graph.

Figure 2: iPhone and web-browser implementation. Markers on screen indicate transfer points, and clicking on the markers or swiping the bar at the bottom of the screen reveals more information, such as the name/location of the stop and waiting times. The arrows on the top right of the touch screen can be used to toggle between different routes, while an information panel displays the total travel time for a selected route.

3.1 Client Side: Mobile device implementation

Transitr was developed on two client-side platforms: the iPhone (programmed in Objective C), and JavaScript enabled web-browsers. The two implementations feature a user-interface overlayed on top of a map, which eases the use of mass transit without requiring significant knowledge about a particular geographic area. Users select their origin and destination, either by using the mobile device's on-board GPS, tower based triangulation, or by manual entry into the device. The origin and destination points are geo-coded into latitude and longitude points and submitted as a query to the server.

Upon receiving the above request, the server creates an XML response which contains information for the five fastest routes. XML provides a simple, open, and extensible format to encode pertinent information about a user's trip, including such as walking directions, location of stops and transfers, and the duration of the trip. Although the application has been developed on two platforms, the same XML feed can be generated in response to a request from any mobile device or phone, thus making the underlying technology portable to multiple platforms.

3.2 Server Side: Routing Engine

The web server contains a routing engine written in Java and deployed as a web application using Java Server Pages. The routing engine determines the optimal routes by performing a database look-up on a set of feasible routes, which are generated from static schedules and route configuration information. The routing engine then communicates with external servers to obtain feeds for real-time bus arrival predictions. The construction of the optimal routes is described in detail in Section 4. The routing engine was implemented on a Linux version 2.6 server, with a QuadCore 64-bit processor, 1024 MB RAM, 400GB of storage, a MySQL database infrastructure, and a Jetty 6 web application server. Tests of the application server were conducted in New York City during the ITS World Congress on November 14-20th as part of the Federal Safe Trip 21 program. Over

five days, approximately 1000 transit riders used the iPhone application to plan trips around Manhattan and the five Burroughs with requests processed within five seconds. Additional load testing and benchmarking of server performance revealed that 72% of that process time is due to communication with the 3rd party real-time data feed. An additional experiment of the first prototype of the system was performed and filmed in Berkeley, CA, which is available online at YouTube [32].

3.3 Third Party Information Providers

3.3.1 *Static Information*

Transitr's routing engine requires a transit agency's static schedules and route configuration. With the growth in popularity of Google Transit, many transit agencies offer this data in the format specified by Google called the *Google Transit Feed Specification* (GTFS) [12]. The type of data needed to by the routing engine is a subset of the Google Transit Feed. Within the GTFS format, the data required for each bus stop in the network includes: names of routes which serve the bus stop, latitude, longitude, the scheduled times each bus arrives at the stop, and an identifier - typically the name of the closest intersection to give directions to transit riders. Other transit agencies store the static information in a proprietary format which is parsed into the GTFS format to create the routing engine.

The first implementation of Transitr system was tested in three agencies in the San Francisco Bay Area, the *San Francisco Municipal Transportation Agency* (SFMTA), *AC Transit, and Bay Area Rapid Transit* (BART). For SFMTA and BART, the data was provided in the GTFS format. The *Metropolitan Transportation Commission* (MTC) provided the data for the AC Transit network, which serves Oakland, Berkeley, and other cities east of San Francisco Bay. Transitr was also implemented for TriMet, which serves the Portland, Oregon area and publishes its static data using GTFS.

3.3.2 *Dynamic Information*

While static information allows for the transit network graph to be constructed, real-time information is required to update the wait and travel times between links. This real-time information is provided in the form of bus arrival predictions from external servers, which aggregate bus data from locations of GPS equipped transit vehicles and generate predictions.

For SFMTA, roughly 1,000 buses equipped with GPS units operate 87 different routes, broadcasting their position at approximately one minute intervals to NextBus, a private company, which performs bus arrival predictions and provides the real-time estimates via an XML feed. NextBus also provides real-time data for a subset of routes run by AC Transit. Delays due to factors such as traffic are calculated by NextBus and incorporated into the bus arrival predictions that are used by the Transitr routing engine. Some transit agencies internally operate their own real-time data feeds which are made available to the public, such as TriMet.

