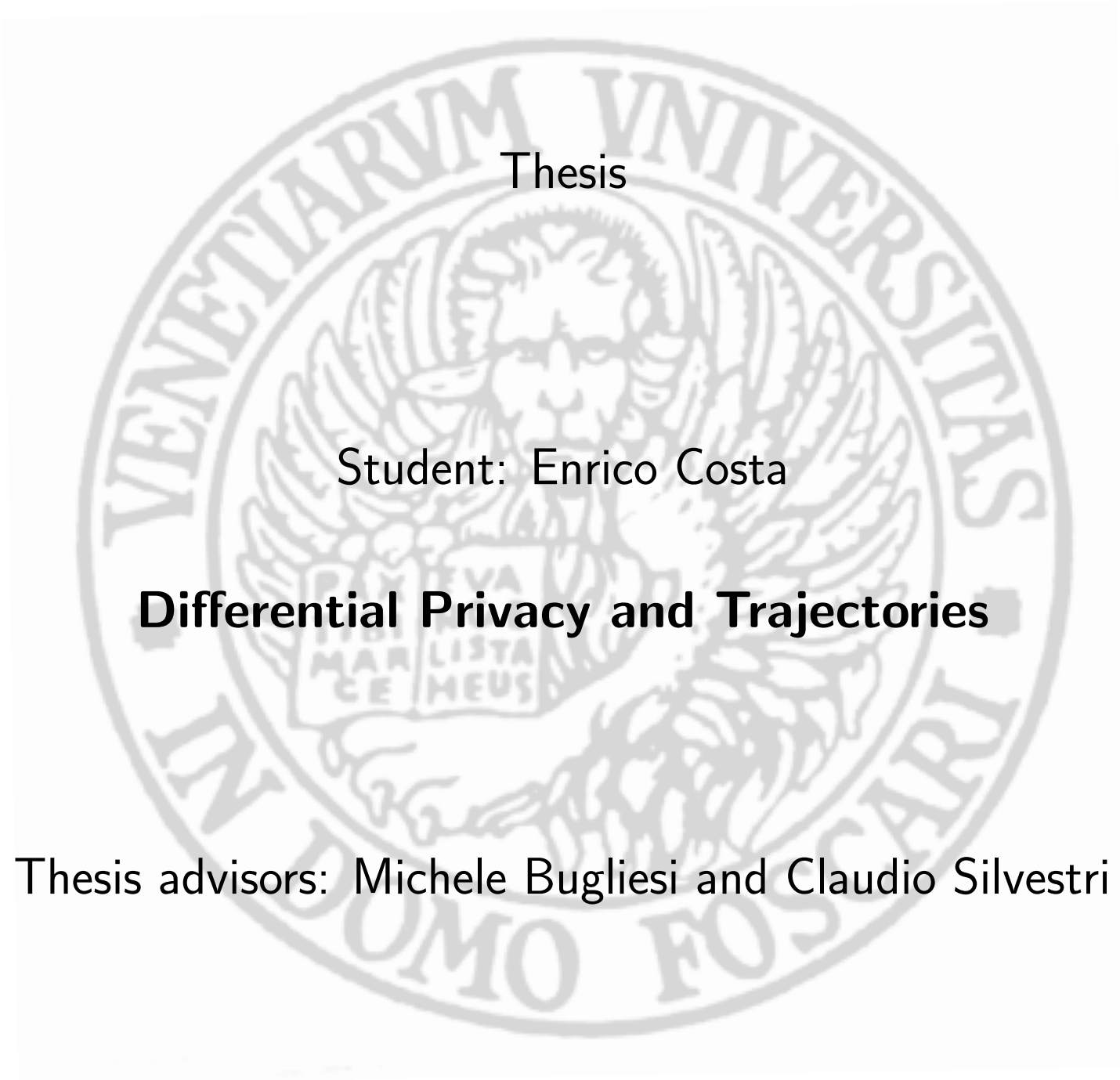
CA' FOSCARI UNIVERSITY OF VENICE
Department of Environmental Sciences, Computer Science and Statistics
MSc Program in Computer Science



Thesis

Student: Enrico Costa

Differential Privacy and Trajectories

Thesis advisors: Michele Bugliesi and Claudio Silvestri

Academic Year 2011 - 2012

Ca' Foscari University

Department of Environmental Sciences, Informatics and Statistics

MSc Program in Computer Science

MASTER THESIS

Enrico Costa

DIFFERENTIAL PRIVACY AND TRAJECTORIES

Supervisors: Michele Bugliesi and Claudio Silvestri

Corso di Laurea magistrale (ordinamento ex D.M. 270/2004) in Informatica

Tesi di Laurea

DIFFERENTIAL PRIVACY AND TRAJECTORIES

Relatori

Ch. Prof. Michele Bugliesi e Claudio Silvestri

Laureando

Enrico Costa

Matricola 815598

Anno Accademico

2011 / 2012

In 1977 Tore Dalenius articulated a desideratum for statistical databases: *Nothing about an individual should be learnable from the database that cannot be learned without access to the database.* We show this type of privacy cannot be achieved.

Cynthia Dwork

Acknowledgments

I would like to thank Michele Bugliesi, Claudio Silvestri and Frank McSherry, my collegues Giorgino Portelli, Andrea Destro and Gianluca Barbon from Team Astrosolid, and also Max Perinato, Francesco Pizzolon, Vadym Boychuck, Kameran Lak, Alessandro Mantovani, Jim, Guru, Juliet, Tewodros, Andrea and Gianpietro Casinei of the Department of Computer Science, my christmas flatmate Joshua Zem and Elena Altare.

Differential Privacy and Trajectories

Enrico Costa

DAIS
University Ca Foscari
Venezia, Italy
2012

ABSTRACT

Publishing meaningful datasets that don't jeopardize the privacy of the participants is still a great challenge for the database community. It has been demonstrated that the recent differentially private approach guarantees a good privacy protection maintaining accurate query results. With the increasing popularity of devices which generate spatio temporal data, trajectories related to human movements are being collected and stored in increasingly large quantities. Trajectory data are extremely useful for data mining tasks but they can expose the participants of the datasets to privacy breaches. This thesis aims to offer a differentially private protection to trajectory datasets, in order to allow their safe publication for data mining tasks. A survey of the most recent researches on the topic is presented, followed by the description of potential solutions to the problem.

Contents

1 Introduction **1**

1.1 Violating the privacy of individual movements 1

1.2 Protecting the privacy of individual movements 2

1.3 Goal of the thesis 3

2 Differential Privacy **5**

2.1 The idea behind Differential Privacy 5

2.2 Differential Privacy formally 6

2.3 The interactive and the non-interactive models 7

2.3.1 Output Perturbation: the interactive setting 8

2.3.2 Limits of the output perturbation mechanism 8

2.3.3 Data Perturbation: the non-interactive setting 9

2.3.4 Limits of the data perturbation mechanism 10

3 Trajectories **12**

3.1 Trajectories formally 12

3.2 Differential Privacy and Trajectories 13

3.2.1 Related work 13

3.2.2 Overview 16

3.3 Queries on trajectory data 16

3.4 Differential Privacy for location data 17

3.5 Data dependent and independent splitting 19

3.6 Local sensitivity 20

4 The Method . . . 22
4.1 Overview of the method . . . 22
4.2 Spatial decomposition as first step . . . 25
4.3 Privacy Level Distribution . . . 29
4.4 Perturbation of the trajectories as second step . . . 30
4.4.1 From points to locations . . . 31
4.4.2 Repetition of locations . . . 32
4.4.3 Timestamps . . . 33
4.5 Prefix tree structure . . . 35
4.6 Prefix tree construction . . . 35
4.7 Release Generation . . . 38
4.8 Calculation of the leakage value . . . 40
4.9 Complexity analysis . . . 41

5 Implementation . . . 43
5.1 Framework . . . 43
5.2 First step: spatial decomposition . . . 44
5.2.1 Z-ordering and Quad-Tree . . . 44
5.2.2 Leafs to Locations . . . 46
5.2.3 Deleting successive duplicates . . . 47
5.3 Second step: trajectory perturbation . . . 47
5.3.1 Preservation of the timestamps coherence . . . 48

6 Experimental Results . . . 49
6.1 Dataset . . . 49
6.1.1 Data preprocessing . . . 49
6.2 Utility . . . 52
6.3 Scalability . . . 52

7 Future Work . . . 53

8 Conclusions . . . 54

Bibliography **55**

List of Figures

1.1 Spontaneous revelation of the current position 1

2.1 Laplace noise distribution 7
2.2 Scheme of the Output Perturbation approach 8
2.3 Non-interactive setting as described in [FWCY10] 10
2.4 Scheme of the Data Perturbation approach 10

3.1 The approach of Abul and others [ABN08] 13
3.2 KD-tree partitioning 19
3.3 Hilbert Curve 20

4.1 Example of trajectory data set in the railway station of Venice 23
4.2 Prefix tree derived from example 4.1 24
4.3 Scheme of the procedure described in 4 26
4.4 Prefix tree generated from example 4.2 36
4.5 Algorithm demonstration (1) 38
4.6 Algorithm demonstration (2) 38
4.7 Algorithm demonstration (3) 39
4.8 Application of the *consistency constraint* 40

5.1 Application of the *Z-ordering curve* (1) 45
5.2 Application of the *Z-ordering curve* (2) 45

6.1 Distribution of Geolife trajectories nearby Beijing 50

List of Tables

3.1 A set of medical records 15

3.2 Example of application of a k-anonymity mechanism to the set of medical records 3.1 15

4.1 Database of trajectories corresponding to the example 4.1 23

4.2 Trajectories containing subsequent repetitions of the same location 32

4.3 Trajectories containing subsequent repetitions after the pruning passage . 33

4.4 A trajectory with associated timestamps 34

4.5 Final release from example 4.1 40

5.1 Representation of the trajectories of figure 4.1 using couples of (x,y) coordinates 44

5.2 List of trajectories between symbolic locations expressed as couples of points 46

6.1 The structure of the fields in the database file 50

Listings

4.1 Pseudo-code algorithm of the spatial decomposition step 27
4.2 Noisy tree construction algorithm from [CFD11] 36
6.1 The Phyton conversion program 51

Chapter 1

Introduction

1.1 Violating the privacy of individual movements

The majority of people do not know that when they move around, it is straightforward for the police and for other entities to track and reconstruct their movements. Our mobile phone emits constant signals that can reveal our current position: Law enforcements everyday exploit informations about people movements in order to catch criminals, thwart terrorism actions and so on. This procedure can be seen as morally right or wrong depending on the point of view, but what is certain is that there exist laws designed to protect the privacy of our data. Daily, traces of our movements are collected and stored in databases, for example when we use a service as public transport, or when using our smartphone we access social applications based on geospatial data and we spontaneously reveal our current position. Privacy laws do not allow the entity that collect data to publish them or to explore and analyse the personal information of

Figure 1.1: Spontaneous revelation of the current position through a portable device

private subjects. Instead, it is possible to perform statistical analysis on a big amount of database records. Clearly every possible participant must give the permission to exploit his data for doing such kind of researches, and he usually does it, thinking that statistics can't reveal so much about him. Unluckily, if an attacker is in possession of auxiliary information about an individual whose data are stored inside a statistical database, he can combine his knowledge with the statistic answers over the data, easily discovering sensitive information. In example, say you want to figure out the average salary of the people in a room without revealing anything about your own salary other than what is inherent in the answer. Intuitively, the obtained information is not harmful since it does not reveal the exact salary amounts. Now, suppose one individual of the group goes to the toilet. Repeating the same question (or query) to the remaining pool will lead to a different average salary. If this new value is lower than the previous one, it is easy to derive that the guy in the toilet earns more money than the average of the others. In other words, removing an individual from the set can cause a privacy breach.

