The Two-Median Problem on Manhattan Meshes *

Mordecai J. GOLIN   Yan ZHANG

March 2, 2005

Abstract

We investigate the two-median problem on an $M \times N$ mesh ($M \geq N$) with a Manhattan (L_1) metric and derive efficient algorithms w. r. t. m, n and r, the number of columns, rows and vertices in the mesh, respectively, that contain requests. Specifically, we give an algorithm with running times $O(mn^2 \log m)$ in general ($m \geq n$). For uniform meshes, we give an $O(MN^2)$ algorithm and an algorithm running in $O(MN \log N)$ time with high probability assuming the weights are i. i. d. random variables satisfying certain natural conditions. These improve upon the previously best known algorithm which ran in $O(mn^2 q)$ time where only unit requests were allowed and q was the total *units* of request.

Keywords: Two-median, mesh, probabilistic analysis.

1 Introduction

In the *k-median* problem we are given a graph $G = (V, E)$ with nonnegative edge costs. We want to choose k nodes (the *medians*) from V so as to minimize the sum of the distances between each node and its closest median. As motivation, the nodes can be thought of as *customers*, the medians as *service centers*, and the distance between a customer and a service center as the cost of servicing the customer from that center. In this view, the k-median problem is about choosing a set of k service centers that minimizes the total cost of servicing all customers. In a parallel computing scenario, the nodes can be thought of as clients and the medians as servers; the k-median problem then corresponds to a form of load-balancing.

The k-median problem is often extended so that each customer (node) has a weight, corresponding to the *amount* of service requested. The distance between a customer and its closest service center (median) then becomes the cost of providing *one unit* of service, i. e., the cost of servicing a customer will then be the weight of the customer-node times its distance from the closest service center.

*Partially supported by HK CERG grants HKUST6082/01E and HKUST6312/04E. The authors are at the Dept. of Computer Science, Hong Kong U. S. T., Clear Water Bay, Kowloon, Hong Kong. Email: {*golin,cszy*}*@cs.ust.hk*.

Lin and Vitter [9] proved that, in general graphs, even finding an $o(\ln n)$-approximate solution to the k-median problem is NP-hard. They were able to show, though, that it is possible in polynomial time to achieve a cost within $O(1+\epsilon)$ of optimal if one is allowed to use $(1+1/\epsilon)(\ln n+1)k$ medians. The problem remains hard if restricted to metric spaces, and Guha and Khuller [5] proved that this problem is still MAX-SNP hard. Charikar, Guha, Tardos and Shmoys [2] showed that constant-factor approximations can be computed for any metric space. In the specific case of points in Euclidean space, Arora, Raghavan, and Rao [1] developed a PTAS.

There are some special graph topologies for which fast polynomial time algorithms for finding exact solutions exist, though. In particular, this is true for trees [11, 12] and lines [6].

In this paper we deal with the specific problem of solving the *two-median* problem on an $M \times N$ orthogonal mesh ($M \geq N$) where the distance function is the Manhattan (L_1) metric. This problem was recently introduced by Lau, Cheng and Tse [8] who were motivated by load-balancing on parallel machines with mesh-topologies such as the iWarp system [4]. In [8] they developed an algorithm that ran in $O(mn^2q)$ time, where $m \geq n$ were respectively, the number of columns and rows in the mesh containing requests and q was the total number of *units* of requests (in their model requests were assumed to come in unit packets; e. g., 5 units of request from node v would add 5 to q. Since $q \geq m$, their algorithm therefore needed at least $O(m^2n^2)$ time). Their algorithm also required that mesh must be a subset of the *uniform mesh.* The main result of this paper is an improved algorithm that runs in $O(mn^2 \log m)$ time, dispensing with the dependence on q. Our algorithm also works for arbitrary meshes, dispensing with the requirement that the mesh be a subset of a uniform one. The algorithm uses $O(r)$ space, where r is the number of vertices containing requests. We also give an algorithm for uniform meshes that runs in $O(MN^2)$ time worst case and in $O(MN \log N)$ time *with high probability* if the requests of all nodes of the mesh are i. i. d. random variables satisfying certain natural conditions. Both algorithms use $O(MN)$ space.

1.1 The Parameters

Let $X = \{x_i : 1 \leq i \leq M\}$ and $Y = \{y_j : 1 \leq j \leq N\}$ be two sets of real numbers sorted increasingly, i. e., $x_1 < x_2 < \cdots < x_M$ and $y_1 < y_2 < \cdots y_N$. The two sets X and Y define an $M \times N$ *mesh*, denoted by G, whose vertex set is $X \times Y$, and whose edges consist of those between pairs of (x_i, y_j) and (x_i, y_{j+1}), for $1 \leq i \leq M$ and $1 \leq j < N$, and between (x_i, y_j) and (x_{i+1}, y_j), for $1 \leq i < M$ and $1 \leq j \leq N$. In general, G is considered to be *nonuniform*, i. e., elements in X and Y can take arbitrary real values. A mesh is *uniform* iff $x_{i+1} - x_i = y_{j+1} - y_j =$ constant for all $1 \leq i < M$ and $1 \leq j < N$. In a uniform mesh, we assume $x_i = i$ and $y_j = j$, i. e., X and Y are the integers starting from 1 running to M, N.

