Anonymous Retrieval of $k$-NN POI in Location Based Services (LBS)

C. Asanya$^1$, and R. Guha$^2$
Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida 32816, USA

Abstract — LBS is a type of location information service accessible through mobile device with the aid of mobile network and mobile device position. Through LBS users can receive information on nearest neighbor (NN) point of interest (POI). LBS need user location and data profile to customize these services. Due to privacy and security concerns, users may be reluctant to share this information. Without this information, it will be difficult to customize these services. Previous solutions offered to process such queries anonymously either imposed too much computation on the user, involve costly transmission, or discloses too much database information. In this paper, we propose idea that allows user to specify and receive exactly $k$ NN (number of POI desired) from LBS with lower transmission cost, minimal user computation, and minimal amount of database information disclosed. We propose two algorithms, first one returns approximate $k$ NN, while the second returns exact $k$ NN.

Keywords: Privacy, Approximate $K$-NN, Exact $K$-NN, Less Communication.

1. Introduction

Location based services are information and entertainment services that are accessible by mobile users through GPS-enabled portable devices and mobile network. It operates by using geographical information of a mobile device [1] to provide real time information such as traffic, entertainment, etc. To effectively provide and customize these services, LBS providers need the location and data profile of a user.

However, users may wish to remain anonymous for various personal reasons. These concerns prompted research into ways to achieve quality of service needed in LBS and at the same time provide user privacy. Privacy requirement may vary between users based on the POI desired. However, the technique to provide this privacy is categorized into three [11]: Two-Tier Spatial Transformation, Three-Tier Spatial Transformation, and Cryptographic Transformation.

Two-tier spatial transformation provide direct communication between user and LBS. User privacy is usually provided through techniques like $k$-anonymity model which demand that every query from a mobile device be indistinguishable related to no fewer than $k-1$ respondents. For instance, for a user (Alice) to issue a query, there has to be other users beside Alice in the vicinity (cloaking region). With this condition satisfied, Bob (server) will be unable to distinguish who the query belongs to. As shown in fig 1, Alice issued a query for $k=7$. With $k=7$, Alice will be availed with the anonymity of the privacy technique. However, Alice has to wait for at least other $k-1$ user, which may not happen on a timely manner, and therefore could delay queries. Some two-tier transformations involve using a dummy location to issue query. Alice issue a query based on a phony location like a landmark, Bob responds to the landmark, Alice then retrieves its POI relative to the landmark.

Three-tier spatial transformations use a trusted central anonymizer. For example, Alice issue a query through a third party server, the server anonymizes Alice location before forwarding the request to the server. However, anonymizer provides not only a single point of attack [8], but also the user has to rely on the honesty of the trusted anonymizer.

Proposed by Ghinita et al, is a cryptographic transformation that does not require a third party intermediary. It is based on the private information retrieval (PIR) scheme that allows a user to retrieve information from a database without revealing the exact information retrieved. In [8], the database is treated as a $X$ bit string represented as a matrix of size $n$, fig 2c. If Alice wants to retrieve the value represented by $X_i$, to preserve privacy, she sends an encrypted query $E(X(q(i)))$, where $E$ is used for encryption. The server responds with an value $r(E(X(q(i))))$, which allows Alice to compute $X_i$. It returns a column to Alice as shown in fig 2b, thereby revealing more information than needed. [22], attempted to minimize the amount of information disclosed by combining PIR and oblivious transfer (OT) scheme to
return NN as shown in fig 2a. However, the content of the cell is still more than the user requested. Remember that each cell of the matrix contains a listing of POI for a particular spatio region. If the LBS is established to make profit by charging per information retrieved, the LBS is disclosing more information than a user will be willing to pay for, or it is willing to give out for free. It also incurs a high transmission cost of at least $O(P_{\max})$, where $P_{\max}$ is the number of POIs in a cell. Also the user has the responsibility of computing its NN from list of NNs. If user wants something different, for instance different gas station, another query has to be issued which results in another transmission and computation just to retrieve one NN.

