A Differential Privacy-Preserving Data Publishing Algorithm for Bus Trajectory Analysis: A Case Study on BC Transit

by

Mahboubeh Bahari Neematabad

B.Sc. in Computer Engineering, Shariaty Technical University

A Thesis Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in the Department of Computer Science

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

A Differential Privacy-Preserving Data Publishing Algorithm for Bus Trajectory Analysis: A Case Study on BC Transit

by

Mahboubeh Bahari Neematabad

B.Sc. in Computer Engineering, Shariaty Technical University

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in the Department of Computer Science

Supervisory Committee

Dr. Yun Lu, Supervisor
Department of Computer Science

Dr. Alex Thomo, Departmental Member
Department of Computer Science

Abstract

The increasing use of trajectory data in location-based services and public transit planning highlights the high analytical value of such data. However, legal, technical, and especially privacy-related concerns have significantly limited public access to these datasets.

This thesis investigates privacy protection in trajectory databases—specifically, passenger movement data from public bus systems—under strong Differential Privacy (DP) guarantees.

We collaborate with BC Transit to make the first publicly available, privacy-preserving analysis of BC Transit's bus tap dataset from Victoria, British Columbia. This work reviews existing DP mechanisms and selects two practical and applicable algorithms for public transit data. These mechanisms are then adapted and optimized to suit the unique characteristics of such data. The goal is to evaluate their practical effectiveness in privacy-preserving publication of transit data while maintaining the utility required for meaningful analysis.

The BC transit bus tap dataset (containing bus tap-ins) enables already-useful analyses such as count or sum queries (e.g., number of visits to a bus stop) used as the benchmark of several related works. However, we aim to demonstrate the power of the state-of-the-art—privacy-preserving trajectory analyses, and with approval from our collaborators at BC Transit, we construct a plausible synthetic trajectory dataset that corresponds to the original given tap dataset based on known weekly role-specific travel patterns. Two privacy-preserving algorithms are then applied:

- **Noisy Prefix Tree** (Rui Chen et al., 2011): A prefix tree-based DP algorithm for sequential data.
- **PPDP** (Yang Li et al., 2020): An improved prefix tree algorithm tailored for transit smart card data.

We also compare the count queries on the original data using the Laplace mechanism with those on the synthetic trajectories, to evaluate how well basic utility is preserved.

For sequential transit data, we introduce the following technical improvements to enhance the effectiveness of prefix tree-based methods:

- A spatio-temporal dimensionality reduction technique to sample noisy nodes with better efficiency;
- An improved post-processing method for achieving consistency in the noisy prefix tree after noise injection.

In addition, A hybrid privacy budget allocation approach is employed, which balances tree depth with the actual distribution of nodes at each level in a more intuitive and effective manner.

Experimental results—conducted on synthetic trajectories generated from real-world tap card data from the BC Transit system—demonstrate that this framework can enforce strong privacy guarantees while answering complex transit-related analytical queries. This work serves as one of the first steps for data sharing among researchers, municipal agencies, and smart service developers, especially in BC, contributing to the design of more efficient, innovative, and human-centered public transportation systems.

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgment

I have been incredibly fortunate to receive the unwavering support, guidance, and friendship of many individuals throughout my two-year journey as a Master's student at the University of Victoria. This thesis is a tribute to all those who have made this such a fulfilling and enriching experience.

First and foremost, I would like to express my deepest gratitude to **my supervisor, Dr. Yun Lu**, whose exceptional mentorship has been invaluable. Her steadfast guidance and encouragement, both personally and professionally, have played a pivotal role in my academic development. I am profoundly grateful for her dedication.

I also wish to extend my heartfelt thanks to my committee members, **Dr. Alex Thomo** and **Dr. Amirali Baniasadi**, for their insightful feedback and thoughtful contributions, which have significantly enhanced the quality of this work.

My sincere appreciation goes to the University of Victoria for its financial and technical support, without which this research would not have been possible.

I would especially like to thank the Enterprise Data & Analytics team at BC Transit for their collaboration and support. In particular, I am grateful to the **Manager of Enterprise Data & Analytics** for facilitating access to the data; the **Senior Data Strategy Advisor** for handling the disclosure process; the **IT Data Analyst** for preparing and sharing the dataset; and the **Information Access and Privacy Officer** for guiding my initial request and connecting me with the appropriate team. Their responsiveness and cooperation were instrumental to this thesis.

Lastly, I am eternally grateful to **my spouse** for his unwavering support and encouragement, which sustained me throughout this journey. I also want to express my deepest thanks to **my dear son, Dayan**, whose patience and resilience during the many ups and downs of this process continually inspired me. His presence has been a source of motivation and strength.

Chapter 1

Introduction

Widespread collection and sharing of personal data have become an integral part of daily life. With the proliferation of smart devices and location-based services (LBS), vast amounts of location data are continuously being generated. While analyzing these data provides significant benefits, such as improving transportation efficiency and offering personalized services, it also introduces serious privacy risks that cannot be overlooked [1].

Trajectory data, which records the movement of individuals over time, plays a critical role in numerous applications, from navigation and route optimization to traffic management and urban planning [2]. Processing such data not only enhances daily experiences through services like navigation and route recommendation, but also has far-reaching implications for institutional data analytics in both public and private sectors [3]. The ability of personal devices (e.g., wearables, smartphones [4]) and navigation systems to accurately collect, process, and analyze this data, coupled with its ubiquitous availability, has driven rapid advancements. Traffic management, urban planning, transportation-system design, routing advice, and homeland security are just a few of the many applications that benefit from trajectory analyses.

Privacy Risks in Trajectory Data

Data analyses offer economic and societal benefits. With the continued rise of urbanization and growing dependence on public transportation systems, data-driven decision-making has become increasingly vital for optimizing transit services. Transportation data, especially trajectory data, is widely employed to optimize transportation systems and better understand individual movement patterns. concerns regarding privacy risks are growing [5], and analyzing such data poses significant risks to privacy, as trajectory data can unintentionally reveal sensitive information such as home locations and daily activities. Given the repetitive nature of human mobility, where individuals often follow predictable patterns, the risk of privacy breaches is heightened. Thus, protecting

user privacy becomes a crucial consideration when utilizing mobility data for analysis [6].

Human movement patterns are shaped not only by geographical constraints but also by social interactions. Human mobility traces are highly unique; even in datasets where the location of an individual is recorded hourly with spatial resolution as defined by carrier antennas, four spatio-temporal points are enough to uniquely identify 95% of individuals [7]. This demonstrates the vulnerability of even seemingly protected mobility data, which can easily lead to the re-identification of individuals.

Data gathered from location-based social networks indicate that individuals tend to move periodically between specific locations, such as home and work, over short distances. However, long-distance movements are more influenced by social ties. While these periodic mobility patterns provide valuable insights for transportation system analysis, All these factors highlight the critical need for robust privacy-preserving methods in this context [8].

Additionally, reconstruction attacks further exacerbate all the aforementioned privacy risks. Adversaries can use external information and advanced techniques like map-matching and spatio-temporal correlation to reconstruct a user's original trajectory, revealing sensitive locations such as their home or workplace, even if differential privacy protections are applied [9]. This highlights the need for privacy safeguards to continuously evolve in order to stay ahead of these sophisticated methods.

Similarly, the rising popularity of location-based services (LBS) on mobile devices has opened the door to a variety of location privacy attacks. These range from single-position attacks to multi-position correlation attacks, where attackers combine background knowledge with location data to deduce sensitive information.

Furthermore, the use of trajectory clustering to analyze movement patterns introduces additional privacy concerns. The uniqueness of trajectories and the spatial-temporal density of movements can unintentionally disclose private information, especially when clustering relies on frequently visited locations. This can inadvertently expose personal habits or routines, making it difficult to maintain a balance between data utility and privacy [10]. Persistent privacy concerns and legal constraints have made it nearly impossible to publicly release detailed passenger trajectory data. As a result, analysts, developers, and even transit authorities themselves are often limited to internal, closed analyses or forced to work with overly generalized, synthetic, or incomplete datasets. This lack of access presents a major obstacle to transparent decision-making, meaningful service improvement, and effective collaboration between governments, academia, and the private sector [11–13].

Various privacy-preserving mechanisms have been developed to address the privacy risks associated with publishing trajectory and mobility data.Traditional approaches such as k-anonymity [14–16], background knowledge attacks [17], the (k, δ)-anonymity model based on the inherent imprecision of sampling [18]. However, these methods often fail in the face of advanced inference

attacks and background knowledge. For example, k-anonymity is vulnerable to composition and foreground knowledge attacks, while obfuscation strategies can be circumvented through correlation with external data sources [19–24]. These limitations necessitate the adoption of stronger, provable privacy guarantees.

All the points mentioned above highlight the need for more advanced measures in bus trajectory data analysis to ensure strong user privacy protection while maintaining data accuracy and utility.

Differential Privacy for Trajectory Data

To address these challenges, differential privacy has emerged as the standard technique that provides strong privacy guarantees, regardless of the adversary's side information. Differential Privacy (DP) offers a formal mathematical guarantee, making it essential to clearly define the specific information it protects. The level of granularity at which DP is applied plays a critical role in understanding the scope of its privacy promises. DP promises that the behavior of an algorithm will be roughly unchanged even if a single entry in the database is modified [25].

However, this guarantee depends on how the concept of "neighboring databases" is defined. The original DP notion aims to protect the entire existence of an individual's records or entries in a database, thus assuming a one-to-one correspondence between record and user. However, in the case of bus trajectory data, where multiple data points collectively represent an individual's movement or each user's record, the notion of granularity becomes especially important. The definition of a neighbouring database, which informally describes *what* kind of information we wish to protect,significantly affects the privacy guarantees provided by DP.

This research aims to analyze various granularity notions within Differential Privacy (DP), especially in the context of bus trajectory data, in order to understand how different definitions of neighboring databases, such as *User-level*, *Event-level*, *W-event*, *ℓ-trajectory*, and *Element-level*, influence privacy guarantees. These granularity notions specify the level of sensitivity to changes in the data, which in turn impacts the amount of noise added. This balance between noise (to ensure privacy) and data utility is crucial to achieving strong privacy protection while retaining the usefulness of the data.

Each definition of neighboring databases provides a different level of protection. For example, in User-level privacy, the entire trajectory of an individual is considered as one unit. In contrast, Event-level privacy offers finer granularity by considering each spatio-temporal point individually. W-event privacy further extends this by grouping multiple consecutive points, ℓ-trajectory privacy defines a sequence of consecutive spatio-temporal points, and Element-level privacy focuses on data clusters, such as locations that share similar characteristics. These variations affect how much noise needs to be added to ensure privacy while preserving data utility.

The implementation in this work adopts the *User-level* granularity. Additionally, to provide a

better understanding of granularity choices, common notions such as *Event-level* and ℓ-trajectory are conceptually introduced and discussed.

In public transportation systems, where trajectory data is extensively collected for tasks like traffic management and route optimization, understanding how different definitions of neighboring datasets influence this balance is vital. Our research aims to provide insights into the optimal application of DP for bus trajectory data, ensuring robust privacy preservation without sacrificing the accuracy needed for effective data analysis.

In the context of public transportation systems, where trajectory data is frequently collected for purposes such as traffic management and route optimization, understanding how these definitions influence privacy and utility is vital. Our research will investigate how different neighboring definitions affect the balance between privacy preservation and the accuracy of bus trajectory data analysis, aiming to provide insights into the optimal application of differential privacy in this domain.

1.1 Motivation

To address the privacy constraints when releasing analyses on trajectory data, Differential Privacy (DP) [3] has emerged as the standard technique that provides strong privacy guarantees, regardless of the adversary's side information.

Challenges in Applying DP to Trajectory Data

However, applying DP to trajectory data poses several challenges and is known *hard* problem [26]. As highlighted in Buchholz et al. (2024), trajectory datasets are inherently high-dimensional and sparse. Each individual's movement trace may vary greatly in length, frequency, and visited locations, making it difficult to apply standard DP techniques effectively.

The high dimensionality stems from the fact that each trajectory comprises a sequence of location-time pairs, and the number of possible sequences grows exponentially with the number of locations and timestamps. Consequently, even relatively small datasets can span an enormous domain space, making statistical analysis and DP-compliant aggregation computationally challenging.

Moreover, sparsity is a critical concern. While the space of all possible trajectories is vast, real-world user trajectories typically occupy only a small subset. This leads to low-frequency patterns and insufficient support for aggregation-based or histogram-based methods. Moreover, transportation datasets often contain no taxonomy trees or hierarchical relationships among station names, limiting the applicability of standard DP methods that rely on such structures.

As a result, a significant amount of noise must be added to meet DP guarantees, which in turn

severely degrades utility.

These challenges underscore the need for mechanisms that are not only privacy-preserving but also structure-aware. Techniques such as prefix trees, generative models, and structure-aware noise addition have been explored in the literature review. However, achieving a good balance between utility and privacy remains an open and active area of research.

State-of-the-Art (SoTA)

Recent efforts have focused on designing data-dependent differential privacy (DP) mechanisms tailored to trajectory data, The two most relevant of which are the Noisy Prefix Tree (Chen et al., 2011) [27] and the Privacy-Preserving Data Publishing (PPDP) algorithm (Yang Li et al., 2020) [28]. These methods construct a prefix tree to represent trajectories, add Laplace noise to node counts, and then apply constrained inference techniques to improve data utility.

Challenge in applying SoTA to BC transit dataset

Many of the DP-based mechanisms proposed for publishing location and mobility data—whether specifically designed for transit smart card data or developed for other types of trajectory data—often remain theoretical or are tested solely on synthetic datasets that fail to capture the complexity and constraints of real-world public transportation systems [3, 28]. In practice, organizations like BC Transit continue to refrain even from releasing aggregated and anonymized versions of their mobility data, due to unresolved privacy concerns. This thesis emerges from a real and operational need: BC Transit possesses large volumes of data, but privacy remains a central barrier to its public use. Moreover, a major limitation of many prior works is that they developed and tested their algorithms only under isolated conditions with custom datasets, rather than in consistent experimental environments using real-world data and unified parameters [27]. This makes it difficult to fairly compare different approaches.

Goal of this thesis

Our goal is to demonstrate that it is possible to answer meaningful transit-related queries—such as those supporting bus service improvements in the Greater Victoria area—without compromising individual privacy. This work aims to help organizations like BC Transit gain the confidence and technical tools needed to share their data securely and responsibly with the research community and the public. In this thesis, we review a widely cited set of prefix-tree–based DP mechanisms with a focus on their applicability to the structure and constraints of real-world bus transit data. We then improve the relevant SoTA, implement these mechanisms, and evaluate them using both the BC Transit tap dataset, and a plausible synthetic dataset that corresponds to and is approved by BC Transit.

1.2 Research Questions (RQs)

Based on the background, motivation, and goals discussed above, this thesis investigates the following research questions. We revisit and answer them in Chapter 6.

- RQ1: Is it possible to publish accurate but privacy-preserving analyses of BC Transit data? What is the impact of privacy mechanisms on accuracy?
- RQ2: What is the effect of Privacy Mechanisms on Sum Queries vs. Count Queries and and more complex queries, such as *Frequent Sequential Pattern Mining* (FSPM)?
- RQ3: What are recommendations for BC Transit for future publication of privacy-preserving data analyses?

1.3 Thesis Contributions

In this thesis, we address the problem of releasing high-utility private trajectory data from tap-in only public bus records under formal differential privacy guarantees.

To overcome the limitations of existing approaches, we introduce the following contributions:

- **Collaborating with BC Transit to access real passenger tap data and create a synthetic trajectory dataset based on that tap data.** We use real-world data from the BC Transit system in Victoria, British Columbia, to design and implement a custom data generation framework that synthesizes realistic passenger mobility patterns. We incorporate domain knowledge such as weekly travel behavior, temporal regularities, and role-based patterns (e.g., student, full-time worker) to reconstruct plausible trajectories. We also create a novel role-based behavioral model that reconstructs realistic sequences from tap-only data.
- **Conducting experiments that are more meaningful to BC Transit.** We apply two state-of-the-art DP mechanisms from the literature—one based on prefix trees and the other an efficient privacy-preserving data publishing (PPDP) algorithm—to sanitize the reconstructed data and ensure privacy. Then, we run not only count queries (often used as the only benchmark in previous work), but also more complex, revealing queries on the privatized dataset.
- **A comprehensive evaluation of DP noise effects across query categories.** We analyze how adding noise for privacy affects the accuracy of different types of queries—the **first work to do so on both real data and more complex queries**.

Technical Contributions:

- **Post-processing improvement.** We enhance the original prefix tree mechanism by introducing a weighted averaging strategy for post-processing. Instead of equally distributing the discrepancy between parent and child nodes (as done in the Chen et al., 2011 [27] method), our approach assigns weights based on the noisy counts of the child nodes, preserving more accurate local structures and reducing error propagation.

- **Faster sampling of new nodes.** We implement an adapted spatio-temporal pruning technique [28] to significantly reduce the candidate node space during prefix tree construction. By narrowing the search space based on domain-informed temporal and spatial constraints, we accelerate the sampling process without compromising utility.

Through these contributions, we bridge the gap between academic research on privacy and the real-world challenges of urban mobility data sharing. We show that it is possible to generate meaningful, privacy-preserving synthetic data that supports public transparency and operational decision-making without compromising individual privacy.

1.4 List of Notations

The list of notations used in this thesis is provided in Table 1.1.

Symbol	Description
ϵ	Privacy budget controlling the strength of differential privacy
$\bar{\epsilon}$	Per-level privacy budget
D	Original trajectory dataset
D'	Sanitized (privatized) trajectory dataset
T	A trajectory (sequence of locations)
$\mathcal{L}$	The set of all distinct locations (stations) within the transit network
$l \in \mathcal{L}$	A single location (station) in the set of locations $\mathcal{L}$
f	A query function applied to the dataset
Δf	Global sensitivity of function f
$\text{Lap}(\lambda)$	Laplace distribution with scale parameter $\lambda = \Delta f/\epsilon$
PT	Prefix tree built from trajectories
$\text{prefix}(v, PT)$	Prefix path represented by node v in the prefix tree PT
$\text{tr}(v)$	Set of trajectories sharing the prefix represented by node v
$c(v)$	True (raw) count of trajectories in node v
$\hat{c}(v)$	Noisy count at node v, obtained by adding Laplace noise
$\bar{c}(v)$	Adjusted (sanity-checked) count for node v used in post-processing
s	Sanity bound in relative error (prevents inflated error for very small counts)
ℓ_i	Privacy budget allocated to level i of the prefix tree
$\text{Root}(PT)$	Virtual root node of the prefix tree
k	Number of empty nodes that passed the boolean test during construction
$\lvert D\rvert$	Size of the trajectory dataset
$\lvert\mathcal{L}\rvert$	Number of distinct locations
$\lvert T\rvert$	Length of a trajectory
U	Candidate set of child nodes generated from $\mathcal{L}$ during prefix tree construction
$\text{NoisyCount}(x, \epsilon)$	Returns $x + \text{Lap}(1/\epsilon)$
θ_i	Pruning threshold at level i
h	Height of the prefix tree
b	Baseline threshold (minimum cutoff)
$\max\{\lvert Q\rvert\}$	Maximum query size (bounded by tree height h)
k	Decay factor controlling how θ_i decreases with depth

Table 1.1: List of notations used in this thesis.

1.5 Thesis Outline

The outline of the thesis is as below:

- **Chapter 2: Preliminaries**
  Introduces key concepts of differential privacy (DP), including basic definitions, the Laplace mechanism, neighboring databases, and essential mathematical foundations.
- **Chapter 3: Literature Review**
  Reviews foundational and advanced DP algorithms for trajectory data, including different neighboring definitions, prefix tree representations, and comparisons of existing methods.
- **Chapter 4: Methodology**
  Describes the selected algorithms, dataset (BC Transit), experimental setup, evaluation metrics for privacy and utility, and parameter choices.
- **Chapter 5: Results and Analysis**
  Presents experimental findings, visual comparisons, and an analysis of utility-privacy trade-offs across tested mechanisms.
- **Chapter 6: Conclusion**
  Summarizes the work, discusses key contributions, and outlines directions for future research.

Chapter 2

Preliminaries

Public transit systems generate vast amounts of spatio-temporal data through smart card logs, GPS tracking, and vehicle logs. These datasets are often stored in what is known as a *trajectory database*, which records sequences of timestamped locations representing the movement of individuals or vehicles over time. More formally, a trajectory database is a collection of data records, each of which describes the historical movement of a moving object within a time interval [2].

Trajectory databases are a subset of spatio-temporal databases designed to model, store, and query data that changes over time [2]. Due to their importance in real-world applications such as urban traffic analysis, transportation planning, and location-based services, trajectory databases have gained significant attention in recent years [9, 29]. However, they often contain sensitive information, and analyzing or publishing them poses substantial privacy risks.

Differential Privacy (DP) [30] has emerged as the leading standard for privacy-preserving data publication. It offers strong theoretical guarantees that the output of an analysis will not change significantly by including or excluding any single individual's data, thereby minimizing privacy leakage [25]. This makes DP a promising tool for releasing aggregate insights from trajectory data while safeguarding individual privacy.

In the remainder of this chapter, we introduce foundational concepts and definitions needed to understand and implement DP mechanisms in the context of trajectory data publication. This includes a formal definition of DP, the Laplace mechanism, neighboring database definitions, a brief overview of the Exponential mechanism, and basic mathematical foundations used throughout this thesis.

Background Definitions

Before we go into the details, this section introduces key terms and definitions that will be used or adapted throughout the thesis. We provide short explanations of the main concepts and methods

discussed in later chapters.

2.1 Differential Privacy

Traditional disclosure limitation mechanisms are designed to prevent the release of sensitive information that could potentially be exposed. These methods often rely on assumptions about the limited background knowledge that an adversary might possess. While these techniques—such as suppression, generalization, and data swapping—can reduce disclosure risks, they do not offer formal guarantees. If an attacker has access to more auxiliary information than anticipated, privacy breaches may still occur.

Two prominent real-world examples illustrate this vulnerability:

- **Netflix Prize Re-identification (2008)**: Netflix released an "anonymized" dataset containing over 100 million movie ratings from around 500,000 users as part of a public competition to improve its recommendation system. Although names were removed, the dataset included user IDs, movie IDs, ratings, and timestamps. Researchers Narayanan and Shmatikov successfully de-anonymized many users by linking this dataset with public IMDb profiles. With only 6–8 known ratings and approximate dates, they could re-identify individuals with up to 96% accuracy. This breach led to the cancellation of a planned second Netflix Prize and legal action against the company. [25]

- **Massachusetts Medical Records (1997)**: Latanya Sweeney demonstrated that supposedly anonymized hospital discharge data released by the state of Massachusetts could be re-identified using publicly available voter registration records. By matching ZIP code, birthdate, and gender, she was able to uniquely identify Governor William Weld's medical records. Her analysis revealed that 87% of the U.S. population could be uniquely identified using just these three demographic attributes. [25]

These cases became foundational examples of the weaknesses of traditional anonymization and motivated the development of stronger privacy models such as *k-anonymity*, and *l-diversity*. However, even these models rely on assumptions about attacker knowledge and cannot guarantee privacy under worst-case conditions. [25]

Differential Privacy (DP) introduces a fundamentally different approach to disclosure limitation [30]. Rather than enforcing a predefined set of rules to mitigate disclosure risks, DP directly limits the influence that the presence or absence of any single data record can have on the outcome of analyses derived from the database.

Initially, differential privacy was introduced as a query-response mechanism [31]. In this model, the data owner delegates query handling to a trusted component, referred to as the *dif-*

ferential privacy interface. This interface processes incoming queries from users and returns noisy answers—answers that have been modified to preserve privacy—ensuring differential privacy guarantees.