Transitr is able to serve any transit agency that provides both static schedule information along with an interface to real-time bus arrival information.

4 ROUTING ALGORITHM

This section describes the construction of the routing engine running inside the web server described in Section 3.2. The routing is solved using K-shortest path techniques specific to this problem, outlined below. The algorithm determines k-shortest paths instead of a single shortest path in order for the user to decide between a set of optimal routes, possibly using personal preferences not captured by the algorithm. The set of optimal routes is constructed in two steps. First, we construct a graph representing all possible ways to go from any point in the network to any other point. The construction of the graph requires static schedules and the resulting graph is stored in a database. Next, the dynamic updates of the graphs are constructed using real-time information in the form of updates to the static graph. The routing operations on the graph utilize the user inputs (origin and destination), obtained from geo-positioning. This method was chosen to minimize computation time when responding to a real-time query while shifting the computation time to the process of pre-calculating the feasible routes in the graph with static information. In this implementation, the dynamic updates are received from a third party source - thus, the problems of finding K-shortest paths and estimation of bus arrivals are decoupled. The bus arrival estimates which update the graph are received from online feeds. They do not take uncertainty into consideration; this is a limitation of the real-time data feeds available online, which only produce a single estimate for an arrival time.

4.1 Static network flow framework

Using a technique similar to time-expanded graphs [3], we construct the transit network as a directed graph. In this graph, the shortest path represents the minimum time to reach a target from a given starting location. For this, we introduce the set of time-indexed vertices $\mathscr{V} \times \mathscr{T}$. A vertex $(v,t) \in \mathscr{V} \times \mathscr{T}$ corresponds to a physical location v at a given time t. Edges can thus be defined to model motion between the different vertices. Three types of edges can be constructed:

- *Waiting*. The action of waiting at vertex v from time t to time t' is encoded by an edge $e_{(v,t),(v,t')}$. This mode occurs when someone is waiting for the bus at a bus stop.
- *Walking*. The action of walking from vertex v to vertex v' starting at time t can be encoded by an edge $e_{(v,t),(v',t+d(v,v')/w)}$ where $d(v,v')$ is the distance between v and v' and w is the *walk speed*. It is assumed that the model works in integer increments, i.e. that $d(v,v')/w \in \mathbb{N}$ for all v, v'.
- *Riding*. The action of taking a transit vehicle from vertex v to vertex v' starting at time t can be encoded by an edge $e_{(v,t),(v,t+d(v,v')/r_{v,v',t})}$ where $r_{v,v',t}$ is the average *ride speed* between v and v' at time t (note that $r_{v,v',t}$ depends on t since transit buses' travel times are contingent on traffic conditions).

We introduce the cost of an edge $e_{(v,t),(v',t')}$ as $c_{(v,t),(v',t')} = t' - t$, which is the time to travel from v to v' using the edge $e_{(v,t),(v',t')}$, if this edge is allowed. We first construct a static network problem encoding published schedules as follows. Let us introduce $\mathscr{T}$ the set of times considered for the scheduling problem.

- *Waiting subgraph.* It is always possible to wait everywhere: $\forall v \in \mathscr{V}$, $\forall t < |\mathscr{T}| - 1$, assign $c_{(v,t),(v',t+1)} = 1$.
- *Walking subgraph.* For every pair (v,v') connected by a physical road walkable in time $d(v,v')/w$, assign $c_{(v,t),(v',t+d(v,v')/w)} = d(v,v')/w$ for all $t < |\mathscr{T}| - d(v,v')/w$. This encodes that it takes $d(v,v')/w$ time units to walk from v to v' and thus that this option is open any time until $|\mathscr{T}| - d(v,v')/w$ (after that time, the walk exceeds the duration of the considered period).
- *Riding subgraph.* For every pair $\big((v,t),\ (v',t + d(v,v')/\ r_{v,v',t})\big)$ of the graph connected by a transit option at time t, assign $c_{(v,t),(v',t+d(v,v')/r_{v,v',t})} = d(v,v')/r_{v,v',t}$.

