Querying Distributed Spatial Datasets with Unknown Regions

Qijun Zhu, Dik Lun Lee, and Wang-Chien Lee

Abstract—This paper studies the problem of querying *Bounded Spatial Datasets* (BSDs). A BSD contains i) objects with known locations, and ii) *unknown regions*, each of which bounds an unknown number of objects, within a coverage area. We consider applications where each BSD is hosted on a server or site connected to a communication network and the BSDs overlap in their coverage areas. The challenge is to query the distributed BSDs to retrieve all objects and to minimize the unknown regions which may contain objects satisfying the query, while minimizing the data transmission volume and number of interactions between the query client and the sites. We develop query processing algorithms for two important types of spatial queries, namely, range and k-nearest-neighbor (kNN) queries. We develop the site-based approach and the area-based approach for efficiently processing range and kNN queries on distributed BSDs. They aim to process only a subset of the sites to obtain the full answer for a query. Thus, optimal site selection and the corresponding site querying methods are important problems studied in this paper. In the area-based approach, we prove an optimal division and derive a practical heuristic to partition a query and select the best processing site for each partition, hence achieving even better efficiency than the site-based approach. Simulation results based on three real spatial datasets show that our proposed approaches significantly outperform the baseline that uses a centralized approach in terms of data transmission volume and the number of interactions between the query client and the distributed sites.

Index Terms—Distributed query processing, spatial databases, incomplete datasets, data integration, range queries, kNN queries, cost-efficient query processing

1 Introduction

Spatial database has undergone significant advancements in the last two decades, from handling only static spatial objects to wide range of applications involving mobile and probabilistic spatial objects. Traditional spatial databases assume that the spatial data items contained in the databases have known and often fixed locations (e.g., geographical databases). Mobile spatial databases deal with spatial data items that are moving objects [1]. While the locations of moving objects may not be exactly known, we can estimate an area that guarantees to bound the location of a moving object based on its last known location and its velocity and trajectory [2]. In probabilistic spatial databases, the location of an object is estimated with a probabilistic distribution function over a certain area (e.g., a circular area around a point) [3], [4]. In general, mobile and probabilistic spatial databases assume that the object does exist and the challenge is to estimate its location.

In this paper, we introduce a new type of spatial database, called *Bounded Spatial Dataset* (BSD), which complements mobile and probabilistic spatial databases by providing a novel but simple way of handling uncertain objects. A BSD contains both *exact objects*, which are objects with known locations, and *unknown regions*, which are areas bounding zero or more objects. Applications that produce BSDs include mobile sensor networks, where mobile sensors navigate a field to detect the occurrences of events (e.g., intruders and abnormal temperatures), which have wide applications in health care, environmental monitoring and surveillance [5]. A practical sensor model is the *limited range interaction* model [6], [7], in which sensing performance degrades with distance. That is, objects within a certain range from the sensor can be detected accurately. However, objects beyond a threshold can only be detected with low certainty. Thus, the mobile sensor marks an area to indicate that zero or more events might have occurred within that area, which is called an *unknown region*. This allows a human operator to further investigate or direct a mobile sensor to scan the unknown region.

Compared to mobile and probabilistic spatial databases, our work on BSD is not concerned with how to identify the location of an object within an area, but instead focuses on distributed spatial query processing on BSDs. Query processing on BSDs is a challenging problem for at least two reasons. First, to ensure proper coverage of large premises, a large number of mobile sensors are needed. This leads to the creation of a large number of BSDs on the sensors. Second, to avoid expensive downloads of the BSDs from the sensors, it is desirable to evaluate spatial queries directly

- *Q. Zhu and D. L. Lee are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. E-mail: qijunzhu@alumni.ust.hk, dlee@cse.ust.hk.*
- *W.-C. Lee is with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 USA. E-mail: wlee@cse.psu.edu.*

Manuscript received 8 July 2013; revised 4 Oct. 2013; accepted 10 Oct. 2013. Date of publication 24 Oct. 2013; date of current version 29 Aug. 2014. Recommended for acceptance by J. Pei.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier 10.1109/TKDE.2013.169

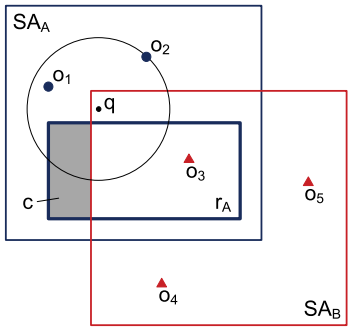


Fig. 1. Invalidated region on one sensor is partially covered by another sensor.

on the sensors with minimal data transfer. The following example illustrates the key issues of query processing in the presence of unknown regions on BSDs.

Example 1. Suppose moving sensors are deployed in an area to detect specific events, with each moving sensor responsible for a sensing area. The example in Fig. 1 shows two overlapping sensing areas SA_A and SA_B covered, respectively, by mobile sensors A and B. Sensor A has recorded two events, o_1 and o_2, and an unknown region r_A, which is suspicious of having some events outbreaking. Sensor B has recorded three events, o_3, o_4 and o_5, and no unknown region, meaning that sensor B is certain that nothing else happened in SA_B. When considering these two BSDs together, we can deduce that only $c = r_A - SA_B$ is an unknown region. For the 2-nearest-neighbor (2NN) query q shown in the figure, the result of q can be derived from both BSDs, i.e., $\{o_1, o_2, o_3, o_4, o_5, c\}$. Thus, we can get the query answer as $\{o_1, o_2, c \cap \odot_{q,o_2}\}$, where $\odot_{q,o_2}$ is the circle centered at q passing through o_2. Notice that $c \cap \odot_{q,o_2}$ is included in the query result as it may contain some abnormal events, supplementary to o_1 and o_2.

We note that although the representation of points and regions in a BSD is similar to that of a conventional spatial dataset, querying BSDs is very different from querying conventional spatial datasets due to the semantics of unknown regions. The above example shows that an unknown region in a BSD can be *resolved* into smaller unknown regions. Specifically, an unknown region r can be eliminated if there exists a BSD D in which the intersection of r and D does not contain any unknown regions. In processing a query on BSDs, we aim to find the *minimum unknown regions* that might contain objects satisfying the query (e.g., $c \cap \odot_{q,o_2}$ in Fig. 1). It is advantageous to minimize the unknown regions in the query result because, as described in the above example, these are the areas that will require further scanning by the mobile sensors.

We assume that each BSD is hosted on a site (or server) connected on a communication network and a query client issues queries to the sites to obtain the results. A straightforward approach to evaluate a query on distributed BSDs is to download all BSDs to the query client and process the query there. This method is obviously unacceptable due to the expensive communication and processing costs. A desirable approach would access only a fraction of the data that is necessary for evaluating the query. Research issues on distributed query processing of non-spatial datasets have been studied in [8]–[11]. These methods cannot be applied in querying distributed BSDs because they address different query semantics. For distributed spatial query processing, most research efforts considered the P2P environment and focused on data allocation and retrieval on the peers [12]–[14], which are not suitable for independent sites considered in this paper.

A challenge of query processing on distributed BSDs is to reduce the number of sites queried and the number of rounds of querying. We propose two distributed query processing strategies, namely, the *site-based approach* and *area-based approach*. In particular, we propose effective site selection and site querying schemes for query processing at a small cost. Moreover, we prove an optimal division method in the area-based approach for effectively reducing redundant transmissions between the sites and the query client. We develop detailed algorithms for two classical spatial queries, namely, range and kNN queries.

In this paper, our main contributions are three-fold:

- We propose a novel spatial database, called Bounded Spatial Datasets (BSDs), consisting of objects and unknown regions and investigate distributed query processing methods for range and kNN queries. We define new semantics for spatial queries when unknown regions are present.
- We propose two cost-efficient approaches, namely, site-based approach and area-based approach, for processing range and kNN queries over distributed BSDs. To reduce data transmission between the sites and the query client, we develop methods to query the sites in an optimal order and methods to further enhance efficiency by partitioning a query into sub-queries and find the optimal site for each sub-query.
- We conduct extensive simulation experiments to evaluate the performance of the proposed approaches. We evaluate them in terms of the data transmission volume and the number of interaction rounds. Experiments show that, compared to the baseline, our proposed approaches can significantly reduce the transmission cost (e.g., up to 95% for kNN queries).

The rest of this paper is organized as follows. Section 2 reviews the related works and Section 3 formulates the problem by analyzing the properties of BSD integration. Then, Section 4 and Section 5 detail the algorithms for range and kNN queries, respectively. Section 6 improves the area-based approach by reducing the computing cost. Section 7 evaluates the performance of our approaches with real data. Finally, Section 8 concludes the paper and discusses future directions.

2 Related Work

2.1 Spatial Query Processing

Generic spatial query processing relies on effective indexing structures to guide the pruning of the search space during query processing to achieve high efficiency. Many index structures, including R-tree [15], kd-tree [16] and quad-tree [17], have been developed for spatial databases. The basic idea of these spatial indexes is to recursively

partition the space or cluster data objects into hierarchical index structures. R-tree and its variants [18], [19] are widely adopted in spatial databases (e.g., Oracle, DB2 and Postgres) to optimize I/O efficiency. R-tree uses minimum bounding rectangles (MBRs) to define the boundaries of object sets and stores the MBRs in the index entries. During query evaluation, MBRs are used to guide traversal in the data space to avoid unnecessary data accesses and thus improve query performance. As distributed query processing on BSDs aims to reduce communication overhead between the BSD sites and the query client, local spatial indexes built on the BSD sites will not impact this cost and thereby are not the major concern of this paper.

2.2 Spatial Data Integration

Spatial data integration has attracted much attention from the research community because of the wide deployment of spatial datasets in various applications. [20] proposed a system for spatial data integration and sharing using Web services technology. [21] considered the query processing issue over integrated spatial database systems. Given an input spatial query Q, they targeted at finding an optimal query execution plan for Q from various plans computed for the integrated spatial data repository. In [22], the authors introduced two integration techniques for heterogeneous geo-spatial data from multiple sources. By using of a federated database of topographic objects and object-relational data in the integration layer, the transparency and middleware services were provided. [23] addressed the data integration and topological consistency problems for distributed and heterogeneous incomplete spatial databases. Here the geometrical information of some spatial objects is missing but topological relations among these objects are available. Thus, the authors proposed topological reasoning methods to achieve effective and efficient query processing. Note that these works mainly consider the incomplete spatial datasets with attribute loss or accuracy loss, which is different from BSD in that all the objects in them can be identified. Thus, the methods cannot be applied in querying distributed BSDs. Moreover, the above works mostly process queries by directly forwarding the rewriting queries to the sources and then integrating the results. Using this approach for querying distributed BSDs is inefficient because many redundant objects and unknown regions may be returned from various sources.

