

Supporting Range Queries on Web Data Using k -Nearest Neighbor Search

Wan D. Bae, Shayma Alkobaisi, Seon Ho Kim, Sada Narayanappa
University of Denver
{wbae, salkobai, seonkim, snarayan}@cs.du.edu

Cyrus Shahabi
University of Southern California
shahabi@usc.edu

ABSTRACT

Access to a large volume of publicly available geospatial data on the web is hindered due to their restricted web interfaces. A typical scenario is the existence of numerous business web sites that provide the address of their branch locations through a limited “nearest location” web interface. For example, a chain restaurant’s web site such as McDonalds can be queried to find some of the closest locations of its branches to the user’s home address. However, even though the site has the location data of all restaurants in, for example, the state of California, the provided web interface makes it very difficult to retrieve this data set. We conceptualize this problem as a more general problem of running spatial range queries by utilizing only k -Nearest Neighbor (k -NN) searches. Subsequently, we propose two algorithms to cover the rectangular shape of the spatial range query with as few k -NN searches as possible. Finally, we evaluate the efficiency of our algorithms through empirical experiments.

1. INTRODUCTION

Due to the recent advances in geospatial data acquisition and emergence of diverse web applications, a large amount of geospatial data have become available on the web. Unfortunately, access to these abundant and useful geospatial data sets is only possible through their corresponding web interfaces [2]. These interfaces are usually designed for one specific query type and hence cannot support the more general access to the data. For example, one type of web interface to search for a number of “nearest locations” from a given geographical point (e.g., a mailing address) or an area (e.g., a zip code) is especially popular for accessing geospatial data on the web. It nicely serves the intended goal of quick and convenient dissemination of business location information to potential customers. However, if the consumer of the data is a computer program, as in the case of web data integration utilities (e.g., wrappers) and search programs (e.g., crawlers), such an interface may be a very inefficient way of accessing the data. To illustrate, suppose a web crawler wants to access the McDonalds web-site to retrieve all the restaurants in the state of California. Even though the site has the required information, the interface

only allows the retrieval of five locations at a time. Even worse, the interface needs the center of the search as input. Hence, the crawler needs to identify a set of *nearest location* searches that both covers the entire state of California (for completeness) and has minimum overlap between the result sets (for efficiency).

In this paper, we conceptualize the aforementioned problem into a more general problem of supporting spatial range queries using k -Nearest Neighbor (k -NN) search. Besides web data integration and search applications, the solution to this more general problem is beneficial for other application domains in the areas of sensor networks, online maps and ad-hoc mobile networks.

Suppose that we want to find the locations of all the points in a given range (rectangle) and the only available interface is a k -NN search with a fixed value of k . Hence, the only parameter that we can vary is the query point for the k -NN search. Given a query point q , the result of the k -NN search is a set of k nearest points to q . It defines a circle centered at q with radius equal to the distance from q to its k^{th} nearest point. The area of this circle, covering part of the rectangle, determines the searched (i.e., covered) portion of the rectangle. To *complete* the range query, we need to identify a set of input locations corresponding to a series of k -NN searches that would result in a set of circles covering the entire rectangle.

The optimization objective of this problem is to perform as few k -NN searches as possible. This is because each k -NN search results in communication overhead between the client and the server in addition to the actual execution cost of the search at the server. Hence, we would like our circles to have as few overlaps as possible. Finally, the fact that we do not know the radius of each circle prior to the actual evaluation of its corresponding k -NN query makes the problem even harder. Hence, the *sequence* of our k -NN searches become important as well. To summarize, the optimal solution should find a *minimal set* of query locations that minimizes the number of k -NN searches to completely cover the given range.

2. RANGE QUERIES USING K -NN SEARCH

Let R be a given query region and C_{R_q} be the circle inscribing R having q and R_q as its center and radius, respectively. Notice that q is also the center point of R . Let $P_k = \{p_1, p_2, \dots, p_k\}$ be the k nearest neighbors, obtained by the k -NN search with q as the query point and ordered by

their distances from q . Let $\overline{qp_k} = r$, i.e., r is the distance from q to the farthest point p_k . Let C_r be the circle of radius r centered at q , which we refer to as a k -NN circle. If $r > R_q$, then the k -NN search must have returned all the points inside C_{R_q} (see Fig. 1). Therefore, we can obtain all the points inside the region R after pruning out any points of P_k that are outside R but inside C_r . Then, the range query is complete. Otherwise, we need to perform some additional k -NN searches to completely cover R .

