RoadAdaptor: An Adaptive Obfuscation Strategy for Vehicle Trajectory Privacy Against Spatial Correlation Aware Attacks

Chenxi Qiu
Computer Science Department
Rowan University
Glassboro, USA
qiu@rowan.edu

Li Yan
Senseable City Lab
Massachusetts Institute of Technology
Cambridge, USA
liyan_20@mit.edu

Ce Pang
Computer Science Department
Rowan University
Glassboro, USA
pangc7@students.rowan.edu

Anna C. Squicciarini
College of Information Science and Technology
Pennsylvania State University
University Park, USA
acs20@psu.edu

Juanjuan Zhao
Shenzhen Institute of Advanced Technology
Shenzhen, P. R. China
jj.zhao@siat.ac.cn

Chengzhong Xu
Computer and Information Science
University of Macau
Macau, P. R. China
czxu@um.edu.mo

ABSTRACT

Vehicles have been increasingly involved in a variety of location-based services (LBS). In many LBS, vehicles have to disclose their locations to servers to perform their services, raising some privacy issues. Currently, one of the popular location privacy-preserving mechanisms applied in many LBS is *location obfuscation*, where mobile users are allowed to report perturbed locations instead of their real locations to servers. However, most existing obfuscation approaches still assume that users can move freely and independently on a 2-dimensional (2D) plane. The 2D plane representation is however inadequate, as vehicles' mobility is highly correlated with external factors, such as traffic conditions, road networks, and even weather or road construction. This additional auxiliary information, unfortunately, helps attackers shrink the search range of vehicles' locations and hence increases the risk of location exposure.

In this paper, we first develop a new inference attack algorithm that leverages traffic information to recover target vehicles' real locations from obfuscated locations (generated by current location obfuscation algorithms). We then develop a two-layer obfuscation strategy, where in Layer 1, a vehicle's locations are obfuscated by a "fake" trajectory that is hard to distinguish from real trajectories, and in Layer 2, the fake trajectory is obfuscated to double-shield vehicles' location privacy. Our experimental results demonstrate that 1) the new inference attack can accurately track vehicles' real locations obfuscated by two state-of-the-arts obfuscation methods, and furthermore, 2) the two-layer obfuscation algorithm can effectively protect vehicles' location privacy under the new inference attack without compromising QoS.

SIGSPATIAL'20, November 2020, Seattle, WA, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

CCS CONCEPTS

• **Security and privacy → Security services**; • **Theory of computation → Mathematical optimization**.

KEYWORDS

location privacy, obfuscation, spatial correlation, traffic flow

ACM Reference Format:
Chenxi Qiu, Li Yan, Ce Pang, Anna C. Squicciarini, Juanjuan Zhao, and Chengzhong Xu. 2018. RoadAdaptor: An Adaptive Obfuscation Strategy for Vehicle Trajectory Privacy Against Spatial Correlation Aware Attacks. In *Proceedings of ACM Conference (SIGSPATIAL'20).* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

With ubiquitous wireless connectivity and continued advances in positioning technologies in mobile/on-board devices, vehicles have been increasingly participating in a variety of location-based services (LBS), from real-time navigation (e.g., Waze [1]) to journalism and crisis response (MediaQ [2]) and commercial transportation systems (e.g., Uber-like platforms [3]). In most of those LBS applications, vehicles need to report their locations to servers to guarantee high quality-of-services (QoS). This location-sharing practice enables vehicle tracking, raising privacy issues that are not limited to whereabouts of the vehicles, but may also relate to some other sensitive information such as drivers' home/working address, sexual preferences, financial status, etc [4, 5].

To enjoy the benefit of LBS without leaking users' sensitive information, location privacy in LBS has become a very active research topic [6–15]. A large body of recent work has been centered on *location obfuscation* [8–12], a location privacy protection mechanism (LPPM) in which mobile users are allowed to report perturbed locations instead of true locations to servers. Compared with traditional cryptographic techniques [7], obfuscation has been acknowledged to be more suitable for mobile LBS applications due to its 1) low computational demand for mobile devices [11, 12], 2) high effectiveness in protecting data privacy from server-side eavesdropping [16], and 3) high accessibility for various service providers [8].

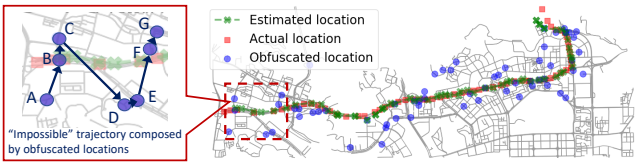


Figure 1: Example: accuracy of location tracking via HMM.

Despite these merits, most existing location obfuscation approaches are still designed against simple threat models, where users' mobility is considered on a 2-dimensional (2D) plane and independent from each other [8, 10, 13, 17]. Clearly, these are strong and unrealistic assumptions, that have led to insufficient solutions for vehicles operating in a road network. First, vehicles cannot drive freely over roads like moving on a 2D plane. Instead, each individual vehicle's mobility is restricted by road network topology and traffic regulations [18]. Second, vehicles' mobility is spatially correlated with other vehicles due to traffic flows. For example, vehicles driving on the same lane are likely to follow a similar speed [18]. Given the pervasiveness of geo-location and mobility services, with limited effort, an attacker may get a hold of vehicle mobility patterns, and attempt to use them as auxiliary information to increase chances of location identification.

To demonstrate the aforementioned potential privacy issue, in this paper, we first develop a new threat model, where an attacker aims to recover a target vehicle's real trajectory from its reported (obfuscated) locations by leveraging the publicly available traffic flow information. We model the vehicle's mobility by a *hidden Markov model (HMM)*: In each round, 1) the vehicle's actual and obfuscated locations are considered as a *hidden state* and an *observable state*, respectively, and 2) the HMM transition matrix describes the probabilities of the vehicle traveling between the locations over the map. With HMM, the vehicle's real trajectory can be estimated with a high accuracy using well-developed hidden state inference algorithms (e.g., the Viterbi algorithm [19]). Figure 1 shows an example of a taxicab tracking in Shenzhen mobility trace dataset [20] using the HMM model. From the figure, we find that the vehicle's trajectory is estimated with a very high accuracy given each single obfuscated location has been obfuscated with a state-of-the-art obfuscation algorithm [13].

It is important to note that when a vehicle obfuscates its location independently per round, the transition between its adjacent reported locations mostly looks "impossible" (see the trajectory $\{A, B, C, D, E, F, G\}$ in Figure 1. This unfortunately helps the inference algorithms like Viterbi eliminate trajectories unlikely to happen in traffic and hence increase the risk of location exposure. As a countermeasure, we design a two-layer obfuscation strategy:
1) In Layer 1, the vehicle generates a "fake" trajectory. The fake trajectory is created to follow the real local traffic flows, which makes it hard to be eliminated by the inference algorithms.
2) In Layer 2, the fake trajectory from Layer 1 will be further obfuscated by the obfuscation function, to double-field the vehicle's location privacy while achieving high QoS.

Besides hiding the real trajectory, the selected fake trajectory in Layer 1 is required to guarantee high QoS, i.e., its deviation from the vehicle's real trajectory should be limited by a threshold. Such a requirement, however, is non-trivial to achieve when the vehicle's mobility is unpredictable. For instance, a fake trajectory originally close to the vehicle may inevitably deviate far away from the vehicle later due to the restriction of traffic flows. As a solution, instead of relying on a single fake trajectory, we maintain a pool of candidate trajectories and select only one to pass to Layer 2 when reporting. The maintenance of trajectory pool follows a similar idea of *natural selection*, i.e., in each round, trajectories leading to high privacy level and high QoS will survive and reproduce a set of new trajectories in the next round, while other trajectories will die off with no more contribution to the pool of further generations.

The obfuscation function in Layer 2 is designed to maximize the privacy level of vehicles (measured by two privacy criteria: expected inference error (EIE) [17] and geo-indistinguishability (GI) [8]) while guaranteeing high QoS. We formulate the obfuscation function generation (OFG) problem as a *linear programming (LP)* problem. By using the angular block structure of the constraint matrix in OFG, we then apply the Dantzig-Wolfe decomposition to solve OFG with high time-efficiency [21].

With respect to performance, simulation results based on Shenzhen taxi trajectory records [20] demonstrate that: 1) Given vehicles' locations are obfuscated by two state-of-the-art obfuscation methods [11, 17], the new threat model can accurately track vehicles' locations, and on average, and its EIE is 87.28% lower than that of the classic Bayesian inference attack [11, 12]. 2) The two-layer obfuscation strategy can effectively protect vehicles' location privacy from the new attack model, e.g., its EIE is at least 499.89% higher than the obfuscation algorithms in [11, 17].

Our contributions can be summarized as follows:
1) We build a realistic threat model for LBS that accounts for vehicles' mobility features over roads. By leveraging traffic flow information, we design a new inference attack, which can accurately recover vehicles' trajectories from their obfuscated locations using the current obfuscation approaches.
2) As a countermeasure, we then develop a two-layer location obfuscation strategy to protect the location privacy of vehicles without compromising the QoS, even assuming attackers can leverage traffic flow information for inference.
3) We conduct a simulation based on a real-world dataset to test the performance of our strategy. The simulation result demonstrates the high accuracy of the new inference attack in tracking vehicles. Furthermore, it demonstrates the effectiveness of the two-layer obfuscation algorithm in protecting vehicles' location privacy under the new threat model.

The remainder of the paper is organized as follows: Section introduces the location obfuscation framework. Section 3 introduces the new threat model and Section 4 describes the obfuscation algorithm. Section 5 evaluates the performance of our algorithm. Finally, Section 6 presents the related work and Section 7 makes a conclusion.

2 FRAMEWORK

Before describing the threat model and our solution, we first introduce the general framework of location obfuscation used in this paper and numerous prior studies [8, 10, 11, 13, 17]. For modeling purposes, time is discretized by way of *rounds*. As Figure 2 shows, in each round, available vehicles need to report their current locations to the server in order to participate in the coming services,

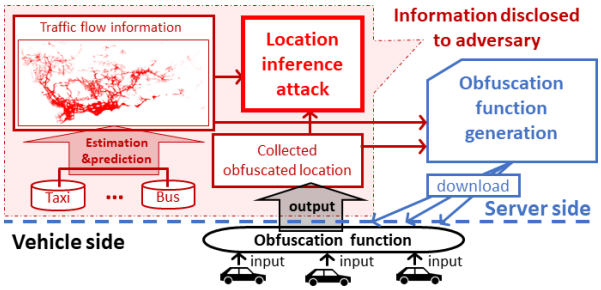


Figure 2: Location obfuscation framework.

which may lead to privacy breaches at the server-side. Like the prior studies, we assume that the server may suffer from a *passive attack, where attackers can eavesdrop vehicles' reported locations breached by the server.* As a solution, before reporting the location to the server, each privacy-aware vehicle will obfuscate its current location via an *obfuscation function.* The obfuscation function takes the vehicle's current true location as input, and returns the probability distribution of obfuscated location, based on which the vehicle can select a location to report. In parallel to protecting vehicles' location privacy, the obfuscation function also aims to minimize the traveling cost for vehicles (i.e. to ensure high QoS).