1.2 Protecting the privacy of individual movements

The state of the art of privacy protection of individual information within statistical databases is in continuous movement. New techniques pop up every year and then new attacks that circumvent them are found. In the example of the previous section i showed how the action of removing a participant from a statistical database can break his privacy. Differential Privacy is a relatively new privacy preservation technique that takes in consideration this particular issue, coming to the definition of a very strong privacy constraint. In chapter 2 the concepts behind differential privacy will be explored in details and compared with other privacy preservation techniques, mainly because differential privacy will be the instrument that this thesis will put in use in order to protect

the information regarding the participant of a database of movements, or Trajectory Database.

1.3 Goal of the thesis

This thesis aims to guarantee the privacy preservation of the participants personal information in statistical trajectory databases, using the strong constraints of the differential privacy. To my knowledge, this thesis is the first work that, starting from raw trajectories (lists of time-ordered points in the space), aims to publish a meaningful differentially private database release. Apart of the abundant literature about other approaches for privacy preservation of trajectory and location data that will be briefly introduced in section 3.2.1, the most similar researches that regard differential privacy preservation for spatial data are not related to trajectories, but only to points in the space, and therefore do not take in consideration queries on connected points, like frequent sequential pattern mining. On the other side, two extremely recent papers, published in the last months of 2012, tried to apply the differentially private approach to ordered sequences of data, such as DNA sequences, web browsing histories and movements between well-defined location. However, there is a difference between these two papers and this thesis. In fact, while the latter is based on linked points in the space, so it has to do with a domain of infinite possible combinations of two spatial coordinates, the previous ones have to do just with a finite number (usually relatively low) of possible locations. The thesis approach is to try to apply the differential privacy constraints for both the points in the space and the trajectories composed by these points. Two linked but different kind of data, each one related to his specific query classes. Intuitively, the perturbation procedure to be applied before the database release must be subdivided in two steps, one grouping points and one perturbing the set of trajectories. Can the differential privacy

be guaranteed for the final release? And how? This is the main question that this thesis tries to answer.

Chapter 2

Differential Privacy

2.1 The idea behind Differential Privacy

Can the privacy of the participants of a database be fully protected? Yes. For example giving totally random answers to every possible query performed on the data. But that's not what database community is looking for. There must be a trade-off between utility of the data and privacy preservation. So the question can be formulated better as: can a database offer meaningful statistical answers while preserving the privacy of his participants? This question corresponds to the *desideratum* presented by Dalenius in 1977 [Dal77]: *Nothing about an individual should be learnable from the database that cannot be learned without access to the database.* Unfortunately, Cynthia Dwork in 2006 presented the mathematical proof [Dwo06] that this kind of privacy can not be achieved. Interestingly, she realized that in the way the desideratum is expressed, even someone that is not in included in the database is in potential danger. She took as an example Terry Gross, an imaginary Lithuanian woman not included in any database. The second actor of the example is a potential attacker that wants to derive her exact height, that is supposed to be a sensitive information, only knowing that she is two inches shorter than the average height of Lithuanian women. Consulting a database that yields the average heights of women of various countries would cause a privacy breach, even if Terry Gross records are not contained in the database. This discovery leaded to an interesting new perspective on privacy preservation: *any output of the privacy mechanism that protects*

the database should be almost totally independent of whether any participant in present or absent from the database. Dwork in [DN08] points out that under this constraint a bad disclosure is still possible, but if differential privacy is respected the individual can be sure that is not the presence or absence of his data that can cause such a disclosure, and that it couldn't be avoided through action or inaction of any kind.

2.2 Differential Privacy formally

In the previous section I introduced the motivations behind the differential privacy approach. This section exposes its mathematical basis and how to apply its constraints in practice. The concept to be formalized is that this privacy mechanism should assure that if a participant removes his data from a database, no outputs would become significantly more or less likely. Suppose that K is a privacy mechanism. For all D_1 and D_2, databases differing on at most one element, and all $S \subseteq Range(K)$:

$$Pr[K(D_1) \in S] \leq exp(\epsilon) \times Pr[K(D_2) \in S]$$

ϵ is a value that represent the leakage and can vary from 0 to 1, meaning respectively stronger or weaker privacy preservation. (ϵ) - differential privacy can be achieved adding noise, whose magnitude is related to the variation that the removal of a single participant can cause on the output. The maximum query output variation when removing an element is represented by the global sensitivity of a query.

Formally, for $f : D^n \rightarrow \mathbb{R}^d$ where R^d is a d-dimensional vector space and $x \in D^n$, the global sensitivity of the query f at x is:

$$GF_f = \max_{x,y:d(x,y)=1} ||f(x) - f(y))||$$

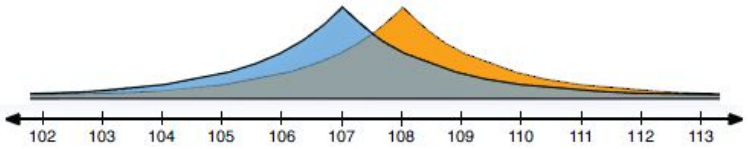


Figure 2.1: Adding symmetric exponential noise to counts causes the probability of any output to increase or decrease by at most a multiplicative factor when the counts are translated

In practice, there are many procedures that can guarantee (ϵ) - differential privacy. One of them is the perturbation of the query output with Laplace noise $Lap(\sigma)$ such that $\sigma = \frac{\Delta f}{\epsilon}$.

Mechanisms that provide differential privacy have two important properties [McS09]:

Sequential Composition

A sequence of M_i mechanisms on an input domain D provides $\sum_i \epsilon_i$ - differential privacy

Parallel Composition

If every M_i acts on a disjoint subset D_i of D, the privacy provided will be $(max(\epsilon_i))$ - Differential Privacy $\forall M_i$

2.3 The interactive and the non-interactive models

In the previous section, differential privacy was formally presented, along with its main properties. It is now good to distinguish the various approaches aimed to guarantee the privacy protection.

In the survey of Adam and Wortmann [AW89] three main settings are identified:

- Query Restriction

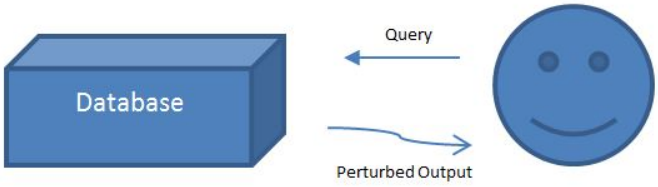


Figure 2.2: Scheme of the Output Perturbation approach

- Output Perturbation
- Data Perturbation

In my opinion focusing on query restriction is superfluous if the thesis goal is to produce a database release that can be used as one wants. Considering this thesis is about perturbation of information, the second and the third approaches are the most relevant to the actual interests.

2.3.1 Output Perturbation: the interactive setting

In the output perturbation approach, the queried database first computes the real answer and then outputs a noisy version of it. In this setting, usually there is a framework protecting the data, that dynamically deciding how much noise add to the outputs, on the basis of a *privacy budget*. The definition of interactive setting derive from this continuous question-answering. PINQ [McS09] is a good example of differential privacy preserving framework based on output perturbation.

2.3.2 Limits of the output perturbation mechanism

A well-known problem within the interactive setting is that question-answering leaks some privacy at every answer. At a certain point, the dataset is not useful any more.

In example, suppose one need to guess once more the average salary on the people in a room. Following the indications of differential privacy constraints, the added noise to the answer is enough to mask the presence or absence of any individual. This is enough satisfying, but only if the query is asked just once. The problem is that if there are no bounds on the number of queries that can be asked, an attacker can just ask the same query multiple times. For the Law of the Large Numbers he will soon be able to guess the exact average salary even if the query answers are always different, because the probability distribution of the replies is always the same and the average value of the query outputs will soon correspond to the searched value. Potential solutions to this issue consist in assigning a *budget* to the data. Each performed query would lower that budget until it will not be possible to ask further information to the database. The budget must be bounded to the data and not to the user, because it is necessary to take in consideration potential information sharing between different attackers. When the privacy budget finally is over, the entire database must be thrown away since no query can be performed on it any more. This, for many contexts (not all, though), is a big waste because harvesting huge quantity of data for statistical databases often requires great efforts.

2.3.3 Data Perturbation: the non-interactive setting

The non-interactive setting offers a possible alternative to the previously described approach. The database can be perturbed under certain constraints and then released. Afterwards, the perturbed (or sanitized) database can be explored and queried by the users without limits. This procedure is also known as Privacy Preserving Data Publishing (PPDP). Fung and others [FWCY10] show a classical mechanism of data publishing as the model described in figure 2.3. If the database in the non-interactive contest is sanitized in the proper way, there is no need to throw away the data, since they don't

represent a privacy threat, independently from the number of queries one can perform on them.

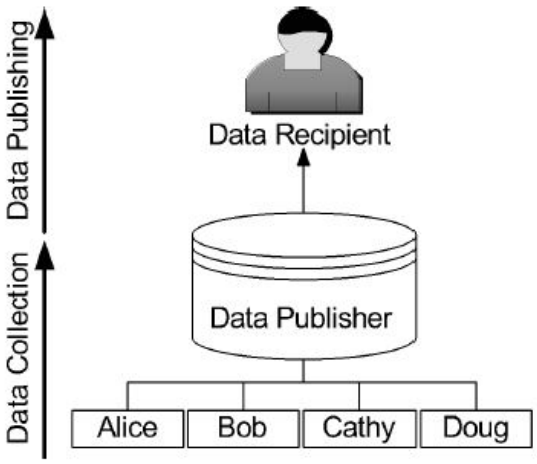


Figure 2.3: Non-interactive setting as described in [FWCY10]

2.3.4 Limits of the data perturbation mechanism

There are issues related to this second approach as well. The most relevant is that in the interactive setting the privacy mechanism to be applied can be easily parametrized at each query. In the case of differential privacy, in example, the parameter ϵ can be dynamically set at each query. This is impossible within the non-interactive contest, since it would require to perturb every time the entire database along with the chosen value of ϵ. For this reason, efforts on the research of differential privacy preservation were

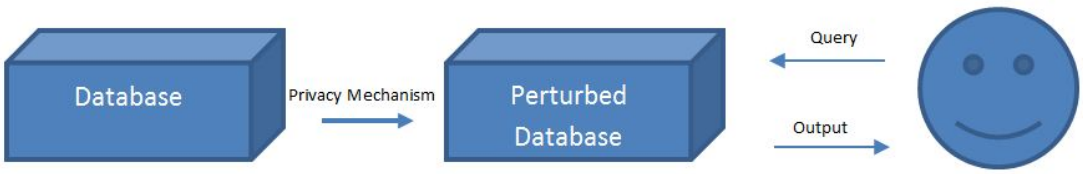


Figure 2.4: Scheme of the Data Perturbation approach

directed mostly toward the interactive setting. However, not all the problems can be reduced to the interactive setting. As an example one can take the release of a database of trajectories in [CFD11]: a smart card automated fare collection system generates and collects daily passengers trajectory data. Trajectory data are stored in a central DBMS, and the IT department of the STM (Societè de transport de Montreal) shares periodically such travel data with the other departments like the marketing department in order to perform data analysis, and also publishes the data to external research institutions for data mining tasks. In such a contest, it's unthinkable to use an interactive setting. In summary, even if the output perturbation approach seems to fit better to the differential privacy approach [McS09], the possibility of publish sanitized trajectory databases protected by differentially private perturbation must be taken in consideration. This thesis will concentrate on the non-interactive setting and on publishing a perturbed database release under the differential privacy constraints.

