Theoretical Computer Science 11 (1980) 1–18

THE COMPLEXITY OF LINEAR PROGRAMMING*

David P. DOBKIN
Department of Computer Science, University of Arizona, Tucson, AZ 85721, U.S.A.

Steven P. REISS
Program in Computer Science, Brown University, Providence, RI 02912, U.S.A.

Communicated by M. Nivat
Received September 1978
Revised April 1979

Abstract. The complexity of linear programming and other problems in the geometry of d-dimensions is studied. A notion of LP-completeness is introduced, and a set of problems is shown to be (polynomially) equivalent to linear programming. Many of these problems involve computation of subsets of convex hulls of polytopes, and require $O(n \log n)$ operations for $d = 2$. Known results are surveyed in order to give an interesting characterization for the complexity of linear programming and a transformation is given to produce NP-complete versions of LP-complete problems.

1. Introduction

The goal of computational geometry is to determine the complexity of problems of a geometric nature. Many previous efforts in this direction have concentrated on finding fast algorithms for geometric operations in 2 and 3 dimensions and have resulted in large families of algorithms for a variety of such problems [5, 12, 15, 31, 34, 35, 36]. Unfortunately, research into extensions of such algorithms into higher dimensions has generally met with varying degrees of success. Problems that involve pairs of points or points and lines (for example finding the closest pair of points [34], or testing which of a set of lines a point lies on [5]) can still be done efficiently in higher dimensions. Problems that actually work with multidimensional objects, however, seem to require time exponential in the dimension. Moreover, some of the major successes in this area have involved demonstrating nontrivial lower bounds on these multidimensional operations [6].

In this paper we study the complexity of some of these multidimensional problems. We are particularly interested in problems which are known to be easily solvable in the plane but which seem quite difficult in the worst case in higher dimensions. The classic example of such a problem is linear programming. This problem and the others we consider are all characterized by the fact that they each require a partial

* Portions of this research were supported by an NSF Graduate Fellowship, NSF Grant MCS76-11460 and ONR Grant N00014-75-C-0450. Most of this work was done when both authors were at Yale.

computation of the convex hull of a set of n points in d dimensions. While it is known that such a hull in two or three dimensions can be computed in $O(n \log n)$ operations [31], it is also known that a convex hull of n points in d dimensions can have $O(n^{d/2})$ facets and that every point on such a hull can be involved in $O(n^{d/2-1})$ facets. Since the techniques used in the plane require constructing the entire convex hull, they are not practical in higher dimensions.

These problems are interesting, however, because they do not actually require the whole convex hull, but rather only a small part of its structure. For example, one problem involves testing if a single point or a set of points lies on a convex hull. The complexity of such problems is not known. There is no nontrivial lower bound based on the size of the output since only a single bit of information and not the entire convex hull is produced. Moreover, while these problems have each been widely studied, no known polynomial algorithms exist.

We use two major tools in our study of the complexity of these problems. The first is the notion of polynomial reducibility as considered by Cook and Karp [18]. In particular, we show that a variety of naturally arising multidimensional geometric problems are polynomial equivalent to the problem of linear programming. For many of these problems it is known that a solution could be found using linear programming techniques. Our results strengthen this relationship by showing that the problems are actually (polynomial) equivalent to linear programming. This is surprising in the light of recent work in computational geometry where some of these problems are conjectured to be of polynomial complexity [35].

The second tool of our study is a combination of intuitions into the complexity of various problems. This includes the common intuition that linear programming is a hard (non-polynomial) problem and the belief that NP and co-NP are not equal. These, combined with a result of Ladner and Karp [26] and the notion of polynomial reducibility allows us to infer that of the following three possibilities for the complexity of linear programming, only the third is likely:

(1) linear programming is NP-complete and NP = co-NP,
(2) linear programming is solvable in polynomial time,
(3) linear programming is not in P and is not NP-complete.

Such a result is quite interesting since it suggests that there is a naturally arising class of problems that are neither polynomial solvable nor NP-complete.

2. Definitions

To begin, we set forth the notions of reducibility which are used throughout this paper. Problems under consideration are phrased either as language recognition problems where we are interested in determining if a given input is a member of the set of acceptable inputs, or as actual problems where we want to produce an answer. In all cases the length of the desired answer is short enough to make lower bound arguments based on output length meaningless. Our basic notions of reducibility and

equivalence are those of Karp (1–1 reducibility) [18] or Ladner (many–one reducibility) [26] and are defined for recognition problems as

Definition 1. Problem A is said to be (polynomial) *reducible* to problem B, denoted $A \propto B$, if and only if there exists a function f, computable in deterministic polynomial time, such that $x \in A$ if and only if $f(x) \in B$.

Definition 2. Problems A and B are said to be (polynomial) *equivalent*, denoted $A \equiv B$, if and only if both $A \propto B$ and $B \propto A$.

In the case of problems where an answer is required, we extend these definitions to allow A to be reducible to B if and only if an algorithm for solving B yeilds an algorithm for solving A after polynomial transformation.

We let P denote the class of problems that are solvable in polynomial time on a deterministic multitape Turing machine and let NP denote the class of problems that can be solved in polynomial time on a nondeterministic multitape Turing machine. A problem is called NP-complete if and only if it is in NP and every problem in NP is reducible to it. A problem is called P-hard if and only if it is in NP, not in P, and is not NP-complete. Finally a problem is said to be LP-complete if and only if it is polynomial equivalent to the problem of Linear Programming (LP).