Differential privacy mechanisms can be implemented in **interactive** or **non-interactive** forms:

- **Interactive mechanisms:** In these mechanisms, the data owner manages the database, and user access is mediated solely through the differential privacy interface. Users submit queries to the interface, which retrieves the true response from the database, applies a differential privacy mechanism (e.g., by adding noise), and returns the result.

  Limitations: These interactive systems have some limitations. Each query consumes part of the privacy budget, meaning only a limited number of queries can be safely answered. Moreover, because noise is added independently for each query, repeated queries may yield different results. Adversaries might exploit this inconsistency to infer the true answer.

- **Non-Interactive mechanisms (privatized dataset) [32]:** To overcome these limitations, **non-interactive mechanisms** apply noise to the data only once and publish the privatized dataset. This allows unlimited queries to be answered consistently and efficiently. Identical queries will always produce the same result.

Figures 2.1 and 2.2 illustrate the differences between interactive and non-interactive approaches.

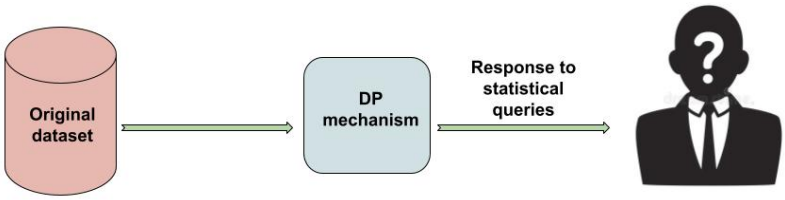

Figure 2.1: Interactive differential privacy mechanism.

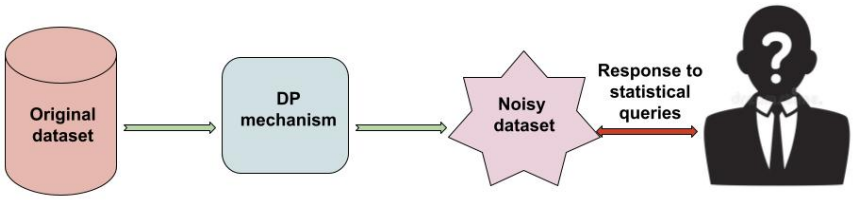

Figure 2.2: Non-interactive differential privacy mechanism.

In recent years, researchers have also begun exploring the use of differential privacy for publishing entire datasets, and several algorithms have been developed for this purpose.

The key idea is that the statistical results produced from a dataset should remain nearly unchanged regardless of whether any particular individual's record is included. This property ensures that an adversary cannot determine the presence or absence of any specific record, even when comparing outputs from neighboring databases (databases that differ in some small way, which captures the kind of information we'd like to protect in the database; For a detailed discussion of neighboring database definitions—particularly in the context of trajectory data—see the next section 2.2.

In the following section, we provide the formal definitions and concepts underlying differential privacy.

2.2 Neighboring Database Definitions

One of the most important concepts in differential privacy (DP) is the idea of *neighboring databases*. The definition of neighboring databases determines what kind of information is protected by a DP mechanism and how privacy is measured [25]. This is also called the *level of granularity*, and it refers to how much information can differ between two databases that are considered neighbors.

Two datasets are considered neighboring if they differ by a small part of the data — for example, one person's whole trajectory, a single spatio-temporal point, or a short sequence of points — depending on how privacy is defined.

Type of privacy	**Difference between neighboring databases**
User-level	A user's whole trajectory
Event-level	A spatio-temporal point visited by a user (an event)
w-event	A window of events over w consecutive timestamps
ℓ-trajectory	A sequence of ℓ consecutive spatio-temporal points from a single user
Element-level	A user's set of points belonging to the same cluster

Table 2.1: Granularity notions and their concept of neighborhood.

Different granularity levels have been proposed for trajectory data:

- **User-level privacy:** Two databases are neighbors if they differ by the entire data of one user (e.g., a full trajectory).
- **Event-level privacy:** Neighbors differ by only one spatio-temporal point (e.g., one stop at one time).
- **w-event privacy:** Neighbors differ in a short sequence of points within a time window of size w.

- **ℓ-trajectory privacy:** Neighbors differ by a sequence of ℓ consecutive spatio-temporal points, regardless of time.
- **Element-level privacy:** Neighbors differ by the presence or absence of points from certain categories or clusters (e.g., "hospital").

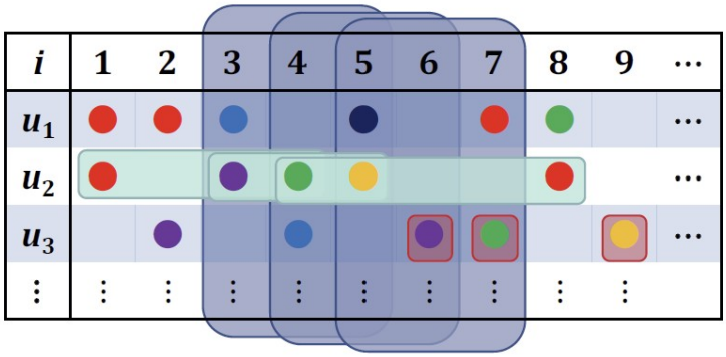

Figure 2.3: An example of a non-periodically recorded streaming database. Colored dots represent different locations. The rounded boxes represent protection scopes of event-level (red), w-event (blue), and ℓ-trajectory privacy (green), for $w = \ell = 3$. Observe that the blue box (w-window) always spans w timestamps independent of how many points they include, and that the green box (ℓ-trajectory) always includes ℓ points independently of the number of timestamps it spans.

Each of these definitions provides a different trade-off between privacy and utility. For example, user-level privacy is the strongest but often leads to high information loss. Event-level privacy allows more utility but can be vulnerable to identity or attribute disclosure [3].

In this thesis, we follow the **user-level** definition of neighboring databases, which is the most popular choice in trajectory-related works [3, 25, 27] … and also represents the granularity used in most non-trajectory differential privacy mechanisms, as discussed in *The Algorithmic Foundations of Differential Privacy* [25]. Informally, this ensures privacy for individual trajectories. That is, we assume each record in the database is a complete trajectory of one passenger. Two databases are considered neighbors if they differ by one user's entire trajectory. This means the DP algorithms used in this work aim to protect the privacy of the full movement history of any single user.

2.3 Differential Privacy (DP) Definition

Differential privacy [30] is a rigorous and mathematical framework for measuring the privacy guarantees provided by data analysis or machine learning algorithms. It aims to ensure that the inclusion

or exclusion of any individual's data does not significantly affect the outcome of the analysis, thus protecting the privacy of individuals in the dataset. In simpler terms, it provides a quantifiable measure of how much privacy is preserved when analyzing a dataset.

Intuition about ϵ. A smaller ϵ means the algorithm adds more noise, which provides stronger privacy protection but reduces the accuracy and utility of the results. A larger ϵ adds less noise, resulting in higher accuracy but weaker privacy protection.

Definition 2.3.1 (Differential Privacy)**.** Let $\epsilon > 0$ be a parameter representing the desired level of privacy, and $\mathcal{M}$ be a randomized algorithm that takes a dataset as input and produces an output. Then, $\mathcal{M}$ satisfies ϵ-differential privacy (ϵ-DP) if for all pairs of neighboring datasets $\mathcal{D}$ and $\mathcal{D}'$, which differ in at most one individual's data, and for all subsets of possible outputs S, the following holds:

$$\Pr\left[\mathcal{M}(\mathcal{D}) \in S\right] \leq e^{\epsilon} \times \Pr\left[\mathcal{M}(\mathcal{D}') \in S\right]$$

where $\Pr\left[\mathcal{M}(\mathcal{D}) \in S\right]$ represents the probability that the algorithm outputs a result in set S when given dataset $\mathcal{D}$, and e is Euler's number (approximately equal to 2.71828).

In essence, this definition states that the ratio of probabilities of observing any output set S when analyzing datasets $\mathcal{D}$ and $\mathcal{D}'$ (which differ only in the data of one individual) is bounded by e^{ϵ}. A smaller value of ϵ indicates a stronger level of privacy protection, as it limits the impact of any single individual's data on the analysis outcome [25].

2.4 Differentially Private Mechanisms

There are several core mechanisms within the framework of Differential Privacy, including the Laplace mechanism, the Exponential mechanism, and the Gaussian mechanism. Among these, the Laplace mechanism is one of the most fundamental and widely used techniques. It is particularly suitable for scenarios involving numerical queries, where the output of a function lies in the real domain.

Due to its conceptual simplicity, analytical tractability, and ease of implementation, the Laplace mechanism has become a standard tool in differentially private data analysis. In this thesis, we adopt the Laplace mechanism and apply Laplace noise to ensure privacy guarantees in our data publishing process.

Laplace Mechanism

One of the most fundamental types of queries in differentially private data analysis involves numeric functions $f : \mathbb{N}^{|X|} \to \mathbb{R}^k$, which return a vector of k real numbers from a dataset. The amount of

noise required to answer such queries privately depends on how much a single individual's data can change the result — a property called ℓ_1-sensitivity.

ℓ_1-Sensitivity

The ℓ_1-sensitivity [25] of a function f is defined as:

$$\Delta f = \max_{\substack{\mathcal{D},\mathcal{D}' \in \mathbb{N}^{|X|} \\ \|\mathcal{D}-\mathcal{D}'\|_1=1}} \|f(\mathcal{D}) - f(\mathcal{D}')\|_1$$

This measures the maximum change in the output of f when one individual's data in the database is added, removed, or modified. It captures the worst-case impact of a single data record, and determines how much uncertainty (i.e., noise) must be added to preserve privacy.

Laplace Distribution

To add noise, we use the *Laplace distribution*, which is symmetric and centered at zero. The probability density function (PDF) of the Laplace distribution with scale parameter b is:

$$\text{Lap}(x \mid b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$$

The variance of this distribution is $2b^2$. We denote a random variable drawn from this distribution as $\text{Lap}(b)$.

Definition 2.4.1 (Laplace Mechanism [25].)**.** Given a function $f : \mathbb{N}^{|X|} \to \mathbb{R}^k$, the *Laplace Mechanism* is defined as:

$$\mathcal{M}_L(\mathcal{D}, f, \varepsilon) = f(\mathcal{D}) + (Y_1, Y_2, \ldots, Y_k)$$

where each Y_i is an independent random variable drawn from the Laplace distribution with scale $\Delta f/\varepsilon$.

This means we compute the true output of $f(\mathcal{D})$, then add Laplace noise to each component of the output vector. The amount of noise depends on the sensitivity Δf and the privacy parameter ε, where smaller ε indicates stronger privacy.

Privacy Guarantee

The Laplace mechanism satisfies $(\varepsilon, 0)$-differential privacy. Intuitively, this means the output distribution of the mechanism does not change significantly whether or not any one individual's data is included in the input. The formal proof uses the triangle inequality [25] and the definition of

ℓ_1-sensitivity to show that the probability densities for two neighboring datasets differ by at most a factor of e^{ε}.

Exponential Mechanisms

Although not used in this study, the Exponential mechanism is briefly presented for completeness. Its inclusion provides context for why the Laplace mechanism—more suitable for our numerical data—was selected.

The exponential mechanism is used when outputs are non-numeric or when adding noise directly may reduce result quality. It selects outputs based on a utility function $u : \mathbb{N}^{|X|} \times \mathcal{R} \to \mathbb{R}$, which assigns a quality score to each output $r \in \mathcal{R}$ for a given dataset $\mathcal{D}$.

The sensitivity of the utility function is:

$$\Delta u = \max_{r \in \mathcal{R}} \max_{\|\mathcal{D} - \mathcal{D}'\|_1 \leq 1} |u(\mathcal{D}, r) - u(\mathcal{D}', r)|$$

Definition 2.4.2 (The Exponential Mechanism [25])**.** Given a utility function $u(\mathcal{D}, r)$, where $\mathcal{D}$ is the dataset and $r \in \mathcal{R}$ is a potential output, the *Exponential Mechanism* $\mathrm{ME}(\mathcal{D}, u, \mathcal{R})$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to:

$$\exp\left(\frac{\varepsilon \cdot u(\mathcal{D}, r)}{2\Delta u}\right)$$

where Δu is the sensitivity of the utility function.

The exponential mechanism satisfies $(\varepsilon, 0)$-differential privacy [25]. Furthermore, it ensures a utility guarantee: with high probability, it outputs a result whose utility is close to the optimum. The probability of selecting an output significantly worse than the optimal decreases exponentially with its utility gap.

2.5 Trajectory Data: Definition and Challenges

Introduction and Formal Definition

Trajectory data refers to the movement history of individuals or objects through space and time, captured as ordered sequences of spatio-temporal points. Each point typically includes a location and a timestamp, forming a chronologically ordered trace of behavior.

Definition 2.5.1. A trajectory T of length $|T|$ is defined as an ordered list of locations [27, 33]:

$$T = t_1 \to t_2 \to \cdots \to t_{|T|}$$

where $t_i \in L$, and $L = \{L_1, L_2, ..., L_{|L|}\}$ is the universe of discrete locations. Locations may repeat and can appear consecutively. For example, given $L = \{L_1, L_2, L_3, L_4\}$, a valid trajectory could be:

$$T = L_1 \rightarrow L_2 \rightarrow L_2$$

When time is included, each element becomes a spatio-temporal point:

$$T = \langle (x_1, y_1, t_1), (x_2, y_2, t_2), ..., (x_n, y_n, t_n) \rangle$$

ensuring $t_{i+1} > t_i$.

Trajectory ID	**Path**
T1	L1 $\rightarrow$ L2 $\rightarrow$ L3
T2	L1 $\rightarrow$ L3
T3	L3 $\rightarrow$ L1 $\rightarrow$ L2
T4	L2 $\rightarrow$ L1 $\rightarrow$ L4 $\rightarrow$ L2
T5	L1 $\rightarrow$ L4 $\rightarrow$ L2
T6	L2 $\rightarrow$ L3 $\rightarrow$ L1
T7	L4 $\rightarrow$ L2 $\rightarrow$ L2
T8	L1 $\rightarrow$ L4

Table 2.2: Example of a simple trajectory database (only contains the spatial component (locations)).

Table 2.2 presents a sample trajectory database with $L = \{L_1, L_2, L_3, L_4\}$. From this dataset, a prefix tree can be constructed, which is illustrated in Fig. 2.4.

Categorizing Trajectory Data

Trajectory data can be collected from:

- GPS devices in phones or vehicles
- Smart card systems in public transit
- Wi-Fi and RFID-based location tracking
- Location-based mobile applications

In this thesis, we use smart card data from Victoria, BC, which includes only tap-in records.

Types of Trajectories

- **Raw Trajectory**: Spatio-temporal sequence like $\langle (x_1, y_1, t_1), ..., (x_m, y_m, t_m) \rangle$
- **Simplified Trajectory**: Spatial-only, time is omitted
- **Semantic Trajectory**: Points include semantic labels (e.g., "Home", "Work")
- **Multi-Aspect Trajectory**: Includes weather, transport mode, or emotional data
- **Streaming Trajectory**: A flow of time-indexed updates $D = \{S_1, ..., S_t\}$

Structure of Trajectory Databases

Trajectory data is typically organized as:

$$D = \begin{cases} T_1 : p_1^{(1)}\, p_2^{(1)} \cdots p_{m_1}^{(1)} \\ T_2 : p_1^{(2)}\, p_2^{(2)} \cdots p_{m_2}^{(2)} \\ \vdots \\ T_r : p_1^{(r)}\, p_2^{(r)} \cdots p_{m_r}^{(r)} \end{cases}$$

Each row is a user's trajectory. Some datasets are regularly sampled; others are sparse.

Key Characteristics of Trajectory Data

- **Spatio-temporal**: Each point has both time and location
- **Sequentiality**: Order of events matters (A→B→C ≠ C→B→A)
- **High Dimensionality**: Long sequences increase complexity
- **Correlation**: Logical or social behavior causes point dependencies

Sequential vs. Non-Sequential Data

In this thesis, we focus on sequential data, as it preserves the temporal and behavioral patterns essential for our modeling and analysis. While non-sequential data exists and may be used in other contexts, it is not considered in our approach.

- **Sequential:** Maintains order, used for behavior modeling
  Example: Home → University → Work

- **Non-Sequential:** Set of locations without order
  Example: {Home, University, Work}

ID	Trajectory
tr_1	1Y $\rightarrow$ 4X
tr_2	2X $\rightarrow$ 3Z
tr_3	2X $\rightarrow$ 3Z $\rightarrow$ 4Y
tr_4	2Y $\rightarrow$ 4X
tr_5	2Y $\rightarrow$ 3Z
tr_6	3X $\rightarrow$ 4Y
tr_7	1Z $\rightarrow$ 2X $\rightarrow$ 3Z
tr_8	1Z $\rightarrow$ 4X

Table 2.3: Trajectory dataset (includes both spatial (locations) and temporal (timestamps) components of trajectories.).

Table 2.3 presents an example of a trajectory dataset consisting of eight individual trajectories. For instance, the first trajectory, tr_1, represents a movement from location **Y** at time slot **1** to location **X** at time slot **4**. It is important to note that the timestamps t_i in each trajectory are strictly increasing, ensuring that $t_i < t_{i+1}$ and thereby preserving the correct temporal order of events.
From this dataset, a corresponding prefix tree can be constructed to represent the spatio-temporal sequences, as shown in Fig. 2.5.

Privacy Risks

Trajectory data is highly sensitive. Studies show that knowing only 3–4 location-time points is often enough to uniquely identify individuals, even without names or IDs.

Privacy threats include revealing:

- Home/work location
- Daily routines
- Visits to sensitive places (clinics, religious centers)
- Social ties (people with similar trajectories)

Real-World Applications

Domain	Use Cases
Public Transit	Route planning, peak load detection, underserved route identification
Urban Planning	Understanding mobility, infrastructure planning
Commerce	Geo-targeted advertising, business location optimization
Healthcare	Disease tracking, evaluating access to services
Mobility Research	Travel simulation, behavioral modeling

Table 2.4: Applications of trajectory data in real-world domains.

Research Focus in This Thesis

This thesis focuses on smart card tap-in data from the BC Transit system in Victoria, BC. Since only boarding locations and times are available (no tap-outs), reconstructing realistic trajectories becomes a challenge.

To ensure privacy while preserving analytical value, we apply differential privacy techniques to generate synthetic trajectory data that:

- Protects individual behavior
- Preserves meaningful travel patterns
- Handles large, sparse, and high-dimensional datasets

2.6 Prefix Tree Mechanism For Trajectory Data

The Prefix Tree Mechanism [27] is one of the most prominent approaches for publishing sequential data, such as mobility trajectories, under differential privacy. This technique is specifically designed to model spatio-temporal sequences in trajectory data, where each passenger's path is represented as a route from the root to a leaf node in a tree structure.

Each node in the tree corresponds to a prefix of a trajectory, consisting of a timestamp-location pair (l_i, t_i), and maintains a *count* indicating how many trajectories have passed through that prefix.

A trajectory prefix tree is formally defined as a triplet:

$$PT = (\text{Root}, E, V)$$

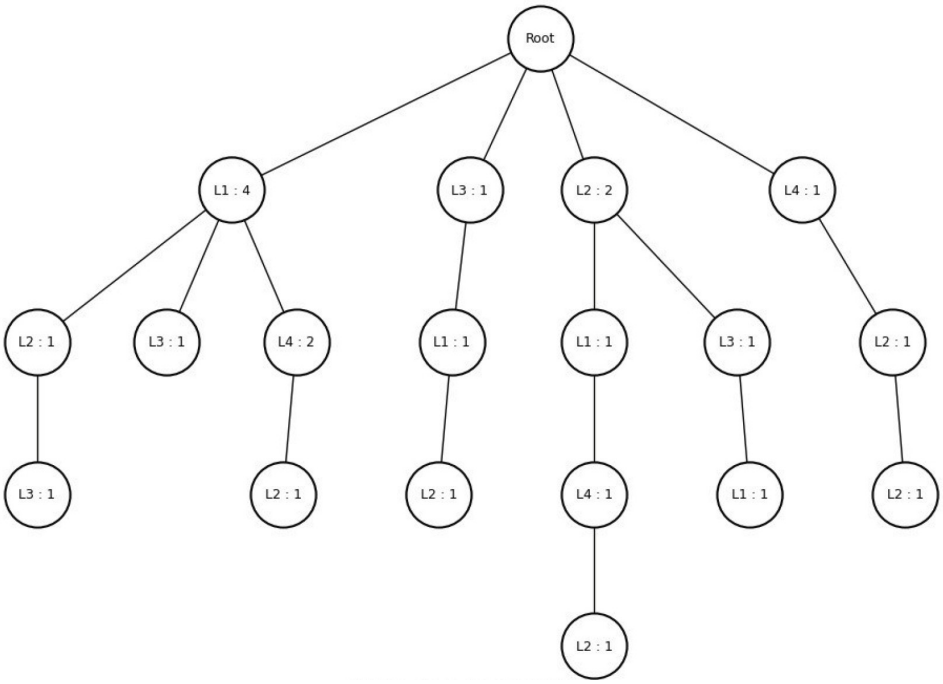

Figure 2.4: The prefix tree constructed from the sample trajectory database in Table 2.2

where:

- **Root** is the root node of the tree, storing the total count of all trajectories in the dataset.
- **V** is the set of nodes, each containing the count of trajectories that pass through the path from the root to that node.
- **E** is the set of edges, each representing a timestamp-location pair, connecting parent nodes to child nodes.

In this structure:

- *In-bound edges* define the link from a node to its parent.
- *Out-bound edges* define the link from a node to its children.

A trajectory prefix tree that corresponds to the dataset in Table 2.3 is shown in Fig. 2.5.

Prefix Tree Mechanism

The Prefix Tree Mechanism is a foundational approach for the differentially private publication of sequential trajectory data. It was first introduced by Chen et al. [27] to effectively model user trajec-

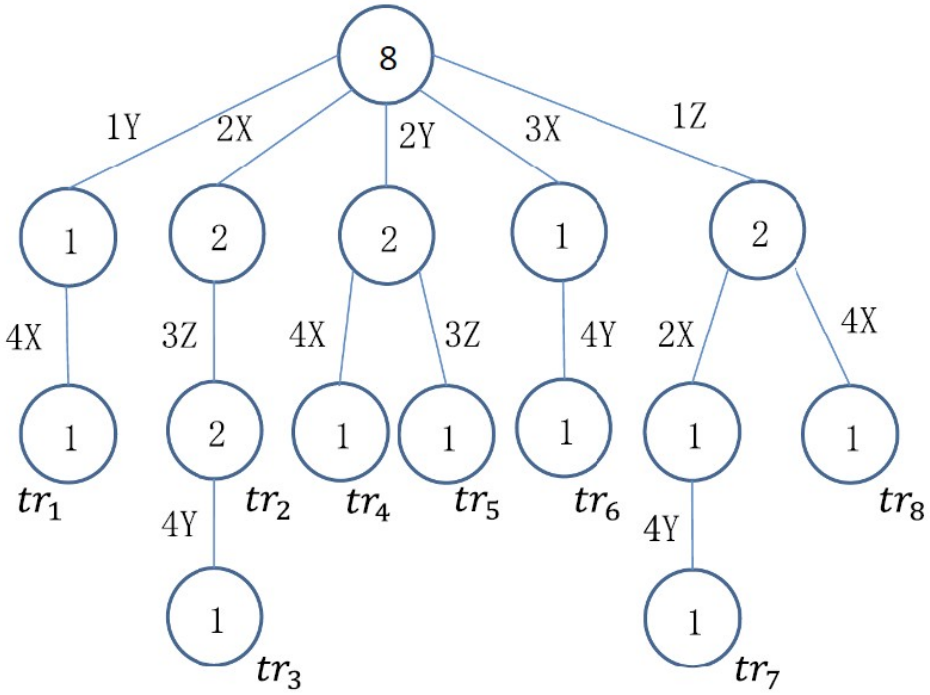

Figure 2.5: The prefix tree constructed from the trajectory dataset in Table 2.3.

tories without relying on hierarchical location generalizations. Instead of using spatial taxonomies, this method directly builds a prefix tree from the raw spatio-temporal data.

