DPT: Differentially Private Trajectory Synthesis Using Hierarchical Reference Systems

Xi He
Duke University
hexi88@cs.duke.edu

Graham Cormode
University of Warwick
g.cormode@warwick.ac.uk

Ashwin Machanavajjhala
Duke University
ashwin@cs.duke.edu

Cecilia Procopiuc
Google Inc.
mpro@google.com

Divesh Srivastava
AT&T Labs-Research
divesh@research.att.com

ABSTRACT
GPS-enabled devices are now ubiquitous, from airplanes and cars to smartphones and wearable technology. This has resulted in a wealth of data about the movements of individuals and populations, which can be analyzed for useful information to aid in city and traffic planning, disaster preparedness and so on. However, the places that people go can disclose extremely sensitive information about them, and thus their use needs to be filtered through privacy preserving mechanisms. This turns out to be a highly challenging task: raw trajectories are highly detailed, and typically no pair is alike. Previous attempts fail either to provide adequate privacy protection, or to remain sufficiently faithful to the original behavior.

This paper presents DPT, a system to synthesize mobility data based on raw GPS trajectories of individuals while ensuring strong privacy protection in the form of \( \varepsilon \)-differential privacy. DPT makes a number of novel modeling and algorithmic contributions including (i) discretization of raw trajectories using hierarchical reference systems (at multiple resolutions) to capture individual movements at differing speeds, (ii) adaptive mechanisms to select a small set of reference systems and construct prefix tree counts privately, and (iii) use of direction-weighted sampling for improved utility. While there have been prior attempts to solve the subproblems required to generate synthetic trajectories, to the best of our knowledge, ours is the first system that provides an end-to-end solution. We show the efficacy of our synthetic trajectory generation system using an extensive empirical evaluation.

1. INTRODUCTION
The ubiquity of sensor-enabled devices (like smartphones, wearable technology, etc.) has significantly advanced our capabilities in real-time data collection. In particular, the popularity of location sensing technology (like GPS) and the rising popularity of location-based applications has resulted in a wealth of data about the movements of individuals and populations. This in turn has sparked renewed interest in studying large-scale human mobility patterns [3] with applications in city/traffic planning, epidemiology and location-driven advertising.

Despite being valuable for research, organizations that collect detailed trajectories are wary about publicly releasing them due to concerns over the privacy of individuals therein. Home and work locations can be easy to discern (based on the frequency with which locations are visited). Additionally, these trajectories may also reveal the health status of individuals (specialties of, and frequency of visits to, clinics), their habitual vices (visits to casinos, alcohol and tobacco shops), and their truthfulness (actual vs. claimed locations). However, anonymizing such data is hard – merely removing identifiers, or coarsening the data in space and time are not enough. A recent study [10] of mobility data of 1.5 million individuals over a period of fifteen months showed that raw trajectories are highly identifiable – 4 spatio-temporal points are enough to uniquely identify 95% of the individuals; further, the study showed that the uniqueness of human mobility trajectories decays significantly as their spatial and temporal resolution coarsens.

What is needed is a principled approach to generate a database of synthetic trajectories, with the provable privacy guarantees of differential privacy [11], while ensuring high utility in practice – i.e., the synthetic trajectories have similar aggregate properties as the original trajectories. In this paper, we present Differentially Private Trajectories (DPT), our system for generating such synthetic trajectories. As in prior work in this area (e.g., [6]), DPT considers regular trajectories, or sequences of locations that are sampled at uniform time intervals. However, in a significant departure from prior work, DPT uses a novel hierarchical reference system set (RSS) model, rather than the straightforward Markov model, to parsimoniously capture correlations between adjacent locations in regular trajectories. While previous work [6] was shown to adequately deal with uniform speed trajectories in small spatial domains, it can result in a considerable loss in utility on trajectories with varying speeds over time even in moderately large spatial domains, such as the movement of taxi cabs in a large metropolitan area [1]. In contrast, the RSS model is designed to naturally deal with such realistic trajectories using a set of hierarchically organized reference systems: movement at slow speeds can be summarized using a fine granularity reference system, while movements at progressively higher speeds are summarized using coarser granularity reference systems. To guarantee differential privacy, DPT uses the standard Laplace mechanism to add noise to counts maintained in each reference system; the novelty of DPT is in the model selection strategy used to build a useful differentially private generative model, and in a direction weighted sampling technique to produce high utility synthetic trajectories from the differentially
private model. Consequently, we maintain that DPT is the first method that can handle realistic trajectories of millions of individuals over large spatial domains to generate useful trajectories with differential privacy guarantees.

Solution Overview. The input to our system is a database of regular trajectories, or sequences of (latitude, longitude) pairs sampled at some uniform rate, corresponding to the movements of a set of individuals. Synthetic trajectories are generated by first fitting a probabilistic model on the data from which new trajectories can be sampled. A popular choice of model for sequential data is the order k Markov process that generates the next location of trajectory based on the previous k locations [16]. The sufficient statistics of this model are given by a prefix tree of depth k + 1, where nodes at the i-th level of the tree record the number of times sequences of i consecutive locations corresponding to the path from the root of the tree up to the node at level i appear in the database. Recent work has proposed a differentially private approach to fit a Markov model by carefully adding noise drawn from a Laplace distribution to parts of the prefix tree [6]. This approach was shown to work well when the domain of locations is small, and can be applied to continuous spatial domains by discretizing locations (e.g., via a uniform coarse grid).

However, the aforementioned approach fails to scale to realistic trajectory databases that span even moderately large geographical regions. Note that the size of the prefix tree is $O(|\Sigma|^{k+1})$, where $\Sigma$ is the domain of possible locations. A sufficiently fine discretization of the spatial domain that captures all the mobility patterns in the data could result in very large domain sizes (of several tens of thousands) making the fitting procedure not only very slow, but also overfitting the data. As the number of nodes in the tree increases, the amount of noise added to ensure differential privacy also grows. On the other hand, while a coarse discretization of the space results in a small prefix tree, much of the spatial correlation information in the original trajectories is lost.

This is quantified in Figure 1. We considered the movement of taxi cabs in a large metropolitan area [11] spanning a geographical region of roughly 34km $\times$ 40km. We discretized this space using uniform grids with varying resolutions of 200m, 400m, 800m, 1600m and 3200m. The number of cells in the discretized domain under the 200m grid is 34000 while the size under the 3200m grid is less than 150. We synthesized trajectories under each of these discretizations using the method of Chen et al. [6] (which we call Seq), and measured the Jensen Shannon divergence (JSD) between the distribution of the diameter, i.e., the distance traveled by trajectories, in the original database to that of the trajectories in the synthetic database. In all cases, the divergence is in the range $[0.33, 0.59]$ (a high value considering that the maximum possible JSD is $\ln(2) = 0.69$). Furthermore, for an input of over 4 million trajectories, the time taken to compute the relevant prefix trees and then reconstruct a synthetic trajectory database is at least 3 hours!

Our system Differentially Private Trajectories (DPT) is a scalable end-to-end system for synthesizing regular trajectories with provable privacy properties. DPT can synthesize trajectories spanning large geographical areas with significantly more utility (JSD error on the diameter of less than 0.05) than Chen et al.’s method and is orders of magnitude faster (fitting the model and sampling trajectories take on the order of 100 seconds for an input of over 4 million trajectories). Figure 2 gives a schematic overview of the steps of DPT, which incorporates the following key novel ideas:

Hierarchical Reference Systems: DPT discretizes the spatial domain at multiple resolutions (see Step 1 in Figure 2) using a hierarchy of reference systems and maintains one prefix tree for each resolution (Step 2). Different reference systems capture movements at different speeds. Within each reference system individuals are restricted to move from each point to only a small number of neighboring points in one step. Thus, while there are a larger number of prefix trees, each tree has a much smaller branching factor resulting in an exponential reduction in the number of counts maintained by the model.

Model Selection: The set of reference systems (corresponding to resolutions at which the spatial domain is discretized) for which prefix trees are materialized, and the depth of the prefix trees are hard to set without looking at the trajectory database. Materializing trees with too large a depth or at resolutions that do not correspond to viable speeds can significantly hurt the accuracy of the system. Hence, DPT uses a novel model selection algorithm (Step 3) to set these parameters in a differentially private manner. As in prior work, noise drawn from the Laplace distribution is added to the chosen prefix trees (Step 4), and these noisy trees are pruned using an adaptive strategy (Step 5) to further improve utility.

Direction Weighted Sampling: Most real world trajectories have an inherent directionality. This is captured to some extent by the Markov model, and trajectories sampled from the true model tend to retain this property. However, this directionality could be lost due to the noise added to the counts of the prefix trees in the private model. In line with prior work on enforcing constraints in differentially private query answering [14], we present a novel postprocessing strategy to restore directionality while sampling synthetic trajectories from the noisy model.

Organization: We first discuss related work in Section 2. Section 3 presents our hierarchical reference system set model. Section 4 describes our differentially private techniques learning the model.
from the data, including model selection and direction weighted sampling. We present a thorough experimental evaluation in Section 5 using real and model-generated data sets.