The *published* transit graph, which encodes all possible options of a pedestrian starting at any vertex v of the physical graph is the union of the waiting subgraph, the walking subgraph and the riding subgraph. A path from a given origin $o \in \mathscr{V}$ at time t, (o,t) to a destination $d \in \mathscr{V}$ at time t', (d,t') on this graph corresponds to a feasible way to travel from o to d in $t' - t$ time units, and is given as a sequence of vertices $\{(o,t),\cdots,(d,t')\}$. Given an origin node $o \in \mathscr{V}$ at time t and a destination node $d \in \mathscr{V}$, the smallest t' such that a path $(o,t) \to (d,t')$ exists defines the fastest route $t' - t$ to go from o to d at time t. Assuming all static schedules are known in advance, riding edges are constructed using $d(v,v')/r_{v,v',t}$ as the cost for these edges. The construction of the shortest path for the static (published) schedule can be computed using standard dynamic programming tools to answer questions such as the shortest path problem [8], the all points shortest path problem [18, 11], and the k shortest paths problem [9]. The set of edges constructed from the static (published) schedule is denoted $\mathscr{E}_s$.

4.2 Dynamic network flow problem

As time t is marched in $\mathscr{T}$, transit schedules are updated based on knowledge of delays from online data feeds which run an estimation engine, predicting the arrival time of the transit vehicles.

When the knowledge of a delay appears at time t in the system, the corresponding edge of the transit vehicle in question must be updated. If the vehicle which left stop $v \in \mathscr{V}$ at $\theta \leq t$, scheduled to arrive at $v' \in \mathscr{V}$ at time $\theta' = \theta + d(v,v')/r_{v,v',\theta}$, incurs a delay t_d (known from a transit monitoring system), the edge $e_{(v,\theta),(v',\theta')}$ is removed and replaced by $e_{(v,\theta),(v',\theta'+t_d)}$ (the length of the corresponding edge is thus extended by t_d). Similarly, adjacent edges $e_{(v',\theta'),(v'',\theta'')}$ which represent the same vehicle scheduled to leave v' at θ' to a third vertex (v'',θ''), are removed and replaced by $e_{(v',\theta'+t_d),(v'',\theta''+t_d)}$ (the corresponding edge is shifted later in time).

The full dynamic graph is represented similarly to the static graph, using a union of a waiting, walking and the updated riding subgraphs. The walking and waiting subgraphs are identical to the static network; it is always possible to wait everywhere, and walking is always allowed on roads which can be physically walked. The static schedule graph is thus updated for all $\mathscr{T}$, leading to an online (time varying) set of edges denoted by $\mathscr{E}_t$, where $\mathscr{E}_t$ is revealed at time t.

4.3 Online shortest paths implementation

A variety of methods have been proposed for computing the solution to shortest path problems

on a dynamic graph. A brute force implementation involves recomputing the shortest paths after every update, using static algorithms such as the all pairs shortest path algorithms of [18, 11], for all $t \in \mathscr{T}$. Alternatively, one can dynamically maintain the all points shortest paths on the dynamic graph as the edges are added and removed [6, 19, 25]. The efficiency gain is made by recomputing (or updating) the solution of the all pairs shortest path problem using the previous solution.

Another implementation technique leverages an offline pre-computation of feasible paths on the graph [20]. Since the feasible paths, denoted by the set $\mathscr{P}$ are precomputed, the the shortest path $p_{(o,t),(d,t')}$ in the set of feasible paths $\mathscr{P}_{(o,t),(d,t')} \subseteq \mathscr{P}$ for an origin destination pair $(o,t),(d,t')$ can be computed by updating the costs on the feasible paths at the time the user query is generated. Thus, the update step is completed on all paths for a given origin destination pair, then the fastest path is returned. This implementation has two advantages for the system presented in this article. First, the main computation expense is paid once upfront. Second, when the number of updates to the graph is large relative to the number of queries, the algorithm is likely to be more efficient than algorithms which place the computational burden on the update step.