As most of the problems we consider here have a geometric flavor, we introduce the necessary geometric concepts before proceeding. $\mathbf{E}^d$ denotes d-dimensional Euclidean space and a point P in $\mathbf{E}^d$ is represented by a d-vector $(p_1, \ldots, p_d)$. The function $(,): \mathbf{E}^d \times \mathbf{E}^d \to \mathbf{E}$ is the usual dot product, i.e., $(P, Q) = \sum_{i=1}^{d} p_i q_i$. An affine sum of a set of points $p_1, \ldots, p_n$ is any weighted sum $\sum_{i=1}^{n} x_i P_i$ such that $\sum_{i=1}^{n} x_i = 1$. A convex sum of a set of points is an affine sum such that each $x_i \geq 0$. An m-flat in $\mathbf{E}^d$, $m < d$, is an m-dimensional surface. A 0-flat is a point; a 1-flat is a line; a 2-flat is a plane. A $(d-1)$-flat is called a hyperplane and can be written as $\{x \in \mathbf{E}^d \mid (a, x) = b\}$ for some d-vector a and some scalar b. A (closed) halfspace is the set of points on or on one side of a hyperplane. It can be written as $\{X \in \mathbf{E}^d \mid (a, X) \geq b\}$ or as $\{X \in \mathbf{E}^d \mid (a, X) \leq b\}$. The corresponding hyperplane, $\{X \in \mathbf{E}^d \mid (a, X) = b\}$, is called the determining hyperplane of the halfspace. A region in $\mathbf{E}^d$ is called convex if and only if for every pair of points in the region, the line segment connecting them lies completely inside the region. In particular, halfspaces and all flats are convex regions, as is the intersection of any number of convex regions. Given a set of points, their convex hull is the smallest convex set containing these points. Convex regions in $\mathbf{E}^d$ determined by the intersection of a finite number of halfspaces are called polyhedra. A bounded polyhedron is called a polytope. A hyperplane is a supporting hyperplane of a polyhedron if and only if it has a non-empty intersection with the polyhedron and the polyhedron lies totally in one of the two halfspaces determined by the hyperplane. The intersection of a supporting hyperplane with the polyhedron is called a face of the polyhedron. An m-face, $m < d$, is a face that has dimension m, that is, the subspace that can be written as an affine sum of points from the face has dimension m.

A 0-face is called a vertex and a $(d-1)$-face is called a facet. For further details the reader is referred to [13, 14].

3. Forms of linear programming

The most prominent example of the class of problems we want to consider is that of linear programming. This problem has been the subject of a vast body of literature which has dealt with various linear programming problems, solutions and applications. In this section we consider how this literature directly relates linear programming to the multidimensional geometric problems we are interested in.

The literature on linear programming includes several (polynomial) equivalent forms of the problem. In this section we define those that are required for our reducibilities. There are numerous proofs of the equivalence of these problem statements and we shall not reproduce them here. The basic problem of linear programming is:

Linear programming
Given: An integer $n \times d$ matrix A, integer n-vector b, integer d-vector c.
Find: A rational d-vector x such that $Ax \leq b$ and $c^t x$ is maximized.

Further assumptions can be added to this statement of the problem. For example, one can assume that elements of A, b and c are rational or that b is positive or that $c^t x$ is bounded [4]. We summarize these assumptions into two modified versions of linear programming that are the focus of our attention in what follows. These statements are:

Linear inequalities
Given: An integer $n \times d$ matrix A, integer n-vector b.
Determine: If there is a rational d-vector x such that $Ax \leq b$.

Relevancy
Given: A set of constraints $(a_0, x) \leq b_0, \ldots, (a_n, x) \leq b_n$.
Determine: If satisfying the last n constraints is equivalent to satisfying the entire set.

In Section 4 we show that each of these problems is polynomial equivalent to linear programming as is the following problem which is the complement of linear programming.

Linear programming complement
Given: An integer $n \times d$ matrix A, integer n-vector b.
Show: That the system $Ax \leq b$ has no rational solution.

It is this result more than any other that shows the difference between problems involving linear programming and NP-complete problems such as integer programming. Since we can show that linear programming and its complement are polynomially equivalent problems, we know that they belong to the same complexity

classes. A long standing conjecture in theoretical computer science, on the other hand, suggests that the same is not true of NP-complete problems. This fact has been noted by Ladner and Karp [26].

It has long been known that linear programming problems can be viewed as problems in the geometry of d-dimensional Euclidean space, $\mathbf{E}^d$. This fact has been the basis of much of the study of linear programming and especially the study of the complexity of linear programming. In the basic linear programming problem, the solution vector x can be thought of as a point in $\mathbf{E}^d$. Each of the constraints, $(a, x) \leq b$, restricts the set of feasible solutions (possible points x that satisfy all the constraints) to a halfspace in $\mathbf{E}^d$. As the solution must satisfy all the constraints simultaneously, the set of feasible solutions is the intersection of the various halfspaces determined by the constraints. This intersection is easily seen to be a convex polyhedron in $\mathbf{E}^d$. Moreover, it has been shown [4] that a solution to a linear programming problem corresponds to a vertex of this polyhedron. Thus, linear programming can be reformulated as the geometric problem:

Geometric linear programming
Given: A set of halfspaces $\{H_1, \ldots, H_n\}$ and a d-vector x.
Find: The vertex v of the polyhedron formed by the intersection of the halfspaces a which (v, x) is maximized.

