Proceedings

EIGHTEENTH ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

October 8-10, 1980

Allerton House, Monticello, Illinois
Sponsored by the
Department of Electrical Engineering and the
Coordinated Science Laboratory of the
University of Illinois at Urbana-Champaign

[2] D. E. Knuth, The Art of Computer Programming, V.3, Sorting and Searching. Addison-Wesley, Reading, Mass., 1973.

[3] J. Vuillemin, A unifying look at data structures, CACM 23, 1980, 229-239.

[4] A. C. Yao and F. F. Yao, On Computing the Rank Function for a Set of Vectors, Technical Report R-75-699, Dept. of Computer Science, University of Illinois, 1975.

LOWER BOUNDS FOR GEOMETRIC PROBLEMS

DAVID AVIS
School of Computer Science, McGill University, Montreal, Quebec H3A 2K6

ABSTRACT

Many lower bounds have been given for 1-dimensional geometric problems under the linear decision tree model. This model is too weak in general to give useful results in higher dimensions. It is shown how to extend such bounds to a decision tree model that includes the primitives usually found in geometric algorithms. A second result is a short proof of a theorem of Yao and Rivest giving a lower bound for the polyhedral decision problem.

1. DECISION TREES

One of the most powerful models for obtaining lower bounds for computational problems is the *decision tree model*. In this model, algorithms are ternary trees, with each internal node representing a test of the form "$f(z_1,\ldots,z_m) : c$". The function f is from a specified class, c is a constant, and $z_1,\ldots,z_m$ are the input numbers. The output of the algorithm takes place at the leaf nodes, and is either a "yes" or "no" answer in the case of a decision problem, or an evaluation of functions from the specified class.

When f is restricted to the class of linear functions, this model is known as the *linear decision tree* model. If the input numbers are real numbers, it is easily seen that the leaf nodes are convex regions of input points. This fact has yielded many lower bounds under the linear decision tree model. The *cost* of a decision tree algorithm is the height of the tree, which is the number of tests required in the worst case.

This paper deals with lower bounds for geometric problems, where the input numbers are usually real numbers of unlimited precision. For example, they may be the co-ordinates of a set of points $x_1,\ldots,x_n$ in k-dimensional euclidean space, R^k. Primitives used in geometric algorithms are usually one of the following three types:

(a) "Is $d_p(x_i,x_j) \geq d_p(x_r,x_s)$?" where i,j,r,s are integers in the range 1 to n and d_p is the pth order Minkowski metric.

(b) "On which side of the hyperplane generated by $\{x_{i_j} : j=1,2,\ldots,k,$ $1 \leq i_j \leq n\}$ does the point x_r lie?"

(c) "$f(x_1,\ldots,x_n) : c$", where f is a linear function and c is constant.

Tests (a) – (b) can be implemented by linear tests only in 1-dimensional space (ie. $k=1$). In fact, as was shown in [1], the linear decision tree model is not strong enough even to decide whether three points in the plane are collinear. Since the proof is short, it is included here for completeness.

Consider any linear decision tree algorithm for deciding whether the points (x_1,y_1), (x_2,y_2), and (x_3,y_3) are collinear. Such a tree must decide whether

$$x_3(y_1-y_2)+y_3(x_2-x_1)+y_2x_1-y_1x_2=0 \; . \qquad (1)$$

Since this is a quadratic function, it is always possible to pick collinear points so that $z=(x_1,y_1,x_2,y_2,x_3,y_3)$ does not lie on any of the hyperplanes defined by internal nodes of the decision tree. Therefore z lies in the interior of some 6-dimensional convex set defined by a leaf node of the tree. By perturbing z slightly, we can destroy the collinearity whilst remaining at the same leaf node. Thus the algorithm fails.

When $k=1$, the linear decision tree model is powerful enough to yield efficient algorithms. Several authors have found sharp linear decision tree bounds, examples of which are given in Section 2. At this stage it is common to make the following statement:

since the lower bound holds for 1-dimensional problems, it must surely hold for all higher dimensional problems.

Whilst this statement is true, it omits the important fact that the *bound is only true under the linear decision tree model*, and is thus useless in general for $k\geq 2$, as shown above. In Section 3 we attempt to overcome this difficulty by showing that linear decision tree bounds for 1-dimensional problems can be used to give bounds for higher dimensional problems, under a decision tree model that includes the primitives (a)-(c) above. Finally, in Section 4, we give a short proof of a theorem of Yao and Rivest [9] on the complexity of deciding whether a point is contained in a polyhedron.

2. SOME EXAMPLES

We begin by presenting three lower bound results for sets of numbers.

ELEMENT UNIQUENESS: (Dobkin, Lipton [4]) Any linear decision tree algorithm for determining whether a set of n real numbers are distinct requires $\Omega(n \log n)$ tests, in the worst case.

INTERSECTION: (Reingold [7]) Any linear decision tree algorithm for determining whether two sets of n numbers has an empty

intersection requires $\Omega(n \log n)$ tests, in the worst case.

