Discrete Comput Geom 2:37-48 (1987)

Discrete & Computational Geometry

A Polynomial-Time Linear Decision Tree for the Traveling Salesman Problem and Other NP-Complete Problems

Martin Kolinek*

Department of Mathematics, University of Washington, Seattle, WA 98195, USA

Communicated by Victor Klee

Abstract. We show that the position of an input point in the Euclidean d-dimensional space with respect to a given set of hyperplanes can be determined efficiently by linear decision trees. As an application, we prove that many concrete problems whose recognition versions are NP-complete, like the traveling salesman problem, many other shortest path problems, and integer programming, have polynomial-time upper bounds in the linear decision tree model of computation.

1. Introduction

In this paper we present a theorem providing an upper bound on computational complexity of a class of problems in the linear decision tree model of computation. The class consists of problems to decide the position of an input point in $\mathbf{R}^d$ with respect to a given set of hyperplanes $H_1, \ldots, H_k$ in $\mathbf{R}^d$, where the hyperplanes have equations with integer coefficients in Cartesian coordinates of $\mathbf{R}^d$. The theorem extends a similar result obtained by Meyer auf der Heide in [5].

In the proof of the theorem, a linear decision tree is constructed in three steps. In the first step, we bring the input point into a cube near the origin of $\mathbf{R}^d$ via a projective transformation. In the second step, we perform a sequence of binary searches inside that cube. After the searches, we have only to decide the position of the transformed input point with respect to a set of hyperplanes having a nonempty intersection. In the third step, we reduce this problem into a problem one dimension lower, which is solved by a recursive call.

A projective geometry approach—the use of homogeneous coordinates—is employed. However, we could not use the analytic projective space $\mathbf{PR}^d$ as the

* Present address: IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA.

underlying space for our discussions, because a hyperplane in $\mathbf{PR}^d$ does not divide $\mathbf{PR}^d$ into two regions. The advantage of homogeneous coordinates over nonhomogeneous (i.e., Cartesian) coordinates is that we do not need to pay attention to technical problems such as that two hyperplanes may be either parallel or intersecting, which arise when nonhomogeneous coordinates are used.

The theorem immediately yields polynomial-time linear decision trees for optimization versions of some combinatorial problems whose recognition versions are NP-complete, e.g., the traveling salesman problem (TSP), various shortest paths problems, the knapsack problem with real (positive or negative) weights, and integer linear programming.

The computational complexity of TSP in the decision tree model was considered by Rabin in [7]. There, he considers three levels of computations: finding a solution, checking a solution, and proof of a solution. The "finding" level, computationally the most difficult, means to actually find the right solution out of a set of many potential solutions. The "checking" level means to check whether a proposed solution is actually the right solution, e.g., given a simple cycle through all vertices of a weighted graph, check whether this cycle is the solution of TSP. The third level, computationally easiest, means to prove that a proposed solution is actually the right solution; the effort related to the discovery of the proof is not part of the proof. In that paper, Rabin established that proof of a solution of TSP can be done in polynomial time. In contrast, our linear decision tree finds a solution of TSP in polynomial time and thus we improve this result of Rabin.

This paper is a generalization of Meyer auf der Heide's paper [5], where the nonnegative knapsack problem is considered. Meanwhile, Meyer auf der Heide has also (independently) generalized [5], see [6].

Here, our goal is achieved by generalizing Theorem 2 of [5] in two ways. First, we remove the restriction limiting the input vectors to a cube. Second, we replace the recognition of whether or not the input $\vec{x}$ is member of a union of some hyperplanes in space by deciding the position of the input $\vec{x}$ with respect to each of the hyperplanes. The second generalization is fairly easy to accomplish, since the method of proof of Theorem 2 in [5] is well suited not only for the above-mentioned recognition but also for the decision of position with respect to the hyperplanes. We note that the implementation of our two generalizations is facilitated if we use homogeneous coordinates.

2. Definitions and Notation

Let $\mathbf{R}^d$ be the d-dimensional Euclidean space. If $x_1, \ldots, x_d$ are Cartesian coordinates in $\mathbf{R}^d$, then the hyperplanes $x_j = 0, j = 1, \ldots, d$, are called the *hyperplanes of coordinates*. The closed cube bounded by the hyperplanes $x_j = \pm 1, j = 1, \ldots, d$, is called the *standard cube* of $\mathbf{R}^d$.