In fact, the derivation of the obfuscation function is computationally intensive and beyond the capability of mobile devices at the vehicle-side. For example, in our prior work [22], calculating the obfuscation function involves millions of decision variables. As such, we adopt a remote computing framework, where the server derives the obfuscation function first in each round, and after then, each vehicle needs to download the function to obfuscate its own location [10, 11]. Note that, *although the server takes charge of calculating the obfuscation function, it cannot obtain vehicles' exact locations since the obfuscated location is chosen randomly by each vehicle - according to the obfuscated location probability distribution.*

On the other hand, even though vehicles' locations are obfuscated at the server-side, attackers may coordinate an inference attack by reverse-engineering obfuscated locations [8, 10, 13, 17]. In particular, an attacker can carry out background information attacks, for instance by leveraging traffic flow information (e.g., estimated by the GPS records of floating cars [23]). As such, the attacker can significantly improve its inference accuracy, as shown in the example in Figure 1. Given the inadequacy of current approaches in dealing with these attack settings, our objective in this paper is to **design a new location obfuscation strategy to protect the location privacy of vehicles operating over roads without compromising QoS.** To accomplish this, we take efforts to address the following two questions:

Q1. How can an attacker use traffic flow information to infer a target vehicle's locations (Section 3)?

Q2. How to develop a location obfuscation strategy to protect vehicles' location privacy even attackers can leverage traffic flow information for interference (Section 4)?

3 THE THREAT MODEL

The mobility of vehicles is relatively consistent over space and time, as vehicles are restricted to follow road topology and traffic regulations. From the perspective of attackers, these environmental factors provide easy to exploit side-channel information, which can be leveraged to filter out unlikely routes and hence increase the risk of vehicles' location exposure. As a first step, we develop a new threat model to infer vehicles' trajectories by leveraging traffic flow information.

Assumptions and notations. We consider the case that a vehicle reports its locations multiple times during its trip and let $t = \{1, 2, ..., T\}$ denote the rounds of its reports. Note that the time duration of each round is possibly different, depending on the time point the vehicle reports its location. Table 1 lists the main notations that will be used throughout the paper and their description.

Table 1: Main notations and definitions

Notation	Description
$\mathcal{S}$	the location set
K	the number of locations in $\mathcal{S}$
T	the number of reported locations in the whole trip
s_k	the k^{th} location in $\mathcal{S}$
$\mathbf{Z}^t$	the obfuscation matrix at round t
$z^t_{i,j}$	the probability of selecting location s_j as the obfuscated location given the true location s_i (or location s_i in the selected fake trajectory (Section 4))
X^t	random variable denoting the vehicle's true location
Y^t	random variable denoting the vehicle's obfuscated location
W^t	random variable denoting vehicle's location in the selected fake trajectory
$\vec{\mathbf{s}}_o$	the observed sequence (the sequence of reported locations)
s_{o^t}	the tth reported location
$\mathbf{P}^{t,t'}$	the transition matrix of the vehicle's location from round t to round t'
$p^{t,t'}_{i,j}$	the transition probability that the vehicle moves from s_i to s_j from round t to round t'
s_{r^t}	the vehicle's actual location in round t
$\vec{\mathbf{s}}_r$	the sequence of real locations
$\vec{\mathbf{s}}_{f^t_m}$	the m^{th} fake trajectory in round t
M	the fake trajectory pool capacity

Like [11], we discretize the road network by a set of *road segments*. Precisely, we create a *connection* when a road intersects, furcates, joins with other roads, or turns into a different direction. These connections divide the road network into a set of *edges*, which only connect with other edges at their endpoints. Each edge can be further partitioned into *road segments* with the same length δ[1]. Considering the computational tractability, instead of finding the true location of the target vehicle, we assume that attackers aim to identify which segment the vehicle is located. Clearly, with smaller δ, the vehicle location can be estimated with higher accuracy. For simplicity, we use the term "location" instead of "road segment" in what follows. For the target vehicle, we let two random variables X^t and Y^t denote its actual and reported locations at round t, respectively, and let $\mathcal{S} = \{s_1, ..., s_K\}$ denote the *location set* over the whole road network. The obfuscation function in each round t can be also represented as a stochastic matrix $\mathbf{Z}^t = \{z^t_{i,j}\}_{K\times K}$, called *obfuscation matrix*, where each $z^t_{i,j}$ denotes the probability

[1]Due to the variety of road segment, there exists some segments with length smaller than δ. But as δ is small enough, we won't discuss these segments in the following part considering the complexity of the formulated problem.

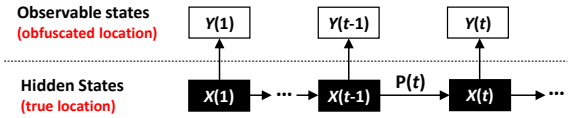


Figure 3: Hidden Markov Model.

of selecting location s_j as the obfuscated location given the true location s_i, i.e., $z_{i,j}^t = \mathbb{P}\mathrm{r}\left(Y^t = s_j | X^t = s_i\right)$.

We let s_{r^t} and s_{o^t} denote the vehicle's true and reported locations in the round t. Then, the sequence of the vehicle's reported locations and the corresponding true locations are represented by $\vec{\mathbf{s}}_o = \left\{s_{o^1}, ..., s_{o^T}\right\}$ and $\vec{\mathbf{s}}_r = \left\{s_{r^1}, ..., s_{r^T}\right\}$, respectively.

Hidden Markov Model. When the vehicle reports its obfuscated location to the server, its true location is hidden from the attacker. Nevertheless, the vehicle's obfuscated location is related to its true location by following a probability distribution determined by the vehicle's true location and the obfuscation matrix $\mathbf{Z}^t$. On the other hand, due to the constraints of road network environment and traffic conditions, the vehicle's true locations in each pair of adjacent rounds are spatially correlated. That is, the vehicle's location in each round is dependent on its location in the previous rounds.

Accordingly, *hidden Markov model (HMM)* offers a conveniently adjustable and straightforward model to characterize the aforementioned vehicles' mobility features. As Figure 3 shows, in each round t, the vehicle's true location X^t and reported location Y^t are considered as its *hidden* and *observable* states, respectively, where X^t follows a *Markov process*, i.e., X^t only depends on X^{t-1}. We let the transition matrix $\mathbf{P}^{t,t'} = \left\{p_{i,j}^{t,t'}\right\}_{K\times K}$ $(t < t')$ describe the transition probabilities between the vehicle's hidden states from round t to t', where $p_{i,j}^{t,t'} = \mathbb{P}\mathrm{r}\left(X^{t'} = s_j | X^t = s_i\right)$.

With HMM, given the observed sequence $\vec{\mathbf{s}}_o = \left\{s_{o^1}, ..., s_{o^T}\right\}$ and the transition matrix $\mathbf{P}^{1,2}, \mathbf{P}^{2,3}, ..., \mathbf{P}^{T-1,T}$, the task of the attacker is to derive the maximum likelihood estimate of the vehicle's true locations (trajectory) $\vec{\mathbf{s}}_r = \left\{s_{r^1}, ..., s_{r^T}\right\}$.

Transition matrix learning via probing vehicles. There is an increasing availability of floating vehicle data, which is both historic, in the form of trajectory datasets [18], and real-time, in the form of continuous data streams [24]. The dataset usually records the vehicles' coordinates along with the timestamps, where time is discretized into *slots* (e.g., second [20]). Given the dataset, we can calculate each the transition probability $p_{i,j}^{t-1,t}$ by

$$p_{i,j}^{\tau-1,\tau} = \frac{\#\text{ of vehicles moving from } s_i \text{ to } s_j \text{ from round } \tau-1 \text{ to } \tau}{\#\text{ of vehicles in } s_i \text{ in round } \tau-1}. \tag{1}$$

Note that vehicles mostly don't report their locations in each slot. Suppose that the target vehicle t^{th} and $(t+1)^{\text{th}}$ reports are at slots τ_t and τ_{t+1}, respectively. The corresponding transition matrix for the adjacent reports s_{o^t} and $s_{o^{t+1}}$, $\mathbf{P}^{t,t+1}$ is calculated by

$$\mathbf{P}^{t,t+1} = \prod_{\tau=\tau_t}^{\tau_{t+1}} \mathbf{P}^{\tau,\tau+1}. \tag{2}$$

Vehicle location tracking. We use the Viterbi algorithm [19] to infer the vehicle's real trajectory $\vec{\mathbf{s}}_r = \left\{s_{r^1}, ..., s_{r^T}\right\}$. Given the observed location sequence $\vec{\mathbf{s}}_o = \left\{s_{o^1}, ..., s_{o^T}\right\}$, the Viterbi algorithm

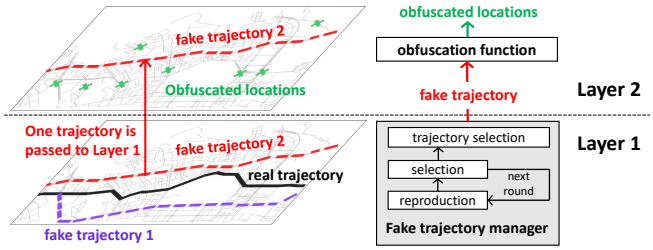


Figure 4: Two layer obfuscation framework.

follows the dynamic programming paradigm to find the most likely real trajectory $\vec{\mathbf{s}}_r = \left\{s_{r^1}, ..., s_{r^T}\right\}$, namely *the Viterbi path*. We let v_j^t represent the probability that the vehicle's t^{th} hidden state is at s_j given the first $t-1$ observations $\left\{s_{o^1}, ..., s_{o^{t-1}}\right\}$, and passing through the most probable state sequence $\vec{\mathbf{s}}$, i.e.,

$$v_j^t = \max_{\vec{\mathbf{s}}\in\mathcal{S}^{t-1}} \mathbb{P}\mathrm{r}(\underbrace{\left\{X^1, ..., X^{t-1}\right\} = \vec{\mathbf{s}}, X^t = s_j}_{\text{hidden states}} \mid \underbrace{Y^1 = s_{o^1}, ..., Y^{t-1} = s_{o^{t-1}}}_{\text{observations}}). \tag{3}$$

The value of each cell v_j^t in iteration t is computed by recursively based on the cells calculated in iteration $t-1$: $v_1^{t-1}, ..., v_K^{t-1}$:

$$v_j^t = \max_{s_k\in\mathcal{S}} v_k^{t-1} p_{k,j}^{t-1,t} z_{j,o^t}. \tag{4}$$

Finally, the Viterbi path (i.e., the estimated real trajectory) can be retrieved by saving back pointers that remember which state was used in Equation (4).

