Forecasting Gathering Events through Trajectory Destination Prediction: A Dynamic Hybrid Model

Amin Vahedian Khezerlou, Xun Zhou, Ling Tong, Yanhua Li, *Senior Member, IEEE*, and Jun Luo

Abstract—Identifying urban gathering events is an important problem due to challenges it brings to urban management. In our prior work, we proposed a hybrid model (H-VIGO-GIS) to predict future gathering events through trajectory destination prediction. Our approach consisted of two models: historical and recent and continuously predicted future gathering events. However, H-VIGO-GIS has limitations. (1) The recent model does not capture the newly-emerged abnormal patterns effectively, since it uses **all** recent trajectories, including normal ones. (2) The recent model is sparse due to limited number of trajectories it learns, i.e., it cannot produce predictions in many cases, forcing us to rely only on the historical model. (3) The accuracy of both recent and historical models varies by space and time. Therefore, combining them the same way at all times and places undermines the overall accuracy of the hybrid model. Addressing these issues, in this paper we propose a Dynamic Hybrid model called (DH-VIGO-TKDE) that addresses the above-mentioned issues. We perform comprehensive evaluations using two large real-world datasets and an event simulator. The experiments show the proposed model significantly improves the prediction accuracy and timeliness of forecasting gathering events, resulting in average precision of 0.91 and recall of 0.67 as opposed to 0.74 and 0.50 of H-VIGO-GIS.

Index Terms—Gathering events, destination prediction, trajectory mining, data mining

1 Introduction

Gathering events are the scenarios where an unexpectedly-large number of moving objects (pedestrians, vehicle, etc.) arrive at the same region during a short period of time. Gathering events in urban areas pose serious challenges for city management as more-than-ordinary resources will be required and public safety concerns will be raised. Example consequences may include traffic jams and high risk of injury, crimes, and terror-attacks. Shanghai's 2014 New Year's Eve stampede is a tragic example [1]. If given timely warning of future gathering events, city officials will have the opportunity to react to these situations in a timely manner, e.g., re-routing usual traffic, adopting necessary provisions, etc.

State-of-the-art techniques on urban event detection [2], [3], [4], [5], [6], [7] are mostly descriptive, i.e., the region and time of events are detected based on available on-site observations such as taxi drop-offs or traffic volume around the venue. These methods lack the ability to forecast future events before the gathering becomes significant.

In our prior work [8], we investigated a gathering event forecasting approach through trajectory destination prediction. The approach worked in two steps. First, a hybrid spatio-temporal destination prediction model was learned from historical and recent trajectories of moving objects (e.g., taxis). Second, we used the model to continuously predict the destination and arrival time distribution of incomplete trajectories, and identify future spatio-temporal regions with high predicted arrival counts as gathering events. We will refer to this method as H-VIGO-GIS (Hybrid VIGO) throughout this paper.

However, H-VIGO-GIS has three important limitations. First, the recent model was incorporated to learn the patterns of "non-typical" trajectories, as opposed to the historical model that learns the "typical" pattern of the trajectories. However, recent model in H-VIGO-GIS learned all the recent trajectories, regardless of them being normal or abnormal. This led to a recent model that captured some non-typical patterns but mostly typical ones. As a result, the hybrid model had little ability in predicting newly-emerged non-typical trajectory patterns. Second, the hybrid model combines historical and recent models' predictions with a fixed weight. Depending on time and location, either of the models could produce superior predictions. Therefore, the weight needs to be adjusted to give more importance to the more accurate model (i.e., historical or recent). Third, the recent model used in H-VIGO-GIS is very sparse, meaning that it cannot produce predictions for many of the sub-trajectories, because it was trained with a limited number of trajectories (i.e., only the recently completed ones). As a result, H-VIGO-GIS relies on the historical model for those sub-trajectories, which in turn results in less consideration of newly-emerged non-typical patterns, reducing the ability of H-VIGO-GIS to forecast rare events.

- *A. V. Khezerlou, X. Zhou, and L. Tong are with the Department of Business Analytics, University of Iowa, Iowa City, IA 52242 USA. E-mail: {amin-vahediankhezerlou, xun-zhou, ling-tong}@uiowa.edu.*
- *Y. Li is with the Worcester Polytechnic Institute, Worcester, MA 01609 USA. E-mail: yli15@wpi.edu.*
- *J. Luo is with the Machine Intelligence Center, Lenovo Group Limited, Hong Kong. E-mail: jluo1@lenovo.com.*

Manuscript received 24 Nov. 2017; revised 17 Apr. 2019; accepted 13 Aug. 2019. Date of publication 26 Aug. 2019; date of current version 3 Feb. 2021. (Corresponding author: Amin Vahedian Khezerlou.) Recommended for acceptance by J. Gama. Digital Object Identifier no. 10.1109/TKDE.2019.2937082

This paper is a significant extension to our prior work, which addresses the above limitations in depth by making the following contributions: *(1)* We adopt a radical approach in training the recent model by only using the trajectories that are significantly different from the historical trajectories. This ensures that if there is a significant change in the trajectory patterns, the recent model will capture all its available evidence. *(2)* We propose a novel parameter setting mechanism to dynamically adjust the combining weight of the recent and historical predictions through space and time. This mechanism tells the model, if the newly-emerged trajectory patterns are to be trusted in future predictions or not. *(3)* We propose a sparsity-improved destination prediction model to be used for training the recent model, which eliminates the sparsity problem and makes it possible to use the few, recent and significantly different trajectories for future predictions. We call the new Dynamic Hybrid model DH-VIGO-TKDE. Our extensive evaluations, using two large real-world datasets and an event simulator [9], show that the proposed model has considerably superior ability in terms of prediction accuracy and timeliness of forecasting statistically significant gathering events, compared to baselines.

The remainder of the paper is organized as follows. In Section 2, we discuss the related work. In Section 3 the destination prediction and event forecasting problems are formulated as computational problems. Sections 4 and 5 discuss the proposed solutions. Section 6 presents our evaluation and Section 7 concludes the paper.

2 Related Work

Availibility of advanced spatial sensing technologies, such as GPS devices, allows us to record veraiety of big spatial data such as spatial trajectories [10]. Therefore, mining of big trajectory data has captured the interest of researchers, from mining interesting locations and travel patterns [11] to semantic mining of locations [12] and destination prediction [13], [14], [15], [16]. In this work we focus on mining of urban gathering events using spatial trajectories.

To the best of our knowledge, trajectory destination prediction had not been used in the context of event detection or forecasting prior to our recent work [8]. The works of Martin Kulldorff and Neill [2], [3], [7], [17] and other recent works on event detection [4], [5] focus on *detecting* events using already observed counts at locations. The works of Zhou et al. and Vahedian et al. [9], [18] are based on real-time monitoring of significantly high flows in space and are categorized as early-detection and not forecasting. Hoang et al. proposed a crowd flow prediction model [19], which could potentially be used to predict gathering events. However this model works on a small number of large regions thus could not precisely locate events. In our recent work, we introduced a hybrid model (H-VIGO-GIS) of destination predictors consisting of a historical and a recent model for event forecasting, which enabled us to forecast gathering events ahead of the time. However, H-VIGO-GIS suffered from several limitations such as, no consideration of statistical significance when attempting to learn rare patterns, a fixed importance given to predictions produced by the historical model and the recent model and prediction sparsity of the recent model, due to limited number of its training trajectories. In this paper, we address those issues by making significant contributions on how and when the recent model is trained, how the predictions of the two models is combined, and how to take full advantage of the limited evidence of abnormality, using the redesigned destination prediction model.

Other works on event detection [20], [21], [22] use different types of data, e.g., geo-tagged social media posts, instead of mobility data. Therefore, they are not directly related to our work. The works of Zheng et al. [23], [24] on gathering patterns, focus on identifying a group of objects, trajectories of which gather together at some points in space and time. Their approach is not predictive and does not have an account of abnormality. These works are not directly related because, they solve a different problem than this paper, i.e., they answer the *who* question, rather than *where* and *when*.

The literature of *destination prediction* problem can be organized into two broad categories based on the data used: (1) using context-related and personal trip data [15], [25], [26], [27], [28], [29], [30], [31], [32] and (2) only using anonymous trip data (e.g., no traveler information). In this paper we only use anonymous trip data to predict destinations. Therefore, this work is not directly related to the first category. A less related work in this category is the work of Cancela et al. [33]. They focus on unsupervised modeling of pedestrian behavior and does not apply to our prediction problem. In the more related work in the second category, an important challenge is the complex dependencies among the segments of of an urban trip. To address this challenge, most researchers have adopted a Markov model-based approach, where the trip is decomposed into a sequence of transitions between locations in space. These transitions are modeled by low-order Markov chains, hence facilitating the calculation of the probability of an incomplete trip. The underlying assumption is that future movements along a trip are independent of the past movements. Xue et al. [13], [14] use this technique to calculate the destination probabilities using the Bayes rule. However, this Markov property assumption is frequently violated in real-world scenarios and adversely affects the prediction accuracy. Also the time and memory costs to learn the prediction models are excessively high and unfeasible for event forecasting. Li et al. [34] use a similar approach but distinguish between transitions among via points and transitions from via points to destinations in their calculations of the transition probabilities. This approach still does not properly address the independence issue in the Markov model-based approaches. Wang et al. [16] propose to condition the destination probability on the start location. They learn three transition probability matrices: source to destination, via points to destination and via point to via point. They also define a direction concept called Mobility Gradient for each sub-trajectory, which is used together with the three transition matrices to calculate the destination probabilities. However, in each of the transition matrices, the probabilities are calculated based on the assumption that they are independent of the other locations in the trip, which also imposes limitations on the accuracy of the predictions.

Kitani et al. [35] proposed an activity forecasting method that included a destination predictor. Their goal was to model the pedestrian behavior. However, their destination predictor is not applicable in our problem, because it only

produces ranking of potential destinations. In our work, to obtain predicted arrival counts at each location, we need to have the destination probability distribution of a moving object and ranking will not be sufficient for our calculations.

