Optimal k-Constraint Coverage Queries on Spatial Objects

Chuanfei Xu **Yanqiu Wang** **Yu Gu** **Shukuan Lin** **Ge Yu**

College of Information Science and Engineering,
Northeastern University,
PO Box 135, Shenyang, China 110819
Email: {xuchuanfei, wangyanqiu, guyu, linshukuan, yuge}@ise.neu.edu.cn

Abstract

Given a set of spatial objects, our task is to assign all the objects to the minimum number of service sites and to find the regions for building these service sites. Each service site has a coverage region (i.e., an area of service) and a capacity (i.e., a maximum number of objects it can serve, called k-constraint). The service sites can provide service for objects located within the coverage regions. Aiming at this problem, we propose a novel kind of spatial queries, called *Optimal k-Constraint Coverage* (*OCC*) queries. An *OCC* query returns some *feasible regions* such that setting up the minimum number of service sites within these regions will guarantee that all the spatial objects can be served. Furthermore, an optimal coverage scheme to assign the objects to these service sites is retrieved by this query as well. Due to the capacity constraints, objects located within the coverage region of a service site may not be assigned to one service site. Therefore, the cost of searching an optimal coverage over all possible coverage schemes becomes prohibitive. To answer *OCC* queries efficiently, we devise a general query framework, which provides two solutions to cope with *OCC* query processing. The naive solution only returns a local optimum without insuring the minimum number of service sites. To improve it, the other solution called *Optimal Coverage Algorithm* (*Opt-C*) is proposed to retrieve an optimal coverage scheme. During the procedure, we present refinement methods for reducing intermediate results of *OCC* queries to improve the efficiency. The performance of the proposed methods is demonstrated by the extensive experiments with both synthetic and real datasets.

Keywords: Optimal Coverage; Minimum number; k-Constraint; Opt-C Algorithm

1 Introduction

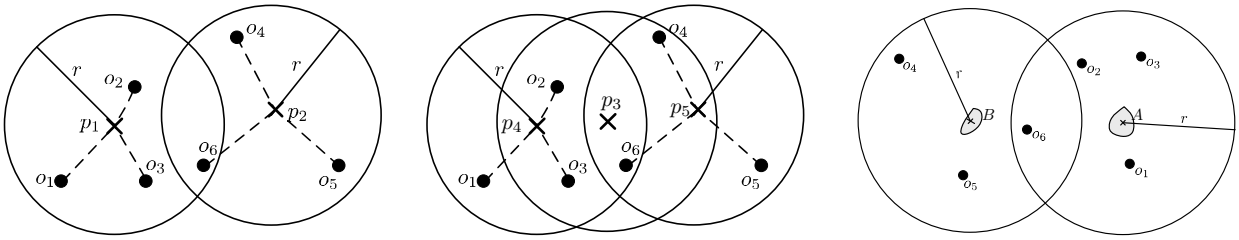
Optimal location queries have gained much research attention (Zhang et al. 2006, Cabello et al. 2005, Xiao et al. 2011) in many real-world applications. Given a set of objects, an optimal location query returns the "best location"such that setting up a service site at this location guarantees the maximum number of objects by proximity. In some cases, the "best location"is denoted by a region (Wong et al. 2009, Zhou et al. 2011) in which any point can be an optimal location.

For existing optimal location query techniques, however, some significant improvements are required when we consider building multiple service sites. For example, we wish to build some service sites to provide assistance for people in the disaster area. To aid people on time, the distance between each person and his/her assigned service site must be not more than a given distance r. For each site, the medical supplies are prepared for at most k people. To minimize the cost, we require a query which can return some regions such that setting up the minimum number of service sites would provide aid for all the persons.

The objective of the above problem we are investigating is serving all the objects (people) with the minimum number of service sites. Compared to optimal location queries, this problem equals to maximizing the average number of served objects for each service site. In addition, the spatial assignment technique also has attracted much research attention (Leong et al. 2008, 2010) in several application domains. This optimized spatial assignment desires an optimal assignment between objects and a set of given service sites. It focuses on minimizing the cost between objects and fixed number of service sites while is unable to minimize the number of service sites.

For our problem, the coverage regions of service sites may have irregular shapes. For ease of presentation, we assume that the coverage region of any service site p is a circular disk with the center at p and the radius r. Moreover, p can provide service for arbitrary k objects within its coverage region. Figure 1(a) illustrates that two service sites p_1 and p_2 serve a set of spatial objects $\{o_1, o_2, o_3, o_4, o_5, o_6\}$. Suppose that the capacity of p_1 and p_2 ($p_1.k$ and $p_2.k$) are both set as 4. As shown in the figure, $\{o_1, o_2, o_3\}$ can be assigned to p_1 and $\{o_4, o_5, o_6\}$ can be assigned to p_2. Another available attempt to solve this problem is illustrated in Figure 1(b). In this solution, the service site p_3 serves the maximum number of objects, which is a result of the optimal location query. However, it is not the optimal coverage for all the objects with the lowest number of service sites since other two service sites p_4 and p_5 must be set up to serve o_1 and o_5, respectively.

Aiming to solve the above variant of optimal location query problem, we propose a novel kind of spatial queries called *Optimal k-Constraint Coverage* (*OCC*) queries which can retrieve a series of regions such that setting up the minimum number of service sites within the regions would guarantee to provide service for all the given objects, where each service site subjects to the capacity constraint (i.e., a maximum number of objects it can serve, called k-constraint). Given a set of spatial objects $\mathcal{O}$, *OCC* queries are particularly



(a) An optimal coverage

(b) Coverage the maximum number of objects for a site

(c) The regions for building service sites

Figure 1: Examples on *OCC* queries

useful for the optimal assignment resource that ensures to maximize the average number of objects covered by each service site. Hence, this type of queries are important to a wide range of applications, such as Location-Based Services, the best locations choice and so on. In these applications, the service sites can be set up in the whole space $\mathcal{U}$ instead of some candidate points. In essence, there is a common problem such as "where are the best regions to build the minimum number of service sites"in these applications. Note that the results of *OCC* queries are the regions for building the service sites and an optimal coverage scheme to assign the spatial objects to these service sites located at any points within these regions.

In Figure 1(c), any two service sites can be built at any locations of the two shadow regions to serve $\{o_1, o_2, o_3\}$ and $\{o_4, o_5, o_6\}$ respectively, where the capacity constraint of each service sites is set as 4. Since the total number of the objects is 6, at least two sites are needed to serve $\{o_1, o_2, o_3, o_4, o_5, o_6\}$. The shadow regions are the results of the *OCC* query over these objects. Besides, there are some other coverage schemes that all the objects are assigned to two service sites (e.g., $\{o_1, o_2, o_3, o_6\}$ are assigned to the service site set up in shadow region A, and $\{o_4, o_5\}$ are assigned to the service site set up in shadow region B). Our task is to return an optimal coverage scheme over all possible coverage schemes which denote assignments between objects and service sites. Along with the increasing number of served objects, the number of available schemes may become prohibitive. Therefore, the *OCC* query evaluation raises serious practicality concerns for the volume of objects in realistic settings.