In geometry there is a well-defined concept of a geometric dual. The dual is formed by a dimension-inverting mapping from $\mathbf{E}^d$ to $\mathbf{E}^d$ that takes objects of dimension into objects of dimension $d-n-1$. In particular, points are mapped into hyperplanes and hyperplanes are mapped into points. There are several methods of constructing such a mapping, and the most common is that of the use of polar sets. Such a mapping can be defined to take an object Q into an object $\bar{Q}$ such that

$$\bar{Q} = \{x \in \mathbf{E}^d \mid (u, x) \leq 1 \text{ for all } u \in Q\}.$$

It is a well-known goemetric result [13] that this mapping has several nice properties

Lemma 1. *Let P be a polytope in $\mathbf{E}^d$ and let $\bar{P}$ be its polar dual. Then*
(1) *$0 \in P$ if and only if $\bar{P}$ is bounded, where 0 is the origin;*
(2) *There is a $1-1$ onto mapping between the k-faces of P and the $d-k-1$ faces of $\bar{P}$*
(3) *$\bar{\bar{P}}$, the dual of $\bar{P}$, is P;*
(4) *if the supporting hyperplane to a facet of P is $\{x \in \mathbf{E}^d \mid (u, x) = 1\}$, then th corresponding point in $\bar{P}$ is u;*
(5) *$\bar{P}$ is the convex hull of U, where $U = \{u \in \mathbf{E}^d \mid u$ corresponds to a facet of $P\}$.*

For any geometric problem it is generally possible to consider the dual problem instead since we can map the original or primal problem into the dual, solve the dual problem, and then apply the dual mapping, which by (3) above is its own inverse, to construct the primal solution. In particular, the dual to the geometric version of the linear programming problem is:

Dual geometric linear programming
Given: A set of points $x_1, \ldots, x_n$ in $\mathbf{E}^d$ such that the origin is interior to their convex hull, and a ray r from the origin.
Find: The facet of the convex hull through which r passes.

In addition to considering the geometric forms of the linear programming problem, we consider the geometric versions of the problem of linear inequalities which is

Intersection of halfspaces
Given: Closed halfspaces $H_1, \ldots, H_n$.
Determine: If $H_1 \cap \cdots \cap H_n$ is non-empty.

This problem and the others we have noted are the forms of linear programming we use in establishing our reducibilities. In the next section of this paper we introduce other problems that are polynomially equivalent to one of these forms. These new problems are important in their own right and have their own applications and associated methods of solution. For many of these it has been noted that linear programming can be used to find a solution. What is surprising though is that these problems which seem quite a bit simpler than linear programming are actually polynomial equivalent to it.

4. Geometric problems

So far we have seen several forms of linear programming that are polynomial equivalent and hence LP-complete. We have also seen several geometric problems which are either restatements of some form of linear programming or are the geometric duals of such problems. In this section we consider several other geometric problems and show that they too are LP-complete. These problems are different than those presented above, and do not correspond to a direct restatement of a linear programming problem. From a geometric point of view, these problems have been the subject of study independent from any connection to linear programming problems. This geometric study has lead to conjectures suggesting that polynomial time and possibly even subquadratic algorithms exist for such problems. It is this connection of problems from different research areas which we view as one of the major contributions of this paper.

The first class of such problems involves the identification of extreme points. We say that a point Q is extreme with respect to points $P_1, \ldots, P_n$ if and only if Q is exterior to the convex hull of $P_1, \ldots, P_n$. The basic problem here is:

Extreme point (EP)
Given: A set of points $P_0, P_1, \ldots, P_n$ in $\mathbf{E}^d$.
Determine: if P_0 is extreme with respect to $P_1, \ldots, P_n$.

We can consider a simplification of this problem where we place all the points $P_1, \ldots, P_n$ on the unit sphere in $\mathbf{E}^d$ and let P_0 be the origin. This yields:

Origin point interior
Given: A set of points $P_1, \ldots, P_n$ on S^{d-1}, the unit sphere in $\mathbf{E}^d$.
Determine: If the origin is extreme with respect to $P_1, \ldots, P_n$.

This version of the problem can also be restated as:

Hemisphere problem
Given: A set of points $P_1, \ldots, P_n$ on S^{d-1}.
Determine: If $P_1, \ldots, P_n$ lie interior to some hemisphere.

A problem that is a generalization of the extreme point problem is that c determining the depth of a set of points. The depth is defined as the numer of neste convex hulls necessary to include all the points. In other words, the points on th original convex hull are at depth one and if these are removed, then the points on th new convex hull are at depth two. The highest depth that is attained in this manner called the depth of the set. The problem here is:

Depth of a set
Given: A set of points $\{P_1, \ldots, P_n\}$ in $\mathbf{E}^d$.
Find: The depth of the set.

The geometric dual problem to testing if a point is extreme is:

Hyperplane-halfspace intersection
Given: A set of halfspaces $H_1, \ldots, H_n$ and a hyperplane h.
Determine: If h intersects $\bigcap_{i=1}^{n} H_i$.

It is interesting to note that this is the geometric form of the problem of relevanc If we let the hyperplane h in this problem be the polar dual of the origin, then th problem becomes:

Boundedness
Given: A set of halfspaces $\{H_1, \ldots, H_n\}$.
Determine: If their intersection is bounded.

All of these extreme point problems can be solved in linear time in the plane an have typically been solved using linear programming techniques in higher dimen sions. In Section 6 we furthermore show that they are all actually LP-complet Moreover, Johnson and Preparata [16] have shown that a modification of th problem is NP-complete. In particular, if we give as input the n points on S^{d-1} and a integer $k < n$ and seek to determine whether k (or more) of the n points share common hemisphere, then the NP-complete problem MAXSAT2 [11] can b reduced to it. Furthermore, by applying their reduction to the hemisphere problen we are able to make contact with the work of Jones and Laaser [17] and show that linear programming is solvable in poly-log space, then all problems in P are solvabl in this space bound [7].

