A Privacy-Enhanced and Personalized Safe Route Planner with Crowdsourced Data and Computation

Fariha Tabassum Islam, Tanzima Hashem, Rifat Shahriyar
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology Dhaka, Bangladesh
fariha.t13@gmail.com, tanzimahashem@cse.buet.ac.bd, rifat@cse.buet.ac.bd

***Abstract*—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. 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. We implement a working prototype of our solution on the Android platform. Extensive experiments using real datasets show that our approach finds the safest route in seconds with 50% less exposure of personalized safety scores.**

***Index Terms*—crowdsource, privacy, safe route, route planner**

I. 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. The road safety in such 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:

2375-026X/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00027

- 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 the computational power and storage in smartphones has enabled researchers to envision for 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 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 develop a heuristic that has less pSSs exposure than the direct algorithm and less communication frequency than the iterative algorithm. The heuristic algorithm sacrifices the answer's accuracy slightly.
- We run extensive experiments with real datasets and evaluate the effectiveness and efficiency of our approach. We implement a working prototype of our solution and show its applicability in the real environment.

II. Problem Formulation

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 I shows the notations used in this paper.

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:

Fig. 1: R_3 is the SR between s and d with $SS = -2$, $\delta = 18$

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 $-S \leq pSS \leq S$.

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 $\lfloor \frac{\Psi_1+\Psi_2+\ldots+\Psi_n}{n} \rfloor$.*

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 IX-C2).

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 into 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 safer than R, where R is any other route between s and d having $dist(R) \leq \delta$.*

TABLE I: Notations and their meaning

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

TABLE II: A comparative analysis with existing safe route planners

	Problem Settings				Privacy	Efficiency
	Safety Level	pSS	Objective	δ		
[4]	Multiple	×	Provide multiple routes with 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	×	Minimize the distance associated with the risky roads	✓	✓	✓

In Fig. 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.

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 our system, the unsafe event type (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.

III. Related Works

A. Safe Route Planners

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

Problem setting. None of the existing work considers minimizing the individual distance associated with risky roads. Thus, the problem settings of existing works are not suitable for safe travel on roads. Furthermore, instead of considering the total 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.

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 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 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.

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.

Other route planners. Variants of orienteering and scheduling problems [12], [13] 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 [14] 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 [13], 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 same length associated with every SS. However, IBM CPLEX is not effective in terms of time and memory when a problem requires to find multiple answers like multiple routes with the same largest minimum SS in the SR query [15].

B. Crowdsourcing

Crowdsourcing has been widely used for route recommendation [16], [17] and POI search [2], [3], package delivery [18] and indoor mapping [19]. In [2], the authors considered protecting 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.

IV. System overview

We develop a privacy-enhanced, personalized, and trustworthy solution for safe route planning with crowdsourced data and computation. Fig. 2 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

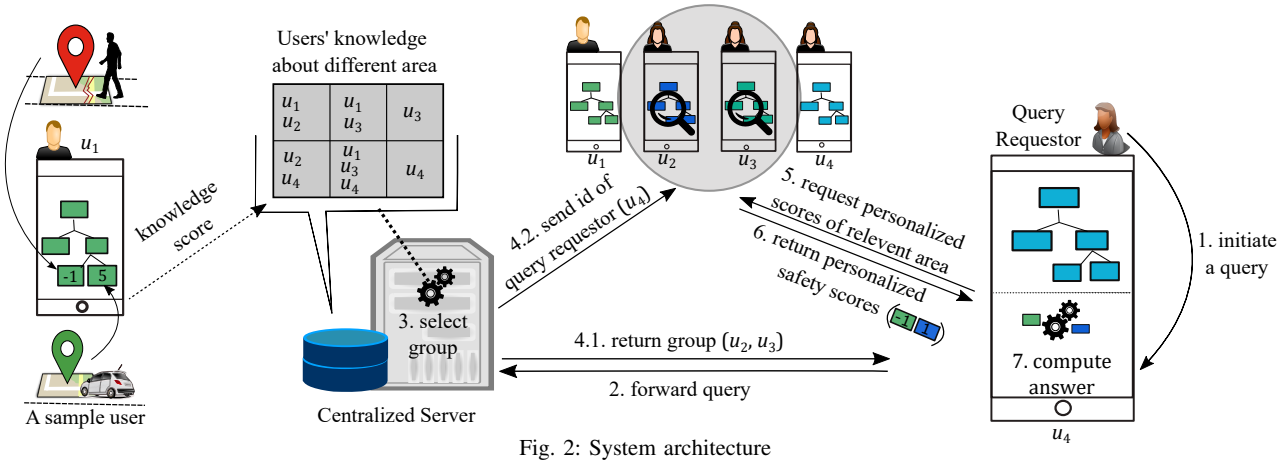


Fig. 2: System architecture

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.

