Computing Least Median of Squares Regression Lines and Guided Topological Sweep

HERBERT EDELSBRUNNER and DIANE L. SOUVAINE*

Given a set of data points $p_i = (x_i, y_i)$ for $1 \le i \le n$, the *least median of squares regression line* is a line $y = ax + b$ for which the median of the squared residuals is a minimum over all choices of a and b. An algorithm is described that computes such a line in $O(n^2)$ time and $O(n)$ memory space, thus improving previous upper bounds on the problem. This algorithm is an application of a general method built on top of the topological sweep of line arrangements.

KEY WORDS: Computational geometry; Duality; Line arrangements; Robust statistics.

1. INTRODUCTION

Let $S = \{p_i = (x_i, y_i) \mid 1 \le i \le n\}$ be a set of n data points and $l : y = ax + b$ be a line in the plane. The *squared residual* of p_i with respect to l is equal to $(ax_i + b - y_i)^2$. Rousseeuw (1984) introduced the least median of squares (LMS) *regression line,* which is the line l that minimizes the median squared residual. The more familiar ordinary least squares (OLS) regression line minimizes the sum of the squared residuals, or equivalently the mean of the squared residuals. It has the disadvantage that a single corrupt data point can give the resulting regression line an arbitrarily large slope. On the other hand, 50% of the data would need to be corrupt to skew the LMS line arbitrarily. Thus LMS regression is far more robust than OLS (see Rousseeuw 1984).

Observe that computing an LMS regression line is equivalent to finding a strip (bounded by two parallel lines) that contains at least half the points of S and whose intersection with the y axis is shortest. The corresponding LMS regression line has the same slope and bisects the strip. As observed by Steele and Steiger (1986), such a strip minimizes its intersection with the y axis only if one bounding line contains at least two points of S and the other contains at least one point of S. Souvaine and Steele (1987) used this fact and the dual relationship between points and lines, and they designed two algorithms for computing an LMS regression line: one constructs the entire arrangement of the lines dual to the n points (see Edelsbrunner 1987) and takes $O(n^2)$ time and memory space, and the other sweeps the arrangement with a vertical line (see Edelsbrunner and Welzl 1986) that requires $O(n^2 \log n)$ time and $O(n)$ space. An approximate algorithm for LMS regression that works in general dimensions (not just two) can be found in Rousseeuw and Leroy (1987).

This article demonstrates that the superior time complexity of the first algorithm in Souvaine and Steele (1987), $O(n^2)$, can be achieved with only $O(n)$ space. The algorithm that does this is based on the topological sweep method of Edelsbrunner and Guibas (1989). Although a certain sophistication is necessary to implement the algorithm, it is probably the most practical (besides being the theoretically most efficient) of all known algorithms that compute LMS regression lines. In a nutshell, the topological sweep method speeds up the conventional sweep from $O(n^2 \log n)$ to $O(n^2)$ by compromising the straightness of the sweep line. More specifically, the sweep is done with a simple unbounded curve that intersects every line in exactly one point. Besides being more efficient, the topological sweep is more robust than the conventional sweep. Further details of this method, to the extent necessary for understanding the techniques of this article, are given in Section 2.

Compromising the straightness of the sweep line might not seem such a well-inspired idea, though, since we need to compute the lengths of *vertical* line segments that connect vertices with edges in the arrangement and intersect a certain number of lines. (See Sec. 3, which explains how the computation of LMS regression lines can be reduced to such tasks.) The problem with the topological sweep is that when we pass a vertex of the arrangement, the relevant edges vertically above and below the vertex may not be available at this time. In fact, the sweep might be way past those edges, and backing up would cost an unaffordable amount of time.

This is the point where some new ideas are needed, and this article describes them. We demonstrate a technique that keeps the sweeping curve straight enough to provide the necessary information for every vertex of the arrangement but leaves enough flexibility (or nondeterminism, if you wish) to retain the $O(n^2)$ time bound. The technique has several other applications (described in Secs. 3 and 4). Philosophically, the technique is a partial answer to the question How straight can we sweep an arrangement of n lines if we use only $O(n^2)$ time? Everybody who feels that it should be possible to conduct the sweep with a perfectly straight vertical line in time $O(n^2)$ is challenged to prove so. As a side result, such an algorithm would imply that $X + Y$ can be sorted in quadratic time (see Fredman 1976) and would thus solve an old open problem in the area of algorithm design.