In our recent work [8], we proposed a destination prediction method that calculated the destination probabilities conditioned on all the three locations: source, current location and destination by proposing an efficient learning algorithm that allows real-time predictions with efficient use of memory. Our method resulted in higher accuracy compared to methods which took advantage of the Markov property. However, this approach potentially results in a sparse model, specially when the number of learning examples (trajectories) is limited. By sparsity we mean the model will not give any predictions to trips whose source and current location does not exactly match a learned trajectory. In this paper, we address the sparsity issue by calculating the destination probability using a different set of conditions.

3 Overview

The Gathering Event Forecasting through Trajectory Destination Prediction problem can be solved in three steps: (1) Build a model for trajectory destination prediction. (2) continuously predict the destinations of currently ongoing trajectories, and (3) Identify gathering events based on the predicted arrivals in every location and time slot. In this section we define basic concepts and formulate these steps into two sub-problems. Then we discuss the challenges in solving these problems as well as an overview of our proposed solution.

3.1 Concepts and Definitions

A *spatial field* S is a two-dimensional geographical region partitioned into a grid with cells $l_1, l_2, ..., l_n$ as distinct locations in space. Given S, the location of any moving object at a certain time can be mapped into a grid cell. For example, locations of a moving taxi in a trip can be represented by a sequence of grid cells, paired with the corresponding time.

Definition 1. *A* trajectory $Y = \{(s, t_s), (v_1, t_{v_1}), (v_2, t_{v_2}), \ldots, (v_n, t_{v_n}), (d, t_d) | s, v_i, d \in S\}$ *represents a trip of a moving object with a sequence of location and time pairs.* s *and* t_s *are the* source *location and start time of the trip, while* d *and* t_d *are the* destination *location and arrival time of the trip. The locations of the rest of the points* v_i *in the trajectory are called* via locations *of* Y.

Taxi trips are examples of trajectories, where the pick-up location is the source and the drop-off location is the destination. Based on Definition 1, we define sub-trajectories to represent incomplete trips (also referred to as ongoing trips).

Definition 2. *A* sub-trajectory $Y_c = \{(s, t_s), (v_1, t_1), ..., (c, t_c)\}$ *is the first few elements of trajectory* Y*, where* $c \in \{v_i\}$.

The first record of a sub-trajectory still represents the source of the trip but the last element of a sub-trajectory is a via-point instead of the destination of the trip. It represent the current location of the traveler.

Definition 3. *A* spatio-temporal region $R = (S_R, T_R)$ *is a pair of spatial region* S_R *and a time window* T_R*, where* S_R *is a rectangular sub-region of the spatial field* S.

In this paper we follow the definition of events in prior work [8]. For a spatio-temporal region R, we calculate the average number of trips ending in S_R during the same time of day as T_R, denoted as B_R or the baseline. The predicted number of trips ending in S_R during time T_R is denoted as C_R. We employ an Expectation-Based Poisson Model [7] to calculate the log-likelihood ratio between the hypothesis that there will be an elevation of arrivals in R versus the hypothesis that the predicted arrival count is normal. The log-likelihood ratio is calculated as follows:

$$LLR(R) = \begin{cases} C_R \log \frac{C_R}{B_R} + (B_R - C_R) & \text{if } C_R \geq B_R \\ 0 & otherwise. \end{cases} \quad (1)$$

As proved in a prior paper [18], the $LLR(R)$ score is statistically significant at α level if 1-$Pr(X < C_R) \leq \alpha$, where $X \sim Po(B_R)$.

Definition 4. *A* gathering event *is a spatio-temporal region* R *such that* $LLR(R)$ *is statistically significant at* α *level.*

3.2 Problem Statements

We formulate the two steps of our approach into two sub-problems, namely, the destination prediction problem and the event forecasting problem.

Sub-Problem 1: Destination Prediction. Given: a spatial field $\mathcal{S}$, a set of historical trajectories X, and a sub-trajectory Y_c, *Find:* the probability of each location in $\mathcal{S}$ to be the destination of Y_c as well as the probability distribution of the arrival time.

Sub-Problem 2: Event Forecasting. Given: a spatial field S, a set of historical trajectories X_h, a list of sub-trajectory U at current time, a target time-window t, and statistical significance threshold α, *Find:* Top-k gathering events at time t with the highest LLR scores. The *Objectives* of both sub-problems are to reduce computation cost while improving the accuracy of results.

3.3 Challenges and Solution Overview

Two challenges arise when designing the computational solutions to our proposed problem. We illustrate them with examples to motivate our solutions.

First, it is challenging to handle the trade-off between destination prediction accuracy and computational cost. Prior research have assumed the urban trips have low-order Markov property [13], i.e., the movement at each stage of the trip is only dependent on the current location but independent of previous steps. This assumption, although helpful in reducing computational cost, is unrealistic and limiting, and might lead to lower accuracy. Fig. 1 shows a counterexample of this assumption. A quiet two-way street (light-shaded, top-down) overpasses a busy one-way express way (darkly shaded, left to right) at c, where the traffic volume on the latter is 9 times of that on the former. A moving object started a trip at s and currently at c will be predicted to go right with 90 percent probability, if assuming first-order Markov property. However, considering the source of the trip, the probability of moving downwards (100 percent) is much higher than moving to the right (0 percent).

In our approach, we relax the Markov assumption to predict the destination of a moving object based on the current location and the start location. However, doing so requires significant amount of memory to store all the combinations

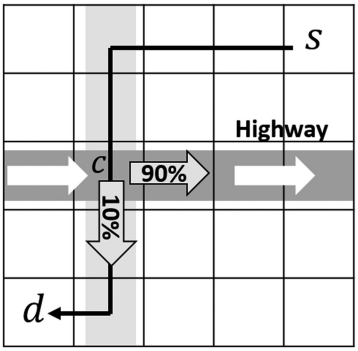


Fig. 1. Example of Markov property assumption.

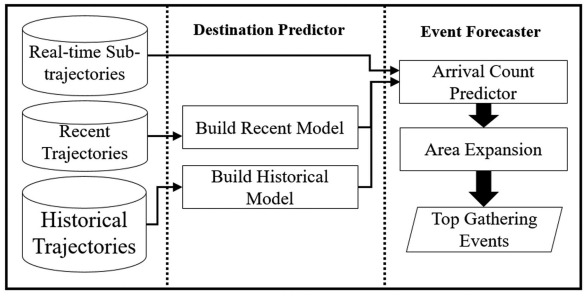


Fig. 2. High-level workflow of the event forecasting framework.

of source, via location, and destination. To address this challenge, we propose a Via Location Grouping (VIGO) approach that combines via locations with the same destination distributions to effectively reduce memory cost.

Another challenge is the temporal non-stationarity of urban trip patterns, which might deviate significantly from the historical patterns in case of rare events. For example, taxis taking passengers from a hotel zone are more likely to end up at the airport. However, during a big event such as a sports game or a concert, the most likely destination might be a stadium instead. This phenomenon is challenging to handle in gathering event forecasting since a global destination prediction model might not be able to accurately predict destinations for trips going to these events.

We address this challenge by proposing a dynamic hybrid event forecasting mechanism. A historical model learned from all the historical trajectories is dynamically combined with a recent model learned from recent trajectories that are significantly different from historical trajectories. The dynamic combination of the historical and recent models produces a powerful hybrid model that is robust to changes in historical patterns.

Fig. 2 demonstrates the high-level overview of the solution framework proposed in this paper and how the two components, destination prediction and event forecasting, interact with each other. A destination predictor is built using complete trajectories, then the event forecaster takes real-time sub-trajectories as input and uses the output of the destination predictor to forecast top gathering events.

4 Trajectory Destination Prediction

4.1 A Simple Classification Model

The destination prediction problem can be defined as a classification problem, i.e., every location is treated as a class. To classify each sub-trajectory, the Bayes classifier is commonly used to compute the probability of a location as the destination, conditioned on the observation of a sub-trajectory Y_c. Based on the definition of the conditional probability and Bayes' Theorem, we have:

$$p(d|Y_c) = \frac{p(d \cap Y_c)}{p(Y_c)} = \frac{p(Y_c|d) \times p(d)}{p(Y_c)}. \quad (2)$$

In this paper, by probability of a location, we mean the probability of the moving object being at that location. Therefore, $p(d)$ in Eq. (2) means probability of being at d. The approach of Eq. (2) involves calculating $p(Y_c|d)$. Commonly, related work (e.g., [13]) solve the problem based on the Markov property assumption. That means, the probability of a sub-trajectory $p(Y_c)$ is the product of the probabilities of all the pair-wise transitions. As previously illustrated in Section 3.3, this assumption is not realistic and may give poor results particularly when predicting destinations of trajectories along unpopular routes.

To address this limitation, we relax the Markov property assumption by using the combination of source and current location (s, c) to replace the entire partial trajectory Y_c. This suggests that the destination is dependent on the combination of source and the current location. We argue that this is a realistic yet computation-friendly simplification of Eq. (2), which achieves higher accuracy compared to the Markov-based approaches. We rewrite Eq. (2) in the following way:

$$p(d|s \cap c) = \frac{p(s \cap c \cap d)}{p(s \cap c)} = \frac{dest(s, c, d)}{via(s, c)} \quad (3)$$

In Eq. (3), s is the source of Y_c and c is its current location. $via(s, c)$ is the total number of trajectories with s as the source and c as a via location. $dest(s, c, d)$ is the total number of such trajectories that end at d in the data. A naive approach to learn the prediction model of Eq. (3) is to store the counts $via(s, c)$ and $dest(s, c, d)$ of every combination of s, c and d in S. In such a case, if S is a 128×64 grid, we will need to store $(128 \times 64)^3 \approx 5.5 \times 10^{11}$ counts. With a 4-byte data-type, we will need 2 TB of memory to learn and apply the model of Eq. (3). Considering the hardware capabilities of an average machine, this approach is infeasible. VIGO [8] is an efficient solution that addresses this challenge by significantly reducing the memory cost.

4.2 VIGO: A Scalable Via-Location Grouping Approach for Destination Prediction

The main idea behind VIGO is that many via locations of the same source share exactly the same destination probabilities. This is particularly true for locations along major roads with high traffic volume. For instance, consider a sequence of locations on a major expressway between two exits. These locations will for sure have the same destination probability distributions for a particular s. In our recent work [8], we proposed a scalable Via-Location Grouping approach which efficiently reduces the memory cost required to learn the model of Eq. (3).

