Detecting Regions of Disequilibrium in Taxi Services Under Uncertainty *

Yan Huang
University of North Texas
Computer Science and Engineering
huangyan@unt.edu

Jason W. Powell
University of North Texas
Computer Science and Engineering
jasonpowell@my.unt.edu

ABSTRACT

Thousands of taxis cruise a metropolitan road network looking for passengers that may be scattered or clustered in highly active locations. Taxicab drivers tend to gravitate to the known clusters, often leading to supply and demand disequilibrium as areas become under or over served. Many cities monitor their taxi fleet's locations using GPS devices and track passenger occupancy through trip meters, thereby producing data streams of taxicab trajectories and passenger activities. This paper presents the Service Disequilibrium Detection (SDD) framework which aims at identifying regions of service disequilibrium using this information. The SDD framework models request wait time and taxicab location uncertainty inherent in the discrete data streams and identifies the disequilibrium regions using two methods: (1) Bayesian spatial scan statistics, and (2) Poisson-based hypothesis testing. We claim the SDD framework can detect emerging disequilibrium and validate this claim using a large Shanghai taxi GPS data set.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithm, Design, Experimentation

Keywords

Spatio-temporal, Geo-streaming, Uncertainty modeling, Poisson, Bayesian, Taxi

1. INTRODUCTION

In large cities, hundreds of thousands of taxis cruise the road network looking for passengers. Some passenger requests may be scattered throughout a region while others may cluster in highly active locations. Taxicab drivers, in their search for a fare, tend to gravitate to known locations that produce many fares, such as an airport or a tourist location. This often leads to an imbalance in supply and demand as areas become under or over served and results in regional disequilibrium.

Many cities monitor their taxi fleet and know the exact location of a taxi every few seconds. Furthermore, once the driver picks up a passenger, the driver manually pushes a button or starts the charging meter, and a trip begins. This produces data streams of taxicab trajectories with starting times and location of request pickups. Please note that this system does not observe the non-served requests. Using this information, the problem is that: given (1) a set of taxis cruising on a road network and being monitored every few seconds and (2) customer pickups in real-time, the goal is to detect emerging regions of service disequilibrium.

The solution to this problem will be useful beyond taxi services. With the wide availability of low cost geo-locating devices, smart phones, and wireless networks, it is possible to monitor demands and services in real-time on road networks for many applications. Other example applications include (1) parking services where demands are parking requests and services are vacant parking spots, and (2) delivery where requests are item pick-up requests and dispatched deliverymen provide the services.

1.1 Challenges

There are several challenges in detecting service disequilibrium. First, uncertainty is inherent in this system, arising in several aspects. The data stream consists of discrete information—the time and location information occurs at discrete intervals. This implies that a system could find locations at discrete intervals; however, not all taxis report their information at every time interval. The sampling frequency and GPS error make the location information of the taxis uncertain at a given observation time. In addition, while the system knows the pickup time of the passenger, the actual time that the passenger makes the request is uncertain. Furthermore, the requests that are never satisfied are unknown even though these unsatisfied requests are very important in modeling disequilibrium.

The second challenge is to design a model to relate demand with service that allows the detection of over-served and under-served regions. A threshold-based counting ap-

*This work partially supported by the National Science Foundation Under Grant No. IIS-1017926

ACM SIGSPATIAL GIS '12 November 6-9, 2012. Redondo Beach, CA, USA

proach seems like a natural solution, but such an approach may be brittle due to the inherent uncertainty, e.g. how do you count a taxi if it's location is uncertain at that time. The third challenge is to deal with large number of requesters and servers in real-time, specifically when taxi service is a product of thousands of requests and servers.

1.2 Contributions

- This paper proposes a framework for detecting under-served and over-served regions. The framework counts idling taxis and requests in a probabilistic manner;
- We adopt a Bayesian spatial scan statistic method in measuring the disequilibrium under uncertainty;
- We then propose to use a Poisson-based hypothesis testing method to label disequilibrium regions. This simpler testing method can achieve similar or better results compared with the Bayesian spatial scan statistic method;
- We evaluate our framework on a large taxi GPS dataset from Shanghai containing 468,000 trips and 17,139 taxicabs. Both Bayesian and Poisson-based methods can detect emerging under-served regions within several minutes with the Poisson method tending to detect faster.

2. RELATED WORK

We can classify the related work into three categories: modeling location uncertainty, taxi service improvement, and spatial-temporal statistics.

Location uncertainty has been an active research area. Location uncertainty in Euclidean space is investigated in paper [8]. Similar to ours, the authors assume that for a trajectory segment represented by two sampling points, the two sampling points have no error and the location in between is uncertain. They derived their general principle of sampling error based on maximal travel speed, time, and sampling rate. The work assumes that objects move in a free space while we assume objects follow the most economical routes. Various papers have proposed indexing schemes for querying trajectories considering uncertainty [2, 14, 10, 1, 4]. For our problem, the main goal is to obtain statistical counts in a stream processing paradigm and indexing is mainly for querying and retrieving.

Improving the matching process of taxi services has attracted research attention in recent years. In [15], the authors proposed a recommendation system to better match taxi drivers with customers. The system uses two major sources of knowledge: 1) passenger mobility patterns and 2) taxi driver pick-up behaviors learned from the GPS trajectories of taxicabs. The idea is to identify, offline, the routes and destinations that historically produce more fares for taxis. In [9], the authors use historical data as experience and derive a Spatio-Temporal Profitability (STP) map to guide cruising taxicabs looking for customers. This method avoids systematic routing, which is impractical in most cases. These methods rely on patterns and do not detect emerging service disequilibrium based on current vacant taxis and requests.

The spatial scan statistic [5] scans for hotspots in spatial regions, producing a ratio statistic between the region being scanned and the outside region. It then compares this ratio with a distribution curve generated by Monte Carlo simulation that assumes a certain distribution of the dataset without hotspots, e.g. a Poisson distribution. A hotspot is determined using the $p - value$ of the ratio. A public domain software SatScan^{TM} is widely used for the detection and evaluation of hotspots in many application domains including infectious diseases, natural and human disasters, criminology, and transportation. In [3], the authors identify neighborhoods with unusually high/low levels of active transportation, e.g. walking and biking. CitySense [6] detects spatial-temporal hotspot such as arrivals or departures of taxicabs and busy nightlife activity regions. Recently, a Bayesian spatial scan statistics was proposed for detecting hotspots [7]. Using the counts of points of interests and baseline population, this method examines the posterior probability of every possible rectangular region under the null hypothesis of no hotspots and the alternative hypothesis of regions of higher density. None of these models considers location uncertainty.