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 query requestor (QR), it selects a group based on the query parameters and the stored KSs of the users. Then the CS returns temporary IDs of the group members to the QR and sends the identity of the QR to the group members. A user's temporary ID is changed periodically and thus a QR cannot identify the user who participates in multiple queries. 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 SS of each road using the pSSs of the group members, and finds the SR.

V. Quantification of Safety

A. 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.

B. 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. 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. Similarly, if a user safely visits an area, after some time, that visit's impact decays, and that is not perceived as safe as before.
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 historical visit data of a user.

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: $\xi' = \xi * e^{-\frac{dist^2}{2*h^2}}$, 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 in every Δ_d days. If the decay rate is r_d and $\Psi \neq 0$, then after every Δ_d days, Ψ becomes $\Psi = \Psi * r_d$, 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).

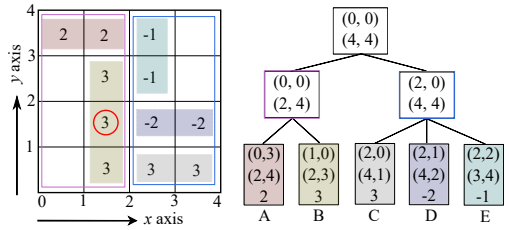
VI. Indexing User Knowledge

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.

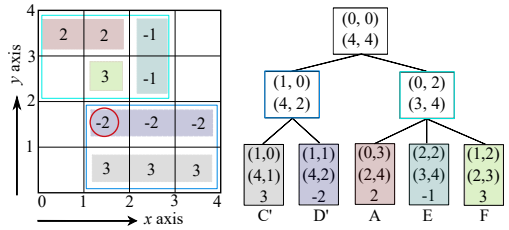
A. Local Indexing.

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 [20] 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 time 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



(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

Fig. 3: A user's pSSs is stored in a modified R-tree

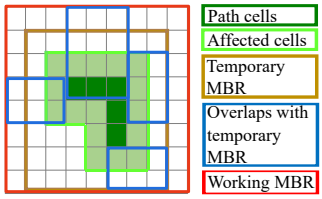


Fig. 4: Necessary MBRs for updating a supercell

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 stores the coordinates of the MBR. The MBR of the root node of the R-tree represents the total grid area. Fig. 3 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.

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. 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).
- *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). 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.
- *Compute working MBR.* Find the working MBR that includes thpse overlapping supercells and the affected cells (Fig. 4).
- *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. 3b shows the updated R-tree for the change of the pSS from 3 to -2 in a grid cell (shown with a red circle).

B. 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 key. For each grid cell, we store the user ids whose KS is 1 for the corresponding grid cell area.

VII. Query Evaluation

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: *Direct Optimal Algorithm (Dir_OA)* and *Iterative Optimal Algorithm (It_OA)*. Dir_OA aims at reducing the processing time, whereas It_OA increases the privacy in terms of the number of retrieved pSSs. Though a pSS does not reveal a user's travel experience (Section VIII) with certainty, the user's privacy is further enhanced by minimizing the number of shared pSSs with the QR. We develop a *Safety Score based Heuristic Algorithm (SS_HA)*. SS_HA has less exposure of pSSs than that of Dir_OA and less communication frequency than that of It_OA. SS_HA sacrifices the accuracy of the answer slightly.

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 to 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;

A. 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 the 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:

- 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.
- 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.

Algorithm 2: It_OA(s, d, δ, N)

```
N', A_q ← compute_query_area(s, d, δ, N);
G_q, M_q ← retrieve_query_group(A_q);
SS_q ← ∅, Q_p ← ∅, v ← s;
while v! = d do
    A_q' ← find_required_cells(v, N, A_q, SS_q);
    SS_q ← SS_q ⋃ compute_SS(G_q, M_q, A_q');
    SR ← get_safest_route(v, N, SS_q, Q_p);
    v ← get_last_vertex(SR);
end
return SR;
```

B. 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.

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 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. We decide the value of X_{it} in experiments.