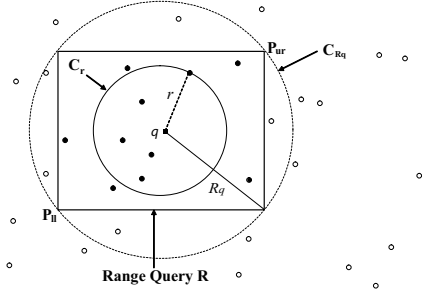
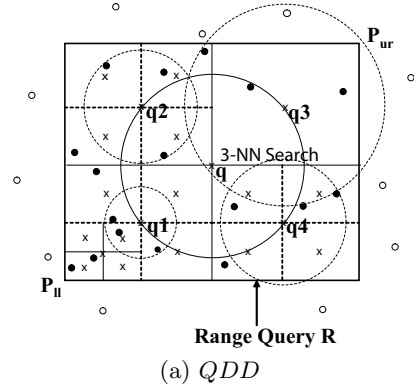


Figure 1: A range query using 5-NN search

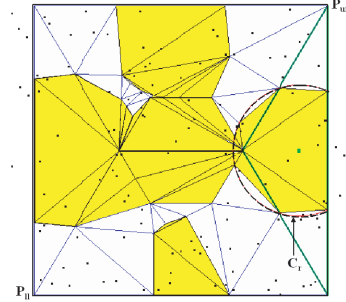
In general, multiple queries are evaluated to cover a reasonably large R . We define the general approach to this problem as follows: 1) divide R into a set of subregions, $R = \{R_1, R_2, \dots, R_m\}$, such that any $R_i \in R$ can be covered by a single k -NN search, 2) obtain the points by calling a k -NN search on each of the subregions, 3) combine the partial results to provide a complete result to the original range query R . The main issue is how to divide the query region R so that we can minimize the total number of k -NN searches. We can consider the following two approaches: 1) recursive approach: conduct a k -NN search in the current query region R , divide R equally if the k -NN search fails to cover R , and call k -NN search for each of the subregions, 2) greedy and incremental approach: divide R into subregions that are not equal in size, and select the largest subregion for a k -NN search. Check the covered region and select the next largest subregion for another k -NN search.

We propose a recursive approach, the Quad Drill Down (*QDD*) algorithm, as a solution to the general range query problem on the web using the properties of the quad-tree. In terms of the tree representation, the root node corresponds to the entire region, and each child node represents a quadrant. The quad-tree represents a recursive quaternary decomposition of space wherein at each level a subregion is divided into four equal sized subregions (quadrants). The properties of the quad-tree provide a natural framework for optimizing decomposition [3]. *QDD* recursively subdivides R into equal-sized quadrants until each subregion is covered by a single k -NN search so that all objects in R are obtained (see Fig. 2.a).

We also propose the Dynamic Constrained Delaunay Triangulation (*DCDT*) algorithm – a greedy and incremental approach to solve the problem using the Constrained Delaunay Triangulation (*CDT*). *DCDT* dynamically partitions R into subregions using a *CDT*. *DCDT* maintains the covered and uncovered regions (triangles) using *CDT* (see Fig. 2.b), and covered regions are bounded by constrained edges. *DCDT* chooses the centroid of the largest uncovered



(a) *QDD*



(b) *DCDT*

Figure 2: Two algorithms

subregion (triangle) as the query point of the next k -NN search to obtain the most coverage. The algorithm terminates when no more uncovered regions exist.

3. EXPERIMENTS

We evaluated *QDD* and *DCDT* for both synthetic (using both uniform and skewed distributions with various number of points ranged from 1K to 4K) and real GIS data sets (U.S. Geological Survey in 2001: Geochemistry of consolidated sediments in Colorado in the US [1]). A typical range of k , 5 - 20, was assumed in all experiments. The number of k -NN calls was measured using each algorithm while varying the query size from 1% to 10% of the entire region. For all experiments, both *QDD* and *DCDT* completely covered the query region. *DCDT* demonstrated around 50% of average reduction in the required number of k -NN calls compared to *QDD*. On the average, *DCDT* required 1.67 times more number of k -NN searches than the theoretical minimum ($\lceil \frac{n}{k} \rceil$, where n is the number of points in R).

4. REFERENCES

- [1] USGS mineral resources on-line spatial data: <http://tin.er.usgs.gov/>. 2001.
- [2] S. Byers, J. Freire, and C. Silva. Efficient acquisition of web data through restricted query interface. In *Poster Proceedings of the World Wide Web Conference*, pages 184–185, 2001.
- [3] S. Wang and M. P. Armstrong. A quadtree approach to domain decomposition for spatial interpolation in grid computing environments. *Parallel Computing*, Vol. 29(No. 3):1481–1504, 2003.