This work focuses on *OCC* query processing, which has two main challenges. (i) How to retrieve the regions such that setting up service sites within these regions will guarantee to provide service for all given objects. (ii) How to choose an optimal coverage scheme that the spatial objects can be assigned to the minimum number of service sites while insuring the capacity constraints. Aiming at the above challenges, we propose a general query framework for answering the *OCC* queries simply and elegantly. Consider a set of spatial objects $\mathcal{O}$=$\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\}$. We will illustrate the process of *OCC* query on these objects in the rest of the paper. In detail, our contributions are summarized as follows.

- We formally define a novel kind of spatial queries, called *Optimal k-Constraint Coverage* (*OCC*) queries, which can choose an optimal coverage scheme that aims to assign all the given objects to the minimum number of service sites, and retrieve some regions to build these service sites for serving all given objects while insuring the capacity constraints of service sites.
- To answer *OCC* queries efficiently, we first propose a local optimum algorithm to serve objects located in each coverage set with the lowest number of service sites.
- To improve the local optimum algorithm, we present another approach for choosing an optimal coverage to assign all objects to the minimum number of service sits, which guarantees that this scheme is global optimum. During this process, refinement methods are proposed to further improve the efficiency.
- We evaluate the performance of our methods through extensive experiments with real and synthetic datasets.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 defines the *OCC* query processing formally. Section 4 describes the query framework for answering *OCC* queries. Section 5 provides some efficient and effective methods for answering our queries in details. Section 6 gives the experimental results and Section 7 concludes this paper.

2 RELATED WORK

2.1 Spatial Queries

There is a large volume of previous works (Prasad et al. 1998, Chen et al. 2005, Beskales et al. 2008, Lian & Chen 2009, Ishikawa et al. 2009, Deng et al. 2009, Nutanong et al. 2008) devoting to spatial query processing in last few years. Especially, this trend has led to the development of spatial database management for spatial queries. Joao Rocha Junior et al. (Rocha-Junior et al. 2010) address the top-k spatial preference query which returns a ranked set of the k best data objects based on the scores of feature objects in their spatial neighborhood. They map the pairs of data and feature objects to the distance-score space, which enables to identify the minimum subset of pairs necessary to answer any ranked spatial preference query. They also improve the efficiency by avoiding examining the spatial neighborhood of the data objects during query execution. Zhenjie Zhang et al. (Zhang et al. 2008) address the continuous k-means problem, where a k-means query returns k points in space, while guaranteeing that the average squared distance between each point in $\mathcal{P}$ and its nearest center is minimized. A novel algorithm is proposed to reduce the computation and communication costs, and a threshold is assigned to each moving object such that the object sends a location update only when it crosses the range boundary. Because these works do not consider the optimal locations (or regions) to build service sites, they are not suitable for our problem.

Furthermore, some other work focuses on finding a region such that building a new service site in this location guarantees the maximum number of objects by proximity. Sergio Cabello et al. (Cabello et al. 2005) address the RNN facility location problems which is actually the MaxBRNN problem when the optimization criteria is maximizing the number of potential customers for the new facility. This work is solved for Euclidean space. In this space, Raymond Chi-Wing Wong et al. (Wong et al. 2009) present the algorithm called Bichromatic reverse nearest neighbors problem, which focuses on finding an optimal region that maximizes the size of BRNNs and an efficient algorithm called Maxoverlap is proposed for that. They define two concepts called consistent region and maximal consistent region to efficiently find an optimal region that maximizes the size of customers by proximity, and it uses the region-to-point transformation to solve the MaxBRNN problem. Considering that a customer may use the service site which is any of his/her k nearest service site, (Zhou et al. 2011) extend the MaXBRNN to the MaxBRkNN which finds an optimal region that deploying a service site in this region attracts the maximum number of customers who would consider the site as one of their k nearest service sites.

Optimal location queries are a significant sort of spatial queries and several query approaches have been proposed for retrieving the optimal locations with respect to different semantics. Du Yang et al. (Du et al. 2005) define and investigate the optimal location problem. They focus on solving the problem in the Manhattan distance space. They retrieve objects of interest in some given order and then use a plane-sweep algorithm to identify an optimal location. Furthermore, Xia et al. (Xia et al. 2005) examine a related problem of finding the top-k most influential sites among a given set of service sites. Note that in the top-t influential sites problem, the search space is limited. This makes the top-t influential sites problem very different from the optimal location problem.

The solutions for MaxBRNN problem and optimal location queries cannot be applied to find optimal regions for multiple service sites and do not consider minimizing the number of required service sites.

2.2 Spatial Assignment

The spatial assignment problem also has attracted much research attention (Leong et al. 2008, 2010) for the optimal assignment domain. Leong Hou U (Leong et al. 2008) consider the capacity constrained assignment, where each service provider can serve at most k customers, when the total size of served customers is maximized and the total assignment cost is minimized. They propose efficient algorithms for optimal assignment that employ novel edge-pruning strategies, based on the spatial properties of the problem. Additionally, they develop approximate CCA solutions that provide a tradeoff between result accuracy and computation cost. They also (Leong et al. 2010) propose the continuous optimal assignment problem, whose objective is to construct an optimal assignment between mobile users and a set of servers and then constantly maintain it. The optimal assignment has the minimum average distance between the users and their assigned servers under constraint that each user is assigned to exactly one server and the maximum possible numbers of users are served. To solve this problem, this work first accelerates the initial assignment computation by exploiting the geometric properties of the problem, subsequently splits the problem into smaller, independent ones, and then solves them using an off-the-shelf optimal assignment algorithm.

Given a set of service sites, however, the spatial assignment problem aims at an optimal assignment of objects. Thus, it is not suitable for choosing the optimal coverage problem proposed by this paper. To the best of our knowledge, this is the first paper that is able to return the regions in which the minimum number of service sites are built to serve all the given objects.

Table 1: Commonly used symbols

Symbol	Description
$\mathcal{O}$	a set of objects
$\mathcal{C}_s$	a coverage set
n	cardinality of $\mathcal{O}$
T	number of k-constraint coverage sets
S	number of coverage sets
o_i	an object in $\mathcal{O}$
k	capacity constraint
k-$\mathcal{C}_s$	a k-constraint coverage set
$\mathcal{U}$	two-dimensional Euclidean space
p_j	a service site in $\mathcal{P}$
$\mathcal{CR}(p_j)$	coverage region of p_j
r	radius of the coverage regions
$dist()$	distance function

3 Problem Definition

In this section, we first summarize some preliminary works. Then we formally define the OCC query processing. Table 1 summarizes the commonly-used symbols in this paper.

3.1 Preliminaries

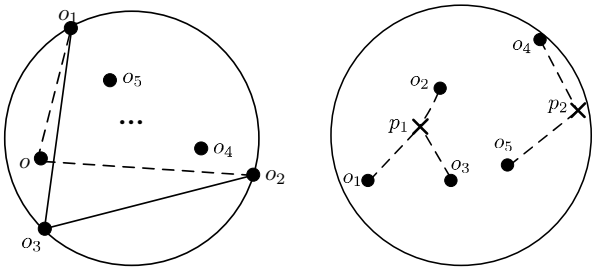
We use $\mathcal{O}$=$\{o_1, \ldots, o_i, \ldots, o_n\}$ to denote the set of spatial objects. Suppose that the capacity constraints of all service sites are fixed at k. We adopt Euclidean distance for realizing distance between objects' locations and service sites' locations in the two-dimensional Euclidean space $\mathcal{U}$, even though our techniques can also apply to higher dimensions and other distance metrics.

