CMSC 754: Fall 2023 Dave Mount

Homework 4: WSPDs, Sampling, and Motion Planning

Handed out Tue, Nov 28. Due, Thu, Dec 7, 9:30am. Late homeworks are not accepted (unless an extension has been prearranged) so please turn in whatever you have completed by the due date. Unless otherwise specified, you may assume that all inputs are in *general position*. Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$.

Problem 1. (15 points) This problem involves a couple problems related to computing the statistics on a sets of n points P in $\mathbb{R}^d$. Each can be solved with the use of WSPDs. In each case, assume that you have access to a subroutine that can compute an s-WSPD of size $O(s^d n)$ for any set of n points in $\mathbb{R}^d$ in time $O((n \log n) + s^d n)$. As we did in class, assume that the WSPD is represented as a set of pairs of nodes from a quadtree for P. Each node of the quadtree can store auxiliary information about the points lying within it subtree (e.g., a representative point, the number of points, or any additional statistics involving just these points).

In your answer, you should explain (i) what separation value do you use, (ii) what auxiliary information is stored in each node of the quadtree to help you, (iii) prove your algorithm's correctness, and (iv) indicate its running time.

(a) (5 points) Given a point set $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$ and $0 < \varepsilon < 1$ the *average interpoint distance* is defined to be

$$\Delta(P) \;=\; \frac{1}{\binom{n}{2}} \sum_{1 \le i < j \le n} \|p_i - p_j\|.$$

Computing this naively would take $O(n^2)$ time. Present an efficient WSPD-based algorithm that computes an ε-approximation to $\Delta(P)$. It computes a value $\widetilde{\Delta}(P)$ such that

$$\frac{1}{1+\varepsilon}\Delta(P) \;\le\; \widetilde{\Delta}(P) \;\le\; (1+\varepsilon)\Delta(P).$$

(b) (10 points) Solve (a), but this time with the generalization that we want to compute the average of the kth powers of the distances. Given an integer $k \ge 1$, define

$$\Delta^{[k]}(P) \;=\; \frac{1}{\binom{n}{2}} \sum_{1 \le i < j \le n} \|p_i - p_j\|^k.$$

(This is also known as the *kth moment* of the interpoint distance.) Assuming that k is a constant (independent of d or n) present an efficient WSPD-based algorithm that computes an ε-approximation to $\Delta(P)$. It computes a value $\widetilde{\Delta}^{[k]}(P)$ such that

$$\frac{1}{1+\varepsilon}\Delta^{[k]}(P) \;\le\; \widetilde{\Delta}^{[k]}(P) \;\le\; (1+\varepsilon)\Delta^{[k]}(P).$$

(**Hint:** The structure is essentially the same as part (a). It will be helpful to have an approximation for $(1+\varepsilon)^k$, assuming $0 < \varepsilon < 1$. For this, it is useful to recall the *Binomial Theorem*, $(a+b)^k = \sum_{i=0}^{k} \binom{k}{i} a^i b^{k-i}$.

Problem 2. (10 points) While coresets provide close approximations to some statistic of interest, it is often useful to have an easily computable lower bound or upper bound, that is just within a constant fraction of the exact value. In this problem, we'll consider a crude lower bound for the weight of the Euclidean minimum spanning tree in $\mathbb{R}^d$. Recall that in dimensions 3 and higher, computing the Euclidean minimum spanning tree exactly takes nearly quadratic time.

You are given a set of n points $P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^d$. Let $\text{emst}(P)$ denote the weight of P's Euclidean minimum spanning tree. We will compute our lower bound as follows. Place the points of P in a unit square grid, that is, a grid of hypercubes, each of side length 1 (see Fig. 1(a)). Let $N(P)$ denote the number of grid squares that contain at least one point of P.

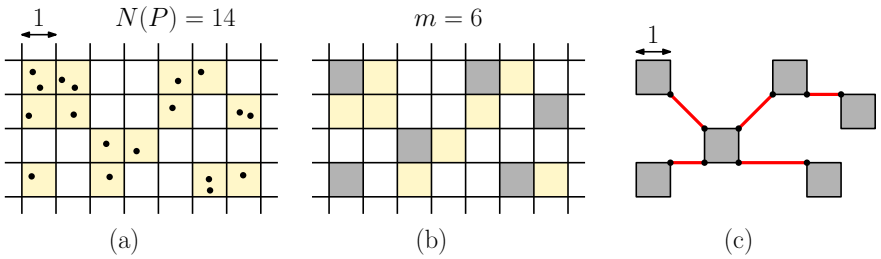

Figure 1: Lower bound for Euclidean minimum spanning tree.

Prove that there exist positive constants a_d and c_d (depending on the dimension d, but not on n), such that if $N(P) \geq a_d$ then

$$\text{emst}(P) \;\geq\; \frac{N(P)}{c_d}.$$

It is not necessary to derive the best values of a_d and c_d, but as a hint, both quantities vary exponentially with d.