To solve this problem, we propose a scheme, which will allow a user to specify $k$ (number of information desired). User will then have $k$ NN to choose from if so desired without issuing another query. This will allow the server to transmit only $k$ nearest neighbor required by the user. It will also protect the server from giving out too many of its valued information. The communication cost is $O(k)$ compare to $O(P_{\max})$ to $O(P_{\max} \cdot \sqrt{n})$ incurred in the existing approach, and $k$ is always less than $P_{\max}$.

Our Contributions are:
1) We developed a method for finding $k$ approximate nearest neighbor to avoid the overhead of transferring more object than necessary, thereby cutting down on the transmission and computation cost.
2) We also propose an improvement that utilizes the exact user location to find the exact $k$ nearest neighbor without disclosing user location.
3) We also show that the transaction cost is much less than the comparable scheme.

The rest of the paper is organized as follows; Section 2 provides the problem formulation, and section 3 presents related works. Section 4 describes the architectural framework in our system, while 5 explains the implementation process. 6 Evaluates our protocol. Section 7 compares our scheme with existing protocol, while conclusion and future work is in 8.

2. Problem Formulation

This section provides definition for the $k$-NN problem set. Find the user location in Euclidean space and return $k$ nearest neighbor to the user without compromising its location or divulging information requested.

2.1 Definition

Let $p$ be data points of interest in Euclidean space represented as a square grid $G$ of cells $c$. Let $P_{\max}$ be the maximum number of $p$ in $c$. For a user $u$ in location $l \in c$ wishing to obtain its nearest neighbor, let $k \in P_{\max}$ represent the number of point of interest desired by the user in $c$. $k$-NN query returns to the user $k$ data object in $c$ whose distances from $u$ are less or equal to the data objects $\in c \in G$.

2.2 Solution 1: (Approximate $k$ NN)

For $p \in c \in G$, assign Hilbert value to $p$. For every $c \in G$, sort $p \in c$ according to their Hilbert value. Create R tree of $k$ size with index key, where $k$ is the number of POI requested by the user. User finds the appropriate key based on its Hilbert value and use double PIR to retrieve its $k$ NN.

2.3 Solution 2: (Exact $k$ NN)

Server sends $G$ and arbitrary point for each $c \in G$. User sends back $k$ and its position relative to the arbitrary point in $c$. For each $c \in G$ server computes NN until k is reached. User retrieves k NN using double PIR.

3. Related Work

Most proposed privacy models in LBS use third party anonymer which acts as intermediary between the user and the server, however, user location is usually known to the third party. Another form is K-anonymity that demands that location information contained in a message sent from a mobile user to a LBS should be indistinguishable from at least k-1 other messages. For the most part it does not protect information requested.

[9], offered a solution that uses spatio-temporal cloaking to transform each original message from a mobile node into a privacy protected message with the k-anonymity guarantee. The cloaking algorithm is run by the protection broker on a trusted server that anonymerizes the message by cloaking the location information contained in the messages before forwarding them to the LBS provider(s).

[14], as in [9], allow each mobile node to specify the minimum level of anonymity desired as well as the maximum temporal and spatial resolutions it is willing to tolerate. For each query, user specifies its waiting interval tolerance, if within this interval other k-1 users issue a query, then all the queries will be combined into a single cloaking region, else the user query will be rejected.

Shin et al [6], proposed a profile based anonymerization model. It generalizes both location and profile to the extent specified by the user. Location is generalized so that the generalized spatio-temporal region includes at least k-1 other users in the region and at the same time contains at least additional k-1 users with identical profiles of the user.

In [2], mobile nodes communicate with external services through a central anonymity server which is part of the trusted computing base. In an initialization phase, the nodes will

![Fig. 3. Approximate k NN](image-url)
set up an authenticated and encrypted connection with the anonymity server. When a mobile node sends position and
time information to an external server, the anonymity server
decrypts the message, removes any identifiers, and distorts the
position data according to the prescribed cloaking algorithms
to reduce the re-identification risk.

One of the downside to the third party anonymizer is the possibility that the trusted server can turn into an adversary
and therefore can compromise privacy.

Cryptographic approach was proposed to help eliminate
third party, though some trusted third party scheme have used
encryption, but user location is known to the third party.
Encryption allow user to secretly retrieve information while
keeping the server oblivious of the exact information retrieved.
The server sends the user encrypted information of which only
the user is able to decrypt.

