ISSN 2395-8618

Location Privacy-Aware Nearest-Neighbor Query with Complex Cloaked Regions

Tran Khanh Dang and Chan Nam Ngo

***Abstract*—The development of location-based services has spread over many aspects of modern social life. This development brings not only conveniences to users' daily life but also great concerns about users' location privacy. In such services, location privacy aware query processing that handles cloaked regions is becoming an essential part in preserving user privacy. However, the state-of-the-art cloaked-region-based query processors only focus on handling rectangular regions, while lacking an efficient and scalable algorithm for other complex region shapes. Motivated by that problem, we introduce enhancements and additional components to the location privacy aware nearest-neighbor query processor that provides efficient processing of complex polygonal and circular cloaked regions, namely the Vertices Reduction Paradigm and the Group Execution Agent. We also provide a new tuning parameter to achieve trade-off between answer optimality and system scalability. Experiments show that our query processing algorithm outperforms previous works, in terms of processing time and system scalability.**

***Index Terms*—Complex cloaked region, database security and integrity, group execution, location-based service, location privacy, nearest-neighbor query.**

I. 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 [1]. In preventing that, the researchers have recently introduced the Location Middleware [2]. 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 exact location to be more general (i.e., from a point to a set of points [3], a rectilinear region [4], [5], [6], [7], or a circular region [8], [9], 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 Middleware 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 be able to 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* [10]. Our query processor can be embedded inside the untrusted location-based database server [10], [11], or plugged into an untrusted application middleware [2]. The query processor is completely independent of the location-based database server in the LBS Provider (as long as it supports basic functions such as range query) and underlying obfuscation algorithms in the Location Middleware (as long as the output is a cloaked region of n-gon shape). Moreover, it also supports various cloaked region shapes, which allows more than one single obfuscation algorithm to be employed in the Location Middleware [2]. 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 Middleware, the Group Execution Agent, to strongly enhance the whole system's scalability. The Group Execution Agent will group queries of the same filter parameters (POI type, keywords, etc.) that have adjacent or overlapped cloaked region into one to reduce the number of queries sent to and executed by the LBS Provider.

The contributions in this paper can be summarized as follows:

- We introduce an enhanced Privacy Aware Nearest-Neighbor Query Processor. With its Vertices Reduction Paradigm (VRP), complex polygonal and circular cloaked regions are handled efficiently in reasonable query processing time. 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 additional component for the Location Middleware, the Group Execution Agent (GEA) and its Group Execution (GE) algorithm, to enhance the whole system's scalability. The basic idea of the GEA is that we group several queries of same filter parameters that have adjacent or overlapped cloaked region into one to reduce computational and communicational cost in the system.
- The applications of both Vertices Reduction Paradigm and Group Execution Agent are not limited to the scope of Privacy Aware Nearest-Neighbor Query. In fact, VRP is able to serve as enhancing component for any algorithms that need to process irregular shapes as general polygons and GEA can be applied to scale up systems that process multiple regions concurrently.

Manuscript received on August 20, 2015, accepted for publication on October 2, 2015, published on October 15, 2015.

T. K. Dang and C. N. Ngo are with the Faculty of Computer Science and Engineering, HCMC University of Technology, VNUHCM, Vietnam (e-mail: khanh@cse.hcmut.edu.vn).

ISSN 2395-8618

– We provide theoretical and experimental evidences to prove 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 cloaked-region-based Casper* Privacy Aware Nearest-Neighbor Query Processor. Section 4 is dedicated for the underlying system architecture. In Section 5, we present two versions of the the proposed Vertices Reduction Paradigm. For the sake of clarity of the first version, we also provide proofs of accuracy and running examples. The Group Execution Agent is discussed in Section 6. Then we present our experimental evaluations in Section 7. Lastly, Section 8 will finalize the paper with conclusion and future works.

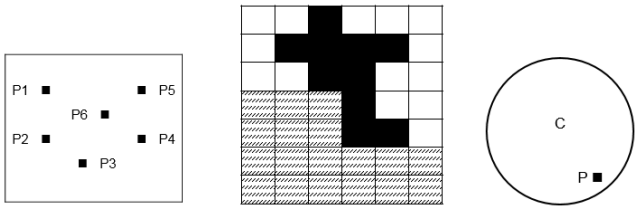
II. Related Work

A. Location Privacy Preservation

To protect user privacy in LBS is to protect the user's location information, includes: identity, position and path. Based on this classification, researchers have proposed algorithms to preserve LBS user's privacy against attackers. The most trivial approach is to remove the information that directly reveals the personal and private information, such as identity (e.g., SSID), but other information, usually location and time, the so-called Quasi-Identifier [1] can also indirectly reveal the user identity. Thus, more complicated and advanced algorithms are introduced for more privacy protection. The algorithms can be classified into four categories as follows:

– Anonymity: to make user indistinguishable among $(k-1)$ other users for user identity protection [2], [12], [5], [7]. However, this category of algorithm has weaknesses where k does not necessarily determine the privacy of users [13]. In anonymous LBS (e.g., navigation), this category of algorithms aims for the protection of identity privacy.
– Obfuscation: to make user location imprecise or inaccurate or generalize the location into a region [8], [4], [3], [6], [14], [15]. Generally, this process is done by the means of perturbation, adding dummies, reducing precision or location hiding. In LBS that require user identity (e.g., paid LBS), this category of algorithms aims at position privacy.
– Transformation: to map user location into another location and repeatedly issue queries to process proximity services [16].
– Encryption: this class of algorithms is based on cryptography, the database server cannot know the data in the query and result [17].

Each privacy algorithm has its own characteristics and application domains. Hence, depending on what kind of objects needing to be protected, privacy algorithms should be chosen carefully and wisely to give the best protection to LBS users. In that manner, [2] proposes a framework that maximize LBS users' privacy level by considering the user and service context factors. Among the algorithms above, obfuscation algorithms have obtained much interest because of their intuitive concepts and simple implementations. Obfuscation algorithms aim at decreasing LBS users' location quality to protect their privacy. To achieve that, from the exact LBS user location (position), the algorithms generate a more general cloaked region, of which the region shape can be very various, from a discrete set of points to a polygonal region or a circular region. Furthermore, the state-of-the-art algorithms deal with not only the geometry of the region, but also its geographic and semantic features. Recently, semantic aware obfuscation algorithm in the PROBE framework [4], or the database level obfuscation algorithm Bob-Tree [6], [15] also consider sensitive features and unreachable region area in the obfuscation process.



(a) Set of Points (b) Polygon (c) Circle

Fig. 1. Cloaked Region Classification

B. Cloaked Region Classification

As discussed in previous subsection, to protect the LBS user's location privacy, various obfuscation algorithms have been proposed in order to reduce the quality of the user's location. This is done by transforming the user's exact location as a point to a more general location as a region. In general, the cloaked regions' shapes can be categorized into 3 different types as follows:

– A set of points: as depicted in Figure 1a, this kind of cloaked regions is proposed by [3], in which the exact location p is transformed to a set of n points {p1, p2, p3, …, pn} based on the concept of imprecision and inaccuracy. In case of inaccuracy, the set of points will not include the original exact location p.
– A polygonal region: this kind of cloaked regions is generated by most of the obfuscation algorithms. The exact location p will be generalized to a polygon represented by an ordered list of points {p1, p2, …, pn, p1}, which are the vertices of the polygon's outer ring. The polygonal region can be as simple as a rectangular region [5], [14], [7] or as complex as an n vertices rectilinear region [4], [6], [15]. For example, in Figure 1b, the complex rectilinear cloaked region (solid black area) is generated by Bob-Tree [15], a database level and geographic aware cloaking algorithm, which considers some parts of the map are unreachable (hatched area)

 http://dx.doi.org/10.17562/PB-52-6

ISSN 2395-8618

such as lakes, mountains, etc. So the cloaked regions will not include the unreachable parts, leads to their indeterminate rectilinear shape.

– A circular region: as depicted in Figure 1c. In the works of [8], the authors propose various obfuscation algorithms to generalize an exact location p to a (c, r) circle, in which c is the center and r is the radius. The radius r then can be reduced, enlarged and the center c can be moved to achieve better obfuscation for LBS user. Or in the works of [9], the authors also provide an obfuscation algorithm to generate a circular cloaked region from the LBS user's exact location with a random method.