The structure of this article is as follows. Section 2 reviews the topological sweep method of Edelsbrunner and

* Herbert Edelsbrunner is Associate Professor, Department of Computer Science, University of Illinois, Urbana, IL 61801. Diane L. Souvaine is Assistant Professor, Department of Computer Science, Rutgers University, New Brunswick, NJ 08903. The first author acknowledges the support of National Science Foundation Grant CCR-8714565. The second author acknowledges the partial support of a Henry Rutgers Research Fellowship and of National Science Foundation Grant CCR-8803549.

Reprinted from: © 1990 American Statistical Association
Journal of the American Statistical Association
March 1990, Vol. 85, No. 409, Theory and Methods

Guibas (1989). Section 3 introduces the so-called alignment graph and shows how it modifies the topological sweep so that an LMS regression line can be computed in quadratic time and linear space. Other applications of the alignment-graph technique are mentioned as well. Section 4 gives another method for controlling the topological sweep that leads to an $O(m \log(m \cdot n) + n^2)$-time and $O(m + n)$-space algorithm for the following point-location problem: Given m points and n lines, for each point determine the line it hits first as it drops vertically downwards. More than this specific result [which is the currently best for values of m close to n^2, e.g., $m = \Theta(n^2/\log n)$], we believe that the potential of the algorithmic method justifies this section. Finally, Section 5 discusses the results of this article and points out a few open problems.

2. SYNOPSIS OF TOPOLOGICAL SWEEP

Let $H = \{l_i : y = x_i x - y_i \mid 1 \le i \le n\}$ be a set of n nonvertical lines in the Euclidean plane. The *arrangement* $\mathcal{A}(H)$ defined by H is the dissection of the plane whose vertices are the intersections of the lines, whose edges are the connected components of the lines after removing all vertices, and whose regions are the connected components of the plane after removing all lines (see Fig. 1). To *sweep* $\mathcal{A}(H)$ means to visit the vertices, edges, and regions of $\mathcal{A}(H)$ in a certain order. The topological sweep algorithm of Edelsbrunner and Guibas (1989) uses a topological line to determine the order. A *topological line* is a simple curve, unbounded at both ends, that intersects each line in exactly one point where it crosses the line [see Levi (1926) for an early account of topological lines]. We require that the topological line avoids all vertices of $\mathcal{A}(H)$ and that its unbounded ends lie in the topmost and bottommost regions of $\mathcal{A}(H)$. Here the topmost (bottommost) region of $\mathcal{A}(H)$ is the one that lies vertically above (below) all lines in H. For simplicity, we assume that no three lines meet in a point and that no two lines are parallel. Algorithmically, this assumption can be simulated by a conceptual perturbation (see Edelsbrunner and Mücke 1988).

The sequence of edges intersecting the topological line is called the *cut* and is represented by a linear array—its ith entry stores the ith edge from the top complete with supporting line and left and right endpoints (if they exist).

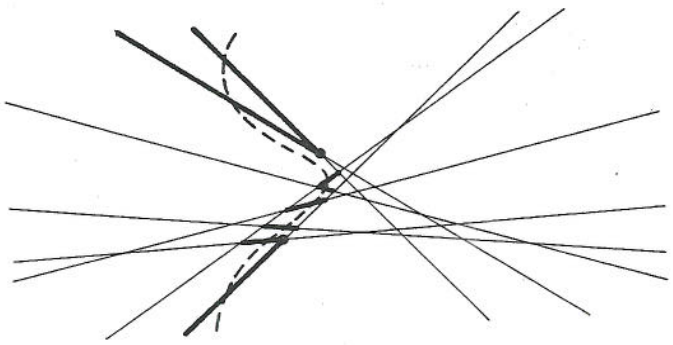

Figure 1. The Arrangement $\mathcal{A}(H)$, a Set of Nonvertical Lines in the Euclidean Plane. The topological line intersects a sequence of edges, one on each line; this is the cut representing the topological line. Common right endpoints of adjacent cut edges are emphasized.

