HAL
open science

On Efficient Processing of Complicated Cloaked Region for Location Privacy Aware Nearest-Neighbor Queries

Chan Nam Ngo, Tran Khanh Dang

To cite this version:

Chan Nam Ngo, Tran Khanh Dang. On Efficient Processing of Complicated Cloaked Region for Location Privacy Aware Nearest-Neighbor Queries. 1st International Conference on Information and Communication Technology (ICT-EurAsia), Mar 2013, Yogyakarta, Indonesia. pp.101-110, 10.1007/978-3-642-36818-9_11 . hal-01480224

HAL Id: hal-01480224

https://inria.hal.science/hal-01480224v1

Submitted on 1 Mar 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

On Efficient Processing of Complicated Cloaked Region for Location Privacy Aware Nearest-Neighbor Queries

Chan Nam NGO, Tran Khanh DANG

Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology, VNUHCM, Vietnam

khanh@cse.hcmut.edu.vn

Abstract. The development of location-based services has brought not only conveniences to users' daily life but also great concerns about users' location privacy. Thus, privacy aware query processing that handles cloaked regions has become an important part in preserving user privacy. However, the state-of-the-art private query processors only focus on handling rectangular cloaked regions, while lacking an efficient and scalable algorithm for other complicated cloaked region shapes, such as polygon and circle. Motivated by that issue, we introduce a new location privacy aware nearest-neighbor query processor that provides efficient processing of complicated polygonal and circular cloaked regions, by proposing the Vertices Reduction Paradigm and the Group Execution Agent. In the Vertices Reduction Paradigm, we also provide a new tuning parameter to achieve trade-off between answer optimality and system scalability. Finally, experimental results show that our new query processing algorithm outperforms previous works in terms of both processing time and system scalability.

Keywords: Location-based service, database security and integrity, user privacy, nearest-neighbor query, complicated cloaked region, group execution

1 Introduction

To preserve the LBS user's location privacy, the most trivial method is to remove the direct private information such as identity (e.g., SSID). However, other private information, such as position and time, can also be used to violate the user's location privacy [3]. In preventing that, the researchers have introduced the Location Anonymizer [3]. It acts as a middle layer between the user and the LBS Provider to reduce the location information quality in the LBS request. The quality reduction is performed by the obfuscation algorithm which transforms the location to be more general (i.e., from a point to a set of points [9], a rectilinear region [10-15], or a circular region [6], etc.). The request is then sent to the LBS Provider to process without the provider knowing the user's exact location. Due to the reduction in location quality, the LBS Provider returns the result as a candidate set that contains the exact answer. Later, this candidate set can be filtered by the Location Anonymizer to receive the request's exact answer for the LBS user. Consequently, to be able to process those requests, the LBS Provider's Query Processor must deal with the cloaked region rather than the

exact location. In this paper, we propose a new Privacy Aware Nearest-Neighbor (NN) Query Processor that extends Casper* [8]. Our Query Processor can be embedded inside the untrusted location-based database server [8], or plugged into an untrusted application middleware [3]. The Privacy Aware Query Processor is completely independent of the location-based database server in the LBS Provider and underlying obfuscation algorithms in the Location Anonymizer. Moreover, it also supports various cloaked region shapes, which allows more than one single obfuscation algorithm to be employed in the Location Anonymizer [3]. In addition, we introduce a new tuning parameter to achieve trade-off between candidate set size and query processing time. Finally, we propose an additional component for the Location Anonymizer, the Group Execution Agent, to strongly enhance the whole system's scalability. Our contributions in this paper can be summarized as follows:

- We introduce a new Privacy Aware NN Query Processor. With its Vertices Reduction Paradigm (VRP), complicated polygonal and circular cloaked regions are handled efficiently. In addition, the performance can be tuned through a new parameter to achieve trade-off between candidate set size and query processing time.
- We propose an addition component for the Location Anonymizer, the Group Execution Agent (GEA), to strongly enhance the whole system's scalability.
- We provide experimental evidence that our Privacy Aware Query Processor outperforms previous ones in terms of both processing time and system scalability.