Algorithm 3: SS_HA(s, d, δ, N, H)

```
N', A_q ← compute_query_area(s, d, δ, N);
G_q, M_q ← retrieve_query_group(A_q);
SS_q, M_q', A_q' ← process_SS(G_q, M_q, A_q, H);
SS_q ← update_SS(G_q, M_q', A_q', SS_q, H);
N'' ← refine_query_area(s, d, δ, N', SS_q);
SR ← compute_safest_route(s, d, δ, N'', SS_q);
return SR;
```

C. Safety Score based Heuristic Algorithm (SS_HA)

SS_HA considers SSs upto H ($-S < H < S$) for ranking the routes in terms of safety. Algorithm 3 shows the pseudocode for SS_HA. SS_HA works in the similar way as Dir_OA except the process of retrieving pSSs and computing the SSs using Functions process_SS and update_SS.

Function process_SS divides the range (H, S] into equal-sized bins and shares them with the group members. If the pSS of a group member's known grid cell in A_q is less than or equal to H, then the user shares it with the QR. Otherwise, the group member shares in which bin the known grid cell belongs to. After retrieving pSS related information from the group members, the function computes either exact or the lower bound of the SS for the grid cells and returns them as SS_q. The function also returns $A_q{}'$ and $M_q{}'$, where $A_q{}'$ represents the grid cells whose lower bounds of SSs are equal to H and $M_q{}'$ includes grid cell wise group members who have not shared exact pSSs for the corresponding grid cell in $A_q{}'$. Function update_SS retrieves actual pSSs from those group members and computes exact SSs for grid cells in $A_q{}'$.

At this stage, the QR knows all SSs that are less than or equal to H. Finally, Function compute_safest_route finds the

SR by considering the lower bound of SSs as the SSs of the grid cells, whose actual SSs are not known. Thus, SS_HA reveals less pSSs than that of Dir_OA with similar computation time and slightly increased communication frequency.

D. Complexity Analysis

The compute_safest_route function in Dir_OA and SS_HA algorithms 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. Thus, the runtime complexity will be $O(b^p)$. Since we utilize two pruning techniques due to which the average depth reduces, the complexity becomes $O(b^{p/r})$, where $r = r_1 * r_2$. The factors r_1 and r_2 represent the effect of our first and second pruning techniques, respectively. In It_OA, edges are expanded till X_{it} depth along with the first pruning in Function find_required_cells, whose runtime complexity is $O(b^{X_{it}/r_1})$. Thus, the runtime complexity of It_OA is $O(b^{\frac{p}{r}+\frac{X_{it}}{r_1}})$.

VIII. Privacy Analysis

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 V). 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 contribution 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 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 result 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 do the validation 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.

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.
- We assign temporary ids to the group members so that when a QR runs multiple queries, the QR cannot accumulate a group member's data from multiple queries.
- The user can choose not to share her the KSs for sensitive areas with the CS.

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.

IX. 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 Section III), we evaluate the performance of our query processing algorithms by varying a wide range of parameters. We compare our solution with the centralized model and show the impact of the missing data on the quality of the SRs. Finally, we show the applicability of our solution in real environments using our implemented prototype.

A. Datasets and Parameters

1) Datasets: We use datasets of three cities: Chicago (C), Philadelphia (P) and Beijing (B). To simulate the environment,

TABLE III: 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

TABLE IV: 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

for each dataset, we need the road network data, the crime data, and the users' visit data to different areas. We use OpenStreetMap [21] to download the road networks. We use the real crime data of Chicago[1]and Philadelphia[2]. For Beijing, instead of crime data, only the locations of crime hotspots[3] are available. We identify the crime hotspots of Chicago using k-means clustering [22] and create similar hotspots around the Beijing hotspots' locations. We generate daily crimes around the hotspots of Beijing following the distribution of Chicago.

For the users' visit data to different areas, we use the day-to-day Foursquare check-in dataset [23], [24] for Chicago and Philadelphia, and real trajectory data of users for Beijing [25]. 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, otherwise, their effects will not be visible due to decay. The details of these datasets are summarized in Table III. 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 of 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 provides 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.