(Here we refer to the above/below relation on the set of cut edges defined by the topological line. In many other cases we use "above" and "below" in the intuitive geometric meaning that implies that there is a vertical line along which one object is above another. To avoid confusion, we sometimes write "vertically above" and "vertically below" to make clear what we mean.) Initially, the cut contains the leftmost edge of each line. An *elementary step* of the sweep replaces two adjacent cut edges with a common right endpoint by the next two edges on the same lines; their common left endpoint is the same as the common right endpoint of the replaced edges. We use the term elementary step for what we just described (namely a certain action of the algorithm) as well as to denote a pair of cut edges with common right endpoint. The sweep is complete when the cut contains the rightmost edge of each line. Edelsbrunner and Guibas (1989) show that an elementary step can be performed unless the sweep is complete and that there is an $O(n)$-space data structure that supports an elementary step in constant amortized time. In fact, in the worst case a single elementary step takes time $\Omega(n)$, but the amount of time spent on all elementary steps is proportional to their number. At any point in time, the possible elementary steps (adjacent cut edge pairs with common right endpoints) are stored in a stack. To perform an elementary step, we just pop the topmost edge pair, replace the two cut edges appropriately, and push newly created elementary steps (if any).

The algorithms described in Sections 3 and 4 differ from the aforementioned sweep algorithm only in the way they push and pop elementary steps. In particular, in Section 3 we will be more picky in choosing elementary steps we wish to perform. In Section 4, we push and pop elementary steps depending on current needs to advance the cut at certain places.

3. ALIGNMENT GRAPHS AND LEAST MEDIAN OF SQUARES REGRESSION

As mentioned in the introduction, computing a least median of squares regression line of a point set $S = \{p_i = (x_i, y_i) \mid 1 \le i \le n\}$ is equivalent to finding a closed nonvertical strip bounded by two parallel lines that minimizes the length of its intersection with any vertical line and contains at least half the points of S. More specifically, we require that the strip contains at least $k = [n/2] + 1$ of the n points. Assuming that no three points are collinear and no two point pairs lie on parallel lines, this strip contains exactly k points, three on its boundary.

Following the approach of Souvaine and Steele (1987), we map each $p_i = (x_i, y_i)$ to the dual line $l_i : y = x_i x - y_i$. Any nonvertical strip corresponds to a vertical line segment with the property that the strip contains a point if and only if the line segment intersects the corresponding line. The length of the line segment is the same as the length of the intersection of the strip with a vertical line. Hence a strip whose axis is a least median of squares regression line corresponds to a vertical line segment of minimum length that intersects k dual lines. Since one

boundary line of such a strip goes through two points of S and the other contains one point, we can restrict our attention to vertical line segments with one endpoint at a vertex of $\mathcal{A}(H)$, $H = \{l_i \mid 1 \le i \le n\}$, and the other on an edge of the arrangement (see Fig. 2). We solve the problem by computing (and taking a minimum of) all such line segments in $O(n^2)$ time and $O(n)$ space. As indicated earlier, the method that achieves these bounds is a topological sweep of the arrangement, $\mathcal{A}(H)$. To make such a sweep work, we have to guarantee somehow that whenever we pass a vertex of the arrangement the edges that hold the other endpoints of the two vertical line segments starting at this vertex are aligned with it.