The rest of the paper is organized as follows. In section 2 and 3, we highlight the related works and briefly review the Casper* Privacy Aware Query Processor. The proposed Vertices Reduction Paradigm and Group Execution Agent are discussed in section 4 and 5 respectively. Then we present our extensive experimental evaluations in section 6. Lastly, section 7 will finalize the paper with conclusion and future works.

2 Related Works

In general, Privacy Aware Spatial Data are classified into Public Data (exact location, *public object* such as Point-Of-Interest or public forces' (police officers) location) and Private Data (cloaked region). Based on that classification, a Privacy Aware Query Processor must have the ability to process four types of query [8], includes one exact query type (exact traditional spatial query) and three privacy aware ones: (1) Private Query over Public Data, e.g., "*Where's the nearest restaurant?*", in which the user's location is a *private object* while the restaurant's (answer) is public, (2) Public Query over Private Data, i.e., "*Who have been around this car during 6AM to 8AM?*", the user is a policeman whose location is a *public object*, he is looking for suspects whose locations are *private objects*, (3) Private Query over Private Data, e.g., "*Where's my nearest friends?*", the user is asking for their nearest friends in friends finder service, in which both locations are *private objects*. Recently, Duckham et al. have proposed an obfuscation algorithm [9] that transforms an exact user location to a *set of points* in a road network based on the concepts of *inaccuracy* and *imprecision*. They also provide a NN query processing algorithm. The idea is that the user will first send the

whole *set of points* to the server, the server will send back a *candidate set of NNs*. Based on that candidate set, the user can either choose to reveal more information in the next request for more accurate result or terminate the process if satisfied with the candidate set of NNs. The other works in [4-5] respectively propose algorithms to deal with *circular* and *rectilinear* cloaked region, those works find the *minimal set of NNs*. In a different approach, Casper* only computes a *superset* of the *minimal set of NNs* that contains the exact NN, in order to achieve trade-off between query processing time and candidate set size for system scalability [8]. In addition, Casper* also supports two more query types: Private and Public Query over Private Data.

Among previous works, only Casper* supports Query over Private Data, while the others either only support Query over Public Data [9] or lack the trade-off for system scalability [4-5]. However, Casper* is only efficient in dealing with rectangular regions. While it can handle polygonal cloaked regions, the application into these cases needs evaluations and modifications. Moreover, in case of systems like OPM [3], the Query Processor must have the ability to deal with various kinds of cloaked region because the system supports more than one single obfuscation algorithm. Motivated by those problems, our proposed Privacy Aware NN Query Processor offers the ability to efficiently handle complicated polygonal and circular cloaked regions with its Vertices Reduction Paradigm and a new tuning parameter for system scalability. Furthermore, we provide an addition component for the Location Anonymizer, the Group Execution Agent, to strongly enhance the whole system's scalability.

3 The Casper* Privacy Aware Query Processor

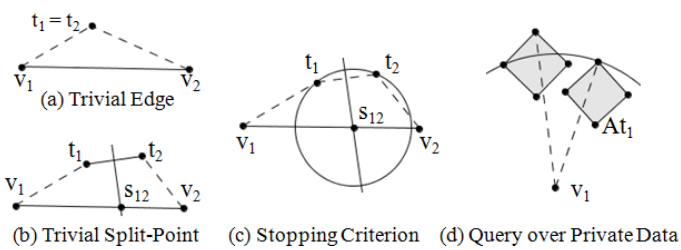


Fig. 1. The Casper* Algorithm

In this section, let us briefly review the Casper* algorithm, start with its terms and notations. For each vertex v_i of the cloaked region A, its NN is called a *filter*, denoted as t_i if that NN is a public object (*public NN*) (Fig. 1b, t_1, t_2 of v_1, v_2). In case the NN is private, it is denoted as At_i. A private object is considered as *private NN* if it has the minimum distance from its cloaked region's furthest corner to v_i (Fig. 1d, At_1). The distance between a vertex and its filter is denoted as $dist(v_i, t_i)$ (public NN) or $min\text{-}max\text{-}dist(v_i, At_i)$ (private NN). For each edge e_{ij} formed by adjacent vertices v_i, v_j, a *split-point* s_{ij} is the intersection point of e_{ij} and the perpendicular bisector of the line segment t_it_j (Fig. 1b, s_{12}). For the purpose of the Casper* NN algorithm [8], given a cloaked region A, it is to find all the NNs of all the points (1) inside A and (2) on its edges. The algorithm can be outlined in the three following steps below.