4.2.1 The VIGO Index Data Structure

First we introduce the concept of a "via group", which is a key idea in our proposed VIGO Index structure.

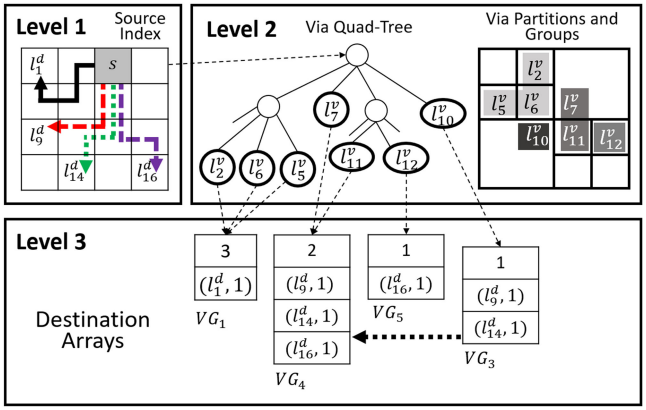


Fig. 3. The proposed VIGO index structure.

Definition 5. *A* via group *of a source location s, denoted as $VG_i(s)$ is a set of via locations l_j^v where for every $l_j^v \in VG_i(s)$ we have $via(s, l_j^v) > 0$ and $\forall l_j^v, l_k^v \in VG_i(s)$ and $d \in S$, $dest(s, l_j^v, d) = dest(s, l_k^v, d)$. Each via location of the same source could belong to only one via group.*

In Fig. 3, the destination counts of via nodes l_2^v, l_5^v, and l_6^v are exactly the same. They should form a via group of s. We only need to store one copy of the destination counts for the two of them. Fig. 3 demonstrates the VIGO index for a particular source location s. The first level is a two dimensional grid indexing all the possible source locations. Each location s points to a via quad-tree at the second level, which stores the counts of via locations c for source s. The via quad-trees save memory through partitioning of space, as the trajectories in the dataset dictate. The quad-trees replace dense matrices that would neglect spatial sparsity and waste memory. The leaf node v of the via quad-tree corresponding to source s stores the via count $via(s, c)$. For each via quad-tree leaf node, we add a pointer to the destination list. Each destination list has an array storing destinations and counts. Via locations pointing to the same destination list form a *via group* and they have the same destination count distribution. In level 2 of Fig. 3 the partitioning of the via tree is shown on the right. Each shaded area in the partitions represents a unique via group. There are four via groups in total in this example.

For each destination list, we track the number of via locations in the group. This count is followed by an array of destinations and their counts. For example, VG_4 has 2 via locations and it has 3 possible destinations: l_9^d, l_{14}^d, and l_{16}^d, all with count = 1.

4.2.2 Learning of the VIGO Model

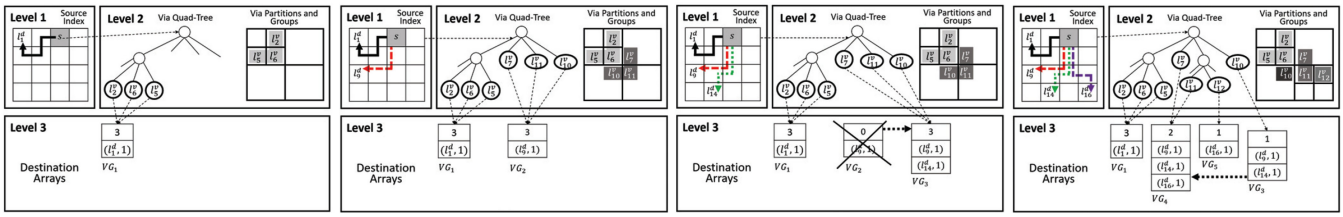
The above data structure can save quite much memory by reducing the number of via and destination counts stored. However, we also need to design an efficient and correct learning algorithm to build this model. To learn the VIGO model, we still read each historical trajectory once and go through its points from the source to the destination once.

Algorithm 1 shows the VIGO-Learner algorithm based on the proposed VIGO index structure. The input is a set of trajectories and the output is the updated VIGO Index. The algorithm takes one trajectory at a time, and updates the underlying VIGO index. For a trajectory Y with source s and destination d, we fetch the corresponding via quad-tree $Q_v(s)$. Then we scan each via location v in Y sequentially (Line 6) and update the VIGO index M according to the following rules.

Algorithm 1. The *VIGO Learner* Procedure

```
    Input: List of all trajectories (X)
    Output: A VIGO Index (M)
1   cur_trj ← 0; M[] ← NULL
2   for each Y ∈ X do
3     Q_v ← M[Y.s]
4     for each via location v ∈ Y do
5       if v not in Q_v then
6         via_node ← Q_v.insert(v)
7         via_node.count = 1
8         if VG_new == NULL then
9           VG_new ← Create new via group
10          VG_new.dst_array[0] ← (Y.d, 1)
11        via_node.group ← VG_new
12      else
13        via_node ← Q_v.get_node(v)
14        via_node.count + +
15        old_group ← via_node.group
16        new_group ← old_group.map
17        if new_group == NULL || old_group.trj ≠ cur_trj then
18          new_group ← Create a copy of old_group
19          new_group.increment_count(Y.d)
20          old_group.map ← new_group
21          old_group.trj ← cur_trj
22        via_node.group ← new_group
23        new_group.v_count + +
24        old_group.v_count − −
25        if (old_group.v_count) == 0 then
26          Delete old_group
27    VG_new ← NULL; cur_trj + +
28  return M
```

1) If a via location $v \in Y$ was not in Q_v, we insert it into Q_v and set $via(s, v) = 1$ (line 5-7).
2) All the "new" via locations v_i along Y that are newly inserted into Q_v should be assigned to the same new group. When scanning the first such via location, we create a new group VG_new and add d as the only destination, with count = 1 (Line 8-10). For all the following new via locations, we assign all of them to this group (Line 11).
3) All the "old" via locations v_j along Y that were already in Q_v also need to be moved to new groups, because a new destination d could potentially change the destination distribution of these via locations. Thus we create a new group to hold these via locations. Suppose v_j was in *old_group* before Y was learned (Line 15-16). If v_j is the first via location in *old_group* to be processed, then we create a new via group *new_group* by copying the destination array of *old_group*. If d is already in the array then we increment the count. Otherwise we append d at the end with count = 1. Then v_j's via group pointer will change to *new_group*(Line 17-19). To avoid creating multiple *new_group* for future via locations in *old_group*, we thereby put a pointer *map* in *old_group* to *new_group* to record this group transfer (Line 20). For future v_k in *old_group*, we use this pointer to find *new_group* and move v_k over (Line 22-23). Also,



(a) Learning Y_1 (b) Learning Y_2 (c) Learning Y_3 (d) Learning Y_4

Fig. 4. An example of learning the VIGO structure.

when a new trajectory comes in, all the group mapping information will be reset. We simply use another variable cur_trj in each destination array to make sure that old mappings are not used when learning the next trajectory (Line 21).

4) After each via location is moved to a new via group, we check the number of via locations remaining in the old_group and delete it if it is empty. This avoids unnecessary memory cost (Line 24-26).

An Illustrative Example. Fig. 4 shows an example of how the trajectories are learned into the VIGO structure. First, $Y_1 = \{s, l_2^v, l_5^v, l_6^v, l_1^d\}$ is fed to the learner (Fig. 4a), via locations l_2^v, l_6^v and l_5^v do not exist in the via tree of s. Therefore, a new via group is created $VG_1(s) = \{l_2^v, l_5^v, l_6^v\}$, which has only one destination l_1^d. When $Y_2 = \{s, l_7^v, l_{11}^v, l_{10}^v, l_9^d\}$ is fed to the learner (Fig. 4b), same scenario happens and group $VG_2 = \{l_7^v, l_{10}^v, l_{11}^v\}$ is created. When $Y_3 = \{s, l_7^v, l_{11}^v, l_{10}^v, l_{14}^d\}$ is fed to the learner (Fig. 4c), it copies VG_2 to create VG_3 and appends l_{14}^d. Then VG_2 gets deleted because no via locations point to it anymore. When $Y_4 = \{s, l_7^v, l_{11}^v, l_{12}^v, l_{16}^d\}$ is fed to the learner (Fig. 4d), VG_3 is copied to create VG_4. Then l_{16}^d is appended to VG_4. When Y_4 visits l_{12}^v, similar to Y_1 and Y_2 where the via point was being visited for the first time, a new group is created and l_{16}^d is added to it. However, VG_3 does not get deleted this time, because it has one remaining member l_{10}^v after losing l_7^v and l_{11}^v. Finally the VIGO index has four destination arrays, one for each via group.

We show, through the following lemma and theorem that the VIGO Learn algorithm can correctly build the VIGO Index with all the necessary counts.

Lemma 1. *Given a set of via locations $l_1^v, l_2^v, ...l_k^v$ that belong to the same via group $VG_i(s)$. If an incoming trajectory Y traverses all of them, then after Y is learned, these via locations will still belong to the same via group in VIGO.*

Proof. Since $l_1^v, l_2^v, ...l_k^v$ were in the same via group $VG_i(s)$, they share the same destination array $Dest(s, VG_i(s))$. Since each trajectory only has one unique destination, the new destination count distributions of v_i, $i = 1...k$ after learning Y are identically $Dest(s, VG_i(s)) \bigcup (d, 1)$. Per the definition of a via group, $l_1^v, l_2^v, ...l_k^v$ still belong to the same via group. □

Theorem 1. *The VIGO Learner algorithm is correct, i.e., the via locations assigned in the same via group in a learned VIGO Index always have the same destination distribution.*

Proof. Algorithm 1 always moves the via locations that were in the same old via group and passed by the same trajectory to the same new via group. According to Lemma 1 and Definition 5, the VIGO Learner algorithm is correct. □

4.3 Spatio-Temporal Destination Prediction

To complete the entire trajectory destination prediction component presented in Fig. 2, we need to predict not only the destination location, but also the arrival time. However, Eq. (3) only predicts the location of the destination. To predict the destination time, we calculate travel time probability distribution between pairs of via and destination locations, defined as $p(\Delta t|c, d)$, where d is destination and c is a via location and $\Delta t = t_d - t_c$. Therefore, we compute the probability of spatio-temporal destination $\{d, t_d\}$ for sub-trajectory Y_c with source s and current location c using the following equation:

$$p(\{d, \Delta t\}|s \cap c) = p(d|s \cap c) \times p(\Delta t|c, d). \quad (4)$$

Eq. (4) is obtained based on the assumption that the travel time between two given points c and d is independent of the points the trajectory visited before c. In other words, the travel time between two points in space only depends on the two points themselves. We argue that this assumption is reasonable because unlike destination, travel time is only determined by the route that will be taken rather than determined by a travel plan made ahead of time.

