An Algorithm for the 2-Median Problem on Two-Dimensional Meshes[1]

F. C. M. LAU, P. K. W. CHENG AND S. S. H. TSE

Department of Computer Science and Information Systems, The University of Hong Kong, Hong Kong, People's Republic of China
Email: fcmlau@csis.hku.hk

We study the p-median problem which is one of the classical problems in location theory. For $p = 2$ and on a two-dimensional mesh, we give an $O(mn^2q)$-time algorithm for solving the problem, where, assuming that $m \geq n$, m is the number of rows of the mesh containing demand points, n the number of columns containing demand points and q the number of demand points.

Received 1 October 1999; revised 12 February 2001

1. INTRODUCTION

The mesh (and its variant, the torus) is a popular topology for processor interconnection in parallel computers. It has practical advantages such as low degree and perfectly compact layout when compared to other well-known topologies, for example the hypercube. A notable example of parallel computers based on the mesh topology is the iWarp system [1]. Dally has shown that low-dimensional networks have lower latency and higher hot-spot throughput than high-dimensional networks [2]. In this paper, we study the problem of finding a 2-median set in a two-dimensional mesh.

The p-median problem is a well-known problem in location theory, which is to locate p identical facilities in a network so that the sum of the distance between every client in the network and its nearest facility is minimized. This has practical application in real mesh-connected parallel computers, which is to locate servers in selected nodes so that service requests from client nodes can be directed to different servers to achieve load balancing, and the traffic so generated can be more spread out over the links. The problem has been proved to be NP-hard for general networks. For a survey on the NP-hardness and algorithms developed in recent years, one can refer to [3]. Mirchandani [4] gives a strong mathematical background for the problem, which also gathers many of the results before 1989. An early survey [5, 6] by Tansel *et al.* is still very useful, especially for beginners. Shmoys *et al.* [7] includes a short summary of some of the results. Recently, Guha and Khuller used the greedy approach to handle the problem [8]. Hamacher *et al.* dealt with a variation of the problem with multicriteria, instead of single criteria [9]. Lai and Chang proposed a method for virtual path layout in ATM networks which is based on solutions to the p-median problem [10]. Peeters proposed some new median problems together with some algorithms for these problems [11].

To our knowledge, no algorithm has yet been proposed for solving the problem on a mesh. Among the several topologies for which efficient algorithms for finding p-medians exist, the tree topology seems to have received the most attention. A recent paper by Auletta *et al.* [12] gives a linear-time algorithm for finding a 2-median set on tree networks having n vertices, which represents a significant improvement over other existing algorithms for the tree [13, 14, 15, 16]. Tree algorithms, however, cannot be easily adapted to solve the problem on the mesh, because in order to directly apply a tree algorithm, one would run into the non-trivial problem of enumerating a large number of trees that are embedded in the mesh.

In this paper, we assume the following system model.

- The mesh is non-oriented—that is, if $d()$ is the distance function, $d(v_i, v_j) = d(v_j, v_i)$ for any vertices v_i and v_j.
- All links (connecting adjacent vertices) have a length of 1.
- Clients and facilities are restricted to vertices of the mesh.
- All clients have a demand of 1, but multiple clients can co-reside in a vertex.
- The number and locations of clients are fixed and do not change over time.
- Facilities are uncapacitated (they could serve any number of clients) and can be located in any vertices.

Vertices having one or more clients are called *demand points*. Since we assume a client has a demand of 1, the (sum of) demand of vertex is equal to the number of clients in that vertex. The first three features listed above

[1]A preliminary version of this paper, entitled 'An Algorithm for the Location Problem in Two-Dimensional Meshes', appeared in *Proc. MFCS '98 Workshop on Communication*, August 1998, pp. 76–90.

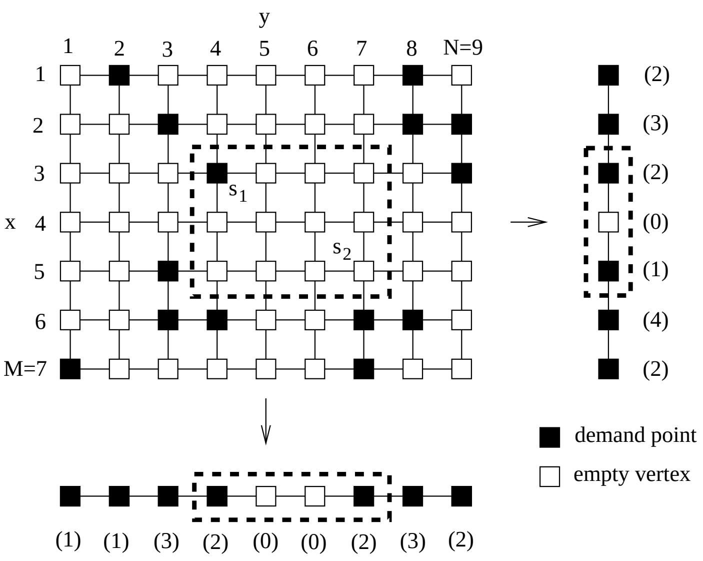

FIGURE 1. $P(1; T)$ decomposed into two chain problems.

are in line with those in real parallel computers where links are bidirectional and the 'cost' of traveling through a link is uniform across the entire network, and links are for communication only [17]. The last three features of the model are for simplicity so that the problem becomes tractable. Further research can consider variants of the present problem that are without these constraints.

