Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Rectilinear path problems in restricted memory setup[☆]

Binay K. Bhattacharya[a], Minati De[b], Anil Maheshwari[c], Subhas C. Nandy[d,*,1], Sasanka Roy[d]

[a] School of Computing Science, Simon Fraser University, Burnaby, Canada
[b] Department of CSA, Indian Institute of Science, Bangalore, India
[c] School of Computer Sciences, Carleton University, Ottawa, Canada
[d] Indian Statistical Institute, Kolkata, India

ARTICLE INFO

Article history:
Received 22 April 2015
Received in revised form 18 January 2016
Accepted 27 May 2016
Available online xxxx

Keywords:
Rectilinear path problem
In-place and read-only model of computation
Constant work-space

ABSTRACT

We study the *rectilinear path problem* in the presence of disjoint axis parallel rectangular obstacles in the *read-only* and *in-place* setup. The input to the problem is a set $\mathcal{R}$ of n axis-parallel rectangular obstacles in $\mathbb{R}^2$. The objective is to answer the following query efficiently.

Path-Query(p, q): Given a pair of points p and q, report an axis-parallel path from p to q avoiding the obstacles in $\mathcal{R}$.

In the *read-only* setup, we show that *Path-Query* (p, q) problem can be solved in $O(\frac{n^2}{s} + n \log s)$ time using $O(s)$ extra space. We also show that the existence of an x-monotone path and reporting it, if it exists, can be done with the same asymptotic time complexity. If the objective is to test the existence of an xy-monotone path between the given pair of points p and q avoiding the obstacles, and report it if exists, then our proposed algorithm needs $O(\frac{n^2}{s} + n \log s + M_s \log n)$ time with $O(s)$ extra space, where M_s is the time complexity for computing the median of n elements in the read-only setup using $O(s)$ extra space. Finally, we show that when the obstacles are unit squares instead of rectangles of arbitrary size, then there always exists a path of $O(\sqrt{n})$ links between a pair of query points, and the path can be reported in $O(n\sqrt{n})$ time using $O(1)$ extra work-space. It is also shown that there is an instance where the minimum number of links in a path between a pair of specified points is $O(\sqrt{n})$.

The objective of the *Path-Query* (p, q) in the *in-place* setup is to preprocess the input rectangles in a data structure in the input array itself such that for any pair of query points p and q, a rectilinear path can be reported efficiently. Here we propose an algorithm with $O(n \log n)$ preprocessing time and $O(n^{3/4} + \chi)$ query time, where χ is the number of links (bends) in the path. Both the preprocessing and query answering need $O(1)$ extra space.

☆ A preliminary version of this work appeared in Bhattacharya et al. (2015).
* Corresponding author.
E-mail addresses: binay@cs.sfu.ca (B.K. Bhattacharya), minati.isi@gmail.com (M. De), anil@scs.carleton.ca (A. Maheshwari), nandysc@isical.ac.in (S.C. Nandy), sasanka@isical.ac.in (S. Roy).
1 Part of the work of this author was done when this author was visiting Carleton University.

http://dx.doi.org/10.1016/j.dam.2016.05.031
0166-218X/© 2016 Elsevier B.V. All rights reserved.

1. Introduction

We study some new variations of *rectilinear path problem* in the restricted memory setup in a given closed planar environment containing a set $\mathcal{R}$ of n disjoint axis-parallel rectangular obstacles. The work is motivated from the *single point query problem* of Rezende et al. [10], and the two point query problem of Bint et al. [5]. Given a set $\mathcal{R}$ of rectangles and a fixed point p, Rezende et al. [10] preprocesses the input in a data structure in $O(n \log n)$ time so that given any arbitrary query point q, they can report the shortest path from p to q in $O(\log n + k)$ time, where k is the number of bends in the path. Bint et al. [5] preprocesses the obstacles in $\mathcal{R}$ in $O(n \log n)$ time, such that given a pair of query points (p, q), they can report the existence of an xy-monotone path from p to q in $O(\log n)$ time. Both these algorithms use $O(n)$ extra work-space in addition to the space allocated for storing the input.