3. THE SERVICE DISEQUIBLIRIUM DETECTION (SDD) FRAMEWORK

DEFINITION 1. *(Taxi Trajectory) A taxi trajectory Tr is a sequence of GPS points pertaining to the taxi's sampling location over time. Each point $Tr_i \in Tr$ consists of a tuple $< (x, y), t, o >$ with location (x, y), location reporting time t, and the taxi's occupancy status o.*

DEFINITION 2. *(Request Pickup) The pickup p of a request r is a tuple $< (x, y), t >$ where (x, y) is the location and t is the time the request is picked up.*

DEFINITION 3. *(Trajectory Segment) A trajectory Tr consists of a sequence of segments $< Tr_i, Tr_{i+1} >$. If a taxi is occupied at Tr_i, i.e. $Tr_i.o$ is "occupied", then segment $< Tr_i, Tr_{i+1} >$ is a live trajectory segment, otherwise it is a cruising trajectory segment.*

3.1 Overview of SDD

Fig. 1 is the Service Disequilibrium Detector (SDD) framework. It has three primary components: *Probabilistic Counter (PC)*, *Equilibrium Finder (EF)*, and *Disequilibrium Detector (DD)*. These components build and maintain statistical information of customer requests and taxis, updates this information at each time interval, and uses it to generate streaming regions of service disequilibrium. The framework begins by partitioning the space S into cells as seen in Fig. 2. At each time interval, requests may enter the system in one of these cells while cruising taxis frequently report their locations along a trajectory. Zooming in a cell, the figure displays the taxi trajectory as a series of sampling points. A taxi may not report its location at time t but it may have been able to serve the request at that time. The ultimate goal in a cruising taxi system is to serve all requests with the minimum number of taxis, thus the framework uses request and taxi counts to identify disequilibrium of these locations.

The problem with simply counting the requests and taxi sampling locations is the uncertainty in the request time and taxi locations. Taxis report at pickup time, not when

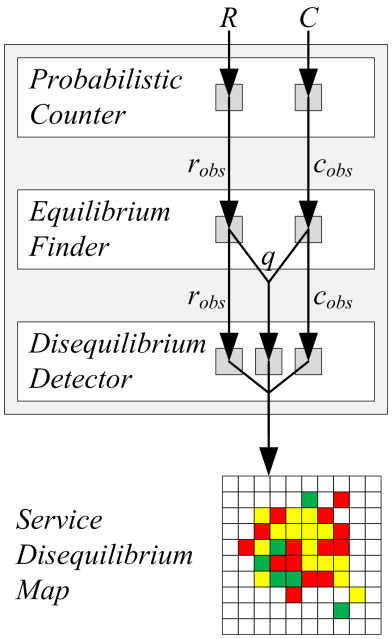


Figure 1: The Service Disequilibrium Detector (SDD) framework receives a set of Request Pickups R and Taxi Location Updates C at each time interval. This input becomes probabilistic observed counts that the framework uses to determine service disequilibrium.

the customer makes the request, and the taxi location between two reported GPS coordinates is unknown. For example, request r appears in the system at pickup time t even though it began at some unknown time $t - \delta_1$. Taxi c may have reported its location at time $t - \delta_2$ and $t + \delta_3$, leaving the actual location at time t unknown. This taxi may have been available to serve the request, along with m other taxis, meaning that the counts are uncertain for that time interval. The framework uses a probabilistic representation for the time and location uncertainty as information arrives from two streams: a set of customer pickups R and a set of taxi location reports C. The PC processes this data into probabilistic counts of observed requests r_{obs} and taxis c_{obs} and updates a cell's statistical information at each time interval to form a streaming history that the framework can use to detect emerging location service disequilibrium.

A major challenge in detecting disequilibrium is to define a notion of equilibrium. In economics, equilibrium is a state at which quantity demanded and supplied are equal [13]. In a cruising taxi system, the ratio between requests and cruising taxis in a equilibrium is decided by several factors. In a perfect knowledge system where all taxi and requests are known to each other and maximum matching [12] is performed by a central system, the ratio can be close to one. When knowledge is imperfect, we will need more cruising taxis to satisfy a given set of requests. The task of the Equilibrium Finder is to estimate the ratio in equilibrium. The idea is to use a request injection and a cruising taxi matching process to learn the parameters at equilibrium.

The Disequilibrium Detector uses the learned equilibrium state and observed counts to perform hypothesis testing to

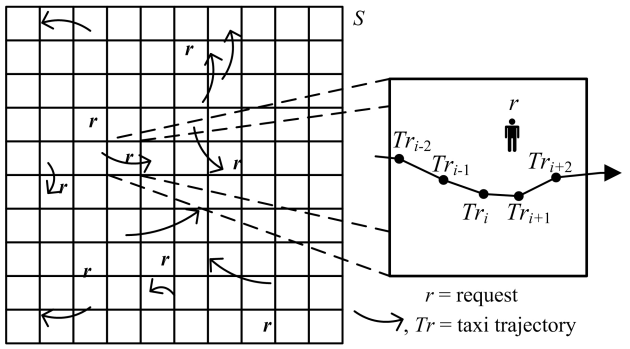


Figure 2: The Service Disequilibrium Detector (SDD) framework partitions the region into cells with request pickups and taxis. Zooming in on one of the cells displays the taxi trajectory as a series of sampling points.

determine if a region is over-served, under-served, or well served at each time interval. We propose two methods. The first one uses a Bayesian spatial scan method on this uncertain data. The second one uses a Poisson hypothesis testing to identify regions of disequilibrium.

3.2 Probablistic Counter (PC)

The Probabilistic Counter counts the requests and taxis in each cell for determining disequilibrium. It begins by partitioning the region S into equal sized cells with count history windows for both requests and taxis. Next, it processes two data streams simultaneously. One is a time-dependent set of request pickups R with each request consisting of the pickup time and location. The second data stream is a time-dependent set of taxi location updates C that contain the taxis' current location and time. At time t, all satisfied requests with that time enter the stream; however, not all taxis report their location nor are their reports necessarily at regular intervals. This input becomes the observed requests r_{obs} and taxis c_{obs} later used to determine a service rate and disequilibrium.