C. Location Privacy Aware Nearest-Neighbor Query Processor

To support the privacy aware queries, recent research efforts have been dedicated to deal with cloaked area processing. The location privacy aware query processor is either embedded directly into an untrusted location-based database server or application middleware to process queries from LBS users or a trusted third party (e.g., the Location Middleware). In addition, to reinforce a higher level of user location privacy, the query processor must support four basic types of query [10] as follows:

– Public Query over Public Data.
– Private Query over Public Data.
– Public Query over Private Data.
– Private Query over Private Data.

In these circumstances, public query or data refers to objects or their information that everyone knows or are willing to be revealed (e.g., the position of POIs, the position of patrol police car, ambulance or public services) while private query or data mentions those whose access is limited (e.g., the user's exact position or private places).

Recently, [3] have proposed an obfuscation algorithm 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 nearest-neighbor 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 nearest-neighbors. 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 nearest-neighbors. The other works by [11] and [9] respectively propose algorithms to deal with circular and rectilinear cloaked regions. Those works find the minimal set of nearest-neighbors based on continuous nearest-neighbor search [18]. In a different approach, Casper* only computes a superset of the minimal set of nearest-neighbors that contains the exact nearest-neighbor, in order to achieve trade-off between query processing time and candidate set size for system scalability [10]. In addition, Casper* also supports two more query types: Private Query over Public Data and Public Query over Private Data.

TABLE I
CASPER* QUERY PROCESSING ALGORITHM NOTATIONS

Cloaked region A. Vertex v (v_1, v_2, …, v_i, …v_n).		
Public Object	Filter (NN) of v_i	t_i
	Distance $v_i t_i$	$dist(v_i, t_i)$
Private Object	Filter (NN) of v_i	At_i
	Distance $v_i At_i$	$min - max - dist(v_i, At_i)$

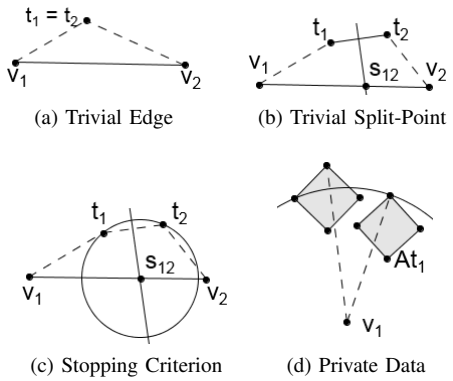
Among previous works, only Casper* supports Query over Private Data, while the others either only support Query over Public Data [3] or lack the trade-off for system scalability [11], [9]. Moreover, Casper* keeps the exact location of LBS users secret from the LBS Providers. However, Casper* is only 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 [2], 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 Nearest-Neighbor Query Processor offers the ability to efficiently handle complex 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 Middleware, the Group Execution Agent, to strongly enhance the whole system's scalability. Simply saying, the GEA groups several queries that have adjacent or overlapped cloaked region into one to reduce computational and communicational cost in the system.

III. THE CASPER* PRIVACY AWARE NEAREST-NEIGHBOR QUERY PROCESSOR

In this section, let us briefly review the Casper* algorithm, start with its terms and notations (Table I). For each vertex v_i of the cloaked region A, its nearest-neighbor is called a $filter$, denoted as t_i if that nearest-neighbor is a public object (public nearest-neighbor) (Figure 2b, t_1, t_2 of v_1, v_2). In case the nearest-neighbor is private, it is denoted as At_i. A private object is considered as a private nearest-neighbor if it has the minimum distance from its cloaked region's furthest corner to v_i (Figure 2d, At_1). The distance between a vertex and its filter is denoted as $dist(v_i, t_i)$ (= $v_i t_i$) (Figure 2b, distance $v_1 t_1$ and $v_2 t_2$ in case of public nearest-neighbor) or $min - max - dist(v_i, At_i)$ (= $v_i At_i$) (Figure 2d, distance from v_1 to the furthest corner of At_1, in case of private nearest-neighbor). 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 $t_i t_j$ (Figure 2b, s_{12}). For the purpose of the Casper* Nearest-Neighbor Query Processing algorithm [10], given a cloaked region A, it is to find all the nearest-neighbors of all the points (1) inside A and (2) on its edges. The algorithm for Query over Public Data 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.

ISSN 2395-8618



(a) Trivial Edge (b) Trivial Split-Point

(c) Stopping Criterion (d) Private Data

Fig. 2. The Casper* algorithm

STEP 2 (Range Selection): For each edge e_{ij} of A, by comparing v_i, v_j's filters t_i and t_j, we consider four possibilities to find the candidate nearest-neighbors and range searches that contain the candidate nearest-neighbors:

- Trivial Edge Condition: If $t_i = t_j$ (Figure 2a, $t_1 = t_2$), t_i (t_j) is the nearest-neighbor of 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 nearest-neighbors (Figure 2b). This means t_i and t_j are the nearest-neighbors of 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 (Figure 2c).

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 nearest-neighbors 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 Middleware to filter the exact nearest-neighbor for the LBS user.

For 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 $min-max-dist(v_i, At_i)$ as a range query into the range queries set R (Fig. 2d, center v_1 and the radius of distance from v_1 to the furthest corner of At_1). The same behavior is applied to v_j and s_{ij} of edge e_{ij}.

IV. System Architecture

Our underlying system architecture consists of two main components, (1) the Location Middleware and (2) the LBS Provider, as depicted in Figure 3.

The Location Middleware manages users' exact positions and their privacy profiles. Upon joining the system, the user agrees to entrust the Location Middleware with the user's precise positions. To preserve the user's location privacy, the user will specify a privacy profile which indicates the desired privacy requirements in interacting with other untrusted components (e.g., the LBS Provider). This privacy profile may include some information such as k (k-anonymity), the minimum and the maximum area of the cloaked regions, or other parameters required by the privacy protection algorithm and the context (time, place, component, etc.) to activate the corresponding profile, etc. In our system, the Location Middleware, according to the privacy profile, will transform the user's precise position into a cloaked region. More than one spatial cloaking algorithm can be employed at the Location Middleware to provide the user with context aware spatial cloaking [2], which selects the best cloaking algorithm given current context of the user. When the user sends an LBS request to the Location Middleware, the request will be forwarded to the LBS Provider with the cloaked region as the input. The LBS Provider will calculate a candidate set of answers based on the cloaked region and send the set back to the Location Middleware. The Location Middleware will then filter the above set for the exact answer and send it to the user. Besides, our Location Middleware also include an additional component, the Group Execution Agent, which will group the LBS queries before sending them to the LBS Provider to strongly enhance the whole system's scalability.

The LBS Provider provides LBS query answers for the Location Middleware. Its main component is the Location Privacy Aware Query Processor. The processor accepts the cloaked region as input instead of the precise position. It will compute a candidate set of answers using the cloaked region and the spatial objects in the database. After that, the LBS Provider will send the set back to the Location Middleware. In this manner, the LBS Provider receives only the cloaked regions, not the exact positions of LBS users. Thus, even if the adversaries gain control over the LBS Provider, they still cannot compromise the LBS users' location privacy. Our Privacy Aware Query Processor is an extended version of the state-of-the-art location privacy aware query processor Casper*. Besides the ability to deal with the four privacy aware query types, it can also efficiently handle complex polygonal cloaked regions, by utilizing the Vertices Reduction Paradigm. Our processor also supports one more type of cloaked region, the circular cloaked region, by transforming it into a polygonal cloaked region. Moreover, the performance of the processor can be tuned through a new parameter, to achieve a trade-off in system scalability, in terms of both query processing time and query candidate set size.

ISSN 2395-8618

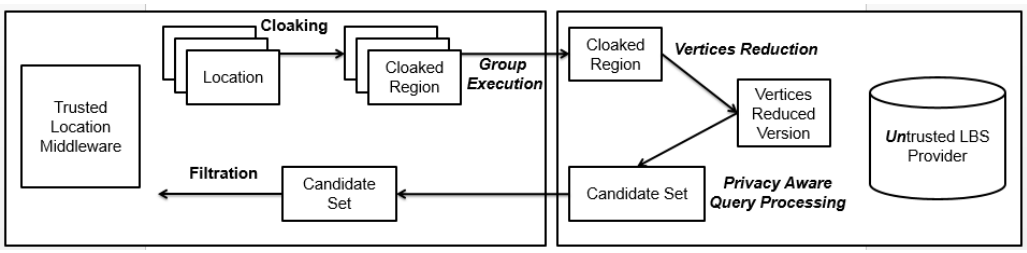


Fig. 3. System Architecture

TABLE II
QUERY PROCESSING ALGORITHM RUNTIME COMPLEXITY UTILIZING VERTICES REDUCTION PARADIGM

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)
# of vertices (4, m, n)	$4 \leq m \leq n$	
Query runtime (Qt_*)	$Qt_4 \leq Qt_m \leq Qt_n$	

In our system, only the Location Middleware and the users themselves are trusted with the exact location information. The LBS provider only receives cloaked regions, which will ensure the users' location privacy. Furthermore, to simplify the system architecture, we assume that the networks between parties are secured and all the pre-processes (including service and user profile registration, trade-offs between participating parties, pay-offs, validation, authentication, authorization, etc.) are already well prepared. It is also worth noting that our proposed enhancements do not affect (not improve and not worsen) the security and privacy aspect of the underlying privacy protection algorithms and query processing processor as we mainly focus on improving the performance and scalability of the system.