Our solution to the problem is based on a solution to the 1-median problem on the mesh. Section 3 presents the 1-median problem and its solution. Section 4 discusses the properties of the specific kind of solutions to the 2-median problem we are after in this paper. Based on these properties, Section 5 then gives and analyses an algorithm that would find a 2-median set in $O(mn^2q)$ time, where m, n ($m \geq n$), and q are respectively the number of rows having clients, the number of columns having clients, and the number of demand points. Section 6 concludes the paper and proposes some further problems.

2. PRELIMINARIES

Given an $M \times N$ mesh and a set T of clients, the 2-median problem is to locate two identical facilities in one or two of the mesh's vertices so that the sum of the distance between every client and its nearest facility is minimized.

We assume that the input is a random list of demand points: $\langle (x, y)_1, z_1 \rangle$, $\langle (x, y)_2, z_2 \rangle, \ldots, \langle (x, y)_q, z_q \rangle$, where $(x, y)_1$ is the location of the first demand point with demand z_1, etc., and q is the number of demand points.

Let D be the sum of the distance between every client and its nearest facility for any median problem discussed in this paper. The objective is to minimize D.

All distances are *shortest path distances*—i.e. $d(v_1, v_2) = |v_1.x - v_2.x| + |v_1.y - v_2.y|$, for any vertices v_1 and v_2.

Let $T_h = |T|/2$.

Let $P(1; T)$ be the 1-median problem—to locate one facility, and $P(2; T)$ the 2-median problem—to locate two facilities.

Figure 1 shows a 7 (M) × 9 (N) mesh with 14 clients. Every vertex of the mesh which is in black is a demand point having a single client.

For a linear chain of t nodes, $v_1, \ldots, v_t$, and $c(i)$ being the number of clients in node v_i, let $L^-(i) = \sum_{1 \leq l \leq i} c(l)$ and $L^+(i) = \sum_{i \leq l \leq t} c(l)$; these quantities equal zero if i is out of range (i.e. $i < 1$ or $i > t$, respectively).

For an $M \times N$ mesh, let $R^-(i, T) = \sum_{1 \leq r \leq i} \mathrm{row}(r, T)$ and $R^+(i, T) = \sum_{i \leq r \leq M} \mathrm{row}(r, T)$, where $\mathrm{row}(r, T)$ is the number of clients of the set T in the rth row of the mesh; similarly, let $C^-(j, T) = \sum_{1 \leq c \leq j} \mathrm{col}(c, T)$ and $C^+(j, T) = \sum_{j \leq c \leq N} \mathrm{col}(c, T)$, where $\mathrm{col}(c, T)$ is the number of clients of the set T in the cth column of the mesh; these quantities have a value of zero if i (resp. j) is out of range.

3. 1-MEDIAN

The problem can be decomposed into two instances of the 1-median problem on a linear chain, one for each dimension. Figure 1 shows a 7 × 9 mesh and the two corresponding linear chains, the *x-chain* and the *y-chain*. The vertices of the linear chains are indexed in the same way as their corresponding rows or columns of the mesh. In the figure, the label next to a vertex in a linear chain is the number of clients in (also demand of) that vertex.

LEMMA 3.1. *Given a linear chain containing the client set T, and some vertex v_i. If $L^-(i) \geq T_h > L^-(i-1)$, then v_i is an optimal location for the facility.*

Proof. Since $L^-(i-1) < T_h$, $L^+(i) > T_h$. Let L' and L'' ($L'' > L'$) be these two quantities, respectively. If instead of v_i we choose v_j, $j < i$, we increase D by at least $(i-j)L''$, and at the same time decrease D by at most $(i-j)L'$—

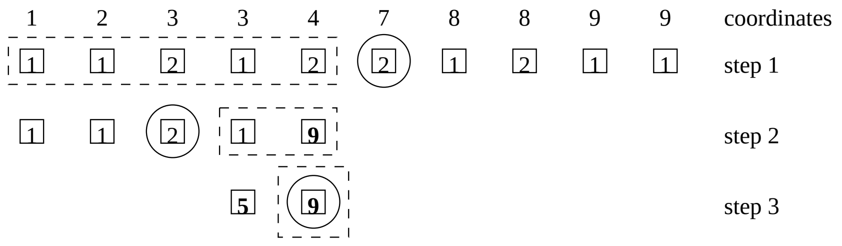

FIGURE 2. Finding a solution to $P(1; T)$.

hence a net increase. Similarly, if we choose v_j, $j > i$, since $L^-(i) \geq T_h$, D either increases or remains the same. Hence, v_i is an optimal location for the facility. □

We say that the v_i above is a *solution* to the 1-median problem on a linear chain. The lemma identifies the solution with the smallest i. There are cases where multiple solutions exist. By symmetry, we can easily find the one with the largest i. It is easy to see that a node (x, y) of the mesh is a solution to $P(1; T)$ if and only if v_x and v_y are solutions to the respective linear chains coming from the decomposition. The example in Figure 1 has multiple solutions, where s_1 is the 'smallest', and s_2 the 'largest' solution. We refer to the set of all solutions to any median problem as the *solution space* of that problem. The solution space for the mesh as well as those for the two linear chains are identified by dashed boxes in the figure.