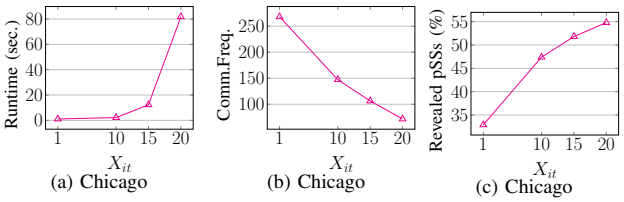
We normalize the crime count in the range [0,1] per grid cell for each day. This count represents the crime probability of each grid cell. For each grid cell, according to the crime probability, we randomly associate the crime events with the visits of the users. Thus, the probability of experiencing 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 as safe events. The pSSs are calculated based on the model of Section V. We choose the model parameters in a way that satisfies Lemma1 for every pSS.

2) Parameters: We show the parameters' default values and ranges in Table IV. 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 meter 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 δ 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 at most 1.5 as a user may not feel comfortable to travel longer than 1.5 times of the shortest distance. The parameter z is used for confidence level (Section IX-C2). 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.

[1]https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present-Dashboard/5cd6-ry5g

[2]https://www.opendataphilly.org/dataset/crime-incidents

[3]http://www.ecns.cn/cns-wire/2013/07-12/72886.shtml



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

Fig. 5: Choosing default value $X_{it} = 15$ based on the effects of X_{it}

B. Comparison of Query Evaluation Algorithms

We provide two optimal and one heuristic query processing algorithms, Dir_OA, It_OA and SS_HA, respectively. We compare the algorithms based on runtime, communication frequency per involved group member (*comm. freq.*), and the total number of revealed pSSs. The less 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 Fig. 6.

1) Choosing the default value of X_{it}: X_{it} significantly impacts the performance of It_OA. Fig. 5 shows clear trade-offs among performance metrics for Chicago. The runtime increases exponentially (undesirable), the comm. freq. 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 comm. freq. is low, and the runtime and the number of revealed pSSs are reasonable. We choose $X_{it} = 15$ as the default value for the Chicago dataset because we see a sharp increase in the runtime for $X_{it} > 15$. In the similar way, for Beijing and Philadelphia, we choose the default values $X_{it} = 20$ and $X_{it} = 10$, respectively (not shown).

2) Comparison of Dir_OA and It_OA: Fig. 6a- 6c shows that It_OA reveals around 47% of the revealed pSSs in Dir_OA. The number of revealed pSSs increases with an increase of δ and d_q because the route length increases. The number of revealed pSSs also increases for large d_G because the number 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.

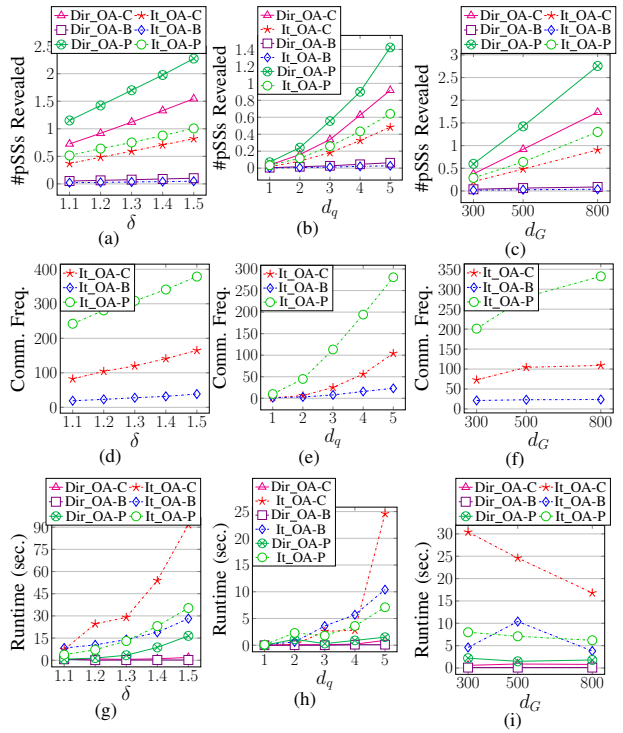


Fig. 6: Dir_OA vs. It_OA in terms of privacy (#pSSs revealed) and computation cost (comm.freq. and runtime) for varying δ, d_q and d_G

Fig. 6d- 6i compare Dir_OA and It_OA in terms of comm. freq. and runtime. The comm. freq. is always 1 for Dir_OA as the group-members are requested once to provide pSSs. For It_OA, the comm. freq. is as high as 378.8 times. To check how reasonable this is, we ran an experiment: a message is sent from one device to another using Firebase Cloud Messaging service and a reply from 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, 378.8 communications take 378.8 * 173.28 ms $\approx$ 1 minute, which is acceptable. The comm. freq. for It_OA increases with an increase of δ, d_q and d_G. For δ and d_q, the reason behind the increased comm. freq. is the increased route length, whereas for d_G, the reason is the number of required grid cells to compute the SR increases.

