M.Sc. Engg. (CSE) Thesis

A PRIVACY-ENHANCED APPROACH FOR PLANNING SAFE ROUTES WITH CROWDSOURCED DATA AND COMPUTATION

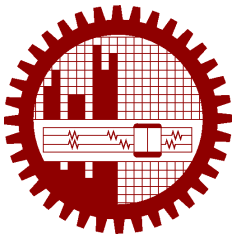
Submitted by

Fariha Tabassum Islam

1018052029

Supervised by

Dr. Tanzima Hashem



Submitted to

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

June 2021

Candidate's Declaration

I, do, hereby, certify that the work presented in this thesis, titled, "A PRIVACY-ENHANCED APPROACH FOR PLANNING SAFE ROUTES WITH CROWDSOURCED DATA AND COMPUTATION", is the outcome of the investigation and research carried out by me under the supervision of Dr. Tanzima Hashem, Professor, Department of CSE, BUET.

I also declare that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Fariha Tabassum Islam

1018052029

The thesis titled "**A PRIVACY-ENHANCED APPROACH FOR PLANNING SAFE ROUTES WITH CROWDSOURCED DATA AND COMPUTATION**", submitted by Fariha Tabassum Islam, Student ID 1018052029, Session October 2018, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents on June 28, 2021.

Board of Examiners

1. _____

   Dr. Tanzima Hashem — Chairman
   Professor — (Supervisor)
   Department of CSE, BUET, Dhaka

2. _____

   Dr. A.K.M. Ashikur Rahman — Member
   Professor and Head — (Ex-Officio)
   Department of CSE, BUET, Dhaka

3. _____

   Dr. A.B.M. Alim Al Islam — Member
   Professor
   Department of CSE, BUET, Dhaka

4. _____

   Mr. Sukarna Barua — Member
   Assistant Professor
   Department of CSE, BUET, Dhaka

5. _____

   Dr. Mohammad Nurul Huda — Member
   Professor — (External)
   Department of CSE
   United International University (UIU), Dhaka

Acknowledgement

Foremost, I am eternally grateful to Allah SWT. the most gracious and the most merciful.

I would like to express my sincerest gratitude, deep respect and admiration to my supervisor, Professor Dr. Tanzima Hashem, who taught me how to do research. I have been extremely lucky to have her as my supervisor. I am indebted to her for her constant supervision, clear guidance, great encouragement and motivation and all the efforts she put on me. This thesis would never be completed without her guidance, directions and help in every aspect. She spent numerous valuable hours of her busy schedule addressing the problems I faced promptly and correcting the manuscript. Her immense wisdom and experience have helped me in all the time of my research life.

I am grateful to Dr. Rifat Shahriyar for his guidance and valuable suggestions and for reviewing the draft. I also want to express my gratitude to the members of my thesis committee for their valuable suggestions. I thank Dr. A.K.M. Ashikur Rahman, Dr. A.B.M. Alim Al Islam, Mr. Sukarna Barua, and specially the external member Dr. Mohammad Nurul Huda.

I gratefully acknowledge the fellowship I received from the ICT Division, Bangladesh (56.00.0000.028.33.108.18). I thank the Department of CSE, BUET, for providing resources during the thesis work.

I remain ever grateful to my beloved parents and my beloved husband Tareq who always inspired me and provided unlimited support in every success and failure.

Dhaka
June 28, 2021

Fariha Tabassum Islam
1018052029

Contents

List of Figures

List of Tables

Abstract

In this thesis, we introduce a novel safe route planning problem and develop an efficient solution to ensure the travelers' safety on roads. Though few research attempts have been made in this regard, all of them assume that people share their sensitive travel experiences with a centralized entity for finding the safest routes, which is not ideal in practice for privacy reasons. As a result, existing systems cannot provide safest routes with high accuracy due to the lack of data related to travel experiences. Furthermore, existing works formulate the safe route planning query in ways that do not meet a traveler's need for safe travel on roads. Our approach finds the safest routes within a user-specified distance threshold based on the personalized travel experience of the knowledgeable crowd without involving any centralized computation. We develop a privacy preserving model to quantify the travel experience of a user into personalized safety scores. Our algorithms for finding the safest route further enhance user privacy by minimizing the exposure of personalized safety scores with others. Specifically, we develop two efficient algorithms, direct and iterative, to evaluate the safest route queries. The direct and the iterative algorithms offer trade-offs among the computation overhead, communication cost and privacy. We run extensive experiments using three real datasets to show the effectiveness and efficiency of our approach. Our iterative algorithm finds the safest route with 50% less exposure of personalized safety scores compared to that of direct algorithm. On the other hand, the computation overhead and the communication overhead for the direct algorithm are lower compared to those of the iterative algorithm. Although the direct algorithm is faster than the iterative algorithm, both of our algorithms take less than a second to process a query. Our experiments also show that lack of data for privacy issues can reduce the answer quality significantly. Our safe route planning system ensures the quality of the safest routes by protecting the privacy of the travel experiences of the users.

Chapter 1

Introduction

Location-based services, especially the journey planners like Google or Bing Maps, have become an integral part of our life for moving on roads with convenience. Existing services mainly consider distance and traffic while planning the routes for the travelers. However, the shortest or the fastest route is not always the best choice. While travelling on roads, people face many inconveniences like theft and pick-pocketing; women face harassment like eve-teasing and unwanted physical touch. People would like to travel a little bit longer on a safer route that avoids those inconveniences. During the outbreak of an infectious disease like COVID-19, a pedestrian may want to avoid a crowded road to keep herself safe from infection. The chance for a virus to exist on the air and road surface increases with the increase of the number of pedestrians. Road safety in such a pandemic period can be measured based on the level of road crowdedness. To meet the traveler's need on roads, we introduce a safe route planner that finds the safest route (SR) between a source-destination pair within a distance constraint.

The data needed for computing the SRs may come from official crime reports and personal travel experiences of the crowd. The latter is more valuable than the former one due to its recency and adequacy. However, travel experiences are often sensitive and private data, and people, especially women, do not feel comfortable sharing their detailed travel experiences and harassment data with others [1]. These factors have inspired us to develop a privacy-enhanced safe route planning system by not sharing the personalized travel experiences of the crowd with a centralized entity or others.

Our approach personalizes the safety score (SS) of a user's travel experience (both safe and unsafe) with respect to the user's travel pattern. If two users face the same crime on two roads, then these roads may have different SSs considering the frequency and recency of the users'

visits on those roads. Ignoring the personal travel pattern of the users would reduce the quality of data and the accuracy of the query answer. We develop a model to quantify a user's travel experience for a visited area into a *personalized safety score* (pSS) based on frequency and recency of the user's visits, location, time, and type of inconveniences faced. Users store their pSSs of their known areas on their own device or any other private storage (e.g., cloud storage) and use them to find the SRs for others. The transformation of a user's travel experience into a pSS is a one-way mapping. From the revealed pSS of a user, it is not possible to pinpoint the type of incident faced by the user. It may only allow an adversary to infer high-level information on a user's travel experience (e.g., a user faced a crime event without knowing the crime type).

To further enhance user privacy, we minimize the amount of pSS information shared to evaluate the SR query. We develop efficient query processing algorithms that find the SRs from the refined search space and minimize the exposure of pSS information. Since the number of possible routes between a source-destination pair is extremely high, a naive algorithm cannot find the SRs in real-time. Our search space refinement techniques allow our query processing algorithms to find the SRs with significantly reduced processing overhead.

Every user is not familiar with all roads, and it is also not feasible to involve a user for all queries. For a specific SR query, we identify the users who are familiar with the query-relevant area and select them as group members. The trustworthiness of the query answer depends on the overall knowledge of the selected group members. To show the credibility of the answer, we present a new measure called *confidence level* [2, 3] in the context of the SR query.

Existing safe route planners involve a centralized entity to find the SRs using crime data collected from reports [4] or crowd [5] or both [6–8]. They have major limitations:

- Ignore the privacy issues of the crowd harassment and incident data and thus suffer from data scarcity problem. Missing incident data can cause a system to return a route that is not actually safe and put a traveler at risk.
- Do not personalize the crowd's travel experiences by considering a user's travel pattern, which is essential to improve the accuracy of the query answer.
- Do not consider individual distances associated with different SSs for ranking the routes. For example, if two routes have the same lowest SS, then the route for which a user has to travel less distance with the lowest SS is the safest one, though its total distance might be greater than that of the other route.
- Do not show any measure to represent the trustworthiness of the identified SRs.

In recent years, the increase of computational power and storage in smartphones has enabled researchers to envision crowdsourced systems [2,3,9]. To the best of our knowledge, we propose the first privacy-enhanced and personalized solution for the SR queries with crowdsourced data and computation. Our solution overcomes the limitations of existing route planners. Our contributions are as follows:

- We present a model to quantify a user's travel experiences into irreversible pSSs and modify the indexing technique, R-tree to store pSSs. Based on pSSs, we design a privacy-enhanced crowd-enabled solution for the SR queries.
- We select the users who have the required knowledge in a query-relevant area, and we guarantee the credibility of the query answer evaluated based on the data of the selected group members in terms of the confidence level.
- We develop optimal algorithms, direct and iterative, to efficiently evaluate the SRs. The direct algorithm reveals group members' pSSs only for the query-relevant area. The iterative one further reduces the amount of shared pSSs at the cost of multiple communications per group member.
- We run extensive experiments with real datasets and evaluate the effectiveness and efficiency of our approach.

This thesis is organized as follows. First, we formulate our problem in Chapter 2. Then, we discuss previous works related to our problem in Chapter 3. In Chapter 4, we elaborate our safe route planning system by providing an overview of our system, our personalized safety quantification model, our indexing techniques, the algorithms and the privacy-preserving aspects of our work. In Chapter 5, we evaluate the effectiveness and efficiency of our algorithms through extensive experiments. Finally, we conclude our work with future directions in Chapter 6.

Chapter 2

Problem Formulation

We define basic terminologies and notations, explain the ranking of routes based on safety and formulate our problem for finding the safest route in this chapter. We also describe our privacy model. In Section 2.1, the necessary notations, definitions, and terminologies are given. Then in Section 2.2, we elaborately discuss our privacy model.

2.1 Preliminaries

The road network $N = (V, E)$ consists of a set of vertices V and a set of road segments E. The vertices represent the start or the end or the intersection points of roads. An edge $e_{ij} \in E$ connects the vertex v_i to the vertex v_j, where $v_i, v_j \in V$. A route R consists of a sequence of vertices $R = (v_{i_1}, v_{i_2}, \ldots, v_{i_{|R|}})$, where $e_{i_{k-1}i_k} \in E$. The total distance $dist(R)$ of R is the summation of distances of all edges in R. Table 2.1 shows the notations used in this thesis.

The total space is divided into grid cells. The knowledge score (KS), the pSS and the SS are computed for each grid cell area, which are defined as follows:

Definition 1. A knowledge score (KS): *The KS of a user for a grid cell area represents whether the user has visited the area of a grid cell. This KS is 0 if the user has not visited the area in the last w days, 1 otherwise.*

Definition 2. A personalized safety score (pSS): Given the safety score bound $[-S, S]$, the pSS of a grid cell area represents a user's travel experience in the area and is quantified between $-S \leq pSS \leq S$.

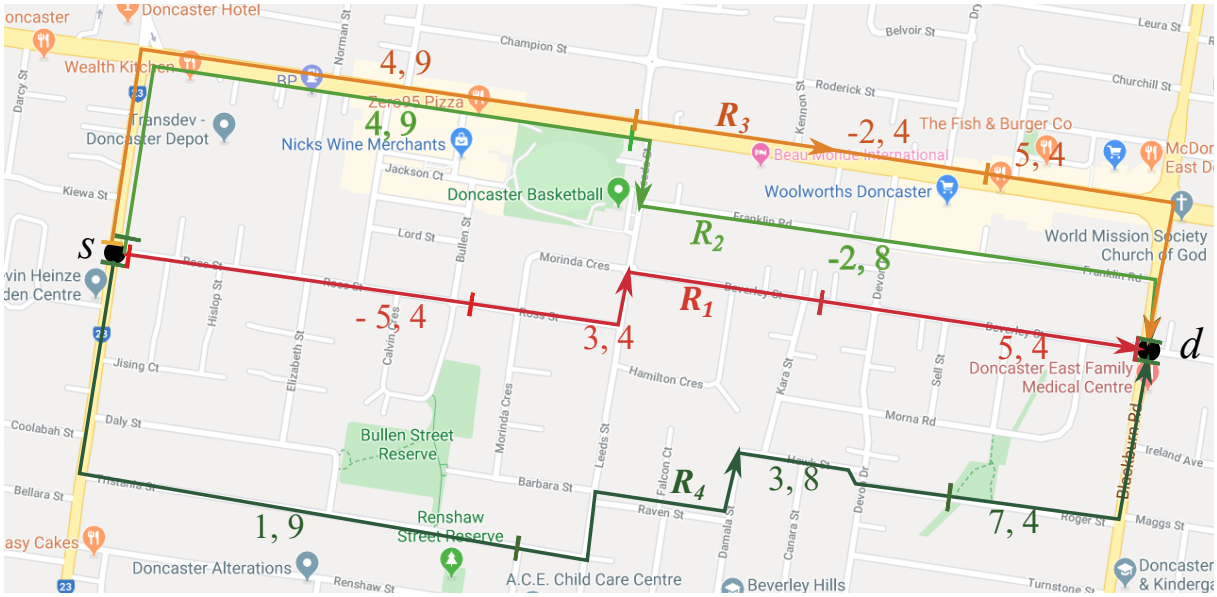


Figure 2.1: R_3 is the SR between s and d with $SS = -2$, $\delta = 18$

Table 2.1: Notations and their meanings

Notation	Meaning
$N(V, E)$	A road network
s, d	Source and destination vertices
δ	Distance constraint
SR	Safest route
pSS	Personalized safety score
SS	Safety score
KS	KS
$[-S, S]$	SS or pSS range

Definition 3. A safety score (SS): *Given a set of pSSs $\Psi_1, \Psi_2, \ldots, \Psi_n$ of n users for a grid cell area, the SS of the grid cell area is computed as* $\left\lfloor \frac{\Psi_1+\Psi_2+\ldots+\Psi_n}{n} \right\rfloor$.

To make the SS measure independent of the number of users who know about an area, we take the average of the pSSs instead of adding them together. The number of users whose pSSs are used to find the SS is considered to determine the credibility of the safest route (Section 5.3.2).

SS based route ranking. The SS of route R is the minimum of all SSs associated with the edges of R. The intuition behind considering the minimum SS instead of the average SS of the route is that even a small distance of road with a bad SS may put a traveler at risk. The route that has the

largest minimum SS among all possible routes between a source-destination pair is considered as the SR. If two routes have the same largest minimum SS, then we consider the smallest SS for which associated distances of two routes differ. The route that has the smallest associated distance for the considered SS, is the SR. We formally define the SR query below:

Definition 4. A safest route (SR) query: *Given a road network $N(V, E)$, distances and SSs of road segments, a source location s, a destination location d and a distance constraint δ, the safest route query returns a route SR between s and d such that $dist(SR) \leq \delta$ and SR is at least as safe as R, where R is any other route between s and d having $dist(R) \leq \delta$.*

In Fig. 2.1, assume that $\delta = 18$. The distance of R_1, R_2, R_3 and R_4 are 12, 17, 17, 21 respectively. R_4 does not satisfy δ. The smallest SSs associated with R_1, R_2 and R_3 are -5, -2 and -2, respectively. Though both R_2 and R_3 have the largest minimum SS, R_3 is the SR because R_3 has the smallest distance associated with the smallest SS -2.

2.2 Privacy model

We assume a semi-honest setting, where participants follow the system protocol but are curious to infer sensitive data from the shared information. In the semi-honest setting, the participants do not send false queries (e.g. unnecessary queries) or wrong pSSs and they do not collude with each other or any centralized entity. In our system, the unsafe event types (e.g., pick-pocketing or harassment) are considered as private data. We assume that anyone can play the role of an adversary for a user. The adversary knows the model to compute the pSSs but does not have any background knowledge about the time and frequency of a user's visits to an area. A user shares the KSs and pSSs for the purpose of the query evaluation. Our solution refrains an adversary from inferring the *unsafe event* type that a user encounters from the shared information of the user. Since a pSS can reveal high-level information like a user faced an unsafe event but not the type, our solution also aims to minimize the number of revealed pSSs for enhancing a user's privacy.

Chapter 3

Related Works

We address the problem of finding the safest route in a crowdsourced manner in this research. Therefore, this problem is closely related to the research works on route planning and crowdsourcing. This chapter provides elaborate discussions on existing research works relevant to our thesis. In Section 3.1, we focus on the existing works on the safe route planners. We compare those works with ours based on the problem setting (Section 3.1.1), privacy (Section 3.1.2), and efficiency (Section 3.1.3). In Section 3.2, we discuss the shortest path algorithms, in Section 3.3, we focus on other route planners, and in Section 3.4, we detail various indexing techniques that are used to store data in an organized way. Finally in Section 3.5, we present the existing works on crowdsourcing.

3.1 Safe Route Planners

Though researchers attempted to solve the safe route planning problem, the works have major limitations. Table 3.1 shows the problem settings and other features of existing works.

3.1.1 Problem Setting

None of the existing works considers individual distances associated with different SSs for ranking the routes after maximizing the minimum SS. Thus, the problem settings of existing works are not suitable for safe travel on roads. Furthermore, instead of considering the total

Table 3.1: A comparative analysis with existing safe route planners

	Problem Settings				Privacy	Efficiency
	Safety Level	pSS	Objective	δ		
[4]	Multiple	×	Provide multiple routes with a trade-off between SS and total distance	×	×	✓
[5]	Safe/Unsafe	×	Minimize the travel in unsafe regions	×	×	×
[6, 7]	Multiple	×	Minimize the weighted combination of SS and total distance	×	×	×
[10]	Safe/Unsafe	×	Minimize the travel in unsafe regions	×	×	✓
ours	Multiple	×	Maximize the minimum SS of the route and then minimize the individual distances associated with the SSs in the increasing order of SSs	✓	✓	✓

distance constraint, selecting appropriate weights in [6, 7] is not easy since it is not intuitive to determine which weights would meet a user's preferred trade-off between safety and distance for a specific source-destination pair. Again, there is no guarantee that the returned routes in [4] satisfy a user's required preference for safety and distance.

3.1.2 Privacy

The crime data for safe route planners may come from crime reports [4, 8, 11] or directly from crowds [5–8]. Crime reports are not regularly updated, and incomplete because many crimes go unreported. Though the crowd knows more and recent information compared to the crime reports, they would not share their incident and harassment data with a centralized service provider, if the privacy of their data is not ensured. Thus, one major limitation of existing works is that they suffer from data scarcity issues for privacy reasons and do not have enough data to provide accurate answers.

3.1.3 Efficiency

None of the existing safe route planning systems except [4, 10] developed efficient algorithms for large road networks. However, as already mentioned, the problem settings of [4, 10] cannot meet a traveler's requirement on roads.

3.2 Shortest Path Algorithms

Researchers have proposed many algorithms to solve the shortest path problem or its variant [12–18] over the last few decades. Research on this topic is still ongoing [19–21]. The shortest path problems can be divided into three types:

1. All-pair shortest path problem
2. Single-source shortest path problem
3. Single-pair shortest path problem

The all-pair shortest path problem can be solved using Floyd–Warshall algorithm [22], Johnson Algorithm [23]. The single-source shortest path problem can be solved by Dijkstra's algorithm [12], Bellman-Ford algorithm [24]. The single-pair shortest path can be solved using the algorithms for the single-source shortest path problem. We can also use A* searching [25] for the single-source shortest path problem. Many improvements have been proposed in these core algorithms for finding the shortest path [26, 27]. Many variants [28–33] of the shortest path problem exist, such as the shortest path with constraints [28, 29] and multi-objective shortest path problem [30]. Meta-heuristic algorithms have been also applied to solve these problems [34, 35]. Some researchers have maintained privacy while computing the shortest path for navigation [17, 36]. The authors of [18] used the Bellman-Ford algorithm to plan energy-efficient driving routes.

For this research, we have used the single-pair shortest path algorithm to refine the search space in Section 4.4.2. Furthermore, in Section 5, we have defined the distance constraint for an SR query based on the shortest road network distance to simplify its implication to users.

3.3 Other Route Planners

Variants of orienteering and scheduling problems [37, 38] have been studied for route planning. An orienteering problem finds a route between a source-destination pair that maximizes the total score within a budget constraint, where a score is obtained when the route goes through a vertex. The scheduling problems focus on incorporating temporal constraints in route planning (e.g., visiting locations to perform services in a timely manner). The problem settings of orienteering and scheduling problems are different from an SR query. Furthermore, their solutions do not consider search space refinement [39] and are not scalable for large road networks. For example, the exact solution of an orienteering problem can be found for a graph of up to 500 vertices [38], whereas the real road networks that we use in our experiments have on average 24 thousand vertices.

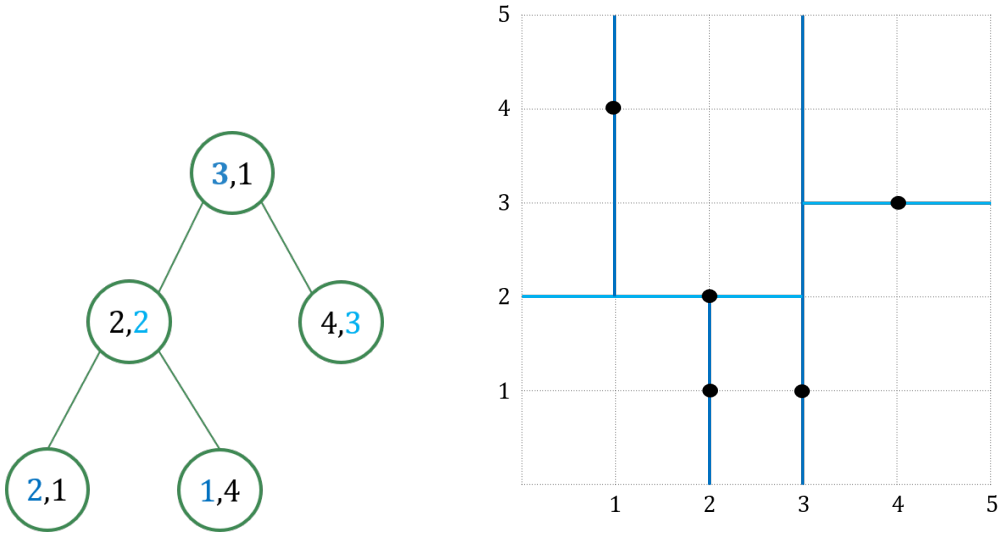
An SR query can be transformed to a multilevel optimization problem for solving it with a commercial optimization tool like IBM CPLEX: (L1) identify all routes that have the largest minimum SS within δ, (L2) consider the smallest SS for which the associated length of the identified routes differs and find the routes that have the smallest length associated with the considered SS, (L3) repeat L2 until the remaining route(s) have the same length associated with every SS. However, IBM CPLEX is not effective in terms of time and memory when a problem requires finding multiple answers like multiple routes with the same largest minimum SS in the SR query [40].

3.4 Indexing

We need to store KSs and pSSs in our system. These are spatial data. Indexing techniques for storing spatial data have been studied extensively in the literature [41–43]. Spatial data can be two dimensional or multidimensional. Spatial data includes points, lines, polygons etc. The number of samples can be huge. We need to store, access and manipulate them efficiently for a variety of applications. Specially, we need to perform range queries in spatial data frequently to access data near or within a particular area. Using a list or map can be costly storage-wise and is not efficient for range queries. Therefore, many indexing techniques for spatial data exploit the tree-based structure to store and access data efficiently. Following we describe some of them.