2. RELATED WORK

Studies on human mobility patterns [10][25] have shown that individual trajectories are both highly unique and predictable. The study by Montjoye et al. [10] shows that the uniqueness of human mobility traces decays only slightly as their spatial and temporal resolution coarsens. This high identifiability indicates the difficulty of achieving properties such as k-anonymity, where at least k individuals have the same trajectory, using generalization, bucketization, and suppression techniques. Nevertheless, there have been numerous attempts in this direction [2][8][15][20][27][29]. At the same time, individual patterns exhibit high predictability, as in the study of Song et al. [25] who showed that each individual has more than 90 percent potential predictability, largely independent of the user’s typical radius. Hence, the release of an individual’s mobility data can make them susceptible to privacy attacks, such as user identification, sensitive location and sequential tracking attacks discussed by Kopanaki et al. [17].

Because of these two properties, we aim to provide a differentially private algorithm, first to give a more robust privacy guarantee than k-anonymity and second to synthesize traces resembling the repetitiveness of an individual’s trajectories. Several prior works have offered approaches to this question [6][7][22][24], but these do not adapt well to general trajectory data. Shao et al. [24] develop sampling and interleaving strategies to a variant (weaker) form of differential privacy. A line of work achieves ε-differential privacy by publishing synthetic traces with hierarchical structure [6][7] and frequency vectors [22]. The hierarchical framework captures the dependency of sequential data with a Markov assumption and achieves scalability with a set of pruning strategies. Chen et al. [7] build a tree by grouping trajectories with the same prefix, but due to the high uniqueness of patterns, the counts for leaf nodes are very sparse. In subsequent work [6], substrings are taken into account for building an exploration tree based on a Markov assumption, with the result that leaf counts are higher and hence offer better utility.

This line of work gives good insight into synthesizing the mobility patterns of individuals, but fails to address challenges faced in building Markov models for real unconstrained trajectory data. First, the large spatial and temporal domain size in trajectory data usually results in a large number of transition possibilities even for a 1st-order Markov model [22]. Pruning strategies proposed in [6][7] help to scale, but their impact on accuracy is unknown. The domain studied in [6][7] is short trajectories logged at subway stops, which are both shorter and over a smaller domain than the cases we wish to study. These works truncate trajectories to bound the sensitivity of queries, but there is no discussion on the optimal methods to extract the most useful information of the trajectories. Last, they all propose post-processing algorithms on the hierarchical counts, but in an ad hoc way, with no connection to a desired utility metric.

Pratesi et al. [22] define adjacency for reference points and remap observations to step between adjacent points, in order to reduce the number of possible transitions. However, only one reference system of homogeneous reference points is used, meaning that many steps take place between the same reference point, and so are discarded. In our work, we use a hierarchical reference system to capture different types of movement, and so minimize the loss of information during the mapping. Related work on adaptive spatial partitioning [4][9][23] has explored the adaptive decomposition of space under differential privacy, but this work looked primarily at point distributions rather than trajectories. Chen et al. [7] build a prefix tree with hybrid-granularity, where each level of the tree has two sub-levels of generalized locations and non-generalized locations. This hybrid granularity helps prune out nodes associated with zero-frequency patterns, but it is not clear how to generalize nodes at each level. Qardaji et al. [23] present an adaptive grid by laying a coarse-grained grid over the dataset, where each cell is then refined according to its noisy counts to answer range queries over the space. Cormode et al. [9] compute private medians for building data-dependent trees. These methods are effective for spatial decomposition when assuming no sequential dependency among locations. We extend the idea of adaptive decomposition of space to a hierarchical reference system set in order to capture trajectories with different speeds and sampling rates.

Recent work [4][6] on providing differential privacy for sequential data under Markov models represent the trajectories as strings (locations). Similar to Chen et al. [6], Bonomi et al. [4] constrain the amount of information collected from the query output in order to bound the sensitivity of queries, but the difference is that Bonomi et al. has a stable transformation over each object in the sequential data by selecting top patterns of each object instead of taking the first few occurrence of sequential data. Taking only top patterns for each object may miss the global information across all individuals while the first few observations of sequential data may capture some global information, but miss information at the end of long trajectories.

3. HIERARCHICAL RSS MODEL

We next describe a Markov model for mobility trajectories and build up to our novel hierarchical reference system set (RSS) model.

3.1 Notation and Background

Given a spatial domain $\Sigma$, a regular trajectory $t$ is a sequence of locations $\sigma_i$ observed and recorded at regular time intervals, where $\sigma_i \in \Sigma$ for $i = 1, 2, \ldots$. The spatial domain is usually a set of latitude-longitude coordinates. We consider a trajectory database $D$ consisting of $|D|$ individuals. Each individual has a personal trajectory table, denoted by $PT^i$ with each tuple representing a regular trajectory, $t$. It is possible for an individual to have multiple trajectories, of varying lengths, where length is defined as the number of points in the trajectory. We call any sequence $z \in \Sigma^*$ of $k$ locations as a $k$-gram. If $x$ is a $k$-gram and $y$ is an $\ell$-gram, then we denote by $xy$ the $(k + \ell)$-gram obtained by appending the sequence $y$ to sequence $x$. We denote by $c(t,x)$ the number of occurrences of $x$ in $t$. $c(\Sigma^k, x)$ the number of occurrences of $x$ in the trajectory table $PT^i$, i.e. $c(\Sigma^k, x) = \sum_{t\in PT^i} c(t,x)$, and $c(D, x)$ the total number of occurrences of $x$ in the database $D$, i.e. $c(D, x) = \sum_{PT\in D} c(PT^i, x)$.

<table>
<thead>
<tr>
<th>$D$</th>
<th>Trajectory database</th>
</tr>
</thead>
<tbody>
<tr>
<td>$PT^i$</td>
<td>Personal Trajectory table of an individual</td>
</tr>
<tr>
<td>$t$</td>
<td>A single trajectory</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>A set of latitude-longitude coordinates</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Latitude-longitude coordinate of a location</td>
</tr>
<tr>
<td>$\Sigma$</td>
<td>Reference system, a set of anchor points</td>
</tr>
<tr>
<td>$a$</td>
<td>Anchor point in reference system $\Sigma$</td>
</tr>
<tr>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>$\top$</td>
<td>Ending symbol</td>
</tr>
<tr>
<td>$\Sigma^k$</td>
<td>k-gram pattern</td>
</tr>
<tr>
<td>$c(t,x)$</td>
<td>The number of occurrences of $x \in \Sigma^k$ in $t$</td>
</tr>
<tr>
<td>$c(\Sigma^k, x)$</td>
<td>The number of occurrences of $x \in \Sigma^k$ in $PT^i$</td>
</tr>
<tr>
<td>$c(D, x)$</td>
<td>The number of occurrences of $x \in \Sigma^k$ in $D$</td>
</tr>
</tbody>
</table>

Table 1: Notation Summary
We summarize the notation of the paper in Table 1. Markov processes \( \Sigma \) to model correlations between contiguous locations in a regular mobility trajectory are defined as follows.

**Definition 3.1 (Markov Process).** A regular trajectory \((\sigma_1, \sigma_2, \ldots, \sigma_n) \in \Sigma^n\) is said to follow an order \(\ell\) Markov process if for every \(\ell \leq i < n\), \(\sigma \in \Sigma\)

\[
P_r[\sigma_{i+1} = \sigma | \sigma_1 \cdots \sigma_i] = P_r[\sigma_{i+1} = \sigma | \sigma_{i-\ell+1} \cdots \sigma_i]
\]

(1)

We refer to the probability \(P_r[\sigma_{i+1} = \sigma | \sigma_{i-\ell+1} \cdots \sigma_i]\) as a transition probability of the Markov process. The collection of transition probabilities for all \(x = \sigma_{i-\ell+1} \cdots \sigma_i \in \Sigma^\ell\) can be estimated using the set of all \(\ell\) - and \(\ell+1\)-gram counts, i.e.

\[
P_r[\sigma_{i+1} = \sigma | \sigma_{i-\ell+1} \cdots \sigma_i] = \frac{c(D, x\sigma)}{c(D, x)}
\]

(2)

Typically, a starting symbol (\(T\)) and a stopping symbol (\(\downarrow\)) are prepended and appended (respectively) to the original trajectories so that the starting and stopping probabilities are captured in the model (and are treated as special locations in \(\Sigma\)). A synthetic trajectory is initialized with a starting symbol (\(T\)) and is continuously sampled by picking the next location from the Markov process till reaching the stopping symbol (\(\downarrow\)). Formally, let \(x = (T \sigma_1 \cdots \sigma_i)\) denote the prefix of a trajectory sampled so far and \(x^\prime\) the longest suffix of \(x\) of length \(\ell \leq (k-1)\), where \((k-1)\) is the maximum order of Markov process considered. The next location is chosen using the transition probability given \(x^\prime\) in the order \(\ell\) model. Thus, sampling synthetic trajectories requires us to maintain all \(\ell\)-gram counts for \(1 \leq \ell \leq k\). These counts can be stored in a prefix tree \(T\) of height \(k\) — nodes in \(T\) are \(\Sigma^1 \cup \cdots \cup \Sigma^\ell\) and edges connect each \(\ell\)-gram \(x\) to an \((\ell+1)\)-gram \(x\sigma\), for all \(\sigma \in \Sigma\).