Let $\mathbf{S} = \{H_1, \ldots, H_k\}$ be a nonempty set of hyperplanes in $\mathbf{R}^d$. If $\bigcap_{i=1}^{k} H_i \neq \emptyset$, then we say that $\mathbf{S}$ is *central*. Following [5], we say that a number $r > 0$ is a *coarseness* of $\mathbf{S}$ if, for every open ball B in $\mathbf{R}^d$ with radius r, it holds that all hyperplanes of $\mathbf{S}$, which intersect ball B, form a central set.

A trivial result is that if $r > 0$ is a coarseness of S, then any r', where $0 < r' < r$, is also a coarseness of S. This result justifies using one value for coarseness throughout the construction of the decision tree in Section 3 of this paper. A value for coarseness is given in the following lemma, the proof of which can be found in [5].

Lemma (Meyer auf der Heide). *Let* $\mathbf{S} = \{H_1, \ldots, H_k\}$ *be a set of hyperplanes in* $\mathbf{R}^d$. *Assume that* H_i *has equation* $\sum_{j=1}^{d} a_{ij}x_j + a_{(d+1)} = 0$, *where the coefficients* a_{ij}, $1 \le j \le d+1$, *are integers. Let*

$$M(\mathbf{S}) = \max\{|a_{ij}|: 1 \le i \le k, 1 \le j \le d+1\}.$$

Then $(M(\mathbf{S})^{2d^2} \cdot d^{2d^2})^{-1}$ *is a coarseness of* $\mathbf{S}$.

For our purposes, it seems advantageous to use the so-called homogeneous coordinates instead of ordinary Cartesian coordinates. For readers unfamiliar with homogeneous coordinates, the following example may be illustrative.

In elementary analytic geometry, the (nonhomogeneous) equation

$$x_1 + 2x_2 + 3 = 0$$

describes a line in the plane. The point P with Cartesian coordinates $(-7, 2)$ is on that line. The vector $V = -2\vec{i} + \vec{j}$ (where $\vec{i}, \vec{j}$ are unit vectors in positive directions of the coordinate axes) is parallel to the line or, in other words, can be placed on the line. In homogeneous coordinates, the same line has homogeneous equation

$$x_1 + 2x_2 + 3x_3 = 0.$$

The point P has homogeneous coordinates $(-7, 2, 1)$ or any nonzero multiple of this triple, e.g., $(-21, 6, 3)$. Thus homogeneous coordinates of point P are not unique. Cartesian coordinates of P can be recovered from homogeneous coordinates of P by dividing the first and second homogeneous coordinate by the third one, which is nonzero. The vector V has homogeneous coordinates $(-2, 1, 0)$. Clearly, this triple satisfies the homogeneous equation of our line and we can say simply that V is on the line. The homogeneous coordinates of vector V are unique. In summary, let $(x_1, x_2, x_3) \in \mathbf{R}^3$. If $x_3 = 0$, then (x_1, x_2, x_3) represents the vector $x_1\vec{i} + x_2\vec{j}$ in the plane. If $x_3 \neq 0$, then (x_1, x_2, x_3) represents the point with Cartesian coordinates $(x_1/x_3, x_2/x_3)$ in the plane. All triples satisfying the homogeneous equation of our line then represent all points and all vectors, including the zero vector, on the line. The triple $(1, 2, 3)$ is called the sequence of homogeneous coordinates of our line. In general, any nonzero triple $(a_1, a_2, a_3) \in \mathbf{R}^3$ represents a line. If $a_1 = a_2 = 0$, then the triple represents the so-called line at infinity. The line at infinity contains all vectors of the plane, but no points. If $a_1 \neq 0$ or $a_2 \neq 0$, then the triple represents an ordinary line in plane. This example generalizes in an obvious way to higher-dimensional spaces.

For more on homogeneous coordinates and analytic projective geometry in d-space; see, for example, [8] and [9].