3.4.1 KD-tree

The KD-tree [44] is essentially a multidimensional binary search tree that stores multidimensional points. A new point is inserted in a leaf node. Each internal node of the tree divides a particular plane. For a k dimensional tree, a level l divides the $(l \mod k)$-th plane (if $(l \mod k)$ is 0, it divides the kth plane). Figure 3.1 shows an example of a KD-tree. A KD-tree has some limitations because its shape depends on the order of input, and in the worst case, it may contain n levels [42]. Variations of the KD-tree has been proposed to overcome these limitations, such as an adaptive KD-tree [45], KDB-tree [46]. However, these variations also have their limitations; e.g. the adaptive KD-tree is not dynamic.



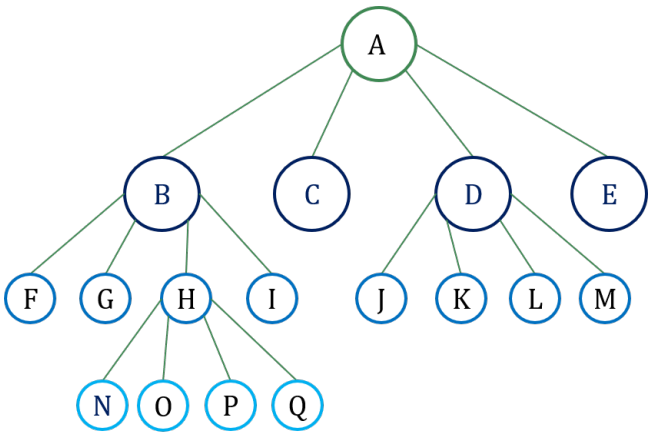
(a) A KD-tree

(b) Plane division by a KD-tree

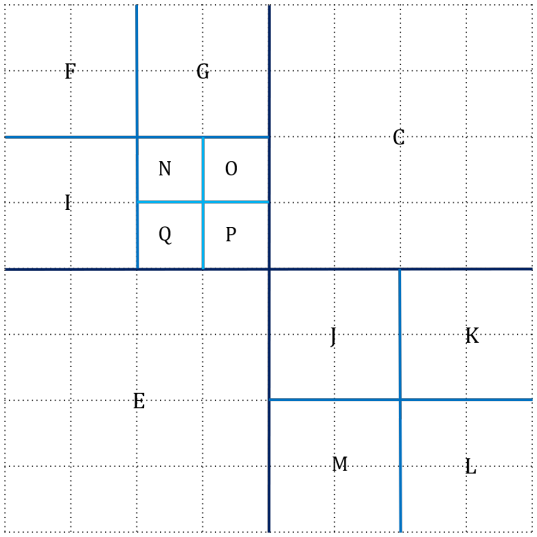
Figure 3.1: A KD-tree

3.4.2 Quadtree

A quadtree recursively decomposes a space into four quadrants. There are different types of quadtree, such as point quadtree, point region quadtree, region quadtree and polygonal map quadtree [42]. The point quadtree is similar to KD-tree, except the fact that each internal node has exactly four children. The point region quadtree is slightly different from the point quadtree. It divides the space into four equal quadrants and does not use the data points for plane decomposition. An example of this type of tree is given in Figure 3.2. A region quadtree generally stores an approximation of a polygon. Finally, a polygon map quadtree is used to store a set of polygons [47].



(a) A quadtree



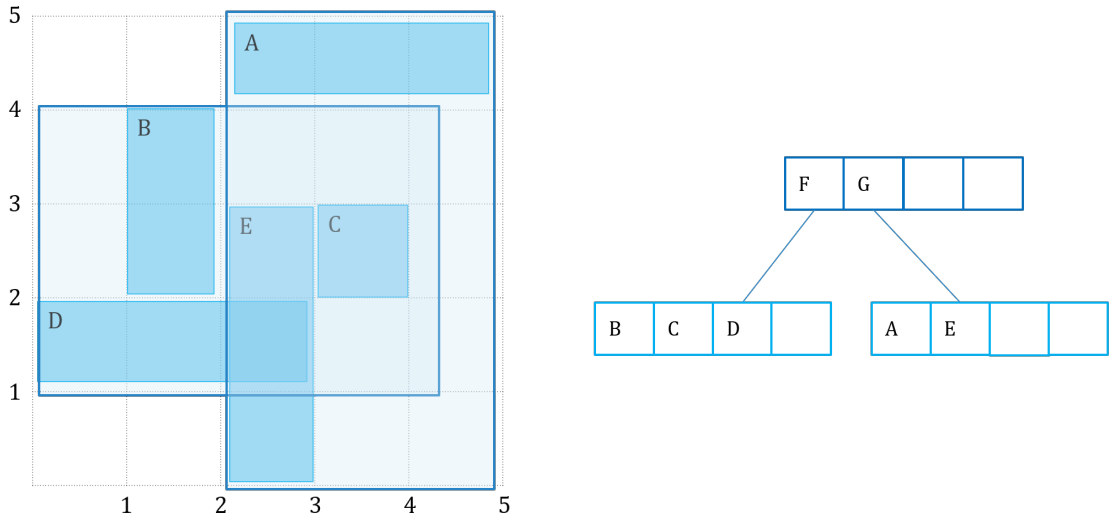
(b) Decomposition of space by the quadtree

Figure 3.2: An point region quadtree

3.4.3 R-tree

An R-tree is a spatial indexing structure proposed in 1984 by Antonin Guttman [48]. Figure 3.3 shows an example of an R-tree that stores rectangles. An R-tree is a multiway tree whose leaf node contains the spatial objects, such as point, line, polygon etc. The internal nodes group nearby objects together using a minimum bounding rectangle (MBR). There is a limit on how many entries each node can have. If that limit is exceeded when inserting data, then that node is split. The R-tree allows the overlap among MBRs of the internals nodes of the same level (e.g. MBR F and G in Figure 3.3) and frequently, the internal nodes cover some empty spaces. Fewer overlaps and empty spaces increase the R-tree's efficiency.

Multiple variations of the R-tree have been proposed including R+-tree [49], R*-tree [50] and Hilbert R-tree [51]. The R+-tree is an efficient version of the R-tree that avoids the overlap



(a) Two-dimensional rectangles (A, B, C, D, E) in a plane

(b) An R-tree constructed from those rectangles

Figure 3.3: An R-tree

among MBRs. However, this comes at the price of more nodes and space [42, 43]. Also, the construction and modification are more complex. $R+$-tree is more efficient for the point query search, however, range queries can be costly [43]. The R*-tree differs from the R-tree in terms of the insertions. When a node becomes overfull while inserting data, instead of splitting that node immediately, some entries are tried to be reinserted in that tree first. The Hilbert R-tree organizes the data based on Hilbert value, which is efficient but might not always be realistic [42].

For storing the pSSs in a user's personal device, we chose to modify an R-tree because we need to perform the range queries for retrieving pSSs by utilizing the low computational power and low space capacity of mobile devices. Its variants are either too complex or require more space. Instead of using the R-tree directly, we modified it because we wanted to reduce space consumption by keeping the same pSS information of nearby grid cells together.

3.5 Crowdsourcing

Crowdsourcing has been widely used for route finding and recommendation [52–62], trip planning [63–65], POI search [2, 3, 66], POI summarization [67], package delivery [68, 69], sensing [70–72], traffic monitoring [73, 74], vehicular network [75], indoor mapping and localization [76–78] and many other tasks [79].

The works in [3,9] eliminate the location-based central service provider to protect users' sensitive location data and divided the query evaluation task among the selected group. While evaluating a query, those works preserve privacy through data imprecision. In [2], the authors considered protecting the privacy of a user's POI knowledge by minimizing the shared POI information with others. Compared to the static POI data, crime data are more complex and challenging to hide from others. We develop a quantification model to hide the type of incident data using pSS and search space refinement techniques to minimize the shared pSS information.

Chapter 4

Our Approach

In this chapter, we explain in detail our approach for finding the safest route in a privacy-preserving manner. We provide our system overview in Section 4.1. Then, in Section 4.2, we establish a practical safety quantification model. Next, we discuss our indexing techniques for pSSs and KSs in Section 4.3. After that, in Section 4.4, we propose two efficient algorithms to compute the safest route and analyze their complexity. Finally, in Section 4.5, we explain in detail the privacy-preserving aspects of our work.

4.1 System Overview

We develop a privacy-enhanced, personalized, and trustworthy solution for safe route planning with crowdsourced data and computation. Fig. 4.1 shows the architecture of our system. Users in our system store their pSSs of their visited areas on their own devices. In the case of storage constraints, users can also consider alternative private storage (cloud storage). The users share their KSs with the centralized server (CS). A KS only provides the information that a user has visited the area. A user can also hide the information of her visit on a sensitive area by not setting corresponding KS to 1 as the user has the control to decide on what the user shares with the CS.

In our system, we set a default ratio between the lengths of the safest and the shortest routes for the distance constraint. To initiate an SR query, the query requestor (QR) provides the source and the destination locations for the query. The QR can also specify the distance constraint as an absolute value or as a ratio between the lengths of the safest and the shortest routes. If the QR specifies the distance constraint, our system replaces the default distance constraint with the

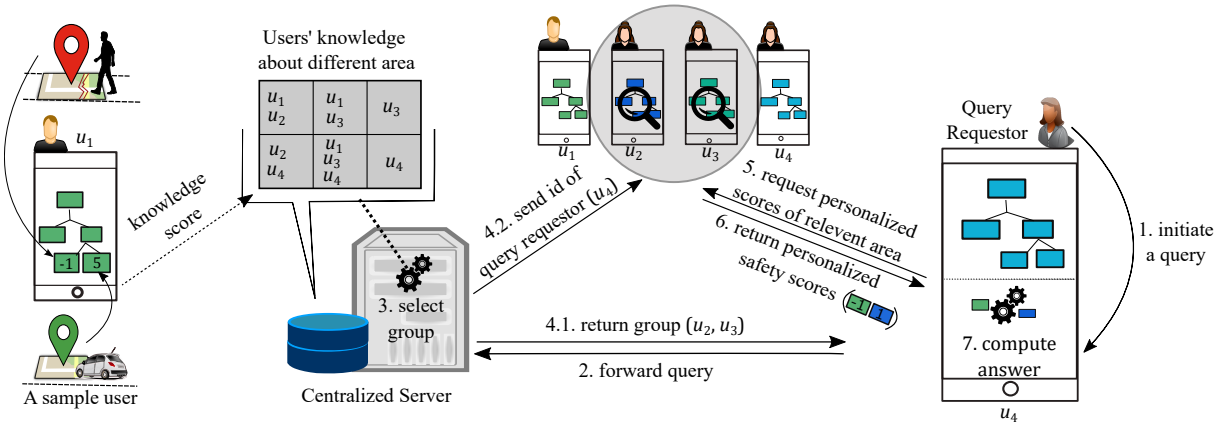


Figure 4.1: System architecture

QR provided distance constraint. It is not realistic to use the computation power of all users for all queries and asking them whether they know any query-relevant area. The availability of KSs allows CS to address this issue. When the CS receives a query from a QR, it selects a group based on the query parameters and the stored KSs of the users. Then the CS returns the IDs of the group members to the QR and sends the identity of the QR to the group members. The QR evaluates the query in cooperation with the group members without involving the CS. The QR retrieves pSSs of the query-relevant area from the group members, computes the SSs of each road using the pSSs of the group members, and finds the SR. Note that the QR communicates with the group members in parallel to retrieve the pSSs. Therefore, the communication with multiple group members does not increase the query processing time significantly. However, in case of the resource constraint in the QR's device, a subset of the group members' pSSs can be retrieved.

4.2 Quantification of Safety

We utilize the personal travel experiences of the crowd to evaluate the SR queries. For this reason, we need to model the diverse travel experiences of each user into a personalized quantifiable metric of safety. In this section, we discuss our model for quantification of safety. We discuss the existing models that have quantified safety and their limitations in Section 4.2.1, and explain our safety quantification model in Section 4.2.2.

4.2.1 Limitations of Existing Models

Existing researches on safe routes have modeled safety in a variety of ways. The authors of [6,7] quantify the safety of a road network edge by simply considering the number of crimes in the particular distance buffer area of that edge. They do not consider the recency and the severity of crimes, the ratio between the unsafe visits and the safe visits by an individual user, and the fact that the impact of a crime decays with distance. Thus, the quantified SSs of roads in [6,7] fail to model the real-scenarios. The work in [4] improves the way to find the SS of a road network edge by considering the crime events of the last few days and weighting the crime events based on their distances from the road. None of the above works [4,6,7] allow the SS to vary in different parts of a road network edge, which is possible for long roads.

In [8], the authors provide a more elaborate model of safety. However, the model suffers from the following limitations: (i) stores historical data and cannot address the constraint of the limited storage of the personal devices, (ii) does not differentiate the weights of crime events based on the frequency of the user's visits, (iii) only considers that the effect of a crime spreads to its nearby places only if no crime occurs there, (iv) does not provide a smooth decay of the effect of older events, rather takes the moving average of the events of the last few days, and discards the impact of previous events, (v) does not consider the severity of a crime event, and (vi) does not allow the SS to vary in different parts of an edge.

4.2.2 Our Model

We develop a model that overcomes the limitations of existing models. In our model, the travel experiences of users are converted into pSSs and then aggregated to infer the SSs of different areas. When a user visits an area, an event occurs. If the user faces a crime, then that event is unsafe; otherwise, it is safe.

Model Properties

Our model has the following properties:

1. The safety of an area depends on the frequency of the users' visits.

- If a user visits an area twice and faces unsafe events both times, then intuitively, that area is riskier than another area where a user visits 10 times and faces unsafe events two times among those visits.
- If a user visits an area 5 times safely, then that area is safer than another area that is visited once safely.

2. The safety of an area also depends on the safety of its nearby places. Therefore, if a user visits an area, the impact of the event is distributed to nearby areas.
3. The safety of an area depends on the recency of the safe and unsafe events. If a user faces an unsafe event in an area, then the crime's effect decays with time. If a user visits an area safely, then the perception of safety due to the safe visit also decays with time.
4. The safety of an area depends on the type and severity of an unsafe event.
5. The pSSs are not allowed to grow indefinitely. They are bounded within a maximum and a minimum value so that while aggregating, a single user's experience does not dominate the SS of an area.
6. A road network edge may go through multiple grid cells and thus, can have different SSs.

An important advantage of our model is that it is storage efficient as it does not store the historical visit data of a user.

Model Computation

Let the impact of a safe event in the occurring area be ξ_+ and the impact of an unsafe one be ξ_-, where $\xi_+, \xi_- \in \mathbb{N}$. ξ_+ is the same for all safe events. ξ_- varies with the type and the intensity of the crime or inconvenience faced.

The impact $\xi(= \xi_+/\xi_-)$ of an event reduces exponentially in nearby areas and becomes ξ' as per the following equation: $\xi' = \xi \times e^{-\frac{dist^2}{2h^2}}$, where the constant h controls the spread of the event. $dist$ represents the distance of the event location from the grid cell. This equation is inspired by the Gaussian kernel density estimation [4].

The pSS, Ψ of an area is bounded within $[-S, S]$ and $\Psi \in \mathbb{N}$ and $0 < \xi_+ < S$ and $-S < \xi_- < 0$. If an event occurs in a place for the first time then $\Psi = \xi$. If another event occurs there, then $\Psi = \Psi + \xi$. If an event ξ occurs nearby, whose effect is ξ' here, then $\Psi = \Psi + \xi'$. If $\Psi > S$ then $\Psi = S$ and if $\Psi < -S$ then $\Psi = -S$. Initially, Ψ is set to *unknown*.

A pSS decays every Δ_d days. If the decay rate is r_d and $\Psi \neq 0$, then after every Δ_d days, Ψ becomes $\Psi = \Psi \times r_d$, where $0 < r_d < 1$ and $r_d \in \mathbb{R}$. Therefore, the decay of older events' impacts is smooth. For example, if $r_d = 0.8$ and $\Delta_d = 2$, then $\Psi = 3$ becomes 2.4 after two days, and becomes 1.92 after two more days.

The values of parameters ξ_+, ξ_-, $S+$, $S-$, Δ_d and r_d are the same for all users and decided centrally. For each grid cell, our model stores only two values: the pSS and when that pSS was last updated. Therefore, this model is storage-efficient and suitable for smart devices. The SS of an area is computed from the shared pSSs of the users (Definition 3).

4.3 Indexing User Knowledge

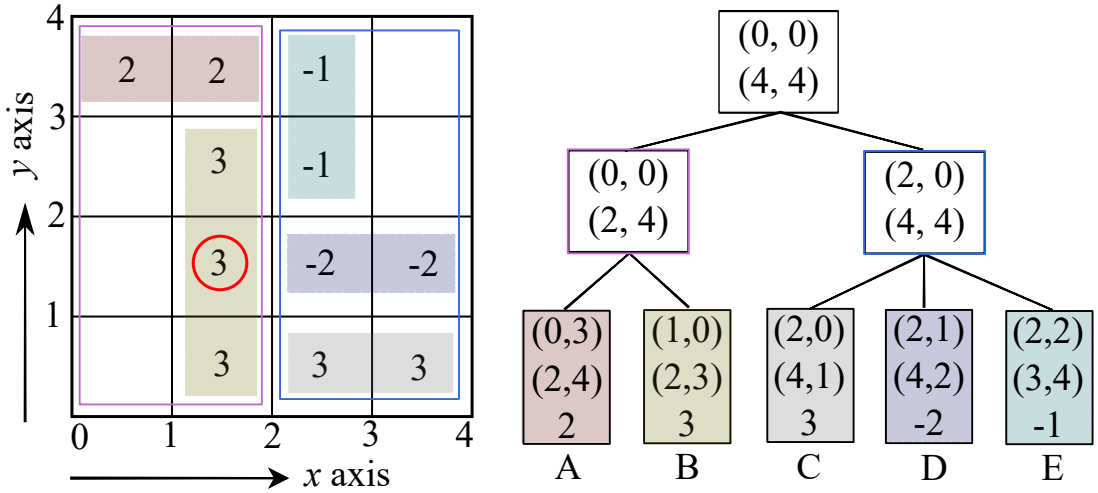
In this section, we elaborate on how we store the pSSs and KSs in our system. A user stores the pSS for every visited grid cell in the local storage and accesses it for evaluating the SR query. The CS stores the KSs of users for every grid cell and uses them for computing query-relevant groups. For efficient retrieval of pSSs and KSs, we use indexing techniques: local and centralized, respectively. In Section 4.3.1, we explain the local indexing mechanism in detail, and in Section 4.3.2, we elaborate the global one.

4.3.1 Local Indexing.

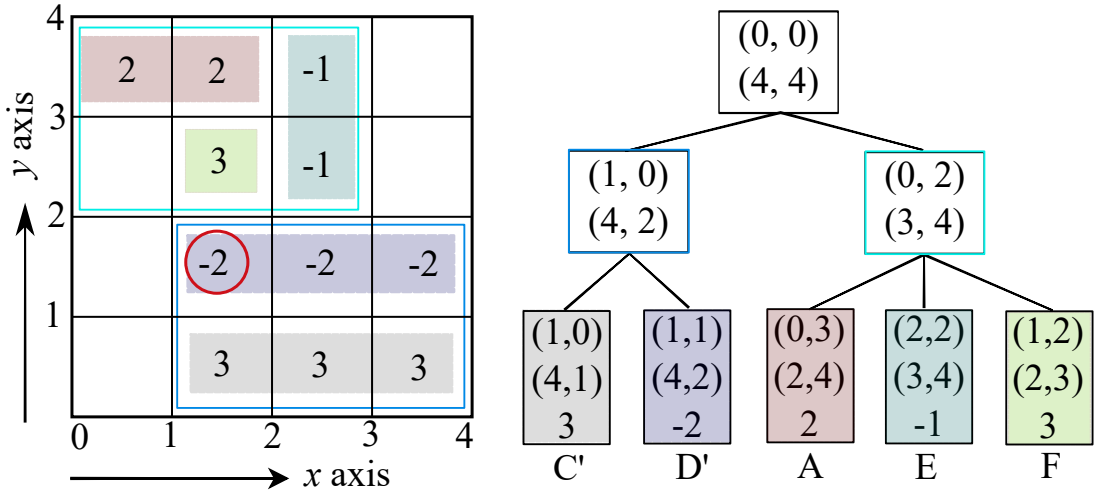
Storing pSSs for the whole grid in a matrix would be storage-inefficient because a user normally knows about some parts of the grid area. We adopt a popular indexing technique R-tree [48] for storing pSSs of the visited grid cells. The underlying idea of an R-tree is to group nearby spatial objects into minimum bounding rectangles (MBRs) in a hierarchical manner until an MBR covers the total space.

For every visited grid cell, a user stores its pSS and the time of its last update. The last update time is required for decaying the pSS. To reduce the storage overhead, we combine nearby adjacent grid cells with an MBR, where the grid cells have the same SS and the difference between the last update times of two cells does not exceed a small threshold. We call this MBR as a supercell and each leaf node of an R-tree represents a supercell. Each leaf node stores the information of the coordinates of MBR, the pSS, and the average of the last update time of the considered grid cells of a supercell. The supercells are recursively combined into MBRs.

The intermediary nodes of the R-tree store the coordinates of the MBR. The MBR of the root node of the R-tree represents the total grid area. Fig. 4.2 shows an example of a grid and the corresponding R-tree. For the sake of clarity, we do not show the last update times in the figure. Note that while creating supercells, there will be a small data loss due to the merging of grid cells for which the differences of their last update times are within a small threshold (e.g. in our experiment we set it to 12 hours).



(a) The pSSs for a 4x4 grid is stored in a modified R-tree



(b) A pSS changed from 3 to -2 and is updated in the R-tree

Figure 4.2: A user's pSSs is stored in a modified R-tree

Supercell Generation

A traditional R-tree only considers the location of the spatial objects for grouping, whereas we consider the location, the pSS and the last update time of the grid cells for grouping them into supercells. To compute the non-overlapping supercells, we scan the grid cells twice: row-wise and column-wise. For row-wise (or column-wise) scan, we maximize the number of grid cells included in a supercell row-wise (column-wise) and then take the supercells of the scan (row-wise or column-wise) that generates the minimum number of supercells. After computing the supercells for the leaf nodes, we insert them into a traditional R-tree.

Supercell Update

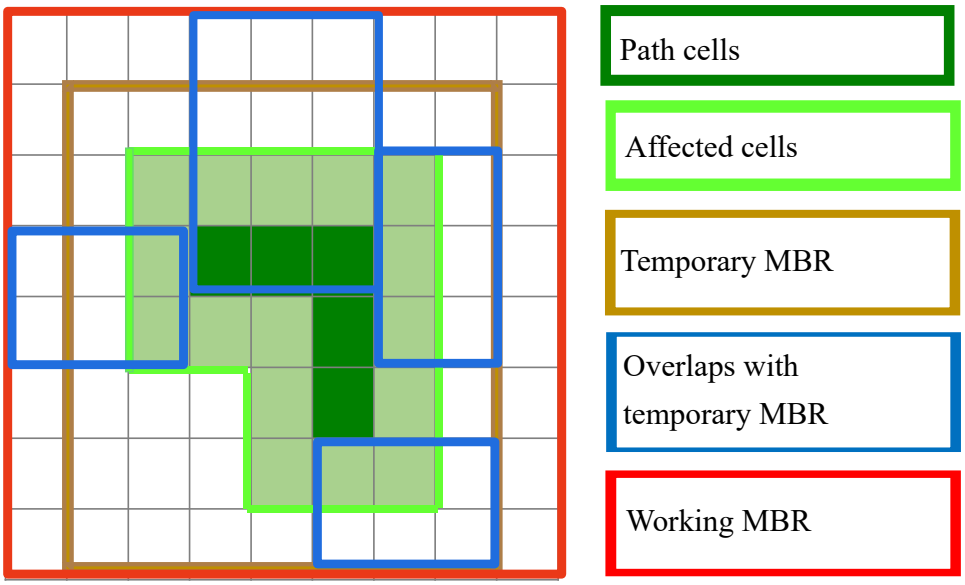


Figure 4.3: Necessary MBRs for updating a supercell

To update the pSSs of grid cells for a visited route R, the following steps are performed:

- *Compute route cells and affected cells.* Compute the grid cells that overlap with R as route cells. The affected cells include the route cells and their nearby cells (Fig. 4.3).
- *Compute temporary MBR.* Find the temporary MBR that includes the affected cells and one extra grid cell besides each affected cell in the boundary (Fig. 4.3). The reason behind considering an extra grid cell is to identify the adjacent existing supercells later.
- *Find overlapping supercells.* Find existing supercells that intersect with the temporary MBR. There are four overlapping supercells in Fig. 4.3.