### 3.2 Discretization and Reference Systems

A trajectory is usually recorded as a sequence of points in some continuous domain \(\Sigma\), e.g. latitude-longitude coordinates. One common way to analyze trajectories on a continuous domain (and to limit the size of the model) is via discretization of the space using a reference system \([6, 7, 22, 26]\).

**Definition 3.2 (Reference System).** A reference system \((RS)\) is a set of anchor points \(\Sigma \subseteq \Sigma\), associated with a mapping function \(f : \Sigma \rightarrow \Sigma\).

In this work, we use a reference system constructed by imposing a uniform grid over the space and choosing the centroids as anchor points. We denote such a reference system by \(\Sigma_m\), where \(v\) denotes the resolution or the length of the side of each grid cell. Varying the resolution \(v\) results in different reference systems; coarse grids (large \(v\)) correspond to fewer anchor points, and fine grids (small \(v\)) result in a larger number of anchor points. The function considered here for the reference system \(\Sigma_m\) is represented by \(f_{m}\) and maps a point in \(\Sigma\) to its closest anchor point in \(\Sigma_m\) (if a point is equidistant to more than one anchor point, ties are broken in a consistent way).

There are of course other methods for instantiating reference systems apart from grids, such as using points of interest (POI) or intersections on a road network as anchors. These methods can also be adopted within the DPT system directly, but for clarity we frame the subsequent discussion in terms of grids.

### 3.3 Hierarchical Reference Systems

Even with the use of a reference system \(\Sigma\), the number of parameters (transition probabilities) to specify a Markov model can be very large. The model requires \(\approx |\Sigma|^{k+1}\) counts (for large \(|\Sigma|\)) in a prefix tree \(T\) of height \(k\) (i.e., \(\ell\)-gram counts for \(1 \leq \ell \leq k\)). For instance, if \(\Sigma\) is a coarse \(16 \times 16\) grid partitioning the space into 256 cells, then for \(k = 3\), \(|\Sigma|^{k+1} \approx 4 \cdot 10^9\). Yet, grids of much finer resolution are needed to model all the transitions in the data.

Therefore, rather than building a Markov model using a single reference system \((\Sigma_m)\), we adopt a novel approach in DPT. First, given a reference system \(\Sigma\), we choose to estimate transition probabilities of \(\ell\)-grams, \(x \in \Sigma^\ell\) to only the anchor points that are “close” to the last observed location in \(x\). In the context of a uniform grid based reference system \(\Sigma_m\), we only consider transitions from a grid cell \(a\) to either itself or to its 8 adjacent cells. We call them the neighboring cells. All other transition probabilities are forced to be 0. Under this constraint, we only need to maintain \(|\Sigma_m|\) 1-gram counts, \(|\Sigma_m|\) 2-gram counts, \(|\Sigma_m|\) 3-gram counts and so on, thus reducing the total number of counts in the prefix tree \(T\) from \(|\Sigma|^{k+1}\) to \(O(9^k \cdot |\Sigma_m|)\).

However, restricting transitions to only neighboring anchor points cannot capture all the transitions in the original trajectories under any single reference system. This is because, even within a single trajectory, objects tend to move at different speed ranges. For instance, a taxi may travel at greater speeds on a highway, but with much slower speeds in downtown, and both types of driving can occur in a single trip. Hence, we use a set of reference systems to capture motion at different speed ranges, and transitions in each of these reference systems occur between anchor points that are close.

**Definition 3.3 (Hierarchical Reference Systems).** Let \(\Sigma\) denote the set of centroids of cells in a uniform grid of resolution \(v\). The reference system set \(RSS = \{\Sigma_1, \ldots, \Sigma_m\}\) is a set of hierarchical reference systems, where \(v_1 < \cdots < v_m\). For any anchor point \(a \in \Sigma_m\), we define (1) its parent in \(\Sigma_{m'}\), where \(m' > m\) as the closest point to \(a\) in \(\Sigma_{m'}\).

\[
par(a, \Sigma_{m'}) = \text{argmin}_{a' \in \Sigma_{m'}} \text{dist}(a', a)
\]

and (2) its children in \(\Sigma_{m'}\), where \(m' < m\) as children\((a, \Sigma_{m'}) = \{a' \in \Sigma_{m'} | \text{dist}(a', a) = a\}\).

If multiple anchor points can be the parent of a single anchor point, ties are broken in a consistent way. In this paper we consider a set of reference systems with geometrically increasing resolutions. That is, we consider \(\{\Sigma_1, \ldots, \Sigma_m\}\) such that for all \(1 \leq m \leq M\),

\[
v_{m}/v_{m-1} = 2.
\]

This set of hierarchical reference systems \(RSS = \{\Sigma_1, \ldots, \Sigma_m\}\) can capture motions at different speed ranges in the raw trajectory \(t\) — long steps (pairs of consecutive points which are far away) are mapped with finer reference systems. While shorter steps are mapped with coarser reference systems. We assume the longest step in the whole database is less than \(v_M\). Given an RSS, we can map a
raw trajectory \( t \) to an RSS-based trajectory \( t_{\text{RSS}} \) that contains anchor points from the RSS, such that any pair of consecutive anchor points \( a_{i-1}, a_i \) in the output trajectory \( t_{\text{RSS}} \) are either (1) neighboring cells or the same cell if they are from the same reference system \( \Sigma_{rs} \), i.e., \( d(a_{i-1}, a_i) \leq v_m \), where \( d(\cdot, \cdot) \) is the \( L_m \)-norm of the coordinates difference between \( a_{i-1} \) and \( a_i \); or (2) in a parent-child relationship if they are from different reference systems, i.e., \( a_{i-1} = \text{par}(a_i, \Sigma_{rs}) \) or \( a_i = \text{par}(a_{i-1}, \Sigma_{rs}) \). In this way, the number of transitions maintained is controlled by the number of neighboring cells in each reference system, and changes in speed are captured by the transitions across different reference systems. The trajectory \( t_{\text{RSS}} \) then becomes a sequence of segments, where a segment is defined as contiguous anchor points from the same reference system. We introduce a new start symbol \( \top \) representing a special location in each reference system \( \Sigma_{rs} \) to indicate the start of a segment falling into that reference system.

**Example 3.1.** Figure 4 considers four grid-based reference systems \( \Sigma_{rs} \), \( \Sigma_{rs} \), \( \Sigma_{rs} \), \( \Sigma_{rs} \), where \( \Sigma_{rs} = (v_{x}, v_{y}) \), and \( v_{x} \) and \( v_{y} \) are the horizontal and vertical indexes in \( \Sigma_{rs} \). The trajectories shown here refer to the same trajectory. The first four points in \( t \) are mapped to anchor points in \( \Sigma_{rs} \) as \( (3,0),(4,1),(5,1) \) (neighbors in \( \Sigma_{rs} \), as each step crosses more than one cell in \( \Sigma_{rs} \). Similarly, the next step after \( (5,1) \) crosses more than one cell in \( \Sigma_{rs} \), and hence a coarser reference system \( \Sigma_{rs} \) is used for the mapping, resulting in the anchor point \( (2,0) \). Though the next step after \( (2,0) \) could be mapped to a finer reference system \( \Sigma_{rs} \) as \( (7,1) \), this is not preferable, since the trajectory is not slowing down. Greedily changing the reference system to a finer resolution is not favored. Hence, we adopt the rule that only when at least 3 consecutive points are mapped to the same cell of the current reference system do we change to a finer reference system. This gives more contiguous points in \( \Sigma_{rs} \) than \( \Sigma_{rs} \), \( \Sigma_{rs} \), \( \Sigma_{rs} \), \( \Sigma_{rs} \). As the last three points remain in the same cell of \( \Sigma_{rs} \), this implies slowing down, and so the following points are mapped to a finer reference system \( \Sigma_{rs} \) as \( (13,3) \), \( (13,4) \), \( (14,3) \). This results in three segments: (1) \( (3,0) \), \( (4,0) \), \( (4,1) \), \( (5,1) \), \( (2,0) \); (2) \( (2,0) \), \( (3,0) \), \( (4,0) \), \( (5,0) \), \( (6,1) \), \( (6,1) \), \( (13,3) \); (3) \( (13,3) \), \( (13,4) \), \( (14,6) \).