In [7], a solution that uses a secure framework for protecting
both the user’s location information and user’s usage profiles
through oblivious transfer and homomorphic encryption was
proposed. However, an intermediate proxy was introduced in
the protocol to interact with the server and the user, which
makes the privacy partly dependent on the proxy’s honesty.
A similar approach was proposed in [10]. It also requires
third party. The server stored information is encrypted by
using double oblivious transfer scheme to prevent server from
knowing which information retrieved by the user.

Similar to our work is the approach used in Ghinita et al [8].
PIR scheme based on the Quadratic Residuosity Assumption
(QRA) was used. The database is modeled as a string X
of n-bit. User wishing to obtain bit X_i, will engage in the PIR
protocol as described in section 4.1. From the list of POIs,
user will be able to compute its NN. However, the protocol
has some shortcomings. It reveals to the user more than what
is required, and also has high transaction cost.

R. Vishwanathan, proposed an improvement that combined
PIR and OT scheme. The scheme used a two phase protocol
(PIR and OT) to minimize server returned information.
However, it also reveals more objects than necessary, imposing
on the user the task of calculating its NN from a
list of NNS.

Neither of these two approaches was able to prevent
the server from disclosing more database information than
necessary, nor minimize user computation and transaction
cost. They also failed to offer user any choice in case a user
is displeased with the information retrieved. For example,
a user in a moving vehicle may receive NN that is already
behind or may be logistically difficult to reach, user has to
issue another query to obtain NN, while with our scheme user
has k choices to choose from without issuing another query.
In [23, 24, 25, 27 and 28], attempt was made to return k-nn,
but it did not offer neither location nor data privacy. [26]
proposed idea that returns only approximate k-nn, but depends
on having other users within the cloaked region to be effective.

4. System Architecture

We used PIR protocol based on computational intractability
of [3]. We also use dual encryption process as in [22], and
[10]. We propose two algorithm to retrieve k-NN; the first
returns approximate k-NN POI to the user with O(log(k))
computation, and O(k) communication cost. The second
returns exact K-NN.

4.1 PIR Framework

Private Information Retrieval (PIR) is a protocol that allows
a user to retrieve information from a database without revealing
the exact information retrieved. Different flavors have been
proposed over the years [4]. The earlier PIR scheme requires
replication of the database [13]. Kushilevitz and Ostrovsky
introduced a computational private information retrieval (cPIR)
that requires one database. We focused our work on Com-
putational PIR applicable to one database tuple distribution
structure as was introduced in [3].

cPIR is based on the premise that there is no known function
in polynomial bounded time that will allow a database of size n
to distinguish a query for the i^{th} bit and a query for the j^{th}
bit \( 1 \leq i, j \leq n \). The PIR scheme is
such that for a database of n-bit string X from which a
user wishes to obtain bit X_i while keeping secret index i
from the database, it requires that the user queries divulge
no information about i. cPIR as described in [3] relies on
Quadratic Residuosity Assumption (QRA).

QRA was used in cryptography by S. Golwasser and M.
Bellare [15], and it states that, it is computational hard to
find the quadratic residues in modulo arithmetic of a large composite number \( N = q \cdot q' \), where \( q, q' \) are large primes. i.e, If \( N \) is a natural number, Define

\[
\mathbb{Z}_N^\ast = \{ x | 1 \leq x \leq N, \gcd(N, x) = 1 \}
\]

(1)

Let the quadratic residuosity predicate be defined as \( Q_N(y) = 0 \) if \( \exists x \in \mathbb{Z}_N^\ast \) such that \( x^2 = y \mod N \) and \( Q_N(y) = 1 \) otherwise. If \( Q_N(y) = 0 \) (i.e. \( y \) is a square \( y \)), then \( y \) is said to be quadratic residue (QR), and if \( Q_N(y) = 1 \) (i.e. \( y \) is a non-squares \( y \)), then \( y \) is said to be quadratic non-residue (QNR). If

\[
\mathbb{Z}_N^{+1} = \{ y \in \mathbb{Z}_N| y = 1\} \}
\]

is true, then half of the numbers in \( \mathbb{Z}_N^{+1} \) are \( QR \), and half are \( QNR \). If \( q \) and \( q' \) are large enough \( \frac{q}{2} \) bit primes, for every constant \( c \) and a family of computational bounded polynomial circuit \( C_{k_0}(y) \) there exist an integer \( k_0 \) such that \( \forall K > k_0 \)

\[
Pr_{y \in \mathbb{Z}_N^+}|C_{k_0}(N, y) = Q_N(y)| < \frac{1}{2} + \frac{1}{K^c}
\]

