THE NATIONAL UNIVERSITY
of SINGAPORE

School of Computing
Lower Kent Ridge Road, Singapore 119260

TRB6/06

Preserving Anonymity in Location Based Services

Panos KALNIS, Gabriel GHINITA, Kyriakos MOURATIDIS and Dimitris PAPADIAS

June 2006

Technical Report

Foreword

This technical report contains a research paper, development or tutorial article, which has been submitted for publication in a journal or for consideration by the commissioning organization. The report represents the ideas of its author, and should not be taken as the official views of the School or the University. Any discussion of the content of the report should be sent to the author, at the address shown on the cover.

JAFFAR, Joxan
Dean of School

Preserving Anonymity in Location Based Services

Panos Kalnis[1] Gabriel Ghinita[1] Kyriakos Mouratidis[2] Dimitris Papadias[2]

[1]Computer Science Department
National University of Singapore
{kalnis, ghinitag}@comp.nus.edu.sg

[2]Computer Science Department
Hong Kong University of Science and Technology
{kyriakos, dimitris}@cs.ust.hk

ABSTRACT

The increasing trend of embedding positioning capabilities (e.g., GPS) in mobile devices facilitates the widespread use of Location Based Services. For such applications to succeed, the privacy and confidentiality issues are of paramount importance. Existing techniques, like encryption, safeguard the communication channel from eavesdroppers. Nevertheless, the queries themselves may disclose the physical location, identity and habits of the user.

In this paper, we present a framework for preserving the anonymity of users issuing *spatial queries* to Location Based Services. We propose transformations based on the well-established $\mathcal{K}$-anonymity technique to compute exact answers for Range and Nearest Neighbor queries, without revealing sensitive information about the user. Our methods optimize the entire process of anonymizing the requests and processing the transformed spatial queries. Extensive experimental studies suggest that our methods are applicable to real-life scenarios with numerous mobile users.

1. INTRODUCTION

In the recent years, positioning devices (e.g., GPS) have gained tremendous popularity. Navigation systems are already a commodity in the automobile industry and, together with wireless communications, facilitate exciting new applications. General Motor's OnStar system, for example, supports on-line rerouting to avoid traffic jams and automatically alerts the authorities in case of an accident. More applications based on the users' location are expected to emerge with the arrival of the latest gadgets (e.g., iPAQ hw6515, Mio A701) which combine the functionality of a mobile phone, PDA and GPS receiver.

Imagine the following scenario: Bob, who is a sports fan, uses his GPS enabled mobile phone to ask the query "Find the nearest fans' club of Juventus". This query can be answered by a Location Based Service (*LBS*) in a publicly available web server (e.g., Google Maps). Bob, however, does not want to disclose to Alice his fondness towards Juventus, so instead of sending the query to the LBS, he uses an anonymizer. He establishes a secure connection (e.g., SSL) with the anonymizer which is a trusted server (services for anonymous web surfing are commonly available nowadays). The anonymizer removes the user ID from the query and forwards it to the LBS; the answer from the LBS is also routed to Bob through the anonymizer.

Nevertheless, the query itself unintentionally reveals sensitive information. In our example, in order to answer the Nearest Neighbor (NN) query, the LBS requires the coordinates of the user and the selection criterion (i.e., Juventus fans' club). Since the LBS is not trusted, the communication channel between the LBS and the anonymizer is not secure. Alice can eavesdrop this connection and acquire all the sensitive information except, of course, the identity of the user. The next step is to relate the coordinates with a specific user. Alice may choose from a variety of techniques ranging from physical observation of Bob, to triangulating his mobile phone's signal[1]. She may also employ publicly available databases[2]. If, for instance, Bob uses his mobile phone within his residence, Alice can easily convert the coordinates to a street address (most on-line maps provide this service) and relate the address to Bob by accessing an on-line white pages service.

The example demonstrates how the identity and favorite football team of Bob may be revealed when using a Location Based Service, despite the encryption. In practice, users would be reluctant to access a service that may disclose sensitive information (e.g., corporate, military), or their political/religious affiliations and alternative lifestyle. Motivated by this fact, we develope methods to guarantee privacy by adapting the well established $\mathcal{K}$-anonymity technique to the spatial domain.

$\mathcal{K}$-anonymity [12, 14] has been used in statistical databases as well as for publishing census, medical and voting registration data. A dataset is said to be $\mathcal{K}$-anonymized, if each record is indistinguishable from at least $\mathcal{K}-1$ other records with respect to certain identifying attributes. In the Location Based Services domain, a similar idea appears in the work of Ref. [5, 6]. Both papers focus on concealing the location of the user. To achieve this, instead of reporting the exact coordinates to the LBS, they construct an *Anonymizing Spatial Region* ($\mathcal{K}$-ASR) which encloses the locations of $\mathcal{K}-1$ additional users. These approaches have several weaknesses: (i) The methods for constructing the $\mathcal{K}$-

[1]Phone companies can estimate the location of the user within 50-300 meters, as required by the US authorities (E911).

[2]According to [14], 87% of the US population is uniquely identified by the combination of $\langle ZIP, Gender, DateOfBirth \rangle$.

Technical Report: TRB6/06, June 2006
Computer Science Department
National University of Singapore

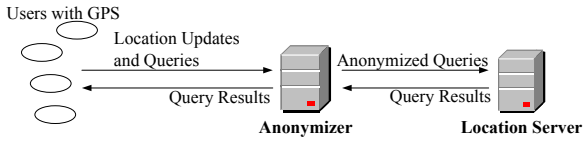


Figure 1.1: System architecture

ASR are inefficient, (ii) they assume approximate answers to the queries, (iii) they do not discuss how queries are executed and, most importantly (iv) anonymization may fail for some data distributions.

In this paper, we propose a framework which deals with the entire procedure of anonymization and query processing for Location Based Services. Our system supports Range and k-Nearest Neighbor[3] (kNN) queries, and retrieves the exact answers without revealing the location or identity of the user. We assume that all the users carry a positioning device (e.g., GPS) reporting their exact coordinates. The system includes a trusted anonymizer and a number of untrusted location servers (LS) providing various Location Based Services (LBS). The system architecture is shown in Figure 1.1.

Initially users set up authenticated and encrypted connections with the anonymizer. Periodically, users report to the anonymizer their coordinates through the secure connection. When a user wants to query the LBS, he sends a message to the anonymizer describing the query type and parameters. Continuing our example, if Bob submits his NN query through our system, the following steps are performed:

1. At the anonymizer's side we identify $\mathcal{K}-1$ auxiliary users in the vicinity of Bob and construct the $\mathcal{K}$-ASR, where $\mathcal{K}$ depends on Bob's privacy requirements.

2. The input to the LBS is the $\mathcal{K}$-ASR. Therefore, instead of finding the NN of Bob, the LBS must retrieve the NN of every point in the $\mathcal{K}$-ASR. By this definition, all the Juventus fan clubs inside the $\mathcal{K}$-ASR and the ones which are closest to any point of the perimeter of the $\mathcal{K}$-ASR, belong to the result G, which is a superset of Bob's answer. Notice that G also contains the NN of each one of the $\mathcal{K}-1$ auxiliary users. Therefore Alice cannot identify Bob as the originator of the query either from the $\mathcal{K}$-ASR or from the result set G.

3. The entire set G is returned to the anonymizer where Bob's answer is extracted. Obviously the size of G, hence the processing cost (CPU and I/O) at the LBS and the communication cost between the LBS and the anonymizer, depend on the choice of the $\mathcal{K}$-ASR.

A naïve solution would pick $\mathcal{K}-1$ auxiliary users and send $\mathcal{K}$ independent NN queries to the LBS. The drawback is that Alice will learn the exact locations of $\mathcal{K}$ users. Assume that Bob sends subsequent queries in the near future (possibly with a refined selection criterion). Then Alice can identify Bob by noticing that a certain location (i.e., Bob's coordinates) appears in the intersection of the sets of $\mathcal{K}$ users. Also note that we cannot choose a random $\mathcal{K}$-ASR. For instance, if Bob lives in a rural area, even a relatively large $\mathcal{K}$-ASR may enclose only his residence, therefore his identity will be compromised. In a densely populated area, on the other hand, an inappropriate $\mathcal{K}$-ASR could return a large amount of unnecessary results.

In summary, our contributions are: (i) We propose a framework for preserving anonymity in LBSs. We also Show how to implement range and kNN queries in our framework. (ii) We describe efficient methods for constructing the $\mathcal{K}$-ASR based on fractals and on Conceptual Partitioning [10]. Our goal is to assemble fast an appropriate $\mathcal{K}$-ASR that preserves anonymity while minimizing the expected cost at the LBS and the required bandwidth between the LBS and the anonymizer. (iii) We develop novel algorithms to compute the kNNs of regions, as opposed to points. (iv) Finally, we conduct an extensive experimental evaluation of our system. The results confirm that our methods minimize the CPU, I/O and communication cost at the anonymizer and the LBS; therefore, they are scalable to a large number of mobile users.

The rest of the paper is organized as follows: Section 2 presents the related work. Next, Section 3 deals with the construction of the $\mathcal{K}$-ASR at the anonymizer, followed by Section 4 where we explain the query processing algorithms at the LBS. The results of our experiments are illustrated in Section 5. Finally, Section 6 concludes the paper and presents the directions of our future work.