2.3 Distributed Query Processing

Distributed query processing has been studied extensively for different data and network environments. For one-dimensional datasets, various methods have been proposed for top-k queries, i.e., finding the k objects with the highest aggregate values. The seminal Threshold algorithm (TH) [24] goes down the sorted lists of objects in parallel, one position at a time, to calculate the sum of the values at that position across all lists. This sum is used as the *threshold*. Every time a new object appears, TH looks up all lists to find its aggregate value. TH stops when it finds k objects whose values are higher than the threshold. The algorithm is correct because any object that it has not seen cannot have a value higher than the threshold. However, TH consumes an excessive amount of bandwidth when the number of sites is large. To address this issue, [9] proposed a *Three-Phase Uniform Threshold* (TPUT) algorithm to reduce network bandwidth consumption by pruning ineligible objects. It terminates in three roundtrips regardless of the data input. [10] developed a framework named KLEE for approximate top-k queries over widely distributed data sources. It shows that high efficiency can be obtained without much penalty on the result quality. Moreover, communication-efficient algorithms for querying distributed uncertain datasets based on the notion of the expected rank have been presented in [11].

For spatial datasets, many research efforts considered peer-to-peer environments and focused on (i) data allocation and retrieval on the peers [12]–[14]; (ii) extending classic Distributed Hashing Table (DHT) techniques [25], [26]; and (iii) exploring spatial properties of the peers [27]–[29]. The basic assumption in these works is that the peers are coordinated to achieve efficient spatial query processing. However, this assumption does not hold in our problem as BSDs are maintained by independent sites.

TABLE 1
Summary of Notations

Notation	Description
A_i	Coverage area of D_i
D_i	Bounded spatial dataset i
o	Spatial object
P	Query client
q	Spatial query
r	Unknown region
s_i	Site holding D_i
t	Spatial item, i.e., an object or an unknown region
$d(q, o)$	Distance between q and o
$\odot(q, d)$	Circle centered at q with radius d

3 Problem Formulation

3.1 Bounded Spatial Datasets

Suppose there are N distributed sites $S = \{s_1, \ldots, s_N\}$. Each site s_i maintains a *Bounded Spatial Dataset* (BSD) D_i, holding n_i spatial items, i.e., $D_i = \{t_{i,1}, \ldots, t_{i,n_i}\}$, where $t_{i,j}$ is either (i) a *spatial object* $o_{i,j}$ with known location stored in D_i,[1] or (ii) an *unknown region* $r_{i,j}$, which is a rectangle bounding zero or more *missing objects*. A missing object of D_i is an object not explicitly stored in D_i. Note that an unknown region is not necessarily the minimum bounding rectangle (MBR) of the missing objects it bounds. Without loss of generality, we assume that an object $o_{i,j}$ is a point object.

Each BSD D_i has a *coverage area* A_i, which may overlap with the coverage areas of other BSDs. The area outside A_i, denoted as $\overline{A}_i = A - A_i$ where A is the global space, is considered an unknown region of D_i, because D_i has no information about any object located in $\overline{A}_i$. Thus, all together, the area $\cup_{r \in D_i} r \cup \overline{A}_i$ may contain any number of missing objects. Note that a spatial object can exist at a location within the bound of an unknown region in a BSD, and they are considered independent to each other.

Table 1 summarizes the symbols frequently used in this paper. Fig. 2 shows an example of two BSDs, D_1 and D_2. $D_1 = \{o_1, o_2, o_3, o_4, o_5, r_{1,1}, r_{1,2}, r_{1,3}, r_{1,4}\}$, in which o_1, o_2,

1. A *spatial object* is hereafter referred to as *object* when no confusion arises.

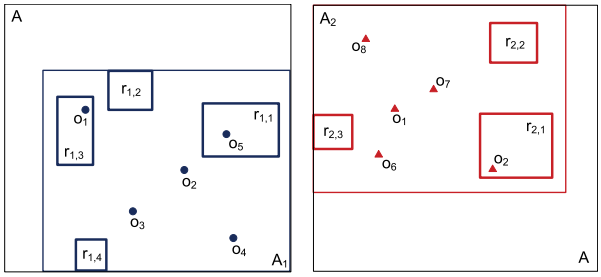


Fig. 2. Example of two BSDs. (a) D_1. (b) D_2.

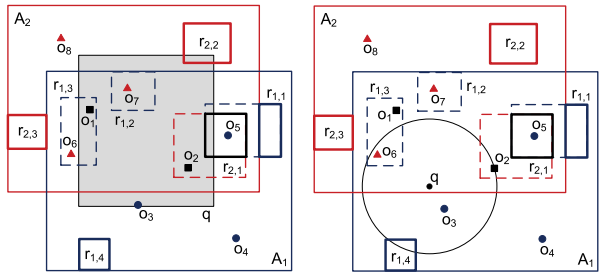


Fig. 3. Spatial queries on distributed BSDs. Square objects are objects stored in both BSDs. (a) Range query. (b) *k*NN query.

o_3, o_4 and o_5 are objects and $r_{1,1}$, $r_{1,2}$, $r_{1,3}$, $r_{1,4}$ are unknown regions, and $D_2 = \{o_1, o_2, o_6, o_7, o_8, r_{2,1}, r_{2,2}, r_{2,3}\}$, in which o_1, o_2, o_6, o_7, and o_8 are objects and $r_{2,1}$, $r_{2,2}$, $r_{2,3}$ are unknown regions. Further, missing objects of D_1, if any, are located in $r_{1,1} \cup r_{1,2} \cup r_{1,3} \cup r_{1,4} \cup \overline{A}_1$, and those of D_2 are location in $r_{2,1} \cup r_{2,2} \cup r_{2,3} \cup \overline{A}_2$.

Integration of BSDs. Based on the containment relationships between the objects and unknown regions in a set of BSDs, we can transform the BSDs into an integrated dataset which contains all of the objects in the BSDs and the *minimum unknown regions*, which are the intersections of the unknown regions in the BSDs, i.e., $\cap_{1 \le i \le N}(\cup_{r \in D_i} r \cup \overline{A}_i)$. Back to the example in Fig. 2. Although D_1 does not know what objects, if any, are located in $r_{1,2}$, we can examine D_1 and D_2 together and conclude that o_7 is the only missing object in $r_{1,2}$ due to the fact that $r_{1,2} \cap D_2$ contains only o_7. That is, $r_{1,2}$ is no longer an unknown region in the integrated dataset. We say that $r_{1,2}$ is *resolved*. Readers may examine o_7 and $r_{1,2}$ in the integrated dataset of D_1 and D_2 shown in Fig. 3(a) (ignore for the moment the range query q in the figure). With the same reasoning, we can conclude that the areas $r_{2,1} - r_{1,1}$ and $(r_{1,1} - r_{2,1}) \cap A_2$ (i.e., the dashed L-shapes in, respectively, $r_{2,1}$ and $r_{1,1}$) are *resolved*, whereas $r_{2,1} \cap r_{1,1}$ and $r_{1,1} - A_2$ (i.e., the two solid rectangles in $r_{1,1}$ and $r_{2,1}$) remain *unresolved* in the integrated dataset. After performing all resolution on D_1 and D_2, we can obtain the final integrated dataset containing objects $\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\}$ and minimum unknown regions $\{A - A_1 - A_2, r_{1,1} - A_2, r_{1,1} \cap r_{2,1}, r_{1,4}, r_{2,2}, r_{2,3}\}$.

3.2 Querying Distributed BSDs

Given a set of BSDs, each of which is hosted on a server connected to a network, a query client P is required to obtain the answer for a spatial query q by querying the BSDs while minimizing the amount of data transmitted over the network. In order to facilitate retrieval of spatial items in BSDs, we assume that each spatial item, $t_{i,j}$, maintains an attribute indicating if $t_{i,j}$ is an object or an unknown region. Further, we assume that the BSDs are updated continuously. To minimize maintenance overhead, P does not maintain any cache, index or summary data about the BSDs. It has to collect from the BSDs all information it needs to answer a query during query processing time.

Since BSDs contain unknown regions, traditional query semantics and query processing techniques are inapplicable. We define the semantics of a spatial query as follows. The answer of a spatial query q contains (i) the set of objects that satisfies q without considering the unknown regions, and (ii) the minimum unknown regions of the BSDs that satisfy q. The former ensures that all qualified objects are obtained when the minimum unknown regions contain no missing objects, whereas the latter ensures that the unknown regions returned in the answer set are the tightest bounds within which objects satisfying q may be found. The formal definition of range queries under BSDs is given in Definition 1.

Definition 1 (Range Query Semantics for BSDs). *Given N distributed BSDs $D_1, \ldots, D_N$ and a range query q, the answer set contains (i) all objects in $\cup_{1 \le i \le N} D_i$ that are inside q, i.e., $\{o | o \in \cup_{1 \le i \le N} D_i \wedge o \in q\}$, and (ii) the parts of the minimum unknown regions inside q, i.e., $\cap_{1 \le i \le N}(\cup_{r \in D_i} r \cup \overline{A}_i) \cap q$.*

Fig. 3(a) shows the results for a range query q on the BSDs in Fig. 2. We can see that the answer set for q consists of o_1, o_2, o_3, o_7, which are all of the objects inside q, and the minimum unknown regions intersecting with q, namely, $q \cap r_{1,1} \cap r_{2,1}$. The formal definition of *k*NN queries under BSDs is given in Definition 2.

Definition 2 (*k*NN Query Semantics for BSDs). *Given N distributed BSDs $D_1, \ldots, D_N$, a kNN query q, and let d be the distance between q and q's k^{th} nearest object in the integrated dataset of the BSDs, and $\odot(q, d)$ denotes the circle centered at q with radius d, the answer of q contains (i) all objects in $\cup_{1 \le i \le N} D_i$ that are inside $\odot(q, d)$, i.e., $\{o | o \in \cup_{1 \le i \le N} D_i \wedge o \in \odot(q, d)\}$, and (ii) parts of the minimum unknown regions that are inside $\odot(q, d)$, i.e., $\cap_{1 \le i \le N}(\cup_{r \in D_i} r \cup \overline{A}_i) \cap \odot(q, d)$.*

Fig. 3(b) shows the results for a 3NN query q on the BSDs in Fig. 2. Let d be the distance between q and o_2. We can see that $\odot(q, d)$ includes exactly three objects o_2, o_3 and o_6. Thus, the answer of q consists of o_2, o_3 and o_6 and the parts of the minimum unknown region intersecting $\odot(q, d)$, i.e., $\odot(q, d) \cap r_{1,4}$.