For notational convenience we will use letters $\vec{a}, \vec{b}, \ldots$ (resp. $\vec{z}, \vec{y}, \ldots$) to denote row (resp. column) vectors of homogeneous coordinates. As mentioned in the introduction, our goal is to establish an upper bound to the "decision problem" in the linear decision tree model of computation. The formal definitions of the "decision problem" and the linear decision tree model follow.

Let $\mathbf{S} = \{\vec{a}_1, \ldots, \vec{a}_k\} \subseteq \mathbf{Z}^{d+1}$ be a nonempty set of row $(d+1)$-vectors with integer entries. Here $\mathbf{Z}$ denotes the set of integers. Given an input $\vec{x} \in \mathbf{R}^{d+1}$, the *integer linear decision problem* (ILDP) is to find, for each $i = 1, \ldots, k$, a relation $\rho_i \in \{<, =, >\}$ such that $\vec{a}_i \vec{x} \, \rho_i \, 0$. In other words, the problem is to find signatures of $\vec{a}_i \vec{x}$ for $i = 1, \ldots, k$.

A *linear decision tree solving the ILDP* is a ternary tree, whose internal nodes contain linear tests of the form $\vec{b}\vec{x}:0$, where $\vec{b} \in \mathbf{R}^{d+1}$ and the colon means "compare to". Each test has three possible outcomes: $<$ or $=$ or $>$. Each leaf contains a solution of the ILDP, i.e., a vector $\vec{\rho} \in \{<, =, >\}^k$. An input $\vec{x} \in \mathbf{R}^{d+1}$ starts at the root and traverses down the tree. At each internal node a branching is made according to the test at that node. When $\vec{x}$ reaches a leaf, the vector $\vec{\rho}$ at that leaf provides the correct solution.

The running time $\text{cost}(\vec{x})$ of the input $\vec{x}$ is the number of tests performed until $\vec{x}$ reaches a leaf. The running time of the tree is $\max_{\vec{x} \in \mathbf{R}^{d+1}} \text{cost}(\vec{x})$.

3. The Main Theorem

We are now ready to state and prove the main theorem.

Theorem. *Let $\mathbf{S} = \{\vec{a}_1, \ldots, \vec{a}_k\} \subseteq \mathbf{Z}^{d+1}$ be a nonempty set of row $(d+1)$-vectors with integer entries. Then there is a linear decision tree solving ILDP whose running time is at most*

$$2d^4 \log_2 d + 2d^4 \log_2 M(\mathbf{S}) + O(d^3),$$

where $M(\mathbf{S}) = \max\{|a_{ij}|: 1 \le i \le k, 1 \le j \le d+1\}$.

Proof. Let $\vec{x} = (x_1, \ldots, x_{d+1})^{tr} \in \mathbf{R}^{d+1}$ be the input vector. A linear decision tree solving ILDP is constructed in the following three steps:

Step 1. We find l so that

$$|x_l| = \max\{|x_1|, \ldots, |x_{d+1}|\}.$$

We need d comparisons of the form $|x_\alpha| : |x_\beta|$ to find the maximum. This comparison can be done by two linear tests $x_\alpha : \pm x_\beta$. Next, we perform test $x_l : 0$. If $x_l = 0$, then $\vec{x}$ is the zero vector and the problem has a trivial solution.

Hence assume $x_l \neq 0$. Let A be the $(d+1)\times(d+1)$ nonsingular matrix

$$
\begin{array}{l}
\\
l^{\text{th}} \rightarrow \\
\\
\\
\\
\end{array}
\left[\begin{array}{ccc|ccc}
1 & & & & & 0 \\
& 1 & & & & \\
& & \ddots & & & \\
& & 1 & & & \\
\hline
& & & 0 & & 1 \\
& & & & 1 & \\
& & & & & \ddots \\
& & & & & 1 \\
0 & & & \pm 1 & & 0
\end{array}\right]_{(d+1)\times(d+1)}
$$

where in the bottom row we take $+1$ (resp. -1) if $x_l > 0$ (resp. $x_l < 0$). We define $\vec{u} = A\vec{x}$ and $\vec{b}_i = \vec{a}_i A^{-1}$ for $1 \leq i \leq k$. This is the "projective transformation" mentioned in the introduction (see [2, p. 4]). We have

$$\vec{a}_i\vec{x} = \vec{a}_i A^{-1} A\vec{x} = \vec{b}_i\vec{u}.$$

Hence we need to find signatures of $\vec{b}_i\vec{u}$ for $1 \leq i \leq k$. Also note that

$$\vec{u} = (x_1, \ldots, x_{l-1}, x_{d+1}, x_{l+1}, \ldots, x_d, |x_l|)^{\text{tr}}.$$