- **STEP 1** (*Filter Selection*): We find the filters for all of cloaked region A's vertices.
- **STEP 2** (*Range Selection*): For each edge e_{ij} of cloaked region A, by comparing v_i, v_j's filters t_i and t_j, we consider four possibilities to find *candidate NNs* and *range searches* that contain the candidate NNs.
  - ***Trivial Edge Condition***: If $t_i = t_j$ (Fig. 1a, $t_1 = t_2$), t_i (t_j) is the NN to all the points on e_{ij}, so we add t_i (t_j) into the *candidate set*.
  - ***Trivial Split-Point Condition***: In this case, $t_i \neq t_j$, but *split-point* s_{ij} of e_{ij} takes t_i, t_j as its NNs (Fig. 1b). This means t_i and t_j are the NNs to the all points on $v_i s_{ij}$ and $s_{ij} v_j$ respectively. So we add t_i, t_j into the *candidate set*.
  - ***Recursive Refinement Condition***: If two conditions above fail, we will consider to split the edge e_{ij} into $v_i s_{ij}$ and $s_{ij} v_j$, then we apply STEP 2 to them recursively. A parameter *refine* is used to control the recursive calls for each edge, it can be adjusted between 0 and ∞ initially in the system. For each recursive call, *refine* will be decreased by 1, and when it reaches 0, we will stop processing that edge. In this case, $refine > 0$, we decrease it by 1 and process $v_i s_{ij}$ and $s_{ij} v_j$ recursively.
  - ***Stopping Criterion Condition***: When *refine* reaches 0, we add the circle centered at s_{ij} of a radius $dist(s_{ij}, t_i)$ as a range query into the *range queries set R* and stop processing current edge (Fig. 1c).
- **STEP 3** (*Range Search*): we execute all the range queries in R, and add the objects into the *candidate set*. As a result, the *candidate set* contains NNs for all the points (1) inside cloaked region A and (2) on its edges. After that, the *candidate set* will be sent back to the Location Anonymizer to filter the exact NN for the user.

In Query over Private Data, STEP 2 is similar to Query over Public Data, with some modifications. Instead of adding At_i directly into the *candidate set*, we will have to add a circle centered at v_i of a radius $min\text{-}max\text{-}dist(v_i, At_i)$ as a range query into the *range queries set R* (Fig. 1d). The same behavior is applied to v_j and s_{ij} of edge e_{ij}.

4 Vertices Reduction Paradigm

Although Casper* can deal with polygonal cloaked region A that has n vertices (*n-gon*), its runtime significantly depends on A's number of vertices (STEP 1) and edges (STEP 2). As shown in Fig. 2a's formula 1 and 2, to process an *n-gon*, Casper* suffers from two aspects. (1) The processing time of STEP 1 increases because it has to find more filters ($4Qt_4 \leq nQt_n$). Besides, the calculation for $min\text{-}max\text{-}dist(v_i, At_i)$ also increase the range query runtime for the *n-gon* ($Qt_4 \leq Qt_n$). (2) The processing time of STEP 2 increases as it has to process more edges ($4(2^{refine} - 1)Qt_4 \leq n(2^{refine} - 1)Qt_n$).