Recently, space-efficient algorithms are becoming popular to the algorithm researchers since it has wide applications in an environment where we need to handle very large data sets, or where we need to work on a input data sharing environment. As the low memory algorithms need only a small amount of extra memory space compared to the input size, a larger part of the data can be kept in a faster memory. This makes the algorithm to run faster. Here two different models are very popular, namely the *read-only model* and the *in-place model*. In the *in-place model*, the elements in the input location are allowed to permute during the execution. However, at any time during the execution, all the input elements should be available in the input locations [6,7]. In the *read-only model*, the input data is not allowed to be permuted. Here one can only read the data and no write instruction can be executed in the input locations [1–3,8,9,11]. Geometric shortest path problems in a restricted memory setup is studied in the literature. Asano and Doerr [2] showed that if the input is a weighted directed *grid graph* given in a read-only memory, then the shortest path between a pair of nodes u and v can be computed in $O(n^{\frac{1}{\epsilon}})$ time using $O(n^{\frac{1}{2}+\epsilon})$ extra space. Asano et al. [3] considered the geometric shortest path problem inside a simple polygon P given in a read-only array. For a pair of points $p, q \in P$, their proposed algorithm runs in $O(n^2)$ time using $O(1)$ extra space. Thus, the algorithm in [3] works in a single connected planar region. Our study in this paper is identifying axis-parallel paths in a planar region with rectangular holes.

In read-only setup, we study the problem of finding an xy-monotone path between a pair of points p and q in the presence of rectangular obstacles. We show that an xy-monotone path between a pair of points in such an environment can be computed in $O(\frac{n^2}{s} + n \log s + M_s \log n)$ time using $O(s)$ extra space, where M_s is the time complexity for computing the median of n elements in read-only setup with $O(s)$ extra space. A comprehensive survey of results for the problem of computing the kth order statistics in a read-only environment is given in the conference version of the paper appeared in [4]. We also consider the case where no xy-monotone path exists between the points p and q. Here we report an x-monotone (or a y-monotone) path between p and q, which always exists. We provide an algorithm for this version of the problem that also runs in $O(\frac{n^2}{s} + n \log s)$ time using $O(s)$ extra space. Finally, we show that if the obstacles are unit squares, then there always exists a path of $O(\sqrt{n})$ links between a pair of query points, and a path with $O(\sqrt{n})$ links can be reported in $O(n\sqrt{n})$ time. We also demonstrate an instance where the number of links on a path is at least $O(\sqrt{n})$.

Next, we consider the query version of the problem in the in-place setup. Here, we can prepare a data structure by permuting the obstacles during the preprocessing. The objective is, given a pair of query points p and q, to report a path between p and q efficiently. Our proposed algorithm needs $O(n \log n)$ preprocessing time, and can answer the query in $O(n^{\frac{3}{4}} + \chi)$ time, where χ is the number of links in the path from p to q. Both preprocessing and query need $O(1)$ extra space.

2. Reporting a path—a simple algorithm

Here the input is a set of disjoint rectangles $\mathcal{R} = \{R_1, R_2 \dots, R_n\}$, in $\mathbb{R}^2$ in a read-only array, and a pair of points $p, q \in \mathbb{R}^2 \setminus \mathcal{R}$. The objective is to compute an axis-parallel path from p to q avoiding the obstacles using $O(1)$ extra space.

Without loss of generality assume that $x(p) < x(q)$ and $y(p) < y(q)$, where $x(.)$ and $y(.)$ represent the x- and y- coordinates of a point respectively. Connect p and q by an "L-path" as shown in Fig. 1(a). If the bend point r of this L-path lies inside a rectangle $R_i \in \mathcal{R}$, and the horizontal and vertical edges of this L-path intersect the rectangle R_i at the points a and b respectively, then update the path as $p \to a \to x \to b \to q$, where x is the appropriate corner of R_i (see Fig. 1(b)). Now, try to get a path from p to a and a path from b to q avoiding the obstacles in $\mathcal{R}$. Otherwise, if the bend-point r does not lie inside any rectangle in $\mathcal{R}$, then the desired path is the concatenation of two obstacle avoiding paths, from p to r and from r to q, respectively. The identification of R_i (if any) can be done in $O(n)$ time by inspecting all the members in $\mathcal{R}$.

To obtain a path from p **to** a: Inspect all the rectangles in $\mathcal{R}$ to find a rectangle (if any) that intersects the line segment $[p, a]$ and is closest to p. Let it be R_1. Now, update the path $p \to a$ as the U-path $p \to a_1 \to x_1 \to y_1 \to b_1 \to a$, where a_1, x_1, y_1 and b_1 are as illustrated in Fig. 1(c). Again find another rectangle (if any) that intersects $[b_1, a]$, and update the path accordingly. Proceed similarly, until the point a or the point r is reached. The same method is adopted to get a path from b (or r) to q.

Since the rectangles are non-overlapping, this method finds an obstacle avoiding path from p to q in $O(n^2)$ time using $O(1)$ extra space. If $O(s)$ extra work-space is available, then the time complexity can be improved to $O(\frac{n^2}{s})$ as follows. Let us consider the method of obtaining a path from p to a.