- *Compute working MBR.* Find the working MBR that includes these overlapping supercells and the affected cells (Fig. 4.3).
- *Generate new supercells.* By considering the location, the pSS and the last update time of the grid cells included in the working MBR, generate the new supercells.
- *Update R-tree.* Remove those overlapping supercells from R-tree and add the new supercells. Update the intermediary nodes based on the change in the leaf nodes. Fig. 4.2b shows the updated R-tree for the change of the pSS from 3 to -2 in a grid cell (shown with a red circle).

4.3.2 Centralized Indexing

The KSs are accessed when the query-relevant groups are computed and updated when a user visits a new area. Since the probability is high that at least a user knows a grid cell area, we store each grid cell's data in a hash-map with the grid cell's coordinates as a key. For each grid cell, we store the user ids whose KS is 1 for the corresponding grid cell area.

4.4 Query Evaluation

We elaborate our approach for finding the answer of an SR query in this section. We provide two optimal algorithms for computing an SR query. In Section 4.4.1, we calculate the query-relevant area and the query-relevant group for an SR query. This part is the same for both optimal algorithms. Then, in Sections 4.4.2 and 4.4.3, we describe two optimal algorithms, direct and iterative, respectively. After that, in Section 4.4.4, we analyze the complexity of our proposed algorithms. Next, in Section 4.4.5, we show the simulation of our algorithms for an SR query. Finally, in Section 4.5, we explain in detail the privacy-preserving aspects of our work.

In our system, a query requestor (QR) retrieves the required pSSs from relevant users and evaluates the SR query. We develop direct and iterative algorithms to find the SR for a source-destination pair s and d within a distance constraint δ.

The number of possible routes between a source-destination pair can be huge. Retrieving the pSSs for all grid cells that intersect the edges of all possible routes and then identifying the SR would be prohibitively expensive. Our algorithms refine the search space and avoid exploring all

routes for finding the SR. We present two optimal algorithms:

1. Direct Optimal Algorithm (Dir_OA)
2. Iterative Optimal Algorithm (It_OA)

Dir_OA aims at reducing the processing time, whereas It_OA increases privacy in terms of the number of retrieved pSSs. Though a pSS does not reveal a user's travel experience (Chapter 4.5) with certainty, the user's privacy is further enhanced by minimizing the number of shared pSSs with the QR.

4.4.1 Query-relevant Area and Group

Query-relevant area A_q.

Our algorithms exploit the elliptical and Euclidean distance properties to find the query-relevant area A_q. We refine the search area using an ellipse where the foci are at s and d of a query and the length of the major axis equals δ. According to the elliptical property, the summation of the Euclidean distances of a location outside the ellipse from two foci is greater than the length of the major axis. On the other hand, the road network distance between two locations is greater than or equal to their Euclidean distance. Thus, the road network distance between two foci, i.e., s and d through a location outside the ellipse, is greater than δ. The refined search area A_q includes the grid cells that intersect with the ellipse. A_q enables us to select a query-relevant group and mitigate unnecessary processing and communication overheads and data exposure.

Query-relevant group G_q.

A query-relevant group G_q consists of the users whose KS is 1 for at least one grid cell in A_q. After receiving a query, the centralized server sends G_q and the list M_q of knowledgeable group members for every grid cell in A_q to the QR.

Algorithm 1: Dir_OA(s, d, δ, N)

1 $N', A_q \leftarrow$ compute_query_area(s, d, δ, N);
2 $G_q, M_q \leftarrow$ retrieve_query_group(A_q);
3 $SS_q \leftarrow$ compute_SS(G_q, M_q, A_q);
4 $N'' \leftarrow$ refine_query_area(s, d, δ, N', SS_q);
5 $SR \leftarrow$ compute_safest_route$(s, d, \delta, N'', SS_q)$;
6 **return** SR;

4.4.2 Direct Optimal Algorithm (Dir_OA)

One may argue that we can simply apply an efficient shortest route algorithm (e.g., Dijkstra) for finding the SR by considering the SS instead of the distance as the optimizing criteria. However, it is not possible because the SR identified in this way in most of the cases may exceed δ.

Algorithm 1 shows the pseudocode for Dir_OA. The algorithm starts by computing the query-relevant area A_q and the query-relevant road network N' that is included in A_q. The edges in N that go through grid cells in A_q but those cells have not been visited by any user are not included in N'. Then the algorithm retrieves the query-relevant group G_q and the list M_q of grid cell wise knowledgeable group members from the centralized server. In the next step, the algorithm retrieves the pSSs from the group members and aggregates them to compute the SSs of the grid cell in A_q using Function compute_SS.

After having the SSs for the grid cells in A_q, the algorithm further refines N' to N'' by pruning the edges that are guaranteed to be not part of the SR (Line 4). The idea of this pruning comes from [4], where edges with the lowest SSs are incrementally removed until s and d become disconnected. To reduce the processing time, we exploit binary search for finding N''. Specifically, we compute the mid-value mid of the lowest and the highest SSs, i.e., $-S$ and S, and remove all edges that have SS lower than or equal to mid. Note that an edge can have more than one associated SSs as it can go through multiple grid cells. For binary search, we consider the minimum of these SSs as the SS of the edge. After removing the edges, we find the shortest route between s and d and check if the length of the shortest route satisfies δ. If no such route exists, then the removed edges are again returned to N'', and the process is repeated by setting the highest SS to mid. On the other hand, if such a route exists, the process is repeated by setting the lowest SS to $mid + 1$. The repetition of the process ends when the lowest SS exceeds the highest one.

Finally, Dir_OA searches for the SR within δ in N'' using Function compute_safest_route.

Dir_OA starts the search from s and continuously expands the search through the edges in the road network graph N'' until the SR is identified. The algorithm keeps track of all routes instead of the safest one from s to other vertices in N'' as it may happen that expanding the SR from s exceeds δ before reaching d.

The compute_safest_route function uses a priority queue Q_p to perform the search. Each entry of Q_p includes a route starting from s, the road network distance of the route, the distance associated with each SS in the route. The entries in Q_p are ordered based on the safety rank, i.e., the top entry includes the SR among all entries in Q_p. Initially, routes are formed by considering each outgoing edge of s. Then the routes are enqueued to Q_p. Next, a route is dequeued from Q_p and expanded by adding the outgoing edges of the last vertex of the dequeued route. The formed routes are again enqueued to Q_p. The search continues until the last vertex of the dequeued route is d. While expanding the search we prune a route if it meets any of the following two conditions:

1. If the summation of the road network distance of the route and the Euclidean distance between the last vertex of the route and d exceeds δ.
2. If the road network distance of the route exceeds the current shortest route distance of the last vertex from s.

Both pruning criteria guarantee that the pruned route is not required to expand for finding the SR. The current shortest route in the second pruning condition for a vertex v from s is determined based on the distances of the dequeued routes whose last vertex is v. Since the dequeued routes to v are safer than a route that has not been enqueued yet, the route can be safely pruned if its length is greater than the current shortest route's distance.

4.4.3 Iterative Optimal Algorithm (It_OA)

It_OA enhances user privacy by reducing the shared pSSs with the QR as it does not need to know the SSs of all grid cells in A_q. Algorithm 2 shows the pseudocode for It_OA. Similar to Dir_OA, It_OA computes N', A_q, G_q, and M_q. It_OA does not apply the binary search to further refine N' as it avoids retrieving the pSSs of all grid cells in A_q. It_OA gradually retrieves the pSSs from the group members only for the grid cells that are required for finding SR. Another advantage of It_OA is that it only involves those group members who know about the required grid cells.

Algorithm 2: It_OA(s, d, δ, N)

$N', A_q \leftarrow$ compute_query_area(s, d, δ, N);
$G_q, M_q \leftarrow$ retrieve_query_group(A_q);
$SS_q \leftarrow \emptyset$, $Q_p \leftarrow \emptyset$, $v \leftarrow s$;
while $v! = d$ **do**
    $A_q{}' \leftarrow$ find_required_cells(v, N', A_q, SS_q);
    $SS_q \leftarrow SS_q \bigcup$ compute_SS$(G_q, M_q, A_q{}')$;
    $SR \leftarrow$ get_safest_route(v, N', SS_q, Q_p);
    $v \leftarrow$ get_last_vertex(SR);
return SR;

It_OA iteratively searches for the SR in N' using a priority queue Q_p like Dir_OA. It_OA expands the search by exploring the outgoing edges of v. Initially v is s and later v represents the last vertex of the dequeued route from Q_p. In each iteration, It_OA identifies the grid cells in $A_q{}'$ through which those outgoing edges pass (Function find_required_cells), and computes their SSs by retrieving pSSs from the group members (Function compute_SS). Next, using Function get_safest_route, It_OA forms the new routes by adding the outgoing edges of v at the end of the last dequeued route, and enqueues them into Q_p if they are not pruned using the conditions stated for Dir_OA. At the end, the function dequeues a route from Q_p for using that in the next iteration. The search for SR ends if the last vertex of the dequeued route is d.

It_OA increases the communication frequency of the QR with the relevant group members. To mitigate this issue, we introduce a parameter X_{it} that trades off between the communication frequency and the number of pSSs shared with the QR. For $X_{it} = 1$, the algorithm considers only the outgoing edges of the last vertex v of the dequeued route for identifying the grid cells for which the pSSs will be retrieved. For $X_{it} > 1$, the algorithm repeats the process X_{it} times by considering the outgoing edges of the last vertices of the newly formed routes. While doing so, the algorithm applies the first and second pruning techniques where applicable. We decide the value of X_{it} in experiments. Please note that the algorithm do not collect the pSSs of nearby edges if the SSs of the immediate edges to be expanded are known.

4.4.4 Complexity Analysis

The compute_safest_route function in Dir_OA algorithm can be drawn as a tree where the source node is the root and the destination node is in the last level. If the average branching factor is b

and the average depth of a route from s to d is p, then Q_p will be dequeued $1 + b + b^2 + \ldots + b^p$ times. The maximum possible number of elements at a time in Q_p is b^p. Therefore, if a binary min heap is used for Q_p, the runtime complexity will be $O(b^p \cdot \log(b^p))$ which is $O(b^p \cdot p \log b)$. Since we utilize two pruning techniques due to which the average depth reduces, the complexity becomes $O(b^{p/r} \cdot \frac{p}{r} \log b)$, where $r = r_1 + r_2$. The factors r_1 and r_2 represent the effects of our first and second pruning techniques, respectively. In It_OA, edges are expanded till X_{it} depth along with the two pruning techniques in Function find_required_cells, whose runtime complexity is $O(b^{X_{it}/r'})$. Here, $r' = r'_1 + r'_2$; r'_1 and r'_2 represent the effect of our first and second pruning techniques in Function find_required_cell, respectively. Thus, the runtime complexity of It_OA is $O(b^{\frac{p}{r} + \frac{X_{it}}{r'}} \cdot \frac{p}{r} \log b)$.

4.4.5 Simulation

In this section, we go through the simulations of Dir_OA and It_OA algorithms in detail. We exhibit the simulations for the same small road network and the same SR query. The road network and the SR query both are shown in Figure 4.4. The source and destination locations of the SR query for the simulation are highlighted in blue. We assume the distance constraint δ is 36 km.

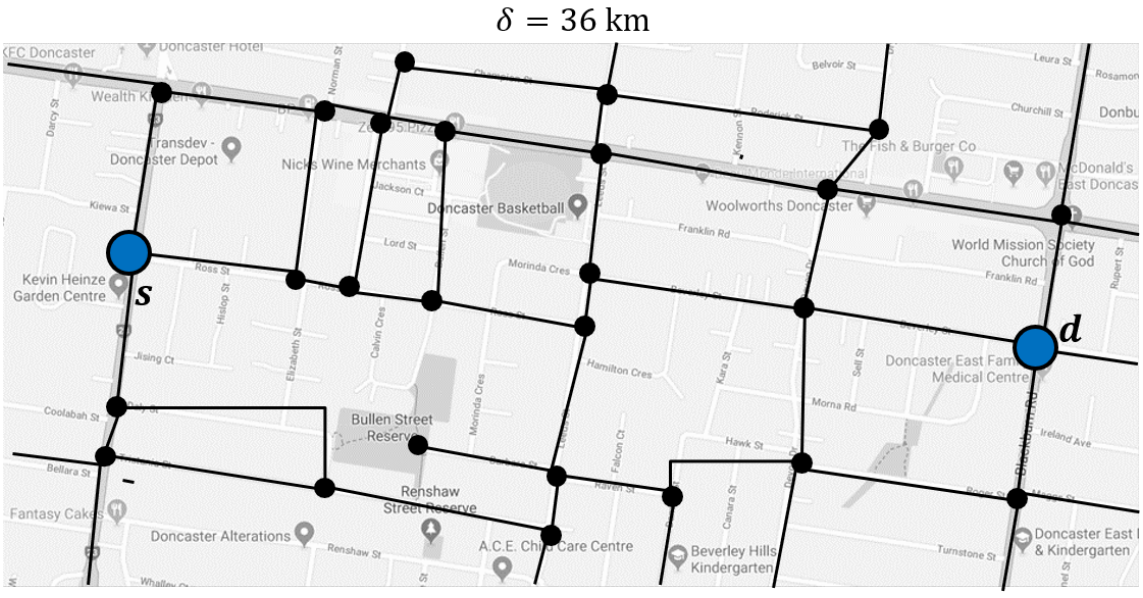
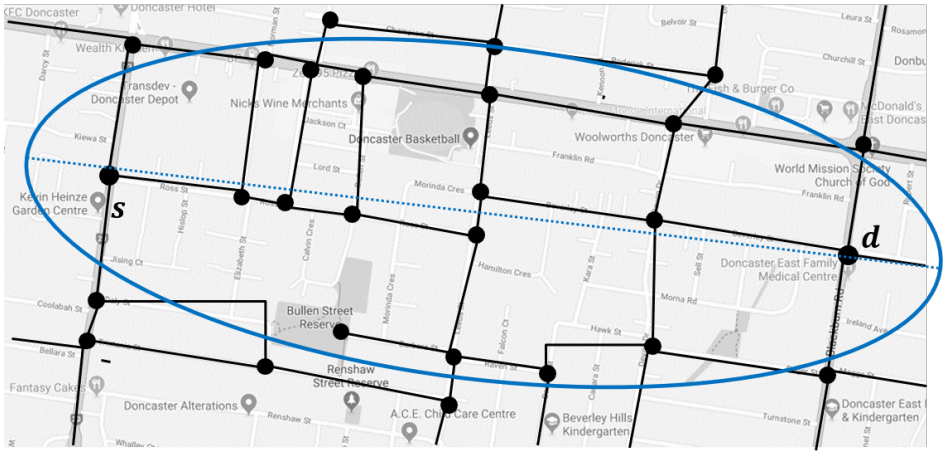


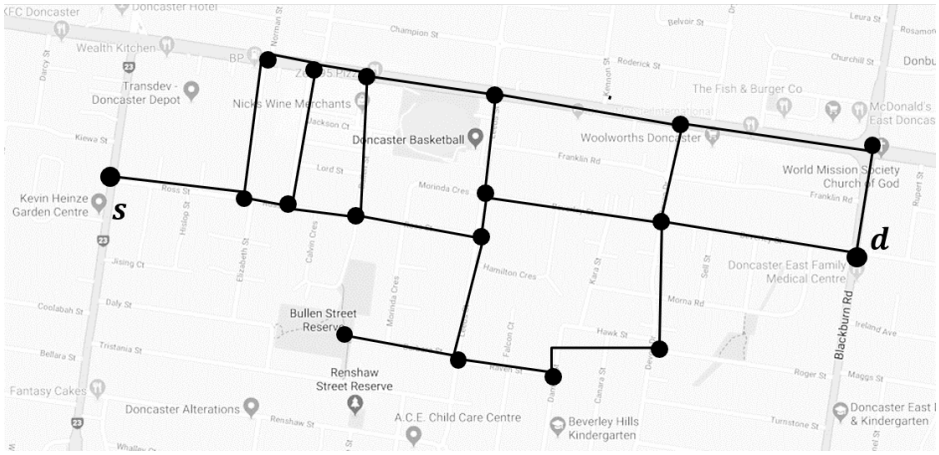
Figure 4.4: A small road network and an SR query for showing the simulations of Dir_OA and It_OA

$\delta = 36$ km



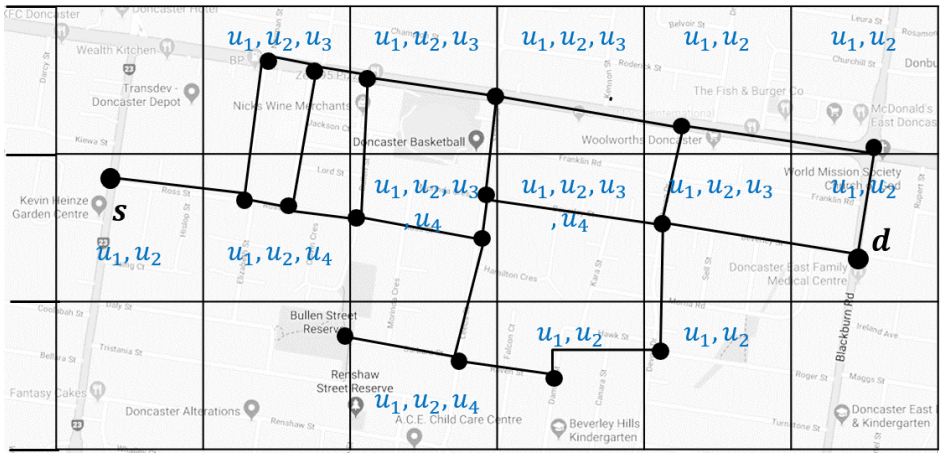
(a)

$\delta = 36$ km



(b)

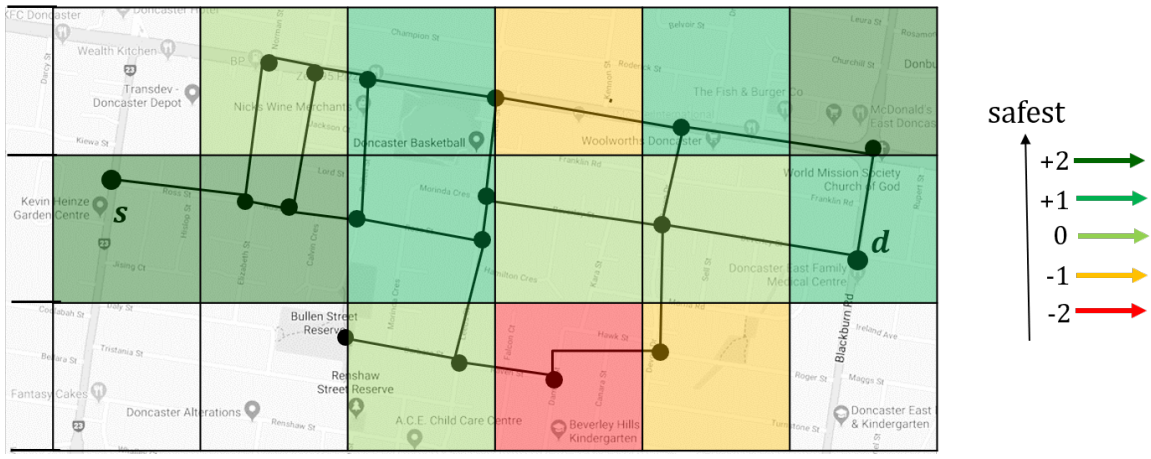
$\delta = 36$ km



(c)

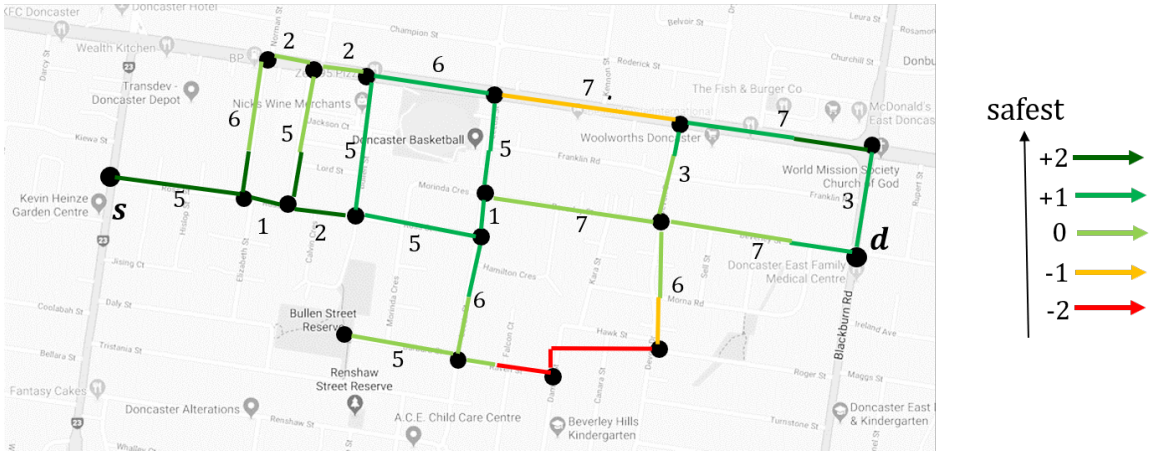
Figure 4.5: Simulation of Dir_OA

$\delta = 36$ km



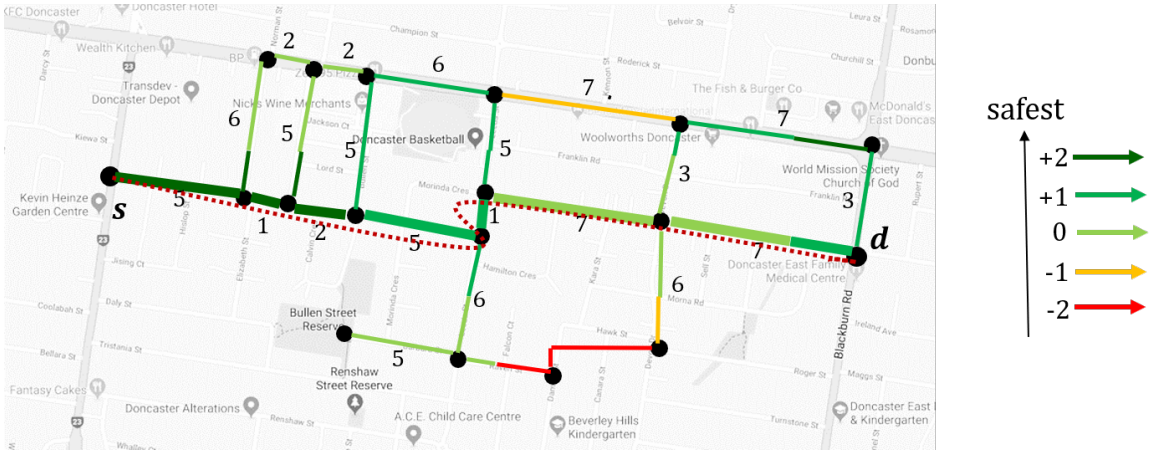
(d)