3.3 Distribute Query Processing Methods

In a distributed environment, it is infeasible for the query client to download all BSDs to produce the integrated dataset and execute the query on the integrated dataset as described in Section 3.2. In this paper, we propose two distributed query processing approaches, namely, the *site-based* and *area-based* approaches, to minimize the data transmission cost, which dominates query processing cost in a distributed environment. Before we describe our approaches,

we first introduce a *baseline approach*, denoted as *CT*, in which (i) the query client P sends q to every site s_i, (ii) s_i executes q locally and returns the *local results* satisfying q to P, and (iii) P integrates the local results into an integrated dataset and execute q on it to produce the final result for q. It is easy to see that each site s_i needs to return the following *local results* to the query client:

Range Query: All objects inside q and all unknown regions which intersect q.

k**NN Query**: All objects inside $\odot(q, d_i)$ and all unknown regions which intersect $\odot(q, d_i)$, where d_i is the distance between q and q's k^{th} nearest object within D_i.

Although *CT* does not download all BSDs into the query client, it has two drawbacks: (i) all sites are queried, and (ii) each site produces the local result from the original query q. As shown in Section 7, *CT* incurs much redundant data transmission compared to our proposed approaches. The general principle of our proposed approaches is to process q on a selected subset of the sites. Each selected site returns the local result for q. Then, the unknown regions in the local results form a *remainder query*, which is used to query another selected subset of the sites to retrieve more objects satisfying q and reduce the unknown regions in the remainder query further. The procedure is repeated until all objects satisfying q are retrieved and the remainder query cannot be reduced further (i.e., the *minimum unknown regions* are obtained).

Our proposed distributed query processing approaches minimize data transmission cost by reducing the total number of sites queried and the remainder query used in each round of querying. However, they incur multiple *interactions* between the query client and the sites. Hence, the query cost is affected by both the *total transmission cost* and the *number of interactions* incurred. These two performance measures will be used in the performance evaluation presented in Section 7.

The site-based and area-based approaches proposed in this paper differ in that the former processes one site at a time and utilizes the site fully to reduce a remainder query, whereas the latter analyzes the remainder query, divides the remainder query into sub-queries and select the best site for each sub-query to process. In Section 4 and Section 5, we describe the detailed algorithms and analysis under these two strategies for range and kNN queries, respectively.

4 Range Query Processing

4.1 Site-Based Approach

The site-based approach consists of two steps. First, a *site selection function*, denoted as $SiteSelRange(q_{j-1})$, selects *one site* for processing the remainder query q_{j-1}. Second, a *site query function*, denoted as $SiteQueryRange(q_{j-1}, D_i)$, queries the remainder query q_{j-1} on the selected site s_i. After site querying, a new remainder query q_j is obtained, which is then processed by the same two steps until the remainder query cannot be resolved further by the sites that are not yet queried. Let's first define the following notations:

- $H_j = \{s_{i_1}, \ldots, s_{i_j}\}$: the first j sites to be processed.
- $\bar{c}(q, H_j)$: the unknown regions within q after processing H_j, denoted as the *remainder query* q_j. q_0 is the initial query q, when no site has been processed yet.
- $\tilde{c}(q_{j-1}, D_{i_j})$: the area of q_{j-1} resolved by s_{i_j}, i.e., $q_{j-1} \cap A_{i_j} - \cup_{r \in D_{i_j}} r$.

The major challenge of the site-based approach is to decide the order of querying the sites to minimize the transmission cost. Thus, we have to first develop the cost model of range query processing in the site-based approach. Without any prior knowledge about the BSDs, the query client has to collect some auxiliary data to make the site selection decision, and the corresponding transmission cost is denoted as $cost_{aux}$. In addition, the transmission cost of querying q_{j-1} on s_{i_j} is denoted as $cost(q_{j-1}, D_{i_j})$. The overall transmission cost of querying the BSDs can be modeled as follows.

$$cost(q, H_n) = cost_{aux} + \sum_{j=1}^{n} cost(q_{j-1}, D_{i_j}). \quad (1)$$

Our goal is to minimize $cost(q, H_n)$, which can be solved in two steps: (i) select the proper site by only considering the data size of the collected items, and (ii) optimize site querying to further reduce $cost_q$.

Site Selection. The site selection function $SiteSelRange(q_j)$ determines the site to be queried next given a remainder query q_j. Given a site processing order H_n, our goal is to minimize the total size of the items collected from the sites during the processing of q, which can be expressed as:

$$size(q, H_n) = \sum_{j=1}^{n} size(q_{j-1}, D_{i_j}), \quad (2)$$

where $size(q_{j-1}, D_{i_j})$ is the size of the items collected from s_{i_j} when q_{j-1} is processed on s_{i_j}. For range queries, $size(q_{j-1}, D_{i_j})$ is the size of the objects and unknown regions in D_{i_j} that are inside q_{j-1}.

An optimized site selection aims to redundant transmission, which is due to the transmission of the same objects more than once and unknown regions not part of the minimum unknown region of the final result. In the example BSDs in Fig. 3(a), if D_1 is queried before D_2, D_1 returns o_1, o_2, o_3, $r_{1,1}$, $r_{1,2}$, and $r_{1,3}$. The remainder query after processing D_1 is $\{r_{1,1} \cap q, r_{1,2} \cap q, r_{1,3} \cap q\}$. Processing it on D_2 returns o_1, o_7, $r_{2,1}$, $r_{2,2}$. In this case, o_1, $r_{1,2}$, $r_{1,3}$ are redundant, since o_1 is retrieved twice, whereas $r_{1,2}$ and $r_{1,3}$ are not part of the minimum unknown regions in the final result (see Section 3.1 for the final result of this example). If D_2 is queried before D_1, D_2 returns o_1, o_2, o_7, $r_{2,1}$, and $r_{2,2}$, and D_1 returns o_2, o_3, and $r_{1,1}$. In this case, only o_2 is redundant. We conclude that D_2 followed by D_1 is a better site selection order.

Theorem 1. *The optimal site selection sequence H_n which minimizes $cost_{H_n}$ must satisfy $size(\tilde{c}(q_{j-1}, D_{i_{j+1}}), D_{i_j}) \leq size(\tilde{c}(q_{j-1}, D_{i_j}), D_{i_{j+1}})$ for any $1 \leq j < n$.*

Theorem 1[2] proves that the optimal H_n satisfies $size(\tilde{c}(q_{j-1}, D_{i_{j+1}}), D_{i_j}) \leq size(\tilde{c}(q_{j-1}, D_{i_j}), D_{i_{j+1}})$ for any $j < n$. Given a remainder query q_{j-1}, the site s_i which resolves the largest area of q_{j-1} usually leads to a large $size(\tilde{c}(q_{j-1}, D_i), D_{i'})$ compared to any other site $s_{i'}$. That is,

2. Proofs of the theorems in this paper are in Appendix I, available online at http://www.cse.ust.hk/faculty/dlee/tkde-bsd/appendix.pdf.

$size(\tilde{c}(q_{j-1}, D_i), D_{i'}) > size(\tilde{c}(q_{j-1}, D_{i'}), D_i)$ if $\tilde{c}(q_{j-1}, D_i) > \tilde{c}(q_{j-1}, D_{i'})$. Thus, $SiteSelRange(q_{j-1})$ selects the site with the largest $\tilde{c}(q_{j-1}, D_i)$ from the unprocessed sites as s_{i_j}. In addition, since $q_j = q_{j-1} - \tilde{c}(q_{j-1}, D_{i_j})$, the site will result in a smaller remainder query, thus having a high capability to prune the unprocessed sites. However, it is not easy to estimate the area of q_{j-1} resolved by s_i, because it requires collecting all of the unknown regions of s_i that intersects q_{j-1}. To achieve proper site selection efficiently, P collects from each site s_i the ratio of the resolved area of q:

$$sel_i(q) = \frac{|q \cap A_i - \cup_{r \in D_i} r|}{|q \cap A_i|}. \quad (3)$$

Assuming that the objects and unknown regions are uniformly distributed, we can estimate $|\tilde{c}(q_{j-1}, D_i)|$ as $sel_i(q)|\tilde{c}(q, H_{j-1}) \cap A_i|$.

Site Querying. The site querying function, denoted as $SiteQueryRange(q_{j-1}, D_{i_j})$, submits q_{j-1} to site s_{i_j} and receives the local result back. To reduce the cost of site querying, we consider two methods of reducing q_{j-1}: (i) $q'_{j-1} = q_{j-1} \cap A_{i_j}$, i.e., we only submit the parts of q_{j-1} that is inside A_{i_j}, and (ii) the minimum bounding rectangle (MBR) of q'_{j-1}, denoted as $M(q'_{j-1})$. The transmission costs of these two choices can be expressed as:

$$cost(q_{j-1}, D_{i_j}) = size(q_{j-1} \cap A_{i_j}) + size(q_{j-1} \cap A_{i_j}, D_{i_j})$$

and

$$cost(q_{j-1}, D_{i_j}) = size(M(q_{j-1} \cap A_{i_j})) + size(M(q_{j-1} \cap A_{i_j}), D_{i_j}),$$

where $size(t)$ denotes the size of an item t. We can see that, compared to q'_{j-1}, $M(q'_{j-1})$ leads to lower upload cost for the query client because the remainder query, which may consist of many parts, is represented as a single *MBR*. However, the download cost is higher, since the $M(q'_{j-1})$ is larger than q'_{j-1} and hence will retrieve more items from a site. Thus, the two strategies are suitable for different situations. The site-based approach estimates the transmission costs of these two strategies before querying the selected site and uses the strategies with the lower cost. To compute $size(c, D_i)$ efficiently, P collects a summary of the data size of D_i as follows.

$$que_i(q) = \frac{\sum_{t \in D_i \wedge t \cap q \neq \phi} size(t)}{|q \cap A_i|} \quad (4)$$

and estimates $size(c, D_i)$ as $que_i(q)|c \cap A_i|$.

Processing Algorithm. To process a range query q on the distributed BSDs, the query client P maintains $\bar{c}$ representing the current remainder query, a set of objects O_q that appear in the intermediate result, and a list of candidate sites S_{cand} which may further contribute to the remainder query. Initially, $\bar{c} = q$, $O_q = \phi$, and $S_{cand} = \{s_1, s_2, \ldots, s_N\}$. The algorithm is detailed below.

Step 1. P sends the range query q to all N sites. Each site s_i returns the overlap between q and A_i, denoted as $A_{i,q} = q \cap A_i$, and the auxiliary data $sel_i(q)$ and $que_i(q)$.

Step 2. P utilizes $SiteSelRange(\bar{c})$ to select a site s_j from S_{cand}, and then performs $SiteQueryRange(\bar{c}, D_j)$ to obtain the objects O_j as well as the unknown regions R_j within $\bar{c}$ from site s_j. After receiving these data, P updates the answer set as follows.