To provide service for these objects, all the objects need to be located within coverage regions of service sites. Furthermore, the number of objects that are assigned to each service site is not more than k. We assume that the coverage region of every service site is a circular disk with the center at p_j and the same radius r. If $dist(o_i, p_j) \leq r$, an object o_i is located within the coverage region of the service site p_j. We summarize the notion of all objects that can be located within the same coverage region as follows.

Definition 1. *Given a set of objects $\mathcal{O}$, a coverage set $\mathcal{C}_s$ is the set of objects, which satisfies (i) $\mathcal{C}_s \subseteq \mathcal{O} \bigwedge dist(o_i, p_j) \leq r \ \forall o_i \in \mathcal{C}_s$, where p_j denotes a service site; (ii) there is not any other set $\mathcal{C}'_s$ that includes $\mathcal{C}_s$.*

As mentioned earlier, some objects located within the coverage region of a service site can be in a coverage set. Next, we discuss which objects are able to be covered by one service site as follows.

Lemma 1. *For the set of objects $\mathcal{C}$, if $dist(o_i, o_{i'}) \leq \sqrt{3}r$, $\forall$ o_i, $o_{i'} \in \mathcal{C}$, all the objects in $\mathcal{C}$ can be covered by one service site.*



(a) The proof on Lemma 1 (b) k-constraint coverage sets in a coverage set

Figure 2: Coverage sets

Proof. As shown in Figure 2(a), due to $dist(o_i, o_{i'}) \leq \sqrt{3}r$, suppose that o_1, o_2 and o_3 satisfy $dist(o_1, o_2) = \sqrt{3}r \in \mathcal{C}$, $dist(o_2, o_3) = \sqrt{3}r$ and $dist(o_3, o_1) = \sqrt{3}r$. O is a circle disk which passes the locations of o_1, o_2 and o_3. For any other object o, according to the given condition, $dist(o_2, o) \leq \sqrt{3}r$ and $dist(o, o_1) \leq \sqrt{3}r$. Thus, $\angle o_1 o o_2 \geq \angle o_1 o_3 o_2 = \frac{\pi}{3}$. It follows that o resides in the circle disk O. Thereby, all the objects in $\mathcal{C}$ can be covered by one service site. □

Given a set of objects $\mathcal{O}$, all the objects are indexed by an R-tree in the space $\mathcal{U}$. According to the above lemma, we can find all the coverage sets from the set of objects $\mathcal{O}$ simply. Due to the capacity constraints, the covered objects are unable to be served by one service site. Set the capacity constraint as k, if the number of objects is more than k in a coverage set $\mathcal{C}_s$, these objects cannot be served by one service site. To ensure that some objects are served by one service site, k-constraint coverage set is described by the following definition.

Definition 2. *Given the capacity constraint k for each service site, a k-constraint coverage set k-$\mathcal{C}_s$ includes at most k objects that are in a coverage set $\mathcal{C}_s$.*

Figure 2(b) shows that a coverage set $\mathcal{C}_s$ and some of its k-constraint coverage sets. The set of objects $\{o_1, o_2, o_3, o_4, o_5\}$ can be covered by one service site. Moreover, it is not included by other coverage sets. According to Definition 1, $\{o_1, o_2, o_3, o_4, o_5\}$ is a coverage set. Assume that the capacity of service sites is 3. Thus, $\{o_1, o_2, o_3\}$ and $\{o_4, o_5\}$ are both 3-constraint coverage sets of the coverage set. Compared to coverage sets, all the objects in a k-constraint coverage set can be served by one service site.

Since objects can be served by one service site in a k-constraint coverage set, we can find some k-constraint coverage sets that satisfy

$$\bigcup_{s=1}^{T}(k\text{-}\mathcal{C}_s) = \mathcal{O}, \tag{1}$$

where T denotes the number of k-constraint coverage sets. For *OCC* queries, if we choose the minimum number of k-constraint coverage sets satisfying Eq.(1) and retrieve the regions to set up service sites for these k-constraint coverage sets, an optimal coverage scheme is returned.

3.2 Problem Formulation

To set up each service site for a k-constraint coverage set k-$\mathcal{C}_s$, we need to find the region such that any point in this region is a possible location at which the service site can be built. We next describe a region such that setting up a service site within this region can serve any object in k-$\mathcal{C}_s$, which is summarized as follows.

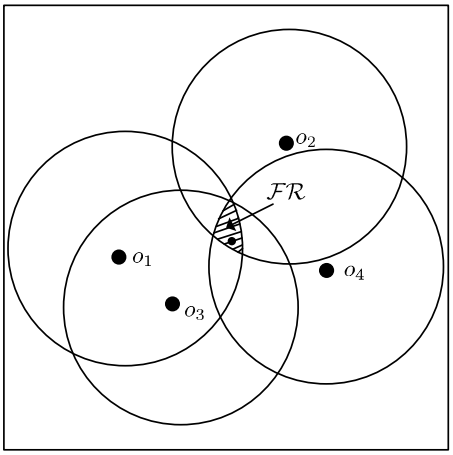


Figure 3: The feasible region of a k-constraint coverage set

Definition 3. *Given a k-constraint coverage set k-$\mathcal{C}_s$, its feasible region, denoted by $\mathcal{FR}_s$, is defined as a region which is composed of all points such that setting up a service site p at any of these points ensures $dist(o, p) \leq r$ for k-$\mathcal{C}_s$.*

The feasible region of a k-constraint coverage set can be given by the intersection among the circular disks with centers at all objects' locations. Figure 3 illustrates the feasible region denoted as shadow region for $\{o_1, o_2, o_3, o_4\}$. Since the distance between any point in the intersection region and the location of any object is not more than r, building a service site in this region can cover all the objects in a coverage set.

To minimize the cost for building server sites, all spatial objects in $\mathcal{O}$ should be served by the minimum number of server sites. In fact, we need to choose the minimum number of k-constraint coverage sets which can include each object in $\mathcal{O}$. We formally define *OCC* queries that are able to return an optimal coverage scheme that includes the minimum number of k-constraint coverage sets and the relevant feasible regions for building the minimum number of service sites to serve all the objects.

Definition 4. *Given a set of $\mathcal{O}$, an* OCC *query,* $\mathrm{OCC}(\mathcal{O}, r, k)$*, retrieves a series of feasible regions to build the minimum number of service sites for the k-constraint coverage sets which include all the objects in $\mathcal{O}$. It is formally defined as*

$$\begin{aligned}\mathrm{OCC}(\mathcal{O}, r, k) \quad &= \{< \mathcal{FR}_s, k\text{-}\mathcal{C}_s > | \forall p_j \text{ located in } \\ &\mathcal{FR}_k \text{ and } \forall o_i \in k\text{-}\mathcal{C}_s, \\ &(i) \bigcup_{s=1}^{Min}(k\text{-}\mathcal{C}_s) = \mathcal{O} \\ &(ii) \neg\exists T, T < Min \bigwedge \bigcup_{s=1}^{T}(k\text{-}\mathcal{C}_s) = \mathcal{O}\},\end{aligned} \tag{2}$$

where Min denotes the minimum number of server sites, and r is the radius of the coverage region of p_j.