To save memory cost, we use a data structure similar to the top two levels of the VIGO Index to store the travel time distributions. Fig. 5 shows an example of this data structure. The top level is a two dimensional grid index for the current location c. Each grid points to a quad-tree of possible destinations from c. Each leaf node of this quad-tree contains the destination d as well as an array of possible travel time and the corresponding counts. Through the analysis of our data we find that more than 93 percent of all the trips are shorter than 30 minutes. Therefore we limit the size of the array to 30. The travel time distribution can be learned simultaneously with the VIGO Index structure. We integrate this process with the VIGO Learner algorithm and design a $VIGO\_ST$ algorithm. The pseudo code is presented in Algorithm 2. The algorithm takes one trajectory at a time and scans all the via locations sequentially. A via location v is used to update the VIGO index first (Line 5), and then the count for travel time $d.t - v.t$ is incremented (Line 6-9). The output is an integrated ST destination predictor M_{ST}. Compared to the learning phase,

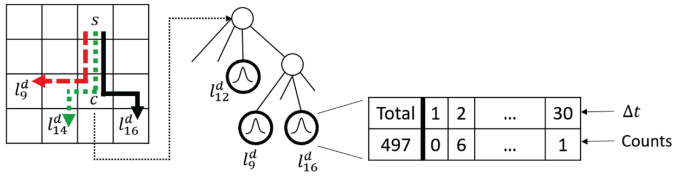


Fig. 5. Travel time distributions.

destination prediction using the ST destination predictor is relatively simple. Given a partial trajectory Y_c with source s and current location c, we first predict the destination location probability. This can be done by finding $via(s, c)$ at the via node of c in VIGO index. Then we also find $dest(s, c, d_i)$ for all the possible destinations d_i by scanning every entry in the destination array of c's via group. The destination probability is calculated according to Eq. (3). Then we find all the possible travel times between c and d_i and their probability. Finally we use Eq. (4) to calculate the ST destination probabilities.

Algorithm 2. VIGO_ST Learner

Input: List of all trajectories (X)
Output: A Spatio-Temporal Destination Predictor M_{ST}
1 $cur\_trj \leftarrow 0$; $M \leftarrow NULL$; $L \leftarrow NULL$
2 **for** *each* $Y \in X$ **do**
3 $Q_v \leftarrow M[Y.s]$
4 **for** *each via location* $v \in Y$ **do**
5 **Same as Line 5-26 in VIGO Learner**
6 **if** $L[v]$.*get_node()* == *NULL* **then**
7 $node \leftarrow L[v]$.insert(d)
8 $node$.count[t_d - t_v]++
9 $node$.total++
10 **Same as Line 27 in VIGO Learner**
11 return $M_{ST} = (M, L)$

Algorithm 3. VIGO_ST Predictor

Input: A VIGO_ST Model $M_{ST} = (M, L)$, Sub-trajectory Y_c
Output: Destination probability D at each location and time
1 $D \leftarrow \emptyset$
2 $via\_node \leftarrow M[Y_c.s]$.get_node($c$)
3 $via\_count \leftarrow via\_node.count$
4 $dst\_array \leftarrow via\_node$.group.$dst\_array$
5 **for** *each* d in dst_array **do**
6 $prob_d = dst\_array[d]/via\_count$
7 **for** *each* Δ_t *in* $L[c]$.*get_node*(d) **do**
8 $prob_t = (L[c]$.get_node$(d))[d]$
9 $D[d][t_c + \Delta_t] \leftarrow prob_l \times prob_t$
10 **return** D

VIGO_ST is the building block of our DH-VIGO-TKDE event forecasting method. It is a state-of-the-art destination prediction model that implements the Bayes Classifier for destination prediction problem. VIGO_ST has two key properties that make it ideal for the gathering event forecasting problem: (1) It produces empirical destination probability distribution of a given sub-trajectory across space and future time. (2) It is memory and time-efficient, therefore can be kept in memory at all times and can be used in a real-time setting. In the next section, we show how we use these capabilities and present how we build a Dynamic Hybrid event forecasting model using VIGO_ST.

5 Gathering Event Forecasting

This section presents the Event Forecasting component of our proposed solution. First, we present our baseline H-VIGO-GIS from our recent work [8]. Through the rest of the section, we propose our novel Dynamic Hybrid model DH-VIGO-TKDE.

5.1 Baseline: A Hybrid Event Forecasting Model (H-VIGO-GIS)

As mentioned in Section 3.3, due to the temporal non-stationarity in the urban trips, a global prediction model may not make accurate predictions at all times, especially for trips to rare gathering events. Instead, recent trajectories may better reflect short-term changes of trip patterns. Therefore, we propose a hybrid destination prediction model, which consists of a historical model learned once using long-term historical data, and a recent model built only based on recently observed trajectories. The final destination probability of each location and arrival time is calculated as a weighted average of the results from these two models. We re-train the recent model for every time-step to continuously predict the arrival count at each location and time.

There are two sets of trajectories: historical trajectories X_h, and recent trajectories X_τ which contains the completed trajectories within the last τ time-steps. A historical model M_h is trained using X_h, and a recent model M_r using X_τ. Then the destinations of each sub-trajectory at time t_g is predicted using both M_r and M_h. The final destination probability is calculated as a weighted average as shown in Eq. (5). β is a weight between 0 and 1 to adjust how much we trust M_h versus M_r. $p_h(\{d, \Delta t\}|s, c)$ and $p_r(\{d, \Delta t\}|s, c)$ are the destination probabilities of d, after Δt time-steps given by M_h and M_r respectively, while $\Delta t = t_d - t_c$. Note that due to the limited amount of data used in the recent model, it is possible that it does not have a prediction for a certain (s, c) combination. In such cases, β is set to 0.

$$p(\{d, \Delta t\}|s \cap c) = (1 - \beta) \times p_h(\{d, \Delta t\}|s \cap c) + \beta \times p_r(\{d, \Delta t\}|s \cap c). \quad (5)$$

Once the destination probability distribution of every sub-trajectory has been predicted, the arrival count is calculated as the expectation of trips ending at each location and time, given the list of sub-trajectories at t_c. We calculate the predicted arrival count of each location at target time t_g as follows:

$$A(d, t_c, t_g) = \sum_{Y_c \in U(t_c)} p(\{d, t_g - t_c\}|s \cap c), \quad (6)$$

Where $U(t_c)$ is the list of sub-trajectories at time t_c, s is the source and c is the current location of Y_c.

After the prediction is made, the recent model for time t_c is discarded and a new recent model is built in the next time-step $t_c + 1$. Meanwhile, all the trajectories completed at time t_c are learned into the historical model M_h.

5.2 DH-VIGO-TKDE: A Dynamic Hybrid Event Forecasting Model

H-VIGO-GIS incorporates a recent model to capture newly-emerged trajectory patterns. The assumption is that if there is a rare event, the trips that go to the event will not have the same behavior as historical trips. While, the recent model will learn the new behavior of the trips going to a certain location, it will learn the same historical pattern from the rest of the trips, which will make it harder for H-VIGO-GIS to learn the rare patterns, if they are accompanied by regular patterns. Next, we propose a method to address this limitation.

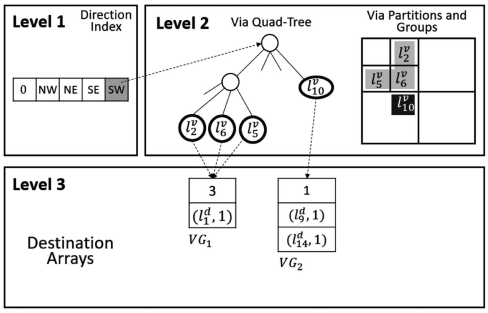


Fig. 6. VIGO-Lite index structure.

5.2.1 A Mechanism to Filter Recent Trajectories

As discussed above, not all the recent trajectories are useful for capturing recently emerged abnormal patterns. Only the trajectories that are significantly different from the historical ones are evidence of the abnormal patterns and should be learned into the recent model. For instance, let c be a via point of a given trajectory Y. Let s and d be its source and destination. We would like to decide whether to learn Y into the recent model or not, based on comparing (s, c, d) with historical trajectories. The pattern of historical trajectories is already captured in the historical model M_h. Given (s, c), the historical model M_h can predict the destination of Y based on the historical patterns. Let $\hat{d}_h$ be the predicted destination of Y according to M_h. By comparing the true destination, d to the predicted destination $\hat{d}_h$, we can determine if Y is abnormal or not. If d is too far from the predicted destination $\hat{d}_h$, then Y is likely to be abnormal. However, $\hat{d}_h$ is not simply one location. It is a probability distribution of the destination of Y among all the locations in S, according to M_h. In our grid setting, $\hat{d}_h$ is a joint distribution of X and Y grid coordinates. Therefore, we would like to calculate a distance between a point and a multi-variate distribution. To accomplish this, we use a distance measure called Mahalanobis distance [36], which is defined as follows:

$$D(\vec{x}) = \sqrt{(\vec{x} - \vec{\mu})^T Cov^{-1}(\vec{x} - \vec{\mu})}, \tag{7}$$

where $\vec{x}$ is an observation, in this case d. $\vec{\mu}$ is the mean of $\hat{d}_h$ and Cov is the covariance matrix of $\hat{d}_h$. Eq. (7) produces a distance measure between d and $\hat{d}_h$. This measure, in conjunction with a chi-square table, is widely used for outlier detection. We use the same table to determine whether Y is significantly different from history. If it is, we learn it into the recent model. Otherwise, it is skipped.