Each vertex may or may not have requests. The *weight* of vertex u, denoted by $w(u)$, represents the amount of requests from vertex u. The weight value can take arbitrary non-negative real numbers. Denote by r the number of vertices that have requests, i. e., nonzero weights, and by m and n the number of columns and rows, respectively, that have requests, i. e., at least one vertex on it have nonzero weights. Without loss of generality, we assume $m \geq n$. In general (i. e., nonuniform meshes), we would like to bound the algorithm by a function of m, n and r since we can throw away columns and rows that have no requests. But for uniform meshes, we cannot since throwing away columns or rows in this way will destroy the uniform property (in general), so the best we can hope for is to bound the algorithm by a function of M, N and possibly r. We will show a general algorithm for nonuniform meshes in Section 3, and then improve it for uniform meshes in Section 4.

1.2 Problem Definitions

In this section we specify the two-median problem considered in this paper. First, the distance $d(u, v)$ between vertices $u = (x_u, y_u)$ and $v = (x_v, y_v)$ is defined as the Manhattan, or L_1, distance, which is $|x_u - x_v| + |y_u - y_v|$. For a vertex set S, denote by $d(u, S)$ the distance between u and the closest vertex in S to u, i. e., $d(u, S) = \min_{v \in S} d(u, v)$. A *region* is just a subset of the mesh. We define the *cost* of vertices in a region R w. r. t. S, denoted by $cost_S(R)$, as the weighted sum of the distances from each vertex in R to S, i. e., $cost_S(R) = \sum_{u \in R} w(u) \cdot d(u, S)$. If S contains only one vertex, say $S = \{s\}$, we write $cost_s(R)$ instead of $cost_{\{s\}}(R)$. The k-median problem is to find a set S with $|S| = k$ that minimizes $cost_S(G)$. In this paper, we consider the case $k \leq 2$. An example of the problem is shown in Figure 1.

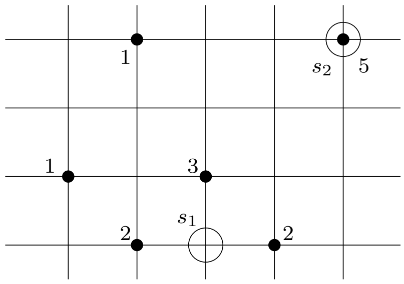

Figure 1: An example of the two-median problem on a uniform mesh. Here, $M = 5$ and $N = 4$. The small black circles indicate the 6 vertices that have requests; their labels indicate their weights. Big empty circles indicate the the optimal two-median s_1 and s_2. In this example, the optimal cost is 13; s_2 services itself and the other vertex in its row. s_1 services the other 4 requests.

Throughout the paper, we use "point" to denote an arbitrary point in the plane, and "vertex" for a vertex of the mesh. For polygonal regions, we use "corner" to denote a vertex of the polygon, which can either be chosen from the vertices of the mesh or not.

2 Preliminaries

2.1 The One-Median Problem

We start with the one-median problem. Note that in the optimal k-median set, each median s_i is an optimal one-median for the set of vertices whose nearest median is s_i.

Let $W_X(x_a, x_b)$, $W_X(x_a, x_b]$, $W_X[x_a, x_b)$, $W_X[x_a, x_b]$ be the sum of weights of vertices (x, y) where $-\infty < y < +\infty$ and, respectively, $x_a < x < x_b$, $x_a < x \leq x_b$, $x_a \leq x < x_b$, $x_a \leq x \leq x_b$. Similarly, let $W_Y(y_a, y_b)$, $W_Y(y_a, y_b]$, $W_Y[y_a, y_b)$, $W_Y[y_a, y_b]$ be the sum of weights of vertices (x, y) where $-\infty < x < +\infty$ and, respectively, $y_a < y < y_b$, $y_a < y \leq y_b$, $y_a \leq y < y_b$, $y_a \leq y \leq y_b$. Denote by W the total weight of all vertices in G. We have the following straightforward lemma from [8].

