6-2011

Real-time trip information service for a large taxi fleet

Rajesh Krishna BALAN
Singapore Management University, rajesh@smu.edu.sg

Nguyen Xuan KHOA
Singapore Management University, xkhguyen@smu.edu.sg

Lingxiao JIANG
Singapore Management University, lxjiang@smu.edu.sg

Citation

BALAN, Rajesh Krishna; KHOA, Nguyen Xuan; and JIANG, Lingxiao. Real-time trip information service for a large taxi fleet. (2011). *MobiSys '11: Proceedings of the 9th International Conference on Mobile Systems, Applications and Services: June 28 - July 1, 2011, Bethesda, MD*. 99-112.
Available at: https://ink.library.smu.edu.sg/sis_research/1678

Real-Time Trip Information Service for a Large Taxi Fleet

Rajesh Krishna Balan, Nguyen Xuan Khoa, and Lingxiao Jiang
School of Information Systems, Singapore Management University
80 Stamford Road, Singapore 178902
{rajesh,xknguyen,lxjiang}@smu.edu.sg

ABSTRACT

In this paper, we describe the design, analysis, implementation, and operational deployment of a real-time trip information system that provides passengers with the expected fare and trip duration of the taxi ride they are planning to take. This system was built in cooperation with a taxi operator that operates more than 15,000 taxis in Singapore. We first describe the overall system design and then explain the efficient algorithms used to achieve our predictions based on up to 21 months of historical data consisting of approximately 250 million paid taxi trips. We then describe various optimisations (involving region sizes, amount of history, and data mining techniques) and accuracy analysis (involving routes and weather) we performed to increase both the runtime performance and prediction accuracy. Our large scale evaluation demonstrates that our system is (a) accurate — with the mean fare error under 1 Singapore dollar ($\approx$ 0.76 US$) and the mean duration error under three minutes, and (b) capable of real-time performance, processing thousands to millions of queries per second. Finally, we describe the lessons learned during the process of deploying this system into a production environment.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organisation and Design—*distributed systems*; H.2.8 [**Database Management**]: Database Applications—*data mining*; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*indexing method*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems*

General Terms

Algorithms, Design, Experimentation, Human Factors, Performance

Keywords

Taxi Fleets, Trip Information Service, Partition-based Predictions, Nearest Neighbour Queries, History-based Predictions

1. INTRODUCTION

In this paper, we describe how we built, tested, improved, and deployed a real-time trip information system for a large GPS-enabled Singaporean taxi company. The company operates a fleet of about 15,000 taxis that report their positions, status, and trip records continuously to a central server.

Our service uses historical data to allow passengers to query the expected duration and fare of a taxi trip that they plan to take. This allows passengers to plan their time and budget accordingly as well as serving as a safeguard against unscrupulous drivers who might take longer than expected routes (especially when servicing tourists). passengers. This type of information system is potentially useful to a wide range of public transportation operators (including buses, trains, and taxis), and even other vehicle fleet operators. For example, a logistics company might find the trip information service useful in better estimating their costs a priori.

We address five key challenges when building this system. First, the amount of spatial data collected by the taxi company was huge — on the order of tens of millions of records per month. Second, the system needs to answer queries in real-time; i.e., for any query, it must be able to quickly find enough previous trip records, from millions of past records, to accurately answer that query.

Third, we need to account for various time-related factors affecting trip fares and durations. For example, trip durations are often longer during peak traffic hours. Also, Singapore has a highly variable taxi fare structure that depends on the day of the week and time of the day (c.f. Section 2). As we show in a later Section, these factors make existing trip prediction systems such as Google Maps highly inaccurate in predicting taxi fares and travel durations.

Fourth, we need to determine how much historical information is necessary to provide accurate answers to trip-related queries. For example, is one month of historical data sufficient for accurate predictions? Would six months of data be better? Would using more data result in more accurate predictions at the cost of more memory and time? In some cases, such as when the taxi fare structure changes or when traffic patterns are affected by transient construction or special events, naively using more data naively actually yields worse results. We need to ensure that our prediction model is flexible enough to leverage large amounts of data (to increase accuracy) while still being robust enough to handle data with unusual and inappropriate patterns.

Fifth, like all real-world systems, our data is quite noisy and contains both outright errors and non-standard behaviour patterns. We need to discover ways to a) filter out bad data, and b) identify and correct prediction errors due to non-standard behaviour. In particular, we identified mainly two types of non-standard behaviour: i) variations due to route choices between any two locations, non-standard routing trips (such as trips that are not point-to-point, i.e., the passenger did no want to go to their destination directly, but rather went somewhere else first), and ii) variations due to weather (torrential tropical thunderstorms are quite common in Singapore),

We describe the algorithms and various optimisations used in our system and present detailed large scale evaluation results. The evaluation is performed on 21 months of historical data (about 250 million data points in total) generated by approximately 15,000 taxis and 35,000 drivers, and shows that our system has excellent performance. In particular, our mean fare prediction error is under 1 Singapore dollar ($\approx$ 0.76 US$) and our mean duration error is under three minutes.

We end the paper with a discussion of the issues we faced in taking our research prototype into a real deployment environment. Overall, this paper makes the following contributions:

- A detailed description of the steps needed to build a real-time service for a large commercial taxi fleet that uses millions of records as input.
- Methods for identifying real-time patterns, both normal and irregular, in spatial data, which can be applicable to other transportation networks in addition to the taxi domain.
- A principled approach to balancing the trade-offs that arise between the amount of history used, accuracy of the results, and real-time performance.
- The use of nearest neighbour techniques to produce self-scaling accurate predictors for this domain.
- Insights into the challenges in moving research prototypes into operational environments.

2. BACKGROUND AND DATA

In this section, we describe the Singaporean taxi system as well as the specific dataset analysed in this paper.

2.1 Overview of Singapore's Taxi System

Singapore is a small densely populated island just 50 by 25 kilometres wide with an area of 710 square kilometres (about 15% larger than the city of San Francisco) that is inhabited by 5 million people. To efficiently transport people across the country, Singapore has a world-class very affordable public transportation system of taxis, buses, and rapid transit rail lines. Taxis, in particular, are widely available and relatively low-priced. Even with various surcharges, metered fares are usually within 2.8 to 50 Singapore dollars with only a few fares exceeding 50 S$. This affordable and accessible public transportation network, coupled with high taxes on both private cars and fuel, result in many Singaporeans choosing not to own a car.

In September 2010, Singapore had 25,624 taxicabs operated by seven companies and several hundred independent owners [17]. These taxis can be flagged down at any time of the day along any public road, with well-marked taxi stands available outside most shopping centres and office buildings.

Except for independent owners, taxi drivers rent a taxi, for a daily fee, from one of the seven operators and are responsible for their fuel expenses and any road usage charges. Many renters sublet their taxis to other drivers for a fraction of the daily rental fee. Hence, it is common to see the same taxi being operated nearly 24 hours a day, 7 days a week.

Each taxi driver earns money by collecting metered fares from passengers. The meter must be used for every trip; ad-hoc pricing is not allowed. The meter rates are mostly standard across taxi companies [25] and combine a fixed starting fare (ranging from 2.80 S$ to 5 S$, depending on taxi types) with time and distance based charges (20 Singapore cents for every 385 meters travelled or 45 seconds of waiting).

To reduce congestion, Singapore has an Electronic Road Pricing (ERP) system. When entering an ERP zone during operational hours, drivers of empty taxis must pay the ERP charge themselves. Otherwise the passenger pays the ERP charge on top of their metered fare. The exact ERP charge depend on the day of the week, the type of vehicle (car, bus, etc.) and the time of the day [16].

In addition to the variable ERP charges, there are other time and location based surcharges affecting taxi fares [25]. For example, the taxi fare is 35% higher during peak periods (7 a.m. to 10 a.m. and 5 p.m. to 8 p.m. on weekdays) and 50% higher for trips that start between midnight and 6 a.m. on any day. In addition, taxi trips that start at the airport, casinos, or the central business district incur additional location-based surcharges ranging from 3 S$ to 5 S$. Additional charges apply if you book a taxi by phone or SMS (the taxi will then come to your current location and pick you up), instead of flagging down a taxi on the road.

Overall, these variable charges make the taxi fare structure quite complicated. This makes other fare prediction systems for Singapore taxis quite inaccurate as they cannot account for this variability, and passengers frequently don't know the expected fare for any given taxi trip. As a result, the taxi operator was eager to implement our solution into their operating environment as it gives them a strong service differentiator from their competitors.

2.2 Data Collected

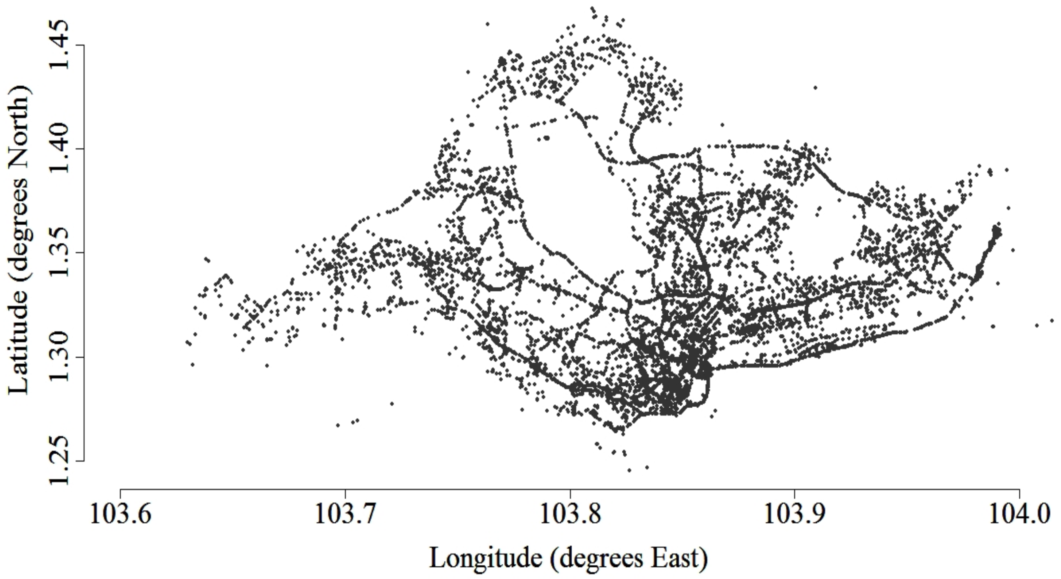
As part of a modernisation drive that took place over the last decade, Singaporean taxi operators added GPS receivers, packet radio equipment (to transmit GPS locations back to a central server), and a touchscreen LCD display to every taxi. This allowed operators to know the location of every taxi at all times, and to get accurate assessments of the earnings of each taxi (obtained directly from the taxi's meter). This allowed them to both better schedule their fleet and to accurately report their drivers' earnings to the tax authorities. It also makes it possible to obtain accurate information about the day-to-day operations of the taxis in the fleet.