2. RELATED WORK

$\mathcal{K}$-anonymity was first discussed in relational databases where published statistical data (e.g., census, medical) should not be linked to specific persons. Samarati and Sweeney [12, 14] proposed the following definition: A relation satisfies $\mathcal{K}$-anonymity if every tuple in the relation is indistinguishable from at least $\mathcal{K}-1$ other tuples with respect to every set of quasi-identifier attributes. Quasi-identifiers are sets of attributes (e.g., date of birth, gender, zip code) which can be linked to publicly available data to uniquely identify individuals. Two techniques are used to transform a relation to a $\mathcal{K}$-anonymized one: *Suppression*, where some of the attributes or tuples are removed and *generalization*, which involves replacing specific values (e.g., phone number) with more general ones (e.g., only area code). Both techniques result to information loss. Ref [2] and Ref [8] discuss efficient algorithms for anonymizing an entire relation while preserving as much information as possible. In Ref [16] the authors consider the case where each individual requires a different degree $\mathcal{K}$ of anonymity, while Aggarwal [1] shows that anonymizing a high-dimensional relation results to unacceptable loss of information due to the dimensionality curse. Finally, Machanavajjhala et.al [9] propose ℓ-diversity, an anonymization method under the assumption that the attacker has domain-specific knowledge.

Our research is closer to the work of Ref [5] and Ref [6]. The authors of the first paper consider mobile users who send queries to the anonymizer together with the required degree $\mathcal{K}$ of anonymity and a spatial cloaking range δ_x, δ_y. The system generates a graph where each vertex represents a user; there is an edge between vertices if one user lies inside the spatial cloaking range of the second and vice-versa. Then the graph is searched for cliques of $\mathcal{K}$ users and their enclosing rectangle is sent to the LBS. Note that the probability of $\mathcal{K}$ users in close proximity to send a query at the same time, is rather low. Therefore, the system allows the users to set an additional temporal cloaking interval δ_t; if a

[3]Notice that the symbol k for kNN queries is different from $\mathcal{K}$, the degree of anonymity.

clique cannot be found within this interval, the corresponding query is rejected. The paper does not discuss how does the user select appropriate cloaking ranges, neither considers query execution.

Ref. [6], on the other hand, assumes that users must report periodically their physical location, and focuses on concealing the exact location without considering queries at all. The anonymizer indexes the locations of all users in a Quadtree [13]. For a user u, it traverses the Quad-tree until it encounters a quadrant which includes u and less than $\mathcal{K}-1$ additional users. Then it selects the parent of that quadrant as the anonymizing region $\mathcal{K}$-ASR. We do not adopt this method in our system for two reasons: (i) $\mathcal{K}$-ASR may include many more users than $\mathcal{K}$, rendering query processing inefficient and (ii) under certain conditions, the location of the user may be revealed (see Section 3).

Our query processing method is related to the Continuous-NN algorithm proposed by Tao et.al. [15] to find the kNNs of line segments. Observe that the algorithm returns the kNNs of *every* point of the line segment; therefore the output may contain more than k objects. Continuous-NN is illustrated in Figure 2.1a, where se is a line segment and $p_{1...4}$ are objects. First, p_1 is processed and becomes the NN of the entire line segment. (s, sp_1) and (e, ep_1) are the vicinity circles of s and e. Any object outside the vicinity circles cannot be closer to se than p_1; for this reason, object p_2 is ignored. Next (Figure 2.1b), p_3 is processed and since it is inside circle (e, ep_1), they calculate the point s_1 where the perpendicular bisector $\perp p_1p_3$ intersects se. Therefore, the NN of ss_1 is p_1 while the NN of s_1e is p_3. The new vicinity circles around s, s_1 and e are calculated and p_4 is ignored since it lies outside them. Continuous-NN can also be used with an R-Tree [3]: if an intermediate entry E does not intersect any vicinity circle, the entire subtree of E can be pruned.

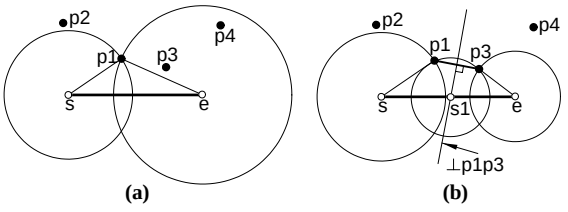


Figure 2.1: The Continuous-NN algorithm: (a) After processing p_1. (b) After processing p_3

The previous method is extended to handle rectangular regions in Ref. [7]. The kNNs of a region $\mathbb{R}$ include all the objects inside $\mathbb{R}$ plus the kNNs of each of the four sides. The authors employ the Continuous-NN algorithm to compute the kNNs of the sides.

3. ANONYMIZER

The anonymizer is a trusted server which acts as a mediator between the users and the location server. Users initiate secure connections with the anonymizer, and periodically report their locations. When a user u needs to access the LS, he sends the query to the anonymizer. We investigate two types of queries: (i) the range query $Q_\varepsilon(u, \varepsilon)$ which returns the objects within distance ε of the current location of u and (ii) the kNN query $Q_{kNN}(u, k)$ which returns the k nearest objects to the location of u. At the anonymizer, the query is transformed in order to satisfy the anonymization criterion, and is forwarded to the LS. The results from the LS are a superset of the user's answer. The anonymizer extracts the exact answer, and returns it to the user.

A user u who issues a location-dependent query is considered to be $\mathcal{K}$-anonymous if his location is indistinguishable from the location of $\mathcal{K}-1$ other users [6]. Formally:

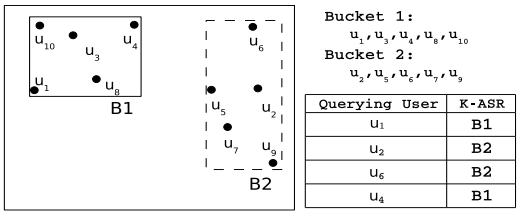
DEFINITION 1. *Let H be a set of $\mathcal{K}$ distinct user entities with locations enclosed in the arbitrary Anonymization Spatial Region $\mathcal{K}$-ASR. Then a user $u \in H$ is said to possess $\mathcal{K}$-anonymity with respect to $\mathcal{K}$-ASR if the probability P of distinguishing u among the other users in H does not exceed $1/\mathcal{K}$. We refer to $\mathcal{K}$ as the required degree of anonymity.*

Although the definition of $\mathcal{K}$-anonymity applies to any shape of the $\mathcal{K}$-ASR, it is likely that only convex regular shapes will present practical interest. We currently consider two different types of enclosing $\mathcal{K}$-ASR: minimum bounding rectangles $\mathbb{R}$ and minimum bounding circles $\mathbb{C}$. The shape affects the area and perimeter of the $\mathcal{K}$-ASR and consequently (i) the processing cost (CPU and I/O) at the LS, (ii) the number of results returned by the LS and thus, the required bandwidth between the LS and the anonymizer and (iii) the CPU cost at the anonymizer due to post-processing of the results. Usually, a rectangular shape $\mathbb{R}$ is preferred, since it minimizes all the above costs. On the other hand, there are cases where a circular shape $\mathbb{C}$ minimizes the area and perimeter of the $\mathcal{K}$-ASR (refer to the SS-Tree paper [19] for a detailed analysis). We show in our experiments that there is a tradeoff of using $\mathbb{C}$: while the bandwidth (between the LS and the anonymizer) and the post-processing cost at the anonymizer decrease, the CPU cost at the LS increases.

The degree of anonymity increases with $\mathcal{K}$; on the other hand, the size of the resulting $\mathcal{K}$-ASR, hence the processing cost, will also increase with $\mathcal{K}$. This is because the $\mathcal{K}$-ASR must be expanded to enclose $\mathcal{K}$ users. Given a target degree of anonymity $\mathcal{K}$, an ideal $\mathcal{K}$-ASR construction technique should minimize the area and perimeter of the $\mathcal{K}$-ASR.

We stress that an *ideal* anonymization technique that is able to guarantee the anonymity of the query source, must employ grouping of users into $\mathcal{K}$-ASRs with fixed user composition. The fixed composition of $\mathcal{K}$-ASR is a *sufficient* condition to guarantee $\mathcal{K}$-anonymity, since the group of users contained in the $\mathcal{K}$-ASR is perceived from the outside as a single entity, with no possibility to distinguish between individual users. Anonymization techniques that do not employ $\mathcal{K}$-ASRs with fixed user composition cannot offer provable anonymity guarantees for all possible user location distributions, but they can still provide strong anonymization features. In Section 3.3 we introduce a *location $\mathcal{K}$-anonymity*, a meaningful criterion for evaluating anonymity of techniques that do not employ $\mathcal{K}$-ASRs with fixed user composition.

An optimal $\mathcal{K}$-ASR construction algorithm would partition the user population into *static* buckets of $\mathcal{K}$ users, such that the sizes of the associated $\mathcal{K}$-ASRs are minimized. Every query from users that reside in the same bucket will have the same associated $\mathcal{K}$-ASR, therefore no matter which user from the bucket generates a query, he cannot be distinguished from others in the same bucket (i.e., the probability of a specific user issuing a query is $1/\mathcal{K}$). Figure 3.1 shows an example. Regardless of which user in bucket B_1 issues a query, the $\mathcal{K}$-ASR which contains the users of bucket B_1 will



Querying User	K-ASR
u_1	B1
u_2	B2
u_6	B2
u_4	B1

Figure 3.1: Anonymization with static $\mathcal{K}$-ASR, $\mathcal{K}$=5

be sent to the LS; the same holds for bucket B_2. Even if the frequency of queries varies for different users, an attacker will not be able to detect which user issued a particular query.

There are three drawbacks with this approach: (i) Calculating the optimal partitioning is an NP-Hard problem [8]. (ii) The partitioning is based on a fixed $\mathcal{K}$. If the degree of anonymization required by a user is larger that $\mathcal{K}$, the request cannot be satisfied. A possible work-around would be to choose a large enough value of $\mathcal{K}$ to satisfy the most demanding user. However, this would increase the average size of the $\mathcal{K}$-ASRs, thus affecting severely the processing cost. Finally (iii) our setting assumes mobile users, therefore static partitioning is not applicable.

In the following, we consider methods for on-the-fly construction of $\mathcal{K}$-ASR. First, we explain the drawbacks of existing approaches. Then, we present two novel $\mathcal{K}$-ASR construction methods: (i) *hilbASR* is based on the Hilbert space filling curve and guarantees that the probability of identifying a user is less than $1/\mathcal{K}$ and (ii) *nnASR* which improves significantly the processing cost by relaxing the anonymization requirement.

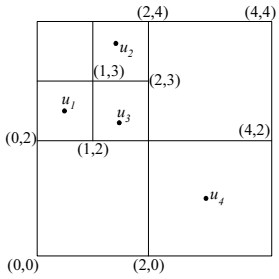
3.1 Drawbacks of Existing Approaches

Current spatial anonymization techniques [6] attempt to build the $\mathcal{K}$-ASR of a given location using the Quadtree [13] space-partitioning technique[4]. When user u issues a query, the Quadtree is traversed until a quadrant which contains u and less than $\mathcal{K}-1$ other users is found. The parent of that quadrant is returned as the $\mathcal{K}$-ASR. We refer to this technique as quadASR.

However, this technique fails to meet the anonymization criterion for some data distributions. Consider the example of Figure 3.2. The data space range is $(0,4)\times(0,4)$. Since a PR-Quadtree is used, quadrants corresponding to nodes with equal depths have equal sizes. Each point resides in its own quadrant, and quadrants are identified by lower-left and upper-right point coordinates: $q_i = ((x_{ll}, y_{ll}), (x_{ur}, y_{ur}))$. When any of the users u_1, u_2 or u_3 issues a query with degree of anonymity $\mathcal{K}$= 3, the quadrant $q_2 = ((0,2),(2,4))$ which encloses $u_{1\ldots3}$ will be returned as the $\mathcal{K}$-ASR. On the other hand, when the isolated user u_4 issues a query with $\mathcal{K}$= 3, the larger quadrant $q_1 = ((0,0),(4,4))$ is returned, which includes all points. Note that if $1 < \mathcal{K} \leq 3$, the only reason to return quadrant q_1 is that u_4 issued a query. If an attacker knows the locations of all users, he will be able to pinpoint u_4 as the query origin.

A second drawback of this technique is that due to the non-uniform distribution of user locations, the number of

[4]Although [6] use the Point-Region (PR) Quadtree, the discussion applies to other versions of the Quadtree.



Non-empty quadrants

Quadrant	Extent	Users
q_1	(0,0),(4,4)	u_1, u_2, u_3, u_4
q_2	(0,2),(2,4)	u_1, u_2, u_3
q_3	(2,0),(4,2)	u_4
q_4	(0,2),(1,3)	u_1
q_5	(1,3),(2,4)	u_2
q_6	(1,2),(2,3)	u_3

Figure 3.2: Limitations of quadASR, $\mathcal{K}$=3

users enclosed by a $\mathcal{K}$-ASR may grow much larger than $\mathcal{K}$. This corresponds to larger spatial extend of the $\mathcal{K}$-ASR hence higher processing cost.

Observe that the same problems also exist if, instead of the Quadtree, we use data partitioning spatial indices such as R-Trees. Next, we propose two methods for constructing the $\mathcal{K}$-ASR, hilbASR and nnASR, which overcome the above drawbacks.

3.2 The hilbASR Anonymization Algorithm

We construct Hilbert $\mathcal{K}$-ASRs by grouping users together into $\mathcal{K}$-sized buckets based on the Hilbert [4] ordering of user locations. The hilbASR algorithm *guarantees* that the probability of identifying a user in a bucket is always bounded by $1/\mathcal{K}$, since it uses a fixed bucket paritioning scheme.

The Hilbert space filling curve transforms the 2D coordinates of each user into an 1D value $\mathcal{H}(u)$. With high probability, if two points are in close proximity in the 2D space, they will also be close in the 1D transformation. We sort the Hilbert values of all users and split them in buckets. Each Hilbert bucket has the same number of users[5], $\mathcal{K}$, except for the last one which is enlarged in order to contain at least $\mathcal{K}$ users.

When user u issues a query Q, we calculate the Hilbert value $\mathcal{H}(u)$ and determine $rank_u$ which is the position of $\mathcal{H}(u)$ in the sorted sequence of all user locations. We associate the query Q with the $\mathcal{K}$-bucket that starts at the index $start = rank_u - (rank_u \mod \mathcal{K})$. Figure 3.3 illustrates an example of determining the buckets for $\mathcal{K}$= 3 and $\mathcal{K}$= 4. For $\mathcal{K}$= 3 for instance, the same bucket B_2^3, is associated to all queries originating at users with Hilbert values 35, 47 or 52. Similarly, for $\mathcal{K}$= 4, bucket B_1^4 is associated to all queries originating at 11, 23, 27 and 35.

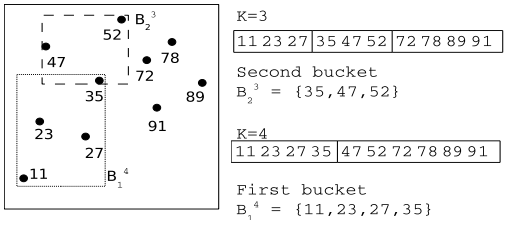


Figure 3.3: hilbASR, $\mathcal{K}$=3 and $\mathcal{K}$=4

As mentioned earlier, $\mathcal{K}$-ASR construction techniques that use fixed $\mathcal{K}$-buckets suffer from lack of flexibility in accom-

[5]Note that even if the Hilbert bucket contains $\mathcal{K}$ users, the corresponding bounding rectangle $\mathbb{R}$ or circle $\mathbb{C}$ may enclose more than $\mathcal{K}$ users.

modating queries with varying anonymization requirements. Our algorithm overcomes this limitation by not physically storing the $\mathcal{K}$-buckets. Instead, a balanced sorting tree is maintained, whose key corresponds to the Hilbert values of users locations. The *start* and *end* indices for a query originating at user u with degree of anonymization $\mathcal{K}_u$ are determined by performing a search for u and computing the bucket boundaries as:

$$start = rank_u - (rank_u \mod \mathcal{K}_u)$$
$$end = start + \mathcal{K}_u - 1$$

For the last bucket, the *start* value is adjusted to $start - \mathcal{K}_u$.

To support efficient range search between two given positions in the tree, we use an annotated tree, where each node stores the number of nodes in its left subtree; the idea is similar to the aR-Tree [11]. The data structure we use is scalable, since search, insert and delete operation all exhibit $O(logN)$ complexity, where N is the number of indexed users; therefore, our technique is applicable to large numbers of mobile users who update their position frequently. The required degree of anonymity $\mathcal{K}$ may vary for different queries, therefore hilbASR can seamlessly accommodate users with diverse anonymization requirements. Note that, while our previous discussion assumes a main memory index, the technique can be easily extended to secondary memory by using B^+-trees.

3.3 The nnASR Anonymization Algorithm

The previous algorithm, hilbASR, guarantees that the anonymization criterion is satisfied. However, the Hilbert transformation may exhibit poor locality for those locations corresponding to the Hilbert curve "turning" points. Therefore the area and perimeter of the corresponding $\mathcal{K}$-ASR is larger, leading to an increase in processing cost.

Recall that finding the optimal $\mathcal{K}$-ASR is NP-Hard. Here, we attempt to minimize the average size of the $\mathcal{K}$-ASR by considering the users which are close to the querying user u. A naïve technique works as follows: Find the $\mathcal{K}-1$ users closest to u and let the $\mathcal{K}$-ASR be the bounding rectangle $\mathbb{R}$ or circle $\mathbb{C}$ which encloses u and his $\mathcal{K}-1$ neighbors. Nevertheless, this naïve $\mathcal{K}$-ASR construction method is likely to disclose the location of u.

Let $index_u$ be the the index of u in the sequence of users enclosed by the $\mathcal{K}$-ASR sorted in ascending order according to distance from the center of $\mathcal{K}$-ASR; for example, if $index_u = 1$ then u is the closest user to the $\mathcal{K}$-ASR center. We propose the following anonymity metric for $\mathcal{K}$-ASR-generation techniques:

DEFINITION 2 (LOCATION $\mathcal{K}$-ANONYMITY). *We consider the location of u to be indistinguishable from the location of the other users in $\mathcal{K}$-ASR if the probability*

$$P[index_u = i] \leq \frac{1}{\mathcal{K}}, \forall i \in \{1, \ldots, \mathcal{K}\} \quad (1)$$

Given an $\mathcal{K}$-ASR and the location of user u within the $\mathcal{K}$-ASR, there are a variety of methods to characterize indistinguishability, such as geometrical simmilarity, pattern matching, etc. However, we believe that our proposed distance-centric *location $\mathcal{K}$-anonymity* is particularly representative for the area of application of LBS.

In Figure 3.4 we show the distribution of the positions inside $\mathcal{K}$-ASR of the querying user u, if we use the naïve method discussed above (for details of the experimental setting, refer to Section 5). For an overwhelming number of cases, u is closest to the center of $\mathcal{K}$-ASR; hence an attacker may easily pinpoint u. The dashed line in Figure 3.4 corresponds to the "flat" index distribution obtained by an ideal anonymization technique, which would always generate $\mathcal{K}$-ASRs with exactly $\mathcal{K}$ users, and the probability of the querying user having index i is exactly $1/\mathcal{K}, \forall i = 1, 2, .., \mathcal{K}$. Note that the probability of the index i being larger than $\mathcal{K}$ is non-zero for the naïve nearest-neighbor method, since the generated $\mathcal{K}$-ASR may contain more than $\mathcal{K}$ users.