In the current system, a pre-computation method is selected. Using the static transit graph, a list of feasible paths from all pairs of vertices is computed. When a transit vehicle is delayed, the specific feasible path on the static graph becomes infeasible (because the $e_{(v,t)(v',t')}$ edge is removed), but a new set of feasible paths can be constructed using the new edge $e_{(v,t+t_d)(v',t'+t_d)}$. We wish to maintain a list of feasible paths on a dynamic graph by modifying the feasible paths on the static graph when updated edge data becomes available.

The static feasible paths are updated with dynamic information as follows. When an edge from the static graph is updated, it is replaced by a new edge. If the new edge begins at a later time t_d, and $t_d < |\mathscr{T}|$, then it is feasible to wait for the later edge. Similarly, at the destination end of the edge, if the destination vertex appears in the static feasible path, then the new dynamic path remains feasible, and the path is successfully updated. If the destination vertex is not in the static feasible path, the new dynamic path is infeasible.

Since the look up cost of the shortest paths in the pre-compute algorithm is linear in the number of feasible paths in the dynamic network, we implement two techniques to decrease the number of feasible paths which are stored after the pre-computation step. First, we remove all paths which exceed a maximal number of transfers. Second, for a fixed origin destination pair, we remove all paths whose fastest historical time is longer than another path's slowest historical time, since no series of updates will likely ever lead to the dominated path becoming the shortest path. These steps reduce the size of the database while still maintaining a list of feasible paths.

4.4 Routing engine summary

To implement the routing engine, we build the published graph from the transit schedules. Then, we compute all of the all pairs feasible paths on the static graph, and store them by origin destination pair. Then, for each origin destination pair, paths which have more than three transfers, or paths which under no edge update scenarios could ever be fastest are removed. The remaining edges are stored permanently. At each time $t \in \mathscr{T}$, the new edge information is obtained. The first time a given origin destination is queried at t, the set of static feasible paths and their costs are updated to find dynamic feasible paths and their associated costs. The shortest k paths are cached for other queries at t for the same origin destination pair, then returned to the user.

5 SYSTEM EVALUATION

In this section we present an evaluation of the Transitr system. In order to quantify the effects of real-time data in trip planning, a schedule-based transit trip planner (TTP) was developed using the same routing algorithm as the real-time trip TTP. The two TTPs differ in two aspects, the sources of data for the wait times at bus stops, and the travel time between bus stops. The schedule-based TTP uses a static database of bus schedules, while the real-time TTP obtains estimated wait times from a real-time bus prediction engine. The real-time TTP takes a snapshot of the actual state of the system at the time the user makes a request, and uses historical data to predict the evolution of the network, and the user's trip. By comparing two nearly identical TTPs, we can examine the scenarios in which real-time data more accurately estimates travel times and more frequently predicts the optimal route.

This section is organized as follows. First, we present a description of an experimental evaluation of the system using the transit system in the San Francisco Bay Area. Next, we characterize the accuracy of the transit vehicle arrival estimates provided by NextBus, on which the real-time TTP relies. For each TTP, the accuracy of the estimated trip travel time along a fixed path is assessed by comparing the estimated travel time given by the TTP at the the start of the trip to the observed travel time to complete the trip. Because of the errors in the estimated trip travel times, the TTPs do not always select the optimal (shortest travel time) route. The frequency with which the optimal route is selected, and the degree of sub-optimality of each TTP when the optimal route is not returned is also assessed.

5.1 Experimental setting

Over six-hundred trips were planned using the schedule-based TTP and the real-time TTP using real–time data from San Francisco Municipal Transit Authority buses and light rail trains running 87 routes across the city. The trip origin and destination pairs were generated using a spatially uniform random distribution over the city. The trips were planned over the period of July 23 - 28, 2009 at various times throughout the day.

Because each transit vehicle is tracked using the on-board GPS receivers, the actual travel times along each path between the *origin-destination* (O-D) pairs can be determined after the trip is completed. These travel times serve as a ground truth upon which the transit vehicle arrival predictor (NextBus), the schedule-based TTP, and the real-time TTP are benchmarked. These GPS tracks also allow for the actual optimal (fastest) route to be determined between the O-D pairs.