V. VERTICES REDUCTION PARADIGM

A. The Polygonal Cloaked Region Problem

As discussed previously, among the polygonal cloaked regions, there are cloaked regions that are complex, which means they have much more than 4 vertices. They can either (1) come directly from the semantic and geographic aware obfuscation algorithms, or (2) they are the group queries' regions that the Group Execution Agent of the Location Middleware sends to the LBS Provider.

For a rectangular cloaked region, the Casper* private query processing algorithm's runtime complexity can be calculated as follows. (1) STEP 1 needs to run exactly 4 nearest-neighbor queries to find 4 filters for the 4 vertices of the rectangular cloaked region, so the runtime is $4Qt_4$ (with Qt_4 is the time of a nearest-neighbor query). (2) In STEP 2, for each edge of the rectangular cloaked region, we need $(2^{refine} - 1)$ nearest-neighbor queries to find the candidates and range searches for STEP 3, so the runtime is $4(2^{refine} - 1)Qt_4$. Hence, the total algorithm runtime is $(4Qt_4 + 4(2^{refine} - 1)Qt_4)$ (worst case). Accordingly, to process a polygonal cloaked region with n vertices, the algorithm's runtime complexity is $(nQt_n + n(2^{refine} - 1)Qt_n)$.

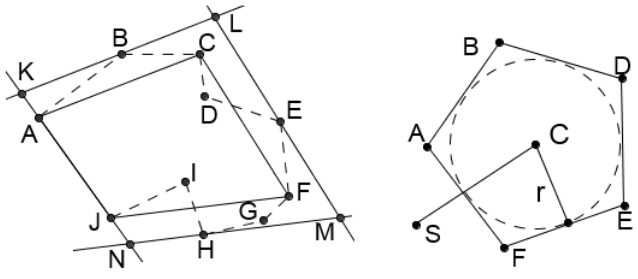
Although the Casper* Privacy Aware Query Processor can deal with polygonal cloaked region A that has n vertices (n-gon), its runtime significantly depends on A's number of vertices (STEP 1) and edges (STEP 2). As shown in Table II's formula 1 and 2, to process an n-gon, Casper* suffers from two aspects. (1) Under the same number of objects in the spatial database and the same access method, the processing time of STEP 1 increases. Because the algorithm has to find n filters for n vertices of the n-gon, it needs to run more nearest-neighbor queries $(4Qt_4 \leq nQt_n)$. Besides, the calculation of $min - max - dist(v_i, At_i)$ in Query over Private Data also increases the nearest-neighbor query runtime for the n-gon because we have to process n vertices of the targets before we can determine their furthest corners $(Qt_4 \leq Qt_n)$. (2) With the same refine value, the processing time of STEP 2 increases as it has to process more edges $(4(2^{refine} - 1)Qt_4 \leq n(2^{refine} - 1)Qt_n)$.

B. The Vertices Reduction Paradigm

To ease the polygonal cloaked region problem, we introduce the Vertices Reduction Paradigm, in which we simplify the polygon so that it has as fewer vertices as possible before processing it with the Casper* algorithm. For that purpose, the Ramer-Douglas-Peucker (RDP) algorithm is employed [19], [20]. For each private object (n-gon) in the database, we maintain only a vertices reduced version (VRV, m-gon, $m \leq n$) of that private object. The vertices reduced version is generated by the Ramer-Douglas-Peucker algorithm and it will be stored inside the database until invalidated, e.g. the user gets a new original cloaked region. For nearest-neighbor query processing, we will use the vertices reduced versions instead of the original ones to reduce processing time ($m \leq n$ and $Qt_m \leq Qt_n$, as depicted in formula 3 of Table II). Accordingly, in the Vertices Reduction Paradigm, the query processing time is only $(mQt_m + m(2^{refine} - 1)Qt_m)$. By using the Vertices Reduction Paradigm, we can achieve at most (nQt_n/mQt_m) speedup in query processing time. In Query over Public Data, $Qt_n = Qt_m$ so the speedup is at most (n/m). In Query over Private Data, $Qt_n/Qt_m = n/m$, so the maximum speedup is $(n/m)^2$.

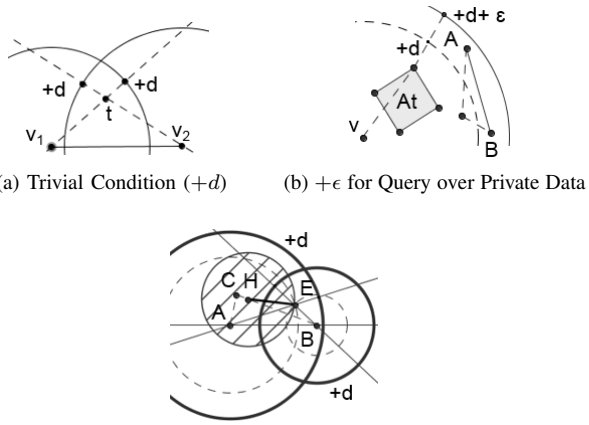
The purpose of the Ramer-Douglas-Peucker algorithm, given an n-gon (ABCDEFGHIJ in Figure 4a), is to find a

ISSN 2395-8618



(a) Polygon (b) Circle

Fig. 4. The Polygonal and Circular Vertices Reduction Versions



(a) Trivial Condition ($+d$) (b) $+\epsilon$ for Query over Private Data

(c) Range searches

Fig. 5. The Query Processing algorithm modifications

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 ($m \leq n$). The inputs are the n-gon's list of vertices and the distance dimension $\epsilon > 0$. 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 the m-gon's list of vertices (Figure 4a, $ACFJ$).

In next subsections, we will discuss two different approaches to utilize the vertices reduced versions. The first one is to use the vertices reduced version directly. The second one, which is better than the first, is to use the bounding polygon of the vertices reduced version. In both approaches, the Ramer-Douglas-Peucker algorithm's overhead in computing the vertices reduced versions is insignificant compared to the total processing time of the query. As depicted in Figure 4a, 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 vertices reduced version is its regular bounding polygon (Figure 4b, $ABDEF$) 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 nearest-neighbor search ($SC + r$ in Figure 4b, where S is the processing vertex, C is the center of the processing private target object and r is the private target object's radius).

C. The Direct Vertices Reduction Paradigm

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 (Table II, formula 3). However, since we use an approximate version of the original cloaked region, we need some modifications in STEP 2 to search for nearest-neighbors of the parts of n-gon that are outside the m-gon (e.g., ABC in Figure 4a). During the Ramer-Douglas-Peucker algorithm's recursive divisions, for each simplified edge, we maintain the distance of the furthest vertex that is not inside the m-gon (B, E and H in Figure 4a). The list's size is at max m. We denote those distances as d (Figure 4a, distance from H to FJ). The modifications (Table III) 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 Figure 4a.

1) Modifications for query over public data: They can be summarized as follows:

- Trivial Edge (**TE**) and Split-Point (**TSP**) 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. As in Figure 5a, given t is the nearest-neighbor of simplified edge v_1v_2, d is the distance of the furthest discarded vertex of edge v_1v_2, we need to add two range queries center at v_1, v_2 with respective radius of $v_1t + d$ and $v_2t + d$. For Trivial Split-Point Condition, we add one more range query centered at sij of a radius $dist(s_{ij}, t_i) + d$ into R. As shown in Figure 5c, given that E is the nearest-neighbor of the simplified edge AB (from edge AC and AB), the nearest-neighbor 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 ($HE' \leq HE$), 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 (**SC**): similarly, we increase the range query's radius by d to ensure including the nearest-neigbors of the outside parts of the original n-gon.