The performance of the location inference attack algorithm based on HMM will be evaluated in Section 5.3, where the algorithm has been demonstrated to have high accuracy in tracking vehicles' locations. In particular, the EIE of the HMM-based inference attack is 87.28% lower than that of the classic Bayesian inference algorithm [11, 12].

4 COUNTERMEASURE: TWO-LAYER LOCATION OBFUSCATION DESIGN

In this section, we will introduce our new obfuscation algorithm to address the vulnerability of vehicles under the inference attack introduced in Section 3. As previously stated, our objective is twofold: 1) to protect vehicles' location privacy over roads and 2) to limit quality loss of LBS. The whole process of location obfuscation at the vehicle-side is composed of two layers (sketched by Figure 4):

Layer 1: The vehicle's locations are first obfuscated by a "fake" trajectory that is hard to distinguish from real trajectories. The generation of the fake trajectory is completed at the vehicle-side by HMM-based inference attack.

Layer 2: The fake trajectory from Layer 1 will be further obfuscated by the obfuscation function to introduce more uncertainty to the vehicle's reported locations without compromising QoS.

The details of the two-layer obfuscation design are introduced in Section 4.1–4.2.

4.1 Layer 1: Fake Trajectory Manager

It is important to note that when the vehicle's location is obfuscated independently per round, the sequence of reported locations

is unlikely to follow the traffic flows. This, unfortunately, helps the HMM-based inference algorithm (like Viterbi) eliminate the obfuscated routes that are impossible to happen in traffic and increases the accuracy of location estimation. Accordingly, instead of generating sporadic "fake" locations, our approach first obfuscates each vehicle's locations by a "fake" trajectory following real traffic flows, which is hard to distinguish from real trajectories by HMM-based inference attack.

To limit quality loss, the fake trajectory is required to be within a certain range with the vehicle's real trajectory. Such a requirement, however, is hard to satisfy by a single fake trajectory when the vehicle's mobility is unpredictable. Due to the restriction of traffic conditions, a fake trajectory may inevitably deviate far away from the real trajectory over time. Accordingly, as opposed to relying on a single fake trajectory, we let the vehicle maintain a pool of candidate fake trajectories dynamically by a *fake trajectory manager (FTM)*. When reporting, FTM randomly selects one fake trajectory from the pool and pass it to Layer 2.

Note that the number of possible trajectories following traffic flows could increase exponentially over time. For the sake of computational tractability, FTM only keeps fake trajectories that can achieve high privacy and low quality loss. We let $\mathcal{F}^t = \left\{\vec{\mathbf{s}}_{f_1^t}, ..., \vec{\mathbf{s}}_{f_M^t}\right\}$ denote the trajectory pool in round t, where M denotes the pool capacity and each $\vec{\mathbf{s}}_{f_m^t} = \{s_{f_m^1}, ..., s_{f_m^t}\}$ $(m = 1, ..., M)$. We define a *fitness value* for each $\vec{\mathbf{s}}_{f_m^t} \in \mathcal{F}^t$

$$w_m^t = w_m^{t-1} + \alpha_d \underbrace{\mathcal{D}\left(s_{f_m^t}, s_{r^t}\right)}_{\text{privacy level}} - \alpha_q \underbrace{\mathcal{Q}\left(s_{f_m^t}, s_{r^t}\right)}_{\text{quality loss}} \tag{5}$$

to evaluate both quality loss and privacy level achieved by $\vec{\mathbf{s}}_{f_m}^t$. Here, $\mathcal{D}\left(s_f, s_r\right)$ and $\mathcal{Q}\left(s_f, s_r\right)$ represent the privacy level and the quality loss generated by the fake location s_f given the real location s_r $(s_f, s_r \in \mathcal{S})$, and α_d and α_q denote their weights, respectively. Like [11–13], we define privacy level $\mathcal{D}\left(s_f, s_r\right)$ as the Euclidean distance between s_f and s_r.

The definition of quality loss $\mathcal{Q}\left(s_f, s_r\right)$ depends on how the utility of LBS is defined in the application. In this work, we consider a specific category of LBS, where vehicles have to physically move to a target location to complete its service. The examples include location-based recommendation systems (e.g., Yelp) and spatial crowdsourcing (e.g., Uber). We assume that the prior distribution of the target location Q is given, then we can define quality loss as the expected distortion of traveling distance [11]:

$$\mathcal{Q}\left(s_f, s_r\right) = \sum_q \mathbb{P}\mathrm{r}\left(Q = s_q\right)\left|C\left(s_r, s_q\right) - C\left(s_f, s_q\right)\right| \tag{6}$$

where $C(s, s_q)$ denotes the traveling distance from s to s_q over the map. Here, we enforce a constraint for the quality loss caused by fake trajectories:

$$\mathcal{Q}\left(s_f, s_r\right) \leq \tilde{\Gamma}, \tag{7}$$

where $\tilde{\Gamma}$ is a predetermined constant. Fake trajectories that cannot satisfy Equation (7) won't be added to the pool.

The maintenance of trajectory pool follows a similar idea of *natural selection*: In each round, trajectories within a certain distance from the current real location will survive and generate a set of new trajectories in the next round, while other trajectories will die off with no more contribution to the pool of further generations. Specifically, the trajectory pool $\mathcal{F}^t$ is updated from $\mathcal{F}^{t-1}$ via two steps: *reproduction* and *selection*:

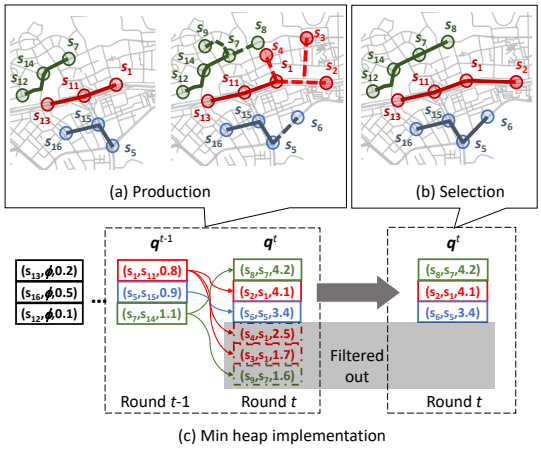


Figure 5: Example of fake trajectory maintenance: At round t, there are three fake trajectories $\vec{\mathbf{s}}_{f_1} = \{s_{12}, s_{14}, s_7\}$, $\vec{\mathbf{s}}_{f_2} = \{s_{13}, s_{11}, s_1\}$, and $\vec{\mathbf{s}}_{f_3} = \{s_{16}, s_{15}, s_5\}$.
(a) Reproduction: $\vec{\mathbf{s}}_{f_2}$ has $\{\vec{\mathbf{s}}_{f_2}, s_2\}$ three possible locations to move to: s_2, s_3, and s_4, so $\vec{\mathbf{s}}_{f_2}$ has three offsprings: $\{\vec{\mathbf{s}}_{f_2}, s_2\}$, $\{\vec{\mathbf{s}}_{f_2}, s_3\}$, and $\{\vec{\mathbf{s}}_{f_2}, s_4\}$. Similarly, $\vec{\mathbf{s}}_{f_1}$ has two offsprings: $\{\vec{\mathbf{s}}_{f_1}, s_8\}$ and $\{\vec{\mathbf{s}}_{f_1}, s_9\}$; $\vec{\mathbf{s}}_{f_3}$ has one offspring: $\{\vec{\mathbf{s}}_{f_3}, s_6\}$.
(b) Selection: Among the six offsprings generated in round t, three offsprings with higher fitness values $\{\vec{\mathbf{s}}_{f_1}, s_8\}$, $\{\vec{\mathbf{s}}_{f_2}, s_2\}$, and $\{\vec{\mathbf{s}}_{f_3}, s_6\}$ are selected for use in the next round.

Step 1. Reproduction: Suppose that a vehicle has followed a fake trajectory $\vec{\mathbf{s}}_{f_m^{t-1}} \in \mathcal{F}^{t-1}$ in rounds 1, ..., $t-1$. In round t, by extending $\vec{\mathbf{s}}_{f_m^{t-1}}$ to a new location $s_{f_{m'}^t}$ that is reachable by the vehicle, we can obtain a new trajectory: $\vec{\mathbf{s}}_{f_{m'}^t} = \left\{\vec{\mathbf{s}}_{f_m^{t-1}}, s_{f_{m'}^t}\right\}$. We call $\vec{\mathbf{s}}_{f_{m'}^t}$ an *offspring* of $\vec{\mathbf{s}}_{f_m^{t-1}}$. Given the transition matrix $\mathbf{P}^{t-1,t}$, we can find the set of all possible offsprings of $\vec{\mathbf{s}}_{f_m^{t-1}}$,

$$\mathcal{R}\left(\vec{\mathbf{s}}_{f_m^{t-1}}\right) = \left\{\left\{\vec{\mathbf{s}}_{f_m^{t-1}}, s_j\right\} \middle| s_j \in \mathcal{S} \text{ with } p_{f_m^{t-1},j}^{t-1,t} > 0\right\}, \tag{8}$$

i.e., $\vec{\mathbf{s}}_{f_m^{t-1}}$ can be possibly extended to a location s_j if only if the transition probability from $s_{f_m^{t-1}}$ to s_j, $p_{f_m^{t-1},j}^{t-1,t}$, is non-zero.

Figure 5(a) gives an example on how fake trajectories' offsprings are reproduced from round $t-1$ to t.

Step 2. Selection: Reproduction generates a set of new fake trajectories $\mathcal{R}^t = \cup_{m=1}^{M} \mathcal{R}\left(\vec{\mathbf{s}}_{f_m^{t-1}}\right)$. To filter out the the trajectories with low privacy level or high quality loss, FTM ranks all the nodes in $\mathcal{R}^t$ by their fitness values and only keeps the M highest ones. Figure 5(b) gives an example on how FTM selects the offsprings in round t.

Algorithm implementation. As Figure 5(c) shows, the fake trajectory $\vec{\mathbf{s}}_{f_m^t}$ in round t is stored as a node (or triple): $\left(s_{f_m^t}, s_{f_m^{t-1}}, w_m^t\right)$, including $\vec{\mathbf{s}}_{f_m^t}$'s current location, previous location, and fitness value. Given $s_{f_m^\tau}$ and $s_{f_m^{\tau-1}}$ in rounds $\tau = 1, ..., t$, the whole trajectory $\vec{\mathbf{s}}_{f_m^t}$ can be derived via backtracking as each node has the pointer to

previous location. When $t = 1$, each $s_{f_m^{t-1}}$ is set by ϕ, indicating no previous location exists.