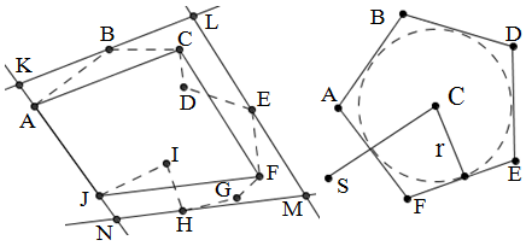
To ease that problem, we introduce the *Vertices Reduction Paradigm* (VRP), in which we simplify the polygon so that it has as less vertices as possible before processing it with the Casper* algorithm. For that purpose, the *Ramer–Douglas–Peucker* (RDP) [1] algorithm is employed. For each private object (*n-gon*) in the database, we maintain a *vertices reduced version* (VRV, *m-gon*, $m < n$) of that private object. The VRV is generated by the RDP algorithm and it will be stored inside the database until invalidated. For NN query processing, we use the VRVs instead of original ones to reduce processing time ($m \leq n$ and $Qt_m \leq Qt_n$, as depicted in formula 3 of Fig. 2a).

The purpose of the RDP [1] algorithm, given an *n-gon* (*ABCDEFGHIJ* in Fig. 2b), is to find a subset of fewer vertices from the *n-gon*'s list of vertices. That subset of vertices forms an *m-gon* that is simpler but similar to the original *n-gon* ($m < n$). The inputs are the *n-gon*'s list of vertices and the *distance dimension* $\varepsilon > 0$. First, we find the vertex that is furthest from the line segment with the first and last vertices as end points. If that furthest vertex is closer than ε to the line segment, any other vertices can be discarded. Otherwise, given the *index* of that vertex, we divide the list of vertices into two: *[1..index]* and *[index..end]*. The two lists are then processed with the algorithm recursively. The output is an *m-gon*'s list of vertices (Fig. 2b, *ACFJ*).

In next subsections, we will discuss two different approaches to utilize the VRVs. The first one is to use the VRV directly. The second one, which is better than the first, is to use the bounding polygon of the VRV. In both approaches, the RDP algorithm's overhead in computing the VRVs is insignificant compared to the total processing time of the query. As depicted in Fig. 2b, the dotted polygon *ABCDEFGHIJ* is the original *n-gon* while *ACFJ* and *KLMN* is the *m-gon* of the first and second approach respectively. For a circular region, the VRV is its bounding polygon (Fig. 2c, *ABDEF*) and we use the distance from another vertex to its center plus the radius as *min-max-dist* of it and that vertex in private NN search (*SC+r* in Fig. 2c).

Rectangle	$4Qt_4 + 4(2^{refine} - 1)Qt_4$	(1)
Polygon	$nQt_n + n(2^{refine} - 1)Qt_n$	(2)
VRP	$mQt_m + m(2^{refine} - 1)Qt_m$	(3)
Number of Vertices (4, m, n)	$4 \le m \le n$	
Range Query Runtime (Qt_4, Qt_m, Qt_n)	$Qt_4 \le Qt_m \le Qt_n$	

(a) Computational Cost



(b) Polygonal VRV (c) Circular VRV

Fig. 2. The Vertices Reduction Paradigm

4.1 The Direct Vertices Reduction Paradigm Approach

In this approach, by using the *m-gons* as the cloaked regions of the query and the private objects, we reduce the query processing time in STEP 1 and STEP 2 of the Casper* algorithm (Fig. 2a, formula 3). However, since we use an approximate version of the original cloaked region, we need some modifications in STEP 2 to search for NNs of the parts of *n-gon* that are outside the *m-gon* (e.g., *ABC* in Fig. 2b). During the RDP's recursive divisions, for each simplified edge, we maintain the distance of the furthest vertex that is not inside the *m-gon* (*A, B, E* and *H* in Fig. 2b). The list's size is exactly *m*. We denote those distances as *d* (Fig. 2b, distance from *H* to *FJ*). The modifications make use of the *distance d* and only apply to the simplified edges that the discarded vertices are not all inside the *m-gon*, e.g. *AC*, *CF* and *FJ* in Fig. 2b.