5.2 Accuracy of the estimated transit vehicle arrival engine

The performance of the real-time TTP is dependent on the estimation of the transit vehicle arrival times at its upcoming stops, which is provided by NextBus. NextBus generates predictions by using historical averages of travel times between segments, calculated at different times of day. Fig. 3 shows the error of the arrival time estimate for 23,349 queries to NextBus. Each time NextBus is queried, an an arrival time is estimated, and the error is calculated after the bus arrives at that stop.

Estimated arrival times between 0 and 10 minutes have a median error of zero, and the interquartile range (the middle 50% of estimates) tend to fall within a minute of the true travel

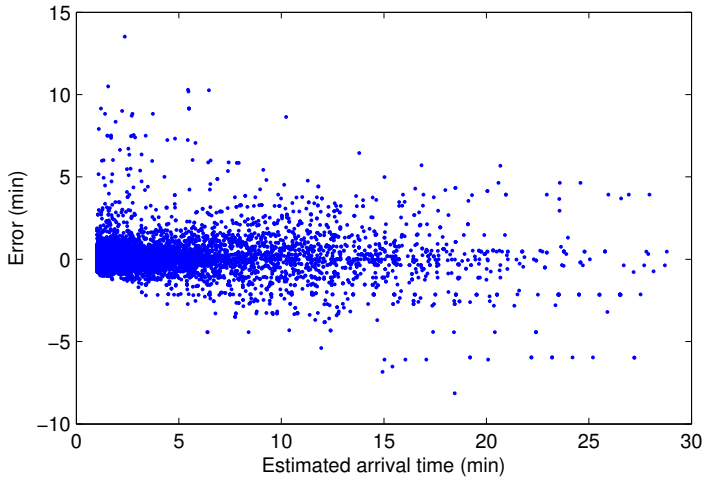

Figure 3: Estimated arrival time error. The error in minutes in the estimated arrival time for 23,349 estimates, as a function of the estimated arrival time. X-axis: estimated arrival time in minutes. Y-axis: error in minutes. Points of the negative half of the Y-axis correspond to bus arriving earlier than predicted, while points on the positive half correspond to the bus arriving later than predicted.

time. However, a small number of outliers, specifically for small arrival times, can potentially cause significant challenges if the information is used to determine if a transfer between buses can be made. As the estimated arrival time increases, variance in error also increases. This variance effects the results of real-time TTP in two ways: determining the travel time of the bus to the transfer point, and determining the wait time at the transfer point for the next leg of the trip. In the set of six-hundred trips planned, there existed over four-hundred trips which required transfers over 10 minutes past the time the trip request was made.

5.3 Accuracy of the estimated trip travel time

The accuracy of of the travel time estimate along the paths suggested by schedule-based and real-time TTPs are compared to the ground truth travel time along the same paths. Fig 4 characterizes the amount of uncertainty in the estimated travel times, divided by the number of cases in which the planned trip required zero transfers, one transfer, or two transfers. Evaluated over all the cases, real-time data increases accuracy for travel time prediction, most noticeable in trips requiring transfers. The Wilcoxon signed-rank test was used to test for statistically significant difference between the two trip planners . The Wilcoxon signed-rank test evaluates the null hypothesis that two related samples have the same distribution. The data compared were the percentage error of the travel times returned by the schedule-based and real-time TTP, each verified against the actual travel times. The median error of the schedule-based TTP is 14.9% vs 11.7% for the real-time TTP. Statistical significance was defined as $z = 4.08$, $P < .00001$ two-tailed, thus the null hypothesis is rejected. It is concluded that the use of real-time data reduces travel time prediction error. However, the small difference between the errors show that the improvement in travel time accuracy is marginal.

