AN EFFICIENT FRAMEWORK FOR NETWORK BASED QUERY PROCESSING IN ROAD NETWORKS

Akula S R Lingeswara Rao¹ and K. Rajani Devi²
¹Student, M.Tech (IT), Siddharth Nagar, Nalanda Institute of Engineering and Technology, A.P, India.
²Head of the Department, (IT), Siddharth Nagar, Nalanda Institute of Engineering and Technology, A.P, India.

Abstract - Advances in sensing and tracking technology enable location-based applications but they also create significant privacy risks. Anonymity can provide a high degree of privacy, save service users from dealing with service providers’ privacy policies, and reduce the service providers’ requirements for safeguarding private information. However, guaranteeing anonymous usage of location-based services requires that the precise location information transmitted by a user cannot be easily used to re-identify the subject. This paper presents middleware architecture and algorithms that can be used by a centralized location broker service. The adaptive algorithms adjust the resolution of location information along spatial or temporal dimensions to meet specified anonymity constraints based on the entities that may be using location services within a given area. Using a model based on automotive traffic counts and cartographic material, we estimate the realistically expected spatial resolution for different anonymity constraints. The median resolution generated by our algorithms is 125 meters. Thus, anonymous location-based requests for urban areas would have the same accuracy currently needed for E-911 services; this would provide sufficient resolution for way finding, automated bus routing services and similar location-dependent services.

Keywords – Anonymization, Query Processing.

INTRODUCTION

The low cost and small size of positioning equipment (e.g., GPS receivers) have allowed their embedding into PDAs and mobile phones. The wide availability of these location-aware portable devices has given rise to a flourishing industry of location-based services (LBS). An LBS makes spatial data available to the users through one or more location servers (LS) that index and answer user queries on them. Examples of spatial queries could be “Where is the closest hospital to my current location?” or “Which pharmacies are open within a 1 km radius?” In order for the LS to be able to answer such questions, it needs to know the position of the querying user. When a user u wishes to pose a query, she sends her location to a trusted server, the anonymizer (AZ)[1], through a secure connection (e.g., SSL). The latter obfuscates her location, replacing it with an anonymizing spatial region (ASR) that encloses u. The ASR is then forwarded to the LS. Ignoring where exactly u is, the LS retrieves (and reports to the AZ) a candidate set (CS) that is guaranteed to contain the query results for any possible user location inside the ASR. The AZ receives the CS and reports to u the subset of candidates that corresponds to her original query. In order for the AZ to produce valid ASRs, the users send location updates whenever they move (through their secure connection). The described model is shown in Figure 1.

RELATEDWORK

Section 2.1 reviews related work on road network databases and Section 2.2 surveys the literature on spatial anonymity.

Spatial Query processing in road networks:

In general, a road network can be modeled as a weighted graph \( G = (N,E) \). \( N \) contains the network nodes, while \( E \) is the set of edges. Nodes \( n \) in \( N \) model road intersections, locations of road turns, or positions where traffic conditions change (e.g., a street gets narrower). On the other hand, every edge \( e \) connects two nodes and is associated with a non-negative weight \( w(e) \). Weight \( w(e) \) may represent, for instance, the traveling time from one node to the other. Figure 2 shows an example of a road network. Edge \( n1n2 \) has weight 3, and its endpoints are nodes \( n1 \) and \( n2 \). Let \( p \) be a point on an edge \( e \) with weight \( w(e) \). The partial weight from \( p \) to an end-node of \( e \) is proportional to their (Euclidean) distance, while the sum of the two partial weights is equal to \( w(e) \). For instance, object \( o1 \) (shown as a solid point) lies on edge \( n3n4 \) and has...
The network distance $dN(u; o)$ between a user $u$ and an object $o$ is defined as the sum of edge weights along the shortest path (in the network) from $u$ to $o$. In our example, the network distance $dN(u; o1)$ between user $u$ and object $o1$ equals to 2+1=3. Its derivation is strongly related to shortest path computation. In case of a small network, main memory shortest path algorithms (e.g., Dijkstra’s algorithm) can be applied to compute $dN(u; o)$. Otherwise, disk-based data structures are utilized. Query Processing by Network Expansion. Users are often interested in location-based queries such as range- and kNN queries [7], in the context of a road network. Given a distance threshold $r$ and a user location $u$, the $r$-range query returns all objects within (network) distance $r$ from $u$. On the other hand, the kNN query retrieves the $k$ objects that are closest to $u$. In the rest of the paper, the term distance refers to the network distance, and the $r$-range and kNN queries refer to their network versions (unless otherwise specified). Developed efficient indexing and processing methods for the above queries. They proposed the following disk-based structures for indexing the road network and the data objects: (i) the adjacency index packs adjacency lists of network nodes into disk blocks, (ii) the edge R-tree ORT organizes the locations of the data objects Network expansion is a well-known technique for evaluating $r$-range and kNN queries. Starting from the user location $u$, it discovers objects on encountered edges while traversing the network like Dijkstra’s algorithm, until the query results (i.e., data objects of interest) are found. Suppose that, in Figure 2, user $u$ issues a range query with $r=9$. First, we access the adjacency index to identify edges within the query range, following the steps in Table 1.