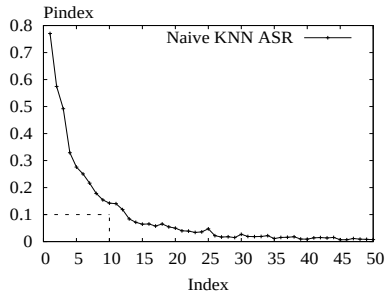


Figure 3.4: Distance from bounding box center measured as $P[index_u = i]$ for the naïve method, $\mathcal{K}$= 10. The points above the dashed line violate Equation 1

Here, we propose a randomized version of the naïve method, called *nnASR*. Our algorithm exhibits the good locality properties of the naïve method, without its drawbacks. nnASR works as follows: Given the location of user u, it first determines the set S containing u and the users who are the $\mathcal{K}-1$ nearest neighbors of u. From S, the algorithm selects a random u'; therefore the probability of selecting the initial user u is $1/\mathcal{K}$. Then nnASR computes the set S' which contains u' and the users who are the $\mathcal{K}-1$ nearest neighbors of u'. Next, we obtain $S'' = S' \cup \{u\}$. This step is essential, since u is not necessarily a $\mathcal{K}-1$ nearest neighbor of u'. Finally, the $\mathcal{K}$-ASR is the bounding rectangle $\mathbb{R}$ or circle $\mathbb{C}$ which encloses the users of S''. Figure 3.5 shows the pseudocode for the algorithm.

nnASR(u, $\mathcal{K}$)
1. $S := \{u\} \cup \{u_i | u_i$ is one of the $\mathcal{K}-1$ NN of $u\}$
2. Randomly choose $u' \in S$
3. $S' := \{u'\} \cup \{u_j | u_j$ is one of the $\mathcal{K}-1$ NN of $u'\}$
4. $S'' := S' \cup \{u\}$
5. **return** $\mathcal{K}$-ASR of S''

Figure 3.5: The nnASR algorithm

We employ the Conceptual Partitioning (*CP*) algorithm [10] to compute efficiently the sets S and S'. CP uses a regular grid to partition the space around u and then follows a branch-and-bound approach to find the nearest neighbors of u. Intuitively, CP is similar to a 2D plain sweep, where the sweep line is a circle centered at u.

The advantage of nnASR is that the average size of the $\mathcal{K}$-ASR is smaller that other anonymization techniques, thus reducing the cost at both the anonymizer and the LS. On the other hand, nnASR does not compute fixed $\mathcal{K}$-buckets, meaning that if user u' is included in the $\mathcal{K}$-ASR of a query issued by user u, u will not necessarily be included in the $\mathcal{K}$-ASR generated for u'. This may cause anonymity problems in the presence of outliers. Figure 3.6 captures such a

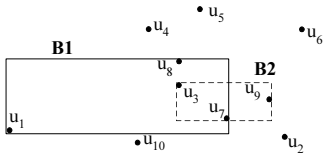


Figure 3.6: nnASR behavior in the presence of outliers, $\mathcal{K}$=3

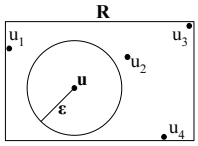


Figure 3.7: Formulation of range queries at the anonymizer side

scenario. When user u_1 issues a query with anonymization degree $\mathcal{K}$= 3, the algorithm will determine its two nearest neighbors, u_3 and u_{10}. Suppose u_3 is chosen at random from u_1's neighbors, and its two nearest neighbors u_7 and u_8 are found. The resulting $\mathcal{K}$-ASR B_1 will include u_1, u_3, u_7 and u_8. However, if a query with $\mathcal{K}$= 3 originates at any of the users other than u_1, the resulting $\mathcal{K}$-ASR will only contain a set of the clustered locations of users $u_2...u_{10}$. Obviously, it is easy for an attacker to identify the queries originating at u_1, as the resulting $\mathcal{K}$-ASR will be much larger than all other $\mathcal{K}$-ASRs, regardless of which of u_1's neighbor is chosen at random.

Summarizing, the Quadtree based algorithm (Section 3.1) exhibits high processing cost and may fail to provide sufficient anonymity for outlier users. nnASR has the same anonymization problem for outliers, but it generates much smaller $\mathcal{K}$-ASRs, therefore decreasing significantly the query processing cost. Finally hilbASR generates larger $\mathcal{K}$-ASRs than nnASR, but it is likely to generate smaller $\mathcal{K}$-ASRs than quadASR. However, hilbASR guarantees that the anonymization criterion is satisfied in all cases.

3.4 Query Formulation

So far, we discussed how to generate the anonymizing region $\mathcal{K}$-ASR. This process is orthogonal to the query type. After the anonymizer selects the $\mathcal{K}$-ASR, the query is forwarded to the location server. Recall that we support two types of queries:

1. The range query $Q_\varepsilon(u, \varepsilon)$ returns the objects (e.g., points of interest from the LS) within distance ε of the current location of u. For this query, the anonymizer sends to the LS the $\mathcal{K}$-ASR and the range ε. The LS returns all the objects within the $\mathcal{K}$-ASR and those which are within range ε from the boundaries of the $\mathcal{K}$-ASR. Then the anonymizer performs a simple filtering and returns to the user the exact answer.

2. The kNN query $Q_{kNN}(u, k)$, returns the k nearest objects to the location of u. The anonymizer sends to the LS the $\mathcal{K}$-ASR and the parameter k (recall that k should not be confused with the degree of anonymity $\mathcal{K}$). The LS returns the objects which are within the $\mathcal{K}$-ASR and those which are the kNNs of *any* point on the boundary of $\mathcal{K}$-ASR. Then the anonymizer finds the actual kNNs of u by a simple linear scan of the result.

With respect to range queries, we employ an additional optimization that aims at reducing query cost. Instead of querying for a range ε around the $\mathcal{K}$-ASR, we use a smaller radius if part of the original query is enclosed in the $\mathcal{K}$-ASR. Consider the example presented in Figure 3.7: user u is issuing a range query $Q_\varepsilon(u, \varepsilon)$. The $\mathcal{K}$-ASR $\mathbb{R}$ obtained by the anonymization algorithm already includes *entirely* the queried region. Therefore, we can query the LS for $\varepsilon = 0$, i.e., return only objects within the $\mathcal{K}$-ASR. Since the anonymity of a user location is guaranteed by the anonymization algorithm within the boundaries of the $\mathcal{K}$-ASR, no information that may pinpoint user location is disclosed by setting $\varepsilon = 0$. Similarly, if the $\mathcal{K}$-ASR does not completely include the range query, then we will formulate the query with the minimum ε value such that the area requested by the user is included in the query (note that the $\mathcal{K}$-ASR is enlarged in *all* directions to preserve anonymity). This value may be considerably smaller than the user query radius, hence the cost of the query may be significantly reduced.

Recall that the shape of the $\mathcal{K}$-ASR may be an axis-parallel rectangle $\mathbb{R}$ or a circle[6] $\mathbb{C}$. There are several cost models [19] to predict which shape will perform better at the LS. In our implementation, for each query, we select the shape which has the smallest area (SA), since our experiments shown that this metric performs better in most of the cases. Other parameters to consider, would include the perimeter of each shape, the value of k, etc. A detailed study is outside the scope of this paper.

4. LOCATION SERVER

The Location Server (LS) receives the query from the anonymizer, processes it and sends the results back to the anonymizer. In our implementation, the data in the LS are indexed by an R*-Tree [3]; our methods, however, are independent of the index structure. As we mentioned, we are interested in two types of queries:

1. Range queries: The LS receives the query range which is either an axis-parallel rectangle $\mathbb{R}$ or a circle $\mathbb{C}$. Processing is straight-forward; the R-tree is traversed from the root to the leaves and any object inside $\mathbb{R}$ (or $\mathbb{C}$) is returned.

2. kNN queries: This case is more complex, since the LS must find the kNNs of the entire range. The rest of this section describes the $\mathbb{R}k$NN and $\mathbb{C}k$NN algorithms which compute the kNNs of rectangular and circular ranges, respectively.

4.1 $\mathbb{R}k$NN - Rectangular Range kNN

We adopt the algorithm from Ref. [7] to compute the kNNs of an axis-parallel rectangle $\mathbb{R}$; we call this algorithm $\mathbb{R}k$NN. Intuitively, the result set consists of (i) all the objects which are inside $\mathbb{R}$ and (ii) the kNNs of each of the four sides (refer to [7] for the proof).

Figure 4.1 presents an example where the query asks for the 1-NNs of $\mathbb{R}_{abcd}$. p_1 and p_2 are included in the result because they are the 1-NN of side ad and bc, respectively.

[6]To compute the minimum enclosing circle of the users, we use the randomized incremental algorithm of Welzl [18] with expected linear time (to the number of users in the K-ASR).

Observe that side ab has two 1-NNs: The 1-NN of segment as_0 is p_1, while that of s_0b is p_2. This is because the distance $|p_1s_0|$ is equal to $|s_0p_2|$. Similarly, the 1-NNs of ds_1 and s_1c are p_1 and p_2, respectively. The final set of 1-NNs of $\mathbb{R}$ is $\{p_1, p_2\}$; object p_3 is not included because all the points in the perimeter of $\mathbb{R}$ are closer to either p_1 or p_2. Note that if there were any objects inside $\mathbb{R}$, they would also be included in the result.

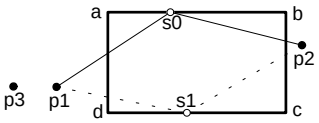


Figure 4.1: The 1-NNs of $\mathbb{R}_{abcd}$ are p_1 and p_2

The $\mathbb{R}k$NN algorithm employs the Continuous-NN algorithm from Ref. [15] (refer to Section 2) to find the kNNs of each of the four sides. A naïve implementation of $\mathbb{R}k$NN requires five traversals of the R-tree: one for each execution of Continuous-NN (for each of the sides) plus one traversal to retrieve the objects inside $\mathbb{R}$. A straight-forward optimization is to combine the five subqueries in a single traversal.