The runtime of Dir_OA is very low (on average 1 second) for all datasets. Though the runtime of It_OA increases with the increase in δ and d_q, they are reasonable (on average 14 seconds). For $\delta = 1.5$ in Chicago, it is 1.5 minutes, which is acceptable. Interestingly, the runtime does not increase with an increase of d_G. Therefore, we conclude that both Dir_OA and It_OA provide practical solution for the SR queries and show a trade-off between runtime and privacy.

TABLE V: Performance of SS_HA on three datasets

Dataset	Top-3 (%)	Revealed pSS (%)	Comm. Freq.	Runtime (sec.)
C	81.8	80.22	1.967	1.20
B	77.5	61.05	1.762	0.11
P	89.8	79.95	1.966	1.63

3) Performance of SS_HA: Since SS_HA does not provide the optimal answer, in addition to the three performance metrics, we measure the accuracy of the provided answer. The accuracy is the percentage of answered routes that is one of the top-3 SRs. We set $H = 1$ and divide $(1, 10]$ to $[2, 4]$, $[5, 7]$ and $[8, 10]$ bins and set other parameters to default values. The performance of SS_HA is shown in Table V. The accuracy is more than 77% for all datasets and at least 20% less pSSs is revealed compared to Dir_OA. The comm. freq. can be at most 2 for SS_HA since one group-member is queried for pSSs at most twice. The runtime of SS_HA is in par with Dir_OA. Therefore, *SS_HA can be a good alternative for quickly finding SRs with high accuracy and reasonable privacy.*

C. 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.

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. Fig. 7 shows that the average accuracy increases with an increase in user data (29% for C50 and 48.6% for C90). Even 10% missing data causes significant (51.4%) 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 (Fig. 7c, 7f, 7i).

*2) Confidence Level (**CL**):* 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 CL 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 CL.

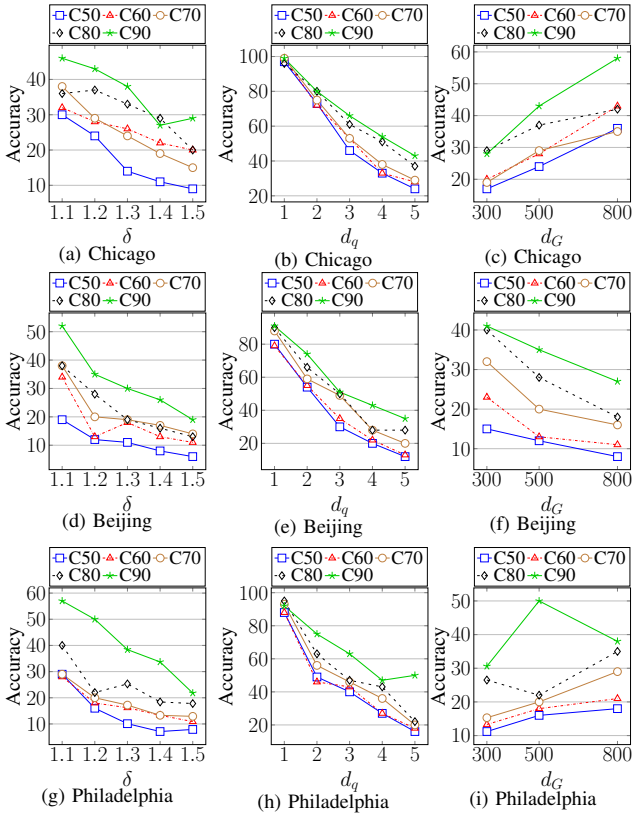


Fig. 7: Accuracy loss in the centralized model for missing data. C50 means 50% of actual data is present.