5.2.2 Sparsity of the Recent Model

Using only a subset of the recent trajectories for training, will result in sparsity. In other words, because the recent model is trained with limited number of trajectories, it will only provide predictions for a small set of (s, c) pairs. This will force us to rely on the historical model for those predictions, which in turn will adversely effect our ability to forecast rare events. To address this issue, we propose a new classification model. The new classification model is based on the model introduced in Eq. (3). However, it predicts the destination probability using the *direction* of the sub-trajectory, instead of its source, in addition to its current location. Sub-trajectory direction is a more generalized property of the sub-trajectory, compared to its source. We define five directions: north-east, north-west, south-east, south-west and null direction. Null direction means the sub-trajectory does not have a direction yet, i.e., current location is the same as the source location. The directions are determined by comparing the current location to the source location. For instance, if the current location is to the south-west of the source, the direction is south-west. Thus the classification model is proposed as follows:

$$p(d|w \cap c) = \frac{dest(w, c, d)}{via(w, c)}, \tag{8}$$

Where $w \in \{0, 1, 2, 3, 4\}$ is the direction of the sub-trajectory based on its source and current location. $dest(w, c, d)$ is the number of trajectories, which had d as their destination while having c as their via point, at which they had the direction of w. $via(w, c)$ is the number of trajectories that had c as their via point, at which they had the direction of w. We adapt the VIGO structure, presented earlier, to this new model, which we call VIGO-Lite. Fig. 6 shows the VIGO-Lite index, learned from the same trajectories as Fig. 3. In the first level of the new structure, the location index is replaced by a direction index. Note that the via quad-tree and the learned groups are different from Fig. 3, because Fig. 6 only shows the via locations which were traversed in south-west direction. The spatio-temporal destination prediction model can be inferred in the same way as Eq. (4):

$$p(\{d, \Delta t\}|w \cap c) = p(d|w \cap c) \times p(\Delta t|c, d) \tag{9}$$

Finally, we have the following formula when considering both historical and recent models:

$$\begin{aligned} p(\{d, \Delta t\}|s \cap c) = (1 - \beta) \times p_h(\{d, \Delta t\}|s \cap c) \\ + \beta \times p_r(\{d, \Delta t\}|w \cap c). \end{aligned} \tag{10}$$

Where w is obtained using s and c. Algorithm 4 shows the pseudo code for the proposed classification model, learned into the VIGO-Lite structure. Algorithm 5 shows the pseudo code for spatio-temporal predictor using the spatio-temporal VIGO-Lite model learned in Algorithm 4. The two algorithms are different in the way they access the via quad-tree. They use the direction at current location, instead of the source location (lines 4-5 in Algorithm 4 and lines 2-3 in Algorithm 5).

Algorithm 4. VIGO-L_ST Learner

Input: List of all trajectories (X)
Output: A VIGO-L_ST Model M_{ST}
1 $cur\_trj \leftarrow 0$; $M \leftarrow NULL$; $L \leftarrow NULL$
2 **for** *each* $Y \in X$ **do**
3 **for** *each via location* $v \in Y$ **do**
4 $w =$ get_direction($Y.s, v$)
5 $Q_v \leftarrow M[w]$
6 **Same as Line 5-9 in VIGO_ST Learner**
7 **Same as Line 10 in VIGO_ST Learner**
8 return $M_{ST} = (M, L)$

Algorithm 5. VIGO-L_ST Predictor

Input: A VIGO-L_ST Model $M_{ST} = (M, L)$, Sub-trajectory Y_c
Output: Destination probability D at each location and time
1 $D \leftarrow \emptyset$
2 $w =$ get_direction($Y.s, v$)
3 $via\_node \leftarrow M[w]$.get_node(c)
4 **Same as Line 3-9 in VIGO_ST Predictor**
5 **return** D

5.2.3 Setting the Value of β Dynamically

Setting β to a suitable value is crucial for the model's ability to make correct predictions in case of unusual trip pattern. To this end, three situations can arise. (1) The trips behave differently from the historical observations because of a rare event. In this case, having a high value for β is desirable, as it will give more emphasis to recent behavior. (2) The trips behave as they have behaved in the historical observations. In this case, the value of β does not matter. Because the predictions resulting from recent observations will be the same as the predictions from the historical ones. (3) The trips are starting to go back to their normal pattern, after behaving differently for a while, e.g., when everybody has arrived at the venue of the rare event. In this case, the recent observations do not represent the future behavior of the trips. If we keep using a high value for β, we will over-estimate the arrival counts. Based on the three cases discussed, we cannot use a fixed value for β at all times.

Based on the ideas discussed above, we propose a mechanism to dynamically set the value of β. To this end, we continuously monitor the errors made by both historical and recent models. The key idea is, if the historical model is making more accurate predictions, we decrease the value of β. However, using one global β for all the locations is potentially problematic. For instance, while historical model can make better predictions at one region, decreasing β globally may prevent us from forecasting an anomalous event elsewhere. Therefore, we use different values of β for each sub-trajectory. That is to say, the value is decided based on the prediction errors of similar sub-trajectories that completed at current time. To measure the destination prediction error, we first define the prediction centroid.

Definition 6. *Consider sub-trajectory Y_c at time t_c. Let $D(Y_c)$ be the set of all (d, t_d) where $p(\{d, t_d - t_c\}|Y_c) > 0$ such that $d = (d_x, d_y)$. The prediction centroid is defined as*

$$\begin{aligned} d_x^{cen}(Y_c) &= \sum_{(d,t_d)\in D(Y_c)} p(\{d, t_d - t_c\}|Y_c) \times d_x \\ d_y^{cen}(Y_c) &= \sum_{(d,t_d)\in D(Y_c)} p(\{d, t_d - t_c\}|Y_c) \times d_y \\ t_d^{cen}(Y_c) &= \sum_{(d,t_d)\in D(Y_c)} p(\{d, t_d - t_c\}|Y_c) \times t_d. \end{aligned} \tag{11}$$

Prediction centroid is the weighted average of all spatio-temporal destinations based on their probability. Next we define the following error function to measure the performance of the models:

$$\epsilon^c(Y) = \quad |x - d_x^{cen}(Y_c)| + |y - d_y^{cen}(Y_c)| + |t - t_d^{cen}(Y_c)|. \tag{12}$$

x, y and t are the coordinates and the time of the true destination of Y. Y_c is the sub-trajectory of Y at location c. The above error function is the Manhattan distance of the prediction centroid at t_c from the true destination plus their time difference. We can use Eq. (12) to calculate prediction errors of both historical and recent models. Next, we define the following measure:

$$e^c(Y) = \begin{cases} \epsilon_o^c(Y) - \epsilon_h^c(Y) & \text{if} \quad \epsilon_o^c(Y) > \epsilon_h^{t_c}(Y) \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Eq. (13) obtains how much the historical model did better given a specific sub-trajectory, so that we can later decide which model to use when encountering a similar one. Therefore, we save all the values calculated using Eq. (13) for each completed trajectory at time t in a matrix E. The rows of E represent via location, and the columns represent the direction of sub-trajectories. An element E_{cw} is the average of e^c values for all the trajectories that completed at this time and had c as their via point at which they had the direction of w. Algorithm 6 shows the calculation of the matrix E.

Algorithm 6. Calculate_E

Input: Completed trajectories (H)
Output: Error Matrix E
1 $E_{count} \leftarrow 0$
2 **for** *each* Y in H **do**
3 **for** *each* c in Y **do**
4 **if** $e^c(Y) > 0$ **then**
5 $w =$ get_direction($Y.s, c$)
6 $E[c][w] = E[c][w] + e^c(Y)$
7 $E_{count}[c][w]$++
8 **for** *all* c *in* S **do**
9 **for** *all directions* w **do**
10 $E[c][w] = E[c][w]/E_{count}[c][w]$
11 **return** E

Matrix E provides us with the most recent evidence of how the two models performed. To take advantage of this evidence, we propose to use a parameter setting system similar to PID [37] to adjust the value of β for each sub-trajectory. A PID system is a parameter correction formula based on linear combination of three terms: Proportion of error, Integral of error and Derivative of error. In the current problem, the correction we need, only depends on the proportion of error, i.e., the larger the value of E_{cw}, the lower β needs to be. Therefore we only use the proportion term as follows:

$$\phi(c, w) = \rho \times E_{cw}. \tag{14}$$

Higher ρ results in more aggressive corrections, while a smaller ρ results in more conservative ones. Finally, to adjust β, we use the following formula:

$$\beta(c, w) = \begin{cases} 0 & \text{if} \quad \phi(c, w) > 1 \\ 1 - \phi & \text{if} \quad 0 \le \phi(c, w) \le 1. \end{cases} \tag{15}$$

Based on Eq. (15), the better the historical model performs, the lower the value of β will be.

5.2.4 Event Forecasting Algorithm

Algorithm 7 shows the proposed event forecasting algorithm. It shows how the novel dynamic parameter setting mechanism and the new proposed destination prediction model (VIGO-Lite) work together with Mahalanobis distance method. Line 1 trains the historical model. Line 3 calculates the error matrix E, using Algorithm 6. Line 4 filters the set of recent trajectories and only keeps the anomalous ones. Then, for each sub-

TABLE 1
Datasets Summary

	Number of Taxis	Number of Points	Number of Trips	Recording Frequency	Area Size	Time Coverage
Shenzhen	20 k	4 Billion	14 Million	10 to 15 (s)	64×32 km	Nov. 2014
Chengdu	181 k	1 Billion	6 Million	3 (s)	8×8 km	Nov. 2016

trajectory, a new β value is set using Eq. (15), denoted as β' (line 7-9). Note that if matrix E returns zero, the old β value is kept. This means historical model did not perform better, thus, β does not need adjustment. Next, the algorithm predicts the destination of the sub-trajectory using both recent and historical models. Then, the centroids are calculated and added to prediction list of the sub-trajectory for future error calculation (lines 10-12). Next, the predicted counts are calculated using the adjusted β (line 13). After obtaining the predicted counts at each location at time t_g, we find the top-k ST regions with statistically significant arrival counts based on Definition 4. Scalable algorithms have been proposed to identify regions of statistically significant hotspots and events [6], [7]. These algorithms find the most likely event by searching all the possible spatio-temporal regions with pruning strategies. However, finding the exact solution is computationally costly and is inapplicable in the context of real-time event monitoring. Thereby we use a heuristic algorithm to identify k events that are statistically significant.