Lemma 1 *[8] Vertex $s = (x_s, y_s)$ is an optimal one-median, iff*

$$W_X(-\infty, x_s) \leq W/2 \leq W_X(-\infty, x_s] \tag{1}$$

and

$$W_Y(-\infty, y_s) \leq W/2 \leq W_Y(-\infty, y_s]. \tag{2}$$

□

Using Lemma 1, we can find all optimal one-medians in $O(m + n + r)$ time. Note that there might be more than one optimal one-medians for some special problem instances: If $W_X(-\infty, x_i] = W_X[x_j, +\infty) = W/2$ for some columns at x_i and x_j, any $x_s \in [x_i, x_j]$ will satisfy Inequality 1. This is also true for the y-direction. So, in general, the set of all optimal one-medians forms an axis-parallel rectangle. To simplify the notations, we will always use the optimal one-median with the smallest x- and y-coordinates. In such a way, the optimal one-median is always a vertex of the mesh, and always on a column and a row that have requests. That means allowing the one-median to be arbitrary point in the plane instead of being restricted to vertices of the mesh will not improve the solution. The same thing holds for the k-median problem: Recall that, as mentioned before, each median s_i in an optimal k-median set is an optimal *one-median* for the set of vertices whose nearest median is s_i. Hence,

Lemma 2 *Allowing the k-median set S to be arbitrary point in the plane $\mathbb{R}^2$ instead of being restricted to vertices of the mesh G will not improve the solution, i. e.*

$$\min_{|S|=k,\ S\subseteq\mathbb{R}^2} cost_S(G) = \min_{|S|=k,\ S\subseteq G} cost_S(G)$$

□

In the sequel we therefore always assume that the k-medians are vertices of the mesh.

2.2 Basic Regions

Denote the two-medians by $s_1 = (x_{s_1}, y_{s_1})$ and $s_2 = (x_{s_2}, y_{s_2})$. W. l. o. g., we assume $x_{s_1} \leq x_{s_2}$. Denote $l = |x_{s_1} - x_{s_2}|$ and $d = |y_{s_1} - y_{s_2}|$. We will focus on how to compute the optimal two-median under the constraints $l \geq d$. The case $l < d$ is similar, and we will discuss it briefly. Further more, we assume $y_{s_1} \leq y_{s_2}$ for clarity. Everything is identical in case $y_{s_1} > y_{s_2}$, so we will not discuss it. See Figure 2.

Define the *Voronoi cell* of s_i to be the region where the nearest median is s_i. If a point has equal distance to both s_1 and s_2, we assign it arbitrarily while keeping the Voronoi cells as continuous regions. We call the boundary curve between the two Voronoi cells the *bisector*. The following lemma describes the shape of the Voronoi cells and bisectors under L_1 distance.

Lemma 3 *Assume $l \geq d$ and $y_{s_1} \leq y_{s_2}$. Let two points $p_1 = (x_{p_1}, y_{p_1})$ and $p_2 = (x_{p_2}, y_{p_2})$ where $x_{p_1} = x_{s_2} - \frac{l-d}{2}$, $y_{p_1} = y_{s_1}$, $x_{p_2} = x_{s_1} + \frac{l-d}{2}$ and $y_{p_2} = y_{s_2}$. The bisector between s_1 and s_2 consists of the following three segments: two are vertical radials L_1 from p_1 downward and L_2 from p_2 upward, and the third is a line segment L_3 connecting p_1 and p_2. The slope of L_3 is always -1.* $\square$

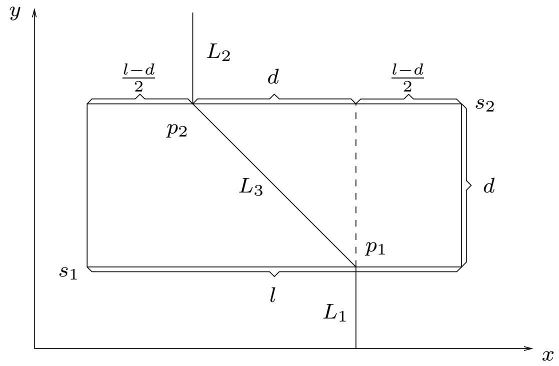

Figure 2: The bisector between s_1 and s_2. We assume $l \geq d$ and $y_{s_1} \leq y_{s_2}$.

Lemma 3 implies that the Voronoi cells under the L_1 metric have the property of being able to be partitioned into very simple shapes, as follows. Define R to be a *basic region* iff R is either an axis-parallel rectangle (possibly unbounded) or an axis-parallel triangle with hypotenuse' slope -1. Furthermore, we call a region R *separable* from a point $s = (x_s, y_s)$ iff $x_s \geq \max_{u \in R} x_u$ or $x_s \leq \min_{u \in R} x_u$, and $y_s \geq \max_{u \in R} y_u$ or $y_s \leq \min_{u \in R} y_u$.

Property 1 *$Vor(s_i)$ can be partitioned into $O(1)$ basic regions which are separable from s_i.*

Proof: We can partition the Voronoi cell so that L_3 is the hypotenuse of the triangular basic region. The rest of the area can be partitioned easily into four rectangular basic regions which are separable from s_i. $\square$

The reason for introducing separability is that the cost of a region w. r. t. any separable point can be computed efficiently. Denote by $w(R)$ the total weight of vertices in R. Define $c_x(R) = \sum_{u\in R} w(u) \cdot x_u$ and $c_y(R) = \sum_{u\in R} w(u) \cdot y_u$.