(3)

If equation 3 holds, and for large enough \( k_0 \), the probability of differentiating between a \( QR \) and \( QNR \) is very small, i.e. the server will be unable to unmask the information requested by user by attempting to find if \( y \in QR \) or \( y \in QNR \).

4.2 k-NN Spatial Search

We implemented method similar to [5] and [12] to find \( k \) NN. It makes use of priority queue to keep track of the points. Readers interested in details can read [5].

4.3 Database Structure

Our scheme is implemented using the database structure as shown in figure 5. It is similar to [22] and [8]. It allows user to secretly retrieve nearest neighbor. It is of size \( n \) represented as a square matrix \( M = \sqrt{n} \times \sqrt{n} \) indexed by \( X_i \), for \( (i = 1, \ldots, \sqrt{n}) \). Figure 5 shows database of \( n = 16 \). \( X_i \) represent the section of the database corresponding to a cell in the Euclidean space as shown in figure 4. The contents of \( X_i \) is the POI found in the grid cell for POI from 1 to \( P_{max} \), where \( P_{max} \) is the maximum number of POI in each cell. All \( X_i \) have equal number of POIs.

4.4 Space Partitioning

Figure 4 shows a Euclidean space enclosing POI’s. A square grid \( G \) is super-imposed on top of the space. User location in the space is indicated by \( U \), and \( p_1, p_2 \), etc. are the points in the space enclosed by the grid.

5. Implementation

The goal is to find \( k \) nearest neighbor POI without revealing user location, or the requested data profile. Recall that \( k \) is the number of nearest neighbor point of interest user wishes to retrieve from the server.

5.1 Approximate k-NN Algorithm

The database for the Approximate k-NN is mapped out to Hilbert curve ordering. Hilbert curve is useful for mapping between 2D and 1D space while still preserving locality. For instance, if \( (x, y) \) are the coordinates of a point within the unit square, and \( d \) is the distance along the curve when it reaches that point, then points that have nearby \( d \) values will also have nearby \( (x, y) \) values [29]. Simply put, if two POIs are close in the 2-D space, they are likely to be close in the Hilbert ordering, as well, therefore nearest neighbor to a user is the point of interest that has Hilbert value closest to that of the user [8]. To retrieve \( k \) NN POI, our algorithm follows the following steps;

1) Step 1: Server creates Hilbert ordering of the database in a matrix \( M = \sqrt{n} \times \sqrt{n} \). Each object in \( M \) represents the POI in cell \( c \). Cells are padded to create cells of equal size, and the POIs are of equal bits. This will prevent the server from inferring the requested POI based on the size of the cell or amount of bit transferred. The POIs were sorted based on their Hilbert values. Figure 3a depicts a cell from a grid \( G \) of \( n \) cells. The cell has 9 POI from \( p_1, \ldots, p_9 \). All the points are assigned Hilbert values based on their distance to one other. Points with closer values are closer in space. For example, \( p_6 \) has \( H \) value of 40, \( p_4 \) has \( H \) of 30, while \( p_2 \) has \( H \) of 50.

2) Step 2, Initialization: User in location \( U \) initiates a query, and sends \( k \) to obtain its \( k \) nearest neighbor, server creates one level R-tree with nodes of size \( k \), and indexed the nodes as shown in figure 8(a,b). Each index key is greater or equal to the left node which values are Hilbert, Server then sends the key to the user together with grid \( G \). User finds the grid enclosing it and its Hilbert value.