$\delta = 36$ km



(e)

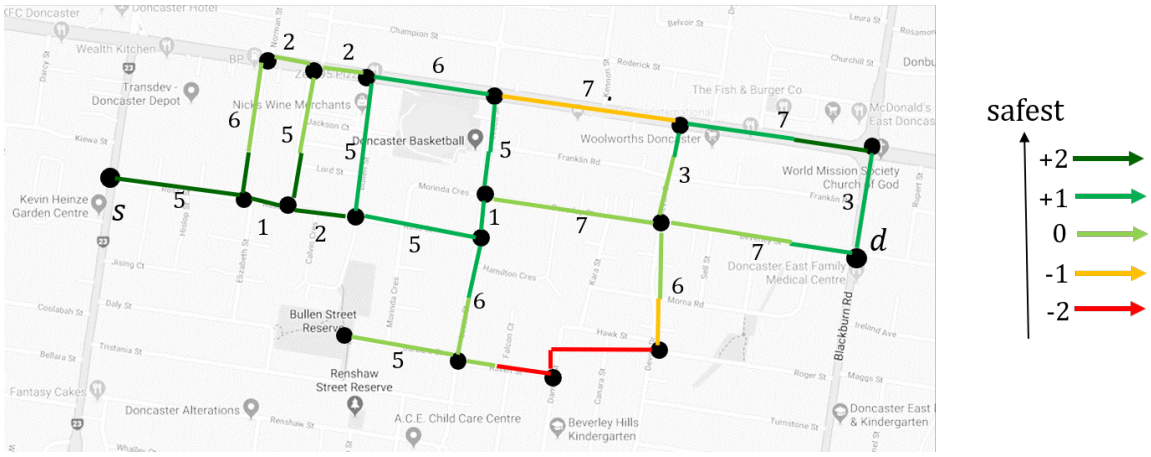
$\delta = 36$ km



(f)

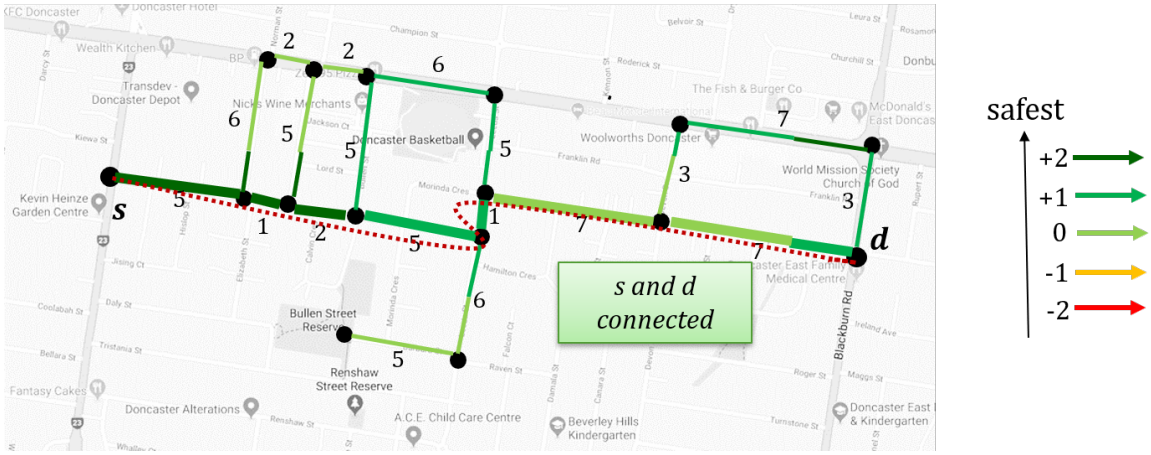
Figure 4.5: Simulation of Dir_OA

$\delta = 36$ km



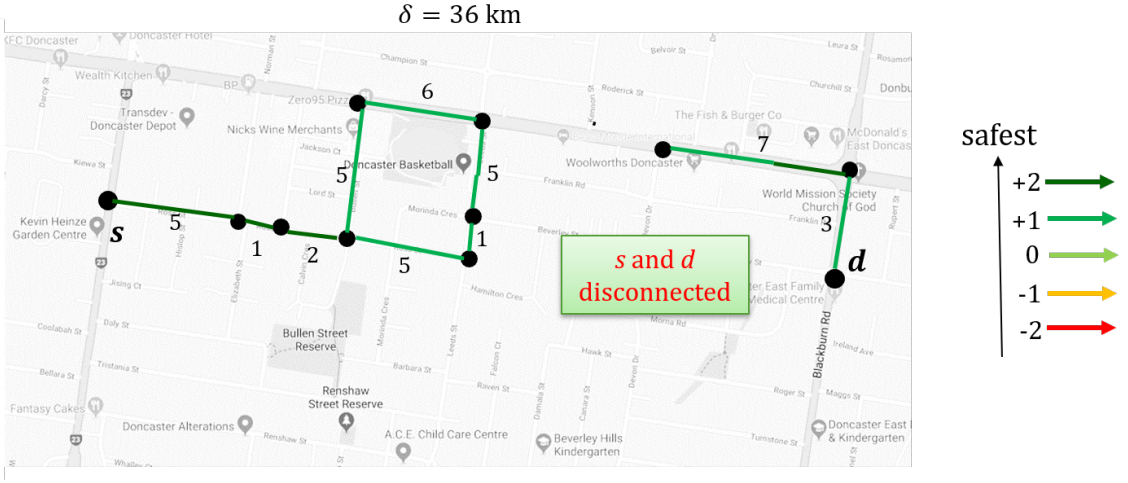
(g)

$\delta = 36$ km



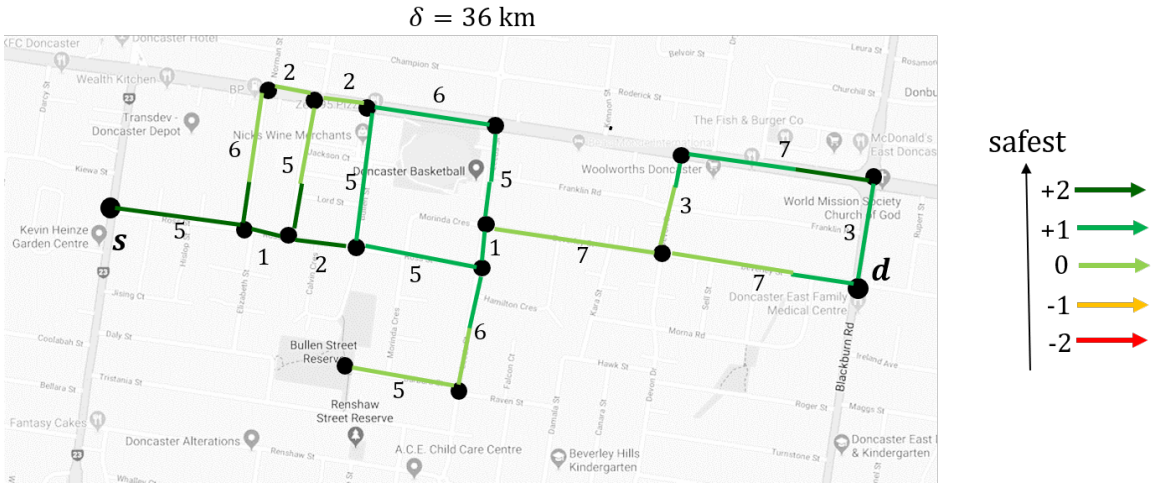
(h)

$\delta = 36$ km



(i)

Figure 4.5: Simulation of Dir_OA



(j)

Figure 4.5: Simulation of Dir_OA

Simulation of Dir_OA

The simulation of computing the given SR query using Dir_OA is demonstrated in detail through Figures 4.5 and 4.6. In Figure 4.5, we simulate up to Line 4 of Algorithm 1, and we simulate the details of Line 5 in Figure 4.6.

In Dir_OA, the query is forwarded to the central server. The query-relevant area for the given SR query is computed by the central server. For this, the road network is refined via an ellipse where the foci are at s and d, and the length of the major axis is 36 km, which is the distance constraint. The ellipse is shown in Figure 4.5a. We only include the edges of the road networks that fall inside the ellipse into the query-relevant area. This query-relevant area is shown in Figure 4.5b. The grid cells of the query-relevant area must include at least one edge of the query-relevant area. Those grid cells are shown in Figure 4.5c.

After computing the query-relevant area, the query-relevant group for this query is computed. The query-relevant group includes the users who know about the grid cells included in the query-relevant area. The ids of the group members who know about each grid cells included inside the query-relevant area are shown in Figure 4.5c. For example, users u_1 and u_2 know about the grid cell that includes the source. From the Figure, it is evident that the query-relevant group for this query includes four users: u_1, u_2, u_3, and u_4. The central server sends the ids of the group members to the QR. After that, all computations take place inside the QR's device.

To compute the SSs of the grid cells of the query-relevant area, the QR collects the pSSs of the query-relevant group members. The computed SSs are illustrated in Figure 4.5d. To keep

the simulation simple, we set $S = 2$. We represent five levels of safety with five colors in the figure. The number against each edge represents the length of the edge. The SSs of the edges are symbolized through colors in Figure 4.5e. Please note that one edge can have multiple SSs.

After computing the SSs, the search space is refined through the binary search. For this binary search-based refinement, Dir_OA first computes the shortest path from s and d shown in Figure 4.5f through a dotted red line. The length of the shortest path is 28 km. This information means that there is at least one route from source to destination within the distance constraint. After that, it computes the mid-value mid of the safety score range: $[low, high]$. Here, $low = -2$, $high = +2$, therefore, $mid = \frac{-2+2}{2} = 0$. It removes all edges with SSs less than zero. The resulting graph is shown in Figure 4.5g. It again computes the shortest path from s to d in the resultant graph and finds the shortest path of length 28 km as shown in Figure 4.5h. The shortest path length is less than the distance constraint, 36 km. From this, we can infer that the source and destinations are still connected. Therefore, it updates low to 0 and recompute $mid = \frac{0+2}{2} = +1$. It removes all the edges with SSs less than $+1$ and gets the resulting graph shown in Figure 4.5i. When Dir_OA computes the shortest path from s to d in this graph, it cannot find any path. It means that the source and destination are disconnected now. Therefore, it brings back the edges that were removed last time and updates $high$ to $+1$. Figure 4.5j exhibits the updated graph after bringing back the removed edges. We recompute $mid = \frac{0+1}{2} = 0$ which is equal to low. Therefore, Dir_OA terminates the binary search-based pruning. It has removed all the unnecessary edges at this point. The safest route is computed in this refined graph as shown in Figure 4.5j.

After refining the search space, Dir_OA computes the SR. To compute the SR from the refined graph, it starts exploring from the source and continues until it reaches the destination. A *safety-based priority queue* is utilized for storing the formed routes; thus, the safest route is always on the top of the queue. In Figure 4.6, the details of this exploration are shown. In the figure, the enqueued routes are shown based on the order of the safety inside the queue for clarity. The currently dequeued route is made bold and the last vertex of that route is circled red. The edges' colors represent the safety levels shown in Figure 4.5j. Following, we describe the detailed steps sequentially:

1. First, Dir_OA enqueues the source s in a queue. When it removes the top of the queue, s is dequeued. The algorithm forms new routes with the outgoing edges of s ($s \rightarrow a$) and enqueues them (Figure 4.6a).

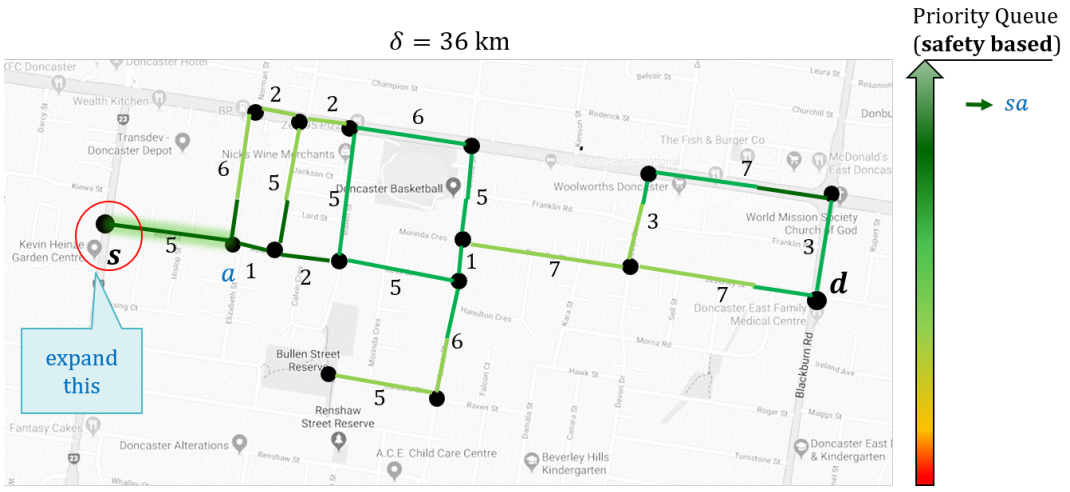
2. Then, this algorithm dequeues the route $s \rightarrow a$ from the top of the queue. It keeps track of the fact that a can be reached from s with a safe route of length 5 km. It forms new routes

$s \to a \to b$ and $s \to a \to c$. The route $s \to a \to b$ is enqueued. The route $s \to a \to c$ of length 11 km is pruned as $11 + \text{euclidean\_dist}(c, d) = 11 + 26 = 37$ km, which is greater than the distance constraint, 36 km (pruning condition 1) (Figure 4.6b).

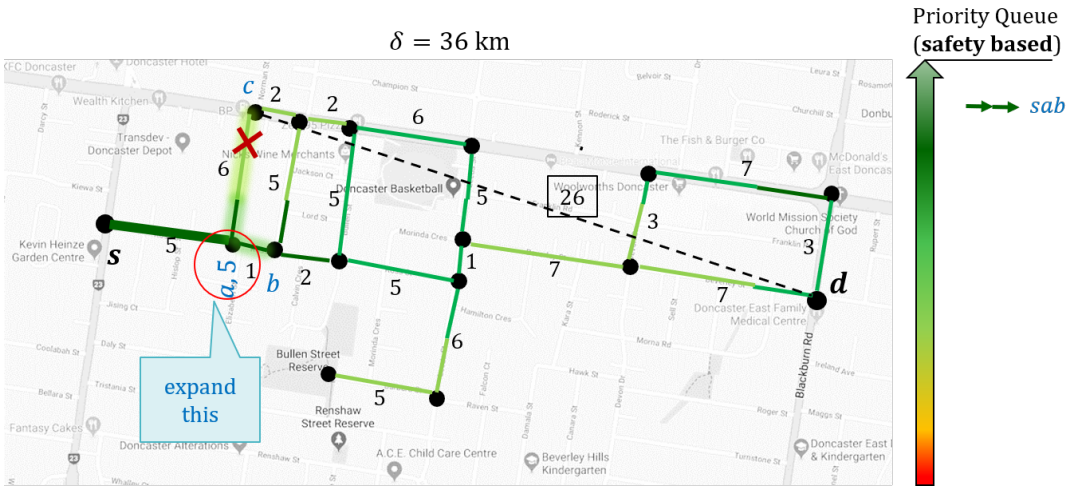
3. After that, Dir_OA dequeues $s \to a \to b$ from the top of the queue and keep track of this route's length, 6 km. It forms two new routes $s \to a \to b \to e$ and $s \to a \to b \to f$, and enqueues them (Figure 4.6c).

4. Next, the top of the queue is removed, and the route $s \to a \to b \to f$ is dequeued. The length, 8 km, is tracked in vertex f. The algorithm forms two routes $s \to a \to b \to f \to g$ and $s \to a \to b \to f \to h$ and enqueues them (Figure 4.6d).

5. Again, the top of the queue is removed, and the route $s \to a \to b \to f \to g$ is dequeued. The length of this dequeued route, 13 km, is tracked in vertex g. Dir_OA forms two new

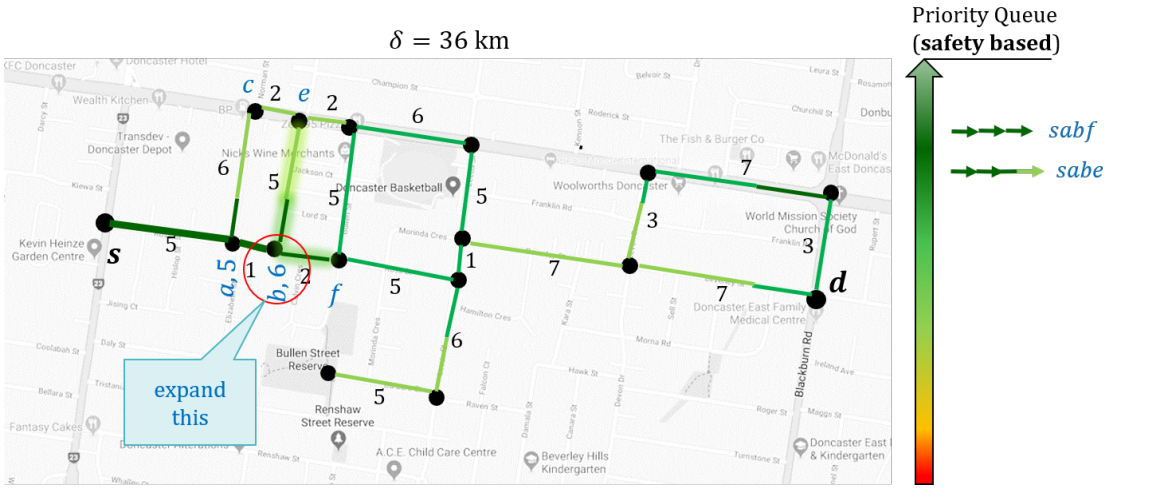


(a)



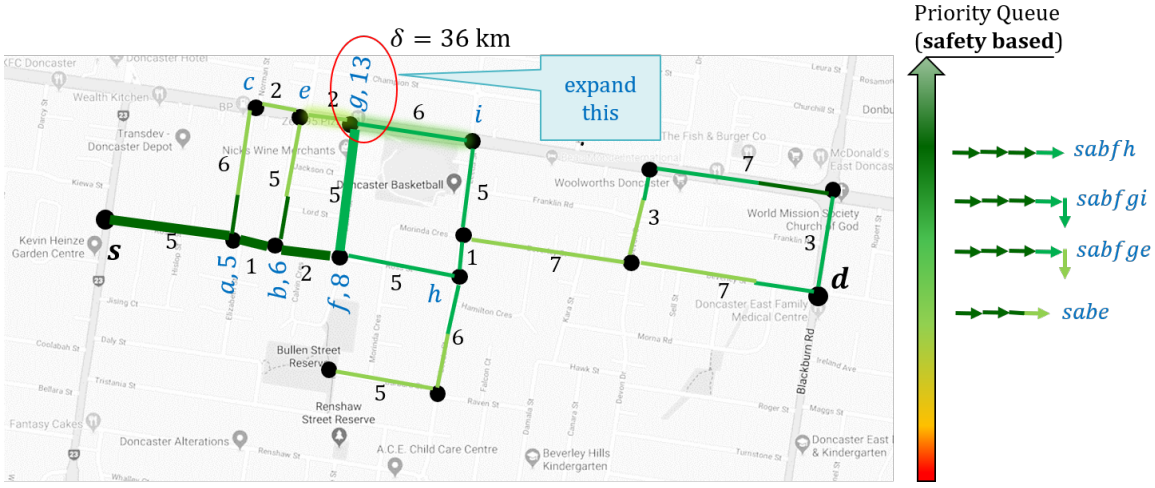
(b)

Figure 4.6: Simulation of Dir_OA (continued)



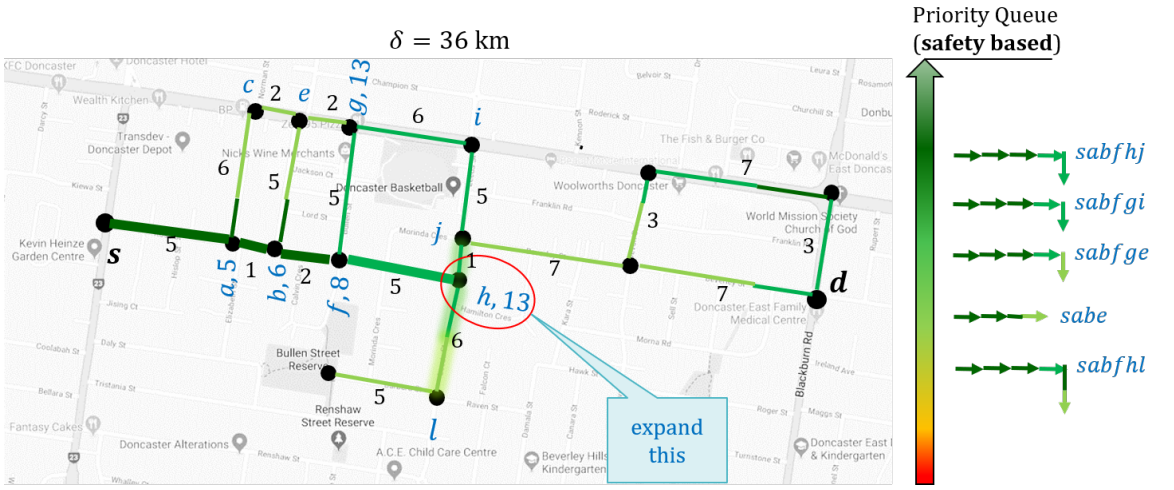
(c)

(d)

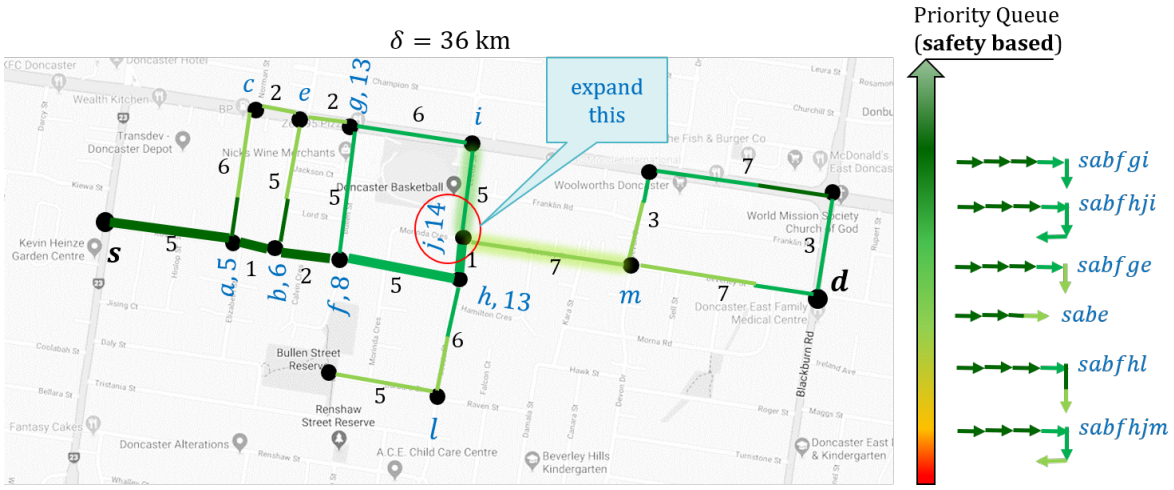


(e)

