Journal of Discrete Algorithms 30 (2015) 1–12

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Range search on tuples of points

Akash Agrawal, Saladi Rahul, Yuan Li, Ravi Janardan *

Dept. of Computer Science and Eng., Univ. of Minnesota-Twin Cities, 4-192 Keller Hall, 200 Union St. S.E., Minneapolis, MN 55455, USA

ARTICLE INFO

Article history:
Received 2 April 2014
Received in revised form 20 October 2014
Accepted 21 October 2014
Available online 11 November 2014

Keywords:
Algorithms
Data structures
Computational geometry
Spatial query processing
Range search
Geometric transformation

ABSTRACT

Range search is a fundamental query-retrieval problem, where the goal is to preprocess a given set of points so that the points lying inside a query object (e.g., a rectangle, or a ball, or a halfspace) can be reported efficiently. This paper considers a new version of the range search problem: Several disjoint sets of points are given along with an ordering of the sets. The goal is to preprocess the sets so that for any query point q and a distance δ, all the ordered sequences (i.e., tuples) of points can be reported (one per set and consistent with the given ordering of the sets) such that, for each tuple, the total distance traveled starting from q and visiting the points of the tuple in the specified order is no more than δ. The problem has applications in trip planning, where the objective is to visit venues of multiple types starting from a given location such that the total distance traveled satisfies a distance constraint. Efficient solutions are given for the fixed distance and variable distance versions of this problem, where δ is known beforehand or is specified as part of the query, respectively.

1. Introduction

1.1. Motivation

Range search is a fundamental query-retrieval problem. In this problem, we are given a set, S, of n points in $\mathbb{R}^d$ and for any query object q (e.g., a rectangle, or a ball, or a halfspace) we wish to report the points of S lying inside q. Since many queries may be asked, it is worthwhile investing effort to preprocess S into a suitable data structure (i.e., index) so that the query time is small. Thus, the resource measures of interest are the query time and the space used by the data structure. Due to its many applications in diverse domains such as spatial databases, robotics, VLSI design, CAD/CAM, computer graphics etc., the range search problem has been studied extensively in the computational geometry and database literature and space- and query time-efficient solutions have been devised for many instances of the problem (see, for instance, [2,3,8,20,22]).

In this paper, we consider a new type of range search problem that is motivated by applications in trip planning. Consider the following scenario: Suppose that we are given a database that contains the locations of venues of different types (e.g., restaurants and theaters) in a large city. A tourist is interested in visiting a restaurant and a theater, in that order, but has a constraint on the maximum travel distance (due to reasons such as cost of travel, time, mileage restriction etc.). She wishes to identify a subset of (restaurant, theater) tuples, among the set of all such tuples, so that, for every tuple in the subset, the distance from her current location to the restaurant plus the distance from the restaurant to the theater is no

* Corresponding author.
E-mail addresses: akash@umn.edu (A. Agrawal), sala0198@umn.edu (S. Rahul), lixx2100@umn.edu (Y. Li), janardan@umn.edu (R. Janardan).

http://dx.doi.org/10.1016/j.jda.2014.10.006
1570-8667/

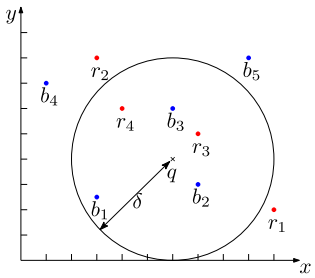

Fig. 1. Locations of two different types of venues (shown as red and blue points). Point q is the query point. The disk with radius δ represents the maximum travel distance constraint. (All figures in this paper are best viewed in color.)

larger than the specified distance. Therefore, she issues such a query from her location using, say, an application running on her smartphone and gets back the set of (restaurant, theater) tuples satisfying the distance constraint. The returned set is usually much smaller than the set of all (restaurant, theater) tuples and enables the user to make a more informed choice of the tuple to visit.