A second class of geometry problems that can be shown to be LP-complete involves the notion of separability. Two point sets are said to be separable if and only if there is a hyperplane such that all points of one set lie on one side of the hyperplane and all points of the other set lie on the other side (i.e., the hyperplane separates the points). This problem has been the subject of several studies. Classical results in geometry show that two point sets in $\mathbf{E}^d$ are separable if and only if every subset of $d+2$ points is separable. This yields an algorithm of complexity $O(n^{d+2})$ for sets of n points in $\mathbf{E}^d$. More recent studies in computational geometry [32, 35] have shown that this problem can be solved in the time $O(n \log n)$ in two or three dimensions. While this is a vast improvement over the classical n^4 or n^5, applying the same techniques in higher dimensions means forming the convex hull and hence yields an algorithm that is still of exponential complexity in $\mathbf{E}^d$. Finally, problems of separability have been studied extensively for classifying data points for pattern recognition purposes [8]. Here two basic solution methods are employed—gradient approximation techniques and linear programming. While experimental results here show that the approximation techniques are more efficient, it is easy to see that they also have worst case complexities that are at least exponential and sometimes infinite.

We consider several separability problems. The basic ones are:

Point–set separation
Given: Points $P_0, P_1, \ldots, P_n$ in $\mathbf{E}^d$;
Determine: If P_0 is separable from $\{P_1, \ldots, P_n\}$,

and

Set–set separation
Given: Points $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ in $\mathbf{E}^d$.
Determine: If $\{P_1, \ldots, P_n\}$ is separable from $\{Q_1, \ldots, Q_m\}$.

We can also consider simpler versions of these basic problems. First of all, the points can be restricted to the unit sphere. This yields:

Spherical separation
Given: Points $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ on S^{d-1} in $\mathbf{E}^d$.
Determine: If $\{P_1, \ldots, P_n\}$ is separable from $\{Q_1, \ldots, Q_m\}$.

In addition, we consider the problem of testing if a set of points and its reflection through the origin are separable:

Hemisphere separation
Given: Points $P_1, \ldots, P_n$ on S^{d-1} in $\mathbf{E}^d$.
Determine: If $\{P_1, \ldots, P_n\}$ is separable from $\{-P_1, \ldots, -P_n\}$.

All of these separability questions have been stated as recognition problems. We can restate each of them as computational problems where we are required to actually find a separating hyperplane if one exists. This yields the problems:

Finding point–set separation
Given: Points $P_0, P_1, \ldots, P_n$ in $\mathbf{E}^d$.
Find: A hyperplane separating $\{P_0\}$ from $\{P_1, \ldots, P_n\}$.

Finding set–set separation
Given: Points $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ in $\mathbf{E}^d$.
Find: A hyperplane separating $\{P_1, \ldots, P_n\}$ from $\{Q_1, \ldots, Q_m\}$.

Finding spherical separation
Given: Points $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ on S^{d-1} in $\mathbf{E}^d$.
Find: A hyperplane separating $\{P_1, \ldots, P_n\}$ from $\{Q_1, \ldots, Q_m\}$.

Finding hemisphere separation
Given: Points $P_1, \ldots, P_n$ on S^{d-1} in $\mathbf{E}^d$.
Find: A hyperplane separating $\{P_1, \ldots, P_n\}$ from $\{-P_1, \ldots, -P_n\}$.

5. Other problems

The techniques of linear programming are used to solve a wide variety of mathematical problems [4]. These include network flow problems, transportation problems, assignment problems, game theory problems, and, in a limited way, traveling salesman problems. The first three of the classes of problems are known to be equivalent [4] and can all be solved using the polynomial algorithm of Edmonds and Karp [9]. Unfortunately, linear programming does not seem to be reducible to these special cases. The traveling salesman problem, on the other hand, is known to be NP-complete [18] and hence can be used to solve linear programming problems. However, in this case linear programming yields an approximate rather than an actual solution, and hence the traveling salesman problem seems more complex.

The most common application of linear programming to game theory, the problem of determining the optimal mixed strategies in a two person game, does involve an LP-complete problem. A two person game is defined by a payoff matrix $A = (a_{i,j})$ where each $a_{i,j}$ represents the cost or benefit of the game if the first player chooses strategy i and the second chooses strategy j. It is known that each player can maximize his earnings by using a mixed strategy where he plays the various individual strategies at random using a fixed set of probabilities. Determining what these probabilities are for the two players is the problem:

Two person game theory
Given: A payoff matrix $A = (a_{i,j})$
Determine: The optimal mixed strategies of the two players.

That this problem is actually equivalent to linear programming (and hence LP-complete) is a classic result that follows from von Neumann's minimax theorem [4].

6. Main result

We are now ready to present our main result:

Theorem 1. *The following prolbems are* LP*-complete*:

(1) *Linear programming,*
(2) *Linear inequalities,*
(3) *Relevancy,*
(4) *Linear programming complement,*
(5) *Geometric linear programming,*
(6) *Dual geometric linear programming,*
(7) *Intersection of halfspaces,*
(8) *Extreme point,*
(9) *Origin interior problem,*
(10) *Hemisphere problem,*
(11) *Depth of a set,*
(12) *Hyperplane-halfspace intersection,*
(13) *Boundedness,*
(14) *Point–set separation,*
(15) *Set–set separation,*
(16) *Spherical separation,*
(17) *Hemisphere separation,*
(18) *Finding point–set separation,*
(19) *Finding set–set separation,*
(20) *Finding spherical separation,*
(21) *Finding hemisphere separation,*
(22) *Two person game theory.*

Proof. Preliminary to proving the main theorem, we prove 18 basic reductions:

(1) Linear programming is LP-complete. This is trivially true based on the definition of LP-complete.

(2) Linear inequalities $\equiv$ LP. Clearly linear inequalities $\propto$ LP. To show $\propto$, one uses a binary search technique over the rational numbers. Since a bound exists on the denominator of the result of a linear programming problem, such a search can be done [30, 33].

