Doing in One Go: Delivery Time Inference Based on Couriers' Trajectories

Sijie Ruan[1,2,3], Zi Xiong[4,2,3], Cheng Long[5], Yiheng Chen[6], Jie Bao[2,3]*, Tianfu He[7,2,3]
Ruiyuan Li[1,2,3], Shengnan Wu[6], Zhongyuan Jiang[1], Yu Zheng[1,2,3]*
[1]Xidian University, Xi'an, China
[2]JD Intelligent Cities Research, Beijing, China [3]JD Intelligent Cities Business Unit, JD Digits, Beijing, China
[4]Wuhan University, Wuhan, China [5]Nanyang Technological University, Singapore
[6]JD Logistics, Beijing, China [7]Harbin Institute of Technology, Harbin, China
sjruan@stu.xidian.edu.cn;zixiong@whu.edu.cn;c.long@ntu.edu.sg;{chenyiheng,baojie,ruiyuan.li,wushengnan1}@jd.com
zyjiang@xidian.edu.cn;{Tianfu.D.He,msyuzheng}@outlook.com

ABSTRACT

The rapid development of e-commerce requires efficient and reliable logistics services. Nowadays, couriers are still the main solution to address the "last mile" problem in logistics. They are usually required to record the accurate delivery time of each parcel manually, which provides vital information for applications like delivery insurances, delivery performance evaluations, and customer available time discovery. Couriers' trajectories generated by their PDAs provide a chance to infer the delivery time automatically to ease the burdens on the couriers. However, directly using the nearest stay point to infer the delivery time is under satisfactory due to two challenges: 1) inaccurate delivery locations, and 2) various stay scenarios. To this end, we propose Delivery Time Inference (DTInf), to automatically infer the delivery time of waybills based on couriers' trajectories. Our solution is composed of three steps: 1) *Data Pre-processing*, which detects stay points from trajectories, and separates stay points and waybills by delivery trips, 2) *Delivery Location Correction*, which infers true delivery locations of waybills by mining historical deliveries, and 3) *Delivery Event-based Matching*, which selects the best-matched stay point for waybills in the same delivery location to infer the delivery time. Extensive experiments and case studies based on large scale real-world waybill and trajectory data from JD Logistics confirm the effectiveness of our approach. Finally, we introduce a system based on DTInf, which is deployed and used internally in JD Logistics.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**.

KEYWORDS

Trajectory Data Mining; Trajectory Annotation; Urban Computing

*Yu Zheng and Jie Bao are corresponding authors.

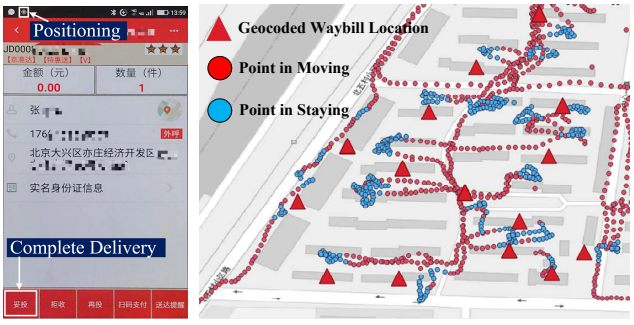
KDD '20, August 23–27, 2020, Virtual Event, CA, USA

ACM ISBN 978-1-4503-7998-4/20/08…$15.00
https://doi.org/10.1145/3394486.3403332

ACM Reference Format:
Sijie Ruan, Zi Xiong, Cheng Long, Yiheng Chen, Jie Bao, Tianfu He, Ruiyuan Li, Shengnan Wu, Zhongyuan Jiang, and Yu Zheng. 2020. Doing in One Go: Delivery Time Inference Based on Couriers' Trajectories. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394486.3403332

1 INTRODUCTION

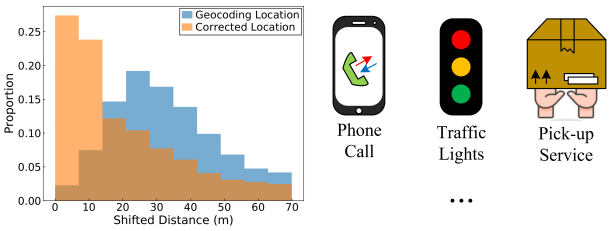
Express couriers are the main solution to address the "last mile" problem in logistics, currently. With the active development of e-commerce, the workloads of couriers become heavier. When delivering a parcel, the courier is asked to perform an important additional task besides the pick-ups and deliveries, i.e., recording the delivery time of each parcel. Figure 1(a) shows the interface of a courier's PDA, displaying detailed information and actions for a parcel delivery task, called *waybill*. The "Complete Delivery" button is required to be clicked immediately when the parcel is delivered. While the task of clicking this button each time a parcel is delivered looks tedious, it helps to reveal accurate delivery time that is vital for many applications in JD.com, e.g., delivery insurances, delivery performance evaluations, and customer available time discovery.



(a) Courier's PDA. (b) Courier's Trajectories & Waybills.

Figure 1: Background and Opportunities.

Therefore, it would be of great value for both the couriers and the logistics company if we can infer the delivery time. Fortunately, a courier's PDA also records his/her locations during the working hours as shown in Figure 1(a), which provides a chance to infer the delivery time automatically. Intuitively, a courier would stay



(a) Inaccurate Delivery Locations. (b) Various Stay Scenarios.

Figure 2: Challenges to Infer the Delivery Time.

at a location for a while when he/she is delivering a parcel, thus generating a stay point [13]. A straightforward solution would be to extract the delivery time based on the stay points of the trajectories.

However, in our preliminary data analysis, there are many exceptions between the stay points and delivery locations. For example, Figure 1(b) shows the point distribution of a trajectory in a region, where circles are GPS points of a courier's trajectory; and triangle markers are the Geocoded waybill locations, which are parsed from the plain text shipping addresses via Geocoding services [1]. According to the figure, there are many more trajectory stay points than the Geocoded waybill locations, and many Geocoded waybill locations are not always close to the stay points. Thus, it is not possible to infer the delivery time directly from stay points due to the following two main challenges:

- **Inaccurate delivery locations.** According to the delivery time of each waybill annotated by couriers, we can find its delivery caused stay point in the trajectory. We plot the distribution of the distance between the Geocoded waybill location and the centroid of such stay point in Figure 2(a). It shows that most of the Geocoded waybill locations have some distance shifts to the delivery caused stay points[2]. Therefore, for each waybill, we can not treat the Geocoded waybill location as the delivery location, and infer the delivery time based on the closest stay point to it.
- **Various stay scenarios.** Even if we find the closest stay point to the true delivery location, we still cannot say that the parcel is delivered at that stay point. The reason is that a courier might stay at a location for various reasons. As shown in Figure 2(b), a courier might stay when he is calling the customers, waiting for the traffic lights, or picking up parcels from customers.