We obtained the GPS-enhanced data logs for a period of 21 months (January 2009 to September 2010) from one Singapore taxi operator. This company operates a fleet of about 15,000 taxis driven by about 35,000 drivers.

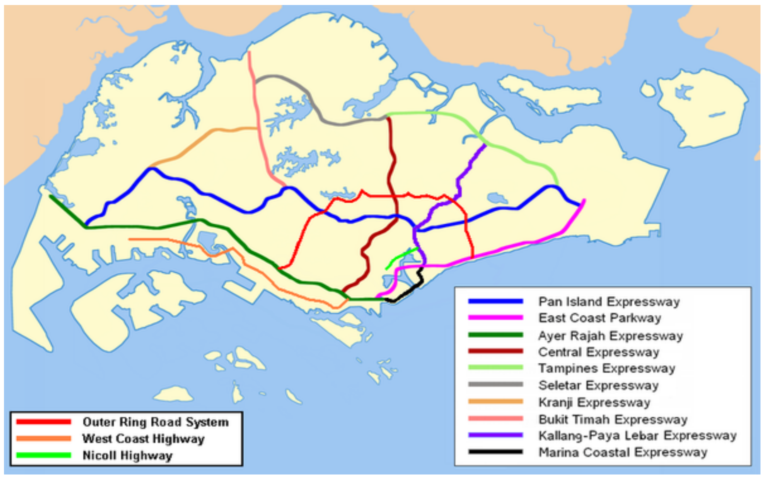
Our dataset contains records about every paid trip that a taxi made. It contains the start and end GPS coordinates of the trip, its start and end times, the distance the taxi travelled, and the final metered fare. In addition, we also had the positions of the taxis as they travelled on the roads. In this paper, we use that position information only to plot the routes of specific anomalous trips (to understand why those trips had anomalous fares and/or times) as it is very computationally intensive in our current setup to do more than that (like track every taxi's route).

Our entire dataset contained about 250 million trip records (average of 12 million paid trips from all the taxis together per month). Naturally, with such a large dataset, we expect to find errors in the data. We found two main types of errors; 1) location errors where trips either started or ended outside Singapore or in unaccessible areas, and 2) semantic errors where we found trips that had trip time errors (negative, 0, or impossibly large trip times), fare errors (fares that were impossibly low or impossibly high), and distance errors (trips that had distances of 0 or impossibly large values).

We reported these errors back to the operator and verified that the errors were valid and that these errors were caused by a number of different reasons such as the canyon effect on GPS accuracy, clock



This is a plot of taxi locations using 10,000 randomly selected GPS points from a single day's worth of data — less than 0.3% of one day's data. Note: the areas inside the Singapore outline without points, are physical areas without accessible public road networks.



Major Highways in Singapore (source: Wikipedia). The taxi locations plot matches these highways well.

Figure 1: Visual Inspection of Taxi Locations

synchronisation issues, and software bugs. Overall, about 3.6% of our trip records were erroneous and were removed.

2.3 Properties of the Taxi Network

Overall, the Singapore taxi network has significantly different characteristics from most US taxi networks (with the possible exception of large cities, e.g., New York City):

1 **Taxis are Cheap**: In particular, most fares are 10 S$ or lower — hardly ever exceeding 50 S$. This is quite affordable for the local population and taxis are thus a more comfortable and usually faster alternative to cheaper public buses and trains.

2 **Taxis Are Common and Found Everywhere**: With 25,000+ taxis on the road, it is quite rare to encounter areas with no taxis. There are peak times where all the taxis on the road may be occupied though — but again, this happens only at certain times and places.

3 **Most Pickups are Street Pickups**: More than 90% of the taxi trips are street pickups (passengers flag down free cabs on the street) and not pre-meditated calls. Booking calls are usually made when there are no free taxis on the road or in other special cases.

4 **Taxis are Used for All Activities**: Due to the low cost and high availability of taxis, taxis are used for all activities — including going to soccer practice, visiting friends, and for travelling to and from the grocery store. This is unlike most US cities where the taxi networks mostly serve the airport routes.

These four factors ensure that taxis are found on every street with very high frequencies — either dropping off or looking for passengers. This is shown in Figure 1 which plots the GPS coordinates of just 10,000 randomly sampled location points from a single day of data (less than 0.3% of a day's worth of taxi GPS location updates).

The figure shows two main things: 1) the error rate of location data is low as only a few outliers are visible (we use the location data to determine the start and end points of each trip), and 2) taxis really do go everywhere as all the main roads and neighbourhoods are clearly visible even with such a small subset of data (the outline of Singapore is also quite clear due to the various coastal roads). Overall, we found that, on average, each taxi was occupied by a paying customer about 30% of the time and spent the remaining 70% of the time either looking for fares (59% of the time), or not in operation (11% of the time).

Finally, we observed that there were many repeated taxi trips between any two locations in Singapore. This could be because individuals have common regular routes such as going to school, work, etc. In addition, the population density in Singapore is incredibly high, and most of the population ($> 95\%$ [7]) stay in high rise apartment or condominium buildings. This leads to a natural "multiplier" effect for taxi routes. These factors suggest that rare taxi routes, where the trip's start and end locations are not observed more than a few times a month, might be uncommon — even in the Singaporean case where taxis are used for every route (and not just a few special routes usually involving the airport). We thus decided to use history as a prediction mechanism as we believed that we could find sufficient similar historical trips for every route.

2.4 Privacy Concerns

In this paper, we do not address the significant privacy concerns that can arise in working with spatial and transactional data. Our focus is on building an information service using data provided under a confidentiality agreement with an interested taxi operator; that collected the data with the consent of its contracted drivers. Our service uses this data in an aggregated way to generate predictions that are generally not traceable to individual taxis or passengers.

3. SERVICE REQUIREMENTS

In collaboration with the taxi operator, we identified four main requirements for the trip information system. Our final system satisfies all four:

1 **Accuracy**. Foremost, the system must produce accurate predictions that match reality. In particular, the operator mentioned that prediction errors must be within a few Singapore dollars (2 S$ at most) and a few minutes (5 minutes at most) of real values. Otherwise, passengers and drivers will not like the system.

2 **Real-Time Capability**. The service must be able to handle queries in real time. Generating a prediction should take on the order of milliseconds.

3 **Low Computational Requirements**. The operator is willing to provide at most one or two 64G server machines and our system must run efficiently on those machines without requiring a cluster environment or more powerful servers. This is a non-negotiable requirement as the operator does not have the budget or manpower to purchase higher-end machines or to setup an in-house cluster, and they are not comfortable with the cost and privacy implications of sending their data to off-site cluster farms like Amazon S3.

4 **Easy to Deploy Operationally**. Finally, the system must be easily deployable at the operator's computing centre with an easy interface for both passengers and in-house users (booking call centre operators etc.)

4. TRIP INFORMATION SERVICE

In this section, we present the design and implementation of the real-time trip information service.

4.1 Failed Solution: Use Google Maps

In developing this service, we considered a fairly straightforward approach based on Google Maps that would use the Google Maps API to retrieve an expected travelling distance and time for that trip. We would then calculate an expected trip fare using the expected distance and time values. However, this approach failed for three main reasons:

First, the Maps API introduced network latencies and rate limits (imposed by Google), slowing down the service. Google offers high-capacity, low-latency Google Maps services; however, the taxi operator is reluctant to pay that additional recurring cost (versus a small one time cost for provisioning an in-house server) for a "nice-to-have" service that is not part of their core operations.

Second, calculating an accurate trip fare given just the estimated distance and time of the trip was hard. As mentioned earlier (Section 2.1), Singapore uses a distance-based fare structure augmented with time and location-based surcharges. Determining the location-based surcharges required tracking the exact route of the trip and this proved to be impossible, with our resources, to do in real time.

Finally, and most importantly, the Google Maps results (as of June 2011) were not very accurate. We tested the Google Maps API with a single month of trips ($\approx$ 12 million clean trip records) and found that, on average, the trip durations predicted by Google Maps were about 35% off from the actual trip durations and about 40% off from the actual trip times. We also tested other local taxi trip prediction systems, such as Gothere.sg (http://gothere.sg), and found them to be similarly inaccurate on fare and duration predictions. We thus needed to find a better solution that can satisfy the requirements stated in Section 3.

4.2 Use Trip History

The insight we have for a viable solution is that taxi drivers generally know the routes between any two locations and often follow the same routes. Hence, historically recorded taxi trips should contain abundant information for predicting the trip duration and fare for a future trip. The key to realise the insight is, for a new trip, to find historical trips *similar* to it and perform predictions based on the durations and fares for these similar trips.

Trip durations and fares can be affected by many factors, including three obvious ones recorded in our dataset: the start location, the end location, and the start time of a trip, which we call the three *basic features* of each trip. Intuitively, in order to have an accurate prediction for a new trip, the historical trips used for prediction should have basic features similar to those of the new trip.

When we have a set of similar trips T found for a new trip t, the prediction is as simple as calculating the average duration and fare of these trips, where $d(t)$ and $f(t)$ represent the duration and fare of a trip t respectively, and avg is the normal arithmetic mean:

$$d(t) \equiv avg_{t_i \in T}\{d(t_i)\}, \quad f(t) \equiv avg_{t_i \in T}\{f(t_i)\} \qquad (1)$$

Thus, the primary challenge lies in quickly searching for similar trips. A naive solution is to search the whole trip database to find a historical trip with exactly the same GPS coordinates and start time as the new trip. This solution is computationally demanding and also incredibly fragile — as it is quite unlikely to find another trip with the exact same start time and start and end GPS coordinates (even with 21 months of historical data).