- $O_q \leftarrow O_q \cup O_j$
- $\bar{c} \leftarrow \bar{c} - (A_{j,q} - \cup_{r \in R_j} r)$.

Then, P removes s_j as well as any site s_i with $A_{i,q} \cap \bar{c} = \phi$ from S_{cand} because they are unable to resolve the remainder query $\bar{c}$. This step is repeated until $S_{cand} = \phi$.

Step 3. P obtains the final answer set $T = O_q \cup \bar{c}$.

Take the range query in Fig. 3(a) as an example. Table 2 lists the steps for site-based range query processing: P first queries s_2 by sending the remainder query inside A_2 (in Step 2) and then queries s_1 by sending the remainder query inside A_1 (in Step 3). This is in fact the optimal procedure for this example.

TABLE 2
Site-Based Range Query Processing for Fig. 3(a)

#	Upload	Download	Processing state
1	$s_1, s_2 : q$	s_1, s_2 : aux. data	$O_q = \phi, \bar{c} = q$
2	$s_2 : A_{2,q}$	$s_2 : o_1, o_2, o_7, r_{2,1}, r_{2,2}$	$O_q = \{o_1, o_2, o_3\}$ $\bar{c} = q \cap (A_2 \cup r_{2,1} \cup r_{2,2})$
3	$s_1 : \bar{c} \cap A_1$	$s_1 : o_2, o_3, r_{1,1}$	$O_q = \{o_1, o_2, o_3, o_7\}$ $\bar{c} = q \cap (r_{1,1} \cap r_{2,1} \cup r_{2,2})$

4.2 Area-Based Approach

Compared to *CT*, which queries all the sites, the site-based approach is efficient because it queries the sites one by one based on an optimal order. Unfortunately, the approach has a limit on performance. As Theorem 1 suggests, querying s_i with the largest $\tilde{c}(q_{j-1}, D_i)$ usually incurs minimal data transmission redundancy. However, although a selected site s_{i_j} may have the largest $\tilde{c}(q_{j-1}, D_{i_j})$, it cannot guarantee that $\tilde{c}(c, D_{i_j})$ is the largest for a part c of q_{j-1}.

To overcome this problem, we propose the *area-based approach*, which partitions a query q into *subqueries* and optimizes site selection and query processing for each subquery of q in each interaction round. This approach can resolve q more than the site-based approach, resulting in a reduction of the number of interaction rounds. However, obtaining an optimal partitioning of q requires the unknown regions (i.e., their coordinates and sizes) in each BSD to be known, which incurs a prohibitively high cost. In this subsection, we introduce *overlap division* on the BSD of a site s_i to generate a coarse representation of the unknown regions for query partitioning and show that the method is efficient. Essentially, the overlap division method partitions the overlap between the coverage area of a BSD and q with horizontal and vertical lines, known as *division lines*, and represents the proportion of unknown regions in each partition. In the following subsections, we address two major issues in the site-based approach: (i) overlap division for optimizing query partitioning, and (ii) range query processing based on overlap division.

4.2.1 Overlap Division

Given a BSD D_i and an area c, we use entropy as a measure of the distribution of unknown regions within c:

$$E_i(c) = -(\frac{|\bar{c}|}{|c|} \log \frac{|\bar{c}|}{|c|} + \frac{|\tilde{c}|}{|c|} \log \frac{|\tilde{c}|}{|c|}), \quad (5)$$

TABLE 3
Procedure of 3-Step Overlap Division (Unit: 10^{-3})

	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
1	**96.2**	36.0	17.8	3.6	59.3	52.8	33.2	7.1	84.4
2	-	44.0	1.9	4.1	16.6	**58.7**	6.3	8.4	3.5
3	-	19.7	8.1	2.9	**36.9**	-	12.9	34.1	2.1

where $\bar{c}$ denotes D_i's unknown regions that are in c and $\tilde{c} = c - \bar{c}$. Particularly, when $E_i(c) = 0$, c either does not contain any unknown region or is entirely composed of unknown regions in D_i; when $E_i(c) = 1$, half of c is unknown regions in D_i.

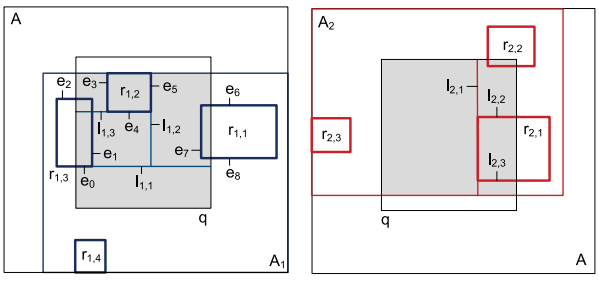
For a range query q, a site s_i can only contribute to the overlap area $A_{i,q}$. Thus, we only need to know the unknown regions in $A_{i,q}$. We propose to divide $A_{i,q}$. Based on $E_i(c)$, an optimal overlap division is realized by recursively selecting a horizontal or vertical division line l_c from a divided part c of $A_{i,q}$ to maximize the following function,

$$I_i(c, l_c) = |c|E_i(c) - \sum_{j=1}^{2} |c_j|E_i(c_j), \tag{6}$$

where c_1 and c_2 are the divided parts of c produced by l_c. Therefore, each division line yields the maximal information gain about the distribution of the unknown regions in $A_{i,q}$. The whole procedure ends when the number of division lines reaches a threshold δ or when every divided part either contains entirely of or does not contain any unknown region of D_i. An overlap division produced by δ division lines is called a *δ-step overlap division*. To find the optimal division line which maximizes Equation 6, we prove a special property as described in Theorem 2.

Theorem 2. *Given a BSD D_i and a rectangular area c, the optimal division line l_c must contain a bounding edge of the unknown regions of D_i in c.*

Based on Theorem 2, given a partition of a BSD, we only need to check all edges of the unknown regions in the partition to obtain the locally optimal division lines of the BSD. Table 3 shows the steps for obtaining the optimal division line for the exemplary BSD D_1 with $\delta = 3$. Each row shows the $I_i(c, l_c)$ values of the edges under consideration. Initially, there are totally nine edges, $e_0, \ldots, e_8$, to be considered. Since e_0 has the largest $I_i(c, l_c)$ value, it is selected as a division line. e_5 and e_4 are selected in a similar way. The complete overlap divisions for both D_1 and D_2 are shown in Fig. 4.



(a) (b)

Fig. 4. Overlap divisions with $\delta = 3$ for the exemplary BSDs. (a) D_1. (b) D_2.

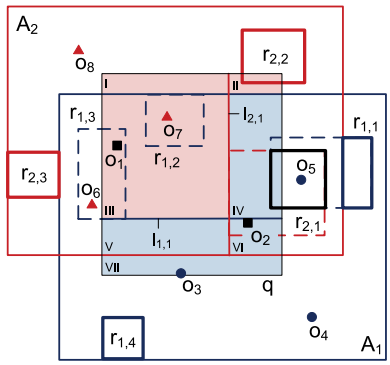


Fig. 5. Partition of the exemplary range query.

4.2.2 Range Query Processing

Based on the intersections of the overlap divisions of all of the sites, we can generate an initial partition of q, which can then be used to partition a remainder query q_j into subqueries during query processing. The best processing site is selected for each subquery. Since there could be more than one subquery mapped to the same site, all of the subqueries mapped to the same site are merged and sent to the site for processing. The whole processing algorithm is detailed below.

Step 1. P sends the range query q to all N sites. Each site s_i makes a δ-step overlap division and returns to P the overlap $A_{i,q} = q \cap A_i$, the δ division lines, and the auxiliary data $sel_i(c_{i,j})$ and $que_i(c_{i,j})$ for each partition $c_{i,j} \subseteq A_{i,q}$ created by the division lines on $A_{i,q}$. Fig. 5 shows the 1-step overlap divisions for the exemplary BSDs. The horizontal division line $l_{1,1}$ (colored in blue) divides $A_{1,q}$ into $c_{1,1}$ and $c_{1,2}$, and the vertical division line $l_{2,1}$ (colored in red) divides $A_{2,q}$ into $c_{2,1}$ and $c_{2,2}$.

Step 2. P partitions q into a set of subqueries $\{c_{q,m}\}$ based on the intersections of the overlap divisions received from the N sites. As shown in Fig. 5, q is partitioned into c_I, c_{II}, c_{III}, c_{IV}, c_V, c_{VI}, and c_{VII}. For each $c_{q,m}$, identify all sites s_i's, in which $c_{q,m} \subseteq A_i$ and is contained by $c_{i,j}$. Create a site list $S_{q,m}$ of s_i's sorted in decreasing order of $sel_i(c_{i,j})$, which approximates $sel_i(c_{q,m})$.

Step 3. For a remainder query $q_j = \bar{c}(q, H_j)$, P selects the first site of $S_{q,m}$ for each subquery $c_{q,m} \cap q_j$. All subqueries $c_{q,m}$ assigned to the same site are merged and sent to the site for processing, and the the items within q_j from the selected sites are obtained. Let O_i and R_i denote the objects and the unknown regions collected from s_i. P updates the answer set as follows.

- $O_q \leftarrow O_q \cup (\cup_i O_i)$
- $q_{j+1} \leftarrow q_j - \cup_i (c_i - \cup_{r \subseteq R_i} r)$.

For any $c_{q,m}$, if $c_{q,m} \cap q_{j+1} = \phi$, P sets $S_{q,m} = \phi$; otherwise, P removes the first site of $S_{q,m}$. This step is repeated until $S_{q,m} = \phi$ for any $c_{q,m}$.

Step 4. P obtains the final answer set $T = O_q \cup q_{j+1}$.