### TABLE 1
Steps for range search, $r=9$

<table>
<thead>
<tr>
<th>Step</th>
<th>De-heaped entry</th>
<th>$H$ contents</th>
<th>Found edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>---</td>
<td>$(n_2, 2)$</td>
<td>$n_3n_5$</td>
</tr>
<tr>
<td>2</td>
<td>$(n_2, 2)$</td>
<td>$(n_3, 2)$</td>
<td>$(n_5, 2)$</td>
</tr>
<tr>
<td>3</td>
<td>$(n_3, 2)$</td>
<td>$(n_1, 5)$</td>
<td>$(n_5, 6)$</td>
</tr>
<tr>
<td>4</td>
<td>$(n_1, 5)$</td>
<td>$(n_4, 6)$</td>
<td>$n_1n_2$</td>
</tr>
<tr>
<td>5</td>
<td>$(n_4, 6)$</td>
<td>---</td>
<td>$n_4n_5$</td>
</tr>
</tbody>
</table>

NCOG Location – Based Queries:

Recently, considerable research interest has focused on preventing identity inference in location-based services. Studies in this area typically assume the model described in Section 1, proposing spatial cloaking (i.e., location obfuscation) techniques. In the following, we describe existing techniques for ASR computation (at the AZ) and query processing (at the LS) [8]. At the end, we cover alternative location privacy approaches and discuss why they are inappropriate to our problem setting. Spatial cloaking at the AZ. In general, the AZ applies the concept of K-anonymity to hide the querying user location $u$. The idea is to compute an anonymizing spatial region (ASR), containing $u$ and at least $K-1$ other user locations. This offers privacy protection in the sense that the actual user position $u$ cannot be distinguished from others in the ASR, even when malicious LS is equipped/advanced enough to possess all user locations. This spatial K-anonymity model is most widely used in location privacy research/applications, even though alternative models are emerging. Casper [6] is the first work on efficient and scalable AZ implementation for ASR computation. A quadtree is utilized for indexing user locations and deriving ASRs. Suppose that the AZ needs to compute a 2-anonymous region (i.e., $K=2$) for querying user $u1$ in Figure 3(a). The AZ first locates the leaf quad that contains $u1$ and traverses the tree upwards until it identifies a region covering at least $K$ users (including $u1$). In this case, the AZ derives rectangle $R1:2,3$ (containing three users) as the 2-anonymous region of $u1$.

![Image](a) Casper  (b) Reciprocal cloaking

Figure 3: Spatial K-anonymous cloaking, $K=2$

OBFUSCATION

Obfuscation is the process of degrading the quality of information about a person’s location with the aim of protecting that person’s location privacy. The
term “obfuscation” is introduced, but several closely related concepts have been proposed in previous work. The “need-to-know principle” aims to ensure that individuals release only enough information that a service provider needs to know inorder to provide the required service. The idea of a need-to-know principle is closely related both to obfuscation and the fundamental fair information practice principle of consent and use. Sleekness investigates a privacy policy-based approach to enforcing the need-to-know principle in location aware computing by adjusting precision of location information in work in develops and tests an algorithmic approach to obfuscating proximity queries based on imprecision.