Here is a way to guarantee that whenever we pass a vertex (it is the common right endpoint of an adjacent cut edge pair) the $(k-2)$nd edges vertically above and below the vertex are current cut edges. We now introduce some definitions. For an edge e_i in a cut $(e_1, e_2, \ldots, e_n)$, we define $\overline{e}_i = e_{i-k+2}$ and $\underline{e}_i = e_{i+k-2}$ (if they exist); so $\overline{e}_i$ is the $(k-2)$nd cut edge above e_i and $\underline{e}_i$ is the $(k-2)$nd cut edge below e_i. The vertical projection of e_i onto the x axis is denoted by $\pi(e_i)$. We say that e_i and e_j are *aligned* if $\pi(e_i) \cap \pi(e_j) \neq \emptyset$, and that e_i is *ahead* of e_j if the right endpoint of $\pi(e_i)$ is to the right of the right endpoint of $\pi(e_j)$. We now redefine when an adjacent pair of cut edges can be advanced in the sweep. A pair (e_i, e_{i+1}) constitutes an elementary step if (a) e_i and e_{i+1} share a common right endpoint, and (b) $\overline{e}_i$ and $\underline{e}_i$ are ahead of e_i and $\overline{e}_{i+1}$ and $\underline{e}_{i+1}$ are ahead of e_{i+1}. In the new algorithm, the stack stores only elementary steps that satisfy the new definition (see Fig. 3). The extra overhead needed for condition (b) is constant per vertex of $\mathcal{A}(H)$. Whenever we perform an elementary step we have to test whether the right endpoints of the two new cut edges correspond to elementary steps according to the new definition that takes constant time. We also have to check whether advancing edges e_i and e_{i+1} makes the right endpoints of $\overline{e}_i$, $\overline{e}_{i+1}$, $\underline{e}_i$, or $\underline{e}_{i+1}$ endpoints of elementary steps.

To prove correctness of the algorithm, we show that the stack does not get empty before the sweep is complete, and that e_i and $\overline{e}_i$ as well as e_i and $\underline{e}_i$ are always aligned (see Fig. 3). The former is true because the leftmost right endpoint of any cut edge always corresponds to an elementary step (also according to the new definition). The latter is true because it holds initially and each elementary step maintains the alignedness of e_i with $\overline{e}_i$ and $\underline{e}_i$. Clearly, every vertical line segment that connects $e_i \cap e_{i+1}$ with $\overline{e}_i$ or $\underline{e}_{i+1}$ intersects k lines. Thus we have the following theorem.

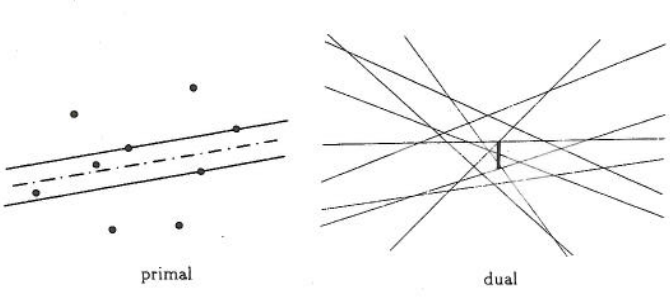

Figure 2. Vertical Line Segments With One Endpoint at a Vertex of $\mathcal{A}(H)$, $H = \{l_i \mid 1 \le i \le n\}$, and the Other on an Edge of the Arrangement. For $n = 9$ points a strip contains $k = 5$ points. If the vertical width of the strip is a minimum, then the middle axis of the strip is an LMS regression line. In dual space, such a strip corresponds to a shortest vertical line segment that intersects $k = 5$ lines. The midpoint of this line segment corresponds to the LMS regression line.

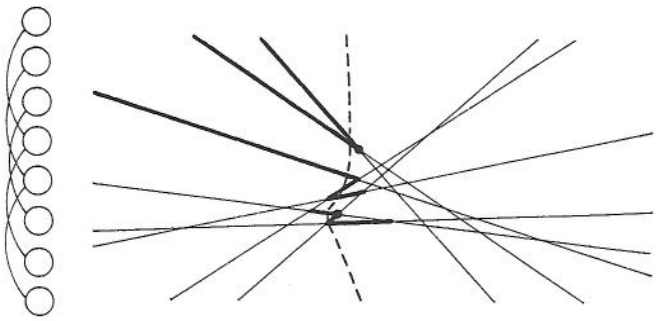

Figure 3. The Arrangement $\mathcal{A}(H)$ and the Alignment Graph. The relevant pairs of cut edges are aligned. Common right endpoints of adjacent pairs of cut edges that satisfy the new definition of an elementary step are emphasized. The graph to the left is the alignment graph (to be defined later) which shows the cut edge dependencies.

