Geometric Applications of Posets*

Michael Segal and Klara Kedem

Department of Mathematics and Computer Science, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

Abstract. We show the power of posets in computational geometry by solving several problems posed on a set $S$ of $n$ points in the plane: (1) find the $k$ rectilinear nearest neighbors to every point of $S$ (extendable to higher dimensions), (2) enumerate the $k$ largest (smallest) rectilinear distances in decreasing (increasing) order among the points of $S$, (3) given a distance $\delta > 0$, report all the pairs of points that belong to $S$ and are of rectilinear distance $\delta$ or more (less), covering $k \geq \frac{n}{2}$ points of $S$ by rectilinear (4) and circular (5) concentric rings, and (6) given a number $k \geq \frac{n}{2}$ decide whether a query rectangle contains $k$ points or less.

1 Introduction

1.1 Problems

Given a set $S$ of $n$ points in the plane and an integer $k$ we solve the following problems in this paper:

1. Find the $k$ ($k \geq \frac{n}{2}$) nearest rectilinear neighbors (under $L_\infty$ metric) for each point of $S$ (by reporting the $n - k$ farthest rectilinear neighbors).
2. Enumerate the $k$ largest (smallest) rectilinear distances in decreasing (increasing) order.
3. Given a distance $\delta > 0$, report all the pairs of points of $S$ which are of rectilinear distance $\delta$ or less (more).
4. Find the smallest "rectangular" axis-aligned (constrained or not constrained) ring that contains $k$ ($k \geq \frac{n}{2}$) points of $S$. A rectangular ring is two concentric rectangles, the inner rectangle fully contained in the external one. As a measure we take the maximum width or area of the ring. By constrained we mean that the center of the ring is one of the points of $S$.
5. Find the smallest constrained circular ring (or a sector of a constrained ring) that contains $k$ ($k \geq \frac{n}{2}$) points of $S$.
6. Given a number $k \geq \frac{n}{2}$, decide whether a query rectangle contains $k$ points or less.

* Work by Klara Kedem has been supported by a grant from the U.S.-Israeli Binational Science Foundation.
1.2 Background

Most of the problems mentioned above have been considered in previous papers [6, 7, 8, 10, 16]. Dickerson et al. [6] present an algorithm for the first problem which runs in time $O(n \log n + nk \log k)$, and works for any convex distance function. Eppstein and Erickson [10] solve the first problem on a random access machine model in time $O(n \log n + nk)$ and $O(n \log n)$ space. In the algebraic decision tree model their time bound increases by a factor of $O(\log \log n)$. Flatland and Stewart [11] present an algorithm for the first problem which runs in time $O(n \log n + nk)$ in the algebraic decision tree model. Finally, a recent paper of Dickerson and Eppstein [8] describes an $O(n \log n + nk)$ time and $O(n)$ space algorithm for the first problem, it works for any metric and is extendable to higher dimensions. For our best knowledge only Dickerson and Shugart [7] present an algorithm for the second problem (for the largest $k$ distances) for any metric, and their algorithm requires $O(n + k)$ space with expected runtime of $O(n \log n + k \log k \log n)$. Dickerson et al. [6] present an algorithm for the problem: enumerate all the $k$ smallest distances in $S$ in increasing order. Their algorithm works in time $O(n \log n + k \log k)$ and uses $O(n + k)$ space. Lenhof et al. [16], Salowe [17], Dickerson and Eppstein [8] also solved this problem but they just report the $k$ closest pairs of points without sorting the distances, spending $O(n \log n + k)$ time and $O(n + k)$ space. An algorithm for solving the second problem (for the smallest $k$ distances) is also presented in [8], spending $O(n \log n + k \log k)$ time and using $O(n + k)$ space. [8] also considered the third problem: find all the pairs of points of $S$ separated by distance $\delta$ or less. They give an $O(n \log n + k)$ time and $O(n)$ space algorithm, where $k$ is the number of distances not greater than $\delta$.

Problem 6 is a variant of the orthogonal range search where we are given a set $S$ of $n$ points and want to find which points are enclosed by the query rectangle. This problem was efficiently solved by Bentley [3] in $O(\log n + m)$ query time, where $m$ is the number of points contained in the given query rectangle, using the range search tree and with preprocessing time and space $O(n \log n)$.

Some variations of problems 4 and 5 have been considered in previous papers. Efrat et al. [9] consider the problem of enclosing $k$ points within a minimal area circle and pose an open problem of covering $k$ points by a ring. They gave two solutions for the smallest $k$-enclosing circle. When using $O(nk)$ storage, the problem can be solved in time $O(nk \log^2 n)$. When only $O(n \log n)$ storage is allowed, the running time is $O(nk \log^2 n \log \frac{n}{k})$. The problem of computing the roundness of a set of points, which is defined as the minimum width concentric annulus that contains all points of the set was solved in [2, 14, 19]. The best known running time is $O(n^{3+\epsilon})$, given in [2], where $\epsilon > 0$ is an arbitrary small constant. Segal and Kedem [18] considered the problem of enclosing $k$ ($k \geq \frac{n}{2}$) points in the smallest axis parallel rectangle. Their algorithm runs in time $O(n + k(n - k)^2)$ and uses $O(n)$ space. Their method and algorithm are one of the tools used in this paper, and we review it below. It is based on posets (partially ordered sets) [1]. A poset being a partially ordered set of elements.

Segal and Kedem [18] describe how to construct a poset such that a subset $R$ of $S$ contains the $n - k$ elements of $S$ with the largest $x$ coordinates. They represent $S$ as a tournament tree. The tournament tree can be implemented as a heap. The operations of creating $R$ and updating the tournament tree run in times $O(n + (n - k) \log n)$.