3) Step 3, k-NN Protocol Procedure with double PIR: For a user wishing to retrieve k-NN, if, for instance user \( H \) value is 25, with the help of the index key, user will be requesting all the points belonging to the values which are to the left of index key 30 of figure 8b.

Let database \( D \) of size \( n \) be organized as a string of \( X \) elements in \( s \times s \) matrix \( M \), where \( s = \sqrt{n} \). Let
with following equation

\[ M_{ab} \text{ figure 5 represent element } X_{ii} \text{ that user wish to retrieve, and let figure 8c depicts the database entry for } M_{ab}. \text{ User randomly generates modulus } N = q \cdot q', \text{ with a query message } y = [y_1, \ldots, y_a], \text{ and } x = [x_1, \ldots, x_k], \text{ such that}
\]

\[ y_b \in QNR, \text{ and } \forall j \neq b, y_j \in QR, \text{ and } x_a \in QNR, \text{ and } \forall r \neq a, x_r \in QR \]

It then sends query PIR(ab) to the server.

4) Step 4, Server Response: Server receives PIR(ab) from the user. Server is unable to tell if \( y \) and \( x \) used for encrypting user query \( y \in QNR \) or QR due to the computational intractability of a large prime. Define

\[ z_r = \prod_{j=1}^{s} w_{rj} \quad (4) \]

and

\[ z_c = \prod_{r=1}^{s} w_{rj} \quad (5) \]

where \( z_r \) and \( z_c \) are vector. Let

\[ Z_\alpha = z_r \times z_c \quad (6) \]

and \( w_{rj} = y_j^2 \) if \( M_{rj} = 0 \), otherwise \( w_{rj} = y \) if \( M_{rj} = 1 \), for \( j = 1, \ldots, s \), and \( w_{rj} = x_j^2 \) if \( M_{rj} = 0 \), otherwise \( w_{rj} = x \) if \( M_{rj} = 1 \), for \( r = 1, \ldots, s \).

Server runs PIR protocol on user request, it computes for every row and column equation 4, 5 and 6 and returns \( Z = z_1 \ldots z_k \) with \( O(k) \).

Since the user knows \( q \) and \( q' \), it will be easy to compute the following equation

\[ \left( Z_{\alpha}^{q-1} = 1 \mod q_1 \right) \land \left( Z_{\alpha}^{q'-1} = 1 \mod q_1 \right) \quad (7) \]

if the above equation is true, then \( Z_\alpha \in QR \) else \( Z_\alpha \in QR \). Therefore, \( M_{ab} \) computes to 0 if \( Z_\alpha \in QR \) else, if \( Z_\alpha \in QNR \) \( M_{ab} \) computes to 1. User does this for all the \( k \) objects requested and that is the \( k \)-NN. For a case where \( k = 3 \), and the user Hilbert value is 25, user will receive \( p_1, p_0 \) and \( p_4 \) as its nearest neighbor. User can then decide which one best suited its need.

Algorithm 1: (Approximate \( k \)-NN)

**Input:** grid size \( n \), number of point of interest \( POI\#\#mb \)

**Output:** \( k \) NN point of interest \( p \) from \( p_1 \) to \( p_k \)

**Procedure:**

1) set the size of the grid; \( cell\#\#mb = n \), and \( POI\#\#mb = P_{max} \);

2) for each grid \( c \):
   - for \( i = 1 \) to \( p_{max} \); \( p_1 \) to \( p_{max} = hV \)

3) Sort \( p \) in order of closeness

4) for each object \( p \) in \( c \):
   - if \( hV \) for \( p_i \), from 1 to \( p_{max} < hV \) \( \text{for } p_i, + 1 \) to \( p_{max} - p_i \) sort \( HV = p_i \);

5) for \( i = 1 \) to \( n \)
   - for \( p = 1 \) to \( p_{max} \)
   - for \( j = 1 \) to \( k \) if \( (j = k), key = hV[j] \);

6) Client

7) for each \( key \)
   - if \( dhV < key \), modulus \( N = q \cdot q' \);

query message \( y = [y_1, \ldots, y_a] \);

and \( x = [x_1, \ldots, x_k] \), such that

\[ y_b \in QNR, \text{ and } \forall j \neq b, y_j \in QR, \text{ and } x_a \in QNR, \text{ and } \forall r \neq a, x_r \in QR \]