- **Modifications for Query over Public Data**

- ***Trivial Edge and Split-Point Condition***: using the corresponding *distance d* maintained above, we add two range queries centered at v_i, v_j of radii $dist(v_i, t_i) + d$, $dist(v_j, t_j) + d$ into the *range queries set R* (Fig. 3a). For Trivial Split-Point Condition, we add one more range query centered at s_{ij} of a radius $dist(s_{ij}, t_i) + d$ into *R*. As shown in Fig. 3c, the NN *E'* of any point *H* on *BC* (*C* is a discarded vertex outside the *m-gon*) must be inside the hatched circle centered at *H* of radius *HE (HE' ≤ HE)*, which is always inside the two bold circles created by the enlarged (+*d*) range queries. It is also the same for any points in *ABC*.
- ***Stopping Criterion Condition***: similarly, we increase the range query's radius by *d* to ensure including the NNs of the outside parts of the original *n-gon*.

- **Modifications for Query over Private Data**: because the private objects are also simplified, we will *increase the search radius by d + ε* for not missing them as candidate NNs (depicted in Fig. 3b, the range query *(+d+ ε)* reaches the simplified edge *AB* of another private object while the range query *(+d)* does not).

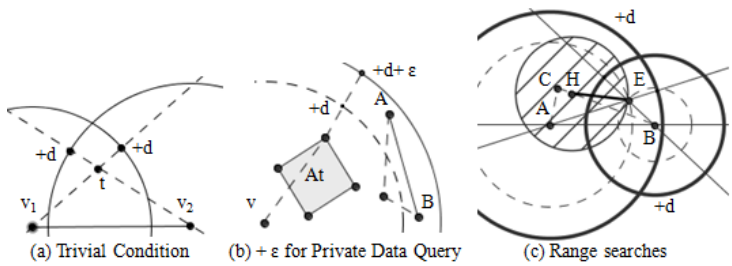


Fig. 3. Modifications in Vertices Reduction Paradigm

4.2 The Bounding Vertices Reduction Paradigm Approach

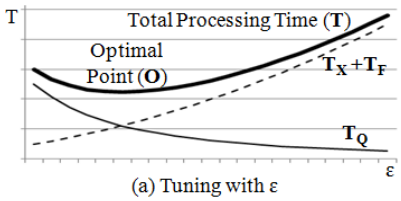
One characteristic of the *m-gon* generated by the RDP algorithm is that it may not contain the original *n-gon*. In this approach, we want to ensure the *m-gon* contains the original one. During the RDP's recursive divisions, for each simplified edge (*m* edges), we maintain the furthest vertex that is not inside the *m-gon* (*A, B, E* and *H* in Fig. 2b). After that, we calculate the *m* lines that are parallel to the respective edges of the *m-gon* and through the respective furthest vertices in the list (e.g., *KL, LM, MN* and *NK* in Fig. 2b). The intersection of those lines forms a new *m-gon* that contains the original *n-gon* inside it (Fig. 2b's *KLMN*). Therefore, the candidate set of the simplified *m-gon* is a superset of the original *n-gon* without directly modifying Casper*.

Although the first approach reduces the query processing time much, it suffers from the moderate increase of the *candidate set* size. Differently, the second approach achieves both better *candidate set* size and query processing time than the first one. Firstly, we can add the filters directly into the *candidate set* without the risk of missing the exact NN because the *m-gon* contains the original *n-gon* (no outside parts). Secondly, although the range query's radius is indirectly enlarged through the enlargement of the original *n-gons* to the bounding *m-gons*, it is kept minimum (+*d*+*d*, an indirect +*d* of the cloaked region and another +*d* of the private object). Thus the

number of results for each range query is also reduced. Furthermore, the reduction in number of range queries also leads to a slight reduction of processing time.