Algorithm 1 summarizes our proposed transformation process. This process starts with adding start symbol \( \top \) to indicate the beginning of a trajectory in Line 1. \( m^* \) represents the optimal reference system used for the current mapping when iterating through the sequence of points in Line 3-10. Line 4 specifies that every step (pair of consecutive points) in the raw trajectory is mapped to neighboring cells \( (a_{i-1}, a_i) \) within the same reference system at the most detailed resolution. This ensures that the finest resolution at the same or coarser level is used (lines 5-6). However, this greedy mapping may lead to many short segments. Hence, in Lines 7-9, before updating the optimal reference system \( \Sigma_{rs} \) to a finer reference system \( \Sigma_{rs} \), we check whether both current and previous steps are sufficiently small (the three consecutive anchor points in the previous optimal reference system are the same). The intuition for this is that if the travelling speed is halved to the previous speed range \( v_m \), at least 2 consecutive steps may fall into the same cell in the current reference system because of the geometrically increasing setting of RSS, i.e., \( v_m/v_{m-1} = 2 \).

![Figure 4: Prefix tree \( T_2 \) for \( \Sigma_{v2} \)](image)

**Algorithm 1:** RSS-based Transformation

**Input:** \( t = \sigma_1 \sigma_2 \cdots \sigma_m \): regular trajectory, \( \text{RSS} = \{\Sigma_1, \ldots, \Sigma_m\} \): \( M \) hierarchical reference systems.

**Output:** \( t_{\text{RSS}} \): sequence of anchor points in \( \text{RSS} \)

1. Initialize \( t_{\text{RSS}} = (\top) / \text{start symbol} \)
2. Initialize optimal reference system index \( m^* \) \( \leftarrow 1 \)
3. For \( i = 2: |t| \) do
   4. \( m^* \leftarrow \text{finest reference system } \Sigma_{rs} \text{ s.t.} \)
      \( d(f_{\text{rs}}(\sigma_{i-1}), f_{\text{rs}}(\sigma_{i})) \leq v_m \)
   5. If \( v_m \leq v_m \) then
   6. \( m^* \leftarrow m^* \text{ / coarser or the same reference system} \)
   7. Else \( f_{\text{rs}}(\sigma_i) = f_{\text{rs}}(\sigma_{i-1}) = f_{\text{rs}}(\sigma_{i-2}) \) then
   8. // stay in the same cell (assume \( \sigma_0 = \sigma_1 \))
   9. \( m^* \leftarrow m^* \text{ / finer reference system} \)
   10. Add \( f_{\text{rs}}(\sigma_{i-1}) \) to \( t_{\text{RSS}} \)
   11. Add \( f_{\text{rs}}(\sigma_i) \) and \( \top \) to \( t_{\text{RSS}} \) / last location and end symbol
   12. return \( t_{\text{RSS}} \)

**3.4 Hierarchical RSS Model**

We now present a probabilistic generative model for trajectories based on hierarchical reference systems \( \text{RSS} = \{\Sigma_1, \ldots, \Sigma_m\} \). Let \( x = (a_1 \cdots a_i) \) denote the prefix of a trajectory that has been mapped to RSS using Algorithm 1. Thus \( x \) may contain anchor points from \( \Sigma_{v1} \cup \ldots \cup \Sigma_{vM} \). Given \( x \), there are two ways to extend \( x \) to the next anchor point \( a_{i+1} \) where (1) \( a_{i+1} \) is a neighboring cell of \( a_i \) in the same reference system; or (2) \( a_{i+1} \) is the parent or a child of \( a_i \) in a different reference system. The cut for each extension is maintained in a set of prefix trees \( \{T_1, \ldots, T_M\} \). Each \( T_m \) has a height \( k \) similar to the basic Markov process described in Section 3.3. We name this set of prefix trees as a forest \( F \) or forest counting query \( F(\cdot) \) as defined in Definition 3.4. Nodes in the prefix tree \( T_m \) correspond to \( \ell \)-grams in \( \Sigma_{v_m} \) for \( 1 \leq \ell \leq k \), as well as \( \ell \)-grams appended with a move to a different reference system \( m^* \neq m \). Thus, every node in \( T_m \) has 9 children representing moves to neighboring nodes in the same reference system (same speed range), and \( M \) children representing moves to different reference systems including \( \top \) as \( \Sigma_{v_m} \) (change speed range). In Example 3.2 we give more details on \( F \) built from the trajectory shown in Example 3.1. Then we continue illustrating how to sample trajectories from \( F \) in the next section (Section 3.5).

**Definition 3.4.** (Forest Counting Query \( F(\cdot) \)). The term ‘Forest counting query’ refers to the set of counting queries on the database \( D \) to build model \( F \), denoted by

\[
F(D) = \{ c(D, x) | x \in T, T \in F \}.
\]

**Example 3.2.** Given the hierarchical reference systems in Example 3.1, \( \text{RSS} = \{\Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1, \Sigma_1\} \), we build a forest consisting of 4 prefix trees: \( F = \{T_1, T_2, T_3, T_4\} \). In Figure 5, we materialize some of the nodes in \( T_2 \) whose counts are partially contributed by the segments resulted from the transformation of trajectory in Figure 4. The highest level of \( T_2 \) consists of all the anchor points in \( \Sigma_{v2} \), the starting symbol \( \top \), and the start of a segment \( \top \). The
I-grams listed in Figure 3 add counts to their corresponding nodes in T2. For instance, \((4,1)_2\) increments the count of node \((4,1)\). All the nodes except \(\tau\) and \(\tau_{iv}\) have two types of children: those that remain with the same step size \(v_2\) and those with a different step size from their parent. For instance, the 2-gram \((4,1)_2(5,1)_2\) corresponds to the node \((4,1)(5,1)\) which stays in the same reference system \(\Sigma_{iv}\); the 2-gram \((14,6)_2(14,6)\) corresponds to the node \((14,6)(14,6)\) which changes from \(\Sigma_1\) to \(\Sigma_0\) (a stop) and the 3-gram \((4,1)_2(5,1)_2(2,0)_3\) corresponds to the node \((4,1)(5,1)(2,0)\) which changes from \(\Sigma_2\) to \(\Sigma_3\). \(\tau\) indicates the start of a trajectory and \(\tau_{iv}\) indicates the start of a segment (changing from another reference system). They could be appended by any anchor points in \(\Sigma_{iv}\). The 2-gram \((3,0)_2\) and \(\tau_{iv}(13,3)_2\) increment the counts of nodes \(\tau(3,0)\) and \(\tau_{iv}(13,3)\) respectively.

3.5 Sampling Trajectories

Given a forest \(F = \{T_1, ..., T_m\}\) of height \(k\) built upon \(\text{RSS} = \{\Sigma_1, ..., \Sigma_{M}\}\), trajectories are sampled independently of each other. We sample the starting reference system \(\Sigma_{iv}\) using probabilities proportional to the count of start symbol \(\tau\) in each \(T_m\). Let \(x = (\tau a_1 \cdots a_i)\) denote the prefix of the trajectory generated thus far, and \(x'\) denote the longest suffix of \(x\) of length \(\ell < k\) from the same reference system. Given \(x'\), the next location is chosen in two steps: (1) choose a reference system \(\Sigma_{iv}\) from RSS or stopping symbol \(\perp\) (represented by \(\Sigma_{iv}\), and), and (2) transition to a neighboring cell of \(a_i\) in the chosen \(\Sigma_{iv}\). Both probabilities depend on the immediate history (of \(\ell \leq k\) events) in the same reference system as \(a_i\). Thus, the first probability can be estimated from \(F\) as:

\[
\Pr[\Sigma_{iv}|x] = \frac{c(D, x(\Sigma_{iv}))}{c(D, x')},
\]

(4)

\(c(D, x(\Sigma_{iv}))\) is the number of trajectories containing \(x'\) followed by a move in reference system \(\Sigma_{iv}\), for \(m \leq M\). Given \(x'\), we first sample the next reference system \(\Sigma_{iv}\) based on the transition probability stated in Eqn. \(\Box\).

For a neighboring cell \(a\) in the new reference system \(\Sigma_{iv}\), the second probability can be estimated as:

\[
\Pr[a|x, \Sigma_{iv}] = \frac{c(D, x(a))}{c(D, x')},
\]

(5)

If a move to the same reference system is sampled, sampling continues using the first 9 children of \(x'\); if a move to a coarser reference system is sampled, sampling continues using the ancestor of the last location in \(x'\); if a move to a finer reference system is sampled, sampling continues using one of the children of the last location in \(x'\). For the last case, rather than uniformly sampling a child of the last location \(a_i\), we sample \(a_{i+1}\) from the children of \(a_i\) in the new reference system \(\Sigma_{iv}\), using probabilities proportional to the count of \(\tau_{iv}a_{i+1}\) in \(T_m\). The trajectory generation is stopped when a maximum length \(L_{max}\) is reached, or if a stopping symbol \(\perp\) is sampled as the reference system \(\Sigma_{iv}\).