Hint: This can be done in two parts. For the first part, let $G(P)$ denote the grid squares that contain at least one point of P, meaning that $N(P) = |G(P)|$. (These are the yellow squares in Fig. 1(a)). Show that if $N(P)$ is sufficiently large (depending on d) there is a subset of $G(P)$ containing at least a constant fraction of squares, so that no two of these squares have boundaries that touch each other. (These are the gray squares in Fig. 1(b)). For the second part, show that if you are given any set of m unit grid squares, where no two have boundaries that touch, then any spanning tree connecting these squares has weight at least $m - 1$ (see Fig. 1(c)). Finally, explain how these two facts can be combined to prove the lower bound on $\text{emst}(P)$.

Problem 3. (15 points) The objective of this problem is to investigate the *VC-dimension* of some range spaces. Recall that a *range space* Σ is a pair $(X, \mathcal{R})$, where X is a (finite or infinite) set, called *points*, and $\mathcal{R}$ is a (finite or infinite) family of subsets of X, called *ranges*.

For each of the following range spaces, derive its VC-dimension and prove your result. (Note that in order to show that the VC-dimension is k, you need to give an example of a k-element subset that is shattered and prove that no set of size $k + 1$ can be shattered.) Throughout, you may assume that points are in general position.

(a) (5 points) $\Sigma = (\mathbb{R}^2, \mathcal{U})$, where $\mathcal{U}$ consists of all orthogonal U-shaped ranges. An *orthogonal U-shaped range* is defined by three numbers (x_0, x_1, y_0). It is the region bounded between the two vertical lines $x_0 \leq x \leq x_1$ and above a horizontal line $y \geq y_0$ (see Fig. 2(a)).

(b) (10 points) $\Sigma = (\mathbb{R}^2, \mathcal{V})$, where $\mathcal{V}$ consists of all orthogonal V-shaped ranges. An *orthogonal V-shaped range* is defined by four numbers (x_0, x_1, a, b). It is the region bounded between the two vertical lines $x_0 \leq x \leq x_1$ and above a line $y \geq ax + b$ (see Fig. 2(b)).

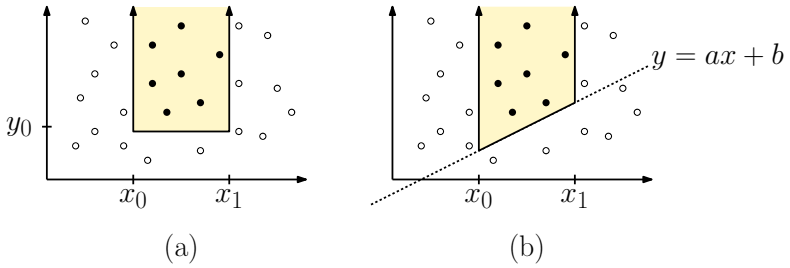

Figure 2: VC-Dimension of some range spaces.

Problem 4. (10 points) Let us consider a motion planning problem in the plane involving a set of rectangular obstacles and a translating "L"-shaped robot (see Fig. 3(a)). The robot $\mathcal{R}$ is an "L"-shaped polygon as shown in Fig. 3(b). Its reference point is its lower-left corner. The obstacles consist of a collection of non-intersecting rectangles $\{O_1, \ldots, O_n\}$, where O_i has lower-left corner p_i and horizontal length ℓ_i and vertical height h_i (see Fig. 3(c)).

You are given a start point s and target point t. Assuming the robot's reference point starts at s, is there an obstacle-avoiding motion that terminates with the endpoint at t? In this problem will explore a solution to this problem.

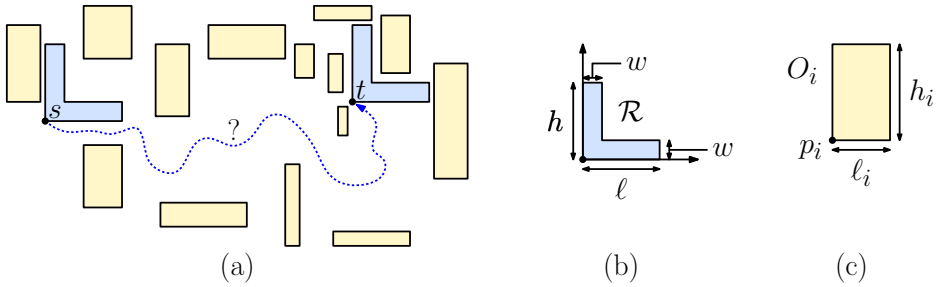

Figure 3: L-shaped robot motion planning.

(a) (5 points) Given an obstacle O_i, provide a clear description and a picture of the associated C-obstacle (its shape, side lengths, and location).

(b) (5 points) Given the starting and target points s and t, sketch an algorithm for determining whether there is a collision-free translational motion of the robot with its reference point starting at s and ending at t.

A high-level sketch of the algorithm is sufficient. To make your life simpler, you may assume that you are given a procedure that will input the C-obstacles from part (a), compute their union, and returns a convenient decomposition of free-space (e.g., as a trapezoidal map). Your algorithm should be efficient, in the sense that its running time should be roughly proportional to the combinatorial complexity of the free space, that is, the union of the C-obstacles. (See the Challenge Problem.)

Challenge Problem. Recalling the setup in Problem 4, what is the worst-case combinatorial complexity of the free space, that is, the union of all the C-obstacles? You may select any values you like for the robot's dimensions (h, ℓ, and w), and you may place the n rectangles anywhere you like. Justify your answer.