4.2 $\mathbb{C}k$NN - Circular Range kNN

The set of kNNs of a circular range $\mathbb{C}$ also consists of two subsets of objects: (i) all the objects which are inside $\mathbb{C}$ and (ii) the kNNs of the circumference of $\mathbb{C}$. In the example of Figure 4.2, let s_0, s_1 be two points on $\mathbb{C}$, such that the distance $|p_1s_0| = |s_0p_2|$ and $|p_1s_1| = |s_1p_2|$. Assuming that the center c of $\mathbb{C}$ is the origin of the coordinate system, the *polar* coordinates of s_0 are $(r, \hat{s_0})$, where r is the radius of $\mathbb{C}$ and $\hat{s_0}$ is the (anti-clockwise) angle between the x-axis and the vector $c\vec{s}_0$. Similarly, the polar coordinates of s_1 are $(r, \hat{s_1})$. The 1-NN of every point in the arc $[\hat{s_0}, \hat{s_1}]$ is p_1; we denote this as:

$$[\hat{s_0}, \hat{s_1}] \rightarrow p_1$$

Likewise $[2\pi - \hat{s_1}, \hat{s_0}] \rightarrow p_2$, since any point in the arc $[2\pi - \hat{s_1}, \hat{s_0}]$ is closer to p_2 than to any other object. Therefore, the set of 1-NNs of $\mathbb{C}$ is $\{p_1, p_2\}$. Note that p_3 is not in this set, even though it is closer to $\mathbb{C}$ than p_2; this is because p_3 is covered by p_1.

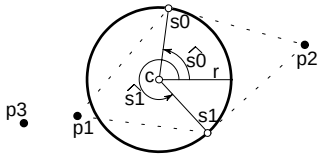


Figure 4.2: The 1-NNs of $\mathbb{C}_{c,r}$ are p_1 and p_2

Here we describe our algorithm, called $\mathbb{C}k$NN-Outer to compute the kNNs of the circumference of $\mathbb{C}$. Conceptually $\mathbb{C}k$NN-Outer is similar to Continuous-NN [15]. However, some of the properties of line segments which are used in Continuous-NN (e.g., continuity, defined in [15]) do not hold for 2-D shapes, rendering the problem more complex.

Let $\mathcal{D} = \{p_0, p_1, \ldots, p_n\}$ be the set of all objects. $\mathbb{C}k$NN-Outer maintains a list SL of mappings $[a, b] \rightarrow p_i$, where a, b are angles defining an arc on $\mathbb{C}$, $0 \leq a < b \leq 2\pi$, and $p_i \in \mathcal{D}$ is the object which is closest to every point of arc $[a, b]$ than any other object $p_j \in \mathcal{D}$. Let $p_1 \in \mathcal{D}$ be the first object encountered by the algorithm (see Figure 4.3a). Since SL is initially empty, p_1 is closest to the entire $\mathbb{C}$. Without loss of generality, we pick two points s_0, s_0', where $\hat{s_0} = 0$ and $\hat{s_0'} = 2\pi$ (i.e., they are the same point), and add the mapping $[\hat{s_0}, \hat{s_0'}] \rightarrow p_1$ in SL.

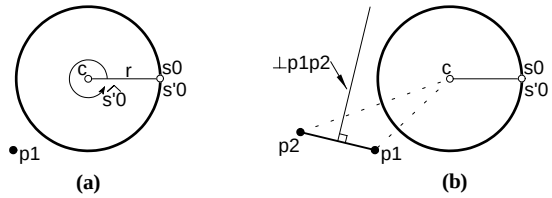


Figure 4.3: $\perp p_1p_2$ does not intersect $\mathbb{C}$

Let p_2 be the next object processed by $\mathbb{C}k$NN-Outer. The algorithm traverses SL and compares p_2 with all existing mappings; in our example, the only mapping is $[\hat{s_0}, \hat{s_0'}] \rightarrow p_1$. Let $\perp p_1p_2$ be the *perpendicular bisector* of line segment p_1p_2. There are two case: (i) $\perp p_1p_2$ does not intersect $\mathbb{C}$ (or is tangent to $\mathbb{C}$) and (ii) $\perp p_1p_2$ intersects $\mathbb{C}$ in two points. Figure 4.3b presents an example of the first case. By definition, any point on the right-hand side of $\perp p_1p_2$, is closer to p_1. Therefore, the entire $\mathbb{C}$ is closer to p_1 than to p_2. Since the mapping to p_1 already exists, there is no change in SL. Furthermore, even if there were more mappings inside SL, it would not be necessary to consider p_2 any further, since p_1 covers p_2. On the other hand, if p_2 was at the right-hand side (and p_1 on the left), then p_2 would be closer to $\mathbb{C}$ than p_1. In this case, the algorithm would remove the $[\hat{s_0}, \hat{s_0'}] \rightarrow p_1$ mapping from SL and add a new one $[\hat{s_0}, \hat{s_0'}] \rightarrow p_2$.

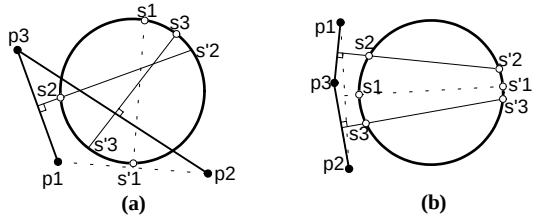


Figure 4.4: $\perp p_1p_2$ intersects $\mathbb{C}$

In the next example (Figure 4.4a), p_1 and p_2 have already been processed. $\perp p_1p_2$ intersects $\mathbb{C}$ in two points, s_1 and s_1'. Since all points on the left of $\perp p_1p_2$ are closer to p_1 and those on the right are closer to p_2, there are two mappings in SL: $[\hat{s_1}, \hat{s_1'}] \rightarrow p_1$ and $[\hat{s_1'}, \hat{s_1}] \rightarrow p_2$. Let p_3 be the next object to be processed. p_3 is compared against the existing mappings. For the first one (i.e., $[\hat{s_1}, \hat{s_1'}] \rightarrow p_1$), $\perp p_1p_3$ intersects $\mathbb{C}$ in s_2 and s_2'. Note that $\hat{s_2'} \notin [\hat{s_1}, \hat{s_1'}]$, so it is not considered further. On the other hand, $\hat{s_2} \in [\hat{s_1}, \hat{s_1'}]$ and p_3 is closer to $\hat{s_1}$ than p_1. Therefore, the arc is split into two parts $[\hat{s_1}, \hat{s_2}]$ and $[\hat{s_2}, \hat{s_1'}]$, which are assigned to p_3 and p_1, respectively. Similarly, for the second mapping (i.e., $[\hat{s_1'}, \hat{s_1}] \rightarrow p_2$), $\perp p_2p_3$ intersects $\mathbb{C}$ in s_3, s_3'. Only $\hat{s_3} \in [\hat{s_1'}, \hat{s_1}]$, so the arc is split into $[\hat{s_1'}, \hat{s_3}]$ and $[\hat{s_3}, \hat{s_1}]$, which are assigned to p_2 and p_3,

```
ℂkNN-Outer(𝒟: the set of objects)
1.  for every object p ∈ 𝒟 do
2.    if SL = ∅ then SL := {[0, 2π] → p}
3.    else
4.      for every interval φ ≡ [a, b] → q, φ ∈ SL do
5.        if ⊥pq ∩ ℂ = ∅ or ⊥pq is tangent to ℂ then
6.          if |pℂ| < |qℂ| then SL := (SL − φ) ∪ {[a, b] → p}
7.          else break
8.        else
9.          let s0, s1 be two points such that ⊥pq ∩ ℂ = {s0, s1}
            // Let ŝ0 < ŝ1 (the other case is symmetric)
10.         if ŝ0 ∈ [a, b] and ŝ1 ∈ [a, b] then
11.           if |pℂa| < |qℂa| then SL := (SL − φ) ∪
                  ∪{[a, ŝ0] → p, [ŝ0, ŝ1] → q, [ŝ1, b] → p}
12.           else SL := (SL − φ) ∪
                  ∪{[a, ŝ0] → q, [ŝ0, ŝ1] → p, [ŝ1, b] → q}
13.         else if ŝ0 ∈ [a, b] or ŝ1 ∈ [a, b] then
              // Let only ŝ0 ∈ [a, b] (ŝ1 ∈ [a, b] is symmetric)
14.           if |pℂa| < |qℂa| then
15.             SL := (SL − φ) ∪ {[a, ŝ0] → p, [ŝ0, b] → q}
16.           else SL := (SL − φ) ∪ {[a, ŝ0] → q, [ŝ0, b] → p}
17.         else if |pℂa| < |qℂa| then
18.           SL := (SL − φ) ∪ {[a, b] → p}
19. return SL

ℂkNN(𝒟: the set of objects)
1.  kNN := {p : p ∈ 𝒟 ∧ p is inside ℂ}
2.  call ℂkNN-Outer(𝒟)
3.  kNN := kNN ∪ {p : p belongs to a mapping of SL}
4.  return kNN
```

Figure 4.5: Find the 1-NNs of a circular range ℂ

respectively. After updating, $SL = \{[\hat{s_2}, \hat{s_1'}] \rightarrow p_1, [\hat{s_1'}, \hat{s_3}] \rightarrow p_2, [\hat{s_3}, \hat{s_1}] \rightarrow p_3, [\hat{s_1}, \hat{s_2}] \rightarrow p_3\}$. The last two mappings can be combined (i.e., $[\hat{s_3}, \hat{s_2}] \rightarrow p_3$) since they are consecutive and are mapped to the same object.