4. PRIVATE MODEL LEARNING

Two aspects of the RSS model need to be learnt: (i) the structure of the model, i.e., the set of reference systems to be included in the model, and the maximum height of the prefix trees for the chosen reference systems, and (ii) the transition probabilities, within and across the chosen reference systems. Transition probabilities are learnt from the data by estimating counts in corresponding prefix trees. Differentially private methods for prefix tree construction have been studied in prior work \([6, 7]\), and our prefix tree construction algorithm (Section 4.2) follows the approach in prior work with a few notable additions. We normalize the counts in the original trajectory table of each individual to bound the sensitivity (see Defn \(\Box\) of the forest counting query. Adding noise to subtrees of the prefix tree with low counts can lead to low signal to noise ratios. We propose an adaptive pruning strategy to eliminate such subtrees, which outperforms prior prefix tree pruning strategies.

The main privacy contributions of our work, however, are (a) a novel differentially private model selection algorithm for learning the structure (Section 4.2), and (b) a direction weighted sampling strategy that recovers the global directional properties from the data that might be lost due to the noisy transition probability learning phase (Section 4.4). Structure learning is crucial, since the set of appropriate reference systems is not known apriori. A range of candidate reference systems that capture most of the transitions can be learnt purely using domain knowledge (of typical speeds) and without access to the data. However, as we show in our experiments, even within this range, only a few of the reference systems may have sufficient data support. Hence, structure learning based on the data can significantly improve the signal to noise ratio, and thus the utility of the learnt model.

In this section, we summarize the model of differential privacy and describe our novel model selection algorithm, followed by a brief overview of our methods for estimating transition probabilities privately. We then outline direction weighted sampling. We conclude with an end-to-end privacy evaluation of the system.

4.1 Differential Privacy

We define \(\varepsilon\)-differential privacy \([12]\) in the context of trajectory databases. Let \(D_1, D_2\) be two neighboring trajectory databases, i.e., \(D_1\) and \(D_2\) differ in only one individual’s trajectory table \(PT\), written as \(|D_1 - D_2| = 1\). This paper consistently takes this to mean that \(PT\) is present in only one of the two databases.

**Definition 4.1 (Differential Privacy).** Let \(D_1, D_2\) be two neighboring databases. Let \(M\) denote a randomized algorithm over databases, and \(O\) be a possible output of \(M\). Mechanism \(M\) is said to be \(\varepsilon\)-differentially private, for \(\varepsilon > 0\), if

\[
\Pr[M(D_1) = O] \leq \exp(\varepsilon) \Pr[M(D_2) = O].
\]

(6)

In our setting, differential privacy guarantees that no individual’s trajectory table can significantly affect the released information — the output distribution generated by \(M\) is nearly the same, whether or not the individual’s trajectory table is present in the database. We make use of the concept of global sensitivity \([12]\).

**Definition 4.2 (Global Sensitivity).** The global sensitivity of a function \(q : D \rightarrow \mathbb{R}^d\), denoted by \(S(q)\), is defined as the largest \(l_1\) difference \(|q(D_1) - q(D_2)|_1\), where \(D_1\) and \(D_2\) are neighboring databases that differ in one personal trajectory table.

Formally,

\[
S(q) = \max_{D_1, D_2: |D_1 - D_2| = 1} \|q(D_1) - q(D_2)\|_1.
\]

(7)

One common technique used to achieve differential privacy is the Laplace mechanism proposed in \([12]\):

**Definition 4.3 (Laplace Mechanism).** The Laplace mechanism privately computes a function: \(q : D \rightarrow \mathbb{R}^d\) by computing \(q(D) + \eta\), where \(\eta \in \mathbb{R}^d\) is a vector of independent random variables, where \(\eta_i\) is drawn from the Laplace distribution with parameter \(S(q)/\varepsilon\). That is, \(\Pr[\eta_i = z] \approx e^{-z/\varepsilon}/S(q)\).
4.2 Private Model Selection

We would like our hierarchical reference system set to be full; i.e., every move in the trajectory dataset \( D \) can be captured as a move between neighboring cells in some reference system. Based on domain knowledge one can construct a full reference system set \( \{ \Sigma_{v_1}, ..., \Sigma_{v_M} \} \) in a data independent fashion (for instance, by picking \( v_m \)'s as powers of 2 that capture the smallest and largest step sizes). We denote the forest \( F \) built on the full model by \( F_{\text{full}} \). Moreover, we would also like the trees in the forest to have height \( k \) large enough to cover high order correlations.

However, choosing the full model or large \( k \) is not desirable under the differential privacy regime: the total variance of the counts in the noisy forest grows linearly in the number of trees and quadratically in the height \( k \). Thus, the variance introduced by the Laplace mechanism can drown out the signal for large forests.

Our approach to solve this problem is to drop some of the trees (resulting in \( F^+ \subset F_{\text{full}} \)) and/or choose a small height \( k \) such that the least bias is introduced into the model (since some of the transitions may no longer be accurately captured). If by introducing a little bias, we can dramatically reduce the variance, the total error will be smaller than the error in \( F_{\text{full}} \). Hence, we would like to identify \( F^+ \) and \( k \) which minimize the sum of the variance and bias. The bias of a model arises from the least coarse moves in \( F^+ \), losing information. We approximate the information loss in each tree \( T_m \in F^- \) of height \( k \) by the squared counts of moves from the corresponding reference system, since we are effectively representing this counts by zeros (i.e. the bias).

Let \( c_m(D) \) be the number of transitions in \( D \) that occur in reference system \( \Sigma_m \) (the sum of 1-gram counts in \( T_m \)). Since \( c_m(D)^2 \) is an upper bound for the squared counts of i-grams for \( i > 1 \), it can be used as an estimate of the information loss at level \( i \) in the tree \( T_m \).

Formally, we express this loss term \( C_m(D,k) \) as:

\[
C_m(D,k) = k \cdot c_m(D)^2,
\]

where \( k \) is the height of the tree. We use a small part of the privacy budget \( \epsilon_r \) to compute the noisy \( c_m \).

The variance for tree \( T_m \in F^+ \) of height \( k \) can be computed as:

\[
N_m(\epsilon_r,k) = \mathbb{E} \left[ \sum_{x \in T_m} (\epsilon_{F^+}(D,x) - \epsilon_{F^-}(D,x))^2 \right]
\]

where, \( \epsilon_r = \epsilon - \epsilon_1 \) is the remaining privacy budget that will be used to learn the transition probabilities (in the next section). \( \text{Var}(\epsilon_r,k) = \frac{2}{k} \cdot \epsilon_r^2 \), where \( \epsilon_r(x) \) is the privacy budget used to compute the noisy count for \( x \) and is dependent on \( \epsilon_r \) and \( k \).

Summing up the variance and bias terms specified in Eqns. (8) and (9), the total error of \( F^+ \) of height \( k \) is:

\[
\text{Error}(F^+,(\epsilon_r,k,D)) = \sum_{T_m \in F^+} N_m(\epsilon_r,k) + \sum_{T_m \in F^-} C_m(D,k).
\]

Our goal is to find the optimal model \( F^{+\star} \) of height \( k \) which minimizes the overall error specified in Eqn. (10) with respect to \( F^+ \) and \( k \), i.e.

\[
F^{+\star}(\epsilon_r,D) = \arg\min_{F^+ \subset F_{\text{full}}, k \geq 2} \text{Error}(F^+,(\epsilon_r,k,D)).
\]

Algorithm 2: Private Model Selection

\[
\begin{align*}
\text{Input: } D &: \text{ database, } F_{\text{full}}; \text{ full model of size } M, \epsilon: \text{ total privacy budget, } \epsilon_r: \text{ privacy budget for model selection, } k_{\text{max}}: \text{ the maximum tree height} \\
\text{Output: } (F^{+\star}, k): &\text{ a subset of } F_{\text{full}} \text{ of order } k \\
1: &\text{ Count moves in each reference system: } (c_1(D),...,c_M(D)) \\
2: &\epsilon_{\text{full}} = (c_1(D),...,c_M(D)) + \text{Lap}(1/\epsilon_r)^M \\
3: &M' = \text{ largest index } m \text{ in } \{T_1, ..., T_M \} \text{ s.t. } \epsilon_m(D) \geq \frac{\epsilon_r}{M'} \\
4: &\text{Initialize } k = k_{\text{max}}, F^{+\star} = \{T_1, ..., T_M \} \\
5: &\epsilon_r = \epsilon - \epsilon_1 \\
6: &\text{ while } k \geq 2 \text{ do} \\
7: &\text{ Compute } \sum_{T_m \in F^{+\star}} c_m(D,k) \forall T_m \in F^{+\star} \\
8: &F^{+\star}(\epsilon_r,D) = \arg\min_{F^+ \subset F^{+\star}} \text{Error}(F^+,(\epsilon_r,k,D)) \\
9: &\text{ if } F^{+\star}(\epsilon_r,D) = \emptyset \text{ then} \\
10: &\text{ return } \{F^{+\star} \cup \{T_m \} \}, k \\
11: &\text{ else} \\
12: &\epsilon_r' = \epsilon_r \\
13: &k = k - 1 \\
14: &\text{ return } \{F^{+\star} \cup \{T_m \} \}, k \\
\end{align*}
\]