(3) Linear programming complement $\equiv$ LP. First note that LI $\propto$ LPC. To solve a LPC problem using linear programming, one merely needs to add artificial variables which are all constrained to be greater than or equal to zero, and solve the linear programming problem of minimizing the sum of these new variables. The details and the proof that this is conclusive is fundamental to the classical study of linear programming as it represents the first phase of the Simplex algorithm [4]. This has also been observed by Karp.

(4) (a) Geometric linear programming $\equiv$ LP,
(b) Intersection of halfspaces $\equiv$ LI,
(c) Hyperplane-halfspace intersection $\equiv$ Relevancy.

These follow immediately since the one problem is the geometric interpretation of the other.

(5) LI $\equiv$ EP. Classical transformations (see e.g. [4]) allow us to assume that the LI problem is bounded and all solutions are positive. The geometric dual of this new problem is then equivalent to the extreme points problem.

(6) (a) Extreme point $\equiv$ Hyperplane-halfspace intersection,
(b) Origin interior problem $\equiv$ Boundedness.

This reduction is based on the geometric duality concept and follow from the properties of the polar dual.

(7) Origin interior problem $\equiv$ Extreme point. $\propto$ follows since an origin interior problem is a special case of an extreme point problem. We need to show that any extreme point problem can be solved by solving a problem where P_0 is the origin and all the other points lie on the unit sphere. We first can do a translation to insure that P_0 is the origin. Now P_0 is interior to $P_1, \ldots, P_n$ if and only if the origin is interior to the translated points $P'_1, \ldots, P'_n$ if and only if $0 = \sum_{i=1}^n x_i P'_i$ where each $x_i \geq 0$ and $\sum_{i=1}^n x_i = 1$. Let $Q_i = P'/|P'_i|$, be a point on the unit sphere corresponding to P'_i. Now let $r = \sum_{i=1}^n x_i |P'_i|$ and for each i, let $y_i = x_i |P'_i|$ and let $z_i = y_i / r$. Then $x_i \geq 0$ if and only if $y_i \geq 0$ if and only if $z_i \geq 0$ as $r > 0$. Also

$$0 = \sum_{i=1}^n x_i P'_i = \sum_{i=1}^n \frac{y_i}{P'_i} P'_i = \sum_{i=1}^n y_i Q_i = \frac{1}{r} \sum_{i=1}^n y_i Q_i = \sum_{i=1}^n z_i Q_i$$

while $\sum_{i=1}^n z_i = 1$. Hence P_0 is interior to $P_1, \ldots, P_n$ if and only if the origin is interior to $Q_1, \ldots, Q_n$, and we are done.

(8) Hemisphere problem $\equiv$ Origin interior problem. These are actually the same problem as the origin is an extreme point if and only if there is a supporting hyperplane through the origin. But such a hyperplane exists if and only if the points on the unit sphere share common hemisphere.

(9) Extreme point $\equiv$ Depth of a set. Since a point is extreme if and only if it has depth one, $\propto$ follows immediately. The depth of all points in a set can be determined by finding all extreme points and assigning them depth 1. Then all the extreme points of the original set with all these points eliminated are assigned depth 2. This process is repeated until all points are assigned a depth. Since it can be repeated at most n/d times for n points in d dimensions, at most $O(n^2)$ extreme point problems must be solved to determine the depth of the set.

(10) Point–set separation $\equiv$ Extreme point. This follows from the definitions of extreme points and separation since P_0 is separable from $\{P_1, \ldots, P_n\}$ if and only if P_0 is extreme with respect to $\{P_1, \ldots, P_n\}$.

(11) (a) Spherical separation $\propto$ Set–set separation,
(b) Hemisphere separation $\propto$ Spherical separation,

(c) Finding spherical separation $\propto$ Finding set–set separation,

(d) Finding hemisphere separation $\propto$ Finding spherical separation.

These reductions all follow as in each case one problem is a special case of the other.

(12) Set–set separation $\propto$ Point–set separation. It is sufficient to show how to express a given set–set separation problem as a point–set separation problem. Let the two sets to be separated be $\{P_1, \ldots, P_n\}$ and $\{Q_1, \ldots, Q_m\}$. Let P be the convex hull of $P_1, \ldots, P_n$ and Q be the convex hull of $Q_1, \ldots, Q_m$. Then the sets are separable if and only if $P \cap Q$ is empty. Define

$$P - Q = \{x \mid x = p - q, P \in P \text{ and } q \in Q\}.$$

Then $P \cap Q$ is empty if and only if the origin is exterior to $P - Q$, or alternatively, if the origin is an extreme point of $P - Q$. It can be shown [3, 4] that $P - Q$ is a subset of the convex hull of U where

$$U = \{u \mid u = P_i - Q_j \text{ for } 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}.$$

Hence it is sufficient to separate the origin from the set U, and since $|U| = mn$, we are done

(13) Hemisphere problem $\propto$ Hemisphere separation. Let the points on the unit sphere be $P_1, \ldots, P_n$. Then these lie interior to some hemisphere if and only if there is a rotation such that every first coordinate is greater than zero. Then the hyperplane with first coordinate zero separates these points from their negatives, and we are done.

(14) Hemisphere separation $\propto$ Finding hemisphere separation. This is true since finding the separating hyperplane determines separability.

(15) Finding set–set separation $\propto$ Finding point–set separation. Here we can use the same techniques as in our reduction from set–set separation to point–set separation (reduction 12). We can thus find a hyperplane $E^0 = \{x \in \mathbf{E}^d \mid (a, x) = b_0\}$ that separates the origin from $P - Q$. Now we assume, with loss of generality, that for all points y in $P - Q$, $(a, y) < 0$. Thus, for all p in P and all q in Q we have

$$(a, p) - (a, q) < 0 \quad \text{or} \quad (a, p) < (a, q).$$

Next we compute $r = \max_i(a, P_i)$ and $s = \min_i(a, Q_i)$. Then $r < s$. Finally, let $t = \frac{1}{2}(r + s)$, and let

$$E = \{x \in \mathbf{E}^d \mid (a, x) = t\}.$$

Now we claim that E separates P from Q. For any p in P and q in Q we have $(a, p) \leq r < t < s \leq (a, q)$.