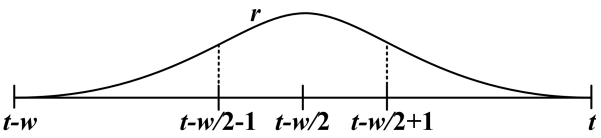
3.2.1 Counting Observed Requests

There is request arrival uncertainty in a cruising taxi system lacking perfect knowledge. When a taxi picks up a customer at time r_t, it is reasonable to assume that the request actually arrived at some previous time $t - \delta$, i.e. the uncertainty is the customer wait time w. This uncertainty carries some probability that the request occurred at a given interval of w, implying that the system can model it as a probability distribution. For the SSD framework, we assume a normal distribution and limit it to three standard deviations about the mean; hence, we define a request as:

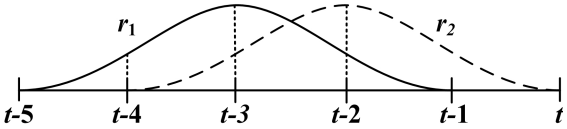
Definition 4. *(Request) A request r assoicated with a pickup p is approximated by a normal distribution $r =<\sim N(p.t - w/2, \sigma_w), p >$ with $p.t - w/2$ as the mean request time and σ_w as the standard deviation. Here $p.t$ refers to the pickup time of p and w is the average customer waiting time.*

Fig. 3 (a) shows the normal distribution curve representing uncertainty in request time. The probablistic request in a time interval $< t_1, t_2 >$ is counted by the cumulative

distribution function $cdf_N(t_2 : p.t - w/2, \sigma_w) - cdf_N(t_1 : p.t - w/2, \sigma_w)$. The total request count is the sum of all probablistic requests observed at that time.



(a) A normal distribution representing a request with pickup time t. The area under the curve of interval $(t - w/2, t - w/2 + 1)$ is the probabilistic request associated with the request count.



(b) Assuming $w = 4$, the request count at $t - 3$ and $t - 2$ is incomplete at time t when r_2 (dashed line) arrives even though the PC already counted r_1. The count at $t - 4$ is complete at time t because future pickups will not affect the count.

Figure 3: Requests model as a distribution.

Because a request is unknown until satisfied, the request count for current time may be incomplete until some future time. As seen in Fig. 3 (b), assuming w is four time units, the request count is incomplete for time $t - 1$ and $t - 2$ at time t because there may be new pickups at and after t. The PC updates a cell's request count history containing observed counts for any previous counts affected by the new request's distribution.

3.2.2 Counting Observed Taxis

The difference between a taxi's previous location and the new location forms a segment with known endpoints Tr_i and Tr_{i+1} but unknown taxi location between endpoints. The PC can count taxis at a location using a similar concept as counting requests by representing the location uncertainty with a probability distribution curve centered on a mean location. Fig. 4 shows this concept by representing each trajectory segment with a distribution curve. For the SSD framework, we assume a normal distribution and limit it to three standard deviations about the mean location of a segment.

Definition 5 (Taxi Location with Static Mean). *An unknown taxi location in a trajectory segment $< Tr_i, Tr_{i+1} >$ is approximated by normal distribution $N((Tr_i.p + Tr_{i+1}.p)/2, \sigma_i)$. Here $(Tr_i.p + Tr_{i+1}.p)/2$ is the middle point of the two ends of the segment.*

Because taxis are moving, the mean location of the normal distribution should move as well. For example in Fig. 5, at time $Tr_i.t$, it should have probability of one at $Tr_i.p$ and, as times approaches $Tr_{i+1}.t$, the probability of taxi being at $Tr_{i+1}.p$ is approaching one. However, with static mean distribution N, at $Tr_{i+1}.t - 1$, the distribution shows the probability as approximately 0.032, which is counter-intuitive because the taxi has almost reached the segment endpoint $Tr_{i+1}.p$ at that time. In other words, an accurate

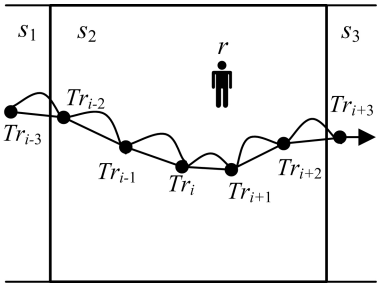


Figure 4: Taxi c forms several trip segments with some crossing cell boundaries. The exact location between each segment's starting and ending location is unknown; however, the PC can approximate it using normal distributions.

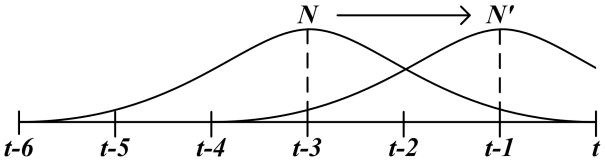


Figure 5: N is the distribution centered on mean time for the trip segment but N' better represents the location probability at time $t - 1$ because it is close to the endpoint.

taxi count for each time in between requires adjusting the distribution curve to a asymmetric distribution.

There are several adjustment mechanisms for normal distributions including skew, kurtosis, and mean adjustments; however, there is a trade-off between adjustment value and computational effort. Calculating skew and kurtosis is typically complicated, and along with variance, is contextually difficult to quantify. In addition, most trip segments are relatively short because GPS logging intervals are often short. Mean adjustment is simple and it can occur through adjusting the input parameter of the distribution's *cdf* function.

Definition 6 (Taxi Location with Variable Mean). *An unknown taxi location at time t where $Tr_i.t < t < Tr_{i+1}.t$ in a trajectory segment $< Tr_i, Tr_{i+1} >$ is approximated by normal distribution $N(\mu_t, \sigma_t)$ with μ_t as the mean location of the taxi with standard deviation σ_t. Here μ_t is the estimated location of the taxi at time t between Tr_i and Tr_{i+1} assuming constant speed in between.*

In the SDD framework, the location used for counting purposes is a cell so a probabilistic count of the taxi becomes the area under the distribution curve bounded by cell boundaries. As seen in Fig. 6, the area under the curve bounded by b_0 and b_1 represents the probabilistic count of the taxi for that cell location. A taxi segment may also cross several cells. We find all cells that intersect $< Tr_i, Tr_{i+1} >$. Then we order all the intersection by their x axis in the direction from $Tr_i.p.x$ to $Tr_{i+1}.p.x$ (if the intersections of all x axis values are the same, we order by y axis). We represent the intersection line as $< Tr_i.p, p_1, \ldots, p_m, Tr_{i+1}.p >$. For each p_i and p_{i+1}, we increase the counts of the cell that the segment $< p_i, p_{i+1} >$ is crossing by the accumulative probability.