As mentioned in the above definition, some regions to set up the minimum number of service sites are retrieved by the *OCC* query. Due to (ii), an optimal coverage is returned by Eq.(2). Then we will describe the framework for the *OCC* query processing.

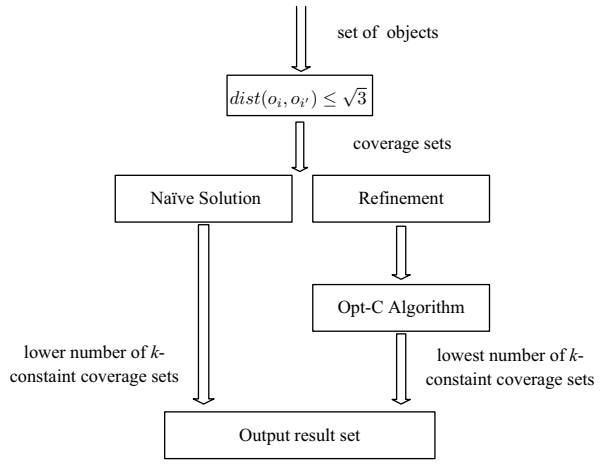


Figure 4: The general query framework for *OCC* queries

4 The Query Framework

As mentioned previously, we have presented *OCC* queries which can find some feasible regions such that setting up the minimum number of service sites within these regions would provide service for all the given objects. Each feasible region is searched by the objects in a k-constraint coverage set. According to the definition of k-constraint coverage sets, any k objects (or lower than k objects) in a coverage set can make up a k-constraint coverage set. Thereby, the key of answering *OCC* queries is finding the minimum number of k-constraint coverage sets that are able to include all the given objects. For a coverage set $\mathcal{C}_s$, hence, the number of k-constraint coverage sets approximates to $C^k_{|\mathcal{C}_s|}$, where $|\mathcal{C}_s|$ denotes the cardinality of $\mathcal{C}_s$. The cost of evaluating k-constraint coverage set for all coverage sets is time-consuming so that the cost of *OCC* queries raises serious practicality concerns in realistic settings. Aiming at this problem, we devise a general query framework for answering *OCC* queries efficiently and elegantly.

As illustrated in Figure 4, the query framework first finds all coverage sets from the given set of objects. According to Lemma 1, all the given objects are partitioned into coverage sets in which objects can be covered by one service site. Next, this framework proposes two approaches to return the coverage schemes and feasible regions for setting up service sites. For a coverage set, the naive solution wishes to insure the maximum number of service sites whose capacities are full. Therefore, this solution approach only retrieves the local optimum that insures the minimum number of service sites for a coverage set.

Aiming at this problem, the other approach is able to return a global optimum that all objects are served by the minimum number of service sites with a high-performance method. To improve the efficiency, this approach refines intermediate results for *OCC* queries.

5 Optimal k-Constraint Coverage Query Processing

In the previous sections, we have formalized *OCC* queries and devised the query framework for answering *OCC* queries. In this section, we describe *OCC* query processing in details.

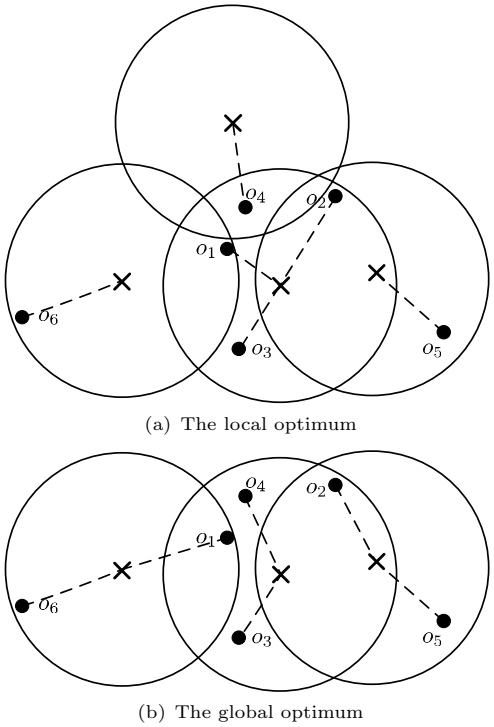


Figure 5: Different coverage schemes

5.1 Naive Solution

Based on Lemma 1, we can find which objects are able to be covered by one service site from the given set of spatial objects $\mathcal{O}$. According to the definition of coverage set, all the coverage sets are returned easily. We use R-tree to index the set of objects $\mathcal{O}$ in the two-dimensional space $\mathcal{U}$. Due to the capacity constraint of k, any service site provides service for at most k objects. If some objects are within a k-constraint coverage set, we say that these objects can be assigned to the same service site. Thereby, we find the minimum number of coverage sets which include all objects in $\mathcal{O}$ to answer the *OCC* query. The query framework gives two approaches to retrieve some k-constraint coverage sets which can cover the set of objects $\mathcal{O}$.

The main idea for the naive solution approach is insuring the maximum number of service sites whose capacities are full in a coverage set. For instance, given a set of objects $\mathcal{O} = \{o_1, o_2, o_3, o_4, o_5, o_6\}$ and $\mathcal{O} = \bigcup_{s=1}^{3} \mathcal{C}_s$ where $\mathcal{C}_1 = \{o_1, o_2, o_3, o_4\}$, $\mathcal{C}_2 = \{o_2, o_5\}$, and $\mathcal{C}_3 = \{o_1, o_6\}$. Assume that the capacity constraint of each service site is fixed at 3. For ease of presentation, the naive solution assigns objects with IDs of objects ordering (i.e., the ID of an object o_i is i). $\{o_1, o_2, o_3\}$ are assigned to a service site, which insures that the capacity of this service site is full in $\mathcal{C}_1$. As shown in Figure 5(a), three other service sites are required to serve o_4, o_5 and o_6. Hence, the naive solution approach returns four service sites to serve $\{o_1, o_2, o_3, o_4\}$ in the example. However, Figure 5(b) illustrates a better coverage scheme which needs three service sites to serve these objects, and the three k-constraint coverage sets are $\{o_1, o_6\}$, $\{o_2, o_5\}$ and $\{o_3, o_4\}$ respectively.

As mentioned previously, the naive solution only returns a local optimum that is an optimal coverage scheme for each coverage set. Using this solution, the objects can be served by the lowest number of service sites in each coverage set (e.g., in $\mathcal{C}_1$ o_1, o_2, o_3 and o_4

are serve by two service sites). However, all the given objects are served by service sites whose number may be not the lowest. The key steps of the naive solution algorithm is illustrated in Algorithm 1.

Algorithm 1: Naive Solution Algorithm

Input : $\mathcal{O} = \bigcup_{s=1}^{S} \mathcal{C}_s$ and the capacity constraint k
Output: $\{k\text{-}\mathcal{C}_1, \ldots, k\text{-}\mathcal{C}_M\}$ and $\{\mathcal{FR}_1, \ldots, \mathcal{FR}_M\}$

```
1  s ⟵ 1
2  i ⟵ 1
3  for each coverage set C_s do
4      while |C_i| ≥ k do
5          take k objects of C_s into k-C_i
6          C_i ⟵ C_i/k-C_i
7          i ⟵ i + 1
8      take the rest objects of C_s into k-C_i
9      O ⟵ O/C_s
10 for each k-C_i do
11     for each object o in k-C_i do
12         generate the circular disk with the
           center at o
13         evaluate the intersection FR_i
```

This algorithm first searches some k-constraint coverage sets whose capacities are full and a k-constraint coverage set whose capacity may not be full for a coverage set (lines 4-7). In order to ensure that objects are assigned to one service site, the algorithm deletes the objects assigned to k-constraint coverage sets (line 9). Lastly, Naive Solution Algorithm evaluates all feasible regions for k-constraint coverage sets. The naive solution can only retrieve a local optimum that is an optimal coverage scheme for each coverage set. To choose an optimal scheme for all coverage sets, we propose an efficient algorithm called Opt-C Algorithm in the next subsection.