Theorem 3.1. A least median of squares regression line of n points in the plane can be computed in time $O(n^2)$ and memory space $O(n)$.

Remark. Recall that an LMS regression line minimizes the median *vertical* distance to the points. To compute a line that minimizes the median *orthogonal* distance instead, we can use the same algorithm with only a small modification: the length of a vertical line segment in dual space is not taken as is but is adjusted to reflect the orthogonal width of the primal strip. If L is the length of the line segment and x is the abscissa of its endpoints, then the adjusted length is $L \cdot \cos(\arctan x) = L/\sqrt{1+x^2}$. Finding a vertical line segment minimizing this new notion of length is no different from before. We thus have a quadratic time linear space algorithm for finding a line that minimizes the median orthogonal distance to the points.

The remainder of this section describes the general method of which the technique that leads to a quadratic time and linear space algorithm for LMS regression lines is a special case. Remember that the main idea was to couple edge e_i with edges $\overline{e}_i$ and $\underline{e}_i$ of the cut and to synchronize their advancement to the right so that e_i is always aligned with $\overline{e}_i$ and $\underline{e}_i$. We make this idea concrete by defining a graph whose nodes are the cut edges and whose arcs are of the form $\{e_i, \overline{e}_i\}$ for $k-1 \le i \le n$ [which includes arcs $\{e_i, \underline{e}_i\}$, since $e_i = (\underline{\overline{e}}_i)$]. We call this graph the alignment graph of the computation.

In the case of the LMS regression line, the alignment graph is like a complete matching except that two or three nodes (depending on whether n is even or odd) are connected to two nodes each. No limit on using other alignment graphs is in sight. If two cut edges are connected by an arc, then the algorithm makes sure that the advancement of both edges is synchronized in the sense that they are always aligned. Using the complete alignment graph

results in a sweep with a vertical and perfectly straight line. Let δ_i be the degree of node e_i and let ε_i be the number of times e_i changes, that is, is involved in an elementary step. Whenever e_i changes, all nodes of the alignment graph connected to e_i are checked in constant time each. This implies that a topological sweep constrained by an alignment graph with node degrees δ_i takes time $O(n^2 + \sum_{i=1}^{n} \delta_i \varepsilon_i)$ and space $O(n + \sum_{i=1}^{n} \delta_i)$. Since $\sum_{i=1}^{n} \varepsilon_i = O(n^2)$, this implies that quadratic time and linear space suffice to support any constant-degree alignment graph. A minimum-height binary tree is an example of such a graph. [Nontrivial results on the behavior of the ε_i can be found in Edelsbrunner 1987 (see also Edelsbrunner and Welzl 1986; Erdös, Lovász, Simmons, and Strauss 1973; Welzl 1986).]

Here is a case of an alignment graph with nonconstant node degrees. The set of arcs is $\{\{e_i, e_j\} \mid i = 1 \text{ or } j = n\}$. The time to support this graph is

$$O\left(n^2 + 2\sum_{i=2}^{n-1} \varepsilon_i + (n-1)(\varepsilon_1 + \varepsilon_n)\right) = O(n^2),$$

because $\varepsilon_1 + \varepsilon_n \leq n + 2$. Using this alignment graph, for each vertex of $\mathfrak{A}(H)$ we can compute the topmost and bottommost edges that intersect the vertical line through the vertex. In addition, for each endpoint of a topmost or bottommost cut edge we get the sequence of lines intersecting the vertical line through this vertex in linear time. This means that, given a set S of n points in the plane, we solve the following problem in time $O(n^2)$ and space $O(n)$: (a) for each point pair compute the smallest strip that contains S and is bounded by two lines parallel to the point pair, and (b) for each edge of the convex hull of S compute the list of points sorted in normal direction.

We remark that there is no reason why a node of the alignment graph should always correspond to the edges at a fixed position of the cut. Applications where each node corresponds to the edges of a fixed line are conceivable.