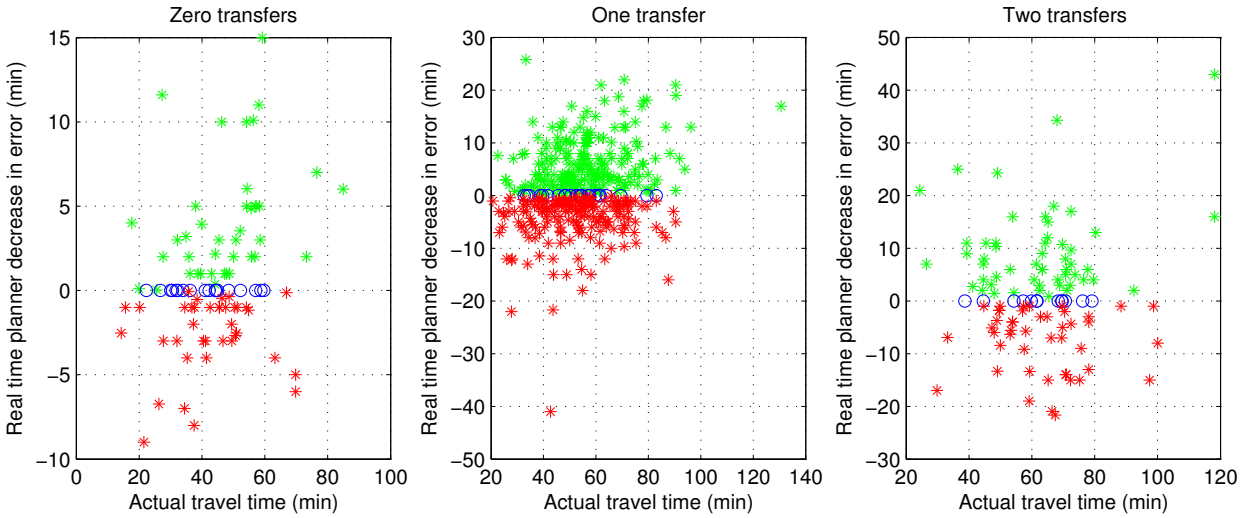

Figure 4: Effect of travel time accuracy when using real-time data. Positive values on the Y-Axis refer to cases in which real-time TTP more accurately predicted total travel time. Trips with zero transfers: 40 (41.6%) more accurate, 38 (39.6%) less accurate 18 (18.8%) same prediction. Trips with one transfer: 232 (51.3%) more accurate, 190 (42.0%) less accurate, 30 (6.6%) same prediction. Trips with two transfers: 58 (48.8%) more accurate, 47 (39.5%) less accurate, 14 (11.8%) same prediction.

5.4 Route selection optimality

The main functions of the TTP are to provide an expected travel time from origin to destination as well as suggest a set of routes based on the state of the transit network at the time of the user's request. The routing algorithm calculates the K-shortest paths and returns the five fastest routes to the user. This allows the user to make a decision based on the total trip time, as well as personal preferences such as walking distances and number of transfers and other variables which the application does not factor in. In this analysis done in this section, only one route predicted by each TTP - the expected optimal route determined by the shortest travel time, is evaluated. The predicted route by both TTPs is compared to the ground truth - the actual optimal route, obtained by tracking the position of multiple buses over the duration of the trips.

Table 1 summarizes the route selection optimality of the TTPs. The percentage of cases in which the two TTPs accurately predicted the actual optimal route was determined. These cases are split into three rows, when both TTPs predicted the optimal route, and when only either one of the TTPs made the correct route prediction. If neither TTP predicted the actual optimal route, the actual travel times of the predicted routes were compared. The next three rows show the percentage of cases in which one of the sub-optimal routes resulted in a shorter actual travel time, or if the same sub-optimal route was predicted by both TTPs. The results are split into four scenarios: long trips (trips longer than 30 minutes), short trips (trips shorter or equal to 30 minutes), trips in which a transfer was involved, and trips which required no transfers. In most cases, the real-time TTP outperformed the schedule-based TTP by a small margin.

	Long	Short	With transfers	No transfers
Both predicted optimal	27.3	44.4	20.5	75.0
Only real-time predicted optimal	20.3	16.7	23.8	0.0
Only schedule-based predicted optimal	14.1	11.0	15.6	4.1
Neither predicted optimal: real-time faster	12.5	11.2	13.9	4.2
Neither predicted optimal: schedule-based faster	10.2	5.6	11.4	0.0
Neither predicted optimal: same prediction	15.6	11.1	14.8	16.7

Table 1: Optimality of fastest route returned by the schedule-based TTP and the real-time TTP with respect to the observed fastest route. Values represent percentage of cases.