Let C be the standard cube of $\mathbf{R}^d$. By our choice of l, $\vec{u}$ provides the homogeneous coordinates of a point P in C, where P has Cartesian coordinates

$$\left(\frac{x_1}{|x_l|}, \ldots, \frac{x_{l-1}}{|x_l|}, \frac{x_{d+1}}{|x_l|}, \frac{x_{l+1}}{|x_l|}, \ldots, \frac{x_d}{|x_l|}\right).$$

Note that altogether $2d+1$ linear tests are performed in Step 1.

Step 2. The vectors $\vec{b}_i$, $1 \leq i \leq k$, can be divided into two groups. The first group consists of vectors $\vec{b}_i$, whose first d entries $b_{i1}, \ldots, b_{id}$ are all equal to zero. Hence for these vectors we have $\vec{b}_i\vec{u} = b_{i(d+1)}|x_l|$. But then we know the signatures of $\vec{b}_i\vec{u}$ for all vectors from the first group, because these signatures depend, at this stage of the decision tree, on $b_{i(d+1)}$'s only and not on the input $\vec{x}$. The remaining vectors constitute the second group. We can assume without loss of generality (by renaming $\vec{b}_i$'s if necessary) that vectors $\vec{b}_{t+1}, \ldots, \vec{b}_k$ are the first group and that vectors $\vec{b}_1, \ldots, \vec{b}_t$ are the second group. Notice that the second group is formed by exactly those vectors which represent an ordinary hyperplane in $\mathbf{R}^d$. We denote these hyperplanes by $H_1, \ldots, H_t$.

Thus we need to find signatures of $\vec{b}_i\vec{u}$ for $1 \leq i \leq t$ only. Let

$$M(\mathbf{S}) = \max\{|a_{ij}|: 1 \leq i \leq k, 1 \leq j \leq d+1\}.$$

Then

$$M(\mathbf{S}) \geq \max\{|b_{ij}|: 1 \leq i \leq t, 1 \leq j \leq d+1\},$$

because $|b_{ij}|$'s are just permuted $|a_{ij}|$'s and $t \leq k$. By the lemma, a coarseness of $H_1, \ldots, H_t$ in $\mathbf{R}^d$ is $r = (M(\mathbf{S})^{2d^2} \cdot d^{2d^2})^{-1}$. We partition the standard cube C in $\mathbf{R}^d$ uniformly into smaller cubes with edge-length 2^{-m}, where

$$m = \left\lceil \log_2 \left(\frac{\sqrt{d}}{r} + 1 \right) \right\rceil - 1,$$

by cutting C by hyperplanes parallel to the hyperplanes of coordinates in $\mathbf{R}^d$. A short calculation below shows that the diagonal of a d-dimensional cube of edge-length 2^{-m}, with m as above, is shorter than $2r$:

$$\begin{aligned} \text{diagonal} &= \sqrt{d} \cdot \text{edge-length} \\ &= \sqrt{d} \cdot 2^{-\lceil \sqrt{d}/r+1) \rceil + 1} \\ &\leq 2\sqrt{d} \cdot 2^{-\log_2(\sqrt{d}/r+1)} \\ &= 2\sqrt{d} \cdot \frac{1}{\sqrt{d}/r+1} \\ &< 2\sqrt{d} \cdot \frac{1}{\sqrt{d}/r} \\ &= 2r. \end{aligned}$$

Therefore our choice of m guarantees that each smaller cube obtained by partitioning is contained in an open ball with radius r.