Also, even if we could find a trip with the same basic features, other factors such as road construction, raining, accidents, etc. may affect the trip duration and fare. To reduce the effect of this variability on prediction accuracy, we need to have predictions aggregated from multiple similar trips. The challenge then becomes finding all trips that start and end within some fixed distance (50 meter for example) of the trip that we are predicting. This "expansion" of search regions must be enabled for finding more similar trips for prediction. An obvious, but inefficient solution to this requirement would be to use a spatial database with appropriate indexes. For example, several simple distance range queries, on one month of data using PostgreSQL with the PostGIS extensions, took about 10 to 30 seconds to return matches. This is not fast enough and the speed gets progressively worse as we add more months of data.

To balance prediction accuracy, efficiency, and hit rate (a measure of how often we can have enough similar trips to make a reasonable prediction), we developed our solutions using a *partitioning* approach that splits the whole continuous search space (formed by the three basic features) into discrete, easily accessible *time-space partitions*, so that expensive queries for similar trips of close start time and close start and end locations can be turned into efficient queries for trips belonging to the same partition.

4.3 Partition-based Prediction

The essence of our solutions is to split the whole trip dataset into discrete partitions indexed by trip start time and start and end locations (i.e., the trip features that can affect durations and fares), and treat all trips belonging to the same partition as similar trips. The following sections describe how we partition both the time and location dimensions.

4.3.1 Time Windows

The trip start time is one of the three basic features and affects trip durations and fares on an hourly and daily basis. As mentioned in Section 2.1, there are numerous time-based fare surcharges such as peak period and midnight surcharges. Thus, we designed four ways to split the start time dimension in the search space into non-overlapping *time windows* so that trips belonging to the same window can be treated as having similar start time and queries for trips with similar start time can be significantly sped up:

1. Hourly Windows. The trip start time is split into 24 time windows based on the hour of the time. For example, trips starting at 2:45 p.m. are considered to be in the 2 to 3 p.m. hourly window. We do not use smaller windows, such as "minutely windows", since they would lead to more windows and take more memory and time to process, without improving prediction accuracy; also, small timestamp fluctuations started to have large effects, causing highly variable answers for similar queries, and the amount of historical data available for each time window dropped significantly, causing extremely low hit rates during prediction.

2. Day-of-Week (DoW) Windows. The trip start time is split by the day of week (Monday, Tuesday, etc.) it belongs to resulting in 7 DoW windows.

3. Hourly DoW Windows. Every hourly window is split further by the day of week to create 24*7=168 finer-grained hourly DoW time windows.

4. Peak Period Windows. The hourly windows and DoW windows are "compacted" into five windows based on taxi fare surcharges: (1) peak traffic hours (7a.m.–10a.m. and 17p.m.–20p.m.) on weekdays which have 35% surcharges, (2) non-peak day hours (6a.m.–7a.m. and 10a.m.–17p.m.) on weekdays, (3) night hours (0a.m.–6a.m.) on weekdays which have 50% surcharges, (4) day hours (6a.m.–2359p.m.) on weekends (no peak hours), and (5) night hours (0a.m.–6a.m.) on weekends which have 50% surcharges.

The effects of time windows on prediction accuracy, efficiency, and hit rate will be illustrated in Section 5.2.

4.3.2 Location Zones

The start and end GPS coordinates of trips obviously affect trip durations and fares, and the whole location space should be split into *zones* of appropriate sizes so that trips with the same start and end zones can be treated as having similar start and end coordinates to speed up our queries. We designed two methods to do this:

Static Zoning: In this method, we establish zones according to pre-chosen zone sizes and use all the similar trips found in these pre-configured zones, called *static zoning*. This method is fast but many particular zones may contain no historical records.

Dynamic Zoning: In this method, we dynamically find the smallest zone that contains the required number of similar historical records for any given query. It is slower but always finds the required number of historical records.

The next section focuses on static zoning while Section 4.3.4 presents dynamic zoning. Section 5.4 discusses the advantages and disadvantages of each method in detail.

4.3.3 Static Zoning

In static zoning, we partition the whole two-dimensional Singaporean map (containing all possible start and end locations) into a series of uniform square areas. To do this, we first found the smallest rectangle that covered all of Singapore — a rectangle 25 kilometres high and 50 kilometres wide. We then split that rectangle into many smaller equally sized squares also known as *location zones*. To understand the effect of zone sizes on prediction accuracy, efficiency, and hit rate, we varied the size of the squares from 50 meters by 50 meters all the way to 5,000 meters by 5,000 meters.

In practice, we found that due to Singapore's irregular shape and road density (Figure 1), quite a few location zones (especially at smaller sizes of 200 meters and below) did not contain any taxi trips at all. To improve efficiency, we thus "compacted" the location zones by removing any zone which did not have a trip either starting or ending in it across the entire 21 month dataset. Table 1 shows the different zone sizes we used along with the total number of location zones created by each size and the effect of our compaction step.

These zones allowed us to quickly convert every trip's GPS start and end coordinates into a specific zone number; i.e., the number of the rectangle containing that portion of the GPS space. This converted the problem of finding trips with similar spatial properties (a slow inaccurate process) into the much easier and faster problem of finding trips with the same integer start and end zone numbers.

The zone size trade-off is that using smaller zones could potentially result in better results (as trips in smaller zones are closer to each other) at the cost of higher computation (as more zones are created) and data sparsity issues (some zones may not contain enough historical data). Section 5.2.1 shows the effects of zone sizes on accuracy and other performance indicators.

Predictors. We combined the location zones with the four time windows (Section 4.3.1) to create five *predictors*. These predictors represent the final time-space partitions of the historical records that we use for making predictions of durations and fares for new trips. The five predictors are "LOC" which uses start and end location zones only to split trips (i.e., no time effects are considered for prediction), "HR" which combines the zones with hourly windows to split trips, "DOW" which combines the zones with DoW windows, "DOW$\times$HR" which combines the zones with both DoW and hourly time windows, and "PEAK" which combines the zones with peak period windows.

Each predictor is thus a large set of partitions of time and locations, which can be implemented as any addressable data container. For example, the LOC predictor with 200m zones contains

Zone Size (meters)	Total No.	After Compaction
50 x 50	565,586	162,730 (71%)
100 x 100	141,148	56,881 (60%)
150 x 150	62,559	31,834 (49%)
200 x 200	35,216	21,346 (39%)
250 x 250	22,374	15,285 (32%)
300 x 300	15,510	11,612 (25%)
350 x 350	11,502	9,197 (20%)
400 x 400	8,804	7,374 (16%)
450 x 450	6,930	6,017 (13%)
500 x 500	5,544	4,960 (11%)

"Total No." is the total number of location zones created for that zone size while "After Compaction" shows the no. of location zones (and the % reduction relative to the previous total) that are left after removing unused location zones.

Table 1: Location Zone Sizes Used

$21{,}346^2 \approx 455$ million partitions (Figure 1) while the DOW predictor with 200m zones contains $21{,}346^2 \times 7 \approx 3.19$ billion partitions.

Implementation. The basic idea is, for each zone size, to partition historical trips according to each predictor and then calculate and store the average historical trip duration and fare for each partition in the predictor based on Equation (1). Then, predicting the duration and fare for a new trip is simply a query of the average trip duration and fare of the partition to which the new trip belongs.

The algorithm below constructs hash tables to store the average trip durations and fares of each partition from each predictor; each entry in a hash table represents a partition and is indexed as follows (lines 3–4): for a zone size z and for a trip t with its start time and start and end GPS coordinates, convert the time to hour of day (h), day of week (d), and peak period (w), and convert the start and end coordinates to the corresponding zone numbers n_s and n_e for z, then the index for the partition containing the trip t is $\langle z, n_s, n_e \rangle$ for LOC predictors, $\langle z, n_s, n_e, h \rangle$ for HR predictors, $\langle z, n_s, n_e, d \rangle$ for DOW predictors, $\langle z, n_s, n_e, h, d \rangle$ for DOW$\times$HR predictors, and $\langle z, n_s, n_e, w \rangle$ for PEAK predictors.

An important feature of this algorithm is that we update the arithmetic means incrementally when new data is added (lines 6–9) without having to store all previous trip details — saving significant amounts of memory at runtime.

```
Algorithm: Construct Trip Prediction Table
input:  T: Trip Data Set
        z, p: Zone Size and Predictor Kind
        (LOC/HR/DOW/DOW×HR/PEAK)
output: P: A Prediction Table
BEGIN:
1:  Initialise P as an empty hash table
2:  For each trip t in T
3:      Based on the zone size z and the predictor kind p,
4:          Extract the index of the partition to which t belongs
5:      Get the entry P[index] which is a 4-tuple: $\langle e_0, e_1, e_2, e_3 \rangle$
6:      If the entry does not exist /* insert a new entry */
7:          $P[index] \leftarrow \langle 1, t.fare, t.duration, t.distance \rangle$
8:      Else /* update the entry */
9:          $P[index] \leftarrow \langle e_0 + 1, \frac{e_0 e_1 + t.fare}{e_0+1}, \frac{e_0 e_2 + t.duration}{e_0+1}, \frac{e_0 e_3 + t.distance}{e_0+1} \rangle$
RETURN P
```

The output of the algorithm is a prediction table for a chosen zone size and predictor kind, which is stored in memory (using efficient hash maps) and/or disk and then queried for the duration and fare of each new trip. If the index of a new trip exists in the table, we return the values stored in that index as prediction values; otherwise, we report an "unsuccessful prediction" for the new trip. The table construction routines and prediction algorithms are both implemented in Java (about 1600 lines of code in total). The prediction table's memory consumption is proportional to the number of location zones and time windows used by that table's predictor (It also increases slightly as more historical data is used)

Also, since we need to evaluate the effects of zone sizes and predictor kinds on on prediction accuracy, efficiency, and hit rate, our code also allows us to load many prediction tables for various predictors into memory at the same time, so that we can compare the prediction accuracy, efficiency, and hit rate of various predictors for each trip easily. Section 5.2 and 5.4 have the detailed results.