FTM stores the nodes in each round t in a *min heap*, denoted by $\mathbf{q}^t$, which is efficient in finding the top M elements from a set [25]. $\mathbf{q}^t$ has two features: (a) The first node in $\mathbf{q}^t$, also called the *top* of $\mathbf{q}^t$, has the minimum fitness value. (b) Two types of operations can be conducted on $\mathbf{q}^t$: *push* (i.e., to insert a node to $\mathbf{q}^t$) and *pop* (i.e., to remove the top node from $\mathbf{q}^t$). The time complexity of both types of operations is $O(\log M)$. Note that pop/push won't change feature (a) of $\mathbf{q}^t$, i.e., the top node always has the minimum fitness value.

To find the top M nodes, FTM traverses each trajectory in $\mathcal{R}^t$, calculates its fitness value, and determines whether to push the corresponding node onto $\mathbf{q}^t$. The first M nodes are directly added to $\mathbf{q}^t$. After then, whether a new node q^t_{new} should be added to $\mathbf{q}^t$ depends on whether q^t_{new} has a higher fitness value than the current top node q^t_{top} in $\mathbf{q}^t$:

i) If no, q^t_{new} won't be pushed onto $\mathbf{q}^t$, since q^t_{new} has a lower fitness value than any of the M nodes in $\mathbf{q}^t$.

ii) If yes, q^t_{top} will be popped off and q^t_{new} will be pushed onto $\mathbf{q}^t$. Note that q^t_{top} cannot be one of the M highest nodes, because 1) its fitness values is lower than q^t_{new}'s and also 2) lower than the fitness values of the other $M-1$ nodes in $\mathbf{q}^t$ (based on feature (a)).

Suppose there are U locations in $\mathcal{R}^t$ satisfying the constraint in Equation (7). To obtain $\mathbf{q}^t$, FTM takes up to UM push/pop operations, which needs totally $O(UM \log M)$ operations. As both M and U are not large in piratical ($M = 100$ and $U \leq 300$ in the experiment in Section 5), such a computation load is acceptable to mobile devices (e.g., smartphones). The detailed pseudo code of the implementation is given in Algorithm 1 in Appendix A.

Fake trajectory determination. After obtaining $\mathbf{q}^t$, FTM randomly picks up a fake trajectory from the pool and pass it to Layer 2 for further obfuscation. Due to the dynamics of the road environment and traffic conditions, the vehicle may have to switch frequently among different fake trajectories when operating over roads, possibly causing the reported trajectory look "impossible". As a solution, we require FTM to select the trajectories that satisfy the constraint $\Pr\left(W^t = s_{f_m^t} | W^{t-1} = s_{f_{m^*}^{t-1}}\right) \geq \eta$, to ensure that reported trajectories are switched smoothly, where the random variable W^t denotes the location of selected fake trajectory in round t and $\eta > 0$ is a predetermined threshold.

4.2 Layer 2: Obfuscation Function

The obfuscation function is first generated by the server-side and then downloaded by each vehicle by the end of each round. In Layer 2, the obfuscation function takes the fake trajectory selected by Layer 1 as the input and outputs an obfuscated location to report.

Obfuscation function generation. Similar to the ordinary obfuscation algorithm introduced in Section 3, the obfuscation function at each round t can be represented as a *stochastic matrix* $\mathbf{Z}^t = \{z^t_{i,j}\}_{K\times K}$, where $z^t_{i,j}$ denotes the probability of selecting s_j as the obfuscated location given the t^{th} location s_i in the selected fake trajectory (note that s_i does not denote the vehicle's real location, which is different from the ordinary obfuscation algorithm introduced in Section 3), i.e., $z^t_{i,j} = \Pr\left(Y^t = s_j | W^t = s_i\right)$. For simplicity, we let $\mathbf{Z}^t = \left[\begin{array}{ccc} \mathbf{z}_1^{t\top} & \dots & \mathbf{z}_K^{t\top} \end{array}\right]^\top$, where $\mathbf{z}^t_k = [z^t_{1,l}, z^t_{2,l}, ..., z^t_{K,l}]^\top$ ($l = 1, ..., K$). The obfuscation function is designed to achieve the following goals:

1) to introduce additional uncertainty to obfuscated location such that higher privacy level can be achieved (measured by both expected inference error (EIE) and geo-indistinguishability (GI)),

2) to limit the quality loss by a threshold.

1a) EIE: which is also known as unconditional expected distortion of estimated location, defined by $\sum_{s_{o^t}\in\mathcal{S}} \Psi\left(\mathbf{z}_{o^t}\right)$, where

$$\Psi\left(\mathbf{z}_{o^t}\right) = \Pr\left(Y^t = s_{o^t}\right) \sum_{s_{f^t}\in\mathcal{S}} \Pr\left(W^t = s_{f^t} | Y^t = s_{o^t}\right) \mathcal{D}(s_{\hat{f}^t}, s_{f^t}), \tag{9}$$

and $s_{\hat{f}^t}$ is the estimated location derived by the attacker. As the obfuscation function generates obfuscated location independently in each round, $s_{\hat{f}^t}$ can be derived using the Bayesian inference attack [17], i.e., $s_{\hat{f}^t} = \arg\min_{s_f\in\mathcal{S}} \sum_k \Pr\left(W^t = s_k | Y^t = s_f\right) \mathcal{D}\left(s_f, s_k\right)$.

1b) GI: EIE assumes certain types of prior information that the attacker may obtain, but does not consider the posterior information leaked from obfuscated location. As such, we require the obfuscation function to achieve GI, which limits the posterior information leakage through a *differential privacy* based criteria. Formally, $\mathbf{Z}^t$ satisfies ϵ-GI if only if, $\forall s_j, s_k \in \mathcal{S}$,

$$\frac{\Pr\left(W^t = s_j | Y^t = s_l\right)}{\Pr\left(W^t = s_k | Y^t = s_l\right)} \leq e^{\epsilon \min\left\{c_{j,k}, c_{k,j}\right\}} \times \frac{\Pr\left(W^t = s_j\right)}{\Pr\left(W^t = s_k\right)}, \ \forall s_l \in \mathcal{S} \tag{10}$$

where ϵ is the parameter to quantify how much the vehicle's actual location is disclosed according to the reported location, i.e., higher ϵ implies more information disclosed and a lower privacy level achieved. We let the polyhedron Φ_l^{GI} represent the constraints of $\mathbf{z}_l$ ($l = 1, ..., K$) in Equation (10), i.e., $\mathbf{z}_l \in \Phi_l^{\text{GI}}$.

2) Quality loss: Given the vehicle's obfuscated location s_{o^t} and actual location s_{r^t}, we measure the quality loss by the expected distortion of estimated traveling distance:

$$\mathbb{E}\left(Q\left(s_{o^t}, s_{r^t}\right)\right) = \sum_{r^t, o^t} \Pr\left(X^t = s_{r^t}, Y^t = s_{o^t}\right) Q\left(s_{o^t}, s_{r^t}\right). \tag{11}$$

To limit the quality loss, we require that

$$\mathbb{E}\left(Q\left(s_{o^t}, s_{r^t}\right)\right) \leq \Gamma, \tag{12}$$

where $\Gamma > 0$ is a predefined constant.

Note that the obfuscation matrix $\mathbf{Z}^t$ defines the relationship between the fake trajectory location s_{f^t} and the final obfuscated location s_{o^t}, but without considering the real location s_{r^t}. Thus, we enforce a stronger constraint: $\forall (s_{o^t}, s_{f^t}) \in \mathcal{S}^2$,

$$z^t_{f^t, o^t} \leq 0, \ \text{ when } \ Q\left(s_{o^t}, s_{f^t}\right) > \Gamma - \tilde{\Gamma}. \tag{13}$$

where $\tilde{\Gamma}$ is a threshold hold to limit the quality loss generated by fake trajectories (defined in Equation 7).

Proposition 4.1. *The constraint in Equation (13) is a sufficient condition for* $\mathbb{E}\left(Q\left(s_{o^t}, s_{r^t}\right)\right) \leq \Gamma$.

Proof. The detailed proof can be found in Appendix B. □

For simplicity, we let the polyhedron Φ_l^{QoS} represent the constraints of $\mathbf{z}_l$ ($l = 1, ..., K$) in Equation (13), i.e., $\mathbf{z}_l \in \Phi_l^{\text{QoS}}$.

Given the above assumptions and definitions, we now formulate the problem of *the obfuscation function generation (OFG)* as a mathematical optimization problem, of which the objective is

to maximize the overall EIE $\sum_l \Psi(\mathbf{z}_l)$, while satisfying both *GI constraints* (Equation (10)) and *QoS constraints* (Equation (13)).

$$\max \quad \sum_l \Psi(\mathbf{z}_l) \tag{14}$$

$$\text{s.t.} \quad \mathbf{z}_l \in \Phi_l^{\text{GI}},\ \mathbf{z}_l \in \Phi_l^{\text{QoS}},\ \mathbf{z}_l \geq \mathbf{0},\ \forall l, \tag{15}$$

$$\sum_l z_{k,l} = 1 \ \text{(prob. unit measure)},\ \forall k. \tag{16}$$

OFG is a linear programming (LP) problem that can be solved by standard LP approaches such as the simplex methods [26]. This, however, introduces challenges with respect to time efficiency and scalability. The number of decision variables in the obfuscation matrix $\mathbf{Z}^t$ is quadratic to the number of discrete locations in $\mathcal{S}$, e.g., thousands of discrete locations will generate millions of decision variables in OFG, leading to an extremely high computation load. Nevertheless, to account for realistic applications where traffic conditions change all the time, the derivation of optimal $\mathbf{Z}^t$ is supposed to be time-efficient to handle the highly dynamic inputs.

To tackle this issue, a promising route is to adopt *decomposition* techniques based on how decision variables in the problems are coupled [27]. We let $\mathbf{\Phi}_l = \mathbf{\Phi}_l^{\text{GI}} \cap \mathbf{\Phi}_l^{\text{QoS}}$. The constraints for $\mathbf{z}$ in OFG has a *block angular* structure, i.e., 1) the constraints $\mathbf{\Phi}_1, ..., \mathbf{\Phi}_K$ (for $\mathbf{z}_1, ..., \mathbf{z}_K$ respectively) are all disjoint; 2) only the joint constraints $\sum_l z_{k,l} = 1$ ($k = 1, ..., K$) link together the different decision vectors $\mathbf{z}_1, ..., \mathbf{z}_K$. Such block angular structure makes OFG well-suited to *Dantzig-Wolfe (DW) decomposition* [21], which can efficently solve the OFG with a cluster. The detailed algorithm for solving OFG via the DW decomposition is introduced in Appendix C.