We find a smaller cube C', obtained by partitioning, which contains the point P. Notice that C' need not be unique since smaller cubes are closed and therefore have intersecting boundaries. We can find C' by performing d binary searches (one search for each dimension of the cube C). Each binary search takes $m+1$ tests, because we start with edge-length 2 and stop when the edge-length is 2^{-m}. Each test of the binary search is of the form $x_\beta / |x_l| : \text{const}$. However, we have performed the test $x_l : 0$ in Step 1 and thus we know if $|x_l| = +x_l$ or $-x_l$. Hence just one linear test $x_\beta : \text{const}\, x_l$ is sufficient. Altogether we perform

$$d(m+1) = d \cdot \left\lceil \log_2 \left(\frac{\sqrt{d}}{r} + 1 \right) \right\rceil$$

linear tests.

We can assume without loss of generality (by renaming H_i's and b_i's if necessary) that

$$H_i \cap C' \neq \varnothing \qquad \text{for} \quad 1 \leq i \leq s,$$

and

$$H_i \cap C' = \varnothing \qquad \text{for} \quad s+1 \leq i \leq t.$$

In the latter case, C' is entirely in one of the two open halfspaces determined by H_i. Hence, for any homogeneous coordinates $\vec{z} \in \mathbf{R}^{d+1}$ representing a point in C' with $z_{d+1} > 0$, we have either always $\vec{b}_i \vec{z} < 0$ or always $\vec{b}_i \vec{z} > 0$. Thus the signatures of $\vec{b}_i \vec{u}$, $s+1 \le i \le t$, are known at this stage of the decision tree.

Hence we need to find the signatures of $\vec{b}_i \vec{u}$ for $1 \le i \le s$. By the definition of coarseness, the hyperplanes $H_1, \ldots, H_s$ are central, i.e., there is a point $Q \in \bigcap_{i=1}^{s} H_i$. Let

$$\vec{y} = (y_1, \ldots, y_d, 1)^{\mathrm{tr}} \in \mathbf{R}^{d+1}$$

be a vector of homogeneous coordinates of Q. Hence we have $\vec{b}_i \vec{y} = 0$ for $1 \le i \le s$.

Step 3. Let $\vec{p} = \vec{u} - |x_l| \vec{y}$ be a column vector. Let B be the $d \times (d+1)$ matrix

$$\left[\begin{array}{ccccc:c} 1 & & & & 0 & 0 \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ 0 & & & & 1 & 0 \end{array}\right]_{d \times (d+1)}$$

The reader can check that

$$B^{\mathrm{tr}} B = \left[\begin{array}{ccccc:c} 1 & & & & & 0 \\ & 1 & & & & \\ & & \ddots & & & \\ & & & 1 & & \\ & & & & 1 & \\ \hdashline 0 & & & & & 0 \end{array}\right]_{(d+1) \times (d+1)}$$

We define $\vec{v} = B\vec{p}$ and $\vec{c}_i = \vec{b}_i B^{\mathrm{tr}}$ for $1 \le i \le s$. We have for $1 \le i \le s$:

$$\begin{aligned} \vec{b}_i \vec{u} &= \vec{b}_i (\vec{u} - |x_l| \vec{y}) \\ &= \vec{b}_i \vec{p} \\ &= \vec{b}_i B^{\mathrm{tr}} B \vec{p} \\ &= \vec{c}_i \vec{v}, \end{aligned}$$

where the third equality follows from $p_{d+1} = 0$, which is true because by definition of $\vec{p}$ we have

$$\begin{aligned} p_{d+1} &= u_{d+1} - |x_l| y_{d+1} \\ &= |x_l| - |x_l| \cdot 1 \\ &= 0. \end{aligned}$$

We have reduced the original problem into a new problem one dimension lower:

Given $\vec{c}_i \in \mathbf{R}^d$, $1 \le i \le s$, and input $\vec{v} \in \mathbf{R}^d$, find the signatures of $\vec{c}_i\vec{v}$ for $1 \le i \le s$.

We can solve the new problem by a recursive call. Note that the coarseness r defined in Step 2 can be used without change in subsequent recursive calls, because $\vec{c}_i = \vec{a}_i A^{-1} B^{\mathrm{tr}}$ implies that $\max|c_{ij}| \le \max|a_{ij}|$. Also observe that a linear test in $\vec{v}$ is actually a linear test in $\vec{x}$, because $\vec{v} = B\vec{p} = B(A\vec{x} - |x_i|\vec{y})$.