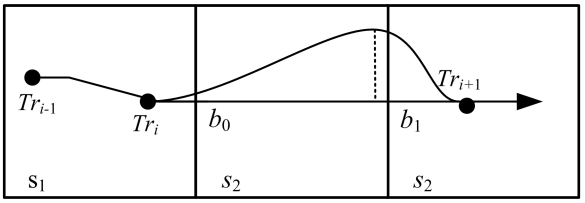


Figure 6: A taxi moves through multiple cells. At each time t, the distribution of location is adjusted and the accumulative count distribution for each intersecting cell is adjusted.

A taxi trajectory segment is an unknown distribution until the taxi reports Tr_{i+1}. Unlike requests, each taxi reports both endpoints so the distribution can represent the variance in segment lengths (Fig. 4). The taxi count may be incomplete until some future time and this can be handled by keeping a window of counts on the taxi streams for T_m time where T_m is the maximal interval between two sampling points.

3.3 Equiblirium Finder (EF)

The Disequilibrium Detector (discussed in Section 3.4) determines the service disequilibrium through Poisson hypothesis testing using the expected number of requests that a given number of taxis can serve. This ratio is called the equilibrium service rate q_{eq}. The service rate is similar to a disease infection rate, but in the taxi service context, the "infection rate" is the number of requests that a location's population of cruising taxis could serve. In a perfect system, the service rate should equal one, implying a perfect matching of requests to taxis. Realistically, this may never occur because of incomplete information and lack of coordination. The Equilibrium Finder's goal is to determine this equilibrium rate given the region's density of requests and taxis.

The service rate is the ratio of request count r to the cruising taxi count c; however, the rate at equilibrium is unknown in a system that only counts the known satisfied requests. The service rate should also include the un-served requests, or a reasonable estimate. The EF estimates this value by simulating the random injection of requests and attempting to match them with a cruising taxis within an arbitrary 100 meter radius until reaching a saturation point. We can define the rate of service rs for the requests using the total number of served requests and total requests as seen in Eq. 1 where r_{obs} is the number of requests served, r_{inj} is the number of requests injected, and $r_{inj,ser}$ is the number of injected requests that are served.

$$rs = \frac{r_{obs} + r_{inj,ser}}{r_{obs} + r_{inj}} \tag{1}$$

We wish to learn the ratio q_{eq} of served requests and cruising taxis as defined in equation 2 at equilibrium. We define the equilibrium as the saturation point where a certain percentage of the requests are satisfied.

$$q_{eq} = \frac{r_{obs} + r_{inj,ser}}{c_{obs}} \tag{2}$$

Algorithm 1 summarizes the injection process used to determine q_{eq}. Given an area A and the desired saturation threshold θ for the rate of service, the algorithm injects random requests by randomly selecting locations in the area based on the historical distribution of requests in the previous 30 minutes. It attempts to match the injected requests with cruising taxis (lines 4-12) while counting them appropriately. This continues while $rs > \theta$ (line 3).

Algorithm 1 INJECTION($area\ A, rs\ threshold\ \theta$)

Require: area A with r_{obs} and c_{obs}, and rs threshold θ
Ensure: The estimated q_{eq} for region S

```
1: r_inj := 0
2: r_inj,ser := 0
3: while rs_A ≤ θ do
4:     create random request r_j ∈ A
5:     r_inj := r_inj + 1
6:     if r_j matches a c_j ∈ A then
7:         c_j.occupied := true
8:         r_j.served := true
9:         r_inj,ser := r_inj,ser + 1
10:    else
11:        r_j.served := false
12:    end if
13:    update rs_A ≤ θ
14: end while
15: return q'_eq = (r_obs + r_inj,ser) / c_obs
```

Algorithm 2 is a supporting algorithm that calls Algorithm 1 after partitioning the entire region S into smaller areas. It performs this partitioning because the distribution of activity in the entire area is not necessarily reflective of localized areas. In addition, because there is a distinct pattern to taxi activity over time, the q_{eq} can vary with time of day. Therefore, for each area and each hour, this algorithm calls the injection algorithm to find an average q_{eq} that is implicitly weighted by each area's activity. The final result of the algorithm is a lookup table of hourly q_{eq} values that the Disequilibrium Detector calls upon. The framework can do this process offline whenever desired because it uses persistent historical data patterns.

Algorithm 2 CREATE_Q_TABLE($region\ S, rs\ threshold\ \theta$)

Require: region S and rs threshold θ
Ensure: The estimated q for region S per hour h

```
1: partition S into area set A with associated r_obs and c_obs
2: for hour h := 1 → 24 do
3:     for area a := 0 → |A| do
4:         q := 0
5:         for injection i := 1 → n do
6:             q := q + INJECTION(A[a], θ)
7:         end for
8:         q := q/n
9:         q_table_insert(A[a], h, q)
10:    end for
11: end for
12: return q table
```

The PC counts the requests and taxis per time unit but the framework is interested in establishing a service rate over a time interval to avoid spurious counts. The EF can establish the request rate by adding the number of requests observed during a subset of the history window. Likewise,

$$P(D|H_0) = \prod_{S_i \in G} P(c_i^r \sim Po(c_i^c \times q_{eq})) \quad (3)$$

$$P(D|H_1) = \prod_{S_i \in S} P(c_i^r \sim Po(c_i^r \times q_{eq})) \times \prod_{S_i \in G-S} P(c_i^c \sim Po(c_i^c \times q_{eq} \times \beta)) \quad (4)$$

it can establish the taxi rate, but with a caveat. The PC distributes a stationary request over a time interval, which means it counts a request only once. The PC distributes the taxis as well, but the adjustment at each time unit effectively forces the PC to count it multiple times. As a simple example, consider a static taxi in a cell. At each time unit, the probability the taxi is located in the cell is always one but counted 300 times in a five-minute window assuming one-second time granularity. The EF adjusts it by dividing the sum of taxis over the history window size to establish the taxi rate.

3.4 Disequiblirium Detector (DD)

We describe two methods of detecting regions of disequilibrium. The first one uses Bayesian scan statistics over the probabilistic counts and the second one uses Poisson testing. The Bayesian method detects hotspots and outliers. It is possible for this method to detect an under-served region even though the region only has slightly more requests than outside and is still over-served. The Poisson-based testing does not depend on a comparison between a region and outside and can detect an under-served region even when the whole space is under-served.