5 EXPERIMENTAL VALIDATION

In this section, we evaluate the performance of the HMM-based inference attack (Section 5.3) and the two layer location obfuscation algorithm (Section 5.4). We carry out an extensive evaluation of our methods using a real dataset, which contains the GPS records of around 28,000 vehicles (Section 5.2).

5.1 Benchmarks

a) Inference algorithm. In Section 5.3, we compare the HMM-based inference attack with a benchmark, called the *Bayesian inference attack* (or optimal inference attack), denoted by **Bayes** [11, 12]. In the Bayesian inference attack, according to the vehicle's reported locations, the attacker derives the posterior of the vehicle's real location via Bayes' theorem and then estimates its real location by finding the location $\hat{s} \in \mathcal{S}$ that minimizes the EIE, i.e., $\hat{s} = \arg\min_{s_r \in \mathcal{S}} \sum_{s_k \in \mathcal{S}} \Pr\left(Y^t = s_k | X^t = s_l\right) \mathcal{D}(s_r, s_k)$.

b) Obfuscation algorithms. In Section 5.4, we compare our location obfuscation strategy with two representative algorithms, both of which obfuscate vehicles' locations independently per round:
(1) *Planar Laplacian noise* (denoted by **Laplace**) [8, 10], where the obfuscation probabilities are calculated by $\Pr\left(Y^t = s_j | X^t = s_i\right) \propto e^{-\frac{\mathcal{D}(s_j, s_i)}{L_{\max}}}$, and $L_{\max}$ is the maximum distance between any two locations in the target region.
(2) *LP based obfuscation* (denoted by **LPO**) [11], which follows a linear programming framework. The objective of [11] is to minimize the traveling distance estimation error of a single vehicle with the ϵ-GI constraints satisfied. In [11], the network-constrained mobility features of vehicles are considered.

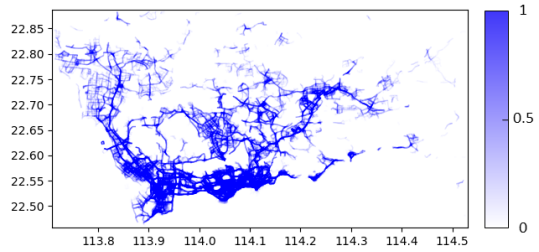


Figure 6: Heat map of the estimated transition probabilities between the road segments in Shenzhen.

We mainly measure the following two metrics:
(i) *Privacy level*, measured by EIE, which is calculated by $\sum_{s_{o^t} \in \mathcal{S}} \Pr\left(Y^t = s_{o^t}\right) \sum_{s_{r^t} \in \mathcal{S}} \Pr\left(X^t = s_{r^t} | Y^t = s_{o^t}\right) \mathcal{D}(\hat{s}, s_{r^t})$, where $\hat{s}$ is the estimated location by the inference attack.
(ii) *Quality loss*, measured by the expected distortion of estimated traveling distance by the server (defined in Equation (11)). Here, we assume the task are uniformly distributed over the location set $\mathcal{S}$.

5.2 Dataset

The dataset used for experiment includes the movement records of various vehicles. In the simulation, we use the data recorded from Jan 1, 2015 to Dec 31, 2015, including:
(1) *Taxicab Dataset.* This dataset records the status (e.g., timestamp, GPS position, velocity, occupancy) of 15,610 taxicabs. The daily size of the uploaded data is around 2 GB.
(2) *Dada Car Dataset.* This dataset is provided by the Dada Car corporation (a customized transit service similar to UberPool), which records the status (e.g., timestamp, position, velocity) of 12,386 customized transit service vehicles.
(3) *Road Map.* The road map of Shenzhen is obtained from OpenStreetMap [28]. According to the municipal information of Shenzhen [20], we use a bounding box with coordinate ($lat = 22.4450, lon = 113.7130$) as the south-west corner, and coordinate ($lat = 22.8844, lon = 114.5270$) as the north-east corner, which covers an area of around 2,926km^2, to crop the road map data.

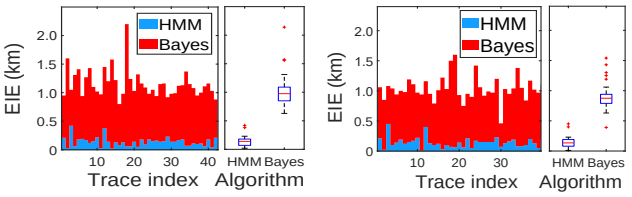
We utilized a 117 TB Hadoop Distributed File System (HDFS) [29] on a cluster consisting of 51 nodes to efficiently manage these datasets. Each node is equipped with 28 cores and 64 GB RAM. All data processing and analysis is accomplished with Apache Spark [30], which is a fast in-memory cluster computing system, deployed over the Hadoop cluster.

5.3 Trajectory Inference Algorithms

We first test the accuracy of the HMM-based trajectory inference algorithm (denoted by **HMM**) introduced in the threat model (Section 3). In what follows, we set the algorithm parameters $\epsilon = 100$, $\Gamma = 1km$, $\tilde{\Gamma} = 0.5km$, $\eta = 0.1$, and $M = 100$ by default.

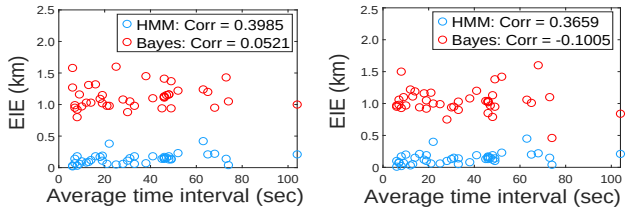
Using the vehicles' historical traffic records in Shenzhen [20], we can train the HMM transition matrices over time by Equation (1). Figure 6 shows a heat map of the transition probabilities between the road segments in Shenzhen.

We pick up 42 taxicab traces from the dataset and consider them as the trajectories of the target vehicles. Figure 7(a)(b) compare the



(a) Obfuscated by Laplace (b) Obfuscated by LPO

Figure 7: Comparison of EIE between HMM and Bayes.



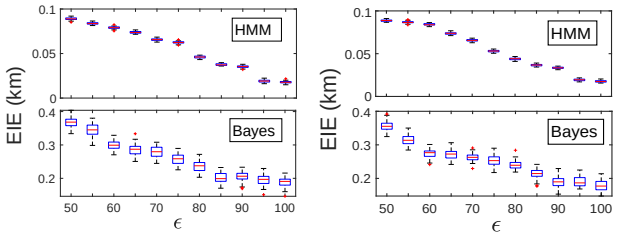
(a) Obfuscated by Laplace (b) Obfuscated by LPO

Figure 8: Correlation between EIE and report time interval.

EIE of HMM and Bayes when the vehicles' locations are obfuscated by Laplace and LPO, respectively. From both figures, we can find that HMM, as a spatial correlation aware inference attack, has significant higher inference accuracy than Bayes, e.g., its EIE is 87.97% and 86.58% lower than that of Bayes when Laplace and LPO are applied for location obfuscation. The figures also demonstrate that the current location obfuscation approaches, like Laplace and LPO, are insufficient to protect vehicles' location privacy from spatial correlation aware attacks such as HMM.

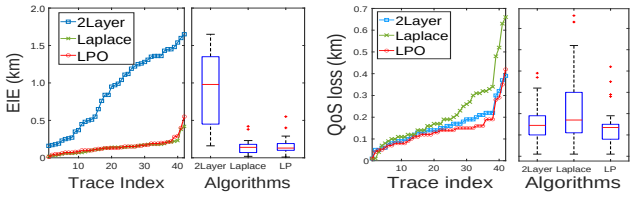
We also note that the location report frequency from vehicles varies across different traces. Hence, it is interesting to check how the report frequency impacts the EIE of HMM and Bayes. In Figure 8(a)(b), we depict the correlation between EIE and the average report interval when the vehicles' locations are obfuscated by Laplace and LPO. We find that in both figures: (1) no significant correlation can be found between the average report intervals and Bayes' EIE, while (2) HMM's EIE is positively correlated with the report intervals. For (1), it is because that Bayes infers vehicles' locations in each round independently, without considering any correlation between adjacent reports. In contrast, the accuracy of HMM highly depends on the spatial correlation information from vehicles' adjacent reports. As such, for (2), when report intervals are larger, the adjacent reports from vehicles are less correlated, which reduces the accuracy of HMM.

Recall that both Laplace and LPO aim to satisfy ϵ-GI, of which the privacy level is quantified by the parameter ϵ (Equation (10)), i.e., lower ϵ indicates higher privacy level. Next, we measure how ϵ impacts the EIE of both HMM and Bayes. The results are depicted in Figure 9, where ϵ is changed from 50 to 100, and the target vehicles' locations are obfuscated by Laplace and LPO in Figure 9(a)(b), respectively. Not surprisingly, EIE decreases with the increase of ϵ in both figures, since higher ϵ enforces more deviation from obfuscated locations to real locations, which reduces the accuracy of HMM and Bayes. However, the impact of ϵ on HMM is not as significant as it is on Bayes, i.e. when an obfuscated location has a higher deviation from the real location, it has less correlation



(a) Obfuscated by Laplace (b) Obfuscated by LPO

Figure 9: EIE of HMM and Bayes with different ϵ.



(a) EIE (b) Quality loss

Figure 10: Comparison between obfuscation algorithms.

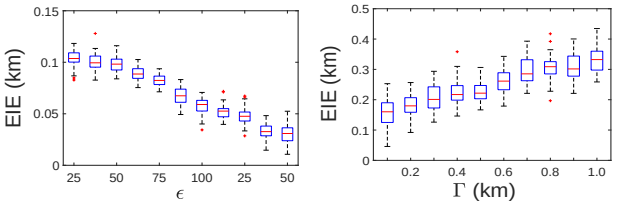
with the last reported location, and hence it is more likely to be eliminated by HMM. This helps increase the accuracy of HMM.

5.4 Location Obfuscation Algorithms

We now test the performance of the two-layer location obfuscation algorithm (denoted by **2Layer**) in terms of both privacy and quality loss, with the comparison of the two benchmarks **Laplace** and **LPO**. Here, we use HMM as the inference attack algorithm to estimate vehicles' locations.