2) Modifications for query over private data: Because the private objects are also simplified, we will increase the search radius by $d + \epsilon$ for not missing them as candidate nearest-neigbors. As depicted in Figure 5b, consider that other private objects are also vertice-reduced versions, regarding vertex v of the m-gon in the query processing phase, the range query $(+d + \epsilon)$ reaches the simplified edge AB of another private object while the range query $(+d)$ does not.

3) Proof of accuracy: Let us consider a simplified edge as in Figure 6a, where the set of vertices A, B, C, D, E is reduced

 http://dx.doi.org/10.17562/PB-52-6

ISSN 2395-8618

TABLE III
ALGORITHM MODIFICATIONS SUMMARY

	Cases	**Ranges** $(center, radius)$
Public Data	TE	$(v_i, v_i t + d)$, $(v_j, v_j t + d)$
	TSP	$(v_i, v_i t_i + d)$, $(v_j, v_j t_j + d)$, $(s_{ij}, s_{ij} t_i + d)$
	SC	$(v_i, v_i t_i + d)$, $(v_j, v_j t_j + d)$, $(s_{ij}, s_{ij} t_i + d)$
Private Data	TE	$(v_i, v_i At + d + \epsilon)$, $(v_j, v_j At + d + \epsilon)$
	TSP	$(v_i, v_i At_i + d + \epsilon)$, $(v_j, v_j At_j + d + \epsilon)$, $(s_{ij}, s_{ij} At_i + d + \epsilon)$
	SC	$(v_i, v_i At_i + d + \epsilon)$, $(v_j, v_j At_j + d + \epsilon)$, $(s_{ij}, s_{ij} At_i + d + \epsilon)$

to edge AE. The box $AFGE$ ($AF = EG = furthest - distance - d$) contains all the outside parts of the simplified edge in this case. If the nearest-neighbors are inside the m-gon, they are already included in the candidate set in STEP3. So we only prove the correctness of STEP2, to find the candidate outside.

Query over Public Data In the trivial edge condition, where there is only one nearest-neighbor T for edge AE, we show that if T is not the nearest-neighbor of all the points in $AFGE$, the real nearest-neighbor Y must also be included in the range searches. There are three possibilities of T's position:

- On the edge AE (1).
- Inside the box $AFGE$ (2).
- On the edge FG or outside the box $AFGE$ (3).

First of all, in three cases, the range searches always cover the whole box $AFGE$, so we only prove that for edge FG, if T is not the nearest-neighbor, the real nearest-neighbor Y must be included in the range searches.

For vertex F, if there is another nearest-neighbor Y, it must satisfy $FY < FT$. Thus, the point Y is included in the range search centered at A, radius $AF + AT$.

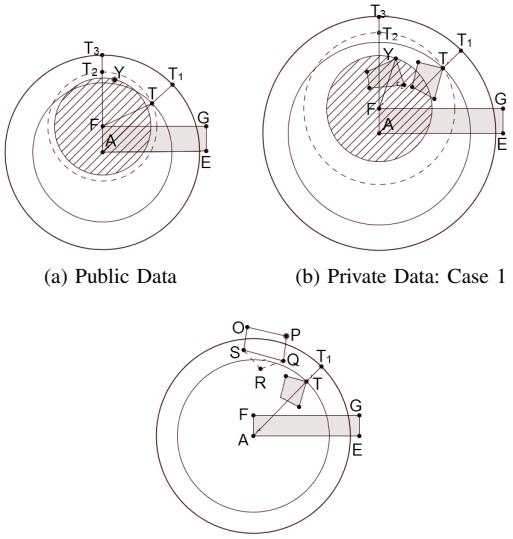
As shown in Figure 6a, A, F, T_2 and T_3 are aligned, given $AT_3 = AT_1$, $FT_2 = FT, TT_1 = AF$, the segment FT_2 is always in the segment AT_3. We can see that the circle center at A, radius AT_3 always contains the circle center at F, radius FT_2, which always contains the circle center at F, radius FY. Hence, for any point Y that is closer to F than T, it will always be included in the circle center at A, radius AT_1.

It is similar to vertex G and all the points on edge FG. So if there is another nearest-neighbor Y of edge FG, it must be included in the range centered at A, radius $AT + AF$ and the range centered at E, radius $ET + EG$.

In the cases of Split-Point or Stopping Criterion Condition, the problems can be considered as the case of Trivial Edge Condition above, by dividing the problem into two sub-problems of edges AS and SE.

Query over Private Data We will provide two proofs in this part, (1) the real nearest-neighbor is included in the range searches and (2) the vertices-reduced private target objects will not be missed in comparison to the original ones.

For the first proof, similarly, the cases of Split-Point or Stopping Criterion Condition can also be considered as the case of Trivial Edge condition. Thus, we only show this case to prove the correctness of the modifications:



(a) Public Data (b) Private Data: Case 1

(c) Private Data: Case 2

Fig. 6. Proof of accuracy: Query over Public/Private Data

- On the edge AE (1).
- Inside the box $AFGE$ (2).
- On the edge FG or outside the box $AFGE$ (3).

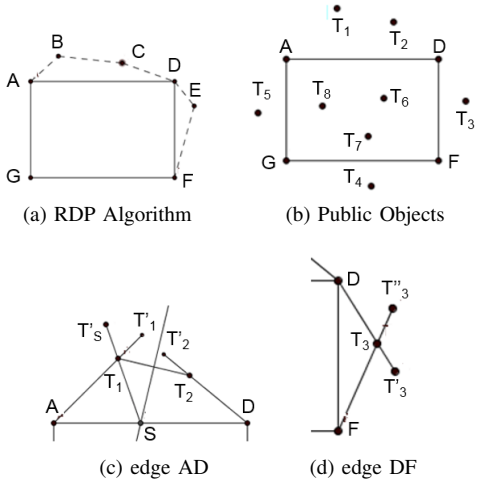
Similar to the proof of query over public data, the range searches cover the whole box and for both vertex F and G. As shown in Figure 6b, the real nearest-neighbor Y is always included in the range searches. So there is no candidate missing in the algorithm.

For the second proof, given the private target object $OPRS$, which is a vertice-reduced version of $OPQRS$. As the set Q, R, S is simplified to edge QS. Let us consider two cases, as shown in Figure 6c:

- Without the enlargement $+\epsilon$, originally, the private target $OPQRS$ will intersect with this range search. But the vertice-reduced version $OPQS$ will not.
- With the enlargement $+\epsilon$, both the private target $OPQRS$ and the vertice-reduced version $OPQS$ will intersect with this range search.

4) Running example: Let us briefly go through a simple running example to illustrate the algorithm more clearly. First, as depicted in Figure 7a, given an n-gon $ABCDEFG$, we process it with RDP, the result is the m-gon $ADFG$. Among all the edges (AD, DF, FG and GA) of the new m-gon, only AD and DF are simplified edge. Hence, the Casper*

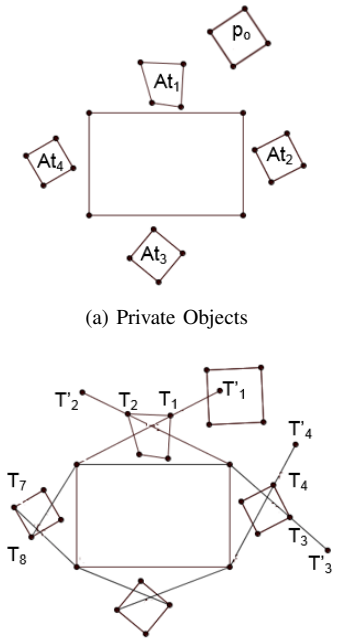
ISSN 2395-8618



(a) RDP Algorithm (b) Public Objects

(c) edge AD (d) edge DF

Fig. 7. Running Example: Private Query over Public Data



(a) Private Objects

(b) Enlarged radius

Fig. 8. Running Example: Private Query over Private Data

algorithm can be applied normally to edge FG and GA. The modifications will only be applied for edge AD and DF.

Query over Public Data Figure 7b shows the public objects related to the m-gon $ADFG$. In STEP1, the filters are found as follows.

A gets the filter T_1 and T_5 ($AT_1 = AT_5$), D gets the filters T_2 and T_3 ($DT_2 = DT_3$), F gets the filter T_3 and T_4 ($FT_3 = FT_4$) and G gets the filter T_4 and T_5 ($GT_4 = GT_5$).