Figure 4.4b presents another example. Again, p_1 and p_2 have already been processed, so $SL = \{[\hat{s_1'}, \hat{s_1}] \rightarrow p_1, [\hat{s_1}, \hat{s_1'}] \rightarrow p_2\}$. Next, p_3 is compared to the first mapping of SL. Note that $\perp p_1p_3$ intersects $\mathbb{C}$ in s_2', s_2 and both $\hat{s_2'}, \hat{s_2} \in [\hat{s_1'}, \hat{s_1}]$. Therefore, the arc is split in three parts and since p_3 is closer to $s'1$ than p_1 the corresponding mappings are: $[\hat{s_1'}, \hat{s_2'}] \rightarrow p_3, [\hat{s_2'}, \hat{s_2}] \rightarrow p_1, [\hat{s_2}, \hat{s_1}] \rightarrow p_3$. Similarly, after considering $\perp p_2p_3$, $[\hat{s_1}, \hat{s_1'}]$ is also split into three parts. Finally, after combining the consecutive mappings, $SL = \{[\hat{s_2'}, \hat{s_2}] \rightarrow p_1, [\hat{s_2}, \hat{s_3}] \rightarrow p_3, [\hat{s_3}, \hat{s_3'}] \rightarrow p_2, [\hat{s_3'}, \hat{s_2'}] \rightarrow p_3\}$.

The $\mathbb{C}k$NN algorithm is shown in Figure 4.5. For simplicity, the presented pseudocode computes only the 1-NNs. To compute the kNNs, instead of a single object, the arcs in our implementation are mapped to an ordered list of k objects: $[a, b] \rightarrow (p_1, \ldots, p_k)$. The procedure is then called for each position i ($1 \leq i \leq k$) of the ordered list. In the ith call, if an object $p \in \mathcal{D}$ already exists in position j ($1 \leq j \leq i-1$), then p is not considered for that mapping. Also, if an arc is split, the objects in positions $1 \ldots i-1$ are not altered. The worst case complexity of $\mathbb{C}k$NN is $O(|\mathcal{D}|^k)$, since any object may cause an arc split. In practice, however, the algorithm is faster, because the objects which are far away from $\mathbb{C}$ do not cause splits.

4.3 R-trees and $\mathbb{C}k$NN

In order to use the $\mathbb{C}k$NN algorithm with an R-tree, we must employ a branch-and-bound heuristic. The process is similar to the Continuous-NN case: starting from the root, the R-tree is traversed either in Depth-First or in Best-First manner. When a leaf entry (i.e., object) p is encountered, the $\mathbb{C}k$NN algorithm is used to check whether p is closer to $\mathbb{C}$ than any of the objects in the current mappings (i.e., p is a *qualifying object*) and updates SL accordingly. For an intermediate entry E we avoid visiting its subtree if it is impossible to contain any qualifying object.

Figure 4.6 presents an example where p_1 and p_2 are the current 1-NNs of $\mathbb{C}$. Next an entry E from an intermediate node of the R-tree is encountered. We observe the following:

LEMMA 1. *Let MBR_E be an axis-parallel MBR and let st be the side which is closest to a circle $\mathbb{C}$. If st does not contain any of the kNNs of $\mathbb{C}$, then the MBR_E cannot contain any kNN.*

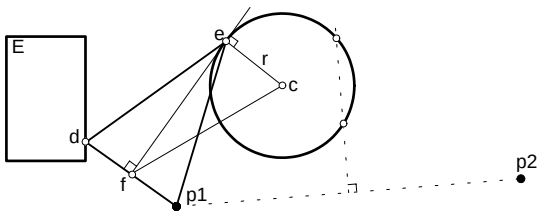


Figure 4.6: Check if E may contain qualifying objects

The proof is straight-forward, since any point in the MBR will be further away from $\mathbb{C}$ than the closest point on st. In our example, the right side st of E is closer to $\mathbb{C}$. Assume that there is a point d on that side, such that the perpendicular bisector $\perp dp_1$ is tangent to $\mathbb{C}$, and let $e = \perp dp_1 \cap \mathbb{C}$. Then we get the following system of equations[7]:

$$\begin{cases} |ce| = r \\ |p_1e| = |de| \\ |p_1e|^2 - |p_1f|^2 = |cf|^2 - r^2 \end{cases} \quad (2)$$

The first equation is derived from the fact that $e \in \mathbb{C}$, while the second one is because the distance from any point on $\perp dp_1$ to d and p_1 is equal. The third equation results from the application of the pythagorean theorem on the orthogonal triangles p_1fe and fec which have a common side ef. After substituting the points with their cartesian coordinates, we get the following system (note that $x_f = \frac{x_d+x_{p1}}{2}, y_f = \frac{y_d+y_{p1}}{2}$, since f is the middle of dp_1):

$$\begin{cases} (x_e - x_c)^2 + (y_e - y_c)^2 = r^2 \\ (x_d - x_e)^2 + (y_d - y_e)^2 = (x_{p1} - x_e)^2 + (y_{p1} - y_e)^2 \\ (x_{p1} - x_e)^2 + (y_{p1} - y_e)^2 - \frac{(x_d - x_{p1})^2 + (y_d - y_{p1})^2}{4} = \\ = \left(\frac{x_d+x_{p1}}{2} - x_c\right)^2 + \left(\frac{y_d+y_{p1}}{2} - y_c\right)^2 - r^2 \end{cases}$$

There are three equations and three unknowns: x_e, y_e, y_d. If there is a real solution of this system, under the condition $(x_d, y_d) \in st$, then there may be a qualifying object inside the subtree of E. Else, if all solutions are imaginary or the condition is not satisfied, all objects in E are further away from $\mathbb{C}$ than the current objects in SL, so the subtree under E can be pruned.

[7]Obviously, if a different side of E is closer to $\mathbb{C}$, the equations are modified accordingly.

Solving this system, however, has two drawbacks: (i) the process is slow (in the order of 100's of msec in an average computer); given that an entry E must be checked against many objects, the running time is prohibitively long and (ii) due to the complexity of the calculations and the possibility of rounding errors, there is a high risk of a false-miss. Therefore, in our implementation, we use the $\mathbb{R}k$NN algorithm to traverse the R-tree and employ the $\mathbb{C}k$NN algorithm only for the objects at the leaf-level. Our strategy is based on the following observation:

LEMMA 2. *Let $\mathbb{C}$ be a circle, MER the maximum enclosed axis-parallel rectangle of $\mathbb{C}$ and S the set of kNNs of MER's perimeter. Let p_i be an object, such that p_i is inside MER and $p_i \notin S$. Then p_i cannot be a kNN for any point of $\mathbb{C}$.*

PROOF. Assume the lemma does not hold. Figure 4.7 shows an example where $p_2 \in MER$ and $p_2 \notin S$. Assume that p_2 is the NN of point $e \in \mathbb{C}$. Let d be the point where the line segment p_2e intersects the perimeter of MER, and p_1 be the object which is the NN of d. Because of our assumption: $|p_2e| < |p_1e|$. Using the triangular inequality, we get: $|p_2d| + |de| < |p_1d| + |de| \Rightarrow |p_2d| < |p_1d|$ which is a contradiction, since p_1 is the NN of d. Therefore, the lemma holds. □

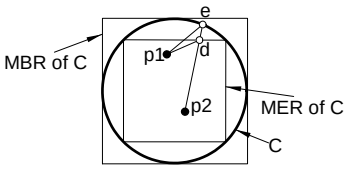


Figure 4.7: The MBR and the MER of $\mathbb{C}$

We construct the Minimum Bounding Rectangle[8] MBR and the Maximum Enclosed Rectangle MER of $\mathbb{C}$ (the side length of MER is $\sqrt{2}r$). Conceptually, our implementation works in three steps:

1. Use the $\mathbb{R}k$NN algorithm to find the set S_1 of kNNs of MBR (including all the objects inside MBR). Recall that S_1 is a superset of the kNNs of any point inside MBR; therefore, it contains all the kNNs of $\mathbb{C}$.

2. Use $\mathbb{R}k$NN to find the set S_2 of kNNs of *only* the perimeter of MER. Use Lemma 2 and S_2 to prune objects from S_1.

3. Call the $\mathbb{C}k$NN algorithm with the objects remaining in S_1.

In practice, these steps can be combined. In a single traversal of the R-tree, steps (1) and (2) can be used at the intermediate levels to prune the tree and step (3) is applied on the leaf-level objects.

[8]Note that for a set of users $u_{1...n}$, the MBR of $\mathbb{C}$ is not the same as their corresponding anonymizing rectangle $\mathbb{R}$.

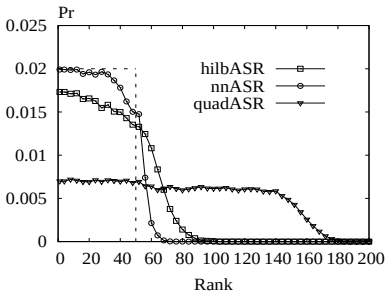


Figure 5.8: Anonymity analysis for hilbASR, nnASR and quadASR, $\mathcal{K} = 50$

5. EXPERIMENTAL EVALUATION

In this section, we present an extensive performance evaluation of our anonymization techniques and the corresponding query processing algorithms. We implemented prototypes for both the anonymizer and the location server. At the anonymyzer side, we used the Conceptual Partitioning code from Ref. [10] to implement nnASR. At the location server, we used the R*-tree and the Continuous-NN code from Ref. [15]. The code is written in C++ and the experiments were run on a Linux cluster consisting of Intel Xeon 2.8GHz machines with 2.5GB of RAM.