8) Server computes for every row and column

9) for \( j = 1 \) to \( s \) and \( r = 1 \) to \( a \); for \( k \) times.

\[ z_r = \prod_{j=1}^{s} w_{rj} \text{ and } z_c = \prod_{r=1}^{s} w_{rj} \]

\[ Z_\alpha = z_r \times z_c \]

return \( Z = z_1 \ldots z_k \);

5.2 Exact k-NN Algorithm

From figure 3a, the actual NN to the user is \( p_0, p_4 \) and \( p_6 \), but the server returned \( p_0, p_4 \) and \( p_1 \). To solve this problem we propose the second algorithm. The algorithm returned the exact \( k \)-NN by using exact user location to find its NN without compromising user privacy.

1) Server offline Phase: Server creates voronoi tessellation as in Ghinita et al using the set of POIs. It then super-impose a regular grid \( M \) of size \( \sqrt{(n)} \times \sqrt{(n)} \) on top of the voronoi diagram, figure 7. For every cell \( c \) of the grid, it determines all voronoi cell intersecting it, and adds the corresponding POI to \( c \). Cell \( c \) therefore contains all potential NNs of every location inside voronoi diagram intersecting it. For example in figure 7, cell \( A1 \) will contain \( P1 \) and \( P3 \), while \( A3 \) contain \( P1, P3 \) and \( P4 \). The cells are padded if necessary to create cells of equal size. All objects in \( M \) are of equal bits. Remember that number of bits for the POI can be of any size as long as they are consistent in \( M \). For every cell \( c \) of the grid, server chooses arbitrary point \( P_a \) and maintains in an offline phase, and updates as necessary.

2) Initialization: When a user initiates a query to obtain its \( k \)-NN, server sends \( G \) and \( P_a \) for each cell in the grid. User finds the cell enclosing it, and calculates its distance \( \pm d \) from \( P_a \). User sends to the server \( \pm d \) and \( k \), representing its distance from \( P_a \), and the number of information desired respectively. Server then add \( \pm d \) to all the \( P_i \) in the grid to get \( P_{new} \). This will ensure that the returned POI by the server is the exact \( k \)-NN to the user. The user location will now be \( P_{new} = P_a \pm d \).

For each cell \( c \) in \( G \), server executes spatial search to find \( K \)-NN using \( P_{new} \). The server does not know the location of the user nor its cell. Server only knows \( P_{new} \) figure 6, which can be located in any cell. Server stops execution of spatial search as soon as \( k \)-NN is found for the entire cell in the grid.

3) User Request: As was in the first case. If \( M_{ab} \) represent element \( X_{ii} \) user wish to retrieve, user randomly generates modulus \( N = q \cdot q' \), with a query message \( y = [y_1, \ldots, y_a] \), and \( x = [x_1, \ldots, x_k] \), such that

\[ y_b \in QNR, \text{ and } \forall j \neq b, y_j \in QR, \text{ and } \]
It then sends query PIR(ab) to the server.

4) Server Response: Server receives PIR(ab) from the user, but is unable to tell if \( y \) and \( x \) used to encrypt user query \( \in QNR \) or \( QR \) due to the hardness of problem involving a large prime.

Server runs PIR protocol on user request. It computes for every row and column equation 4, 5 and 6 and returns \( Z = z_1 \ldots z_k \) with \( O(k) \).

Since the user knows \( q \) and \( q' \), it will be easy to compute equation 7. If equation 7 is true, then \( Z_\alpha \in QR \) else \( Z_\alpha \in QNR \). Therefore, \( M_{ab} \) computes to 0 if \( Z_\alpha \in QR \) else, if \( Z_\alpha \in QNR \) \( M_{ab} \) computes to 1. User does this for all the \( k \) objects requested and that is its \( k \)-NN.

6. Evaluation

Our goal is to show that our system incurs less communication cost when transmitting nearest neighbor POI compare to other similar scheme. We also show that at the maximum user request our scheme is equal or less in communication cost to other comparable method.