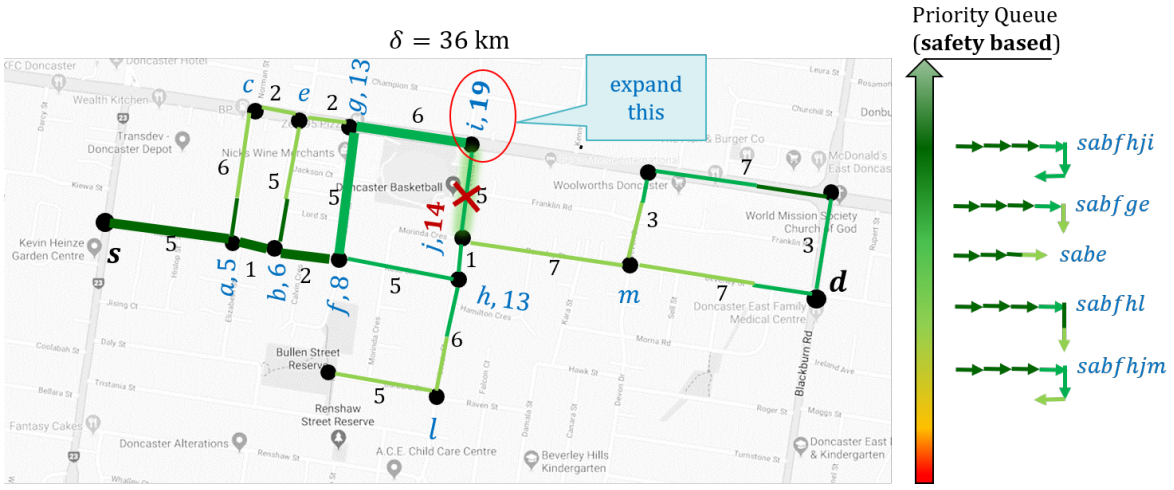
Figure 4.6: Simulation of Dir_OA (continued)



(f)

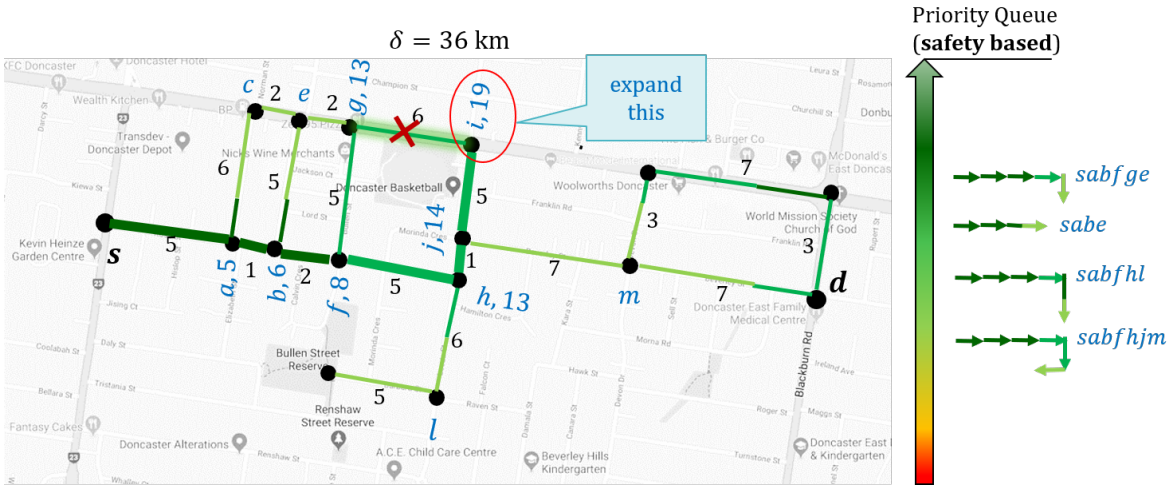


(g)

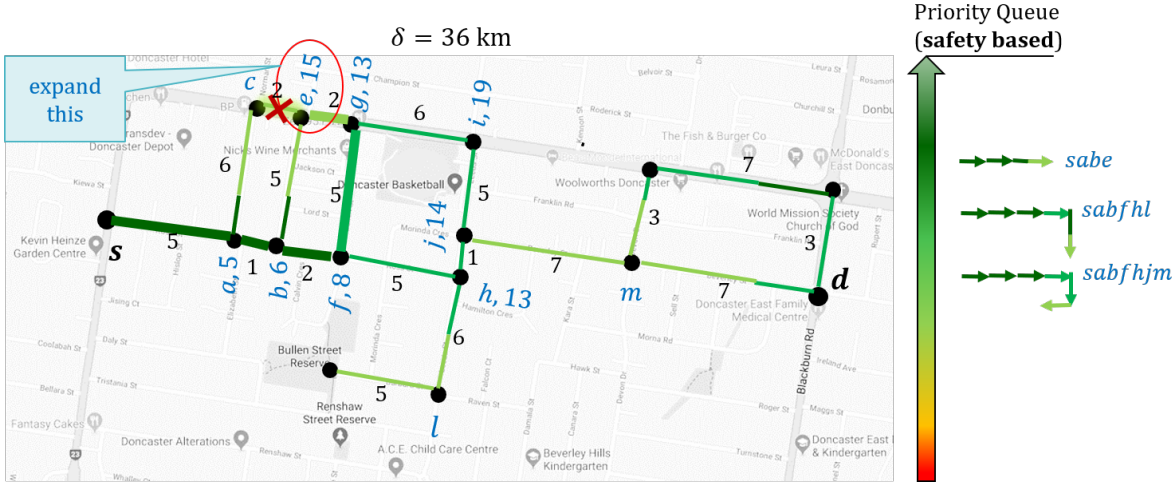


(h)

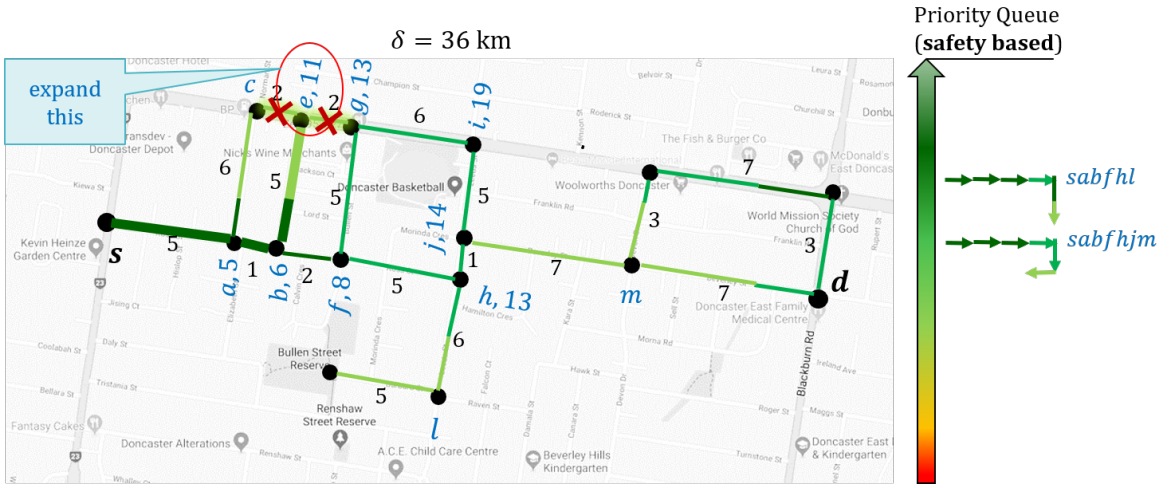
Figure 4.6: Simulation of Dir_OA (continued)



(i)

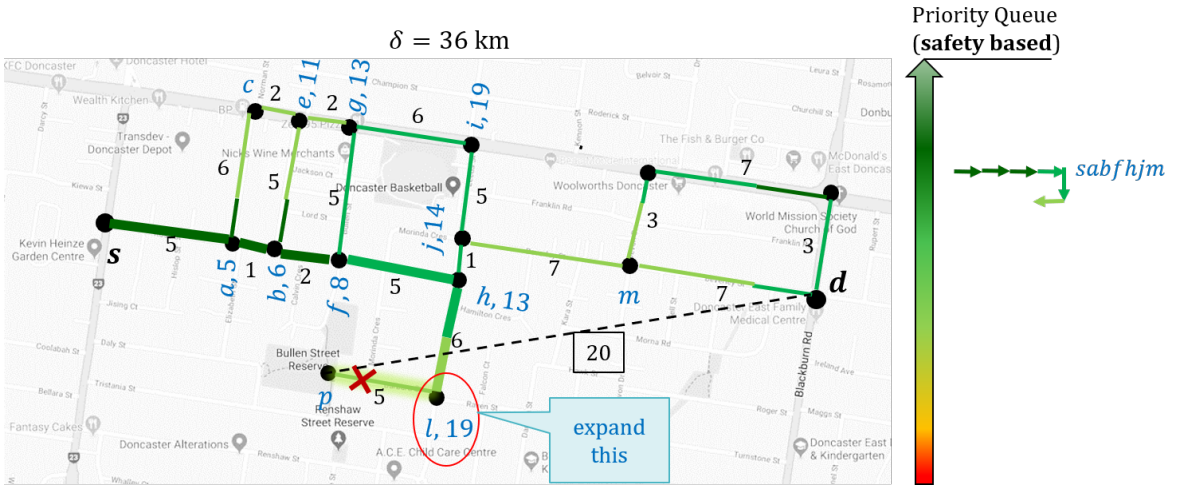


(j)

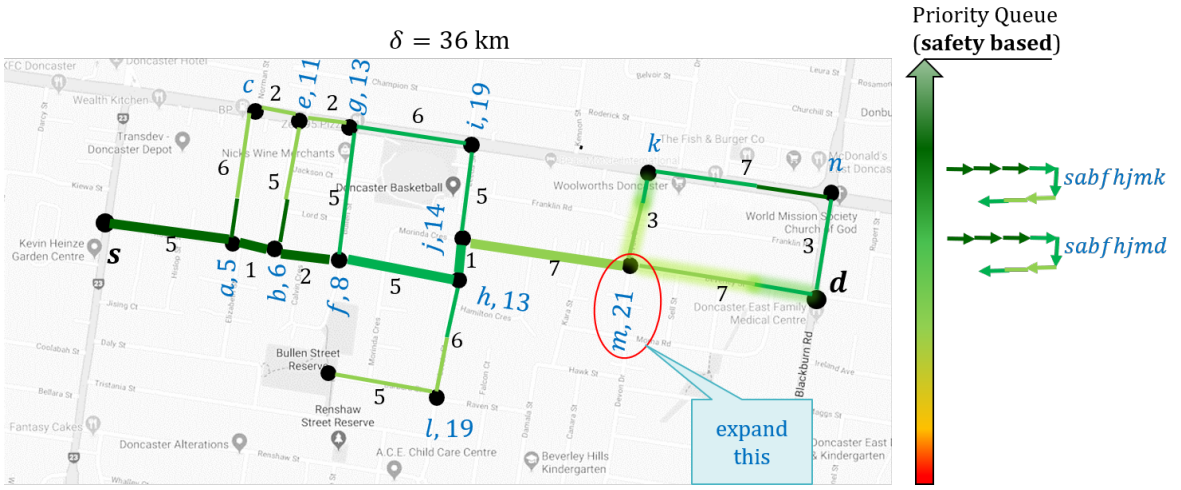


(k)

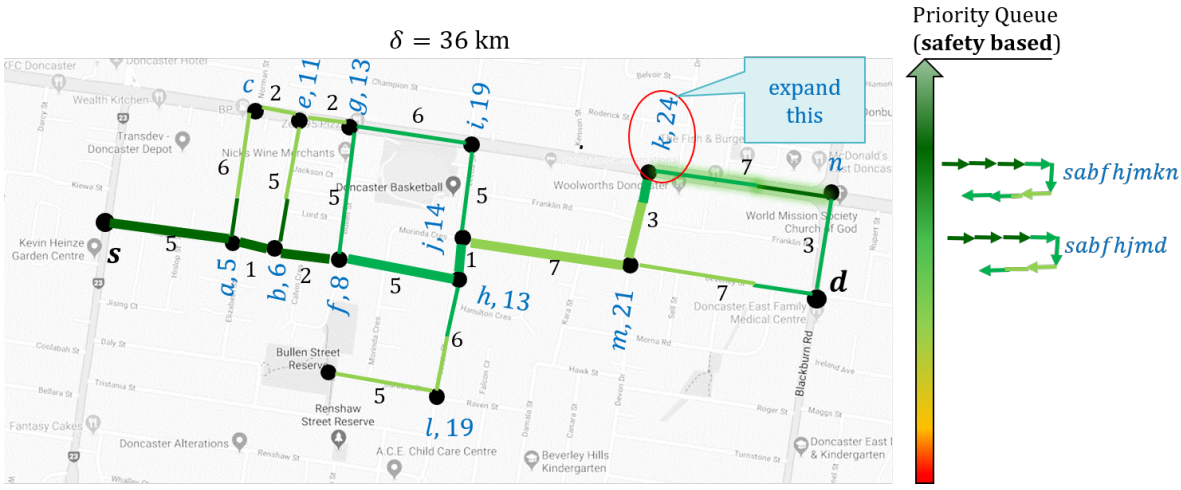
Figure 4.6: Simulation of Dir_OA (continued)



(l)

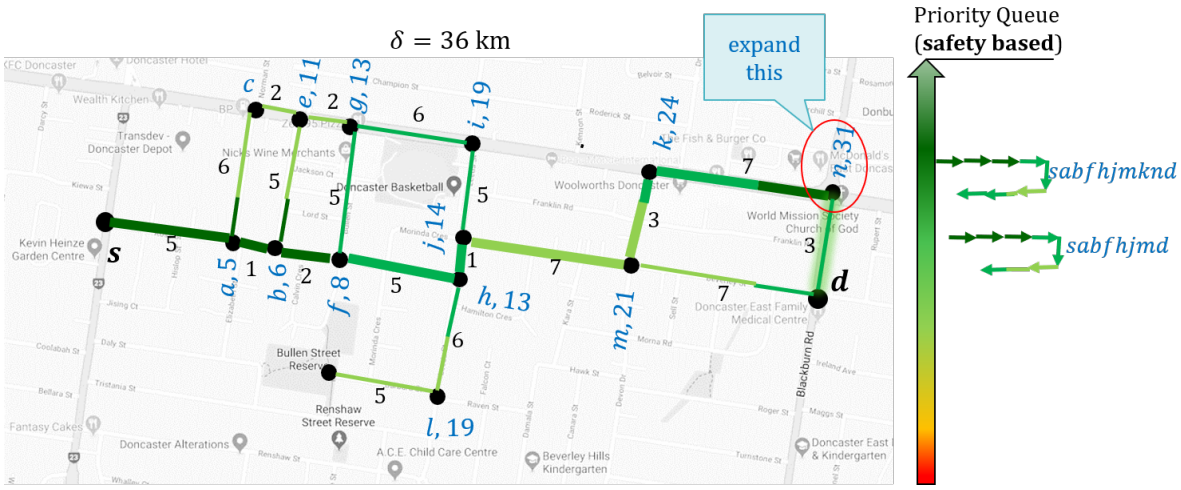


(m)

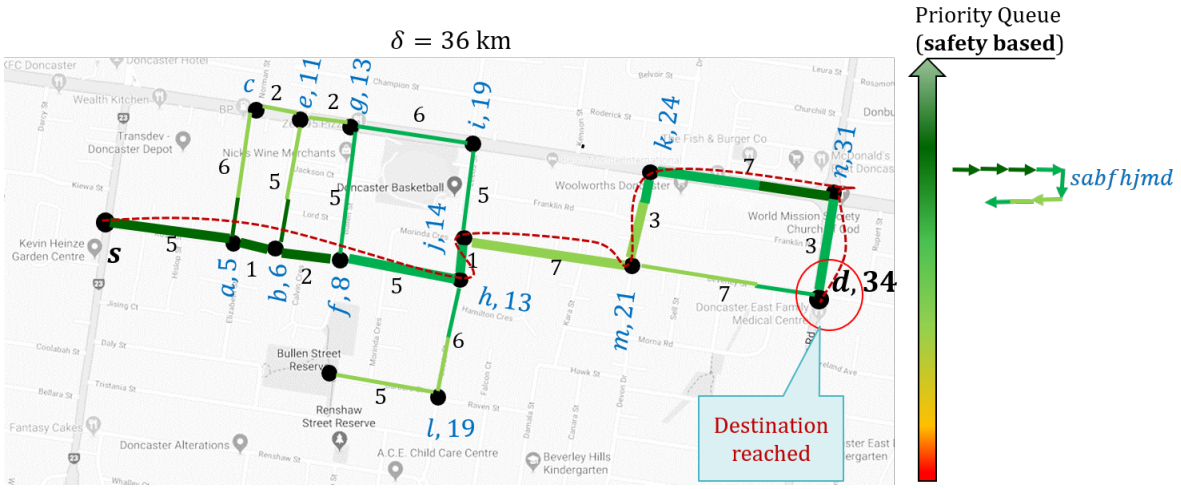


(n)

Figure 4.6: Simulation of Dir_OA (continued)



(o)



(p)

Figure 4.6: Simulation of Dir_OA (continued)

routes $s \to a \to b \to f \to g \to e$ and $s \to a \to b \to f \to g \to i$ and enqueues them (Figure 4.6e).

6. Dir_OA removes the top of the queue over again, and the route $s \to a \to b \to f \to h$ is dequeued. It tracks the dequeued route's length, 13 km, in vertex h. It forms two new routes $s \to a \to b \to f \to h \to j$ and $s \to a \to b \to f \to h \to l$, and enqueues them (Figure 4.6f).

7. Furthermore, the route $s \to a \to b \to f \to h \to j$ is dequeued from the top of the route, and its length, 14 km, is tracked in vertex j. The algorithm forms two new routes $s \to a \to b \to f \to h \to j \to i$ and $s \to a \to b \to f \to h \to j \to m$ and enqueues them (Figure 4.6g).

8. Following that, Dir_OA dequeues $s \to a \to b \to f \to g \to i$ from the top of the queue, and tracks its length, 19 km, in vertex i. It forms two new routes

$s \to a \to b \to f \to g \to i \to g$ and $s \to a \to b \to f \to g \to i \to j$. The first newly formed route is pruned because it is not a simple route. Moreover, the second one is also pruned; because the length of the second newly formed route is 24 km, however, we observe that vertex j has been reached by a safer route of a smaller length, 14 km, already (pruning condition 2) (Figure 4.6h).

9. Dir_OA removes the top of the queue again and retrieves $s \to a \to b \to f \to h \to j \to i$. It prunes the newly formed route $s \to a \to b \to f \to h \to j \to i \to g$ of length 25 km because vertex g has been reached by a safer route of length 13 km (Figure 4.6i).

10. The algorithm dequeues $s \to a \to b \to f \to g \to e$ from the top of the queue and tracks the length 15 km in the vertex e. However, it cannot enqueue the newly formed route $s \to a \to b \to f \to g \to e \to c$ of length 17 km because $17 + \text{euclidean\_dist}(c, d) = 17 + 26 = 43$ km exceeds the distance constraint 36 km (Figure 4.6j).

11. The algorithm dequeues $s \to a \to b \to e$ from the top of the queue and updates the tracked length to 11 km in vertex e. It cannot enqueue the newly formed route $s \to a \to b \to e \to c$ due to pruning condition 1, and $s \to a \to b \to e \to g$ due to pruning condition 2 (Figure 4.6k).

12. After that, $s \to a \to b \to f \to h \to l$ is dequeued from the top of the queue and the tracked length is set to 19 km in vertex l. Dir_OA cannot enqueue the newly formed route $s \to a \to b \to f \to h \to l \to p$ because of pruning condition 1 (Figure 4.6l).

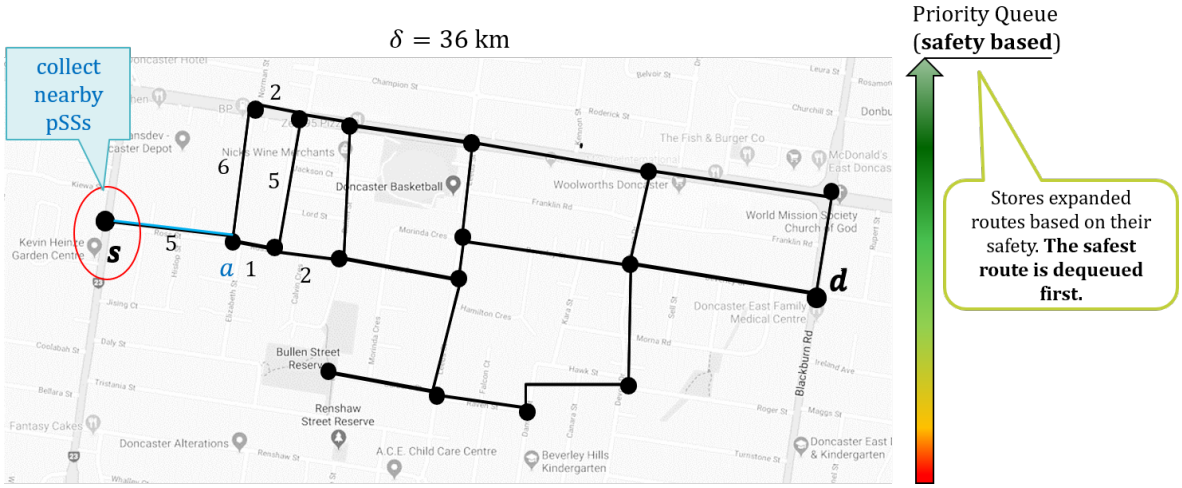
13. After that, it dequeues $s \to a \to b \to f \to h \to j \to m$ from the top of the queue and sets the tracked length to 21 km in vertex m. It enqueues the two newly formed route $s \to a \to b \to f \to h \to j \to m \to d$ and $s \to a \to b \to f \to h \to j \to m \to k$ (Figure 4.6m).

14. Next, it dequeues $s \to a \to b \to f \to h \to j \to m \to k$ from the top of the queue and sets the tracked length to 24 km in vertex k. Dir_OA enqueues the newly formed route $s \to a \to b \to f \to h \to j \to m \to k \to n$ to the queue (Figure 4.6n).

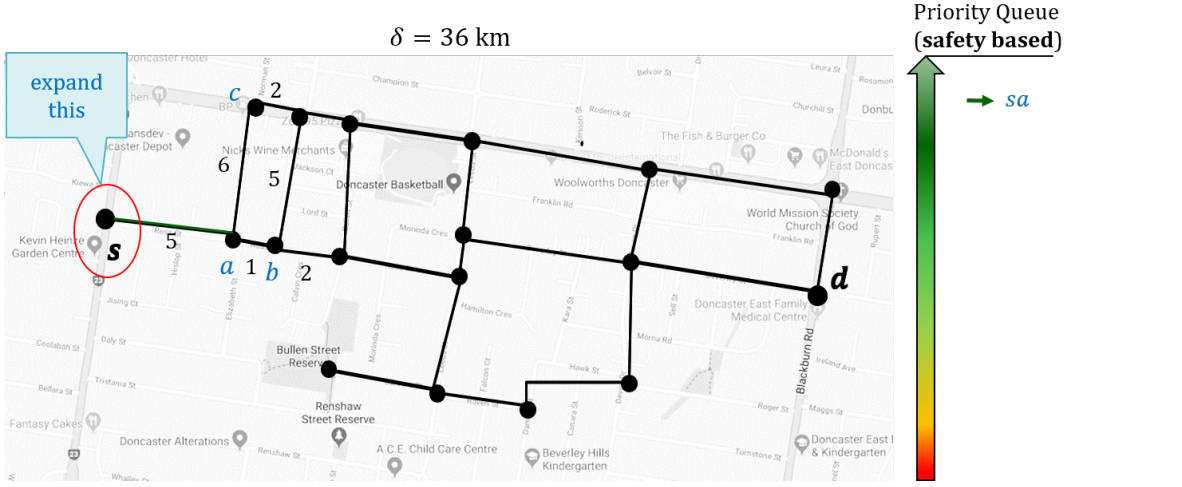
15. Then, it dequeues $s \to a \to b \to f \to h \to j \to m \to k \to n$ from the top of the queue and sets the tracked length to 31 km in vertex n. It enqueues the newly formed route $s \to a \to b \to f \to h \to j \to m \to k \to n \to d$ to the queue (Figure 4.6o).