Algorithm 7. Event Forecasting Procedure (DH-VIGO-TKDE)

Input: Historical trajectories (X_h), Recent trajectories (X_τ), Sub-trajectories (U), Completed trajectories (H), Current and target time t_c, t_g, baseline arrival count (B), k, α, β_{old}
Output: k significant gathering events

1 $M_h \leftarrow VIGO\_ST\_Learner(X)$; $A \leftarrow 0$; $E \leftarrow 0$
2 **while** *program not terminated* **do**
3 $E \leftarrow$ Calculate_E(H)
4 $X'_\tau \leftarrow$ Mahalanobis(X_τ)
5 $M_r \leftarrow VIGO - L\_ST\_Learner(X'_\tau)$
6 **for** *each* Y_c *in* U **do**
7 $w \leftarrow$ get_direction($Y_c.s, c$); $\beta' \leftarrow \beta_{old}$
8 **if** $E[c][w] > 0$ **then**
9 $\beta' \leftarrow \beta(c, w)$
10 $D_h \leftarrow VIGO\_ST\_Predictor(Y_c, M_h)$
11 $D_r \leftarrow VIGO - L\_ST\_Predictor(Y_c, M_r)$
12 $Y_c.pred.push(d_h^{cen}(Y_c), d_r^{cen}(Y_c))$
13 $A \leftarrow A + D_h \times (1 - \beta') + D_r \times \beta'$
14 $R, R_0 \leftarrow \emptyset$
15 **for** *all locations* d **do**
16 **if** *is_significant*$(A[d], B[d], \alpha)$ **then**
17 $R_0 \leftarrow R_0 \bigcup d$
18 $R \leftarrow$ *area_exapnsion*(R_0)
19 Sort R on $LLR(G)$ in descending order
20 **output** R.top(k)
21 Update X_τ, U, M_h; $t_0 = t_0 + 1$

Given the predicted arrival count for each location at time t_g, we first find the grid locations with predicted arrival counts significantly higher than their respective baselines. We feed them as seeds to an area expansion algorithm to summarize the footprints of potential events. See supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2019.2937082 for details of this algorithm. Finally, the top k elements of R are returned. To continue the real-time prediction, X_τ and U are updated and the trajectories that completed at t_c are learned into M_h. See the supplemental material, available online, for a detailed time complexity analysis of DH-VIGO-TKDE.

6 Evaluations

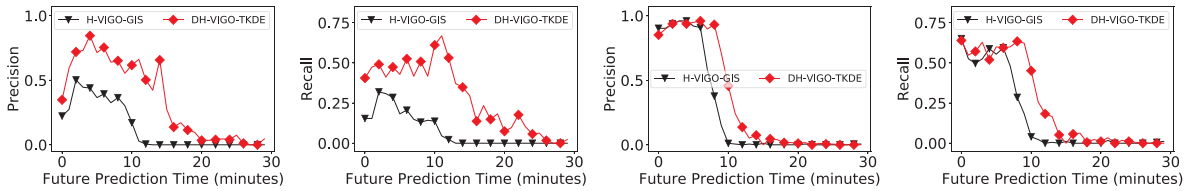
6.1 Dataset and Settings

We use trajectory datasets from two cities in China. The first dataset is recorded in Shenzhen during November 2014. It contains 4 billion GPS points of around 20 thousand taxis, recorded every 15 seconds. The dataset includes a field that marks whether the taxi is carrying a passenger or it is vacant. Using this field, we have extracted 14 million trajectories from this dataset. We map every point onto a 128×64 grid, with cells of size 500×500 meters. Our time-step is one minute. The second dataset is recorded in Chengdu during November 2016 and is publicly released by Didi Chuxing Technology Co., which offers a popular ride-sharing service in China. It contains around 6 million trajectories with GPS points recorded every 3 seconds [38]. We map every point onto a 64×64 grid, with cells of size 130×130 meters. Our time-step is one minute. Our default setting for the parameters is $\rho = 5$, $\tau = 30$ and $k = 5$. Table 1 shows a summary of the datasets. The methods were implemented in C++ and the experiments were run on an Intel Xeon E5 2.4 GHz with 256 GB of memory.[1]

6.2 Prediction Accuracy Evaluation

In this experiment, we analyze the ability of the proposed method in predicting gathering events. We split both datasets into training and testing sets. The training sets are the first 23 days of the month and the testing sets are the remaining 7 days. It is important to note that there is no ground truth for abnormal gathering events available. However, for the purpose of this evaluation, we follow the definition of gathering event in Section 3.1. For each dataset, we compile a list of all the spatio-temporal regions that satisfy definition 4, based on true counts. Then we compare the predicted list of events with the true list of events. If the predicted event is within 4 grid cells of the true event and within half an hour, we mark the prediction as a true positive. Fig. 7 shows the result. Figs. 7a and 7c show the precision of the proposed method compared to the baseline in Shenzhen and Chengdu. The x-axis is the time in the future. For Shenzhen, DH-VIGO-TKDE has a precision of above 0.5 for prediction of events between 2 minutes in the future and 15 minutes in the future, reaching max precision of 0.85. H-

1. All the implementation source code, along with a sampled version of the Shenzhen dataset is available at:
https://www.biz.uiowa.edu/faculty/xzhou/paper/VIGO_TKDE/data_sample/



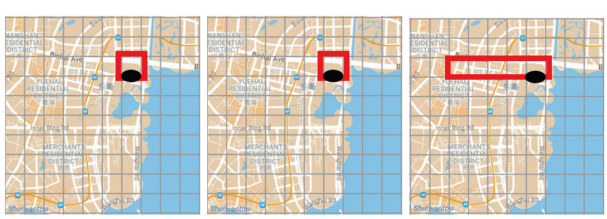
(a) Precision (Shenzhen). (b) Recall (Shenzhen). (c) Precision (Chengdu). (d) Recall (Chengdu).

Fig. 7. Prediction performance of DH-VIGO-TKDE versus H-VIGO-GIS.

VIGO-GIS has a precision of below 0.5 except the third minute in the future. For Chengdu, DH-VIGO-TKDE has a precision of above 0.85 for prediction of events 9 minutes in the future, reaching max precision of 0.97. The precision for H-VIGO-GIS drops much earlier in the future. Figs. 7b and 7d show the recall of the proposed method compared to the baseline. For Shenzhen, DH-VIGO-TKDE demonstrates a similar superiority to H-VIGO-GIS in terms of recall. It reaches the max recall of 0.67 while H-VIGO-GIS shows a max recall of 0.31. For Chengdu, DH-VIGO-TKDE also demonstrates superiority to H-VIGO-GIS. It remains accurate further in the future compared to the baseline.

6.3 Case Study

In this section we demonstrate a case that shows an example of algorithm's success when applied to real-world dataset. We train different models for weekdays, Saturdays and Sundays in the month, excluding the day on which we perform the forecast. Then we run Algorithm 7 on all the days. Fig. 8 shows a predicted event on November 21st, 2014 in Shenzhen. The black dot in Figs. 8a, 8b, 8c is the location of Shenzhen Bay Sports Center. The red box in Figs. 8a-8b is the area output by Algorithm 7 at $\alpha = 0.01\%$. After we observed this output, we looked into public records and found that it corresponds to a real event, i.e., a concert that started at 20:00 with nearly 30,000 attendees [39]. Fig. 8a shows the forecast 27 minutes before the event (i.e., 19:32). Fig. 8b shows the prediction 5 minutes before the event (i.e., 19:45). Fig. 8c shows footprint of the event obtained by applying the area expansion algorithm to true arrival counts. Figs. 9a and 9b show heatmaps of the predicted counts around the event location. Fig. 9a shows predicted arrival counts when only the historical model is used, i.e., no recent model. We can see that the event at the stadium is completely missed, as the predicted counts at the stadium are very low. This is a failure of the historical model in capturing a rare pattern. Fig. 9b shows the predicted counts for the same period by DH-VIGO-TKDE. This time the predicted counts are correctly concentrated at the event location because the recent model captures the rare pattern of this particular event and successfully forecasts it.



(a) 27 minutes ahead of the time. (b) 5 minutes ahead of the time. (c) Event based on true arrivals.

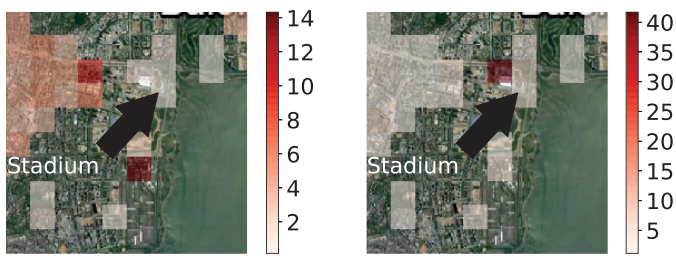
Fig. 8. Event forecast on day 21.

Fig. 10a shows the arrival count errors when only using historical model versus DH-VIGO-TKDE. It is clear that the historical model consistently underestimates the counts, hence is unable to capture the significant increase in the arrival counts, which is necessary to forecast the event. However, one can notice that shortly after the scheduled time of the event ($t = 0$) recent model over-estimates the counts. This over-estimation is caused by the drop in true counts after the start of the event. During this time period, the recent model still keeps predicting the destinations based on recently observed abnormal pattern of trips while the trip patterns have started to go back to normal. Dynamic setting of β is proposed to handle this situation by decreasing the value of β for the appropriate sub-trajectories. Fig. 10b compares the dynamic β (DH-VIGO-TKDE) with fixed $\beta = 0.9$ (H-VIGO-GIS) during the over-estimation period. H-VIGO-GIS consistently over-estimates the arrival counts, while count error of DH-VIGO-TKDE stays closer to 0.