A prefix tree is incrementally constructed where each node represents a prefix of a trajectory—typically a location or a timestamp-location pair. Each path from the root to a leaf corresponds to a complete user trajectory, and each node maintains a count of how many trajectories pass through that prefix.

Key Advantages Unlike many location generalization techniques that rely on coarse hierarchical groupings (e.g., grids or regions), the prefix tree mechanism:

- Works directly on raw trajectory data, avoiding information loss from generalization,
- Allocates the privacy budget more efficiently by only adding noise where necessary,
- Allows for deeper and more expressive tree structures due to its adaptive construction,
- Supports high-utility publishing even when datasets lack tap-out or complete location hierarchies.

Mechanism Overview As shown in Algorithm 1, the construction of the prefix tree involves recursive expansion and noise addition.
The main parameters of the Prefix Tree Mechanism are as follows:

- **Privacy budget (ϵ):** Controls the amount of Laplace noise added. A smaller ϵ provides stronger privacy but reduces accuracy. The budget can be split across the tree levels as $\bar{\epsilon} = \epsilon/h$.
- **Threshold (θ):** Determines whether a node is kept or pruned. Nodes with noisy count $\geq \theta$ are expanded, otherwise discarded.
- **Maximum tree height (h):** Defines the maximum depth of the prefix tree, usually chosen according to the maximum trajectory length.
- **Per-level budget allocation ($\bar{\epsilon}$):** Since the tree is built level by level, the global budget ϵ is typically distributed across h levels.

The construction of the prefix tree consists of the following key steps:

1. **Initialization:** Start from a root node that represents all trajectories in the dataset.
2. **Recursive Expansion:** At each level of the tree, examine all possible child nodes that could extend the current prefix.
3. **Noise Injection:** For each node, add Laplace noise Lap$(1/\epsilon)$ to the true count of trajectories reaching that node, where ϵ is the privacy parameter (or a portion of the total privacy budget).
4. **Threshold Check:** Compare the noisy count of each candidate node against a predetermined threshold θ:
    - If the noisy count is above θ, the node is kept and expanded further.
    - Otherwise, the node is pruned and not included in the final tree.
5. **Coverage Guarantee:** To ensure that all points, including those from short or rare trajectories, are considered, the algorithm maintains a full search space during expansion. This helps prevent privacy leakage through omission.
6. **Post-Processing (Constrained Inference):** After constructing the tree, constrained inference techniques are applied to fix inconsistencies caused by added noise. In particular, the algorithm ensures that:

$$\text{count}(parent) \geq \sum \text{count}(children)$$

This preserves logical consistency and enhances utility by correcting anomalies introduced by the noise.

Algorithm 1: Trajectory Data Sanitization Algorithm

Input: Raw trajectory dataset $\mathcal{D}$
Input: Privacy budget ϵ
Input: Height of the prefix tree h
Output: Sanitized dataset $\widetilde{\mathcal{D}}$

$\mathcal{PT} \leftarrow \texttt{BuildNoisyPrefixTree}(\mathcal{D}, \epsilon, h)$;
$\widetilde{\mathcal{D}} \leftarrow \texttt{GeneratePrivateRelease}(\mathcal{PT})$;
return $\widetilde{\mathcal{D}}$;

These steps are formalized and summarized in Algorithm 2, which outlines the **BuildNoisyPrefixTree** procedure is used to construct the differentially private prefix tree.

Algorithm 2: BuildNoisyPrefixTree Procedure

Input: Raw trajectory dataset $\mathcal{D}$
Input: Privacy budget ϵ
Input: Height of the prefix tree h
Output: Noisy prefix tree $\mathcal{PT}$

$i \leftarrow 0$;
Create an empty prefix tree $\mathcal{PT}$;
Insert a virtual root $Root(\mathcal{PT})$ to $\mathcal{PT}$;
Add all trajectories in $\mathcal{D}$ to $tr(Root(\mathcal{PT}))$;
$\bar{\epsilon} \leftarrow \frac{\epsilon}{h}$;
while $i < h$ **do**
    foreach *node* $v \in level(i, \mathcal{PT})$ **do**
        Generate a candidate set of nodes $\mathcal{U}$ from $\mathcal{L}$;
        foreach $u \in \mathcal{U}$ **do**
            Consider u as v's child;
            Add trajectories D in $tr(v)$ such that prefix$(u, \mathcal{PT}) \preceq D$ to $tr(u)$;
            $c(u) \leftarrow$ NoisyCount$(|tr(u)|, \bar{\epsilon})$;
            if $c(u) \geq \theta$ **then**
                Add u to $\mathcal{PT}$ as v's child;
    $i \leftarrow i + 1$;
return $\mathcal{PT}$;

The procedure NoisyCount(x, ϵ) adds Laplace noise with scale $1/\epsilon$ to a true count x, consistent with the standard Laplace mechanism.

Summary The Prefix Tree Mechanism provides:

- Fine-grained modeling of spatio-temporal sequences,
- Controlled expansion of the tree using a noisy threshold,
- Efficient and targeted use of the privacy budget,
- Improved accuracy through post-processing,
- Applicability to non-hierarchical data such as smart card tap-in records.

These properties make the Prefix Tree Mechanism one of the most widely used and effective tools for differentially private trajectory data publication, particularly in transportation systems with incomplete or non-uniform data such as transit logs [27, 28].

Theorem 2.6.1. *Given $\epsilon > 0$, and any threshold θ and maximum height h, the Prefix Tree Mechanism [27] satisfies ϵ-differential privacy under the user-level granularity.*

Chapter 3

Literature Review

Introduction

In this chapter, we review the background of the research. As previously stated, the issue addressed in this thesis is the differentially private publication of bus passenger trajectory data. The goal is to publish real passenger trip data, obtained as tap card records from the public transportation system of the city of Victoria, in a way that is both privacy-preserving and analytically useful.

Since none of the previous studies have specifically addressed real-world data related to the bus system with a tap-in only structure and most of the earlier research has either focused on general user movement data (such as GPS or LBS data in mobile applications) or has concentrated on metro data or general trajectories, a significant gap exists in the literature. Many of the proposed methods were also designed without considering the specific and challenging structure of real bus data. As a result, directly applying these methods to such data often leads to conceptual or performance-related issues. Even in comprehensive studies such as the *SoK: Differentially Private Publication of Trajectory Data* paper [3], which reviews various privacy-preserving methods for trajectory data, the focus is mainly on theoretical evaluation of algorithms, rather than their practical application to real bus data with a tap-in structure. Experimental comparisons of methods based on real-world public transportation data at the urban scale—especially while preserving the precise location of stations—have received little attention.

Therefore, there is a need for revisiting, adapting, and localizing existing algorithms to match the characteristics of real-world bus data so that, on the one hand, theoretical privacy guarantees can be maintained, and on the other hand, the practical applicability and analytical value of the final published data can be ensured.

Accordingly, in the following, we review the related works connected to the topic of this thesis, which primarily fall under the following areas:

- Differential privacy in trajectory databases, including the foundational works and early attempts to define privacy guarantees in sequential data;
- Privacy-preserving trajectory data publishing algorithms, with a focus on prefix-tree based methods and their variants.
- Synthetic data generation methods based on location and time.
- Practical models and case studies for public transportation systems, highlighting the gap in the literature for tap-in only datasets and motivating the need for adapting existing algorithms to this context.

Overview of Related Work

Differential privacy, since its introduction in 2006 [30], has been applied to a wide range of data analysis tasks and applications [34–46]. In this section, we review the main and recent works and ideas in this field that have been proposed specifically for trajectory databases. [27, 33, 47–61].

The problem of publishing data from trajectory databases can be discussed in two different forms. In the first type, each trajectory is considered as a database, and therefore each point of the trajectory constitutes a data record. In the second type, each trajectory is considered as a single data record. In the domain of privacy, the first type has not received much attention, and limited work has been done on this form. In contrast, many studies have been conducted on the second type. For example, Shao et al. [50] proposed two techniques for publishing data from trajectory databases under differential privacy. These techniques were specifically designed for the publication of ship trajectories under differential privacy, where each ship's trajectory is treated as a database. These techniques use a combination of sampling and interpolation. The first technique performs sampling followed by interpolation, while the second applies interpolation followed by sampling. In the sampling phase, a fixed number of points are sampled from the trajectory and the remaining points are discarded. In the interpolation phase, the discarded points are recovered using interpolation techniques. They showed that both techniques guarantee $(0, \delta)$-DP.

For publishing trajectory data of the first type, Jiang et al. [51] demonstrated, a simple solution is to add noise to the trajectory. This can be done in three ways: in the first approach, the entire trajectory is treated as a single point in a high-dimensional space, and differential privacy is ensured by adding a high-dimensional noise vector to the actual trajectory. In the second approach, a two-dimensional noise vector is added to each point in the trajectory. In the third approach, noise is added to each coordinate of the trajectory points.

However, our focus is mainly on the second type of trajectory data, and on achieving $(\epsilon, 0)$-DP (which is generally preferred over $(0, \delta)$-DP as considered in [50]). In recent years, several mechanisms have been proposed to apply differential privacy to trajectory databases of the second type. Chen et al. [27] were the first to study the differential privacy problem for publishing trajectory data. They proposed a non-interactive, data-independent differential privacy algorithm that constructs a noisy prefix tree over the trajectory database. In this approach, trajectory records with the same prefix are grouped into the same branch. Each node in the tree stores a number representing the count of trajectory records whose prefix matches the path from the root to that node. Their proposed method is one of the most relevant techniques for bus transportation data and has served as a basis for many other methods. Given its compatibility with the needs of bus data, this paper was selected for further analysis. We implemented and evaluated this paper in Chapters 5 and 6 of this thesis, with our own improvements in the post-processing phase.

Not all DP sequential data publication algorithms are compatible with our bus dataset. For example, the variable-length n-gram algorithm introduced by Chen et al. [33] was designed to publish sequential data under differential privacy. This method models the data as a sequence of events and extracts a set of variable-length n-grams from it. A Markov assumption is applied in modeling these sequences, meaning that the probability of the next item depends only on the previous $n-1$ items. Then, an exploration tree is constructed based on these sequences, where the frequency of each n-gram in the original data is counted and Laplace noise is added to ensure privacy protection. Finally, private synthetic sequential data is generated from this noisy tree.

Despite the novelty of this algorithm in combining the n-gram structure with differential privacy, a closer examination reveals that it is not well-suited for public transit data—especially tap-in-only bus data. The main reasons for this incompatibility are outlined below:

- **Incompatibility with the structure of tap-in data:** The n-gram algorithm relies on the assumption of regular and continuous sequences, where each event is directly influenced by preceding ones. However, this assumption does not hold for tap-in data, as passenger trips may occur sporadically, across varying time and space intervals, and may even include repeated entries. For example, a passenger may tap in at stop 23 today and tap in again at stop 95 several days later, without any clear temporal or spatial connection. In such cases, the actual route is unknown and highly inconsistent with the regular patterns assumed by the n-gram model.

- **Inability to realistically reconstruct trajectories:** The goal of data publishing algorithms is not only to protect privacy but also to generate data that remains useful and realistic for analysis. However, the n-gram algorithm bases predictions only on prefix sequences (i.e.,

the previous $n-1$ steps) and does not account for temporal or spatial structures. This leads to generated paths in public transit data—where routes are defined by bus lines and stops—that lack coherence and logic. In other words, the algorithm cannot reconstruct semantically meaningful or operationally consistent routes aligned with the real transportation network.

- **Scalability and memory challenges:** Constructing an exploration tree over all possible n-grams causes the tree size to grow exponentially with the number of locations (e.g., bus stops). In real-world datasets like the Victoria tap-in dataset, which contains thousands of stops and millions of records, this results in excessive memory usage and significantly reduced performance. The paper by Chen et al. [33] also explicitly states that their method suffers from low accuracy and poor efficiency when applied to large and sparse datasets.

Due to the above suitability issues, this algorithm was not used in this thesis.

Ruan and Ho [53] proposed a differential privacy approach for mining geographic location patterns. They focused on two types of spatial queries, namely the *count of points within a region* and the *center of a region*. Due to the high global sensitivity of these queries, and in order to mitigate this issue, they employed a *local sensitivity* dependent on the dataset instead of relying on global sensitivity. Moreover, they utilized a quadtree structure along with a clustering algorithm. After partitioning the space using the quadtree, the clustering algorithm was applied within each partition. Then, the number of points in each cluster was counted, and if the count exceeded a predefined threshold, the cluster was considered interesting.

Li et al. [62] proposed a trajectory data publishing algorithm under differential privacy that combines a *bounded noise generation algorithm* with a *trajectory merging algorithm*. The bounded noise generation algorithm is designed such that the noise added to the true trajectory values for privacy protection is sampled from a restricted (reasonable) range. In other words, the added noise is constrained within a specific interval to ensure both privacy preservation and data utility.

A number of studies build sequence or exploration trees and only perturb the counts of subsequences that actually appear in the data, leaving hypothetical (zero-count) trajectories completely outside the output space. Representative works in this category include Zhao [63], Yuan et al. [64, 65], and the "trajectory count" approach [66]. The main issue is that the output is defined only on the observed trajectories, while no probability mass is assigned to zero-count trajectories. This contradicts the formal definition of DP, since under DP the output distribution must be well-defined and comparable for all neighboring datasets, including those that differ by the presence of a previously unseen trajectory. When the output does not account for zero-count cases, an adversary may directly infer the existence or absence of a particular trajectory.

The works of Hua et al. [67], the extended versions such as Chen et al. [68], and the bounded-noise method of Li et al. [62] fall under the category of clustering with the exponential mechanism

and a data-dependent output space. The key problem here is that in the exponential mechanism the output set must be predefined and independent of the data. However, in these works the output space is constructed directly from the data (data-dependent). This dependency allows the very choice of clusters to leak information about the raw data, rendering the formal DP proof invalid. Therefore, these approaches also lack DP guarantees.

In the SafePath algorithm [47], after applying noisy counts to nodes, those with counts below a given threshold (θ) are pruned, and only the surviving nodes are retained as generalized nodes. Instead of outputting the exact number of passengers at, for example, "station 152," the data is released in an aggregated form such as "the number of passengers across stations 150 to 160." In our bus data, however, we require synthetic data that explicitly indicates at which exact stop passengers tapped in. If the algorithm groups stops, it becomes impossible to identify high-traffic stations or to perform fine-grained analyses of travel patterns. For instance, if busy stops are merged with less busy ones, the resulting analysis for transportation planning would be severely misleading. While such generalization may be acceptable for broader applications (e.g., regional mobility analysis or GPS-level data), this approach loses its utility when applied to tap-in bus card data, where station-level preservation is crucial.

In contrast, Li et al. (2020) [28] is a state-of-the-art mechanism that is compatible with our dataset and restrictions. It also builds on the prefix tree idea, but focuses on spatial-temporal data (combinations of station and time). To address the dimensionality explosion problem in such data, the paper introduces three key innovations:

1. A new prefix tree without using a taxonomy tree: instead of relying on data-dependent clustering, the tree structure is predefined. This avoids the generalization problem.

2. An incremental privacy budget allocation model: the DP budget is distributed gradually and proportionally to tree depth, with higher levels receiving less noise and deeper, more detailed levels receiving more.

3. Spatial-temporal reduction: only unreachable or nonsensical nodes are pruned, without making the output space data-dependent.

As a result, the final output is defined over a data-independent space covering all possible station–time combinations, with privacy budget allocation and noise ensuring privacy. In this approach, the exact station identifiers are preserved in the output, and they are never replaced with geographic points outside the station set or with generalized groups. This feature is particularly valuable for bus transportation data, where one of the main challenges in selecting an algorithm is the need to preserve station-level granularity. For these reasons, this paper was retained as one of the core methods to be implemented and evaluated in this thesis.

The work of Wen et al. (2020) [69] introduces a Personalized DP mechanism in which each trajectory point is assigned a different privacy level (e.g., stay-points are considered more sensitive). While the core idea is valuable, the method is not suitable for our dataset, which requires the preservation of exact station IDs. The algorithm perturbs locations by replacing them with "nearby" alternatives. Although this may be reasonable for GPS data, it is inconsistent with bus card data where the exact stop must be preserved. Consequently, the utility at the station level is lost.

Wang et al. (2020) [70] proposes a method for publishing trajectory data under differential privacy with a focus on protecting sensitive places (e.g., hospitals or private venues). The core idea is highly valuable, since it combines privacy preservation with consideration of location sensitivity, which is important for future research directions, including ours. However, the main limitation of this method is generalization: - To protect sensitive places, stops or locations are aggregated into larger clusters or cells. - As a result, the exact station identifier or precise location is lost in the published data, with only a generalized area (such as a range of stops) being reported. - This is problematic for our tap-in only dataset, since the main goal of our project is to preserve station-level granularity. Thus, given this algorithm's reliance on generalization and the resulting loss of station-level detail, it is unsuitable for direct application to our dataset. For this reason, it is classified in the literature review as not applicable to our project, although its concept will be revisited in the conclusion chapter as an important avenue for future research.

Chapter 4

BC Transit Data Collection and Generation

Introduction

In this chapter, we describe the collaboration with BC Transit, in particular the tap dataset, as well as steps in generating a plausible synthetic trajectory dataset in order to showcase more complex privacy-preserving queries.

4.1 Data Collection and BC Transit Collaboration

This thesis focuses on the Victoria region in British Columbia, in particular, analyzing real data from the city's public transit system. Although various public datasets are available through BC Transit, no data related to individual boarding records was accessible, as such datasets have not been released publicly.

During the data collection phase, multiple governmental sources and open data platforms were examined, including BC Data Catalogue, Passenger Transportation Board, BC Geographic Warehouse, TransLink, as well as Data Systems & Services Client Hub, Data Governance, and Open Data Portals. Numerous requests were submitted through emails, online forms, and service portals. However, most organizations either did not possess the required data or were unable to share it due to privacy, jurisdictional, or capacity constraints.

Eventually, a formal collaboration was established with **BC Transit**. Initially, concerns were raised even about releasing anonymized data. However, after providing detailed explanations about the academic purpose of the thesis, the differential privacy framework, and the ethical safeguards in place, BC Transit responded positively.

Through a multi-step process—including briefing sessions, completion of formal data request forms, and responding to ethical and technical inquiries—BC Transit provided a de-identified dataset of tap-in events.

This collaboration marked BC Transit's first experience supporting an academic thesis focused on privacy. Their growing interest in understanding the application of differential privacy to passenger data highlights the practical and research value of this work. Their support played a key role in shaping a realistic, locally grounded approach to the privacy-preserving generation of sequential trajectory data. One of the goals of our work is to demonstrate that privacy-preserving analyses of various kinds of transit data are possible and practical. Thus, in addition to analyzing the tap dataset, we also motivate the privacy-preserving analysis of sequential trajectory data (which we were not able to obtain due to privacy concerns). To do so, we first describe a *new* host of queries possible when given sequential trajectories. Then, we demonstrate the efficacy of DP mechanisms on such queries in Ch. 6. Instead of running these mechanisms on random and thus less realistic datasets (as is done in some previous work [27]), we take this demonstration a step further, by creating a plausible synthetic trajectory dataset using the real tap dataset. We describe the generation of this synthetic dataset in Section 4.2.
Further details regarding the data request process and communications with BC Transit are provided in Appendix A.

Understanding the Tap-In Dataset and Its Analytical Potential

The tap-in dataset provided by BC Transit consists of de-identified records of boarding events recorded by onboard sensors. Each record includes the following attributes:

- **routeName**: The route number of the bus
- **stopId**: The unique identifier of the stop where the passenger tapped in
- **ts**: Timestamp of the tap-in event
- **vehicle**: Internal identifier of the vehicle
- **bus_number**: Bus fleet number
- **seats**: Capacity of the vehicle

These fields enable a variety of useful transit analyses. For example, this dataset supports:

- Estimating the number of tap-ins per stop or per route
- Identifying peak hours and temporal usage trends
- Measuring route popularity or stop-level usage patterns

- Aggregating demand by bus or by day to estimate vehicle utilization

This level of detail allows researchers and transit planners to better understand demand patterns, optimize routes, and improve operational efficiency.
Beyond the direct insights available from the tap-in data alone, its value increases significantly when combined with other publicly available data from BC Transit's website (like schedule data). This integration opens the door to rich, policy-relevant questions in operational planning and rider experience. For example:

- **Delay Analysis**: By comparing actual tap-in timestamps with scheduled arrival times for each stop, one can identify buses that arrived late or early. If many passengers tapped in shortly after a delayed bus arrived, it may indicate an accumulated wait, signaling a high impact of delay at that stop [71].
- **Load vs. Capacity Trends**: Combining tap-in counts with the `seats` field allows estimation of bus occupancy levels over time. When correlated with scheduled frequencies, this supports capacity planning or fleet adjustments during peak hours.
- **Missed Transfers and Service Gaps**: Although individual passenger journeys cannot be tracked, clusters of tap-ins at adjacent stops and times can indicate systemic transfer patterns. When such clusters appear disrupted, it may point to service unreliability or coordination issues across routes.
- **Event or Weather Impact**: Tap volume fluctuations across routes on specific dates or times (e.g., during storms or public events) offer insights into demand elasticity and resilience planning.
- **Service Equity and Accessibility**: Spatial patterns in tap-in activity across neighborhoods, when combined with demographics or land-use maps, support evaluations of equitable service distribution.

These types of analyses are highly valuable for transit authorities. For example, delay analysis alone is a critically important domain [72]—helping improve rider satisfaction, reduce missed connections, and guide resource allocation. Even in the absence of full trajectories or tap-out data, the structural richness of the tap-in dataset makes it a powerful foundation for data-driven decision-making in transit operations [72, 73].
However, as discussed in the following section, some questions—especially those involving complete passenger paths, behavioral clustering, or role-based movement modeling—require the generation of plausible synthetic trajectory data under formal privacy protections.

Broader Impact and Motivations

BC Transit's involvement has the potential to lead to productive conversations around the following opportunities:

- **Improving Urban Planning:** Accurate yet private data enables better identification of underserved routes and high-demand locations. Making urban planning decisions based on publicly-available (privatized) data also creates more trust between the government and the public.
- **Fostering Innovation:** Synthetic data can fuel transit-related software development, accessibility tools, and simulation systems.
- **Reducing Public Research Costs:** Open access to synthetic datasets allows universities and private labs to perform transit research more efficiently and at lower cost.
- **Advancing Equity:** Fairness analysis becomes possible when data is safely shareable.
- **Leadership in Ethical Data Publishing:** BC's adoption of differential privacy could position it as a national or global model.

4.2 Synthetic Generation of Sequential Trajectories

The tap dataset allows us to answer already-useful queries, such as ... (will state a couple queries used for tap dataset). However, with access to even more informative datasets (such as sequential tap data), we can understand the state of BC Transit even better. To demonstrate this, in this section we, in collaboration with BC Transit, generate a synthetic dataset of trajectories (sequential location data that follows an individual's trajectory within the bus system) that aligns with the original real tap dataset, and known passenger behaviours in BC Transit. This dataset will be used in Ch. 5 to showcase the potential power of DP mechanisms on trajectory data for BC Transit.