Table 4 lists the steps of the area-based approach for the exemplary range query. We can see that the area-based approach effectively reduces the number of collected items at a small cost by obtaining finer remainder queries and additional auxiliary data. Thus, the area-based approach in general is more cost-efficient for querying distributed

TABLE 4
Area-Based Range Query Processing for Fig. 5

#	Upload	Download	Processing state
1	$s_1, s_2 : q$	s_1, s_2 : aux. data	$O_q = \phi, \bar{c} = q$
2	$s_1 : c_{IV-VII}$ $s_2 : c_{I-III}$	$s_1 : o_2, o_3, r_{1,1}$ $s_2 : o_1, o_7, r_{2,2}$	$O_q = \{o_1, o_2, o_3, o_7\}$ $\bar{c} = q \cap (r_{1,1} \cup r_{2,2})$
3	$s_2 : q \cap r_{1,1}$	$s_2 : r_{2,1}$	$O_q = \{o_1, o_2, o_3, o_7\}$ $\bar{c} = q \cap (r_{1,1} \cap r_{2,1} \cup r_{2,2})$

TABLE 5
Site-Based *k*NN Query Processing for Fig. 3(b)

#	Upload	Download	Processing state
1	$s_1, s_2 : q$	$s_1 : d(q, o_1)$ $s_2 : d(q, o_1)$	$\bar{d} = d(q, o_1), O_q = \phi$ $\bar{c} = \odot(q, \bar{d})$
2	$s_1, s_2 : \bar{d}$	s_1, s_2 : aux. data	$\bar{d} = d(q, o_1), O_q = \phi$ $\bar{c} = \odot(q, \bar{d})$
3	$s_1 : \bar{d}$	$s_1 : o_3, o_2, o_1, r_{1,3}, r_{1,4}, r_{1,2}$	$\bar{d} = d(q, o_1)$ $O_q = \{o_3, o_2, o_1\}$ $\bar{c} = (A_1 \cup r_{1,2} \cup r_{1,3} \cup r_{1,4}) \cap \odot(q, \bar{d})$
4	$s_2 : M(\bar{c})$	$s_2 : o_6, o_1$	$\bar{d} = d(q, o_2)$ $O_q = \{o_3, o_6, o_2\}$ $\bar{c} = \odot(q, \bar{d}) \cap r_{1,4}$

BSDs. On the contrary, the site-based approach is more straightforward and usually incurs less computing cost.

4.3 Analysis

Since the proposed site-based approach collects little auxiliary data for querying distributed BSDs, the overall transmission size can be represented as:

$$\begin{aligned} cost(q, H_n) &= \sum_{j=1}^{n} cost(q_{j-1}, D_{i_j}) \\ &\le \sum_{j=1}^{n} (size(M(q_{j-1} \cap A_{i_j})) + size(M(q_{j-1} \cap A_{i_j}), D_{i_j})) \\ &\le \sum_{j=1}^{N} (size(q) + size(q, D_{i_j})). \end{aligned}$$

In other words, it always incurs equal or less transmissions compared to the baseline CT if $size(c, D_i)$ is properly estimated in *SiteQueryRange*. Moreover, the area-based approach in general is superior to the the site-based approach considering that the requesting order of sites for each subquery of q is optimized. In the following, we analyze the overhead of the area-based approach, i.e., the cost of query partitioning. Theorem 3 gives the upper bounds of the storage and computing cost for generating the partitions given N δ-step overlap divisions.

Theorem 3. *Given a threshold δ, a partition of q generated by N δ-step overlap divisions has size of $O(\delta^2 \sum_{i=1}^{N} \mu_i)$, where μ_i denotes the number of overlaps which intersect the i^{th} overlap, and the computing cost is $O(\delta^2 N^2)$.*

We can see that the overhead incurred by the area-based approach is minor for small N and δ. Further, although a delicate partition of q can improve the site querying plan, the requesting queries may become unnecessarily complicated, thus lowering the performance. Therefore, small values of δ are preferred in the area-based approach. Besides, the partition of q can be simplified by just using k coverage divisions with the largest $|\tilde{c}(q, D_i)|$. More discussions about δ can be found in Section 7.

5 Distributed *k*NN Query Processing

5.1 Site-based Approach

Conventional *k*NN query processing using R-trees explores the spatial space one object at a time, whereas our proposed site-based *k*NN query processing on distributed BSDs explores a set of objects and unknown regions in each interaction with the sites to save interaction rounds. Given a *k*NN query q, the query client P probes the sites and estimates the minimum distance $\bar{d}$ that guarantees at least k objects can be found within $\odot(q, \bar{d})$. Thus, P can answer q by exploring the bounding range $\odot(q, \bar{d})$. Let $\bar{c}$ denote the currently unknown regions inside $\odot(q, \bar{d})$. Site selection and querying at each round is similar to that of range queries. The difference is that the bounding range $\odot(q, \bar{d})$ in *k*NN queries is dynamic and will be updated when new items are collected. The detailed processing algorithm is presented below. Table 5 lists the steps of site-based query processing for the exemplary *k*NN query in Fig. 3(b).

Step 1. P sends the *k*NN query q to all N sites. Each site s_i returns the minimal distance d_i which guarantees that at least k objects of D_i have equal or shorter distances than d_i from q.

Step 2. P gets $\bar{d} = \min\{d_1, \ldots, d_N\}$ and $\bar{c} = \odot(q, \bar{d})$, and sends $\bar{d}$ to all N sites. Each site s_i returns the coverage area A_i and the auxiliary data $sel_i(\odot(q, \bar{d}))$ and $que_i(\odot(q, \bar{d}))$.

Step 3. P applies *SiteSelRange*$(\bar{c})$ to select a site s_j from the candidate sites S_{cand}, and then performs *SiteQueryRange*$(\bar{c}, D_j)$ to obtain the objects O_j as well as the unknown regions R_j within $\bar{c}$. Based on the collected data, P reevaluates $\bar{d}$ and updates the answer set as follows.

- $O_q \leftarrow \{o | o \in O_q \cup O_j \wedge d(o, q) \le \bar{d}\}$
- $\bar{c} \leftarrow (\bar{c} - (A_j - \cup_{r \subseteq R_j} r)) \cap \odot(q, \bar{d})$.

P removes from S_{cand} s_j and any other site s_i with $A_i \cap \bar{c} = \phi$ from S_{cand}. This step is repeated until $S_{cand} = \phi$.

Step 4. P obtains the final answer set $T = O_q \cup \bar{c}$.

5.2 Area-based Approach

By introducing bounding ranges for *k*NN queries, we present an area-based processing algorithm similar to that of range queries. A distinct feature of the area-based approach for *k*NN queries is that the parts of the bounding ranges are explored in groups, though not all at a time, according to their distances to the query point. This is because a part closer to the query point usually contains more valuable data for query processing and should be processed earlier in order to prune the search space as early as possible. Moreover, for a *k*NN query q and the bounding range $\odot(q, \bar{d})$, the overlap division on a site s_i can be approximated by dividing the MBR of $\odot(q, \bar{d}) \cap A_i$ to simplify the computation. The detailed processing algorithm is presented below.

Step 1. P sends the *k*NN query q to all N sites. Each site s_i returns the minimal distance d_i which guarantees that at

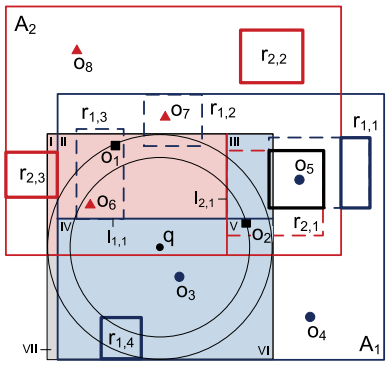


Fig. 6. Partition of the bounding range for the exemplary *k*NN query.

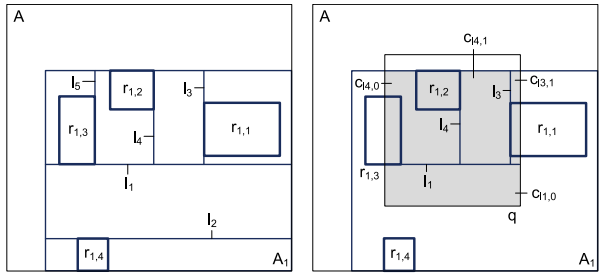


Fig. 7. Coverage-area division for the exemplary BSD D_1 and the derived overlap division for range query q. $c_{l,0}$ (or $c_{l,1}$) denotes the left/bottom (or right/up) part produced by l in A_1. (a) coverage-area division. (b) generated overlap division.

least k objects of D_i have equal or shorter distances than d_i from q.

Step 2. P gets $\bar{d} = \min\{d_1, \ldots, d_N\}$ and $\bar{c} = \odot(q, \bar{d})$, and sends $\bar{d}$ to all N sites. Each site s_i then makes a δ-step division on the $M(\odot(q, \bar{d}) \cap A_i)$, and returns A_i, the δ division lines, and the auxiliary data $sel_i(c_{i,j})$ and $que_i(c_{i,j})$ for each divided part $c_{i,j} \subseteq M(\odot(q, \bar{d}) \cap A_i)$. Fig. 6 shows the results of 1-step MBR division for the exemplary sites.

Step 3. P generates a partition of the MBR of the bounding range $\odot(q, \bar{d})$ formed by the intersections of the overlap divisions. As shown in Fig. 6, $M(\odot(q, \bar{d}))$ is divided into seven parts: c_I, c_{II}, c_{III}, c_{IV}, c_V, c_{VI}, and c_{VII}. For each part $c_{q,m}$, if $c_{q,m} \subseteq A_i$ and is contained by $c_{i,j}$, we use $sel_i(c_{i,j})$ to approximate $sel_i(c_{q,m})$. Accordingly, a list $S_{q,m}$ of sites whose coverage areas contain $c_{q,m}$ is created and sorted in decreasing order of $sel_i(c_{q,m})$.

Step 4. P computes $\bar{d} = \min_{S_{q,m} \neq \phi} d_{max}(c_{q,m}, q)$ and selects the first site of $S_{q,m}$ for each part $c_{q,m}$ with $d_{min}(c_{q,m}, q) \leq \bar{d}$, where $d_{max}(c_{q,m}, q)$ and $d_{min}(c_{q,m}, q)$ denote the maximum distance and the minimum distance from $c_{q,m}$ to q. Then, all subqueries $c_{q,m}$ assigned to the same site are merged and sent to the site for processing. Let O_i and R_i denote the objects and the unknown regions collected from s_i. P reevaluates $\bar{d}$ and updates the answer set as follows.

- $O_q \leftarrow \{o | o \in O_q \cup (\cup_i O_i) \wedge d(o, q) \leq \bar{d}\}$
- $\bar{c} \leftarrow (\bar{c} - \cup_i (c_i - \cup_{r \subseteq R_i} r)) \cap \odot(q, \bar{d})$.

For any $c_{q,m}$, if $c_{q,m} \cap \bar{c} = \phi$, P sets $S_{q,m} = \phi$; otherwise, P removes the first site of $S_{q,m}$. This step is repeated until $S_{q,m} = \phi$ for any $S_{q,m}$.

Step 5. P obtains the final answer set $T = O_q \cup \bar{c}$.

The area-based approach uses a distance metric d to decide which parts of the bounding range should form a group to be explored together. This group-based exploring scheme has the potential to reduce the transmission cost at the cost of more interaction rounds. However, experiments show that the area-based approach only incurs a slightly larger number of interaction rounds compared to the site-base approach. Table 6 lists the steps of the area-based query processing for the exemplary *k*NN query. We can see that the area-based approach collects fewer items compared to the site-based approach.