5.5 Implications for further research

The study shows that the inclusion of real-time information in the Transitr system provides two benefits over schedule-based TTPs: more accurate predictions for total trip times, and a larger number of cases in which the optimal route was suggested. However, these benefits were found to be marginal, leaving room for improvement in three areas: removal or detection of outliers as shown in Figure 3, improved long-term estimation of buses arriving in over 15 minutes, and incorporation of uncertainty into bus arrival estimates.

Currently, Transitr is dependent on third party data sources to perform bus arrival estimation, which is based on time-of-day historical averages of segment trip times. As described in Section 2, newer estimation algorithms exist which take into consideration factors such as real-time traffic, but have not been tested in the context of a trip planning application. As shown in Figure 3, NextBus accurately predicts bus arrivals when the expected arrival time is under 15 minutes, with the exception of outliers in 1-5% of cases. However, the incorporation of real-time data into a trip planner requires estimation of bus arrivals at transfer points, which often occur greater than 30 minutes from the time the trip is initially planned. In these problems, the current estimation engine is less accurate in predicting bus arrivals, leading to missed transfers and large discrepancies in estimated trip times and optimality of suggested routes. Further studies will be done to incorporate more advanced bus arrival estimation techniques into the Transitr application, while characterizing uncertainty into the predictions to compensate for potential missed transfers.

In addition, the experiment examined the ability of the TTP to determine the optimal route based on information only at a single time instant prior to the trip. Table 1showed that the real-time TTP performed better in shorter trips. Larger errors in estimation for the total time of longer trips, and trips which include transfers, lead to greater number inaccuracies in the suggestion of optimal routes. The conclusion from these results is that accuracy drops with respect to the amount of time into the future the TTP must predict the state of the system. A solution to this problem is to allow for the algorithm to re-calculate the expected optimal route after the trip has been planned, allowing a user to receive updated information about their route as the state of the transit system evolves. As more recent information becomes available, users will receive new estimates for their route travel time. It is also possible that the prediction for the optimal route may change after the user has reached a transfer point, or even while riding the bus. Further evaluation of the system includes re-calculating the optimal route at different stages of a trip, and examining the accuracy of the total travel time at these stages.

6 CONCLUSION

This paper described an implementation of a transit trip planning system, which includes an application developed for mobile phones, a routing engine running a K-shortest paths algorithm, and interfaces to bus arrival prediction engines provided by third parties. It was discovered through over six-hundred experiments in San Francisco that the real-time data provided marginal improvements to current schedule-based trip planners. The study showed that there is potential for much improvement in transit trip planning by using real-time data in conjunction with more advanced estimation techniques which focus on solving long term estimates for bus arrivals, and characterizing uncertainty into the predictions. The growth of geo-positioning systems in transportation and the availability of real-time data feeds from transit agencies and other transportation modes is promising, as more opportunities for research are opened in the field of real-time transit, and multi-modal navigation.

7 ACKNOWLEDGMENTS

The authors of the paper would like to acknowledge NextBus Inc. for providing the online feeds to the real-time data.

References

[1] Service standards report. Technical Report Q3 FY08, San Francisco Municipal Transportation Authority, July 2008.

[2] Abdelfattah and Khan. Models for predicting bus delays. *Transportation Research Record*, 1623(0361-1981):8–15, 1998.

[3] R. Ahuja, T. Magnanti, and J. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1993.

[4] D. Chien and Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, pages 430–438, September/October 2002.

[5] C. Daganzo. How to improve bus service. *Institute of Institute of Transportation Studies, Working Paper UCB-ITS-VWP-2008-6, University of California, Berkeley, CA*, 2008.

[6] C. Demetrescu and G. F. Italiano. A new approach to dynamic all pairs shortest paths. *Journal of the ACM*, 51(6):968–992, 2004.