Chapter 3

Trajectories

In the previous chapter the differential privacy was presented, along with possible approaches aimed to preserve the privacy of the database participants. This chapter will focus on the description of the data to be protected: trajectories. How can intuitively be defined a trajectory? As an ordered sequence of points in the space.

3.1 Trajectories formally

Following the convention of [Ho12], let a trajectory $T = (p_1, p_2, ..., p_k)$ be a sequence of k measurements, $p_i = (x_i, y_i, t_i) \forall 1 \leq i \leq k$ where x_i is the latitude, y_i the longitude, t_i is a timestamp of the i-th measurement, and $t_i < t_{i+1}$.

The database records corresponding to this definition are then supposed to be in this "raw" form:

```
Trajectory ID | x coordinate | y coordinate | time stamp
```

A trajectory T is the list of the records having the same Trajectory ID. A database of trajectories can be defined as $D = T_1, T_2, ..., T_n$ where n is the total number of trajectories.

3.2 Differential Privacy and Trajectories

3.2.1 Related work

Related work to this thesis ranges among two main topics: differential privacy theory and privacy preserving trajectory publication. With regards to the latter topic, I can list between the most interesting researches th (k, δ)-anonymity solution elaborated by Abul and others [ABN08], that encloses k trajectories in a cylinder of radius δ as shown in figure 3.1.

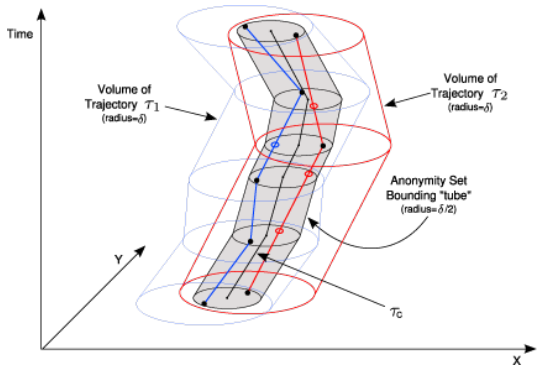


Figure 3.1: The approach of Abul and others [ABN08]

Other works related to k-anonymity are [YBLW09], [HXO+10] and [AAG+09]. The first uses timestamps as QIDs (Quasi-identifiers, or QID, are combinations of attributes within the data that can be used to identify individuals), the second ideates a new technique called local enlargement, the latter makes use of a particular form of data-dependent spatial generalization. Anyway, k-anonymity privacy definition is slightly distinct from differential privacy, in fact it captures the fact that an individual should be hidden between a set of equal ones. In the example 3.1, taken from [MKGV07], suppose that Brijesh Gohel is the only 37-year old Indian whose ZIP code is 13053. After a 4-anonymity algorithm is applied on the database, there are at least four individuals that can be him. However, if an attacker is in possession of the auxiliary information that Bri-

jesh Gohel record is located in the row number 10 of the database. It is straightforward to guess that he has cancer. So actually the database in example 3.1 is not differentially private. Even without explicitly querying for the specific individual information there exist many ways to obtain his medical status once auxiliary information is in possession of the attacker. Translating the same issue to trajectory data publication, it is clear that k-anonymity is a weaker privacy preservation technique if compared with differential privacy, the same reasoning can be applied with respect to all the partition based privacy models, including l-diversity and many others. Interested readers can examine the extensive survey [FWCY10] for a list of the major privacy preservation approaches and their respective weaknesses. Even if k-anonymity is a weaker approach to privacy preservation, researches took place with the goal of finding a parallelism between the two privacy definitions. Li and others [LQS12] defined a stronger version of k - anonymity called $safe\ k$ - anonymity, and a weaker version of ϵ - differential privacy, called $(\beta, \epsilon, \delta)$ - differential privacy, stating that it is reasonable to relax the ϵ - differential privacy to the situation in which the adversary knows all attributes of a tuple, but not if it is or not in the database, and he has only statistical informations on the database, and not the perfect knowledge of all the tuples. This weaker definition exploits adversary uncertainty. In this new definition, the β parameter stands for the probability of including a tuple in the sample dataset during an initial sampling procedure. Unfortunately it was proven in [LQS12] that this privacy notion does not compose, in the sense that if the procedure is applied two times to a database, obtaining two sampled database A and B, and these two database are published, differential privacy does not hold any more. This issue leads to the same problem of the privacy budget explained in section 2.3.1, since the procedure can be applied to produce just one sampled database. For this reason, I think that try to modify the existing k - anonymity approaches with the goal of achieve a differentially private release may not be very productive. Therefore, I

focused on the researches that regard differential privacy, trying to adapt them to the particular case of trajectories database perturbation. The literature about differential privacy, applied to the most various context, is growing huge month after month, but rarely facing the issue of privacy preserving trajectory data publication. In the next sections, anyway, the most important researches on the topic are taken in consideration if considered useful for a better understanding.

	Zip Code	Age	Nationality	Condition
1	13053	28	Russian	Heart Disease
2	13068	29	American	Heart Disease
3	13068	21	Japanese	Viral Infection
4	13053	23	American	Viral Infection
5	14853	50	Indian	Cancer
6	14853	55	Russian	Heart Disease
7	14850	47	American	Viral Infection
8	14850	49	American	Viral Infection
9	13053	31	American	Cancer
10	13053	37	Indian	Cancer
11	13068	36	Japanese	Cancer
12	13068	35	American	Cancer

Table 3.1: A set of medical records

	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

Table 3.2: Example of application of a k-anonymity mechanism to the set of medical records 3.1

3.2.2 Overview

How can differential privacy be applied on databases of trajectories? Differential privacy deals with the removal of a single individual from the database. Notice that an user can produce few or many trajectories, but if the approach has to be generally applicable to the all types of moving object measurements, it is necessary to consider even databases in which only trajectory identifiers are stored, and not the identifiers of the data owners. Also, the first step of anonymization of this type of databases often consists in removing of the user identifier. Henceforth in this thesis *individual* will stand for trajectory, if not differently specified. Recalling the definition of differential privacy given in section 2.2, it is clear that the usefulness of the privacy mechanism applied is strictly related to the global sensitivity of the queries performed on the database. For this reason, in the next section a description of the queries that will most likely performed on the released database is introduced.

3.3 Queries on trajectory data

According to Cheng [CFD11] and others, queries on trajectories can be subdivided in two main classes: queries that requires only the locations information, and queries that requires also informations about the relationships between locations. Therefore intuitively the global sensitivity and the privacy mechanisms must be guaranteed for both these two query classes. Let's focus on two typical queries:

- Count queries: *how many trajectories share the same location?*
- Frequent sequential pattern mining: *how frequent is a certain sequence of locations?*

The former query relates only to the locations, the latter is necessarily bounded to entire trajectories. Intuitively, in the first case the goal is to protect the privacy of the locations, and in the second case the goal is to protect the trajectories as well. Since the database to be released must be just one, it is straightforward to think that two different differentially private mechanisms should be applied to the data: the first one based on the removal of one single point and on the count query sensitivity, the second one based on the removal of a trajectory and based on the frequent sequential pattern mining sensitivity too. The next section will describe how to potentially perform a differentially private perturbation for the points data, keeping in mind that trajectories consistency and coherence have to be preserved in order to obtain meaningful query outputs.

3.4 Differential Privacy for location data

The idea of differentially private data-partitioning index structures was previously suggested by Inan and others [IKGB10] in the context of private record matching, but the first publication strictly related to spatial data partitioning is the one of Xiao and others [XXY10], and created the bases for the significant work of Cormode and others [CPS+11]. The effort of these researches was directed to answer an annal question of spatial based applications: to know how many points fall within a certain region. These researches are a fundamental basis towards the count query privacy preservation within a trajectory database. The major part of the publications on the topic states that the existing privacy-preserving data publishing methods derived from partition-based privacy models, for example k-anonymity, are unable to provide sufficient privacy protection, while differential privacy approaches may lead to interesting results. The common idea is to divide the two-dimensional space domain in regions or clusters of points. Focusing

on this thesis, such operation will have two goals:

- Application of a noisy space partitioning procedure aimed to guarantee differential privacy for the location data released.
- Proportioned discretization of the space domain for the successive trajectory perturbation procedure.

My choice of space partitioning over clustering comes from the fact, not underlined in the previously listed articles, that trajectories pass through regions. Differentially private clustering, instead, tends to group points if they are nearby, and tends to remove isolated points. Now, imagine a trajectory over the space of the entire North-America that pass through some desert glacial region of north Canada. Clustering the points would cause the removal of that specific location. In order to maintain the consistency of the trajectory the only solution would be to map the former couple of coordinates on the nearest cluster centre. That would potentially lead to enormous modifications of the trajectories if the clusters are not many. This is the main reason for which i prefer to stick on the idea of a space partitioning procedure that will result in a domain of location such that every possible point in the world domain would have a correspondent location. Moreover, Cormode [CPS+11] states that applying the classical differential privacy mechanisms on spatial data generates just noise without any useful information. This theory is shared by [CFD11] when related to trajectory publication. For example, suppose we represent a set of 10^7 GPS locations in a grid of 10×10 meter squares over the USA territory: this will generate approximately 10^{11} records, and almost all of the counts for the locations would be 0 or 1. Outputs of the spatial queries will be therefore a large number of noisy count with almost no information useful for accurate query outputs. Spatial decomposition appropriate methods, instead, allow to obtain compact regions with sufficiently many points and a more uniform distribution, and therefore is

expectable that query answers would be more accurate.

3.5 Data dependent and independent splitting

Techniques of space splitting can be divided in data-dependent and data-independent. Quad tree decomposition is obtained recursively splitting the areas in four equal squares. This type of partitioning procedure is data-independent because it does not take in consideration the distribution of the points in the space. When however regions are split on the basis of the points distribution, such as KD-Trees approaches, the setting is considered data-dependent.

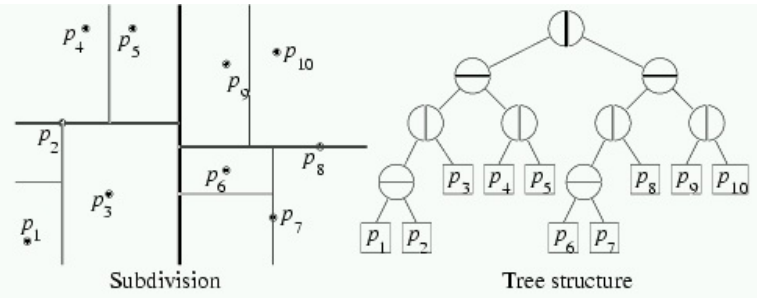


Figure 3.2: KD-tree partitioning

There exist clearly many methods that can be used to identify the points in the space, one of them is Hilbert R-Tree, shown in figure 3.3, that uses the Hilbert curve to find the points in the space and subdivide the regions, other ones will be explored in the implementation chapter 5. Data-dependent approaches were proven to be more efficient in some contests [CPS+11]. However, for the purposes of this thesis even the simpler Quad-tree representation is fine.