4.3 The Distance Dimension ε as VRP Tuning Parameter

The *Total Processing Time* (T) of a query consists of three components. (1) The *Query Processing Time* (T_Q), which is for the Query Processor to compute the candidate set. (2) The *Data Transmission Time* (T_X) which is for the candidate set to be transmitted to the Location Anonymizer for NNs filtration. (3) The *Answers Filtration Time* (T_F) which is for the candidate set to be filtered for exact NN of the query request. T_Q is monotonically decreasing with the decrease of number of vertices, while T_X and T_F are monotonically decreasing with the decrease of candidate set size. Thus, we can utilize the distance dimension ε as a tuning parameter for VRP since it affects the number of vertices in the VRV and the search radius of range queries in the range query set R. We will consider 2 cases in respect to the ε value: (1) $T_Q > T_X+T_F$. Initially, the ε is too small that query processing takes too much time. To resolve this, we must increase ε. (2) $T_Q < T_X+T_F$. This indicates the candidate set size is too large that (T_X+T_F) is longer than T_Q. We have to decrease ε. Thus, in order to find an optimal value of ε for the best T, we increase ε until it reaches the optimal point O (Fig. 4a).



(a) Tuning with ε

VRP	$N(T_Q + T_X) + NT_F$	(1)
GEA	$K(T_Q + T'_X) + NT'_F$	(2)
Number of Queries (N, K)		$K \leq N$
Average Transimision Time (T_X, T'_X)		$KT'_X \leq NT_X$

(b) Group Execution Agent

Fig. 4. System Scalability

5 Group Execution Agent

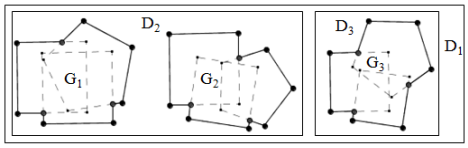


Fig. 5. Group Execution

As shown in Fig. 5, there are many queries with adjacent and overlapped regions at a time (the dotted regions), or even better, a query's region is contained inside another's. Obviously, such queries share a part of or the whole candidate set. To take advantage of that, we propose the Group Execution (GE) algorithm for the Location Anonymizer's additional component, the Group Execution Agent (GEA). The GEA will group as many queries as possible for one query execution before sending them

to the Query Processor (N queries into K query groups, $K \leq N$, i.e., 9 queries to 3 groups in Fig. 5, the bold $G_{1,2,3}$ are used as cloaked regions in NN queries).

Algorithm Group Execution

```
function GroupExecution(list of query regions, system-defined max_A)
while true and list size > 1 do
    for each region r_i, find another region r_j that has least enlargement if r_i is
grouped with r_j, the new region's area a ≤ max_A, and add into list(r_i,r_j,a)
    break if list(r_i,r_j,a) is empty
    sort list(r_i,r_j,a) by a ascending
    for each r_i in list(r_i,r_j,a)
        if r_i already grouped in this loop
            if r_j already grouped in this loop continue
            else groupedRegions = groupedRegions U {r_j}
        else groupedRegions = groupedRegions U GroupedRegionOf(r_i, r_j)
    regions = groupedRegions
return regions (maximum number of regions grouped, minimum area each group)
```

The algorithm is outlined in the pseudo code above. Its purpose, given a *list of query regions* (of size N) and a parameter max_A, is to group the regions in the list into K *grouped regions* ($K \leq N$) of which areas are smaller than max_A. Then the queries are also put into *query groups* accordingly. The *grouped regions* are used as the cloaked regions of those *query groups* in NN query processing. The *query group*'s candidate set is a superset of the candidate set of the queries in the group, so the GEA does not miss any exact NN of those queries. The system benefits from the GEA as shown in Fig. 4b. (1) The query processing time for each *query group* is the same as a single query (T_Q) because we only execute the *query group* once with the *grouped region* as input. Thus, the sum of all queries' processing time decreases ($KT_Q \leq NT_Q$). This also leads to the decrease of average query processing time. (2) The *query group*'s candidate set size increases because it is a superset of the candidate set of those queries in the group, but the average transmission time decreases as we only transmit the common candidates once (as $KT'_X \leq NT_X$). The average filtration time increases ($T'_F \geq T_F$), but it is minor in comparison to the benefits above. Furthermore, for optimization, the algorithm's input list must satisfies two conditions: (1) the list's regions are adjacent to each other for easier grouping, (2) the list size is small enough to avoid scalability problem because the algorithm's complexity is $O(n^2)$. To find those suitable lists, we maintain an R*-Tree [2] in the Location Anonymizer. When a query is sent to the Anonymizer, its cloaked region is indexed by the R*-Tree. By finding the R*-Tree's nodes of which the directory rectangle's area are smaller than a predefined area value $kmax_A$, we will get the suitable lists from those nodes' regions. In Fig. 5, we find two suitable lists from the nodes D_2 and D_3's regions (D_1's area > $kmax_A$). Later, the algorithm returns *grouped regions* G_1, G_2 and G_3, which reduces the number of queries from 9 to 3. In fact, the GEA's speedup is dependent of how much overlapped the regions are. The worst case could be that we cannot group any query but still have the overhead of the R*-Tree and the GE algorithm. However, in most cases, when the number of queries is large enough, the GEA does strongly reduce the system's average query processing and transmission time and improve the system scalability.