Property 2 *If region R is separable from a point s, the value of $cost_s(R)$ can be computed in $O(1)$ time given the values of $w(R)$, $c_x(R)$ and $c_y(R)$.*

Proof: $cost_s(R) = \sum_{u\in R} w(u) \cdot |x_u - x_s| + \sum_{u\in R} w(u) \cdot |y_u - y_s|$. We will only show how to compute $\sum_{u\in R} w(u) \cdot |x_u - x_s|$. The computation of $\sum_{u\in R} w(u) \cdot |y_u - y_s|$ is similar. Since R is separable from s, either $x_s \geq \max_{u\in R} x_u$ or $x_s \leq \min_{u\in R} x_u$. If $x_s \geq \max_{u\in R} x_u$, we have $|x_u - x_s| = x_s - x_u$ for all vertices $u \in R$. Then, $\sum_{u\in R} w(u) \cdot |x_u - x_s| = w(R) \cdot x_s - c_x(R)$. If $x_s \leq \min_{u\in R} x_u$, then $|x_u - x_s| = x_u - x_s$, and $\sum_{u\in R} w(u) \cdot |x_u - x_s| = c_x(R) - w(R) \cdot x_s$. $\square$

Combining Properties 1 and 2, if we can compute $w(R)$, $c_x(R)$ and $c_y(R)$ of any basic region R efficiently, we can compute $cost_S(G)$ efficiently. For clarity, we postpone the computations of w, c_x and c_y to Section 5. For the time being,

Definition 1 *Define $Q = Q(M, N, m, n, r)$ to be some function depending on the various parameters, such that the values of $w(R)$, $c_x(R)$ and $c_y(R)$ of any basic region R can be computed in amortized $O(Q)$ time.*

Lemma 4 *We can compute $cost_S(G)$ for any given two-median set S in amortized $O(Q)$ time.*

Proof: We first partition the Voronoi cells into separable basic regions as we did in Property 1, then, compute the cost of each basic region w. r. t. the corresponding median. The total cost $cost_S(G)$ is simply the sum of the costs of all the basic regions. This approach takes $O(Q)$ time since the number of the basic regions is constant. $\square$

3 An $O(mn^2Q)$ Worst-Case Time Algorithm

The following lemma generalized from [8] is a key to the two-median problem. We repeat the proof here in our notation since we will need it later.

Lemma 5 *[8] If $s_1 = (x_{s_1}, y_{s_1})$ and $s_2 = (x_{s_2}, y_{s_2})$ are the optimal two-median under the constraint $l \geq d$ and $y_{s_1} \leq y_{s_2}$, then*

$$W_X(-\infty, x_{s_1}) + W_X(x_{s_2}, +\infty) \leq W/2 \leq W_X(-\infty, x_{s_1}] + W_X[x_{s_2}, +\infty). \quad (3)$$

Proof: From Lemma 3, vertices with $x \in (-\infty, x_{s_1}]$ are in $Vor(s_1)$ and vertices with $x \in [x_{s_2}, +\infty)$ in $Vor(s_2)$. Denote by W_i the sum of weights of vertices in $Vor(s_i)$. Since s_i is the optimal one-median for vertices in $Vor(s_i)$, using Lemma 1, we have

$$W_X(-\infty, x_{s_1}) \leq W_1/2 \leq W_X(-\infty, x_{s_1}] \tag{4}$$

and

$$W_X(x_{s_2}, +\infty) \leq W_2/2 \leq W_X[x_{s_2}, +\infty). \tag{5}$$

The lemma follows by summing Inequalities 4 and 5. □

Recall that the optimal two-medians are on vertices of the mesh. So, we will choose x_{s_1} and x_{s_2} to be column coordinates of the mesh. We call $<x_{s_1}, x_{s_2}>$ a *candidate* column-pair if it satisfies Inequality 3. Lemma 5 says we only need to consider the two-medians on candidate column-pairs to find the optimal one. Note that for each x_{s_1} there might be more than one x_{s_2} to form a candidate column-pair. However as we said before, we always choose the optimal medians with the smallest x-coordinates, so we only need to consider at most one x_{s_2} for each x_{s_1}. Thus it is only necessary to consider $O(m)$ candidate column-pairs, leading to the following algorithm.

Theorem 1 *The optimal two-median of an $m \times n$ nonuniform mesh can be computed in $O(mn^2Q)$ time.*

Proof: We first show how to compute the optimal two-median under the constraint $l \geq d$. It is easy to calculate the $O(m)$ candidate column-pairs in $O(m + n + r)$ time using Inequality 3. For each candidate column-pair, we simply compute the costs of all $O(n^2)$ possible two-medians on those columns. So, the running time for the case $l \geq d$ is $O(mn^2Q)$.