Figure 7c depicts the processing result of edge AD in STEP2. The distance from B to edge AD is used as distance d for this edge processing step. As A and D get different filters (T_1 and T_2), a split-point S is found. We add 3 search ranges (1) centered at A, radius AT'_1, (2) centered at D, radius AT'_2 and (3) centered at S, radius AT'_S, where $T_1T'_1 = T_2T'_2 = T_1T'_S = d$. Figure 7d depicts the processing result of edge DF. Similarly, distance d is the distance from E to edge DF. As D and F get the same filter T_3, we add 2 search ranges centered at D, radius DT'_3 and centered at F, radius FT''_3, where $T_3T'_3 = T_3T''_3 = d$.

For edge FG and AG, as the two vertice of both edges get the same filters (T_4 and T_5), T_4 and T_5 are added into candidate set.

In STEP3, we add all objects inside $ADFG$ (T_6, T_7, T_8) into the candidate set. We also find all the objects inside the range searches found in STEP2.

Query over Private Data Figure 8a shows the private objects related to $ADFG$. The filters found in STEP1 are (At_1 and At_4 for A), (At_1 and At_2 for D), (At_2 and At_3 for F) and (At_3 and At_4 for G).

Figure 8b shows the range searches added in STEP2 for edge AD and DF. We add 2 range searches for each edge, (centered at A, radius AT'_1, $TT'_1 = d + \epsilon$) and (centered at D, radius DT'_2, $TT'_2 = d + \epsilon$) for edge AD, (centered at D, radius DT'_3, $TT'_3 = d + \epsilon$) and (centered at F, radius DT'_4, $TT'_4 = d + \epsilon$) for edge DF.

For edge FG and GA, as shown in Figure 8b, we process them normally with Casper* and gets another 2 range searches for each edge, (centered at F, radius FT_5) and (centered at G, radius GT_6) for edge FG, (centered at G, radius GT_7) and (centered at A, radius AT_8) for edge GA.

In STEP3, we add all private objects that intersect with the range searches added in STEP2 into the candidate set.

D. The Bounding Vertices Reduction Paradigm

One characteristic of the m-gon generated by the Ramer-Douglas-Peucker 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 Ramer-Douglas-Peucker algorithm's recursive divisions, for each simplified edge (m edges), we maintain the furthest vertex that is not inside the m-gon (B, E and H in Figure 4a). 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 Figure 4a). The intersection of those lines forms a new m-gon that contains the original n-gon inside it (Figure 4a's $KLMN$). Therefore, the candidate set of the simplified m-gon is a superset of the original n-gon without directly modifying the Casper* query processing algorithm.

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

 http://dx.doi.org/10.17562/PB-52-6

ISSN 2395-8618

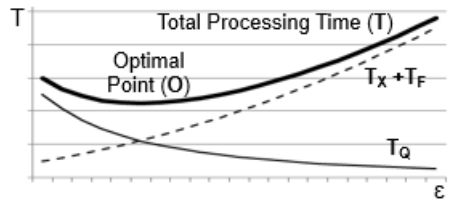
nearest-neighbor because the simplified m-gon contains the original n-gon (no outside parts). Secondly, although the range query's radius is indirectly enlarged through the enlargement of the original n-gons to the bounding m-gons, it is kept minimum as $+d + d$, an indirect $+d$ of the cloaked region and another $+d$ of the private object. Thus the number of results for each range query is also reduced in comparison to the first approach. Furthermore, the reduction in number of range queries also leads to a slight reduction of processing time.

E. The Distance Dimension ϵ as Vertices Reduction Paradigm Tuning Parameter

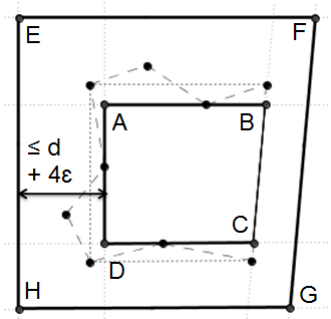
The Total Processing Time (T) of a query consists of three components. (1) The Query Processing Time (T_Q), which is for the Query Processor to compute the candidate set. In the Vertices Reduction Paradigm, T_Q also includes the overhead of the Ramer-Douglas-Peucker algorithm to simplify the query's cloaked region. (2) The Data Transmission Time (T_X), which is for the candidate set to be read from the objects database and transmitted to the Location Middleware for nearest-neighbors filtration. (3) The Answers Filtration Time (T_F), which is for the candidate set to be filtered for the exact nearest-neighbor of the query request. T_Q is monotonically decreasing with the decrease of the 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 Vertices Reduction Paradigm since it affects the number of vertices in the vertices reduced versions and the search radius of range queries in the range query set R. As a result, a large value of ϵ leads to less number of vertices and less query processing time, but larger candidate set size. We will consider two 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 for fewer vertices and edges in query processing. (2) $T_Q < T_X + T_F$. This indicates the candidate set size is too big because the range searches' radius is too large that $(T_X + T_F)$ is longer than T_Q. We have to decrease ϵ to reduce the radii. Thus, in order to find an optimal value of ϵ for the best T, we increase ϵ until it reaches the optimal point O (Figure 9a).

As discussed previously, by utilizing VRP, we can achieve at most (n/m) speedup of Query Processing Time in Query over Public Data and $(n/m)^2$ speedup in Query over Private Data. Thus, the total decrease in Query Processing Time T_Q is accordingly $(1 - m/n)T_Q$ (Query over Public Data) and $(1 - m/n)^2 T_Q$ (Query over Private Data).

Let us consider a uniform-distributed data set, if the distance between public objects is u, the distance from a vertex to its nearest public object is $(u\sqrt{2}/2)$, we denote this distance as d_{pub} $(dist)$. Accordingly, if the longest edge of the bounding rectangle of the private objects is b, so the distance from a vertex to its nearest private object is $((u + b)\sqrt{2}/2)$, we denote this distance as d_{pri} $(min - max - dist)$. In the



(a) The tuning parameter ϵ



(b) The bound of candidate set

Fig. 9. Performance tuning

Vertices Reduction Paradigm, the increase in candidate set size depends on the difference of the areas covered by itself and the original Casper*. In the original Casper*, the area covered is at least less than the inner bounding polygon of the original polygon, as depicted in Figure 9b $(ABCD)$. Meanwhile, in the Vertices Reduction Paradigm, the area covered is at most the polygon enlarged from the outer bounding polygon of the original one, as depicted in Figure 9b $(EFGH)$. The reason of this enlargement is to cover the area of the range queries in STEP 2. As the distance in the respective parallel lines of the inner and the outer bounding polygon is at most 2ϵ and the range query's radius is at most $(d_{pub} + \epsilon)$ (for Query over Public Data) and $(d_{pri} + 2\epsilon)$ (for Query over Private Data). Thus the area difference is at most the difference between the two polygons above, which is at least $C(d_{pub} + 3\epsilon)$ for Query Over Public Data and $C(d_{pri} + 4\epsilon)$ for Query over Private Data, with C is the perimeter of the enlarged polygon $(EFGH$ in Figure 9b). Therefore, for Query over Public Data, the total time increase is $(C/u+1)((d+3\epsilon)/u+1)(t_X+t_F)$, with t_X and t_F is time to transmit and filter one candidate. Respectively, the total time increase is $(C/u + 1)((d + 4\epsilon)/u + 1)(t_X + t_F)$ for Query over Private Data. By collecting the required information (u, average of b, and average of C) of the formula above, we can explain and consider the impact of the increase in candidate set size of the Vertices Reduction Paradigm and adjust the distance dimension ϵ for better system performance. In general, the impact of the Vertices Reduction Paradigm is summarized in Table IV.

It is worth noting that the Vertices Reduction Paradigm works best when the query processing time is much longer than the data transmission and data filtration time $(T_Q >>$

ISSN 2395-8618

TABLE IV
SYSTEM PERFORMANCE IMPACT UTILIZING VERTICES REDUCTION PARADIGM

Public	$\frac{(1-m/n)T_Q \text{ -}(C/u+1)((d+3\epsilon)/u+1)(t_X+t_F)}{d=d_{pub}=(u\sqrt{2}/2)}$	(1)
Private	$\frac{(1-m/n)^2T_Q \text{ -}(C/u+1)((d+4\epsilon)/u+1)(t_X+t_F)}{d=d_{pri}=((u+b)\sqrt{2}/2)}$	(2)

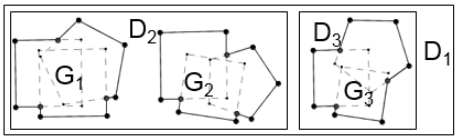


Fig. 10. Effective grouping with R-Tree

$T_X + T_F$), which means the system cannot provide high computational power. In that case, the paradigm will simplify the polygons to reduce the query processing time. But it will also increase the candidate set size to ensure the inclusion of the exact answer, leads to the increase in data transmission and data filtration time. Hence, the best situation to apply the Vertices Reduction Paradigm is that the system has fast read access and high network bandwidth so the increase in data transmission time will not cancel out the benefits of the paradigm. The faster read access and the higher the network bandwidth is, the better effect the Vertices Reduction Paradigm can achieve.