Objective and Problem Definition

This section presents a synthetic trajectory generation method based on real tap-in data from BC Transit in Victoria, BC. The objective is to produce plausible passenger movement sequences that are produced using typical human mobility behaviors while preserving privacy. The method simulates plausible travel patterns for different user roles (e.g., full-time employee, part-time employee, self-employed, student, unemployed) and purposes (e.g., work, study, shopping, travel, personal activity), despite the limited data available.

Limitations of Tap Dataset

- Only tap-in data is available. Tap-out data (where and when a passenger gets off) is not recorded in this type of bus card system. This is a structural feature of the system and not a data quality issue. Only the boarding moment (tap-in) is logged.
- There is no user ID. Passengers appear completely anonymous in the raw data, and no identifier is available to track an individual's trips over time.
- Each tap-in record includes timestamp, stop ID, route name, bus number, stop coordinates (latitude and longitude), and seats (indicating bus type and capacity).

Challenges and Requirements

Given these limitations, our algorithm will:

- Group tap-in records into synthetic passenger trajectories (without having real user IDs).
- Align generated temporal and spatial patterns with realistic human behavior (e.g., commute patterns to work or school, irregular travel for seniors or tourists, etc.).
- Avoid unrealistic artifacts such as:
  - Too many tap-ins at the same stop in one day.
  - Very fast or physically impossible trips in terms of time and distance.
  - Unreasonably filling daily travel limits in a very short time.
  - Travel patterns that do not match natural human routines (like trips without return or trips without any time gaps).

This process results in synthetic trajectories that both protect user privacy and can be used for urban transport analysis, evaluating new policies, and designing privacy-preserving data-sharing approaches.

Research Sources

To ensure higher realism, various reports and sources were used, and our assumptions were reviewed and adjusted with suggestions from BC Transit employees. Most behavioral parameters and numerical metrics come from the *Customer Satisfaction Tracking Research Annual Report 2024–2025* [74] by BC Transit. Additionally, other sources like Statistics Canada [75] were used to define work hours and employment patterns (such as definitions of full-time and part-time work), so the distribution of roles and travel patterns reflect real conditions as much as possible.

Passenger Behavior Modeling Assumptions

Using the above sources and with feedback from our source at BC Transit (Andrew Miller, Manager, Enterprise Data & Analytics), we make the following assumptions about a bus passenger: a passenger can use the bus system multiple times a day, which usually includes combinations of:

- Regular daily trips (e.g., going to work or school in the morning and returning in the evening).
- Combined trips (e.g., dropping a child at school in the morning and then going to work).
- Other occasional trips during the day (e.g., shopping, medical visits, social meetings).

The behavioral assumptions described above were developed in consultation with BC Transit representatives (notably Andrew), whose experience and familiarity with passenger dynamics helped ensure the plausibility and relevance of the modeled patterns.
Repeating travel patterns are expected. The algorithm models this repetition and reflects it in synthetic trajectories:

- Daily trips between home and work/school.
- Semi-regular trips (e.g., weekly shopping or sports activities).
- Irregular or leisure trips.

Main Algorithm Assumptions

1. **Passengers have a fixed identity.** Each synthetic passenger has a fixed social role (e.g., full-time worker, student, unemployed) that does not change during the simulation.
2. **Passenger role distribution (`ROLE_DISTRIBUTION`).** When a new passenger is created, a social role is assigned that remains fixed for all simulation days. These roles and their percentages are based on BC Transit reports and Statistics Canada.

   Exact distribution:

   - 20% full-time employees
   - 8% part-time employees
   - 6% students
   - 17% business owners or self-employed
   - 49% unemployed

Total: 100%. The selection is done using `np.random.choice` with the specified weights.

3. **Behavior depends on role.** Passengers with different roles follow different weekly and daily travel patterns. These include the number of travel days per week, trip counts, start times, stay durations, and return times.

4. **Weekly active days (`WEEKLY_ACTIVE_DAYS_PER_ROLE`).** This parameter controls how many days per week each passenger role is active.

   From the code:

   - Full-time employee: exactly 5 days per week (Monday–Friday), fixed
   - Part-time employee: 2 to 4 days, random
   - Student: 3 to 5 days, random
   - Business owner: 4 to 6 days, random
   - Unemployed: 2 to 5 days, random

   Non-full-time roles choose days randomly from the whole week; full-time employees are limited to weekdays.

5. **Travel purpose distribution (`GENERAL_NON_PRIMARY_PURPOSE_DISTRIBUTION`).** Besides their primary purpose (e.g., work or school), passengers may also have general trips.

   General purpose breakdown (45% of total):

   - Shopping: $17/45 \approx 0.378$
   - Social: $16/45 \approx 0.356$
   - Airport: $12/45 \approx 0.267$

   Logic in the `infer_trip_purpose` function:

   - *Primary purpose priority:* On weekdays and during working hours:
     - Full-time employee: 6 AM–12 PM
     - Part-time employee: 5 AM–6 PM
     - Student: 6 AM–4 PM

     If the current tap matches the pattern, the purpose is set to work or school with high probability.

- *Sequential or return trips:* If the tap is close in time and distance to the last one, the same purpose is inherited.
- *General or personal purposes:* Otherwise, there is a 55% chance of assigning a purpose related to the passenger's role, and a 45% chance of selecting from the general purpose distribution above.

Logical Temporal Extension

The algorithm is designed to consider realistic time and distance constraints when generating trips.

Time and distance rules per role (extttROLE_BASED_TIME_RULES and distance constraints)

- **Assumption:** The time and distance between taps show the activity type and purpose, and these patterns depend on social role.
- *Details:*
  - Time gaps for consecutive taps: `min/max_immediate_consecutive_tap_minutes` define the min and max time between taps to treat them as one trip or transfer.
  - Activity duration: `min/max_primary_activity_duration_hours` determine how long a passenger stays at their destination.
  - Minimum gap for new trip: `min_new_distinct_trip_minutes` is the minimum time between two taps to treat them as separate trips.
  - Latest daily activity end: `max_daily_activity_end_hour` defines the latest time a passenger may tap in.
  - Distance constraints (in km):
    - `MAX_DISTANCE_KM_FOR_IMMEDIATE_CONSECUTIVE_TAP`
    - `MAX_DISTANCE_KM_FOR_PRIMARY_RETURN_TRIP`
    - `MAX_DISTANCE_KM_FOR_NEW_TRIP_START`

Avoiding Unrealistic Patterns

To ensure the realism of synthetic trajectories, the following controls are implemented:

- **Time gaps between taps:** Prevent unrealistic tap sequences.
- **Exceptions allowed with lower probability:** Rare behaviors like sudden midday return are allowed but uncommon.

- **Limit daily taps:** Controlled via `DAILY_TAP_LIMITS_PER_PURPOSE_AND_ROLE` to ensure average of 4.5 taps/week.
- **Behavioral diversity:** Some passengers have no taps or varying tap counts per day.

For more detail regarding the generation of the synthetic data, please refer to Appendix B.

4.3 Selection of Methodology: Justification and Comparison

Current Method: Rule-Based + Scoring Function

- Developed and tested iteratively
- Interpretable and configurable
- Works well with tap-in-only data
- Computationally light

Limitations:

- Incomplete trip information
- No network-wide learning
- Empirical tuning

Comparison with Other Methods

Markov Models: Require full trajectories; not suitable due to lack of tap-out.

Clustering: Too restrictive; travel patterns not always clustered.

Graph-Based: Require complete sequences to build graphs.

Learning-Based: Require massive labeled datasets and complex training.

پ

Other Approaches and Future Directions

Recent advances in generative modeling, including GANs, VAEs, and Transformer-based architectures, have shown promise in synthesizing sequential data in domains such as healthcare and natural language processing [26]. These models can, in principle, capture long-range dependencies and generate highly realistic trajectories. However, they often require large training datasets, substantial computational resources, and rely on differentially private training procedures (e.g., DP-SGD) that can significantly degrade model utility and are challenging to explain to non-technical stakeholders.

In this study, we deliberately focused on prefix-tree mechanisms, which provide clear and auditable differential privacy guarantees, are lightweight to implement at scale, and produce interpretable outputs that transit agencies can readily validate. For applied settings such as BC Transit, where organizational trust, reproducibility, and transparency are as important as accuracy, prefix-tree approaches offer a defensible and practical first step. Nevertheless, deep generative models remain a promising avenue for future research, and hybrid approaches combining the interpretability of tree structures with the expressive power of neural models could further improve utility while preserving privacy.

Final Conclusion

Given the nature of the data and the practical goals of the thesis, a rule-based and scoring-function approach is currently the most suitable, justifiable, and defensible methodology for generating synthetic trajectories from tap-in-only transit data.

Chapter 5

Experimental Methodologies and Technical Contributions

5.1 Introduction

This chapter presents the experimental methodology used to implement and evaluate two algorithms proposed by Chen et al. [27] and Li et al. [28] for differentially private trajectory data publishing, evaluated using real-world smart card transit data from Victoria, BC.

While the theoretical foundations and general principles of prefix tree mechanisms were introduced in Chapter 2, here we specifically focus on:

- Adapting the algorithms for the prefix tree mechanisms to the BC Transit tap dataset (particularly, the trajectory data generated in Chapter 4).
- Improvements in implementation details and parameter configurations.
- Refinements in the post-processing of privatized data to enhance consistency and utility.

Both algorithms employ the *Noisy Prefix Tree* mechanism, but differ in internal design and privacy budget allocation. In particular, Chen et al. (NPT) model trajectories as sequences of locations, while Li et al. (PPDP) extend the tree to encode (location, timestamp) pairs, thereby capturing both spatial and temporal information. In our adaptation, modifications were introduced, including improved post-processing for the Chen et al. [27] algorithm and replacing the θ-based pruning with a *Binomial Test* for the Li et al. [28] algorithm.

The adaptation process includes multiple preprocessing steps, data cleaning, and converting raw tap-in records into trip sequences. These preprocessing steps, common to both DP mechanisms' algorithms, are detailed below before discussing each algorithm individually.

Data Preparation and Cleaning

The initial dataset consisted of **312,659** passenger tap-in events between December 15–20, 2024. Preliminary inspection revealed missing or invalid values:

- **1,833** records with `routeName` equal to "Default" and empty `stopId`.
- **1,010** records also missing `vehicle`, `bus_number`, and `seats`.

To ensure high-quality input, the following cleaning procedure was applied:

1. Remove records missing essential operational fields (`vehicle`, `bus_number`, `seats`).
2. Identify incomplete records (`routeName` missing, `stopId` missing, or `routeName` = "Default").
3. Use complete records as a reference set to fill missing values by matching on `bus_number`.
4. Replace missing `routeName` and `stopId` using corresponding values from matched records.
5. Remove records still containing invalid `routeName` or `stopId`.
6. Reset dataset index after cleaning.

Generation of Sequential Trajectories

In this stage, the goal is to prepare the cleaned dataset for use in the **Noisy Prefix Tree** algorithm. The details of how trajectory data is generated, are found in Ch. 4. We summarize the steps below.

As explained in Ch. 4, most bus transit systems worldwide are designed to record only **tap-in** events (boarding), without storing information about the time or location of passengers alighting.

Furthermore, in the dataset released by **BC Transit**, passenger *User IDs* have been completely removed to ensure security and protect passenger privacy. As a result, all passengers are recorded anonymously, and no information is available to directly trace their travel paths. Consequently, direct reconstruction of complete passenger trajectories is not possible, and synthetic passenger trajectory generation methods must be employed to obtain realistic tap-card travel sequences.

The trajectory generation process in this study is designed to:

1. Preserve temporal and spatial travel patterns as close as possible to real passenger behavior,
2. Incorporate attributes related to passengers' social roles (e.g., student, worker, retiree) and trip purposes (e.g., commuting, shopping, personal business),

3. Respect operational constraints of the transit system and maintain reasonable time gaps between trips.

To **ensure realism and accuracy of the simulation**, numerical indicators and passenger behavioral information were extracted from the **BC Transit – Customer Satisfaction Tracking Research Annual Report 2024–2025** [74] and supplemented with insights obtained from direct inquiries to BC Transit. This ensured that modeling was grounded in actual operational conditions and real-world travel patterns.

Finally, the records for each passenger in a given day were chronologically ordered and converted into a **travel sequence**, preserving the order of stops and routes. If a sequence exceeded the maximum length $L_{\max}$ (determined based on the statistical distribution of actual trip lengths), it was truncated to comply with algorithmic requirements, as described in Chapter 4, section 4.2

5.2 Framework of the Proposed Algorithms

Bus tap card data presents unique challenges that are not observed in other types of transportation datasets (such as GPS-based data in the Geolife dataset, taxi datasets, or maritime and other transportation data). Unlike many existing algorithms that can generalize locations to some extent, in bus tap card data such generalization is not acceptable. In other words, each stop ID must be preserved exactly, and merging or spatial shifting of stops is not permitted.

This unique characteristic rendered many algorithms and studies—originally designed for other types of transportation data—unsuitable for this thesis. Consequently, we selected the NPT (Noisy Prefix Tree, Rui Chen et al.) [27] and PPDP (Yang Li et al.) [28] algorithms, which our comprehensive review identified as more practical and compatible with bus tap card data.

Motivation for testing NPT and PPDP algorithms

- **NPT as Benchmark and as a Baseline Algorithm (Conceptual Simplicity, Computational Efficiency, and Extensibility):** The NPT algorithm represents the first major work in the domain of privacy-preserving trajectory publishing. Many subsequent studies have either built upon it or compared their results against it. Therefore, including NPT as a benchmark makes our results both comparable and credible in the research literature.

  Importantly, NPT does not incorporate time into the prefix tree structure; it only considers the sequence of locations (stops). As a result, computations are simpler ($O(n \cdot h)$), focusing solely on sequence length. If time (e.g., time-of-day or intervals) were also introduced, each location would need to be expanded into multiple time-stamped states $\rightarrow$ leading to a significantly larger tree, higher computational complexity, and more noise.

- **PPDP:** The PPDP algorithm explicitly incorporates time from the start, building a spatio-temporal prefix tree. Instead of just modeling "from stop A to stop B," it models "from stop A at time t to stop B at time $t + \Delta$." While this increases complexity, it provides higher accuracy by preserving travel-time patterns (such as peak and off-peak hours).

- **Advantages of NPT over PPDP:**

  - **Simplicity and transparency:** Conceptually and in implementation, NPT is simpler, since it is essentially a prefix tree with added noise. This makes it easier for readers to understand and follow.
  - **Computational efficiency:** The complexity of NPT is lower, as it only focuses on spatial sequences without introducing the temporal dimension.
  - **Reliable baseline:** Due to its popularity and widespread use, NPT remains a standard benchmark in the field.

- **Advantages of PPDP over NPT:**

  - **Incorporation of time:** In public transit data, time plays a critical role. PPDP integrates this dimension, reconstructing data more realistically.
  - **Progress beyond 2011:** Despite higher complexity (particularly due to the spatio-temporal matrix), PPDP demonstrates how accuracy and utility can be better preserved in practical applications.
  - **Optimized budget allocation:** PPDP allocates the privacy budget adaptively across levels, avoiding waste at the deeper levels of the tree.

Conclusion: We test NPT because it is a historical, simpler, and computationally efficient baseline, making results comparable with prior literature. We test PPDP because it is the more advanced model that incorporates time and is therefore more suitable for smart card transit data. Comparing the two allows us to evaluate both efficiency and simplicity (NPT) as well as realism and practical utility (PPDP).

It is also worth noting that in cases where queries are coarse and location-based (and not time-dependent), NPT may still be sufficient, particularly when very high accuracy is not required.

5.2.1 Noisy Prefix Tree (Chen et al., 2011)

Summary

The Noisy Prefix Tree (NPT) proposed by Chen et al. [27] is a benchmark and highly cited DP mechanism for publishing privatized sequential data, such as trajectories, by constructing a prefix

tree that encodes all observed sequences. In this thesis, the algorithm is applied directly to tap-in transit smart card records from Victoria, BC, preserving its original structure but incorporating an enhanced post-processing step to improve the consistency and practical utility of the released data.

Prefix Tree Construction

The execution of the NPT algorithm begins with the creation of a **prefix tree** capable of encoding all travel sequences present in the input dataset. Unlike spatial hierarchy-based methods that first generalize locations into coarser levels (such as zones or cities), this mechanism directly uses the exact spatial points from the raw data and preserves their order and details in the prefix tree structure. This approach enables a more accurate representation of trajectories compared to generalized methods.

The tree construction process consists of the following steps:

1. **Root Node Initialization** An empty node is created as the root, representing the start of all trajectories. This node is considered to be at level zero, and its count is set to the total number of sequences in the dataset.

2. **Tree Depth Configuration** The depth of the tree is limited to $L_{\max}$, where each level corresponds to a specific position in the travel sequence. The value of $L_{\max}$ is determined based on the statistical distribution of trip lengths and operational considerations of the transit system.

3. **Sequence Insertion** For each travel sequence in the input data, the corresponding path from the root node down to the required level is created or updated. Common subsequences among different trajectories are stored only once, which optimizes storage usage and improves tree-building efficiency.

4. **Count Storage** Each node maintains a counter that records the frequency of its associated prefix. These counts are later used for adding Laplace noise and for applying the *pruning* strategy.

This stage serves as the foundation for two critical subsequent steps:

- Allocation of the privacy budget ε across the tree levels (Privacy Budget Allocation),
- Pruning of branches using the threshold parameter θ to prevent unnecessary tree expansion.

Privacy Budget Allocation

To control the amount of noise added and ensure compliance with the total privacy budget ε, this budget is evenly distributed across all levels of the prefix tree. If the tree height is h, the privacy budget allocated to each level is:

$$\bar{\varepsilon} = \frac{\varepsilon}{h}$$

This uniform allocation ensures a balanced level of perturbation at each depth and prevents early exhaustion of the privacy budget.

Noisy Prefix Tree Construction

The construction process proceeds level-by-level until the maximum height h is reached:

1. **Initialization (Level 0)**: The root node represents the start of all trajectories, with its count equal to the total number of sequences in the dataset.

2. **Node Expansion**: For each level i from 0 to $h-1$, each node v is examined and a set of candidate child nodes U is generated. This set contains all possible next locations from the location set:

$$\mathcal{L} = \{l_1, l_2, \ldots, l_n\}.$$

3. **Sequence Assignment**: For each candidate $u \in U$, all sequences whose prefix matches u are assigned to it. The true count for node u is given by:

$$c(u) = |tr(u)|$$

   where $tr(u)$ is the set of sequences ending with prefix u.

4. **Adding Laplace Noise**: To ensure ε-differential privacy, Laplace noise with sensitivity $\Delta f = 1$ and budget $\bar{\varepsilon}$ is added to each node's true count:

$$\tilde{c}(u) = c(u) + \text{Lap}\left(\frac{\Delta f}{\bar{\varepsilon}}\right)$$

   or more concisely:

$$\tilde{c}(u) = \text{NoisyCount}(|tr(u)|, \bar{\varepsilon}).$$

5. **Pruning**: The noisy counts $\tilde{c}(u)$ are then compared to a pruning threshold θ to decide whether a node should be expanded further or removed from the tree.

Pruning Strategy and Threshold θ

After computing the noisy counts for each node, the NPT algorithm decides whether the node should expand further in the tree. This decision is made based on a threshold parameter θ.

1. Threshold Calculation. The threshold is computed as:

$$\theta = \frac{2\sqrt{2}}{\bar{\varepsilon}}, \quad \text{where} \quad \bar{\varepsilon} = \frac{\varepsilon}{h}$$

Here, $\bar{\varepsilon}$ is the privacy budget allocated to each tree level, ε is the total privacy budget, and h is the height of the tree.

2. Non-Empty Node Check. For each node u, if the noisy count $\tilde{c}(u) \geq \theta$, the node is considered significant and is expanded into the next level of the tree. Otherwise, the corresponding branch is pruned.

3. Empty Node Check. To prevent information leakage and maintain differential privacy guarantees, low-frequency paths cannot be entirely removed, as doing so may allow reconstruction of sensitive information. Instead, some unseen or low-frequency paths are intentionally preserved in a controlled way by adding *empty nodes*.

Depending on dataset size, two main approaches are used:

(a) Small Dataset. For smaller datasets, where the number of empty nodes is limited:

1. Identify empty nodes: Nodes with actual count $|tr(u)| = 0$.
2. Add Laplace noise:
$$\tilde{c}(u) = 0 + \text{Lap}\left(\frac{1}{\bar{\varepsilon}}\right)$$
3. Threshold check: If $\tilde{c}(u) \geq \theta$, the node is added to the tree; otherwise, it is pruned.

(b) Large Dataset. For larger datasets, where the potential number of empty nodes is very large, a statistical sampling approach is applied:

1. Compute the total number of candidate empty nodes:

$$m = (|\mathcal{L}| \times \text{number of parents passing } \theta) - \text{number of non-empty nodes}$$

where $|\mathcal{L}|$ is the total number of locations.

2. Compute the probability of selecting an empty node:

$$p_\theta = \frac{\exp(-\bar{\varepsilon}\theta)}{2}$$

3. Sample the number of empty nodes to be added:

$$k \sim \text{Binomial}(m, p_\theta)$$

4. Randomly select k empty nodes from the m candidates and assign noisy counts according to the probability distribution described in the following paragraph.

This probabilistic inclusion of empty nodes ensures that even unseen or rare paths have a controlled chance of appearing in the published data. It preserves the structural consistency of the prefix tree, improves the utility of the released data, and prevents privacy breaches due to deterministic removal of infrequent paths.

Probability Distribution for Noisy Counts of Empty Nodes. For the k selected empty nodes, the probability density function (PDF) of the noisy counts x, conditional on $x \geq \theta$, is:

$$P(x) = \begin{cases} 0, & x < \theta, \\ 1 - \exp(\bar{\epsilon}\theta - \bar{\epsilon}x), & x \geq \theta. \end{cases}$$

where

$$p_\theta = \frac{\exp(-\bar{\varepsilon}\theta)}{2}$$

is the probability that a noisy count for an empty node exceeds the threshold θ.

Substituting p_θ into the PDF yields the simplified form:

$$p(x \mid x \geq \theta) = \bar{\varepsilon} \exp(\bar{\varepsilon}\theta - \bar{\varepsilon}x), \quad x \geq \theta.$$

The corresponding cumulative distribution function (CDF) is:

$$P(x) = \begin{cases} 0, & x < \theta, \\ \displaystyle\int_\theta^x \bar{\varepsilon} \exp(\bar{\varepsilon}\theta - \bar{\varepsilon}t)\, dt = 1 - \exp(\bar{\varepsilon}\theta - \bar{\varepsilon}x), & x \geq \theta. \end{cases}$$

This formulation ensures that the noisy counts assigned to the selected empty nodes follow an exponential decay distribution shifted by θ, thereby preserving the differential privacy guarantees while allowing a controlled inclusion of infrequent or unobserved paths in the prefix tree.

Proof of Distribution Preservation for Binomial Technique

The Binomial selection method yields the same output distribution for the private prefix tree as the standard Laplace-based (or its discrete counterpart, the Geometric) thresholding that tests each empty node independently. In the standard approach [76, 77], for each empty node, Laplace noise with scale $1/\bar{\epsilon}$ is added to the count (which is initially zero), and the node passes if the noisy value exceeds a threshold θ. The probability of passing is

$$p_\theta = \frac{\exp(-\bar{\epsilon}\,\theta)}{2},$$

and the number of passing empty nodes therefore follows the binomial distribution $B(m, p_\theta)$, where m is the number of empty nodes to be tested.