Eqn. (11) satisfies an interesting, and intuitive monotonicity property of the optimal reference systems as a function of privacy budget \( \epsilon_r \) and data size \( |D| \): (1) given fixed data size \( |D| \), a larger \( \epsilon_r \) allows either more reference systems to be chosen, or trees to have larger heights; (2) given fixed privacy budget \( \epsilon_r \), a larger dataset allows either more reference systems to be chosen, or trees to have larger heights. We omit formal statement and proof of these claims for brevity. This monotonicity property suggests that given fixed dataset \( D \) and privacy budget \( \epsilon_r \),

\[
\text{argmin } \text{Error}(F^+,\epsilon_r,k+1,D) \subseteq \text{argmin } \text{Error}(F^+,\epsilon_r,k,D) \\
F^+ \subset F_{\text{full}}
\]

Hence, we propose Algorithm 2 to search for the optimal \( F^{+\star} \) and also \( k \) by decrementing the value of \( k \) from its maximum possible value. In this way, the search space of the optimal model at \( k \) is not the subsets of \( F_{\text{full}} \) any more, but the subsets of the optimal model at \( k+1 \) as shown in Line 8. In Algorithm 2 Lines 1-2 use a privacy budget of \( \epsilon_1 \) to compute the noisy bias for each reference system if the corresponding system is dropped. Line 3 chooses the coarsest possible reference system \( \Sigma_m \), which has a noisy root count square greater than the variance of the noise \( \frac{\epsilon_r}{k} \). Lines 4-5 initialize the full model's maximum possible tree height \( k \) and remaining budget \( \epsilon_r \). Then the variance in \( F^{+\star} \) is estimated based on the remaining budget \( \epsilon_r \). Lines 6-14 identify the best model with the lowest Error \( F^+, \epsilon_r, k, D \) given the height \( k \). The coarsest reference system is included in final output (in Line 10 and 14) in order to record any large steps which the optimal model fails to capture. This algorithm may not guarantee the optimal solution for Eqn. (11), but reduces the searching space based on Eqn. (12) and is shown to be effective in our evaluation (Section 5.4).

4.3 Learning Transitions

Noisy Forest Counts. The transition probabilities in the optimal model \( F^{+\star}, k \) can be learned by adding Laplace noise to the counts of the prefix trees corresponding to \( F^{+\star} \). Each transition maps to exactly one prefix tree in \( F^{+\star} \), and affects the counts of exactly one node at each of the \( k \) levels of the tree. Thus, adding or removing a user’s trajectory table with at most \( L \) transitions affects at most \( L \cdot k \) nodes. However, there is no apriori bound on \( L \), resulting in an unbounded sensitivity for the forest counting query.
Rather than resorting to truncate a user's trajectories [6,7] or sampling a bounded number of transitions from a user's trajectory table [4,28], we choose to normalize the counts such that the total weight contributed by all transitions in a user's trajectory table is 1. That is, if \( PT \) has \( L \) transitions, then each transition will contribute a weight of \( 1/L \) to the counts in the prefix tree. Thus, a user's trajectory table contributes at most weight 1 to level \( i \) of all prefix trees, resulting in a sensitivity of \( k \) for the counting query.

We follow Cormode et al.'s [9] strategy of splitting the privacy budget \( \epsilon \) geometrically across the different levels of the prefix trees. The roots of the prefix trees are assigned a privacy budget \( \epsilon_0 \). Level \( i \) nodes receive a budget of \( \epsilon_i = \epsilon_0 \cdot \gamma^i \), where \( \gamma = 2^+ \) is a multiplicative factor that increases the budget (resulting in smaller noise) as we move down the prefix tree (to account for lower counts). \( \epsilon_0 \) is set such that \( \sum_{i=0}^{\infty} \epsilon_i = \epsilon \). For a node at level \( i \) in a prefix tree, a noisy count \( \tilde{c}(D,x) = c(D,x) + \eta \) is released, where \( \eta \) is drawn from Laplace(1/\( \epsilon_i \)).

**Adaptive Pruning.** Model selection prunes away prefix trees \( T \) with low support and chooses a height \( k \). However, in each remaining tree \( T \), a large fraction of the nodes may have a count of 0. Releasing noisy counts for nodes with 0 counts increases the size of the tree and dilutes the signal to noise ratio. Hence, like in prior work [6], after adding Laplace noise to the nodes in the prefix tree, (including nodes with a count of 0), we prune subtrees that could result in very low signal to noise ratios. Our pruning strategy is different from prior work: Nodes in tree \( T \) are traversed breadth-first. With each node \( x \in T \), we associate a threshold \( \tilde{\theta}(x) \) based on signal to noise ratio. If the noisy count \( \tilde{c}(x) \) is greater than the threshold \( \theta(x) \), we release the noisy count and continue. Otherwise, we prune away all the descendants of \( x \). If the privacy budget \( (\epsilon') \) that would have been used for a descendant of \( x \) is greater than the privacy budget for node \( x \), then we recompute the noisy count for \( x \) using \( \epsilon' \).

**Choice of Threshold \( \tilde{\theta}(x) \):** There are many choices for setting the thresholds \( \tilde{\theta}(x) \). A simple data independent method, which we call fixed pruning, is to apply a fixed threshold \( \theta \) for all the nodes in the tree. When \( \theta \) is too small, we still add noise to many nodes with a true count of 0. If \( \theta \) is too large, we may prune away nodes with large true counts. Both scenarios result in poor utility.

Hence, we propose an adaptive pruning strategy to select different thresholds for different nodes such that pruning results in a reduction in total error with high probability. The intuition behind adaptive pruning is the same as that for our model selection strategy: if the count of a node \( x \) is much smaller than the number of its descendants, then the bias introduced by pruning the subtree below \( x \) is much smaller than the variance due to noisy counts (if the subtree is not pruned). This is formalized in the following lemma:

**Lemma 4.1.** Let \( T \) be an input tree of height \( k \) and let \( x \) be a node at level \( \ell \). Let \( \text{desc}(x,i) \) denote the number of descendants of \( x \) in level \( i \). Let \( \tilde{T} \) be the noisy tree resulting from the Laplace mechanism (with geometric budgeting). Let \( \tilde{T}_{\text{pruned}} \) be the result of pruning the subtree under \( x \) whenever

\[
c(D,x) < \tilde{\theta}(x) = \frac{2}{k+1} + \frac{1}{\epsilon_i} + \frac{1}{\epsilon'(x)^2} - \frac{1}{\epsilon'^2}
\]

where \( \epsilon_i \) is the privacy budget allocated to nodes at level \( i \) by geometric budgeting, and \( \epsilon' = \epsilon - \sum_{i} \epsilon_i \). Then, \( \tilde{T}_{\text{pruned}} \) has lower error than \( \tilde{T} \).

Note that for regular trees (where all nodes have the same degree), nodes in the same level of the tree are assigned the same threshold \( \theta(x) \). However, adaptive pruning works after noise has been added to a node, and hence does not have access to the true count. We use \( \tilde{\theta}(x) = \theta(x) - \ln(1/\delta)/\epsilon(x) \), and prune \( x \)'s descendants when the noisy count \( \tilde{c}(D,x) < \tilde{\theta}(D,x) \). This is because with probability \( 1-\delta \) if the noisy count \( \tilde{c}(D,x) \) is smaller than \( \theta(x) \) then true count \( c(D,x) \) is smaller than the threshold \( \theta(x) \). This pruning choice seems to work well on real datasets in our evaluation.

**Stochastic Transition Probabilities.** In order for a prefix tree to represent stochastic transition probabilities, the counts in the tree should be non-negative. Additionally, the counts should satisfy prefix consistency i.e., the count at a node \( x \) must equal to the sum of the counts at its children. Consistency is ensured via constrained inference using least squares minimization [14]. We modify the original algorithm and proofs to work for the non-regular trees that result from pruning (we omit details for brevity). Non-negativity is ensured by (a) setting negative counts to 0 (if a node is not a leaf node, all the nodes in its subtree are set to 0 as well); (b) normalizing the counts among children of the same parent node to obtain transition probabilities from the parent to the children; (c) computing the counts in a top-down approach from the original root count (obtained by consistency), given the stochastic transition probabilities from (b).

**4.4 Direction Weighted Sampling**

The order \( \ell \) Markov process captures global properties of the trajectories like direction, since the next value depends on the previous \( \ell \) values. However, the noise added for ensuring privacy can mask this behavior. We introduce a novel direction weighted sampling scheme to recover the directionality in the data.