To end the proof, we estimate the running time of the constructed tree. Let $T(d)$ be the running time of the tree for the d-dimensional input (i.e., $\vec{a}_i$'s and $\vec{x}$ are $(d+1)$-tuples of homogeneous coordinates).

If $d \ge 2$, we get a recursive relation:

$$T(d) \le (2d+1) + d \cdot \left\lceil \log_2\left(\frac{\sqrt{d}}{r} + 1\right)\right\rceil + T(d-1).$$

When $d = 1$, we perform Steps 1 and 2 as described. However, a central set of hyperplanes in $\mathbf{R}^1$ contains at most one hyperplane. Thus we can finish by at most one additional test. Hence

$$T(1) \le 3 + \left\lceil \log_2\left(\frac{1}{r} + 1\right)\right\rceil + 1.$$

Solving the recurrence we obtain

$$T(d) \le d^2 \cdot \left\lceil \log_2\left(\frac{\sqrt{d}}{r} + 1\right)\right\rceil + O(d^2).$$

Substituting $r = (M(\mathbf{S})^{2d^2} \cdot d^{2d^2})^{-1}$, we obtain

$$T(d) \le 2d^4 \log_2 d + 2d^4 \log_2 M(\mathbf{S}) + O(d^3). \qquad \square$$

The theorem can be readily recast into a Cartesian setting:

Corollary. *Let $\mathbf{S} = \{H_1, \ldots, H_k\}$ be a nonempty set of hyperplanes in $\mathbf{R}^d$. Assume H_i has equation $\sum_{j=1}^{d} a_{ij}x_j + a_{i(d+1)} = 0$, where the a_{ij}'s are integers and x_j's are Cartesian coordinates in $\mathbf{R}^d$. Then there is a linear decision tree, performing affine tests of the form $\sum_{j=1}^{d} b_j x_j + b_{d+1} : 0$, which determines the position of an input $\vec{x} \in \mathbf{R}^d$ with respect to all hyperplanes in $\mathbf{S}$ in time at most*

$$2d^4 \log_2 d + 2d^4 \log_2 M(\mathbf{S}) + O(d^3),$$

where $M(\mathbf{S}) = \max\{|a_{ij}|: 1 \le i \le k, 1 \le j \le d+1\}$.

An important feature of the upper bound in the theorem or corollary is that k is not explicitly in the upper bound formula. Instead, $\log_2 M(\mathbf{S})$ does appear. In practical applications, $\log_2 M(\mathbf{S})$ is relatively small, whereas k is exponential in d.

4. Applications

The linear decision tree is a good model of computation for proving lower bounds for many concrete problems in which the input data are treated just as real numbers and various solutions of the problem correspond to a partitioning of the "space of inputs" into regions bounded by hyperplanes. But among problems of such nice geometrical character are also variations of finding shortest paths (e.g., the traveling salesman problem), the knapsack problem, and integer linear programming, which are believed to be hard combinatorial problems, especially when the input numbers are allowed to be positive or negative. We will show on a few examples how our theorem or its corollary easily yield polynomial-time upper bounds. In contrast, all known algorithms require exponential time in the worst case to solve these problems.

Example 1 (The Traveling Salesman Problem). Given a complete directed graph on n cites with real weights x_{ij}, $1 \le i, j \le n$, $i \ne j$, on the edges; i.e., given an input $\vec{x} \in \mathbf{R}^d$, where $d = n(n-1)$ is the number of edges. Find a shortest cycle visiting each city exactly once.