The Binomial approach first samples $k \sim B(m, p_\theta)$, then selects k empty nodes uniformly at random without replacement, and finally assigns noisy counts to them drawn from the conditional Laplace distribution given that the noisy value exceeds θ:

$$P(x) = \begin{cases} 0 & x < \theta, \\ 1 - \exp(\bar{\epsilon}\,\theta - \bar{\epsilon}\,x) & x \geq \theta. \end{cases}$$

As shown in [76, 77], this procedure produces exactly the same joint distribution over the set of surviving empty nodes and their noisy counts as the per-node Laplace thresholding, while avoiding the need to process all m empty nodes individually.

It should be noted that some prior works (e.g., trajectory publishing approaches without explicit empty-node handling, as discussed in Jin et al., 2021 – SOK) skip zero-count nodes entirely and thus cannot directly benefit from this Binomial efficiency improvement, because their construction step does not require testing all empty nodes. Works that do handle empty nodes explicitly can adopt this technique without changing the distribution.

Theorem 5.2.1. *Chen et al. [27] with the binomial technique produces the same distribution as the original mechanism.*

The proof structure in [76, 77], is provided in **Appendix C**.

Post-processing and Parent–Child Count Consistency

After constructing the Noisy Prefix Tree (NPT) and applying pruning, the injected Laplace noise and the removal of certain paths may result in violations of logical consistency constraints between parent and child counts. One of the key constraints is that the count of a parent node must be greater than or equal to the sum of its children's counts. Violating this constraint may lead to logically impossible paths, such as partial routes without their corresponding complete routes.

Following the approach of Chen et al. [27], a post-processing step is applied to enforce consistency. This is done in two stages: path smoothing and parent–child adjustment.

Definition 5.2.1 (Consistency Constraints)**.** The consistency rules for the noisy prefix tree are defined as follows:

1. **Path Constraint:** Along a path from the root to a leaf, the count of each node must be no less than the count of its child:

$$\forall v_i \in p, \quad |tr(v_i)| \geq |tr(v_{i+1})|.$$

2. **Parent–Child Constraint:** The count of a parent must be at least the sum of the counts of all its direct children:

$$|tr(v)| \geq \sum_{u \in children(v)} |tr(u)|.$$

Chen et al. (2011) Approach As described in Definition 5.2.1, these consistency rules may be violated after noise injection and pruning. Chen et al. [27] proposed a constrained inference method in two stages to enforce them:

Stage 1: Path Smoothing

- All root-to-leaf paths are extracted.
- For each path p, the counts are stored as a sequence:

$$S = \langle c(v_1), c(v_2), \ldots, c(v_{|p|}) \rangle,$$

  where v_i is the child of v_{i-1}.
- Using the L_2-minimization method from [31], counts are adjusted so that they satisfy the path constraint.

- Since a node may appear in multiple paths, the mean of all its adjusted values is used as its intermediate estimate $e_c(v)$.

Stage 2: Parent–Child Adjustment

- A top-down process ensures that the parent–child constraint holds.
- If $\sum_{u \in children(w)} e_c(u) > \bar{c}(w)$, the excess is subtracted *equally* from all children:

$$\bar{c}(v) = e_c(v) + \min\left(0, \frac{\bar{c}(w) - \sum_{u \in children(w)} e_c(u)}{|children(w)|}\right).$$

- Counts are only decreased (never increased) to avoid creating artificial extensions of routes.

Limitations of the Chen et al. Method

- Equal budget allocation distributes the reduction evenly across all child nodes, regardless of their frequency or importance. As a result, differences in counts among the nodes are overlooked, which may lead to smaller nodes being disproportionately reduced or even eliminated.
- May produce negative counts if the reduction is large.
- May distort high-utility paths by over-reducing their counts.

5.2.2 PPDP (Yang Li et al., 2020)

Summary

The algorithm proposed by Li et al. [28] is the current state-of-the-art which addresses the issue of excessive privacy budget consumption by preserving timestamp–location pairs in trajectories and eliminating the need for taxonomy trees. It employs an incremental privacy budget allocation model, where the per-level budget increases with tree depth, combined with a decreasing threshold function for more accurate node retention or pruning. Noise is injected using the Laplace mechanism with sensitivity 1, and nodes with counts below the level-specific threshold are pruned.
To improve efficiency, a spatial–temporal dimensionality reduction model is introduced, incorporating a network geometric accessibility constraint and a minimum required travel time matrix (K_L) to restrict candidate locations to those reachable within the allowable time interval, without consuming additional privacy budget.

In our implementation, we extend Li et al.'s framework by introducing an additional binomial hypothesis testing step during node expansion. This step filters out nodes whose likelihood of appearing in the real dataset is statistically negligible, even after applying the K_L constraints. This optimization significantly reduces the number of comparisons, improves tree construction speed, and lowers the accumulated noise in the final published dataset.

Spatial-Temporal Prefix Tree Construction

As discussed in the *Prefix Tree Mechanism* section of Chapter 2, this method directly models the raw spatio-temporal data. Unlike hierarchy-based approaches, which first generalize locations to coarser levels (such as regions or cities), this mechanism operates directly on the precise spatial coordinates present in the raw data. This preserves the order and detail of the spatio-temporal pairs during prefix tree construction, enabling a more accurate representation of user trajectories.

In the algorithm proposed by Li et al. [28], the input data are represented as sequences of $(\text{timestamp}, \text{location})$ pairs. The goal of constructing the **spatio-temporal prefix tree** is to represent all observed sub-trajectories in the dataset, such that each node corresponds to a prefix of the trajectories.

1. Tree Structure Definition. A prefix tree $PT = (Root, E, V)$ consists of:

- **Root:** The starting point of all trajectories, whose initial count equals the total number of trajectories.
- **Edges** E: Each edge encodes a $(\text{time}, \text{location})$ pair.
- **Nodes** V: Each node stores the count of a specific sub-trajectory.

For a node v_i:

- The incoming edge e_{in} carries a (t, l) pair from its parent.
- The outgoing edges e_{out} represent possible subsequent steps.

2. Difference from Prior Approaches. Previous methods such as SeqPT [33] and SafePath [47] relied on constructing separate taxonomy trees for time and location, which split the privacy budget across multiple dimensions and reduced accuracy as the tree depth d increased. In contrast, this algorithm allocates the entire per-level privacy budget directly to the actual nodes without building additional classification structures.

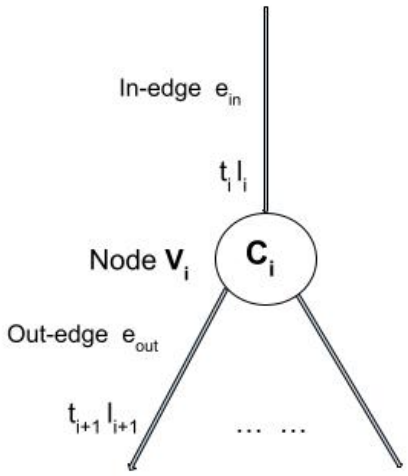

Figure 5.1: In- and out- edge of node.

3. Tree Construction Process

Step 1 – Initial Tree Building:

1. Read the raw dataset of $(\text{time}, \text{location})$ sequences.
2. Starting from the root, create or update nodes for each trajectory.
3. Continue until reaching the end of the trajectory or the maximum length $L_{\max}$.

Step 2 – Spatio-temporal Background Matrix:

- Compute a matrix K_L representing the minimum travel time between every pair of locations:

$$K_L[i][j] = \text{minimum travel time from location } l_i \text{ to } l_j$$

- This matrix is computed from the network distance and free-flow speed, and requires no additional privacy budget.
- When extending trajectories, only locations reachable within the time interval between two timestamps according to K_L are considered.

Step 3 – Geographic Accessibility Constraint: Let the incoming edge to node v_i be (t_i, l_i). To create a new edge (t_{i+1}, l_{i+1}):

1. Ensure $t_{i+1} > t_i$.
2. Compute $\Delta t = t_{i+1} - t_i$.

3. Select only locations l_{i+1} such that $K_L[l_i][l_{i+1}] \leq \Delta t$.

This restriction narrows the search space and reduces computational cost.

Step 4 – Noise Addition and Pruning:

- Add Laplace noise with sensitivity 1 to the count of each node in the current level.
- Prune nodes whose noisy counts fall below the level-specific threshold.

Privacy Budget Allocation

In this algorithm, the total privacy budget ε is allocated in a level-by-level manner across the nodes of the prefix tree. This approach differs fundamentally from earlier models such as *SeqPT* and *SafePath*.

1. Core Idea

- The prefix tree is constructed starting from the root, expanding level-by-level.
- Nodes at the same level are disjoint with respect to the dataset; therefore, according to the *Parallel Composition* property of Differential Privacy (DP), all nodes at a given level share the same budget.
- In previous approaches, the budget was evenly divided across all levels ($\ell_i = \ell_{i+1}$). However, Li et al. argue that:
    1. Counts c_i in upper levels are generally large, so the relative impact of noise is smaller.
    2. Counts in lower levels are smaller, making them more sensitive to even small amounts of noise.
- To address this, they propose an **Incremental Privacy Budget Allocation** model.

2. Incremental Budget Model

Let:

$$h = \text{height of the tree}, \quad \ell_i = \text{budget for level } i, \quad \theta_i = \text{threshold for level } i.$$

Formulas:

1. **Budget function** (increases with depth):

$$\ell_i = k \times i + b \quad \text{where} \quad k > 0,\ b > 0$$

ensuring $\ell_i < \ell_{i+1}$ so that deeper levels receive larger budgets.
k controls the decay rate of the threshold across levels, while b provides a baseline minimum threshold that prevents spurious noisy nodes from surviving at deeper levels.

2. **Threshold function** (decreases with depth):

$$\theta_i = \theta_0 \times \frac{1}{\log(i+1)}$$

meaning that upper levels apply stricter thresholds, while deeper levels are more permissive.

3. Rationale

Because $c_i \geq c_{i+1}$ (the sum of a node's children equals the parent's count but usually decreases with depth), a fixed budget would cause the noise-to-signal ratio to grow significantly in deeper levels. Incremental allocation mitigates this by providing more budget to deeper levels, thus reducing relative noise and producing more realistic released data.

4. Use in the Algorithm

Within the `HandleSubTree` module:

1. At the start of each level:

$$\ell_i \quad \text{and} \quad \theta_i \quad \text{are computed based on level } i.$$

2. Laplace noise is added to each node count:

$$c' = c + \text{Lap}\left(\frac{1}{\ell_i}\right)$$

since the sensitivity of the counting function is 1.

3. Node retention or deletion:

   - If $c' \geq \theta_i \rightarrow$ retain the node.
   - Otherwise $\rightarrow$ prune the node.

4. For non-existent nodes:

   - Start with count 0, then add Laplace noise.

- If the noisy count exceeds θ_i, the node is added.

5. Consistency check: the noisy sum of children must not exceed the noisy parent count; adjustments are made if necessary.

5. Output

After noise addition and pruning across all levels, the result is a noisy prefix tree that satisfies ε-Differential Privacy. A single traversal of this tree yields the sanitized trajectories.

Pruning Strategy and Threshold θ

Pruning is guided by a level-specific threshold θ_i:

- Nodes with noisy counts below θ_i are removed.
- This serves two purposes:
  1. Reducing cumulative noise in the final dataset.
  2. Preventing unnecessary tree growth and lowering computational complexity.
- The threshold decreases with depth so that lower-level nodes, which tend to have smaller counts, are more likely to be preserved.

Empty Node Checking and Spatio-Temporal Matrix

One of the main innovations of Li et al. (2020) compared to *SeqPT* and *SafePath* is the method for selecting and expanding candidate nodes during prefix tree construction:

- Previous methods considered all possible time–location combinations at each expansion step, leading to many empty nodes and wasted budget.
- This algorithm uses a spatio-temporal prior knowledge matrix to enforce two constraints:
  1. **Network Geometric Accessibility Constraint** – Only locations reachable within the available time window between parent and child timestamps are considered.
  2. **Minimum Required Travel Time Matrix** K_L is a $|L| \times |L|$ matrix where $K_L[i][j]$ stores the minimum travel time from location l_i to l_j under free-flow speed conditions. Candidate locations must satisfy $K_L[i][j] \leq \Delta t$.
- This filtering step:

- Reduces empty nodes,
- Lowers runtime,
- Improves accuracy without consuming additional privacy budget.

Post-processing and Parent–Child Count Consistency Guarantee

In the implementation of the Li et al. (2020) algorithm, post-processing is not performed as a separate phase after the prefix tree is fully constructed. Instead, it is integrated into the `HandleSubTree` module as an *in-process* step. This design ensures that structural consistency is enforced *during* the tree growth process, minimizing the negative impact of noise on the logical integrity of the tree.

The process works as follows:

1. **Removal of Low-Importance Noisy Nodes** – If the noisy count of a node is less than the level-specific threshold λ_l, the node is immediately discarded.
2. **Retention of Zero-Count Nodes with Positive Noise** – If a node has an initial count of zero in the raw data but the added noise increases it to a value $\geq \lambda_l$, the node is temporarily retained.
3. **Parent–Child Consistency Constraint** – Nodes retained solely due to positive noise are kept *only* if the sum of the noisy counts of all their children does not exceed the noisy count of their parent. This condition, similar to that used in Chen et al. (2011) and Cormode et al. (2018), ensures that hierarchical counts remain valid.
4. **On-the-Fly Enforcement** – Unlike approaches that first construct the entire tree and then apply post-processing, this method prevents inconsistent or unnecessary nodes from entering the tree in the first place.

This approach provides two key advantages:

- *Reduced structural distortion from noise*, since consistency checks are applied at the decision-making stage rather than after full construction.
- *A final sanitized tree that is inherently hierarchy-consistent*, without requiring major node removal or adjustment at the end of the process.

Improvements of Li et al. (PPDP) over Li et al. (NPT)

Improvement 1 – New Prefix Tree Structure

- No Taxonomy Tree is used.

- The tree consists solely of nodes representing $(\text{timestamp}, \text{location})$ pairs.
- This avoids the indirect consumption of privacy budget caused by building separate spatial and temporal taxonomy trees, as in earlier works.

Improvement 2 – Incremental Privacy Budget Allocation Model Instead of equally dividing the privacy budget among all tree levels:

- **Upper levels**, with high data density, receive less budget (since noise has less relative impact).
- **Lower levels**, with sparse branches, receive more budget (to reduce the distortion caused by noise).

The budget and threshold functions are defined as:

$$\lambda(l) = \frac{\log(l+1)}{\log(h+1)}, \quad \theta(l) = k \times \lambda(l) + b$$

where:

- l = current tree level,
- h = total tree height,
- k, b = tunable parameters.

Improvement 3 – Spatial–Temporal Domain Reduction

- Instead of considering all possible $(\text{timestamp}, \text{location})$ combinations for expanding new branches:
  - Only locations **temporally reachable** from the current node are considered.
  - A **minimum travel time matrix** KL is precomputed, storing the shortest travel time between each pair of locations based on free-flow speed.
  - Invalid expansions (e.g., traveling 5 km in 1 minute) are eliminated without consuming any privacy budget.

Up to this point, we have described our implemented algorithms — including both the original versions and the improved variants based on Li et al. (NPT) and Li et al. (PPDP) — along

with detailed interpretations and our proposed modifications. These explanations covered the construction of the spatial–temporal prefix tree, the incremental privacy budget allocation model, geographic and temporal accessibility constraints, and statistical test–based optimizations for pruning low-importance nodes. From this section onward, we focus on the elements common to all implementations. We first present the experimental parameters, evaluation metrics, and performance analysis methods, followed by an assessment of algorithm efficiency and privacy preservation.

5.2.3 Our Technical Improvements

While the original NPT [27] and PPDP [28] algorithms provide strong theoretical frameworks, several practical limitations emerge when adapting them to real-world bus tap card data. In particular, two aspects required refinement: (1) the post-processing step of NPT, which may distort trajectory counts when reductions are applied equally to all children, and (2) the treatment of low-count nodes in PPDP, which can result in excessive noise and unnecessary computational cost. This subsection summarizes our technical contributions to improve both algorithms.

Improvements to NPT

Motivation. The original NPT post-processing applies equal reductions across all children when the sum of children's counts exceeds the parent's count. This approach disregards the differences in frequency or importance of nodes, leading to several problems:

- Smaller nodes may be disproportionately reduced or eliminated.
- Large reductions may even yield negative counts.
- High-utility paths can be distorted due to uniform adjustments.

Proposed Weighted Adjustment Method. To address these issues, we propose a **weighted reduction strategy**, where adjustments are distributed proportionally according to each child's relative weight:

1. **Step 1:** Remove pruned nodes and recompute counts for the remaining tree.
2. **Step 2:** If
$$\sum_{u \in children(w)} e_c(u) > \bar{c}(w),$$
the reduction amount is distributed proportionally to each child's relative frequency:
$$c_{\text{new}}(u) = c_{\text{old}}(u) - \left\lceil \frac{c_{\text{old}}(u)}{\sum_{j \in children(w)} c_{\text{old}}(j)} \times \text{total\_to\_reduce} \right\rceil.$$

where

$$\text{total\_to\_reduce} = \sum_{u \in children(w)} e_c(u) - \bar{c}(w).$$

Advantages. This weighted adjustment method:

- Respects differences in node frequency (larger nodes take a proportionally larger share of the reduction, while smaller nodes are reduced less, preventing their elimination).
- High-utility paths can be distorted under uniform adjustment, but the weighted strategy preserves them more faithfully.
- Prevents negative counts.
- Preserves the statistical distribution of the original noisy counts more faithfully.

Together, these refinements make the improved NPT post-processing step more reliable and better aligned with both theoretical consistency constraints and practical data utility.

Improvements to PPDP

Motivation. The PPDP algorithm already incorporates spatio-temporal constraints and a minimum travel-time matrix (K_L) to reduce empty-node expansion. However, in our dataset—which only contains tap-in events without tap-out information—many nodes with extremely low or noise-only counts still remain, due to the possibility of arbitrarily long gaps between trips. These nodes inflate computational cost and contribute to excess noise in the released data.

Technical Improvement: More Efficient Noising of Low-Count Nodes. To mitigate this issue, we introduce a **Binomial hypothesis test** during node expansion, applied in addition to K_L filtering. This test statistically prunes nodes whose likelihood of appearing in the real dataset is negligibly small, ensuring that only plausible nodes are retained.

Advantages. This optimization provides three key benefits:

- Fewer comparisons in later tree-building stages,
- Reduced processing time due to a smaller search space,
- Less accumulated noise in the final released dataset.

Together, these refinements make our improved PPDP algorithm more computationally efficient and enhance the utility of the privatized trajectories.

5.3 Dataset

The experiments in this thesis were conducted using a real-world smart card dataset provided by BC Transit for the Victoria, BC, region. The dataset contains tap-in records collected over a continuous six-day period, from December 15 to December 20, 2024.

In total, the dataset comprises 312,659 records, each representing a single passenger boarding event. For each record, the following attributes are included:

- **Route name:** The official name or code of the bus route.
- **Stop ID:** A unique identifier for the bus stop.
- **Timestamp:** The exact date and time of the boarding event.
- **Bus number:** The vehicle's identification number.
- **Stop coordinates (latitude, longitude):** The geographical location of the stop.
- **Seats:** The number of seats available on the bus for that record, indicating the vehicle type and capacity.

Unlike synthetic datasets used in previous studies, this dataset is entirely derived from real operational data and reflects the actual passenger boarding patterns within the Victoria transit network during the study period.

In addition to the main dataset, several publicly available datasets provided by BC Transit were used for pre-processing and executing certain queries, building spatio-temporal matrices for travel pattern analysis, as well as for the synthetic trajectory generation algorithm. These include both static and real-time GTFS data files such as:

- Stops, Routes, Trips, Stop Times, Real-time updates (including trip updates), vehicle updates, and alerts.

These supplementary datasets were utilized for purposes such as:

- Finding the latitude and longitude of stops
- Determining the valid stop domain for the Victoria region
- Identifying inactive stops during the study period
- Linking tap-in records to their corresponding route, trip, and vehicle information

This combination of primary and supplementary data ensured that both the spatial and temporal aspects of bus stop locations were accurately incorporated into preprocessing, trajectory reconstruction, and algorithm evaluation.

5.4 Parameters and Evaluation Metrics

Parameter / Metric	**NPT(Chen et al.)**	**PPDP(Li et al.)**	**Purpose / Notes**
Total privacy budget	ϵ	ϵ	Controls noise level; smaller $\epsilon \rightarrow$ stronger privacy, lower accuracy.
Per-level privacy budget	$\bar{\epsilon} = \frac{\epsilon}{h}$	$\epsilon_l = \frac{\lg(l+\sigma)}{\sum_{l=1}^{h} \lg(l+\sigma)} \times \epsilon$	Defines noise allocation per tree level.
Smoothing factor	—	$\sigma > 0$	Adjusts budget allocation curve so deeper levels receive some budget.
Max tree height h	chosen from real trip length distribution	chosen from real trip length distribution	Limits maximum trajectory length.
Pruning threshold	$\theta = \frac{2\sqrt{2}}{\bar{\epsilon}}$	$\theta_l = k \times l^{-1} + b$	Removes nodes with very noisy counts to reduce error from noise propagation.
Pruning decay factor	—	$k > 0$	Controls how quickly pruning threshold decreases with depth.
Baseline threshold	—	$b > 0$	Ensures a minimum pruning cutoff regardless of depth.
Level index	l (1 to h)	l (1 to h)	Used in formulas for ϵ_l and θ_l.
Dataset	Sequential location IDs only (no timestamps)	Sequential location IDs, timestamps	Real BC Transit dataset.
Evaluation metrics			Precision, Recall, Top-k Path Accuracy, Runtime, Count Query Relative Error, running practical queries.

Table 5.1: Summary of Parameters and Evaluation Metrics for NPT and PPDP algorithms.

Note: Here, l^{-1} denotes the inverse of level l, i.e., $1/l$.

Experimental Parameters

ϵ : Privacy Budget Tested values: 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, and 2.0. A smaller ϵ implies stronger privacy protection but lower accuracy due to higher noise. A larger ϵ implies weaker privacy but higher accuracy due to reduced noise.

h : Maximum Tree Height The maximum length from the root to a leaf in the prefix tree. Choosing h:

- If h is too small, true trajectories are truncated and lose details.
- If h is too large, the noise budget is split across more levels, increasing per-level noise and reducing accuracy.

Before selecting h, the distribution of actual trip lengths in the Victoria dataset was analyzed (Figure 5.2). The chosen h:

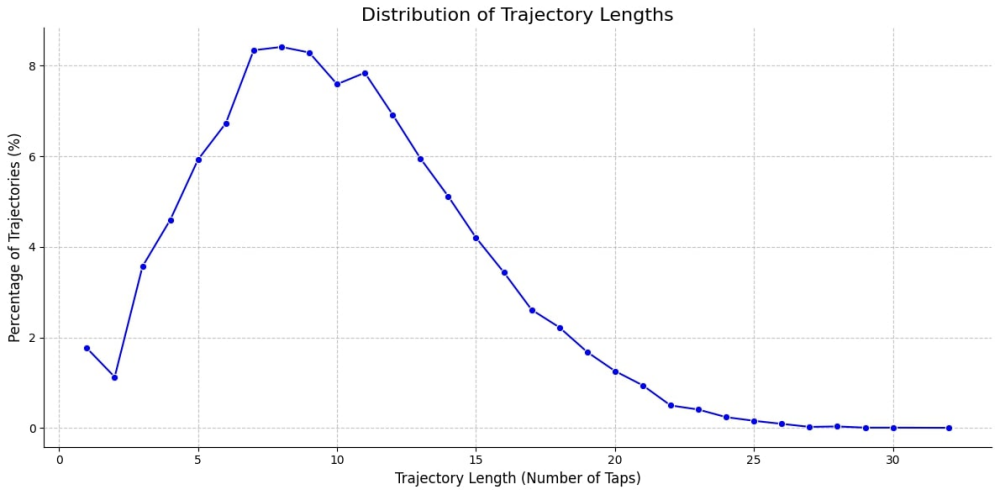

Figure 5.2: The distribution of trajectory length in the Victoria dataset

1. Covers the majority of real-world paths.
2. Ignores a small number of unusually long trips that would unnecessarily increase noise.