4.3.4 Dynamic Location Zones

The second kind of location zones we use have dynamically adjustable sizes for each trip we are predicting. Unlike static zoning, dynamic zoning aims to find, for each new trip, the minimum zone size that can give us a specified number of similar historical trips that we then use to make reasonable predictions. Thus, it has almost-zero "unsuccessful" predictions. We used the nearest neighbour model [3] from data mining community to achieve dynamic zoning.

k-Nearest Neighbour (kNN). Given a set of data points T and a query point t, the *k-nearest neighbours* (kNNs) for t are the k data points in T whose *distances* from t are the top-k shortest among all points in T. Using the k-nearest neighbour model for finding similar trips has these benefits:

- Predefined location zones are no longer required. The approach can naturally collect k "closest" trips for each new trip, no matter whether they are in the same small region or scattered across larger, or multiple regions.
- If k is chosen appropriately, the low hit rates associated with smaller static zone sizes will no longer be an issue, and the effects of anomalous data (assuming most trip data is normal) on prediction accuracies can also be offset by enough normal data among the k trips.

The distance metric. The key to make the kNN approach work for our dataset is to define a reasonable *distance* metric between any two trips. As mentioned earlier in Section 4.2, the three basic features of a trip (start position, end position, and start time) can affect its duration and fare and should be used for calculating the distance metric. Two of the basic features, start and end locations, are GPS coordinations and standard Euclidean distances can be easily computed for them. The main challenge lies in incorporating the last feature, the trip start time, into the distance metric.

Our solution uses a 5-tuple to represent each trip: $\langle SLong, SLat, ELong, ELat, STime \times factor \rangle$, where $SLong$, $SLat$, $ELong$, $ELat$ are the start and end longitudes and latitudes of a trip respectively; $STime$ is the trip start time of day, and $factor$ is a constant that allows us to relate GPS coordinates to time. The $factor$ value is determined intuitively as follows: we estimated that a 0.0001 degree difference in longitude or latitude around Singapore corresponds to about 10 meters, and that a taxi's average speed is about 25 kilometres per hour; thus, one hour is roughly comparable to $25000/10 * 0.0001 = 0.25$ degree difference, and 0.25 may be used as the $factor$. With this solution, the kNN distance metric between any two trips can be computed as standard Euclidean distances between their 5-tuples.

Predictors. Dynamic zoning can also be combined with time windows to create five kinds of *predictors* similar to but different from the predictors for static zoning: LOC which only uses the distance metric to split trips (i.e., queries for nearest neighbours will be carried out in the whole trip dataset), HR which splits the trip dataset by hourly windows before querying for nearest neighbours, DOW which splits trips by DoW windows, DOW×HR which splits trips by both DoW and hourly windows, and PEAK which splits trips by the five peak period windows (c.f. Section 4.3.1).

The indices used for identifying partitions of each predictor are also similar to but different from the indices used in static zoning: for a trip t with a given start time and start and end GPS coordinates, we do not need to index a partition for the LOC predictor since the whole trip dataset is the partition; for other predictors, we convert the start time to hour of day (h), day of week (d), and peak period number (w), then the index for the partition containing the trip t is $\langle h \rangle$ for the HR predictor, $\langle d \rangle$ for the DOW predictor, $\langle h, d \rangle$ for the DOW×HR predictor, and $\langle w \rangle$ for the PEAK predictor.

Implementation. The following algorithm illustrates how we use kNN queries to make fare and duration predictions. We used k-dimensional trees (kd-trees) [20] as our kNN data structure (lines 1–3) as it provides a convenient interface for both creating a kd-tree for a dataset and for returning k nearest neighbours for a given query.

```
Algorithm: kNN Search for Trip Prediction
input:   $T$: Trip Data Set
         $p$: Predictor Type (LOC/HR/DOW/DOW×HR/PEAK)
         $k$: Required number of similar trips
         $t$: A Trip Inquiry
output:  Predicted duration and fare for $t$
BEGIN:
0:  /* construct $kd$-trees (only done once) */
1:  Partition $T$ according to the predictor type $p$
2:  For each trip partition $T'$
3:      Create a $kd$-tree using trips belonging to $T'$

4:  Based on the predictor type $p$,
5:      Extract the index of the partition to which $t$ belongs
6:  Identify the $kd$-tree ($x$) created for the partition indexed by index
7:  If $x$ is empty, report an "unsuccessful prediction"
8:  Else: Get $k$ nearest trips of $t$ from $x$
9:      Calculate the duration and fare of $t$ based on Equation (1)
RETURN
```

Unlike the static zoning algorithms, the dynamic zoning algorithms keep the details of every historical trip in memory to calculate the distances for each query. Its computational memory cost is thus dependent on the size of the historical dataset. It can consume much more memory and be significantly slower than static zoning when additional months of historical data are used. The main benefits of dynamic zoning, though, are flexible zone sizes and significantly higher hit rates. Section 5.4 presents more comparative results.

Finally, kd-trees are not the only way to realise kNN queries. Many other data structures, such as B^{**}-trees [27] and R^{+}-trees [24], can also be used to search spatially near neighbours with varying memory and computational costs. In this paper, although the kd-trees-based dynamic zoning approach was much slower than static zoning, it was still within the expected performance limits and its accuracy and hit rate were better than static zoning (Section 5.3). In the future, we plan to investigate other data structures to determine if they are better suited for our particular dataset and problem.

5. EMPIRICAL EVALUATION

This section presents our experimental setup, and illustrates the effects of various settings (e.g., time windows, zones, and amount of history) on the three main performance indicators of our system: prediction accuracy, efficiency, and hit rate (i.e., the prediction success rate, which is the % of trips having prediction values). The results show that on average, our system can efficiently predict fares to within 1 S$ of the actual fares and durations to within 3 minutes of the actual durations, beyond the expectation of the taxi operator.

5.1 Evaluation Methodology

To evaluate our system, we divided the data into two sets. Set 1 contained the data from January 2009 to August 2010 while Set 2 contained the data for just September 2010. We created 20 different trip subsets from Set 1 that differed in the amount of data they contained: (1) The first trip subset contained all trips from August 2010 (just 1 month of data), (2) the second subset contained all trips from July 2010 and August 2010 (2 months of contiguous data), and so forth, with the 20^{th} trip subset containing all of the Set 1 data (20 months from January 2009 to August 2010). We call these 20 subsets *history sets* and used them with various zone size and predictor settings to generate prediction values.

We used Set 2 as the query data (about 12 million trips) to test our system. In particular, for each trip t in Set 2, we produced predicted durations and fares for t using every predictor configured with various static or dynamic zones and/or time windows (c.f. Section 4.3) and constructed with each history set. We then compared the predicted values with the actual trip values to determine the prediction accuracy under each setting. Also, for each setting, we recorded the prediction speed (the number of predictions made per second), the memory consumption, and the hit rate (the % of trips in Set 2 for which the predictor had enough similar historical records to make a prediction for).

In the rest of this section, we compare the performance of the various settings along different dimensions (e.g., zone size, predictor kind, amount of history used, etc.) to understand the best operational settings for our system.

5.2 Results: Static Zoning

We first present the results of our static zoning approach.

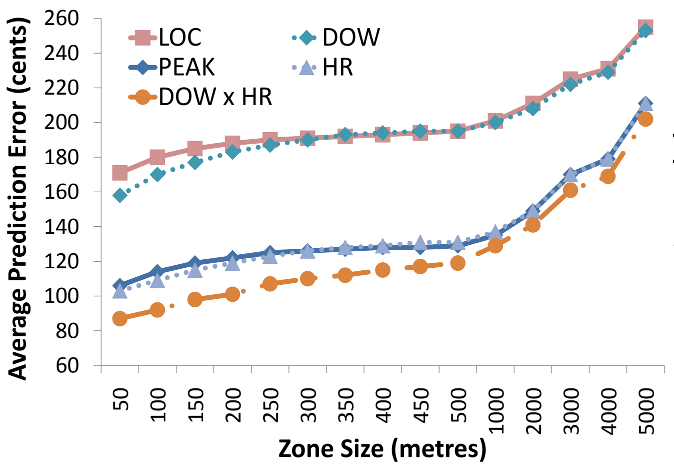
5.2.1 *Time Windows and Zones versus Accuracy*

Figures 2 and 3 show the overall accuracies of our system with respect to various time windows and location zones of different sizes, using the 1-month history set. The y-axis of both figures are the average absolute differences between the predicted values and the actual values of every test trip, which we call *average prediction errors*. We observe in Figure 2 that the average prediction errors for fares go from 0.87 S$ for 50m zones to 2.53 S$ for large 5km zones. These results, especially those with errors within 1 S$, greatly exceed the expectation of the taxi operator (Section 3).

The system also predicts trip durations well with average errors ranging from slightly over 2 minutes (50m zones) up to 4 minutes (5km zones). However, the duration prediction errors also showed higher standard deviations, indicating that anomalous predictions could occur more often.

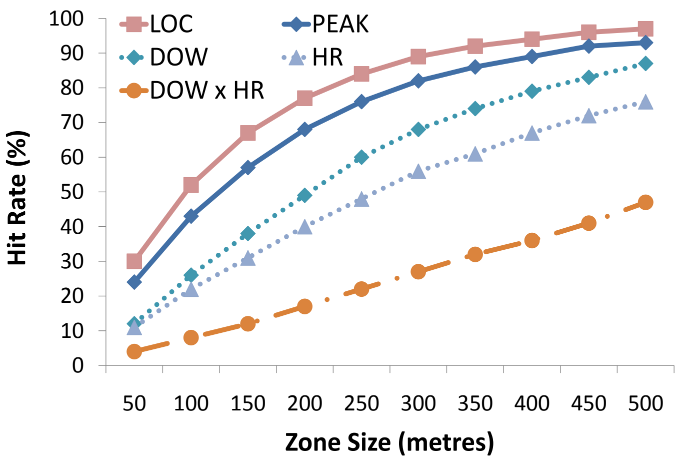
5.2.2 *Prediction Success Rate*

From Figures 2 and 3, one may be tempted to use 50m zones all the time. However, as Figure 4 shows, when static location zones get smaller, the hit rate (i.e., the prediction success rate, which is the % of test trips having a non-empty entry in a corresponding prediction table) goes down significantly. In particular, the rate for



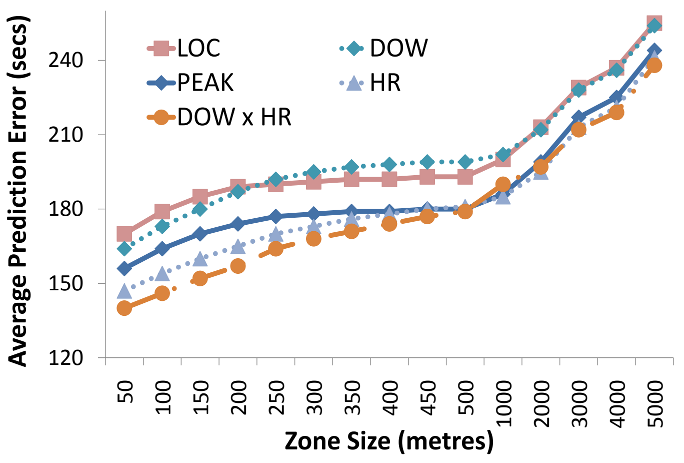
The max. std. deviation of the prediction errors was 25%.