If γ_1 and γ_2 are two cycles visiting each city exactly once, then the equation

$$\sum_{(i,j)\in\gamma_1} x_{ij} = \sum_{(i,j)\in\gamma_2} x_{ij}$$

defines a hyperplane in $\mathbf{R}^e$. The summations are over edges forming γ_1 (resp. γ_2). Let $\mathbf{S}$ be the set of all such hyperplanes. Clearly $M(\mathbf{S}) = 1$. By the corollary, we can find the position of input $\vec{x}$ with respect to the hyperplanes in $\mathbf{S}$ in time $4n^8 \log_2 n + O(n^6)$. If we know the position of $\vec{x}$ with respect to the hyperplanes in S, then we have actually sorted all cycles according to their lengths. Thus in particular we know, in that time, a shortest cycle which solves TSP. Moreover, we also know whether this solution is unique.

Example 2 (Shortest Paths Problems). The method used in Example 1 clearly yields the upper bound $4n^8 \log_2 n + O(n^6)$ to many other variations of shortest paths problems, e.g., *find a shortest simple path from vertex v to vertex v′ in a complete directed graph on n vertices with real weights on edges* (both negative and positive edges are permitted). Here, a path is simple if it visits each vertex at most once. The recognition version of this particular variation is NP-complete (see [3, p. 213] or [1]). However, the same problem with "simple path" replaced by just "path" admits a polynomial algorithm.

Example 3 (The Knapsack Problem). Given real numbers $(x_1, \ldots, x_n) \in \mathbf{R}^n$, decide if there exists a nonempty subset $I \subseteq \{1, \ldots, n\}$ such that $\sum_{i\in I} x_i = 1$.

The set $\mathbf{S}$ consists of $2^n - 1$ hyperplanes $\sum_{i\in I} x_i = 1$. Hence $M(\mathbf{S}) = 1$ and we get upper bound $2n^4 \log_2 n + O(n^3)$, which extends a similar result in [5] for the nonnegative knapsack problem.

Example 4 (Integer Linear Programming). Let n, m, M be integers. The problem is to maximize $\sum_{j=1}^{n} a_j x_j$ subject to

$$(1) \qquad \sum_{j=1}^{n} a_{ij} x_j \leq a_{i(n+1)} \quad \text{for} \quad 1 \leq i \leq m,$$

$$(2) \qquad \vec{x} = (x_1, \ldots, x_n) \in \mathbf{Z}^n \quad \text{and} \quad |x_j| \leq 2^M \quad \text{for} \quad 1 \leq j \leq n.$$

The input is a real vector $\vec{a} \in \mathbf{R}^d$, formed by a_j's and a_{ij}'s, where $d = n + m(n+1)$. The solution is either an integer vector $\vec{x} \in \{-2^M, \ldots, +2^M\}^n$ or "no" (if the constraints in (1) and (2) are inconsistent). What is the partitioning of the space of inputs $\mathbf{R}^d$? If $\vec{x} \in \{-2^M, \ldots, +2^M\}^n$, then the polyhedron $P(\vec{x})$ in $\mathbf{R}^d$ defined by the halfspaces

$$\sum_{j=1}^{d} x_j a_j \geq \sum_{j=1}^{d} y_j a_j \quad \text{for any} \quad \vec{y} \in \{-2^M, \ldots, +2^M\}^n,$$

$$\sum_{j=1}^{d} x_j a_{ij} \leq a_{i(n+1)} \quad \text{for} \quad 1 \leq i \leq m$$

is the region of inputs for which $\vec{x}$ is a solution. The polyhedrons $P(\vec{x})$, $\vec{x} \in \{-2^M, \ldots, +2^M\}^n$, will not in general cover the input space $\mathbf{R}^d$. Hence the solution is "no" outside $\bigcup P(\vec{x})$. Let $\mathbf{S}$ be the set of boundary hyperplanes of all polyhedrons $P(\vec{x})$. Then $M(\mathbf{S}) = 2^{M+1}$, because the x_j's and y_j's are coefficients of equations of hyperplanes in $\mathbf{S}$. If we know the position of an input $\vec{a} \in \mathbf{R}^d$ with respect to hyperplanes in $\mathbf{S}$, then we know which $P(\vec{x})$ contains $\vec{a}$ or whether $\vec{a}$ is outside $\bigcup P(\vec{x})$, and thus we know the solution of the problem. The corollary gives (after obvious simplifications) the upper bound $O(n^4 m^4 M) + O(n^4 m^4 \log_2 nm)$.