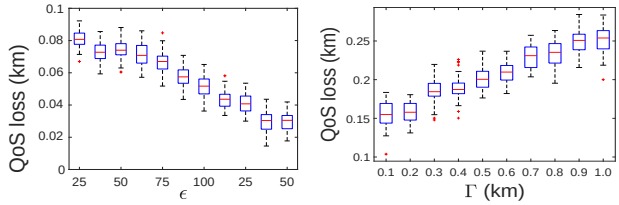
Figure 11(a) compares the EIE of 2Layer, Laplace, and LPO in different traces (without loss of generality, we sort the vehicle traces according to the EIE achieved by 2Layer). The figure demonstrates that 2Layer outperforms both Laplace and LPO in term of privacy level: On average, the EIE of 2Layer is 547.17% and 499.89% higher than that of Laplace and LPO, respectively. As analyzed in Section 4, the higher privacy level achieved by 2Layer is due to its two layer obfuscation framework, particularly the obfuscation by a fake trajectory in Layer 1 that is hard to distinguish from real trajectories by HMM. Moreover, we depict the box plot of the three algorithms' EIE in Figure 11(a) to show the variability of their EIE. According to the box plot, we can see that 2Layer has higher EIE variance than Bayes, as 2Layer has the second layer obfuscation that introduces additional uncertainty to the vehicles' reported locations.

Figure 11(b) compares the quality loss of the three algorithms in different traces (without loss of generality, we sort the vehicles based on the quality loss achieved by 2Layer). The figure shows that Laplace > 2Layer ≈ LPO in terms of quality loss. On average, the quality loss caused by 2Layer is 26.48% lower than that of Laplace, and 8.25% higher than that of LPO. Laplace has higher quality loss than both HMM and LPO as it simply considers users' mobility on a 2D plane, in which the sensitivity of quality loss to location obfuscation is different than in a road network. In particular, due to the restriction of the road network, a small location deviation on the 2D plane may lead to a significant different estimated traveling distance over roads (e.g., think about when a vehicle has to take a detour to reach a nearby destination). Therefore, obfuscated



(a) Varying ϵ (b) Varying Γ

Figure 11: EIE of HMM given different ϵ and Γ.



(a) Varying ϵ (b) Varying Γ

Figure 12: Quality loss of HMM given different ϵ and Γ.

locations generated by Laplace is more likely to generate high traveling cost. Although LPO has a slightly lower quality loss than 2Layer, it minimizes the quality loss at the expense of privacy, as demonstrated in Figure 11(a).

Next, we evaluate the impact of the two parameters ϵ and Γ on HMM's accuracy (recall that, in Equation (12), Γ is a threshold limiting the quality loss). Figure 11(a)(b) show the change of HMM's EIE with ϵ increased from 50 to 100, and Γ increased from 0.1km to 1.0km, respectively. From the two figures, we observe that: HMM's EIE (1) decreases with an increase of ϵ, and (2) increases with an increase of Γ. For (1), it is because higher ϵ requires smaller distortion between vehicles' obfuscated and real locations, making vehicles more susceptible to the inference attack. For (2), since higher Γ allows larger deviation from real locations to obfuscated locations, EIE can be achieved at a higher level.

Finally, we test how the two parameters ϵ and Γ impact the quality loss of HMM. Figure 12(a)(b) show the change of HMM's EIE given different ϵ and Γ. The figures imply that the quality loss of HMM (1) decreases with the increase of ϵ and (2) increases with the increase of Γ. By comparing Figure 12(a)(b) with Figure 11(a)(b), we can also find that lower ϵ (higher privacy level) and higher Γ allow more space to select obfuscated locations for vehicles, but on the other hand, are likely to introduce more errors to service (e.g., task assignment/recommendation), leading to a higher quality loss. In fact, there is a trade-off between privacy and QoS. Therefore, in real applications, it is of great importance to balance these two metrics based on users' preferences.

6 RELATED WORK

In this section, we summarize the existing works that are most relevant to ours:

Sporadic location privacy model. In fact, the discussion of location privacy criteria can date back to more than ten years ago, when Gruteser and Grunwald [31] first introduced the notion of *location k-anonymity* on the basis of Sweeney's well-known concept of *k-anonymity* for data privacy [32]. Location-based k-anonymity was originally used to hide a user's identity in a location-based service [33]. This notion has been extended to obfuscate users' location by means of *l-diversity*, i.e., a user's location cannot be distinguished with other $l-1$ dummy locations [12, 34], which creates a cloaking area that includes l locations sharing some property of interest. However, l-diversity is hard to achieve in many real applications as it assumes dummy locations are equally likely to be the real location from the attacker's point of view [8, 12].

In recent years, two practical privacy notions have been proposed based on statistical quantification of attack resilience, e.g., *expected inference error (EIE)* [17, 35], and *geo-indistinguishability* (*GI*) [8], i.e., if two location points are geographically close, their obfuscated locations will be generated with a similar probability distribution. On the basis of EIE and GI, a large body of location obfuscation strategies have been proposed to achieve either of these two privacy criteria (e.g., [8–10, 17, 35–37]) or their combination [12]. As location obfuscation based methods inevitably introduce errors to users' reported locations, which may lead to quality loss in LBS. Therefore, a key issue has been discussed in those works is how to trade-off QoS and privacy. In particular, many existing works follow a global optimization framework: given the privacy (measured by EIE/GI) or QoS constraints, a math optimization model is formulated to maximize QoS or privacy respectively [6, 10, 13, 17].

Yet, both privacy notions have their own limitations: EIE based approaches assume certain types of prior information that the attacker may obtain, but require no restriction on the posterior information gain from the exposure of obfuscated locations. GI-based approaches limit the posterior information leakage through a *differential privacy (DP)* based criteria, but they are susceptible to inference attacks based on attackers' prior knowledge [12]. As such, recent works (e.g., [12]) have proposed to strategically combine the two privacy notions to double shield users' location privacy. *More importantly, both EIE and GI are based on isolated obfuscated location with no consideration of spatial correlation between reported (obfuscated) locations from mobile users.* Such sporadic location privacy criteria are insufficient for applications in real-world settings, leaving severe concerns on the actual robustness of these methods.

Spatiotemporal location privacy model. A variety of privacy protection location/trajectory inference algorithms have been proposed to date. These algorithms focus on spatiotemporal correlation of users' reported locations, either from a single user at different time points (e.g., trajectory) [38–43] or from multiple users [44]. These efforts are based on the assumption that users' mobility follows a Markov process [38, 42], i.e., users' current locations depend on the locations attained in the previous round. For instance, Liao *et. al* [38] applied a hierarchical Markov model to learn and infer a mobile user's trajectory based on the places and temporal patterns the user visited. Given uncertain locations of moving objects, Emrich *et. al* [42] proposed a modified matrix computation method to efficiently compute the probability of a user appearing in certain region during certain time period. In addition to the above, some inference algorithms focus on other statistical features of users' mobility (e.g., visiting frequency at certain interests of points [45] or users' location spatial similarity [46]).

In recent years, research works closer to ours have proposed privacy criteria and solutions that account for statistical features of users' location data [40, 44, 47, 48]. For example, under the assumption that attackers use Markov models to describe users' mobility,

Cao *et al.* [40] defined a criterion to quantify the privacy level that existing methods can achieve. Furthermore, Cao et al. [44] extended the notion of DP to a new criteria to protect *spatiotemporal event privacy* and provided a framework to calculate the privacy loss of a given location privacy protection mechanism. By counting for the temporal correlations in location data, Xiao et al. [48] proposed a new definition, called δ-location set based DP, and presented a planar isotropic mechanism for location obfuscation. *While elegant, all these formulations are designed for general moving objects without considering vehicles' mobility features over roads, i.e., road network topology, traffic regulations, and many other road network environments. Therefore, they fall short in regard to their applicability in vehicles' location privacy protection.*

7 CONCLUSIONS

In this paper, by leveraging vehicle traffic information, we have developed a new spatial correlation aware inference attack that can accurately recover vehicles' trajectories from their obfuscated locations. As a countermeasure, we have proposed a two-layer location obfuscation algorithm to protect the location privacy of vehicles under the new threat model. The trace-driven simulation results have shown the current obfuscation algorithms are insufficient to address the vulnerability of vehicles when the new inference algorithm is applied. Furthermore, the experimental results have demonstrated the effectiveness of our approach over the state of the arts in terms of both privacy and QoS.

We can see several promising directions to continue this research. First, our current work accounts only for a single vehicle, without considering the spatial correlations between multiple vehicles. Also, this work can be extended to general LBS applications, where service utilities can be defined in different ways. Moreover, we will consider different threat models where the information disclosed to attackers is in different formats (e.g., smartphones' accelerometer and gyroscope).