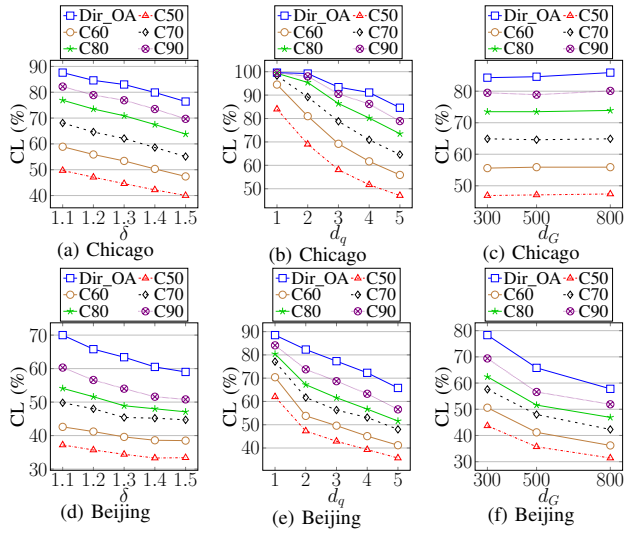


Fig. 8: CL for our system is higher than that of the centralized model.

Fig. 8 shows that the CL for our system is always the highest (on average 86.6%). Since both Dir_OA and It_OA provide optimal solutions, their CL is the same. In the centralized model, CL predictably increases with the increase of missing data. CL 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 (not shown).

D. Effect of Different Datasets

The datasets in our experiments provide variation in the number of users and road network structure and size. The runtime (Fig. 6) increases for Chicago (3554 users) due to more users compared to Beijing (87 users) even though the runtime is still reasonable. Both comm. freq. and the number of revealed pSSs are the highest for Philadelphia (Fig. 6), which does not have the highest number of users or road network size (Table III). Therefore, these two metrics might have been affected by the road network structure. The CL is less for Beijing than other datasets (Fig. 8), which shows its dependency on the number of users.

E. Prototype

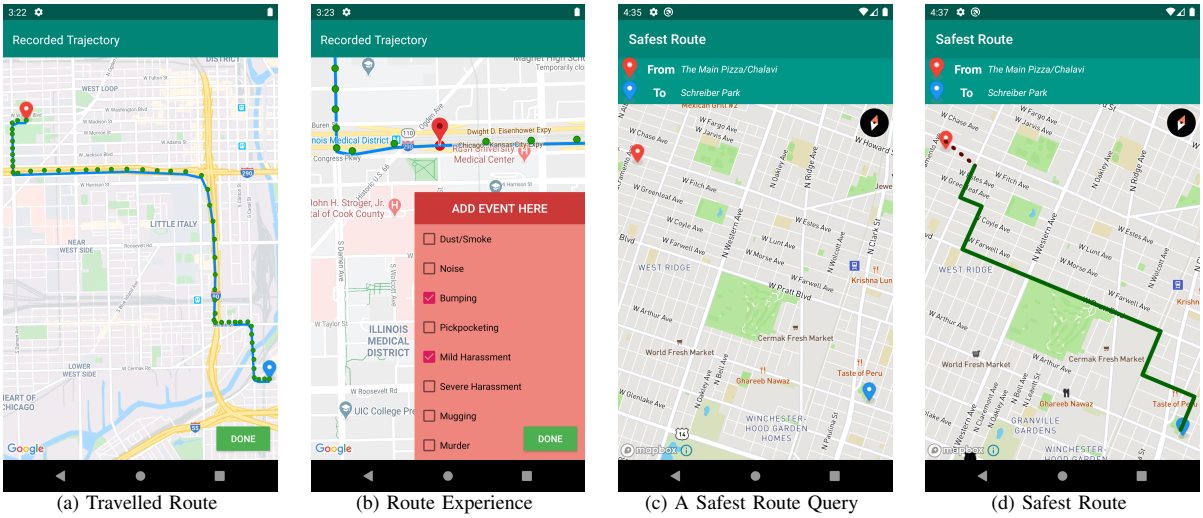
We implemented a prototype of our system. When a user starts travelling via a route, we temporally store the route trajectory as shown in Fig. 9a until the user shares her experiences on different points of the trajectory, or press the done button to indicate that the travel was safe (Fig. 9b). The user initiates an SR query in Fig. 9c and it is answered in Fig. 9d. We test Dir_OA and It_OA with this prototype in a pedestrian scenario for 5 users. Their experiences were assigned from top-5 users' 10 days data of the Chicago dataset. We set $X_{it} = 5$ and run 30 queries. We observe that Dir_OA takes 2.03 seconds and It_OA takes 3.29 seconds per query on average, which is very reasonable for the SR query. The communication frequency is 5.11 for the iterative approach on average and 1 for Dir_OA in all queries. Finally, It_OA reveals only 31.69% of pSSs revealed by Dir_OA, and thus enhances privacy within reasonable query processing time.