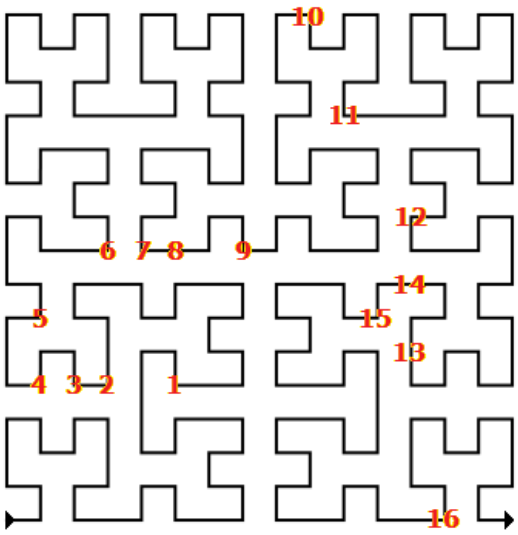


Figure 3.3: Hilbert Curve

3.6 Local sensitivity

Splitting the spatial domain in sub-problems using quad trees or KD-trees allows to obtain a good practical result: achieve a good local sensitivity for the subregions. What is local sensitivity? And how can it be used to improve a spatial decomposition algorithm? As an example, suppose Michael, a database participant, goes to vacation leaving his GPS device working at his home for days. The database would be enriched by thousands of measurements related to the Michael's house location. The global sensitivity of the entire database would be affected, influencing therefore the noisy counts of locations on the other side of his city, totally unrelated to it. Spatial decompositions procedures solve this problem through the use of the local sensitivity. The results are: better accuracy for both high and small density of points within a location. But what exactly is the local sensitivity? And how a differentially private mechanism be calibrated on local sensitivity instead of the global one? This section exposes the definition of local sensitivity in the contest of differential privacy, and how it can be used keeping the privacy preserved while improving the accuracy of count queries.

Local sensitivity in the contest of differential privacy was introduced by Nissim and others [NRS07] when they realized for some classes of queries, like the median calculation, the global sensitivity formula yields to highly noisy privacy mechanisms that does not reflect a possible function's typical insensitivity to individual inputs. Local sensitivity, differently from global sensitivity 2.2, is related with a specific instance of the database. For $f : D^n \rightarrow \mathbb{R}^d$ and $x \in D^n$, the local sensitivity of f at x is:

$$LF_f(x) = max_{y:d(x,y)=1} ||f(x) - f(y))||$$

Comparing this formula with the global sensitivity definition 2.2, it can be noticed that notion of local sensitivity is analogue to the maximum magnitude of the partial derivative in different directions. Local sensitivity value is calculated over a specific x and all the possible databases y that differ from x only for one element. Why then performing spatial partitioning in a recursive way would guarantee ϵ - differential privacy for the entire database? Every region would be perturbed with different noise quantities, but a single region can be indeed seen as a self-standing database, and so its global sensitivity equals its local sensitivity. Recursively calculating the noisy counts on subregions therefore guarantees differential privacy at every level of recursion and for the all regions and subregions. In the final step, the area correspondent with the measure of the spatial database domain would be in practice perturbed with global sensitivity.

Chapter 4

The Method

In the previous chapters the differential privacy concepts were introduced, followed by a description of the data this thesis aims to protect: trajectories. Additionally, the main issues related to this kind of data, the most popular query classes and the their privacy constraints were analysed. In this chapter a possible approach aimed to produce a differentially private release from a dataset of raw trajectories is introduced. The method acts in two steps. The first step aims to map the highly-private original spatio temporal coordinates of the participants movements on a set of trajectories based on a differentially private bag of locations of finite dimension. The second step of the method further perturbs the database with the goal of achieving differential privacy for the trajectories. An brief description of the method is summarized in the following overview.

4.1 Overview of the method

As a small example, suppose that a database is composed by eight trajectories correspondent to a total of 22 couples of coordinates, sensitive data for which must be guaranteed the maximum privacy preservation. Real life databases are composed by thousands of points and trajectories, but for the sake of simplicity I will stick to this tiny example. The first operation is to group the points into regions using an appropriate spatial decomposition algorithm which will be better described later in this chapter. In figure 4.1

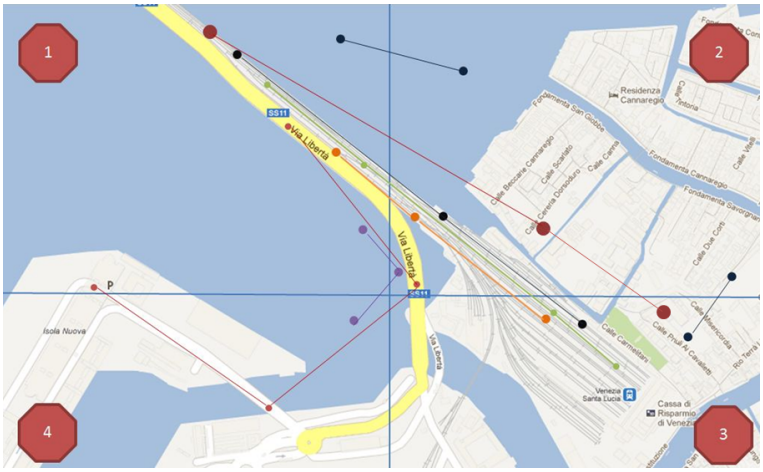


Figure 4.1: Example of trajectory data set in the railway station of Venice

a possible spatial decomposition is shown using two blue lines that divide the space in four quadrants. Suppose that the four regions obtained correspond to the four spatial decomposition tree leafs, and therefore consists in the space domain $\mathcal{L} = (L_1, L_2 ... L_{|\mathcal{L}|})$, where a single trajectory is an ordered list of locations $T = (L_1, L_2 ... L_{|T|})$ $\forall i$ such that $1 \leq i \leq |T|$. In the example, the location domain is $\mathcal{L} = (L_1, L_2, L_3, L_4)$. The trajectories are listed in table 4.1:

T_1	$L_1 \rightarrow L_2 \rightarrow L_3$
T_2	$L_1 \rightarrow L_2$
T_3	$L_3 \rightarrow L_2 \rightarrow L_1$
T_4	$L_1 \rightarrow L_2 \rightarrow L_4$
T_5	$L_1 \rightarrow L_2 \rightarrow L_3$
T_6	$L_3 \rightarrow L_2$
T_7	$L_1 \rightarrow L_2 \rightarrow L_4 \rightarrow L_1$
T_8	$L_3 \rightarrow L_1$

Table 4.1: Database of trajectories corresponding to the example 4.1

The main concern of this step is to split the space in an appropriate way, obtaining a set of locations of different dimensions, but having a proportionate distribution of measurements. Differential privacy, when applied to this step, allows to guarantee a certain degree of uncertainty with regards to the number of points in a region (See section 4.2 for details). After this procedure, trajectories can be stored as lists of locations. From these lists it is straightforward to build a prefix tree (figure 4.2)

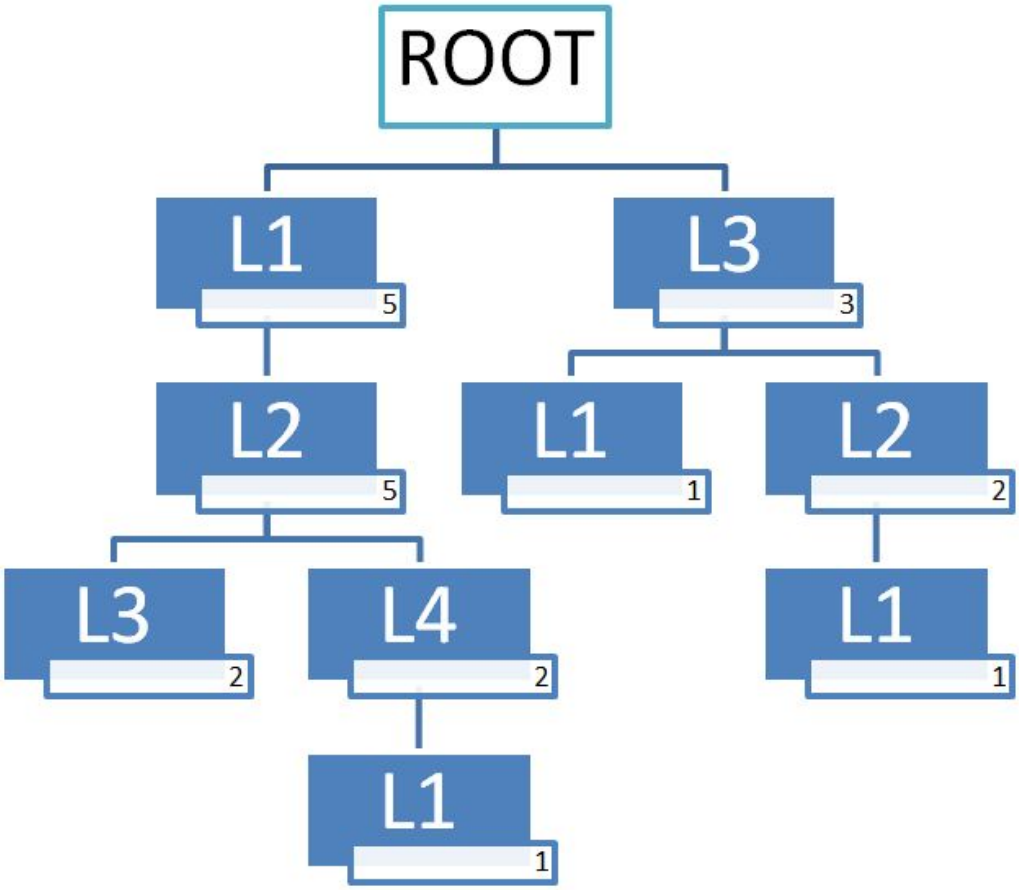


Figure 4.2: Prefix tree derived from example 4.1

Cheng and others [CFD11] use an algorithm based on a prefix tree in order to perturb the *Montreal transport society* dataset *(www.stm.info)*, the public transit agency of Montreal. The prefix tree of a trajectory dataset D is composed by an abstract root

connected to a first level of nodes. These nodes are the starting locations of the trajectories. Sticking to our example, the root has just the locations L_1 and L_3 as children. The node L_1 contains also informations about how many trajectories have the location L_1 as starting point. Each path that continues from L_1 is represented with a new sub-tree of the node L_1, and so on. See figure 4.2 for a graphical sketch of the prefix tree derived from example 4.1. The records 3, 6 and 8 share the same starting location L_3, therefore they are represented by the sub-tree which root is the node L_3. The associate number is the count of the trajectories that share L_3 as starting location. In this way the entire tree can be built.

Now, the prefix tree contains at level n informations about the quantity of trajectories that share the same initial sequence of $L_1...L_n$ locations. In section 4.4 it will be explained in deep how a noisy version of this prefix tree can be produced. As final step, the perturbed prefix tree can be re-mapped on the original form of the dataset and can be published without potential harm for the participants. Recalling that the locations at this stage are not couples of coordinates, but areas, they can be seen as couple of points, where the two points are the left-upper and the right-lower extremes of the square which corresponds to the area derived from the spatial decomposition. There are besides other possible representations that will be explored in section 4.4, dedicated to this part of the method.

