World Wide Web Journal manuscript No.
(will be inserted by the editor)

Trip Planning Queries with Location Privacy in Spatial Databases

Subarna Chowdhury Soma[1] · **Tanzima Hashem**[2] · **Muhammad Aamir Cheema**[3] · **Samiha Samrose**[4]

Received: date / Accepted: date

Abstract Privacy has become a major concern for the users of location-based services (LBSs) and researchers have focused on protecting user privacy for different location-based queries. In this paper, we propose techniques to protect location privacy of users for trip planning (TP) queries, a novel type of query in spatial databases. A TP query enables a user to plan a trip with the minimum travel distance, where the trip starts from a source location, goes through a sequence of points of interest (POIs) (e.g., restaurant, shopping center), and ends at a destination location. Due to privacy concerns, users may not wish to disclose their exact locations to the location-based service provider (LSP). In this paper, we present the first comprehensive solution for processing TP queries without disclosing a user's actual source and destination locations to the LSP. Our system protects the user's privacy by sending either a false location or a cloaked location of the user to the LSP but provides exact results of the TP queries. We develop a novel technique to refine the search space as an elliptical region using geometric properties, which is the key idea behind the efficiency of our algorithms. To further reduce the processing overhead while computing a trip from a large POI database, we present an approximation algorithm for privacy preserving TP queries. Extensive experiments show that the proposed algorithms evaluate TP queries in real time with the desired level of location privacy.

Keywords Location-based services · privacy · trip planning queries

1 Introduction

Due to the exponential increase in the usage of smartphones and other GPS enabled devices, cheap wireless network bandwidth and the availability of huge amount of data related to the locations, location-based services (LBSs) have become immensely popular. Despite the usefulness of LBSs, privacy concerns are serious and cannot be ignored [27]. To access an LBS (e.g., asking for a nearest point of interest (POI) such as a restaurant or a gas station), a user provides her location to a location-based service provider (LSP) and the LSP returns the query answer based on the location. The LSP may infer private information about a user's

[1,2,4]Bangladesh University of Engineering and Technology, Bangladesh
· [3]Monash University, Australia
E-mail: {[1]subarna089, [2]tanzimahashem, [4]samiha.mumu}@gmail.com, [3]aamir.cheema@monash.edu

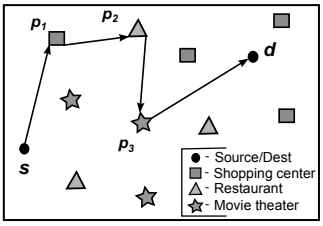


Fig. 1 Trip Planning

health, habits and preferences from the user's revealed location. For example, if a user is located at a heart clinic while requesting an LBS then the LSP may predict that the user is suffering from a heart disease. Similarly, the LSP may also predict user's home and work address and sell it to marketing companies.

An important and widely used type of LBS is to provide the users with information of their nearby POIs using nearest neighbor queries and their variants. In recent years, researchers have developed privacy preserving techniques for nearest neighbor queries. In this paper, we focus on protecting location privacy of users for a novel variant of nearest neighbor queries, *trip planning (TP) queries* [4,25,36]. For example, a user may plan a trip to go to a shopping center from her workplace, then have a dinner at a restaurant and finally watch a movie in a movie theater before returning home. A TP query enables the users to plan such a trip with the minimum trip distance.

Specifically, in a TP query, a user sends to LSP her source and destination locations, and the required types of POIs she wants to visit (e.g., a restaurant, a movie theater, a shopping centre). The LSP returns the locations of POIs that minimize the user's trip distance, where the trip starts from the user's source location, passes through at least one POI of each type, and ends at the user's destination. For example, in Figure 1, POIs p_1, p_2, and p_3 provide a user with the smallest trip distance.

A TP query is called a sequenced TP query if the sequence of visiting the types of POIs is fixed, e.g., first a shopping center, then a restaurant, and a movie theater at the end. Although we focus on sequenced TP queries in this paper, our solution can be easily applied to general TP queries when the sequence of visiting POI types is not fixed.

Due to the privacy concerns, users may not want to disclose their source and destination locations. In this paper, we assume that the users do not worry about disclosing the types of POIs they are planning to visit. This is because, unlike source and destination locations, a user may or may not visit a specific POI even if returned by the LSP. Moreover, usually the users do not mind revealing the types of POIs they are interested in (e.g., restaurants, shopping centers etc.). To protect a user's location privacy, revealing a user's false or cloaked location instead of the exact one has been shown to be an effective technique [6,13,28,38] in the literature. We are the first to propose efficient privacy preserving algorithms to evaluate TP queries with respect to both false and cloaked locations of users. Next, we briefly highlight our techniques for both of the cases.

1.1 Protecting privacy by sending a false location

Our first approach evaluates TP queries based on a user's false location, i.e., the user provides a false location instead of her actual source and destination to the LSP. The key idea of our approach is to incrementally retrieve the nearest POIs of different types with respect

to the false location from the LSP until the POIs that minimize the trip distance with respect to the user's actual source and destination locations have been identified. According to the elliptical property, the trip distance from a user's source to destination through any POI outside an ellipse is larger than the length of the major axis of the ellipse if the foci of the ellipse are at the user's source and destination, respectively. We set the length of the major axis of this ellipse as the minimum trip distance computed from the retrieved POIs from the LSP, and thus, the elliptical region represents the required POI search space as any POI outside the ellipse cannot further minimize the computed minimum trip distance. With the incremental retrieval of POIs, the length of the major axis, i.e., the minimum trip distance, remains constant or reduces, which in turn shrinks the area of the ellipse. The search for the optimal answer terminates when all POIs inside the ellipse are retrieved.

Though a user does not reveal her actual locations, the LSP can infer the user's location from other revealed information such as retrieved POIs and the technique used by the user to determine the optimal answer for actual locations. We quantify the level of location privacy of a user in terms of the obfuscation level [13], the area in the total space that is refined as the user's location. Our approach allows a user to specify her required privacy level and guarantees that the achieved obfuscation level is greater than or equal to the specified one.

1.2 Protecting privacy by sending cloaked locations

Our second approach is based on cloaked location where a user sends a cloaked location (e.g., an MBR) that contains her actual location. Specifically, a user sends her source and destination regions containing her actual source and destination locations, respectively. The LSP returns a candidate POI set to the user that includes optimal answers for all possible source-destination pairs within the revealed regions. The user knows her actual source and destination locations and finds the required answer from the candidate answer set. In this approach, the level of privacy, i.e., the obfuscation level, is measured in terms of the area of the cloaked location with respect to the total space.

Evaluating optimal answer for every possible source destination pairs within the source and destination regions independently would be prohibitively expensive. We refine the POI search space with respect to a user's source and destination regions by integrating the elliptical property with the concept of triangular inequalities. We develop an efficient algorithm to evaluate the candidate POI set that includes optimal answers for all possible source-destination pairs with a single search on the database. The key advantage of our solution based on the cloaked location is that the user needs a single communication with the LSP to have the answer. On the other hand, for our solution based on false location, the user needs multiple communications with the LSP.

1.3 Approximate results

Computing the exact answer for TP queries is computationally expensive and is shown to be NP-hard [4, 25]. Although the optimal solutions can still be computed if the number of POIs and types of POIs are not large, the cost may still be quite high especially if the privacy is also to be preserved and the sequence of visiting POI types is not fixed. Therefore, we develop approximation algorithms that might be preferable when a trip is computed from a large POI database or the required privacy level is high. We allow the user to input her desired accuracy guarantee and the proposed techniques optimize the query processing while

satisfying the desired accuracy guarantee. For example, if a user's required accuracy level is 90%, our approximation algorithm ensures that the optimal trip distance is not less than 90% of the returned trip distance. Our experimental study shows that our approximation algorithms actually return answers with better accuracy than the guaranteed accuracy. Our proposed approximation algorithms can compute trips for both false and cloaked locations.

1.4 Contributions

To the best of our knowledge, we are the first to address the privacy issues for TP queries. Unlike expensive cryptographic techniques, our approach does not guarantee strong location privacy [31] to the users with a modified system architecture. Our approach aims to develop a practical solution that trades privacy with query processing cost. *An important benefit of our solutions is that they do not require any modification in the existing framework and indexing method that an LSP uses for processing traditional incremental nearest neighbor queries [18].* Similar to existing solutions [29,38] in the literature, in our privacy preserving approaches, the query processing overhead is shared between the LSP and the user who requests the TP query. With the continuous advancement of mobile devices and communication media, we envision that the processing power of mobile devices and the communication bandwidth would not cause any bottleneck to deploy our approach in reality.

In summary, our contributions are as follows:

- We develop novel techniques to refine the search space for privacy preserving TP queries by exploiting elliptical properties and triangular inequalities.
- We develop efficient algorithms to evaluate optimal answer for TP queries with respect to both false and cloaked locations of users.
- We present approximation algorithms to compute trips from a large POI database with reduced processing overhead in a privacy preserving manner.
- We conduct extensive experiments to show that our approaches can evaluate TP queries in real time even for a high level of location privacy. We present a comparative analysis of our proposed approaches in terms of both computational and communication overhead.

The remainder of the paper is organized as follows. In Section 2, we formulate the research problem. Section 3 provides an overview of the related work. In Sections 4 and 5, we present privacy preserving algorithms for computing exact results of TP queries for false and cloaked locations, respectively. In Section 6, we present efficient privacy preserving algorithms for computing approximate results of TP queries satisfying specified accuracy guarantees. Section 7 presents experimental results and a comparative analysis of our approaches. Section 8 concludes the paper and presents future research directions.

2 Problem Formulation

Let D represent a set of POIs in a 2 dimensional space, p_i represent the location of a POI of type i in D, and $dist(.,.)$ denote the Euclidean distance between two points. The computation of the trip distance $Tdist(s,d,P)$ for a set of m types (e.g., restaurant, movie theater) of POIs $P = \{p_1, p_2, \ldots, p_m\} \in D$ with respect to a source s and a destination d, differs based on whether the sequence of visiting POI types has been specified by the user or not. For $m = 3$, if the user specified sequence is p_1, p_3, p_2, the trip distance $Tdist(s,d,P)$ is computed as

$dist(s,p_1)+dist(p_1,p_3)+dist(p_3,p_2)+dist(p_2,d)$. On the other hand, if the sequence of visiting POI types is not fixed, the trip distance is computed for every possible sequence of POI types and the minimum of these computed distances is considered as the trip distance $Tdist(s,d,P)$ for P.

A trip planning (TP) query is formally defined as follows:

Definition 1. Given a source s and a destination d, and m types of POIs, a TP query returns a set of POIs $P = \{p_1, p_2, \ldots, p_m\}$ from D, one POI from each type, that minimizes the trip distance, i.e, $Tdist(s,d,P) \leq Tdist(s,d,P')$ for any other set of POIs of m types $P' \in D$.