θ — Pruning Threshold Set according to the formula in [27]:

$$\theta = \frac{2\sqrt{2}}{\bar{\epsilon}},$$

where $\bar{\epsilon} = \frac{\epsilon}{h}$. If the noisy count of a node is less than θ, that node and its subtree are pruned.

Evaluation Metrics

1. **Precision:** Percentage of retrieved trajectories that are correct.
2. **Recall:** Percentage of actual trajectories present in the released dataset.
3. **Top-k Path Accuracy:** Agreement between the most frequent paths in the original and private datasets.
4. **Runtime:** Time taken to build the prefix tree and generate private data.
5. **Count Query Relative Error:** Mean relative error over a set of random counting queries.

Queries used for Utility Evaluation

To evaluate the algorithm's performance in real-world scenarios, we first reviewed BC Transit's original wishlist for valuable operational insights. According to BC Transit, key areas of interest include:

- Understanding where, how often, and how busy transit services are to assess service delivery effectiveness.
- Determining whether services are provided in the right locations, at the right frequency, and with appropriate vehicle types (High-Capacity vs. Medium vs. Light Duty).
- Exploring the potential impact of population density changes on ridership.
- Identifying opportunities to reduce travel time by closing low-impact bus stops or re-routing vehicles.
- Gaining insight into peak ridership patterns, particularly around the University and government offices, with bi-modal peaks during morning and afternoon commutes, highest ridership mid-week, and seasonal variations (e.g., September busiest, August lowest).

Based on these priorities, we designed and executed the following operational queries on both original and noisy datasets:

1. Most frequent stop over the full period — comparing rank and passenger count.
2. Most frequent stop in a specific time range (e.g., 7–9 a.m.) — comparing rankings and passenger count.
3. Least frequent stops — comparing rank and passenger count.
4. Route with the highest passenger flow — comparing consistency of results.

Proof of Differential Privacy

The purpose of this section is to provide a formal proof that the proposed algorithms satisfy the requirements of ϵ-differential privacy (DP) in publishing trajectory data. In DP, the guarantee must hold for the *entire* data release process.

Theorem 5.4.1. *The NPT Algorithm is ϵ-differentially private.*

Proof of ϵ-Differential Privacy for the NPT Algorithm. Let $\mathcal{D}$ and $\mathcal{D}'$ be user-level neighboring datasets. The Chen et al. algorithm [27] consists of:

Stage 1 – `BuildNoisyPrefixTree`: At each level i ($1 \leq i \leq h$), nodes correspond to disjoint subsets of trajectories. Per-level budget: $\bar{\epsilon} = \epsilon/h$. By the parallel composition theorem (Theorem 3.4 in [27]), the cost for processing one level is $\bar{\epsilon}$. By sequential composition (Theorem 3.3 in [27]), the total cost is:

$$\sum_{i=1}^{h} \bar{\epsilon} = h \times \frac{\epsilon}{h} = \epsilon.$$

Thus, Stage 1 is ϵ-DP.

Stage 2 – `GeneratePrivateRelease`: Applies deterministic post-processing to noisy counts from Stage 1. By the post-processing immunity theorem [25], ϵ-DP is preserved.

Therefore, the NPT Algorithm is ϵ-differentially private.

□

Theorem 5.4.2. *The PPDP Algorithm is ϵ-differentially private.*

Proof of ϵ-Differential Privacy for the PPDP Algorithm. Let $\mathcal{D}$ and $\mathcal{D}'$ be user-level neighboring datasets. The Li et al. algorithm [28] uses:

Stage 1 – Trajectory Segmentation & Transformation: This stage transforms raw trajectories into time–location records and applies Laplace noise to each count query with parameter ϵ_1. By the Laplace mechanism theorem [25], each query is ϵ_1-DP. Queries on disjoint subsets use parallel composition; queries on overlapping subsets use sequential composition. Total privacy loss in this stage is bounded by ϵ_1.

Stage 2 – Consistency Enforcement: Enforces logical constraints (e.g., non-negativity, parent $\geq$ sum of children) via deterministic post-processing. By the post-processing theorem, this stage consumes no additional privacy budget.

Budget Accounting: If the total privacy budget ϵ is split as $\epsilon = \epsilon_1 + \epsilon_2$, with ϵ_2 assigned to any additional noisy releases (e.g., aggregated flows), then by sequential composition the total cost is ϵ. If $\epsilon_2 = 0$ (pure post-processing), the cost is exactly $\epsilon_1 = \epsilon$.

Thus, the PPDP algorithm satisfies ϵ-differential privacy. □

Computational Complexity

The total runtime complexity of the algorithm is:

$$O(|D| \cdot |L|),$$

where:

- $|D|$ is the number of trajectories in the input dataset.
- $|L|$ is the size of the location universe (number of distinct stops).

BuildNoisyPrefixTree The dominant cost in constructing the Noisy Prefix Tree (NPT) comes from:

1. **Node generation:** For each tree level, the number of new nodes is proportional to $k|D|$, where $k \ll |L|$ depends on the branching factor and $|L|$.
2. **Trajectory distribution:** At each level, up to $|D|$ trajectories are assigned to newly generated nodes.

Thus, the complexity per level is $O(|D| \cdot |L|)$. With at most h levels, the complexity of building the NPT is:

$$O(h \cdot |D| \cdot |L|).$$

Since h is a small constant (bounded by the maximum real trajectory length), this term simplifies to $O(|D| \cdot |L|)$.

GeneratePrivateRelease Post-processing has three main steps:

1. **Intermediate estimates:** Computing estimates for a single root-to-leaf path is $O(h^2)$. With at most $|D|$ distinct paths, the total cost is $O(h^2 \cdot |D|)$.
2. **Consistency enforcement:** Each node is visited twice, resulting in $O(|D| \cdot |L|)$ complexity.
3. **Private release generation:** A single postorder traversal has complexity $O(|D| \cdot |L|)$.

Overall Complexity Since h is much smaller than $|D|$ and $|L|$ in practice, the overall computational complexity is:

$$O(|D| \cdot |L|).$$

As reported in [27], the cost scales linearly with both the number of trajectories and the number of distinct locations.

5.5 Operational Queries

To evaluate the practical value of privatized trajectory data for BC Transit, we define a set of **operational queries**. These queries go beyond synthetic metrics such as relative error and are designed to answer real-world planning and analysis needs of a transit agency. In this section, we describe seven operational queries that will later be evaluated on both NPT and PPDP outputs in Chapter 6.

1. **Total number of tap-ins.** Purpose: Estimate overall ridership demand.
2. **Tap-ins per stop.** Purpose: Identify stop-level demand and highlight busy vs. underutilized stops.
3. **Tap-ins by time interval (e.g., hourly).** Purpose: Detect peak demand periods.
4. **Most frequent stops (from Query 2 sorted descending).** Purpose: Locate high-demand stops that may require additional service or larger buses.
5. **Least frequent stops (from Query 2 sorted ascending).** Purpose: Detect underused stops that may justify reducing service.
6. **Frequent short sequences (2-stop patterns).** Purpose: Reveal common short travel paths that inform service adjustments or new route planning.

These operational queries are intentionally simple and directly computable from privatized tap-in trajectories. They provide a bridge between theoretical differential privacy guarantees and the practical information needs of transit planners.

Chapter 6

Experiment Results and Analysis

6.1 Chapter Objective

The objective of this chapter is to present and analyze the results of experiments conducted to evaluate the performance and accuracy of two selected algorithms—*Noisy Prefix Tree (NPT)* and *Privacy-Preserving Data Publishing (PPDP)*—on real BC Transit data and a set of synthetic trajectories reconstructed from this real data.

In this chapter, the performance of these two algorithms is evaluated in response to a set of analytical scenarios relevant to BC Transit's operational needs. These scenarios include *Count Queries* with various parameters and categories, as well as *Frequent Sequential Pattern Mining*, executed on both the original and privatized datasets.

Beyond the conventional count queries common in related studies [27, 28], we also assess algorithm performance based on practical, real-world queries required by BC Transit. These queries are derived from BC Transit's initial list of operational needs, as described in Ch. 5.

This combination of quantitative evaluation (using statistical metrics) and qualitative evaluation (using operational queries) allows us to assess not only the numerical accuracy of the algorithms but also the analytical value of their results for operational decision-making.

To measure quality and performance, we use metrics such as Relative Error, Precision, Recall, and F-score. We examine the impact of key parameters including the privacy budget ϵ, tree height h, and the number of frequent patterns k on the evaluation results.

This thesis evaluates improved variants of both the Noisy Prefix Tree (NPT) method and the Privacy-Preserving Data Publishing (PPDP) method. The NPT method lacked a robust post-processing step to enforce parent–child consistency; this limitation was addressed through a weighted post-processing method. The PPDP method, while accurate, incurred long runtimes due to reliance on location–time matrices; this was mitigated by applying spatio-temporal filtering to reduce the

search space.

Finally, the results in this chapter provide both quantitative and qualitative evidence demonstrating how public trajectory data can be released under strong differential privacy guarantees without losing the ability to answer important analytical queries.

6.2 Experimental Setup

The experiments in this chapter were conducted on the real-world BC Transit tap-in dataset described in detail in Ch. 5. The dataset contains 312,659 boarding records over six consecutive days (December 15–20, 2024) and includes attributes such as route name, stop ID, timestamp, bus number, stop coordinates, and seat capacity. During preprocessing, missing route and stop information was first completed using an internal reference based on bus numbers, and geographic coordinates of stops were then attached from the BC Transit stops file. Finally, records that could not be completed or categorized by any method were removed.

We tested multiple privacy budgets ϵ ranging from 0.5 to 2.0. The maximum tree height h was selected based on the observed distribution of trip lengths in the Victoria dataset (see Figure 5.2 in Ch. 5), covering the majority of real-world paths while ignoring a small number of unusually long trips that would unnecessarily increase noise.

For Frequent Sequential Pattern Mining tasks, the number of patterns k was varied in $\{5, 10, 15, 20\}$ to assess the trade-off between coverage and false positives.

The experiments were designed to measure both conventional count-query utility and operationally relevant queries based on BC Transit's needs. Detailed parameter definitions, pruning thresholds, and evaluation metrics are provided in Ch. 5.

Average Relative Error (ARE). To quantify the accuracy of query answers in the privatized dataset, we compute the *Average Relative Error (ARE)* across all evaluated queries. For a given query q, let $A(q)$ denote the true answer on the original dataset and $\tilde{A}(q)$ denote the answer on the privatized dataset. The relative error for q is defined as:

$$RE(q) = \frac{|\tilde{A}(q) - A(q)|}{\max\{A(q),\ s\}}, \tag{6.1}$$

where s is a *sanity bound* to stabilize the denominator for small-answer queries. Consistent with prior work, we set

$$s = 0.1\% \times N,$$

where N is the dataset size.

This sanity-bound definition avoids inflated ratios when $A(q)$ is very small, yields numerically

stable errors within each query family, and makes our results directly comparable to prior literature.

The ARE over a query set Q is then

$$ARE = \frac{1}{|Q|} \sum_{q \in Q} RE(q). \tag{6.2}$$

This metric provides a normalized measure of deviation that is comparable across queries of different scales.

6.3 Count Query Utility

Introduction

To assess the accuracy of noisy prefix trees in responding to count queries, a total of 40,000 queries—drawn from the four groups defined in Chapter 5—were executed. These queries were applied to the original dataset as well as to multiple differentially private releases produced using two widely cited prefix-tree–based methods [27, 28] and their enhanced variants. The evaluation was carried out under consistent experimental conditions, enabling direct comparison between the baseline approaches and their improved forms.

6.3.1 Effect of Privacy Budget (ϵ)

Setup: The noisy prefix tree height h was fixed at values ranging from 3 to 12 (four different settings), and the average relative error (ARE) was measured for ϵ values from 0.5 to 2.0. Each point in the plots represents the average error over the 40,000 queries.

Results:

- In Figure 6.1 (NPT) and Figure 6.2 (PPDP), higher ϵ values always reduced the average relative error (ARE). This happens because less noise is added and the counts are reconstructed more accurately.

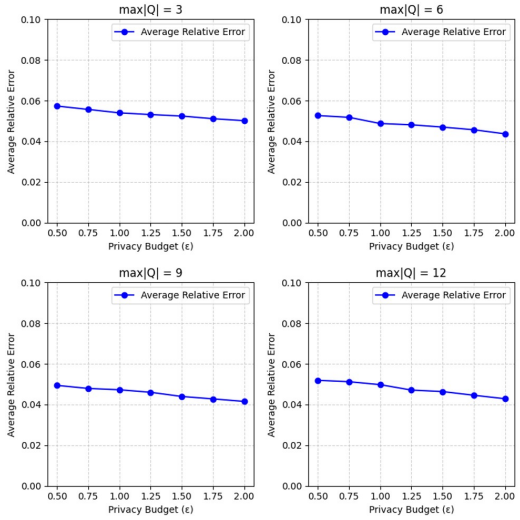

Figure 6.1: Average relative error vs. privacy budget-NPT.

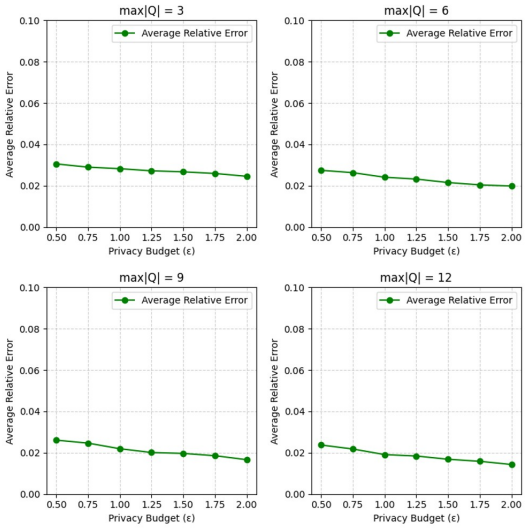

Figure 6.2: Average relative error vs. privacy budget-PPDP.

- Both prefix-tree methods kept good accuracy for count queries in all tested privacy budgets. In the strictest case ($\epsilon = 0.5$, max $|Q| = 3$), ARE was below 6% for NPT and below 3% for PPDP. These results are not the same — PPDP performed better because it uses progressive budget allocation and keeps both spatial and temporal details.
- Figure 6.3 shows that the improved (Chen et al.) NPT algorithm always had lower ARE than the original, with the gap larger at higher ϵ. This is because the improved method fixes parent–child differences based on each child's noisy count (weighted), instead of sharing the difference equally. This keeps the data more consistent and reduces errors.
- PPDP generally gave lower ARE than NPT in all ϵ values. Its spatio-temporal filtering and progressive budget allocation are the main reasons for this advantage. In Figure 6.4, PPDP lines are always below the NPT lines.
- Adding weighted post-processing to NPT gave small but clear improvements, especially when ϵ was low, by reducing mismatches between parent and child counts (Figure 6.3).

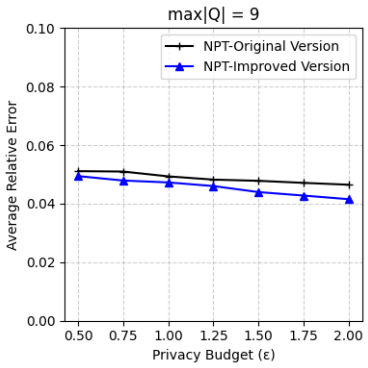

Figure 6.3: Average relative error comparison between the original algorithm and improved NPT at $h = 9$.

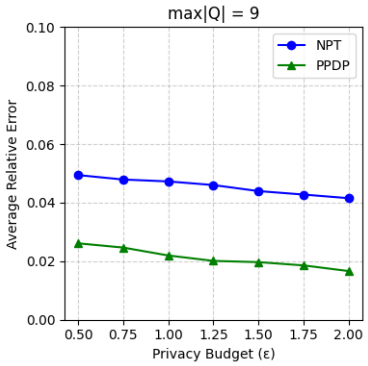

Figure 6.4: Average relative error comparison between the NPT method (Chen et al., 2011) and the PPDP method (Li et al., 2020) at $h = 12$.

6.3.2 Effect of Tree Height (h)

Setup: The privacy budget ϵ was fixed at four values (0.5, 0.75, 1, and 1.25), and the tree height h was varied from 3 to 15 (where h corresponds to the maximum possible height based on the trip length distribution in the original data). The average relative error was computed for the same 40,000 count queries defined in Chapter 5.

Results: Figures 6.5 and 6.6 show the changes in the *average relative error* at different noisy prefix

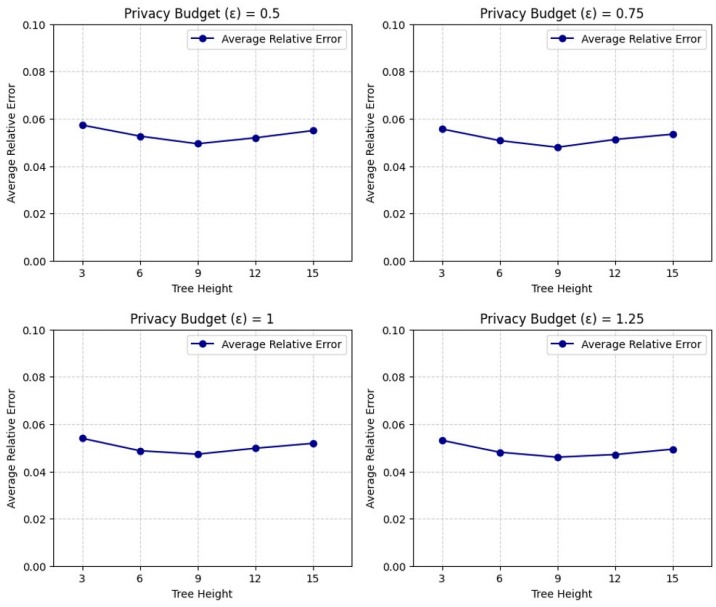

Figure 6.5: Average relative error vs. prefix tree height in NPT method (Chen et al., 2011).

tree heights (h).

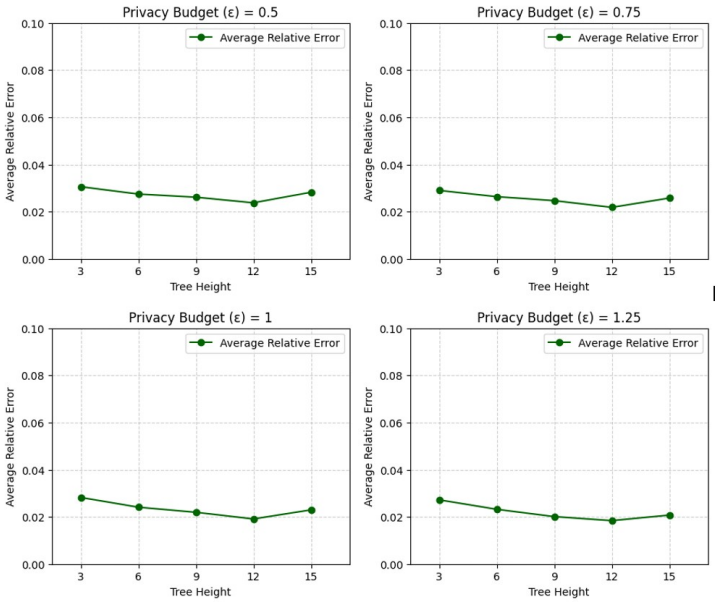

Figure 6.6: Average relative error vs. prefix tree height in PPDP method (Li et al., 2020).

NPT Method (Uniform Budget Allocation)

In the NPT method, increasing the tree height does not lead to a monotonically decreasing error.

- Initially, increasing h reduces the error because a higher tree retains more information from the original dataset.
- However, after a certain threshold (around $h = 9$ in this case), the error rises again. This happens because the *privacy budget* is equally divided among all levels of the tree ($\varepsilon_{\text{level}} = \frac{\varepsilon}{h}$). As h grows, each level receives a smaller share of the budget, resulting in larger added noise. Consequently, although increasing the height initially preserves more information, beyond a certain point the significant decrease in $\varepsilon_{\text{level}}$ and the increase in noise cause the relative error to rise again.
- Moreover, increasing h does not necessarily mean that the private tree will actually have that depth, because reaching depth h requires enough trajectories. In such cases, the original tree may still have deeper nodes and a broader information range, but the privatized tree either lacks those nodes or contains nodes with high noise, leading to higher errors.

PPDP Method (Incremental Privacy Budget Allocation)

In the PPDP method, the decreasing trend in error continues for a longer range of h, mainly due to two reasons:

1. **Incremental privacy budget allocation:** Unlike the 2011 method, where $\varepsilon_i = \varepsilon_{i+1}$, here the budget for each level is calculated based on the depth and statistical characteristics of that level, and then adjusted using a linear model ($\lambda_l = k \times l + b, \quad (k > 0, b > 0)$) In this approach:
    - As the depth increases, the budget also increases.
    - The *threshold* for each level decreases with depth, so that lower levels (with smaller counts) get less noise and retain more accurate data.
    - Higher levels with large counts are given less budget (more noise), while lower levels with small counts are given more budget (less noise) to balance the noise impact.
2. **Preservation of more details at greater depths:** Higher tree heights preserve more trajectory details. Moreover, increasing the query length ($|q|$) reduces the hit rate, which in turn lowers the error value.

However, even in the PPDP method, after a certain point (around $h = 15$ here), the error rises again. This is because the actual trajectory lengths are at most 15, but the privatized tree is usually pruned to depth 12. As a result, deeper queries on the original data return more results, while in the privatized tree, either no nodes exist at those depths or the nodes have high noise.

Overall Comparison and Qualitative Insight

In the NPT method, increasing h from 3 to 9 reduces the error because more detailed prefixes are retained, but beyond this point the error begins to rise as the per-level privacy budget becomes too small and noise dominates. In contrast, in the PPDP method this decreasing trend continues up to around $h = 12$, since its incremental budget allocation prioritizes deeper levels and preserves more details before noise starts to dominate.

This comparison highlights the fundamental trade-off between tree depth and noise: choosing h that is too small discards useful information, while choosing h that is too large spreads the privacy budget too thinly and reduces accuracy. These observations help practitioners select an h that balances privacy and utility for a given dataset.

6.4 Frequent Sequential Pattern Mining

Introduction

To further evaluate the analytical value of privatized trajectory data, we apply a concrete data mining task: *Frequent Sequential Pattern Mining (FSPM)*. For this, we use the *PrefixSpan* algorithm [78].

This task is of key importance for *BC Transit*, since identifying common travel sequences (e.g., "home $\rightarrow$ transfer station $\rightarrow$ workplace") not only helps improve bus scheduling but also enables optimization of the entire public transportation network, demand management, and planning of future infrastructure projects.

Unlike point-based data (e.g., the number of boardings at a single station), sequential data allows us to analyze:

- whether a station is a final destination or simply a transfer hub,
- which routes contribute most to congestion (e.g., $B \rightarrow A \rightarrow C$ compared to $D \rightarrow A \rightarrow C$),
- how transportation policies (such as adding a new BRT line or changing schedules) alter passenger travel patterns,
- how machine learning algorithms (e.g., Markov chains or deep learning models) can be used to predict future demand.