Figure 2: Fare Prediction Errors Versus Zone Sizes



There are no error bars for this graphs. The values are exact.

Figure 4: Predication Success Rate Versus Zone Size



The max. std. deviation of the prediction errors was 70%.

Figure 3: Duration Prediction Errors Versus Zone Sizes

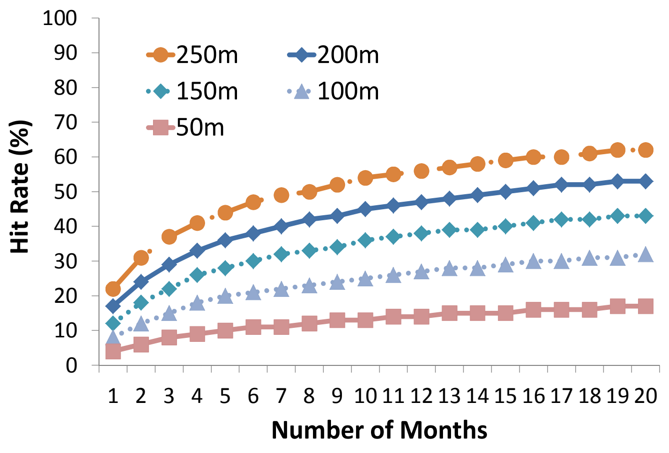
the DOW×HR predictor with 50m zones is a paltry 4% with 1 month of data.

5.2.3 *Effect of Amount of History*

As suggested by previous results, we may not be able to use the 50m zones due to low hit rates. Figure 5 further shows the hit rates, for zones of various sizes, as we add up to 20 months of historical data (Jan 2009 to August 2010).

We observe that hit rates go up consistently with more historical data for all zone sizes. However, DOW×HR with 50m zones still has very low hit rate (about 17%) for 20 months of data. Trading accuracy for hit rate, we notice that the PEAK predictor with 200m zones is the best predictor with an average fare prediction error of 1.2 S$, an average duration error of 2.8 minutes, and a hit rate of 93%.

On the other hand, we observe that prediction accuracies do not necessarily go up when more months of data is used. Table 2 illustrates this for the DOW×HR and PEAK predictors using 200m zones. This effect is understandable as aggregate traffic patterns, driver behaviour, fare structures, etc. do not usually change that quickly.



The plotted lines are for the DOW×HR predictor.

Figure 5: Amount of History versus Hit Rate

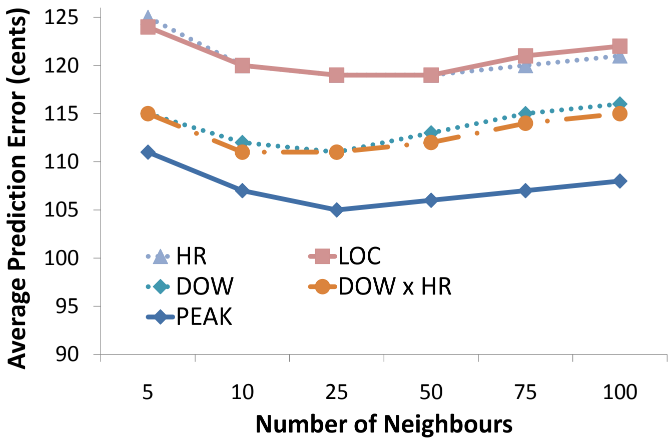
5.2.4 *Further Evaluation and Results Summary*

We also conducted various sensitivity tests that we only briefly describe here in the interest of space. First, we used different months as the test set (with the other months as the training set) to check if our current test (September 2010) and training set partition (remaining months from January 2009 to August 2010) was causing unusual results. We found no significant differences in our results when we used other months as the test set. We then partitioned the trips data according to the type of taxi that was used to service that trip and found that this partitioning had no effect on the results. Next, we tested various values for the minimum number of historical trips (e.g., three historical trips versus 10 historical trips) that needed to have been used to compute that prediction table index before a successful prediction could be made. We found that, with small zones, the number of historical entries used to compute a prediction table index had little effect as the index was describing a small enough portion of time and space that unusual variations were uncommon. Finally, we tested the effect of on-line learning where we make a prediction for a new trip and then immediately augment the prediction tables using the real data of that trip record. We found that this had minimal effect on the prediction accuracy and a very small positive effect on the prediction success rate.

No. of	DOW×HR Diff.		PEAK Diff.	
Months	Fare (cents)	Duration (s)	Fare (cents)	Duration (s)
1	101	157	122	174
5	104	160	121	172
10	106	162	120	171
15	106	162	120	170
20	107	163	120	170

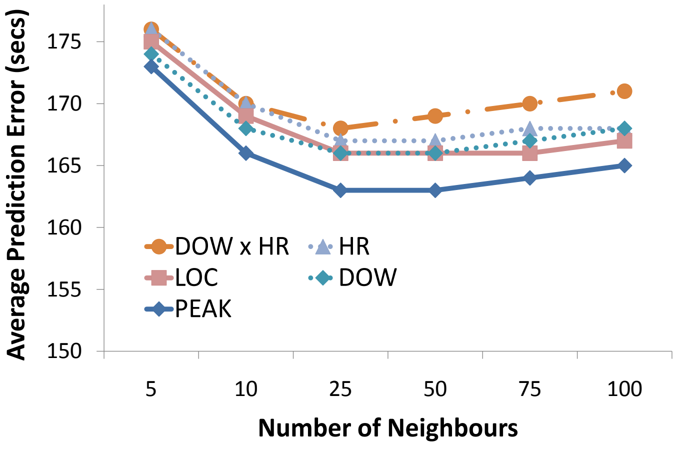
This table shows the accuracies of DOW×HR and PEAK predictors using 200m zones and up to 20 months of history.

Table 2: Amount of History and Accuracies



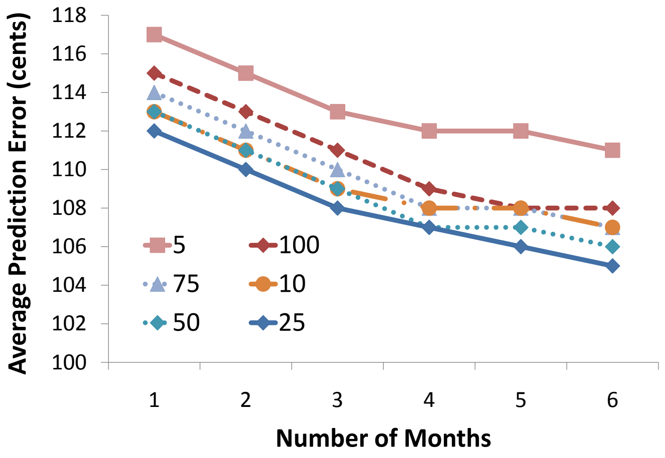
The max. std. dev. was 18% with a mean value of 13.7%.

Figure 6: Fare Errors vs. No. of Neighbour Trips



The max. std. dev. was 56% with a mean value of 46.37%.

Figure 7: Duration Errors vs. No. of Neighbour Trips



This plot is for the PEAK predictor with various k. The max. std. dev. was 16% with a mean value of 13.28%.

Figure 8: Amount of History vs. Fare Errors

Overall, our system based on static zoning satisfies all the performance requirements (accuracy, speed, and low computational cost). However, the performance depends on choosing the right zone size, and the right predictor that best matches the data characteristics — determining these accurately can be tedious in practice. In the next section, we evaluate dynamic zoning which can automatically determine the optimal zone size for a given set of historical and test data.

5.3 Results: Dynamic Zoning

We use the same accuracy metric as Section 5.2 for dynamic zoning. Figures 6 and 7 show the overall accuracies of dynamic zoning with various predictor types and various numbers of neighbour trips (k) using the 6-month history set. We observe that k has fluctuating effects on accuracy. This is due to the fact that trips vary from each other so much that using too few similar trips cannot sufficiently offset the variations, but using too many may introduce variations from unrelated trips. Overall, we empirically decided $k = 25$ is an optimal choice for most of our settings.

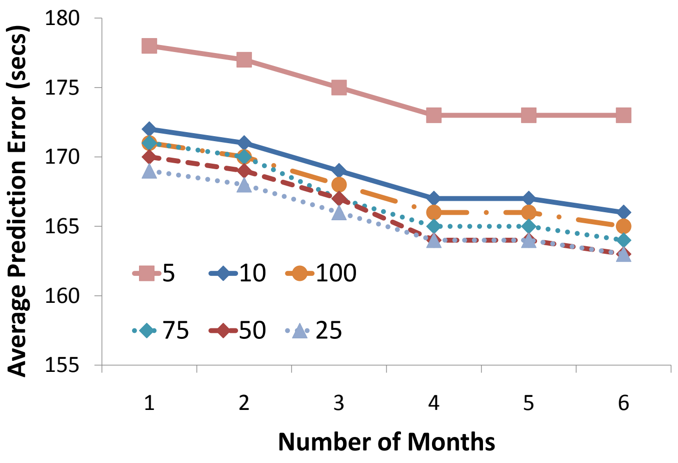
We observe that the average fare prediction errors range from 1.05 S$ to 1.25 S$ for $k \in \{5, \ldots, 100\}$. The average duration prediction errors are larger but still within the acceptable 3-minute range. These results are comparable to the 200m zone results in static zones (Figures 2 and 3) except that dynamic zoning always has 100% hit rates, while static zoning may have very low hit rates.

The amount of history used in dynamic zoning has little effects on prediction accuracy when k is set appropriately. Note that the accuracy improvements shown in Figures 8 and 9 are very small: fare and duration predictions are improved by up to 15 cents and 15 seconds respectively, which are negligible in practical uses. This is one benefit of dynamic zoning since it is almost always able to find enough neighbour trips for prediction calculation.