3.4.1 Baysian Scan Statistics

The space is partitioned into an $N \times N$ grid G. We want to find regions S in G where the ratio of the probabilistic count of requests over the count of cruising taxis is higher than the region outside S. For a given dataset D at a time snapshot, the null hypothesis H_0 is that the requests follow a Poisson distribution based on cruising taxis $R_{obs} \sim Po(q \times C_{obs})$ against the alternative set of hypothesis $H_1(S)$, each representing a higher number of requests in a region S.

Now we want to calculate the probability $P(H_0|D)$ and the probability of $P(H_1|D)$ as in Equation 3 and 4. Here for each cell S_i of G, we use c_i^r to represent the probabilistic request count and c_i^c to represent the count for probabilistic cruising taxis.

Using the Bayesian rule, we have $P(H_0|D) = \frac{P(D|H_0)P(H_1)}{P(D)}$ and $P(H_1|D) = \frac{P(D|H_1)P(H_1)}{P(D)}$ where $P(D) = P(D|H_0)P(H_0) + \sum_S P(D|H_1)P(H_1)$. We assume the prior probability of an under-served region is p. Then $P(H_0) = 1 - p$. We also assume the probability of the outbreak is equally distributed to any region. Thus, $P(H_1(S)) = \frac{p}{N_S}$ where N_S is the number of possible regions. For under-served detection, β is larger than 1 while β will be between 0 and 1 for over-served detection. We discuss setting β for under-serving but the over-serving case can be derived similarly. Since we do not know the exact value of β, we use a discrete uniform distribution for β, ranging from 1...3 at intervals of 0.2. We call $P(H_1|D)$ the F^* score of the dataset D and use F^* to detect service disequilibrium which will be discussed in details in section 4.2.2.

3.4.2 Region-based Poisson Testing

The Disequilibrium Detector determines a region's service disequilibrium through Poisson hypothesis testing. For each region $S \in G$, given the service rate and the count of observed taxis and requests, it labels a region according to Table 1. The detector labels a region as over served if the requests are more than expected given the number of cruising taxis in that region; however, this may also result from no reported requests. Similarly, it labels a region as under served if the requests are more than expected given the taxis in that region; however, this could also results from no taxis. There is an 'unknown' category representing region with no known requests and taxis, which may indicate an un-servable region or that there is no data for that time. Well-served regions are those with observed requests that do not significantly deviate from the expected requests.

Category	Level	Requests	Taxis
over served	-2		x
over served	-1	x	x
well served	0	x	x
under served	+1	x	x
under served	+2	x	
unknown	NA		

Table 1: Service Balance Categories. The level depends whether there are requests or taxis in the region (x indicates that data exists for that attribute). Unknown represents regions that have no known requests or taxis.

We compare the null hypothesis that the observed request does not significantly deviate from the expected requests with the alternative hypothesis of deviation for a region. It uses the service rate at equilibrium to calculate the likelihood it deviates by first determining the expected number requests as $E_R = q_{eq} \times C_{obs}$. E_R then becomes the Poisson distribution mean (Equation 5).

For a given region, multiplying the service rate by the observed taxi count produces the expected number of requests. The goal is to use this as the mean of a Poisson distribution to determine if the observed number of requests significantly deviates from the normal expectation. Depending on the results of the two-tail hypothesis test (Equation 6), the framework can label it according to the table. Rejection of the null hypothesis means the counts determine the level, otherwise, the null hypothesis is accepted and it labels the location as well served. This occurs for each time unit leading to a time-series of maps.

$$R_{obs} \sim P_0(\lambda = E_R = q_{eq} \times C_{obs}) \quad (5)$$

$$H_0 : \lambda = E_R \; Assume \; H_0 \; is \; true \quad (6)$$

$$H_1 : \lambda \neq E_R \; Reject \; H_0 \; if :$$
$$P(R_{obs} \leq E_R) \leq \alpha/2 \quad or$$
$$P(R_{obs} \geq E_R) \geq 1 - \alpha/2$$

4. EVALUATION

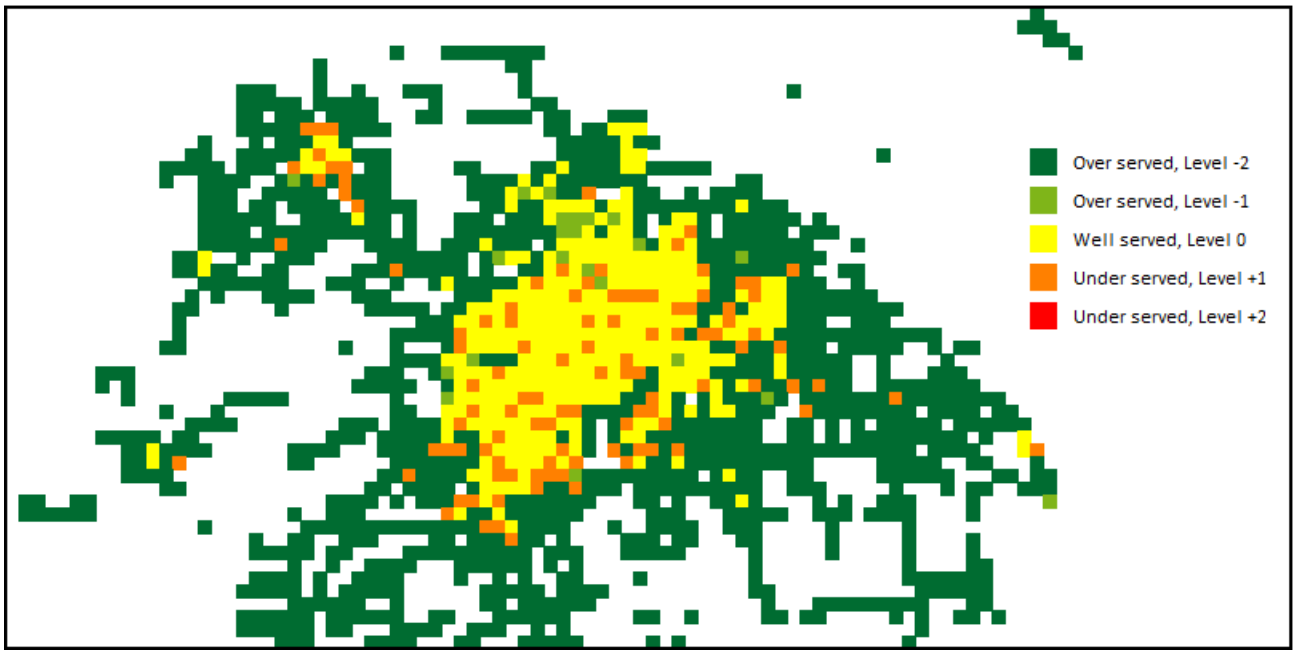


Figure 7: An example result of the Region-based Poisson Test on the Shanghai metropolitan region at 8 am. The region is divided in to $1km$ by $1km$ cells with the shading indicating the disequilibrium.