A user may be interested in k sets of POIs $\{p_1^1, p_2^1, \ldots, p_m^1\}$,$\{p_1^2, p_2^2, \ldots, p_m^2\}$, $\ldots$, $\{p_1^k, p_2^k, \ldots, p_m^k\}$ that have the k smallest trip distances for the trip. The user then selects one set by considering other priorities such as cost and recommendations. If a user queries for k sets of POIs then the query is called a *k trip planning (kTP) queries*. We remark that the source and destination may refer to the same location, e.g., a user may want to return to the source location after the trip.

For *privacy preserving kTP (PkTP) queries*, the kTP query is evaluated without disclosing a user's actual source and destination locations to the location-based service provider (LSP). We assume that users are connected to the LSP via mobile networks or the Internet for communication purpose and the locations and types of POIs are indexed using an R-tree [10] on the LSP's database.

The symbols used in this paper are summarized in Table 1.

Table 1 Notations and their meanings

Symbol	Meaning
s and d	Source and destination locations of a user
m	The number of required POI type
k	The number of required sets of POIs
p_i	The location of a POI of type i
$dist$	The Euclidean distance between two points
$Tdist$	The trip distance
f	A false location
s_r and d_r	Source and destination rectangles of a user

2.1 Privacy Model

In this paper, we consider the scenario where the users consider locations as sensitive data but are not concerned about revealing other information such as the types of POIs or their identities. Note that, in many real world applications, it is not always possible to hide identities from the LSP due to authentication issues [39] and personalized services [16].

2.1.1 Adversaries

We consider the LSP as the adversary. We do not consider an eavesdropper as a separate adversary because the maximum knowledge about a user's location that an eavesdropper can have is no more than that of the LSP. We assume that the adversary does not have any background knowledge about a user's location (e.g., distribution). For our approach based on a false location, a user sends a false location instead of her actual source and destination

locations to the LSP and retrieves nearest POIs with respect to the false location. Thus, the LSP only knows the false location and the returned POIs to a user. On the other hand, for our approach based on cloaked location, a user sends source and destination rectangles to the LSP and retrieves a candidate answer set. Thus, in this case the LSP knows the source and destination rectangles and the returned candidate answer set to a user. Our approaches guarantee a user's required privacy level and provide the user with accurate answers for TP queries.

2.1.2 Privacy Level

We quantify the level of location privacy in terms of *obfuscation level* (o_l). Although the LSP cannot determine the exact location of a user, it can infer that the exact location of a user is contained in a region R, e.g., in the cloaked region. Assuming that the area of the whole data space is 1 and y is the area of the region R. The obfuscation level is $y\%$. Please note that larger the obfuscation level the better it is from the user's perspective. For the case when the user reports cloaked location, the obfuscation level can be easily determined using the area of the cloaked region. For the case when the reported location is a false location, the details on how to compute the obfuscation level are given in Section 4.5.

3 Related Work

In Section 3.1, we discuss existing work on trip planning queries and in Section 3.2, we present privacy preserving techniques for other type of queries.

3.1 Trip Planning Queries

A large body of research works focus on developing algorithms for processing route and trip planning queries [4,30,36]. In [36], Sharifzadeh et al. have developed algorithms to evaluate an optimal sequenced route (OSR) query that returns a route with the minimum length passing through a set of POIs in a particular order from the source location of a user, where both order and type of POIs are specified by the user. Chen et al. [4] have proposed a generalization of the trip planning query, called multi-rule partial sequenced route (MRPSR) query. A MRPSR query provides a uniform framework to evaluate both OSR query [36] and trip planning query [25]. In [30], the authors propose a method that plans a trip by searching the shortest route from the current position with stops at one of each specified POI type from the visiting sequence before reaching the final destination. In [2,11,12,32], the authors have developed trip planning algorithms for groups.

A keyword-aware optimal route (KOR) query [3] returns an optimal route that starts from a user's source, passes through the locations that match with the user's specified keywords such as a restaurant, a shopping mall, and ends at the target destination of the user within a budget constraint. For example, a user may want to maximize the popularity of a route while satisfying a budget constraint (e.g., duration of the trip). In [3], Cao et al. have developed approximation algorithms to evaluate KOR queries. In [26], Li et al. have extended KOR queries by integrating preferences for the specified keywords (e.g., a user may have the highest preference for the shopping mall).

On the other hand, the proliferation of GPS-enabled mobile devices enables users to share trajectory data and thus, facilitates trajectory matching queries in trajectory databases.

In [33,34], the authors have proposed trajectory matching approaches, where a user can specify her preferences for different locations of a trajectory and the trajectory matching query returns the set of trajectories from the datbase that are similar to the user's given trajectory and preferences. These types of personalized trajectory matching can be used for route recommendations to the users.

However, none of the above mentioned variants of route and TP queries consider a user's location privacy, i.e., the LSP is aware of user's exact source and destination. These existing trip planning techniques can not provide optimal answer without knowing user's actual source-destination and thus are not suitable for our privacy preserving kTP queries.

Shortest path and distance queries and route planning for two fixed locations have also been extensively studied in the literature [35,37,40,41]. Shortest path and distance queries [40,41] return the shortest path between two points in a road network and the length of the shortest path, respectively. In [35], Shang et al. have proposed an approach to plan a route from a source to a destination by avoiding the potential traffic congestions. In [37], Wang et al. have proposed algorithms to identify routes between two stations in a public transport network based on different criteria such as earliest arrival time, latest departure path and shortest duration path. These approaches are different from trip planning queries as they do not consider the selection of POIs of required types while planning a route and the route between two fixed locations does not pass through the POIs.

3.2 Location Privacy

In the literature, there exist a number of privacy models that include K-anonymity [20,28], obfuscation [6,38], space-transformation [21,22] and cryptography [8].

In a K-anonymity technique, a user provides a region that includes the locations of $K-1$ users in addition to the user's location and the user becomes k-anonymous. In the k-anonymity technique, though a user is not identified even if an adversary knows the locations of all users included in the rectangle. However, it incurs high processing overhead and requires involving other parties to compute the user's K-anonymous location. In an obfuscation technique, a user reveals her false or cloaked location to the LSP and does not need any third party to compute her cloaked or false location. Thus, K-anonymity and obfuscation techniques apply for two different scenarios. In this paper, we assume that adversaries do not have any knowledge about a user's location and the user reveals her identity while requesting TP queries, and thus we adopt obfuscation model to protect the user's location privacy.

With the space transformation strategy, the query is evaluated in a transformed space and a key drawback of this technique is that the query results may not always be accurate. On the other hand, as shown in [7] cryptographic techniques provide strong location privacy in return of high query processing overhead for nearest neighbor queries. Since we aim for a lightweight solution for trip planning queries, a complex variant of nearest neighbor queries, we adopt the obfuscation technique as our privacy model.

Researchers have developed privacy preserving techniques for variant spatial queries [15,16,19,28]. Some of these privacy preserving algorithms [28] provide exact answers and others [23] provide approximate answers for requested queries. We develop the first privacy preserving algorithm for TP queries that provide users with the exact answer for both cloaked and false locations.

Efficient approaches [5,9,24,38] have been developed for nearest neighbor queries and group nearest neighbor queries for both false and cloaked locations. However, these solu-

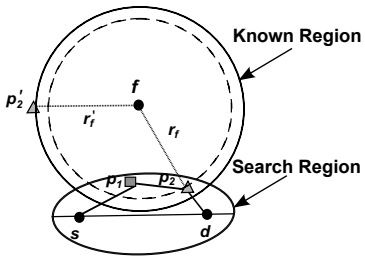


Fig. 2 Known Region and Search Region

tions do not apply for TP queries, as nearest and group nearest neighbor queries involve finding the nearest POI of a specific type, whereas TP queries identify POIs of different types that minimize the trip distance with respect to source and destination locations of a user. On the other hand, nearest neighbor and group nearest neighbor queries evaluate the answer with respect to a single location, i.e., current location of the user.

4 Our Solution : False Location

In this section, we present a solution to evaluate kTP queries based on a user's false location revealed to the LSP. The key idea of our algorithm is to incrementally retrieve the nearest POIs with respect to a false location f from the LSP until the POIs that minimize the trip distances have been identified. We exploit elliptical properties to determine the optimal trip with respect to s and d from the retrieved POIs.

The user computes the *known region* [38] using the retrieved POIs from the LSP. The known region is the area, where locations of all POIs of required types are known. Then the user identifies the POIs (one from each required type) within the known region that minimize the k^{th} smallest trip distance with respect to s and d. Based on the current minimum trip distance and the source-destination pair, the user determines the *search region* [38], the area where the user needs to know locations of all POIs of required types. The search region is an ellipse and we prove that the user's trip through a POI outside the ellipse cannot provide the minimum trip distance. More specifically, the elliptical search region contains the POIs that minimize the user's trip distance with respect to s and d among all POIs on the LSP's database.

With the incremental retrieval of POIs, the known region expands and the search region shrinks or remains same. The search for optimal answer terminates when the known region includes the search region.

In Sections 4.1, and 4.2, we discuss techniques to compute known region and search region, respectively. In Section 4.3, we present the terminating condition of the search for POIs. In Sections 4.5 and 4.6, we present the technique to compute privacy level of users and the privacy analysis for our approach based on a user's false location, respectively. Appendix A presents the detailed algorithms, pseudocodes, and descriptions for processing PkTP queries with respect to a false location.

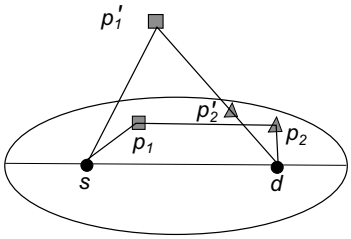


Fig. 3 Search region that includes all POIs of required types

4.1 Computing Known Region

The known region is a circle with center at f and radius r_f equal to the distance between f and the farthest retrieved POI from f. With the incremental retrieval of nearest POIs with respect to f, r_f gradually increases and the known region expands.

Figure 2 shows an example, where $m = 2$ and $k = 1$. In the first iteration, the user retrieves 1st and 2nd nearest POIs p_1 and p_2 with respect to f from the LSP. The dotted circle represents the known region since the locations of POIs of type 1 and 2 are known to the user. There is no other POI of type 1 and 2 within this area because if there exists such a POI, the POI's distance from f would be less than that of p_2 and the LSP must have returned that POI before p_2. In the next iteration, the known region (shown with solid line) expands by receiving the 3rd nearest POI $p_1^{'}$ from the LSP.