VI. GROUP EXECUTION AGENT

As shown in Figure 10, there are many queries of same filter parameters 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 add one additional component, the Group Execution Agent (GEA), into the Location Middleware. We also propose the Group Execution (GE) algorithm for the Agent. The Group Execution Agent will group as many queries as possible for one query execution before sending them to the query processor (group N queries into K group queries, $K \leq N$, i.e., group 9 queries to 3 groups as shown in Figure 10, the bold $G_{1,2,3}$ are used as cloaked regions in nearest-neighbor queries). The query processor will calculate the candidate set for the whole group as one single query. After that, the Location Middleware will filter the exact answer from the group query's candidate set for each query in the group.

TABLE V
QUERY PROCESSING ALGORITHM RUNTIME COMPLEXITY UTILIZING GROUP EXECUTION AGENT

VRP	$N(T_Q+T_X)+NT_F$	(1)
GEA	$K(T_Q+T'_X)+NT'_F$	(2)
# of queries (N,K)		$K \leq N$
Average T_X time (T_X, T'_X)		$KT'_X \leq NT_X$

Algorithm 1 Group Execution Algorithm ($list\_region$, max_A)

```
1: list_group ← {}
2: for all region r_i in list_region do
3:     min_a ← ∞; min_d ← ∞; min_r_j ← null
4:     for all region r_j ≠ r_i do
5:         a ← bound_area(r_i, r_j);
           d ← intersect_area(r_i, r_j)
6:         if a < min_a and a < max_A then
7:             min_a ← a; min_d ← d; min_r_j ← r_j
8:         end if
9:     end for
10:    list_group ← list_group ∪ {(r_i, min_r_j, d, min_a)}
11: end for
12: sort(list_group) by d descending and a ascending
13: list_grouped_region ← {}
14: for all group (r_i, r_j, d, a) in list_group do
15:    if r_i is not grouped then
16:        grouped_region ← null
17:        if r_j is not grouped then
18:            grouped_region ← group(r_i, r_j);
               mark_grouped(r_i); mark_grouped(r_j)
19:        else
20:            grouped_region ← r_i; mark_grouped(r_i)
21:        end if
22:        list_grouped_region ←
               list_grouped_region ∪ {grouped_region}
23:    end if
24: end for
25: return list_grouped_region
```

The Group Execution algorithm is outlined in Algorithm 1. 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 grouping regions ($K \leq N$) of which areas are smaller than max_A. First, we find the best region pairs (r_i, r_j) that have the least enlargement in area when we group them together, the area a of the grouping regions must be less than max_A. For each pair, we also compute the intersection area d, and put all of them into $list(r_i, r_j, a, d)$. Then we sort the list by $d : descending$, $a : ascending$ and put the queries into group queries accordingly. The main reasons for this list computation and sorting are as follows. (1) $a \leq max_A$. We want to control the areas of the final grouping regions so that they will not be too large, leads to a large candidate set for each group query and long transmission time. (2) Sort $list(r_i, r_j, a, d)$ by $d : descending$, $a : ascending$. We want the cloaked regions of the queries in a group has maximum overlapping area since the larger intersection area, the larger number of common candidates. If two or more groups have the same intersection areas, we will prioritize the group that has smaller union area, according to reason 1.

The grouping regions are the bounding regions of each

 http://dx.doi.org/10.17562/PB-52-6

ISSN 2395-8618

group's regions and will be used as the cloaked regions of those group queries in nearest neighbor query processing. The group query's candidate set is a superset of the candidate sets of the queries in the group (as the final region contains all the regions in the group), so the Group Execution Agent does not miss any exact nearest neighbor of those queries. Thus, the Group Execution Agent does not affect the accuracy of the query processing algorithm. To sum up, from N queries $(q_1, q_2, q_3, ..., q_n)$, we create K group queries $(g_1, g_2, g_3, \ldots, g_k)$. Then we send the K group queries to the query processor to compute their candidate sets. From each group query's candidate set, we calculate the exact answer for each query in the group and send it back to the LBS user.

The system benefits from the Group Execution Agent as shown in Table V. (1) The query processing time for each group queries is the same as a single query (TQ) because we only execute the group query once with the grouping 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 group query's candidate set size increases because it is a superset of the candidate sets 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, to achieve reasonable runtime, 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 runtime complexity is $O(N^2)$. To find those suitable lists, we maintain an R-Tree [21] in the Location Middleware. The R-Tree is not the fastest spatial access method, but its index criterion is that the directory rectangle's area is minimized which suits the Group Execution Agent the best. When a query is sent to the Middleware, its cloaked region is indexed by the R-Tree. By finding the R-Tree's nodes of which the directory rectangle's area is smaller than a predefined area value $kmax_A$, we will get the suitable lists from those nodes' regions. For example, in Figure 10, we find two suitable lists from the nodes D_2 and D_3's regions (given D_1's area exceeds $kmax_A$). Later, the algorithm returns grouping regions G_1, G_2 and G_3, that reduces the number of queries from 9 to 3. The runtime complexity is now only $O(n_1^2+n_2^2+\ldots+n_d^2)$, with n_1 to n_d is the number of regions in the directory rectangles. Because $(n_1 + \ldots + n_d = N)$, so $O(n_1^2 + n_2^2 + \ldots + n_d^2) < O(N^2)$. In this case, we sacrifice the actual best combinations because we only run the Group Execution algorithm on each suitable list instead of all queries in the system at that time. But by that trade-off, we achieve reasonable runtime with nearly best results.

In fact, the Group Execution Agent's speedup effect 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 Group Execution algorithm. However, in most cases, when the number of queries is large enough, the Group Execution Agent does strongly reduce the system's average query processing and transmission time and improve the system scalability. Furthermore, it is also worth noting that the grouping regions are complex polygonal regions since they are the bounding of all cloaked regions in each group. After the grouping regions are sent to the query processor, to reduce total processing time, the Vertices Reduction Paradigm will simplify them before calculating the candidate set. In other words, the Group Execution Agent and the Vertices Reduction Paradigm are complements for each other. The Group Execution Agent eases the drawback of the candidate set size of the Vertices Reduction Paradigm. The Vertices Reduction Paradigm reduces the total processing time when the query processor calculates the candidate set for the grouping regions. Together, they strongly enhance the whole system's performance and scalability.

VII. Experimental Evaluations

A. Overview

In this section, we evaluate both two Vertices Reduction Paradigm approaches and the Group Execution algorithm for the Private Query over Public and Private Data. The Vertices Reduction Paradigm approaches 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 will also examine the system's average total processing time speedup and candidate set size reduction ratio in case of with and without the Group Execution Agent in the Location Middleware.

TABLE VI
Experimental Settings

Cell area	$10000m^2$	**GEA** $kmax_A$	$300000m^2$
Grid size	100x100	**GEA** max_A	$100000m^2$
Circle radius	$180m$-$200m$	**Public Data**	9801
Object size	$< 1KB$	**Circular Private**	3200
Mobile Network	$14.4Mbits/s$	**Polygonal Private**	3200

Table VI summarizes our experimental settings. For all our experiments' private data, first we generate the private moving objects' precise positions using Siafu, a context simulator and the map of the crowded district 1 in Ho Chi Minh city, Vietnam. Siafu, the context simulator takes input of the map (consists of reachable and unreachable regions) and a list of moving objects (with start positions, target positions, etc.) and simulates the movement of those objects over time. We then capture the moving objects' position at a certain random time for uses in the experiments. The total number of private moving objects is 6400, which is big enough to show the effects of the improvements clearly. Then 50% of the precise positions are cloaked into complex rectilinear regions with Bob-Tree [15]. In our experiments, the map is divided into a grid of $100x100$ cells with each cell's area is of $10000m^2$. All the moving objects are obfuscated to reach a total of at least 10

ISSN 2395-8618

cells in their cloaked region. As a result, the complex cloaked regions' numbers of vertices are very high and all range from 15 to 30. The remaining 50% of the private moving objects are cloaked into circular regions utilizing the works in [8] with the radius size at least $180m$ and less than $200m$ so that the area of the circle will be at least the area of the polygon of Bob-Tree. For the public data, we distribute 9801 objects uniformly in all our experiments ($99x99$, each at the center of the Bob-Tree cell above). The sizes of both the public and private data objects are at max $1KB$. We consider the network between Location Middleware and LBS Provider is optimistically high-speed of $1Gbits/s$ while the mobile network is HSPA 3G network of $14.4Mbits/s$ download rate.