The following theorem shows that a solution to $P(1; T)$ can be easily derived. Note that there is no need to actually build the linear chains. The input is the set of demand points in random order. We do not need to consider rows or columns of the mesh that do not contain any demand points—i.e., blank nodes in the x- or the y-chain (refer to Corollary 3.2).

THEOREM 3.1. *$P(1; T)$ can be solved in $O(q)$ time, where q is the number of demand points.*

Proof. We apply the standard $O(n)$-time algorithm for finding the median of an unsorted set of n elements [18].

First, we find the total sum of demands, and denote that by t. Next, we consider the x-coordinates (i.e. the x-chain). In $O(q)$ time, using the standard find-median algorithm, we can find an element v such that the number of demand points with a smaller x-coordinate is no more than $\lfloor q/2 \rfloor$ and the number of demand points with a larger x-coordinate is no more than $\lfloor q/2 \rfloor$. We then divide the q demand points into three sets, containing points having a smaller, the same, and a larger x-coordinate, respectively. Summing up the demand in each set, we then know which set the optimal facility should belong to. If the first set (smaller coordinates) has a total demand $\geq t/2$, the target facility is in the first set. Otherwise, if total demand of the third set (larger coordinates) is $> t/2$, the target facility is in the third set. If the target facility is not in the first or the third set, it must be in the second set (same coordinate), and we record the x-coordinate (call it i) thus found. Obviously at this point $L^-(i) \geq T_h > L^-(i-1)$ for the x-chain. All the above takes kq time, where k is a constant. If indeed the target facility is in the first or the third set, we apply the above recursively to the set using $kq/2 + kq/4 + \ldots \leq kq$ time. Note that at the end of each of these steps except the last, the sum of demand of the sets not chosen would be added to the demand of the boundary node (the one closest to the middle set) in the chosen set (as shown in Figure 2). The result is the x-coordinate of the smallest solution to $P(1; T)$.

In other words, we can use $O(q)$ time to find the smallest i such that $L^-(i) \geq T_h > L^-(i-1)$ for the x-chain. Similarly, we can find the y-coordinate for the smallest solution in $O(q)$ time. □

Figure 2 shows a simple example of the above procedure being applied to an x-chain, where $q = 10$, $t = 14$. At each step, the circled demand point is the chosen median; the dashed rectangle is the set containing the target demand point. Note how the sum of demand of the sets not chosen is added to the boundary node of the chosen set. The procedure finds the smallest solution. Symmetrically, we can find the largest i such that $L^+(i) \geq T_h > L^+(i+1)$ in $O(q)$ time, if we are interested in all solutions.

Using the mesh example in Figure 1 without loss of generality, we have the following set of inequalities governing the solution space of $P(1; T)$.

COROLLARY 3.1. *The solution space of $P(1;T)$ is bounded by s_1 and s_2 (assuming that $s_1.x \leq s_2.x$ and $s_1.y \leq s_2.y$) where*

$$R^-(s_1.x, T) \geq T_h > R^-(s_1.x - 1, T),$$
$$C^-(s_1.y, T) \geq T_h > C^-(s_1.y - 1, T),$$
$$R^+(s_2.x, T) \geq T_h > R^+(s_2.x + 1, T),$$
$$C^+(s_2.y, T) \geq T_h > C^+(s_2.y + 1, T).$$

Calling the rows containing $s_1.x$ and $s_2.x$ and the columns containing $s_1.y$ and $s_2.y$ the *boundary rows* and *columns* respectively of the solution space, we have the following.

COROLLARY 3.2. *All boundary rows and columns of the solution space must contain a client.*

Proof. Suppose without loss of generality that the row containing $s_1.x$ does not contain a client, then $R^-(s_1.x, T) = R^-(s_1.x - 1, T)$, violating Corollary 3.1. □

As a closing note to this section on 1-median, we should mention the work by Hassin and Tamir [19] as well as the work by Hsu *et al.* [20] on solving the p-median problem

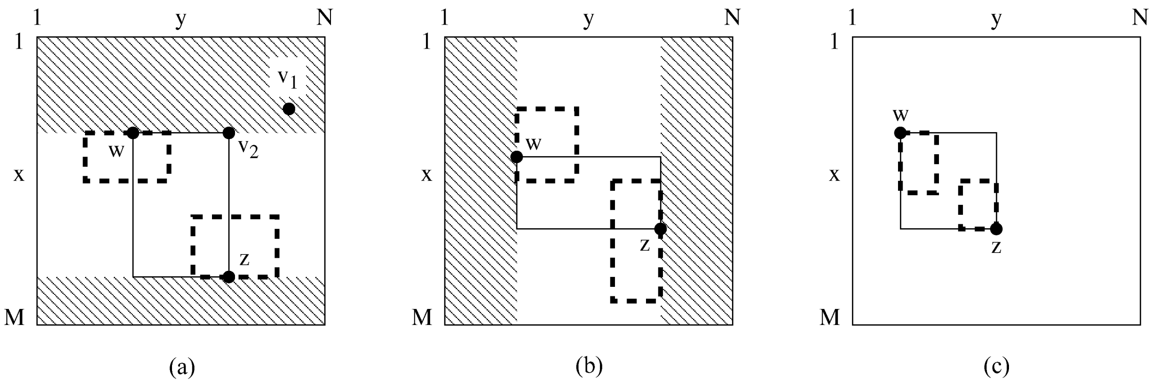

FIGURE 3. Solution to $P(2; T)$ forming (a) an upright rectangle, (b) a flat rectangle and (c) a square.

on a 'line'. We could have applied their algorithms by mapping vertices with k clients to a sub-chain of k clients with a negligibly small edge cost assigned to the edges of the sub-chain. For $p = 1$, however, the solution we give here is simple enough to not warrant the use of their more comprehensive algorithms.