ϵ-CLOSENESS: (Fredman, Weide [5]) Any linear decision tree algorithm for determining whether any two of n numbers are within ϵ apart requires $\Omega(n \log n)$ tests, in the worst case.

Consider now, the following set of geometric problems. In each case, it has been shown that one of the preceding bounds is applicable for the 1-dimensional case.

CLOSEST PAIR: (Shamos [8]) Given a set of n points in k-space, determine the pair whose distance is smallest. (Use ELEMENT UNIQUENESS).

FIXED RADIUS NEAREST NEIGHBOUR: (Bentley [2]) Given a set of n points in k-space, and some constant d, list all pairs of points whose distance is no more than d. (Use ϵ – CLOSENESS)

SEGMENT INTERSECTION: (Shamos [8]) Given a set of n line segments in k-space, determine if any pair of segments intersect. (Use ELEMENT UNIQUENESS).

SET DISTANCE: (Bhattacharya [3]) Given two sets of n points in k-space, determine the pair of points (one from each set) whose distance is smallest. (Use INTERSECTION).

Algorithms known to the author for solving the four geometric problems above, use primitives of the form (a)-(c) of Section 1. In the next section we show how to extend the lower bounds above to this class of primitives.

3. LIFTING LOWER BOUNDS TO HIGHER DIMENSIONS

Consider problems where the input is an n-tuple $X_k=(x_1,\ldots,x_n)$ of n points in k-space. Let $f\colon R^k\times\ldots\times R^k \longrightarrow R$ be a real valued function of nk variables, on the input data. Let $E^k = R \times 0^{k-1}$, be the set of points in R^k that have zeros in every co-ordinate, except possibly the first. Then we say that f is *1-linear*, if the function $f\colon E^k\times\ldots\times E^k \rightarrow R$ is a linear function. It is easily shown that tests of type (a)-(c) in Section 1 are implementable by 1-linear functions. A *1-linear decision tree* is a decision tree where all tests are by 1-linear functions.

Consider a sequence of geometric problems $\pi = \pi_1,\pi_2,\pi_3 \ldots$ on inputs $X_1,X_2,X_3,\ldots$. π_k is a problem on k-dimensional inputs; its output on X_k is denoted $\pi_k(X_k)$. π is said to be (1-dimensionally) *consistent* if

$$\pi_1(x_1,\ldots,x_n) = \pi_k(y_1,\ldots,y_n) \;\; \forall (x_1,\ldots,x_n)\in R^n,\; k=2,3,4,\ldots,$$

where

$$y_i = (x_i, 0, \ldots, 0) \in E^k .$$

It can be verified that the four geometric problems in section 2 are 1-dimensionally consistent. We can now state and prove the following result.

Theorem 1. If π is a consistent sequence and π_1 has a linear decision tree bound of $\Omega(g(n))$, then π_k has a 1-linear decision tree bound of $\Omega(g(n))$.

Proof: Let T_k be a 1-linear decision tree algorithm for π_k. Let f_j be the 1-linear function at some node j of T_k. It follows from the definitions that there exists some function $h: R^n \to R$ such that $f(y_1,\ldots,y_n) = h(x_1,\ldots,x_n)$, $\forall (x_1,\ldots,x_n) \in R^n$, where $y_1,\ldots,y_n$ is defined above. Since f is 1-linear, it follows that h is linear. Construct a linear decision tree T_1 by substituting h_j for f_j at each node j of T_k. Since π_k is consistent, it follows that T_1 is a linear decision tree algorithm for π_1, and hence has height at least $\Omega(g(n))$. Thus T_k has height at least $\Omega(g(n))$ and the theorem follows. □

Corollary The problems CLOSEST PAIR, FIXED RADIUS NEAREST NEIGHBOUR, SEGMENT INTERSECTION and SET DISTANCE have $\Omega(n \log n)$ 1-linear decision tree bounds.

Thus any improvement in $O(n \log n)$ algorithms for these problems must use primitives other than those listed (a)-(c) in Section 1.

4. THE POLYHEDRAL DECISION PROBLEM

Let $A \in R^{m\times n}$ be an m×n matrix, and let $b \in R^m$ be an m element vector. The set $P = \{x \in R^n : Ax \le b\}$ is called a *polyhedron*. P is the intersection of m *half-spaces* defined by the m rows of A. Let C(P) denote the minimum height of any linear search tree algorithm T(P) for resolving the decision problem Q(x): "Is $x \in P$?". We call any leaf-node $t \in T(P)$ that gives the output "yes", a *yes-leaf*. For each yes-leaf,t,define the polyhedron, P_t, by

$$P_t = \text{closure } \{x \in R^n : T(P) \text{ resolves } Q(x) \text{ at } t\} .$$

It is easy to construct P_t given T(P). Simply trace the path in T(P) from t to the root, converting any "<" to "≤" and any ">" to "≥". Any equality encountered can be replaced by two inequalities.

We require a few definitions from the field of convex sets, the reader is referred to the book by McMullen and Shephard for further information [6].