REFERENCES

[1] M. Amin-Naseri and et al. Evaluating the reliability, coverage, and added value of crowdsourced traffic incident reports from waze. *Transportation Research Record*, 2672(43):34–43, 2018.
[2] MediaQ. https://imsc.usc.edu/platforms/mediaq/, 2019. Accessed: 2019-07-22.
[3] Y. Tong, L. Chen, and C. Shahabi. Spatial crowdsourcing: Challenges, techniques, and applications. *VLDB Endow.*, 10(12):1988–1991, August 2017.
[4] H. To, G. Ghinita, L. Fan, and C. Shahabi. Differentially private location protection for worker datasets in spatial crowdsourcing. *IEEE TMC*, pages 934–949, 2017.
[5] B. Liu, L. Chen, X. Zhu, Y. Zhang, C. Zhang, and W. Qiu. Protecting location privacy in spatial crowdsourcing using encrypted data. In *Proc. of EDBT*, 2017.
[6] K. Fawaz, H. Feng, and K. G. Shin. Anatomization and protection of mobile apps' location privacy threats. In *Proc. of USENIX Association*, pages 753–768, 2015.
[7] G. Ghinita and et al. Private queries in location based services: Anonymizers are not necessary. In *Proc. of ACM SIGMOD*, pages 121–132, 2008.
[8] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. In *Proc. of ACM CCS*, pages 901–914, 2013.
[9] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi. Optimal geo-indistinguishable mechanisms for location privacy. In *Proc. of ACM CCS*, pages 251–262, 2014.
[10] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma. Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation. In *Proc. of ACM WWW*, pages 627–636, 2017.
[11] C. Qiu and A. C. Squicciarini. Location privacy protection in vehicle-based spatial crowdsourcing via geo-indistinguishability. In *Proc. of IEEE ICDCS*, pages 1061–1071, 2019.
[12] L. Yu, L. Liu, and C. Pu. Dynamic differential location privacy with personalized error bounds. In *Proc. of IEEE NDSS*, 2017.
[13] K. Fawaz and K. G. Shin. Location privacy protection for smartphone users. In *Proc. of ACM CCS*, pages 239–250, New York, NY, USA, 2014. ACM.
[14] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *Proc. of IEEE ICDCS*, pages 620–629, June 2005.
[15] H. To, G. Ghinita, and C. Shahabi. A framework for protecting worker location privacy in spatial crowdsourcing. *VLDB Endow.*, 7(10):919–930, June 2014.
[16] H. To, C. Shahabi, and L. Xiong. Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In *Proc. of IEEE ICDE*, 2018.
[17] R. Shokri, G. Theodorakopoulos, C. Troncoso, J. Hubaux, and J. L. Boudec. Protecting location privacy: Optimal strategy against localization attacks. In *Proc. of ACM CCS*, pages 617–627, 2012.
[18] L. Yan, H. Shen, J. Zhao, C. Xu, F. Luo, and C. Qiu. Catcharger: Deploying wireless charging lanes in a metropolitan road network through categorization and clustering of vehicle traffic. In *Proc. of IEEE INFOCOM*, pages 1–9, 2017.
[19] G. D. Forney. The viterbi algorithm. *Proc. of the IEEE*, 61(3):268–278, 1973.
[20] Yanhua Li, Jun Luo, Chi-Yin Chow, Kam-Lam Chan, Ye Ding, and Fan Zhang. Growing the charging station network for electric vehicles with trajectory data analytics. In *Proc. of ICDE*, 2015.
[21] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, (8):101–111, 1960.
[22] C. Qiu, A. C. Squicciarini, and S. M. Rajtmajer. Rating mechanisms for sustainability of crowd-sourcing platforms. In *Proc. of ACM International Conference on Information and KnowledgeManagement (CIKM)*, 2019.
[23] Wang T. and Hu J. Applying floating car data in traffic monitoring. In *2014 IEEE ICCSSE*, pages 96–99, 2014.
[24] Y. Zhu, Z. Li, H. Zhu, M. Li, and Q. Zhang. A compressive sensing approach to urban traffic estimation with probe vehicles. *IEEE TMC*, 12:2289–2302, 11 2013.
[25] Harsh Bhasin. *Algorithms: Design and Analysis.* Oxford Univ Press, 2015.
[26] Frederick S. Hillier. *Linear and Nonlinear Programming.* Stanford University, 2008.
[27] D. P. Palomar and Mung Chiang. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications*, 24(8):1439–1451, Aug 2006.
[28] openstreetmap. https://www.openstreetmap.org/, 2020. Accessed: 2020-04-07.
[29] Apache Hadoop 2.7.3. hadoop.apache.org, 2020. Accessed in February, 2020.
[30] Apache Spark 1.5.2. spark.apache.org, 2020. Accessed in February, 2020.
[31] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of ACM MobiSys*, 2003.
[32] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, 2002.
[33] L. Zheng, H. Yue, Z. Li, X. Pan, M. Wu, and F. Yang. k-anonymity location privacy algorithm based on clustering. *IEEE Access*, 2018.
[34] T. Wang and L. Liu. Privacy-aware mobile services over road networks. *VLDB Endow.*, 2(1):1042–1053, August 2009.
[35] R. Shokri, G. Theodorakopoulos, J. Y. Le Boudec, and J. P. Hubaux. Quantifying location privacy. In *2011 IEEE SP*, pages 247–262, 2011.
[36] C. Ardagna, M. Cremonini, S. Vimercati, and P. Samarati. An obfuscation-based approach for protecting location privacy. *IEEE TDSC*, 8:13 – 27, 03 2011.
[37] R. Shokri, G. Theodorakopoulos, J. Le Boudec, and J. Hubaux. Quantifying location privacy. In *Proc. of 2011 IEEE SP*, pages 247–262, May 2011.
[38] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5):311 – 331, 2007.
[39] F. Xu, Z. Tu, Y. Li, P. Zhang, X. Fu, and D. Jin. Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data. In *Proc. of ACM WWW*, page 1241–1250, 2017.
[40] Y. Cao, M. Yoshikawa, Y. Xiao, and L. Xiong. Quantifying differential privacy under temporal correlations. In *Proc. of IEEE ICDE*, pages 821–832, 2017.
[41] Z. Li, B. Ding, J. Han, and R. Kays. Swarm: Mining relaxed temporal moving object clusters. *Proc. VLDB Endow.*, 3(1–2):723–734, September 2010.
[42] T. Emrich, H. Kriegel, N. Mamoulis, M. Renz, and A. Zufle. Querying uncertain spatio-temporal data. In *Proc. of IEEE ICDE*, pages 354–365, 2012.
[43] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, and W. Ma. Mining user similarity based on location history. In *Proc. of SIGSPATIAL*, 2008.
[44] Y. Cao, Y. Xiao, L. Xiong, and L. Bai. Priste: From location privacy to spatiotemporal event privacy. In *Proc. of IEEE ICDE*, pages 1606–1609, 2019.
[45] A. Korkmaz, F. Elik, F. Aydin, M. Bulut, S. Kul, and A. Sayar. Modeling trajectory data as a directed graph. *MIKE*, pages 168–176, 2018.
[46] W. Li, H. Chen, W. Ku, and X. Qin. Scalable spatiotemporal crowdsourcing for smart cities based on particle filtering. In *Proc. of ACM SIGSPATIAL*, 2017.
[47] G. Ghinita, M. L. Damiani, C. Silvestri, and E. Bertino. Preventing velocity-based linkage attacks in location-aware applications. In *Proc. of ACM SIGSPATIAL*, page 246–255, 2009.
[48] Y. Xiao and L. Xiong. Protecting locations with differential privacy under temporal correlations. In *Proc. of CCS*, page 1298–1309, 2015.
[49] J. Puchinger, P. Stuckey, M. Wallace, and S. Brand. Dantzig-wolfe decomposition and branch-and-price solving in g12. *Constraints*, 16(1):77–99, Jan 2011.
[50] N. Maculan, M. Passini, B. Moura, and I. Loiseau. Column-generation in integer linear programming. *RAIRO*, 37(2):67–83, 2003.

APPENDIX

A. The pseudo code of trajectory maintenance

Algorithm 1 provides the pseudo-code of the trajectory maintenance algorithm by FTM in each round t:

Algorithm 1: Fake trajectory manager.

Input : t, $\mathbf{q}^1$, ..., $\mathbf{q}^{t-1}$, s_r^t
Output: $\mathbf{q}^t$

1 $\mathbf{q}^t$ is initialized by empty;
2 **if** *t equals 1* **then**
3 `// Initialization in the first round`
4 Randomly pick up M locations around the current real location s_r^t: $s_{f_1^t}$, ..., $s_{f_M^t}$;
5 **for** *each* $s_{f_m^t}$ $(m = 1, ..., M)$ **do**
6 Calculate $s_{f_m^t}$'s fitness value w_m^t using Equation (5);
7 $\mathbf{q}^t$.push$\left(\left(s_{f_m^t}, \phi, w_m^t\right)\right)$;
8 **else**
9 Initialize the location set $\mathcal{R}^t$ by an empty set;
10 `// S1: Production`
11 **for** *each* $\left(s_{f_m^{t-1}}, s_{f_{m'}^{t-1}}, w_m^{t-1}\right)$ *in* $\mathbf{q}^{t-1}$ **do**
12 **for** *each* $s_j \in \mathcal{S}$ *with* $p_{f_m^{t-1},j}^{t-1,t} > 0$ **do**
13 Add s_j to $\mathcal{R}^t$;
14 `// S2: Selection`
15 **for** $\forall s_j \in \mathcal{R}^t$ **do**
16 Calculate s_j's fitness value w_j^t using Equation (5);
17 **if** $\left|\mathbf{q}^t\right| < M$ **then**
18 `// q^t is not full`
19 $\mathbf{q}^t$.push$\left(\left(s_{f_{j'}^{t-1}}, s_{f_j}, w_j^t\right)\right)$; `// s_{f_{j'}^{t-1}} is s_j's previous location`
20 **else**
21 **if** $\mathbf{q}^t$.top() *has higher fitness value than* w_j^t **then**
22 $\mathbf{q}^t$.pop();
23 $\mathbf{q}^t$.push$\left(\left(s_{f_{j'}^{t-1}}, s_{f_j}, w_j^t\right)\right)$;
24 **return** $\mathbf{q}^t$;

At the beginning of each round t, the min heap $\mathbf{q}^t$ is initialized by empty (line 1). If it is the first round (line 2–7), FTM randomly picks up M locations around the current true location: $s_{f_1^t}$, ..., $s_{f_M^t}$ (line 4), calculates their fitness values w_1^t, ..., w_M^t (line 6), and pushes the corresponding nodes $\left(s_{f_m^t}, \phi, w_m^t\right)$ $(m = 1, ..., M)$ sequentially onto $\mathbf{q}^t$ (line 7).

After the first round, FTM needs to carry out the following two steps:

S1: *Production* (line 9–13): FTM initialize an empty location set $\mathcal{R}^t$ (line 9). Given the locations of fake trajectories in the last round, which can be found in $\mathbf{q}^{t-1}$ (line 11), FTM finds a set of locations that are reachable (line 12) and add them in $\mathcal{R}^t$ (line 13);

S2: *Selection* (line 14–23): For each location $s_j \in \mathcal{R}^t$, FTM calculates its fitness value w_j^t (line 16). The first M nodes are pushed directly onto $\mathbf{q}^t$ (line 19). Once $\mathbf{q}^t$ reaches its capacity, i.e., $|\mathbf{q}^t| = M$, FTM first checks whether the top node in $\mathbf{q}^t$ has higher fitness value than w_j^t (line 21): If yes, then FTM pops the top node off $\mathbf{q}^t$ (line 22) and pushes the new node $\left(s_{f_{j'}^{t-1}}, s_{f_j}, w_j^t\right)$ onto $\mathbf{q}^t$ (line 23).