4. 2-MEDIAN

The 2-median problem is to find a pair of nodes, $\{w, z\}$, such that D, the sum of distances from the clients to their nearest facility, is minimized. In the following, for convenience but without loss of generality, when we say $\{w, z\}$ is a solution to $P(2; T)$, we assume $w.x \le z.x$ and $w.y \le z.y$.

The following is obvious.

LEMMA 4.1. *For any solution $\{w, z\}$ to $P(2; T)$, we have the following.*

- *The client set T can be divided into two disjoint subsets, U and V, such that w is a solution to $P(1; U)$, and z is a solution to $P(1; V)$.*
- *For any w' and z' within the solution spaces of $P(1; U)$ and $P(1; V)$, respectively, $\{w', z'\}$ can replace $\{w, z\}$ as a solution.*

If a client is of equal distance from either facility, we assume that an arbitrary choice of one of the facilities would be assigned to the client. The solution $\{w, z\}$ forms either an upright rectangle, a flat rectangle, or a square, as shown in Figure 3. Note that Corollary 3.1 and 3.2 apply to the solution spaces of $P(1; U)$ and $P(1; V)$. Such solution spaces are shown as the dashed boxes in Figure 3.

LEMMA 4.2. *There exists a solution $\{w, z\}$ to $P(2; T)$ such that if $|w.x - z.x| \ge |w.y - z.y|$ (the solution forms an upright rectangle), then*

$$R^-(w.x, T) + R^+(z.x, T) \ge T_h > R^-(w.x - 1, T) + R^+(z.x + 1, T); \qquad (1)$$

and if $|w.x - z.x| \le |w.y - z.y|$ (the solution forms a flat rectangle), then

$$C^-(w.y, T) + C^+(z.y, T) \ge T_h > C^-(w.y - 1, T) + C^+(z.y + 1, T). \qquad (2)$$

Proof. Without loss of generality, consider Figure 3a, the case of an upright rectangle, where w is a solution to $P(1; U)$ and z a solution to $P(1; V)$. By Lemma 4.1, we can choose w to be among those solutions with the smallest x-coordinate in the solution space (the dashed box in the figure) of $P(1; U)$. Then, by Corollary 3.1, we have

$$R^-(w.x, U) \ge \frac{|U|}{2} > R^-(w.x - 1, U).$$

Similarly, choosing z to be one of the solutions with the largest x-coordinate in the solution space of $P(1; V)$, we have

$$R^+(z.x, V) \ge \frac{|V|}{2} > R^+(z.x + 1, V).$$

Combining, we have

$$R^-(w.x, U) + R^+(z.x, V) \ge T_h > R^-(w.x - 1, U) + R^+(z.x + 1, V).$$

From Figure 3a, $d(v_1, w) = d(v_1, v_2) + d(v_2, w)$, and $d(v_1, z) = d(v_1, v_2) + d(v_2, z)$. But $d(v_2, z) > d(v_2, w)$, and hence w is the facility for both v_1 and v_2 as well as all other clients of T with x-coordinate $\le w.x$. Likewise, z is the facility for all clients of T with x-coordinate $\ge z.x$. Therefore we can substitute T for U and V in the above inequality, giving

$$R^-(w.x, T) + R^+(z.x, T) \ge T_h > R^-(w.x - 1, T) + R^+(z.x + 1, T).$$

By symmetry, the second inequality, corresponding to Figure 3b, can be proved in exactly the same way. □

Note that both inequalities are true if the solution forms a square (Figure 3c). In the algorithm, to be presented in Section 5, the square case is treated as part of the upright-rectangle case as well as the flat-rectangle case. The overlap between the two cases (and hence duplicated effort) should be minimal when compared to the number of upright and flat rectangles the algorithm has to consider. It is important to note that given an instance of $P(2; T)$, any solution must form either an upright rectangle (and possibly a square) or a

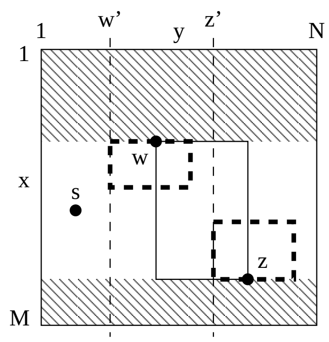

FIGURE 4. Solution to $P(1;T)$ falling outside the solution to $P(2;T)$.

flat rectangle (and possibly a square), but the existence of a solution forming a square cannot be guaranteed.

We refer to a solution $\{w, z\}$ that satisfies Lemma 4.2 as a *max-solution* because for fixed U and V, w and z are chosen to be on the two boundary edges of the solution spaces of U and V respectively and therefore have a maximum separation (along the x- or the y-dimension).