TABLE 6
Area-Based *k*NN Query Processing for Fig. 6

#	Upload	Download	Processing state
1	$s_1, s_2 : q$	$s_1 : d(q, o_1)$ $s_2 : d(q, o_1)$	$d = d(q, o_1), O_q = \phi$ $\bar{c} = \odot(q, \bar{d})$
2	$s_1, s_2 : d$	s_1, s_2 : aux. data	$d = d(q, o_1), O_q = \phi$ $\bar{c} = \odot(q, \bar{d})$
3	$s_1 : c_{III-VI}$ $s_2 : c_{I-II}$	$s_1 : o_3, o_2, r_{1,4}$ $s_2 : o_6, o_1$	$d = d(q, o_2)$ $O_q = \{o_3, o_6, o_2\}$ $\bar{c} = \odot(q, \bar{d}) \cap r_{1,4}$

6 Efficient Overlap Division

The area-based approach requires overlap division to be performed on each site for every query. This may incur excessive computing cost which may be too high for mobile devices such as sensor nodes. To address this issue, we introduce a *coverage-area division* scheme, in which the division of the coverage area is precomputed on each site before any query is received. Particularly, the coverage-area division scheme maintains three auxiliary data items for each division line l: 1) sequence ID, 2) $sel_i(c_{l,0})$ and $sel_i(c_{l,1})$, and 3) $que_i(c_{l,0})$ and $que_i(c_{l,1})$, where $c_{l,0}$ and $c_{l,1}$ are the divided parts produced by l.

When performing overlap division, the site selects the first δ division lines which contain some bounding edges of the unknown regions within the overlap and generates the division information based on the auxiliary data. That is, if a divided part c of the overlap is generated by l and contained by $c_{l,j}$ ($j \in \{0, 1\}$), we set $sel_i(c) = sel_i(c_{l,j})$ and $que_i(c) = que_i(c_{l,j})$. An example of the coverage-area division is presented in Fig. 7(a), and the first 3 selected division lines for a range query q are shown in Fig. 7(b). Compared to Fig. 4(a), the result of the overlap division is slightly different from that of the optimal 3-step overlap division. We will discuss and compare the performance of the coverage-area division and the optimal overlap division in Section 7.

7 Experiments

This section evaluates the performance of the proposed approaches with simulation experiments. All experiments were run on a machine with a 2.13 Ghz CPU and 2GB of main memory.

7.1 Spatial Data and Experimental Settings

We use three real spatial datasets, namely, the postal addresses of three metropolitan areas (New York, Philadelphia and Boston) (PNE), the roads of Germany

TABLE 7
System Parameter Settings

Param.	Value	Param.	Value
N	**20**	$\sqrt{\lvert A_i \rvert}$	$0.05, 0.1, \mathbf{0.15}, 0.2, 0.25$
p	**0.4**	$\sqrt{\lvert F_i \rvert / \lvert A_i \rvert}$	$0, \ldots, 0.3, \mathbf{0.4}, 0.5 \ldots, 1$
$\lvert R_i \rvert$	$500 \sim 1500$	$\sqrt{\lvert W \rvert}$	$0.05 \sim 0.1$
k	$300 \sim 500$		

TABLE 8
Average Overlaps for Different Sizes of Coverage Areas

$\lvert A_i \rvert$	0.05	0.1	0.15	0.2	0.25
Avg. Overlap (%)	3.56	13.6	27.8	40.9	50.5

(RGE) and the roads of Greece (RGR), to evaluate the proposed approaches.[3] The cardinalities of the three datasets are, respectively, 123,593, 30,674, and 23,268. The data space is projected on to a unit space with each dimension between $[0, 1]$. We construct N distributed BSDs from each dataset as follows. Each BSD D_i is assigned a *coverage area* A_i and a *focus area* $F_i \subseteq A_i$. The location of A_i is randomly chosen within the unit space with default size 0.15×0.15. F_i is an area in A_i that contains no unknown region. That is, all unknown regions are located within $A_i - F_i$.[4] The location of F_i within A_i is randomly chosen with average size $0.4 \times 0.4 \times |A_i|$ by default. $|R_i|$ unknown regions are generated in $A_i - F_i$. The number and locations of the objects in each dataset are prescribed by the dataset. To generate the unknown regions for the experiments, we randomly select a proportion p of the objects residing in $A_i - F_i$, clustered them into $|R_i|$ unknown regions and thereafter treat them as missing objects.

Range query and kNN query are randomly picked for execution in the experiments. The window of a range query is centered at a randomly chosen location within the unit space with average size $|W| = 0.075 \times 0.075$. For a kNN query, the query point is randomly chosen within the unit space, and k is randomly selected from the range between 300 and 500. The system parameter settings are summarized in Table 7 with default values in bold. In the following, we simply use the edge length of an area to indicate the size of the area since all areas in the experiments are assumed to be squares.

We implement both range and kNN query processing algorithms for *CT* and our two proposed distributed approaches, namely, the *site-based approach* (*SS*) and the *area-based approach*, with δ-step overlap division (*AS*δ). Moreover, we implement the state-of-art *threshold approach* (*TH*) for kNN queries and a variant of *SS*, denoted as *SR*, which is the same as *SS* except that in each interaction it selects a random site from the candidate sites and directly sends the remainder query to the selected site.

The performance of the approaches are measured with two metrics: data transmission size and number of interaction rounds. We assume that an object consists of a 8-byte ID (with type indicator to distinguish if the item is an object or unknown region), and a pair of 8-byte double-precision values indicating the x and y coordinates. Further, the location of an unknown region is 32 bytes indicating the locations of its opposite corners. Likewise, a remainder query is encoded by a list of the coordinates of its vertices, each of which is an 8-byte double-precision value. The algorithms under evaluation were implemented, and the data transmission volume between the query client and sites was logged to obtain the total transmission cost of a method. The final performance figures are averaged from the results of 200 randomly selected spatial queries.

3. The datasets can be downloaded at www.rtreeportal.org/spa-tial.html

4. This simulates the situation that a sensor can detect the locations of objects in its vicinity but is uncertain about objects beyond its vicinity.

7.2 Overall Performance

In this subsection, we investigate the performance of spatial query processing on distributed BSDs by varying the coverage size of the BSDs. Since all BSDs are confined to a unit space, when the coverage areas are large, the BSDs will have large overlaps as well. Table 8 shows the the average overlap of the 20 BSDs at different coverage sizes, which indicates the average proportion of the coverage area of a BSD covered by another BSD. As our experiments show, when the overlap is small (in the extreme case, the BSDs could be disjoint), the performance of *CT* and our proposed approaches become similar. However, when the overlap is large, the result from one BSD decides which site(s) to be processed next. A good site selection can significantly reduce the remainder query and hence the number of interaction rounds.

Figs. 8 and 9 show the data transmission sizes of various approaches for range queries and kNN queries, respectively. We can see that our proposed approaches *SS* and *AS*δ clearly outperform *CT* and *SR* for both range and kNN queries. Take the range queries on PNE with default setting as an example. *SS* and *AS*3 save, respectively, 49.1% and 69.1% of the transmission size compared to *CT*. The superiority of our proposed approaches is even more significant for kNN queries. Under PNE, *SS* and *AS*1 save, respectively, 89.8% and 91.5% of the transmission size compared to *CT*. In particular, *SS* always incurs less transmissions than *SR*, which is up to 72.7% saving for range queries and 54.1% saving for kNN queries. It verifies the effectiveness of our proposed site selection and site querying functions. Without these functions, *SR* may even perform much worse than *CT*, e.g., as can be observed in Fig. 8(a). Further, *AS*δ outperforms *SS* for both range and kNN queries, saving, respectively, up to 39.2% and 33.9% of transmission size for range and kNN queries. It again verifies that our area-based processing approach is better than the site-base approach in terms of transmission size for querying distributed BSDs.

For kNN queries, *AS*δ always outperforms the state-of-art approach *TH*. However, since *SS* and *SR* collects objects and unknown regions in larger areas than necessary to reduce the number of interaction rounds, they sometimes incur more transmission than *TH* (e.g., in RGE with coverage area 0.1). As the size of the coverage area grows, the superiority of our proposed approaches over *CT* and *TH* is more significant. This is because as the overlaps between

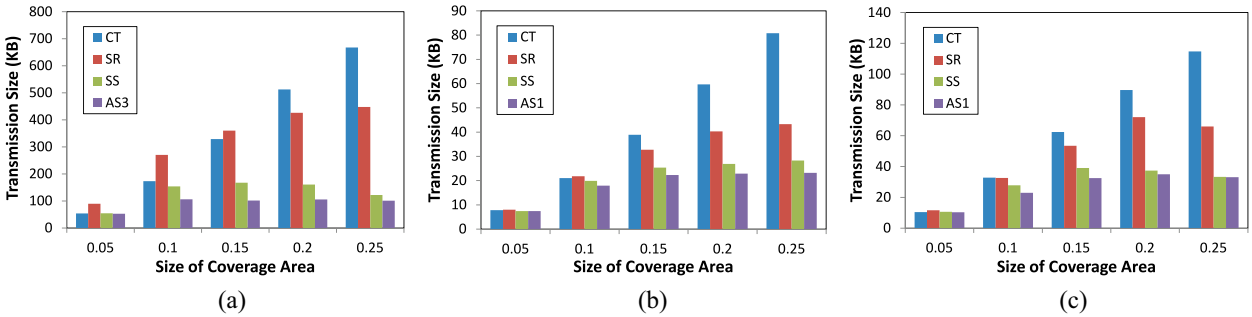


Fig. 8. Data transmission sizes for range queries. (a) PNE. (b) RGE. (c) RGR.

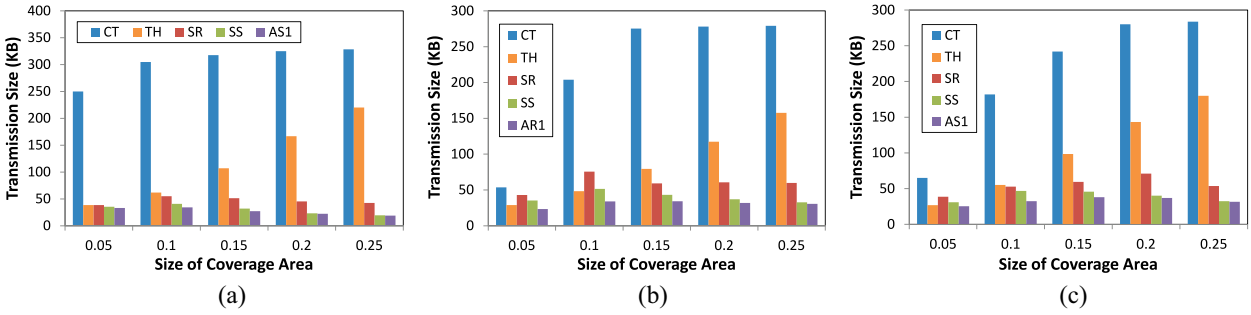


Fig. 9. Data transmission sizes for *k*NN queries. (a) PNE. (b) RGE. (c) RGR.