A simplified version of the algorithm introduced in issummarized in. The algorithm assumes a graph-based representation of a geographic environment (for example, a road network). An individual protects his or her location privacy by only reporting a set of locations (an obfuscation set), one of which is that individual’s actual location (figure 3.1a). For an obfuscation set O, the location-based service provider must compute the relation d (figure 3.1b), where od p means o, p < O are most proximal to the same point of interest (POI). The algorithm then proceeds according to three possibilities. First, all the locations in the obfuscation set may be most proximal to a single POI (O < O/d), in which case that POI can be returned to the Second, the individual may agree to reveal a more precise representation of his or her location, in which case the algorithm can reiterate. Otherwise, the best estimate of the most proximal POI is returned.

The analysis in shows that efficient mechanisms for computing the relation d can ensure that the entire algorithm has the same computational (time) complexity as a conventional algorithm for proximity queries, and that the algorithm must terminate in a finite number of iterations. Obfuscation has several important advantages that complement the other privacy protection strategies. Obfuscation and anonymity are similar, in that both strategies attempt to hide data in order to protect privacy.

The crucial difference betweenobfuscationandanonymity isthatwhileanonymityaimsto hide a person’s identity, obfuscation is an explicitly spatial approach to location privacy that aims to allow a person’s identity to be revealed. Potentially, this combats one of the key limitations of anonymity algorithms: the need to authenticate users. At the same time, degrading the quality of location information may make it hard for third-party location providers to determine the location of an individual. Obfuscation is flexible enough to be tailored to specific user requirements and contexts, unlike regulatory strategies; it does not require high levels of infrastructure and is less vulnerable to inadvertent disclosure of personal information, unlike privacy policies; and it is less weighty enough to be used without the need for trusted privacy brokers, unlike many anonymity approaches.

Obfuscation aims to achieve a balance between the level of privacy of personal information and the quality of service of a location-based service. Current research has indicated that there exist many situations where it is possible to expect high quality location-based services based on low quality positional information. Consequently, situations where the user requires high quality of service in a data set where a minimum acceptable level of privacy is chosen, other privacy protection strategies must be relied upon. Further, obfuscation is that the individual, able to choose what information about his or her location to reveal, service provider. Whilenot all service providers for network-based positioning systems, and when sharing location with a third party, location-based service provider, dealing with the entities that administer network-based positioning systems still requires privacy protection based on regulatory or privacy policy [2][3] approaches.

**NETWORK BASED ANONYMIZATION**

In this section, we present the cloaking algorithm [4] of ourNAP framework. Our primary objective is to guarantee reciprocity based anonymity. In NAP, the AZ Anonymizes u with a set of line segments/edges instead of a spatial region (ASR). The crux of our cloaking method is to utilize a global edge ordering; i.e., an ordered sequence that contains all network edges exactly once. The edge ordering is setting-sensitive, i.e., it specifies which end-node of the edge precedes the other. We refer to the position and setting of an edge in the ordering as the edge order and the edge setting, respectively. To avoid confusion, the setting of an edge depends solely on the ordered sequence, and has nothing to do with the direction (in the case of directed networks) of the road segment it models. Figure 5(a) shows a road network, and an ordering of sets. Then, number next to each edge indicates its order and the arrow its setting. The edge ordering defines an implicit linear order among the usersthemsevles. In particular, a user precedes another u if the edge goes from u to v.

If they fall on the same edge, then (with setting from u to v), the next user proceeds.
es usefulto consider. Tie sonogencocincidence of users are resolved arbitraril. This resiidence relationship defines the order of each user. The position of a user in the defined sequence is referred to as the user order. The example in Figure 5(a) contains 10 users whose subscription indicates the order (i.e., user 1 has order 3, etc.). Reciprocity in NAP is achieved by conceptually partitioning the user ordering into buckets of K users each, and forwarding to the bucket of the querying user. This selection is called the Anonymizing Edge List (AEL) of user. Specifically, let U be the set of users registered with the AZ and a user u selects a bucket of size K, where K is the last one which may contain up to 2^K. The bucket of user u is denoted by B_u = K, and the bucket assigned to user v is denoted by B_v. The set of all buckets is denoted by B = \{B_1, B_2, \ldots, B_n\}

Given the edge ordering, the next question is how AEL computation can be implemented efficiently at the AZ. Parameter K is not known in advance and varies, since different users have different anonymity requirements, and even queries by the same user may specify different K, depending on the nature of the query. As a result, AELs are defined according to K, they cannot be explicitly materialized. Instead, the AZ employs an index that keeps the user sort ed on their order and allows efficient AEL computation for arbitrary K. The edge is defined as the aggregate B-tree (similar to an aggregate R-tree), whose internal nodes keep for each child the number of users in the corresponding subtree. Figure 6 shows the tree in the example of Figure 5(a). Foreach user (e.g., u_4), the AEL is a Red-Black tree with a node for each edge (u_2, 7), and its distance from the edge's first node (u_5, u_4). The latter two values are used (primarily) for edge ordering, secondarily for distance from the first node as the sorting key of the tree.