16. Finally, this algorithm dequeues the route $s \to a \to b \to f \to h \to j \to m \to k \to n \to d$ of 34 km from the top of the queue and observes that we have reached destination d within distance constraint. Therefore, this is the SR of the given query (Figure 4.6p). The computation ends here.

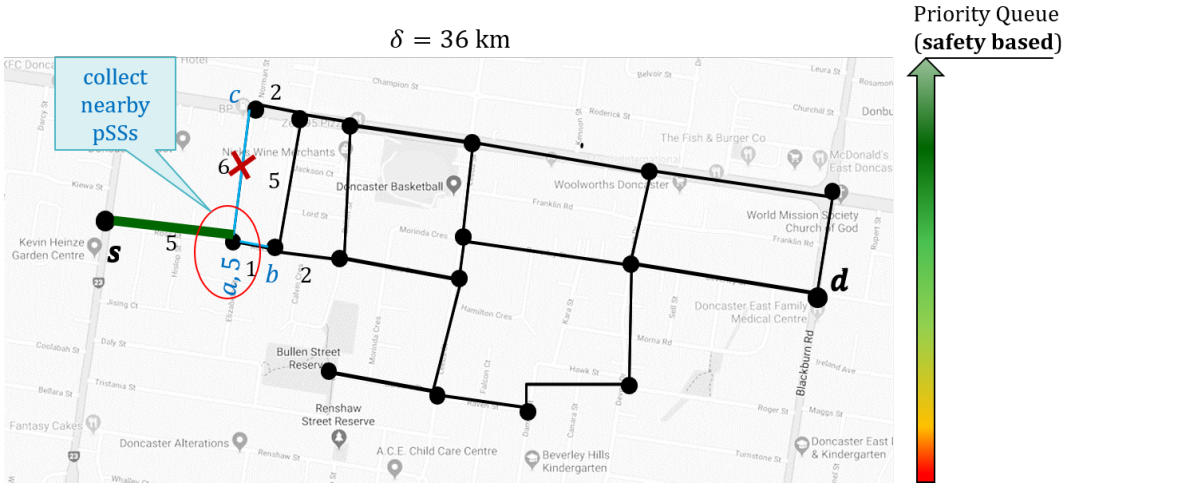
Simulation of It_OA



(a)

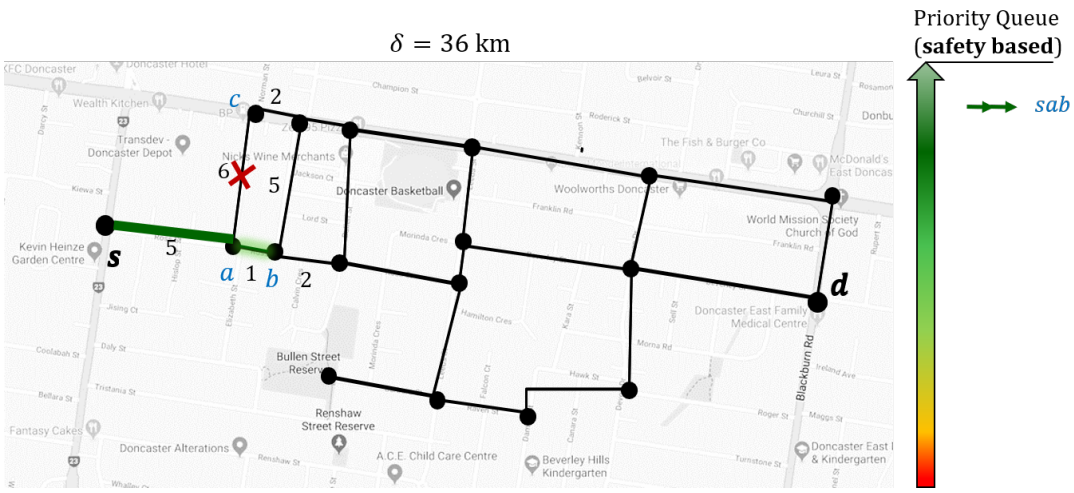


(b)



(c)

Figure 4.7: Simulation of It_OA



(d)

Figure 4.7: Simulation of It_OA

We shall simulate It_OA to compute the SR of the given query here. Line 1 and 2 of Algorithm 2 are demonstrated through Figures 4.5a- 4.5c. Like the direct algorithm, the iterative algorithm refines the search space elliptically (same as Figures 4.5a and 4.5b) and computes the query-relevant group u_1, u_2, u_3, u_4 (same as Figure 4.5c). However, it does not retrieve pSSs at once. That is also why it does not perform the binary search-based pruning; it avoids collecting unneeded pSSs. In Line 3 to 8, It_OA expands the search from the source using a priority queue based on safety. We illustrate this expansion of search in detail through Figure 4.7 and describe it in the following.

1. The search starts from the source s. First, s is enqueued in the queue. After that, the top of the queue is removed, and s is dequeued. To expand from s, we need to know the SSs of the edge $s \rightarrow a$. Therefore, It_OA retrieves the pSSs of the grid cells that intersects with the edge $s \rightarrow a$ and computes the SSs (Figure 4.7a). After that, the route $s \rightarrow a$ (Figure 4.7b) is enqueued.

2. The top of the queue is removed again, and thus, the route $s \rightarrow a$ is dequeued. To expand from vertex a, It_OA needs to know the SSs of the edges $a \rightarrow b$ and $a \rightarrow c$. However, according to pruning condition 1, $s \rightarrow a \rightarrow c$ will be pruned; thus, it only retrieves the pSSs of grid cells of the edge $a \rightarrow b$ and computes the SSs (Figure 4.7c). Next, this algorithm enqueues the route $s \rightarrow a \rightarrow b$ (Figure 4.7d).

3. This process continues until the destination is reached.

Computing the safest route in this way can incur high communication costs. Therefore, while expanding a vertex v, It_OA computes the SSs of the outgoing edges of v up to depth X_{it} to

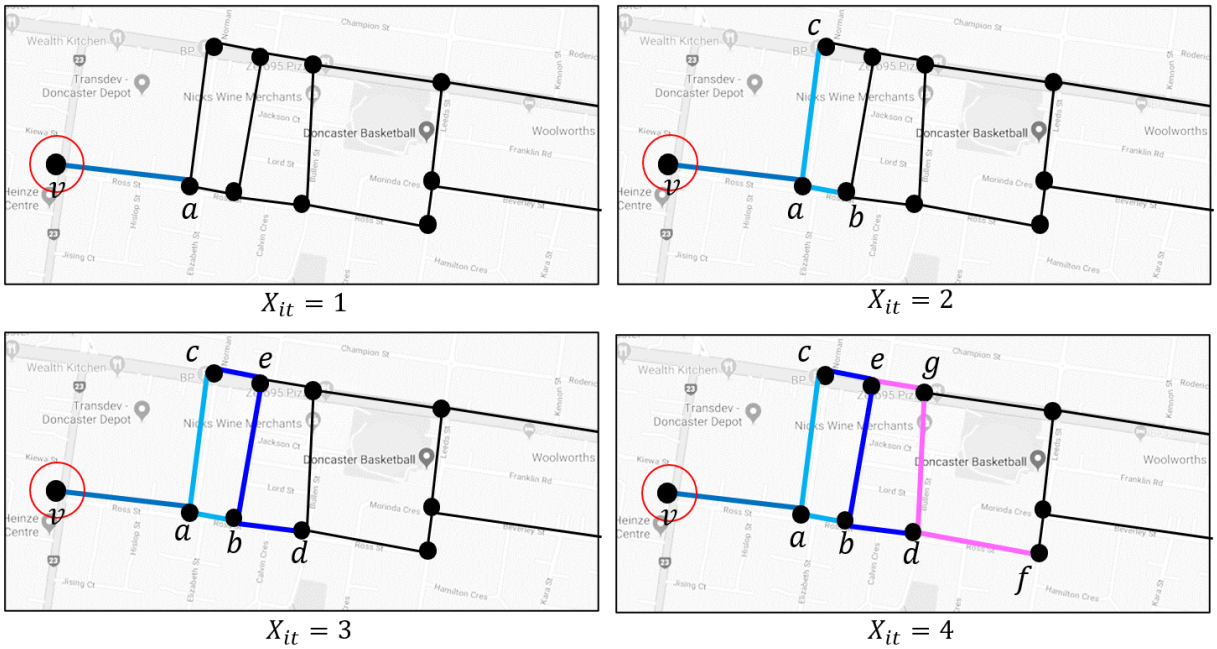


Figure 4.8: Simulation of X_{it}

reduce the communication cost. This has been illustrated in Figure 4.8, where the edges whose SSs are computed are colored in shades of blue and pink. When expanding from vertex v, if $X_{it} = 1$, then the algorithm computes the SSs of the edge $s \to a$. If $X_{it} = 2$, then it computes the SSs of the edges $s \to a$, $a \to b$, $a \to c$. If $X_{it} = 3$, then it computes the SSs of the edges $s \to a$, $a \to b$, $a \to c$, $b \to d$, $b \to e$, and $c \to e$. And if $X_{it} = 4$, then it computes the SSs of the edges $s \to a$, $a \to b$, $a \to c$, $b \to d$, $b \to e$, $c \to e$, $d \to f$, $d \to g$ and $e \to g$. However, if some of the edges can be exempted from expanding due to pruning conditions 1 and 2, then those edges' SSs are not computed.

4.5 Privacy Analysis

In this section, we empirically prove that our system preserves privacy. We also highlight the privacy-preserving features of our safe route planning system. Section 4.5.1 provides the proof that our system maintains privacy. We discuss other privacy-enhancing measures that we have taken to protect the sensitive data of users in Section 4.5.2.

4.5.1 Privacy Guarantee

Our solution ensures that an unsafe event type that a user faces cannot be inferred from the user's pSSs and KSs. A KS does not reveal the frequency and the time of the user's visits (event) in an area. It only discloses whether the user visited an area or not. Thus, even if the adversary knows that an unsafe event occurs at a grid cell, the user's KS for the grid cell does not provide any clue for the adversary to associate the unsafe event with the user and reverse engineer the user's pSS.

Quantification model parameters like the impact of an event type (ξ), recency (r_d and Δ_d), frequency, and the distance between the grid cell location and the event location ($dist$) contribute to the computation of pSSs based on the events that a user has encountered (please see Section 4.2). The following lemma shows the condition required for hiding an unsafe event type that a user encounters from others.

Lemma 1. *Given a user's revealed pSS Ψ for a grid cell and the values for quantification model parameters: ξ, r_d and Δ_d, the unsafe event type that a user encounters cannot be inferred from Ψ if (i) more than one event combinations cause the model to result in Ψ and (ii) every unsafe event type is not included in at least one event combination that result in Ψ.*

Proof. The contributing factors of the model parameters in computing a pSS change with the type, location, time and frequency of an event in an event combination, where an event combination consists of any number of events. Since an adversary does not know about any unsafe event faced by the user, the adversary cannot identify the actual event combination that results in Ψ for the user and cannot infer the user's unsafe event type from the combination. Again, if an unsafe event type is included in all possible event combinations that result in Ψ then the adversary can easily identify the user's unsafe event type. However, the second condition ensures that an unsafe event type is not included in at least one event combination that results in Ψ and thus, does not allow the adversary to infer the user's unsafe event type from Ψ. □

Thus, our system can refrain others from knowing the unsafe event type that a user encounters by selecting the values for the model parameters in a way that satisfies the condition of Lemma 1 for every possible pSS in the range $[-S, S]$.

Empirical method. We show an empirical method that can validate whether the chosen values for the model parameters are appropriate for ensuring privacy. Since an adversary does not know a user's events, it is sufficient to validate for any event setting. Without loss of generality, we consider the events of 3 days, where one event occurs per day in a grid cell. We allow $dist$

for an event to be either 0 or 1. For every pSS, we compute the possible event combinations that result in the pSS and checks whether the lemma condition is satisfied. For simplicity, we assume that there are three event types with impact $\{-3, -5, +2\}$, $r_d = 0.5$ and $\Delta_d = 1$. For the above-mentioned event setting and parameter values, we find that the condition of Lemma 1 is satisfied for every pSS, and the number of event combinations per pSS is in the range [304k, 4724k] and the average is 2869k. We leave the detailed study for generating the rules for selecting the parameter values that satisfy Lemma 1 as our future work.

4.5.2 Privacy-Enhancing Measures

In our system, the following measures further enhance the privacy of user data related to the user's travel experience.

- We refine the search space and minimize the number of shared pSSs with the QR. Since a negative pSS reveals that a user faced an unsafe event (not the type), reducing the number of shared pSSs enhances user privacy.
- In our system, a user shares pSSs with the QR instead of a centralized server (CS). A CS is fixed and thus, a user would not feel comfortable to share all pSSs with the CS, whereas a QR changes with a query and the user only shares limited query-relevant pSSs with the QR.
- The user can choose not to share her KSs for sensitive areas with the CS.
- The system restricts a QR if the QR shows suspicious behaviours, such as sending queries frequently or queries with source and destination locations distributed far apart in the road network.

Our solution does not need to store the event data. It transforms a user's events to pSSs and stores them on the local device. Thus, an adversary can only retrieve a user's pSSs by applying a malicious attack on the local device. It is not possible to infer the unsafe event type that the user encounters from the pSSs.

Chapter 5

Experiments

We evaluate our safe route planner on real datasets with experiments in both simulated and real environments. Since there is no solution that can find the SRs *in our problem setting* (please see Chapter 3), we evaluate the performance of our query processing algorithms by varying a wide range of parameters. Finally, We compare our solution with the centralized model and show the impact of the missing data on the quality of the SRs.

We discuss the datasets and parameters in Section 5.1. We evaluate the performance of our query processing algorithms in Section 5.2. In Section 5.3, we compare our work with the centralized model. Then, we evaluate the effectiveness of the SRs with respect to the shortest routes in Section 5.4. Finally, in Section 5.5, we discuss the effects of datasets on the experiment result.

Table 5.1: Datasets

Dataset (30 days)	#Users	#Checkins	#Crimes	Road Network	
				#Nodes	#Edges
Chicago (**C**)	3554	60922	30843	28468	74751
Beijing (**B**)	87	-	-	33923	75131
Philadelphia (**P**)	2275	26923	82363	24800	59987

5.1 Experiment Setup

5.1.1 Datasets

We use datasets of three cities: Chicago (C), Philadelphia (P) and Beijing (B). To simulate the environment, for each dataset, we need the road network data, the crime data, and the users' visit data to different areas. We use OpenStreetMap [80] to download the road networks. We use the real crime data of Chicago [81] and Philadelphia [82]. For Beijing, instead of crime data, only the locations of crime hotspots [83].

For the users' visit data to different areas, we use the day-to-day Foursquare check-in dataset [84, 85] for Chicago and Philadelphia, and real trajectory data of users for Beijing [86]. We use crime and check-in data of the same 6 months for Chicago and Philadelphia and one year trajectory data of 87 users for Beijing. To increase the number of events, we map these data to one month (30 days), otherwise, their effects will not be visible due to decay. All of the experiments are run on the 31st day. The details of these datasets are summarized in Table 5.1. We use datasets of three cities to show the performance of our solution irrespective of the variation in the number of users, check-in and crime data.

From check-in data, we generate the users' visits. Specifically, we take two consecutive check-ins of a user in a day and generate an elliptical area, where the foci of the ellipse are located at the check-in locations and the length of the major axis equals to 1.25 times the distance between two check-in locations. We consider that the user visited the grid cells in the elliptical area. On the other hand, the user trajectories in Beijing directly provide the grid cell area visited by the users. Since most of the trajectory data is located around the center of Beijing city, we consider the area ([39.7, 40.12, 116.1, 116.6]) around the center of Beijing for our experiments.

We normalize the crime count in the range [0,1] per grid cell for each day. This count represents the crime probability of each grid cell. For each grid cell, according to the crime probability, we randomly associate the crime events with the visits of the users. Thus, the probability of experiencing crime in a grid cell increases for a user who visits the cell multiple times. The visits of the users that are not associated with any crime are considered safe events. The pSSs are calculated based on the model of Chapter 4.2. We choose the model parameters in a way that satisfies Lemma1 for every pSS.

Table 5.2: Parameter settings

Metric	Range	Default
Query Distance, d_q (km)	1, 2, 3, 4, 5	5
Grid Size, d_G	300, 500, 800	500
Distance Constraint, δ	1.1, 1.2, 1.3, 1.4, 1.5	1.2
Confidence Level Parameter, z (%)	25, 50, 75, 100	50

5.1.2 Parameters

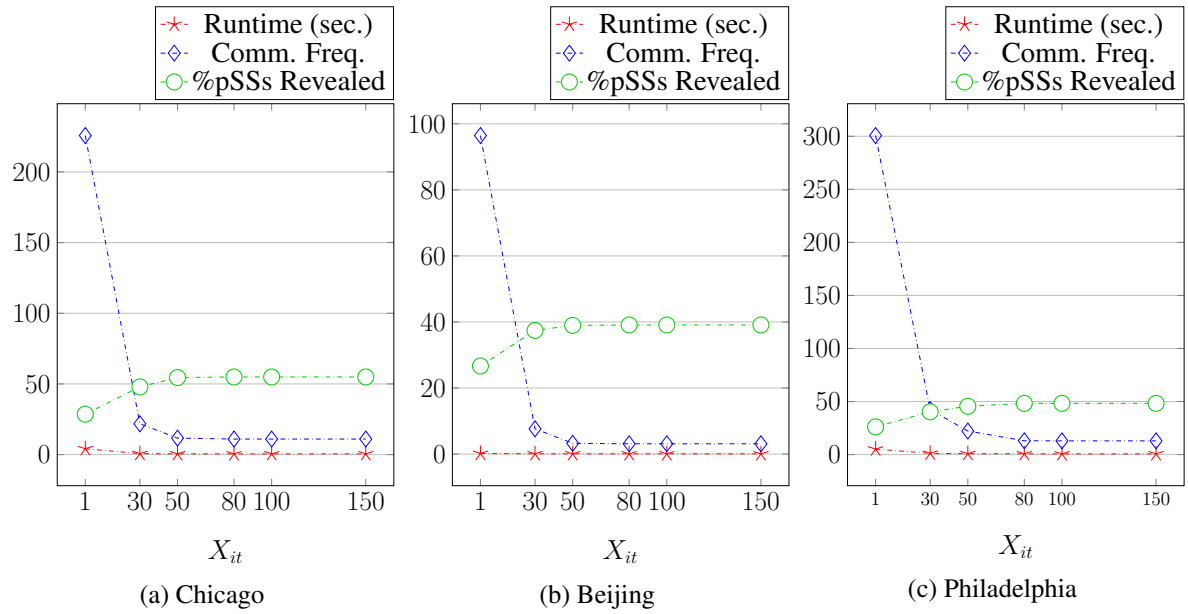
We show the parameters' default values and ranges in Table 5.2. Similar to [4], we vary the query distance d_q, the Euclidean Distance between s and d, from 1 to 5. The parameter d_G represents the grid size: $d_G \times d_G$. The range of d_G changes the grid cell area within 30x30 to 150x150 square meters and we vary d_G to show the impact of the grid resolution (and the grid cell area) on our solution performance. The distance constraint $\delta = \delta_R \times dist_{shortest}$, where $dist_{shortest}$ represents the shortest road network distance from s to d and δ_R represents the ratio of the allowed road network distance of the safest route and the road network distance of the shortest route from s to d. We keep δ_R at most 1.5 as a user may not feel comfortable travelling longer than 1.5 times of the shortest distance. The parameter z is used for confidence level (Section 5.3.2). For each experiment, we set $S = 10$ because a smaller S does not capture the variation of safety and a large S increases the computation cost by adding insignificant detail. We generate 100 safest route queries randomly and take the average performance. Our system is written in Java. We run our experiments on an Intel Core i7-7770U 3.60 GHz CPU and 16GB RAM machine.

5.2 Comparison of Query Evaluation Algorithms

We provide two optimal query processing algorithms, Dir_OA and It_OA, respectively. We compare the algorithms based on runtime, communication frequency per involved group member (*comm. freq.*), and the total number of revealed pSSs. The runtime of a query consists of the time to calculate the distance constraint δ from δ_R and the steps shown in the pseudocodes (Algorithm 1 or 2). We assume the pSSs are retrieved parallely from the group members. The KSs and pSSs are updated in offline (i.e., when a user's device is idle), not during query evaluation. Therefore, they do not affect the query response time. In addition, note that the fewer the number of revealed pSSs, the better the privacy is. We append the initial letter of the dataset

after the algorithm name with a hyphen in Figure 5.2.

5.2.1 Choosing the default value of X_{it}



(a) Chicago (b) Beijing (c) Philadelphia

Figure 5.1: Choosing default value $X_{it} = 50$ based on the effects of X_{it}

The parameter X_{it} significantly impacts the performance of It_OA. Figure 5.1 shows clear trade-offs among performance metrics for Chicago. The runtime decreases (desirable), the communication frequency decreases (desirable) and more pSSs are revealed (undesirable) with the increase of X_{it}. Thus, we have to carefully choose a value for X_{it} so that the communication frequency is low, and the runtime and the number of revealed pSSs are reasonable. From the figure, it is clear that there is a saturation point after which all three performance metrics do not change much. Therefore, we choose $X_{it} = 50$ as the default value for all datasets because we see a sharp decrease in the communication frequency and then for $X_{it} >= 50$ there is not much change. Please note that X_{it} can be used as a regulator to control the runtime, communication frequency and data exposure. In scenarios where privacy matters more, we can choose a lower value for X_{it} and in scenarios where the communication frequency matters, we should choose the value of X_{it} for which communication frequency reaches a saturated value.