5.4 Pros and Cons: Static vs. Dynamic Zoning

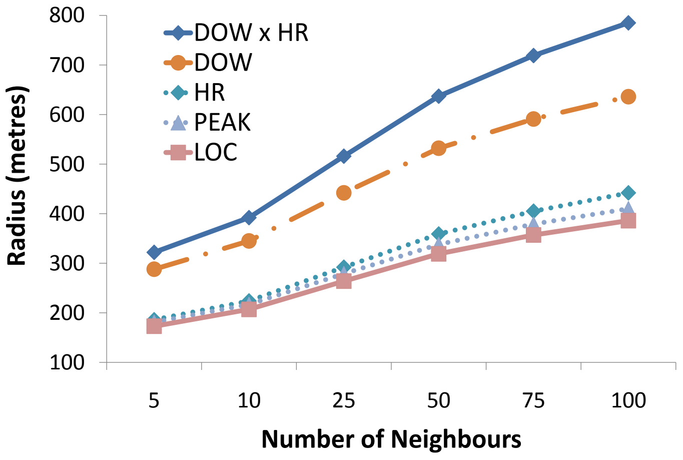
In addition to the difference in hit rates, static and dynamic zoning have various characteristics that balance their prediction accuracy and efficiency differently.

In terms of location zone sizes, static zoning requires pre-chosen zone sizes, while dynamic zoning does not require explicit zones; it only requires k trips which may form *implicit* zones. As a comparison, Figure 10 shows the average size of the implicit zones, which are the average radii of the k nearest neighbours for all test trips. The average radii were calculated as follows: Given a trip t and its k neighbours $\{t_1, \ldots, t_k\}$, we call the *maximum* spatial distance between t and its neighbours the *radius* for t, while the distance between two trips t and t_i here is defined as the *larger* of the distance between the two start locations and the distance between the two end locations; then, the average radius for all test trips under



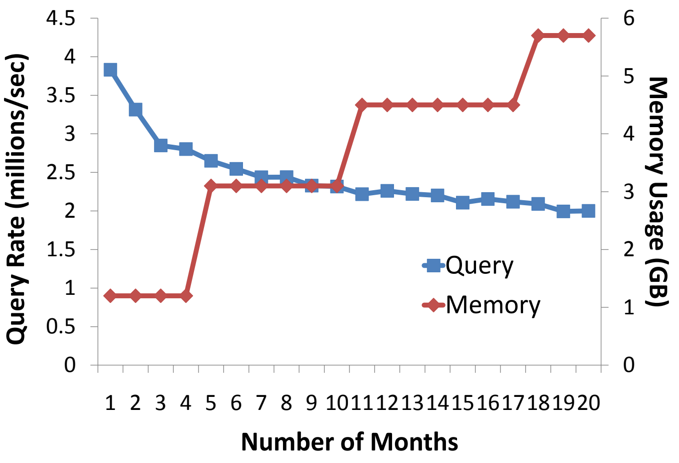
This plot is for the PEAK predictor with various k. The max. std. dev. was 56% with a mean value of 46.92%.

Figure 9: Amount of History vs. Duration Errors



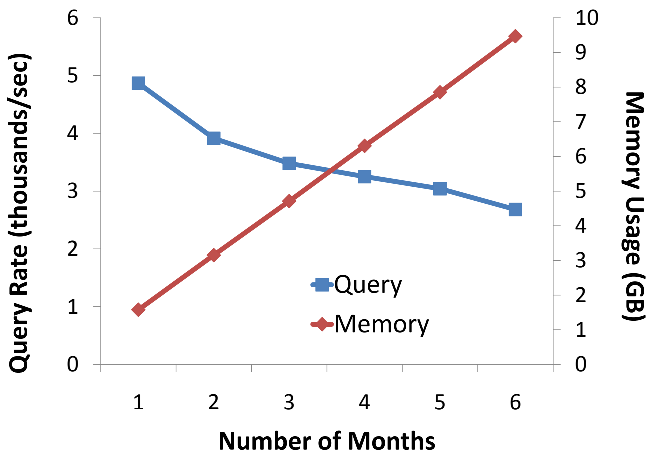
y-axis is the average sizes of implicit zones used by dynamic zoning for each of the five predictors with various k. The max. std. dev. was 741m with a mean value of 375m.

Figure 10: Radii versus Number of Neighbour Trips



The left y-axis is the average number of queries carried out per second DOW×HR with 200m zones; the right y-axis is the corresponding memory cost.

Figure 11: History vs. Performance–Static Zoning



The left y-axis is the average no. of queries carried out per second for the PEAK predictor with $k=25$ and up to 6 months of data; the right y-axis is the corresponding memory cost.

Figure 12: History vs. Performance–Dynamic Zoning

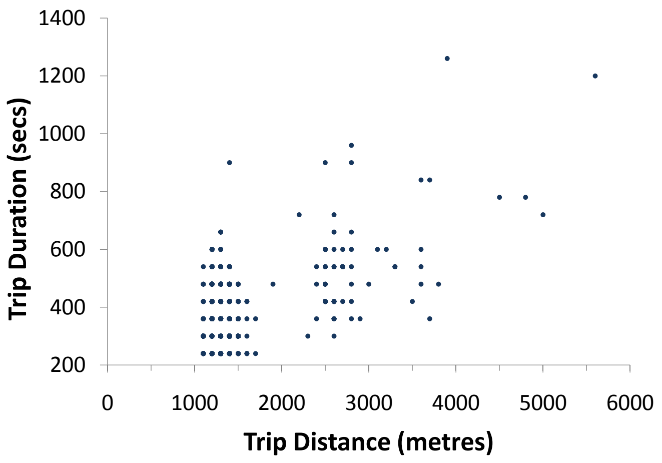
a particular setting can be calculated. We see, as expected, that a smaller k leads to a smaller radius.

In terms of prediction performance, static zoning is more efficient than dynamic zoning as predictions can be calculated incrementally without storing individual trip details, while dynamic zoning needs to store details of every trip (in the kd-trees) to decide neighbour trips (a more time and memory intensive process). The more months of data used, the more memory dynamic zoning will use and the slower the queries will be, while the memory consumption of static zoning only depends on zone sizes and predictor types and its queries take constant time.

As mentioned in Section 4.3.3, we used sparse hash maps to store the static zoning prediction tables. Our experiments used only up to 2GB of memory, regardless of the number of months of data used, for zones greater than 400 meters in size. For the smaller zones (400 meters and below) and more months of data, we exploited the spatial locality of the entries in the prediction tables and stored parts of the hash maps on disk when memory consumption reached certain thresholds, only keeping memory mapped pointers to those disk blocks in memory. In such cases, our experiments used up to 6GB of memory only. For dynamic zoning, the memory cost is proportional to the number of months of data but not related to the time windows used. In our experiments, with up to six months of data, dynamic zoning required less than 2GB of memory for each month.

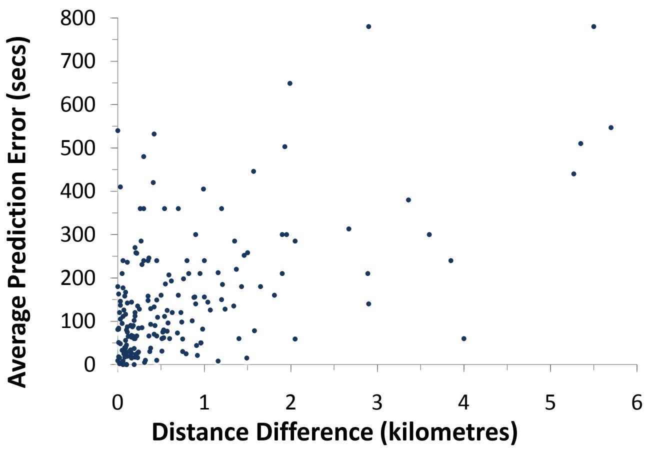
In terms of speed, static zoning on average returns the result for a prediction query within half a microsecond (μs) when the prediction tables are either in memory or disk, while dynamic zoning on average returns a query within half a millisecond (ms) — capable of real-time performance in all cases (and meeting the taxi operator's requirements). Figures 11 and 12 illustrate the performance of static and dynamic zoning respectively in more details.

A very interesting idea for future work is to improve our system's accuracy and efficiency by using static zoning with varying zone sizes at different locations, with the zone size determined by the radii from dynamic zoning. This might give us the high hit rate and accuracy of dynamic zoning with the low memory consumption and speed of static zoning.



This plot is for a sampled set of trips belonging to a trip partition generated by the PEAK predictor with 200m zones.

Figure 13: Trip Distances versus Durations



This plot is for a sampled set of test trips belonging to a trip partition generated by the PEAK predictor with 200m zones.

Figure 14: Distance Deviations vs. Duration Errors

5.5 Accuracy Analysis

Even though the prediction accuracy of our system exceeds the requirements, it can still be improved significantly. In this section, we present sources of prediction errors that can be eliminated.

In our solutions, for either static or dynamic zoning, we have only taken the three basic features into account. However, there are intuitively many other factors that could affect trip durations and fares, including but not limited to (1) indirect routes, either taken by honest or dishonest taxi drivers or due to special requests by passengers (e.g., multi-leg trips whose intermediate destinations are not recorded in the trip records), and (2) traffic conditions.

The main challenge for addressing these factors is that our dataset contains no direct information about them. Thus, we need to infer them from other available information.

5.5.1 Routing

We observed that different routes often have different distances, and that these distances appear in the trip dataset. We can thus easily distinguish trips with different routes from each other by measuring the differences among their travelled distances, without needing to track the actual path of each taxi using its location data (c.f. Section 2.2). It is also computationally much cheaper to use distance deviations to distinguish trips with different routes than to identify actual routes by tracking the physical movement patterns of every taxi. We show in the following section that our prediction accuracies are only slightly affected by alternative routes.

We first examined the correlation between distances and durations of historical trips. We observed that distances and durations can vary a lot — even for trips that started at the same time and start and end locations. Figure 13 shows a scatter plot for a set of such trips, where the distances and durations can differ by up to 6 kilometres and 1000 seconds. Taxi fares have similar large variations. This illustrates an inherent reason affecting our prediction accuracy.