4.2 Computing Search Region

Using the retrieved POIs from the LSP, the user computes the search region using elliptical properties. The elliptical property states that the distance from one focus of the ellipse to the other through any point outside the ellipse is larger than the length of the major axis of the ellipse. We adopt this property to compute the search region, which is elaborated in the following lemma:

Lemma 1 *Let s and d represent foci of an ellipse with the major axis equal to the k^{th} smallest trip distance computed from the retrieved POIs in the known region. The ellipse is the search region that includes all POIs of required types that provide k smallest trip distances with respect to s and d.*

Proof (By contradiction) Let p_1' be a POI located outside the ellipse and minimizes the user's k^{th} smallest trip distance for source destination pair s and d. Assume that POIs of other required types are located in the Euclidean path between p_1' and d (see Figure 3) so that they do not add extra distances.

According to the elliptical property, the trip distance from s to d through p_1' is larger than the length of the major axis, which is the upper bound of the k^{th} smallest trip distance. Thus, p_1' cannot further minimize the trip distance, which contradicts the assumption. The ellipse is the search region that includes all POIs of required types that provide k smallest trip distances with respect to s and d. □

Therefore, the search region is an ellipse with two foci at source s and destination d of the user. The major axis of the search region ellipse is the upper bound of the k^{th} smallest

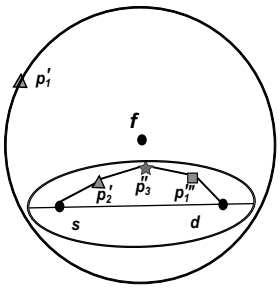


Fig. 4 The known region covers the search region

trip distance, which is computed based on the POIs of the required types retrieved from the LSP. The length of the major axis gradually reduces or remains same with the retrieval of more POIs from the LSP.

Figure 2 also shows a search region for $k = 1$ and $m = 2$. The upper bound of the k^{th} smallest trip distance is computed as $Tdist(s,d,\{p_1,p_2\})$. Thus, the search region is an ellipse with two foci at s and d and the major axis equal to $Tdist(s,d,\{p_1,p_2\})$.

4.3 Terminating Condition

With the incremental retrieval of nearest POIs from f, the known region gradually expands and when the known region includes the search region, all POIs within the search region have been retrieved. The incremental retrieval of nearest POIs continues until the known region includes the search region because at this stage, the POIs of required types that produce the optimal answer with respect to s and d, have been identified (see Figure 4).

4.4 An Example

In this section, we explain the steps of our approach with an example. Figure 5 shows a scenario for $k = 1$, $m = 2$ and the sequence of visiting POI type is first 2 then 1. First time, the LSP returns 3 POIs p_1, p_2, p_2' of two types 1 and 2 with respect to a user's false location f. The known region is the circle with radius r_f and center f as shown in Figure 5(a), where r_f is set to the distance between f and p_1 because p_1 is the farthest POI from f. The search region is computed by selecting the POIs p_2, p_1 that minimize the total travel distance. The major axis d_{min} of the search region, i.e, ellipse, is the total distance returned by $Tdist(s,d,\{p_2,p_1\})$ and the foci are the user's actual source and destination s and d. Note that at this stage of the evaluation the search region is not included in the known region. The TP query answer is initialized as $\{p_2,p_1\}$.

In the next phase, in Figure 5(b), the next nearest POI p_1' from f is retrieved from the LSP and the known region expands. We observe that, POI p_1' further minimizes the trip distance because $Tdist(s,d,\{p_2,p_1'\}) < Tdist(s,d,\{p_2,p_1\})$. Thus the length of the major axis d_{min} is updated as $Tdist(s,d,\{p_2,p_1'\})$ and the search region shrinks. The TP query answer is updated as $\{p_2,p_1'\}$.

Finally, the user retrieves the next nearest POI p_2'' (see Figure 5(c)) from f and the known region expands. Since the POI p_2'' does not minimize the trip distance for any combination

of POIs, the search region remains same. Dotted regions represent the previous positions of the known and search regions. As the expanded known region encloses the search region, the terminating condition satisfies and we get p_2, p_1' as the final POI set that minimizes the trip distance from s to d.

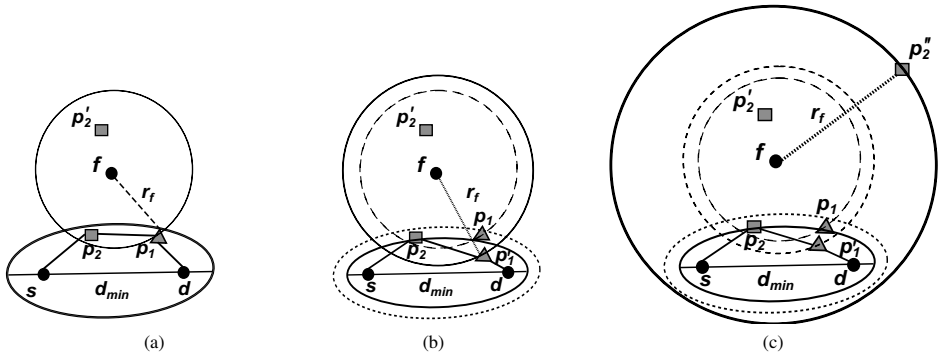


Fig. 5 An example of PkTP query evaluation process for $k = 1$.

4.5 Computing the Obfuscation Level o_l

Though the user reveals a false location instead of her actual source and destination locations to the LSP, the LSP can refine the user's actual source destination pair within a set of source destination pair from other revealed information. During an access of a kTP query, the LSP can have the following information: f, k, POIs retrieved by the user, and the termination condition of the search.

The LSP can compute the known region in the similar way to the user from the set of POIs retrieved by the user. However, the LSP cannot compute the search region as it does not know the actual source destination pair of the user. Since the LSP knows the terminating condition, i.e., the search terminates when the known region covers the search region, the LSP may take a source destination pair s' and d' randomly within the known region and compute the search region, where the foci of the search region are s' and d' and the major axis is equal to the k^{th} smallest trip distance from s' to d' via the POIs of required types in the known region. Then the LSP checks whether the search region is included in the known region. If yes, then the LSP considers the selected source destination pair as a candidate for the actual source destination pair. The LSP repeats the process for all possible source destination pair within the known region and refines the user's actual source destination pair in a set of candidate source destination pairs.

We express the level of location privacy in terms of the obfuscation level, where the obfuscation level (o_l) is the percentage of the area covering the refined set of source destination pairs with respect to the total space. The uncertainty increases with the increase of the obfuscation level. The larger the known region, the higher the percentage of location pairs that satisfy the terminating condition. Thus, to increase the level of location privacy, a user may continue the search, even if the current known region includes the user's required search region. More specifically, when the known region covers the user's search region, the user can determine her obfuscation level achieved at that stage of the PkTP query evaluation

process. If the achieved obfuscation level is less than the required one, the user can continue to request additional POIs to the LSP for an arbitrary number of times until the obfuscation level reaches a satisfactory value.

4.6 Privacy Analysis

The following lemma shows that our approach based on a user's false location guarantees a user's required level of obfuscation (o_l) and the LSP cannot reduce the user's privacy level by reverse engineering the proposed privacy preserving technique.

Lemma 2 *Let o_l represent a user's required privacy level, f the user's false location, and s-d the user's actual source destination pair. Then our approach based on a user's false location satisfies the user's required obfuscation level o_l.*

Proof The LSP knows a user's false location f, the distribution of returned POIs to the user, and the technique to find the optimal answer of a TP query. From these revealed information, the LSP can compute the refined set *SD* of source destination pairs that include the user's actual source destination pair, when our approach based on a user's false location terminates. A source destination pair (s', d') is included in *SD*, if an ellipse with foci at s' and d', respectively, is located within the known region, where the major axis of the ellipse equals to the smallest trip distance from s' to d' via the POIs of required types among the retrieved set of POIs from the LSP. Our approach based on a user's false location continues until the cardinality of *SD* becomes large enough to satisfy the user's required obfuscation level o_l.

Further, our technique to compute a false location f ensures that the LSP cannot prune any source destination pair from *SD*. The false location is randomly selected on the boundary of an ellipse, whose foci are located at the user's actual source destination pair and the length of the major axis is randomly set from a range of the distance between s and d to the maximum distance between two points in the total space. Since the length of the major axis can vary, for any source destination pair (s', d') in *SD*, the LSP can find an ellipse with foci at s' and d' such that f resides on the boundary of that ellipse. Thus, the LSP cannot exclude any source destination pair s' and d' from *SD* by reverse engineering the computation technique of f.

In summary, our approach based on a user's false location computes obfuscation level with respect to *SD*, terminates when the obfuscation level is equal to or higher than the user's required obfuscation level. Since the LSP cannot further refine *SD* using f, Algorithm 1 guarantees the user's required obfuscation level o_l. □

5 Our Solution: Cloaked Location

In this section, we present a solution to evaluate kTP queries based on a user's cloaked locations. In our solution based on the cloaked location of a user, we determine the obfuscation level as the percentage of the source/destination rectangle area with respect to the total space. Thus, to request a kTP query, a user computes two rectangles s_r and d_r that include the user's source location s and destination location d, respectively, and satisfy the user's required obfuscation level. The user sends s_r, d_r, and the required types to the LSP. The LSP

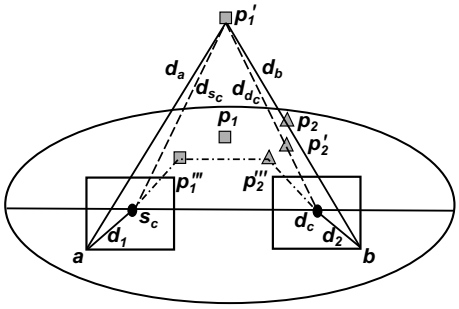


Fig. 6 Lemma 3

returns candidate POIs that provide k smallest trip distances for all possible source destination pairs within s_r and d_r, respectively. The user finds the POIs from the candidate answer set that minimize trip distances for her actual source and destination s and d.

The major challenge to evaluate the candidate answer set is to find k optimal trips for all possible source destination pairs within s_r and d_r, respectively. Evaluating the POIs for each source destination pair independently would be prohibitively expensive. In this paper, we develop an approach to evaluate the candidate answer set that includes optimal answers for all source destination pairs with a single search on the database. Based on elliptical properties and triangular inequalities, we gradually refine the search space, which plays a key role to reduce the processing overhead significantly.

Similar to our approach for a user's false location, the LSP (instead of the user) updates the known region and the search region with the incremental retrieval of POIs from the database. The candidate answer set is identified when the known region covers the search region.