(16) Finding point–set separation $\propto$ Dual geometric linear programming. Let $\{P_1, \ldots, P_n\}$ be the set of points we are separating from P_0. Then $Q = (1/n) \sum_{i=1}^{n} P_i$ is interior to the convex hull of $P_1, \ldots, P_n$. We can transform all the points to place Q at the origin. Let r be the ray from Q through P_0 under the transformation. We can now find the facet F that this ray passes through. Given this facet, its affine hull is its

supporting hyperplane. Let this hyperplane be $E_0 = \{x \mid (a, x) = b_0\}$. Let a paralle hyperplane containing the point P_0 be $E_1 = \{x \mid (a, x) = b_1\}$. Then let $E_2 = \{x \mid (a, x) = \frac{1}{2}(b_0 + b_1)\}$. Then E_2 is a separating hyperplane between P_0 an $\{P_1, \ldots, P_n\}$.

(17) Dual geometric linear programming $\equiv$ LP. Classical transformations (see e.g [4]) allow us to assume that the LP problem is bounded and has only positiv solutions. Taking the geometric dual and observing that these transformations als hold in the geometric domain then allows us to complete the equivalence.

(18) Two person game theory $\equiv$ LP. This is a classic result due to Von Neumann' minimax theorem[4]

We can now prove the theorem using these reductions as follows:

- Linear programming $\equiv$ LP	1
- Linear inequalities $\equiv$ LP	2
- Linear programming complement $\equiv$ LP	3
- Geometric linear programming $\equiv$ LP	4a
- Intersection of halfspaces $\equiv$ LI	4b
- Extreme point $\equiv$ LI	5
- Hyperplane-halfspace intersection $\equiv$ EP	6a
- Origin interior problem $\equiv$ EP	7
- Boundedness $\equiv$ Origin interior problem	6b
- Hemisphere problem $\equiv$ Origin interior problem	8
- Depth of a set $\equiv$ EP	9
- Point–set separation $\equiv$ EP	10
- Relevancy $\equiv$ Hyperplane-halfspace intersection	4c
- LP $\equiv$ Hemisphere problem $\propto$ Hemisphere separation	13, 11b, 11a, 12
$\propto$ Spherical separation $\propto$ Set–set separation	
$\propto$ Point–set separation $\equiv$ LP	
$\propto$ Dual geometric linear programming $\equiv$ LP	17
- LP $\equiv$ Hemisphere separation	14, 11d, 11c, 15, 16
$\propto$ Finding hemisphere separation	
$\propto$ Finding spherical separation	
$\propto$ Find set–set separation	
$\propto$ Finding point–set separation	
$\propto$ Dual geometric linear programming $\equiv$ LP	17
- Two person game theory $\equiv$ LP	18

7. Complexity of linear programming

So far we have demonstrated a class of natural problems related to d-dimensiona geometry that all have the same relative complexity. What this complexity i however, is not clear. While many of the problems have been considered for som time, no polynomial time algorithms are known for solving them. Moreover, thes

problems also do not appear to be NP-complete. These two observations make the existence of this class of LP-complete problems quite interesting from both an applied and a theoretical point of view. In this section we investigate this class of problems by collecting previous results that reflect on their complexity.

One of the outstanding features of these problems is that they are typically solved quite efficiently in the average case. This is principally illustrated in the class of algorithms that first find an approximation to the solution and then methodically proceed to a new approximation until the actual solution is found. The most widely known example of such an algorithm is the Simplex algorithm [4]. This algorithm solves a linear programming problem by finding an initial feasible solution and then, if it does not maximize the objective function, a new feasible solution is found and the check for maximizing the objective function is made again. The value of the objective function with the new solution is always greater than or equal to its value at the previous solution. In geometric terms this algorithm involves finding some vertex of a polytope defined by the intersection of a set of halfspaces, and then, if this vertex does not represent the desired solution, finding a new vertex adjacent to the current one and considering it. This process is repeated, following a path of connected vertices along the polytope until the proper one is reached. Under the proper assumptions, it can be shown that these techniques always find the proper solution. The Simplex algorithm can be used to solve any LP-complete using the proper reducibilities. Moreover, this method of incremental search can be employed directly in most of the problems under consideration.

Empirical evidence for the complexity of the Simplex algorithm shows it to be quite efficient, usually running in time linear with the number of constraints and variables [1, 4, 25]. However, in recent years it has been shown that there are cases where any Simplex-like algorithm requires exponential time [24]. Thus, although LP-complete problems can be solved efficiently in the average case, the best upper bound known for their worst-case complexity remains exponential.

Given this behavior, we know turn to a study of the relationship of LP-complete problems to NP-complete prolbems. To begin with, we observe that all LP-complete problems belong to NP.

Lemma 2. $LP \in NP$.

Proof. Consider the problem of linear inequalities. A solution to this problem consisting of a rational d-vector x can be checked in polynomial time to insure that $Ax \leq b$. Thus this problem is in the class NP. Then, by Theorem 1, all LP-complete problems are in NP, and, in particular, $LP \in NP$.