6.4 Forecasting Simulated Events

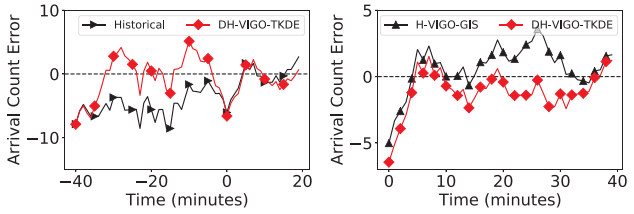
To evaluate the proposed event forecasting method in terms of timeliness and location accuracy, we need to compare the forecasts with ground truth. However, as mentioned above, the ground truth of the gathering events in urban areas is often not reliably available. To address this challenge, Vahedian et al. [9] proposed an event simulation mechanism in which gathering events are simulated by having simulated vehicles travel to a pre-determined location during a certain period of time. We use this simulation mechanism to evaluate our proposed method and compare it with the [9].

We use the same five locations and times as Vahedian et al. [9] for the simulated events. At each location we simulate two gathering events, one at noon and another at 7 PM, resulting in a total of 10 gathering events. Fig. 11 shows the locations of the simulated events in the study area. To simulate the trips, we first analyze the already verified gathering event in Section 6.3 in terms of travel distance and arrival time of the trips to the event. Fig. 12 shows the scatter plot of trip distance versus arrival time as well as the accumulated number of arrivals based on time. Note that 0 is the scheduled time of the event



(a) Historical model. (b) DH-VIGO-TKDE

Fig. 9. Predicted arrival counts (best viewed in color).



(a) Event Starting. (b) Event Ending.

Fig. 10. DH-VIGO-TKDE versus H-VIGO-GIS and historical model in the studied case.

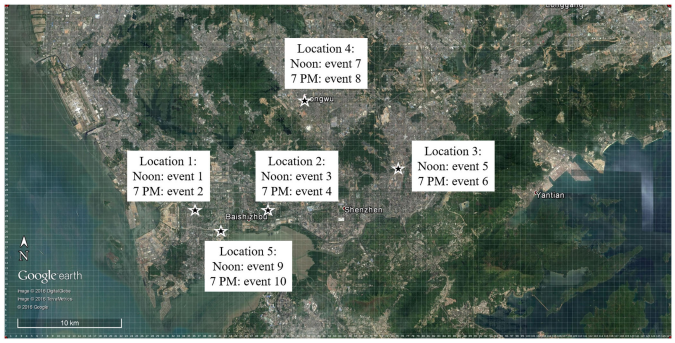


Fig. 11. Simulated events' locations [9].

and the time period covered by the plot is the time when the true counts were significantly high at the event location. To create each simulated trip for event e at time t_e and location d_e, we first randomly select a point (t, l) from Fig. 12. Then we randomly select a location at distance l from d_e to be the starting point of the trip and we set the arrival time of the trip to $t + t_e$. Then, we query the Google Maps Directions API [40] to obtain the trip trajectory. Based on our analysis in Fig. 12, the total number of arrivals in the period when the number of arrivals were significantly high was 797. It means in addition to 153 arrivals on average at this location and time of day, there were 644 more arrivals. Therefore, we create 644 simulated trips for each simulated event. Then we run DH-VIGO-TKDE to evaluate its performance based on simulated events. We would like to evaluate the performance of the method in terms of accuracy and timeliness. To evaluate the accuracy at each time step, we use the following error function called *destination error*:

$$e_{dest} = \min\{dist_M(R, R_k) | R_k \in Result(k)\}, \tag{16}$$

Where $dist_M(R, R_k)$ is the Manhattan distance between R and R_k, and $Result(k)$ is the set of top k gathering events forecast by the algorithm. Eq. (16) is the Manhattan distance between the closest forecast event and the event location. We set the maximum value of e_{dest} to be 10 grid cells, when it returns 10 or larger. We compare DH-VIGO-TKDE with H-VIGO-GIS and the SmartEdge algorithm [9]. We use default settings for our methods and the default settings for SmartEdge presented in [9]. Fig. 13 shows the average destination error through time. The error of DH-VIGO-TKDE starts dropping well before H-VIGO-GIS and much lower than SmartEdge. Moreover, DH-VIGO-TKDE reaches destination error of zero, while SmartEdge never reaches that value.

Next in this simulation analysis, we compare the timeliness of DH-VIGO-TKDE to the baseline. First, we consider the predicted count at a simulated event location, as we get closer to the simulated event time. Fig. 14 shows the

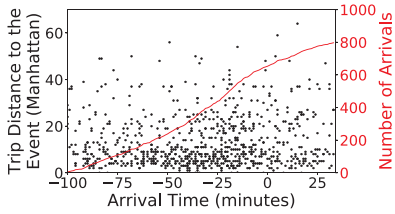


Fig. 12. Trip distance versus arrival time (Section 6.3).

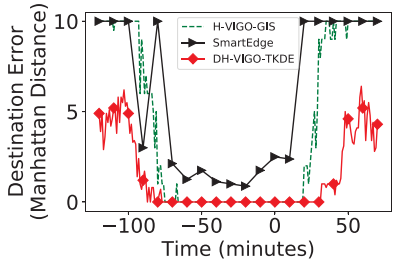


Fig. 13. Average destination error over time.

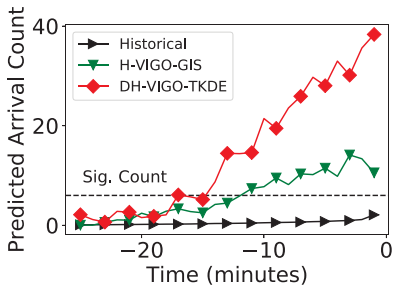


Fig. 14. Predicted counts at event location.

predicted counts of the proposed method and the baselines prior to the start of the event. The x-axis shows the time the prediction is performed. Time 0 is the time of the event. The dashed horizontal line is the cut-off value of statistically significant count. Fig. 14 shows that DH-VIGO-TKDE predicts a statistically significant count at around 17 minutes before the event. This value is 12 minutes for H-VIGO-GIS. Meanwhile, the historical model completely fails to predict any significantly high counts.

Fig. 15 shows how early a statistically significant count is predicted for a simulated event. The y-axis is the prediction time and the x-axis is the target time. The dashed line shows real-time detection, i.e., observation, no prediction. Both plots, being below the dashed line, show that the simulated event is consistently predicted ahead of the time i.e., before its observed arrival count became significant. Moreover, DH-VIGO-TKDE stays generally bellow the baseline, meaning it generally predicts the simulated event earlier.

6.5 Running Time Evaluation

Fast processing is crucial for continuous event forecasting. Thus, in this experiment we evaluate the running time of the proposed method and compare with baseline. It is expected that DH-VIGO-TKDE will have a longer running time than H-VIGO-GIS, because it performs more operations to reach higher accuracies. In this experiment, we evaluate its running time to determine whether this increased running time is justifiable, given the gained performance improvements. Fig. 16 shows the time cost of doing forecasting by varying grid size and τ. Event forecasting time increases by

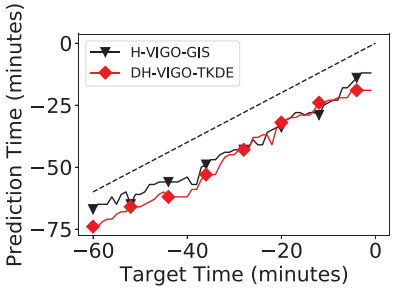
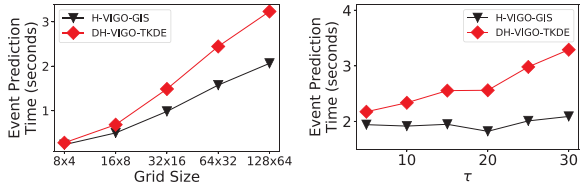


Fig. 15. Earliest prediction of the simulated event.



(a) Varying grid size. (b) Varying τ

Fig. 16. Running time of event forecasting.

increasing both parameters. In Fig. 16b as τ increases, the event prediction time also increases since more trajectories are used to train the recent model. For DH-VIGO-TKDE, setting β dynamically requires us to save predictions in advance and calculate prediction errors at each time-step for completed trips. Moreover, the method also calculates Mahalanobis distance for each recent trajectory. Fig. 16 shows that these operations have very small impact on the event forecasting time. The results show that even with finest grid resolution and largest τ the forecasting time cost is around 3 seconds. This level of performance keeps it possible for real-time forecasting of gathering events at 1-minute level.

6.6 VIGO Accuracy, Memory Cost and Running Time Evaluations

We proposed our destination prediction method, VIGO, in our prior work [8]. Therefore, we have not included its evaluations in the main text of this article. However, for the sake of this paper being self-contained, we have included the those evaluations in the supplemental material, available online. The evaluations show that VIGO outperforms stat-of-the-art destination prediction methods [14], [16], [41] in terms of prediction accuracy. VIGO also outperforms the baselines in terms of memory cost and running time. For detailed results, please refer to the supplemental material, available online.

7 Conclusions

In this paper, we addressed the gathering event forecasting problem through destination prediction of incomplete trips. Event forecasting in urban setting is important to traffic management and public safety. Prior event detection techniques are mostly descriptive, which only rely on on-site observations such as taxi drop-offs therefore lacking the ability to make forecasts ahead of the time. In our prior work [8], we proposed a predictive approach to identify future events through trajectory destination prediction. We designed a hybrid prediction mechanism (H-VIGO-GIS), which combined two destination prediction models, one was learned from historical trajectories and the other learned from recent ones. The role of the recent model was to capture the non-typical patterns of trajectories in case of rare gathering events, which the historical model is incapable of capturing. However, H-VIGO-GIS had a number of limitations regarding the training process of the recent model, the prediction sparsity of the recent model and fixed combination weight in the hybrid mechanism. Addressing these issues, in this paper we proposed a Dynamic Hybrid model (DH-VIGO-TKDE) that addressed the above-mentioned limitations. We performed comprehensive evaluations using two large real-world datasets and an event simulator. The experiments showed the proposed model significantly improved the prediction accuracy and timeliness of forecasting gathering events, resulting in superior precision and recall values, compared to H-VIGO-GIS.

Acknowledgments

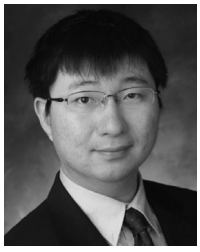
This work is partially supported by the NSF under Grant Number IIS-1566386. Yanhua Li was supported in part by NSF grants CNS-1657350 and CMMI-1831140, and a research grant from DiDi Chuxing Inc. This research uses datasets provided by DiDi Chuxing Technology Company [38].

References

[1] Shanghai stampede - wikipedia, 2014. [Online]. Available: https://en.wikipedia.org/wiki/2014_Shanghai_stampede
[2] M. Kulldorff, "A spatial scan statistic," *Commun. Statist.-Theory Methods*, vol. 26, no. 6, pp. 1481–1496, 1997.
[3] M. Kulldorff, R. Heffernan, J. Hartman, R. Assunçao, and F. Mostashari, "A space-time permutation scan statistic for disease outbreak detection," *PLoS Med.*, vol. 2, no. 3, 2005, Art. no. 216.
[4] L. Hong, Y. Zheng, D. Yung, J. Shang, and L. Zou, "Detecting urban black holes based on human mobility data," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2015, Art. no. 35.
[5] Z. Li, H. Xiong, and Y. Liu, "Mining blackhole and volcano patterns in directed graphs: A general approach," *Data Mining Knowl. Discovery*, vol. 25, no. 3, pp. 577–602, 2012.
[6] D. B. Neill and A. W. Moore, "Rapid detection of significant spatial clusters," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 256–265.
[7] D. B. Neill, "Expectation-based scan statistics for monitoring spatial time series data," *Int. J. Forecasting*, vol. 25, no. 3, pp. 498–517, 2009.
[8] A. Vahedian, X. Zhou, L. Tong, Y. Li, and J. Luo, "Forecasting gathering events through continuous destination prediction on big trajectory data," in *Proc. 25th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2017, Art. no. 34.
[9] A. V. Khezerlou, X. Zhou, L. Li, Z. Shafiq, A. X. Liu, and F. Zhang, "A traffic flow approach to early detection of gathering events: Comprehensive results," *ACM Trans. Intell. Syst. Technol.*, vol. 8, no. 6, 2017, Art. no. 74.
[10] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, 2014, Art. no. 38.
[11] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 791–800.
[12] X. Cao, G. Cong, and C. S. Jensen, "Mining significant semantic locations from gps data," *Proc. VLDB Endowment*, vol. 3, no. 1/2, pp. 1009–1020, 2010.
[13] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu, "Destination prediction by sub-trajectory synthesis and privacy protection against such prediction," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 254–265.
[14] A. Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang, and Y. Li, "Solving the data sparsity problem in destination prediction," *VLDB J.*, vol. 24, no. 2, pp. 219–243, 2015.
[15] D. Xue, L.-F. Wu, H.-B. Li, Z. Hong, and Z.-J. Zhou, "A novel destination prediction attack and corresponding location privacy protection method in geo-social networks," *Int. J. Distrib. Sensor Netw.*, vol. 13, no. 1, pp. 1–16, 2017, Art. no. 15.

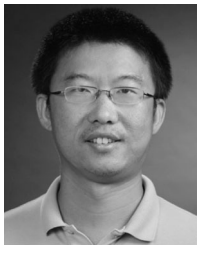
[16] L. Wang, Z. Yu, B. Guo, T. Ku, and F. Yi, "Moving destination prediction using sparse dataset: A mobility gradient descent approach," *ACM Trans. Knowl. Discovery Data*, vol. 11, no. 3, 2017, Art. no. 37.
[17] M. Kulldorff, W. F. Athas, E. J. Feurer, B. A. Miller, and C. R. Key, "Evaluating cluster alarms: A space-time scan statistic and brain cancer in los alamos, new mexico," *Amer. J. Public Health*, vol. 88, no. 9, pp. 1377–1380, 1998.
[18] X. Zhou, A. V. Khezerlou, A. Liu, Z. Shafiq, and F. Zhang, "A traffic flow approach to early detection of gathering events," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, Art. no. 4.
[19] M. X. Hoang, Y. Zheng, and A. K. Singh, "Fccf: Forecasting city-wide crowd flows based on big data," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2016, Art. no. 6.
[20] F. Chen and D. B. Neill, "Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1166–1175.
[21] J. Krumm and E. Horvitz, "Eyewitness: Identifying local events via space-time signals in twitter feeds," in *Proc. 23rd SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2015, Art. no. 20.
[22] C. Zhang, L. Liu, D. Lei, Q. Yuan, H. Zhuang, T. Hanratty, and J. Han, "Triovecevent: Embedding-based online local event detection in geo-tagged tweet streams," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 595–604. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098027
[23] K. Zheng, Y. Zheng, N. J. Yuan, and S. Shang, "On discovery of gathering patterns from trajectories," in *Proc. IEEE 29th Int. Conf. Data Eng.*, 2013, pp. 242–253.
[24] K. Zheng, Y. Zheng, N. J. Yuan, S. Shang, and X. Zhou, "Online discovery of gathering patterns over trajectories," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1974–1988, Aug. 2014.
[25] L. Chen, M. Lv, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervasive Mobile Comput.*, vol. 6, no. 6, pp. 657–676, 2010.
[26] K. Tanaka, Y. Kishino, T. Terada, and S. Nishio, "A destination prediction method using driving contexts and trajectory for car navigation systems," in *Proc. ACM Symp. Appl. Comput.*, 2009, pp. 190–195.
[27] J. Krumm, R. Gruen, and D. Delling, "From destination prediction to route prediction," *J. Location Based Serv.*, vol. 7, no. 2, pp. 98–120, 2013.
[28] L. Chen, M. Lv, Q. Ye, G. Chen, and J. Woodward, "A personal route prediction system based on trajectory data mining," *Inf. Sci.*, vol. 181, no. 7, pp. 1264–1284, 2011.
[29] J. J.-C. Ying, W.-C. Lee, T.-C. Weng, and V. S. Tseng, "Semantic trajectory mining for location prediction," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2011, pp. 34–43.
[30] J. Krumm and E. Horvitz, "Predestination: Inferring destinations from partial trajectories," in *Proc. Int. Conf. Ubiquitous Comput.*, 2006, pp. 243–260.
[31] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1345–1352.
[32] J. A. Alvarez-Garcia, J. A. Ortega, L. Gonzalez-Abril, and F. Velasco, "Trip destination prediction based on past gps log using a hidden markov model," *Expert Syst. Appl.*, vol. 37, no. 12, pp. 8166–8171, 2010.
[33] B. Cancela, A. Iglesias, M. Ortega, and M. G. Penedo, "Unsupervised trajectory modelling using temporal information via minimal paths," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 2553–2560.
[34] X. Li, M. Li, Y.-J. Gong, X.-L. Zhang, and J. Yin, "T-desp: Destination prediction based on big trajectory data," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2344–2354, Aug. 2016.
[35] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 201–214.
[36] P. C. Mahalanobis, "On the generalized distance in statistics," in *Proc. Nat. Inst. Sci. India*, vol. 2, no. 1, pp. 49–55, Sept. 2016.
[37] K. J. Åström and T. Hägglund, *PID Controllers: Theory, Design, and Tuning*, vol. 2. Research Triangle Park, NC, USA: Instrument society of America, 1995.
[38] Gaia open datasets, 2019. [Online]. Available: https://outreach.didichuxing.com/research/opendata/en/
[39] 10th anniversary of the mixc - super stars concert, 2014. [Online]. Available: http://news.ifeng.com/a/20141128/42597678_0.shtml
[40] "Routes and directions on google maps platform - google cloud," 2018. [Online]. Available: https://cloud.google.com/maps-platform/routes/, Accessed on: Aug. 17, 2018.
[41] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Yu, and Y. Tang, "Desteller: A system for destination prediction based on trajectories with privacy protection," *Proc. VLDB Endowment*, vol. 6, no. 12, pp. 1198–1201, 2013.

Amin Vahedian Khezerlou received the PhD degree in business administration from the University of Iowa, in 2019. He is currently an assistant professor of information technology with the College of Business and Economics, University of Wisconsin-Whitewater. His research interests include big data analytics and spatial and spatio-temporal data mining. He has published articles in ACM and IEEE Transactions and ACM conference proceedings.

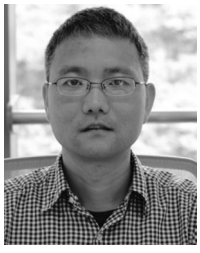


Xun Zhou received the PhD degree in computer science from the University of Minnesota, Twin Cities, in 2014. He is currently an assistant professor with the Department of Business Analytics, University of Iowa. His research interests include big data management and analytics, spatial and spatio-temporal data mining, and geographic information systems (GIS). He has published more than 40 papers in these areas and has received four best paper awards. He also served as a co-editor-in-chief of the *Encyclopedia of GIS*, 2nd Edition.

Ling Tong received the BBA degree in business analytics & information systems and the BS degree in mathematics. She is currently working towards the PhD degree in business administration at the University of Iowa. Her research interests include data mining and mathematical optimiziation. As an undergraduate, she won the third place in the Syngenta Crop Challenge in Analytics in 2016.



Yanhua Li (S'09–M'13–SM'16) received two PhD degrees in electrical engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009 and in computer science from the University of Minnesota at Twin Cities, in 2013, respectively. He has worked as a researcher with Noahs Ark LAB, Hong Kong, from August 2013 to December 2014, and has interned with Bell Labs, New Jersey, Microsoft Research Asia, and FutureWei Research Labs of America from 2011 to 2013. He is currently an assistant professor with the Department of Computer Science, Worcester Polytechnic Institute (WPI), in Worcester, Massachusetts. His research interests include big data analytics and urban computing in many contexts, including urban network data analytics and management, urban planning, and optimization. He is a senior member of the IEEE.



Jun Luo received the PhD degree in computer science from the University of Texas at Dallas, in 2006. He is a principal researcher with the Lenovo Machine Intelligence Center in Hong Kong. His research interests include big data, machine learning, spatial temporal data mining, and computational geometry. He has published more than 100 journal and conference papers in these areas.