6.1 Experimental Setup

The experiment was performed on a PC running Windows 7 with i5 processor and 2.30GHz CPU cycle with 3.85GB RAM. We used synthetic dataset of an independent and uniform distributed POI. We varied \( k \) from 5 to 15, and \( n \) from 100 to 144, and POI for each cell from 15 to 20. Each POI has 8 bits. Transmission payload is 50 bits. We implemented a 10MBits data transmission rate. We did not consider any propagation delay as we take the medium to be the same.

6.2 Results

Table 1 shows the transmission in bits for varying \( k \) up to \( P_{max} \). As the table depicts, the payload transferred in our scheme \((k\text{-NN})\) increases as user request increase, as expected up to or less than the minimum for previous scheme. In the previous scheme, it is always the same regardless of user request. Table 11 shows the effect of increasing grid size \( n \) and \( P_{max} \) in a grid. As the table show, it has no effect on our scheme, while it increases for the previous method. Scheme 1 is the worst with a 36% increase, while scheme 2 has 19% increase. Figure 9a shows transmission cost for different \( k \). Our scheme has lower cost when \( k < P_{max} \), it catches up with PS1 as user request equals \( P_{max} \). As figure 9b shows, as \( P_{max} \) increases, our scheme performs better.

7 Discussion and Comparison

In [8], server returns an answer set as shown in figure 2a. It will take \( O(P_{max} \cdot \sqrt{q}) \) computation to retrieve NN. In [22], server returns are shown in figure 2a, which requires \( O(P_{max}) \) computation to retrieve NN. We were able to reduce the computation to \( O(k) \) to retrieve \( k \) NN by returning only \( k \) NN to the user. There will be no need for user to compute and find its NN from the returned set as in [8] and [22] since server only returned what user requested. In the previous scheme, server has to re-issue another query to receive a different NN, while our scheme offered \( k \) choices at one snap shot. Our scheme also prevented server from disclosing too much database information by sending only what the user requested.

We also theoretically showed that server was unable to decipher user location, or request based on computational intractability of [3]. The transmission time is less than comparable scheme, unless for cases where \( P_{max} < k \), then transmission time will be equal, since only \( P_{max} \) will be transmitted. Hilbert ordering and voronoi diagram creation are all done by the server in an offline phase. The largest piece of computation after user request is the NN search in algorithm 2, however it only runs \( k \) times.

### Table 1: Bit Size For n=100, and POI per Cell=15

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k ) NN</td>
<td>90</td>
<td>114</td>
<td>130</td>
<td>146</td>
<td>170</td>
</tr>
<tr>
<td>Previous Scheme 1</td>
<td>170</td>
<td>170</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
<tr>
<td>Previous Scheme 2</td>
<td>1250</td>
<td>1250</td>
<td>1250</td>
<td>1250</td>
<td>1250</td>
</tr>
</tbody>
</table>

### Table 2: Bit Size for n=114, and POI per Cell=20

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>8</th>
<th>10</th>
<th>12</th>
<th>15</th>
<th>17</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k ) NN</td>
<td>90</td>
<td>114</td>
<td>130</td>
<td>146</td>
<td>170</td>
<td>186</td>
<td>210</td>
</tr>
<tr>
<td>Prev Sch 1</td>
<td>210</td>
<td>210</td>
<td>210</td>
<td>210</td>
<td>210</td>
<td>210</td>
<td>210</td>
</tr>
</tbody>
</table>
8. Conclusion and Future Work

In this paper we proposed a new idea for finding \( k \) NN without the overhead imposed in some previous method. We were able to experimentally show that transmitting \( k \) NN has less communication cost. Requesting \( k \) NN is more efficient since it prevents query re-issue in the case of unsatisfied user. It provides opportunity for a user to make a choice without additional communication. By returning exact number of POI requested by the user, the protocol minimizes the number of objects returned in an answer set thereby reducing the transmission cost of sending the answer set to the user, also the user does not need to compute its NN, and user has more choice that does not require re-issuing of query to obtain next NN. In future we intend on using the parallel processing power of GPU to optimize our algorithm. We also intend on exploring a different way of reducing user computation when finding user distance from arbitrary point.

REFERENCES


Fig. 9. Transmission Cost