4.2 Spatial decomposition as first step

Accordingly to the research in section 3.4, it appears that the most recent researches on differential privacy for points in the space are focused on spatial decomposition. I will take as an example one particular trajectory (T_1) of figure 4.1: $L_1 \rightarrow L_2 \rightarrow L_3$, coloured in red, with bigger points with respect to the other trajectories. This trajectory, as

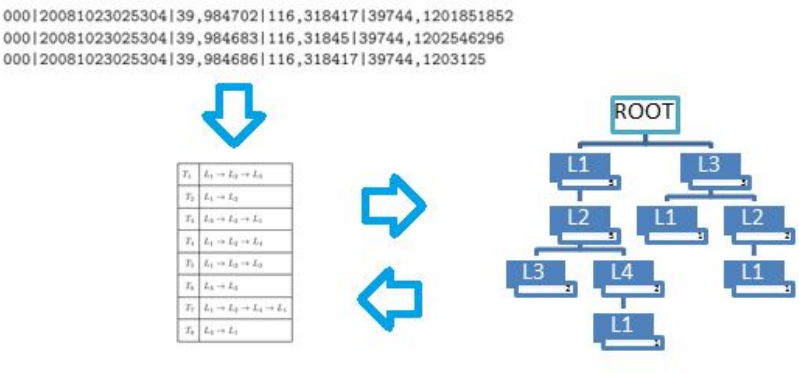


Figure 4.3: Scheme of the procedure described in 4

it is, reveals informations that perhaps the participant wants to keep private, like the specific house where he lives (and therefore his identity!), or the fact that at a certain time he was using the train. Many privacy techniques, surveyed in example in [DG] , can be used in order to protect the privacy of the participants, but this thesis is focused on differential privacy. A deep analysis of the problem of differential privacy for spatial points datasets was already performed by [CPS+11], [XXY10] and others, and briefly described in section 3.4. The most appropriate solution is to split the space into regions, and publish statistics on the points within each region in a differentially private manner. Then one can get the desired answers by intersecting the query regions with the spatial areas obtained with the splitting. It is important to understand the reason why the method presented in this thesis requires a differentially private space decomposition, and not a normal quad-tree representation. Actually, the only reason why this splitting is executed is to create from the leaf of the quad-tree a set on symbolic locations that group the points within a certain area. It's then task of the subsequent trajectory perturbation step to protect the differential privacy of the count queries outputs. Unfortunately, performing a non-perturbed spatial decomposition can lead a potential attacker to retrieve the exact count of the points within an area simply comparing the dimension of the subregion analysed with the corresponding counts performed on the

perturbed trajectory database. In order to make this issue clear, suppose the threshold used to decide if split or not an area is equal to 100. Obviously if the non-perturbed version quad-tree spatial decomposition is applied, every leaf would have at maximum 100 points. Suppose now that a count query intersecting only one region of the released perturbed database outputs the value 120. I recall that both ϵ and the exact dimension of the region, are known in the final result. Giving a sight to the region extension, an attacker can guess that the number of points in that area has to be 100, because it would be impossible for them to be 120 or more, since the threshold was 100. For this reason the splitting itself must be perturbed in a way that does not allow to the attacker to guess this information. This result can be achieved recalling the differential privacy idea that an attacker shouldn't be able to guess if a particular point is or not inside the database and how many points fall within a certain area. The result of applying the differential privacy to the space decomposition is basically the following: some areas that should be split are kept entire while some others that should be kept entire are split. This result captures the fact that a potential attacker will not be able to guess how exactly many point correspond to each leaf on the base of the threshold used. The algorithm 4.2 for the subdivision of the areas within the dataset is a modification of the one presented in [HR11], in which it is used to mine interesting locations with DBSCAN clustering algorithm. It takes as input a set S of points, that in this case are couples of coordinates with additive information regarding the trajectories, e.g. the timestamps corresponding to the points and the trajectory identifiers. The spatial region R must be taken as input too. A threshold T must be used as well to stop the recursion of the algorithm when a subregion is small enough or contains too few points. The output is a set of spatial partitions P and a set Sp of sets of points for the corresponding partitions in P. The global variables $P = \{\}$ and $Sp = \{\}$ are used in order to store these two sets.

Listing 4.1: Pseudo-code algorithm of the spatial decomposition step

```
1 BuildDPQuadTree(S, R, T)
2   noisycount = |S| + Lap(σ_qt)
3   if noisycount < T then
4     P = P ∪ {R}; Sp = Sp ∪ {S};
5     return
6   else
7     Split spatial region R into 4 equal quadrants:
8     BuildDPQuadTree(S_{nw},R_{nw},T)
9     BuildDPQuadTree(S_{ne},R_{ne},T)
10    BuildDPQuadTree(S_{sw},R_{sw},T)
11    BuildDPQuadTree(S_{se},R_{se},T)
12  end if
13  return
```

The algorithm executes a noisy count of the current area points accordingly to the local sensitivity correspondent to the current region taken in consideration, and comparing it with the threshold T it decides if keep on splitting the area, or to stop. The possibilities are:

- noisycount $<$ T - There are enough points within the region to create a meaningful location, and there is no need to keep splitting the area. A new location is created and added to the set.
- noisycount $>$ T - There are still too many points within the region: a new split is performed.

After the spatial decomposition stop, it is possible to mine the interesting locations, using for example DBSCAN Density Based Clustering Algorithm [EpKSX96]. Alter-

natively, an easy method to generate an universe of locations is to consider the centre of the subregions identified by the leaf of the quad-tree, or, in order to maintain more information on the dimension of the areas, characterize them using a couple of coordinates correspondent to opposite angles of the squares. These options regards the implementation choices and are left for the apposite chapter 5.

4.3 Privacy Level Distribution

Property 2.2 states that one can achieve an overall ϵ - differential privacy guarantee applying a sequence of differential privacy mechanisms. If the desired privacy to be achieved is ϵ, and n perturbation mechanisms need to be applied, the single ϵ_i $\forall i$ such that $1 \leq i \leq n$ can be chosen accordingly with the best possible accuracy reachable and $\epsilon = \sum_{i=1}^{n} \epsilon_i$. From the previous sections, it is clear why two different mechanisms need to be applied to the database, the first that noisy locates the points into regions and the second, to be described in the next sections, that perturbs the trajectories. The leakage parameter ϵ can be decomposed in $\epsilon = \epsilon^P + \epsilon^T$ where ϵ^P is the leakage parameter used for the spatial decomposition and ϵ^T is the leakage parameter used for trajectories perturbation. Suppose now that the database must be secured with 0.5 - Differential Privacy. One can subdivide the leakage parameter in $0.5 = 0.2 + 0.3$. This means that in order to guarantee a quite strong privacy preservation, two stronger privacy mechanisms must be used. This is fine until we realize that even the ϵ^P parameter is the sum of the leakage at every recursive step of the spatial decomposition procedure and therefore $\epsilon^P = \sum_{i=1}^{n} \epsilon_i^P$. It is easy to guess that even if very few recursion are performed, the value of parameter ϵ_i^P at level i of the quad-tree would be extremely low, producing an intolerable amount of noise for each decomposition step. Luckily, property 2.2 comes in help in this case. In fact, each recursive spatial decomposition step acts of disjoint sub-

sets (see references [XXY10] and [CPS+11] for further clearances) of the points domain, therefore: $\epsilon = max(\epsilon_i^P) + \epsilon^T$. One could question if it the sequential composition property is valid for two different mechanisms, so different that they even acts on different domains and data types like points and symbolic locations. According to [McS09] any sequence of differentially private computations provides differential privacy interdependently even if subsequent computations depend arbitrarily on results of previous ones. Moreover, no possible post-processing can affect the validity of differential privacy, once it is guaranteed on the original privacy mechanism output.

4.4 Perturbation of the trajectories as second step

The previous sections showed how indeed spatial decomposition is a fundamental step for achieving a set of symbolic locations, each one representing a set of points that fall within a certain area. The question now is: what about the trajectory data? Is it reasonable to think that if the locations are now grouped, then also the trajectories (and consequently the users) are protected as well? I remark that differential privacy constraints among a database where an individual is represented by a trajectory must be verified removing an entire trajectory from the database, and not just a single point. Therefore, releasing the trajectories database as the actual set of lists of symbolic locations is not guaranteeing any type of differential privacy. Luckily, recent researches on differential privacy for sequence of elements (both [CFD11] and [CAC12]) come to help with some interesting ideas on how to produce a differentially private release of a database of sequential data. In the next sections these works will be explored and their ideas will be modified to be adapted to the actual aims.

The focal point of this thesis is to find a way to guarantee differential privacy for trajectory data without perturbing the output of the queries, but instead by publishing

a perturbed database that would give sufficiently accurate answers to the query classes listed in chapter 3.

4.4.1 From points to locations

The starting format of a trajectory is $T = (p_1, p_2, ..., p_k)$, sequence of k measurements where $p_i = (x_i, y_i, t_i)$ and x_i is the latitude, y_i is the longitude, t_i is a timestamp for the measurement, with $t_i < t_{i+1}$. A database record corresponds to one measurement, along with a trajectory identifier:

```
Trajectory ID | x coordinate | y coordinate | timestamp
```

Once the spatial decomposition step is performed and a quad-tree is obtained, its numbered leafs correspond to symbolic locations. Each leaf containing a certain amount of points. The informations preserved for the single record at the start of the second step of the method are:

- The location in which the measurement falls;
- The measurement timestamp;
- The trajectory ID.

A database row, depending on the implementation, after the decomposition step, takes this shape:

```
Trajectory ID | Location ID | timestamp
```

Formally, the set of all the records with the same trajectory identifier T can be expressed as $T = (L_k1, L_k2...L_{k|T|})$ where $k_1, k_2, ..., k|T|$ are all the symbolic locations

corresponding to each measurement of the trajectory. $\mathcal{L} = (L_1, L_2 ... L_{|\mathcal{L}|})$ is now the new anonymized location domain. Two problems raised in this step are the repetition of sub-sequent locations in a trajectory and the association of timestamps with tree nodes.

4.4.2 Repetition of locations

Depending on the threshold specified for the spatial decomposition, trajectories are going to contain sub-sequences of elements corresponding to the same location, like in example 4.2

T_1	$L_1 \rightarrow L_2 \rightarrow L_3 \rightarrow L_3 \rightarrow L_3$
T_2	$L_1 \rightarrow L_1 \rightarrow L_1 \rightarrow L_2$
T_3	$L_1 \rightarrow L_1 \rightarrow L_1 \rightarrow L_1 \rightarrow L_1 \rightarrow L_2$
T_4	$L_1 \rightarrow L_1 \rightarrow L_1 \rightarrow L_1 \rightarrow L_2$

Table 4.2: Trajectories containing subsequent repetitions of the same location

What can be said for T_2, T_3 and T_4? The difference between these three trajectories is the number of repetitions of L_1, otherwise they would be equal. I remark that for the query classes taken in consideration, the sequentiality among locations is more interesting than the exact time in which the individual was in the single location, mostly because a sequence of measurements within the same location correspond on different coordinate couples, but after the spatial decomposition step, they all refer to the same area. Therefore this information is only a waste of database space. It would be useful for data mining if there was the guarantee that the measurements are following the same distribution over the time, but actually this is not true for all databases. Geospatial databases in particular, including the GeoLife dataset [ZXM10] taken in consideration

in chapter 6, mix trajectories obtained by pedestrian, bikers, train travellers and so on, without information on the speed of the users. Before applying the second step of the method, therefore, a pruning procedure is performed deleting the records corresponding to sub-sequences of measurements that cover the same location. The records in example 4.2 become:

T_1	$L_1 \rightarrow L_2 \rightarrow L_3$
T_2	$L_1 \rightarrow L_2$
T_3	$L_1 \rightarrow L_2$
T_4	$L_1 \rightarrow L_2$

Table 4.3: Trajectories containing subsequent repetitions after the pruning passage

4.4.3 Timestamps

The final issue is to decide if carry the timestamp values to the final database release. The choice depends on the future use of the database. A conspicuous quantity of queries can be performed even without being in possession of the time information in the perturbed database release, but a count query asking how many individuals were in a certain region during a specific time interval would not be possible. Both the analogous researches [CFD11] and [CAC12] avoid to explore this issue, and just ignore the time informations. In particular, [CFD11] purposes as possible solution to consider as different two trajectories with different time stamps. This suggestion, however, would not be efficient at all for this case. It is easy to understand why: recalling the overview of section **??**, if two trajectories with same locations and different timestamps are considered different, almost all branches of the prefix tree would be mono-representative (e.g. they would represent only one trajectory). Perturbation on

that kind of tree would produce extremely unfaithful data.