X. Extension

Our algorithms can be adopted for a variant of the SR query that takes γ, the minimum required SS for the route as input and returns the shortest route that satisfies γ. An SR query guarantees that the minimum SS of the returned SR is better than that of any other route within δ. If a user wants to maximize her safety on roads within δ, the user would prefer the SR query, whereas a user who wants to fix the minimum required SS would prefer the SR query variant. However, it would be hard to decide γ without having the idea of the safety condition of an area as for the inappropriate choice of γ the variant of the SR query may not find any route that meets γ. Our algorithms for the SR query can be applied by finding the shortest route in the graph with the edges whose minimum SS is greater than or equal to the required minimum SS of the user. Since the variant of the SR query is orthogonal to the main focus of this paper, we leave the detail study of this variant of the SR query as our future work.

XI. Conclusion

We developed a novel journey planner for finding SRs with crowdsourced data and computation. In experiments, we observe that 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 36% and 41% times when



(a) Travelled Route (b) Route Experience (c) A Safest Route Query (d) Safest Route

Fig. 9: Safe Route Prototype

a centralized route planner has 30% and 20% missing data, respectively. Our privacy-enhanced solution encourages more users to share their data and improve the quality of the SRs.

Experiments show that our approach can evaluate a query in seconds. Our iterative query processing algorithm enhances user privacy by not revealing, on average, 53% 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. The communication frequency and the number of revealed pSSs in the heuristic algorithm are in between those of the direct and iterative algorithms.

Acknowledgments This research has been done in Bangladesh University of Engineering and Technology (BUET). Fariha Tabassum Islam is supported by the ICT Division, Bangladesh (56.00.0000.028.33.108.18).

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*, 2014, pp. 2695–2704.

[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," *IMWUT*, 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," *PMC*, 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 *WWW*, 2014, pp. 179–182.

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

[7] K. Fu, Y. Lu, and C. Lu, "TREADS: a safe route recommender using social media mining and text summarization," in *ACM SIGSPATIAL GIS*, 2014, pp. 557–560.

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

[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 *UbiComp*, 2013, pp. 559–562.

[10] S. Aljubayrin, J. Qi, C. S. Jensen, R. Zhang, Z. He, and Z. Wen, "The safest path via safe zones," in *ICDE*, 2015, pp. 531–542.

[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 *DCOSS*, 2018, pp. 77–84.

[12] A. Allahverdi, C. T. Ng, T. C. E. Cheng, and M. Y. Kovalyov, "A survey of scheduling problems with setup times or costs," *Eur. J. Oper. Res.*, vol. 187, no. 3, pp. 985–1032, 2008.

[13] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *Eur. J. Oper. Res.*, vol. 209, no. 1, pp. 1–10, 2011.

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

[15] *How to enumerate all solutions*, Accessed: 2020-09-11. [Online]. Available: https://ibm.co/2Fjq482

[16] T. Hultman, A. Boudjadar, and M. Asplund, "Connectivity-optimal shortest paths using crowdsourced data," in *PerCom*, 2016, pp. 1–6.

[17] H. Su, K. Zheng, J. Huang, H. Jeung, L. Chen, and X. Zhou, "Crowdplanner: A crowd-based route recommendation system," in *ICDE*, 2014, pp. 1144–1155.

[18] 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 Trans. Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1478–1496, 2017.

[19] 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," *IMWUT*, vol. 2, no. 1, pp. 1:1–1:26, 2018.

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

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

[22] J. Agarwal, R. Nagpal, and R. Sehgal, "Crime analysis using k-means clustering," *International Journal of Computer Applications*, vol. 83, no. 4, 2013.

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

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

[25] Y. Zheng, H. Fu, X. Xie, W.-Y. Ma, and Q. Li, *Geolife GPS trajectory dataset - User Guide*, 1st ed., July 2011.