In other words, point-based data only shows that a station is crowded, while sequential data explains *why* it is crowded and which routes contribute to that congestion. Without sequential data, analyses remain incomplete and accurate prediction or smart optimization of the transit system becomes infeasible.

Parameters and Methods. In this experiment, the following additional parameters and methods were applied:

- **Number of frequent patterns:** $k \in \{20, 30, 40, 50\}$.
- **Extraction method:** PrefixSpan algorithm.

Handling Sparsity in PPDP. In PPDP, incorporating time information greatly expands the trajectory space, resulting in significant sparsity in the privatized tree. In our dataset, the spatio-temporal pattern space is especially sparse, and due to the limited data size, the probability of

observing multiple identical $(\text{location}, \text{time})$ sequences among the Top-k patterns is very low. To ensure that this analysis remains both **practical** and **fair**, we considered two possible strategies: time-binning and temporal projection. In our evaluation, we adopted the **Projection for Fair Spatial Comparison** approach.

1. **Time Binning (PPDP-ST-Binned).** Time values were merged into a few meaningful intervals (e.g., morning peak, mid-day, evening peak, night). This reduces sparsity while retaining the overall temporal structure, allowing PrefixSpan to mine frequent spatio-temporal patterns.

2. **Temporal Projection for Fair Spatial Comparison (PPDP-Projected).** To enable a fair comparison between PPDP and NPT, we also report a *time-projected* version of PPDP. In this method, the privatized PPDP output is post-processed by mapping each $(\text{location}, \text{timebin})$ to its corresponding location only, and aggregating noisy counts. This is purely a *post-processing* step and therefore incurs **no additional privacy cost**.

   Specifically:

   $$(A, t_1) \to (B, t_2) \to (C, t_3) \;\mapsto\; A \to B \to C$$

   PrefixSpan is then executed on these purely spatial sequences, producing Top-k spatial patterns that can be directly compared with NPT (which is inherently spatial).

Evaluation Protocol. Top-k results were reported under two configurations:

- NPT (spatial only),
- PPDP-Projected (spatial, with time removed via post-processing).

6.4.1 Effect of Number of Frequent Patterns (k)

Table 6.1 shows how utility changes under different values of k while fixing $\epsilon = 1.25$ and $h = 12$.

- At smaller values of k, both NPT and PPDP achieve relatively high accuracy, with PPDP consistently outperforming NPT.
- As k increases, accuracy (the ratio of correctly identified patterns to k) decreases.
- At $k = 20$, both methods achieve high accuracy, and even at $k = 50$, accuracy remains acceptable.
- For example at $k = 40$, NPT achieves 80% accuracy, while PPDP is slightly higher, demonstrating its stronger performance.

Table 6.1: Utility of FSPM vs. k ($\epsilon = 1.25$, $h = 9$).

k	NPT - True Positives	NPT - False Positives	NPT Acc. (%)	PPDP - True Positives	PPDP - False Positives	PPDP Acc. (%)
20	17	3	85.0	19	1	95.0
30	25	5	83.33	26	4	86.66
40	32	8	80.0	34	6	85.0
50	36	14	72.0	36	14	72.0

6.4.2 Effect of Privacy Budget (ϵ)

Table 6.2 shows the utility of frequent sequential pattern mining under different privacy budgets, with $h = 9$ and $k = 40$ fixed.

In general:

- Larger privacy budgets result in more true positives and fewer false positives.
- This matches theoretical expectations: a larger privacy budget means less noise, and therefore more accurate results.
- Since most frequent patterns are short (typically length ≤ 3), they have high support in the underlying dataset. Thus, utility is relatively insensitive to privacy budget variation and remains high even at smaller budgets.
- For example, when $\epsilon = 0.5$, NPT achieves 75% accuracy while PPDP reaches 80%.

Table 6.2: Utility of FSPM vs. ϵ ($h = 9$, $k = 40$).

ϵ	NPT - True Positives	NPT - False Positives	NPT Acc. (%)	PPDP - True Positives	PPDP - False Positives	PPDP Acc. (%)
0.5	30	10	75.0	32	8	80.0
0.75	31	9	77.5	32	8	80.0
1.0	31	9	77.5	33	7	82.5
1.25	32	8	80.0	34	6	85.0
1.5	33	7	82.5	34	6	85.0

6.5 Operational Queries for BC Transit

Beyond the conventional count queries common in related studies [27, 28], we also evaluate algorithm performance based on practical, real-world queries identified as operational needs by *BC Transit* (see Ch. 5).

This combination of **quantitative evaluation** (using statistical metrics) and **qualitative evaluation** (using operational queries) allows us to assess not only the numerical accuracy of the algorithms but also the analytical value of their results for operational decision-making.

Query 1 — Total tap-ins (overall ridership)

This query is a fully aggregated statistic (the sum of all tap-ins). As seen in the Figures 6.7, for NPT and PPDP at max $|Q| = 9$, partial cancellation of positive/negative noise across nodes and tree-consistency post-processing keep the *relative error* for this query much smaller than the average over random count queries. In random count queries, each query stands alone (no cross-cancellation), whereas in an aggregated total, increases in some nodes and decreases in others partially offset.

Baseline for comparison. With max $|Q| = 9$, the average ARE over 40,000 random count queries decreases from **4.94%** to **4.15%** for NPT and from **2.61%** to **1.66%** for PPDP as ε increases from 0.5 to 2.0; these are conservative, stress-test figures for sparse queries. For both methods, the relative error of the *total tap-ins* is **below 1%** and decreases with larger ε, consistent with the highly aggregated nature of this query.

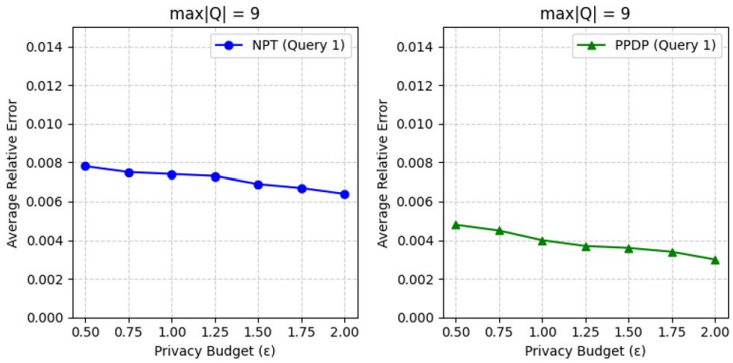

Figure 6.7: Average relative error (ARE) vs. privacy budget ϵ for the NPT method and the PPDP method, evaluated on Query 1.

Query 2 — Tap-ins per stop

As shown in Figure 6.8, this query measures the number of tap-ins at each stop separately. Unlike Query 1, where positive and negative noises across different nodes partially cancel out and reduce the overall error, here the results are more similar to count queries of length one. Consequently, the relative error does not diminish through aggregation.

As illustrated in the figure, the average relative error (ARE) under max $|Q| = 9$ decreases gradually as ϵ increases. In the NPT method, the ARE ranges from 0.0371 at $\epsilon = 0.5$ to 0.0311 at $\epsilon = 2.0$. In contrast, the PPDP method achieves a lower error, ranging from 0.0209 at $\epsilon = 0.5$ to 0.0125 at $\epsilon = 2.0$.

Compared to the large set of random count queries (40,000 queries across multiple lengths), Query2 yields significantly lower error because it is limited to queries of length one.

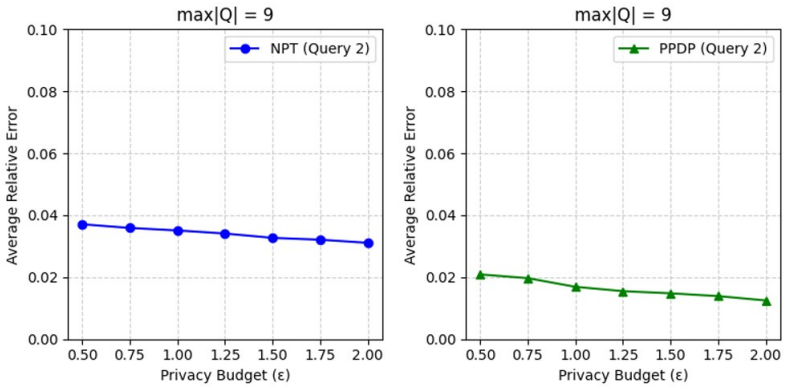

Figure 6.8: Average relative error (ARE) vs. privacy budget ϵ for the NPT method and the PPDP method, evaluated on Query 2.

Query 3 — Tap-ins by time interval

This query was *not* executable under NPT (Rui Chen et al.) because that method does not model the temporal dimension; we therefore evaluate it only with PPDP (Yang Li et al., 2020). As shown in Fig. 6.9, counts are aggregated across all stops *within each time bin*, so positive/negative noise partially cancels *within* a bin. Consequently, errors are consistently lower than for sparse random count queries, but typically higher than the fully aggregated total (Q1), since cancellation does not occur *across* bins. Errors decrease as ε increases and with coarser time binning; off-peak bins with smaller support exhibit slightly larger relative error.

Query 4 — Most frequent stops

Table 6.3 shows the utility of Query 4 (Most Frequent Stops) under different values of k with $\epsilon = 1.25$ and max $|Q| = 9$. At smaller values of k, both NPT and PPDP achieve high accuracy, with PPDP consistently outperforming NPT. For instance, when $k = 20$, NPT reaches an accuracy of 85.0%, while PPDP achieves 95.0%. As k increases, the accuracy gradually decreases for both methods due to the inclusion of more patterns with lower support. At $k = 40$, NPT attains 80.0% accuracy compared to 85.0% for PPDP. When $k = 50$, accuracy drops to 74.0% for NPT and 76.0% for PPDP, though both remain acceptable. Overall, PPDP demonstrates stronger performance across all values of k, particularly at smaller k, where it provides more reliable identification of the most frequent stops.

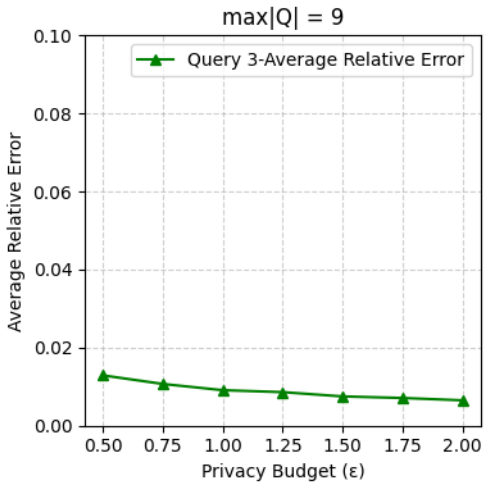

Figure 6.9: Query 3—Tap-ins by time interval (PPDP, max $|Q| = 9$): average relative error (ARE) vs. privacy budget ϵ.

Analysis of Query 4 (Most Frequent Stops)

Table 6.3 presents the utility of Query 4 under different values of k with $\epsilon = 1.25$ and $h = 9$. The results show that at smaller values of k, both NPT and PPDP achieve relatively high accuracy, with PPDP consistently outperforming NPT. As k increases, the accuracy of both methods gradually decreases. For example, when $k = 40$, NPT correctly identifies 32 out of 40 frequent stops (80.0%), while PPDP achieves 34 out of 40 (85.0%). At larger k values, such as $k = 50$, the accuracy of both methods drops further (74.0% for NPT and 76.0% for PPDP), though still remains within an acceptable range. Overall, PPDP demonstrates more stable and stronger performance across different values of k.

This comparison is very similar to the results of **Frequent Sequential Pattern Mining (FSPM)**, with the difference that in FSPM the focus is on frequent *trajectories* (paths), whereas Query 4 examines frequent *stops* within the trajectories. These frequent stops can be single stops or short subsequences. In practice, depending on the data size, the most common trajectories usually have a length of less than three or four. Thus, Query 4 is dominated by single stops, while FSPM includes both single and multi-stop subsequences.

When comparing Table 6.2 and Table 6.3, we observe that the results for the top 30–40 patterns are nearly identical. This is likely because the majority of the top-40 frequent patterns are single-stop subsequences. However, at $k = 50$, Query 4 yields slightly lower error than FSPM. This can be explained by the fact that FSPM includes longer subsequences among the top-50, and because algorithms like PrefixSpan examine all subsequences, errors in counting a single stop (e.g., A1) can propagate to multiple patterns (e.g., both A1 and A1→A4), amplifying the overall error.

Table 6.3: Utility of Query 4 (Most Frequent Stops) vs. k ($\epsilon = 1.25$, max $|Q| = 9$).

k	NPT - True Positives	NPT - False Positives	NPT Acc. (%)	PPDP - True Positives	PPDP - False Positives	PPDP Acc. (%)
30	25	5	83.33	26	4	86.66
40	32	8	80.0	34	6	85.0
50	37	13	74.0	38	12	76.0

6.6 Discussion and Answers to the Research Questions

RQ1: Is it possible to publish accurate but privacy-preserving analyses of BC Transit data? What is the impact of privacy mechanisms on accuracy?

Yes—BC Transit analyses can be released under differential privacy while retaining *acceptable accuracy*.

Mechanism comparison: PPDP consistently yields lower error than NPT; the improved NPT with weighted post-processing outperforms the vanilla NPT.

Across query types. All count queries, operational queries, and frequent sequential pattern mining achieve acceptable accuracy under both algorithms.

Across the tested privacy budgets ($\epsilon \in [0.5, 2.0]$) and multiple tree heights, both prefix-tree approaches retained acceptable accuracy for *aggregate count* queries, with PPDP consistently outperforming NPT. Even in the strictest setting ($\epsilon = 0.5$, max $|Q| = 3$), ARE remained below 6% for NPT and below 3% for PPDP (Figs. 6.1, 6.2). The weighted post-processing improved the NPT baseline by reducing parent–child inconsistencies (Fig. 6.3). For *sequential* analyses, short patterns remained accurate; e.g., Tables 6.1 and 6.2 show high top-k recovery (e.g., $197/250 \approx 78.8\%$ for NPT, with PPDP slightly higher, and 75–80% accuracy at $\epsilon = 0.5$). Overall, higher ϵ improves utility; very shallow or very deep trees can hurt it (Figs. 6.5, 6.6); and overly fine temporal/spatial granularity increases error. **Verdict:** With moderate aggregation and careful parameterization, DP releases can provide decision-useful accuracy.

RQ2: Effect of Privacy Mechanisms on Sum Queries vs. Count Queries and FSPM.

Our results indicate that *count queries* and *frequent sequential pattern mining (FSPM)* typically exhibit a higher average relative error (ARE) than sum queries/totals. The main reason is that count queries and FSPM are more tightly coupled to the fine-grained structure of the data; in these workloads, differential-privacy noise is not easily canceled or absorbed. By contrast, many some queries are highly *aggregated* (e.g., total tap-ins, or time-binned totals), so their results are often large. The effect of noise is thus smaller, yielding noticeably smaller final errors.

Conceptually, Q4 (most frequent stops) and FSPM share a ranking/pattern flavor, but they differ in scope: FSPM reasons about *sequences* (single- or multi-step paths), whereas Q4 measures the *marginal* frequency of each stop. In FSPM (implemented with *PrefixSpan*), all *subsequences* are also considered: if a frequent pattern $A1 \rightarrow A4$ appears, then $A1$ and $A4$ are implicitly frequent on their own. Consequently, perturbing the count of a single node can affect multiple dependent patterns, amplifying error beyond a single item. As k grows or sequences become longer, FSPM behaves more stringently and its accuracy degrades compared to aggregated sum queries.

Summary. Count queries and especially FSPM (for larger k or longer sequences) are more sensitive to DP noise, while aggregated sum queries generally attain lower error. Nevertheless, in all cases the error remains within a reasonable range for operational analysis. In most scenarios, structure-aware mechanisms (e.g., PPDP) are more stable and accurate than simpler approaches (e.g., NPT).

RQ3: What are recommendations for BC Transit for future publication of privacy-preserving data analyses?

This study has demonstrated that it is feasible to publish transit data under strong Differential Privacy (DP) guarantees while maintaining both acceptable accuracy and computational efficiency, using algorithms such as the **Noisy Prefix Tree (NPT)** and **PPDP**. Based on the findings, the following recommendations can be made for BC Transit:

1. **Publish noised raw data with DP guarantees (when possible).** If legal requirements and privacy concerns allow, the best-case scenario is to publish sanitized raw data protected with DP. This provides the highest level of transparency and the greatest analytical value for researchers and planners.

2. **Release privacy-preserving statistics/aggregate results when raw data cannot be published.** In cases where publishing raw data is not feasible due to privacy or policy restrictions, releasing statistical summaries and targeted outputs (e.g., most/least frequent stops, short travel patterns, or temporal demand distributions) can still be highly valuable. Such outputs provide direct answers to operational questions without exposing raw records.

3. **Adopt prefix-tree based algorithms.** As shown in this work, NPT and PPDP, along with improved versions (e.g., weighted post-processing and binomial pruning), are well-suited for sequential transit data and can balance privacy protection with analytical utility. This thesis provides a guideline for tuning the parameters of these algorithms to achieve superior privacy/utility tradeoffs.

4. **Ensure transparent privacy budget management.** Each data release should clearly document the value of ϵ and how it is allocated, allowing results to be compared over time and strengthening stakeholder trust.

5. **Establish a sustainable framework for periodic data publication.** As future work, BC Transit could develop a framework for the periodic release of privacy-preserving data and analyses. Such a framework could include multiple levels of access: statistical and analytical outputs for the public, and richer or more detailed datasets for researchers in controlled environments.

In summary, whether through the publication of raw data with DP guarantees, or—if that is not possible—through the release of statistical and analytical outputs, BC Transit can establish itself as a model for responsible and privacy-preserving data sharing in the public transit sector.

Chapter 7

Conclusion

This thesis addresses the problem of publishing bus passenger trajectory data while maintaining high utility under strong Differential Privacy (DP) guarantees. In collaboration with BC Transit, de-identified *tap-in-only* smart card records from the Victoria region were obtained and analyzed. Due to the inherent characteristics of bus transit data—where, after noise injection and algorithm execution, the output must correspond to an actual bus stop in the network and it is not permissible to produce arbitrary coordinates or non-existent stops—many common trajectory data algorithms lose their applicability in this context. This specific constraint limited the choice of methods and required precise adaptation to the rules and structure of bus data.

To enable the practical implementation and realistic evaluation of these algorithms, part of the work was devoted to reconstructing synthetic trajectory data based on real tap-in records. The creation of these reconstructed datasets, which required substantial time and attention to detail, allowed for the testing of mechanisms and conducting comparisons without disclosing actual passenger data.

Subsequently, a comprehensive review of existing DP mechanisms was carried out, and two algorithms—*Noisy Prefix Tree (NPT)* and *privacy-preserving data publishing (PPDP)*—were selected for their greater compatibility with the data. During the implementation process, several limitations and challenges were identified and were largely addressed through technical enhancements, including a *weighted post-processing method* to improve accuracy and the use of a *binomial method for checking empty nodes* to enhance efficiency.

The privatized datasets were evaluated using basic count queries, sum queries, and frequent sequential pattern mining to assess the balance between privacy preservation and utility.

Research Contributions

The key contributions of this thesis are as follows:

1. **First DP-based study on BC Transit passenger data** – Conducted in collaboration with BC Transit, this work provides a practical framework for the privacy-preserving release of public transportation data. The framework directly addresses an operational need and can help similar organizations securely share data for research and service improvement without disclosing sensitive passenger information.

2. **Synthetic trajectory generation from tap-in-only data** – Developed a role-based behavioral model capable of reconstructing realistic weekly travel patterns from limited tap-in-only data. The generated synthetic trajectories enable algorithm evaluation without exposing real passenger trips.

3. **Technical improvements to prefix tree algorithms** – Designed and implemented two key enhancements:

    - A *weighted post-processing method* to improve the accuracy of privatized data.
    - *Spatio-temporal pruning* and a *binomial test for empty nodes* to reduce runtime and improve computational efficiency.

4. **Comprehensive evaluation under real-world conditions** – Performed experiments both simple and complex queries to assess the privacy–utility trade-off for two state-of-the-art algorithms (NPT and PPDP) in realistic public transportation analysis scenarios.

Key Findings

The main findings of this research can be summarized as follows:

1. **Feasibility of differentially private data release for real transit datasets** – The study demonstrates that it is possible to publish useful public transportation data from the BC Transit system while preserving strong privacy guarantees, even under the operational constraints of real-world datasets.

2. **Realistic reconstruction from limited tap-in-only data** – By introducing a role-based behavioral model, the research successfully reconstructed plausible weekly passenger travel patterns, enabling the generation of high-utility synthetic datasets for analysis.

3. **Enhanced prefix tree algorithms** – The integration of a weighted post-processing strategy, spatio-temporal pruning, and a binomial test for empty nodes significantly improved accuracy and computational efficiency compared to the original implementations.

4. **Balanced privacy–utility trade-off** – Through extensive experiments using count queries, *Frequent Sequential Pattern Mining* tasks, and complex analytical queries, the evaluation confirmed that the proposed improvements achieve a better balance between privacy preservation and analytical usefulness for two state-of-the-art methods (NPT and PPDP).

Conclusion and Future Work

Future research can be expanded in several directions:

Personalized Differential Privacy for Trajectory Data

Although this area has been previously explored in the literature, existing methods often rely on generalization and do not require the output to map back to an actual stop in the transit network. As such, these methods cannot be directly applied to bus transit data but could be adapted for this purpose in the future. Two primary models in this domain include:

- **Location-Sensitivity Model:** The trajectory database contains only spatial–temporal features, and each location has a different sensitivity level. The privacy budget for each moving object is allocated according to the sensitivity of the locations it passes through.
- **Attribute-Sensitivity Model:** In addition to spatial–temporal features, the trajectory database contains other sensitive attributes. In this case, each moving object has a privacy requirement defined by the data owner, which can be determined based on any chosen criteria.

In smart card (tap-in) transit data, such as that used in this thesis, personalized sensitivity could be implemented by assigning different weights to specific stops or routes. For example, stops near hospitals or political institutions could be assigned higher privacy levels. Furthermore, passengers could be given the option to specify their desired privacy level at the time of card purchase or registration, which would then be taken into account during data processing.

Aggregate Delay Analysis

Using bus route scheduling and real-time tracking data, an aggregate model could be developed to estimate the *total passenger delay load* at the station level. This model would not only record the delay time for each service but also weight it by the number of passengers waiting during each time interval. Implementing such an approach within a differential privacy framework presents challenges such as accurately estimating the number of waiting passengers (without revealing their identities), managing large-scale spatial–temporal data, and optimally allocating the privacy budget to delay-related variables. Such analysis could enable transit operators to optimize routes and schedules while preserving passenger privacy.

Conclusion

This research aimed to develop and evaluate a practical framework for the publication of bus passenger trajectory data under strong differential privacy guarantees. By combining industry collaboration, a thorough analysis of the characteristics of bus transit data, and the adaptation of existing algorithms to the specific constraints of such data, the study achieved results that can benefit both the academic community and public transit operators.

Although certain limitations remain, the findings offer pathways for future developments. It demonstrates that releasing privacy-preserving real-world bus data is feasible, while showing practical technical improvements to achieve better privacy-utility tradeoffs.

References

[1] Fatima Zahra Errounda and Yan Liu. An analysis of differential privacy research in location data. In *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, pages 53–60. IEEE, 2019.

[2] Xiangjie Kong, Menglin Li, Kai Ma, Kaiqi Tian, Mengyuan Wang, Zhaolong Ning, and Feng Xia. Big trajectory data: A survey of applications and services. *IEEE access*, 6:58295–58306, 2018.