To this end, we design, implement and deploy a delivery time inference system, Delivery Time Inference (DTInf), which can automatically infer the delivery time of each completed waybill based on couriers' trajectories. The proposed system contains three main components: 1) *Data Pre-processing*, which cleans trajectories from couriers, detects stay points, and separates stay points and waybills by delivery trips; 2) *Delivery Location Correction*, which corrects the Geocoded waybill locations based on their historical deliveries to them; and 3) *Delivery Event-based Matching*, which forms several delivery events by grouping waybills according to their delivery locations, and matches each delivery event with the most likely stay point in its neighborhood. Our contributions are four folds:

[1]https://en.wikipedia.org/wiki/Geocoding

[2]In our study region, there does not exist express cabinets, which are usually placed at the entrances of residential areas. Cabinets could make the shifts even larger.

- We present the first attempt to formalize the delivery time inference problem based on trajectories and identify its challenges.
- We propose a three-stage delivery time inference solution DTInf, which not only overcomes the distance shifts of delivery locations, but also takes various factors into the inference modeling.
- Experiments as well as case studies on real-world datasets from JD Logistics show the effectiveness of our proposed method. Results show that DTInf outperforms the best baseline by 31.8%.
- A system based on DTInf is deployed in JD Logistics and used internally.

2 OVERVIEW

2.1 Preliminaries

Definition 1 (Waybill). A waybill is a parcel delivery task assigned to a courier, denoted as a 4-tuple $w = (l_a, \mathbf{F}_p, t_r, t_d)$. l_a is the Geocoded waybill location of the shipping address, $\mathbf{F}_p$ are features of the parcel, e.g., the weight and the volume, t_r is the timestamp, at which a courier receives the parcel, and t_d is the delivery time.

We note that the shipping address in plain text is not available in this study due to privacy protection issues. The delivery time t_d normally needs to be manually recorded by couriers. In this study, we aim to infer t_d so as to reduce couriers' burden of recording it.

Definition 2 (Delivery Location). A delivery location is a spatial point, denoted as $l_d = (x, y)$, where a courier gives the parcel to the corresponding customer, or leaves it at an express cabinet.

Definition 3 (Trajectory). A trajectory is a sequence of spatio-temporal points, denoted as $tr =< p_1, p_2, ..., p_n >$, where each point $p = (x, y, t)$ indicates the physical presence at a location (x, y) (e.g., longitude and latitude) at time t. Points in a trajectory are organized chronologically.

Definition 4 (Stay Point). A stay point is a subsequence of the trajectory, which semantically means that a moving object stays in a geographic region for a while. Formally, given a distance threshold D_{max} and a time threshold T_{min}, $< p_i, p_{i+1}, ..., p_j >$ is called a stay point sp if $distance(p_i, p_k) \leq D_{max} (\forall k \in [i+1, j])$, $distance(p_i, p_{j+1}) > D_{max}$ (if $j < n$), and $|p_j.t - p_i.t| \geq T_{min}$. The time interval of a sp is $[p_i.t, p_j.t]$.

The location of a sp is estimated using its spatial centroid:

$$sp.x = \frac{\sum_{k=i}^{j} p_k.x}{j-i+1} \quad \text{and} \quad sp.y = \frac{\sum_{k=i}^{j} p_k.y}{j-i+1} \tag{1}$$

The time of a sp is defined as the middle point of its time interval:

$$sp.t = p_i.t + \frac{p_j.t - p_i.t}{2} \tag{2}$$

Particularly, if a stay point is caused by a delivery, we call it a *delivery caused stay point*. In historical data, it can be identified by checking whether there is a parcel delivered during the time interval of the stay point based on the delivery time of the waybill.

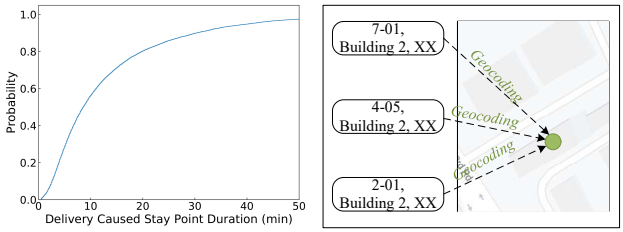
Definition 5 (Delivery Trip). A delivery trip is a process that a courier delivers a batch of parcels to customers.

2.2 Problem Definition

We aim to infer the delivery time for parcels in each delivery trip based on stay points in couriers' trajectories. We assume that a courier should report the parcels failed to be delivered (or successfully delivered) after the trip. Therefore, we know those waybills whose delivery time could be inferred.

We propose to identify for a waybill the stay point, at which the parcel of the waybill is delivered, and then use its time as the inferred time of the waybill. There are two reasons for this strategy:

- *Short delivery stay*: According to delivery caused stay points, the average delivery duration is 13 minutes, and for 80% waybills, the delivery duration does not last for longer than 20 minutes as shown in Figure 3(a). Such granularity is acceptable for target applications, e.g., customer available time discovery.
- *Anonymized shipping address*: One or more parcels can be delivered at the same stay point. However, Geocoding anonymizes the detailed floor information of shipping addresses as shown in Figure 3(b). Therefore, there is no guidance for deciding the orders by which the waybills are finished. Thus, it is impossible to infer finer-grained time.



(a) Delivery Stay Duration Distribution. (b) Anonymized Shipping Addresses.

Figure 3: The Motivations of Problem Formulation.

Therefore, the problem of inferring the delivery time of a waybill is transformed to be one of identifying the delivery caused stay point for the waybill. We define the problem as follows:

Given courier's stay points $SP = \{sp_j | j \in 1, .., m\}$ *detected from the trajectory of a delivery trip, and the waybills* $W = \{w_i | i \in 1, ..., p\}$ *he/she completed in the trip, the objective is to match each waybill* w_i *with its delivery caused stay point* sp_j.

2.3 System Framework

The system framework of DTInf is elaborated in Figure 4, consisting of three components:

Data Pre-processing. This component takes couriers' trajectories and waybills and performs three main tasks: 1) *Noise Filtering*, which removes the outlier GPS points; 2) *Stay Point Detection*, which detects all the stay points from the trajectories; 3) *Delivery Trip Identification*, which separates waybills and stay points by the identified delivery trips (detailed in Section 3).

