THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

Framework for Visualizations of Location Privacy for Users

HAILEY ONWELLER
SPRING 2025

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Computer Science
with honors in Cybersecurity Analytics and Operations

Reviewed and approved* by the following:

Anna Squicciarini
Professor of Information Sciences and Technology
Thesis Supervisor

Edward Glantz
Teaching Professor and Assistant Director of Masters Programs, Information Sciences and Technology
Thesis Honors Adviser

* Electronic approvals are on file.

ABSTRACT

The applications we use are constantly collecting data about us from our social networks regarding our locations. Oftentimes we accept the terms and conditions of such applications without thinking much about how this private information can impact us in the future. Previous work on privacy has investigated how we can make datasets obfuscated using mathematical noise, known as differential privacy. This body of research has expanded to be applied to location information resulting in a new concept, referred to as Geo-Indistinguishability. Geo-Indistinguishability research has produced promising results in ensuring user privacy protection and can be very customizable because of the parameter epsilon it takes in. However, because of the mathematical nature of the application, this concept has not been widely adopted yet. Plus, individuals often do not have a good understanding of how data sharing impacts their privacy. In this paper, an application is designed to serve as a tool to help users understand (and eventually deploy) how Geo-Indistinguishability works, and to choose parameters for privacy that work for their use cases. In this paper, three user use cases are identified which are an end user, a developer and a stakeholder along with two cities, New York City and San Francisco, to explain how each would benefit from such an application to explain privacy, as well as how to use Geo-Indistinguishability. Results obtained from these use cases show that all users benefit from such an application. This application has the potential to be expanded and better implemented into everyday technologies to explain privacy, and to help users make informed decisions about the data they share.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

In completing this thesis, I would like to give special thanks to Anna Squicciarini and Primal Pappachan. I have been working with Professor Squicciarini and Professor Pappachan since my first year at Penn State. It is through both their instruction and guidance that I have learned how to perform research and learned about privacy. They have guided me through my research experience at Penn State and have built my passion for designing technology that helps individuals understand privacy to protect their data. The skills that they have taught me are skills that I will take into my future career and research endeavors.

Next, I would like to thank the Millennium Scholars Program for their guidance and support throughout my time at Penn State. The Millennium Scholars Program staff including Dr. Amy Freeman and my advisors, Errol Wizda, Georjanne Rosa, Latisha Franklin and Katelyn Jones have always encouraged me and guided me through my classes, internships and research experiences. Their support and encouragement have given me the skills to overcome challenges and confidence that I know will guide me through my career.

In addition, I would like to thank my family and friends for supporting me along the way. Their constant support and guidance have helped me to succeed and are the reason that I am where I am today.

Finally, I would like to thank Penn State for the opportunities that have shaped me into the person that I am today. Through the research and leadership experiences I have had the privilege of taking part in, I have grown into a confident researcher and leader who will take the lessons and experience that I have had into the future.

Chapter 1

Introduction

Privacy is of the top concerns of users of data intensive applications. As applications continue to use our data, individuals grow increasingly weary of how their data is being collected and shared. Location based services often ask individuals to share their own private data, but many users do not understand what this means and how they can protect their data. Data privacy attempts to ensure that individuals maintain control over their own data by deciding how companies collect, store and use data [8]. Many companies are even required by law to disclose privacy statements; however, these statements are often long and hard for everyday users to understand.

Often the idea of a "privacy paradox" is used to explain how users may seem concerned about their privacy but often fail to take action to protect their information. Research has shown that this idea requires more research and understanding to offer solutions to this paradox. One promising solution has been to provide cues and education to help users understand their privacy more [2]. Often users do not quite understand what privacy means, how their data is being used and implications of that data usage and would benefit from instruction on how to properly set their preferences.

Data privacy techniques have been thoroughly researched and are referred to as privacy-enhancing technologies (PETs). PETs allow researchers to use and share private information in their research. The goal of a PET is to ensure high utility from the data in the data set while protecting the user's data. One way that a PET can work is through data obfuscation. This means

that the tool ensures privacy over the data set by adding noise to the entries [11]. One such tool is Differential Privacy (DP). Differential privacy is the idea of adding noise to inputs to obscure data within a dataset. This means that when running a query on a dataset, the output cannot be traced back to the dataset itself. DP ensures that the output of a request does not reveal personal information about users using math [1]. DP attempts to use enough obfuscation so that one entry in the data set does not impact the output. DP is very promising for protecting personal information such as health or demographic datasets. One problem with differential privacy is that it is still being studied theoretically, and few organizations understand how to implement it or use it properly [11].

Differential privacy has been extended to location-based services through a method known as Geo-Indistinguishability. This is the idea of adding noise to locations to obscure an original location as a point or in a dataset. Location privacy refers to protecting user data from location-based services. Often, users do not know if their devices and applications are collecting their location which can threaten their privacy as well as safety [7]. Geo-indistinguishability differs from differential privacy in that it applies the methods from DP onto a user's location (typically represented in longitude and latitude) [9]. It introduces the idea that within a radius, a user's privacy is guaranteed. As is true with DP, Geo-indistinguishability injects math into this data which can be confusing for users to fully understand. The parameters for this algorithm can be difficult for users to understand and work with, making geo-indistinguishability an abstract idea for many individuals. There has been extensive research on how to use DP and Geo-Ind but there is less research on how to explain these functions to non-technical users [11].

One such solution to helping users understand privacy better is by visualizing the parameters and data. DP has little research on these types of visualizations while Geo-Ind has

close to no research on these visualizations. As is common with privacy, location privacy techniques come with tradeoffs in utility and privacy that can be hard for different users to understand. Thus, in this paper we seek to create visualizations to describe privacy to a variety of users which include technical users, non-technical users and stakeholders.

Research Objective

This paper intends to provide a framework which is designed to help users of different backgrounds and experience levels to create a location privacy input, based on their comfort level. In exploring previous methods for visualizing differential privacy, we will apply them to the setting of location privacy and suggest new methods. Geo-Ind differs from DP in the number and types of parameters needed, thus we will augment existing DP systems as well as adding more explanations to help users fully understand how their location data is used.

Chapter 2

Literature Review

Differential Privacy

Differential Privacy (DP) is a method of obscuring data by adding controlled random noise [4]. Currently, DP is widely accepted and used in privacy applications using data. More technically, differential privacy is the property of a randomized algorithm such that if there are two databases D and D' and they vary by at most one row, the probability that a randomized function applied to the original database is less than or equal to e^{ε} times the probability of that function, applied to the neighboring database.

Given a random function K, we have ε-differential privacy if for all data-sets D and D' that differ at most one row with output S and all S within K we have [4]:

$$\Pr[K(D) \in S] \leq e^{\varepsilon} * \Pr\left[K(D')\epsilon S\right]$$

Figure 1: Definition of Differential Privacy

Differential privacy typically utilizes the Laplacian mechanism to add such noise to a query.

Location Based Services (LBS)

Location Based Services are any application that utilizes a user's location to perform a function and provide a target location [12]. Examples include any application that would ask a user to share their location. According to a study from 2012, 43% of US adults own a

smartphone and 76% of them use LBSs. Various uses of LBS include maps, navigation systems, weather services, ridesharing, delivery of goods and services and more [1].

Geo-Indistinguishability

Given a Euclidean space X, function A satisfies ε-geo-indistinguishability if for all $x_1, x_2 \in X$ and $Z \in \mathbb{Z}$ there is:

$$A(x_1)(Z) \leq e^{\varepsilon * d(x_1, x_2)} * A(x_2)(Z)$$

Figure 2: Definition of Geo-Indistinguishability

Geo-Indistinguishability is one application/ extension of DP applied to location data [17]. Geo-Indistinguishability is essentially differential privacy when the distance between points is 1. This form of DP adds controlled noise to a location to obtain an approximate location. This guarantees that within a radius, r, a theoretical attacker has a hard time telling the difference between a perturbed location and a real location. There has been research on algorithms to perturb a user's location that serves as the basis for this research.

One such approach requires a technique called location perturbation that considers both Geo-Indistinguishability and location semantics. Location semantics include details like time of being at that location or popularity of that location. The first proposed algorithm takes a point and epsilon and then repeatedly adds noise to the point to find a radius within which other locations can be the perturbed point. The algorithm works by iterating through a user defined number of iterations to randomly select points and use the lambert function to generate a total distance traveled which accounts for the privacy parameter epsilon. Then the algorithm averages the total distance with the number of iterations and returns that as the radius.