By symmetry, the optimal two-median under the constraint $l < d$ can be computed in a similar way. There are $O(n)$ candidate row-pairs, and for each candidate row-pair, there are $O(mn)$ possible two-medians since $l < d \leq n$. So, the running time for the case $l < d$ is also $O(mn^2Q)$. □

4 A Probabilistic Algorithm

In this section, we consider uniform meshes. The algorithm in Theorem 1 runs in $O(MN^2Q)$ time for an $M \times N$ uniform mesh, we will improve it in the probabilistic case. We assume the request weights of *all* of the MN vertices are independent and identically distributed (i. i. d.) random variables with the same distribution X. Denote $\mu = E(X)$, the *expectation* of X. We will show that our algorithm runs in $O(MNQ \log N)$ time with high probability if X satisfies certain natural conditions; at the same time the algorithm will also run in $O(MN^2Q)$ time worst case.

The technique we use is simple binary search. Assume $l \geq d$ as before. Fix a candidate column-pair $<x_{s_1}, x_{s_2}>$ as before and consider all possible two-medians on those columns that arise by allowing all possible values of y_{s_1} and

y_{s_2}. In Theorem 1, we simply tried all $O(N^2)$ possible two-medians in arbitrary order. However, if we group the $O(N^2)$ possibilities according to the vertical distance $d = |y_{s_1} - y_{s_2}|$, we can use binary search, as follows.

Consider the set of two-medians with fixed vertical distance d on the candidate column-pair $<x_{s_1}, x_{s_2}>$. As before, we assume $y_{s_1} \leq y_{s_2}$, the case $y_{s_1} > y_{s_2}$ is identical. In this case, the locations of the two-medians are completely determined by the value of y_{s_1}. From the proof of Lemma 5, we only need to compute costs for those two-medians satisfying Inequality 4. Recall that W_i is the sum of the weights of the vertices in $Vor(s_i)$, $i = 1, 2$. Note from Figure 2 that W_1 increases when the two-medians move upward, i. e., y_{s_1} increases, while all other values in Inequality 4 are fixed. Computing W_1 for a given two-median takes $O(Q)$ time. So, a binary search can, in $O(Q \log N)$ time, find the upper and lower bounds of the value of y_{s_1} for which the corresponding two-medians satisfy Inequality 4. We call these two-medians *valid* and the range of y_{s_1} in which the two-medians are valid the *valid interval.* Denote the valid interval by $[y_{min}, y_{max}]$, where y_{min} and y_{max} are the minimum and maximum values of y_{s_1} over all valid two-medians. From the argument above we find that, instead of checking all n possibilities for y_{s_1}, as was essentially done in Section 3, we only need to examine the valid two-medians to find the optimal one. What remains is to bound the *length* of the valid interval.

Lemma 6 *Assume $l \geq d$. Fix a candidate column-pair and consider the two-medians with fixed vertical distance d. If $X \leq B$ for some constant B, then for any $c \geq 2$,*

$$\Pr(y_{max} - y_{min} \geq 8cN/d) \leq 4\exp(-2cN\mu^2/B^2) \tag{6}$$

Proof: From Inequality (4), we see that, for a fixed column-pair $<x_{s_1}, x_{s_2}>$, the maximum difference among the possible W_1 values of valid two-medians is at most $2(W_X(-\infty, x_{s_1}] - W_X(-\infty, x_{s_1})) = 2W_X[x_{s_1}, x_{s_1}]$ which has the same distribution as $2Y$ where Y is the sum of N independent copies of X.

We will only consider the case $y_{s_1} \leq y_{s_2}$ since the case $y_{s_1} > y_{s_2}$ is identical. Refer to Figures 2 and 3. Note that if we move the two-medians upward by one row, the Voronoi cell of s_1 will contain exactly d new vertices since the width of the part of the bisector curve with slope -1 is d. Denote by $W_1(y)$ the sum of weights of vertices in Voronoi cell s_1 when $y_{s_1} = y$. Let $Z_t = W_1(y_{min} + t) - W_1(y_{min})$. Z_t is therefore the sum of dt independent copies of X. Note that Y and Z_t are *independent* since Y only depends upon requests in column x_{s_1} while Z_t only depends upon requests in columns to the *right* of column x_{s_1}. Our goal is to prove that when $t = 8cN/d$, $\Pr(Z_t \leq 2Y) \leq 4\exp(-2cN\mu^2/B^2)$.