Delivery Location Correction. This component takes historical waybills and stay points, and generates the location mapping from the Geocoded waybill location to the delivery location [3]. It includes three steps: 1) *Inverted Indexing*, which finds all historical delivery

[3]We assume a Geocoding waybill location corresponds to one delivery location. If the plain text shipping addresses are available, we can correct based on buildings.

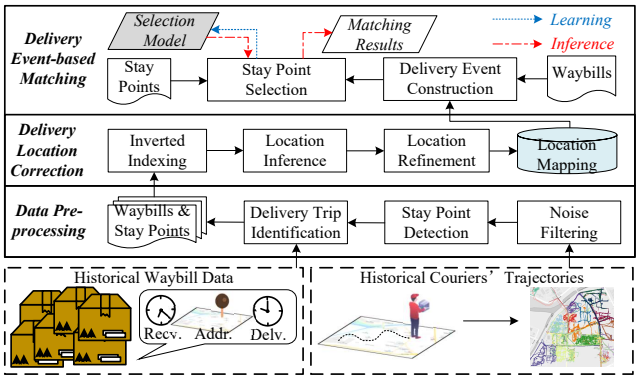


Figure 4: System Framework.

caused stay points for each Geocoded waybill location; 2) *Location Inference*, which infers the raw delivery location based on the inverted index; 3) *Location Refinement*, which refines the raw delivery location by merging it with its nearby delivery locations discovered by other Geocoded waybill locations (detailed in Section 4).

Delivery Event-based Matching. This component takes the stay points and waybills in a trip, and identifies the most likely delivery caused stay point for each waybill. Two steps are conducted: 1) *Delivery Event Construction*, which uses the location mapping to group waybills based on the corrected delivery location, where each grouped waybills are called a delivery event; 2) *Stay Point Selection*, which selects the best-matched stay point for each delivery event, and waybills in each delivery event share the same stay point matching result (detailed in Section 5).

3 DATA PRE-PROCESSING

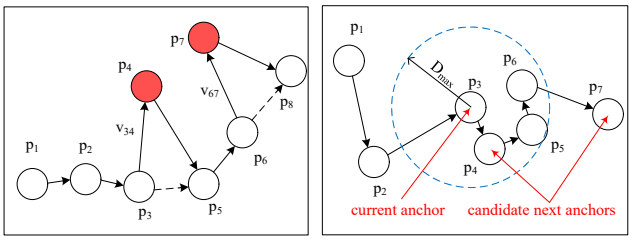
In this component, trajectories are cleaned and stay points are extracted. Then the stay points and waybills are separated and organized by identified delivery trips.

3.1 Noise Filtering

The trajectories generated by a courier's PDA usually contain noise points. For example, as shown in Figure 5(a), the error of p_4 and p_7 might be several hundred meters away from its true location. Such noise points would affect the quality of stay point detection. A heuristic-based approach proposed in [31] is used to filter noise points in trajectories. The algorithm sequentially calculates the traveling speed for each point in a trajectory based on its precursor and itself. If the speed is larger than a threshold, the current examined point is removed from the trajectory. In this example, if v_{34} and v_{67} are larger than the speed threshold, they are removed from the trajectory. The speed threshold is set to 54km/h since the moving speed of a courier would rarely exceed this threshold.

3.2 Stay Point Detection

Based on the cleaned trajectories, we extract all stay points from them. We use stay points not only to infer the delivery time, but also to find the real delivery locations. The stay point detection algorithm proposed in [13] is employed. The algorithm first checks if the distance between an anchor point and its successors in a trajectory is larger than a given threshold D_{max}. In the example



(a) Noise Filtering. (b) Stay Point Detection.

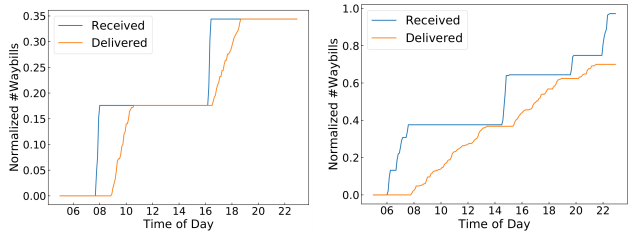
Figure 5: Trajectory Pre-processing.

shown in Figure 5(b), p_3 is the current anchor point, and p_4 to p_6 are its successors within D_{max}. It then calculates the duration between the anchor point and the last successor within D_{max} (p_3 and p_6). If the duration is larger than the given temporal threshold T_{min}, a stay point is detected (p_3 to p_6), and the anchor point moves to the next point after the current stay point (p_7). Otherwise, the anchor point moves forward by one (p_4). This process is repeated until the anchor point moves to the end of the sequence. The algorithm has the chance to generate stay points that are temporally consecutive, which makes little difference for the delivery time inference. Therefore, we also merge those consecutive stay points. We tried different parameter combinations and found that most delivery time of waybills can be included in stay points when we set $D_{max} = 20m$ and $T_{min} = 30s$.

3.3 Delivery Trip Identification

According to workloads, a courier can have one, two, or several delivery trips each day. For example, a normal workday usually contains 2 trips, while a promotion day (e.g., "6.18", Double 11) might contain 3~4 trips given the tremendous workloads. A courier will start a delivery trip after he/she receives the newly arrived parcels. Figure 6 shows the (normalized) number of parcels received and delivered by a courier during a day for a normal workday and a promotion day. It is noticeable that the number of sharp increases and the time of sharp increases of the parcel receiving curve are dynamic due to the upstream logistics arrangements, which further influences the start time of the delivery trip.

We propose to identify delivery trips of a courier based on the following two rules: 1) a trip begins when the number of receiving parcels stops increasing, and the number of delivering parcels begins to increase; and 2) a trip ends if the opposite condition holds.



(a) A Normal Workday. (b) A Promotion Day.

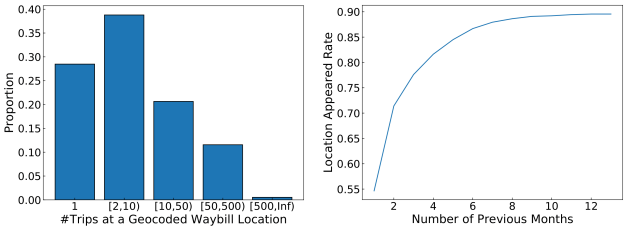
Figure 6: #Waybills Received&Delivered w.r.t. Time of Day.

Based on the identified delivery trips, we separate the waybills and stay points by the delivery trips that contain them.

4 DELIVERY LOCATION CORRECTION

In this component, we infer the delivery location for each Geocoded waybill location, so that we can model the delivery events more accurately based on the corrected locations in the later component.