Next the location semantics are considered to narrow down the data set. In this algorithm, it takes in the point and radius and first finds all points that fit within the radius. It removes locations that fail to have as many users as desired by the user. Then it calculates the cosine similarity between each point and the real location and averages the results. Then it compares each cosine similarity with the average and removes those that are larger (meaning that they are too different). After removing the values, it returns an optimized area.

Finally, the last algorithm takes in the optimized area and minimizes the quality loss of choosing a point using an optimization algorithm. The algorithm creates a zero matrix to represent the perturbations and probabilities of a point being chosen. Then it defines the constraints as seen in Figure 3. The constraints declare that the sum of perturbations must satisfy Geo-Indistinguishability as defined in Figure 2, each should be larger than 0 and sum to one across rows. After running a minimization software, the algorithm returns a random point that minimizes the quality loss [17].

$$Minimize: QL(K, \pi, d)$$

$$Subject\ to \begin{cases} k_{x,z} \leq e^{\varepsilon * d(x,x')} * k_{x',z} & \forall x, x', z \in X \\ k_{x,z} \geq 0 & \forall x, z \in X \\ \sum_{z \in X} k_{x,z} = 1 & \forall x \in X \end{cases}$$

Figure 3: Linear Programming Function and Constraints

Additional work has demonstrated the importance of customizable location privacy. Research on Customizable and Robust Geo-Indistinguishability has shown how linear programming and optimization functions based on location type, privacy and location can improve utility of Geo-Indistinguishability [12]. This also aids in user understanding of privacy, through this demo paper users benefit from interacting firsthand with privacy to understand privacy and trade-offs between utility and privacy.

This series of algorithms and work on customizable location privacy is currently considered the best use of Geo-Indistinguishability, but its process seems very obscure and complicated to many users. Thus, this research seeks to better explain this process.

Types of Users

When thinking about privacy, there are many stakeholders to take into account. These users have different needs and preferences when it comes to location data. For example, an end user might be more concerned with their data being shared than the developer who is accessing the data. And even in this example, one end user might be more inclined than another to share their real location based on the use of the application. Research done by Columbia University has identified the key users who should be considered when explaining or utilizing differential privacy [3]. The idea behind implementing Geo-Indistinguishability in applications is that each user would only receive the information needed to complete their job. Thus, it is interesting to consider how differential privacy can be and should be explained to the different users who interact with it.

Users include:

- **End-users**:
    - These are users whose data is used by the application and are assumed to have little to no mathematical knowledge. These users should receive information on the technical guarantees of differential privacy and are more likely to share information when information leaks are less likely to occur. In addition, users should be given guidance on DP parameters by explaining tradeoffs in the specific

use cases. Suggestions for developers would be taking into consideration user preferences for privacy and use cases when setting parameters.

- **Engineers and Technical Managers**:
    - Engineers and technical managers are individuals who implement and test DP systems to determine if they meet the privacy needs of their organization. They must make the decisions regarding managing privacy parameters based on use cases. They are assumed to have a high level of technical knowledge but not necessarily in the field of differential privacy. Thus, they need guidance on setting parameters and in testing applications.
- **Data curators**
    - Data curators are those who assemble data sets and facilitate access. These individuals understand when access should or should not be granted to users. When privacy is integrated into datasets, the job of a curator can be more complicated. They need to be concerned about privacy budgets, levels and access by multiple users. They thus need more education on DP and examples.
- **Executives**
    - These are the individuals who decide whether their organization would benefit from DP. They often are more concerned with improving the performance of their organization in terms of revenue and customer satisfaction as opposed to implementing cutting edge technology. They don't benefit from knowledge on privacy-accuracy trade-offs as much as user input such as user trust, better data

collection, and protection against leaks. They want to do what makes their customers trust them and what policy makers may require by law.

- **Regulators**
    - Regulators include lawyers and policymakers. Lawyers are primarily concerned with whether new privacy tools violate any privacy laws. Given that there is little guidance on the interpretation of such laws, challenges in understanding DP arise from the mathematical nature of DP and privacy laws that change in geographical regions. Policymakers are primarily concerned with augmenting or regulating privacy laws. Given the new and unstandardized nature of DP development, there are not necessarily 'best practices' of DP usage, which can make the job of a policymaker hard. They might not understand the nuances of DP, such as that the epsilon value is dependent on the given use case, which makes regulation harder.

Visualizing Differential Privacy

There has been prior research on visualizing differential privacy [10]. These visualizations are often about static and general datasets. Static datasets are datasets where the data entered in a database will not be altered or added to. In these cases, visualizations for differential privacy can be more straightforward. In datasets that refer to numeric entries, the visualizations can also be more straightforward. Prior research has investigated how to incorporate odds-based explanations for privacy with graphs to explain privacy [10]. These are examples that tell users the odds of their information being leaked. This work has understood the different users who access and interact with data including data curators, providers and

consumers, and aims to visualize DP to benefit each user. More specifically, the work aims at helping individual users determine their own epsilon for their use case. The work focuses on statistical ideas to explain the risk of leaking data, given a user's choice. The work relies on quantile dot plots, confidence intervals, hypothetical outcome plots and cumulative density functions to explain the probability of the outcomes of the differentially private equation. These techniques are examples of odds-based explanations, since odds and probabilities are used to explain the privacy parameter in terms of the risk of revealing information. This has proven successful in these types of data, but location data is 2-dimensional and is much harder to explain in these terms [18]. Explaining 2-dimensional data often requires the use of maps and other visualizations.

Visualizing Geo-Indistinguishability

The problem with visualizing location privacy is that not only does one have to explain differential privacy to a user, but they also must explain longitudinal and latitudinal numbers to the user. Thus, previous visualizations for differential privacy cannot be applied to location privacy. Thus, there is a need to explain differential privacy in a new way.

Some research has attempted to explain differential privacy applied to location privacy. Research has shown how researchers can visualize privacy by taking user location data points and adding noise (via a Laplace mechanism) and plotting these noisy points via a heatmap [18]. These visualizations often require knowledge of mathematical topics such as logarithmic scales or the Laplace mechanism to understand what the maps are trying to show and are thus not appropriate for many types of users.

In this prior research, researchers explain Geo-Indistinguishability but fail to explain how the user's choices impact the result of the geo-indistinguishable mechanism. Thus, we aim to create an application to allow users to understand how their choices and preferences can impact Geo-Indistinguishability to understand how to use privacy for their particular use cases.

Chapter 3

Methodology

Objectives

The objective of this paper is to develop a framework and baseline for visualizing location privacy. We have seen how earlier research has attempted to visualize differential privacy for decision makers but has failed to be applied to Geo-Indistinguishability. Thus, this paper will apply differential privacy visualization techniques to location privacy applications to allow users to make their own decisions about privacy. This will allow users to balance the privacy-utility tradeoff when considering location-based applications' access to location. By walking through this dashboard, users will be able to modify parameters to their comfort and to apply this to their own use case. We will also describe the value of visualizations like these to the different user types.

Assumptions, Dependencies and Constraints

Assumptions:

- This application will demonstrate to users of varying skillsets the tradeoff between privacy and utility
- The radius will show how the epsilon impacts privacy and the pins on the map will show where a user's privacy and utility is at, at each algorithm
- Radius is inversely proportional to epsilon as seen in Figure 2
- In more densely populated cities, more individuals will check in giving more check-in data for these cities

- Similarly, in more densely populated areas, more choices of locations will return from each algorithm
- This algorithm would be better suited to serve in densely populated cities

Dependencies:

- This application relies heavily on the SNAP Gowalla dataset [6]
- The application is built using Flask, IBM CPLEX and Google Maps [5,13,14]

Constraints:

- The application can only use locations from check-in information in the SNAP Gowalla dataset, meaning it is dependent on a high number of individuals checking in at each location
- Privacy in this application is applied to an individual's privacy
- The application also assumes that a user is only using it within the privacy budget they are allotted

Implementation and Development

This application was built using Python, Flask, HTML, CSS and Google Maps API along with the Stanford SNAP Gowalla Dataset [6, 14,13]. On the backend, Python was used with the Gowalla dataset to program and run the algorithms with real world data. In addition, IBM CPLEX was used to run the linear algebra optimization in Algorithm 3 [13]. The implementation of this followed from the work done on developing the perturbation algorithms [17] and from the work done on Customizable and Robust Geo-Indistinguishability [12]. On the front end, the application generates a map for each algorithm with information on how the algorithm works, and there are fields for the user to customize the algorithms. For Algorithm 1, there is the ability

to select an epsilon to populate the radius and pins on the map, and for Algorithm 2 there is a drop down to select the upper and lower limit of people at a given location. The front end and back-end connect using Flask.