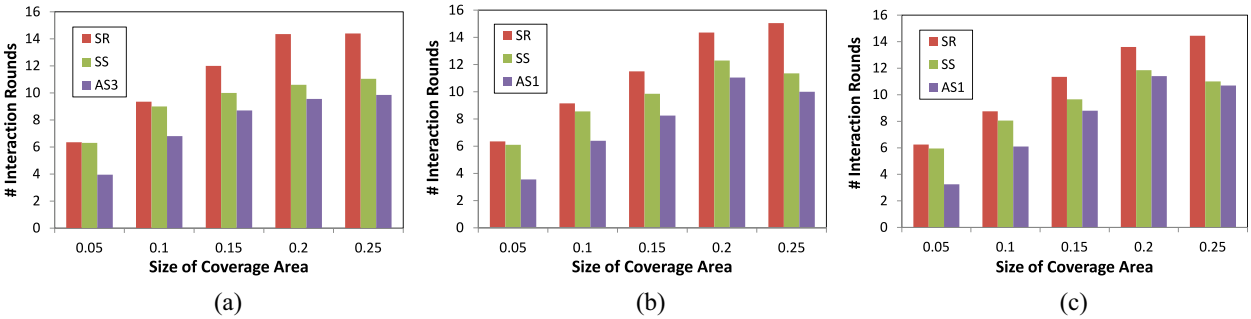


Fig. 10. Numbers of interaction rounds for range queries. (a) PNE. (b) RGE. (c) RGR.

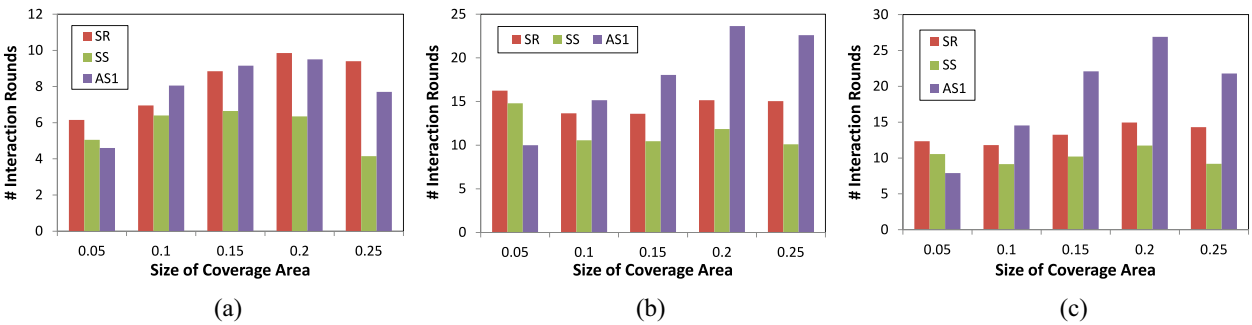
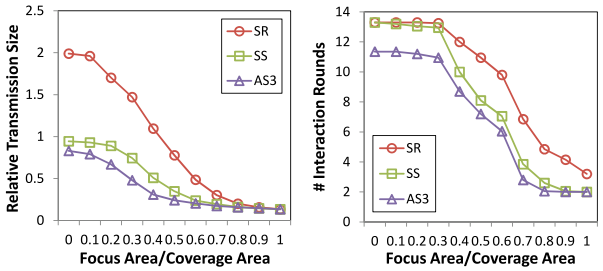


Fig. 11. Numbers of interaction rounds for *k*NN queries. (a) PNE. (b) RGE. (c) RGR.

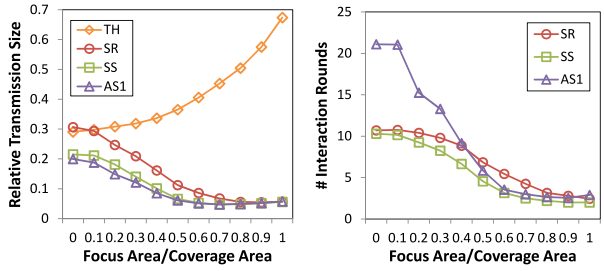
BSDs increase, the objects and unknown regions in the overlapping areas may be stored in more than one site, hence increasing the amount of redundancy data retrieved. *SR*, *SS* and *ASδ* can effectively reduce redundant transmission but *CT* and *TH* cannot. Based on the same reason, we can observe that the transmission size of *CT* and *TH* persistently increases while our proposed approaches reach the maximum at the middle. All of the approaches (except *CT* for *k*NN queries) achieve similar performance when the coverage area is 0.05 because the BSDs nearly have no overlaps and hence has little data redundancy between the BSDs (see Table 8).

Figs. 10 and 11 show the numbers of interaction rounds of the proposed approaches for range queries and *k*NN



(a) (b)

Fig. 12. Performance comparison for range queries w.r.t. various focus areas under PNE. (a) Relative transmission size. (b) Number of interaction rounds.



(a) (b)

Fig. 13. Performance comparison for *k*NN queries w.r.t. various focus areas under PNE. (a) Relative transmission size. (b) Number of interaction rounds.

queries, respectively. Fig. 11 does not include *TH*'s since its numbers of interaction rounds are too large to be shown together with *SR*, *SS* and *AS*1.[5] We can see that *SS* always incurs fewer transmission rounds than *SR* for both range and *k*NN queries. Moreover, *ASδ* requires fewer interaction rounds than *SS* and *SR* for range queries. However, for *k*NN queries, *ASδ* sometimes incurs more interaction rounds because of its group-based exploring scheme, which aims to reduce data transmission by first exploring the nearby partitions instead of all of the partitions. In conclusion, both the site-based and area-based approaches require significantly fewer rounds of interactions than *TH* for *k*NN queries.

7.3 Impacts of the Sizes of the Focus Areas

Query performance is affected by the proportions of unknown regions in the BSDs, because unknown region resolution among the BSDs is most costly. This section investigates its impact by adjusting the sizes of the focus areas. We will first study the performance of range queries. Since the performance of *CT* does not depend on the size of the focus area, we choose it as the baseline and show the relative transmission size of the approaches against the transmission size incurred by *CT*. Fig. 12 shows the performance of the proposed approaches for various sizes of the focus areas under PNE.[6] From the figure, we can see that the relative transmission sizes and the numbers of interaction rounds of all three approaches decrease as the average size of the focus areas grows. When the focus-to-coverage area ratio is near one, the unknown regions are nearly zero and the query processing costs are spent mostly on the retrieval of objects from the BSDs. At the other end when the ratio is near zero, unknown regions are large and the cost of unknown region resolution is large. We can see that the query processing costs are flat at both ends, showing that the proposed approaches are effective in both redundant object retrieval and unknown region resolution. Among the three methods, the performance of *ASδ* is better than that of *SS*, which in turn is better than that of *SR* for both transmission size and the number of interaction rounds, no matter how large the focus areas are. Notice that the gap of the transmission sizes between *ASδ* and *SS* (*SR*) is largest in the middle. This is because the partitioning scheme of *ASδ* is most effective when half of the area being partitioned are unknown regions.

For *k*NN queries, Fig. 13 shows the performance of the proposed approaches against various sizes of the focus areas under PNE. We can observe that the transmission sizes and the numbers of interaction rounds of the proposed approaches follow the same trend as for range queries. The only difference is that *ASδ* requires more rounds of interaction for small focus areas because of the fine grouping and exploration of unknown regions for the sake of reducing the total transmission cost. This number decreases quickly to that of *SS* as the size of focus area grows because all divided parts of the bounding region tend to be explored at the same time. On the contrary, *TH* incurs more transmission costs and rounds of interactions (ranging from 1891 to 3340) as the focus area expands, which also justifies the superiority of the proposed approaches. Similar results can also be found under RGE and RGR.

7.4 Coverage-Area Division and Impacts of δ

This subsection investigates the impact of overlap division on the performance of the area-based approaches. Besides the optimal division scheme, we implement two other overlap division schemes: 1) coverage-area division scheme proposed in Section 4.2 (denoted as ACδ), and 2) uniform division scheme which divides the overlap into $\delta+1$ equal rectangles along the x-axis (denoted as ARδ). Figs. 14 and 15 show the transmission sizes of these area-based approaches for range queries and *k*NN queries, respectively. We can see that *ASδ* outperforms the other two approaches in all three datasets. This verifies that the proposed overlap division scheme can effectively improve the distributed query performance on BSDs. Moreover, *ACδ* achieves a performance that is quite close to *ASδ*. Therefore, it is a good substitute for *ASδ* when limited computing power is available on the sites.

The transmission sizes of the three area-based approaches for both range and *k*NN queries do not always decrease as δ grows. Particularly, they first decrease and then increase after some point (e.g., $\delta = 3$ in PNE for range queries), i.e., there exists an optimal value of δ for the area-based approaches. However, for small values of δ as shown in Figs. 14 and 15, the performance variation

5. *TH*'s numbers of interaction rounds for *k*NN queries range from 742 to 4394 for PNE, 433 to 2479 for RGE and 400 to 2854 for RGR.

6. The same performance behavior was observed for all three datasets, but we only show the result for PNE, which is the largest dataset, because of space limitation.

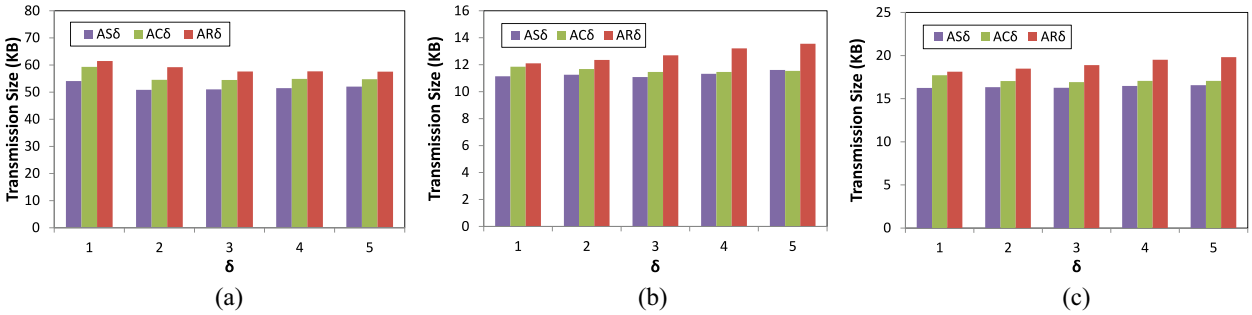


Fig. 14. Data transmission sizes of the area-based approaches for range queries, varying δ. (a) PNE. (b) RGE. (c) RGR.