Let $X' = X/B$, then $X' \in [0, 1]$. Denote $\mu' = E(X') = \mu/B$. Similarly, let $Y' = Y/B$ and $Z'_t = Z_t/B$. From the discussion above, $E(Y') = N\mu'$ and $E(Z'_t) = dt\mu' = 8cN\mu'$ when $t = 8cN/d$. Since Y and Z_t are independent, Y' and Z'_t are also independent. So, we have

$$\begin{aligned}
\Pr(Z_t \leq 2Y) &= \Pr(Z'_t \leq 2Y') \\
&\leq \Pr(Y' \geq 2cN\mu') + \Pr(Y' \leq 2cN\mu') \cdot \Pr(Z'_t \leq 4cN\mu')
\end{aligned}$$

$$\begin{aligned}
&\leq \Pr(Y' \geq 2cN\mu') + \Pr(Z'_t \leq 4cN\mu') \\
&\leq \Pr(Y' - E(Y') \geq cN\mu') + \Pr(Z'_t - E(Z'_t) \leq -4cN\mu') \\
&\leq 2\exp(-2c^2N\mu'^2) + 2\exp(-4cN\mu'^2) \\
&\leq 4\exp(-2cN\mu'^2) \\
&= 4\exp(-2cN\mu^2/B^2)
\end{aligned}$$

The fourth inequality comes from the Hoeffding bound [7], which says $\Pr(|S_n - E(S_n)| \geq \delta n) \leq 2\exp(-2n\delta^2)$ where $S_n = \sum_{k=1}^{n} X_k$ and $X_k \in [0,1]$ are mutually independent. □

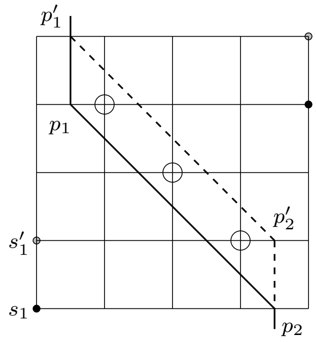

Figure 3: The change of the Voronoi cell when the two-median moves upward by one row. We can see d new vertices (with big circles on them) are added to $Vor(s_1)$ when the two-median $\{s_1, s_2\}$ moves to $\{s'_1, s'_2\}$.

Lemma 6 will be used in Theorem 2 to bound the length of valid intervals. Before continuing, we quickly look at the symmetric case $l < d$. Figure 4 shows the bisector and the Voronoi cells when $l < d$. As stated in the proof of Theorem 1, when $l < d$, there are $O(n)$ candidate row-pairs, and on each candidate row-pair, there are $O(mn)$ possibilities for the two-medians. In the probabilistic algorithm, we group these $O(MN)$ possible two-medians according to the horizontal distance l, and for each group, we use binary search to find the valid interval $[x_{min}, x_{max}]$, where x_{min} and x_{max} are the minimum and maximum values of x_{s_1} over all valid two-medians. Similar to Lemma 6, we also have the following lemma to bound the length of the valid intervals when $l < d$.

Lemma 7 *Assume $l < d$. Fix a candidate row-pair and consider the two-medians with fixed horizontal distance l. If $X \leq B$ for some constant B, then for any $c \geq 2$,*

$$\Pr(x_{max} - x_{min} \geq 8cN/l) \leq 4\exp(-2cM\mu^2/B^2) \tag{7}$$

□

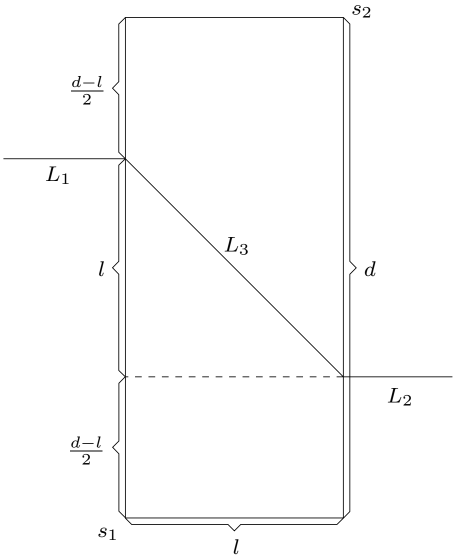

Figure 4: The bisector between s_1 and s_2 when $l < d$.

The proof of Lemma 7 is very similar to the proof of Lemma 6 and is therefore omitted. Before proceeding to Theorem 2, we clarify some notations used in this paper. We say an event happens *with high probability* iff the probability is $1 - O(N^{-c})$ for all constants $c \geq 0$. To bound the algorithm with high probability, we also need that the value of M is polynomially bounded by the value of N, i. e., $M = O(N^{\beta})$ for some constant $\beta \geq 0$. In what follows, $g(n) = \omega(f(n))$ implies $g(n)/f(n) \to \infty$ as $n \to \infty$.

Theorem 2 *Assume $M = O(N^{\beta})$. The optimal two-median of an $M \times N$ uniform mesh can be computed in $O(MNQ \log N)$ time with high probability, if the weight values of all vertices are i. i. d. random variables with some distribution X, where $X \leq B$ for some constant B and $E(X) = \omega(\sqrt{\ln N/N})$.*

Proof: We first consider the case $l \geq d$. The set of two-medians on a fixed candidate column-pair is partitioned into $2N$ groups classified according to the vertical distance d, where $1 \leq d \leq N$, and whether $y_{s_1} \leq y_{s_2}$ or not. For each group, finding the valid interval takes $O(Q \log N)$ time, and by Lemma 6, computing the costs for the valid two-medians takes at most $8cNQ/d = O(NQ/d)$ time with probability at least $1 - 4\exp(-2cN\mu^2/B^2)$. Summing over all $2N$ groups, the time to find the optimal two-median on a fixed candidate column-pair is $O(NQ \log N) + O(\sum_{d=1}^{N}(NQ/d)) = O(NQ \log N)$ with probability at least $1 - 8N\exp(-2cN\mu^2/B^2)$. So, for all M candidate column-pairs, computing the optimal two-median under the constraint $l \geq d$ requires $O(MNQ \log N)$ time with probability at least $1 - 8MN\exp(-2cN\mu^2/B^2)$.