Algorithm 1 was implemented as:

Algorithm 1: Finding the Perturbation Area

Input: Real point x_0, Number of iterations N and Privacy Parameter ε
Output: Area $\{x_0, \text{radius}\}$

for 1 to N
  θ = uniform random number between [0,π]
  ρ = uniform random number between [0,1]
  Assert that $\lambda = W_{-1}\left(\frac{\rho-1}{e}\right) + 1$ is real
  Radius $= \left(-\frac{1}{\varepsilon}\right) * \lambda$
  $x.lat = x_0.lat + radius * \cos(\theta)$
  $x.lon = x_0.lon + radius * \sin(\theta)$
  $total\_distance = total\_distance$ + Euclidean distance between x and x_0
Radius $= \frac{total\_distance}{N}$
Area $\{x_0, \text{radius}\}$

Figure 4: Algorithm 1 Implementation

Algorithm 2 was implemented as:

Algorithm 2: Optimizing the Perturbation Area

Input: Real point x_0, Area, Lower Limit of people num, user data df
Output: Optimized Area2 $\{x_0, \text{radius}\}$ as a data frame with correct data

initialize the cos_sum = 0
for all entries in data frame
  filter out locations outside of the radius and center point
  filter out locations with fewer than num entries
  add up the cosine similarity between the center point and all points in the data frame
calculate the average cosine similarity
for each location:
  check that each cosine similarity is greater than the average cosine similarity
Add the remaining locations to Area2

Figure 5: Algorithm 2 Implementation

Finally, Algorithm 3 was implemented using the equation in Figure as:

Algorithm 3: Returning the Perturbed Point

Input: Number of entries, Area2, Prior Probabilities
Output: Perturbed Point, P

create a nxn matrix and fill it with the distance between each entry
Using IBM CPLEX:
    create the perturbation matrix K
    add each constraint:
        for i in n:
            for j in n:
                if i≠j:
                    for z in n
                        add constraint $K[i,z] \leq e^{\varepsilon * dist[i,j]} * K[j,z]$
        for i in n:
            sum of each row must be 1
    solve optimizer for K
generate l = uniform random number [0,1]
P = the point with the closest probability in the matrix to l

Figure 6: Algorithm 3 Implementation

Parameters

The main parameters in this application are real location, epsilon and lower limit on number of users at a location. The real location is assumed to be given by the user. The epsilon is given by the user with guidance from the application description. The epsilon impacts the user's privacy with a smaller epsilon meaning more privacy/obfuscation. The lower limit on users at a location is another parameter given by the user. The lower limit allows a user to describe whether they want the application to consider how popular the other locations are (the more populus giving more privacy). This parameter aids in demonstrating how location semantics impact privacy [1]. Other parameters include the dataset that the applications run on, and the radius generated by the algorithms. The dataset used in this application is the Stanford SNAP Gowalla dataset [6].

Visualizing Epsilon and Radius

To visualize the privacy, the application utilizes the relationship between epsilon and radius. The visualization of the epsilon as the radius allows a user to look at the region in which they are protected by Geo-Indistinguishability. The radius is decided in algorithm 1 and stays consistent throughout the application. There is an inversely proportional relationship between radius and epsilon which is demonstrated by the choice in epsilon and the resulting radius. The application allows the user to play with these values to see how they change.

Utility and Privacy

This application can provide a customizable utility and privacy tradeoff. The application allows users to visualize their utility and privacy based on the parameters that they choose. When a user selects a larger epsilon, they can see how the radius returned is larger and there are more points to choose from with similar probabilities to the original point. This shows a high level of privacy because of the options of points to choose from but also means potentially poorer utility. The utility is worse because the possible points are further away, thus have the potential for worse utility. This can be visualized from the application by changing the epsilon and the number of people at each location. A user can see how there are less points, and therefore the result is less private.

Scalability

This application can be further expanded to benefit a wider variety of users. The location of the application can be changed, as well as the dataset that it is built off, to provide better utility and examples for individual use cases. The application could even be imbedded in applications that request privacy so that a user can better understand how to give their data.

Demonstration of Application

To use the application, a real location is chosen (in this example the real location is in Austin, Texas). The user then reads the algorithm 1 description that explains how the algorithm works and suggestions of the parameters. This statement is there to help give guidance to users who are new to privacy algorithms. They are then able to select an epsilon value from a dropdown. Clicking “Run Algorithm 1” runs the algorithm and prints the radius and updates the map with possible values. The red point is the real location, and the blue points are the values from the dataset.

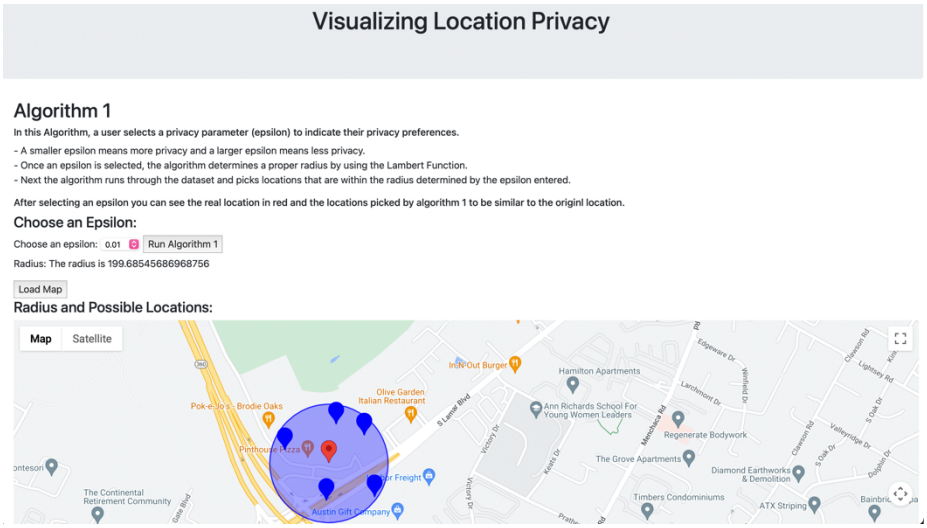

Figure 7: Demo of Algorithm 1

After Algorithm 1 runs, the user moves to Algorithm 2 where they see a similar explanation of the algorithm and similar suggestions. They then can pick a lower limit on the number of users at each location. After selecting from the drop down, they can select to run algorithm 2 which populates the map with the remaining location choices.

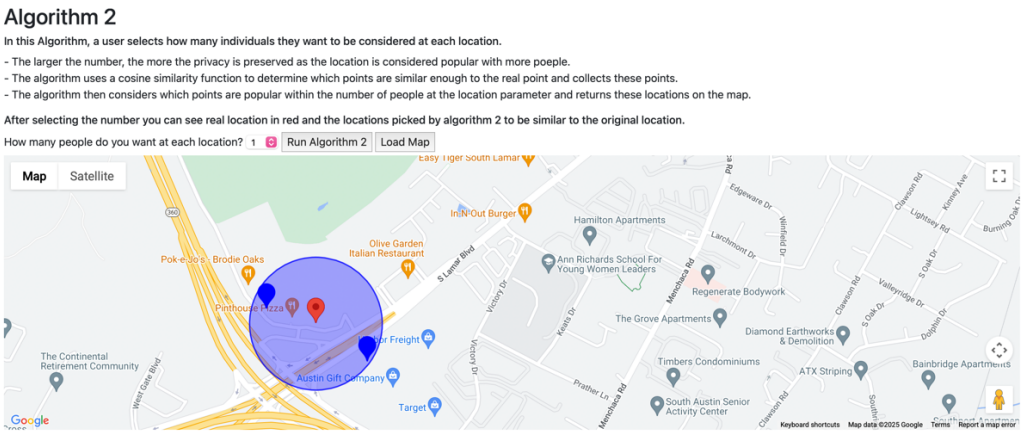

Figure 8: Demo of Algorithm 2

Finally, the user gets to Algorithm 3 and sees a similar description of the algorithm with suggestions. A user clicks run algorithm 3 and load map to see the results of the application. The final point left is the perturbed location.