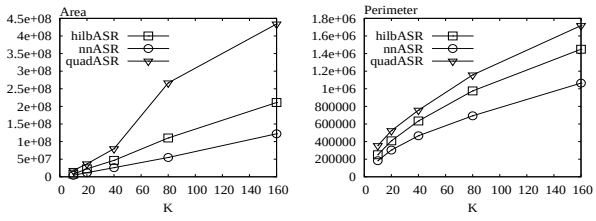
We used two real datasets to evaluate the performance of our system: the ST set and the NA set [17]. The ST dataset represents the public road system in California, and consists of 2.2M road segments. The NA dataset consists of 560,000 locations on the north-American continent. For both sets, the dataspace is a rectangle with coordinates $[0, 10000] \times [0, 10000]$. We used these datasets to generate workloads for both user locations and landmarks/points of interest.

Performance is measured in terms of CPU time, I/O time and communication cost. At the anonymizer we employed main-memory structures, therefore we measure only the CPU time. At the location server, we use an R*-Tree and measure the total time which consists of the I/O and CPU time; in all experiments we used a cache with size equal to 10% of the corresponding R*-Tree. The communication cost is measured in terms of number of results sent back from the LS to the anonymizer.

5.1 Anonymity Analysis

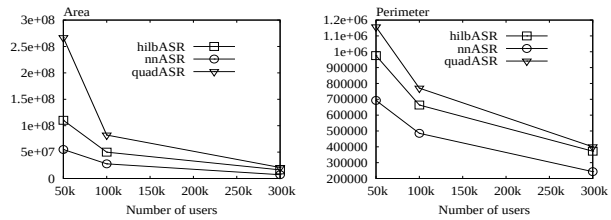
In this section, we evaluate the anonymization capabilities of the three discussed techniques: hilbASR, nnASR and quadASR. We consider a workload of 1000 user queries, originating at a set of users randomly chosen from the ST data set. The requested degree of anonymization is $\mathcal{K} = 50$. As a measure of anonymity strength, we consider the metric introduced in Section 3: the rank of the querying user location in terms of distance from the $\mathcal{K}$-ASR center, compared to the other users in the $\mathcal{K}$-ASR.

Figure 5.8 shows the probability P_r of the querying user to have the rank r in terms of distance from the $\mathcal{K}$-ASR center. Note that P_r can be non-zero for values of r larger than $\mathcal{K} = 50$, since there may be more than $\mathcal{K}$ users in a $\mathcal{K}$-ASR. The dashed line corresponds to the distribution of ranks obtained by the ideal anonymization technique: a "flat" distribution in the interval $[1, \mathcal{K}]$. We observe that nnASR achieves the requested anonymization degree, as P_r never exceeds the threshold value $1/\mathcal{K}$. nnASR is sub-optimal in terms of $\mathcal{K}$-



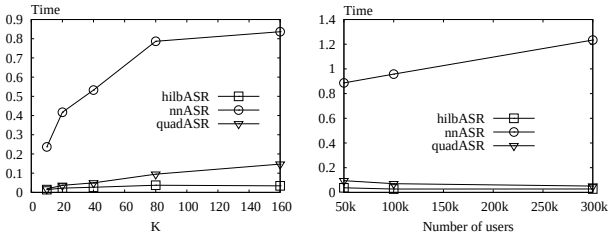
(a) Area (b) Perimeter

Figure 5.9: Area and perimeter for rectangular $\mathcal{K}$-ASR, varying $\mathcal{K}$, 50000 users



(a) Area (b) Perimeter

Figure 5.10: Area and perimeter for rectangular $\mathcal{K}$-ASR, $\mathcal{K} = 80$, varying number of users



(a) Varying $\mathcal{K}$, 50000 users (b) $\mathcal{K} = 80$, varying number of users

Figure 5.11: $\mathcal{K}$-ASR generation time

ASR size, in the sense that the resulting $\mathcal{K}$-ASR may enclose more than $\mathcal{K}$ users, but only by a small margin. However, it is superior to hilbASR, which generates larger $\mathcal{K}$-ASRs; both hilbASR and nnASR are better than quadASR, which generates $\mathcal{K}$-ASRs much larger than those necessary to achieve the required anonymization degree. quadASR is the farthest from the ideal anonymization technique, by a considerable margin compared to hilbASR and nnASR. For this reason, it yields increased $\mathcal{K}$-ASR generation and query processing cost, as we show in the following experiment. We have obtained the same trend for both rectangular and circular $\mathcal{K}$-ASR.

5.2 Anonymizer Evaluation

At the anonymizer site, we consider three different user workload sizes: 50000, 100000 and 300000 users, with locations randomly selected from the ST and NA sets. The chosen user sets are representative for medium-to-large scale systems; the largest size (300000) corresponds to approximately 1% of the population of California. We measure the spatial extent of the $\mathcal{K}$-ASR generated by all three anonymization techniques.

At each run, we generate 1000 queries, having as origin a randomly chosen user. For each query, we determine the corresponding $\mathcal{K}$-ASR and we measure its spatial extent. From space considerations, we present our results only for the ST dataset. The results obtained for the NA set exhibit a similar behavior.

First, we fix the number of users to 50000 and we measure the $\mathcal{K}$-ASR area and perimeter for different degrees of anonymity $\mathcal{K}$. Figure 5.9 shows the measurements obtained for the ST dataset. Both hilbASR and nnASR outperform quadASR in terms of area, by a factor of up to 2.1 and 3.7 respectively. nnASR outperforms hilbASR by a factor of up to 1.8 in terms of $\mathcal{K}$-ASR area. Furthermore, both methods scale well, with the $\mathcal{K}$-ASR area increasing linearly with $\mathcal{K}$. A similar trend can be observed for $\mathcal{K}$-ASR perimeter.

For a fixed value of $\mathcal{K} = 80$, we plot the area and perimeter versus the number of users. Since the size of the dataspace remains constant, an increase in user population translates to higher user density, hence reduced $\mathcal{K}$-ASR size. In Figure 5.10, we observe that hilbASR and nnASR both outperform quadASR for all user cardinalities.

In terms of processing cost (Figure 5.11), hilbASR outperforms both quadASR and nnASR. Since hilbASR is better than quadASR in terms of both $\mathcal{K}$-ASR size and generation time, and furthermore it features stronger anonymization capabilities (hilbASR uses fixed $\mathcal{K}$-ASRs), we conclude that hilbASR is clearly superior to quadASR in all aspects. hilbASR and nnASR provide a clear tradeoff between $\mathcal{K}$-ASR generation cost and $\mathcal{K}$-ASR size: if $\mathcal{K}$-ASR generation latency is of prime importance, then hilbASR is the best choice. On the other hand, if the $\mathcal{K}$-ASR size needs to be reduced, nnASR is preferable. A reduction in $\mathcal{K}$-ASR area translates to both reduced LS query processing time, and a smaller number of query results returned, which saves communication bandwidth.

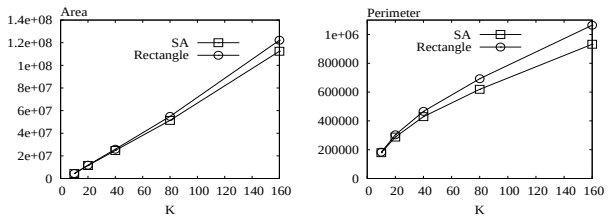
So far, we have focused on rectangular $\mathcal{K}$-ASRs. Now, we consider the case of circular $\mathcal{K}$-ASRs by evaluating the "smaller-area" (SA) optimization method (introduced in Section 3), which for a given query computes both the enclosing rectangle and circle $\mathcal{K}$-ASR and retains the shape with smaller area. Since the nnASR technique exhibits best locality, and is likely to generate "balanced" $\mathcal{K}$-ASRs with equal spatial extent in both dimensions, we expect circular $\mathcal{K}$-ASRs to bring a reduction in $\mathcal{K}$-ASR size only for the case of nnASR. Figure 5.12 shows the reduction in $\mathcal{K}$-ASR size for varying degrees of anonymization $\mathcal{K}$. We observe that SA manages to reduce $\mathcal{K}$-ASR size by a margin of up to 10%.

In the LS performance analysis section, we investigate further the advantages of using circular $\mathcal{K}$-ASR with respect to query processing.

5.3 Location Server Evaluation

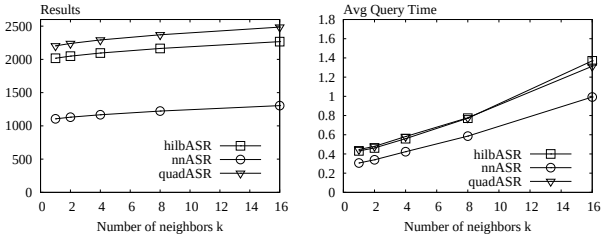
At the LS site, the dataset consists of points of interest corresponding to the entire NA dataset. The query workload of the LS consists of the output of the anonymizer for a user population corresponding to subsets of NA of different sizes. For each $\mathcal{K}$-ASR-generation technique, we consider a workload of 1000 queries, i.e., 1000 $\mathcal{K}$-ASR instances. Due to lack of space, we do not include our results for the ST dataset. However, the trends are similar for both NA and ST datasets.

We compare the average processing time and number of results for all three $\mathcal{K}$-ASR generation techniques. The processing time includes both CPU time and I/O time. The



(a) Area (b) Perimeter

Figure 5.12: Area and perimeter for nnASR, rectangular vs circular $\mathcal{K}$-ASR, varying $\mathcal{K}$, 50000 users



(a) Number of Results (b) Avg. Processing Time

Figure 5.13: kNN queries, varying number of neighbors, 50000 users, $\mathcal{K} = 80$

number of results is a measure of the communication cost incurred by the query.