The LSP incrementally finds nearest POIs with respect to the midpoint m_p of the centers s_c and d_c of rectangles s_r and d_r, respectively. The known region is centered at m_p and the radius is the distance between m_p and the furthest POI from m_p. With the incremental retrieval of POIs, the LSP updates the radius of the known region and the k^{th} smallest travel distance from s_c to d_c via POIs of required types inside the known region. The required search region is an ellipse, which is defined by the following lemma:

Lemma 3 *Let s_c and d_c represent foci of an ellipse with the major axis equal to $d+2\times(d_1+d_2)$, where d is the upper bound of the k^{th} smallest trip distance with respect to the source destination pair s_c and d_c, and d_1 (d_2) is the distance between s_c (d_c) and the corner point of the rectangle s_r (d_r). The ellipse is the search region that includes all POIs of required types that provide k smallest trip distances with respect to all possible source destination pairs in s_r and d_r, respectively.*

Proof (By contradiction) Let p_1' be a POI located outside the ellipse and minimizes the k^{th} smallest trip distance with respect to a source destination pair a and b, where a and b are corners of s_r and d_r, respectively. Assume that POIs of other required types are located in the Euclidean path between p_1' and d_c (please see Figure 6) so that they do not add extra distances.

According to the elliptical property, the trip distance from s_c to d_c through p_1', $(d_{s_c}+d_{d_c})$ is larger than the length of the major axis, which is $d+2\times(d_1+d_2)$.

$$(d_{s_c}+d_{d_c})>d+2\times(d_1+d_2) \tag{1}$$

Let d_a and d_b represent the distances from a and b to p'_1, respectively. According to triangular inequalities, $(d_a + d_1) > d_{s_c}$ and $(d_b + d_2) > d_{d_c}$. Thus from Equation 1, we have

$(d_a + d_1 + d_b + d_2) > d + 2 \times (d_1 + d_2)$

$\triangleright (d_a + d_b) > d + d_1 + d_2$

Given that d is the upper bound of the k^{th} smallest trip distance with respect to the source destination pair s_c and d_c. Thus according to the triangular inequality, the upper bound of the k^{th} smallest trip distance with respect to source destination pair a and b is $d + d_1 + d_2$.

We already have $(d_a + d_b) > d + d_1 + d_2$. Thus, p'_1 cannot further minimize the trip distance for source destination pair a and b, which contradicts the assumption.

Further a and b are the farthest points within s_r and d_r from s_c and d_c, respectively. Thus for any other source destination pair s' and d' within s_r and d_r, the distances between s' and s_c, and d' and d_c are less then d_1 and d_2, respectively.

The elliptical search region with foci s_c and d_c, and the major axis equal to $d + 2 \times (d_1 + d_2)$ include all POIs of required types that provide k smallest trip distances with respect to all possible source destination pairs in s_r and d_r, respectively. □

In the next section, we present the privacy analysis for our approach based on a user's cloaked locations. Appendix B presents the detailed algorithms, pseudocodes, and descriptions for processing PkTP queries with respect to a cloaked location.

5.1 Privacy Analysis

The following lemma shows that our approach based on a user's cloaked locations guarantees a user's required level of obfuscation (o_l) and the LSP cannot reduce the user's privacy level by reverse engineering the proposed privacy preserving technique.

Lemma 4 *Let o_l represent a user's required privacy level, s_r and d_r the user's source and destination rectangles, s-d the user's actual source destination pair. Our approach based on a user's cloaked locations satisfies the user's required obfuscation level o_l.*

Proof A user computes her source and destination rectangles s_r and d_r and sends them to the LSP. The LSP returns a candidate answer set that includes POIs for TP query answer for every possible source-destination pair within s_r and d_r, respectively. The area of s_r (or d_r) represents o_l% area with respect to the total space. Since the LSP does not know about which POIs are selected by the user, the LSP cannot refine the user's actual source and destination pair *s-d* within s_r and d_r, respectively. Even if the candidate answer set includes a single set of POIs of required types, that POI set is the TP query answer for all possible source-destination pairs within s_r and d_r and thus, the LSP cannot refine the user's actual source and destination pair *s-d* within s_r and d_r, respectively. Thus, our approach based on a user's cloaked locations guarantees the user's required obfuscation level o_l. □

6 Approximation Solutions

As discussed earlier, computing the exact answer for TP queries may be computationally expensive especially when the desired level of privacy is high (e.g., large cloaked region) and the sequence of visiting POI types is not fixed. A user may be willing to compromise the accuracy of the returned results in order to improve the query processing cost, e.g., a user may be happy to obtain a trip that is similar to the optimal trip but is retrieved much quickly.

Motivated by this, in this section, we extend our solutions proposed in the previous sections to enable a user to specify the required level of accuracy for the answer and in return reduce the query processing overhead significantly.

Let x be the desired accuracy level. We propose approximate algorithms that guarantee that the optimal trip distance is not smaller than $x\%$ of the distance of the trip returned by the proposed approximate algorithms. For example, if a user's required accuracy level is 90, our approximation algorithm guarantees that if the trip distance for the approximated answer is 10 km, the trip distance for the optimal answer is not less than 9 km.

Our approximation algorithm to process a PkTP query with respect to a user's false location works in a similar way to the optimal solution proposed in Section 4 except that the major axis of the search region (i.e., ellipse) is set as the $x\%$ of the current best k^{th} smallest trip distance. The trip distance via any POI outside the search region is greater than $x\%$ of the current best k^{th} smallest trip distance. The following lemma shows the correctness of our approximation algorithm.

Lemma 5 *Let s, d and x represent the source, destination and accuracy level specified by a user. The optimal trip distance is not less than x% of k smallest trip distances returned by our approximation algorithm to process a kTP query with respect to a user's false location.*

Proof The approximation algorithm incrementally retrieves POIs from the LSP until the known region includes the search region. The search region is an ellipse with foci at s and d and the major axis equal to $x\%$ of the current k^{th} smallest trip distance computed from the already retrieved POIs. For every retrieved POI, the algorithm checks whether it is in the search region and minimizes the current k^{th} smallest trip distance. If yes, the length of the major axis of the search region is updated.

A POI outside the ellipse cannot provide a trip distance less than the length of the major axis of the ellipse. Let p'_1 be a POI located outside the ellipse and POIs of other required types are located in the Euclidean path between p'_1 and d so that they do not add extra distances. According to the elliptical property, the distance of a trip from s to d via p'_1 is greater than the length of the major axis of the ellipse, i.e., $x\%$ of the k^{th} smallest trip distance computed from the retrieved POIs. Thus, the optimal trip distance is not less than $x\%$ of k smallest trip distances returned by our approximation algorithm to process a kTP query with respect to a user's false location. □

The approximation solution for a user's cloaked location works in a similar way to the optimal solution proposed in Section 5 except that the LSP returns POIs that provide trip distances with a guaranteed accuracy level x with respect to all possible source-destination pairs in s_r and d_r, respectively. To ensure the approximation of the answer with a $x\%$ accuracy level instead of the optimal answer, d is changed to $x\%$ of the current best k^{th} smallest trip distance with respect to the source destination pair s_c and d_c, and the length of the major axis is set as $d + 2 \times (d_1 + d_2)$. Due to similarity with Lemma 5, we omit the correctness proof of the approximation algorithm for processing a PkTP query with respect to a user's cloaked location.

7 Experiments

In this section, we evaluate the performance of our proposed algorithms using false and cloaked locations through an extensive set of experiments. In our experiments, we use both real and synthetic data sets. The real dataset [1] consists of 100K POIs of 63 different types

from California. We generate two synthetic POI data sets U and Z using uniform and Zipfian distribution, respectively. We vary the size of datasets U and Z as 15000, 30,000, 60,000, and 120,000 POI locations. For all data sets, the data space is normalized into a span of 10,000 x 10,000 square units. The location of POIs of all types are indexed using a single *R**-trees. Though using an independent *R**-tree for every type may reduce query processing time, it is not realistic to deploy such a large number of *R**-trees; for example, in case of California dataset, it would require 63 independent *R**-trees for 63 POI types. Further, using separate *R**-tree for every POI type would increase the IO cost significantly.

Since there is no existing algorithm to process trip planning queries with location privacy, in our experiments, we perform a comparative analysis of our proposed algorithms for false and cloaked locations in terms of IO costs, query processing time, and communication overhead. In all experiments, we measure the query processing time for the LSP and the user independently to measure the efficiency of our algorithms. Our solution based on cloaked locations requires a single communication between a user and the LSP, whereas our solution based on a false location requires several communications between a user and the LSP. We also measure the answer set size returned to the user as the answer set size can be used to approximate the communication overhead irrespective of the bandwidth of the network used for communicating with the LSP.

Table 2 Experimental Setup

Parameter	Range	Default
Distance between s and d (in %)	2, 4, 8, 16, 32, 64	8
Type m	1, 2, 3, 4, 5	3
k	2, 4, 8, 16	4
Obfuscation level (o_l)	0.004%, 0.006%, 0.008%, 0.01%	0.01%
Data set size (Synthetic)	15K, 30K, 60K, 120K	30K

We vary different parameters: the distance between source s to destination d, the number of required type (m), the number of required sets of POIs (k), obfuscation level (o_l) and the dataset size in different sets of experiments. Table 2 summarizes the range of values used for each parameter in our experiments and their default values. We set 0.01% of the total data space as the default area for the obfuscation level as it represents around 20 km^2 with respect to the total area of California, which is even larger than the area of a small suburb in California. We also choose default values for other parameters considering practical scenarios. We run the experiments on a desktop with a Intel Core 2 Duo 2.40 GHz CPU and 4 GBytes RAM.

We vary the distance between source and destination as 2%, 4%, 8%, 16%, 32%, and 64% of the maximum distance between two pints in the total space. For every source-destination distance, we generate 100 sample TP queries by computing the source, destination, and false locations randomly in the total space and take the average of their performances in terms of IO costs, query processing time, and communication overhead. Specifically, we first randomly generate the source point inside the total space and then we generate the destination point randomly in the total space at the required distance from the source location. Since the focus of this paper is on sequenced kTP queries, we also consider sequenced kTP queries for our experiments.

In Sections 7.1, 7.2, 7.3, 7.4, and 7.5, we present our experimental results for our optimal algorithms for varying distance between s to d, m, k, o_l and dataset size, respectively. In Section 7.6, we perform a comparative analysis between our proposed algorithms for false

and cloaked locations based on the experimental results. In Section 7.7, we present our experimental results for our approximation algorithms for varying the accuracy level.