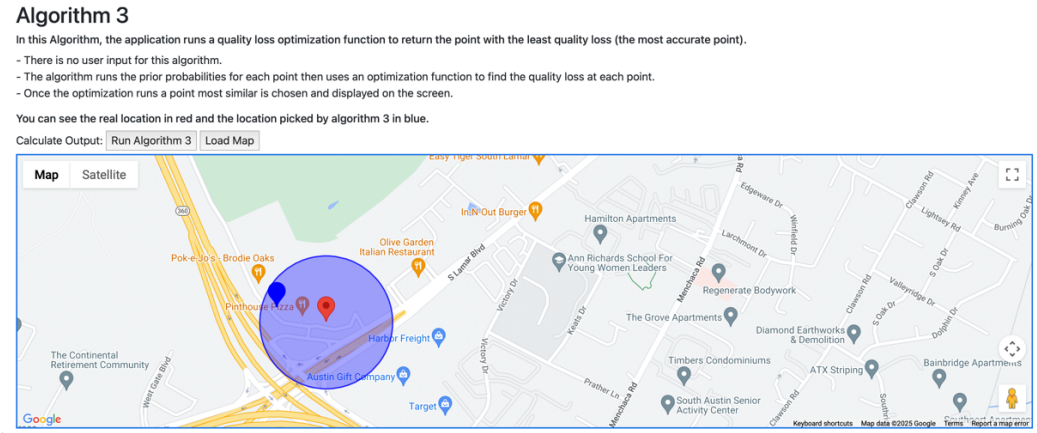

Figure 9: Demo of Algorithm 3

Use Case

In the context of Geo-Indistinguishability, the users and their roles in the system are different from the users in differential privacy. Thus, in creating an application to visualize location privacy the users can walk through the application and decide their privacy preferences by seeing the results of each algorithm. Three individual user types can be defined as:

- User 1 - End User
    - This is an end user, they might be asked by an application to share their location, and they need to weigh their comfort and privacy with their utility needs
    - User needs to use a rideshare application, wants one close to home but not too far
- User 2 – Developer
    - This is a developer; they need to check the results of a privacy algorithm. Their priority might be utilization and implementing the algorithm correctly.
    - Developer is designing an application that asks for location data, prioritizes utility over privacy
- User 3 – Stakeholder
    - This is a stakeholder, they are deciding whether their application should use Geo-Indistinguishability, they weigh the trust from users against utility.
    - Deciding whether to invest in geo-indistinguishable algorithms for an application, their choice depends mainly on location and utility

Based on the application and the 3 use cases, two cities of varying sizes were selected to compare results from the application. The cities chosen were New York City and San Francisco.

New York city is defined as a large city while San Francisco is defined as a medium sized city. According to 2023 US census data, New York City is the largest city in the US and San Francisco ranks as the 17th largest city [15]. By choosing these two cities with sufficient data from SNAP Gowalla dataset, data was collected on the sizes and results returned from the application to explain how these three user types would benefit from the application, as well as assessing the limitations of the applications.

Chapter 4

Results

Upon running this application with the users in mind and for each of the two cities the following results were obtained. First multiple radius values were tested to obtain radius values and number of locations within that radius for each city. To calculate these values, an area was created geographically around the area and the center of the city was chosen as the "real location". Then Algorithm 1 was run to calculate the radius and number of locations left for each city. New York City worked well with larger privacy parameters such as 0.01 and 0.02. This means that within smaller radii, the utility of the application worked. This makes sense because in a more densely populated city there are more likely to be people checking in within a smaller radius. With San Francisco, the application only returned values starting at an epsilon of 0.001 which returned a radius of 2007.1 meters. This shows that Geo-Indistinguishability within the city only works for radii larger than 2007.

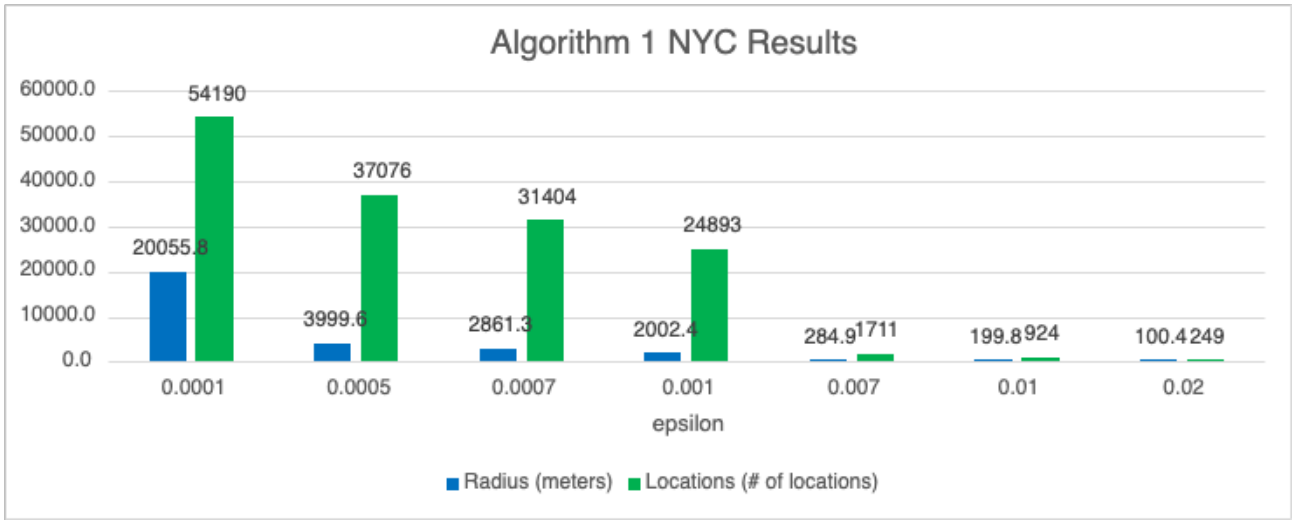

Figure 10: Algorithm 1 NYC Results

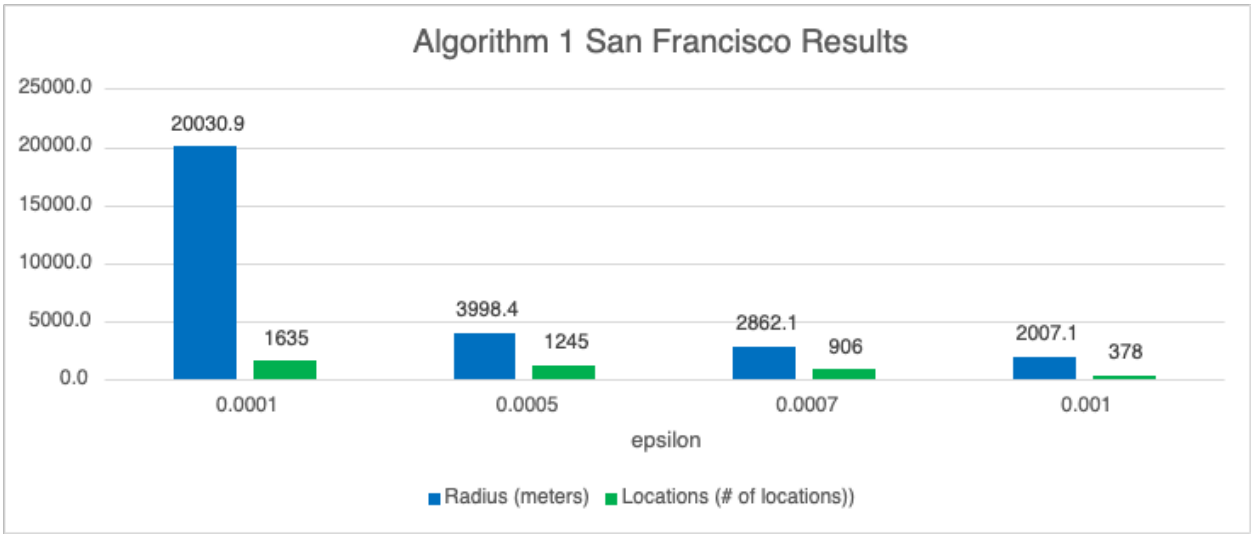

Figure 11: Algorithm 1 San Francisco Results

Based on these results, User 1 might see that both offer strong privacy guarantees, however a bigger city like New York City allows for better utility because of the number of locations within a smaller radius. For this user, the usefulness of this application would depend on their particular use of the application and whether they would be willing to use data farther from their real location. User 2 might use this to verify that Geo-Indistinguishability is working as they would expect, as the trends verify the algorithms. They might also see higher utility with cities like New York City but also see that the algorithm works in other cities. Finally, User 3 would see the difference between a more densely populated city, and they might only invest in cities of this size and density. They might see merit in smaller cities but for the most part they would likely care more about larger cities.