B. Proof of Proposition 4.1

Proof. According to the triangle inequality,

$$\begin{aligned} Q\left(s_{o^t}, s_{r^t}\right) &= \sum_q \mathbb{P}\mathrm{r}\left(Q = s_q\right)\left|C\left(s_{r^t}, s_q\right) - C\left(s_{o^t}, s_q\right)\right| \\ &\le \sum_q \mathbb{P}\mathrm{r}\left(Q = s_q\right)\left|C\left(s_{f^t}, s_q\right) - C\left(s_{o^t}, s_q\right)\right| \\ &+ \underbrace{\sum_q \mathbb{P}\mathrm{r}\left(Q = s_q\right)\left|C\left(s_{r^t}, s_q\right) - C\left(s_{f^t}, s_q\right)\right|}_{\le \tilde{\Gamma} \text{ according to Equation (7)}} \\ &\le Q\left(s_{o^t}, s_{f^t}\right) + \tilde{\Gamma}. \end{aligned}$$

Accordingly, when the constraint in Equation (13) is satisfied (i.e., $z_{f^t,o^t}^t \le 0$, when $Q\left(s_{o^t}, s_{f^t}\right) > \Gamma - \tilde{\Gamma}$), we have

$$\mathbb{E}\left(Q\left(s_{o^t}, s_{r^t}\right)\right) = \sum_{r^t,o^t} \mathbb{P}\mathrm{r}\left(X^t = s_{r^t}, Y^t = s_{o^t}\right) Q\left(s_{o^t}, s_{r^t}\right) \quad (17)$$

$$\le \tilde{\Gamma} + \sum_{r^t,o^t \text{ with } Q(s_{o^t},s_{r^t}) \le \Gamma - \tilde{\Gamma}} \mathbb{P}\mathrm{r}\left(X^t = s_{r^t}, Y^t = s_{o^t}\right) Q\left(s_{o^t}, s_{f^t}\right)$$

$$\le \tilde{\Gamma} + \Gamma - \tilde{\Gamma} = \Gamma, \quad (18)$$

indicating that the constraint in Equation (12) is also satisfied. The proof is completed. □

C. The DW decomposition

DW formulation. We let $\mathcal{Z}_l = \left\{\hat{\mathbf{z}}_l^1, ..., \hat{\mathbf{z}}_l^{T_l}\right\}$ denote the set of extreme points of Φ_l (recall Φ_l is the polyhedron that constraints the decistion vector $\mathbf{z}_l$). According to Minkowski-Weyl's Theorem [26], $\mathbf{z}_l \in \Phi_l$ can be represented as a convex combination of $\hat{\mathbf{z}}_l^1, ..., \hat{\mathbf{z}}_l^{L_l}$:

$$\mathbf{z}_l = \sum_{t=1}^{T_l} \lambda_{l,t} \hat{\mathbf{z}}_l^t, \quad (19)$$

where $\sum_{t=1}^{T_l} \lambda_{l,t} = 1, \lambda_{l,t} \ge 0$. Based on Equation (19), the original OFG can be rewritten as:

$$\max \quad \sum_l \sum_t \lambda_{l,t} \hat{z}_{K+1,l}^t \quad (20)$$

$$\text{s.t.} \quad \sum_l \sum_t \lambda_{l,t} \hat{z}_{k,l}^t = 1, \forall k, \quad (21)$$

$$\sum_{t=1}^{T_l} \lambda_{l,t} = 1, \lambda_{l,t} \ge 0, \forall l \quad (22)$$

called the *master program (MP)*. Here, $\lambda_{l,t}$ ($t = 1, ..., T_l, l = 1, ..., K$) are the decision variables in MP, where each $\lambda_{l,t}$ corresponds to an extreme point in $\mathbf{\Phi}_l$.

According to [49], most extreme points in DW formulation are non-basic (i.e., they won't be visited during the whole search process), indicating that MP formulated in Equation (20)-(22) can be solved by involving only a small portion of extreme points. As such, we can adopt the *column generation* algorithm [50] to solve MP:
(I) The algorithm starts by formulating a *restricted MP (RMP)* (Definition 7.1), where only a subset of extreme points in MP are included.
(II) In the algorithm, we solve the RMP and test the optimality of its solution by solving its dual problem D-RMP (definition 7.2). If the optimal of MP hasn't been reached, a new extreme point is added to the RMP to improve the objective value. This process is repeated until MP's optimal is found.

The definitions and proposition used in the DW decomposition are introduced in Appendix D.

The algorithm details. Algorithm 2 gives the details of the column generation algorithm. The superscript $^{(n)}$ denotes the values set/derived in iteration n.

Algorithm 2: The column generation algorithm.

input : $\Lambda_1, ..., \Lambda_K$
output: The optimal solution of MP

1 $n \leftarrow 1$; // Index of iteration
2 **do**
3 // Layer 1: Master program
4 $(\overline{\boldsymbol{\pi}}^{*(n)}, \overline{\boldsymbol{\mu}}^{*(n)}) \leftarrow$ D-RMP $(\overline{\mathcal{Z}}_1^{(n)}, ..., \overline{\mathcal{Z}}_K^{(n)})$;
5 // Layer 2: Subproblems
6 **for** *each* $l = 1, ..., K$ **do**
7 $\left(\zeta_l^{(n)}, \hat{\mathbf{z}}_l^{(n)}\right) \leftarrow \text{sub}_l\left(\overline{\boldsymbol{\pi}}^{*(n)}, \overline{\boldsymbol{\mu}}^{*(n)}; \boldsymbol{\lambda}\right)$;
8 **if** $\zeta_l^{(n)} < 0$ **then**
9 $\overline{\mathcal{Z}}_l^{(n+1)} \leftarrow \overline{\mathcal{Z}}_l^{(n)} \cup \hat{\mathbf{z}}_l^{(n)}$;
10 $n \leftarrow n + 1$;
11 **while** $\min_l \left\{\zeta_l^{(n-1)}\right\} < -\xi$;
12 $\overline{\boldsymbol{\lambda}}^{*(n)} \leftarrow$ RMP $(\overline{\mathcal{Z}}_1^{(n)}, ..., \overline{\mathcal{Z}}_K^{(n)})$;
13 **return** $\overline{\boldsymbol{\lambda}}^{*(n)}$;

The algorithm is composed of two layers: master program and subproblems. The master program (line 3-4) and the subproblems (line 5-9) deliver their results to each other until a near-optimal solution is achieved, i.e., when each $\zeta_l^{(n)} \geq \xi$, where ξ ($\xi \leq 0$) is a predefined threshold.

Layer1: The optimal $(\overline{\boldsymbol{\pi}}^{*(n)}, \overline{\boldsymbol{\mu}}^{*(n)})$ in D-RMP is derived and delivered to each sub_l (line 4).

Layer2: Each sub_l calculates the optimal $\left(\zeta_l^{(n)}, \hat{\mathbf{z}}_l^{(n)}\right)$ (line 7). If exists $\zeta_l^{(n)} < 0$, the optimal solution hasn't been found (by Proposition 7.3). An extreme point $\hat{\mathbf{z}}_l^{(n)}$ is added to $\overline{\mathcal{Z}}_l^{(n)}$ to improve the solution (line 9). The results calculated by subproblems are then sent back to the master program.

Finally, the optimal solution $\overline{\boldsymbol{\lambda}}^{*(n)}$ is derived in RMP (line 12). [26].

D. Definitions and Propositions used in DW Decomposition

Definition 7.1. (*Restricted master program (RMP)*) Given a subset of extreme points $\overline{\mathcal{Z}}_l$ ($\overline{\mathcal{Z}}_l \subseteq \mathcal{Z}_l$) in each $\mathbf{\Phi}_l$, we define the corresponding *restricted master program*, denoted by RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$), as the MP with only extreme points $\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$ considered:

$$\max \quad \sum_l \sum_{t \in \overline{\mathcal{Z}}_l} \lambda_{l,t} \hat{z}_{K+1,l}^t \tag{23}$$

$$\text{s.t.} \quad \sum_l \sum_{t \in \overline{\mathcal{Z}}_l} \lambda_{l,t} \hat{z}_{k,l}^t = 1, \forall k, \tag{24}$$

$$\sum_{t=1}^{T_l} \lambda_{l,t} = 1, \lambda_{l,t} \geq 0, \forall l \tag{25}$$

and we let $\overline{\boldsymbol{\lambda}}^*$ denotes the optimal solution of RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$).

Definition 7.2. (D-RMP) The dual problem of RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$) [26], denoted by D-RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$), is defined as:

$$\min \quad \sum_k \pi_k + \sum_l \mu_l \tag{26}$$

$$\text{s.t.} \quad \sum_k \hat{x}_{k,l}^t \pi_k + \mu_l \geq \hat{z}_{K+1,l}^t, \forall t \in \overline{\mathcal{Z}}_l,\ l = 1, ..., K. \tag{27}$$

where $(\overline{\boldsymbol{\pi}}^*, \overline{\boldsymbol{\mu}}^*)$ ($\overline{\boldsymbol{\pi}}^* = [\overline{\pi}_1^*, ..., \overline{\pi}_K^*]$ and $\overline{\boldsymbol{\mu}}^* = [\overline{\mu}_1^*, ..., \overline{\mu}_K^*]$) denote the optimal solution of D-RMP($\overline{\mathcal{Z}}_1, ..., \overline{\mathcal{Z}}_K$).

Proposition 7.3. *(Optimality test criteria) To test $\overline{\boldsymbol{\lambda}}^*$'s optimality in MP, it is sufficient to test whether $(\boldsymbol{\pi}^*, \overline{\boldsymbol{\mu}}^*)$ satisfies*

$$\min_{t \in \mathcal{Z}_l} \left\{\sum_k \hat{z}_{k,l}^t \overline{\pi}_k^* + \overline{\mu}_l^* - \hat{z}_{K+1,l}^t\right\} \geq 0,\ l = 1, ..., K, \tag{28}$$

where the derivation of

$$\min_{t \in \mathcal{Z}_l} \left\{\sum_k \hat{z}_{k,l}^t \pi_k + \mu_l - \hat{z}_{K+1,l}^t\right\} \tag{29}$$

is essentially a LP problem (labeled by sub_l) with the decision variables $\mathbf{z}_l$ constrained in the polyhedron $\boldsymbol{\lambda}$:

$$\min \quad \sum_k z_{k,l} \overline{\pi}_k^* + \overline{\mu}_l^* - z_{K+1,l} \tag{30}$$

$$\text{s.t.} \quad \mathbf{z}_l \in \mathbf{\Phi}_l. \tag{31}$$

Proof. First, $\boldsymbol{\lambda}^*$ achieves the optimal of MP if only if $(\overline{\boldsymbol{\pi}}^*, \overline{\boldsymbol{\mu}}^*)$ is a feasible solution of the dual problem of MP, D-RMP($\mathcal{Z}_1, ..., \mathcal{Z}_K$) [26]. As such, to test $\overline{\boldsymbol{\lambda}}^*$'s optimality, it is sufficient to test whether $(\boldsymbol{\pi}^*, \overline{\boldsymbol{\mu}}^*)$ satisfies the following condition:

$$\min_{t \in \mathcal{Z}_l} \left\{\sum_k \hat{z}_{k,l}^t \overline{\pi}_k^* + \overline{\mu}_l^* - \hat{z}_{K+1,l}^t\right\} \geq 0,\ l = 1, ..., K. \tag{32}$$

where the derivation of $\min_{t \in \mathcal{Z}_l} \left\{\sum_k \hat{z}_{k,l}^t \pi_k + \mu_l - \hat{z}_{K+1,l}^t\right\}$ is essentially the problem sub_l (defined in Equation (30)–(31)). □