4.1 Data

Shanghai is a metropolitan area in eastern China with over 23 million denizens and a taxicab service industry with approximately 45 thousand taxis operated by over 150 companies [11]. To demonstrate our framework, we use a collection of GPS traces for May 29, 2009. The data set contains over 48.1 million GPS records (WGS84 geodetic system) for three companies between the hours of 12am and 5:30pm and over 468,000 predefined live trips of 17,139 taxicabs.

The region was limited to $31.0 - 31.5°N$, $121.0 - 122.0°E$ to remove extreme outliers and limit trips to the greater metropolitan area. Furthermore, only trips greater than five minutes are included since erratic behavior occurred more often in those below that threshold. Similar erratic behavior occurred with trips above three hours, often the result of the taxi going out of service, parking, and showing minute but noticeable movement from GPS satellite drift. The three-hour threshold is partially arbitrary and partially based on the distribution of trips times. While relatively rare, there are times when taxis spend over an hour on a cruising trip, but cruising trips over three hours occur much less frequently. These reductions left over 306 thousand live trips from which we constructed a stream of requests using live trip start times and locations. This resulted in an average of approximately 5 pickups per second, distributed as shown in Fig. 8.

For each taxicab, we defined a cruisin trips as the time and distance between the ending of one live trip and the beginning another. Using the cruising trip endpoints, we form the cruising trip segments from the 48 million individual GPS records. This resulted in over 21.4 million cruising trip segments that become the stream of taxicab location reports. The average trip segment is approximately 24 seconds and 100 meters. Fig. 9 is the distribution of cruising trips segments using the segment start time and Fig. 10 shows how

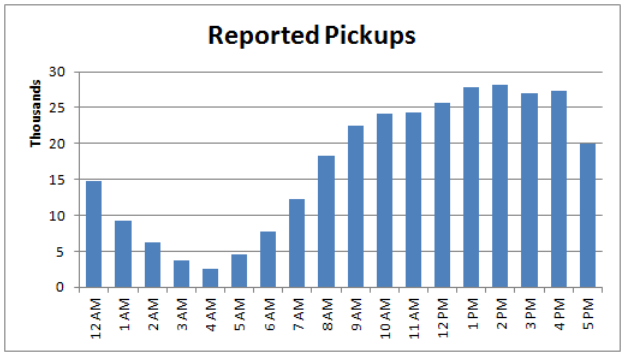


Figure 8: Distribution of requests per hour using the live trip start time.

the segment length varies throughout the day. As shown, the average length of two consecutive sampling points is smaller between 3 and 5am, indicating cruising taxis are less mobile in that time period.

4.2 Framework Evaluation

To validate our framework, we compare the results of the Bayesian Scan Statistic and the Region-based Poisson Testing method by examining both methods over synthetically created outbreaks designed to make an area under-served. The outbreaks result from injecting requests into specific areas using the same method as the Equilibrium Finder. The injection process matches each request with a taxicab within 100 meters, which it then removes from the dataset as it adds the injected request data as a pickup. The goal is to determine if both methods detect the outbreak and the quickness of detection.

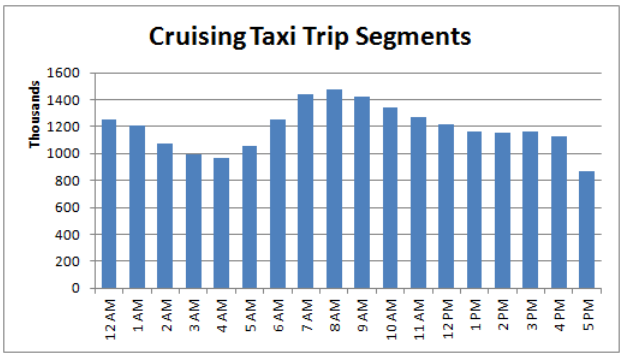


Figure 9: Distribution of cruising trip segments per hour based on segment starting time.

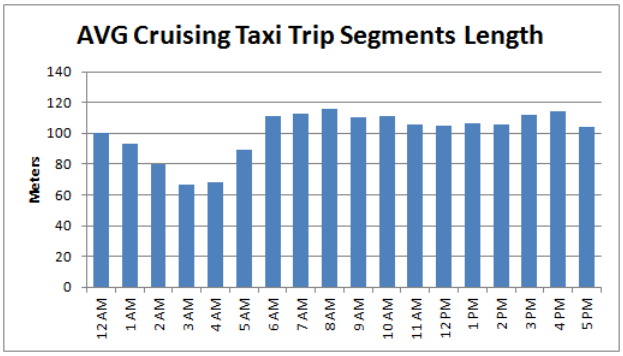


Figure 10: Distribution of cruising trip segments lengths per hour based on segment starting time. While the average length is 100 meters, there is variation depending on time of day.

4.2.1 Parameters

We pseudo-randomly selected five areas over the Shanghai region with interesting characteristics based on preliminary data examination of $1km$ x $1km$ cells. One selected area is relatively highly active yet routinely over served. Three areas are more moderately active and contain a mixture of over-served, under-served, and well-served cells. The fourth area often has a split between over-served locations with a high taxi-to-request ratio and under-served locations with a high request-to-taxis ratio. The last area is a relatively active location apart from the Shanghai downtown area. Figs. 11 and 12 is an example of the request and taxicab activity in the five selected areas at 8 am.

The three independent variables include the injection intensity (the number of injected and satisfied requests), the length of the outbreak in minutes, and the size of the outbreak area. The injection intensity ranged from 50 to 200 requests, in intervals of 50 requests over four experiments, with injection based on a normal distribution using mean outbreak time. This created a period of intensity, followed by a peak, and that a decrease in intensity. The outbreak lengths ranged from 10 to 30 minutes in 10 minute intervals and the outbreak area was $9km^2$. A second set of experiments using a $16km^2$. The control variable $P(H_0)$ is assumed as 0.90 For the Bayesian Scan Statistic method and all experiments used the 7 AM to 12 PM period data with the outbreak injected requests centered during the 8am hour.