6 Experimental Evaluations

We evaluate both two VRP approaches and the GE algorithm for the Private Query over Public and Private Data. The algorithms are evaluated with respect to the tuning parameter ε. For all two types of private query, we compare our algorithms with the Casper*, the performance evaluations are in terms of total processing time and candidate set size. We conduct all experiments with 100K private data and 200K public data. The polygonal cloaked regions are generated by Bob-Tree [14-15], and the circular ones are generated by the works in [6]. The polygonal cloaked regions' number of vertices range from 20 to 30, while the value of ε is varied in the range of 10% to 50% of the Bob-Tree's grid edge size. For GEA, the number of queries is 10K and the parameter max_A and $kmax_A$ are 30 and 100 times of the above grid's area respectively.

The charts in Fig. 6 show our experimental results. As shown in the processing time charts, the VRPs perform significant improvements compared to Casper*. When ε increases, the processing time of VRPs and GEA decrease while Casper*'s remains constant. Because the larger the ε value is, the larger reduction we can achieve, leads to the larger reduction in query processing time, especially in Private Query over Private Data. At the largest ε (50%), the VRPs reduce the total processing time by 98% for Private Query over Private Data (Fig. 6b) with the standard variation at only 92ms (10% of average total processing time). However, the candidate set size increases moderately (Direct VRP) and slightly (Bounding VRP). Lastly, with the additional component GEA, the total processing time and candidate set size are reduced at best (ε = 50%) by 66% and 33% respectively in comparison to Bounding VRP, the best VRP approach. This helps ease the increase in candidate set size of VRP.