Now consider the case $l < d$. For each candidate row-pair, we partition the set of two-medians into N groups according to the values of l, where $1 \leq l \leq N$, and whether $x_{s_1} \leq x_{s_2}$ or not. For each group, finding the valid interval takes $O(Q \log M)$ time, and by Lemma 7, computing the costs takes $O(MQ/l)$ time with probability at least $1 - 4\exp(-2cM\mu^2/B^2)$. So, a fixed candidate row-pair takes $O(NQ \log M) + O(\sum_{l=1}^{N} MQ/l) = O(MQ \log N)$ time with probability at least $1 - 8N \exp(-2cM\mu^2/B^2)$. Hence, all N candidate row-pairs will take $O(MNQ \log N)$ time with probability at least $1 - 8N^2 \exp(-2cM\mu^2/B^2)$.

Therefore, the running time of the algorithm will be $O(MNQ \log N)$ with probability at least $1 - 16MN \exp(-2cN\mu^2/B^2) \geq 1 - 16N^{\beta+1-2c\cdot\omega(1)/B^2}$, which satisfies our definition of high probability. □

An example of using the probabilistic algorithm is as follows. Assume $M = N$ so we have an $N \times N$ square mesh, and the requests are mutually independent unit requests, i. e., $X = 1$ with probability $p \geq \ln N/\sqrt{N}$, and $X = 0$ otherwise. Then $E(X) = \omega(\sqrt{\ln N/N})$ and the conditions of Theorem 5 are satisfied. Note that, for these parameters, with high probability, every row and every column of the uniform mesh will contain at least one request, i. e., $m = M$ and $n = N$. Therefore, with high probability, applying the worst-case algorithm of Theorem 1 would require $O(MN^2Q)$ time. On the other hand, with high-probability, the algorithm of Theorem 2 requires only $O(MNQ \log N)$ time.

5 Range Queries on Meshes

In this section we will discuss appropriate values for Q, the time for computing $w(R)$, $c_x(R)$ and $c_y(R)$ of a basic region R. Since all the three values w, c_x and c_y are calculated in some commutative groups, this problem is the classic range searching problem. In particular, all the three operations of the groups are simply ordinary arithmetic addition. For example, $c_x(R) = \sum_{u \in R} w(u) \cdot x_u$. The value $c_x(u) = w(u) \cdot x_u$ can be regarded as a property of u represented by a constant, and the value of $c_x(R)$ is just the arithmetic sum of the values of $c_x(u)$ for all $u \in R$.

The basic rectangular regions are orthogonal ranges. See e. g. [3]. The basic triangular regions are also orthogonal ranges, since the slopes of the hypotenuses are always -1. Furthermore, we only need counting-type queries, i. e., we do not need to *list* vertices in a region. In the counting case, orthogonal range searching is equivalent to solving the 2-sided case, also known as dominance search, where the search region is $[-\infty, a] \times [-\infty, b]$. Refer to Figure 5. The basic rectangular region $ABCD$ (the left figure) can be decomposed into the sum/difference of four 2-sided rectangle regions P_1AQ_1, P_1BQ_2, P_2DQ_1 and P_2CQ_2. If we can do 2-sided queries efficiently, then we can do the 4-sided query in the same time up to a constant factor. Similarly, a basic triangular region ABC (the right figure) can be decomposed into two 2-sided triangular regions P_1AQ_1 and P_1CQ_2 plus a 3-sided rectangular region Q_1BCQ_2. The 3-sided rectangles can be further decomposed into the difference of two 2-sided rectangles, so we only need to find ways to query 2-sided triangular regions efficiently.

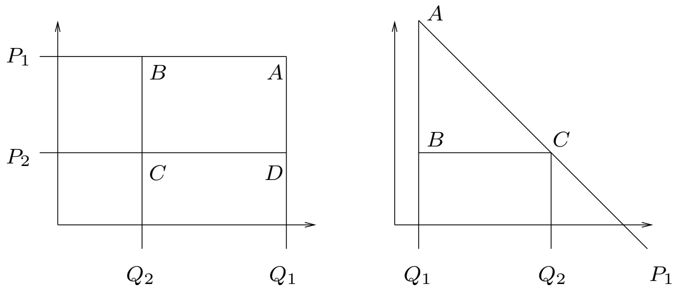

Figure 5: The decomposition of rectangular and triangular regions into 2-sided regions. P_1, P_2, Q_1, Q_2 are points at infinity of the corresponding directions.