LEMMA 4.3. *Given any solution s to $P(1;T)$, there exists a max-solution $\{w, z\}$ to $P(2;T)$ such that*

$$w.x \le s.x \le z.x \quad \textit{and} \quad w.y \le s.y \le z.y.$$

Proof. Without loss of generality, consider the upright-rectangle case—i.e. Inequality 1 of Lemma 4.2. By symmetry, the flat-rectangle case can be dealt with similarly. Suppose that $s.x$ does not lie between $w.x$ and $z.x$; then, we have two cases: $w.x \le z.x < s.x$ and $s.x < w.x \le z.x$. Consider the former. We have

$$\begin{aligned}|T| &> 2(R^-(w.x-1,T) + R^+(z.x+1,T))\\ &\ge 2(R^-(w.x-1,T) + R^+(s.x,T))\\ &\ge 2R^+(s.x,T).\end{aligned}$$

But since s is a solution to $P(1;T)$, by Corollary 3.1, $R^+(s.x,T) \ge T_h$. Substituting, we have $|T| > |T|$, a contradiction. The case of $s.x < w.x \le z.x$ is symmetric to this one. Hence, all max-solutions to $P(2;T)$ must 'surround' $s.x$.

For the y-axis, the situation is as shown in Figure 4. By Lemma 4.1, we could move w and z over to w' and z' respectively and still have a max-solution. Assume without loss of generality that $z' > w'$. If the new rectangle is a flat one, then we can deal with it using the argument for the x-axis as in the above, and then $s.y$ must lie within w' and z'. Assume that the rectangle is upright and s is outside of it (as shown in the figure). Since $C^-(w'.y-1,U) < |U|/2$ and $C^-(z'.y-1,V) < |V|/2$ (Corollary 3.1), we have $C^+(w'.y,U) > |U|/2$ and $C^+(z'.y,V) > |V|/2$. Hence,

$$\begin{aligned}C^+(w'.y,T) &= C^+(w'.y,U) + C^+(w'.y,V)\\ &\ge C^+(w'.y,U) + C^+(z'.y,V) > \frac{|U|}{2} + \frac{|V|}{2}\\ &= T_h.\end{aligned}$$

It follows that if $s.y < w' < z'$ then $C^-(s.y,T) < T_h$, which is a contradiction to Corollary 3.1. The case of s being on the other side of the rectangle can be dealt with by symmetry. □

Hence, given a solution s to $P(1;T)$, all max-solutions that belong to the upright-rectangle case must surround $s.x$, and there exists at least *one* max-solution belonging to the upright-rectangle case that surrounds $s.y$.

LEMMA 4.4. *If $\{w, z\}$ satisfies Inequality 1, then $\{w', z'\}$, where $w'.x < w.x$ and $z'.x > z.x$, cannot be a max-solution to $P(2;T)$; neither can $\{w'', z''\}$, where $w''.x > w.x$ and $z''.x < z.x$, be a max-solution. Similarly for any pair satisfying Inequality 2.*

Proof. If $\{w, z\}$ satisfies Inequality 1, $R^-(w'.x,T) + R^+(z'.x,T) \le R^-(w.x-1,T) + R^+(z.x+1,T) < T_h$; hence, $\{w', z'\}$ does not satisfy Lemma 4.2. Then if $\{w'', z''\}$ is a max-solution, it should satisfy Inequality 1, and therefore $T_h > R^-(w''.x-1,T) + R^+(z''.x+1,T) \ge R^-(w.x,T) + R^+(z.x,T)$; hence, $\{w, z\}$ cannot satisfy Inequality 1. □

We say that $\{w', z'\}$ in the above lemma *surround* $\{w, z\}$ and $\{w, z\}$ *surround* $\{w'', z''\}$.

5. ALGORITHM

Lemma 4.3 says that for any given solution s to $P(1;T)$, there exists a solution to $P(2;T)$ that surrounds s. This solution to $P(2;T)$ satisfies Inequality 1 or 2 or both. The following algorithm capitalizes on this proven fact. It begins with an arbitrary solution s to $P(1;T)$. Then it enumerates all pairs of x's that surround s and satisfy Inequality 1. This appears to be an $O(M^2)$ operation, but by the lemmas in Section 4, a substantial number of such pairs can be skipped, resulting in an $O(M)$ operation. Then for each pair of x's, the algorithm considers all pairs of y's so that every solution thus formed surrounds s and is an upright rectangle or a square; the solution generating the smallest D is recorded. The same is then applied to the y dimension. The final result is one pair of (x, y) whose D is the smallest.

Algorithm Find2Median is given in Figure 5.

LEMMA 5.1. *Step 3 of Algorithm Find2Median will find a pair of $\{x_1, x_2\}$, $x_1 = 1$, satisfying Inequality 1 and surrounding $s.x$.*

Proof. By Corollary 3.1, $R^+(s.x,T) \ge T_h$, and so $R^-(x_1,T) + R^+(s.x,T) \ge T_h$. Note that we also have $R^-(x_1-1,T) = 0$. But there must exist some x, $s.x \le x \le M$, such that $R^+(x+1,T) < T_h \le R^+(x,T)$. When x_2 reaches x, we have $R^-(x_1-1,T) + R^+(x_2+1,T) < T_h \le R^-(x_1,T) + R^+(x_2,T)$, and $\{x_1, x_2\}$ satisfying Inequality 1 and surrounding $s.x$. □

We denote the value of x_2 after Step 3 by x_2°.