For this reason in this work I try to keep the time information maintaining a reasonable quantity of tree branches. Actually, I elaborated a method based on similarity between timestamps corresponding to neighbour measurements within a trajectory. But before examining this procedure, we must consider which time values correspond to the locations at this point. While the timestamps for the locations that appear without repetitions are conserved, the value of the timestamps to be assigned to the *collapsed* locations sequences should be decided. Sticking to example 4.2, suppose T_4 has associated the timestamps written is 4.4. There exist two reasonable options to choose:

- Conservation of the initial and the final timestamps
- Calculation of an average value of all the timestamps

In my opinion, the latter option does not represent a heavier loss of information compared to the former one. In fact the former carries information about the precise moment in which the individual entered the area, but the latter carries information about the distribution of the measurements. Considering that the average value is only one, while the timestamps at the extremes of the interval are two, the second option is chosen. Everything is now ready for the next step of the algorithm: trajectory perturbation.

T_4	$L_1\|1359576759 \rightarrow L_1\|1359577116 \rightarrow L_1\|1359580716 \rightarrow L_1\|1359663516 \rightarrow L_2\|1360181916$

Table 4.4: A trajectory with associated timestamps

The next section will deal with this issue, purposing a possible strategy for releasing a database where differential privacy for the trajectories is guaranteed.

4.5 Prefix tree structure

In the method overview **??** I already introduced the prefix tree structure. There exist an exact correspondence without loss of information between the trajectory database that comes out from the computations described in the previous section and its prefix tree. Possible implementations for this transformation are described in the implementation chapter 5. The definition of a prefix is intuitive: a trajectory $P = p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_{|P|}$ is prefix of $T = t_1 \rightarrow t_2 \rightarrow ... \rightarrow t_{|T|}$ if and only if $|S| \leq |T|$ and $p_i = t_i \; \forall i$ such that $1 \leq i \leq |P|$. Formally, a prefix tree $PT = (V, E, Root(PT))$ is a tree built over a set of prefixes, each one represented by a node v. Prefix(v, PT) is an ordered list of locations from $Root(PT)$ to v. In the prefix tree used in this thesis, each node v must carry the time informations about all the measurements that it represents. Recalling example 4.2 and 4.4, the prefix tree of trajectory T_4 can be represented as 4.4. Each node of the prefix tree contains the information on the measurements timestamps in an appropriate structure. In figure 4.4, the time information for L_1 relative to trajectory T_4 is the average of all the *collapsed* locations timestamps: $\frac{1359576759+1359577116+1359580716+1359663516}{4} = 1359599527$.

Chapter 5 will rely on possible implementations for the prefix tree structure, while in the next section the theory behind trajectories perturbation is introduced.

4.6 Prefix tree construction

The fundamental idea behind this method phase is that *every possible* trajectory (e.g. each one that can be derived from $\mathcal{L}$) has a certain probability of appearing in the final release such that sensitive information is masked. Intuitively, exploring a prefix path every possible location of the domain has a minimum probability of appearing, even those which are not included in any trajectory having that prefix. This is the basic dif-

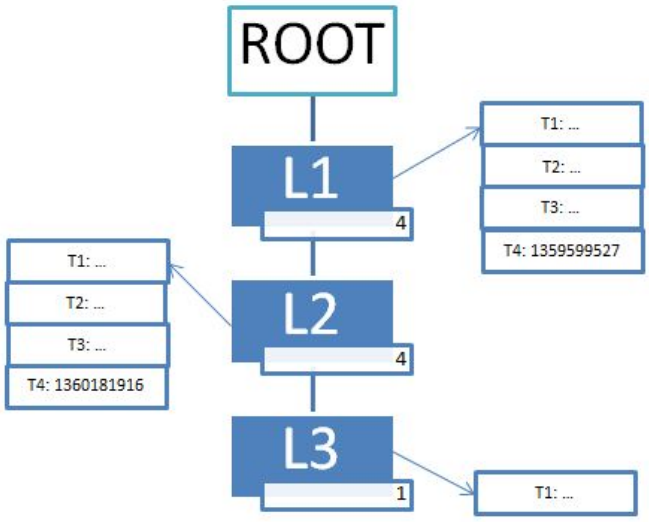


Figure 4.4: Prefix tree generated from example 4.2

ference between differential privacy and anonymization algorithms using generalization or sampling: the probability of having in the perturbed database a certain quantity of brand new fake trajectories.

Algorithm 4.6 is inspired to [CFD11]. It builds a prefix tree for the database D is a noisy way such that the final result is differentially private.

Listing 4.2: Noisy tree construction algorithm from [CFD11]

```
1 BuildNoisyPrefixTree(D, ϵ, h)
2 i = 0
3 Create Root(PT)
4 Add the trajectories of D to tr(Root(PT));
5 ϵ̄ = f(ϵ, h);
6 while i < h do
7   for each v ∈ level(i, PT)
8     Create a set of nodes U, each one
```

```
 9        representing a location of L
10      for each u ∈ U
11        Select the prefix trajectories of u
12        Add to tr(u) the trajectories
13          of tr(v) which are Prefix(u, PT)
14        c(u) = noisycount(|tr(u)|, ε̄);
15        if c(u) ≥ θ and u ≠ v then
16          Add u to the set of v children;
17        end if
18      end for
19    end for
20    i++;
21  end while
```

The algorithm behaviour is not trivial, example 4.1 can be recalled for better understanding. After the root is created, in line 4 all the trajectories in the database are added to its field *tr* (figure 4.5). During the first iteration of the *while* loop, the only node in level 0 of PT is the root and $U = L_1, L_2, ..., L_8$. Now, L_1 is the first element of U taken in consideration. The trajectories that are prefixes of this node are T_1, T_2, T_4, T_5, T_7, so they are added to its field *tr*. The *noisycount* result depends on the value $\bar{\epsilon}$. This value is calculated by the function f in relation to the height of the tree. A simple solution is $f = \frac{\epsilon}{h}$, even if other researches like [CPS+11] found that a different distribution of the budget can lead to better accuracy results. In chapter 5 it will be shown how to calibrate the $\bar{\epsilon}$ and which h and θ should be chosen. Anyway, suppose that L_1 passes the test in line 15: The next step is to add L_1 as *root* child, conserving the value $c(u)$ that will be used for the release generation. A possible $c(u)$ value for L_1 is shown in

red in figure 4.6. This procedure goes on for each location in the universe, and then the second level of three is taken in consideration. Timestamps, not represented in the figures for space reasons, are memorized in correspondence with each relative trajectory in the *tr* field.

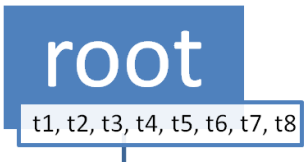


Figure 4.5: Algorithm demonstration (1)

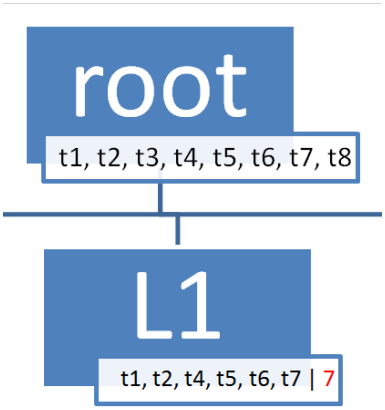


Figure 4.6: Algorithm demonstration (2)

4.7 Release Generation

Suppose that from the procedure described in the previous section the noisy tree generated is 4.7 with threshold $\theta = 2$. The node L_4, child of L_1 is generated from nothing, just because his corresponding noisycount is equal to 2.

The procedure suggested in order to generate a private release is to traverse the tree in *postorder* and create the trajectories from the leafs on the basis of the noisy counts

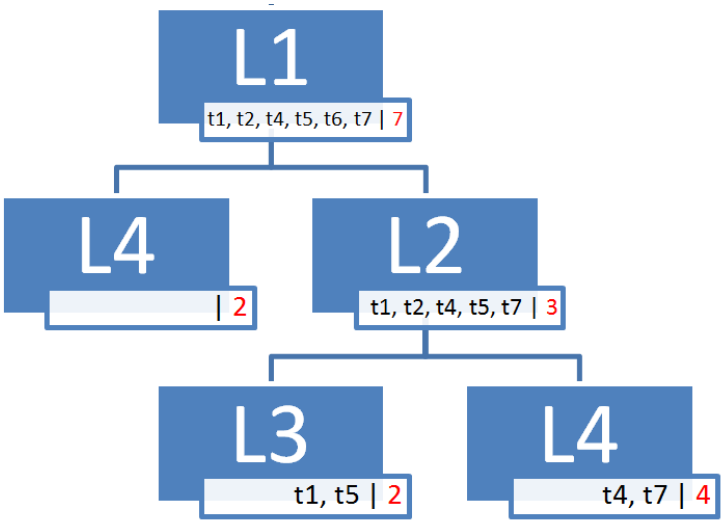


Figure 4.7: Algorithm demonstration (3)

values $c(u)$. Sticking to the example, from the tree it is impossible to derive a meaningful database, because there would be 6 trajectories whose end is $L_1 \rightarrow L_4$ or $L_1 \rightarrow L3$, while L_1 has just tree associated trajectories. A *consistency constraint* can be used to control this issue:

The sum of the noisycounts of the children of a node v must be less than the noisycount value of v.

A possible method to be applied for satisfying this constraint is to perform a post-processing on the noisycounts of the prefix tree that starting from the leafs will equal the noisycount of each father node to the sum of his children nodes, but only if the noisycount of the father is lower than this sum. After this post-processing the tree of the example would look like figure 4.8. There exist better methods in order to achieve this result without generating such a big quantity of additional records, even if they are not directly applicable to this structure. The interested reader can refer to [HRMS10] for further researches.

The released database from the tree 4.8 is represented in table 4.5. Carrying meaningful

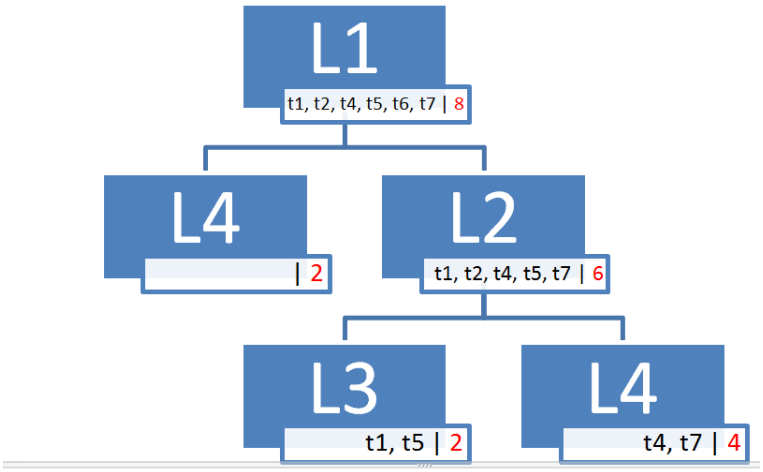


Figure 4.8: Application of the *consistency constraint*

timestamps information to the final release can seem difficult, but in the next chapter, that deals with the possible implementation choices, this issue will be solved.