This lemma puts an upper bound on the complexity of linear programming. It leaves three broad possibilities for the actual complexity. These can be summarized as:

Theorem 2. *One of the following is true:*
(a) $LP \in P$,
(b) LP *is* P*-hard,*
(c) LP *is* NP*-complete and* NP *is closed under complement.*

Proof. If $P = NP$, then all the statements are true. Assume then that $P \neq NP$. Then either LP is or is not doable in polynomial time. If it is, then $LP \in P$. Otherwise, either LP is NP-complete or it is not. If it is not, then $L \in NP - P$ and LP not NP-complete imply that LP is P-hard. If LP is NP-complete then LI is also NP complete and the complement of LI, LPC, is also NP-complete. But then NP is closed under complement, and the theorem is proved. Parts of this theorem have been alluded to in [26].

Of these three possibilities, the last one must be considered unlikely. Although it is currently unknown whether NP is closed under complement or if $P = NP$, it is widely believed that both of these statements are false. Hence, we may conjecture that either statement (a) or statement (b) is true. If statement (a) is true, then polynomial time algorithms exist for all of the problems that are LP-complete. This would be interesting in terms of its impact on the study of geometric complexity as well as being of possible practical interest in operations research. If, however, statement (b) is true, then we would have demonstrated a natural problem belonging to $NP - P$ which is not NP-complete (assuming $P \neq NP$). Such a problem would be of considerable interest to theoreticians. Thus, the resolution of Theorem 2 will be an important and interesting result no matter which of the alternatives is true. Hence the problems associated with determining the complexity of linear programming are important problems of complexity theory.

It seems unlikely that the notions of LP-completeness and NP-completeness are equivalent. Yet, we may extend the observation of Johnson and Preparata [16] to transform each of these problems into an NP-complete problem by merely adding an additional parameter, k. For example, determining if any subset of size k of a set of n points on S^{d-1} share a common hemisphere is NP-complete. We observe that in the case where $k = n$, the problem becomes LP-complete. We can similarly form NP-complete versions of the other LP-complete problems. These transformations are similar to methods introduced in [11]. Typical of these problems would be:

Intersection of common halfspaces
Given: Halfspaces $H_1, \ldots, H_n$ in $\mathbf{E}^d$ such that the origin is interior to their intersection, an integer k.
Determine: If there is a subset consisting of k of the halfspaces which have the origin as their intersection.

Extreme point
Given: A set of points $P_0, P_1, \ldots, P_n$ in $\mathbf{E}^d$, an integer k.
Determine: If there exist $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ such that P_0 is extreme with respect to $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$.

Intersection of halfspaces
Given: Closed halfspaces $H_1, \ldots, H_n$, an integer k.
Determine: If there are $1 \leq i_1 \leq i_2 \leq \cdots \leq i_k \leq n$ such that $\bigcap_{j=1}^{k} H_{i_j}$ is non-empty.

Relevancy
Given: A set of constraints $(a_0, x) \leq b_0, \ldots, (a_n, x) \leq b_n$, an integer k.
Determine: If satisfying some set of k of the last n constraints is equivalent to satisfying these constraints along with the first constraint.

Linear inequalities
Given: An integer $n \times d$ matrix A, integer n-vector b, integer k.
Determine: If there is a rational d-vector x such that k components of $Ax = b$ are negative.

These results expressing linear programming as an interesting limiting case of integer programming take on added interest as a possible means of finding a hierarchy of natural problems in their complexities. Furthermore, as observed in [7], connections with logspace completeness can also be made.

8. Conclusion

The results in this paper can be divided into two parts. In the first part we introduced the class of LP-complete problems. These problems are interesting and important. Some of them are fundamental to the study of d-dimensional geometry and others, especially those directly involved with linear programming, have vast practical applications. Moreover, these problems represent a wide range of geometric problems that all have the same intrinsic complexity. Hence, using our notions of reducibility, information regarding the complexity of any single problem, be it a fast algorithm or a good lower bound, is directly applicable to all the other LP-complete problems as well.

The second part of the paper was devoted to a survey of known results on the complexity of linear programming. Here we have shown that it is probable that linear programming is either of polynomial complexity or is P-hard. In the light of the efforts that have been made toward finding a polynomial algorithm, and considering that these efforts have failed, it seems most probable that linear programming is indeed P-hard. Further connection is made with NP-complete problems by introducing a simple transformation which converts each LP-complete problem into an NP-complete problem.

References

[1] M.L. Balinsky, An algorithm for finding all vertices of convex polyhedral sets, *SIAM J. Appl. Math.* **9** (1961) 72–88.