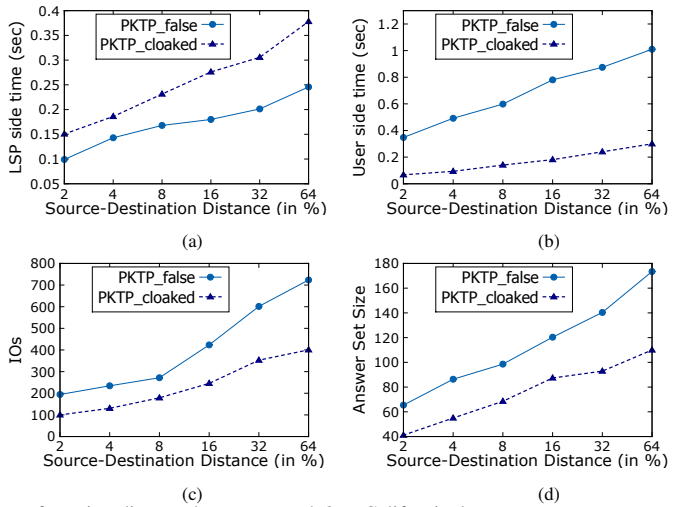


Fig. 7 Effect of varying distance between s and d on California dataset

7.1 Effect of Distance between s and d

The experimental results of varying distance between s and d have been shown in the Figure 7. In Figure 7(a), the LSP side processing time for our approach based on a user's cloaked location, PkTP_cloaked is higher than that of our approach based on a user's false location, PkTP_false. For the user side time, IOs, and answer set size, our approach based on cloaked locations outperforms the privacy protection technique based on a false location (see Figures 7(b)-(d)).

This is because for PkTP_cloaked, the LSP side algorithm performs most of the complex and time consuming tasks. The trends in graphs show that the LSP side processing time, user side processing time, IOs and answer set size increase with the increase of the distance between source-destination.

7.2 Effect of m

Figures 8(a), (b) and (c) show the effect of varying m from 1 to 5 on the LSP and user side processing time and IO cost, respectively, for California dataset. The experimental results show that the privacy protection technique based on cloaked location PkTP_cloaked, outperforms the privacy protection technique based on false location, PkTP_false in terms of IO cost and user processing time. We also measure the communication cost and answer set size for varying m. From Figure 8(d), we see that the answer set size increases with the increase of m. On average PkTP_cloaked requires 1.6 times more LSP side processing time whereas PkTP_false requires 2.2 times more user side processing time. IO cost and answer set size are on average 1.4 times higher for PkTP_false than those of PkTP_cloaked.

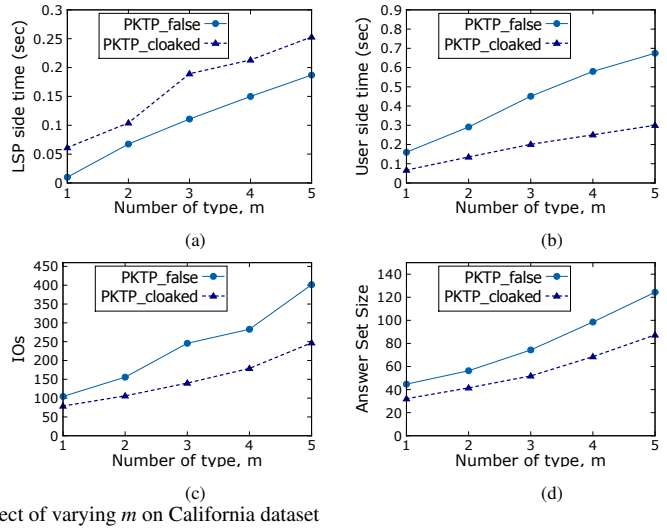


Fig. 8 Effect of varying m on California dataset

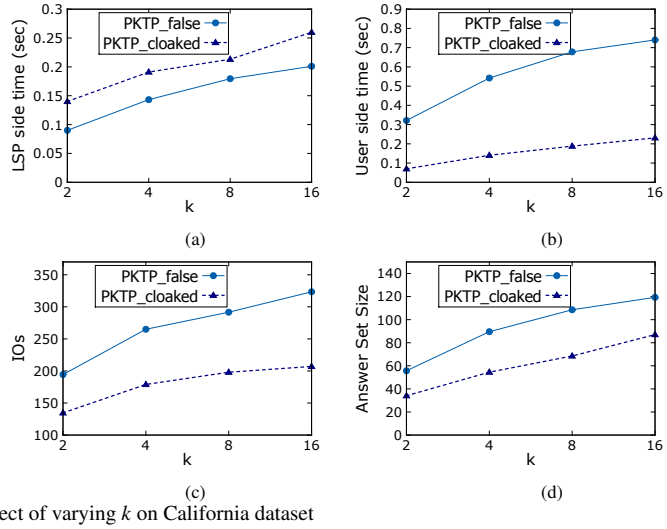


Fig. 9 Effect of varying k on California dataset

7.3 Effect of k

In this set of experiments, we vary the number of required sets of POIs (k) to observe its effect on LSP and user side processing time, IO costs, and answer set size. Both user side processing cost and IO cost increase with k (see Figure 9(b)-(c)). The answer set size also shows the same behavior in Figure 9(d). Our privacy protection technique based on cloaked location PkTP_cloaked outperforms our privacy protection technique based on false location PkTP_false in terms of all parameters except the LSP side processing time (Figure 9(a)) for the same reason explained above in Section 7.1. The rate of change in the LSP side processing time, IO cost and answer set size is almost same for both PkTP_false and PkTP_cloaked. On the other hand, the rate of change in the user side processing time is high for PkTP_false.

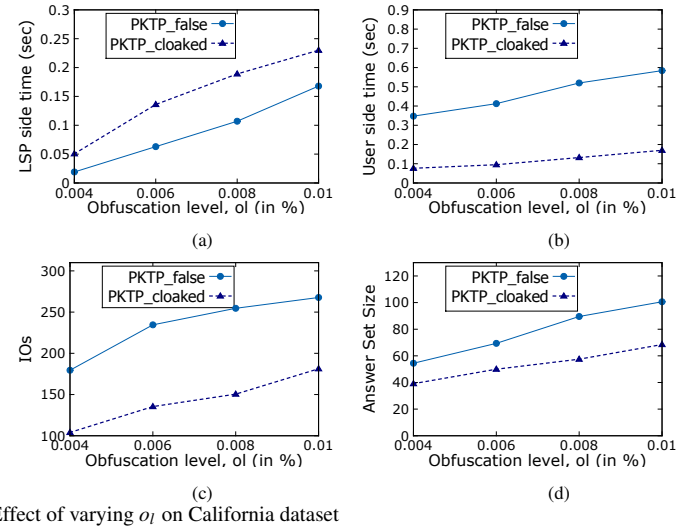


Fig. 10 Effect of varying o_l on California dataset

7.4 Effect of Obfuscation Level (o_l)

In this set of experiments, we vary the obfuscation level from 0.004% to 0.01%. For PkTP_false, a user approximates her obfuscation level (o_l) with monte carlo simulation. The process of retrieving POIs from the LSP continues until a user's approximated obfuscation level meets the required obfuscation level.

For PkTP_cloaked, we generate a cloaked region which is a rectangle. To compute the rectangles according to the privacy requirements, we use the algorithm proposed in [13, 14].

A larger obfuscation level o_l ensures a higher level of privacy which also comes with higher processing time and IO costs (see Figures 10(a), (b), (c)). The answer set size also increases with the increase of the obfuscation level for both solutions based on cloaked and false locations (Figure 10(d)).

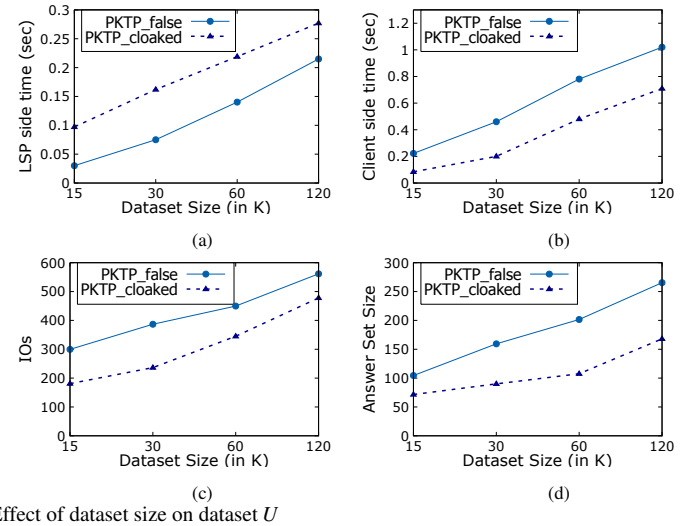


Fig. 11 Effect of dataset size on dataset U

7.5 Effect of Data Set Size

In this set of experiments, we vary the dataset size for both uniform (U) and Zipfian (Z) distributions. Figures 11(a)-(d) show the LSP and user processing time, IO cost and answer set size respectively, for different data set sizes with U distribution and Figures 12(a)-(d) show the experimental results for Z distribution. The experimental results show that the privacy protection technique based on cloaked location PkTP_cloaked, outperforms the privacy protection technique based on false location, PkTP_false in terms of IO cost, LSP and user side processing time. Specially the user side processing time is much higher for PkTP_false due to the obfuscation level computation overhead in the user side. We observe that the experimental results for Z distribution follow similar trends to U distribution.

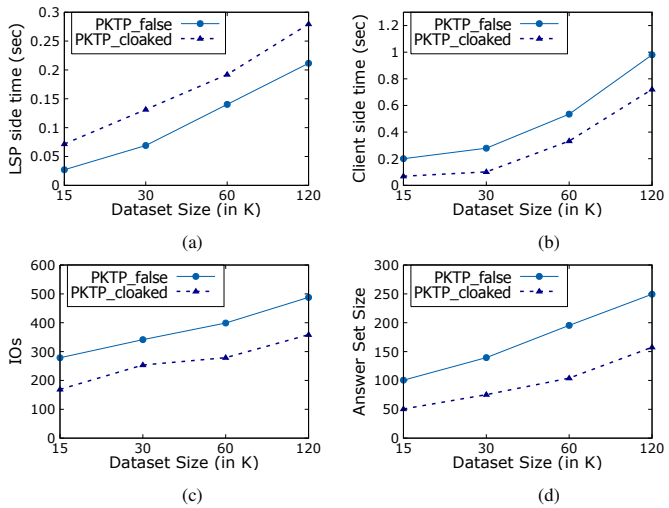


Fig. 12 Effect of dataset size on dataset Z

7.6 Comparative Analysis

From our experimental results we observe that the privacy preserving technique based on the cloaked location performs better than the privacy preserving technique for the false location in terms of IOs, user side processing time and communication overhead. On the other hand, the privacy preserving technique for the false location incurs less LSP side processing time than that of the privacy preserving technique based on the cloaked location. Though the processing overhead in terms of computational and communication overhead of both PkTP_false and PkTP_cloaked increase with the increase of different parameters such as m, k, o_l, the rate of increase of the processing overhead for PkTP_cloaked is less than PkTP_cloaked. Thus, PkTP_cloaked is more scalable than PkTP_false.

The LSP side processing time for PkTP_cloaked is on average 1.6 times more than that of PkTP_false but the user side processing time is on average 3.02 times less than that of PkTP_false. This is because, PkTP_false only retrieves POIs of required types from the database of the LSP and performs other computations on the user's mobile device. On the other hand, in addition to retrieving POIs from the database, PkTP_cloaked performs the

checking whether the required POIs to compute k optimal trips for all possible source destination pairs within source and destination regions have been retrieved. On the other hand, the reason behind such a high processing time on the user side for PkTP_false is in addition to computing optimal answer with respect to the actual source destination pair, the algorithm also checks whether the user's required obfuscation level is satisfied using monte carlo simulation, which consumes most of the user side processing time.