5.2.2 Comparison of Dir_OA and It_OA

Figures 5.2a- 5.2c show that It_OA reveals around 50% of the revealed pSSs in Dir_OA. The number of revealed pSSs increases with an increase of δ and d_q because the length of SR

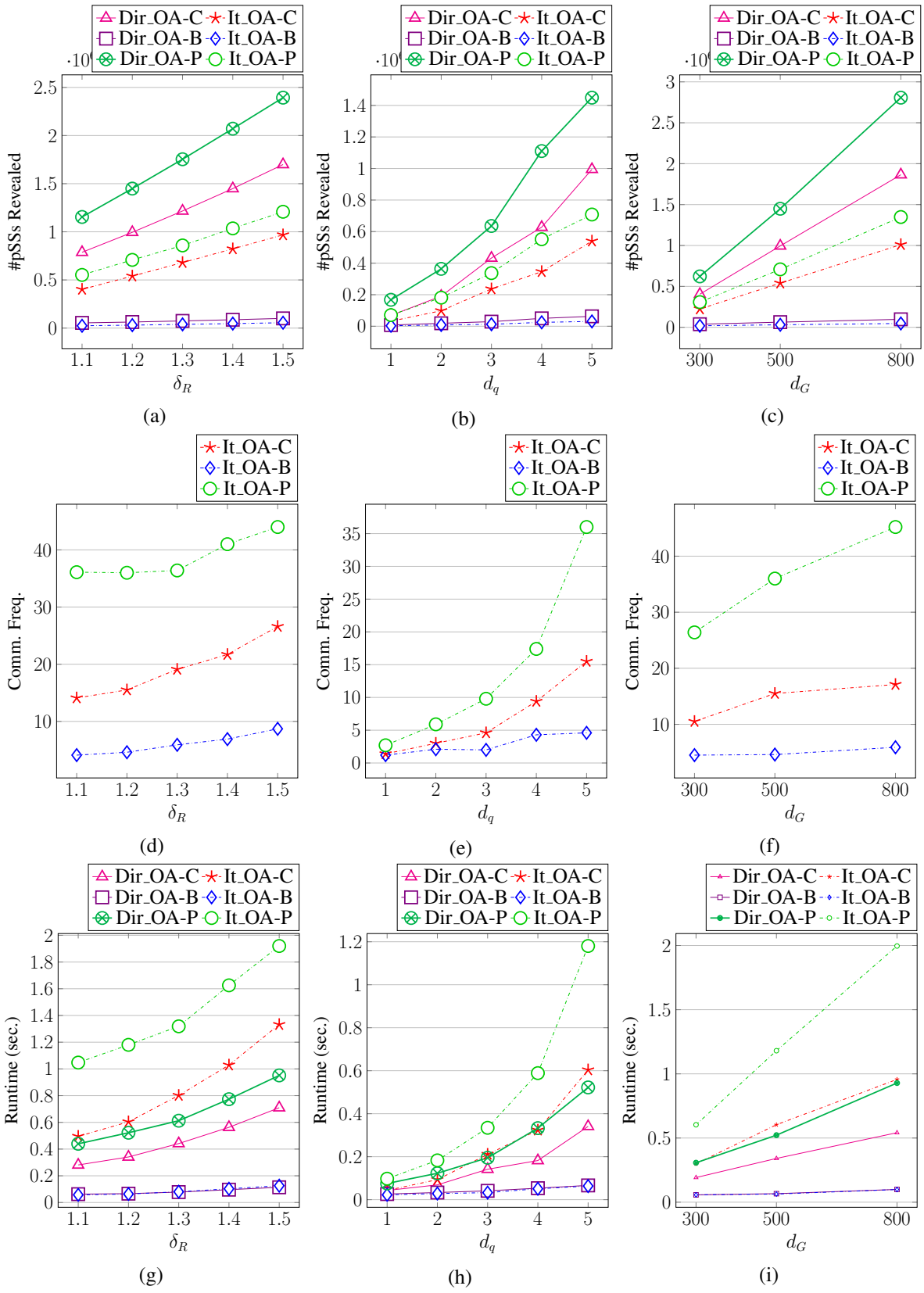


Figure 5.2: Dir_OA vs. It_OA in terms of privacy (#pSSs revealed) and computation cost (comm. freq. and runtime) for varying δ, d_q and d_G

increases. The number of revealed pSSs also increases for large d_G because the number of grid cells through which the route passes increases. Please note that the number of revealed pSSs increases more rapidly for Dir_OA than that of It_OA.

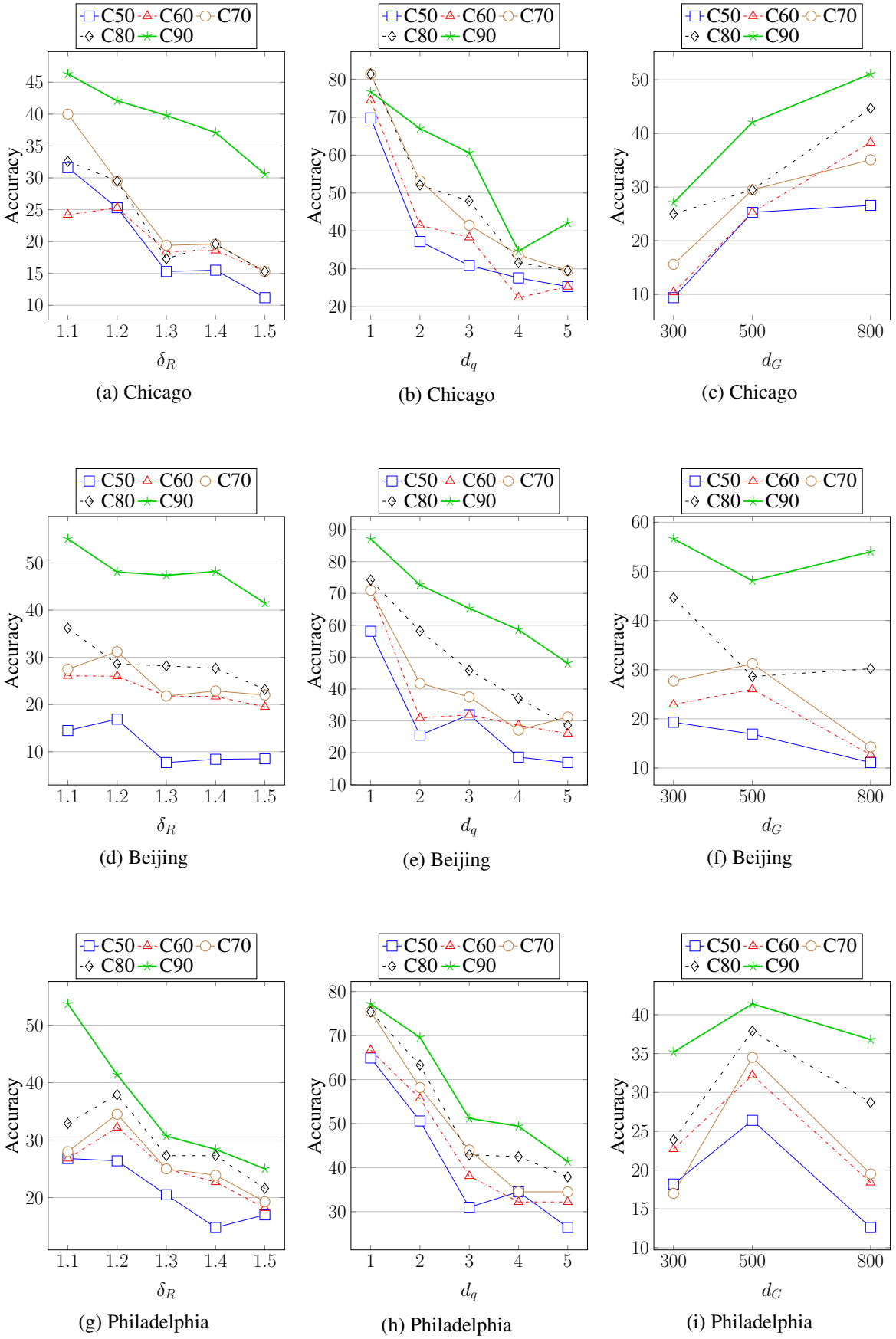
Figures 5.2d- 5.2f compare Dir_OA and It_OA in terms of communication frequency. The communication frequency is always 1 for Dir_OA as the group-members are requested once to provide pSSs. For It_OA, the communication frequency is on average 15 times; it can be as high as 45.2 times (Philadelphia dataset). To check how reasonable this is, we ran an experiment: a message is sent from one device to another using the Firebase Cloud Messaging service and a reply from the recipient is received. This is a cycle, and we ran 500 such cycles which took a total of 86641 ms, so on average, 173.28 milliseconds per cycle. Therefore, 45.2 communications take 45.2×173.28 milliseconds ≈ 8 seconds, which is acceptable. The communication frequency for It_OA increases with the increase of δ, d_q and d_G. For δ and d_q, the reason behind the increased communication frequency is that the route length increases, whereas, for d_G, the reason is that the number of required grid cells to compute the SR increases.

Figure 5.2g- 5.2i compare Dir_OA and It_OA in terms of runtime. The runtime of Dir_OA is very low (on average 0.3 second) for all datasets. Though the runtime of It_OA increases with the increase in δ, d_q and d_G, they are reasonable (on average 0.54 second). Therefore, we conclude that both Dir_OA and It_OA provide practical solutions for the SR queries and show a trade-off between runtime and privacy.

5.3 Comparison with the Centralized Model

A centralized architecture assumes that users share their travel experiences with a centralized server (CS) without considering privacy issues. However, in reality, this does not happen and the centralized solution has missing data. We investigate the impact of missing data on the quality of SRs.

As mentioned before, there exists no solution for finding SRs in our problem setting. Thus, for this experiment, we adopt our solution for the centralized model, where users share pSSs with the CS. We compare the accuracy and confidence level of our system with the centralized architecture. We vary the percentage of available data for the centralized model as 50%, 60%, 70%, 80%, and 90% and denote them with C50, C60, C70, C80, and C90, respectively.



(a) Chicago (b) Chicago (c) Chicago

(d) Beijing (e) Beijing (f) Beijing

(g) Philadelphia (h) Philadelphia (i) Philadelphia

Figure 5.3: Accuracy loss in the centralized model for missing data. C50 means 50% of actual data is present.

5.3.1 Accuracy

In our system, users do not hesitate to share their pSSs as there is no fear of privacy violation. Thus, our system always provides the actual SR. We measure the accuracy as the percentage of the answers that are within the top-5 SRs. Figure 5.3 show that the average accuracy increases with an increase in user data (25.4% for C50 and 49.9% for C90). Even 10% missing data causes significant (50.1%) accuracy loss. Hence it is important to adopt privacy-preserving solution to find the SRs. For the same amount of available data, the accuracy decreases with the increase of δ and d_q, because the number of possible routes from s to d increases. The accuracy does not depend on d_G (Figures 5.3c, 5.3f, 5.3i).

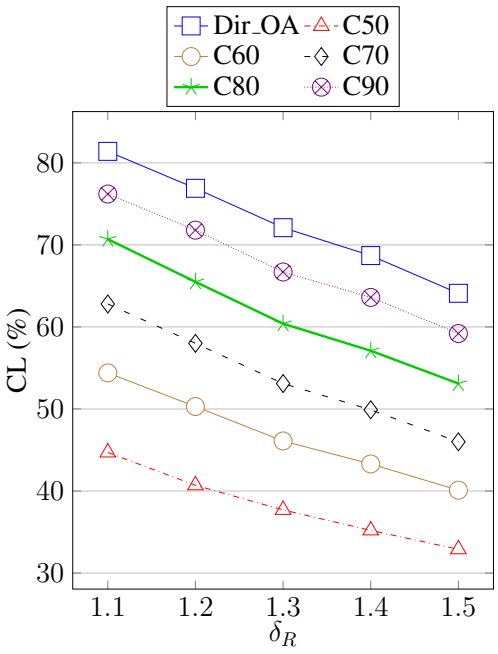
5.3.2 Confidence Level

The confidence level of a query answer expresses its reliability from the viewpoint of a QR. In our case, the more the number of users supports an answer, the more reliable it is to the QR. For a route R, its confidence level $CL(R)$ is expressed as follows.

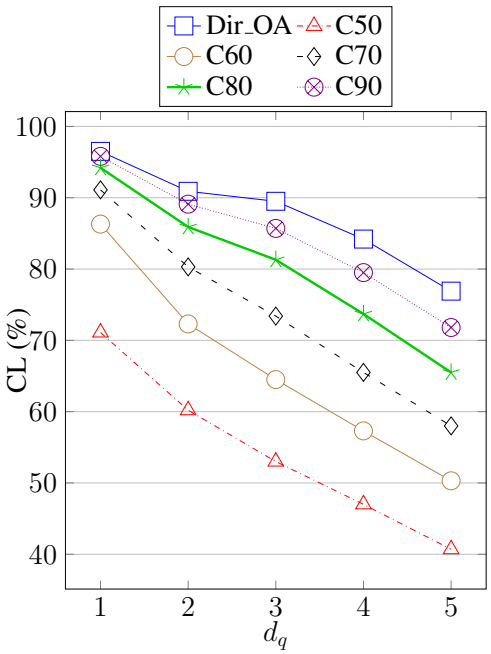
$$CL(R) = \frac{100}{z} \times \frac{\sum_{c_i} l_i \times m_{c_i}}{dist(R) \times m}$$

Here, l_i is the length of R that crosses grid cell c_i and m_{c_i} is the number of group members who know c_i. Intuitively, the confidence level indicates the average percentage of query-relevant group members who know each unit length of the route. The QR might be satisfied when on average $z\%$ members among the m query-relevant group members know about each unit length. Thus, we include z in the definition of the confidence level. The parameter z is varied within $\{25, \mathbf{50}, 75, 100\}$ to cover the full range.

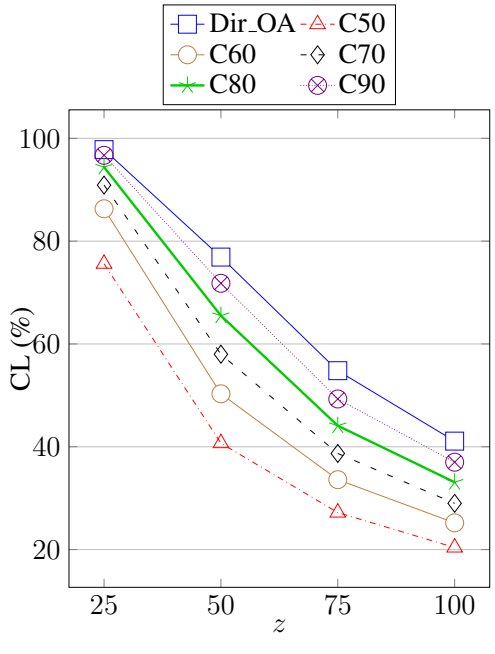
Figures 5.4, 5.5 and 5.6 show that the confidence level for our system is always the highest (on average 75.7%). Since both Dir_OA and It_OA provide optimal solutions, their confidence level is the same. In the centralized model, confidence level predictably decreases with the increase of missing data. confidence level decreases when the SRs become longer (for δ and d_q). No particular trend is visible for d_G. Philadelphia dataset shows same trends as Chicago. The CL decreases with the increase in d_G as the SR contains more cells and ensuring on average 50% (the default value of z is 50) knowledgeable users per cell becomes more difficult. For an increase in z, the CL decreases as expected.



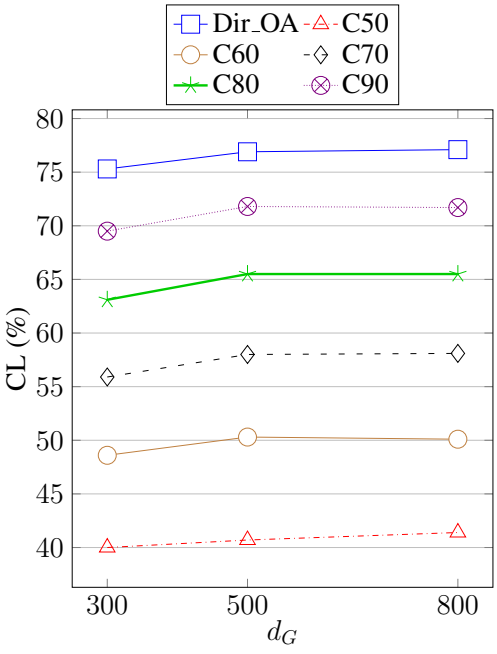
(a) Chicago



(b) Chicago

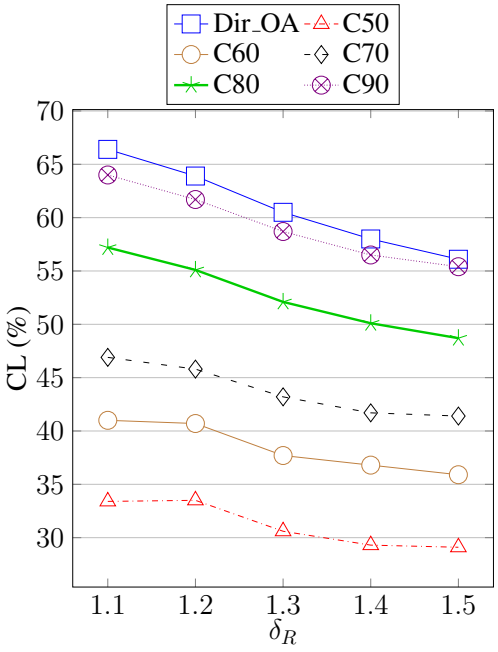


(c) Chicago

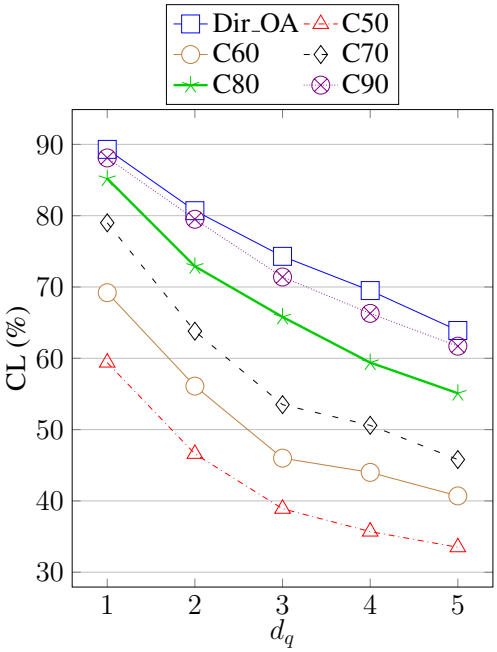


(d) Chicago

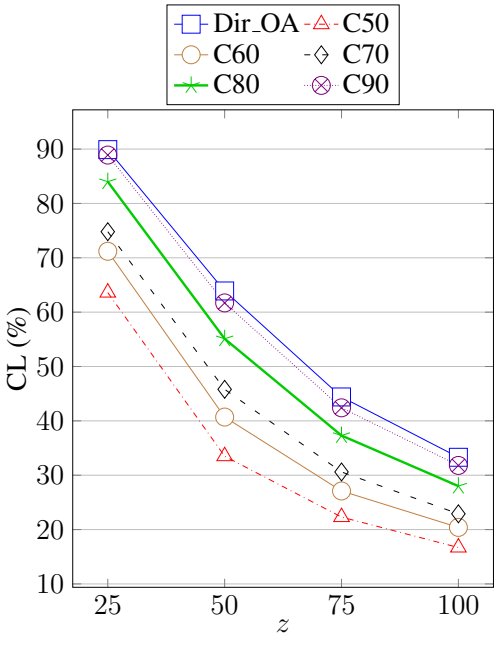
Figure 5.4: Confidence level for our system is higher than that of the centralized model (Chicago dataset).



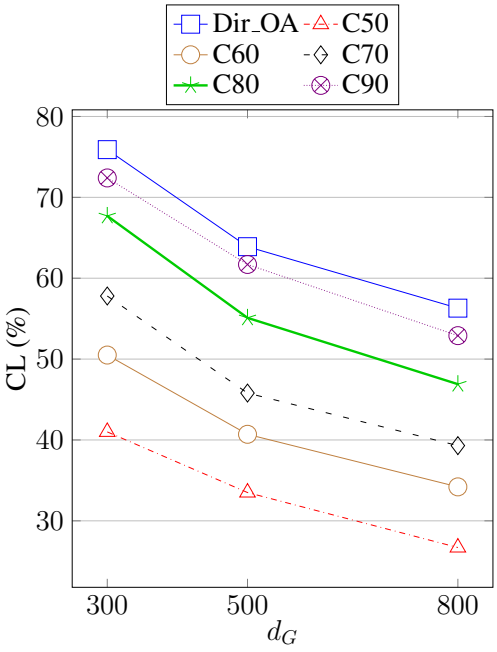
(a) Beijing



(b) Beijing

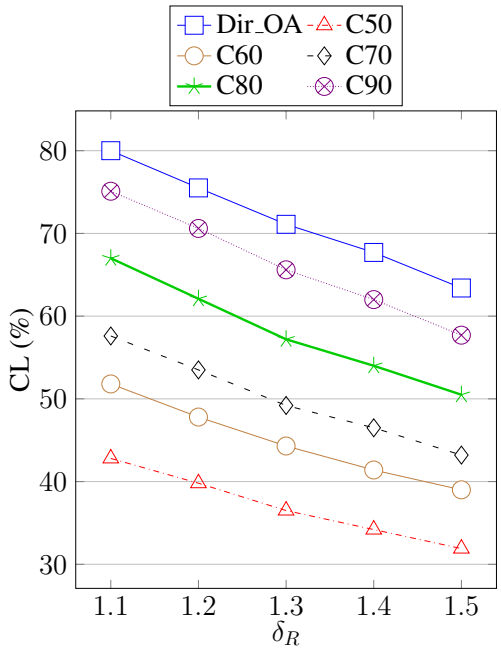


(c) Beijing

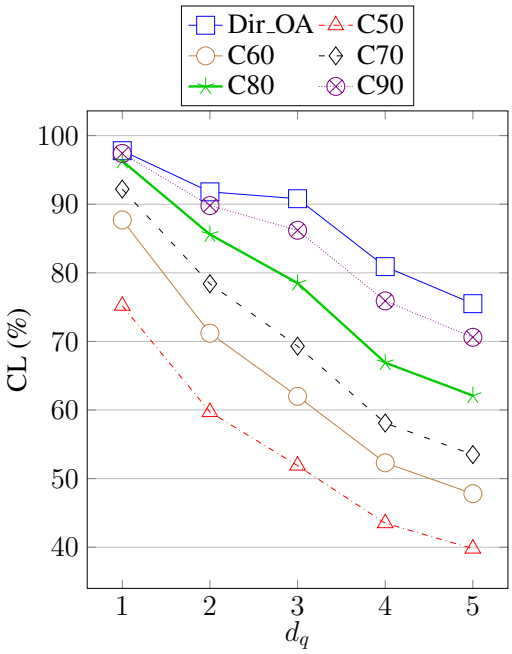


(d) Beijing

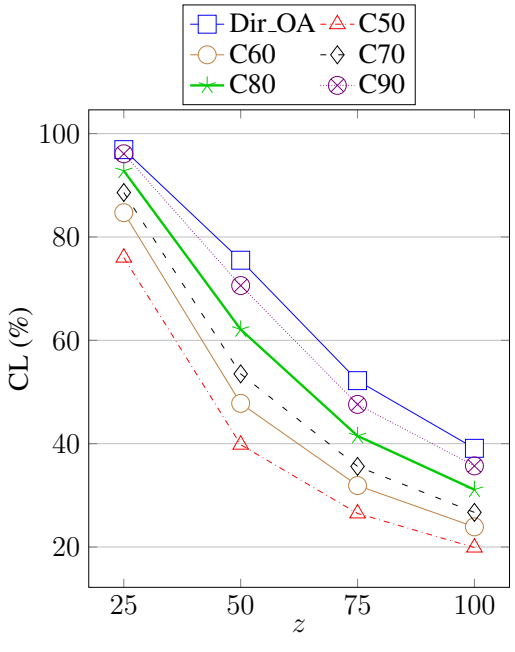
Figure 5.5: Confidence level for our system is higher than that of the centralized model (Beijing dataset).