[2] C.-A. Burdet, Generating all the faces of a polyhedron, *SIAM J. Appl. Math.* **26** (1974) 479–489.
[3] D.R. Chand and S.S. Kapur, An algorithm for convex polytopes, *J. ACM* **17** (1970) 78–86.
[4] G.B. Dantzig, *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ, 1963).
[5] D.P. Dobkin and R.J. Lipton, Multidimensional search problems, *SIAM J. Comput.* **5** (2) (1976) 181–186.
[6] D.P. Dobkin and R.J. Lipton, A lower bound of $\frac{1}{2}n^2$ on linear search tree programs for the knapsack problem, *J. Comput. System Sci.* **16** (3) (1978) 413–417.
[7] D.P. Dobkin, R.J. Lipton and S. Reiss, Linear programming is P-complete, *Information Processing Lett.* **8**(2) (1978) 96–97.
[8] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).
[9] J. Edmonds and R.M. Karp, Theoretical efficiency in algorithmic efficiency for network flow problems, *J. ACM* **19** (1972) 248–264.
[10] D. Gale, On the number of faces of convex polytope, *Canadian J. Math.* **16** (1964) 12–17.
[11] M. Garey, D. Johnson and L. Stockmeyer, Some simplified NP-complete problems, *Theoret. Comput. Sci.* **1** (3) (1976) 237–267.
[12] R.L. Graham, An efficient algorithm for determining the convex hull of a finite planar set, *Information Processing Lett.* **1** (1972) 132–133.
[13] B. Grunbaum, *Convex Polytopes* (Wiley, London, 1967).
[14] B. Grunbaum, Polytopes, graphs, and complexes, *Bull. AMS* **76** (1970) 1131–1201.
[15] R.A. Jarvis, On the identification of the convex hull of a finite set of points in the plane, *Information Processing Lett.* **2** (1973) 18–21.
[16] D. Johnson and F. Preparata, The densest hemisphere problem, *Theoret. Comput. Sci.* **6** (1978) 93–107.
[17] N. Jones and W. Laaser, Complete problems for deterministic polynomial time, *Theoret. Comput. Sci.* **3** (1977) 105–117.
[18] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J. W. Thatcher, Eds. *Complexity of Computer Computations* (Plenum Press, New York, 1972).
[19] V. Klee, A class of linear programming problems requiring a large number of iterations, *Numer. Math.* **7** (1965) 313–321.
[20] V. Klee, Convex polyhedra and mathematical programming, University of Washington Tech. Report #50 (1974).
[21] V. Klee, Convex polytopes and linear programming, in: *Proc. IBM Scientific Computing Symposium on Combinatorial Problems* (IBM Data Processing Division, White Plains, NY, 1966) 123–158.
[22] V. Klee, Heights of convex polytopes, *Math. Anal. Appl.* **11** (1965) 176–190.
[23] V. Klee, On the number of vertices of a convex polytope, *Canadian J. Math.* **16** (1964) 701–720.
[24] V. Klee and G.J. Minty, How good is the simplex algorithm, in: O. Shisda, Ed., *Inequalities*, Vol. III (Academic Press, New York, 1972) 159–175.
[25] H.W. Kuhn and A.W. Tucker, Linear inequalities and related systems, *Ann. Math. Studies* **38** (Princeton University Press, Princeton, NJ, 1956).
[26] R.E. Ladner, On the structure of polynomial time reducibility, *J. ACM* **22** (1975) 155–171.
[27] T.M. Liebling, On the number of iterations of the simplex method, Institut fur Operations Research der ETHZ.
[28] T.H. Mattheis, An algorithm for determining irrelevant constraints and all vertices in systems of linear inequalities, *Operations Research* **21** (1973) 247–260.
[29] W.B. McRae and E.R. Davidson, An algorithm for the extreme rays of a pointed convex polyhedral cone, *SIAM J. Comput.* **2** (1973) 281–293.
[30] C. Papadimitriou, Efficient search for rationals, *Information Processing Lett.* **8** (1) (1978) 1–4.
[31] F.P. Preparata and S.J. Hong, Convex hulls of finite sets of points in two and three dimensions *Comm. ACM* **20** (2) (1977) 87–93.
[32] F.P. Preparata and D.E. Muller, Finding the intersection of n half-spaces in time $O(n \log n)$ Univ. of Illinois Report (1977).
[33] S. Reiss, Rational search, *Information Processing Letters* **8** (2) (1978) 87–90.
[34] M.I. Shamos and D. Hoey, Closest-point problems, in: *Proc. 16th Annual Symposium on Foundations of Computer Science* (1975) 151–162.

[35] M.I. Shamos, Computational geometry, Ph.D. Thesis, Yale University, New Haven, CT (1978).
[36] M.I. Shamos, Geometric complexity, in: *Proc. Seventh Annual ACM Symposium on Theory of Computing* (1975) 224–233.
[37] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions* (Springer-Verlag, New York, 1970).
[38] R., J.-B. Wets and C. Witzgall, Algorithms for frames and linearity spaces of cones, *J. Res. Nat. Bur. Standards Sect. B* **71** (1967) 1–7.
[39] N. Zadeh, A bad network problem for the simplex method and other minimum cost flow algorithms, *Math. Programming* **5** (1973) 255–266.

Note added in proof

A recent result of Khinchin shows that LP is in P. This resolves the discussion of Section 7 and raises new open problems concerning the application of this algorithm to other LP-complete problems.

Theoretical Computer Science 11 (1980) 19–37

INVERTIBLE TERMS IN THE LAMBDA CALCULUS

Jan BERGSTRA
Mathematical Institute, University of Leiden, Leiden, The Netherlands

Jan Willem KLOP
Mathematical Institute, University of Utrecht, Utrecht, The Netherlands

Communicated by C. Böhm
Received June 1977
Revised May 1978

1. Introduction

It is well-known that the set of λ-terms modulo $\beta\eta$-convertibility is a semi-group with I as identity element and composition $\circ$, defined by $M \circ N = BMN$, where $B \equiv \lambda xyz.x(yz)$. In [6, pp. 167, 168] the question is raised under what conditions an element in this semi-group has an inverse.

Dezani-Ciancaglini gave in [8] a characterization of (w.r.t. $\lambda\beta\eta$-calculus) invertible terms having a normal form as the 'finite hereditary permutators', and she conjectures that these are all the $\beta\eta$-invertible terms, i.e. a term without normal form cannot have an inverse.

In this paper we confirm her conjecture. Two proofs are given for this fact, of which the first is more direct. The second proof uses the in itself interesting fact that certain 'λ-trees' can be represented as Böhm-trees of λI-terms (in fact we prove something more), plus Hyland's characterization of the equality in the Graph model $P\omega$ (see [9, 1]).

The result on representation of λ-trees is further used to characterize the λ-terms invertible in D_∞, Scott's well-known lattice model (see [12]).

Since for this last result a slightly more general form of the main lemma in [8] is needed, we have included a new proof of that lemma.

2. Preliminaries

In this section we collect the ingredients necessary for the sequel, without the proofs which can be found in the literature. The basic definitions and facts about the λ-calculus are supposed to be known.