The experimental results show that PkTP_cloaked always outperforms PkTP_false in terms of both IOs and answer set size for any parameter. On an average PkTP_cloaked needs 1.6 times less IOs for finding the answer set. Similarly, the answer set size is on average 1.5 times larger for PkTP_false than that of PkTP_cloaked. PkTP_false needs to retrieve more POIs than PkTP_cloaked even for similar obfuscation level because of the techniques used to compute obfuscation levels in these two algorithms.

Further, for PkTP_cloaked, the user communicates once with the LSP, whereas for PkTP_false, the user needs to communicate several times with the LSP.

In summary, we conclude that a user with high performance mobile device may go for the privacy preserving technique using a false location. Otherwise the privacy preserving technique based on the cloaked location is in general a good option due to its less communication and computational overhead.

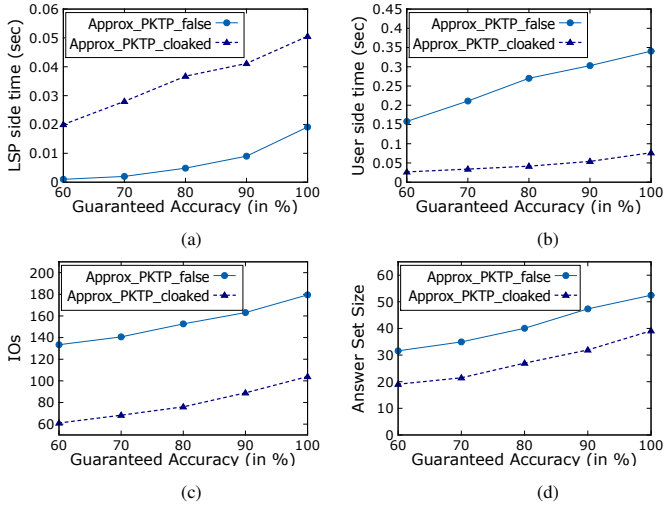


Fig. 13 Effect of varying the guaranteed accuracy level on California dataset when the sequence of visiting POI types is fixed

7.7 Approximation Algorithm

In this set of experiments, to observe the effect of reducing the accuracy level on the query performance, we set the obfuscation level to the minimum value, i.e., 0.004%, of the used range because to ensure a higher obfuscation level, our privacy preserving technique based on the false location often requires to retrieve the same number of POIs from the LSP irrespective of the required accuracy level. The other parameters are set into default values according to Table 2. Since the benefit of using an approximation algorithm is more pronounced when the sequence of visiting POI types is not fixed, in this set of experiments,

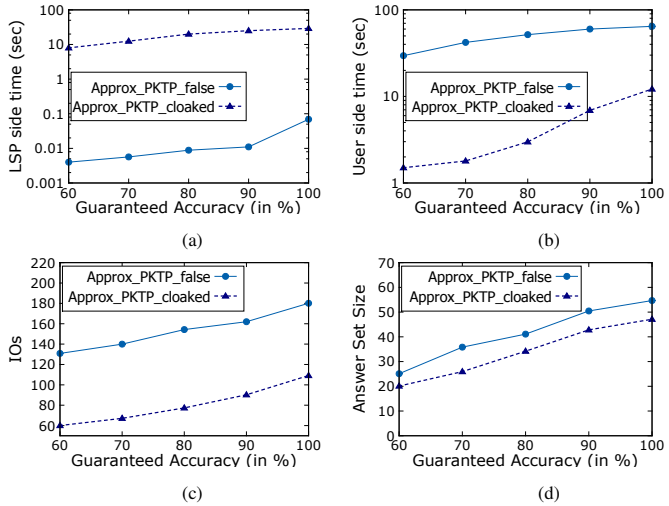


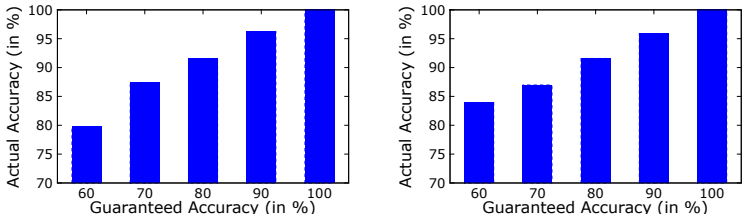
Fig. 14 Effect of varying the guaranteed accuracy level on California dataset when the sequence of visiting POI types is not fixed

we consider both scenarios, i.e., by keeping the sequence of visiting POI types fixed and flexible.

Figures 13(a)–(d) show the effect of varying the guaranteed accuracy level on LSP and user side processing time, IO cost and answer set size, respectively for California dataset when the sequence of visiting POI types remains fixed. The performance of both privacy preserving techniques improve with the decrease of the guaranteed accuracy level. For example, for sacrificing 10% accuracy, the LSP side processing time decreases 53% and 19% for our techniques based on false and cloaked locations, respectively.

Figures 14(a)–(d) show the experimental results for the scenario when the sequence of visiting POI types is not fixed. In Figures 14(a) and 14(b), we observe that the LSP side processing time for our approach based on cloaked locations and the user side processing time for both of our approaches based on false and cloaked locations are higher than those shown in Figures 13(a) and 13(b), respectively. This is expected because when the sequence of visiting POI types is not fixed, our approach requires to consider all possible sequences of POI types while computing the trip distance. However, in Figures 14(a) and 14(b), we find that both LSP and user side processing time decrease significantly in return of sacrificing the accuracy of query answers.

The actual accuracy of the returned query answer to the user is greater than or equal to the guaranteed accuracy level achieved using our approximation algorithms. For example, for a guaranteed accuracy level 80%, if the trip distance for the returned approximated answer is 100 meter, the trip distance for the optimal answer can be greater than or equal to 80 meter. In our experiments, we observe that the actual accuracy is higher than the guaranteed accuracy most of the times. Figures 15 shows the average actual accuracy obtained for varying the guaranteed accuracy level. For example, for 80% guaranteed accuracy level, the actual accuracy achieved, on average, is 92%.



(a) (b)

Fig. 15 Guaranteed Accuracy vs. Actual Accuracy when the sequence of visiting POI types is (a) fixed and (b) not fixed

8 Conclusion

In this paper, we have proposed the first comprehensive solution for privacy preserving k trip planning (PkTP) queries. We have developed both optimal and approximation algorithms to evaluate kTP queries with respect to false and cloaked locations of users. We have exploited geometric properties to refine the search space and reduce the query processing overhead, which is a major challenge for a privacy preserving queries as a user does not reveal her location to the LSP. Experiments show that our approach can provide users the optimal answer with a reduced communication and computational overhead for a high level of location privacy and the processing overhead can be further reduced with the proposed approximation algorithms in return of sacrificing the accuracy of the query answer slightly. Our comparative analysis through experiments reveals that the privacy protection technique based on a user's cloaked location is scalable and outperforms the privacy protection technique based on a user's false location in terms of IOs, communication and user side computational overhead.

Though we have shown experiments in the Euclidean space, our approach can be also adopted in road networks. In future, we have plan to protect user privacy for kTP queries with other privacy models such as K-anonymity and cryptographic approach.

Appendix A Detailed Algorithm: False Location

In this section, we present the detailed algorithm to evaluate a PkTP query based on false location. Algorithm 1, PkTP_false, shows the pseudocode to process a PKTP query that runs on a user's mobile device. Inputs of the algorithm are k, required types $\{1,2,\ldots,m\}$, a user's actual source s and destination d, and the obfuscation level o_l. The output is $R_s = \{p_1^1, p_2^1, \ldots, p_m^1\}, \{p_1^2, p_2^2, \ldots, p_m^2\}, \ldots, \{p_1^k, p_2^k, \ldots, p_m^k\}$, k sets of POIs that have the k smallest trip distances from s to d.

The notations that we have used for this algorithm are as follows:

- $MinD[1..k]$: An array of k entries, where $MinD[j]$ represents the j^{th} smallest trip distance from s to d via visiting POIs of required types, where $1 \leq j \leq k$. $MinD[j]$ is updated with the incremental retrieval of POIs from the LSP. In addition, $MinD[k]$ represents the length of the major axis of the search region (ellipse).
- r_f: The distance between f and the farthest retrieved POI from f. It also represents the radius of the known region (circle) centering at f.
- IsInsideKnownRegion(s,d,$MinD[k]$,f,r_f): A function that checks whether the known region(circle) covers the search region(ellipse) or not and returns yes or no, respectively.
- $P^{'}$: A set that stores all POIs retrieved from the LSP, which fall inside the search region.

The first step of our algorithm is the computation of the false location f based on the source s and destination d of a user. Any irreversible technique to compute the false location can be used for this purpose. In our implementation, we have used the following technique to compute the false location. At first an ellipsoid area is calculated with two foci at s and d respectively. The length of the major axis is randomly selected from a range that can vary from the distance between s and d to the maximum distance between two points

Algorithm 1 PkTP_false ($input : k, \{1,2,\ldots,m\}, s, d, o_l$)

1: $f \leftarrow Compute(s,d)$
2: $R_s \leftarrow \{\emptyset\}$
3: $MinD[1...k] \leftarrow \{\infty\}$
4: $r_f \leftarrow 0$
5: $P^{'} \leftarrow \emptyset$
6: $flag \leftarrow 0$
7: **repeat**
8: $P \leftarrow INN(f,k,m,flag)$
9: update (r_f)
10: $P \leftarrow IsInsideSearchRegion(s,d,MinD[k],P)$
11: $S \leftarrow$ GenerateSet$(P,P^{'},m,s,d,MinD[k],flag)$
12: **for** each set $S^{'} \in S$ **do**
13: **if** $Tdist(s,d,S^{'}) < MinD[k]$ **then**
14: update $(R_s, MinD)$
15: **end if**
16: **end for**
17: $P^{'} \leftarrow IsInsideSearchRegion(s,d,MinD[k],P^{'})$
18: $flag \leftarrow 1$
19: **until** $IsInsideKnownRegion(s,d,MinD[k],f,r_f)$=0
20: **while** $o_l < CompObLev()$ **do**
21: $D \leftarrow INN(f,k,m,flag)$
22: **end while**
23: Return R_s

in the total space, which ensures that four extreme points of the ellipse, remain within the total space. Then a false location is chosen randomly on the boundary of the ellipsoid area. Intuitively, if s, d and f are close then POIs are retrieved in low cost. To reduce query processing cost, the length of the major axis can be also set by the user instead of computing randomly. The output array R_s is initialized with $\{\emptyset\}$, the entries of distance array $MinD$ are initialized with $infinity$, and r_f is initialized with 0.