In Figure 6, the numbers in the shaded boxes correspond to the aggregate information maintained, i.e., the cardinalities of the sub-trees rooted at the nodes. Note that we use a B-tree instead of a R-tree (i.e., user information is stored in internal nodes), because it is faster for in-memory indexing.

**Anonymous Query Processing:**

In this section, we describe AEL query processing at the LS; we present algorithms for minimal and inclusive CS computation for asingle query, we propose additional optimizations for the cases where multiple AEL queries are processed in a batch. Wedemonstrate the scalability of NAP with respect to the number of storage nodes used at the LS.

**Single Query Processing:**

Processing consists of three phases: direct, block which uses the theorem (network-based) search algorithms. The shelf building blocks. Thus, the NAP query evaluation methodology is readily deployable existing systems, and can be easily adapted to different environment storage schemes, as we discuss in Section 5.3. As a case study, this section we focus on the storage scheme and the network expansion framework, in order to provide a concrete NAP prototype.

**Batch Query Processing:**

The LS processes queries asdiscrete time stamps, and multiple AEL-based queries may bear on the same timestamp. In this case, the queries are evaluated in batch.
Below we propose strategies aiming at maximizing computation sharing among different queries.

**EXPERIMENTAL EVALUATION**

We evaluate the robustness and scalability of our proposed methods on a real road network [5]. Our algorithms were implemented in C++ and were executed on a Pentium D2, 8GHzPC. We measured the average of the following performance values over a query workload of 100 queries: (i) anonymization time and refinement time at the anonymizer AZ, (ii) I/O time and CPU time for query processing at the location server LS, and (iii) the communication cost (in terms of transmitted points) for the anonymized edge list AEL and the candidate set CS.

**Scalability Experiments:**

In this section, we investigate the scalability of NAP with respect to various factors. To provide an indication of the space requirements, note that for the largest tested data sizes (i.e., $|U| = 2000000$ and $|O| = 1024000$), the AZ uses only 12.5 M Bytes of main memory (including the network graph) and the LS needs a total of 23.5 M Byte shared disk storage. End-to-end-time. Before a lower level study, we present an experiment on the overall response latency.

Specifically, from the user’s viewpoint, the end-to-end time captures the elapsed time between issuing a query and obtaining the results. It includes the processing time at AZ, the computation time at LS, and the communication time between AZ and LS.

![Figure 7: End-to-end time vs. anonymity degree K](image)

Figure 7 shows the end-to-end time as a function of the anonymity degree $K$, assuming a communication bandwidth of 10 Mbps. Clearly, the processing cost at LS dominates the end-to-end time, while the communication (between AZ and LS) and the AZ computation account only for a small percentage of the total time. It is worth mentioning that the processing (including anonymization and refinement) at AZ takes 0.000620 seconds for RE. HN and DF have similar costs. This simplifies that the AZ is capable of serving 1600 requests per second.

**CONCLUSION**

In this paper, we propose the network-based anonymization and processing (NAP) framework, the first system for $K$-anonymous query processing in road networks. NA Priellosa global user ordering and bucketization that satisfies reciprocity and guarantees $K$-anonymity. We identify the ordering characteristics that affect subsequent processing, and qualitatively compare alternatives. Then, we propose query evaluation techniques that exploit these characteristics. In addition to user privacy, NAP achieves low computational and communication costs, and quick responses overall. It is readily deployable, requiring only basic network operations. In the traditional spatial anonymity model, the data owner (e.g., allocation-based service) makes its data available using a location server. It may, however, be the case that the owner is out sourcing its data base to a third-party (and, thus, untrusted) location server. A challenge here is how to encrypt the owner’s data so that they are hidden from the location server, while it can still process anonymous queries. Another interesting question is how (anonymous) users could verify that the location server did not tamper with the original owner data.

**REFERENCES**