5.2 Optimal Coverage Scheme

In this subsection, we present refinement methods for improving the efficiency of *OCC* queries. Then we discuss an optimal coverage scheme algorithm called Opt-C Algorithm.

Using the naive algorithm to answer *OCC* queries, the number of service sites that are required to be built for serving all the objects is more than that of the optimal coverage. If we choose an optimal coverage with the exhaustive method, the time cost would become prohibitive. Aiming at this problem, we first propose a method to refine coverage sets before choosing an optimal coverage, which is summarized as follows.

Lemma 2. *For OCC queries, if* $\bigcup^{m} \mathcal{C}_{s1} \subset \bigcup^{m'} (k\text{-}\mathcal{C}'_{s2}) \wedge m \geq m'$, *m coverage sets* $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ *can be refined safely.*

Proof. For number m of coverage sets, $\forall o_i$ $o_i \in \bigcup^{m} \mathcal{C}_{s1}$. According to the definition of coverage sets, o_i can be covered by number m of service sites that are set up for these coverage sets. For k-constraint coverage sets of these coverage sets, at least m service sites are required to serve all the objects in $\bigcup^{m} \mathcal{C}_{s1}$. Since $\bigcup^{m} \mathcal{C}_{s1} \subset \bigcup^{m'} (k\text{-}\mathcal{C}'_{s2})$, $o_i \in \bigcup^{m'} (k\text{-}\mathcal{C}'_{s2})$. Note that, these objects are can also be served by number m' of service sites. Due to $m \geq m'$, these objects can be served by a lower number of service sites. Therefore, for our queries, $\{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ cannot generate any optimal coverage and they should be refined safely. □

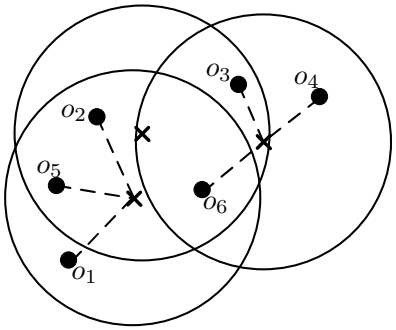


Figure 6: Refining coverage sets

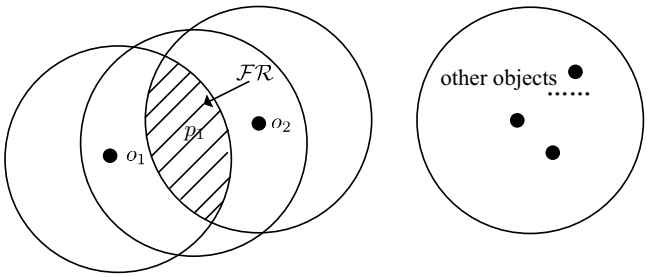


Figure 7: Refining objects

From the above refinement method, we can refine coverage sets which are unable to generate any optimal coverage schemes. This method reduces coverage schemes that are certainly not the optimal coverage schemes to improve the query efficiency. As shown in Figure 6, for the set of objects $\{o_1, o_2, o_3, o_4, o_5, o_6\}$, the coverage sets are $\{o_1, o_2, o_5, o_6\}$, $\{o_2, o_3, o_5, o_6\}$ and $\{o_3, o_4, o_6\}$. This figure illustrates a coverage scheme that objects $\{o_1, o_2, o_5\}$ are assigned to a service site and objects $\{o_3, o_4, o_6\}$ are assigned to another service site, where the capacity constraint of each service site is equal to 3. Thus, the scheme requires two 3-constraint coverage set. According to Lemma 2, $\{o_1, o_2, o_5\}$, $\{o_2, o_3, o_5, o_6\}$ and $\{o_3, o_4, o_6\}$ can be refined since any coverage scheme requires at least two service sites over this set of objects.

To further improve the efficiency of *OCC* queries, another refinement method is proposed to reduce the number of objects.

Lemma 3. *For an OCC query, if some objects are only within a coverage set* $\mathcal{C}_s$ *and the number of these objects is not more than the capacity constraint* k, *then the set of these objects is certain to be included by an optimal coverage.*

Proof. Since some objects are only within a coverage set $\mathcal{O}_k$, we suppose that a service site p_j is set up to serve these objects. Due to $|\mathcal{C}_s| \leq k$, all objects covered by p_j are within a k-constraint coverage set, denoted by $k\text{-}\mathcal{C}_k$. It follows that these objects are certain to be served by one service site. Therefore, the set of these objects is included by an optimal coverage. □

As mentioned in the above lemma, some objects satisfying this lemma can be taken into a k-constraint coverage set which are included by an optimal coverage. Thus, the number of objects that have not been assigned to any service sites is reduced, which results in the number of k-constraint coverage sets decreases.