4. BINARY-SEARCH-DRIVEN TOPOLOGICAL SWEEP

This section presents a method, different from the alignment-graph technique of Section 3, for exploiting the nondeterminism of the topological sweep. We use the following point-location problem to demonstrate the method. Given a set $P = \{p_j = (x_j, y_j) \mid 1 \leq j \leq m\}$ of m points and a set $H = \{l_i : y = a_i x + b_i \mid 1 \leq i \leq n\}$ of n nonvertical lines in the plane, for each p_j determine the line l_i that minimizes $y_j - a_i x_j - b_i \geq 0$ (that is, l_i is the first line p_j hits when it drops vertically downwards). We assume that the line for each point is unique, which is certainly the general case.

The global algorithm that we use for this problem sorts the points from left to right and then performs a topological sweep of the lines. The difficult part of the algorithm is to detect for each point when the sweep passes the edge right below the point. To avoid the danger of passing such an edge without realizing it, we design the topological sweep as an extremely conservative (not to say lazy) process: It does not move unless told so by another process. This other process, which takes care that things get done, performs a binary search in the cut for each point p_j, taking the points in sequence from left to right. A typical comparison in the binary-search process asks whether point p_j lies vertically above, on, or vertically below the ith edge, e_i, of the cut. If p_j and e_i are aligned, then it is easy to provide a satisfying answer in constant time. Otherwise, we have to make sure that the ith edge proceeds to the right until it is aligned with p_j. We do this through the following recursive procedure.

```
while p_j is ahead of e_i do ADVANCE(i) endwhile;
Decide upon the relative position of p_j and e_i.

procedure ADVANCE(i);
begin
  Let l be the line through the right endpoint of e_i and
    let e_i' be the cut edge supported by l.
  while e_i and e_i' do not share the right endpoint do
    ADVANCE(i') endwhile;
  Perform the elementary step defined by e_i and e_i'.
end.
```

Note that if e_i and $e_{i'}$ do not share the right endpoint, then e_i is ahead of $e_{i'}$; otherwise, we get a contradiction to the property that the cut is defined by a topological line.

An important invariant of the algorithm is that the current point, p_j, is aligned with or ahead of every cut edge. This implies the correctness of the algorithm. For the analysis observe that $O(\log n)$ time per point is required for the point/edge comparisons. Time spent to advance the cut can be charged in constant shares to the advanced edges, which implies that the point location problem for m points and n lines can be solved in $O(m \log(m \cdot n) + n^2)$ time and $O(m + n)$ space. This is the currently best result for values of m asymptotically close to n^2; better complexity bounds are possible for smaller values of m (see Edelsbrunner, Guibas, and Sharir 1988).

The solution to the point-location problem suggests that it is possible to influence the topological sweep line so that its sweep satisfies certain requirements necessary for solving a problem. With a different application, for instance, we could require that the topological line is straight at a given x coordinate (that is, there is a point in time at which all n cut edges are aligned with the given point, x_0, on the x axis). This can be achieved by advancing the ith cut edge until it is aligned with x_0, for $1 \leq i \leq n$. The extra overhead for making the sweep straight at x_0 costs $O(n)$ time. To make it straight at m x values thus takes $O(mn + n^2)$ time.

5. DISCUSSION AND OPEN PROBLEMS

An $O(n^2)$-time and $O(n)$-space algorithm for finding a least median of squares regression line for n points in the plane is described in this article. It would be interesting to prove a matching lower bound, although this seems rather difficult. Another interesting problem is to extend the results to three and higher dimensions.

The problem of computing least median of squares

regression lines motivated a study of methods for adding certain straightness conditions when topologically sweeping an arrangement of lines. This study focuses on the gray area between the topological sweep—which takes $O(n^2)$ time—and the conventional sweep with a vertical straight line, which takes $O(n^2 \log n)$ time for n lines. The alignment-graph technique of Section 3 and the externally driven topological sweep of Section 4 are two such methods.