Next, a query is sent to the LSP using a function ***INN*** with the parameters f, k, m and $flag$. ***INN*** incrementally retrieves nearest POIs from f. Any existing nearest neighbor algorithm [17] can be used for the function ***INN***. For the first time $flag = 0$ and ***INN*** returns at least k POIs of one type and at least one POI of remaining types, which are sufficient to get k initial POI sets. For rest of the time $flag$ will be 1 and ***INN*** returns k POIs, where POIs can be of any of the required types $1,2,\ldots,m$. In Line 8, the retrieved POIs with respect to f are stored in P. Each time P contains the latest POIs retrieved from the LSP. For a dense distribution of POIs, the known region may expand slowly and require a large number of communications between the user and the LSP. To avoid such a scenario, the user may adjust the number of POIs that need to be retrieved incrementally as $k+\delta$ instead of k in Function ***INN***, where δ is a positive integer. The parameter r_f is updated as the distance between f and the farthest POI from f in P.

After retrieving POIs from the LSP, Function ***IsInsideSearchRegion*** checks and prunes the newly retrieved POI in P, if any POI falls outside the overlapping region of the known region and the search region. For the first time, no POI is pruned as the search region has not been yet computed, i.e., $MinD[k]$ is ∞.

Then the algorithm computes S, all possible new eligible candidate sets of POIs computed with POIs in P and $P^{'}$ using the function ***GenerateSet***. Note that P and $P^{'}$ store all POIs retrieved from the LSP in the current iteration and previous iterations, respectively, which fall inside the search region. In addition to computing S, the function ***GenerateSet*** also updates $P^{'}$ by adding POIs in P to $P^{'}$. The pseudocode for ***GenerateSet*** is shown in Algorithm 2 (please see Section Appendix A.1).

For each set $S^{'} \in S$, the algorithm computes the trip distance $Tdist(s,d,S^{'})$ and updates R_s and $MinD$, if $Tdist(s,d,S^{'}) < MinD[k]$. At this stage, ***IsInsideSearchRegion*** function again prunes the POIs of $P^{'}$ because $MinD[k]$ may have been reduced, which means the area of the search region may have been reduced. A POI that was previously inside the search region may now go outside of the search region. At the end, the ***IsInsideKnownRegion*** function checks whether the known region(circle) covers the search region(ellipse) or not. If yes, the loop terminates, otherwise the loop continues and repeats the process by retrieving more POIs from the LSP. When the known region covers the search region, the k optimal POI sets have been found and stored in R_s.

After that the algorithm checks whether the privacy level achieved so far is greater than or equal to o_l using the function ***CompObLev***. The incremental retrieval of POIs from the LSP continues until the user specified privacy level o_l is achieved by expanding the known region. To compute the obfuscation level, ***CompObLev*** needs to check whether a source destination pair within the known region satisfies the terminating condition of the search and ***CompObLev*** needs to repeat the test for all possible source destination pairs within the known region, which is computationally very expensive. Thus, to make the process faster, in our proposed approach, a user approximates her obfuscation level with monte carlo simulation. For this, we randomly generate 1 million source destination pairs within the known region and compute the percentage of source destination pairs that satisfy the termination condition, and thereby approximate the area of known region that can be considered by adversaries as a refined location of a user. Finally, we compute the obfuscation level as the percentage of the area considered as the user's refined location with respect to the total space.

Appendix A.1 *GenerateSet*

Algorithm 2 *GenerateSet* $(P, P^{'}, m, s, d, MinD[k], flag)$

1: $S \leftarrow \{\emptyset\}$
2: **if** $flag = 0$ **then**
3: $S \leftarrow$ GetSet($NULL, P, m$)
4: $P^{'} \leftarrow P$
5: **end if**
6: **if** $flag = 1$ **then**
7: **for** each POI $p \in P$ **do**
8: $S \leftarrow S \cup$ GetSet($p, P^{'}, m$)
9: $P^{'} \leftarrow p \cup P^{'}$
10: **end for**
11: **end if**
12: **for** each set $S^{'} \in S$ **do**
13: **if** $(OneSetDist(S^{'}) > MinD[k])$ **then**
14: Remove $S^{'}$ from S
15: **end if**
16: **end for**
17: Return S

The ***GenerateSet*** function takes the parameters m, P, $P^{'}$, $MinD[k]$ and $flag$ and returns all possible eligible candidate sets of POIs from the retrieved dataset. Note that for sequenced kTP queries, the algorithm considers the sequence while generating the sets and if the sequence is not fixed, the algorithm considers all possible sequences of visiting POIs while generating the sets. Algorithm 2 describes the detailed steps of the function.

For the first time $flag = 0$ and $P^{'}$ is empty. Thus, sets are generated using the POIs in P with the function ***GetSet***, stored in S and P is copied in $P^{'}$.

For subsequent call of the algorithm, $flag = 1$ and P contains the latest POIs retrieved from the LSP and $P^{'}$ contains the previously retrieved POIs. For each POI $p \in P$, all possible sets are generated with p and other POIs in $P^{'}$ with the function ***GetSet***, sets are added in S, and p is added in $P^{'}$.

After generating the sets, ***OneSetDist*** calculates the distance for each set of POIs. For example, if a set is $\{p_1, p_2, p_3\}$, the distance is computed as $dist(p_1, p_2) + dist(p_2, p_3)$ The distances of eligible sets are also stored to avoid recomputation for future trip distance computation (not shown in Algorithm 2).

For better understanding, Table 3 represents a simulation of function ***GenerateSet*** with an example scenario for $k = 1$ and $m = 2$. The table is divided into two parts based on values of $flag$ (0 or 1). First time when ***GenerateSet*** is called, $MinD[k]$ is ∞ and $flag$ is 0. The first part of the table assumes $P = \{p_1, p_2, p_2^{'}\}$. After execution of ***GetSet***, S includes sets $\{(p_1, p_2), (p_1, p_2^{'})\}$ (as shown in Line 3 of Table 2). Then $P^{'}$ keeps a copy of P for future use. In Line 12.(a), for set $\{(p_1, p_2)\}$, calculated distance from ***OneSetDist*** is less than $MinD[k]$ because $MinD[k]$ is $infinity$. Thus, the condition of Line 13.(a) is False. Same for Lines 12.(b) and 13.(b). Hence no set will not be removed from S and it contains $\{(p_1, p_2), (p_1, p_2^{'})\}$.

Table 3 GenerateSet : Simulation with example

Line	Variable	Value
1.	S	$\emptyset$
3.	S	$\{(p_1,p_2),(p_1,p_2^{'})\}$
4.	$P^{'}$	$\{p_1,p_2,p_2^{'}\}$
12.(a)	$S^{'}$	$\{(p_1,p_2)\}$
13.(a)	Condition	False
12.(b)	$S^{'}$	$\{(p_1,p_2^{'})\}$
13.(b)	Condition	False
17.	S	$\{(p_1,p_2),(p_1,p_2^{'})\}$
1.	S	$\emptyset$
6.	$flag$	1
7.(a)	p	$p_1{'}$
8.(a)	S	$\{(p_1^{'},p_2),(p_1^{'},p_2^{''})\}$
9.(a)	$P^{'}$	$\{p_1,p_1^{'},p_2,p_2^{'}\}$
7.(b)	p	$p_2{''}$
8.(b)	S	$\{(p_1^{'},p_2),(p_1^{'},p_2^{''}),(p_1,p_2^{''}),(p_1^{'},p_2^{''})\}$
9.(b)	$P^{'}$	$\{p_1,p_1^{'},p_2,p_2^{'},p_2^{''}\}$
12.(a)	$S^{'}$	$\{(p_1^{'},p_2)\}$
13.(a)	Condition	True
14.(a)	S	$\{(p_1^{'},p_2^{''}),(p_1,p_2^{''}),(p_1^{'},p_2^{''})\}$
12,13,14...	...	...
17.	S	$\{(p_1,p_2^{''})\}$

We assume that the trip distance via (p_1,p_2) and $(p_1,p_2^{'})$ are 10 and 20, respectively. Thus, the next time when ***GenerateSet*** is called $MinD[k]$ is 20 and $flag$ is 1. The second part of the table assumes $P = \{p_1^{'},p_2^{''}\}$. In Line 7.(a), for $p = p_1{'}$, two sets $\{(p_1^{'},p_2),(p_1^{'},p_2^{''})\}$ are generated and stored in S. After that $p_1{'}$ is added to the $P^{'}$ which stores all the retrieved POIs. Following the same procedure for $p_2^{''}$, S contains sets $\{(p_1^{'},p_2),(p_1^{'},p_2^{''}),(p_1,p_2^{''}),(p_1^{'},p_2^{''})\}$ and all the POIs are added to $P^{'}$. Assume that for set $\{(p_1^{'},p_2)\}$ in 13.(a), ***OneSetDist*** distance, say 30, is greater then $MinD[k]$. Thus, the condition is true and this set will be removed from S. Present S contains $\{(p_1^{'},p_2^{''}),(p_1,p_2^{''}),(p_1^{'},p_2^{''})\}$ in Line 14.(a). Similar process continues for each remaining sets in S. After pruning all the unnecessary sets, finally S contains $\{(p_1,p_2^{''})\}$.

Appendix B Detailed Algorithm: Cloaked Location

In this section, we present our algorithm to evaluate a PkTP query based on the cloaked location of the user. Algorithm 3, PkTP_cloaked_User, shows the pseudocode to process a PKTP query that runs on a user's mobile device. Similar to Algorithm 1, inputs of the algorithm are k, required types $\{1,2,\ldots,m\}$, a user's actual source s and destination d, and the obfuscation level o_l. The output is $R_s = \{p_1^1,p_2^1,\ldots,p_m^1\},\{p_1^2,p_2^2,\ldots,p_m^2\},\ldots,\{p_1^k,p_2^k,\ldots,p_m^k\}$, k sets of POIs that have the k smallest trip distances from s to d.

In the first step of the algorithm, the function ***GenerateRectangle*** generates the source and destination rectangle s_r and d_r based on the user defined obfuscation level o_l. We use the algorithm proposed in [13, 14] to randomly compute the rectangles according to the privacy requirements. Then the algorithm retrieves a candidate answer set that includes optimal answers for all possible source-destination pairs in s_r and d_r, respectively, using the function PkTP_cloaked_LSP (discussed in detail in the later part of this section). The candidate POIs are stored in P.