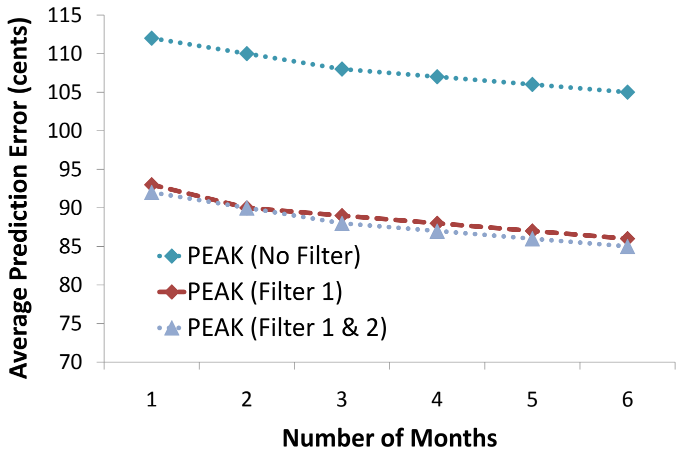
We examined some of the trips having anomalously long distances or durations and noticed several kinds of anomalous routing behaviour. For example, a passenger going from location *A* to *B* may instruct a taxi to wait at *A* for an extended period first while the fare-meter is kept running. Such a trip may show a normal distance between *A* and *B* but an anomalous high trip duration and fare. For another example, a passenger may go from location *A* to a close location *C* via another location *B* which is far away from both *A* and *B*, but the whole trip is only recorded as a trip from *A* to *C*, causing anomalously high trip distance, duration, and fare.

These observations prompted us to filter trips with "anomalous" distances and durations from the dataset used for prediction in the following ways: (1) trips with distances 2 times longer than the straight distance between start position and end position are filtered; (2) trips with average speeds slower than 20 km/h or faster than 100 km/h are filtered (speed is estimated by dividing trip distances by durations). We picked the lower and upper bound speeds after analysing the average speed for all trips in our dataset. We found that these cutoffs were able to eliminate outliers without removing trips that were slowed by regular traffic conditions. In particular, taxis did not normally average less than 20 km/h for any trip (even if particular stretches of the trip were slowed by traffic during peak hours etc.). The main exceptions we discovered were due to a) unusual congestion due to accidents or weather (tree falling across the road etc.), or b) the taxi waiting at some location (presumably for the passenger to finish some activity) during the trip for a long period of time.

In addition, we examined the correlation among actual trip distances, durations, fares, and the prediction errors of trips using our Set 2 dataset. We observed that the prediction accuracy for each test trip is related to the "irregularity" of the trip: when the trip distance or duration is anomalously long or short, its prediction error is also much higher than the average.

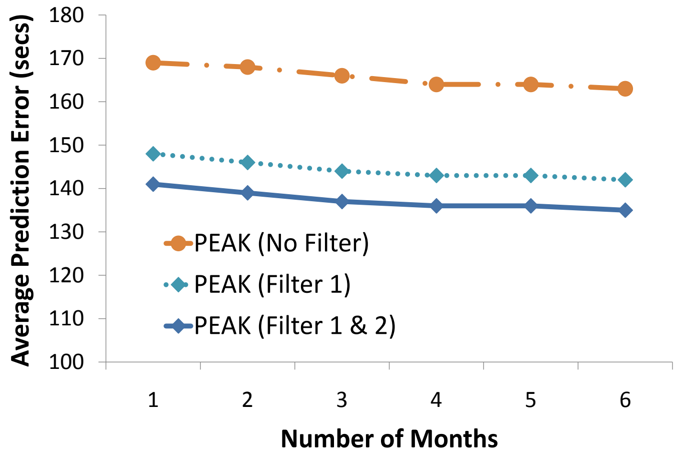
Figure 14 illustrates the relation between duration prediction errors and the distance deviation of each trip where distance deviation is defined as the difference between the straight line distance of the trip and the actual trip distance. We found that more than 75% of the trips were condensed in a small area (180 seconds by 1 kilometre) near the origin, while about 10% of the trips had fairly anomalous distances (deviating more than 3 kilometres) and durations (deviating more than 300 seconds). We observed the same behaviour for fare prediction errors. These results validated the use of the filters mentioned above as just a small percentage of trips had extremely high deviations. An acceptable practical implication of these irregularity filters is that our system may not be suitable for predicting the durations or fares of trips that require anomalous routes.

Overall, 9.5% of trips from our entire dataset were filtered by the filter (1), and 21.0% of the trips were filtered by both filters (1) and (2). An immediate benefit of such irregularity filters is improved



Error may be reduced, up to 25 cents, by the irregularity filters.

Figure 15: History vs. Fare Errors



Error may decrease, up to 30s, with the irregularity filters.

Figure 16: History vs. Duration Errors

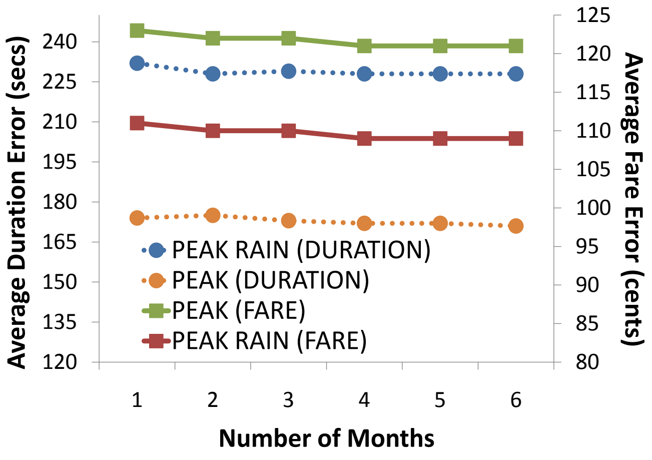
prediction accuracies for "normal" trips. We performed the same set of experiments as those shown in Section 5.2 and 5.3, and fare and duration prediction errors were reduced by up to 20 cents and 30 seconds, respectively, for both static and dynamic zoning, as illustrated in Figure 15 and 16. Also, the filter (1) achieved similar accuracies to the filter (2) without dropping as many trips.

In summary, even though we had no direct information about the route taken by each trip, we could efficiently infer the normality of the route by examining the trip distance and duration. We incorporated this analysis into our system resulting in improved prediction accuracy. Also, this analysis has an additional potential benefit for passengers: our system now better identifies the expected fares and durations for "normal" trips, hopefully eliminating anomalous routing from dishonest drivers.

5.5.2 *Traffic Conditions*

The second main error inducing factor we considered was traffic conditions such as (1) peak hours, (2) special events in the city, and (3) road conditions (e.g., weather, accidents, construction).

Our time windows have taken the peak hours into account and indeed improve prediction accuracy. For the other factors, we were only able to test the effect of rain on the accuracy. Note that Singapore frequently experiences torrential tropical thunderstorms that can severely affect road conditions. The lowest monthly rainfall is about 5.5 inches and the highest is about 12 inches (c.f. `http://www.weatherunderground.com`). This is significantly higher and more consistently wet than most cities in the world.



This plot is for the PEAK predictor with 200m zones. Fare lines use the right y-axis and duration lines the left y-axis

Figure 17: Effect of Rain on Prediction Accuracy

To test the effect of thunderstorms on our predictions, we obtained hourly rainfall data, from March 2010 to September 2010, from Singapore's National Environment Agency. This data was from 80 reporting stations.

With this data, we classified each trip that occurred within the 6 month window into either a raining or non-raining trip. However, the rain data does not contain the area that was affected by the rain or the exact length of the rain. It also does not directly say, for each trip, whether it rained throughout the entire trip route either. We thus conservatively estimated that each reporting station covered a 1km by 1km location zone, and classified a trip as raining only if the rainfall in the start zone exceeded a certain threshold in the same hour when the trip started *and* the rainfall in the end zone exceeded the threshold in the same hour when the trip ended.

We then compared the prediction accuracies of various predictors for raining trips and non-raining trips. Figure 17 shows that rain indeed has effects on prediction accuracy: the duration errors for raining trips only were higher by 60 seconds on average, while fare errors were reduced by 12 cents. The results are reasonable since raining trips have less distance deviations due to less specially requested routes but more duration deviations due to affected traffic conditions. On the other hand, since the raining trips account for only about 0.6% of all trips (because of our conservative estimation), the prediction errors for non-training trips were almost the same as the errors for all trips. We plan to improve predictions for raining trips when we obtain more detailed historical rainfall data and integrate trip predictions with weather prediction services.

5.6 Summary of Results

Overall, our system exceeds the performance requirements stated in Section 3. In particular, our dynamic solution with 6 months of history and the best error filters achieves real-time performance (Figure 12) with fare and duration errors of under S$ 0.90 (Figure 15) and 2 $\frac{1}{2}$ minutes (Figure 16) respectively. We also found that i) a more efficient (Figure 11) static approach may not be sufficient due to serious data spareness issues and low hit rates (Figures 4 and 5), and that ii) domain-specific factors that affect prediction accuracy, such as indirect routing (Section 5.5.1) and rain (Section 5.5.2), need to be identified and mitigated.

6. DISCUSSION

6.1 Deployment in a Real Environment

Our dynamic zoning solution has been installed at the taxi operator's site. This section describes the process, challenges and lessons learned from this real-world deployment.

Real systems have many non-uniform parts: We quickly learned that the IT systems used in transportation companies tend to be a mix of systems from different vendors. As such, our system had to be modified to operate seamlessly in this mix-and-match environment. For example, we needed a separate data collection module to integrate with the operator's live data feed and a separate web service component to output our results into the operator's call centre system (for dissemination to customers and booking call operators).

Integrating the user: Our system predicts the expected fare and time of a trip. However, as stated earlier, the average fare error can be up to 1 S$ and the time error up to 3 minutes. Should these errors be shown to the end user? If so, how? In consultation with the operator, we decided to tell the customer that the predicted fare/time would likely fall between just two values; 1) the predicted value and 2) the predicted value + the average error. The operator felt that this was best as customers would probably be happy about lower-than-expected fares and times but get annoyed about higher-than-expected fares and times.

Real-world deployments take time: It has taken about 6 months to deploy one version of our system. This was much longer than we expected as the process involved code-reviews and deployments to multiple test environments before deployment to the live environment. For example, we were first given access to a small test server, then given access to a sand boxed development server (after passing a code review), and finally given access to the secure production server after another code review. We have since realised, from talking to colleagues at various industry research labs, that this process is fairly typical.