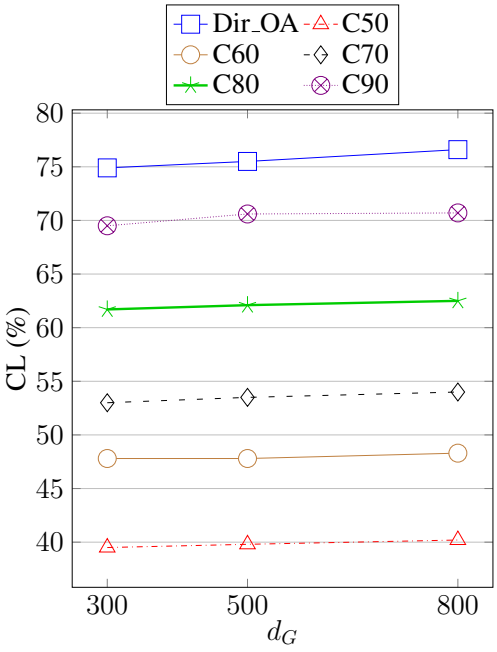
(a) Philadelphia



(b) Philadelphia



(c) Philadelphia



(d) Philadelphia

Figure 5.6: Confidence level for our system is higher than that of the centralized model (Philadelphia dataset).

5.4 Safest Route versus Shortest Route

Dataset	$K = 1$	$K = 10$	$K = 50$	$K = 100$	$K = 250$	$K = 500$
Chicago	1.02	3.06	3.06	3.06	3.06	3.06
Beijing	4.76	11.90	13.10	19.05	25.00	27.38
Philadelphia	2.27	2.27	2.27	2.27	2.27	3.41

Table 5.3: The percentage of query samples for which top-K shortest routes include the respective SRs

Dataset	$\delta_R = 1.1$	$\delta_R = 1.2$	$\delta_R = 1.3$	$\delta_R = 1.4$	$\delta_R = 1.5$
Chicago	1.08	1.18	1.27	1.35	1.44
Beijing	1.07	1.15	1.21	1.29	1.36
Philadelphia	1.08	1.19	1.28	1.37	1.45

Table 5.4: The actual distance constraint ratio of the lengths between the SR and the shortest route

In this set of experiments, we evaluate the effectiveness of the SRs with respect to the shortest routes. To show that, we run an experiment for 100 queries on default setting (Section 5.1). For each query, we compute the top-K shortest routes and among those routes, whose lengths are within distance constraint are considered. We compute the safety score of each of those considered routes and match with the safety score of SR. Then we check whether any of the top-K shortest routes has the same safety score as that of the safest routes. Table 5.3 shows the results of this experiment. When we consider the shortest route (K = 1), only 2.7% among them are the SRs. As we increase the value of K, the percentage increases slightly. Even for the high value of $K = 500$, it is only 27%. This is because the consecutive shortest routes (e.g., the third and the fourth ones) normally have very small differences in terms of the included roads in the routes. Table 5.4 shows the actual ratio of the lengths between the SR and the shortest route. Note that this ratio cannot exceed the distance constraint ratio δ_R.

5.5 Effect of Different Datasets

The datasets in our experiments provide variation in the number of users and road network structure and size. The runtime (Figures 5.2g-5.2i) increases for Chicago (3554 users) and Philadelphia (2275 users) due to more users compared to Beijing (87 users) even though the

runtime is still reasonable. Both communication frequency and the number of revealed pSSs are the highest for Philadelphia (Figure 5.2), which does not have the highest number of users or road network size (Table 5.1). Therefore, these two metrics might have been affected by the road network structure. The confidence level is less for Beijing than other datasets (Figure 5.4, 5.5 and 5.6), which shows its dependency on the number of users.

Chapter 6

Conclusion

In this thesis, we have developed a novel journey planner for finding SRs with crowdsourced data and computation. For this purpose, we developed a realistic safety quantification model, provided efficient indexing techniques to store our SSs and KSs, proposed two optimal algorithms to find the SRs, and validated the effectiveness and efficiency of our solution through extensive experiments.

We proposed the first privacy-preserving system architecture for computing the SRs which enhances the privacy of travelers' sensitive travel information. Our quantification model for transforming a user's travel experience into safety scores is personalized based on the user's travel parameters like frequency and recency of visits. The evaluation of the SR queries using the pSSs of the users improves the quality of the query answer. The quantification model is also storage efficient and does not allow an adversary to reverse engineer the personalized SS of a user and identify the user's precise travel data. One of our two indexing techniques modifies the R-tree for the efficient storage and fast retrieval of pSSs. The other indexing technique also identifies the efficient way to store KSs.

Both of our proposed algorithms: direct and iterative, find the optimal answer to the SR queries. Experiments show that our approach can evaluate a query in less than a second. Although the search space for finding the SR is huge, our refining techniques enable this faster processing of the SR queries. Both of our algorithms aim to protect user privacy by minimizing the number of shared pSSs with others. Our iterative query processing algorithm enhances user privacy by not revealing, on average, 50% of the pSSs revealed by the direct query processing algorithm. The direct one is better than the iterative algorithm in terms of the processing time and communication frequency. To show the credibility of the query answers, the query answers in our system come

with a personalized confidence level parameter and make the system trustworthy for the users.

In experiments, we observe that the data scarcity problem can have a significant impact on lowering the quality of SRs. For example, the actual SR is only identified for on average 34% and 38% times when a centralized route planner has 30% and 20% missing data, respectively. Thus, protecting the privacy of a user's travel experience is essential to solve the data scarcity problem and ensure the quality of the SRs. Our privacy-enhanced solution encourages more users to share their data and improves the quality of the SRs.

In the future, this work can be extended by addressing the following issues:

- An extension of this work could be the safe route planning for flexible destinations and the safe route planning for group. In the case of safe route planning for flexible destinations, the user wants to reach any one of a set of specified destinations (e.g., a number of branches of a bank or a superstore) within distance constraints. In the case of the safe route planning for group, a group of users want to meet at a fixed or flexible destination locations within distance constraints. One can use our provided direct and iterative algorithms to solve these two problems straightforwardly. However, that would require exploring the search space multiple times and incur high processing cost. In the future, more efficient algorithms can be designed to solve these two problems in a privacy-preserving way.
- Another extension of this work can adopt a tuning parameter that provides a trade-off between safety and length of the route within a distance constraint. The users (e.g. vulnerable group) can prioritize safety over length or vice versa by setting the value of this parameter.
- In our safety quantification model, we only considered regular events. Future researches could incorporate the effects of temporary events (e.g., cricket matches, political public meetings etc.) that affect the safety of an area for a limited time in our model. Moreover, further studies are needed to analyze the effects of varying the values of the model parameters, like S, in different aspects of the SR query and choose the appropriate values for these model parameters. Moreover, future studies can be conducted on the feasibility of adopting our safety quantification model for the law agency's safety data, which is not personalized.
- Peer-to-peer anonymous communication protocols [87–89] can be utilized to prevent a query requestor from collecting pSSs of a single user using multiple queries. However, further investigation is needed in the selection of this anonymous protocol.
- We considered the semi-honest adversary model in this thesis. In the future, the safe route

planning system for the malicious adversary model can be explored, where the participants can send wrong pSSs and collude with each other or the central entity. In addition, existing models can be adopted to detect suspicious behaviours of the users and decrease the security vulnerabilities of our safe route planning system.

- In the future, a working prototype of our safe route planner can be implemented and its usability can be tested in the real environment. This will also allow the measurement of the real query processing overhead for finding the SRs.

References

[1] S. I. Ahmed, S. J. Jackson, N. Ahmed, H. S. Ferdous, M. R. Rifat, A. S. M. Rizvi, S. Ahmed, and R. S. Mansur, "Protibadi: a platform for fighting sexual harassment in urban bangladesh," in *CHI*, pp. 2695–2704, 2014.

[2] T. Hashem, R. Hasan, F. D. Salim, and M. T. Mahin, "Crowd-enabled processing of trustworthy, privacy-enhanced and personalised location based services with quality guarantee," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 4, pp. 167:1–167:25, 2018.

[3] M. T. Mahin, T. Hashem, and S. Kabir, "A crowd enabled approach for processing nearest neighbor and range queries in incomplete databases with accuracy guarantee," *Pervasive and Mobile Computing*, vol. 39, pp. 249–266, 2017.

[4] E. Galbrun, K. Pelechrinis, and E. Terzi, "Urban navigation beyond shortest route: The case of safe paths," *Information Systems*, vol. 57, pp. 160–171, 2016.

[5] J. Kim, M. Cha, and T. Sandholm, "Socroutes: safe routes based on tweet sentiments," in *International World Wide Web Conference, Companion Volume*, pp. 179–182, ACM, 2014.

[6] S. Shah, F. Bao, C. Lu, and I. Chen, "CROWDSAFE: crowd sourcing of crime incidents and safe routing on mobile devices," in *International Symposium on Advances in Geographic Information Systems*, pp. 521–524, ACM, 2011.

[7] K. Fu, Y. Lu, and C. Lu, "TREADS: a safe route recommender using social media mining and text summarization," in *International Conference on Advances in Geographic Information Systems*, pp. 557–560, ACM, 2014.

[8] N. Goel, R. Sharma, N. Nikhil, S. D. Mahanoor, and M. K. Saini, "A crowd-sourced adaptive safe navigation for smart cities," in *International Symposium on Multimedia*, pp. 382–387, IEEE Computer Society, 2017.

[9] T. Hashem, M. E. Ali, L. Kulik, E. Tanin, and A. Quattrone, "Protecting privacy for group nearest neighbor queries with crowdsourced data and computing," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 559–562, ACM, 2013.

[10] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z. He, and Z. Wen, "The safest path via safe zones," in *International Conference on Data Engineering*, pp. 531–542, IEEE Computer Society, 2015.

[11] A. M. de Souza, L. C. Botega, and L. A. Villas, "Fns: Enhancing traffic mobility and public safety based on a hybrid transportation system," in *International Conference on Distributed Computing in Sensor Systems*, pp. 77–84, IEEE, 2018.

[12] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[13] K. Mehlhorn, J. Orlin, and R. Tarjan, "Faster algorithms for the shortest path problem," tech. rep., Technical Report CS-TR-154-88. Princeton University, Department of Computer Science, 1987.

[14] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.

[15] G. Gallo and S. Pallottino, "Shortest path algorithms," *Annals of operations research*, vol. 13, no. 1, pp. 1–79, 1988.

[16] A. V. Goldberg, "Scaling algorithms for the shortest paths problem," *SIAM Journal on Computing*, vol. 24, no. 3, pp. 494–504, 1995.

[17] Y. Xi, L. Schwiebert, and W. Shi, "Privacy preserving shortest path routing with an application to navigation," *Pervasive Mobile Computing*, vol. 13, pp. 142–149, 2014.

[18] A. Schambers, M. Eavis-O'Quinn, V. Roberge, and M. Tarbouchi, "Route planning for electric vehicle efficiency using the bellman-ford algorithm on an embedded gpu," in *International Conference on Optimization and Applications*, pp. 1–6, 2018.

[19] S. Saunders and T. Takaoka, "Improved shortest path algorithms for nearly acyclic graphs," *Theoretical Computer Science*, vol. 293, no. 3, pp. 535–556, 2003.

[20] Y. Han, "Improved algorithm for all pairs shortest paths," *Information Processing Letters*, vol. 91, no. 5, pp. 245–250, 2004.

[21] H. Ortega-Arranz, Y. Torres, D. R. Llanos, and A. Gonzalez-Escribano, "A new gpu-based approach to the shortest path problem," in *International Conference on High Performance Computing & Simulation*, pp. 505–511, 2013.

[22] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[23] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *Journal of the ACM (JACM)*, vol. 24, no. 1, pp. 1–13, 1977.

[24] B. Richard, "On a routing problem," *Quarterly Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[25] N. Hart, "Raphael; a formal basis for the heuristic determination of minimum cost path," *IEEE Transactions on Systems Science and Cybernetics SSC4*, vol. 4, no. 2, 1968.

[26] W. Shu-Xi, "The improved dijkstra's shortest path algorithm and its application," *Procedia Engineering*, vol. 29, pp. 1186–1190, 2012.

[27] M. Xu, Y. Liu, Q. Huang, Y. Zhang, and G. Luan, "An improved dijkstra's shortest path algorithm for sparse network," *Applied Mathematics and Computation*, vol. 185, no. 1, pp. 247–254, 2007.

[28] J. E. Beasley and N. Christofides, "An algorithm for the resource constrained shortest path problem," *Networks*, vol. 19, no. 4, pp. 379–394, 1989.

[29] G. Y. Handler and I. Zang, "A dual algorithm for the constrained shortest path problem," *Networks*, vol. 10, no. 4, pp. 293–309, 1980.

[30] E. Q. V. Martins, "On a multicriteria shortest path problem," *European Journal of Operational Research*, vol. 16, no. 2, pp. 236–245, 1984.

[31] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan, "Fuzzy dijkstra algorithm for shortest path problem under uncertain environment," *Applied Soft Computing*, vol. 12, no. 3, pp. 1231–1237, 2012.

[32] M. Randour, J.-F. Raskin, and O. Sankur, "Variations on the stochastic shortest path problem," in *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pp. 1–18, 2015.

[33] S. Broumi, A. Bakal, M. Talea, F. Smarandache, and L. Vladareanu, "Applying dijkstra algorithm for solving neutrosophic shortest path problem," in *International Conference on Advanced Mechatronic Systems*, pp. 412–416, 2016.

[34] A. W. Mohemmed, N. C. Sahoo, and T. K. Geok, "Solving shortest path problem using particle swarm optimization," *Applied Soft Computing*, vol. 8, no. 4, pp. 1643–1653, 2008.

[35] Y. Marinakis, A. Migdalas, and A. Sifaleras, "A hybrid particle swarm optimization–variable neighborhood search algorithm for constrained shortest path problems," *European Journal of Operational Research*, vol. 261, no. 3, pp. 819–834, 2017.

[36] D. J. Wu, J. Zimmerman, J. Planul, and J. C. Mitchell, "Privacy-preserving shortest path computation," *CoRR*, vol. abs/1601.02281, 2016.

[37] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *European Journal of Operational Research*, vol. 187, no. 3, pp. 985–1032, 2008.

[38] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[39] R. Jahan, T. Hashem, F. D. Salim, and S. Barua, "Efficient trip scheduling algorithms for groups," *Information Systems*, vol. 84, pp. 145–173, 2019.

[40] *How to enumerate all solutions*, Accessed: 2020-09-11.

[41] V. Gaede and O. Günther, "Multidimensional access methods," *ACM Computing Surveys (CSUR)*, vol. 30, no. 2, pp. 170–231, 1998.

[42] P. van Oosterom, "Spatial access methods," *Geographical information systems*, vol. 1, pp. 385–400, 1999.

[43] Y. Manolopoulos, Y. Theodoridis, V. J. Tsotras, and J. Vassilis, "Spatial indexing techniques.," 2009.

[44] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[45] J. L. Bentley and J. H. Friedman, "Data structures for range searching," *ACM Computing Surveys (CSUR)*, vol. 11, no. 4, pp. 397–409, 1979.

[46] J. T. Robinson, "The kdb-tree: a search structure for large multidimensional dynamic indexes," in *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pp. 10–18, 1981.

[47] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[48] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *SIGMOD'84*, pp. 47–57, 1984.

[49] T. Sellis, N. Roussopoulos, and C. Faloutsos, "The r+-tree: A dynamic index for multi-dimensional objects.," tech. rep., 1987.

[50] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The r*-tree: An efficient and robust access method for points and rectangles," in *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pp. 322–331, 1990.

[51] I. Kamel and C. Faloutsos, "Hilbert r-tree: An improved r-tree using fractals," tech. rep., 1993.

[52] A. Efentakis, D. Theodorakis, and D. Pfoser, "Crowdsourcing computing resources for shortest-path computation," in *International Conference on Advances in Geographic Information Systems*, pp. 434–437, 2012.

[53] H. Su, K. Zheng, J. Huang, H. Jeung, L. Chen, and X. Zhou, "Crowdplanner: A crowd-based route recommendation system," in *International Conference on Data Engineering*, pp. 1144–1155, IEEE Computer Society, 2014.

[54] C. J. Zhang, Y. Tong, and L. Chen, "Where to: Crowd-aided path selection," *Proceedings of the VLDB Endowment*, vol. 7, no. 14, pp. 2005–2016, 2014.

[55] A. Jain, D. Das, J. K. Gupta, and A. Saxena, "Planit: A crowdsourcing approach for learning to plan paths from large scale preference feedback," in *IEEE International Conference on Robotics and Automation*, pp. 877–884, 2015.

[56] B. Nushi, A. Singla, A. Gruenheid, E. Zamanian, A. Krause, and D. Kossmann, "Crowd access path optimization: Diversity matters," in *AAAI Conference on Human Computation and Crowdsourcing* (E. Gerber and P. Ipeirotis, eds.), pp. 130–139, 2015.

[57] T. Hultman, A. Boudjadar, and M. Asplund, "Connectivity-optimal shortest paths using crowdsourced data," in *International Conference on Pervasive Computing and Communication Workshops*, pp. 1–6, IEEE Computer Society, 2016.

[58] X. Fan, J. Liu, Z. Wang, and Y. Jiang, "Navigating the last mile with crowdsourced driving information," in *IEEE Conference on Computer Communications Workshops*, pp. 346–351, IEEE, 2016.

[59] G. Delnevo, A. Melis, S. Mirri, L. Monti, and M. Prandini, "Discovering the city: Crowdsourcing and personalized urban paths across cultural heritage," in *International Conference on Smart Objects and Technologies for Social Good*, vol. 233, pp. 132–141.

[60] G. Jossé, M. Franzke, G. Skoumas, A. Züfle, M. A. Nascimento, and M. Renz, "A framework for computation of popular paths from crowdsourced data," in *ICDE*, pp. 1428–1431, 2015.

[61] F. A. Santos, D. O. Rodrigues, T. H. Silva, A. A. F. Loureiro, R. W. Pazzi, and L. A. Villas, "Context-aware vehicle route recommendation platform: Exploring open and crowdsourced data," in *ICC*, pp. 1–7, 2018.

[62] D. Sun, K. Xu, H. Cheng, Y. Zhang, T. Song, R. Liu, and Y. Xu, "Online delivery route recommendation in spatial crowdsourcing," *World Wide Web*, vol. 22, no. 5, pp. 2083–2104, 2019.

[63] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu, "Tripplanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1259–1273, 2015.

[64] Z. Yu, H. Xu, Z. Yang, and B. Guo, "Personalized travel package with multi-point-of-interest recommendation based on crowdsourced user footprints," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 151–158, 2016.

[65] Y. He, J. Ni, B. Niu, F. Li, and X. S. Shen, "Privbus: A privacy-enhanced crowdsourced bus service via fog computing," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 156–168, 2020.

[66] Y. Cui, L. Deng, Y. Zhao, B. Yao, V. W. Zheng, and K. Zheng, "Hidden POI ranking with spatial crowdsourcing," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 814–824, ACM, 2019.

[67] X. Qian, C. Li, K. Lan, X. Hou, Z. Li, and J. Han, "POI summarization by aesthetics evaluation from crowd source social media," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1178–1189, 2018.

[68] C. Chen, D. Zhang, X. Ma, B. Guo, L. Wang, Y. Wang, and E. H. Sha, "crowddeliver: Planning city-wide package delivery paths leveraging the crowd of taxis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1478–1496, 2017.

[69] X. Fan, J. Liu, Z. Wang, Y. Jiang, and X. S. Liu, "Crowdnavi: Demystifying last mile navigation with crowdsourced driving information," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 2, pp. 771–781, 2017.

[70] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, and H. Qi, "Personalized privacy-preserving task allocation for mobile crowdsensing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 6, pp. 1330–1341, 2019.

[71] G. Singh, D. Bansal, S. Sofat, and N. Aggarwal, "Smart patrolling: An efficient road surface monitoring using smartphone sensors and crowdsourcing," *Pervasive and Mobile Computing*, vol. 40, pp. 71–88, 2017.

[72] X. Kong, F. Xia, J. Li, M. Hou, M. Li, and Y. Xiang, "A shared bus profiling scheme for smart cities based on heterogeneous mobile crowdsourced data," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 1436–1444, 2020.

[73] X. Kong, X. Song, F. Xia, H. Guo, J. Wang, and A. Tolba, "Lotad: long-term traffic anomaly detection based on crowdsourced bus trajectory data," *World Wide Web*, vol. 21, no. 3, pp. 825–847, 2018.

[74] A. Karakaya, J. Hasenburg, and D. Bermbach, "Simra: Using crowdsourcing to identify near miss hotspots in bicycle traffic," *Pervasive Mobile Computing*, vol. 67, p. 101197, 2020.

[75] A. Rahim, K. Ma, W. Zhao, A. Tolba, Z. Al-Makhadmeh, and F. Xia, "Cooperative data forwarding based on crowdsourcing in vehicular social networks," *Pervasive Mobile Computing*, vol. 51, pp. 43–55, 2018.

[76] M. Abdelaal, D. Reichelt, F. Dürr, K. Rothermel, L. Runceanu, S. Becker, and D. Fritsch, "Comnsense: Grammar-driven crowd-sourcing of point clouds for automatic indoor mapping," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 1, pp. 1:1–1:26, 2018.

[77] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: zero-effort crowdsourcing for indoor localization," in *International Conference on Mobile Computing and Networking* (Ö. B. Akan, E. Ekici, L. Qiu, and A. C. Snoeren, eds.), pp. 293–304, ACM, 2012.

[78] S. H. Jung, S. Lee, and D. Han, "A crowdsourcing-based global indoor positioning and navigation system," *Pervasive Mobile Computing*, vol. 31, pp. 94–106, 2016.

[79] S. Manen, M. Gygli, D. Dai, and L. V. Gool, "Pathtrack: Fast trajectory annotation with path supervision," in *IEEE International Conference on Computer Vision*, pp. 290–299, 2017.

[80] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org ." https://www.openstreetmap.org, 2017.

[81] "Chicago Data Portal: Crimes - 2001 to present." https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g. Accessed: 2021-06-08.

[82] "OpenDataPhilly: Crime Incidents." https://www.opendataphilly.org/dataset/crime-incidents. Accessed: 2021-06-08.

[83] "Beijing maps 19 crime hot spots." http://www.ecns.cn/cns-wire/2013/07-12/72886.shtml. Accessed: 2021-06-08.

[84] D. Yang, D. Zhang, and B. Qu, "Participatory cultural mapping based on collective behavior data in location-based social networks," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 3, pp. 30:1–30:23, 2016.

[85] D. Yang, D. Zhang, L. Chen, and B. Qu, "Nationtelescope: Monitoring and visualizing large-scale collective behavior in lbsns," *Journal of Network and Computer Applications*, vol. 55, pp. 170–180, 2015.

[86] Y. Zheng, H. Fu, X. Xie, W.-Y. Ma, and Q. Li, *Geolife GPS trajectory dataset - User Guide*, 1.1 ed., July 2011.

[87] P. Mittal, N. Borisov, C. Troncoso, and A. Rial, "Scalable anonymous communication with provable security," in *5th USENIX Workshop on Hot Topics in Security, HotSec'10, Washington, D.C., USA, August 10, 2010* (W. Z. Venema, ed.), USENIX Association, 2010.

[88] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach, "AP3: cooperative, decentralized anonymous communication," in *Proceedings of the 11st ACM SIGOPS European Workshop, Leuven, Belgium, September 19-22, 2004* (Y. Berbers and M. Castro, eds.), p. 30, ACM, 2004.

[89] H. Peng, S. Lu, J. Li, A. Zhang, and D. Zhao, "An anonymity scheme based on pseudonym in P2P networks," in *Forensics in Telecommunications, Information, and Multimedia - Third International ICST Conference, e-Forensics 2010, Shanghai, China, November 11-12, 2010, Revised Selected Papers* (X. Lai, D. Gu, B. Jin, Y. Wang, and H. Li, eds.), vol. 56 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 287–293, Springer, 2010.

Generated using Postgraduate Thesis LaTeX Template, Version 1.03. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Wednesday 30th June, 2021 at 7:38am.