Real-Time Trip Information Service for a Large Taxi Fleet

Rajesh Krishna Balan, Nguyen Xuan Khoa & Lingxiao Jiang
School of Information Systems
Singapore Management University
• Predicting:
  – Expected fare
  – Trip Duration
Challenges

• Huge Data
• Real-time results (ms)
• Highly variable taxi fare structure
• How much data is needed for prediction?
• Noisy data
• Error to be within $2$ and 5 mins
Related Work

• Real-time taxi dispatching using global positioning systems (Liao. Z.) – 2003

• Road traffic prediction with spatio-temporal correlations (Min, Winter, & Amemiya) – 2007

• Surface street traffic estimation – MobiSys (Yoon, Noble, & Liu) – 2007

• Differences: (a) the domain (taxi network) (b) specific analysis (trip information) or (c) the volume of data analysed
Commercial traffic monitoring & intelligent routing systems

Global Leader of Connected Car Services and Movement Analytics

Big data, analytics and innovation to help make roads better, commutes faster and people safer

LEARN MORE
Commercial traffic monitoring & intelligent routing systems

IMS Traffic Intelligence
Get on top of traffic before it slows you down

More real time comprehensive data than any other system on the market.

Improve Traffic Movement:

Quality of life can be impacted in cities where infrastructure hasn't grown as fast as the population. Traffic problems affect everything from commute times to productivity, and can diminish resident satisfaction.

In truth, the future prosperity of your city and the quality of life for your residents depend on a properly managed traffic infrastructure.
Data

- 21 Months duration
- 250 million trip records
- 15000 taxis/35000 Drivers
- Start/Stop GPS locations/times
- Distance travelled/meter fare
- Mean fare prediction error: under 1 Singaporean 
- Mean duration error: under 3 mins.
Why the Singapore’s Taxi fare system is complicated?

• Meter to be used always.
• Fixed starting fare & time/distance based charges
• Booking by phone or SMS
• ERP Zones
• Time & location based surcharges
Plot of taxi locations using 10,000 randomly selected GPS points from a single day’s data
Failed Solutions

- Google Maps API
- PostgreSQL with the PostGIS
Trip History

• 3 basic features for each trip:
  – start location
  – end location
  – start time

• Time Windows:
  – Hourly - 24
  – Day-of-Week (DoW) - 7
  – Hourly DoW - 24x7
  – Peak Period (peak, non-peak, day, night)
Location Zones

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

- Time-Space Partitions (Zones + Time-Windows)
  - LOC
  - HR
  - DoW
  - DoW x HR
  - PEAK
System Design (Static Zones)

• Training:
  – Partition historical trips according to each predictor
  – Store average historical trip duration/fare for each partition in the predictor.

• Prediction:
  – Find the partition P to which the trip belongs
  – Return average trip duration/fare of P
  – If not found, show ‘not found’
Algorithm: Construct Trip Prediction Table

input:  \( T \): Trip Data Set
        \( z, p \): Zone Size and Predictor Kind
        (LOC/HR/DOW/DOW \times 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 \( P \)
Indexing the hash-table

- $<z,ns,ne>$ for LOC
- $<z,ns,ne,h>$ for HR predictors,
- $<z,ns,ne,d>$ for DOW predictors,
- $<z,ns,ne,h,d>$ for DOW×HR predictors,
- $<z,ns,ne,w>$ for PEAK
Dynamic Zoning

- Dynamic zone sizes for each trip
- k-NN (kd-trees)
- Almost zero unsuccessful predictions.
- 5-tuple: \((S\text{Long}, S\text{Lat}, E\text{Long}, E\text{Lat}, S\text{Time} \times \text{factor})\)
- Factor: \((25000/10 \times 0.0001) = 0.25\)
Predictors (Dynamic Zones)

- LOC
- HR
- DOW
- DOW × HR
- PEAK
Algorithm: kNN Search for Trip Prediction

input:  
\( T \): Trip Data Set  
\( p \): Predictor Type (LOC/HR/DOW/DOW \times 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
Dynamic Zoning

- Slower
- High hit rates
- More memory consumption
Evaluation

- Set 1: training (20 months)
- Set 2: test (1 month; 12 million)
- 20 subsets from Set 1

- Prediction accuracy
- Speed
- Memory efficiency
- Hit Rate (%)
Static Zoning Results

- LOC
- DOW
- PEAK
- HR
- DOW x HR

Average Prediction Error (cents)

Zone Size (metres)
Static Zoning Results

Average Prediction Error (secs)

Zone Size (metres)

LOC, PEAK, DOW, HR, DOW x HR

2 to 4 mins
Static Zoning Results

The graph shows the Hit Rate (%) against Zone Size (metres) for different zoning techniques: LOC, PEAK, DOW, HR, and DOW x HR.

- LOC: The least effective for small zone sizes.
- PEAK: Moderate performance across all zone sizes.
- DOW: Shows a steady increase with some fluctuations.
- HR: Improved performance compared to LOC and DOW, especially at larger zone sizes.
- DOW x HR: The most effective, maintaining a high hit rate across all zone sizes.

The notation “Not good for small zone sizes” is particularly relevant for LOC, indicating it's not ideal for applications requiring small zone definitions.
Still not good for small zone sizes

The plotted lines are for the DOW×HR predictor.

Hit % goes up with more data
For 20 months of data, 50m zone size gave only 17% hit rate.

The plotted lines are for the DOW \times HR predictor.
Trading accuracy for hit rate, **PEAK** predictor with **200m zones** was found to be the best predictor with an average fare prediction error of **1.2$**, an average duration error of **2.8 minutes**, and a hit rate of **93%**.
Dynamic Zoning

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

**Figure 6: Fare Errors vs. No. of Neighbour Trips**
Dynamic Zoning

Only 6 months history set was used.

100% hit rate for dynamic zoning results.

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

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

Only 6 months history set was used.

![Graph showing fare errors vs. number of neighbour trips.](image)

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

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

Only 6 month history set was used.

1.05$ to 1.25$

PEAK classifier seems to be better

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

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

![Graph showing average prediction errors for different numbers of neighbours.](image)

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

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

$k = 25$ was chosen as optimum

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

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

Again 6 months of data & 100% hit rate.

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

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

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

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

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

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

Not too much difference (only 8 cents)

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
Dynamic Zoning Results:

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
Dynamic Zoning Results:

Similar results.

![Graph showing average prediction error across months with different k values. The graph 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]
Dynamic Zoning is almost always able to find enough neighbors to make good predictions.

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
Static vs Dynamic Zoning

- Hit rates
- Pre-chosen zone sizes: Static
- Implicit zones as k varied: Dynamic
- Prediction speed (micro/milli sec)
- Memory
- Hash-map & kd-trees
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**
Figure 12: History vs. Performance—Dynamic 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.
• Improving Accuracy:
  – Indirect routes (21%)
  – Filter 1 (2 times longer than the straight distance)
  – Filter 2 (\( \leq 20 \text{ km/h} \) or \( \geq 100 \text{ km/h} \))
  – Weather (rain)
Error may be reduced, up to 25 cents, by the irregularity filters.
Error may decrease, up to 30s, with the irregularity filters.
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
• Summary:
  – Dynamic zoning + 6 months data + filters
  – Data Cleaning necessary
  – Avoid seduction of complexity
  – Data partitioning for efficiency
  – Deployment takes time
  – Applicability to Other Domains