Motivation. Given that the deviation from the Geocoded waybill location and the true delivery location distributes arbitrarily as shown in Figure 2(a), it is difficult to set a global consistent judgment about which stay point might be the delivery caused stay point of a certain waybill if we treat the Geocoded waybill location as the delivery location. Fortunately, because a customer might place orders multiple times using the same shipping address, a Geocoded waybill location might appear several times. Figure 7(a) shows the distribution of the number of delivery trips at a Geocoded waybill location during a period of 15 months. It is noticeable that for 72% Geocoded waybill locations, there exist multiple deliveries. Besides, those locations can also appear in the future. As shown in Figure 7(b), the Geocoded waybill locations of waybills in the previous 4 months cover more than 80% Geocoded waybill locations in the last month. Therefore, it is valuable if we can infer the delivery location for each Geocoded waybill location appeared in history to improve the time inference in the future.



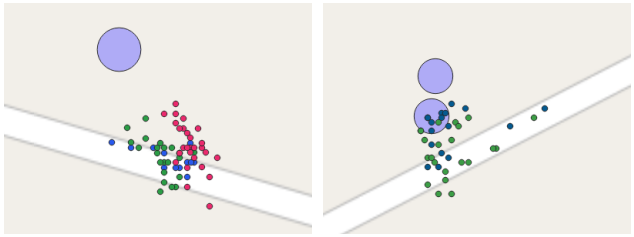
(a) #Trips at a Geocoded Location. (b) Geocoded Waybill Location Overlap.

Figure 7: Feasibility of Delivery Location Correction.

Main Idea. The delivery location correction mainly consists of three steps: 1) *Inverted Indexing*, which stores all historical delivery caused stay points for each Geocoded waybill location; 2) *Location Inference*, which infers the raw delivery location; 3) *Location Refinement*, that clusters the raw delivery locations which are spatially very close to generate the final delivery location. This is inspired by two insights discovered in the dataset:

- *Multiple delivery caused stay points*: Figure 8(a) shows 3 delivery caused stay points (points with the same color belong to one stay point) of a Geocoded waybill location (in the purple circle) in different trips. It is noticeable that although those stay points are quite close, there are still minor differences. If all stay points are leveraged, the delivery location correction can be more accurate.
- *Redundant Geocoded waybill locations*: Due to different Geocoding data sources, data source updates, and/or inaccurate plain text addresses input in history, redundant Geocoded waybill locations might be generated. Figure 8(b) shows two waybills with different Geocoded locations that are from different trips. It can be noticed that their delivery caused stay points have considerably large overlaps, which indicates they potentially correspond to the same delivery location. If we only infer the delivery location for each Geocoded waybill location individually, those delivery locations

have high possibility to be distinct even though they are spatially very close. As a result, we may have different delivery locations for those waybills even if they are completed during the time interval of the same stay point.



(a) Multiple historical stay points. (b) Redundant Geocoded locations.

Figure 8: Insights of Delivery Location Correction.

The pseudo code of the delivery location correction is presented in Algorithm 1, which takes historical trips TR and a distance threshold D, and returns the location mapping $\mathcal{R}$ from the Geocoded waybill location to the delivery location. We first build the inverted index $\mathcal{M}$ to store the delivery caused stay points for each Geocoded waybill location l_a by iterating over all historical trips. In each trip tr, we iterate over waybills delivered during the trip, i.e., $tr.W$, and find the delivery caused stay point for each waybill w by querying trip stay points $tr.SP$ with the delivery time $w.t_d$ (Line 2-5). Note that, we only store unique stay points for each l_a to avoid adding duplicated stay points given multiple waybills in one delivery location for each trip. Then, for each index key l_a in $\mathcal{M}$, we infer the raw delivery location using the centroid of its delivery caused stay points, and store the location mapping to the hashmap $\mathcal{A}$ (Line 6-7). Next, a hierarchical clustering algorithm [22], which is denoted as $cluster(\cdot,\cdot)$, is applied on all discovered raw delivery locations to form several clusters bounded by a given distance threshold D. The clustering algorithm first treats each raw location as a cluster, and then iteratively merges two clusters with the minimum distance to form a new cluster, until there does not exist two clusters whose distance is smaller than D. The clustering result $\mathcal{H}$ stores the mapping from the raw delivery location to its cluster centroid (Line 8). Eventually, the final delivery location for each l_a is the centroid of the cluster, which includes the raw delivery location. The final location mapping is stored in $\mathcal{R}$ and returned (Line 9-11).

Algorithm 1 Delivery Location Correction.

Input: The historical trips TR; the distance threshold D.
Output: The location correction mapping $\mathcal{R}$.

1: $\mathcal{M} \leftarrow \emptyset$; $\mathcal{A} \leftarrow \emptyset$; $\mathcal{R} \leftarrow \emptyset$;
2: **for** $tr \in TR$ **do**; ▷ Inverted Indexing
3: **for** $w \in tr.W$ **do**
4: $sp \leftarrow temporal\_query(tr.SP, w.t_d)$;
5: $\mathcal{M}[w.l_a] \leftarrow \mathcal{M}[w.l_a] \cup \{sp\}$;
6: **for** $l_a \in \mathcal{M}$ **do** ▷ Location Inference
7: $\mathcal{A}[l_a] \leftarrow centroid\_calculation(\mathcal{M}[l_a])$;
8: $\mathcal{H} \leftarrow cluster(\{\mathcal{A}[l_a] | \forall l_a \in \mathcal{A}\}, D)$; ▷ Location Refinement
9: **for** $l_a \in \mathcal{A}$ **do**
10: $\mathcal{R}[l_a] \leftarrow \mathcal{H}[\mathcal{A}[l_a]]$;
11: **return** $\mathcal{R}$;

5 DELIVERY EVENT-BASED MATCHING

In this component, we infer the delivery time for each waybill in a delivery trip by performing the delivery event-level stay point matching, which is based on the aforementioned location mapping mined from the historical data. We first introduce how to construct the delivery events, and then describe the matching strategy.

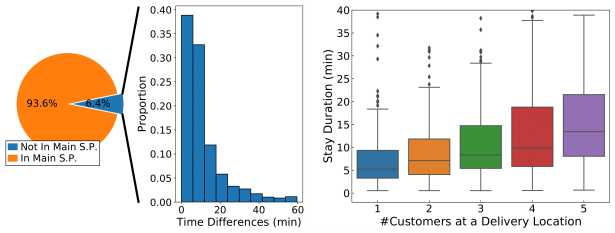
5.1 Delivery Event Construction

Instead of handling each waybill separately, we group waybills into several delivery events according to their corrected delivery locations. Then, in the later step, we can select the most probable stay point for each delivery event based on its delivery location, and assign that stay point to all waybills in that event.