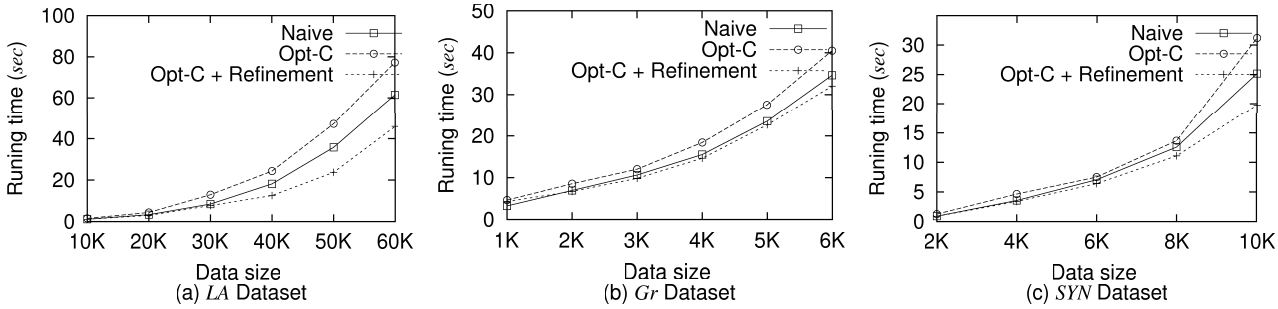


Figure 9: Effect of data size on running time

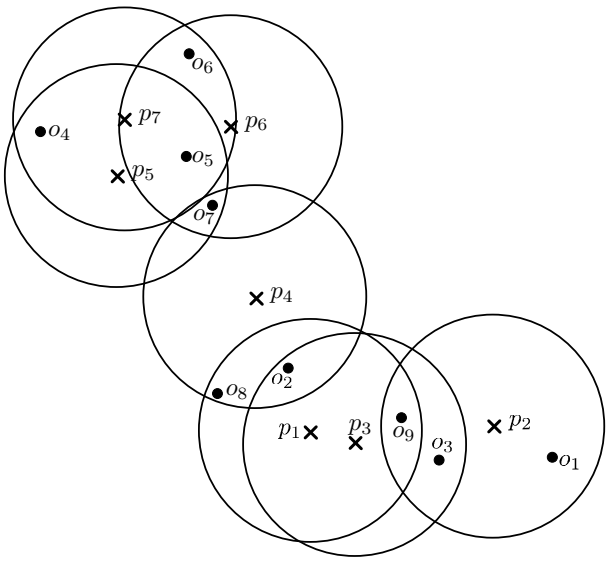


Figure 8: An example of choosing an optimal coverage scheme

Figure 7 illustrates the case in details. Objects o_1 and o_2 are only within a coverage set $\mathcal{C}_1$. Because o_1 and o_2 are not in any other coverage set, they are assigned a service p_1 which can be built at the shadow region. Using Lemma 3, some coverage sets can be the results for an optimal coverage without complex computation. Hence, the cost of OCC queries can be reduced with this refinement method. For instance, $\mathcal{C}_1$ is a result for an optimal coverage in Figure 7 because $\mathcal{C}_1$ can be represented as a k-constraint coverage set ($k = 3$).

Next, to answer OCC queries, we present an effective algorithm for retrieving an optimal coverage. To find the global optimum, all the possible coverage scheme is needed to be considered. We use Opt-C$[n, Min]$ to represent an optimal coverage scheme, which represents number Min of service sites to serve number n of objects. Note that, an optimal coverage scheme is defined as

$$\text{Opt-C}[n, Min] = \text{Opt-C}[n-m, Min-s] \bigcup (\cup_{j=1}^{s}(k\text{-}\mathcal{C}_j)), \quad (3)$$

where $\bigcup(\cup_{j=1}^{s}(k\text{-}\mathcal{C}_j))$ denotes k-constraint coverage sets including some objects which is not included by the union of a lower number of coverage sets. Thus, Opt-C$[n, Min]$ insures that number n of objects are served by the minimum number of service sites. From Eq.(3), an optimal coverage scheme Opt-C$[n, Min]$ can be evaluated with Dynamic Programming method. For any objects, this method is able to serve them with the minimum number of k-constraint coverage sets. We propose Opt-C Algorithm representing the above processing, which is illustrated in Algorithm 2.

Algorithm 2: Opt-C Algorithm

Input : $\mathcal{O} = \bigcup_{s=1}^{S} \mathcal{C}_s$ and the capacity constraint k
Output: $\{k\text{-}\mathcal{C}_1, \ldots, k\text{-}\mathcal{C}_{Min}\}$ and $\{\mathcal{FR}_1, \ldots, \mathcal{FR}_{Min}\}$

```
1  i ⟵ 1
2  for each object o_i in O do
3      take o_i into a set O'
4      if O' ⊆ ∪_{j=1}^{s}(k-C_j) then
5          r ⟵ 1
6          if r > s then
7              r ⟵ s
8          i ⟵ i + 1
9      Min ⟵ r
10 while s ≤ Min do
11     take (k-C_j) into result set
12     s ⟵ s + 1
13 for each k-C_j do
14     for each object o in k-C_j do
15         generate the circular disk with the
           center at o
16         evaluate the intersection FR_j
```

Before Opt-C Algorithm, we can use refinement methods to reduce intermediate results for OCC queries. As mentioned in Algorithm 2, this algorithm returns an optimal coverage which is the global optimum instead of the local optimum. At first, it initializes the set of objects and coverage sets. Then this algorithm insures that some objects are within the minimum number of k-constraint coverage sets (lines 4-8). Finally, it chooses an optimal coverage scheme and returns the results for an OCC query (lines 13-16).

Compared to the naive solution, Opt-C Algorithm chooses the minimum number of k-constraint coverage sets. Figure 8 illustrates an example for an OCC query. As mentioned in Section 1, we consider a set of objects $\{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8\}$. According to Lemma 1, these objects can be transformed into some coverage sets, $\mathcal{C}_1 = \{o_2, o_8, o_9\}$, $\mathcal{C}_2 = \{o_1, o_3, o_9\}$, $\mathcal{C}_3 = \{o_2, o_3, o_9\}$, $\mathcal{C}_4 = \{o_2, o_7, o_8\}$, $\mathcal{C}_5 = \{o_4, o_5, o_7\}$, $\mathcal{C}_6 = \{o_6, o_5, o_7\}$ and $\mathcal{C}_7 = \{o_4, o_5, o_6\}$. Assume that the capacity of each service sits is set as 3. The sets $\{o_4, o_5, o_6\}$ and $\{o_2, o_7, o_8\}$ are both 3-constraint coverage sets. Since $\mathcal{C}_5 \bigcup \mathcal{C}_6 \bigcup \mathcal{C}_7 \subset \{o_4, o_5, o_6\} \bigcup \{o_2, o_7, o_8\}$, $\mathcal{C}_5$, $\mathcal{C}_6$ and $\mathcal{C}_7$ can be refined

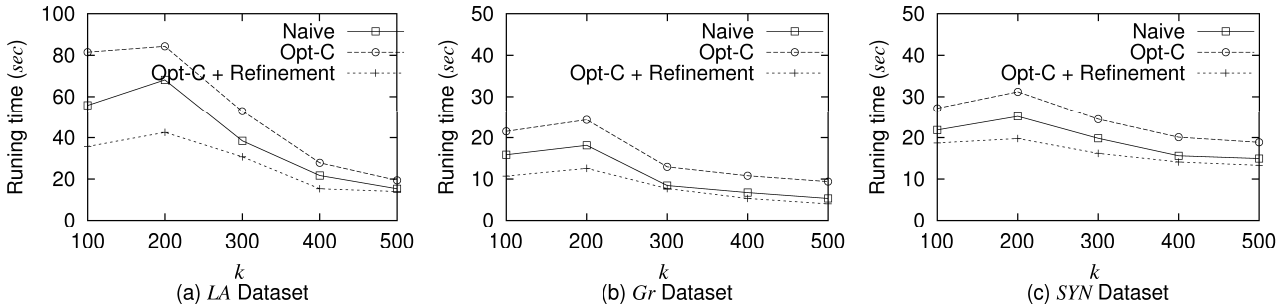


Figure 10: Effect of k on running time

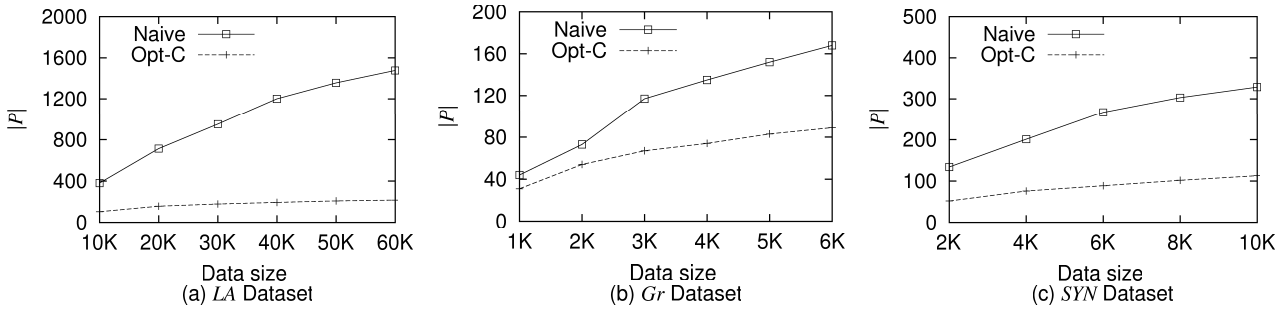


Figure 11: Effect of data size on $|\mathcal{P}|$

by Lemma 2. Next, if some objects are only within a coverage set (e.g., o_1 in $\mathcal{C}_2$), the 3-constraint coverage set $\{o_1, o_3, o_9\}$ is certain to be included by an optimal coverage due to Lemma 3. Because these coverage sets (or k-constraint coverage sets) are refined by the refinement methods, the efficiency of OCC queries would be improved considerably.