A naïve solution for this problem is to first find all the restaurants and theaters that are reachable from the query location within the specified travel distance and then check this set for the tuples to output. Unfortunately, this can be very expensive. For example, consider Fig. 1 which depicts a small data-set of restaurants (modeled as blue points b_i in the plane) and theaters (red points r_j). The user's location is denoted by the query point q. Let us assume that the maximum travel distance, denoted by δ, is 4 units. This distance constraint is represented as a disk of radius δ centered at q. In this example, it is clear that even though points b_1, b_2, b_3, r_3, and r_4 are all reachable within travel distance δ, only the (blue, red) tuples (b_2, r_3), (b_3, r_3), and (b_3, r_4) with total travel distance 3.4, 3.4, and 4, respectively, are part of the answer set. The other (blue, red) tuples formed by the points reachable from q within travel distance δ have total travel distance more than δ. (The other possible (blue, red) tuples are (b_2, r_4), (b_1, r_4), and (b_1, r_3) with total travel distance 5.7, 7 and 8.1, respectively.) Now consider a scenario where all the blue points lie on the perimeter of a disk centered at q and of radius δ and all the red points lie inside the disk. It is clear that all the points are reachable from q and yet the total travel distance for every (blue, red) tuple is larger than δ, so that the answer set is empty! This is the worst case scenario for the naive solution.

Our search problem generalizes naturally to more than two types of venues (e.g., restaurants, theaters, gas stations, grocery stores, etc.) where the objective is to start from the query point, q, and visit one venue of each type, in a pre-specified order, so that the total distance traveled is no larger than the specified maximum travel distance δ.

In fact, depending on the application there are two versions of the problem that are of interest. In the first version, δ is known beforehand (during preprocessing) while in the second version δ is known only at query time. (In both versions, q is known only at query time.) We will refer to these versions as the *fixed distance* and the *variable distance* problems, respectively.

For example, some car rental companies place a restriction on the maximum distance a user can travel. In this case the maximum travel distance, δ, is fixed for all the users and is pre-specified. This is an instance of the fixed distance problem. On the other hand, for some applications, the distance constraint can be imposed by reasons like travel time, travel cost etc. In this scenario, the maximum travel distance, δ, depends on the user and is specified only during the query. This is an instance of the variable distance problem. One might suspect that the fixed distance problem might admit a more efficient solution than the variable distance problem and, as we will see, this is indeed the case.

1.2. Problem statement and contributions

In what follows, it will be convenient to associate with each type of venue a distinct color and formulate our problem over colored point-sets in the plane.

Let S be a set of n points in $\mathbb{R}^2$, where each point is assigned a color c_i from a palette of m colors ($m \geq 2$). Let C_i be the set of points of color c_i and let $n_i = |C_i|$. Note that $C_i \cap C_j = \emptyset$ if $i \neq j$ and $\bigcup_{i=1}^{m} C_i = S$. Let $CS = (c_1, \ldots, c_m)$ be a given ordering of the colors. We wish to preprocess S into a suitable data structure so that for any query point q in $\mathbb{R}^2$ and a distance $\delta > 0$, we can report all tuples $(p_1, \ldots, p_m)$, where $p_i \in C_i$, such that $d(q, p_1) + \sum_{i=2}^{m} d(p_{i-1}, p_i) \leq \delta$. (Here $d(\cdot, \cdot)$ denotes Euclidean distance.) Hereafter, we will call this the *Colored Tuple Range Query* (CTRQ) problem.

Throughout, we will discuss the CTRQ problem for $m = 2$ colors, which will henceforth be called the 2-CTRQ problem. As we will discuss in Section 4, our solutions for 2-CTRQ generalize to $m > 2$ colors. For simplicity, we define the 2-CTRQ problem as follows.

Let B be a set of blue points, $B = \{b_1, b_2, \ldots, b_{n_1}\}$, and R be a set of red points, $R = \{r_1, r_2, \ldots, r_{n_2}\}$, in $\mathbb{R}^2$, where $n_1 + n_2 = n$. We wish to preprocess the sets, B and R, so that for any query point $q : (x_q, y_q) \in \mathbb{R}^2$ and distance $\delta > 0$, we can report all tuples (b_i, r_j), such that $d(q, b_i) + d(b_i, r_j) \leq \delta$.

In this paper, we present efficient solutions to the fixed and variable distance versions of the CTRQ problem. Our results include: