# Structured Sparsity Model Based Trajectory Tracking Using Private Location Data Release

Minglai Shao, Jianxin Li, *Member, IEEE,* Qiben Yan, *Member, IEEE,* Feng Chen, *Member, IEEE,* Hongyi Huang, Xunxun Chen

**Abstract**—Mobile devices have been an integral part of our everyday lives. Users' increasing interaction with mobile devices brings in significant concerns on various types of potential privacy leakage, among which location privacy draws the most attention. Specifically, mobile users' trajectories constructed by location data may be captured by adversaries to infer sensitive information. In previous studies, differential privacy has been utilized to protect published trajectory data with rigorous privacy guarantee. Strong protection provided by differential privacy distorts the original locations or trajectories using stochastic noise to avoid privacy leakage. In this paper, we propose a novel location inference attack framework, iTracker, which simultaneously recovers multiple trajectories from differentially private trajectory data using the structured sparsity model. Compared with the traditional recovery methods based on single trajectory prediction, iTracker, which takes advantage of the correlation among trajectories discovered by the structured sparsity model, is more effective in recovering multiple private trajectories simultaneously. iTracker successfully attacks the existing privacy protection mechanisms based on differential privacy. We theoretically demonstrate the near-linear runtime of iTracker, and the experimental results using two real-world datasets show that iTracker outperforms existing recovery algorithms in recovering multiple trajectories.

**Index Terms**—Location privacy, differential privacy, multiple trajectory recovery, structured sparsity model.

## 1 Introduction

Nowadays, mobile devices, e.g., smartphones, have become an integral part of people's daily lives. These mobile devices collect almost indelible trajectories of user activities. Generally, a trajectory, comprised of a set of locations, can provide a wealth of useful information, such as individuals' habits, interests, activities, and relationships. Thus, publishing or analyzing the trajectory data, e.g., traffic flow trajectory, is of great value to many applications.

However, publishing the trajectory data poses serious threats to individual's location privacy [1], [2], cautious mobile users are reluctant to expose their locations. Consequently, in previous studies, location privacy preserving mechanisms [3], [4], [5], [6], [7] have been proposed to anonymize user identities using pseudonyms or obfuscate location coordinates by replacing real locations with forged locations or regions. Recently, differential privacy [8], [9], [10], [11], [12], [13], [14], which adds stochastic noises to protect data release, has been adapted to protect location or trajectory data release. Unfortunately, adding stochastic noises distorts the original locations or trajectories and reduces the utility of published trajectories [15], [16].

On the other hand, a variety of trajectory recovery methods have been proposed to eliminate noises and recover trajectories, which perform trajectory recovery on privacy-preserving location data as shown in Figure 1. In order to effectively capture spatio-temporal characteristic of moving trajectory, most of the existing methods for trajectory recovery, to the best of our knowledge, adopt a Markov model for modeling temporality to infer locations or trajectories [17], [18], [19], [20]. However, Markov-based methods can only recover a single trajectory at one time and achieve low recovery accuracy, since the Markov transition matrix used to recover trajectories is constructed by the perturbed/noisy locations without the consideration of suppressing noise.

In this paper, in order to advance the trajectory recovery capability and raise concerns about users' trajectory privacy, we propose a private trajectory recovering framework, iTracker, which attacks the differential privacy-based location protection model and simultaneously recovers multiple trajectories from the perturbed locations. iTracker utilizes a trajectory structured sparsity model that is capable of recovering multiple trajectories simultaneously, due to the fact that the model can effectively capture the interdependency among the locations and adaptively group trajectories. Then, an efficient approximation algorithm, based on the trajectory structured sparsity model, is designed to simultaneously recover multiple trajectories from privacy-preserving trajectories. Moreover, we provide theoretical analysis and experimental evaluation to examine the convergence and accuracy of iTracker framework.

The main contributions are summarized as follows:

- **Formulating a trajectory structured sparsity model.** In iTracker, the protected locations are labeled and mapped into a location sparsity matrix. Trajectory Earth Mover's Distance (TEMD) is employed to measure the quantity of the changed locations and the moving distance of the locations in adjacent time

- *M. Shao and J. Li are with Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University. E-mail: shaoml@act.buaa.edu.cn, lijx@act.buaa.edu.cn.*
- *Q. Yan is with Michigan State University. E-mail: qyan@msu.edu.*
- *F. Chen is with University of Texas at Dallas. E-mail: feng.chen@utdallas.edu.*
- *H. Huang is with Tsinghua University. E-mail: hhy17@mails.tsinghua.edu.cn*
- *X. Chen is with CNCERT/CC. E-mail: xx-chen@139.com.*

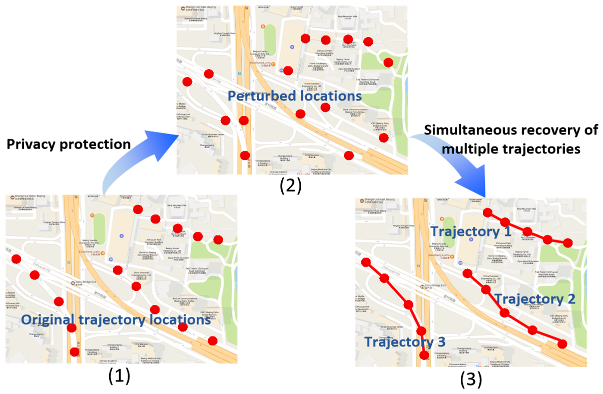


Fig. 1. Problem setting: the simultaneous recovery of multiple trajectories from unmarked perturbed locations protected by differential privacy based protection mechanisms.

slices, based on which we construct a trajectory structured sparsity model.

- **Designing an efficient approximation algorithm attacking the location protection mechanism (differential privacy based) and recovering multiple trajectories from perturbed locations simultaneously.** We propose an efficient attack framework, iTracker, based on the trajectory structured sparsity model. The iterative approximation algorithm suppresses noise at each iteration, which strengthens the recovery capability and improves recovery accuracy. We utilize a model projection oracle that finds the best approximation for recovering multiple trajectories simultaneously. In addition, we theoretically analyze the convergence rate and accuracy of the iTracker framework.
- **Performing comprehensive experiments to validate the effectiveness and efficiency of the proposed techniques.** Extensive experiments based on real-world datasets demonstrate that our proposed method outperforms state-of-the-art methods customized for multiple trajectories recovery.

The rest of the paper is organized as follows. Section 2 introduces the related work. Section 3 first defines the time-series trajectory and reviews the theory of differential privacy. Then, we build the structured sparsity model and formulate the trajectory recovery problem. Section 4 details the proposed approach for multiple trajectories recovery. Comprehensive experiments are provided in Section 5. Conclusion and future work are presented in the final section.

## 2 Related Work

In this section, we review the state-of-the-art models and techniques for recovering or analyzing human location records, which rely on a high degree of temporal and spatial regularities of human trajectories, e.g., simple and reproducible daily moving patterns [21]. Thus, such models and techniques can be employed by adversaries to infer sensitive locations from individuals' records. The proposed approaches can be summarized into three distinct types according to different data modeling methods, namely: (1) *state-space approaches*, (2) *data mining approaches* and (3) *template matching approaches*.

***State-space approaches.*** These approaches attempt to model the changes in terms of spatial sequences via time-series approaches such as the discriminative Conditional Random Fields (CRFs) [22], Hidden Markov Model (HMM) [23], or extensions of both two methods [24], [25]. More recently, DPSense [26] is proposed for the purpose of obtaining a good balance between efficiency and location privacy, in which spectrum-sensing tasks are published by the spectrum providers for the specific locations and time. In [27], the authors present the approach PriCSS, a framework proposed for the sensing service provider to choose the spectrum-sensing participants in a privacy-preserving manner. Generally speaking, these models have been successfully employed in coping with uncertainty, but they also suffer from high training complexity. For location and trajectory prediction, naturally, generative approaches such as HMM can be applied, since they support the generation of possible next locations with associated probabilities. Unfortunately, such techniques that recover trajectories based on probabilistic inference will induce excessive recovery error, and are incapable of simultaneously recovering multiple trajectories.

***Data mining approaches.*** *Data mining* explores association rules and frequent patterns by treating the trajectories as ordered sequences of time-stamped locations and employing the corresponding sequence analysis approaches such as the improved Apriori algorithm [28]. However, most previous data mining approaches measure the recovered locations by maximizing the confidence of these locations based on previous occurrences, and they disregard spatial and temporal distances among trajectories and locations, resulting in a low recovery accuracy. In addition, previous works have also analyzed the records of location through non-sequential unsupervised approaches such as probabilistic topic approaches known as Latent Dirichlet Allocation (LDA) [29]. Yet, the data mining technique is not easily applicable to recover private trajectories perturbed by stochastic noise.

***Template matching approaches.*** *Template matching* compares the extracted feature information with the pre-stored templates or patterns for recovering trajectories, which usually employs the longest common sub-sequence matching, or other heuristic string matching algorithms [30], [31]. However, in most cases, pre-stored patterns and templates are hard to establish.

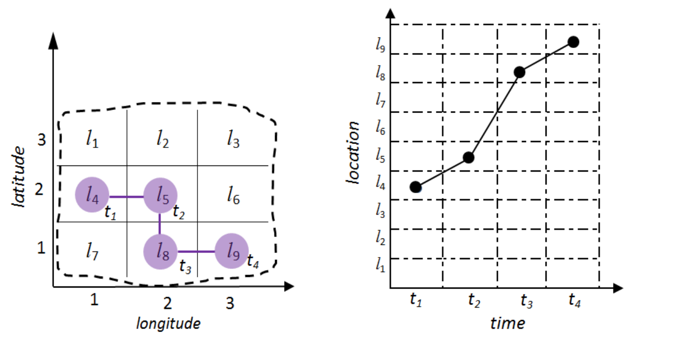
## 3 Background and Problem Formulation

In this section, we first define the time-series trajectory and review the theory of differential privacy. Then, we construct the structured sparsity model and formulate the problem.

### 3.1 Time-Series Trajectory

Here, we use a time-location coordinate framework and a map coordinate framework to represent the trajectories. They can be transformed into one another. $\boldsymbol{L}$ denotes the domain of space, which can be partitioned into a finer granularity, denoted as "unit", i.e, $\boldsymbol{L} = \{l_1,\ l_2,\ ...,\ l_n\}$, where each $l_i$ represents a location unit.

If the space is viewed as a map with longitude and latitude, a two-tuple coordinate can be used to represent



(a) Map coordinate framework (b) Time-location coordinate framework

Fig. 2. Two coordinate frameworks of trajectory: map coordinate framework and time-location coordinate framework. They can be transformed into one another.

a user's location. A trajectory can be represented by a sequence of the two-tuple coordinate and time slices. Meanwhile, we propose to project the map coordinate framework into time-location coordinate framework. In time-location coordinate, if no trajectory goes through a certain location, the state of this location is marked as "0". As time proceeds, the trajectory of a user can be denoted by a series of locations. Note that the two coordinate representations can be transformed into one another. Furthermore, all trajectories can be projected into any of these two coordinate frameworks. A formal definition of the time-series trajectory is presented below.

**Definition 1** (Time-series trajectory). *A trajectory is an ordered list of time-location pairs: $T = (l_1, t_1) \rightarrow (l_2, t_2) \rightarrow \ldots \rightarrow (l_i, t_i) \rightarrow \ldots \rightarrow (l_{|T|}, t_{|T|})$, where $|T|$ is the length of this trajectory, $\forall i (1 \leq i \leq |T|)$, and $l_i$ is a discrete spatial location.*

**Example 1.** Figure 2 demonstrates an example using the two coordinate frameworks to represent a trajectory. Consider user $u$ whose actual location in region $l_4$ at a given time $t_1$, then the location $l_4$ can be expressed by $(4, 1)$ in the form of the two-tuple coordinate. After being mapped into the time-location coordinate framework, a trajectory $T = (l_4, t_1) \rightarrow (l_5, t_2) \rightarrow (l_8, t_3) \rightarrow (l_9, t_4)$ is shown in Figure 2(b).

The set of time-series trajectories is defined as $\mathbb{T} = \{T_1, T_2, ..., T_{|\mathbb{T}|}\}$, where $|\mathbb{T}|$ is the size of $\mathbb{T}$. For simplicity, we assume that the trajectories in $\mathbb{T}$ are recorded for the same set of time stamps. For example, $1 \leq m, n \leq |\mathbb{T}|$, $\mathcal{T}(T_m) = \mathcal{T}(T_n)$, where $\mathcal{T}(T) = \{t_1, t_2, ..., t_{|T|}\}$ denotes the set of time slices in $T$. The set of locations of $\mathbb{T}$ is composed of the locations from all trajectories in $\mathbb{T}$.

Since every location is a discrete spatial point, we assume it difficult to find two trajectories $T_m, T_n \in \mathbb{T}$ that share the same location at any time point as discussed in [10], [32], namely $\forall t \in \mathcal{T}(T_m), T_m(t) \neq T_n(t)$.

## 3.2 Threat Model and Differential Privacy Based Trajectory Publishing

Trajectory data provides a great deal of benefits, but it gives rise to serious threats to the individual privacy [1], [2]. After aggregating sufficient trajectory data, the data collectors often publish these data to the internal departments or external partners for data analysis [16]. Even if the collection process is secure, some people may be curious about other individuals' private information. If the trajectory data is published in an inappropriate form, these adversaries may leverage background knowledge to link mobility traces to individuals [33] and further infer the sensitive individual information such as home address, employment, health condition, religion, etc.

To avoid the potential sensitive information leakage, differential privacy-based protection models have been used to protect the published locations or trajectories [9], [12], [34], [35], [36], since they provide a strong privacy guarantee. In this work, we consider the following differential privacy-based protection mechanism as a case study, while the proposed attack in Section 4 can also be appropriate for attacking other state-of-the-art protection mechanisms such as [12], [14].

**Definition 2** ($\epsilon$-geo-indistinguishability) [37]. *A mechanism $K$ satisfies $\epsilon$-geo-indistinguishability iff for all locations $l, l'$:*

$$K(l)(Z) \leq e^{\epsilon d(l,l')} K(l')(Z), \tag{1}$$

*where $l, l' \in \mathcal{X}$ and $\mathcal{X}$ is the set of possible locations of a user at a time slice, $K(l)(Z)$ is the probability that the reported point belongs to the set $Z \subseteq \mathcal{Z}$ and $\mathcal{Z}$ is the set of possible reported locations, $d(\cdot, \cdot)$ is the Euclidean distance metric, and $\epsilon$ ($\epsilon \geq 0$) represents the strength of protection.*

Moreover, given the parameter $\epsilon$ and an actual location $l$, the probability density function $D_\epsilon$ of the applied noise on another released location $l'$ is [37]:

$$D_\epsilon(l)(l') = \frac{\epsilon^2}{2\pi} e^{-\epsilon d(l,l')}. \tag{2}$$

For an actual location, we randomly select a noisy location according to the probability density function defined in Eq. (2) [37].

## 3.3 Structured Sparsity Model for Recovering Multiple Trajectories Simultaneously

In order to recover multiple trajectories simultaneously from locations protected by differential privacy, the trajectory structured sparsity model is formulated based on the *location sparsity matrix*.

**Definition 3** (Location sparsity matrix). *Based on the time-location coordinate framework, the location sparsity matrix can be defined as: a matrix $\mathbb{X} \in \mathbb{R}^{h \times w}$, where $w$ denotes the number of time slices, and $h$ denotes the number of locations. Each element $(l_i, t_j)$ in $\mathbb{X}$ represents a location $l_i$ released based on the differential privacy at time $t_j$, where $l_i \in \{l_1, ..., l_h\}, t_j \in \{t_1, ..., t_w\}$. If no trajectory crosses the location $l_i$ at time $t_j$, the state of this location is marked by zero in $\mathbb{X}$; otherwise, marked by $l_i$.*

For example, Figure 2(b) can be formulated as a location sparsity matrix $\mathbb{X} \in \mathbb{R}^{9 \times 4}$, where $(l_4, t_1)$, $(l_5, t_2)$, $(l_8, t_3)$ and $(l_9, t_4)$ in a trajectory are marked by $l_4$, $l_5$, $l_8$ and $l_9$ in different time respectively. Others are marked by 0.

Let $[h]$ and $[w]$ denote the set $\{1, 2, ..., h\}$ and $\{1, 2, ..., w\}$, respectively, and the indices of $\mathbb{X}$ can be denoted as $[h] \times [w]$. Let $S$ be a subset of $\mathbb{X}$'s indices, i.e., $S \subseteq [h] \times [w]$. And let $\mathbb{X}_S$ be the submatrix of a matrix

$\mathbb{X} \in \mathbb{R}^{h \times w}$ and $\mathbb{X}_S$ is identical to $\mathbb{X}$ but the entries not contained in $S$ are set to zero. A matrix $\mathbb{X} \in \mathbb{R}^{h \times w}$, is said to be $k$-sparse if at most $k \leq h \times w$ coordinates are nonzero. The support of $\mathbb{X}$, $\text{supp}(\mathbb{X}) \subseteq [h] \times [w]$, is also the set of indices corresponding to nonzero entries. For a matrix support set $S$, we denote the support of $L_t$ in $S$ as $\text{supp}_{\text{col}}(S, L_t) = \{l_i | (l_i, L_t) \in S\}$, where $L_t$ is the location set at time slice $t$ in $\mathbb{X}$.

A trajectory $T \subseteq [h] \times [w]$ is a set of locations in $\mathbb{X}$ with one location per column and $|T| = w$. The support of $\mathbb{T}$ is $\text{supp}(\mathbb{T}) = \{(l,t) \mid (l,t) \in T_i,\ for\ i \in \{1, 2, ..., |\mathbb{T}|\}\}$, where $|\mathbb{T}|$ is the total number of trajectories in $\mathbb{X}$. Furthermore, the Trajectory Earth Mover's Distance (TEMD) is defined based on [38] to measure the number of the changed locations and the moving distance of the locations in adjacent time slices.

**Definition 4** (TEMD). *The Trajectory Earth Mover's Distance of two adjacent time slice location sets $L_t, L_{t+1}$ is defined as:*

$$\text{TEMD}(L_t, L_{t+1}) = \min_{\pi : L_t \to L_{t+1}} \sum_{l_t \in L_t} |l_t - \pi(l_t)|, \tag{3}$$

*where $\pi$ ranges over all one-to-one mappings from $L_t$ to $L_{t+1}$, $L_t$ and $L_{t+1}$ are location sets at time slices $t$ and $t+1$ respectively. $L_t$ and $L_{t+1}$ contain the same number of locations. $|\cdot|$ is the absolute value of the variation of a location in adjacent time slices.*

**Definition 5** (STEMD). *Let $S \subseteq [h] \times [w]$ be the support of the location matrix $X$, $|\text{supp}_{\text{col}}(S, L_t)| = |\mathbb{T}|$ for $t \in [w]$. The TEMD of Support $S$ is defined as:*

$$\text{STEMD}(S) = \sum_{t=1}^{w-1} \text{TEMD}(\text{supp}_{\text{col}}(S, L_t), \text{supp}_{\text{col}}(S, L_{t+1})), \tag{4}$$

*where the $\text{supp}_{\text{col}}(S, L_t)$ is the support of column $L_t$ in $S$ and the $\text{supp}_{\text{col}}(S, L_{t+1})$ is the support of column $L_{t+1}$ in $S$.*

STEMD of the location matrix $\mathbb{X}$, denoted as $\Theta$, represents all the locations that change in consecutive time slices. Next, we provide the definition of trajectory structured sparsity model as follows.

**Definition 6** (Trajectory structured sparsity model). *Given the location matrix $\mathbb{X} \in \mathbb{R}^{h \times w}$, the trajectory structured sparsity model can be defined as:*

$$\begin{aligned} \mathbb{M}(\Theta, |\mathbb{T}|) = \{ & S \subseteq [h] \times [w] \mid \text{STEMD}(S) \leq \Theta, \\ & |\text{supp}_{\text{col}}(S, L_t)| = |\mathbb{T}|\ for\ t \in [w]\}. \end{aligned} \tag{5}$$

*where $|\mathbb{T}|$ is the number of trajectories, $w$ is the total time slices, $\text{supp}_{\text{col}}(S, L_t)$ is the support of column $L_t$ in $S$, and $\Theta$ is the* STEMD *of location matrix $\mathbb{X}$.*

Given the location matrix $\mathbb{X}$ of the protected trajectories and the trajectory structured sparsity model $\mathbb{M}(\Theta, |\mathbb{T}|)$, the multiple trajectory recovery can be realized by finding the **best approximation** for the real trajectory location matrix $X$ with the support $S \in \mathbb{M}(\Theta, |\mathbb{T}|)$ based on the structured sparsity model $\mathbb{M}(\Theta, |\mathbb{T}|)$:

$$\arg \min_{X' \in \mathbb{R}^{h \times w}} ||X - X'||_2^2 \quad s.t.\ \text{supp}(X') \in \mathbb{M}(\Theta, |\mathbb{T}|), \tag{6}$$

where location matrix $X \in \mathbb{R}^{h \times w}$ has the same basic definition and properties as $\mathbb{X}$, the trajectory structured sparsity model $\mathbb{M}(\Theta, |\mathbb{T}|)$ is defined as a family of structured supports: $\mathbb{M}(\Theta, |\mathbb{T}|) = \{S_1,\ S_2,\ \dots,\ S_N\}$ and $S_i \subseteq [h] \times [w]$.

## 4 iTracker Design and Theoretical Analysis

In this section, we first present the design of iTracker. iTracker attacks the location protection mechanism (differential privacy based) by recovering multiple trajectories from perturbed locations simultaneously. Then, we present the theoretical analysis on its convergence and accuracy.

### 4.1 iTracker

In order to find the best recovered multiple trajectories simultaneously from released locations protected by differential privacy, **a novel framework named as iTracker is introduced based on the trajectory structured sparsity model $\mathbb{M}(\Theta, |\mathbb{T}|)$ which can effectively capture the interdependency of the locations and adaptively group trajectories**. However, the problem in Eq. (6) is NP-hard. Instead of seeking for the exact solution to the problem, iTracker utilizes two approximation algorithms, namely UP-approximation Trajectories (UP(X)) and DOWN-approximation Trajectories (DOWN(X)):

1) An UP-approximation Trajectories oracle returns a support $S$, namely the indices of most of the relevant trajectory locations in $X$, such that the norm of $||X_S||_2$ is approximately maximized as shown in Section 4.2.

2) A DOWN-approximation Trajectories oracle returns a support $S$, namely the indices of most of the relevant trajectory locations in $X$, such that the norm of $||X - X_S||_2$ is approximately minimized as shown in Section 4.3.

**Algorithm 1** iTracker Framework

```
1:  Input: location matrix $\mathbb{X}$ of published trajectories, the number of iterations: $I$, measurement: $\mathfrak{M}$.
2:  Output: Estimated location matrix.
3:  $i = 0$, $X_0 = 0$;
4:  repeat
5:      $X_a = \mathbb{X} - \mathfrak{M}X_i$;
6:      $X_b = \mathfrak{M}^T X_a$;
7:      $X_c = X_i + \text{UP}(X_b)$;
8:      $X_{i+1} = \text{DOWN}(X_c)$;
9:  until $i = I$;
10: Return $X_{I+1}$.
```

**iTracker uses these two approximations to bounce between upper bound and lower bound to approach the optimal solution of the problem shown in Eq. (6) [39]. Furthermore, iTracker integrates these two approximations with a linear measurement $\mathbb{X} = \mathfrak{M}X + \tau$, where the matrix $\mathfrak{M}$ satisfies the variant of restricted isometry property named as Model-RIP**, the definition of which is presented below.

**Definition 7**. *The matrix $\mathfrak{M} \in \mathbb{R}^{p \times q}$ has Model-Restricted Isometry Property (Model-RIP) [40] with constant $\rho$, if for all $\text{supp}(X) \in \mathbb{M}(\Theta, |\mathbb{T}|)$:*

$$(1 - \rho)||X||_2^2 \leq ||\mathfrak{M}X||_2^2 \leq (1 + \rho)||X||_2^2. \tag{7}$$

**Integrating the two approximation algorithms and the measurement, iTracker reduces the noise in each iteration,**

**and its solution will converge to the most probable trajectories efficiently** (See Alg. 1).

As shown in Algorithm 1, $\mathfrak{M}^T X_a$ is used to update the location matrix in each iteration. Then, a subset of locations is identified via $\mathrm{UP}(X_b)$ (UP-approximation Trajectories) that returns locations with UP value, which is at least a constant fraction of the optimal UP value (Section 4.2). The locations in this subset are then merged with the current estimate to obtain the merged location matrix. Then a subset of locations is identified via $\mathrm{DOWN}(X_c)$ (DOWN-approximation Trajectories) that returns locations with DOWN value, which is at most a constant times larger than the optimal DOWN value (Section 4.3). After several iterations, the corresponding approximation oracle returns most of the relevant trajectory locations and exhibits provably robust convergence and recovery property.

## 4.2 UP-approximation Trajectories (UP(X))

Given the location matrix $X = X_b$ obtained in each iteration of Algorithm 1 and the trajectory structured sparsity model $\mathbb{M}(k, |\mathbb{T}|)$, UP-approximation Trajectories oracle returns a support $S$, namely the indices of most of the relevant trajectory locations in $X$, such that the norm of $||X_S||_2$ is approximately maximized, formally written as:

$$||X_S||_2 \geq C_{\mathrm{U}} \cdot \max_{S' \in \mathbb{M}(\Theta, |\mathbb{T}|)} ||X_{S'}||_2, \tag{8}$$

where $0 < C_{\mathrm{U}} < 1$ is a constant.

**Algorithm 2** UP-approximation Trajectories

```
1: Input: location matrix $X = X_b$ obtained in each
   iteration of Algorithm 1, $\Theta$.
2: Output: $|\mathbb{T}|$ trajectories.
3: $X_1 = X$;
4: for $i \in \{1, ..., |\mathbb{T}|\}$ do
5:     $T_i = \arg\max_{T_i \in X_i} \mathbb{L}_i(T_i)$ $s.t.$ $\mathrm{STEMD}(T_i) \leq \lfloor \frac{\Theta}{i} \rfloor$;
6:     $X_{i+1} = X_i$;
7:     for $(l, t) \in T_i$ do
8:         $X_{i+1_{(l,t)}} = 0$;
9:     end for
10: end for
11: Return $\bigcup_{i=1}^{|\mathbb{T}|} T_i$.
```

Generally, a trajectory $T$ in $X$ is a set of locations with $\mathbb{L}(T) = ||X_T||_2^2 = \sum_{l_i \in T} |l_i|^2$ and $\mathrm{STEMD}(T) = \sum_{i=1}^{w-1} |l_i - l_{i+1}| = \mathrm{STEMD}(\mathrm{supp}(X_T))$, where $l_1$, ..., $l_w$ are the locations of trajectory $T$ from time 1 to $w$. Then, we can get the locations support $S$ by finding $|\mathbb{T}|$ trajectories iteratively. The details of UP-approximation Trajectories are presented in Algorithm 2 and an example of finding $|\mathbb{T}|$ trajectories is shown in Example 2.

**Example 2.** A matrix including protected trajectories based on differential privacy is shown in Figure 3, where $\Theta = 5$, $|\mathbb{T}| = 3$, location $l_i \in \{1, 2, 3, 4, 5, 6\}$, $t \in \{t_1, t_2, t_3\}$. In the first iteration, $\lfloor \frac{\Theta}{i} \rfloor = \lfloor \frac{\Theta}{1} \rfloor = 5$, trajectory $T_1 = (5, t_1) \to (6, t_2) \to (5, t_3)$ is selected based on the constraint that $\max_{T_i \in X} \mathbb{L}(T_i)$ subject to $\mathrm{STEMD}(T_i) \leq \lfloor \frac{\Theta}{i} \rfloor$. After that, all locations selected in $T_1$ are set to 0 marked by pink in Figure 3. Then, $i = 2$, $\lfloor \frac{\Theta}{2} \rfloor = 2$. Repeat the above operation, we can get $T_2 = (3, t_1) \to (4, t_2) \to (4, t_3)$. Similarly, in iteration 3, $T_3 = (2, t_1) \to (0, t_2) \to (2, t_3)$. Through this process, almost all locations in $X$ are found. Moreover, all locations are distributed into three disjoint trajectories. Note that the perturbed locations that are distant from the original trajectories will be removed.

In Algorithm 2, given $\mathbb{L}_i(T_i) = ||X_{i,T}||_2^2 = \sum_{l_i \in T_i} |l_i|^2$, we derive the following results.

**Theorem 1.** *The approximation results of the trajectories satisfy:*

$$||X_S||_2 \geq \frac{1}{2} \max_{S' \in \mathbb{M}(\Theta, |\mathbb{T}|)} ||X_{S'}||_2. \tag{9}$$

*Proof.* Let $S_O$ be the optimal support of $X$ and $S$ be the support of trajectories returned by UP-approximation Trajectories algorithm. And let $\{O_1, ..., O_{|\mathbb{T}|}\}$ with $\mathrm{STEMD}(O_1) \geq \mathrm{STEMD}(O_2) \geq ... \geq \mathrm{STEMD}(O_{|\mathbb{T}|})$ be the trajectories corresponding to the decomposition of the $S_O$. In the algorithm of UP-approximation Trajectories, when we obtain the trajectory $T_i$ in $X_i$, there are two cases: 1) the trajectories $\{T_1, ..., T_{i-1}\}$ only contain less than half of the locations of $O_i$ in $X$, namely $\mathbb{L}_i(O_i) \geq (1/2)\mathbb{L}(O_i)$; 2) the trajectories $\{T_1, ..., T_{i-1}\}$ contain more than half of the locations. Let $A = \{i \in [|\mathbb{T}|] \mid$ case 1) holds for $T_i\}$, $B = \{i \in [|\mathbb{T}|] \mid$ case 2) holds for $T_i\}$. Then we can get:

$$\begin{aligned} ||X_S||_2^2 = \sum_{i=1}^{|\mathbb{T}|} \mathbb{L}_i(T_i) &= \sum_{i \in A} \mathbb{L}_i(T_i) + \sum_{i \in B} \mathbb{L}_i(T_i) \\ &\geq \sum_{i \in A} \mathbb{L}_i(T_i) \geq \frac{1}{2} \sum_{i \in A} \mathbb{L}(O_i). \end{aligned} \tag{10}$$

For every $O_i$ and $i \in B$, let $\iota_i = O_i \cap \bigcup_{j<i} T_j$, namely the locations of $O_i$ have been covered by some $T_j$ when we target $T_i$. Then we can get:

$$\sum_{(l,t) \in \iota_i} |X_{l,t}|^p = \mathbb{L}(O_i) - \mathbb{L}_i(O_i) \geq \frac{1}{2} \mathbb{L}(O_i), \tag{11}$$

$$\sum_{i \in B} \sum_{(l,t) \in \iota_i} |X_{l,t}|^p \geq \frac{1}{2} \sum_{i \in B} \mathbb{L}(O_i), \tag{12}$$

$O_i$ and $\iota_i$ are pairwise disjoint for every $i \in B$, so we can get $\iota_i \subseteq \bigcup_{j<i} T_j$. Therefore,

$$||X_S||_2^2 = \sum_{i=1}^{|\mathbb{T}|} \mathbb{L}(T_i) \geq \sum_{i \in B} \sum_{(l,t) \in \iota_i} |X_{l,t}|^p \geq \frac{1}{2} \sum_{i \in B} \mathbb{L}(O_i). \tag{13}$$

Combining Eqs. (10) and (13), we can get:

$$2||X_S||_2^2 \geq \frac{1}{2} \sum_{i \in B} \mathbb{L}(O_i) + \frac{1}{2} \sum_{i \in A} \mathbb{L}(O_i), \tag{14}$$

which is equivalent to (9).

**Theorem 2.** *The time complexity of UP-approximation Trajectories algorithm is $O(|\mathbb{T}| w h^2 \Theta)$.*

*Proof.* In the process of recovering each trajectory in location-comprised matrix $X$, we obtain the trajectory of the largest weight at corresponding locations and the corresponding amount of STEMD support $\theta \in \{1,\ 2,\ \dots,\Theta\}$. Since there are $h \times w$ location nodes in $X$ and each location has $h$ outgoing edges to the locations in the next time slice. The time complexity of finding a trajectory is $O(w h^2 \Theta)$. As there are $|\mathbb{T}|$ trajectories for recovery, we need to run the above procedure $|\mathbb{T}|$ times. As a result, the time complexity of UP-approximation Trajectories algorithm is $O(|\mathbb{T}| w h^2 \Theta)$.

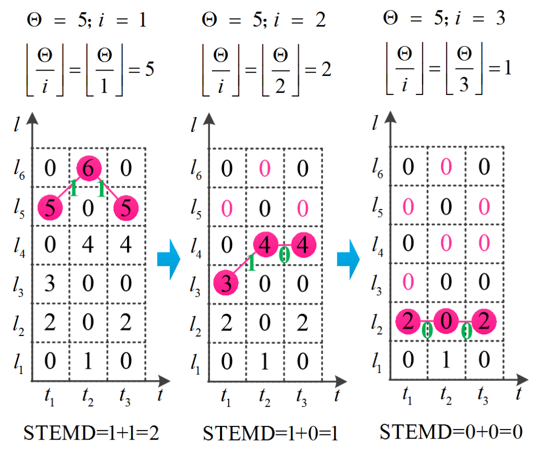

$\Theta = 5; i = 1$
$\lfloor \frac{\Theta}{i} \rfloor = \lfloor \frac{\Theta}{1} \rfloor = 5$
$\Theta = 5; i = 2$
$\lfloor \frac{\Theta}{i} \rfloor = \lfloor \frac{\Theta}{2} \rfloor = 2$
$\Theta = 5; i = 3$
$\lfloor \frac{\Theta}{i} \rfloor = \lfloor \frac{\Theta}{3} \rfloor = 1$
STEMD=1+1=2
STEMD=1+0=1
STEMD=0+0=0

Fig. 3. The process of finding $|\mathbb{T}| = 3$ trajectories based on UP approximation. Trajectory $T_1 = (5, t_1) \to (6, t_2) \to (5, t_3)$, $T_2 = (3, t_1) \to (4, t_2) \to (4, t_3)$, $T_3 = (2, t_1) \to (0, t_2) \to (2, t_3)$.

## 4.3 DOWN-approximation Trajectories (DOWN(X))

Given the location matrix trajectories $X = X_c$ obtained in each iteration of Algorithm 1 and the trajectory structured sparsity model $\mathbb{M}(\Theta, |\mathbb{T}|)$, DOWN-approximation Trajectories oracle returns a support $S$, namely the indices of most of the relevant trajectory locations in $X$, such that the norm of $||X - X_S||_2$ is approximately minimized, formally written as:

$$||X - X_S||_2 \leq C_{\mathrm{D}} \cdot \min_{S' \in \mathbb{M}(\Theta, |\mathbb{T}|)} ||X - X_{S'}||_2, \quad (15)$$

where $C_{\mathrm{D}} > 1$ is a constant.

In order to approach an optimal solution to the problem in Eq. (15), we reformulate the problem into *min-cost max-flow problem* which is a generation of the classical maximum flow problem [41]. Given the location matrix $X \in \mathbb{R}^{h \times w}$, we reconstruct $X$ as the graph, named as $\mathrm{Graph}_{\mathrm{X},\lambda}$, based on *min-cost max-flow problem*. In $\mathrm{Graph}_{\mathrm{X},\lambda}$, the numbers on edges indicate the edges costs; the edges costs are the location distances between the start and the end locations, multiplied by $\lambda$; the locations costs are the negative absolute values of the corresponding location information. An example of the graph is shown in Figure 4. The main elements of this graph are described as:

- Nodes: locations in $X = X_c$ got in each iteration of Algorithm 1.
- Edge: an edge from $\forall l_m$ at time $t$ to $\forall l_n$ at time $t+1$, where $m, n \in [h]$, $t \in [w-1]$.
- Costs: the cost of location $l_i$ is $-|l_i|^2$, where $i \in [h]$. The cost of an edge form $l_m$ at time $t$ to $l_n$ at time $t+1$ is $\lambda|m-n|$, where $\lambda > 0$.
- Capacity: the capacity of both edges and nodes is 1.

The main idea is to find the disjoint trajectories via the $\mathrm{Graph}_{\mathrm{X},\lambda}$, which correspond to the support of $X$. For any fixed $\lambda$, a solution of the *min-cost max-flow problem* on the graph includes a subset of the locations. These locations correspond to a support with exactly $|\mathbb{T}|$ locations per time slice that minimize the cost of $-||X_S||_2^2 + \lambda \cdot \mathrm{STEMD}(S)$, where $-||X_S||_2^2$ is the cost of the location nodes. Since the paths in $S$ do not intersect vertically, they represent a min-

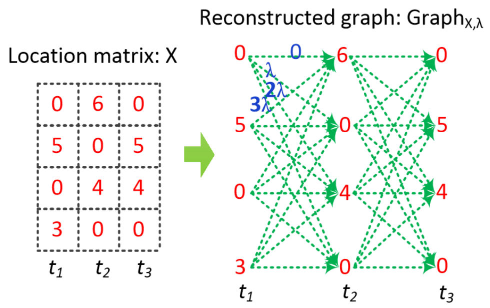


Fig. 4. A location matrix $X$ and the corresponding reconstructed graph: $\mathrm{Graph}_{\mathrm{X},\lambda}$, where $t = 3$, $|\mathbb{T}| = 2$. The numbers on edges indicate the edges costs. And the edges costs are the location distances between the start and the end locations, multiplied by $\lambda$. The locations costs are the negative absolute values of the corresponding location information. The capacity of both nodes and edges is 1.

cost matching for the locations in $S$. Hence, the cost of edges between columns of $X$ sums up to $\lambda \cdot \mathrm{STEMD}(S)$.

Note that $||X - X_S||_2^2 = ||X||_2^2 - ||X_S||_2^2$. Since $||X||_2^2$ does not depend on $S$, minimizing $||X - X_S||_2^2 + \lambda \cdot \mathrm{STEMD}(S)$ with respect to $S$ is equivalent to minimizing $-||X_S||_2^2 + \lambda \cdot \mathrm{STEMD}(S)$, which means the min-cost graph solves a Lagrangian relaxation of the original problem. Then, the next important question is how to select the parameter $\lambda$, which defines a trade-off between STEMD and approximation location error. For any fixed $\lambda$, the min-cost graph provides the optimal solution to the Lagrangian relaxation [42]. Moreover, we show that we can find such a good location support efficiently via a binary search over $\lambda$. And the details of DOWN-approximation Trajectories are presented in Algorithm 3. In this algorithm, $\varrho$ and $\delta$ are two constant parameters and we assume that $\mathrm{GRAPH}(\mathrm{Graph}_{\mathrm{X},\lambda})$ returns the support of locations corresponding to an integral min-cost graph in $\mathrm{Graph}_{\mathrm{X},\lambda}$. Theorem 3 presents the approximation results of the DOWN-approximation Trajectories algorithm, and Theorem 4 presents the time complexity of this algorithm.

**Algorithm 3** DOWN-approximation Trajectories

```
1:  **Input:** location matrix $X = X_c$ obtained in each iteration of Algorithm 1, $\Theta$, parameters $\varrho$ and $\delta$.
2:  **Output:** $S$.
3:  $l_{min} = \min_{l_i \in X} |l_i|^2$, $\lambda_0 = \frac{l_{min}}{2wh^2}$, $\eta = \frac{l_{min}}{whh}\varrho$;
4:  $S = \mathrm{GRAPH}(\mathrm{Graph}_{\mathrm{X},\lambda_0})$;
5:  **if** there is a $S \in \mathbb{M}(k, \mathbb{T})$ and $||X - X_S||_2 = 0$ **then**
6:      **return** $S$;
7:  $\lambda_{right} = 0$, $\lambda_{left} = ||X||_2^2$;
8:  **while** $\lambda_{left} - \lambda_{right} > \eta$ **do**
9:      $\lambda_m = (\lambda_{left} - \lambda_{right})/2$;
10:     $S = \mathrm{GRAPH}(\mathrm{Graph}_{\mathrm{X},\lambda_{\mathrm{m}}})$;
11:     **if** $\mathrm{STEMD}(S) \geq \Theta$ and $\mathrm{STEMD}(S) \leq \delta\Theta$ **then**
12:         **return** $S$;
13:     **if** $\mathrm{STEMD}(S) > \delta\Theta$ **then** $\lambda_{right} = \lambda_m$;
14:     **else** $\lambda_{left} = \lambda_m$;
15: **end while**
16: **Return** $S = \mathrm{GRAPH}(\mathrm{Graph}_{\mathrm{X},\lambda_{\mathrm{left}}})$.
```

**Theorem 3.** *Let $S$ be the location support got from Algorithm 3, $O$ be the approximation error of the best location support under* *the support of* STEMD *no more than* $\Theta$, $\delta > 1$, $\varrho > 0$, $S_{left} = \text{GRAPH}(\text{Graph}_{\text{X},\lambda_{\text{left}}})$, $S_{right} = \text{GRAPH}(\text{Graph}_{\text{X},\lambda_{\text{right}}})$, *Then the following two results can be got: (1)* $||X - X_S||_2^2 \le O$ *s.t.* $\Theta \le \text{STEMD}(S) \le \delta\Theta$*; (2)* $||X - X_S||_2^2 \le \frac{\delta+\delta\varrho-\varrho}{\delta-1}O$ *s.t.* $\text{STEMD}(S) \le \Theta$.

*Proof.* (1) If the $S$ is returned in line 6 of the Algorithm 3, the first result is satisfied obviously. When the $S$ is returned in line 12 of Algorithm 3, $\Theta \le \text{STEMD}(S) \le \delta\Theta$. Moreover, $||X - X_S||_2^2 + \lambda_{mid}\text{STEMD}(S) \le \min_{S' \in \mathbb{M}(\Theta,|\mathbb{T}|)} ||X - X_{S'}||_2^2 + \lambda_{mid}\text{STEMD}(S') \le O + \lambda_{mid}\Theta$. Then we can get $||X - X_S||_2^2 \le O$.

(2) When $S$ is returned in the last line of Algorithm 3, $\lambda_{left} - \lambda_{right} > \eta$. In the process of iterations, we keep two invariants: 1) $\text{STEMD}(S_{left}) \le \Theta$; 2) $\text{STEMD}(S_{right}) \ge \delta\Theta$. Based on the $\text{GRAPH}(\text{Graph}_{\text{X},\lambda})$, we can get:

$$\begin{aligned} ||X - X_{S_{right}}||_2^2 + \lambda_{right}\text{STEMD}(S_{right}) &\le O + \lambda_{right}\Theta \\ \lambda_{right}\delta\Theta &\le O + \lambda_{right}\Theta \\ \lambda_{right} &\le \frac{1}{\delta\Theta - \Theta}O. \end{aligned} \tag{16}$$

At the end of the iteration, we get $\lambda_{left} - \lambda_{right} < \eta$, which is equivalent to $\lambda_{left} - \lambda_{right} < \frac{l_{min}}{whh}\varrho$. Based on $\text{GRAPH}(\text{Graph}_{\text{X},\lambda})$, we can also get:

$$\begin{aligned} ||X - X_{S_{left}}||_2^2 + \lambda_{left}\text{STEMD}(S_{left}) &\le O + \lambda_{left}\Theta \\ ||X - X_{S_{left}}||_2^2 &\le O + \lambda_{left}\Theta \\ &\le O + (\lambda_{right} + \frac{l_{min}}{whh}\varrho)\Theta. \end{aligned} \tag{17}$$

Combining Eqs. (16) and (17), we can get:

$$\begin{aligned} ||X - X_{S_{left}}||_2^2 &\le \frac{\delta}{\delta-1}O + \frac{l_{min}}{whh}\varrho\Theta \\ &\le \frac{\delta}{\delta-1}O + \varrho l_{min} \\ &\le \frac{\delta+\delta\varrho-\varrho}{\delta-1}O. \end{aligned} \tag{18}$$

As a result, we prove the second result.

**Theorem 4.** *The time complexity of DOWN-approximation Trajectories is* $O(|\mathbb{T}|wh^2(log(wh/\varrho) + log(l_{max}/l_{min})))$*, where* $l_{max} = \max_{l_i \in X} |l_i|^2$, $l_{min} = \min_{l_i \in X} |l_i|^2$, $i \in [h]$, $\varrho > 0$.

*Proof.* Since the capacity of all edges and location nodes is 1, we focus on finding $|\mathbb{T}|$ trajectories to get the min-cost graph. Each trajectory in $X$ can be found in $O(wh^2)$ time [43]. Then, to find the best $\lambda$ , the upper bound of the total number of iterations is: $log\frac{||X||_2^2 - 0}{\epsilon} = log\frac{wh^2||X||_2^2}{\varrho l_{min}} \le log\frac{w^2h^3 l_{max}}{\varrho l_{min}} \le log\frac{w^3h^3}{\varrho} + log\frac{l_{max}}{l_{min}} \le 3log\frac{wh}{\varrho} + 3log\frac{l_{max}}{l_{min}}$. In summary, the total runtime of the DOWN-approximation Trajectories algorithm is $O(|\mathbb{T}|wh^2(log(wh/\varrho) + log(l_{max}/l_{min})))$.

## 4.4 Theoretical Analysis of iTracker

This section analyzes the convergence property, the recovered trajectory signal estimates for various number of released trajectories, and the time complexity of iTracker.

**Theorem 5** (Convergence of iTracker). *Let* $\Delta_i = X - X_i$ *for various number of released trajectories, where* $X_i$ *is the result estimated by iTracker in iteration* $i$*, then,*

$$||\Delta_{i+1}||_2 \le \omega||\Delta_i||_2 + \xi||\tau||_2, \tag{19}$$

*where* $\omega = (1 + C_{\text{D}})[\rho + (1 - (C_{\text{U}}(1-\rho) - \rho)^2)^{1/2}]$, $\xi = (1 + C_{\text{D}})[\frac{(1+C_{\text{U}})(1+\rho)^{1/2}}{C_{\text{U}}(1-\rho)-\rho} + \frac{(1+C_{\text{U}})(1+\rho)^{1/2}(C_{\text{U}}(1-\rho)-\rho)}{(1-(C_{\text{U}}(1-\rho)-\rho)^2)^{1/2}} + (1+\rho)^{1/2}]$.

*Proof.* From the iTracker, $X_c = X_i + \text{UP}(X_b)$. Based on the triangle inequality, for various number of released protected trajectories, $||\Delta_{i+1}||_2$ can be upper bounded as:

$$||\Delta_{i+1}||_2 \le (1 + C_{\text{D}})||\Delta_i - \text{UP}(\mathfrak{M}^T\mathfrak{M}\Delta_i + \mathfrak{M}^T\tau)||_2. \tag{20}$$

Furthermore, let $\Psi = \text{supp}(\text{UP}(X_b))$, we can get the upper bound: $||\Delta_{i,\Psi^c}||_2 \le (1 - (C_{\text{U}}(1-\rho) - \rho)^2)^{1/2}||\Delta_i||_2 + [\frac{(1+C_{\text{U}})(1+\rho)^{1/2}}{C_{\text{U}}(1-\rho)-\rho} + \frac{(1+C_{\text{U}})(1+\rho)^{1/2}(C_{\text{U}}(1-\rho)-\rho)}{(1-(C_{\text{U}}(1-\rho)-\rho)^2)^{1/2}}]||\tau||_2$ ($\circledast$), where $\Delta_{i,\Psi^c}$ is the set of coordinates in the complement of $\Psi$ in $\Delta_i$. Then based on RIP and ($\circledast$), $||\Delta_i - \text{UP}(\mathfrak{M}^T\mathfrak{M}\Delta_i + \mathfrak{M}^T\tau)||_2$ can be upper bounded as:

$$\begin{aligned} &||\Delta_i - \text{UP}(\mathfrak{M}^T\mathfrak{M}\Delta_i + \mathfrak{M}^T\tau)||_2 \\ &\le [(1 - (C_{\text{U}}(1-\rho) - \rho)^2)^{1/2} + \rho]||\Delta_i||_2 \\ &\quad + [\frac{(1+C_{\text{U}})(1+\rho)^{1/2}(C_{\text{U}}(1-\rho)-\rho)}{(1-(C_{\text{U}}(1-\rho)-\rho)^2)^{1/2}}]||\tau||_2 \\ &\quad + [\frac{(1+C_{\text{U}})(1+\rho)^{1/2}}{C_{\text{U}}(1-\rho)-\rho}]||\tau||_2 \\ &\quad + [(1+\rho)^{1/2}]||\tau||_2. \end{aligned} \tag{21}$$

Combining the above, we can get the (19) [39].

In (19), the second item of the right side of this inequality can be ignored when there is no noise, and then we can only focus on the factor $\omega$. In order to achieve the convergence , $\omega$ needs to be smaller than 1. Namely:

$$\omega = (1 + C_{\text{D}})[\rho + (1 - (C_{\text{U}}(1-\rho) - \rho)^2)^{1/2}] < 1. \tag{22}$$

Note that we can make $\rho$ as small as we desire since this assumption only affects the measurement bound by a constant factor. Therefore, for guaranteed convergence of various number of released trajectories, the condition $C_{\text{U}}^2 > \frac{C_{\text{D}}^2 + 2C_{\text{D}}}{(1+C_{\text{D}})^2}$ must hold.

iTracker uses two approximation algorithms (UP-approximation Trajectories and DOWN-approximation Trajectories) to bounce between upper bound and lower bound and to reduce the noises in each iteration to find the best recovered multiple trajectories simultaneously from the released data protected by differential privacy. Finally, iTracker converges for different number of trajectories as discussed in Theorem 5. Furthermore, the convergence of iTracker implies that this approach quickly recovers estimated trajectories for various number of released trajectories. Formally, we present the Theorem 6.

**Theorem 6.** *Let* $X$ *be a true optimum, iTracker returns a signal estimate* $\hat{X}$ *such that* $supp(\hat{X}) \in \mathbb{M}(\Theta, |\mathbb{T}|)$ *and*

$$||X - \hat{X}||_2 \le (\frac{1 - \omega + \xi}{1 - \omega})||\tau||_2 \tag{23}$$

*after* $i = \lceil log(||X||_2/||\tau||_2)/log(1/\omega) \rceil$ *iterations. Moreover, the runtime of iTracker can be written as:*

$$O((\mathfrak{T} + |\mathbb{T}|wN(\Theta + log\frac{N}{\varrho} + log\frac{l_{max}}{l_{min}}))log\frac{||X||_2}{||\tau||_2}), \tag{24}$$

*where* $\mathfrak{T}$ *is the time complexity of one execution of the subproblem in Line 5 and Line 6 in Algorithm 1,* $N = w * h$ *is the number of* *elements in $X$.*

*Proof.* (1) Based on the Theorem 5 and the simple inductive argument which are applicable to various number of released trajectories, we can get that:

$$||X - X_{i+1}||_2 \leq \omega^i ||x||_2 + \xi ||\tau||_2 \sum_{j=0}^{i} \omega^j. \tag{25}$$

After $i = \lceil log(||X||_2/||\tau||_2)/log(1/\omega) \rceil$ iterations, $\omega^i ||X||_2 \leq ||\tau||_2$ can be derived. Further, the bound of $\sum_{j=0}^{t} \omega^j$ can be bounded by $1/(1-\omega)$. Combining the above, we can get (23).

(2) The time complexities of the UP and DOWN approximations are $O(|\mathbb{T}|wh^2\Theta)$ and $O(|\mathbb{T}|wh^2(log(wh/\varrho) + log(l_{max}/l_{min})))$, respectively. The time complexity of one iteration in Algorithm is $O(\mathfrak{T} + |\mathbb{T}|wN(\Theta + log\frac{N}{\varrho} + log\frac{l_{max}}{l_{min}}))$, and the total number of the iterations is $\lceil log(||X||_2/||\tau||_2)/log(1/\omega) \rceil$. The overall time complexity follows.

## 5 EXPERIMENTS

This section evaluates the effectiveness and efficiency of the proposed approach based on comprehensive experiments using two real-world datasets.

### 5.1 Experimental Setup

**Datasets:** In experiments, two real data sources (Geolife and Gowalla) are used, the details of which are described below.

**Geolife:** Geolife is collected from 182 users from April 2007 to August 2012. This dataset is represented by a series of tuples including latitude, longitude and time, and records a wide range of users' movements. 17,621 trajectories with the total duration of 50,176 hours and the total distance of 1,292,951 kilometers are contained in this dataset. 91.5 percents of the trajectories are logged in a relatively dense representation, such as every $5 \sim 10$ meters or every $1 \sim 5$ seconds per location. In experiments, we randomly select 100 original trajectories from this dataset, each of which contains at least 100 sampling points, to train our method and baseline methods. Moreover, these selected original trajectories are used as the ground truth to evaluate all the comparison methods in the experiments.

**Gowalla:** Gowalla is a location-based social networking website. In this website, users share their locations by checking-in. There are 6,442,890 checking-in locations of 196,586 users over the period of February 2009 to October 2010 in this dataset. We select all the records in Los Angeles as the ground truth to train our approach and the competitive approaches, with the map partitioned into the cells of $0.9 \times 0.9$ km$^2$. Moreover, check-ins are logged in a relatively low frequency, such as every $1 \sim 50$ minutes. The difference in Gowalla dataset and Geolife dataset allows for a fair evaluation of our approach.

**Protection Method:** In the experiments, differential privacy based on Laplace mechanism is used to perturb locations and protect the real trajectories. The desired differential privacy parameter $\epsilon$ is set as $0.1, 0.2, ..., 0.9$ respectively. For different trajectories, we can have different privacy-preservation setting based on different $\epsilon$. In this work, we use the same privacy-preservation setting as a case study for ease of comparison among different methods.

**iTracker:** In this work, an efficient multiple trajectory recovery framework, named as iTracker, is proposed based on the trajectory structured sparsity model. Specifically, in our proposed method, $\mathfrak{M}$ is an i.i.d. Gaussian matrix. We set $\Theta = 10,000$ by default. Moreover, we note that, to obtain the best performance of our proposed method iTracker, we try a set of different $\Theta$ values ($\Theta$ = 1,000, 2,000, 3,000,..., 20,000) and return the best.

**Comparison Methods:** We compare iTracker with three baseline methods. Since there is no method for recovering multiple trajectories simultaneously, we propose to combine the traditional classification methods and single trajectory prediction methods as the baseline methods to implement multiple trajectories recovery.

The three baseline methods have two main steps. First, *K* Nearest Neighbor (KNN), Support Vector Machine (SVM) and K-means are selected as the classification methods to assign the locations into different trajectories. Then, we infer the trajectories based on the results of classifications. We compare against Markov [44], PutMode (*P*rediction of *u*ncertain *t*rajectories in *M*oving *o*bjects *d*atabas*e*s) [45] and a recently proposed method, AT [24]. Specifically, the order-*k* Markov approach supposes that the locations can be predicted based on the context, namely, the sequence of the *k* most recent records in the location history. Moreover, the underlying Markov model represents the states as the contexts, and the transitions are used to represent the possible locations which follow the context [44]. PutMode is a framework proposed for predicating uncertain trajectories based on Continuous Time Bayesian Networks (CTBNs) in databases of moving objects [45]. AT provides the adversary with both the locations where users conduct activities and the information when users stay at each of these locations [24]. The attack is formulated as an optimization problem based on Bayesian theorem. We strictly follow the strategies recommended by the authors in their papers to estimate the model parameters. We use 10-fold cross validation to identify the best combination of all the related parameters.

**Performance Metrics:** In the experiments, trajectory similarity measures (Euclidean distance and Hausdorff distance) and classification performance measure (F-measure) are used to evaluate the performance, where F-measure provides an integrated result of precision and recall.

(1) **Trajectory Similarity Measures:** Two measures are selected to compare the performance of our method and the baseline methods.

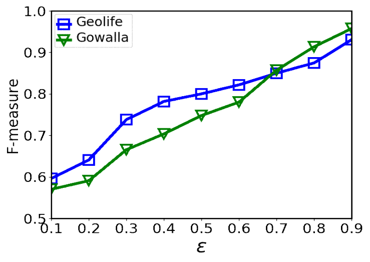
**Euclidean distance**, also known as $\mathcal{L}_2$-norm, is a distance measure in literature for measuring trajectory similarity. Given two trajectories $T1$, $T2$, the Euclidean distance $d(T_1, T_2)$ can be calculated as:

$$d(T_1, T_2) = \frac{1}{n} \sum_{i=1}^{n} d(t_{1,i} - t_{2,i}), \tag{26}$$

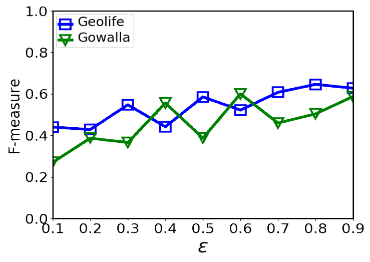
where $d(t_{1,i} - t_{2,i})$ is the distance on spatial space.

**Hausdorff distance**. It is widely used to express the spatial similarity between two trajectories. For two trajectories $T_1$ and $T_2$, the Hausdorff distance, $D(T_1, T_2)$, is defined as:

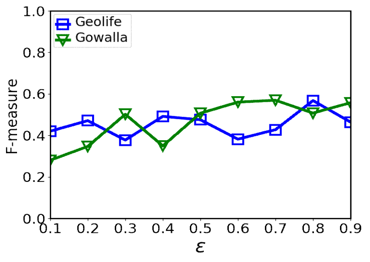
$$D(T_1, T_2) = \max\{d(T_1, T_2), d(T_2, T_1)\}, \tag{27}$$



(a) The average F-measure of KNN.



(b) The average F-measure of SVM.



(c) The average F-measure of K-means.

Fig. 5. The comparison of the F-measure among KNN, SVM and K-means approaches based on the Geolife dataset and Gowalla dataset.

where

$$d(T_1, T_2) = \max_{l_{T_1} \in T_1} \min_{l_{T_2} \in T_2} ||l_{T_1} - l_{T_2}||. \tag{28}$$

(2) **Classification Performance Measure:** F-measure, based on the precision and recall, is used to measure the results of classification. We define $\partial(X)$ as the set of the locations in $X$, $X_0$ as the set of locations of true trajectories and $X_1$ as the set of locations of detected trajectories by a specific method. Then the precision ($\mathbf{p}$), recall ($\mathbf{r}$) and F-measure can be defined as:

$$\mathbf{p} = \frac{|\partial(X_1) \bigcap \partial(X_0))|}{|\partial(X_1)|}, \ \mathbf{r} = \frac{|\partial(X_1) \bigcap \partial(X_0))|}{|\partial(X_0)|}, \\ \mathrm{F-measure} = \frac{2 \times \mathbf{p} \times \mathbf{r}}{\mathbf{p} + \mathbf{r}}. \tag{29}$$

## 5.2 Experimental Results and Analyses

**Comparison of Recovery Accuracy:** Using Geolife and Gowalla datasets, we run all competition approaches under different protective degrees respectively. The privacy protection based on Laplace mechanism perturbs the locations of original trajectories. The $\epsilon$, as a privacy budget, controls the protection intensity. The smaller its value is, the stronger the privacy guarantee is. The results are shown in Figure 5-10.

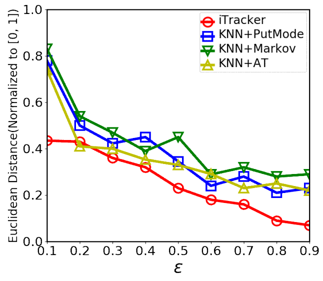
(1) **F-measure:** The comparison of the F-measure based on different methods is shown in Figure 5. In Figure 5(a), we can see that the results of KNN classification become better with both Geolife dataset and Gowalla dataset when $\epsilon$ increases from 0.1 to 0.9. Specifically, when $\epsilon$ equals to 0.7, 0.8 and 0.9 respectively, there is a better classification with Gowalla dataset. However, with the $\epsilon$ decreasing from 0.7 to 0.1, the classification results become worse. The main reason for this is that the strength of protection become stronger with smaller $\epsilon$. The locations of all trajectories largely deviates from their original positions and are intricately mixed together. Moreover, when $\epsilon$ increases from 0.1 to 0.9, the F-measure of SVM slowly increase with fluctuations as shown in Figure 5(b) and there are also fluctuations of the F-measure of K-means as shown in Figure 5(c). As shown in Figure 5, the KNN performs the best classification results among these three methods, especially, when the $\epsilon$ grows from 0.6 to 0.9.

Then, based on these results of classification, we train the traditional Markov method, PutMode method and AT method for each trajectory recovery. Moreover, we also train our proposed method on the Geolife and Gowalla datasets. The average Euclidean distance and the average Hausdorff distance of all trajectories, *normalized to* $[0, 1]$, are presented in Figure 6, 7, 8 and 9 respectively.

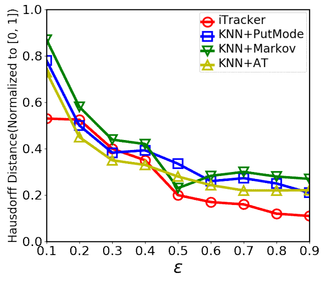
(2) **Euclidean Distance:** Using both Geolife dataset and Gowalla dataset, we can see that even when the results of all methods become worse with increasing protection gradually, our proposed approach retains the best performance under different $\epsilon$. Specifically, based on KNN classification method, there is a better classification result that F-measure is nearly 0.9 when the $\epsilon$ is set from 0.7 to 0.9 as shown in Figure 5. At this time, the classification exerts little impacts on the recovery since almost all locations are correctly assigned to trajectories that they belong to. Moreover, all the average Euclidean distance of the baseline methods with KNN are smaller than the baseline methods with other classification methods. Nevertheless, our proposed method yields the best result since it can capture the interdependency of the locations and adaptively group trajectories. With the decrease of the $\epsilon$, the results of classification become worse, which further reduces the accuracy of the recovery. Moreover, we can see that the classification results of SVM and K-means are not as good as the results of KNN approach, and the average Euclidean distance based on SVM method or K-means method are larger than the distance based on KNN classification method respectively. Moreover, the results show that $\epsilon$ exerts a great impact on the accuracy of recovery in both Geolife dataset and Gowalla dataset.

(3) **Hausdorff Distance:** The results of Hausdorff distance are shown in Figure 6(b), Figure 6(d), Figure 7(b), Figure 7(d), Figure 8(b) and Figure 8(d). The $\epsilon$ shows a greater impact on the results of Hausdorff distance in both Geolife dataset and Gowalla dataset. Specifically, with the increase of the $\epsilon$, the Hausdorff distance based on KNN classification method sharply decreases in Gowalla dataset. Although all baseline methods present the similar results when $\epsilon$ equals to 0.5, 0.6, 0.7, 0.8 and 0.9 as shown in both Figure 6(b) and Figure 6(d), our proposed method obtains shorter Hausdorff distance and shows the best performance in Gowalla dataset. Moreover, as shown in Figure 7 and Figure 8, the average Hausdorff distance of baseline methods based on the results of SVM or K-means seems to be not as good as the Hausdorff distance of the methods based on the results of KNN approach. To sum up, iTracker gets the shorter distance than all baseline methods and obtains the best recovery effect among all comparison approaches, due to the fact that iTracker can effectively capture the interdependency of the locations to accurately group trajectories.

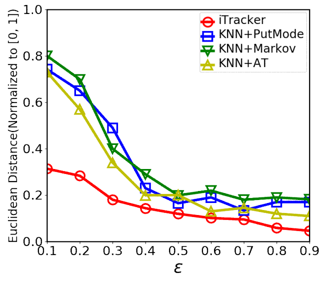
(4) **Sensitivity to the Parameter $\Theta$:** We set $\Theta = 10^4$ by



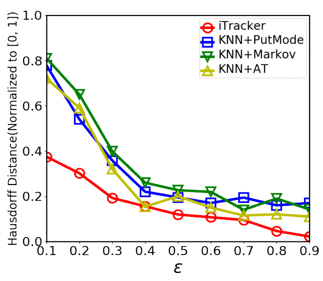
(a) Euclidean distance vs $\epsilon$ based on Geolife dataset.



(b) Hausdorff distance vs $\epsilon$ based on Geolife dataset.

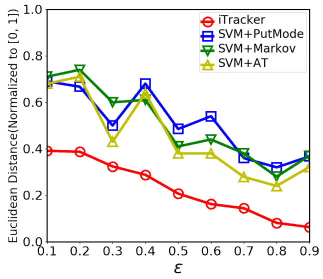


(c) Euclidean distance vs $\epsilon$ based on Gowalla dataset.

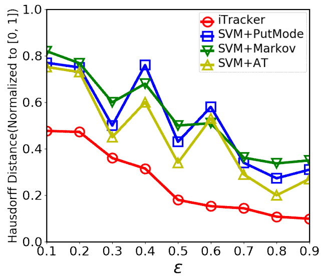


(d) Hausdorff distance vs $\epsilon$ based on Gowalla dataset.

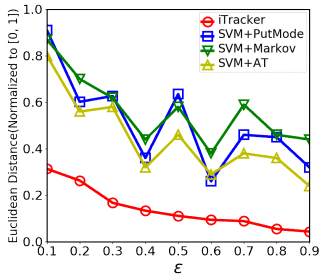
Fig. 6. The comparison between iTracker and baseline methods with KNN under different $\epsilon$ based on the Geolife dataset and Gowalla dataset.



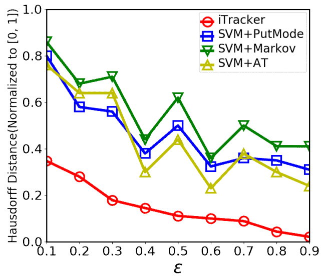
(a) Euclidean distance vs $\epsilon$ based on Geolife dataset.



(b) Hausdorff distance vs $\epsilon$ based on Geolife dataset.

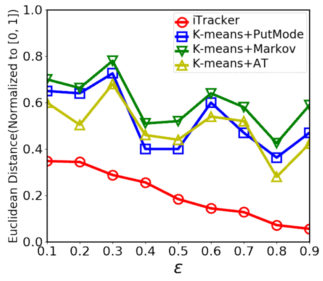


(c) Euclidean distance vs $\epsilon$ based on Gowalla dataset.

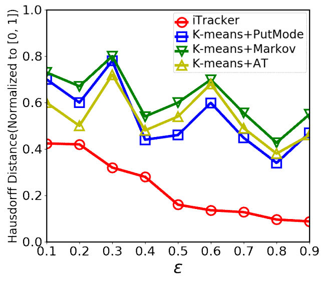


(d) Hausdorff distance vs $\epsilon$ based on Gowalla dataset.

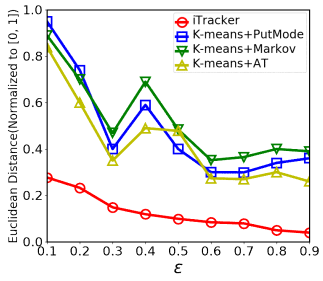
Fig. 7. The comparison between iTracker and baseline methods with SVM under different $\epsilon$ based on the Geolife dataset and Gowalla dataset.



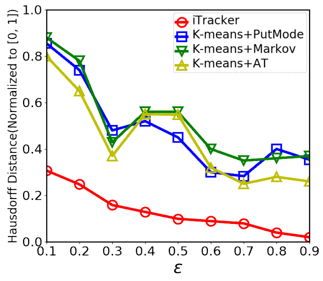
(a) Euclidean distance vs $\epsilon$ based on Geolife dataset.



(b) Hausdorff distance vs $\epsilon$ based on Geolife dataset.



(c) Euclidean distance vs $\epsilon$ based on Gowalla dataset.

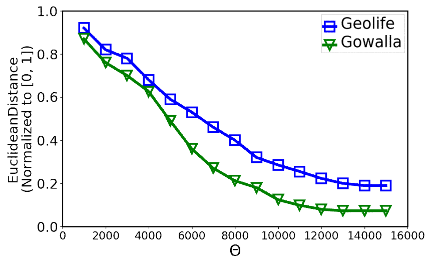


(d) Hausdorff distance vs $\epsilon$ based on Gowalla dataset.

Fig. 8. The comparison between iTracker and baseline methods with K-means under different $\epsilon$ based on the Geolife dataset and Gowalla dataset.

default. Moreover, we note that, to obtain the best performance of our proposed method iTracker and evaluate the sensitivity to the parameter $\Theta$, we try a set of different $\Theta$ values ($\Theta$ = 1,000, 2,000, 3,000,..., 20,000). The sensitivity of Euclidean distance and Hausdorff distance to the parameter $\Theta$ based on Geolife dataset and Gowalla dataset is shown in Figure 9. We can see that both the Euclidean distance and the Hausdorff distance decrease with the increase of the parameter $\Theta$. When the parameter $\Theta$ reaches a certain value, the best value is obtained. After that, with the increase of the parameter $\Theta$, the results will not change. Specifically, when the parameter $\Theta$ is close to 13,000, the best distance results are achieved based on the Geolife dataset. However, based on the Gowalla dataset, the best distance results can be achieved when $\Theta$ is close to 12,000. Moreover, the Gowalla dataset is more sensitive to parameter $\Theta$ than Geolife dataset. Overall, we achieve better distance results using Gowalla dataset compared with the results using Geolife dataset.

(5) **Comparison of the Time Cost:** Figure 10 shows the time costs of iTracker and all competitive baseline methods based on KNN, SVM and K-means on the two benchmark datasets (Geolife dataset and Gowalla dataset). The results indicate that our method is faster than PutMode method which is based on KNN, SVM or K-means classification method in both Geolife dataset and Gowalla dataset. Furthermore, iTracker runs faster than any other methods based on K-means approach. The main reason is that iTracker eliminates the dependency on classification methods, simplifies the recovery process, and retains competitive time complexity. For instance, PutMode and classification methods based baseline approach include steps such as location classification, continuous time Bayesian networks construction, trajectories clear up, and trajectories step prediction.


(a) The Euclidean distance of iTracker under different Θ.

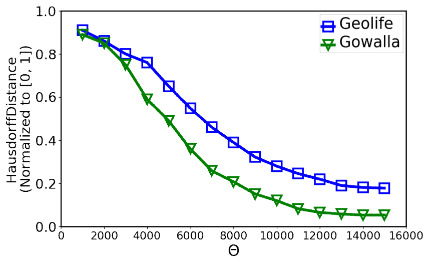
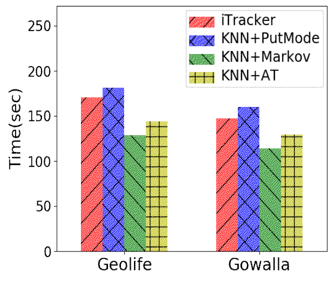
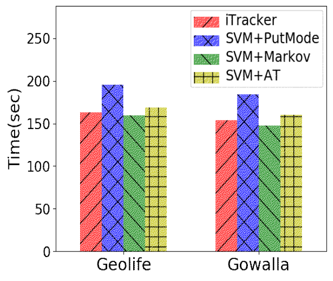

(b) The of Hausdorff distance of iTracker under different Θ.

Fig. 9. The comparison of Euclidean distance and Hausdorff distance of iTracker under different Θ based on the Geolife and Gowalla datasets.


(a) The runtime of iTracker and baseline methods with KNN approach.


(b) The runtime of iTracker and baseline methods with SVM approach.

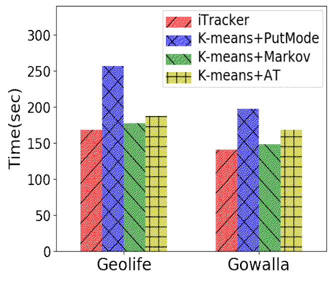
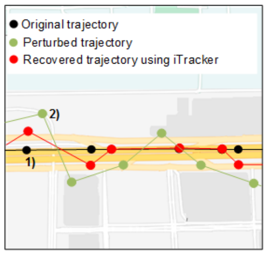
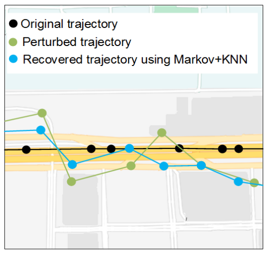

(c) The runtime of iTracker and baseline methods with K-means approach.

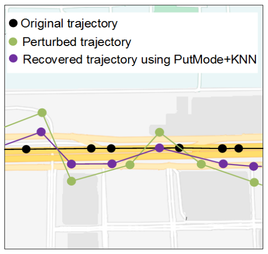
Fig. 10. The comparison of runtime between iTracker and baseline methods with different classfication methods (KNN, SVM and K-means) based on the Geolife dataset and Gowalla dataset.



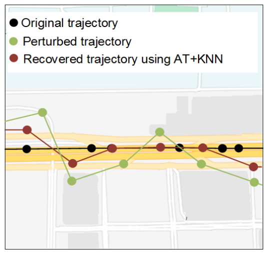
(a) Trajectory recovery based on iTracker.



(b) Trajectory recovery based on Markov and KNN.



(c) Trajectory recovery based on PutMode and KNN.



(d) Trajectory recovery based on AT and KNN.

Fig. 11. An illustration of a randomly selected original trajectory and the recovered trajectories based on iTracker and baseline methods using Geolife dataset.

On the other hand, iTracker can directly predict multiple trajectories simultaneously based on our proposed structured sparsity model. Although the runtime of our proposed method is sometimes longer than Markov method, our approach achieves much higher recovery accuracy within acceptable runtime. As a result, iTracker presents the best overall performance compared with the existing methods considering the efficiency and accuracy based on both Geolife dataset and Gowalla dataset.

## 5.3 Case Study: Trajectory Recovery

An illustration of the original trajectory, perturbed trajectory and recovered trajectories based on the competitive methods, including iTracker, Markov+KNN, PutMode+KNN and AT+KNN, in Geolife dataset are shown in Figure 11. Here, KNN is selected as the classification method due to its performance edge against SVM and K-means. We randomly select a section of a trajectory from original Geolife dataset. The original trajectory and perturbed trajectory have been shown in each subfigure of Figure 11. As shown in these subfigures, the locations of the original trajectory are perturbed and the original trajectory is changed. For instance, the location 1) in the original trajectory is replaced by the location 2) in the perturbed trajectory shown as Figure 11(a). The recovered trajectories based on different recovery methods have been presented in different subfigures of Figure 11. We can see that the trajectory recovered by iTracker resembles the original trajectory more than the trajectories recovered by all baseline methods. For example, the locations of recovered trajectory based on Morkov+KNN largely deviate from the original trajectory locations as shown in Figure 11. The main reason for the underperformance is due to the misclassification of some locations. Even worse, the Markov transition matrix used to recover trajectories is constructed by wrong locations with the unavoidable classification error and the perturbed/noisy locations without any consideration of suppressing noise and alleviating errors.