As shown in Figure 8, an optimal coverage scheme represents that all the objects can be within the minimum number of 3-constraint coverage sets (e.g., $\mathcal{O} = \{o_1, o_3, o_9\} \bigcup \{o_2, o_7, o_8\} \bigcup \{o_4, o_5, o_6\}$), where each 3-constraint coverage set can be served by a service sites. The feasible regions are retrieved by the intersections among the circular disks with the centers at the locations of objects.

6 Experiments

In this section, we present results from an extensive empirical study over several datasets to illustrate the performance of our proposed methods.

6.1 Experimental settings

All the experiments were conducted on a PC with a 2.6GHz Processor and 2GB main memory. We use two real world datasets "Los Angeles (LA)"and "Greece (Gr)", which were also available in Topologically integrated geographic encoding and referencing (tiger) system[1]. Furthermore, one synthetic dataset (SYN) was generated by Uniform distribution in the two-dimensional Euclidean space. The range of the space is standardized in 1000×1000. We used R-tree to index these datasets respectively. In order to evaluate methods exactly, we randomly generated 20 queries and computed the expectation for each query evaluation. We describe the three datasets in detail.

***LA* Dataset:** LA dataset, a two-dimensional real dataset, is composed of $60K$ geographical objects described by ranges of longitudes and latitudes. We used the centers to represent the locations of these objects.

***Gr* Dataset:** Gr dataset contains 5, 922 cities and villages in Greece. For the cities and villages, we also used the spatial objects to represent the locations of the cities and villages.

***SYN* Dataset:** SYN dataset is composed of $10K$ spatial objects. Each object is generated by Uniform distribution in the whole space.

Alternative Techniques Considered. The aim of the experiments is to study the time cost and the number of required service sites for the algorithms to solve the optimal coverage problem under various settings. To the best of our knowledge, there is no reference method to solve this optimal problem. Therefore, we contrast Opt-C Algorithm returning the global optimum with the naive solution returning the local optimum.

6.2 Experimental Results

We first study the effect of data size on the performance of Opt-C Algorithm, Opt-C Algorithm with the refinement method (Opt-C Algorithm+Refinement) and the naive solution method under different datasets. We fix the radius r of each service site's coverage region at 5, the capacity constraint k of each service site at 200, and vary the size of datasets. We observe the running time and the number of service sites that provide service for all the objects in the datasets.

Figure 9 shows the running time by the naive solution method, Opt-C algorithm and Opt-C Algorithm+Refinement algorithm in the three datasets. For any dataset, with the refinement method, Opt-C algorithm is more efficient than any other methods. The reason is that the refinement method improves the efficiency of answering OCC queries by reducing unnecessary k-constraint coverage sets.

We also observe that the effect of the capacity constraint k on the running time of these algorithms. We fix the radius r of each service site's coverage re-

[1] http://www.census.gov/geo/www/tiger/.

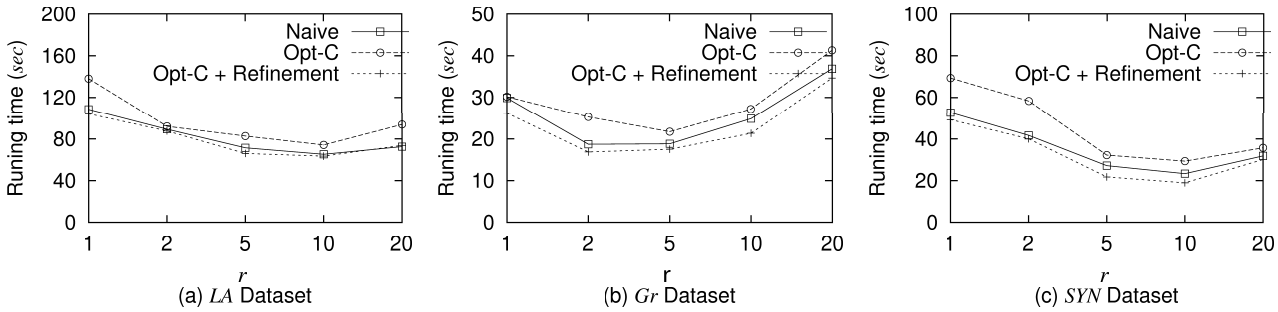


Figure 12: Effect of r on running time

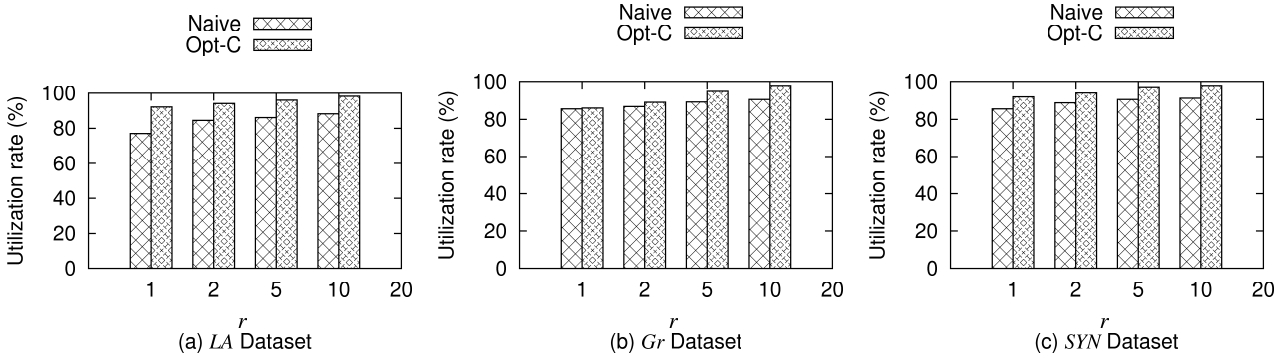


Figure 13: Effect of r on utilization rate

gion at 5. As shown in Figure 10, the running time of all the algorithms first increases and then reduces along with increasing k. This is because the number of k-constraint coverage sets for a coverage set $\mathcal{C}$ approximates $C_{\mathcal{C}}^k$ whose value first increases and then reduces along with increasing k. In addition, the running time of Opt-C Algorithm+Refinement is lower than the running time of the others.

Then we compare the number of service sites $|\mathcal{P}|$ that provides service for all the objects in these datasets with different algorithms. We fix the radius r of each service site's coverage region at 5. As shown in Figure 11 obviously, more service sites are required with the size of datasets increasing. Furthermore, Opt-C Algorithm can cover these objects with less number of service sites than that of the naive solution for all datasets. It can be explained that the naive solution which chooses a local optimum that needs more number of service sites to serve these objects. Therefore, Opt-C Algorithm is more effective than the naive solution.