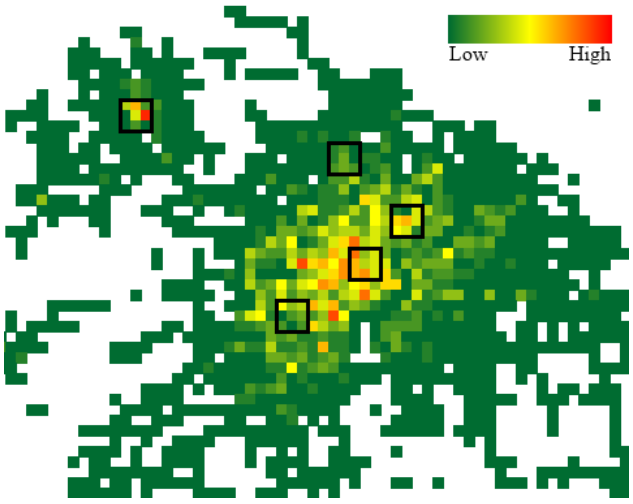


Figure 11: An example of the 8 am request activity for the five selected areas. Three areas focus on downtown Shanghai while one area is located some distance away. The selected area just north of downtown is a region that preliminary experiments revealed as routinely over served.

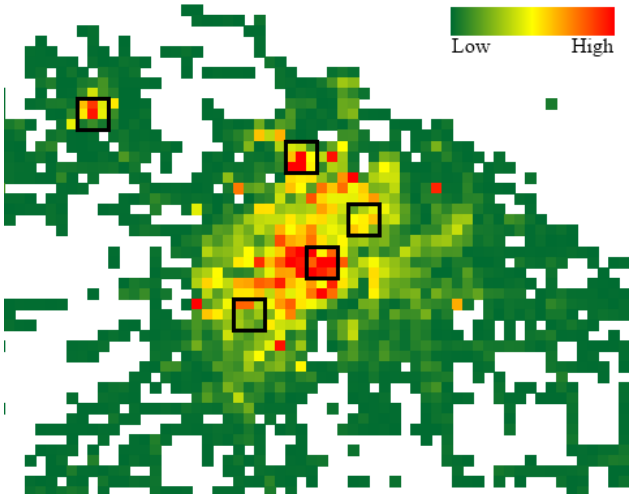


Figure 12: The 8 am taxi activity for the same five areas as seen in Fig. 11.

4.2.2 Outbreak Detection

The Bayesian Scan Statistic outbreak detection uses a similar process to [7]. We first generate the score F^* (see Section 3.4.1) for each minute between 7 AM and 12 PM before any outbreak is injected. We then partition these scores into consecutive 10-minute interval sets, thus each set contains ten F^* scores which serve as a baseline. After the outbreak is injected, the process calculates the new score $F^*(t)$ for each minute and compares it to each of the ten original F^* scores for that 10-minute interval. If $F^*(t)$ is greater than all original F^* scores within the interval, the process marks the time as the start of an outbreak. This process ensures that the false positive is bounded by one every 10 minutes. We can also determine false positives, false negatives, and how quickly it detected the outbreak because we know when the injected outbreak begins. We focus on detecting the outbreak and the detection time.

Detecting outbreaks for the Region-based Poisson Test uses a simpler method. Since the outbreak is affecting both the request count and taxi count, the result of the Poisson test can change at each minute; therefore, the first reported change in an area to under-served after the start of an outbreak signals outbreak detection.

4.2.3 Results

Figure 13 shows the results of performing the detection under the various parameter values with area sized fixed at $9km^2$. Both methods detect the outbreak within a few minutes; however, the Region-based Poisson Test often identifies the outbreak quicker than the Bayesian method. The anomaly in the 30-minute outbreak experiment appears to be a missing result; however, this is actually the result of a missed detection. The Region-based Poisson Test detected the outbreak during the first minute for four of the areas, but failed to detect the outbreak during for the fifth area. An examination of data revealed that for several minute before to the outbreak, the test identified the area as over served. Then during the outbreak, it became well served before going back to over served at the end. This results from a distribution of 50 injections over a 30-minute period as shown in Figure 14. The per-minute injections are simply not enough to change it from over served to under served in this case. Technically, the outbreak was detected therefore we label this as a pseudo-false negative.

We also performed experiments by expanding the injection areas to $16km^2$ as shown in Fig. 15. The inner box is the original area and the outer box is the expanded area. The results of these experiments were virtually identical as the other results except the Bayesian often identified area 4 (shown encircled in the figure) later than in the previous experiment. This appears to only be a factor of the newly included data of the expand areas.

5. CONCLUSION

Thousands of taxis cruise a metropolitan road network looking for passengers that may be scattered or clustered in highly active locations. Taxicab drivers tend to gravitate to the known clusters, often leading to supply and demand disequilibrium as areas become under or over served. Our proposed Service Disequilibrium Detection (SDD) framework identifies regions of service disequilibrium by modeling the uncertainty resulting from discrete customer requests and taxicab location information. This modeling allow the

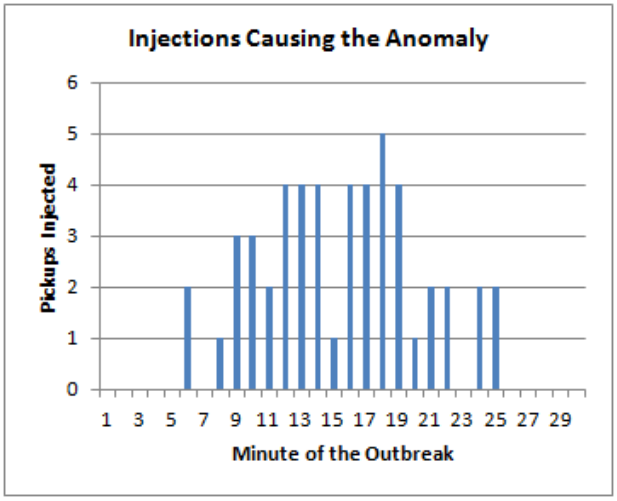


Figure 14: The injection distribution causing the pseudo-false negative anomaly. The injections were enough to change the area to well served from over served, but not enough to identify it as underserved.