We call a pair of $\{x_1, x_2\}$ that surrounds $s.x$ in Algorithm Find2Median a *success* if it satisfies Inequality 1; a *failure* otherwise.

```
1. find a solution, $s$, to $P(1; T)$;
2. $x_1 = 1; x_2 = s.x$;
3. while [ $\{x_1, x_2\}$ not satisfying Inequality 1 ]
       $x_2 = x_2 + 1$;
4. repeat
       if [ $\{x_1, x_2\}$ satisfies Inequality 1 ]
           $X = X \cup \{x_1, x_2\}$;
           $x_1 = x_1 + 1$;
       else
           $x_1 = x_1 - 1; x_2 = x_2 + 1$;
   until [ $x_1 > s.x$ or $x_2 > M$ ];
5. if [ $x_2 < M$ ]
       $x_1 = x_1 - 1$;
       for $x_2 = x_2 + 1$ to $M$
           if [ $\{x_1, x_2\}$ satisfies Inequality 1 ]
               $X = X \cup \{x_1, x_2\}$;
6. similar to 3.–5., for the $y$-axis; results in $Y$;
7. $D_{\mathrm{opt}} = \infty$;
8. for each $\{x_1, x_2\}$ in $X$
       $l = x_2 - x_1$;
       for $y_1 = \max(s.y - l, 1)$ to $s.y$
           for $y_2 = s.y$ to $\min(y_1 + l, N)$
               $D$ = sum of the distance between every
                   client and $\{x_1, y_1\}$ or $\{x_2, y_2\}$,
                   whichever is nearer;
               if [ $D < D_{\mathrm{opt}}$ ]
                   $D_{\mathrm{opt}} = D$ and record $\{x_1, y_1\}$
                   and $\{x_2, y_2\}$;
9. similar to 8., for pairs in $Y$ and flat rectangles;
10. output $D_{\mathrm{opt}}$ and the recorded $\{x_1, y_1\}$ and $\{x_2, y_2\}$;
```

FIGURE 5. Algorithm Find2Median.

LEMMA 5.2. *During Step 4 of Algorithm Find2Median, if $\{x_1, x_2\}$ is a failure, then $\{x_1 - 1, x_2 + 1\}$ must be a success.*

Proof. At the beginning of Step 4, by Lemma 5.1, $R^-(x_1, T) + R^+(x_2, T) \geq T_h$, where $x_1 = 1$ and $x_2 = x_2^\circ$. As the loop progresses, as long as $\{x_1, x_2\}$ is a success, $R^-(x_1 + 1, T) + R^+(x_2, T) \geq T_h$ continues to be true because more rows are covered by x_1. The else clause is taken when the loop comes to an $\{x_1, x_2\}$ pair that fails Inequality 1 (refer to Figure 6). Obviously, the only possible reason for the failure as far as Inequality 1 is concerned is that $T_h \not> R^-(x_1 - 1, T) + R^+(x_2 + 1, T)$, or $R^-(x_1 - 1, T) + R^+(x_2 + 1, T) \geq T_h$. Since there exists at least one success pair before every failure (Lemma 5.1), $\{x_1 - 1, x_2\}$ is a success and satisfies Inequality 1. And so we have $T_h > R^-(x_1 - 2, T) + R^+(x_2 + 1, T)$, implying that $T_h > R^-(x_1 - 2, T) + R^+(x_2 + 2, T)$. Recalling that $R^-(x_1 - 1, T) + R^+(x_2 + 1, T) \geq T_h$, therefore $\{x_1 - 1, x_2 + 1\}$ satisfies Inequality 1. In other words, there cannot be any consecutive failures. □

THEOREM 5.1. *Steps 4 and 5 of Algorithm Find2Median cover all the pairs of x-coordinates that satisfy Inequality 1 and surround $s.x$.*

Proof. First of all, Step 4. By Lemma 4.2, we do not need to consider any $\{1, x\}$, $x < x_2^\circ$. Step 4 uses x_2 as a running base, each time considering all the pairs $\{x, x_2\}$, $x \leq x_2$. x_2 moves gradually from x_2° to M over Steps 4 and 5. For any x_2, the progression of x_1 stops when a failure ($\{x_1, x_2\}$) occurs. x_2 will change base, to $x_2 + 1$. By Lemma 4.4, the remaining $\{x, x_2\}$'s, where $x > x_1$, need not be checked because they would all be surrounded by $\{x_1 - 1, x_2 + 1\}$ or $\{x_1', x_2'\}$ which, by Lemma 5.2, is a success. For the new base, x_2', the checking needs only to begin from x_1' because any $\{x, x_2'\}$, $1 \leq x < x_1'$, would surround a previous success pair, $\{x_1 - 1, x_2\}$.

Step 4 terminates when incrementing x_1 or x_2 or both would cross the respective boundaries, $s.x$ or M. There exist four possible scenarios when this happens, as shown in Figure 7.

- (a) and (b) x_1 and x_2 are at their respective boundaries. Either it is a success or failure, this should be the end of Steps 4 and 5, because anything further would not surround $s.x$.
- (c) $x_2 = M$ and $\{x_1, x_2\}$ is a failure. By Lemma 5.2, $\{x_1 - 1, x_2 + 1\}$ or $\{x_1', x_2'\}$ would be a success (x_2', being out of range, does not matter here). Therefore, the search can stop here because by Lemma 4.4, no $\{x_1, x_2\}$, $x_1 > x_1'$ and $x_2 < x_2'$, can be a success.
- (d) $x_1 = s.x$ and $\{x_1, x_2\}$ is a success. Let these x_1 and x_2 be x_1' and x_2'. Step 4 terminates at this point. But x_2 has not reached its boundary. So Step 5 needs to finish the remaining x_2's—i.e. $M \geq x_2 > x_2'$. For each of these x_2's, the search needs only to check $\{x_1', x_2\}$ because any $\{x, x_2\}$, $x < x_1'$, would surround the success pair $\{x_1', x_2'\}$. Moreover, there is no need to respond to a failure like in Step 4, because if we decrement x_1 and increment x_2, the result would surround $\{x_1', x_2'\}$. □

Figure 6 shows a possible scenario of the execution of Steps 4 and 5. By symmetry, Step 6 of Algorithm Find2Median covers all the pairs of y-coordinates that satisfy Lemma 4.2 (Inequality 2) and surround $s.y$.