Live environments are not easily changed, accessed, or instrumented: The currently deployed system is an older version that does not have the error filters described in Section 5.5, as each new version needs to be approved before it can be deployed and this takes time (weeks or even months). We are currently getting our latest version approved. In addition, all results in this paper were generated from tests conducted in our research lab and not at the operator's site because (1) we had to physically visit the operator's data centre to access the production servers — offsite access was not possible, and (2) we could not instrument the code on the production server as the operator did not want our instrumentation to slow down their production environment in any way. These reasons also turned out to be fairly typical.

6.2 Applicability to Other Domains

The solutions presented have a few domain-specific components (e.g., the distance metric in the nearest neighbour search and the specific error filters used). However, beyond those components, the various algorithms should be applicable to other problems in the transportation space that involve finding patterns in a large body of data. Indeed, we plan to apply the same techniques to other transportation problems such as scheduling bus routes, and planning an optimal driving route for a courier delivery service.

7. RELATED WORK

Our work shares similarities with and draws inspiration and ideas from prior work in two different areas: (1) systems work focusing on various aspects of transportation networks such as traffic analysis, privacy preservation, and ad-hoc network creation, and (2) prior analytical work on taxicab and transport networks.

Systems Research. Our work is most closely related, at least data-wise, to the analysis performed by Liao [18]. In particular, he used similar GPS data from a Singaporean taxi operator for his analysis. However, our work extends into areas (such as passenger demand prediction) that were not previously tackled. Min et. al [19] also used taxi data coupled with real-time traffic information (from pressure sensors under the road and traffic cameras) to create a predictive traffic model for Singaporean roads. Our work looks at a different aspect of the same data.

Our work is also similar to prior traffic behaviour analysis; including both deployed commercial traffic monitoring and intelligent routing systems such as Inrix [14], Intellione [15], Onstar [21], and TeleNav [26] as well as research systems. For example, Yoon et al. [30] use a combination of vehicles equipped with GPS technology plus low-bandwidth cellular updates to dynamically estimate street traffic. Cayford and Johnson [4], Chen and Chien [5], Turner and Holdener [29], also provide solutions for determining traffic conditions using vehicles as probes. Our work uses similar techniques but differs in at least one of these attributes: (a) the domain (taxi network), (b) the specific analysis (trip information), and (c) the volume of data analysed (21 months of data).

Looking a little further upfield, our work can be neatly complemented by the ongoing research in track tracing and privacy preservation of traffic and location information. For example, Gruteser and Grunwald's [10] work in maintaining anonymity in location-aware systems, Hoh et. al's work on anonymising real-time traffic updates [12] and GPS traces [13], and the StarTrack [11] system that might allow us to use, in computationally efficient ways, the taxis' location data in our prediction models.

Analytical Models. There is a substantial amount of prior work in developing analytical models of transport networks and taxicab markets in particular. For example, in the area of traffic signal optimisation, Dresner and Stone presented schemes to (a) allow emergency vehicles to go through signal junctions faster [8], and (b) to improve general throughput at intersections [9]. Separately, Bazzan [1] and Oliveira et. al [6] showed that it is possible to effectively control a series of distributed traffic signals. Finally, Tumer and Agogino [28] showed that agents could be used to dynamically reduce congestion in an air traffic network.

In addition, economic principles have been used by Beesley [2], and Schroeter [23] explain the theoretical underpinnings of the taxicab market. These theories and models have provided solid theoretical insights, but estimating and testing them has been difficult due to a lack of empirical data. Schaller [22] helps address this shortcoming by assembling a data set based on taxi meter and odometer readings that the New York City Taxi and Limousine Commission collects in its periodic inspections of New York taxicabs. Our work complements these prior analytical studies. In particular, we hope to (a) use the techniques developed in these studies to aid our analysis, and (b) provide real-world data for theoreticians to develop even better traffic models.

8. CONCLUSION AND FUTURE WORK

This paper describe the design, testing, iterative improvements, and real-world deployment of a real time trip information system for providing passengers with accurate predicted taxi fares and trip times that is accurate, fast, and can run on commodity server hardware. This process made us re-learn many of the systems building "rules of thumb", namely: (a) reducing the data size through aggregation and smart filtering is essential to real-time performance on commodity hardware (especially with hundreds of millions of records), (b) real world data needs to be cleaned before use, (c) avoid the seduction of complexity — simple algorithms are frequently good enough and make it much easier to debug problems (especially when the output contains millions of records), (d) proper partitioning of large amounts of data is vital to achieve both runtime-efficiency as well as a high hit-rate, (e) don't optimise prematurely — once you know what you need, you can develop specific simple solutions, using smart data structures, that allow better runtime performance, and (f) deploying a research prototype into a real production environment requires far more work than we naively expected. We hope these observations will prove useful in reducing the systems building time for other researchers and developers.

9. ACKNOWLEDGEMENTS

The authors would like to thank Jason C. Woodard, Darshan Santani, David Lo, Ankit Guglani, and Jacek Szymczyk for their invaluable help with earlier versions of this work. We would also like to thank the anonymous reviewers and our shepherd, David Wetherall, for their help in improving this paper.

10. REFERENCES

[1] Bazzan, A. L. A distributed approach for coordination of traffic signal agents. *Autonomous Agents and Multi-Agent Systems*, 10(1):131–164, 2005.

[2] Beesley, M. E. and Kemp, M. A. Urban transportation. In Mills, E. S., editor, *Handbook of Regional and Urban Economics, Volume II*, chapter 26, pages 1023–1052. Elsevier Science Publishers, 1 edition, Dec. 1987.

[3] Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. When is "nearest neighbor" meaningful? *International Conference on Database Theory (ICDT)*, Jerusalem, Israel, Jan. 1999.

[4] Cayford, R. and Johnson, T. Operational parameters affecting the use of anonymous cell phone tracking for generating traffic information. *Transportation Research Board Annual Meeting*, Washington, DC, Jan. 2003.

[5] Chen, M. and Chien, S. I. J. Determining the number of probe vehicles for freeway travel time estimation. *Transportation Research Record*, 1719:61–68, 2000.

[6] de Oliveira, D., Bazzan, A. L. C., and Lesser, V. Using cooperative mediation to coordinate traffic lights: a case study. *International Conference on Autonomous Agents and Multiagent systems (AAMAS)*, Utrecht, Holland, July 2005.

[7] Department of Statistics Singapore. *Statistics Singapore - Census of Population 2010, Households and Housing*, Feb. 2011. `http://www.singstat.gov.sg/pubn/popn/c2010sr2.html`.

[8] Dresner, K. and Stone, P. Human-usable and emergency vehicle-aware control policies for autonomous intersection management. *Workshop on Agents in Traffic and Transportation (ATT)*, Hakodate, Japan, May 2006.

[9] Dresner, K. and Stone, P. Multiagent traffic management: Opportunities for multiagent learning. *Workshop on Learning and Adaption in Multi-Agent Systems (LAMAS)*, Utrecht, Holland, 2006.

[10] Gruteser, M. and Grunwald, D. Anonymous usage of location-based services through spatial and temporal cloaking. *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Francisco, CA, May 2003.

[11] Haridasan, M., Mohomed, I., Terry, D., Thekkath, C. A., and Zhang, L. Startrack next generation: A scalable infrastructure for track-based applications. *Symposium on Operating System Design and Implementation (OSDI)*, Vancouver, Canada, Oct. 2010.

[12] Hoh, B., Gruteser, M., Herring, R., Ban, J., Work, D., Herrera, J.-C., Bayen, A. M., Annavaram, M., and Jacobson, Q. Virtual trip lines for distributed privacy-preserving traffic monitoring. *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, Breckenridge, Colorado, June 2008.

[13] Hoh, B., Gruteser, M., Xiong, H., and Alrabady, A. Preserving privacy in GPS traces via density-aware path cloaking. *Computer and Communications Security (CCS)*, Alexandria, Virginia, Oct. 2007.

[14] Inrix. `http://www.inrix.com`, 2006.

[15] Intellione. `http://www.intellione.com`, 2006.

[16] Land Transport Authority. *Electronic Road Pricing Rates in Singapore*, Dec. 2010. `http://www.onemotoring.com.sg/publish/onemotoring/en/on_the_roads/ERP_Rates.html`.

[17] Land Transport Authority of Singapore. *Breakdown of Taxi Ownership for 2010*, Sept. 2010. `http://www.lta.gov.sg/corp_info/doc/Taxi%20Info%20for%20LTA%20Website%202010.pdf`.

[18] Liao, Z. Real-time taxi dispatching using global positioning systems. *Communications of the ACM*, 46(5):81–83, 2003.

[19] Min, W., Winter, L., and Amemiya, Y. Road traffic prediction with spatio-temporal correlations. Technical Report RC24275 (W0706-018), IBM Research Watson, Yorktown Heights, New York, June 2007.

[20] Moore, A. *Efficient Memory-based Learning for Robot Control*. PhD thesis, Technical Report UCAM-CL-TR-209, University of Cambridge, Cambridge, England, Oct. 1990.

[21] Onstar. `http://www.onstar.com`, 2006.

[22] Schaller, B. Elasticities for taxicab fares and service availability. *Transportation*, 26(3):283–297, 1999.

[23] Schroeter, J. R. A model of taxi service under fare structure and fleet size regulation. *Bell Journal of Economics*, 14(1):81–96, 1983.

[24] Sellis, T. K., Roussopoulos, N., and Faloutsos, C. The r^+-tree: A dynamic index for multi-dimensional objects. *International Conference on Very Large Data Bases (VLDB)*, Brighton, England, Sept. 1987.

[25] SMRT. *Taxi Fare Structure in Singapore*, Dec. 2010. `http://www.smrt.com.sg/taxis/fares.asp`.

[26] TeleNav. `http://www.telenav.net`, 2004.

[27] Toptsis, A. A. b^{**}-tree: A data organization method for high storage utilization. *International Conference on Computing and Information*, pages 277–281, 1993.

[28] Tumer, K. and Agogino, A. K. Distributed agent-based air traffic flow management. *AAMAS*, 2007.

[29] Turner, S. M. and Holdener, D. J. Probe vehicle sample sizes for real-time information: The houston experience. 1995.

[30] Yoon, J., Noble, B., and Liu, M. Surface street traffic estimation. *International Conference on Mobile Systems, Applications, and Services (MobiSys)*, San Juan, Puerto Rico, June 2007.