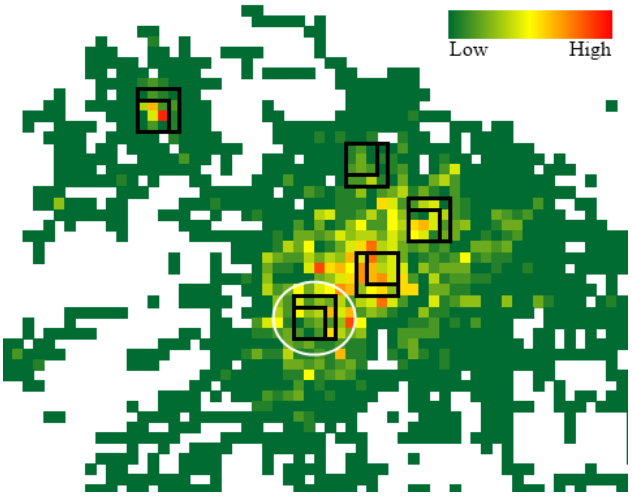


Figure 15: The expanded area used for testing $16km^2$ areas. The expansion attempts to include some of the more active surround cells.

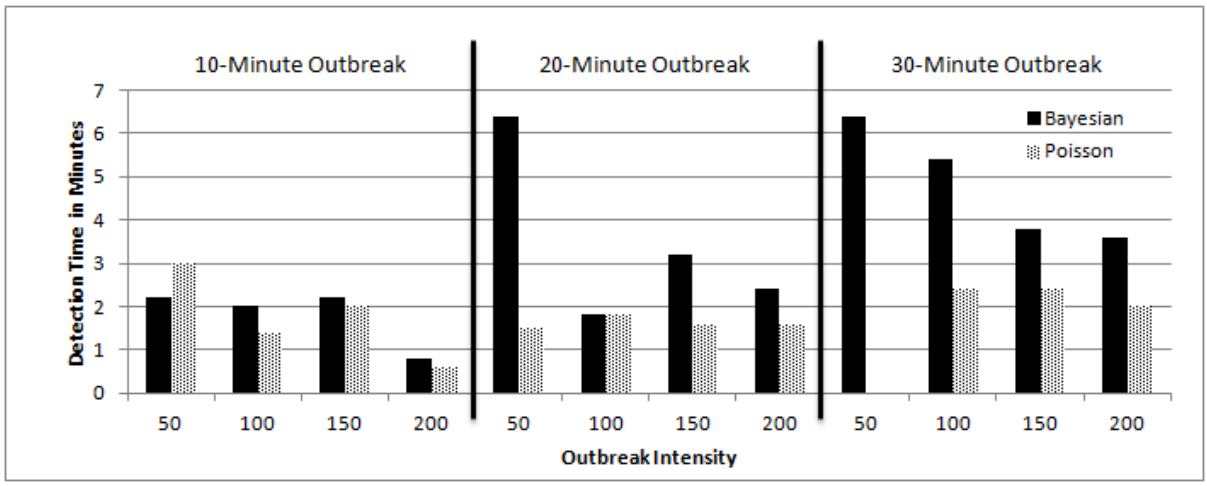


Figure 13: The results of four experiment sets with five fixed sized regions of $9km^2$ and varying outbreak intensity and length. Both methods detect the outbreaks, but overall, the Region-based Poisson Test detected the outbreak sooner. The apparent missing data is the result of a pseudo-false negative detection in one area while it detected the other four areas immediately.

probabilistic counting of taxis and customers such that the framework can use the counts to determine an equilibrium point from which two methods, the Bayesian Scan Statistic and our own Region-based Poisson Test, can quickly identify emerging regions of disequilibrium. We validate our framework by comparing methods on a set of real taxicab data from Shanghai, China.

There are several potential future directions for this framework. First, the grid-based approach could become a road network approach in which the framework analyzes each road segment instead of a cell. Second, the framework could provide a recommendation system that can redistribute taxis and customer to balance the disequilibrium as it emerges. In addition, the framework could also make recommendations as a share-ride utility by recommending passengers share taxis in under-served areas. The key challenge for these projects would be to ensure real-time performance.

6. REFERENCES

[1] Jinchuan Chen and Reynold Cheng. Efficient evaluation of imprecise location-dependent queries. In *ICDE*, pages 586–595. IEEE, 2007.

[2] Reynold Cheng, Dmitri V. Kalashnikov, and Sunil Prabhakar. Querying imprecise data in moving object environments. *IEEE Trans. on Knowl. and Data Eng.*, 16(9):1112–1127, September 2004.

[3] L Huang, DG Stinchcomb, LW Pickle, J Dill, and Berrigan D. Identifying clusters of active transportation using spatial scan statistics. 37:157–166.

[4] Dmitri V. Kalashnikov, Yiming Ma, Sharad Mehrotra, and Ram Hariharan. Index for fast retrieval of uncertain spatial point data. In *Proc. of Int'l Symposium on Advances in Geographic Information Systems (ACM GIS 2006)*, Arlington, VA, USA, November 10–11 2006.

[5] Martin Kulldorff. A spatial scan statistic. *Communications in Statistics - Theory and Methods*, 26(6):1481–1496, 1997.

[6] M. Loecher and T. Jebara. CitySense: Multiscale space time clustering of gps points and trajectories. In *Proceedings of the Joint Statistical Meeting*, 2009.

[7] Daniel B. Neill, Andrew W. Moore, and Gregory F. Cooper. A bayesian spatial scan statistic. In *NIPS*, 2005.

[8] Dieter Pfoser and Christian S. Jensen. Capturing the uncertainty of moving-object representations. In *Proceedings of the 6th International Symposium on Advances in Spatial Databases*, pages 111–132, 1999.

[9] Jason W. Powell, Yan Huang, Favyen Bastani, and Minhe Ji. Towards reducing taxicab cruising time using spatio-temporal profitability maps. In *Proceedings of the 12th international conference on Advances in spatial and temporal databases*, SSTD'11, pages 242–260, 2011.

[10] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst.*, 29(3), September 2004.

[11] TravelChinaGuide.com. Get around shanghai by taxi, shanghai transportation. Online; accessed 9-February-2011, 2011.

[12] Leong Hou U, Kyriakos Mouratidis, Man Lung Yiu, and Nikos Mamoulis. Optimal matching between spatial datasets under capacity constraints. *ACM Trans. Database Syst.*, 35(2):9:1–9:44, May 2010.

[13] Hal R. Varian. *Microeconomic Analysis*. Norton, 1992.

[14] Simonas Šaltenis, Christian S. Jensen, Scott T. Leutenegger, and Mario A. Lopez. Indexing the positions of continuously moving objects. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 331–342, 2000.

[15] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. Where to find my next passenger. In *Proc. of the 13th international conf. on Ubiquitous computing*, UbiComp '11, pages 109–118, 2011.