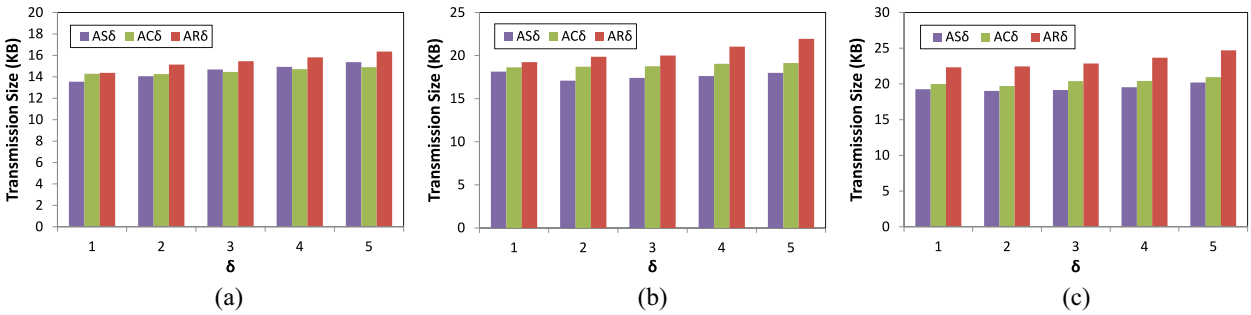


Fig. 15. Data transmission sizes of the area-based approaches for *k*NN queries, varying δ. (a) PNE. (b) RGE. (c) RGR.

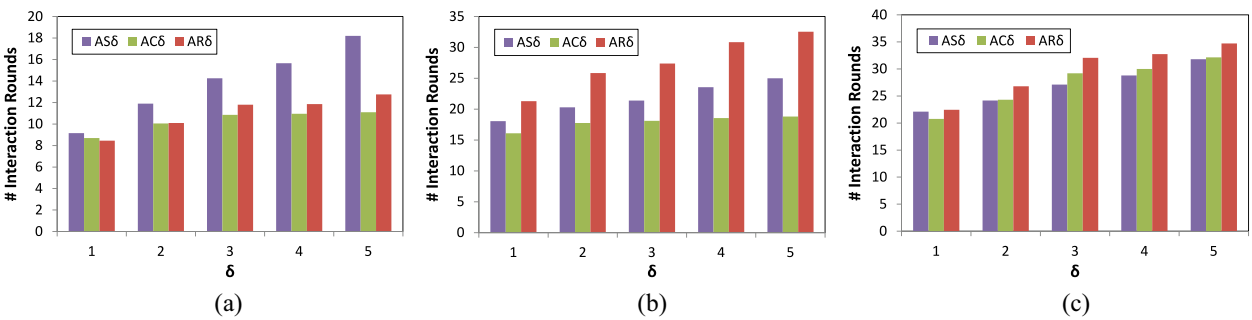


Fig. 16. Numbers of interaction rounds of the area-based approaches for *k*NN queries, varying δ. (a) PNE. (b) RGE. (c) RGR.

of $ASδ$ and $ACδ$ is insignificant (while ARδ does not guarantee this property) but their transmission sizes are still lower than other approaches such as *SS*. Therefore, little effort is required to obtain the proper value for δ. Finally, Fig. 16 shows the numbers of interaction rounds for *k*NN queries (the results for range queries are omitted due to space limitation). We can see that all of the three approaches with the optimal δ require about the same number of interaction rounds. Beside, $ACδ$ is always superior to ARδ with different δ.

8 Conclusion

This paper addresses the problem of querying distributed BSDs, which contain known objects and unknown regions containing zero or more objects. We develop the notion of unknown region resolution to reduce the size of unknown regions in the query result. We proposed two distributed query processing approaches, namely, the site-based approach and the area-based approach, for range and *k*NN query processing on BSDs. They both aim at obtaining the query result without exhaustively searching all BSDs by devising effective site selection and site querying functions. For the area-based approach, we proved the optimal area division and developed efficient heuristics that achieves a performance close to the optimal overlap division at low computing cost during query processing. The experimental results show that our proposed approaches are more cost-efficient than existing and baseline approaches.

In our future research, we plan to extend the definition of BSDs when the lower and upper bounds of missing objects in the unknown regions are known. We also plan to study query processing techniques when the sites are connected by a P2P network, which is common for mobile sensors.

References

[1] R. Cheng, S. Prabhakar, and D. V. Kalashnikov, "Querying imprecise data in moving object environments," in *Proc. ICDE*, 2003, pp. 723–725.

[2] H. Hu, J. Xu, and D. L. Lee, "A generic framework for monitoring continuous spatial queries over moving objects," in *Proc. SIGMOD*, Baltimore, MD, USA, 2005, pp. 479–490.

[3] X. Dai, M. L. Yiu, N. Mamoulis, Y. Tao, and M. Vaitis, "Probabilistic spatial queries on existentially uncertain data," in *Adv. Spatial Temporal Databases*, Angra dos Reis, Brazil, 2005, pp. 400–417.

[4] R. Li, B. Bhanu, C. Ravishankar, M. Kurth, and J. Ni, "Uncertain spatial data handling: Modeling, indexing and query," *Comput. Geosci.*, vol. 33, no. 1, pp. 42–61, 2007.
[5] H. Djidjev and M. Potkonjak, "Coverage problems in sensor networks," *Intelligent Sensor Networks: Across Sensing, Signal Processing, and Machine Learning*, CRC Press, 2012, pp. 459–482.
[6] J. Cortes, S. Martinez, and F. Bullo, "Spatially-distributed coverage optimization and control with limited-range interactions," *ESAIM, Control Optim. Calc. Var.*, vol. 11, no. 4, pp. 691–719, 2005.
[7] R. Tan, G. Xing, B. Liu, J. Wang, and X. Jia, "Exploiting data fusion to improve the coverage of wireless sensor networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 2, pp. 450–462, Apr. 2012.
[8] B. Babcock and C. Olston, "Distributed top-k monitoring," in *Proc. SIGMOD*, 2003.
[9] P. Cao and Z. Wang, "Efficient top-k query calculation in distributed networks," in *Proc. 23rd PODC*, 2004.
[10] S. Michel, P. Triantafillou, and G. Weikum, "KLEE: A framework for distributed top-k query algorithms," in *Proc. 31st Int. Conf. VLDB*, Trondheim, Norway, 2005.
[11] F. Li, K. Yi, and J. Jestes, "Ranking distributed probabilistic data," in *Proc. SIGMOD*, Providence, RI, USA, 2009, pp. 361–374.
[12] E. Tanin, A. Harwood, and H. Samet, "A distributed quadtree index for peer-to-peer settings," in *Proc. ICDE*, 2005.
[13] H. V. Jagadish, B. C. Ooi, Q. H. Vu, R. Zhang, and A. Zhou, "VBI-Tree: A peer-to-peer framework for supporting multi-dimensional indexing schemes," in *Proc. 22nd ICDE*, 2006.
[14] J. Wang, S. Wu, H. Gao, J. Li, and B. C. Ooi, "Indexing multi-dimensional data in a cloud system," in *Proc. SIGMOD*, Indianapolis, IN, USA, 2010.
[15] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. SIGMOD*, 1984.
[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[17] R. Finkel and J. Bentley, "Quad trees: A data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.
[18] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles," in *Proc. SIGMOD*, 1990.
[19] M. L. Lee, W. Hsu, C. S. Jensen, B. Cui, and K. L. Teo, "Supporting frequent updates in R-trees: A bottom-up approach," in *Proc. 29th Int. Conf. VLDB*, Berlin, Germany, 2003.
[20] X. Ma, Q. Pan, and M. Li, "Integration and share of spatial data based on web service," in *Proc. 6th IEEE Int. Conf. PDCAT*, 2005, pp. 328–332.
[21] M. Essid, O. Boucelma, F. Colonna, and Y. Lassoued, "Query processing in a geographic mediation system," in *Proc. ACM Int. Symp. Adv. GIS*, Washington, DC, USA, 2004.
[22] A. Cali, D. Lembo, and R. Rosati, "Query rewriting and answering under constraints in data integration systems," in *Proc. 18th IJCAI*, 2003, pp. 16–21.
[23] A. Cuzzocrea and A. Nucita, "Enhancing accuracy and expressive power of range query answers over incomplete spatial databases via a novel reasoning approach," *Data Knowl. Eng.*, vol. 70, no. 8, pp. 702–716, 2011.
[24] S. Nepal and M. V. Ramakrishna, "Query processing issues in image (multimedia) databases," in *Proc. 15th ICDE*, Sydney, NSW, Australia, 1999, pp. 22–29.
[25] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. SIGCOMM*, San Diego, CA, USA, 2001.
[26] S. Ratnasamy, P. Francis, M. Handley, and R. Karp, "A scalable content-addressable network," in *Proc. SIGCOMM*, San Diego, CA, USA, 2001.
[27] M. Demirbas and H. Ferhatosmanoglu, "Peer-to-peer spatial queries in sensor networks," in *Proc. 3rd Int. Conf. P2P*, 2003.
[28] H.-Y. Kang, B.-J. Lim, and K.-J. Li, "P2P spatial query processing by delaunay triangulation," in *Proc. 4th Int. Conf. W2GIS*, Goyang, Korea, 2004, pp. 136–150.
[29] Y. Xu, W.-C. Lee, J. Xu, and G. Mitchell, "Processing window queries in wireless sensor networks," in *Proc. ICDE*, 2005.

Qijun Zhu received the B.Sc. and the M.Sc. degrees in computer science from Tianjin University, China, in 2004 and 2007, respectively. He is currently a Ph.D. candidate in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. His current research interests include mobile data management, distributed query processing, pervasive computing, information retrieval, and fault tolerant computing.

Dik Lun Lee received the Ph.D. degree in computer science from the University of Toronto, Canada. He is a Professor in the Department of Computer Science and Engineering at the Hong Kong University of Science and Technology. He was the Founding Conference Chair for the International Conference on Mobile Data Management and served as the Chairman of the ACM Hong Kong Chapter in 1997. His current research interests include information retrieval, search engines, mobile computing, and pervasive computing.

Wang-Chien Lee is an Associate Professor of Computer Science and Engineering at Pennsylvania State University, University Park, PA, USA. After receiving the Ph.D. degree from the Computer and Information Science Department, Ohio State University, Columbus, OH, USA, he worked as a Principal Member of the Technical Staff at Verizon/GTE Laboratories, Inc. Dr. Lee leads the Pervasive Data Access (PDA) Research Group at Penn State University to pursue cross-area research in data management, pervasive/mobile computing, and networking.