We encode the neighboring cells in a grid relative to the current cell. For instance, in Figure 11, the cell \((12,7)_i\) in reference system \( \Sigma_i \) has eight neighboring cells, one in each compass direction: \((11,6)_i\) in the southwest direction, \((12,6)_i\) in the south direction, etc. In order to avoid sudden unrealistic changes of direction, we modify our sampling procedure to remember the recent trend. Each direction, \( \text{dir} \), has an equal weight when we start sampling a new trajectory, and the weight is updated along with the sampling process. Given the prefix of the trajectory generated so far, denoted by \( x \), we set the weight of the direction \( \text{dir} \) to \( w_{\text{dir}}(x) = \alpha^{c(\text{dir},x)} \), where \( \alpha \) is a weighting factor greater than 1. We count the number of times \( x \) has followed direction \( \text{dir} \) by \( c(\text{dir},x) \). However, in order to allow a long trajectory to turn locally, we use only the window of the last \( w \) moves of \( x \), denoted by \( x(w) \), to compute the weight of direction \( \text{dir} \), i.e. \( w_{\text{dir}}(x) = \alpha^{c(\text{dir},x(w))} \). Sampling with window size \( x = 0 \) corresponds to the basic sampling method without considering the direction of the prefix of the trajectory. We show in our experiments that direction weighted sampling with a reasonable window size can improve the utility of synthetic trajectories.

**4.5 End-to-End Privacy Analysis**

**Theorem 4.2.** The DPT system satisfies \( \epsilon \)-differential privacy.

**Proof Sketch.** Recall that we maintain normalized counts in the prefix trees and not the number of transitions so that adding or removing a user’s trajectory table results in a total change of 1 in the \( r \)th level of all prefix trees (for all \( r \)). This normalization does not require global knowledge and can be performed by each user independently. Such a transformation is called 1-stable [19], and does not violate differential privacy.

The private model selection procedure in Algorithm 2 uses the noisy count at the root of all the prefix trees in \( F_{\text{full}} \), and thereafter does not use the trajectory database. Since we normalize trajectories such that the sum of the weights associated with the transitions...
of a single user sum to 1, adding or removing a user results in a total change of 1 to the set of root counts. Hence, by adding Lap(1/εr) to each root count Algorithm 2 satisfies εr-differential privacy.

Noisy prefix trees are constructed by adding Laplace noise with sensitivity 1 to each of the prefix trees. The εr-budget is split geometrically across the levels. This again ensures εr-differential privacy. Adaptive pruning does not alter the differential privacy guarantee, since (a) pruning decisions are made after noise is added, and (b) after pruning the subtree below x, the count of x maybe re-estimated using unused privacy budget. Due to the composition properties of differential privacy, DPT ensures εr + εs = ε-differential privacy.

5. EMPIRICAL EVALUATION

In this section, we empirically evaluate DPT over two large trajectory datasets. The evaluation results are presented in two parts:

- Synthetic trajectories released using DPT are more useful and scalable compared to prior work in our end-to-end evaluations.
  
  We show that the synthetic trajectories mirror the original trajectories on three utility metrics — distribution of diameter (i.e., distance traveled), conditional distributions of destinations given starting regions, and frequent patterns.

- The individual algorithms powering DPT (namely differentially private model selection, adaptive pruning and direction weighted sampling) outperform alternate solutions derived from prior work.

We first describe our datasets and utility metrics in Section 5.1, then list the design choices for DPT in Section 5.2. Next, we present empirical end-to-end scalability and utility results in Section 5.3 followed by utility evaluation on components of DPT in Section 5.4.

5.1 Datasets and Utility Metrics

Table 2 summarizes the two large trajectory datasets used in our experiments. Given Σv, a full set of hierarchical reference systems of size 6 are considered for both datasets, i.e. Σv1, Σv2, ..., Σv6, where vmin = v_{m+1}/2. We will define Σv for each dataset below.

**Taxi dataset:** A set of GPS trajectories was recorded by 8602 taxi cabs in Beijing, China, during May 2009 [1]. The trajectories cover the region of Beijing within the bounding box (39.788N, 116.148W) and (40.093N, 116.612W) — approximately 34 km × 40 km. The finest resolution v1 considered is 100 m × 100 m. The raw sampling rate of these trajectories ranges from 30 seconds to 5 minutes. The dataset consists of approximately 4.3 million trips with passengers. Each trip is linearly interpolated into a sequence of location points at 30-second sampling intervals.

**Network dataset:** A set of trajectories synthesized by Thomas Brinkhoff’s network-based generator [2] of moving objects. The data consists of 300 files of 30000 individuals’ trajectories of length up to 1000 time units, sampled at equal time intervals. This data covers the region of Oldenburg, a city in Germany, within a bounding box of approximately 9 km × 10 km. The finest resolution v1 considered is about 50 m × 50 m.

We developed three utility metrics Qd, Qp, Qp_0 to evaluate whether synthetic trajectories Dsyn preserve aggregate properties of the original trajectories Dsyn. We sampled 5 sets of 50000 trajectories from the DPT model for Dsyn, and evaluated against 5 sets from Dsyn each of which consists of 50000 randomly selected raw trajectories. We then reported the relevant mean error or accuracy with its standard deviation. The three metrics are described below.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ε (km)</th>
<th>f_{min} (m)</th>
<th>f_{max} (m)</th>
<th>2σ (km)</th>
<th>v1 (m)</th>
<th>s_m (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Taxi</td>
<td>4,268.780</td>
<td>1919</td>
<td>20.03</td>
<td>34 × 40</td>
<td>100</td>
<td>306 × 463</td>
</tr>
<tr>
<td>Network</td>
<td>1.3 × 10^7</td>
<td>971</td>
<td>19.12</td>
<td>9 × 10</td>
<td>50</td>
<td>177 × 201</td>
</tr>
</tbody>
</table>

**Table 2: Datasets Summary**

| Metric 1: Qd - Diameter Distribution. 
| The diameter for a trajectory t = s_1, s_2, ..., s_n is the maximum distance between any pair of locations in t, i.e. max_{i,j} d(s_i, s_j) ∀i,j = 1, 2, ..., n. Let Qd(D) denote the empirical distribution of the diameter on trajectory database D, where the diameter is quantized into 25 equal width buckets \{ [0, x), [x, 2x), ..., [24x, 25x) \}. x is set to 800m for Taxi dataset, and 400m for Network dataset (both correspond to the resolution of Σv).

\[ \mathcal{E}_{diameter} = JSD(Q_d(D_{syn}), Q_d(D_{raw})) \]  

where JSD(·, ·) is the Jensen Shannon divergence with range [0, ln2].

**Metric II: Qp - Trip Distribution.**

Given a coarse grid Σv, a starting region a, is a square consisting of 3 × 3 cells from Σv. Let Da−ag denote the set of trajectories in D which starts from the region a, and ends at the cell a ∈ Σv. Trip distribution Qp measures the distribution of the ending point a in Da−ag given a, i.e. P(·|a, D). A set of 1000 starting regions S are randomly chosen. v is set to 4km for Taxi and 1km for Network dataset. The error in Qp is measured as

\[ \mathcal{E}_{trip} = \frac{1}{|S|} \sum_{a \in S} JSD(P(·|a, D_{syn}), P(·|a, D_{raw})) \]  

where JSD(·, ·) is the Jensen Shannon divergence with range [0, ln2].

**Metric III: Qp_0 - Frequent Patterns.**

We first project the path taken by each trajectory in D on a grid Σv, where the path is formed by connecting consecutive points with a straight line. We obtain the sequence of cells the trajectory passes through, and represent this sequence of cells with their corresponding anchor points in Σv, for m ∈ \{2, 3, ..., 5\}. The query Qp_0 takes in D and outputs the top p patterns from Σv, with the highest count, for k = 2, 3, ..., 6. We consider p to be 1000 and 10000 in our evaluations. The utility in Qp_0 of Dsyn w.r.t. Dsyn is measured as

\[ \mathcal{A}_{pattern} = F_1(Q_{p_0}(D_{syn}), Q_{p_0}(D_{raw})) \]  

where F1(·, ·) is the F1 score with range [0, 1] (i.e., harmonic mean of precision and recall) which measures the similarity between two item sets.

5.2 Design Choices

We consider alternative implementations of DPT by varying the structure of the RSS model (namely the reference system set used), the pruning strategy and the sampling method as shown in Table 3.

In the default setting [opt, adapt, dir], opt refers to the model output by private model selection, adapt refers to adaptive pruning, and dir refers to direction weighted sampling. Two alternate models are presented to opt: full and Σv, where full refers to the full model consisting of all the reference systems from Σv to Σv, Σv is not in full as we do not expect to see transitions in Σv, since transitions in Σv corresponds to movements at the speed of greater than 240 miles per hour. Alternatives to the pruning strategy are fixed which uses a fixed threshold of 0 (described in Section 4.3), and rgram, the pruning strategy used in [6]. np refers to the non-private model without noise. Lastly, direction-weighted sampling is studied by varying the window size Ω. nodir refers to the case when Ω = 0.
5.3 End-to-End Evaluation

In the end-to-end evaluation, we compare DPT with the most relevant prior work [6] denoted by seq. This prior work computes a noisy prefix tree on the entire domain by adding Laplace noise followed by pruning (sanitization phase), and then uses the noisy prefix tree to reconstruct a database of sequences (reconstruction phase). For seq, we transform the first \( \ell_{\text{max}} \) sampled points of each trajectory with a single reference system \( \Sigma_{v_m} \), for \( m = 1, \ldots, 6 \), where \( \ell_{\text{max}} \) is the truncation length used in seq to bound sensitivity as the input. Then this input is processed by the original code of seq provided by [6], resulting in a differentially private synthetic trajectory database. For end-to-end evaluations, we use the default setting of DPT—\{opt, adapt, dir\}.