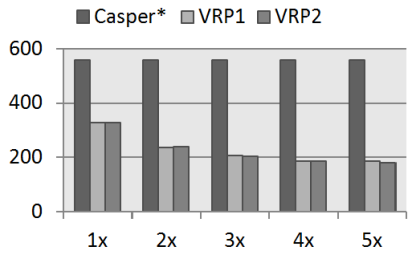
B. Vertices Reduction Paradigm

We will investigate the effect of the tuning parameter ϵ in this section. The baseline algorithm is the original Casper* query processing algorithm. Since Casper* cannot process circular regions, we only compare the total processing time of the polygonal regions. The ϵ values are respectively 1 time to 5 times of the Bob-Tree grid edge size.

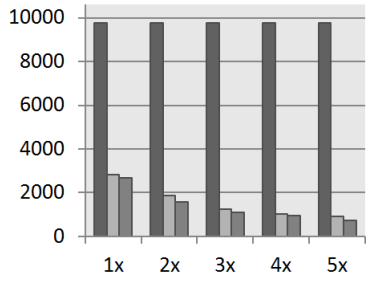
For the charts, the x-axis shows the respective values of the tuning parameter ϵ, as the ϵ increases from 1 time (1x) to 5 times (5x) of the Bob-Tree grid cell edge size. When the ϵ exceeds $5x$, the effects do not increase clearly since we cannot make the cloaked regions simpler than the previous ones in our experiments. The total processing times are measured in milliseconds (ms) and the candidate sets' sizes are measured as the number of data in the result. The abbreviation VRP1 is for the Direct Vertices Reduction Paradigm, while VRP2 is for the Bounding Vertices Reduction Paradigm and Casper* for the original algorithm.

For Private Query over Public Data, as depicted in Figure 11a, when the ϵ value increases, the total processing time decreases. At best (5x), the total processing time is only 32.19% of the original Casper* algorithm's processing time. As explained in Section 5, the main reason for this improvement in total processing time to happen is that the number of range queries is significantly reduced in Vertices Reduction Paradigm. In our experiment, at max reduction (5x), the number of vertices in Vertices Reduction Paradigm is only 24.85% of the original algorithm. Thus we can achieve at most 4 times speedup in Query over Public Data and 16 times speedup in Query over Private Data. For Private Query over Private Data, Figure 11b describes the effect of Vertices Reduction Paradigm. Total processing time in this case is also significantly improved. At best (5x), the total processing time is only 7.34% of the original algorithm's processing time. Besides the reduction in the number of the range queries, using the vertices reduced versions of the cloaked regions also simplifies the process of determining the furthest corners in finding the nearest private objects, thus the total processing time decreases greatly.

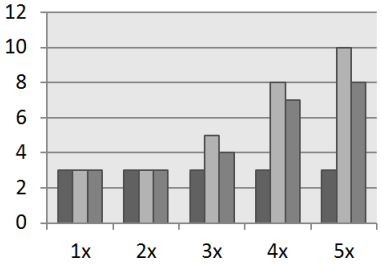
However, as a drawback, the candidate set sizes increase in both cases of Private Query over Public Data and Private Data. In our cases, as illustrated in Figure 11c and Figure 11d, at worst (5x), the candidate set sizes increase by 2.6 times (Private Query over Public Data) and 2.1 times (Private Query over Private Data). But this is not a big problem, since the data size is small (less than $1KB$) and the network between Location Middleware and LBS Provider is high-speed ($100Gbits/s$).



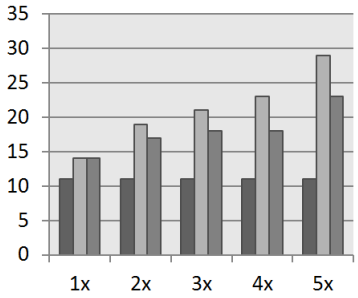
(a) Total Processing Time (ms) of Private Query over Public Data



(b) Total Processing Time (ms) of Private Query over Private Data



(c) Candidate Set Size of Private Query over Public Data



(d) Candidate Set Size of Private Query over Private Data

Fig. 11. Vertices Reduction Paradigm

 http://dx.doi.org/10.17562/PB-52-6

ISSN 2395-8618

To sum up, due to the application of the Vertices Reduction Paradigm, as the tuning parameter ϵ increases, the total processing time decreases significantly while the candidate set size only increases slightly.

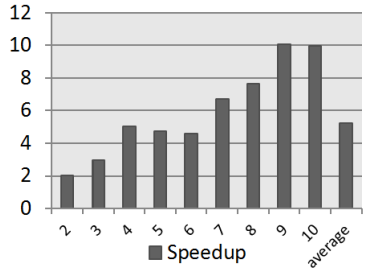
C. Group Execution Agent

We will examine the improvement when we integrate the Group Execution Agent in the Location Middleware in addition to the VRP included in the LBS Provider. In this experiment, we investigate the effect of the Group Execution Agent in the case of grouping 2 to 10 queries (the charts' x-axis is the number of queries grouped), as well as the average benefits in the whole system, in terms of average total processing time and average candidate set size. Similar to the Vertices Reduction Paradigm experiment, we perform this experiment on two private query types, i.e. the Private Query over Public Data and Private Query over Private Data. The pre-defined system parameters max_A and $kmax_A$ are respectively 1.5 and 3 times of the minimum cloaked region area. In other words, $max_A = 150000m^2$ and $kmax_A = 300000m^2$. Meanwhile, the concurrent query number is 500, which means 500 users are issuing the same query content (but with different cloaked regions) at the same time. In this experiment, the total query processing time also includes the overhead of the R-Tree indexing time and region grouping time.

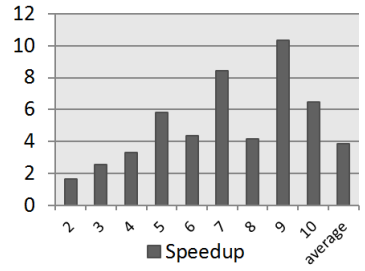
As depicted in Figure 12a and Figure 12b, the system receives great speedup in average total processing. On average, the speedups are 5.24 times (Private Query over Public Data) and 3.84 times (Private Query over Private Data). In other words, the average total processing time decreases at least 3.84 times when the Location Middleware is integrated with the Group Execution Agent. The main reason for this benefit is that the actual number of queries is reduced as we group the queries before sending them to the Privacy Aware Query Processor in the Location-based Database Server.

Figure 12c and Figure 12d illustrate the average candidate set size reduction ratios when we apply the Group Execution Agent in the Location Middleware. The reduction ratios are at max the number of queries in a group, but this is not always guaranteed, as the actual reduction ratio depends on two main factors, (1) the data distribution and (2) the union and intersection area of the cloaked regions. On average, the total numbers of data to transmit are reduced by 5.12 times (Private Query over Public Data) and 3.33 times (Private Query over Private Data).

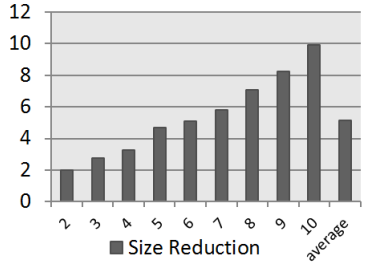
Generally speaking, when the Location Middleware is included with the Group Execution Agent, the whole system scalability increases and we gain speedups in average total processing time and reduction ratios in average candidate set size.