In Line 5, the function ***ComputeSet*** is called to generate all possible sets with POIs in P and the sets are stored in S. After that for each set $S^{'}$ in S, the algorithm computes the trip distance with respect to s to d. If the trip distance is less than $MinD[k]$ then the answer set R_s and the distance array $MinD$ are updated. Since $MinD[k]$ may have been updated, the algorithm checks whether it is possible to prune some sets from S to reduce the computational overhead using the function ***PruneSet***. The function ***PruneSet*** removes a set

Algorithm 3 PkTP_cloaked_User($input : s, d, k, \{1,2,\ldots,m\}, o_l$)

```
1: s_r, d_r ← GenerateRectangle(s, d, o_l)
2: R_s ← {∅}
3: MinD[1...k] ← {∞}
4: P ← PkTP_cloaked_LSP (k, {1,2,...,m}, s_r, d_r)
5: S ← ComputeSet(P, m)
6: for each set S' ∈ S do
7:     if Tdist(s, d, S') < MinD[k] then
8:         update(R_s, MinD)
9:         PruneSet(S, MinD[k])
10:    end if
11: end for
12: Return R_s
```

$\{p_1^1, p_2^2, ..., p_m^i\}$ from S if $\sum_{i=1}^{m-1} dist(p_i, p_{i+1})$ is already greater than $MinD[k]$. Finally R_s contains k sets of POIs that minimize trip distances from s to d.

Algorithm 4 PkTP_cloaked_LSP ($input : s_r, d_r, k, \{1,2,\ldots,m\}$)

```
1: MinD[1...k] ← {∞}
2: P' ← ∅
3: d' ← ∞
4: flag ← 0
5: repeat
6:     P ← INN_sc(m_p, k, m_p, flag)
7:     Update(r_mp)
8:     P ← IsInsideSearchRegion(s, d, d', P)
9:     S ← GenerateSet(P, P', m_p, s_c, d_c, MinD[k], flag)
10:    for each set S' ∈ S do
11:        if Tdist(s_c, d_c, S') < MinD[k] then
12:            Update(MinD)
13:        end if
14:    end for
15:    d' ← MinD[k] + 2 × (d_1 + d_2)
16:    P' ← IsInsideSearchRegion(s_c, d_c, d', P')
17:    flag ← 1
18: until IsInsideKnownRegion(s_c,d_c,d',m_p,r_mp)=0
19: Return P'
```

Algorithm 4, PkTP_cloaked_LSP shows the details steps for LSP side algorithm. A set $P^{'}$, initialized with $\emptyset$, stores the POIs that provide k smallest trip distances with respect to all possible source-destination pairs in s_r and d_r, respectively. The input to the algorithm are $s_r, d_r, k, \{1,2,\ldots,m\}$ and the output is $P^{'}$.

The notations that we have used for this algorithm are summarized below:

- $s_c(d_c)$: The center of the source (destination) rectangle, $s_r(d_r)$ and one of the foci of the search region (ellipse).
- m_p: The mid point of s_c and d_c. It is also the center point of the known region (circle).
- $d_1(d_2)$: The Euclidean distance from s_c to the corner point of the s_r
- $MinD[1..k]$: An array of k entries, where $MinD[j]$ represents the j^{th} smallest trip distance from s_c to d_c via visiting POIs of required types, where $1 \leq j \leq k$.
- $d^{'}$: The sum of d_1, d_2 and $MinD[k]$. It also represents the length of the major axis of the search region (ellipse).
- r_{m_p}: The radius of the known region centering at m_p.
- IsInsideKnownRegion(s_c,d_c,$d^{'}$,m_p,r_{m_p}): A function that checks whether the known region(circle) covers the search region(ellipse) or not and return yes or no, respectively.

Similar to Algorithm 1, this algorithm uses a function ***INN*** with parameters f, k, m_p and $flag$ to incrementally retrieve nearest POIs with respect to m_p. For the first time $flag = 0$ and ***INN*** returns at least k POIs of one type and at least one POI of remaining types, which are sufficient to get k initial POI sets. For rest of the time $flag$ is 1 and ***INN*** returns k POIs, where POIs can be of any of the required types $1, 2, \ldots, m$. The retrieved POIs with respect to m_p are stored in P. Each time P contains the latest POIs retrieved by ***INN***. Note that with every call of ***INN***, the length of the radius of the known region r_{m_p} increases. Function ***IsInsideSearchRegion*** in Line 8 prunes a POI in P if it falls outside the overlapping region of the known region and the search region. Note that initially $MinD[k]$ is ∞ and thus, no POI from P is pruned in the first iteration.

Algorithm 4 also uses the function ***GenerateSet*** (described in Section Appendix A.1) to generate possible candidate sets and stores them in S. For each set $S^{'}$ in S, the algorithm calculates the trip distance $Tdist(s_c, d_c, S^{'})$ for source destination pair s_c and d_c. If $Tdist(s_c, d_c, S^{'}) < MinD[k]$, the array $MinD$ is updated. In Line 15, the algorithm updates the length of the major axis $d^{'}$ of the search region and uses ***IsInsideSearchRegion*** function to prune the POIs from $P^{'}$ that are not included in the overlapping region of computed known and search regions.

Finally, the algorithm checks whether the current known region covers the search region using the function ***IsInsideKnownRegion***. If yes, the algorithm returns $P^{'}$ to PkTP_cloaked_User. Otherwise, the algorithm repeats the process to identify the candidate answer set that includes POIs for k smallest trip distances with respect to all possible source-destination pairs in s_r and d_r, respectively.

References

1. California dataset: http://www.cs.utah.edu/~lifeifei/spatialdataset.htm.
2. E. Ahmadi and M. A. Nascimento. A mixed breadth-depth first search strategy for sequenced group trip planning queries. In *MDM*, pages 24–33, 2015.
3. X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-aware optimal route search. *PVLDB*, 5(11):1136–1147, 2012.
4. H. Chen, W. Ku, M. Sun, and R. Zimmermann. The multi-rule partial sequenced route query. In *SIGSpatial*, page 10, 2008.
5. C. Chow, M. F. Mokbel, and W. G. Aref. Casper*: Query processing for location services without compromising privacy. *ACM Trans. Database Syst.*, 34(4), 2009.
6. M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive*, pages 152–170, 2005.
7. G. Ghinita. Private queries and trajectory anonymization: a dual perspective on location privacy. *Transactions on Data Privacy*, 2(1):3–19, 2009.
8. G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD*, pages 121–132, 2008.
9. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. *Communications of the ACM*, pages 31–42, 2003.
10. A. Guttman. R-trees: a dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.
11. T. Hashem, S. Barua, M. E. Ali, L. Kulik, and E. Tanin. Efficient computation of trips with friends and families. In *MDM*, pages 931–940, 2015.
12. T. Hashem, T. Hashem, M. E. Ali, and L. Kulik. Group trip planning queries in spatial databases. In *SSTD*, pages 259–276, 2013.
13. T. Hashem and L. Kulik. Safeguarding location privacy in wireless ad-hoc networks. In *Ubicomp*, pages 372–390, 2007.
14. T. Hashem and L. Kulik. "Don't trust anyone": Privacy protection for location-based services. *Pervasive and Mobile Computing*, 7:44–59, 2011.
15. T. Hashem, L. Kulik, and R. Zhang. Privacy preserving group nearest neighbor queries. In *EDBT*, pages 489–500, 2010.
16. T. Hashem, L. Kulik, and R. Zhang. Countering overlapping rectangle privacy attack for moving knn queries. *Inf. Syst.*, 38(3):430–453, 2013.
17. G. R. Hjaltason and H. Samet. Ranking in spatial databases. In *SSD*, pages 83–95, 1995.
18. G. R. Hjaltason and H. Samet. Distance browsing in spatial databases. *ACM TODS*, 24(2):265–318, 1999.
19. H. Hu and D. L. Lee. Range nearest-neighbor query. *IEEE TKDE*, 18(1):78–91, 2006.
20. H. Hu and J. Xu. Non-exposure location anonymity. In *ICDE*, pages 1120–1131, 2009.
21. P. Indyk and D. Woodruff. Polylogarithmic private approximations and efficient matching. *IEEE Pervasive Computing*, pages 245–264, 2006.

22. A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD*, pages 239–257, 2007.
23. A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD*, pages 239–257, 2007.
24. H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *ICPS*, pages 88–97, 2005.
25. F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S. Teng. On trip planning queries in spatial databases. In *SSTD*, pages 273–290, 2005.
26. Y. Li, W. Yang, W. Dan, and Z. Xie. Keyword-aware dominant route search for various user preferences. In *DASFAA*, pages 207–222, 2015.
27. Microsoft. Location & privacy: Where are we headed?, 2011 (accessed September 2, 2011). `http://www.microsoft.com/privacy/dpd`.
28. M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: query processing for location services without compromising privacy. In *VLDB*, pages 763–774, 2006.
29. S. Nutanong, R. Zhang, E. Tanin, and L. Kulik. The v*-diagram: a query-dependent approach to moving KNN queries. *PVLDB*, 1(1):1095–1106, 2008.
30. Y. Ohsawa, H. Htoo, N. Sonehara, and M. Sakauchi. Sequenced route query in road network distance based on incremental euclidean restriction. In *DEXA*, pages 484–491, 2012.
31. S. Papadopoulos, S. Bakiras, and D. Papadias. Nearest neighbor search with strong location privacy. *PVLDB*, 3(1):619–629, 2010.
32. S. Samrose, T. Hashem, S. Barua, M. E. Ali, M. H. Uddin, and M. I. Mahmud. Efficient computation of group optimal sequenced routes in road networks. In *MDM*, pages 122–127, 2015.
33. S. Shang, R. Ding, B. Yuan, K. Xie, K. Zheng, and P. Kalnis. User oriented trajectory search for trip recommendation. In *EDBT*, pages 156–167, 2012.
34. S. Shang, R. Ding, K. Zheng, C. S. Jensen, P. Kalnis, and X. Zhou. Personalized trajectory matching in spatial networks. *VLDB Journal*, 23(3):449–468, 2014.
35. S. Shang, J. Liu, K. Zheng, H. Lu, T. B. Pedersen, and J. Wen. Planning unobstructed paths in traffic-aware spatial networks. *GeoInformatica*, 19(4):723–746, 2015.
36. M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi. The optimal sequenced route query. *The VLDB Journal*, 17(4):765–787, 2008.
37. S. Wang, W. Lin, Y. Yang, X. Xiao, and S. Zhou. Efficient route planning on public transportation networks: A labelling approach. In *SIGMOD*, pages 967–982, 2015.
38. M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *ICDE*, pages 366–375, 2008.
39. M. L. Yiu, C. S. Jensen, J. Møller, and H. Lu. Design and analysis of a ranking approach to private location-based services. *ACM TODS*, 36(2):10, 2011.
40. A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, and S. Zhou. Shortest path and distance queries on road networks: towards bridging theory and practice. In *SIGMOD*, pages 857–868, 2013.
41. A. D. Zhu, X. Xiao, S. Wang, and W. Lin. Efficient single-source shortest path and distance queries on large graphs. In *SIGKDD*, pages 998–1006, 2013.