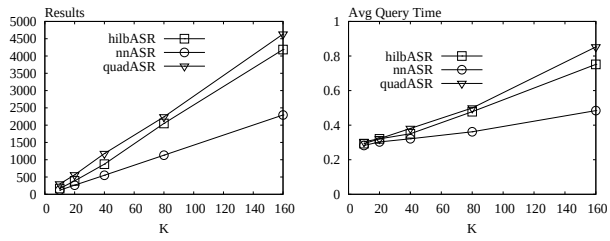
First, we focus on kNN queries. Fig 5.13 shows the number of results and average processing time for varying number of nearest neighbors k. nnASR generates a significantly lower number of results, roughly 50% compared to the other techniques. This is expected, since the size of nnASR $\mathcal{K}$-ASRs is significantly smaller than for hilbASR and quadASR. hilbASR outperforms quadASR in number of results by approximately 10%. In terms of processing time, nnASR incurs 70-75% of the cost of hilbASR and quadASR.

The experimental results for fixed number of neighbors $k = 2$ and varying degree of anonymization $\mathcal{K}$ are presented in Figure 5.14. We notice that nnASR outperforms both hilbASR and quadASR, in terms of both result set size and query processing time. The difference is more significant for larger $\mathcal{K}$ values, as the size of the $\mathcal{K}$-ASR grows.

While, nnASR reduces the load at the LS site, recall from the previous section that, it spends more time at the anonymizer to construct the $\mathcal{K}$-ASR, compared to hilbASR. Hence, there is a clear tradeoff in terms of processing time: if the system load at the LS site is an important concern, then nnASR is the recommended anonymization technique. If, on the other hand, the anonymizer has limited processing capacity, then hilbASR is more appropriate. For varying number of users (Figure 5.15), both the average cost and the number of results *per query* decrease, since the size of the $\mathcal{K}$-ASR decreases.

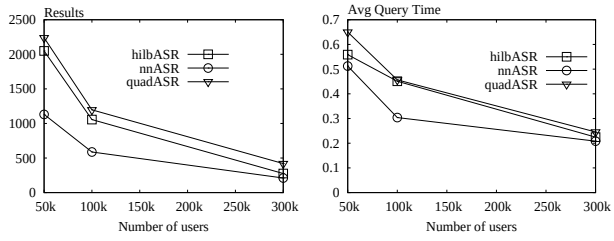
The results for range queries, presented in Figure 5.16, exhibit a similar trend as for NN queries. Again, we observe a significant advantage of nnASR over the other techniques, while hilbASR outperforms quadASR in terms of both processing cost and result set size. A trend similar to the one for NN queries shown in Figure 5.15 was observed for a varying number of users.

We now compare the relative performance of NN queries



(a) Number of Results (b) Avg. Processing Time

Figure 5.14: kNN queries, varying $\mathcal{K}$, $k = 2$ neighbors, 50000 users



(a) Number of Results (b) Avg. Processing Time

Figure 5.15: kNN queries, varying number of users, $k = 2$ neighbors, $\mathcal{K} = 80$

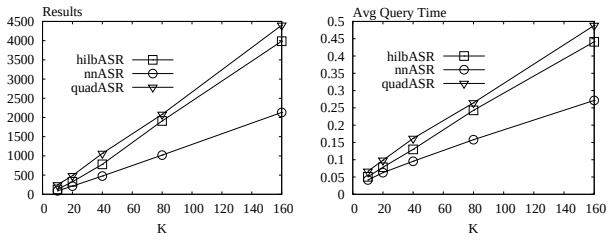
using rectangular and circular nnASR-generated $\mathcal{K}$-ASRs. In Section 4 we introduced the $\mathbb{C}k$NN algorithm that computes the nearest neighbors of a given circular spatial region. We now show that using the $\mathbb{C}k$NN algorithm, we can improve the accuracy of the nearest neighbor search, returning a smaller result set to the anonymizer.

We consider the case of rectangular $\mathcal{K}$-ASR and $\mathcal{K}$-ASR generated using the "smaller-area"(SA) method introduced earlier. Out of the 1000 queries in the SA set, 45% of the $\mathcal{K}$-ASRs for small values of $\mathcal{K}$, and up to 90% for large values of $\mathcal{K}$are circles. Figure 5.17 shows our results for $k = 2$ neighbors. The reduction in number of results for the SA method can be as significant as 10%, compared to the rectangular $\mathcal{K}$-ASR. This reduction comes at the cost of increased processing time of the $\mathbb{C}k$NN algorithm. Therefore, circular $\mathcal{K}$-ASRs are appropriate if minimizing the communication cost is our primary goal; otherwise, if the processing time at the LS is our main concern, rectangular $\mathcal{K}$-ASRs are preferred.

Figure 5.18 shows the performance of query processing for SA-generated $\mathcal{K}$-ASR in comparison with the rectangular $\mathcal{K}$-ASR, for a varying number of users. We observe the same decreasing trend in cost per query as the number of users increases similar to the case of rectangular $\mathcal{K}$-ASR.

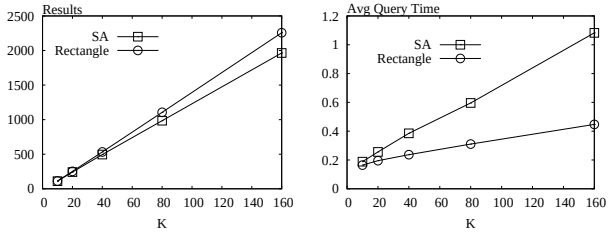
6. CONCLUSIONS

In this paper we propose a framework for preserving anonymity in Location Based Services. The main idea is to conceal the user coordinates, by replacing them with a spatial region (either a circle or a rectangle). This region covers the query initiator and at least $\mathcal{K}-1$ other users, where $\mathcal{K}$ is a user-specified parameter determining the required degree of privacy. We propose methods that construct appropriate anonymization regions, and investigate their tradeoffs. We also design algorithms that run at untrusted location servers, and compute exact answers to Range and Nearest



(a) Number of Results (b) Avg. Processing Time

Figure 5.16: Range queries, 50000 users, varying $\mathcal{K}$



(a) Number of Results (b) Avg. Processing Time

Figure 5.17: nnASR, rectangular vs SA $\mathcal{K}$-ASR, $k = 2$ neighbors, 50000 users, varying $\mathcal{K}$

Neighbor queries given the anonymizing region (instead of the user coordinates). Realistic experiments demonstrate the efficiency of our methods. To the best of our knowledge, this is the first work addressing the entire process of anonymization and query evaluation in Location Based Services.

Our work protects the user's privacy against attacks within reasonable effort. If the attacker has access to excessive resources, the proposed methods will suffer. For example, if the attacker can deploy a large number of decoys near a user u, he may intercept a query from u with high confidence, since the anonymizing range is likely to include the known decoys. For the majority of applications, however, such an attack is impractical.

Our initial findings reveal interesting directions for future research. One such direction is to anonymize multiple user queries in a single region if they are sufficiently close to each other. Another one is to ensure anonymity for users issuing continuous spatial queries. These queries monitor result changes as the user moves, and they have recently attracted considerable research interest. Intuitively, preserving anonymity is harder in this case, because asking the same query from successive locations may disclose the moving habits and the identity of the user. Finally, it would be interesting to investigate methods that do not require an anonymizer. Assuming that the users trust each other, the query initiators could communicate and collaborate with peers in their vicinity to compute their anonymization region.

7. REFERENCES

[1] C. C. Aggarwal. On k-Anonymity and the Curse of Dimensionality. In *Proc. of VLDB*, pages 901–909, 2005.

[2] R. Bayardo and R. Agrawal. Data Privacy through Optimal k-Anonymization. In *Proc. of ICDE*, pages 217–228, 2005.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *Proc. of ACM SIGMOD*, pages 322–331, 1990.

[4] A. R. Butz. Alternative Algorithm for Hilbert's Space-Filling Curve. *IEEE Trans. on Computers*, pages 424–426, April 1971.

[5] B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *Proc. of ICDCS*, pages 620–629, 2005.

[6] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *Proc. of USENIX MobiSys*, 2003.

[7] H. Hu and D. L. Lee. Range Nearest-Neighbor Query. *IEEE TKDE*, 18(1):78–91, 2006.

[8] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient Full-Domain K-Anonymity. In *Proc. of ACM SIGMOD*, pages 49–60, 2005.

[9] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *Proc. of ICDE*, 2006.

[10] K. Mouratidis, D. Papadias, and M. Hadjieleftheriou. Conceptual Partitioning: An Efficient Method for Continuous Nearest Neighbor Monitoring. In *Proc. of ACM SIGMOD*, pages 634–645, 2005.

[11] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP Operations in Spatial Data Warehouses. In *Proc. of SSTD*, pages 443–459, 2001.

[12] P. Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE TKDE*, 13(6):1010–1027, 2001.

[13] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[14] L. Sweene. k-Anonymity: A Model for Protecting Privacy. *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

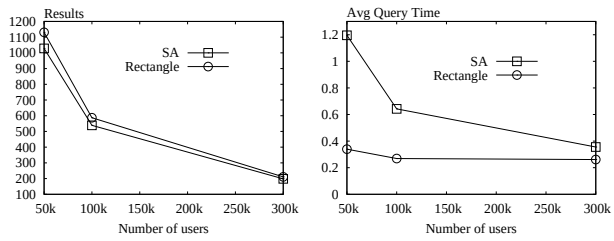
[15] Y. Tao, D. Papadias, and Q. Shen. Continuous Nearest Neighbor Search. In *Proc. of VLDB*, pages 287–298, 2002.

[16] Y. Tao and X. Xiao. Personalized Privacy Preservation. In *Proc. of ACM SIGMOD*, 2006.

[17] Y. Theodoridis. The R-tree-portal, 2003.

[18] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In *In New Results and New Trends in Computer Sciences*, page 359370, 1991.

[19] D. A. White and R. Jain. Similarity Indexing with the SS-tree. In *Proc. of ICDE*, pages 516–523, 1996.



(a) Number of Results (b) Avg. Processing Time

Figure 5.18: nnASR, rectangular vs SA $\mathcal{K}$-ASR, $k = 2$ neighbors, varying number of users, $\mathcal{K} = 80$