Note that during the online inference phase, there may exist some Geocoded waybill locations that have never appeared in history. In such cases, we directly group waybills according to their Geocoded waybill locations to construct the delivery event.

The reasons to perform the delivery event-level matching for waybills in a trip are two folds:

- *Location by location delivery*: A courier usually continuously delivers several parcels at the same delivery location, e.g., a residential building. For a delivery location in each trip, we can find a stay point that is the most frequently matched by waybills at that location. We call such stay point as the *main stay point* of that delivery location in that trip. The pie chart in Figure 9(a) shows that 93.6% waybills match to the main stay points ($D = 3m$). Though there are still 6.4% waybills matching to other stay points, their delivery time differences with respect to the main stay points are small as the histogram shows. Therefore, if we perform the delivery event-level matching and correctly infer the main stay point for each delivery event, the time inference errors for waybills are acceptable.
- *Correlations between delivery events and stay points*: The parcels delivered at the same delivery location will affect the characteristics of the stay point. The box plot in Figure 9(b) shows the duration distribution with respect to different number of customers at a delivery location. It is obvious that a courier would stay longer if he/she needs to deliver for more customers. Such characteristics cannot be captured if we perform the inference task for each waybill individually.



(a) Waybill Matching Distribution. (b) Duration w.r.t #Customers.

Figure 9: Insights of Delivery Event Construction.

5.2 Stay Point Selection

After we construct several delivery events in a trip, we issue a spatial range query to find stay points within a certain radius for each delivery event, and then our task is to select a stay point, which corresponds to the main stay point of that delivery event.

Given the various stay scenarios as shown in Figure 2(b), we propose to use a model to capture the correlation between delivery events and the main delivery caused stay points, and ultimately improve the inference accuracy. We model the stay point selection as a binary classification problem. For a delivery event and one stay point in its neighborhood, we extract features of them and predict whether the stay point is the main delivery caused stay point of the delivery event. We use a MLP [8] as our classifier, whose output gives a probability between 0 and 1. The cross entropy loss is used during the training. In the inference phase, for each delivery event, we select the stay point in its neighborhood, which gives the highest probability. The following four types of features are extracted:

- *Location features*: We obtain the POI category of the Geocoded waybill location via the reverse Geocoding service, which is encoded by the one-hot vector.
- *Delivery event features*: We extract four aggregated information from the delivery events of waybills, namely, the number of waybills, number of customers, total weight, and total volume.
- *Stay point features*: We extract the duration and the area of the stay point.
- *Matching feature*: The geographical distance between the centroid of the stay point and the delivery location is extracted.

We also train a different model which does not employ the matching feature. During the online inference, such model is used to select stay points for delivery events whose delivery locations cannot be corrected, since the distance between the Geocoded waybill location and the delivery caused stay point can be arbitrary.

After the stay point is selected for each delivery event based on the stay point selection model, the time of stay point is used as the inferred delivery time for all waybills in the delivery event.

6 EXPERIMENTS

6.1 Experimental Settings

Datasets. The datasets contain trajectories and waybills of 5 couriers at a delivery station in Tongzhou District, Beijing over a period of about 15 months (from Apr. 12nd, 2018 to Jul. 7th, 2019).

- **Couriers' trajectories.** They are raw GPS logs generated by couriers' PDAs, where each record contains a courier ID, a location, and a timestamp. The average sampling time interval is 7.4 seconds. The datasets contain 5.93 million GPS points.
- **Waybills.** Each record contains a customer ID, a courier ID, parcel information (e.g., weight and volume), the time when the parcel is received, the time when the parcel is delivered, and a Geocoded waybill location. The datasets contain 274 thousand waybills. Besides, there are 16 POI categories we obtained via the reverse Geocoding service.

After the data pre-processing step, waybills and stay points detected from trajectories are organized by delivery trips. The trips without trajectories are dropped. There are 3,653 delivery trips in total. For each courier, we use his/her former 80% trips for training, the following 10% trips for validating, and the last 10% trips for testing. The delivery location correction is conducted based on the training and validation trips, which contain 2,506 unique Geocoded waybill locations. 87.1% Geocoded waybill locations in the test trips appear at least once in former trips.

Evaluation Metrics. We use the accuracy, which is defined as the proportion of waybills whose corresponding delivery caused stay points are correctly classified (i.e., their inferred delivery times are accurate). We also report the RMSE and the MAE based on the inferred delivery time and the time of delivery caused stay points.

Baselines. To the best of our knowledge, there is no existing solution that can exactly tackle our problem. Therefore, we design the following three baselines for comparison:

- Random Inference (RDInf): We randomly select a stay point from each waybill's neighborhood as its delivery caused stay point.
- Spatial Nearest Inference (SNInf): SNInf matches each waybill with its closest stay point in its neighborhood.
- Temporal Longest Inference (TLInf): TLInf selects the stay point in waybill's neighborhood with the longest duration.

Variants. We also compare DTInf with its three variants:

- DTInf-nC: This variant does not correct the delivery locations. The model is trained based on the Geocoded waybill locations.
- DTInf-nM: This variant corrects the locations, but it selects the stay point that is the closest to the corrected location.
- DTInf-nE: This variant also corrects the location, but it does not construct delivery events. Instead, it infers delivery caused stay point for each waybill based on the same model, but the delivery event features are replaced with individual waybill features.

Parameter Settings. The query radius R is set to $70m$ in order to cover all delivery caused stay points according to Figure 2(a). Our MLP for the stay point selection contains 3 layers, and the hidden layer contains 16 hidden units.