Similar trends are seen when running Algorithm 2. In Algorithm 2 a user can set a lower limit on the number of people at each location. This aids privacy by letting a user pick popular locations so that it is more deceiving to an adversary. The results from this follow very similarly to Algorithm 1. The trends are similar in that as the number of people at each location increases and the epsilon values increase the number of locations returned decreases. With New York City, more locations are returned than with San Francisco

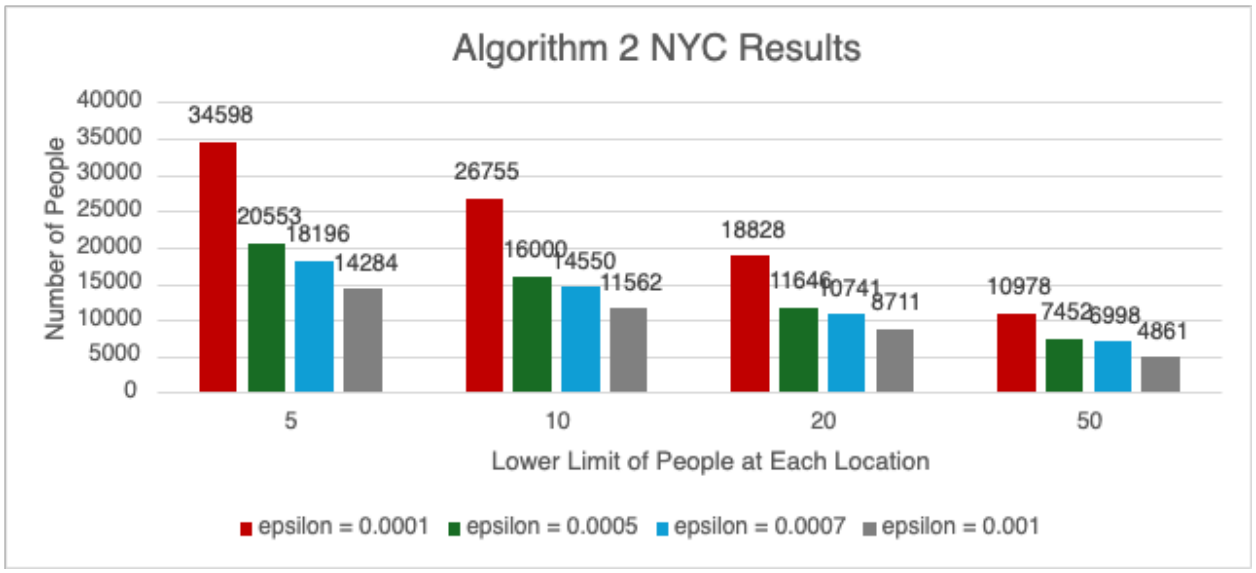

Figure 12: Algorithm 2 NYC Results

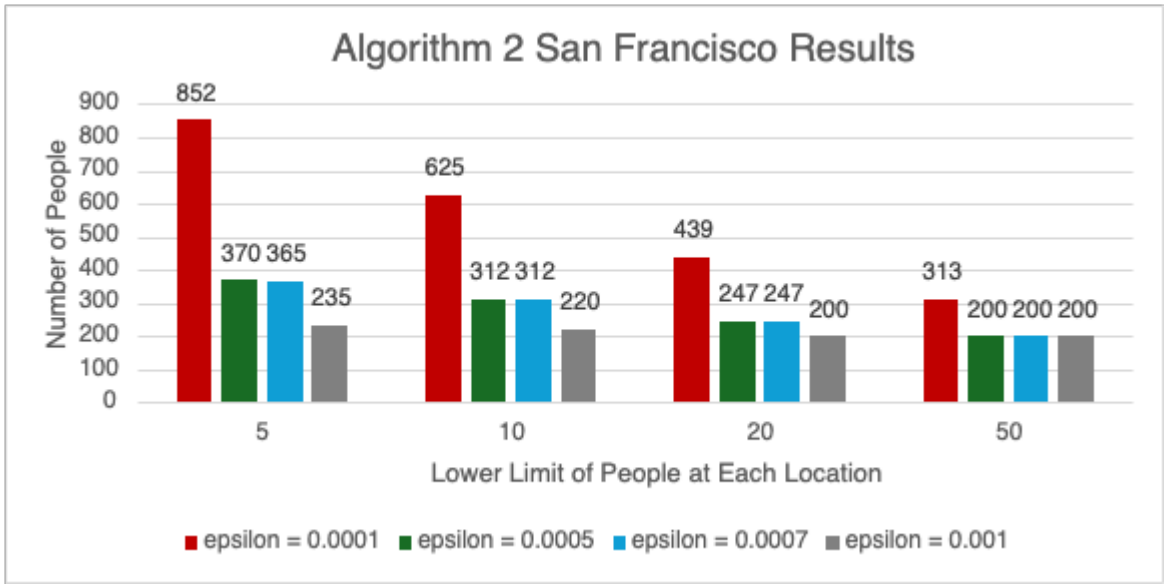

Figure 13: Algorithm 2 San Francisco Results

These results explain how different users can utilize the application for their needs. For User 1 who cares about their individual privacy, they might find that larger cities offer more obfuscation for their needs but in some cases, it might not matter as much. For applications such as weather or traffic, this might not matter as much as applications such as ride shares or deliveries. User 2 would mainly use the application to verify their programming and to see how the application fits into their use case. They might

care about utility but mainly care about verification of the algorithms. Finally, as a stakeholder, User 3 cares most about deciding whether to invest resources into implementing Geo-Indistinguishability. They might see that Geo-Indistinguishability is promising in larger locations and has potential in smaller locations.

Chapter 5

Conclusion

This application demonstrates the importance of teaching users about how their data is being used and how to use tools that can help them make better informed decisions about sharing their data. The tool serves as a framework for understanding privacy from different user perspectives. Users such as end users, developers and stakeholders all can use the tool to explore how Geo-Indistinguishability can be applied to their use cases. Results show that the application gives results that help explain to the different users how to make better informed choices about sharing data or investing in the implementation of Geo-Indistinguishability in applications. Privacy tools such as differential privacy have been successfully implemented in applications, and Geo-Indistinguishability has offered promising results. Frameworks like this better explain how to use these tools and how they can improve user trust and utility.

Limitations

This application serves as a framework for explaining visualizations of privacy. The tool is mainly for those new to privacy and Geo-Indistinguishability. Because of this some features are left out for simplicity. In addition, because the tool relies on check-in data, some of the data might not be as representative of real situations in which someone might use this type of privacy.

One main limitation of this application is the dataset used. The data set used is a comprehensive set of locations and real user check-ins. This offers realistic data but might not provide the best data available. This was seen when analyzing cities of different population densities. Because some cities are less populous, they might not have been represented as well.

In addition, because this is a visualization tool, the algorithms do not account for repetition and therefore a privacy budget is not explained and utilized. In the future, the application would benefit from an explanation of the privacy budget and potential privacy leakage.

Future Work

Future work in this area would include expanding this application to be used in more use cases and as an actual tool for deciding privacy parameters and implementation of Geo-Indistinguishability. As seen with the 3 users and their results, multiple users have a stake in privacy levels and would benefit from being able to interact with and explore Geo-Indistinguishability. To better serve these users, offering this application as a tool to explain privacy would be very beneficial. Thus, this tool could be expanded and personalized to individual use cases.

In addition, testing the application in different regions or countries could provide another interesting experiment. Different datasets of real-world data might provide more interesting results. Also, more comprehensive user testing would be beneficial to ensure that the application works with individual user types as expected. Feedback from end users, and from corporate employees and executives would offer valuable insight into how to make privacy more accessible to individuals.

This application has the potential to give more access to understanding privacy tradeoffs and how private data such as location data can be used and protected. By expanding this work, there is potential to make applications more equitable and fairer.