THEOREM 5.2. *For a mesh of size $M \times N$, $M \geq N$, the complexity of Algorithm Find2Median is $O(MN^2q)$, where q is the number of demand points in the mesh.*

Proof. Consider Steps 4 and 5. Let $A = s.x$, $B = M - x_2^\circ$, and K be the number of failures by the time Step 4 terminates (as shown in Figure 6). Note that K cannot exceed B. The number of times Step 4 would execute is equal to $A + 2K$. The number of times Step 5 would execute is $\leq B - K$. Hence the total time for Steps 3–5 is $\leq (x_2^\circ - s.x) + A + 2K + B - K = (x_2^\circ - s.x) + A + B + K < 2M$ or is $O(M)$. Similarly, the time for Step 6 is $O(N)$.

Consider Step 8. For each pair of $\{x_1, x_2\}$ and $l = x_2 - x_1$, we have $\leq l \times (l + 1)/2$ rectangles to consider, each of which would be used to compute the total distance from $|T|$ clients. For vertices having more than one client, the computation needs to be performed only once—i.e. the distance is multiplied by the number of clients. The value

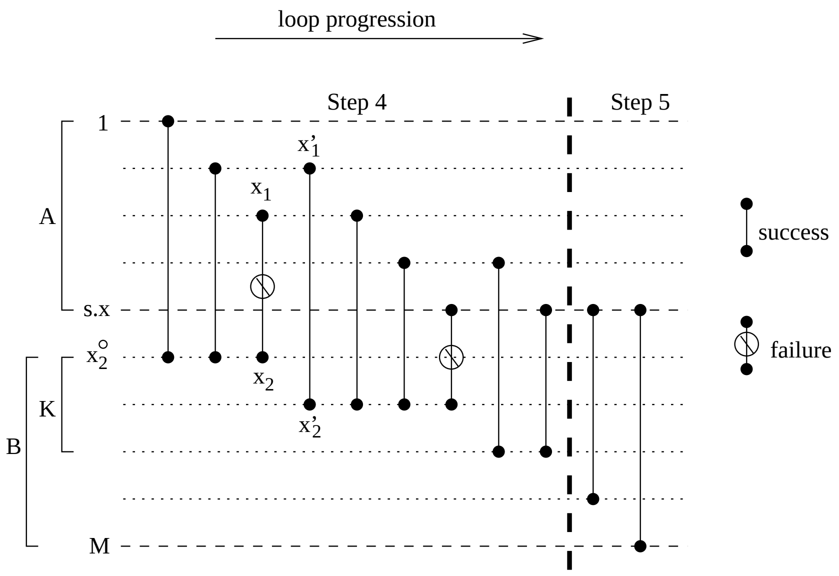

FIGURE 6. Execution of Steps 4 and 5.

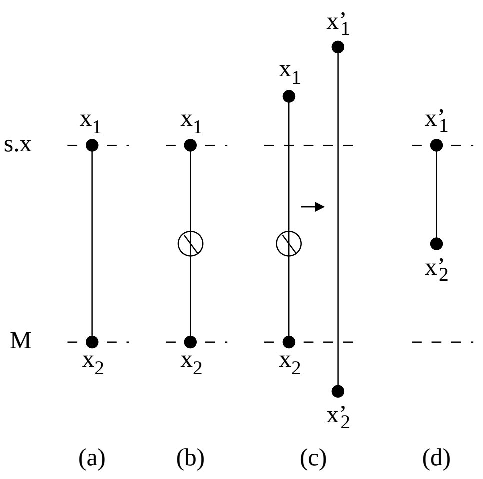

FIGURE 7. Between Step 4 and Step 5.

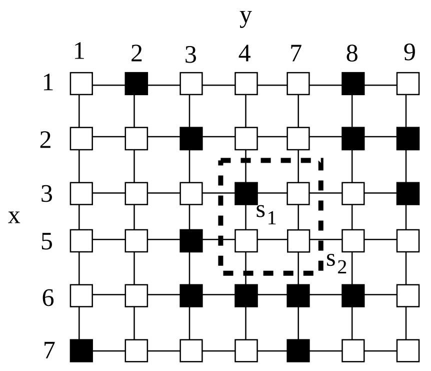

FIGURE 8. A compressed mesh.

of l could be as large as M, but since $M \geq N$, the width of a rectangle is bounded by N. Hence the number of rectangles to consider at each step is $O(N^2)$, and so the time for each step is $O(N^2q)$, where $q \leq |T|$ is the number of demand points. The total time for the entire Step 8 would be $O(MN^2q)$. For Step 9, l could be as large as N, and hence the time for this step would be $O(N^3q)$. □

Given the input which is a set of q demand points in random order, we then build an $m \times n$ 'compressed' mesh, as a 2D array, which has no empty row or column. This can be done in $O(q \log q + mn)$ time, or $O(mnq)$ since $q \leq mn$, where $q \log q$ is the sorting time for lining up the columns and the rows as well as the time to insert the demand points into the array; and mn the time to construct and initialize the 2D array. Figure 8 shows a compressed 6×7 mesh derived from the 7×9 mesh in Figure 1. Since all max-solutions are composed of two solutions to the corresponding 1-median problems for two subsets of T, by Corollary 3.2, they must be located in columns or rows that contain demand points. Hence, we can apply Algorithm Find2Median as it is to the compressed mesh, and the result will be correct.