5. Additional Remarks

If $H_1, \ldots, H_k$ are hyperplanes in $\mathbf{R}^d$, not necessarily with integer coefficients, then the *linear decision problem* (LDP) is to decide the position of an input $\vec{x} \in \mathbf{R}^d$ with respect to all hyperplanes $H_1, \ldots, H_k$. An important open problem in the theory of linear decision trees is to determine the complexity of LDP. Dobkin and Lipton [2] obtained the upper bound $(2^d - 1)\lfloor \log_2 k \rfloor + d$. The essentially best known lower bound is a straightforward information theory lower bound $\log_2\left(\sum_{n=0}^{d} \binom{k}{n}\right) \approx d \log k$, which follows immediately from the fact that k hyperplanes may divide $\mathbf{R}^d$ into up to $\sum_{n=0}^{d} \binom{k}{n}$ nonempty regions. Clearly the bounds are far apart in d. A lower bound method presented by Yao and Rivest in [10] for the polyhedron membership problem can be adapted for the

LDP, but it does not give any significant improvement over $d \log k$. It would be interesting to prove an upper bound polynomial in d and $\log k$, if it exists.

The linear decision trees and comparison trees are useful tools for studying the computational complexity of various concrete problems. However, our results and the results of Meyer auf der Heide [5], [6] show that many NP-complete problems can be solved in polynomial time by linear decision trees, whereas it is widely conjectured that they require exponential time on Turing machines.

We note that the polynomial-time linear decision trees constructed here do not provide us with polynomial algorithms for the concrete problems considered. We use TSP as an example to illustrate this fact. We have actually constructed one tree for each input size of TSP (where the input size is the number of cities). The trees for various numbers of cities may vary widely. In contrast, an algorithm for TSP should work for any number of cities.

The power of the linear decision tree model stems, at least partially, from the following property. At a given node of the tree, any fact that is implied by the tests on the path from the root to the given node is assumed to be known. In this context, there is a difficulty related to transforming our linear decision tree into an algorithm. The algorithm would have to find which hyperplanes do not intersect the smaller cube and decide the position of the input with respect to these hyperplanes. In the linear decision tree this information is known, since binary searches tests imply that the input is in the smaller cube and this in turn implies the position of input with respect to any hyperplane missing the smaller cube. Another difficulty arises when the algorithm should determine and work with a common point Q of the hyperplanes intersecting the smaller cube. In contrast, the knowledge of Q is automatic in the linear decision tree model.

Acknowledgments

The author wishes to express appreciation to Victor Klee, Faith Fich, and the referee for careful reading and comments on the manuscript.

References

1. G. B. Dantzig, W. O. Blattner, and M. R. Rao, All shortest routes from a fixed origin in a graph, *Theory of Graphs* (Proc. Internat. Sympos., Rome, July 1966), 85–90, Gordon and Breach, New York, 1967.
2. D. Dobkin and R. J. Lipton, On some generalizations of binary search, preprint.
3. M. R. Garey and D. S. Johnson, *Computers and Intractibility: A Guide to the Theory of NP-completness*, Freeman, San Francisco, 1979.
4. B. Grünbaum, *Convex Polytopes*, Interscience, New York, 1967.
5. F. Meyer auf der Heide, A polynomial linear search algorithm for the n-dimensional knapsack problem, *J. Assoc. Comput. Mach.* **31** (1984), 668–676.
6. F. Meyer auf der Heide, Fast algorithms for the N-dimensional restrictions of hard problems, *Proceedings of the 17th STOC*, 413–420, 1985.
7. M. O. Rabin, Proving simultaneous positivity of linear forms, *J. Comput. System Sci.* **6** (1972), 639–650.

8. O. Schreier and E. Sperner, *Modern Algebra and Matrix Theory*, translated by M. David and M. Hauser, Chelsea, New York, 1951.
9. O. Schreier and E. Sperner, *Projective Geometry in n Dimensions*, translated by C. A. Rogers, Chelsea, New York, 1961.
10. A. C. Yao and R. L. Rivest, On the polyhedral decision problem, *SIAM J. Comput.* **9** (1980), 343–347.

Received May 6, 1985, *and in revised form February* 24, 1986.