Several computational geometry problems remain for which $O(n^2 \log n)$-time solutions are known based on the conventional sweep method and which successfully resisted our attempts to improve the complexity of $O(n^2)$ using the topological sweep method. In each case, such an improvement hinges on how straight a sweep line is necessary to solve the problem. The input data for each problem is a set of n pairwise disjoint line segments in the plane. The problems are given as follows.

1. Compute a direction of nonintersecting shadows (if it exists); this is a direction so that any line parallel to it intersects at most one line segment. This problem was originally stated by Lee and Preparata (1984), who studied certain shortest-path problems in the plane.
2. Compute a direction of the largest (smallest) shadow. Here we define the size of a shadow as the total length of the orthogonal projection onto a line normal to the direction.
3. Compute a direction so that the shadow is connected.
4. For each direction, compute the best balanced partition of the set of line segments defined by a line that avoids all line segments and is parallel to the direction. Note that the set of directions can be decomposed into $O(n^2)$ intervals so that two directions in the same interval give rise to the same partition. To avoid overhead caused by storing the partitions explicitly, we represent each partition by a line that induces it.
5. For each direction, compute a largest subset of the line segments that meet a common line parallel to the direction. As in problem 4, the infinite set of directions can be decomposed into $O(n^2)$ intervals of invariant answers. A set of line segments meeting a common line is represented by the stabbing line.

We remark that the first three problems seem to require a straight sweep line, because the desired properties are global in the sense that they are defined in terms of *all* lines parallel to a given direction. Problems 4 and 5 are made difficult by the requirement that a piece of information is to be computed for *every* direction.

[*Received August 1988. Revised May 1989.*]

REFERENCES

Edelsbrunner, H. (1987), *Algorithms in Combinatorial Geometry,* Heidelberg: Springer-Verlag.

Edelsbrunner, H., and Guibas, L. J. (1989), "Topologically Sweeping an Arrangement," *Journal of Computer and System Sciences,* 38, 165–194.

Edelsbrunner, H., Guibas, L. J., and Sharir, M. (1988), "The Complexity of Many Faces in Arrangements of Lines and of Segments," in *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry,* New York: Association for Computing Machinery, pp. 44–55.

Edelsbrunner, H., and Mücke, E. P. (1988), "Simulation of Simplicity: A Technique to Cope With Degeneracies in Geometric Algorithms," in *Proceedings of the Fourth Annual ACM Symposium on Computational Geometry,* New York: Association for Computing Machinery, pp. 118–133.

Edelsbrunner, H., and Welzl, E. (1986), "Constructing Belts in Two-Dimensional Arrangements With Applications," *SIAM Journal on Computing,* 15, 271–284.

Erdös, P., Lovász, L., Simmons, A., and Strauss, E. G. (1973), "Dissection Graphs of Planar Point Sets," in *A Survey of Combinatorial Theory,* eds. J. N. Srivastava et al., Amsterdam: North-Holland, pp. 139–149.

Fredman, M. (1976), "On the Information Theoretic Lower Bound," *Theoretical Computer Science,* 1, 355–361.

Lee, D. T., and Preparata, F. P. (1984), "Euclidean Shortest Paths in the Presence of Rectilinear Barriers," *Network,* 14, 393–410.

Levi, F. (1926), "Die Teilung der Projektiven Ebene Durch Gerade Oder Pseudogerade," *Berichte Mathematische-Physikalischer Klassen der Sächsischen Akademie der Wissenschaften Leipzig,* 78, 256–267.

Rousseeuw, P. J. (1984), "Least Median of Squares Regression," *Journal of the American Statistical Association,* 79, 871–880.

Rousseeuw, P. J., and Leroy, A. M. (1987), *Robust Regression and Outlier Detection,* New York: John Wiley.

Souvaine, D. L., and Steele, J. M. (1987), "Efficient Time and Space Algorithms for Least Median of Squares Regression," *Journal of the American Statistical Association,* 82, 794–801.

Steele, J. M., and Steiger, W. L. (1986), "Algorithms and Complexity for Least Median of Squares Regression," *Discrete Applied Mathematics,* 14, 93–100.

Welzl, E. (1986), "More on k-Sets of Finite Sets in the Plane," *Discrete and Computational Geometry,* 1, 95–100.