Next, we examine the effect of the radius r of each service site's coverage region on the running time of different algorithms under the three datasets. We fix the capacity of each service site at 200. As described in Figure 12, for all the datasets, Opt-C Algorithm+Refinement is more efficient than the other two algorithms. When r raises from 1 to 20, the running time first decreases and then increases. The efficiency of these algorithms is lower with larger r or smaller r, since larger r results in more objects for each coverage set and smaller r generates more coverage sets, which would reduce the efficiency.

We study the utilization rate of service sites under different capacity constraints with Opt-C Algorithm and the naive solution. The rate is equal to dividing k by the average number of objects served by a service site. For any dataset, we vary the radius r from 1 to 20. Figure 13 illustrates that the utilization rate of Opt-C Algorithm is much higher than that of the naive solution. The reason is that Opt-C Algorithm returns an optimal coverage scheme that has the minimum number of service sites. Therefore, the average number of objects served by a service site is higher.

Finally, we test the performance of our refinement methods under the three datasets. The refinement methods can refine massive k-constraint coverage sets before implementing Opt-C Algorithm. We fix the radius r of each service site's coverage region at 5 and 10 respectively, and vary k from 100 to 500. As shown in Figure 14, our refinement methods reduce the number of k-constraint coverage sets for all the datasets effectively. Furthermore, the reducing rate is not lower than 30%.

7 Conclusions

In this paper, we have proposed a new type of spatial queries called *Optimal k-Constraint Coverage* (*OCC*) queries. Given a set of objects, these queries are able to return an optimal coverage scheme and a series of regions such that setting up the minimum number of service sites within these regions would guarantee all the objects to be served, where each service site has the capacity constraint and the coverage region in which the service site only can provide service for at most k objects. To answer *OCC* queries efficiently, we have presented a general query framework. In the framework, we first have proposed the naive solution algorithm for choosing an optimal coverage in each coverage set, which retrieves the local optimum. To improve it, we have addressed another algorithm called Opt-C Algorithm that can choose an optimal coverage scheme. Using this algorithm, all the objects can be included by the lowest number of k-constraint coverage sets. Furthermore, two refinement methods have been developed to reduce massive k-constraint coverage sets before answering *OCC* queries. Finally, we have studied the performance of the proposed methods through the theoretical analy-

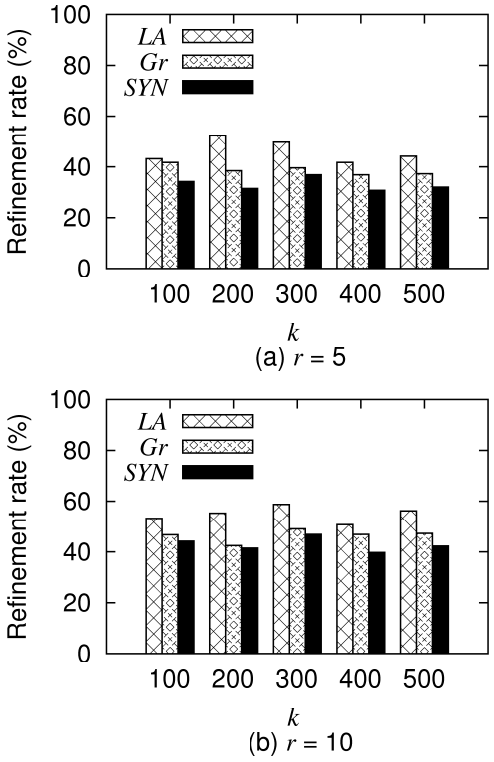


Figure 14: The refinement rate of k-constraint coverage sets

sis and extensive experiments with synthetic and real datasets.

8 Acknowledgments

The research is supported by the National Natural Science Foundation of China under Grant Nos. 60873009, 61003058 and the Fundamental Research Funds for the Central Universities No.N100604014.

References

Beskales, G., Soliman, M. A. & IIyas, I. F. (2008), 'Efficient search for the top-k probable nearest neighbors in uncertain databases', *Proc. VLDB Endow.* **1**, 326–339.

Cabello, S., Díaz-Báez, J. M., Langerman, S., Seara, C. & Ventura, I. (2005), Reverse facility location problems, *in* 'CCCG', pp. 68–71.

Chen, L., Özsu, M. & Oria, V. (2005), Robust and fast similarity search for moving object trajectories, *in* 'SIGMOD'.

Deng, K., Zhou, X., Shen, H. T., Sadiq, S. & Li, X. (2009), 'Instance optimal query processing in spatial networks', *The VLDB Journal* **18**, 675–693.

Du, Y., Zhang, D. & Xia, T. (2005), The optimal-location query, *in* 'Symposium on Large Spatial Databases', pp. 163–180.

Ishikawa, Y., Iijima, Y. & Yu, J. X. (2009), Spatial range querying for gaussian-based imprecise query objects, *in* 'ICDE', pp. 676–687.

Leong, H. U., Mouratidis, K. & Mamoulis, N. (2010), 'Continuous spatial assignment of moving users', *The VLDB Journal* **19**, 141–160.

Leong, H. U., Yiu, M. L., Mouratidis, K. & Mamoulis, N. (2008), Capacity constrained assignment in spatial databases, *in* 'SIGMOD', pp. 15–28.

Lian, X. & Chen, L. (2009), 'Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data', *The VLDB Journal* **18**, 787–808.

Nutanong, S., Zhang, R., Tanin, E. & Kulik, L. (2008), 'The v*-diagram: a query-dependent approach to moving knn queries', *Proc. VLDB Endow.* **1**, 1095–1106.

Prasad, A. S., Ouri, W., Sam, C. & Son, D. (1998), 'Querying the uncertain position of moving objects', *In Temporal Databases: Research and Practice* .

Rocha-Junior, J. a. B., Vlachou, A., Doulkeridis, C. & Nørvåg, K. (2010), 'Efficient processing of top-k spatial preference queries', *Proc. VLDB Endow.* **4**, 93–104.

Wong, R. C.-W., Özsu, M. T., Yu, P. S., Fu, A. W.-C. & Liu, L. (2009), 'Efficient method for maximizing bichromatic reverse nearest neighbor', *Proc. VLDB Endow.* **2**, 1126–1137.

Xia, T., Zhang, D., Kanoulas, E. & Du, Y. (2005), On computing top-t most influential spatial sites, *in* 'VLDB', pp. 946–957.

Xiao, X., Yao, B. & Li, F. (2011), Optimal location queries in road network databases, *in* 'ICDE', pp. 804–815.

Zhang, D., Du, Y., Xia, T. & Tao, Y. (2006), Progressive computation of the min-dist optimal-location query, *in* 'VLDB', pp. 643–654.

Zhang, Z., Yang, Y., Tung, A. K. H. & Papadias, D. (2008), 'Continuous k-means monitoring over moving objects', *IEEE Trans. on Knowl. and Data Eng.* **20**, 1205–1216.

Zhou, Z., Wu, W., Li, X., Lee, M.-L. & Hsu, W. (2011), Maxfirst for maxbrknn, *in* 'ICDE', pp. 828–839.