T_1	$L_1 \rightarrow L_2 \rightarrow L_4$
T_2	$L_1 \rightarrow L_2 \rightarrow L_4$
T_3	$L_1 \rightarrow L_2 \rightarrow L_4$
T_4	$L_1 \rightarrow L_2 \rightarrow L_4$
T_5	$L_1 \rightarrow L_2 \rightarrow L_3$
T_6	$L_1 \rightarrow L_2 \rightarrow L_3$
T_7	$L_1 \rightarrow L_4$

Table 4.5: Final release from example 4.1

4.8 Calculation of the leakage value

In section 4.3 it was proven that in order to produce an ϵ - differentially private release from a raw trajectory database using an algorithm articulated in two successive steps,

the sequential composition 2.2 property holds and therefore the leakage value must be distributed in the two steps such that $\epsilon = \epsilon^P + \epsilon^T$, where ϵ^P is used for the quad-tree spatial decomposition and ϵ^T for the perturbation of trajectories using the noisy prefix tree. In addition, it was demonstrated that the first part of the algorithm works on disjoint subsets and therefore $\epsilon = max(\epsilon_i^P) + \epsilon^T$ for the differential privacy parallel composition property. For the second step, each time a v node from the current tree level is examined, all the possible children u are potential originators on disjoint trajectories, so for every u the level budget can be used in full. Moreover, supposing that for the noisy prefix tree construction the privacy budget ϵ^T is equally partitioned among the h prefix tree levels such that $\overline{\epsilon}^T = \frac{\epsilon^T}{h}$, the final leakage value, for each quad-tree leaf i and each symbolic location j is given by this formula:

$$\epsilon = max(\epsilon_i^P) + h \times max(\overline{\epsilon_j^T})$$

The most obvious choice is to use the highest possible value of ϵ^P and ϵ^T, since higher ϵ means less noise applied. The two values should be chosen during the implementation in order to minimize the queries errors, on the basis of the type of database to be released.

4.9 Complexity analysis

The quad-tree creation algorithm complexity is unbounded in the worst case. In real life moving objects databases, however, the worst case situation is very unlikely, and the complexity in the generic case is $O(n)$, for a set of n measurements. The symbolic locations removal and the timestamps adjustment complexities are linear too. The complexity of the second step of the algorithm is bounded to the Noisy tree construction function. In fact, for each level of the noisy prefix tree, [CFD11] states that the number of nodes to generate approximates $k|T|$, where $k \ll |L|$, if $|T|$ is the number of trajectories

in the database and $|L|$ is the number of symbolic locations generated by the quad-tree. For each level, at most $|D|$ trajectories are added under the u nodes created. The total complexity for the second step hence becomes $O(h|T| \times |L|)$. The entire algorithm complexity is therefore given by the second step of the procedure.

Chapter 5

Implementation

In this chapter, the most interesting implementation choices are presented, from the selection of the most suitable framework to the choice between the different quad-tree creation procedures, to the choices of the structures to be used for keep in the calculator memory the records and the trees presented in the previous chapters.

5.1 Framework

The method was implemented in C# within the PINQ environment [McS09]. This choice was motivated by various reasons. First of all, PINQ offers a set of basic differentially private functions, such as COUNT, SUM, GROUP BY and so on. PINQ perturbation is based on Laplace noise, which is the most utilized by the thesis algorithm. Other reasons for this choice are related to the velocity of C# for parallel computation, and for his Lazy Initialization. The Lazy Loading design pattern in fact actually gives the developer the capability of providing data only when a property is called for. In other words, it is a on-demand loading. Lazy Initialization is therefore an efficient technique that improves drastically the performances when huge amounts of data are loaded. For example, loading around 25 millions of records (see chapter 6 for details) and then performing a noisy count is a matter of few seconds, since the properties of the single objects are not explored. What PINQ requires for the constructor of a PINQueryable is any object implementing the IQueryable<T> interface, for some type T. An easy

example consists on an array of type T. So, what I did was to create a struct type with four fields (trajectory id, x, y, time), allocate an array with as many elements as the amount of entries in the database, and then fill in all the results. Once I have done this, I just constructed a PINQueryable on the base of this array.

5.2 First step: spatial decomposition

5.2.1 Z-ordering and Quad-Tree

In this section it will be discussed how to transform in practice the raw coordinates couples in symbolic locations. The raw database records of figure 4.1 can be represented as in table 5.1, where the coordinates values are expressed in pixels, over a figure dimension of 1021×632 pixels.

T_1	red	$(281, 38) \rightarrow (728, 302) \rightarrow (843, 438)$
T_2	blue	$(455, 47) \rightarrow (621, 91)$
T_3	orange	$(732, 425) \rightarrow (555, 285) \rightarrow (451, 200)$
T_4	violet	$(487, 302) \rightarrow (537, 359) \rightarrow (474, 425)$
T_5	black	$(314, 69) \rightarrow (539, 286) \rightarrow (781, 430)$
T_6	blue	$(923, 451) \rightarrow (982, 366)$
T_7	red	$(386, 145) \rightarrow (558, 377) \rightarrow (359, 544) \rightarrow (124, 382)$
T_8	green	$(827, 488) \rightarrow (741, 414) \rightarrow (488, 217) \rightarrow (356, 109)$

Table 5.1: Representation of the trajectories of figure 4.1 using couples of (x,y) coordinates

In the previous chapter it was explained that a noisy quad-tree representation of the

database would be optimal for the spatial decomposition procedure. An efficient way to build a quad-tree starting from a set of coordinates in a finite space is through a z-ordering of the records. The z-order maps multidimensional data to one dimension and it is calculated by interleaving the binary representations of the points coordinates. It maps quadrants recursively in order NW, NE, SW, SE, as shown in figures. The result then can be simply stored in an array.

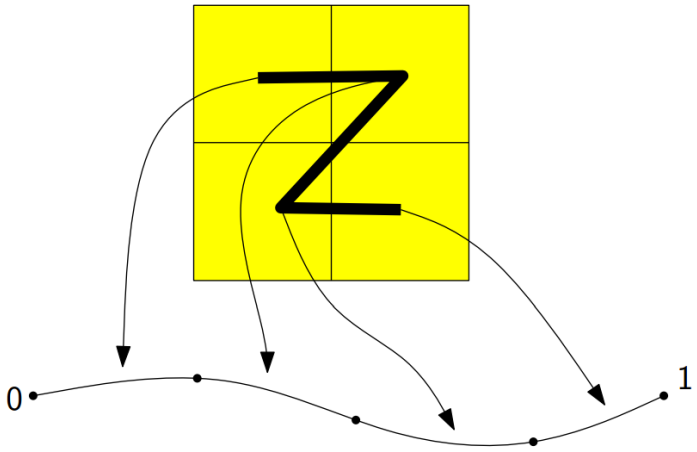


Figure 5.1: Application of the *Z-ordering curve* (1)

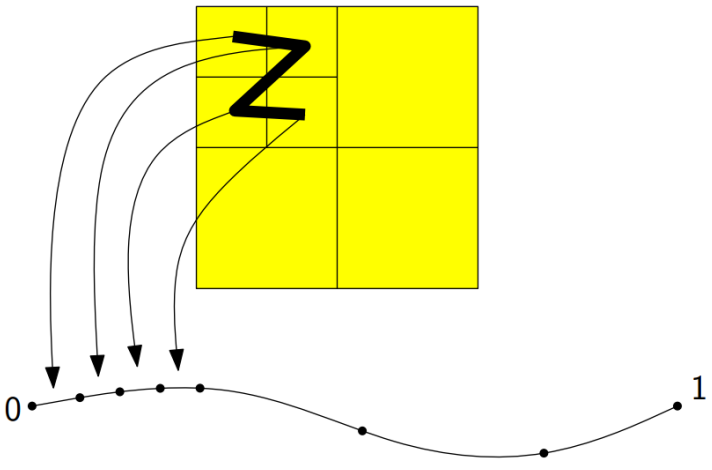


Figure 5.2: Application of the *Z-ordering curve* (2)

5.2.2 Leafs to Locations

Each leaf of a simple linear quad-tree contains a locational key and a value field. In this case, a leaf contains a certain number of points, each one along with its timestamp value and its trajectory ID. In order to pass from the quad-tree representation to a database of trajectories among symbolic locations, it is sufficient to change the coordinates of all the points to the coordinates corresponding to the location dimension. In the example 4.1, the quadrants can be expressed through their upper-left and lower-right coordinates:

- L_1: $[(0,0),(522,394)]$
- L_2: $[(522,0),(1021,394)]$
- L_3: $[(0,394),(522,632)]$
- L_4: $[(522,394),(1021,632)]$

So the records of table 5.1 can be transformed as in table 5.2 :

T_1	$[(0,0),(522,394)] \rightarrow [(522,0),(1021,394)] \rightarrow [(0,394),(522,632)]$
T_2	$[(0,0),(522,394)] \rightarrow [(522,0),(1021,394)]$
T_3	$[(0,394),(522,632)] \rightarrow [(522,0),(1021,394)] \rightarrow [(0,0),(522,394)]$
T_4	$[(0,0),(522,394)] \rightarrow [(522,0),(1021,394)] \rightarrow [(522,394),(1021,632)]$
T_5	$[(0,0),(522,394)] \rightarrow [(522,0),(1021,394)] \rightarrow [(0,394),(522,632)]$
T_6	$[(0,394),(522,632)] \rightarrow [(522,0),(1021,394)]$
T_7	$[(0,0),(522,394)] \rightarrow [(522,0),(1021,394)] \rightarrow [(522,394),(1021,632)] \rightarrow [(0,0),(522,394)]$
T_8	$[(0,394),(522,632)] \rightarrow [(0,394),(522,632)] \rightarrow [(0,0),(522,394)] \rightarrow [(0,0),(522,394)]$

Table 5.2: List of trajectories between symbolic locations expressed as couples of points

5.2.3 Deleting successive duplicates

As anticipated in chapter 4, removing the series of successive locations within trajectories can improve the data usefulness and boost the algorithm performance. Trajectory T_8 in table 5.2 has two series of duplicates: $L_3 \rightarrow L_3$ and $L_1 \rightarrow L_1$. Duplicated can be harmlessly removed from each location examining each database trajectory. If two consecutive points of the trajectory fall within the same location, the average value of the timestamps in calculated and substituted to the first of the points of the sequence. The other points records are then deleted. Trajectory T_8, in example, becomes as shown in table 5.2.3 :

T_8	$[(0, 394), (522, 632)] \rightarrow [(0, 394), (522, 632)] \rightarrow [(0, 0), (522, 394)] \rightarrow [(0, 0), (522, 394)]$

5.3 Second step: trajectory perturbation

For the trajectory perturbation step it is possible to use the algorithm 4.6, shown in the previous chapter. However, there are some parameter that can be accurately chosen in order to boost the speed of the algorithm and improve the utility of the results. The h value taken as input is needed in order to stop the iterations of the algorithm. One can think that the value of h should correspond to the maximum length of the trajectories. Actually, this is not the best choice, since the result would be an extremely noisy tree, with a large number of fake trajectories. Evfimievsky and others [ESAG02] state that it is typically impossible make both useful and private transaction of size 10 and more. This observation holds for trajectory data as well. Long trajectories have small support for count queries, and for sequential pattern mining eliminating the tail of the trajectories does not significantly affect the found patterns, that are usually short (see [CFD11] for more informations). That's why the value h should be limited to be

under the value of 10, and should be bounded to the average length of the trajectory set taken into consideration. Not much is worth to say about the leakage value ϵ for each level of the tree. Since the higher is this value, the higher is the perturbation of the specific level, it seems reasonable to divide equally the ϵ between the levels. Finally, in [CFD11] other boosting techniques are shown, especially in order to avoid to check each possible location in the domain, a really slow procedure. Instead, it is possible to exploit the similarity between the binomial distribution and the pass / not pass test that decides if add a specific node to the tree.