COROLLARY 5.1. *For a compressed mesh of size $m \times n$, $m \geq n$, the complexity of Algorithm Find2Median is $O(mn^2q)$.*

6. CONCLUSION

The naive method to find a 2-median set—examining all pairs of vertices—would take $O(M^2N^2q)$ time. In the worst case where all vertices of the mesh have clients, our result represents an $O(M)$ times improvement over the naive method, assuming $M \geq N$. For meshes that are relatively sparse, our $O(mn^2q)$ time solution, where m is the number of rows having clients, n the number of columns having clients ($m \geq n$), and q the number of vertices having clients, is clearly much more efficient. The following should be worth pursuing.

- To find all solutions to $P(2; T)$ instead of just one solution (the naive method finds all solutions).
- To find approximate solutions to $P(2; T)$ that are based on simple but fast heuristical algorithms, such as dividing the mesh into two submeshes according to the distribution of the clients. Or to improve on existing approximation results, such as the work by Charikar *et al*. [21].
- To solve $P(f : T)$, $f > 2$.
- To consider meshes with wrap-around links, i.e. the torus and higher-dimensional meshes and tori.
- To consider meshes by lifting some of the restrictions as discussed in Section 1.

ACKNOWLEDGEMENTS

The authors wish to thank the referees for their valuable comments.

REFERENCES

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

[2] Dally, W. J. (1990) Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Computers*, **39**, 775–785.

[3] Hakimi, S. L., Schmeichel, E. F. and Labbé, M. (1993) On locating path- or tree-shaped facilities on networks. *Networks*, **23**, 543–555.

[4] Mirchandani, P. B. (1990) The p-median problem and generalizations. In Mirchandani P. B. and Francis R. L. (eds.), *Discrete Location Theory*, pp. 55–117. John Wiley, New York.

[5] Tansel, B. C., Francis, R. L. and Lowe, T. J. (1983) Location on networks: a survey. Part I: The p-center and p-median problems. *Management Science*, **29**, 482–497.

[6] Tansel, B. C., Francis, R. L. and Lowe, T. J. (1983) Location on networks: a survey. Part II: Exploiting tree network structure. *Management Science*, **29**, 498–511.

[7] Shmoys, D. B., Tardos, E. and Aardal, K. (1997) Approximation algorithms for facility location problems. *Proc. 29th Ann. ACM Symp. on Theory of Computing*, May 1997, pp. 265–274. ACM Press, New York.

[8] Guha, S. and Khuller, S. (1999) Greedy strikes back: improved facility location algorithms. *J. Algorithms*, **31**, 228–248.

[9] Hamacher, H. W., Labbé, M. and Nickel, S. (1999) Multicriteria network location problems with sum objectives. *Networks*, **33**, 79–92.

[10] Lai, W. K. and Chang, R. F. (1999) Virtual path layout in ATM networks based on the P-median problem. *Comp. Commun.*, **22**, 224–231.

[11] Peeters, P. H. (1998) Some new algorithms for location problems on networks. *Eur. J. Operat. Res.*, **104**, 299–309.

[12] Auletta, V., Parente, D. and Persiano, G. (1996) Dynamic and static algorithms for optimal placement of resources in a tree. *Theoret. Comput. Sci.*, **165**, 441–461.

[13] Gavish, B. and Sridhar, S. (1995) Computing the 2-median on tree networks in $O(n \log n)$ time. *Networks*, **26**, 305–317.

[14] Kariv, O. and Hakimi, S. L. (1979) An algorithmic approach to network location problems. Part II: The p-medians. *SIAM J. Appl. Math.*, **37**, 539–560.

[15] Mirchandani, P. B. (1980) Localizing 2-medians on stochastic networks. *Networks*, **10**, 329–350.

[16] Tamir, A. (1996) An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs. *Operat. Res. Lett.*, **19**, 59–64.

[17] Duato, J., Yalamanchili, S. and Ni, L. (1997) *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press, Los Alamitos, CA.

[18] Cormen, T. H., Leiserson, C. E. and Rivest, R. L. (1996) *Introduction to Algorithms*. MIT Press, Cambridge, MA.

[19] Hassin, R. and Tamir, A. (1991) Improved complexity bounds for location problems on the real line. *Operat. Res. Lett.*, **10**, 395–402.

[20] Hsu, V. N., Lowe, T. J. and Tamir, A. (1997) Structured p-facility location problems on the line solvable in polynomial time. *Operat. Res. Lett.*, **21**, 159–164.

[21] Charikar, M., Guha, S., Tardos, E. and Shmoys, D. (1999) A constant-factor approximation algorithm for the k-median problem. *Proc. 31st Ann. ACM Symp. on the Theory of Computing*, 1–10. ACM Press, New York.