Hence, we have reduced our problem into the 2-sided orthogonal range counting problem. Moreover, our problem is static (queries are known in advance), but we will not take advantage of this in this paper.

The range counting algorithm depends on whether the mesh is uniform or not. We first deal with the uniform case. Range counting on a uniform mesh is easy. A simple "table lookup" approach can do everything optimally in constant time.

Lemma 8 *In an $M \times N$ uniform mesh, the values of $w(R)$, $c_x(R)$ and $c_y(R)$ of any basic region R can be computed in $O(1)$ time using $O(MN)$ preprocessing time and $O(MN)$ additional space.*

Proof: For each vertex u, assume A is the location of u in Figure 5, we denote $R^R(u)$ the 2-sided rectangular region P_1AQ_1 in the left figure, and $R^T(u)$ the 2-sided triangular region P_1AQ_1 in the right figure. The table lookup approach is to simply pre-compute the values of w, c_x, c_y for the two regions $R^R(u)$ and $R^T(u)$ for each u. It is easy to do this in $O(MN)$ time, hence the lemma follows. $\square$

But the table lookup approach does not work well on nonuniform meshes. If we use table lookup, we need to pre-compute $O(mn)$ 2-sided rectangular regions and $O(mb^-)$ 2-sided triangular regions, where b^- is the number of slope -1 diagonals that have requests. The value of b^- is bounded by r, which results in a preprocessing time bigger than the bound in Theorem 1. Moreover, the query time is still $O(\log m)$ since we need a binary search.

There are many results on range searching on meshes, but to the best of the authors' knowledge, none of them can do better than $O(\log m)$ in counting-type queries while at the the same time keeping the preprocessing time low enough compared to the bound in Theorem 1 to be useful to us. Hence, we will therefore use the classic priority search tree [10] data structure. This data structure is not only practical, but also allow arbitrary real values instead of only integers.

Lemma 9 *[10] In an $m \times n$ nonuniform mesh, the values of $w(R)$, $c_x(R)$ and $c_y(R)$ of any basic region R can be computed in $O(\log m)$ time using $O(r)$ preprocessing time and $O(r)$ additional space.*

Combining this Lemma with Theorem 1 immediately gives

Theorem 3 *The optimal two-median on an $m \times n$ nonuniform mesh can be computed in $O(mn^2 \log m)$ time in the worst case, using $O(r)$ space.*

Combining Lemma 8 with Theorem 1 gives

Theorem 4 *The optimal two-median on an $M \times N$ nonuniform mesh can be computed in $O(MN^2)$ time in the worst case, using $O(MN)$ space.*

Finally, combining Lemma 8 with Theorem 2 gives

Theorem 5 *Assume $M = O(N^\beta)$ for some constant β. The optimal two-median on an $M \times N$ uniform mesh can be computed in $O(MN \log N)$ time with high probability if the weights of all vertices of the mesh are i. i. d. random variables with some distribution X, where $X \leq B$ and $E(X) = \omega(\sqrt{\ln N/N})$. The algorithm also runs in $O(MN^2)$ time in the worst case and uses $O(MN)$ space.*

References

[1] S. Arora, P. Raghavan, and S. Rao. Approximation schemes for Euclidean k-medians and related problems. In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 106–113, 1998.

[2] M. Charikar, S. Guha, E. Tardos, and D. B. Shmoys. A constant-factor approximation algorithm for the k-median problem. *Journal of Computer and System Sciences*, 65(1):129–149, 2002.

[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry - Algorithms and Applications*. Springer-Verlag, second edition, 2000.

[4] T. Gross and D. R. O'Hallaron. *iWarp: Anatomy of a Parallel Computing System*. MIT Press, 1998.

[5] S. Guha and S. Khuller. Greedy strikes back: Improved facility location problems. In *Proc. 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*, pages 649–657, 1998.

[6] R. Hassin and A. Tamir. Improved complexity bounds for location problems on the real line. *Operations Research Letters*, 10(7):395–402, 1991.

[7] W. J. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of American Statistics Association*, 58:713–721, 1963.

[8] F. C. M. Lau, P. K. W. Cheng, and S. S. H. Tse. An algorithm for the 2-median problem on two-dimensional meshes. *The Computer Journal*, 44(2):101–108, 2001.

[9] J.-H. Lin and J. S. Vitter. ϵ-approximations with minimum packing constraint violation. In *Proc. 24th Annual ACM Symposium on Theory of Computing (STOC'92)*, pages 771–782, 1992.

[10] E. M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, 1985.

[11] A. Tamir. An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operations Research Letters*, 19(2):59–64, 1996.

[12] A. Vigneron, L. Gao, M. J. Golin, G. F. Italiano, and B. Li. An algorithm for finding a k-median in a directed tree. *Information Processing Letters*, 74(1-2):81–88, 2000.