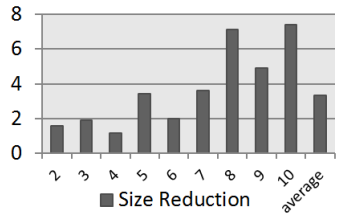
(a) Average Total Processing Time Speedup in Private Query over Public Data



(b) Average Total Processing Time Speedup in Private Query over Private Data



(c) Average Candidate Set Size Reduction Ratio in Private Query over Public Data



(d) Average Candidate Set Size Reduction Ratio in Private Query over Private Data

Fig. 12. Group Execution

VIII. Conclusions and Future Works

This paper introduces an extended version of the state-of-the-art Location Privacy Aware Query Processor, to efficiently deal with complex polygonal cloaked regions, by proposing the Vertices Reduction Paradigm. We also support one more type of cloaked region, the circular shape ones, by transforming them into polygonal ones. The extended query processor can be embedded inside a location-based database server or an untrusted application middleware, allows users to completely keep their exact locations private, by sending only the cloaked regions to the server. In addition, the performance of the query processor can be tuned through a new parameter, to achieve a trade-off in system scalability, in terms of both

ISSN 2395-8618

query processing time and query candidate set size.

As the experimental results show, our works reduce the query processing in both private queries over public and private data for complex polygonal and circular cloaked regions alike, while the increase in candidate set size is very small, especially in private queries over private data. However, the drawback of increase in candidate set size is insignificant since both transmission time and filtration time is very small in comparison to query processing time.

For further development, we are working on improving the Vertices Reduction Paradigm and the Group Execution Agent, i.e. faster algorithms and better effects, supports for continuous queries. In addition, the support of k nearest-neighbor should also be taken into account.

In LBS, location privacy aware query processing that handles cloaked regions has become an essential part in preserving user privacy. However, the state-of-the-art cloaked-region-based query processors only focus on handling rectangular regions, while lacking an efficient and scalable algorithm for other complex region shapes. Motivated by that problem, we introduce enhancements and additional components to the location privacy aware nearest-neighbor query processor that provides efficient processing of complex polygonal and circular cloaked regions, namely the Vertices Reduction Paradigm (VRP) and the Group Execution Agent (GAE).

Generally, our processor provides better support for the Application Middleware in systems that utilize more than one single obfuscation algorithm, as it supports various cloaked region shapes. Besides, as it provides efficient processing of indeterminate rectilinear regions, this result also allows and encourages the development of cloaking algorithms that utilize the geographic, semantic and sensitive features to protect user location privacy, e.g., Bob-Tree. The basic idea of the VRP is that we only maintain a set of vertices-reduced (VR) regions that are computed from the set of private objects' original cloaked regions. The processor will only use the VR versions to reduce computational cost. However, that improvement in total processing time also leads to the increase in candidate set size. To ease that problem, we introduce a new tuning parameter ϵ to achieve trade-off for total processing time and candidate set size. On the other hand, the GAE groups the queries before sending them to the Privacy Aware Query Processor in Location-based Database Server. To achieve that, the queries' cloaked regions are first indexed with the in-memory R-Tree. Then the GAE will scan the R-Tree and perform the group operations. This component strongly enhances the whole system's scalability as it reduces the number of queries to process.

As the experimental results show, our works reduce the query processing in both private queries over public and private data for complex polygonal and circular cloaked regions alike, while the increase in candidate set size is very small, especially in private queries over private data. However, the drawback of increase in candidate set size is insignificant since both transmission time and filtration time is very small in comparison to query processing time.

The applications of both Vertices Reduction Paradigm and Group Execution Agent are not limited to the scope of Privacy Aware Nearest-Neighbor Query. In fact, VRP is able to serve as enhancing component for any algorithms that need to process irregular shapes as general polygons and GEA can be applied to scale up systems that process multiple regions concurrently.

For further development, we are working on improving the Vertices Reduction Paradigm and the Group Execution Agent, i.e., faster algorithms and better effects. In addition, the support of k nearest-neighbor should also be taken into account. Moreover, current private query processing algorithms only focus on position privacy, however, more attention and effort should be put into path privacy too, as the privacy breach in user trajectory also does harms to the development of LBS.

REFERENCES

[1] C. Bettini, X. S. Wang, and S. Jajodia, "How anonymous is k-anonymous? look at your quasi-ID," in *Proceedings of the 5th VLDB Workshop on Secure Data Management*, ser. SDM-08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 1–15.

[2] T. K. Dang, C. N. Ngo, T. N. Phan, and N. N. M. Ngo, "An open design privacy-enhancing platform supporting location-based applications," in *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, ser. ICUIMC-12. New York, NY, USA: ACM, 2012, pp. 59:1–59:10. [Online]. Available: http://doi.acm.org/10.1145/2184751.2184824

[3] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Proceedings of the Third International Conference on Pervasive Computing*, ser. PERVASIVE-05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 152–170.

[4] M. L. Damiani, E. Bertino, and C. Silvestri, "The PROBE framework for the personalized cloaking of private locations," *Trans. Data Privacy*, vol. 3, no. 2, pp. 123–148, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1824401.1824404

[5] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, ser. MobiSys-03. New York, NY, USA: ACM, 2003, pp. 31–42. [Online]. Available: http://doi.acm.org/10.1145/1066116.1189037

[6] T. T. B. Le and T. K. Dang, "Semantic-aware obfuscation for location privacy at database level," in *Proceedings of the 2013 International Conference on Information and Communication Technology*, ser. ICT-EurAsia-13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 111–120.

[7] Q. C. Truong, A. T. Truong, and T. K. Dang, "Memorizing algorithm: Protecting user privacy using historical information of location-based services," *IJMCMC*, pp. 65–86, 2010.

[8] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. di Vimercati, and P. Samarati, "Location privacy protection through obfuscation-based techniques," in *Proceedings of the 21st Annual IFIP WG 11.3 Working Conference on Data and Applications Security*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 47–60. [Online]. Available: http://dl.acm.org/citation.cfm?id=1770560.1770566

[9] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," *IEEE Trans. on Knowl. and Data Eng.*, vol. 19, no. 12, pp. 1719–1733, Dec. 2007. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2007.190662

[10] C.-Y. Chow, M. F. Mokbel, and W. G. Aref, "Casper*: Query processing for location services without compromising privacy," *ACM Trans. Database Syst.*, vol. 34, no. 4, pp. 24:1–24:48, Dec. 2009. [Online]. Available: http://doi.acm.org/10.1145/1620585.1620591

ISSN 2395-8618

[11] H. Hu and D. Lee, "Range nearest-neighbor query," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 78–91, Jan 2006.

[12] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous location-based queries in distributed mobile systems," in *Proceedings of the 16th International Conference on World Wide Web*, ser. WWW-07. New York, NY, USA: ACM, 2007, pp. 371–380. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242623

[13] R. Shokri, C. Troncoso, C. Diaz, J. Freudiger, and J.-P. Hubaux, "Unraveling an old cloak: K-anonymity for location privacy," in *Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society*, ser. WPES-10. New York, NY, USA: ACM, 2010, pp. 115–118. [Online]. Available: http://doi.acm.org/10.1145/1866919.1866936

[14] Q. C. To, T. K. Dang, and J. Kung, "OST-tree: An access method for obfuscating spatio-temporal data in location based services," in *2011 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Feb 2011, pp. 1–5.

[15] Q. C. To, T. K. Dang, and J. Küng, "A hilbert-based framework for preserving privacy in location-based services," *Int. J. Intell. Inf. Database Syst.*, vol. 7, no. 2, pp. 113–134, Apr. 2013. [Online]. Available: http://dx.doi.org/10.1504/IJIIDS.2013.053546

[16] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "SpaceTwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," in *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ser. ICDE-08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 366–375. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2008.4497445

[17] A. Khoshgozaran and C. Shahabi, "Private information retrieval techniques for enabling location privacy in location-based services," in *Privacy in Location-Based Applications*, C. Bettini, S. Jajodia, P. Samarati, and X. S. Wang, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 59–83.

[18] Y. Tao, D. Papadias, and Q. Shen, "Continuous nearest neighbor search," in *Proceedings of the 28th International Conference on Very Large Data Bases*, ser. VLDB-02. VLDB Endowment, 2002, pp. 287–298. [Online]. Available: http://dl.acm.org/citation.cfm?id=1287369.1287395

[19] D. H. Douglas and T. K. Peucker, *Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature*. John Wiley & Sons, Ltd, 2011, pp. 15–28. [Online]. Available: http://dx.doi.org/10.1002/9780470669488.ch2

[20] J. Hershberger and J. Snoeyink, "Speeding up the douglas-peucker line-simplification algorithm," University of British Columbia, Vancouver, BC, Canada, Canada, Tech. Rep., 1992.

[21] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD-84. New York, NY, USA: ACM, 1984, pp. 47–57. [Online]. Available: http://doi.acm.org/10.1145/602259.602266