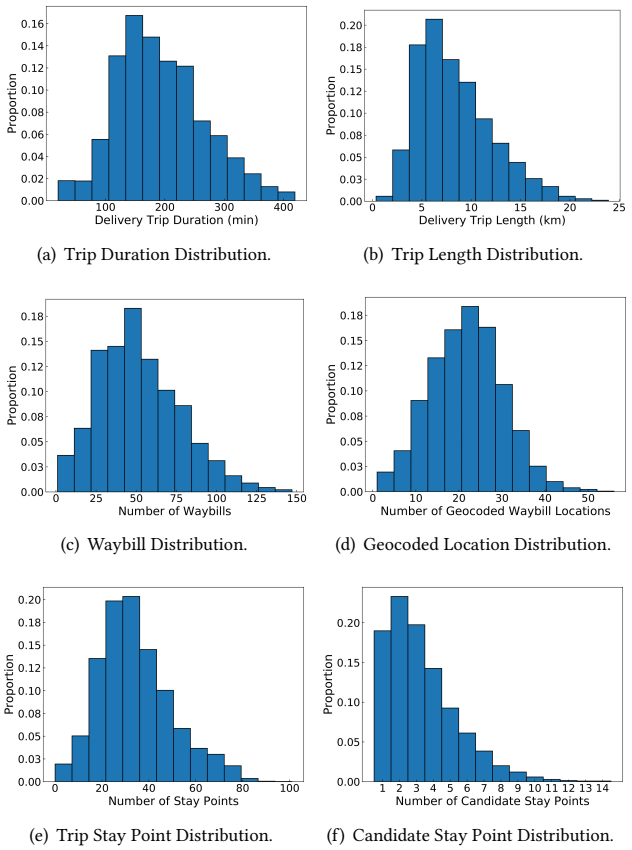
Implementations. Our algorithms are implemented in Python. Experiments are conducted on a workstation with an Intel(R) Core(TM) CPU i7-8700K @ 3.7GHz, 32GB memory, and Windows 10 OS.

6.2 Data Descriptions

Delivery Trip Distribution. Figure 10 gives the distribution of the trip duration and the trip length detected from the delivery trip identification step in Section 3.3. Figure 10(a) shows that the average duration of a delivery trip is about 3.2 hours, and the maximal duration does not exceed 7 hours, because a courier needs to go back to the station to receive newly arrived parcels after a certain time period. Figure 10(b) illustrates the average length is 8.3 km. Since a courier is assigned to deliver parcels for some regions that are spatially close, the length of the trip usually is not long.

Waybill Distribution. Figure 10(c) and 10(d) show the distribution of the number of waybills and unique Geocoded waybill locations in each delivery trip, respectively. As can be observed, a courier needs to deliver 52 waybills in each delivery trip on average. Since some waybills are in the same building, or belong to the same customer, there are 22 unique Geocoded locations to be delivered on average.

Stay Point Distribution. Figure 10(e) shows the distribution of the number of stay points in each delivery trip. The average number of stay points is 34, which is larger than the average number of Geocoded waybill locations. It validates our claim that a courier



(a) Trip Duration Distribution. (b) Trip Length Distribution.

(c) Waybill Distribution. (d) Geocoded Location Distribution.

(e) Trip Stay Point Distribution. (f) Candidate Stay Point Distribution.

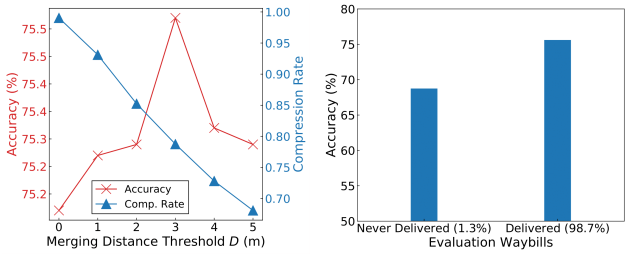
Figure 10: Dataset Descriptions.

stays at a location during a delivery trip not only because of the delivery, but also for some other purposes. We also plot the distribution of the number of stay points near each waybill in Figure 10(f). As shown in the figure, 81% waybills contain more than one stay point in their neighborhood ($R = 70m$), which implies that the stay point selection is necessary.

6.3 Effectiveness Evaluation

Merging Distance Selection. In order to model the delivery event accurately, we first select an appropriate location merging parameter D by varying it from 0m (no merging) to 5m. The stay point selection accuracy is reported in Figure 11(a), which shows the accuracy first increases and then drops. The reason is that when D becomes larger, redundant Geocoded waybill locations are corrected to the same delivery locations, which makes the delivery event modeling more accurate. However, when D is larger than 3m, the performance is degraded, because we might merge adjacent delivery locations by mistake. We also report the ratio between the number of detected delivery locations and Geocoded waybill locations (denoted as the compression rate) in the same figure. It shows that the compression rate is decreasing quickly, which also demonstrated the redundancy issue. Therefore, we set $D = 3$m. We also plot the distribution of the distance shifts of correction locations with respect to the delivery caused stay points in Figure 2(a).

Overall Evaluation. The overall performance of DTInf compared with baselines and variants is shown in Table 1. Among 3 baselines,



(a) Different Merging Distance. (b) Different Types of Waybills.

Figure 11: Effectiveness Experiments.

RDInf only achieves 41.0% accuracy. Comparing SNInf and TLInf, we can find that the matching based on duration is a better heuristic. DTInf outperforms the best baseline (TLInf) by 31.8% in terms of MAE, and is better than either of its variants. Comparing SNInf and DTInf-nM, we can see that after the locations are corrected, a 22.3% performance gain is witnessed using the spatial nearest heuristic. Comparing DTInf-nC and DTInf, we find that if the model is trained based on the biased locations, its effectiveness is degraded. Finally, the performance gaps between DTInf and DTInf-nM as well as DTInf-nE show the superiority of the delivery event modeling than using heuristics and modeling for each waybill individually. We also note that if express cabinets exist in the delivery region, the performance of non-correction-based methods (i.e., except for DTInf-nM, DTInf-nE, and DTInf) could be even worse since a much larger R needs to be set in order to cover the delivery caused stay point for each Geocoded waybill location, which brings more stay point candidates. While the correction-based methods are less affected, since our delivery locations are inferred based on couriers' annotation, and the distance to the delivery location is considered.

Table 1: Comparison with Baselines.

Methods	Accuracy (%)	RMSE (s)	MAE (s)
RDInf	41.0	2725.8	1254.5
SNInf	55.8	2361.5	868.9
TLInf	71.3	1713.4	588.1
DTInf-nM	62.6	2023.6	674.8
DTInf-nC	74.3	1518.5	470.8
DTInf-nE	74.8	1430.8	418.3
DTInf (ours)	**75.5**	**1365.6**	**401.0**

Different Types of Waybills. We are also interested in the performance differences of DTInf when faced with waybills whose Geocoded waybill locations have ever been seen or not. Figure 11(b) shows that for waybills whose locations have never appeared in history, we achieve 68.8% accuracy, while 75.6% accuracy is witnessed in the other case. It not only shows the effectiveness of the delivery location correction, but also tells that even a waybill whose location has never been delivered in history, we still have a high chance to have an acceptable inference. Another interesting point is that although there are only 87.1% Geocoded waybill locations appeared in history, those waybills only correspond to 1.3% in total test waybills, which indicates that the locations that frequently place orders have a high chance to appear in history, and the delivery location correction is applicable to the majority of waybills.