A *supporting hyperplane* H of P is a hyperplane such that (i) $H \cap P \neq \emptyset$ and (ii) P is contained in one of the closed halfspaces determined by H. The set $H \cap P$ is then called a *face* F of P. If F has dimension s, then it is called an *s-face*. It is well known that an s-face F can be written as the intersection of n-s of the halfspaces that define P. Thus if the path from a yes-leaf t to the root involves d inequalities, P_t can have at most $\binom{d}{n-s}$ s-faces. The Upper Bound Theorem [6] gives a much sharper bound on this number. We may now state and prove the following result of Yao and Rivest [9].

Theorem 2. $2^{C(P)} \binom{C(P)}{n-s} \ge f_s(P)$.

Proof: Let T(P) be a linear decision tree algorithm for Q(x), and let F be an s-face of P. Since the number of yes-leaves is finite, there must be some yes-leaf t such that $F_t = P_t \cap F$ is an s-dimensional convex set. We will show that F_t is an s-face of P_t. First observe that $P_t \subseteq P$, since each point $y \in P_t$ is the limit of a sequence $\{y_j\}$ of points resolved at the yes-leaf t, and hence in the closed set P. Further $F = P \cap H$, for some supporting hyperplane H. Thus $F_t = P_t \cap H$ and H is a supporting hyperplane for P_t. Hence F_t is an s-face of P_t.

The theorem now follows from the fact that P_t can have at most $\binom{C(P)}{n-s}$ s-faces, and the fact that there are at most $2^{C(P)}$ yes-leaves. □

5. REFERENCES

[1] D. Avis, Comments on a lower bound for convex hull determination, *Information Processing Letters*, to appear.

[2] J. Bentley, Multidimensional divide and conquer, *CACM* 23, 4 (1980).

[3] B. Bhattacharya, The application of computational geometry to pattern recognition problems, *Ph.D. Thesis Proposal*, McGill University, March 1980.

[4] D. Dobkin, R. Lipton and S. Reiss, *Excursions into Geometry*, Computer Science Tech. Rep. 71, Yale University, New Haven, CT. 1976.

[5] M. Fredman and B. Weide, On the complexity of computing the measure $U[a_i,b_i]$, *CACM* 21, 7 (1978) 540-544.

[6] P. McMullen and G. Shephard, *Convex Polytopes and the upper bound conjecture*, Cambridge University Press, 1971.

[7]. E. Reingold, On the optimality of some set algorithms, *JACM* 19, 4 (1972) 649-659.

[8] M. Shamos, *Computational Geometry*, Ph.D. Thesis, Yale University (1978).

[9] A. Yao and R. Rivest, On the polyhedral decision problem, *SIAM J. Computing* 9, 2 (1980) 343-347.

An Analysis of Two Heuristics for the Euclidean Traveling Salesman Problem[1]

Jon Louis Bentley
James B. Saxe
Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

Abstract -- In this paper we study the performance of the *Nearest Neighbor* and *Greedy* heuristics for constructing traveling salesman tours of N cities (points) in the plane. Our first result is negative: we give a particular configuration of cities such that the ratio of heuristic tour length to optimal is $\Theta[(\lg N)/(\lg \lg N)]$, for both heuristics. We then show that if the cities are constrained to lie within the unit square, then both tour lengths are $O(\sqrt{N})$, which implies that the tours are almost surely within a constant factor of optimal if the point sets are randomly drawn from a uniform distribution. Simulation results show that Nearest Neighbor tours are usually very close to optimal. Finally, we discuss several efficient implementations of the Nearest Neighbor heuristic.

1. Introduction

The traveling salesman problem of N cities (points) in the plane calls for constructing a polygon through the cities of minimum perimeter; Papadimitriou [1977] has shown that this problem is NP-complete. Karp [1977] has given an asymptotically efficient algorithm for producing near-optimal tours, but his algorithm is difficult to code and has a high constant factor. In this paper we will investigate the following two heuristics for constructing traveling salesman tours:

- **The Nearest Neighbor Heuristic:** Start at an arbitrary city, and always proceed to the closest unvisited city; when all cities have been visited, return to the starting city.
- **The Greedy Heuristic:** Consider the $\binom{N}{2}$ edges of the graph in increasing order of length, and add each edge to the tour if it will not prohibit the current tour from being extended to a complete tour.

The performance of the Nearest Neighbor heuristic when applied to graphs satisfying the triangle inequality has been studied by Rosenkrantz, Stearns and Lewis [1977]; they showed that its tour is never more than a logarithmic factor from optimal, and gave a family of graphs that exhibit this behavior. They did not, however, consider the performance of the heuristic on graphs induced by point sets, which are a proper subclass of the graphs they studied. In this paper we will study the performance of the above heuristics on cities in the plane.

In Section 2 we discuss the efficacy of the tours produced by the two heuristics. We then turn in Section 3 to efficient implementations of the Nearest Neighbor heuristic. Finally, conclusions are offered in Section 4.

[1]This research was supported in part by the Office of Naval Research under contract N00014-76-C-0370.