## 6 Conclusions and Future Work

This paper presents a completely new angle of attacking a standard location protection mechanism based on differential privacy, and implements multiple trajectories recovery simultaneously from perturbed locations. iTracker, an efficient framework, is designed to build the trajectory structured sparsity model and execute the model projection oracles to find the best approximation for the multiple trajectories. Furthermore, the convergence and accuracy of the proposed approach are theoretically analyzed and experimentally evaluated. In future, we will consider the overlapped trajectories and focus on the design of trajectory protection methods to protect against iTracker.

## References

[1] M.-P. Pelletier, M. Trépanier, and C. Morency, "Smart card data use in public transit: A literature review," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 4, pp. 557–568, 2011.

[2] R. Clarke, "Person location and person tracking-technologies, risks and policy implications," *Information Technology and People*, vol. 14, no. 2, pp. 206–231, 2001.

[3] B. Lee, J. Oh, H. Yu, and J. Kim, "Protecting location privacy using location semantics," in *SIGKDD*, pp. 1289–1297, ACM, 2011.

[4] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *2011 IEEE Symposium on Security and Privacy*, pp. 247–262, IEEE, 2011.

[5] Y. Wang, D. Xu, X. He, C. Zhang, F. Li, and B. Xu, "L2p2: Location-aware location privacy protection for location-based services," in *INFOCOM*, pp. 1996–2004, IEEE, 2012.

[6] Y.-S. Chen, T.-T. Lo, C.-H. Lee, and A.-C. Pang, "Efficient pseudonym changing schemes for location privacy protection in vanets," in *ICCVE*, pp. 937–938, IEEE, 2013.

[7] V. A. Kachore, J. Lakshmi, and S. Nandy, "Location obfuscation for location data privacy," in *2015 IEEE World Congress on Services*, pp. 213–220, IEEE, 2015.

[8] Y. Cao and M. Yoshikawa, "Differentially private real-time data release over infinite trajectory streams," in *MDM*, pp. 68–73, 2015.

[9] C. Yin, J. Xi, R. Sun, and J. Wang, "Location privacy protection based on differential privacy strategy for big data in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3628–3636, 2017.

[10] J. Hua, Y. Gao, and S. Zhong, "Differentially private publication of general time-serial trajectory data," in *INFOCOM*, pp. 549–557, IEEE, 2015.

[11] W. Qardaji, W. Yang, and N. Li, "Differentially private grids for geospatial data," in *ICDE*, pp. 757–768, IEEE, 2013.

[12] Y. Xiao and L. Xiong, "Protecting locations with differential privacy under temporal correlations," in *Proceedings of the 2015 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1298–1309, ACM, 2015.

[13] K. Gu, L. Yang, and B. Yin, "Location data record privacy protection based on differential privacy mechanism," *Information Technology And Control*, vol. 47, no. 4, pp. 639–654, 2018.

[14] P. Xiong, T. Zhu, L. Pan, W. Niu, and G. Li, "Privacy preserving in location data release: A differential privacy approach," in *Pacific Rim International Conference on Artificial Intelligence*, pp. 183–195, Springer, 2014.

[15] M. E. Gursoy, L. Liu, S. Truex, and L. Yu, "Differentially private and utility preserving publication of trajectory data," *IEEE Transactions on Mobile Computing*, 2018.

[16] R. Chen, B. Fung, and B. C. Desai, "Differentially private trajectory data publication," *arXiv preprint arXiv:1112.2020*, 2011.

[17] A. Mannini and A. M. Sabatini, "Accelerometry-based classification of human activities using markov modeling," *Computational Intelligence and Neuroscience*, vol. 2011, p. 4, 2011.

[18] E. Kim, S. Helal, and D. Cook, "Human activity recognition and pattern discovery," *IEEE Pervasive Computing*, vol. 9, no. 1, p. 48, 2010.

[19] M. Götz, S. Nath, and J. Gehrke, "Maskit: Privately releasing user context streams for personalized mobile applications," in *SIGMOD*, pp. 289–300, ACM, 2012.

[20] L. Liao, D. J. Patterson, D. Fox, and H. Kautz, "Learning and inferring transportation routines," *Artificial Intelligence*, vol. 171, no. 5-6, pp. 311–331, 2007.

[21] S. Gambs, M.-O. Killijian, and M. N. del Prado Cortez, "Next place prediction using mobility markov chains," in *Proceedings of the 1st Workshop on Measurement, Privacy and Mobility*, pp. 1–6, ACM, 2012.

[22] D. L. Vail, M. M. Veloso, and J. D. Lafferty, "Conditional random fields for activity recognition," in *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, p. 235, ACM, 2007.

[23] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[24] X. Chen, A. Mizera, and J. Pang, "Activity tracking: A new attack on location privacy," in *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 22–30, IEEE, 2015.

[25] N. Nguyen and Y. Guo, "Comparisons of sequence labeling algorithms and extensions," in *ICML*, pp. 681–688, ACM, 2007.

[26] X. Jin, R. Zhang, Y. Chen, T. Li, and Y. Zhang, "Dpsense: Differentially private crowdsourced spectrum sensing," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 296–307, ACM, 2016.

[27] X. Jin and Y. Zhang, "Privacy-preserving crowdsourced spectrum sensing," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 3, pp. 1236–1249, 2018.

[28] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "Wherenext: a location predictor on trajectory pattern mining," in *SIGKDD*, pp. 637–646, ACM, 2009.

[29] K. Farrahi and D. Gatica-Perez, "Discovering routines from large-scale human locations using probabilistic topic models," *TIST*, vol. 2, no. 1, p. 3, 2011.

[30] D. E. Riedel, S. Venkatesh, and W. Liu, "Recognising online spatial activities using a bioinformatics inspired sequence alignment approach," *Pattern Recognition*, vol. 41, no. 11, pp. 3481–3492, 2008.

[31] O. Ossama and H. M. Mokhtar, "Similarity search in moving object trajectories," in *Proceedings of the 15th International Conference on Management of Data*, pp. 1–6, 2009.

[32] M. Li, L. Zhu, Z. Zhang, and R. Xu, "Achieving differential privacy of trajectory data publishing in participatory sensing," *Information Sciences*, vol. 400, pp. 1–13, 2017.

[33] M. Srivatsa and M. Hicks, "Deanonymizing mobility traces: Using social network as a side-channel," in *CCS*, pp. 628–637, ACM, 2012.

[34] E. Elsalamouny and S. Gambs, "Differential privacy models for location- based services," *Transactions on Data Privacy*, vol. 9, no. 1, pp. 15–48, 2016.

[35] R. Dewri, "Local differential perturbations: Location privacy under approximate knowledge attackers," *IEEE Transactions on Mobile Computing*, vol. 12, no. 12, pp. 2360–2372, 2013.

[36] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal geo-indistinguishable mechanisms for location privacy," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 251–262, ACM, 2014.

[37] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, pp. 901–914, ACM, 2013.

[38] E. Levina and P. Bickel, "The earth mover's distance is the mallows distance: Some insights from statistics," in *ICCV*, vol. 2, pp. 251–256, IEEE, 2001.

[39] C. Hegde, P. Indyk, and L. Schmidt, "Approximation algorithms for model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 61, no. 9, pp. 5129–5147, 2015.

[40] R. G. Baraniuk, V. Cevher, M. F. Duarte, and C. Hegde, "Model-based compressive sensing," *IEEE Transactions on Information Theory*, vol. 56, no. 4, pp. 1982–2001, 2010.

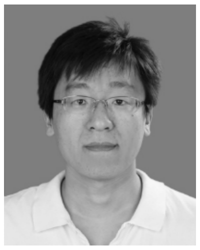
[41] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows: theory, algorithms, and applications," *Journal of the Operational Research Society*, vol. 45, no. 11, pp. 791–796, 1993.

[42] A. Kyrillidis and V. Cevher, "Sublinear time, approximate model-based sparse recovery for all," *arXiv preprint arXiv:1203.4746*, 2012.

[43] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, vol. 6. MIT press Cambridge, 2001.

[44] L. Song, D. Kotz, R. Jain, and X. He, "Evaluating location predictors with extensive wi-fi mobility data," in *INFOCOM*, vol. 2, pp. 1414–1424, IEEE, 2004.

[45] S. Qiao, C. Tang, H. Jin, T. Long, S. Dai, Y. Ku, and M. Chau, "Putmode: prediction of uncertain trajectories in moving objects databases," *Applied Intelligence*, vol. 33, no. 3, pp. 370–386, 2010.

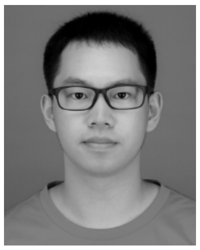


**Minglai Shao** is a PhD candidate in Beijing Advanced Innovation Center for Big Data and Brain Computing, School of Computer Science and Engineering, Beihang University, Beijing, China. He was a visiting scholar in the State University of New York at Albany in 2018. He received the M.S. degree from Guangxi University in 2015. His research interests include trajectory privacy, anomaly detection, botnet detection, graph mining, event detection and forecasting and machine learning.

**Jianxin Li** is a professor in Beijing Advanced Innovation Center for Big Data and Brain Computing, School of Computer Science and Engineering, Beihang University, Beijing, China. He received the PhD degree from Beihang University in 2008. He was a visiting scholar in the Machine Learning Department of CMU in 2015, and a visiting researcher of MSRA in 2011. His current research interests include big data, anomaly detection, machine learning.

**Qiben Yan** is an Assistant Professor in Department of Computer Science and Engineering of Michigan State University. He received his Ph.D. in Computer Science department from Virginia Tech, an M.S. and a B.S. degree in Electronic Engineering from Fudan University in Shanghai, China. His current research interests include wireless communication, wireless network security and privacy, mobile and IoT security, and big data privacy.

**Feng Chen** received the PhD degree in computer science from Virginia Tech, Blacksburg, Virginia, in 2012. He is an associate professor in Department of Computer Science, University of Texas at Dallas, USA. He is a recipient of the 2018 NSF CAREER award. His research interests include anomalous pattern detection, event detection and forecasting, graph mining, and machine learning.



**Hongyi Huang** is currently a Ph.D. student in Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China. He received his B.S. degree in Computer Science from Beihang University, Beijing, China. His current research topics cover network function virtualization and data-driven network management.

**Xunxun Chen** received the PhD degree in School of Computer Science, Harbin Institute of Technology, Harbin, China. He is a professor in CNCERT/CC, Beijing, China. His current research interests include anomaly pattern detection, intelligent transportation, event detection and forecasting, machine learning and data mining.