Appendix

Index.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<script src="/static/maps.js" defer></script>
<title>Visualizing Location Privacy</title>
<style>
#googleMap {
width: 100%;
height: 400px;
}
#googleMap2 {
width: 100%;
height: 400px;
}
#googleMap3 {
width: 100%;
height: 400px;
}
</style>
<script src="https://maps.googleapis.com/maps/api/js?key=&libraries=places,marker"></script>

</head>
<body>
<div class="jumbotron text-center">
<h1>Visualizing Location Privacy</h1>
</div>
<div class="itembox one mt-4 mb-3">
<div class="col mx-2" >
<div class="header1"><h2>Algorithm 1</h2></div>
<div class="slidecontainer">
<h6>In this Algorithm, a user selects a privacy parameter (epsilon) to indicate their privacy preferences.</h6>
<p> - A smaller epsilon means more privacy and a larger epsilon means less privacy.
<br> - Once an epsilon is selected, the algorithm determines a proper radius by using the Lambert Function.
<br> - Next the algorithm runs through the dataset and picks locations that are within the radius determined by the
epsilon entered.
</p>
<h6> After selecting an epsilon you can see the real location in red and the locations picked by algorithm 1 to be
similar to the originl location.</h6>
<h4>Choose an Epsilon:</h4>
<label for="epsilon">Choose an epsilon:</label>
<select id="dropdown">
```

```
<option value="0.01">0.01</option>
<option value="0.02">0.02</option>
<option value="0.007">0.007</option>
</select>
<button onclick="sendDropdown1()">Run Algorithm 1</button>
<p>Radius: <span id="result"></span></p>
<script>
function sendDropdown1(val) {
const dropdownValue = document.getElementById("dropdown").value;
fetch('/process1', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ number: dropdownValue }),
})
.then(response=>response.json())
.then(data=>{document.getElementById('result').innerText=data.result;})
.catch(error=>console.error('Error:',error));
}
</script>
</div>

<button onclick="loadMap1()">Load Map</button>
<h4>Radius and Possible Locations:</h4>
<div id="googleMap"></div>

</div>
</div>

<div class="itembox two mt-4 mb-3">
<div class="col mx-2">
<div class="header2"><h2>Algorithm 2</h2></div>
<h6>In this Algorithm, a user selects how many individuals they want to be considered at each location.</h6>
<p> - The larger the number, the more the privacy is preserved as the location is considered popular with more people.
<br> - The algorithm uses a cosine similarity function to determine which points are similar enough to the real point and collects these points.
<br> - The algorithm then considers which points are popular within the number of people at the location parameter and returns these locations on the map.
</p>
<h6> After selecting the number you can see real location in red and the locations picked by algorithm 2 to be similar to the original location.</h6>
<label for="numberppl">How many people do you want at each location?</label>
<select name="numberppl" id="numberppl">
<option value="1">1</option>
<option value="10">10</option>
<option value="20">20</option>
<option value="50">50</option>
</select>
<button onclick="sendDropdown2()">Run Algorithm 2</button>
<script>
function sendDropdown2(val) {
// Update the displayed selected value
const dropdownValue = document.getElementById("numberppl").value;
fetch('/process2', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ number: dropdownValue }),
})
.then(response=>response.json())
.catch(error=>console.error('Error:',error));
```

```
}
</script>
<button onclick="loadMap2()">Load Map</button>
<div id="googleMap2">
</div>
</div>
</div>
</div>
<div class="itembox three mt-4 mb-3">
<div class="col mx-2">
<div class="header3"><h2>Algorithm 3</h2></div>
<h6>In this Algorithm, the application runs a quality loss optimization function to return the point with the least quality loss (the most accurate point).</h6>
<p> - There is no user input for this algorithm.
<br> - The algorithm runs the prior probabilities for each point then uses an optimization function to find the quality loss at each point.
<br> - Once the optimization runs a point most similar is chosen and displayed on the screen.
</p>
<h6> You can see the real location in red and the location picked by algorithm 3 in blue.</h6>
<label>Calculate Output:</label>
<button onclick="sendDropdown3()">Run Algorithm 3</button>
<script>
function sendDropdown3(val) {
const dropdownValue = document.getElementById("dropdown");
fetch('/process3', {
method: 'POST',
headers: { 'Content-Type': 'application/json' },
body: JSON.stringify({ number: dropdownValue }),
})
.then(response=>response.json())
.then(data=>{document.getElementById('result').innerText=data.result;})
.catch(error=>console.error('Error:',error));
}
</script>
<button onclick="loadMap3()">Load Map</button>
<div id="googleMap3"></div>
</div>
</div>
```

Maps.js

```
// Setting Algorithm 1 map
function loadMap1() {
  fetch("/static/data1.json")
  .then(response => response.json())
  .then(data => {
    var center_lat = data.center_lat;
    var center_lon = data.center_lon;
    var rad = data.radius;
    var latitudes = data.latitudes || [];
    var longitudes = data.longitudes || [];
    var locs = {'lat':latitudes,'lon':longitudes}
    console.log(rad)
    const temp = [1,2,3]
```

```
  if (isNaN(center_lat) || isNaN(center_lon)) {
      console.error("Invalid coordinates in JSON data");
      return;
    }

initMap(center_lat,center_lon,rad,locs,latitudes,longitudes);

})
.catch(error => console.error("Error loading JSON:", error));



function initMap(center_lat,center_lon,rad,locs,latitudes,longitudes){
  const { Map } = google.maps.importLibrary("maps");
  const { AdvancedMarkerElement } = google.maps.importLibrary("marker");
    const mapProp = {
      center: new google.maps.LatLng(center_lat, center_lon),
      zoom: 10,
      mapId: 'map_1'
    };
    const map = new google.maps.Map(document.getElementById("googleMap"), mapProp);

    const circle = new google.maps.Circle({
      map: map,
      center: new google.maps.LatLng(center_lat, center_lon), // Center of the circle
      radius: rad, // Radius in meters
      fillColor: 'blue', // Circle fill color
      fillOpacity: 0.35, // Circle opacity
      strokeColor: 'blue', // Circle border color
      strokeOpacity: 0.8, // Circle border opacity
      strokeWeight: 2 // Circle border weight
    });


      // console.log(locs);
      for (let i = 0; i < latitudes.length; i++) {
        const pin = new google.maps.marker.PinElement({
          background: "blue",
          borderColor: "blue",
          glyph: '',
        });
        const position = { lat: latitudes[i], lng: longitudes[i] };

        // Create a marker
        new google.maps.marker.AdvancedMarkerElement({
          position: position,
          map: map,
          title: `Marker ${i + 1}`,
          content: pin.element,
        });
      }
      const marker = new google.maps.marker.AdvancedMarkerElement({
        map: map,
        position: { lat: center_lat, lng: center_lon},
      });
    // Create the map
  }
}
```

```
// map 2
function loadMap2(){
  fetch("/static/data2.json")
  .then(response => response.json())
  .then(data => {
    var center_lat = parseFloat(data.center_lat);
    var center_lon = parseFloat(data.center_lon);
    var rad = parseFloat(data.radius);
    var latitudes = data.latitudes || [];
    var longitudes = data.longitudes || [];
    var locs = {'lat':latitudes,'lon':longitudes}
  //   console.log(typeof(locs));
  //   console.log(longitudes.length)
    if (isNaN(center_lat) || isNaN(center_lon)) {
         console.error("Invalid coordinates in JSON data");
         return;
      }

  initMap2(center_lat,center_lon,rad,locs,latitudes,longitudes);
  })
  .catch(error => console.error("Error loading JSON:", error));
  function initMap2(center_lat,center_lon,rad,locs,latitudes,longitudes){
     const { Map } = google.maps.importLibrary("maps");
     const { AdvancedMarkerElement } = google.maps.importLibrary("marker");
      const mapProp = {
         center: new google.maps.LatLng(center_lat, center_lon),
         zoom: 10,
         mapId: "map_2"
      };
      const map2 = new google.maps.Map(document.getElementById("googleMap2"), mapProp);

      const circle = new google.maps.Circle({
         map: map2,
         center: new google.maps.LatLng(center_lat, center_lon), // Center of the circle
         radius: rad, // Radius in meters
         fillColor: 'blue', // Circle fill color
         fillOpacity: 0.35, // Circle opacity
         strokeColor: 'blue', // Circle border color
         strokeOpacity: 0.8, // Circle border opacity
         strokeWeight: 2 // Circle border weight
      });

        //   console.log(locs);
         for (let i = 0; i < latitudes.length; i++) {
            const position = { lat: latitudes[i], lng: longitudes[i] };
            const pin = new google.maps.marker.PinElement({
              background: "blue",
              borderColor: "blue",
              glyph: '',
            });
            // Create a marker
            new google.maps.marker.AdvancedMarkerElement({
               position: position,
               map: map2,
               title: `Marker ${i + 1}`,
               content: pin.element,
            });
         }
```

```
          const marker = new google.maps.marker.AdvancedMarkerElement({
            map: map2,
            position: { lat: center_lat, lng: center_lon},
          });
        // Create the map
    }
}

function loadMap3(){
  fetch("/static/data3.json")
  .then(response => response.json())
  .then(data => {
    var center_lat = parseFloat(data.center_lat);
    var center_lon = parseFloat(data.center_lon);
    var rad = parseFloat(data.radius);
    var pt_lat = parseFloat(data.pt_lat);
    var pt_lon = parseFloat(data.pt_lon);

    console.log(pt_lat);