5.3.1 Preservation of the timestamps coherence

During the private release generation, some issues can take place with regards to the consistency constraint introduced in the previous chapter. Specifically, more trajectories are generated than the quantity of available timestamps. However, it is possible to reconstruct meaningful timestamps starting from the ones already present in correspondence to the leafs. Recalling the representation 4.8 and the consequent release 4.5, it is evident the preponderance of the trajectories generated with respect to the ones really represented in the tree, that consequently are associated with an existing timestamp. An possible solution is to compare the timestamps of the children with the timestamps of their father and connect within a new trajectory the most similar timestamps, keeping the chronological order intact. The measurements that after this operation remain without a timestamp should be associated to a slightly higher timestamp than the father node, if it has still available timestamps. If not, a new timestamp can be created from the values of the other timestamps used. The procedure can be even more efficacious if the timestamps are assigned once the release is created, exploring the tree from the root and deciding the timestamps values on the basis of the children nodes at each level. However, this procedure is less efficient than the one previously described.

Chapter 6

Experimental Results

6.1 Dataset

For the experiments the GeoLife dataset [ZXM10] was used. GeoLife trajectories can be downloaded for free, and consists on a folder hierarchy containing text files. Each file represents a trajectory and each row inside the file represents a measurement of the trajectory. This dataset is interesting mainly for the disproportionate distribution of points and trajectories in the space, and for the disparity of transport vehicles adopted, that does not allow to guess an averagely valid time range between different measurements. These two characteristics qualified this dataset as particularly indicated for testing the usefulness of the spatial decomposition algorithm and the timestamp-related procedures presented in section 4.4.3.

6.1.1 Data preprocessing

Some preprocessing steps on the data were performed. Using python (the program is shown in the listing 6.1.1) the hierarchical structure of the trajectories was transformed in a unique text file having the structure 6.1. The database text file was then loaded in the PINQ framework and each record was added as a member of an array of type *Trajectory*.

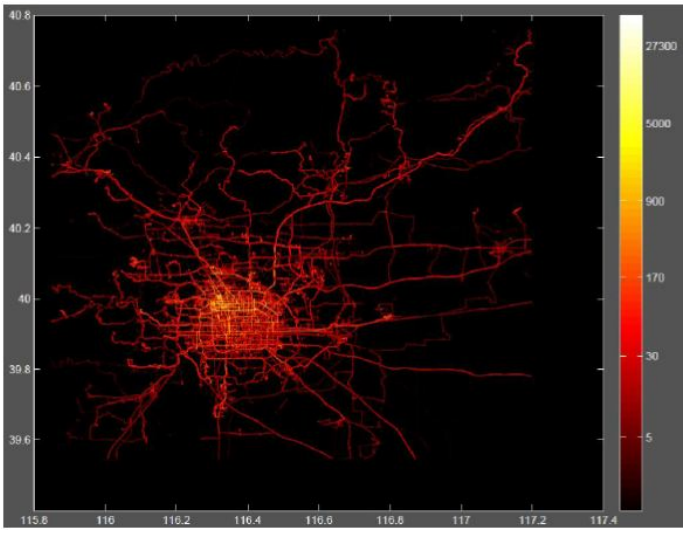


Figure 6.1: Distribution of Geolife trajectories nearby Beijing

Trajectory ID	Longitude coordinate	Latitude coordinate	Date*

* = number of days (with fractional part) that have passed since 12/30/1899

Table 6.1: The structure of the fields in the database file

Listing 6.1: The Phyton conversion program

```
1 file_database = file('database.txt', 'a')
2 count = 0
3 for id_user in os.listdir('Data'):
4   os.chdir('Data/%-s/Trajectory' % (id_user))
5   for id_trajectory in os.listdir('.'):
6      plt_file = file(id_trajectory)
7      for i in xrange(6):
8         plt_file.readline()
9         while True:
10              line = plt_file.readline().strip()
11              if line == '': break
12              count = count+1
13              fields = line.split(',')
14              file_database.write(
15              id_trajectory[:-4] + '|'
16              + fields[0].replace('.',',') + '|'
17              + fields[1].replace('.',',') + '|'
18              + fields[4].replace('.',',') + '\n')
19              file_database.flush()
20     plt_file.close()
21   os.chdir('../../..')
22 file_database.close()
23 file_count = file('count.txt', 'a')
24 file_count.write(str(count))
25 file_count.close()
```

6.2 Utility

In this section, the utility of the perturbed database is measured comparing the results of specified sets of queries on a unperturbed version of database and then on a differentially private database release. Experiments on Count Queries and Frequent Sequential Pattern Mining were performed.

- Count Queries: The utility is measured on the relative error $\frac{|Q(D)-Q(\widetilde{D})|}{max(Q(D),c)}$, where c is a constant.
- Frequent Sequential Pattern: The utility is measured in terms of false positives, true positives and false drops (see reference [AVEG04] for more details on the motivation of this choice).

In order to obtain a meaningful comparisons, the location domain should be the same. Therefore, the comparisons takes place between releases generated on the base of the trajectory database derived from the noisy quad-tree, with removed location duplicates. The *clear* release and the *noisy* release differ between each other in the sense that the latter is produced applying the second step of the algorithm based on the noisy prefix tree, while the former is left at the status obtained after the removal of the sequences of duplicates. Variations on queries outputs are taken in consideration on the basis of the changes for two parameters: the privacy budget ϵ and the noisy prefix tree height h.

6.3 Scalability

The scalability of the method is measured on the basis of the database dimension, e.g. the number of trajectories loaded in the framework in relation to the runtime complexity. All the experiments are performed on an Intel Core i3 2.40 GHz PC with 3GB RAM.

Chapter 7

Future Work

Most part of the researches taken in consideration are extremely recent. The differentially private approach to spatial data is currently in embryonic state, especially if the research regards moving objects databases. However, this work hopefully showed the possible advantages of this approach. Future work can proceed in various ways. In particular, PINQ framework can be used for an interactive implementation of trajectory data differentially private analysis, using the differentially private GROUP BY functions applied to locations, or entire trajectories, or finally entire user entries. From that result one can infer which is the best solution in terms of efficiency and accuracy. On the side of non-interactive approach, there are for sure many possible optimization for the trajectory tree representation and perturbation. My approach is simplified, but a database should handle way more complex data, without such a big loss on informations not trajectory related. Future research goals may be addressed to find the solution of this problem. Sampling as first step of the algorithm of sanitization and pruning as last step were in similar works proven to give satisfying result, but the lack of theoretical proofs on the correctness of the approaches should be fixed. Not only count queries and frequent sequential pattern should be supported by the differential privacy mechanisms, but other different data mining task as well. However, the sensitivity of these query classes must be calculated before proceeding with the research.

Chapter 8
Conclusions

This thesis explored the world of privacy preservation of trajectory data from the novel point of view of differential privacy. Extremely recent researches on the topic obtained encouraging result that motivated this work. However, still does not exist an efficacious approach that, starting from a database of spatio-temporal measurements, is able to product a meaningful and differentially private release of sensitive data. The goal of this thesis was to show that such a procedure is possible, allowing to generate a perturbed database release instead of be forced to use an interactive framework or query restrictions. The main result of this research is an algorithm that produce a differentially private trajectory database release, along with a series of suggestions and possible improvements that can be used in order to boost the algorithm speed and the utility of the release.

Bibliography

[AAG+09] Gennady Andrienko, Natalia Andrienko, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Movement data anonymity through generalization. In *Proceedings of the 2nd SIGSPATIAL ACM GIS 2009 International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '09, pages 27–31, New York, NY, USA, 2009. ACM. 13

[ABN08] Osman Abul, Francesco Bonchi, and Mirco Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *In ICDE*, pages 376–385. IEEE, 2008. x, 13

[AVEG04] R. Agrawal A. V. Evfinievski, R. Srikant and J. Gehrke. Privacy preserving mining of association rules. In *Information Systems, vol. 29, no. 4, pp. 343 - 364*, 2004. 52

[AW89] Nabil R. Adam and John C. Worthmann. Security-control methods for statistical databases: a comparative study. *ACM Comput. Surv.*, 21(4):515–556, December 1989. 7

[CAC12] 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*, CCS '12, pages 638–649, New York, NY, USA, 2012. ACM. 30, 33

[CFD11] Rui Chen, Benjamin C. M. Fung, and Bipin C. Desai. Differentially private trajectory data publication. *CoRR*, abs/1112.2020, 2011. xii, 11, 16, 18, 24, 30, 33, 36, 41, 47, 48

[CPS+11] Graham Cormode, Magda Procopiuc, Entong Shen, Divesh Srivastava, and Ting Yu. Differentially private spatial decompositions. *CoRR*, abs/1103.5170, 2011. 17, 18, 19, 26, 30, 37

[Dal77] T. Dalenius. Towards a Methodology for Statistical Disclosure Control. *Statistik Tidskrift*, 15:429+, 1977. 5

[DG] George Danezis and Seda Gurses. A critical review of 10 years of privacy technology. 26

[DN08] Cynthia Dwork and Moni Naor. On the Difficulties of Disclosure Prevention in Statistical Databases or The Case for Differential Privacy, 2008. 6

[Dwo06] Cynthia Dwork. Differential privacy. In *in ICALP*, pages 1–12. Springer, 2006. 5

[EpKSX96] Martin Ester, Hans peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996. 28

[ESAG02] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. pages 217–228, 2002. 47

[FWCY10] Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010. x, 9, 10, 14

[Ho12] Shen-Shyang Ho. Preserving privacy for moving objects data mining. In *ISI*, pages 135–137, 2012. 12

[HR11] Shen-Shyang Ho and Shuhua Ruan. Differential privacy for location pattern mining. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '11, pages 17–24, New York, NY, USA, 2011. ACM. 27

[HRMS10] Michael Hay, Vibhor Rastogi, Gerome Miklau, and Dan Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, September 2010. 39

[HXO+10] Haibo Hu, Jianliang Xu, Sai Tung On, Jing Du, and Joseph Kee-Yin Ng. Privacy-aware location data publishing. *ACM Trans. Database Syst.*, 35(3):18:1–18:42, July 2010. 13

[IKGB10] Ali Inan, Murat Kantarcioglu, Gabriel Ghinita, and Elisa Bertino. Private record matching using differential privacy. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT '10, pages 123–134, New York, NY, USA, 2010. ACM. 17

[LQS12] Ninghui Li, Wahbeh Qardaji, and Dong Su. On sampling, anonymization, and differential privacy or, k-anonymization meets differential privacy. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 32–33, New York, NY, USA, 2012. ACM. 14

[McS09] Frank D. McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD*

International Conference on Management of data, SIGMOD '09, pages 19–30, New York, NY, USA, 2009. ACM. 7, 8, 11, 30, 43

[MKGV07] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007. 13

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM. 21

[XXY10] Yonghui Xiao, Li Xiong, and Chun Yuan. Differentially private data release through multidimensional partitioning. In *Secure Data Management*, pages 150–168, 2010. 17, 26, 30

[YBLW09] Roman Yarovoy, Francesco Bonchi, Laks V. S. 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*, EDBT '09, pages 72–83, New York, NY, USA, 2009. ACM. 13

[ZXM10] Yu Zheng, Xing Xie, and Wei-Ying Ma. *GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory*. 2010. 32, 49