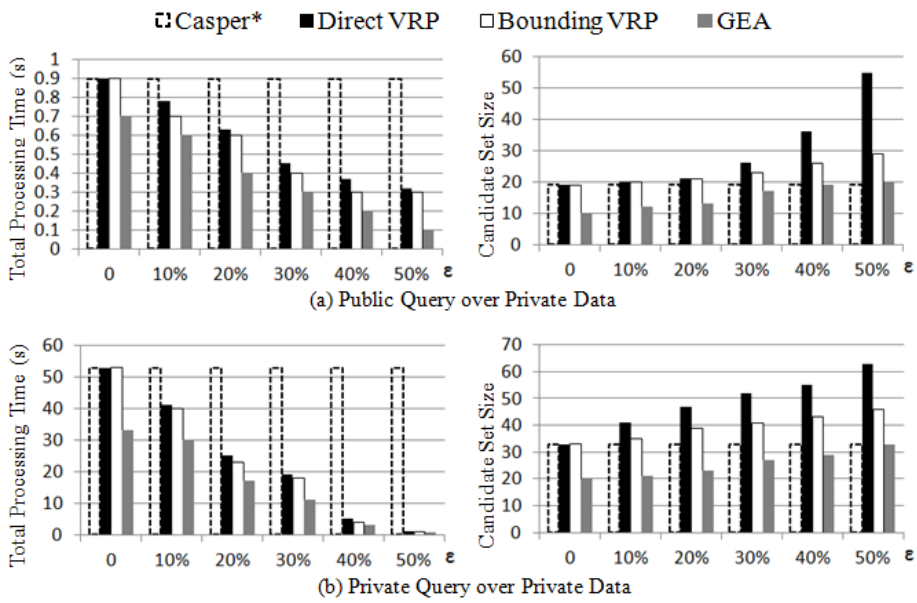


Fig. 6. Experimental Evaluations

7 Conclusion and Future Works

In this paper, we introduce a new Privacy Aware Query Processor that extends Casper*. With the new Query Processor's Vertices Reduction Paradigm and its tuning parameter ε, complicated polygonal [12-15] and circular [6] cloaked regions are handled efficiently. The main idea is that we employ the Ramer–Douglas–Peucker algorithm to simplify the region's polygon before processing it. Furthermore, we propose the Group Execution Agent to strongly enhance the system scalability. Experimental results show that our works outperform Casper* in dealing with such kinds of region above. For future, we will consider supporting k nearest-neighbor query, continuous query [7-8] and trajectory privacy [16] in our Privacy Aware Query Processor.

8 References

1. Douglas, D. H., Peucker, T. K.: Algorithms for the Reduction of The number of Points required to represent a Digitized Line Or its Caricature. In: Cartographica: The Intl. Journal for Geographic Information and Geovisualization, Univ. of Toronto Press (1973) 112-122
2. Beckman, N., Kriegel, H., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: SIGMOD. (1990) 322-331
3. Dang, T. K., Phan, T. N., Ngo, C. N., Ngo, N. N. M.: An open design privacy-enhancing platform supporting location-based applications. In: ICUIMC. (2012) 1-10
4. Hu, H., Lee, D. L.: Range Nearest-Neighbor Query. IEEE TKDE 18(1) (2006) 78-91
5. Kalnis, P., Ghinita, G., Mouratidis, K., Papadias, D.: Preventing Location-Based Identity Inference in Anonymous Spatial Queries. IEEE TKDE 19(12) (2007) 1719-1733
6. Ardagna, C. A., Cremonini, M., Damiani, E., Vimercati, S. D. C., Samarati, P.: Location privacy protection through obfuscation-based techniques. In: DBSec. (2007) 47-60
7. Truong, Q. C., Truong, T. A., Dang, T. K.: The Memorizing Algorithm: Protecting User Privacy in Location-Based Services using Historical Services Information. In: IJMCMC. 2(4) (2010) 65-86
8. Chow, C., Mokbel, M. F., Aref, W. G.: Casper*: Query processing for location services without compromising privacy. ACM TODS (2009) 1-48
9. Duckham, M., Kulik, L.: A formal model of obfuscation and negotiation for location privacy. In: PerCom. (2005) 243-251
10. Dang, T. K., Truong, T. A.: Anonymizing but Deteriorating Location Databases. In: POLIBITS. No. 46 (2012) 73-81
11. Truong, T. A., Dang, T. K., Küng, J.: On Guaranteeing k-Anonymity in Location Databases. In: DEXA. (2011) 280-287
12. Damiani, M. L., Bertino, E., Silvestri, C.: The PROBE Framework for the Personalized Cloaking of Private Locations. In: TDP. 3(2) (2010) 123-148
13. Le, T. B. T., Dang, T. K.: Semantic-aware Obfuscation for Location Privacy at Database Level. In: ICT-EurAsia (2013)
14. To, Q. C., Dang, T. K., Küng, J.: Bob-tree: an efficient B+-tree based index structure for geographic-aware obfuscation. In: ACIIDS. (2011) 109-118
15. To, Q. C., Dang, T. K., Küng, J.: A Hilbert-based Framework for Preserving Privacy in Location-based Services. In: IJIIDS. (2013)
16. Phan, T. N., Dang, T. K.: A Novel Trajectory Privacy-Preserving Future Time Index Structure in Moving Object Databases. In: ICCCI. (2012) 124-134