     // Call function to initialize the map after data is loaded
  initMap3(center_lat,center_lon,rad,pt_lat,pt_lon);
  })

  function initMap3(center_lat,center_lon,rad,pt_lat,pt_lon){
    const { Map } = google.maps.importLibrary("maps");
    // const { AdvancedMarkerElement } = google.maps.importLibrary("marker");
      const mapProp = {
          center: new google.maps.LatLng(center_lat, center_lon),
          zoom: 10,
          mapId: "map_3"
      };
      const map3 = new google.maps.Map(document.getElementById("googleMap3"), mapProp);

      const circle = new google.maps.Circle({
          map: map3,
          center: new google.maps.LatLng(center_lat, center_lon), // Center of the circle
          radius: rad, // Radius in meters
          fillColor: 'blue', // Circle fill color
          fillOpacity: 0.35, // Circle opacity
          strokeColor: 'blue', // Circle border color
          strokeOpacity: 0.8, // Circle border opacity
          strokeWeight: 2, // Circle border weight
      });

      const marker = new google.maps.marker.AdvancedMarkerElement({
        map: map3,
        position: {lat: center_lat, lng: center_lon},
      });
      console.log(pt_lat);
      console.log(typeof(pt_lat));
      const pin = new google.maps.marker.PinElement({
              background: "blue",
              borderColor: "blue",
              glyph: '',
            });
      new google.maps.marker.AdvancedMarkerElement({

        position: {lat: pt_lat, lng: pt_lon},
```

```
        map: map3,
        content: pin.element,
        });
    //enter real location
  }
}
```

Funcs.py

```
import numpy as np
import math
from scipy.special import lambertw
import pandas as pd
import json
from geopy import distance
class area:
    r = 0
    center = 0

class point:
    lon = 0
    lat = 0
# Set up for algorithm 1:
x_0 = point() #real location
x_0.lat = 30.233651595938202
x_0.lon = -97.79147033337382
epsilon = 0.007 #privacy parameter
N = 100000 #number of iterations

def alg_1(x_0,epsilon, N):
    from geopy import distance
    total_dist = 0
    for i in range(1,N):
        theta = np.random.uniform(0,math.pi)
        rho = np.random.uniform(0,1)
        lambert = lambertw((rho-1)/math.e,-1)+1
        assert np.isreal(lambert), "Lambert is imaginary"
        lambert = float(lambert)
        r = (-1/epsilon)*lambert
        x = point()
        x.lat = x_0.lat + r* math.cos(theta)
        x.lon = x_0.lon + r* math.sin(theta)
        total_dist = total_dist + np.sqrt(((x_0.lat-x.lat))**2+((x_0.lon-x.lon))**2)
    radius = total_dist/N
    global area_1
    area_1 = area()
    area_1.radius = radius
    area_1.center = x_0
    print(f'radius: {area_1.radius}')
    return area_1

def fix_dists_1(area_1):
    locations = []
    df = pd.read_csv('~/desktop/thesis/loc-gowalla_totalCheckins (1).csv',sep="\t",names=["User",
"Time","latitude",'longitude','id'])
```

```
            for i,row in df[:50000].iterrows():
                dist = distance.distance((float(area_1.center.lat),float(area_1.center.lon)),(row['latitude'],row['longitude'])).meters
                if float(dist) <= area_1.radius:
                    print(dist,row['latitude'],row['longitude'])
                    locations.append(row)
            t = pd.DataFrame(locations).reset_index()
            loc_lat = t['latitude'].to_list()
            loc_lon = t['longitude'].to_list()
            locs = []
            for x,y in zip(loc_lat, loc_lon):
                locs.append([x,y])
            print(f'Area 1 radius in fun is {area_1.radius}')
            data = {
            "center_lat":float(area_1.center.lat),
            "center_lon":area_1.center.lon,
            "radius": area_1.radius,
            "locations": locs[:1000],
            'latitudes':loc_lat[:1000],
            'longitudes':loc_lon[:1000]
            }
            with open("./static/data1.json", "w") as outfile:
                json.dump(data, outfile)
            new_df = pd.DataFrame(locations, columns=df.columns).reset_index()
            location_counts = new_df['id'].value_counts()
            return [new_df,locations]



        def cos_sim(x_0lat,x_0lon,xn_lat,xn_lon):
            orig = [x_0lat,x_0lon]
            new = [xn_lat,xn_lon]
            dot_prod = np.dot(orig, new)
            norm_orig = np.linalg.norm(orig)
            norm_new = np.linalg.norm(new)
            return dot_prod / (norm_orig*norm_new)

        def alg_2(area_1,locations,new_df,num):
            area_2 = area_1
            location_num = len(locations)
            lower_lim = num #temporary lower limit on people at a location
            opt_locations = []
            indices_to_drop = []
            location_counts = new_df['id'].value_counts()
            # Step 2: Get location IDs that appear at least 10 times
            valid_locations = location_counts[location_counts >= lower_lim].index
            filtered_df = new_df[new_df['id'].isin(valid_locations)].reset_index()
            print(filtered_df)

            cos_sum = 0
            n_df = []
            #find cos average for filtered dataframe
            for i in range(len(filtered_df)):
                cos_sum +=
cos_sim(area_2.center.lat,area_2.center.lon,float(filtered_df.iloc[i]['latitude']),float(filtered_df.iloc[i]['longitude']))
            try:
                cos_avg = cos_sum / len(filtered_df)
            except ZeroDivisionError:
                cos_avg = 0
                print("Error: Division by zero")
```

```
    if cos_avg==0:
        data = {
        "center_lat":float(area_1.center.lat),
        "center_lon":area_1.center.lon,
        "radius": area_1.radius,
        'latitudes':[],
        'longitudes':[]
        }
        with open("./static/data2.json", "w") as outfile:
            json.dump(data, outfile)
        id_counts = 0
        filtered_df = []
        return [id_counts,filtered_df]
    for i in filtered_df.index:
        cos_sim_i = cos_sim(area_2.center.lat,area_2.center.lon,filtered_df.iloc[i]['latitude'],filtered_df.iloc[i]['longitude'])
        if cos_sim_i > cos_avg:
            n_df.append(filtered_df.iloc[i])
            indices_to_drop.append(i)
            print(filtered_df.iloc[i]['latitude'])
    filtered_df = pd.DataFrame(n_df)
    print("Filtered DataFrame:")
    print(filtered_df)
    temp_df = filtered_df
    print(temp_df)
    id_counts = filtered_df['id'].value_counts()
    print(id_counts)
    data = {
    "center_lat":float(area_1.center.lat),
    "center_lon":area_1.center.lon,
    "radius": area_1.radius,
    'latitudes':list(filtered_df['latitude'].unique()),
    'longitudes':list(filtered_df['longitude'].unique())
    }
    with open("./static/data2.json", "w") as outfile:
        json.dump(data, outfile)
    return [id_counts,filtered_df]


def prior_prob(id_counts,area_1,filtered_df):
    if type(id_counts) == int:
        return [0,[],0]
    else:
        percents = []
        ids = id_counts.index.tolist()
        nums = id_counts.tolist()
        print(nums)
        total = 0
        for i in nums:
            total += i
        print(total)
        for i in nums:
            percents.append(i/total)
        print(percents)
        sum_percents = 0
        for i in percents:
            sum_percents +=i

        print(ids)
        print(sum_percents)
        longs = []
```

```
            lats = []
            dists = []
            real_x = area_1.center
            for i in ids:
                lo = filtered_df.loc[filtered_df['id'] == i, 'longitude'].iloc[0]
                la = filtered_df.loc[filtered_df['id'] == i, 'latitude'].iloc[0]
                longs.append(lo)
                lats.append(la)
                dist = distance.distance((float(real_x.lat),float(real_x.lon)),(la,lo)).meters

                dists.append(dist)

            print(dists)
            data = {"id":ids,"prior": percents, "dist from real":dists, "longitude":longs,"latitude":lats}
            opt_df = pd.DataFrame(data)
            return [1,opt_df,sum_percents]