[7] M. Dessouky, R. Hall, L. Zhang, and A. Singh. Real-time control of buses for schedule coordination at a terminal. *Transportation Research Part A: Policy and Practice*, 37(2):145–164, February 2003.

[8] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[9] D. Eppstein. Finding the k shortest paths. In *Proc. the Annual Symposium on Foundations of Computer Science (FOCS)*, pages 154–165, 1994.

[10] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 29–39, Breckenridge, CO, June 17-18 2008.

[11] R. W. Floyd. Algorithm 97 (shortest path). *Communications of the ACM*, 5(6):345, June 1962.

[12] Google. Google transit feed secification. *http://code.google.com/*.

[13] Graphserver. The open-source multi-modal trip planner. *http://graphserver.sourceforge.net/*.

[14] I. P. H. Jula, M. Dessouky. Real-time estimation of travel times along the arcs and arrival times at the nodes of dynamic stochastic networks. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):97–110, 2008.

[15] M. Hickman and N. Wilson. Passenger travel time and path choice implications of real-time transit information. *Trans. Research C*, 3(4):211–226, 1995.

[16] B. Hoh, M. Gruteser, R. Herring, J. Ban, D. B. Work, J. Herrera, A. M. Bayen, M. Annavaram, and Q. Jacobson. Virtual trip lines for distributed privacy-preserving traffic monitoring. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 15–28, Breckenridge, CO, June 17-18 2008.

[17] E. Horvitz, J. Apacible, R. Sarin, and L. Liao. Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service. *In Twenty-First Conference on Uncertainty in Artificial Intelligence, (UAI-05), Edinburgh*, 2005.

[18] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM*, 24(1):1–13, 1977.

[19] V. King. Fully dynamic algorithms for maintaining all-pairs shortest paths and transitive closure in digraphs. In *In Proc. 40th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 81–91, 1999.

[20] J. Miller and E. Horowitz. Algorithms for real-time gathering and analysis of continuous-flow traffic data. In *Proc. IEEE Intelligent Transportation Systems Conference ITSC '06*, pages 1454–1459, 2006.

[21] R. Mishalani, S. Lee, and M. McCord. Evaluating real-time bus arrival evaluating real-time bus arrival information systems. *Transportation Research Record*, 1731(00-0739):81–87, 2000.

[22] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Trafficsense: Rich monitoring of road and traffic conditions using mobile smartphones. *Technical Report MSR-TR-2008-59*, 2008.

[23] G. F. Newell and R. B. Potts. Maintaining a bus schedule. *Proc. Second Conference Australian Road Research Board*, pages 388–39, 1964.

[24] B. C. M. O'Mahony. An examination of the public transport information requirements of users. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):21–30, 2007.

[25] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest-path problem. *Journal of Algorithms*, 21(2):267–305, 1996.

[26] K. Rehrl, S. Bruntsch, and H. Mentz. Assisting multimodal travelers: Design and prototypical implementation of a personal travel companion. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):31–42, 2007.

[27] M. D. Rossetti and T. Turitto. Comparing static and dynamic threshhold based control strategies. *Trans. Research A 32A, 607-620*, 1998.

[28] Shalaby and Farhan. Prediction model of bus arrival and departure times using avl and apc data. *Journal of Public Transportation*, Vol. 7(No. 1):41–61, 2004.

[29] D. B. Work, O.-P. Tossavainen, S. Blandin, A. M. Bayen, T. Iwuchukwu, and K. Tracton. An ensemble kalman filtering approach to highway traffic estimation using gps enabled mobile devices. *47th IEEE Conference on Decision and Control, Cancun, Mexico*, 2008.

[30] J. Xu, N. Wilson, and D. Bernstein. The holding problem with real-time information available. *Trans. Sci. 35, 1-18*, 2001.

[31] J. Yoon, B. Noble, and M. Liu. Surface street traffic estimation. In *6th International Conference on Mobile Systems, Applications, and Services*, pages 220–232, San Juan, Puerto Rico, June 11-14 2007.

[32] YouTube. Real-time transit application. *http://www.youtube.com/watch?v=fSeTuisLw6c*.