6.4 Case Study

We further give a case study of a delivery trip on the morning of Jun. 16th, 2019, which is one of the delivery trips in the evaluation dataset. Figure 12 shows the satellite image of a region in Tongzhou District. There is a waybill whose Geocoded location is displayed with the red triangle. The blue dots are the centroids of stay points detected from the courier's trajectory of the corresponding delivery trip, where the stay point with the longest duration in the neighborhood is sp_4. However, according to the ground-truths, the parcel is delivered during the time interval of sp_3, which leads to a great time inference error. Fortunately, this Geocoded waybill location has been delivered multiple times in history, so we are able to correct it. The corrected location is shown with the green triangle, which is much closer to the delivery caused stay point. Nevertheless, if we just employ the spatial nearest heuristic, sp_{18} would be inferred as the matched stay point, which also leads to large inference error. The DTInf successfully selects sp_3 among candidate stay points because it considers various factors.

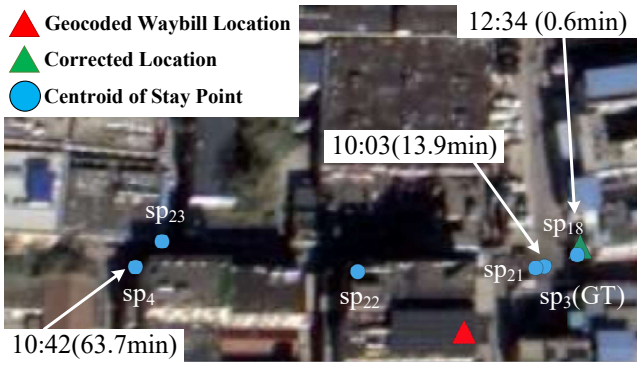


Figure 12: Case Study.

7 SYSTEM DEPLOYMENT

Our delivery time inference system is deployed internally in JD Logistics. In order to process massive couriers' trajectories, we leverage our self-developed platform, JD Urban Spatio-Temporal Data Engine (JUST) [14], to efficiently perform the noise filtering and the stay point detection in the distributed environment based on Apache Spark and HBase. The spatio-temporal index is also built over detected stay points based on JUST to accelerate the process of querying stay points near the waybills. When running online, our inference process is activated as soon as a courier ends his/her trips and commits parcels he fails to deliver. For waybills whose Geocoded waybill locations newly appeared, couriers would be asked to record the delivery time optionally in order to correct the delivery locations for better delivery time inference in the future.

The interface of our system is shown in Figure 13, which allows operators to visualize couriers' delivery trips and understand the inference process. The interface contains four components:

Operation View. In this view, the operator can perform several operations. There are four main buttons: 1) *Retrieve*, which is used to query waybills and trajectories of a trip that is specified; 2) *Correct*, which corrects the delivery location of waybills in the current trip; 3) *Query*, that issues spatio-temporal query to find stay points near each waybill; and 4) *Infer*, that infers the delivery caused stay point for each waybill.

Figure 13: System Interface.

Main Map View. The right part is the main map view. When the delivery trip is retrieved, courier's trajectories (grey line), stay point centroids (blue circle), and Geocoded waybill locations (red triangle) are visualized. After *Correct* is clicked, the corrected delivery locations are displayed with the green circles, and the big red circles indicate the querying neighborhood. Finally, when the operator clicks *Infer*, a link would be generated between the corrected location and the inferred delivery caused stay point.

Waybill Information View. This view displays detailed information about the retrieved waybills. The waybill ID, the customer ID, the weight and the volume are shown. If one of the waybills is selected, the location of the waybill would appear within the map view, and the inferred stay point will appear in the result view.

Result View. The view shows the detailed information of the inferred stay point for the selected waybill. It displays the start time, the end time, and the duration of the stay point.

8 RELATED WORK

Trajectory Annotation. In this work, we essentially want to annotate some stay points in trajectories with parcels delivered during the time interval of them, which is related to the trajectory annotation. The trajectory annotation aims to enrich trajectories with the semantic information [1]. The annotation techniques are mainly concerned with annotating trajectories with maps [2, 5, 17, 19, 23, 25] and recognizing the transportation modes [6, 33]. Annotating moving trajectories with roads is also known as the map matching [17, 25]. Annotating stay points with POIs is usually based on the geometric intersection or the spatial nearest neighbor [2, 5]. Many candidate POIs may exist in some densely populated urban areas, therefore, Yan et al. [23] attempted to infer the POI categories based on a Hidden Markov Model to maximize the visiting sequential probability. Keles et al. [12] predicted POI categories using a Bayesian Network, which considers the time of the day, the day of the week, and the duration of the stay point. Suzuki et al. [19] inferred the exact POIs a user visited under the integer linear programming framework, which also considers various features from POIs and stay points. For the POI assignment tasks, the stay duration differences are usually caused by different POI categories,

while in our problem, the number of customers at a delivery location plays the dominant role. Apart from that, for the trajectory annotation, a specific POI would be annotated for multiple times or not be assigned, both of which are not acceptable in our scenario.

Trajectory Data Mining. The trajectory data mining [31] studies discovering various knowledge from massive trajectory data. To enhance the existing maps, [9, 18, 21, 29] studied the road network generation or refinement based on crowd sourced trajectories, and [3, 30, 34, 35] studied discovering interesting places from trajectory hotspots. To help urban planning, [4, 28] gave the bike path lane planning or electric fence construction recommendation. To increase the commercial profits, [15, 27] aimed to select the best location for the billboard placement. To improve the user experience, [11, 34] studied the traveling recommendation. In this work, we discover the delivery locations based on trajectories and the recorded delivery time, which are further used to help the delivery caused stay point recognition.

Urban Computing. Urban computing [32] aims to solve the issues caused by human's rapid progress in urbanization, such as anomaly detection [7, 26], crime rate inference [20], air quality prediction [24], and resource rebalancing [10, 16]. In our work, we focus on easing the burden of couriers by automatically inferring the delivery time based on their trajectories.

9 CONCLUSION

In this paper, we propose DTInf, a system to infer the delivery time based on couriers' trajectories. Our method first separates waybills and stay points detected from trajectories by delivery trips, then corrects delivery locations of waybills, finally constructs delivery events for waybills based on corrected locations, and predicts the delivery caused stay point for each event, which is further used to infer the delivery time. Experiments show our method significantly outperforms baselines by at least 31.8%. And a case study is further conducted to illustrate the advantage of our solution. Finally, a system is deployed in JD Logistics.

ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China (2019YFB2101805), the NSFC Grant (61976168, 61672399, 61502375), the NSFC-Zhejiang Joint Fund (U1609217), and Beijing Academy of Artificial Intelligence (BAAI). This work was also supported by the Nanyang Technological University Start-UP Grant from the College of Engineering under Grant M4082302 and by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (RG20/19 (S)).

REFERENCES

[1] Basma H Albanna, Ibrahim F Moawad, Sherin M Moussa, and Mahmoud A Sakr. 2015. Semantic trajectories: a survey from modeling to application. In *IF&GIS*. Springer, 59–76.
[2] Luis Otavio Alvares, Vania Bogorny, Bart Kuijpers, Jose Antonio Fernandes de Macedo, Bart Moelans, and Alejandro Vaisman. 2007. A model for enriching trajectories with semantic geographical information. In *Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems*. 1–8.
[3] Daniel Ashbrook and Thad Starner. 2002. Learning significant locations and predicting user movement with GPS. In *Proceedings. Sixth International Symposium on Wearable Computers*. IEEE, 101–108.
[4] Jie Bao, Tianfu He, Sijie Ruan, Yanhua Li, and Yu Zheng. 2017. Planning Bike Lanes Based on Sharing-Bikes' Trajectories. In *KDD*. 1377–1386.
[5] Dong-Wan Choi, Jian Pei, and Thomas Heinis. 2017. Efficient mining of regional movement patterns in semantic trajectories. *VLDB* 10, 13 (2017), 2073–2084.
[6] Sina Dabiri and Kevin Heaslip. 2018. Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transportation research part C: emerging technologies* 86 (2018), 360–371.
[7] Takashi Fuse and Keita Kamiya. 2017. Statistical anomaly detection in human dynamics monitoring using a hierarchical dirichlet process hidden markov model. *TITS* 18, 11 (2017), 3083–3092.
[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT.
[9] Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. 2018. RoadRunner: improving the precision of road network inference from GPS trajectories. In *SIGSPATIAL*. 3–12.
[10] Shenggong Ji, Yu Zheng, Zhaoyuan Wang, and Tianrui Li. 2019. A Deep Reinforcement Learning-Enabled Dynamic Redeployment System for Mobile Ambulances. *IMWUT* 3, 1 (2019), 1–20.
[11] Shuhui Jiang, Xueming Qian, Tao Mei, and Yun Fu. 2016. Personalized travel sequence recommendation on multi-source big social media. *TBD* 2, 1 (2016), 43–56.
[12] Ilkcan Keles, Matthias Schubert, Peer Kröger, Simonas Šaltenis, and Christian S Jensen. 2017. Extracting visited points of interest from vehicle trajectories. In *Proceedings of the Fourth International ACM Workshop on Managing and Mining Enriched Geo-Spatial Data*. 1–6.
[13] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma. 2008. Mining user similarity based on location history. In *SIGSPATIAL*. ACM, 34.
[14] Ruiyuan Li, Huajun He, Rubin Wang, Yuchuan Huang, Junwen Liu, Sijie Ruan, Tianfu He, Jie Bao, and Yu Zheng. 2020. JUST: JD Urban Spatio-Temporal Data Engine. In *ICDE*. IEEE.
[15] Yuhong Li, Jie Bao, Yanhua Li, Yingcai Wu, Zhiguo Gong, and Yu Zheng. 2016. Mining the most influential k-location set from massive trajectories. In *SIGSPATIAL*. 1–4.
[16] Junming Liu, Leilei Sun, Weiwei Chen, and Hui Xiong. 2016. Rebalancing bike sharing systems: A multi-source data smart optimization. In *SIGKDD*. 1005–1014.
[17] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL*. 336–343.
[18] Sijie Ruan, Cheng Long, Jie Bao, Chunyang Li, Zisheng Yu, Ruiyuan Li, Yuxuan Liang, Tianfu He, and Yu Zheng. 2020. Learning to generate maps from trajectories. AAAI.
[19] Jun Suzuki, Yoshihiko Suhara, Hiroyuki Toda, and Kyosuke Nishida. 2019. Personalized visited-poi assignment to individual raw GPS trajectories. *TSAS* 5, 3 (2019), 1–28.
[20] Hongjian Wang, Daniel Kifer, Corina Graif, and Zhenhui Li. 2016. Crime rate inference with big data. In *KDD*. 635–644.
[21] Suyi Wang, Yusu Wang, and Yanjie Li. 2015. Efficient map reconstruction and augmentation via topological methods. In *SIGSPATIAL*. ACM, 25.
[22] Joe H Ward Jr. 1963. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association* 58, 301 (1963), 236–244.
[23] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. 2013. Semantic trajectories: Mobility data computation and annotation. *TIST* 4, 3 (2013), 1–38.
[24] Xiuwen Yi, Junbo Zhang, Zhaoyuan Wang, Tianrui Li, and Yu Zheng. 2018. Deep distributed fusion network for air quality prediction. In *KDD*. 965–973.
[25] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guang-Zhong Sun. 2010. An interactive-voting based map matching algorithm. In *MDM*. IEEE, 43–52.
[26] Huichu Zhang, Yu Zheng, and Yong Yu. 2018. Detecting urban anomalies using multiple spatio-temporal data sources. *IMWUT* 2, 1 (2018), 1–18.
[27] Ping Zhang, Zhifeng Bao, Yuchen Li, Guoliang Li, Yipeng Zhang, and Zhiyong Peng. 2018. Trajectory-driven influential billboard placement. In *KDD*. 2748–2757.
[28] Yongping Zhang, Diao Lin, and Zhifu Mi. 2019. Electric fence planning for dockless bike-sharing services. *Journal of cleaner production* 206 (2019), 383–393.
[29] Lisheng Zhao, Jiali Mao, Min Pu, Guoping Liu, Cheqing Jin, Weining Qian, Aoying Zhou, Xiang Wen, Runbo Hu, and Hua Chai. 2020. Automatic Calibration of Road Intersection Topology using Trajectories. (2020).
[30] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative location and activity recommendations with GPS history data. In *WWW*. 1029–1038.
[31] Yu Zheng. 2015. Trajectory data mining: an overview. *TIST* 6, 3 (2015), 29.
[32] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *TIST* 5, 3 (2014), 38.
[33] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. 2008. Learning transportation mode from raw gps data for geographic applications on the web. In *WWW*. 247–256.
[34] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. 2009. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*. 791–800.
[35] Changqing Zhou, Dan Frankowski, Pamela Ludford, Shashi Shekhar, and Loren Terveen. 2004. Discovering personal gazetteers: an interactive clustering approach. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*. 266–273.