[3] Àlex Miranda-Pascual, Patricia Guerra-Balboa, Javier Parra-Arnau, Jordi Forné, and Thorsten Strufe. Sok: Differentially private publication of trajectory data. *Proceedings on Privacy Enhancing Technologies*, 2023.

[4] Andrew J Blumberg and Peter Eckersley. On locational privacy, and how to avoid losing it forever. *Electronic frontier foundation*, 10(11):1–7, 2009.

[5] Shira Ovide. Just collect less data, period. *New York Times*, 2020.

[6] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[7] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3(1):1–5, 2013.

[8] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[9] Erik Buchholz, Alsharif Abuadbba, Shuo Wang, Surya Nepal, and Salil Subhash Kanhere. Reconstruction attack on differential private trajectory protection mechanisms. In *Proceedings of the 38th Annual Computer Security Applications Conference*, pages 279–292, 2022.

[10] Yuqing Yang, Jianghui Cai, Haifeng Yang, Jifu Zhang, and Xujun Zhao. Tad: A trajectory clustering algorithm based on spatial-temporal density analysis. *Expert Systems with Applications*, 139:112846, 2020.

[11] BC Transit. Customer Satisfaction Tracking Research Annual Report 2024–25. `https://www.bctransit.com/corporate-reports/customer-tracking-surveys/`, 2024. Available at `https://www.bctransit.com/corporate-reports/customer-tracking-surveys/`.

[12] Rob Kitchin and Martin Dodge. Moving violations: Data privacy in public transit. `https://www.maynoothuniversity.ie/sites/default/files/assets/document/NIRSA%20Working%20Paper%2062.pdf`, 2011. National Institute for Regional and Spatial Analysis (NIRSA), National University of Ireland Maynooth.

[13] Brian D. Taylor and Lisa A. Schweitzer. Assessing the experience of women in transit: Evidence from los angeles. *Transportation Research Record*, 1927(1):145–153, 2005.

[14] Roman Yarovoy, Francesco Bonchi, Laks VS Lakshmanan, and Wendy Hui Wang. Anonymizing moving objects: How to hide a mob in a crowd? In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 72–83, 2009.

[15] Anna Monreale, Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Mirco Nanni, Dino Pedreschi, Salvatore Rinzivillo, and Stefan Wrobel. Movement data anonymity through generalization. *Transactions on Data Privacy*, 3(2):91–121, 2010.

[16] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. A classification of location privacy attacks and approaches. *Personal and ubiquitous computing*, 18:163–175, 2014.

[17] Manolis Terrovitis and Nikos Mamoulis. Privacy preservation in the publication of trajectories. In *The Ninth international conference on mobile data management (mdm 2008)*, pages 65–72. IEEE, 2008.

[18] Pierangela Samarati. Protecting respondents identities in microdata release. *IEEE transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2002.

[19] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, 2003.

[20] David E Bakken, R Rarameswaran, Douglas M Blough, Andy A Franz, and Ty J Palmer. Data obfuscation: Anonymity and desensitization of usable data sets. *IEEE Security & Privacy*, 2(6):34–41, 2004.

[21] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proceedings 38th annual symposium on foundations of computer science*, pages 364–373. IEEE, 1997.

[22] John Krumm. Inference attacks on location tracks. In *Pervasive Computing: 5th International Conference, PERVASIVE 2007, Toronto, Canada, May 13-16, 2007. Proceedings 5*, pages 127–143. Springer, 2007.

[23] Reza Shokri, George Theodorakopoulos, Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec. Protecting location privacy: optimal strategy against localization attacks. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 617–627, 2012.

[24] Mohammad Saiful Islam, Mehmet Kuzu, and Murat Kantarcioglu. Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *Ndss*, volume 20, page 12. Citeseer, 2012.

[25] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and trends® in theoretical computer science*, 9(3–4):211–407, 2014.

[26] Erik Buchholz, Alsharif Abuadbba, Shuo Wang, Surya Nepal, and Salil S Kanhere. Sok: Can trajectory generation combine privacy and utility? *arXiv preprint arXiv:2403.07218*, 2024.

[27] Chen Rui, C Benjamin, M Fung, and Bipin C Desai. Differentially private trajectory data publication. *CoRR, abs/1112.2020*, 2011.

[28] Yang Li, Dasen Yang, and Xianbiao Hu. A differential privacy-based privacy-preserving data publishing algorithm for transit smart card data. *Transportation Research Part C: Emerging Technologies*, 115:102634, 2020.

[29] Marco Fiore, Panagiota Katsikouli, Elli Zavou, Mathieu Cunche, Françoise Fessant, Dominique Le Hello, Ulrich Matchi Aivodji, Baptiste Olivier, Tony Quertier, and Razvan Stanica. Privacy in trajectory micro-data publishing: a survey. *Transactions on Data Privacy*, 13(2):91–149, 2020.

[30] Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer, 2006.

[31] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially-private histograms through consistency. *arXiv preprint arXiv:0904.0942*, 2009.

[32] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2):1–25, 2013.

[33] Rui Chen, Gergely Acs, and Claude Castelluccia. Differentially private sequential data publication via variable-length n-grams. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 638–649, 2012.

[34] George Kellaris, Spiros Papadopoulos, Xiaokui Xiao, and Dimitris Papadias. Differentially private event sequences over infinite streams. *Proceedings of the VLDB Endowment*, 7(12):1155–1166, August 2014.

[35] Daniele Riboni and Claudio Bettini. Incremental release of differentially-private check-in data. *Pervasive and Mobile Computing*, 16:220–238, January 2015.

[36] Naoise Holohan, Douglas J Leith, and Oliver Mason. Differential privacy in metric spaces: Numerical, categorical and functional data under the one roof. *Information Sciences*, 305:256–268, June 2015.

[37] Ping Xiong, Tianqing Zhu, Wenqi Niu, and Guangyan Li. A differentially private algorithm for location data release. *Knowledge and Information Systems*, 47(3):647–669, June 2016.

[38] Jun Zhang, Xiaokui Xiao, and Xing Xie. Privtree: A differentially private algorithm for hierarchical decompositions. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD)*, pages 155–170, New York, NY, USA, June 2016. ACM.

[39] Yu Sei and Akihiko Ohsuga. Differential private data collection and analysis based on randomized multiple dummies for untrusted mobile crowdsensing. *IEEE Transactions on Information Forensics and Security*, 12(4):926–939, April 2017.

[40] Scott Fletcher and Md Zahidul Islam. Differentially private random decision forests using smooth sensitivity. *Expert Systems with Applications*, 78:16–31, July 2017.

[41] Chenxu Xu, Jing Ren, Yifan Zhang, Zhen Qin, and Kui Ren. Dppro: Differentially private high-dimensional data release via random projection. *IEEE Transactions on Information Forensics and Security*, 12(12):3081–3093, December 2017.

[42] Dan Feldman, Cheng Xiang, Runmin Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 3–15, New York, NY, USA, 2017. ACM.

[43] Yibo Wang, Liang Yang, Xiaohui Chen, Xiaoqing Zhang, and Zhihong He. Enhancing social network privacy with accumulated non-zero prior knowledge. *Information Sciences*, 445–446:6–21, June 2018.

[44] Graham Cormode, Somesh Jha, Tejas Kulkarni, Ninghui Li, Divesh Srivastava, and Ting Wang. Privacy at scale: Local differential privacy in practice. In *Proceedings of the 2018 International Conference on Management of Data (SIGMOD)*, pages 1655–1658, New York, NY, USA, June 2018. ACM.

[45] Meng Wang, Cheng Xu, Xiaohui Chen, Hong Hao, Lei Zhong, and Shui Yu. Differential privacy oriented distributed online learning for mobile social video prefetching. *IEEE Transactions on Multimedia*, 21(3):636–651, March 2019.

[46] Jochen Steil, Ivan Hagestedt, Mx Huang, and Andreas Bulling. Privacy-aware eye tracking using differential privacy. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. ACM, 2019.

[47] Khalil Al-Hussaeni, Benjamin CM Fung, Farkhund Iqbal, Gaby G Dagher, and Eun G Park. Safepath: Differentially-private publishing of passenger trajectories in transportation systems. *Computer Networks*, 143:126–139, 2018.

[48] Mehmet Ercan Gursoy, Ling Liu, Stacey Truex, and Lei Yu. Differentially private and utility preserving publication of trajectory data. *IEEE Transactions on Mobile Computing*, October 2018.

[49] Mehmet Ercan Gursoy, Ling Liu, Stacey Truex, Lei Yu, and Wenqi Wei. Utility-aware synthesis of differentially private and attack-resilient location traces. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 196–211, New York, NY, USA, 2018. ACM.

[50] Donghui Shao, Kui Jiang, Thomas Kister, Stéphane Bressan, and Kian-Lee Tan. Publishing trajectory with differential privacy: A priori vs. a posteriori sampling mechanisms. In H. Decker, L. Lhotská, S. Link, J. Basl, and A. M. Tjoa, editors, *Database and Expert Systems Applications*, volume 8055 of *Lecture Notes in Computer Science*, pages 357–365, Berlin, Heidelberg, 2013. Springer.

[51] Kui Jiang, Donghui Shao, Stéphane Bressan, Thomas Kister, and Kian-Lee Tan. Publishing trajectories with differential privacy guarantees. In *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, pages 12:1–12:12, New York, NY, USA, 2013. ACM.

[52] Rui Chen, Benjamin C. M. Fung, Bipin C. Desai, and Nicolas M. Sossou. Differentially private transit data publication: A case study on the montreal transportation system. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 213–221, New York, NY, USA, 2012. ACM.

[53] Shou-Shii Ho and Shuo Ruan. Preserving privacy for interesting location pattern mining from trajectory data. *Transactions on Data Privacy*, 6(1):87–106, 2013.

[54] Luca Bonomi and Li Xiong. A two-phase algorithm for mining sequential patterns with differential privacy. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 269–278, New York, NY, USA, 2013. ACM.

[55] Xin He, Graham Cormode, Ashwin Machanavajjhala, Cecilia M. Procopiuc, and Divesh Srivastava. Dpt: Differentially private trajectory synthesis using hierarchical reference systems. *Proceedings of the VLDB Endowment*, 8(11):1154–1165, July 2015.

[56] Ming Li, Lei Zhu, Zheli Zhang, and Rui Xu. Differentially private publication scheme for trajectory data. In *Proceedings of the First International Conference on Data Science in Cyberspace (DSC)*, pages 596–601, Changsha, China, 2016. IEEE.

[57] Shuang Wang and Richard O. Sinnott. Protecting personal trajectories of social media users through differential privacy. *Computers & Security*, 67:142–163, June 2017.

[58] Ming Li, Lei Zhu, Zheli Zhang, and Rui Xu. Achieving differential privacy of trajectory data publishing in participatory sensing. *Information Sciences*, 400:1–13, August 2017.

[59] Ling Ou, Zhen Qin, Shan Liao, Ying Hong, and Xiaohua Jia. Releasing correlated trajectories: Towards high utility and optimal differential privacy. *IEEE Transactions on Dependable and Secure Computing*, July 2018.

[60] Chen Xu, Lei Zhu, Yifeng Liu, Jihong Guan, and Shui Yu. Dp-ltod: Differential privacy latent trajectory community discovering services over location-based social networks. *IEEE Transactions on Services Computing*, July 2018.

[61] Shuang Wang, Richard Sinnott, and Surya Nepal. Privacy-protected statistics publication over social media user trajectory streams. *Future Generation Computer Systems*, 87:792–802, October 2018.

[62] Meng Li, Liehuang Zhu, Zijian Zhang, and Rixin Xu. Achieving differential privacy of trajectory data publishing in participatory sensing. *Information Sciences*, 400:1–13, 2017.

[63] Xiaodong Zhao, Dechang Pi, and Junfu Chen. Novel trajectory privacy-preserving method based on clustering using differential privacy. *Expert Systems with Applications*, 149:113241, 2020.

[64] Shuilian Yuan, Dechang Pi, Xiaodong Zhao, and Meng Xu. Differential privacy trajectory data protection scheme based on r-tree. *Expert Systems with Applications*, 182:115215, 2021.

[65] Xiaodong Zhao, Yulan Dong, and Dechang Pi. Novel trajectory data publishing method under differential privacy. *Expert Systems with Applications*, 138:112791, 2019.

[66] Jianzhe Zhao, Jie Mei, Stan Matwin, Yukai Su, and Yuancheng Yang. Risk-aware individual trajectory data publishing with differential privacy. *IEEE Access*, 9:7421–7438, 2020.

[67] Jingyu Hua, Yue Gao, and Sheng Zhong. Differentially private publication of general time-serial trajectory data. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 549–557. IEEE, 2015.

[68] Si Chen, Anmin Fu, Jian Shen, Shui Yu, Huaqun Wang, and Huaijiang Sun. Rnn-dp: A new differential privacy scheme base on recurrent neural network for dynamic trajectory privacy protection. *Journal of Network and Computer Applications*, 168:102736, 2020.

[69] Ruxue Wen, Wenqing Cheng, Haojun Huang, Wang Miao, and Chen Wang. Privacy preserving trajectory data publishing with personalized differential privacy. In *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 313–320. IEEE, 2020.

[70] Nana Wang and Mohan S Kankanhalli. Protecting sensitive place visits in privacy-preserving trajectory publishing. *Computers & Security*, 97:101949, 2020.

[71] Ehab I Diab, Madhav G Badami, and Ahmed M El-Geneidy. Bus transit service reliability and improvement strategies: Integrating the perspectives of passengers and transit agencies in north america. *Transport Reviews*, 35(3):292–328, 2015.

[72] Georgios Georgiadis, Ioannis Politis, and Panagiotis Papaioannou. Measuring and improving the efficiency and effectiveness of bus public transport systems. *Research in Transportation Economics*, 48:84–91, 2014.

[73] Nancy J Obermeyer. Moving violations: Data privacy in public transit. *Geographical Review*, 97(3):351–364, 2007.

[74] BC Transit. Customer satisfaction tracking research – annual report 2024–25. Technical report, BC Transit, March 2024. Prepared by NRG Research Group.

[75] Statistics Canada. Average weekly working hours, 1976 to 2022. Quality of employment in canada, catalogue no. 14-28-0001, Statistics Canada, June 2023. Release date: June 13, 2023.

[76] Graham Cormode, Magda Procopiuc, Divesh Srivastava, and Thanh TL Tran. Differentially private publication of sparse data. *arXiv preprint arXiv:1103.0825*, 2011.

[77] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. Universally utility-maximizing privacy mechanisms. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 351–360, 2009.

[78] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Meichun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224. IEEE Piscataway, NJ, USA, 2001.

Appendix A
BC Transit Data Request Process

This appendix provides a detailed account of the efforts undertaken to access real-world bus trajectory data for this research.

Initial Discovery and Request Efforts

Multiple governmental and data organizations were explored during the data discovery phase. These included the BC Data Catalogue, Passenger Transportation Board, BC Geographic Warehouse, TransLink, as well as internal data and governance portals such as the Data Systems & Services Client Hub.

Several data access requests were submitted via email, online request forms, and support portals. While some organizations responded, they either did not possess the required boarding-level data or cited restrictions related to privacy, jurisdiction, or internal limitations.

Establishing Collaboration with BC Transit

The data request was eventually routed to **BC Transit**. Initially, the organization had concerns about releasing even anonymized tap-in data due to the lack of precedent for such academic requests.

However, after several formal communications explaining the academic nature of the thesis, the use of the differential privacy framework, and the scope and ethical protections, BC Transit began to express interest in supporting the research. A multi-stage engagement followed, which included the submission of official data request documentation, attendance at a clarification and privacy-focused meeting, responding to inquiries regarding data use, ethics, and implementation, and revising the scope based on BC Transit's internal feedback.

Dataset Details and Restrictions

Eventually, BC Transit shared a de-identified dataset containing tap-in events recorded by automatic passenger counters (APCs). The dataset included Bus Stop IDs, timestamps of tap-in events,

Bus IDs, seated capacities, and Route IDs.

Restrictions: No personal identifiers were included. Complete sequential trajectories were not shared due to re-identification risks. Additionally, the data could not be reused or redistributed beyond the scope of this thesis.

Organizational Insight and Support

The BC Transit team—led by **Andrew Miller**, Manager of Enterprise Data & Analytics—provided meaningful support and insights throughout the process. He shared observations about transit ridership, including:

- The University of Victoria is the largest ridership driver (students/faculty)
- Public sector employees commuting to downtown Victoria are the second-largest group
- Ridership is bi-modal (AM and PM peaks)
- Tuesday through Friday are the busiest days
- September has the highest ridership; August the lowest

Andrew emphasized the potential value of synthetic datasets in future data-sharing efforts:

> "Being able to provide representative synthetic datasets is a capability we should work towards… There are huge opportunities for research and analysis of our data, and we have limited internal capacity to do things that aren't addressing immediate business processes."

Conclusion

This collaboration marked BC Transit's first effort to support an academic research focused on differential privacy and synthetic trajectory generation. The success of this partnership demonstrates the shared value in bridging academia and public institutions to create privacy-preserving, high-impact solutions in public transportation systems.

Appendix B
Synthetic Tap Data Assignment Workflow and Scoring Methodology

This appendix provides a detailed explanation of the synthetic data generation process.

Assignment Process and Scoring Logic

Pre-processing and Preparation of Raw Data

Before starting to generate synthetic trajectories, the raw tap-in data must go through a careful pre-processing and preparation step. This is essential to ensure the input data is clean, complete, and in a proper format for modeling.

The steps are:

- **Clean and remove incomplete records:** Any rows missing key information such as vehicle ID, bus number, or number of seats are identified and removed. This helps maintain data quality and consistency.
- **Repair route and stop information:** For records with missing or default values for route name or stop ID, the algorithm tries to fill these fields using complete information from other records for the same bus. This is done using a dedicated function for data recovery.
- **Final removal of invalid records:** After repair attempts, any row that still has invalid or empty route names or stop IDs is completely removed to avoid introducing errors in the next steps.
- **Add geographic coordinates:** To enable spatial analysis, the latitude and longitude of each stop are added to each tap-in record. This is done by merging the cleaned tap-in data with a separate station information source.

- **Convert timestamps and create helper features:** The timestamp column is converted to a standard date-time format. Any row where this conversion fails is deleted.

Main Logic of Tap Simulation and Assignment

1. **Process data day by day:** The cleaned tap-in data (`df_real_data`) is grouped by date. The algorithm proceeds day by day.

2. **Per-day shuffle:** For each day, tap-ins are shuffled randomly to avoid any bias from the original order in the file and to maintain a realistic distribution across the day.

   Why per-day shuffle and not overall shuffle? If the entire dataset is shuffled at once, it disrupts daily passenger behavior (e.g., records from day 1 mix with day 4), and daily limits per passenger would break. By shuffling within each day, we preserve daily structure while keeping diversity.

3. **Reset passenger status:** At the start of each new day, each passenger's daily tap counters (`daily_taps_count`) are reset. The algorithm also checks whether this day is an active day for the passenger (based on their weekly plan).

4. **Search for a suitable passenger (multi-step filters):** For each new tap-in:

   - **Review all active simulated passengers.**
   - **Weekly active day filter:** If the tap-in occurs on a day when a passenger is not scheduled to travel, they are excluded.
   - **Daily total taps limit filter:** If a passenger has already reached their maximum allowed taps for that day, they are excluded.
   - **Four main travel scenarios are checked for remaining candidates:**
     - **Continuation/Transfer:** Tap is close in time (between `min_immediate_consecutive_tap_min` and `max_immediate_consecutive_tap_minutes`) and in distance (`MAX_DISTANCE_KM_FOR_IM` to the passenger's last tap.
     - **Return trip:** Tap happens after a reasonable activity duration (based on `min/max_primary_acti` and within the allowed return distance (`MAX_DISTANCE_KM_FOR_PRIMARY_RETURN_TRIP`) from the last tap.
     - **New distinct trip:** Tap happens after a longer gap (`min_new_distinct_trip_minutes` and above), during an acceptable time window (`max_daily_activity_end_hour`), and within acceptable starting distance (`MAX_DISTANCE_KM_FOR_NEW_TRIP_START`).

  - **First tap of the day or ever:** If there is no previous tap or it is a new day for that passenger.
- **Determine trip purpose:** For each scenario, the trip purpose (e.g., work, school, shopping, social, personal, airport) is guessed using the `infer_trip_purpose` function.
- **Purpose-specific taps limit filter:** If the passenger has already reached the allowed number of taps for that purpose today (e.g., two "work" taps for a full-time worker), they are excluded.

Scoring Logic

Qualified passengers receive scores based on:

- Time gap from last tap
- Distance from last tap location
- Bonus score for proximity to primary location (e.g., work/school)

Lower total score = better match.

Best Passenger Selection

Choose the passenger with the lowest score.

Tap Assignment

- Assign tap to selected passenger
- Update their profile (location, time, purpose, counts)
- Save primary location if needed

Fallback: No Match Found

If no passenger qualifies:

- Create new passenger (assign ID, role using `ROLE_DISTRIBUTION`)
- Generate weekly active plan and limits
- Start their profile with this tap

Storing Results

Final outputs:

- `simulated_taps_with_passenger_profiles.csv`
- `simulated_trajectories_wide.csv` (one row per passenger)

Summary

- Closely follows real-world patterns (time and geography)
- Uses role-based behavior models for realistic synthetic data
- Builds trustworthy privacy-preserving transit data

Appendix C
Proof of Equivalence of Binomial Selection and Explicit Laplace Testing

This appendix provides a detailed mathematical proof demonstrating the equivalence between the Binomial selection method and explicit Laplace-based threshold testing.

Mathematical Proof of Equivalence of Binomial Selection and Explicit Laplace Testing

Let m be the number of empty candidate nodes at a given level of the prefix tree. For each empty node u_i, the Laplace mechanism with scale $b = 1/\bar{\epsilon}$ outputs:

$$c(u_i) = 0 + \text{Laplace}(b).$$

A node passes if $c(u_i) \geq \theta$. The probability of passing is:

$$p_\theta = \Pr[c(u_i) \geq \theta] = \int_\theta^\infty \frac{\bar{\epsilon}}{2} e^{-\bar{\epsilon}x} dx = \frac{e^{-\bar{\epsilon}\theta}}{2}.$$

Since the m tests are independent Bernoulli trials with success probability p_θ, the number k of passing nodes follows:

$$k \sim B(m, p_\theta).$$

Original Method

1. Perform m independent Laplace draws.
2. Keep node u_i if $c(u_i) \geq \theta$.

3. Result: exactly k successes, where k is distributed as $B(m, p_\theta)$.

Binomial Method

1. Draw $k \sim B(m, p_\theta)$ directly.
2. Uniformly select k nodes from the m candidates without replacement.
3. For each selected node, draw its noisy count from the *conditional* Laplace distribution:

$$f(x \mid x \geq \theta) = \frac{\frac{\bar{\epsilon}}{2} e^{-\bar{\epsilon}x}}{p_\theta} = \bar{\epsilon} e^{\bar{\epsilon}\theta - \bar{\epsilon}x}, \quad x \geq \theta.$$

Equivalence Proof

In the original method:

- The set of passing nodes is obtained as "all successes among m independent Bernoulli trials".
- This is equivalent to first drawing $k \sim B(m, p_\theta)$ and then selecting exactly k nodes uniformly.

Because the conditional noisy counts for passing nodes are identical in both methods, the joint distribution over **which nodes survive** and **their noisy counts** is identical.

Relation to Other Works

This equivalence applies to works that explicitly process all empty nodes, such as Chen et al. (2011) and Cormode et al. (2011), where efficiency gains are achieved without altering the statistical properties of the output. However, works that skip empty nodes entirely (e.g., certain trajectory publishing approaches surveyed in Jin et al., 2021) do not require this step and thus cannot directly benefit from the Binomial optimization.