Scalability. We run seq on a single machine with 2.40GHz 30MB Cache (256GB RAM - 48 hyperthreaded cores) for 5 times, and report the average time for sanitization and reconstruction. Similarly, we run DPT with its default setting for 5 times and report the average time for sanitization and sampling of 50000 trajectories. In Figure 6 on page 2 we show the runtime of seq with increasing resolution from \( v_6 \) to \( v_2 \) (a 4 \times increase in domain size each time) for \( \ell_{\text{max}} = 20 \) and \( k \)-grams for \( k \leq 4 \) when \( \epsilon = 1.0 \). The time for sanitization increases exponentially with the domain size, i.e. \( O(\Sigma^4) \), since seq considers the transitions between all pairs of points in the domain. In comparison, DPT uses the optimal model \( \{\Sigma_{v_4}, \Sigma_{v_5}\} \) with a fanout of constant size, which takes orders of magnitude less time to complete the sanitization process. In addition, seq aims to reconstruct a database of all possible trajectories by joining shorter trajectories to form longer ones. When a database has a large size, this approach takes a very long time. When resolution is \( \Sigma_{v_6} \), the reconstruction takes more than 5 hours to complete. We observe that the reconstruction time decreases from \( v_2 \) to \( v_1 \) and then increases again from \( v_3 \) to \( v_6 \). This is due to the fact that many consecutive points are mapped to the same cell indices in the coarse reference system. This allows the reconstruction process to form longer trajectories from shorter trajectories of repeated points. For instance, the maximum length of trajectories reconstructed in \( \Sigma_{v_6} \) reaches 20, while the maximum length in \( \Sigma_{v_5} \) is only 5. Therefore, a sampling approach is preferred as it can quickly produce a set of good representative trajectories with various lengths and can be parallelized easily. Results for seq at resolution \( v_1 \) are not reported as the program does not terminate even after a day.

Utility. In Figure 7 we show the JSD error for diameter distribution \( Q_d \), and trip distribution \( Q_t \), and \( F_1 \) score for frequent patterns \( Q^p \) respectively when \( \epsilon = 1.0 \). For both error metrics, as the resolution gets finer (from \( v_6 \) to \( v_2 \)), the error gets worse. DPT improves upon seq by > 80% on \( Q_d \) error and > 60% on \( Q_t \) error. For \( Q^p \), given the resolution at which seq is processed, we report the accuracy score of the top 1000 frequent patterns at the same resolution: both trajectories generated by seq and DPT are projected on \( \Sigma_{v_4} \) for the evaluation of frequent patterns. The comparison shows that DPT obtains better \( F_1 \) score at resolutions \( v_4 \) to \( v_3 \), with 30% to 200% improvements, even though DPT is not necessarily generated at \( \Sigma_{v_4} \). At \( v_2 \), DPT performs poorly because its optimal model selected is \{v_4, v_5\}, and hence it is unlikely to capture information at fine resolution. Experiments at smaller \( \epsilon \) also show that DPT outperforms seq. We omit the figures due to space constraints.

5.4 Component Utility Evaluation

In this section, we report the evaluation on three components: 1) private model selection, 2) adaptive pruning and 3) direction-weighted sampling on both Taxi (T) and Network (N) datasets. For each component, we will consider alternate choices defined in Section 4.2. The setting \{full, np, dir\} is chosen as the benchmark for the optimal non-private (np) model, because full is unbiased and close to optimal among all non-private models. All three utility metrics are measured and frequent patterns \( Q^p \) are reported at \( \Sigma_{v_4} \) for \( p = 1000 \) and at \( \Sigma_{v_5} \) for \( p = 10000 \).

Component I: Model Selection. Table 4 shows the optimal model selected by private model selection (Algorithm 2) on Taxi dataset. As \( k \) increases and \( \epsilon \) decreases, the signal to noise ratio gets weaker, and only a smaller subset of reference systems can be supported. This validates the monotonicity property from Section 4.2. We observe similar trends for Network dataset.

In addition, we have found that the optimal model performs considerably better than its alternatives. In Figure 8 we present two obvious alternate models: the private full model \{full, adapt, dir\} and the private coarsest reference system \{v_5, adapt, dir\}. The optimal model opt approached the high accuracy of the non-private full model \{full, np, dir\} as \( \epsilon \) increases. At \( \epsilon = 0.1 \) opt achieves accuracy/error within 85% of \{full, np, dir\} for Taxi dataset.

Though the non-private full model preserves good utility, the optimal model consistently beats the private version of the full model \{full, adapt, dir\} as the private full model incurs too much noise. Moreover, opt outperforms \( \Sigma_{v_5} \) at most settings. When \( \epsilon \) is large opt significantly outperforms \( \Sigma_{v_5} \). For instance, for the Network dataset, the optimal model for \( \epsilon = 1 \) is \{\( \Sigma_{v_5}, \Sigma_{v_4}, \Sigma_{v_5} \}\}, and thus captures a lot more finer details than \( \Sigma_{v_5} \), resulting in lower error. On the other hand, at small \( \epsilon \), \( \Sigma_{v_5} \) may be the optimal model (but opt picks a different model due to noise). In those cases, \( \Sigma_{v_4} \) can outperform opt. Furthermore, the good performance of \( \Sigma_{v_5} \) for \( Q^p \) and \( Q^t \) can be explained by the fact that \( Q^p \) and \( Q^t \) are measured at the same resolution as \( \Sigma_{v_5} \), or at a coarser resolution.

Component II: Adaptive pruning. We substitute the default setting of the second component (sanitizing method) — geometric noise with adaptive pruning methods (adapt), by its alternatives —
geometric noise with fixed threshold pruning at threshold 0 (fixed), and n-gram methods (ngram) [9]. In Figure 7, we present the evaluation results on synthetic trajectories generated from these three settings [opt, adapt, dir], [opt, fixed, dir], [opt, ngram, dir]. We observe that adapt performs just as ngram for $Q_4$, $Q_2$ on Taxi dataset, but gives better accuracy than ngram for other metrics. We conjecture two reasons for the poor performance of ngram: First, ngram can effectively estimate the right subtree height at each interval node and prune away subtrees with low support when the given tree height is large. However, our private model selection has considered the optimal tree height in opt and hence, the height estimation in ngram is no longer useful for opt. Second, the privacy budget is split equally amongst nodes on the root-to-leaf path which survive after ngram pruning. Uniform budget splits have been shown to be less accurate than geometric noise in [9]. Furthermore, fixed seems to perform as well as adapt. We further studied the noisy prefix trees output by the three pruning strategies. We find that fixed adds the most number of false nodes (nodes with true count 0) to the noisy trees leading to zigzag trajectories when sampled without dir. Direction-weighted sampling, however, lifts the accuracy of fixed.

Component III: Direction-weighted Sampling. The intuition behind direction-weighted sampling is to minimize unnecessary local turns in the synthetic trajectories. In this evaluation, we vary the direction setting (dir) in the default setting of DPT [opt, adapt, dir], by changing the window size $\omega$ from 0 to 32. $\omega = 0$ refers to no-direction weighted sampling (nodir). Default window sizes are set to 25% of the average length of raw trajectory dataset (6 and 16 for Taxi and Network respectively). As direction-weighted sampling has a larger impact on longer trajectories, we filter Taxi synthetic trajectories consisting of more than 15 sampled points for evaluation, where 15 is the median number of location points in the ground truth. For Network dataset has an average length greater than the maximum window size evaluated, we use all synthetic trajectories for evaluation. As shown in Figure 8, direction-weighted sampling re-enforces the directionality by smoothing the local turns, and hence results in lower errors and higher $F_1$ scores. For non-private model (np), the improvement by this direction constraint is not as significant as in the private models. For $\varepsilon = 0.5, 0.1$, the utility of private models improves as window size increases and stabilizes at a window size slightly greater than 4 (the order of the Markov process). The JSD errors for $Q_4$ and $Q_2$ are reduced up to 20% and the $F_1$ scores are improved up to 30%.
6. CONCLUSIONS

The problem of releasing realistic but private trajectory data takes on new importance as more mobility data is produced, and more pressures arise to share versions of the data. With DPT, we introduce the first system devoted to doing so for natural, speed-varying trajectories under differential privacy. Our analysis and empirical study demonstrate that this is an effective way to reveal information, while protecting privacy both in theory and in practice.

There is much opportunity for extending this line of work. While effective in practice, we have yet to show strong analytic guarantees for the utility of data released via DPT. We have focused on reference systems based on multi-resolution grids, but other systems could lead to better results for certain trajectories where paths are more constrained. Last, we focused on trajectories within the (two-dimensional) plane. As airborne sensors and unmanned aerial vehicles become more prevalent, it may be necessary to extend to three-dimensional trajectories, where the number of possible cells increases while the data density further decreases.

7. REFERENCES