        def alg_3(code,opt_df,sum_percents):
            if code == 0:
                print
                data = {
                "center_lat":float(area_1.center.lat),
                "center_lon":area_1.center.lon,
                "radius": area_1.radius,
                'pt_lat':[],
                'pt_lon':[]
                }
                with open("./static/data3.json", "w") as outfile:
                    json.dump(data, outfile)
                return
            else:
                from docplex.mp.model import Model
                sum = 0
                n = len(opt_df)
                distances = np.zeros((n, n))
                for i in range(n):
                    for j in range(n):
                        distances[i][j] =
float(distance.distance((opt_df.iloc[i]['latitude'],opt_df.iloc[i]['longitude']),(opt_df.iloc[j]['latitude'],opt_df.iloc[j]['longitude'])).m
eters)

                mdl = Model(name="Alg 3")
                K = mdl.continuous_var_matrix(n, n, lb=1e-6, ub=1, name="K")

                quality_loss = mdl.sum(float(opt_df.iloc[i]['prior']) * K[i, j] * float(distances[i, j])
                               for i in range(n) for j in range(n))
                #replace float with double
                exp_values = np.exp(epsilon * distances.astype(float))
                for i in range(n):
                    for j in range(n):
                        if i != j:
                            for z in range(n):
                                mdl.add_constraint(K[i, z] <= float(exp_values[i,j]) * K[j, z])

                for i in range(n):
                    mdl.add_constraint(mdl.sum(K[i, j] for j in range(n)) == sum_percents)
                print(mdl.lp_string)
                solution = mdl.solve()
```

```
    # Output results
    if solution:
        K_matrix = []
        print("Optimal perturbation matrix K:")
        for i in range(n):
            row = [solution[K[i, j]] for j in range(n)]
            K_matrix.append(row)
            print(row)
        K_matrix = np.array(K_matrix)
        prob = np.random.uniform(0,1)
        differences = np.abs(K_matrix - prob)
        closest_index = np.unravel_index(np.argmin(differences), K_matrix.shape)
        closest_value = K_matrix[closest_index]
        print("Minimum Quality Loss:", solution.objective_value)
    else:
        print("No solution found")
    final_lat = opt_df.iloc[int(closest_index[1])]['latitude']
    final_lon = opt_df.iloc[int(closest_index[1])]['longitude']
    print(f'final lat: {prob}{closest_index},{closest_value}')
    print(f'final lat: {final_lat},{final_lon}')

    data = {
        "center_lat":float(area_1.center.lat),
        "center_lon":area_1.center.lon,
        "radius": area_1.radius,
        'pt_lat':final_lat,
        'pt_lon':final_lon
    }
    with open("./static/data3.json", "w") as outfile:
        json.dump(data, outfile)
    # flask needs:
    #     "center_lat":float(area_1.center.lat)
    #     "center_lon":area_1.center.lon
    #     "radius": area_1.radius
    #     'pt_lat':30.234664
    #     'pt_lon':-97.793158
```

App.py

```
from flask import Flask, render_template, request, jsonify
import funcs
from funcs import point,area
global result,res_2, res_3
app = Flask(__name__)
print("hi")
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/process1', methods=['POST'])
def process1():
    data = request.json
    epsilon = float(data.get('number'))
    x_0 = point()
    x_0.lat = 30.2359091167
```

```
    x_0.lon = -97.7951395833
    N = 100000
    global result
    global res_2
    result = funcs.alg_1(x_0,epsilon,N)
    res_2 = funcs.fix_dists_1(result)
    return jsonify({"result": f"The radius is {result.radius}"})

@app.route('/process2', methods=['POST'])
def process2():
    data2 = request.json

    number = int(data2.get('number'))
    global res_3
    res_3 = funcs.alg_2(result,res_2[0],res_2[0],number)

    return jsonify({"result": f"The square of {number} is {res_3}"})

@app.route('/process3', methods=['POST'])
def process3():
    data3 = request.json
    res_4 = funcs.prior_prob(res_3[0],result,res_3[1])
    final = funcs.alg_3(res_4[0],res_4[1],res_4[2])

    return jsonify({"result": f"The square is {final}"})


if __name__ == '__main__':
    app.run(debug=True)
```

BIBLIOGRAPHY

[1] Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., & Palamidessi, C. (2013). Geo-Indistinguishability: differential privacy for location-based systems. Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, 901–914. Presented at the Berlin, Germany. doi:10.1145/2508859.2516735

[2] Barth, S., & de Jong, M. D. T. (2017). The privacy paradox – Investigating discrepancies between expressed privacy concerns and actual online behavior – A systematic literature review. *Telematics and Informatics*, *34*(7), 1038–1058. doi:10.1016/j.tele.2017.04.013

[3] Cummings, R., Desfontaines, D., Evans, D., Geambasu, R., Huang, Y., Jagielski, M., … Zhang, W. (2024). Advancing Differential Privacy: Where We Are Now and Future Directions for Real-World Deployment. arXiv [Cs.CR]. Retrieved from http://arxiv.org/abs/2304.06929

[4] Dwork, C. (2011, January 1). A firm foundation for Private Data Analysis. Communications of the ACM. https://dl.acm.org/doi/10.1145/1866739.1866758
Jiang, H., Li, J., Zhao, P., Zeng, F., Xiao, Z., & Iyengar, A. (2021). Location Privacy-preserving Mechanisms in Location-based Services: A Comprehensive Survey. ACM Comput. Surv., 54(1). doi:10.1145/3423165

[5] *Get Started*. Google. (n.d.) https://developers.google.com/maps/get-started

[6] *Gowalla*. SNAP. (n.d.). https://snap.stanford.edu/data/loc-gowalla.html

[7] Konstantinos Chatzikokolakis, Catuscia Palamidessi, Marco Stronati. Geo-indistinguishability: A Principled Approach to Location Privacy. ICDCIT 2015 - Proceedings of the 11th International Conference on Distributed Computing and Internet Technology, Feb 2015, Bhubaneswar, India. pp.49- 72, ff10.1007/978-3-319-14977-6_4ff. ffhal-01114241f

[8] Kosinski, M., & Forrest, A. (2025a, March 17). *What is data privacy?*. IBM. https://www.ibm.com/think/topics/data-privacy#:~:text=Data%20privacy%2C%20also%20called%20%22information,to%20actively%20manage%20their%20data.

[9] Mendes, R., Vilela, J.P. (2022). Geo-Indistinguishability. In: Jajodia, S., Samarati, P., Yung, M. (eds) Encyclopedia of Cryptography, Security and Privacy. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27739-9_1535-1

[10] Nanayakkara, P., Bater, J., He, X., Hullman, J., & Rogers, J. (2022). Visualizing Privacy-Utility Trade-Offs in Differentially Private Data Releases. *CoRR*, *abs/2201.05964*. Retrieved from https://arxiv.org/abs/2201.05964

[11] OECD (2023), “Emerging privacy-enhancing technologies: Current regulatory and policy approaches”, *OECD Digital Economy Papers*, No. 351, OECD Publishing, Paris, https://doi.org/10.1787/bf121be4-en.

[12] Pappachan, P., Hunsur Manjunath, V. S., Qiu, C., Squicciarini, A., & Onweller, H. (2023). CORGI: An interactive framework for Customizable and Robust Location Obfuscation. 2023 IEEE 39th International Conference on Data Engineering (ICDE), 3667–3670. doi:10.1109/ICDE55515.2023.00294

[13] *Quickstart*. Flask Documentation (3.1.x). (n.d.). https://flask.palletsprojects.com/en/stable/quickstart/

[14] *Setting up the python API OF CPLEX*. IBM. (n.d.). https://www.ibm.com/docs/en/icos/22.1.0?topic=cplex-setting-up-python-api

[15] Wikimedia Foundation. (2025, February 20). *List of United States cities by population*. Wikipedia. https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population#cite_note-PopEstCities-3

[16] Wood, Alexandra, Micah Altman, Kobbi Nissim, and Salil Vadhan. 2020. “Designing Access with Differential Privacy.” In: Cole, Dhaliwal, Sautmann, and Vilhuber (eds), *Handbook on Using Administrative Data for Research and Evidence-based Policy*. Accessed at https://admindatahandbook.mit.edu/book/v1.0/diffpriv.html on 2025-03-23.

[17] Yan, Y., Xu, F., Mahmood, A. et al. Perturb and optimize users’ location privacy using Geo- Indistinguishability and location semantics. Sci Rep 12, 20445 (2022). https://doi.org/10.1038/s41598-022-24893-0

[18] Zhang, D., Miklau, G., Hay, M., & O’Connor, B. (n.d.). Challenges of visualizing differentially private data. https://people.cs.umass.edu/~dzhang/dp_viz_challenges.pdf