Cowles Foundation Paper 522

Econometrica, Vol. 49, No. 2 (March, 1981)

PRODUCTION SETS WITH INDIVISIBILITIES PART II. THE CASE OF TWO ACTIVITIES[1]

BY HERBERT E. SCARF

Part I of this paper introduces a general framework for the discussion of discrete production sets and the associated programming problems which arise when a particular endowment of factors is specified. In this part of the paper we shall apply these ideas to integer programming problems with two activities and bring to bear some of the basic considerations of the theory of computational complexity. The numbering of sections, figures, and equations will follow those used in Part I.

7. THE COMPLEXITY OF DISCRETE PROGRAMMING PROBLEMS

THE ASSUMPTIONS of constant returns to scale in production, implicit in neoclassical analysis, is inadequate to account for one of the major facts of economic life: the existence of large industrial enterprises which form the basic units of economic activity. To the extent that a purely technical explanation can be offered for this phenomenon it is to be found in the property of increasing returns to scale which is implied by the presence of indivisibilities and other forms of nonconvexities in production.

The systematic study of maximization problems involving indivisibilities is now over two decades old. At its inception there was considerable optimism that this area of research would be as fruitful for economic analysis as linear programming had previously been. But this optimism has faded. For in the intervening period, virtually no general theoretical results have been produced nor have universally applicable algorithms been developed which are capable of even the most tentative economic interpretation. The bulk of research on this topic has moved far from economics and become concerned instead with intricate and detailed problems in combinatorial analysis and graph theory.

The concentration on combinatorics does not represent an arbitrary disengagement from real issues, but is based on a growing appreciation of the enormous complexity of discrete programming problems. The existence of combinatorial problems which seem to possess no efficient algorithms finds its counterpart in that body of economic literature which replaces the concept of a profit maximizing firm by that of an organization concerned with heuristic and tentative searches for suboptimal solutions. In this literature there is a persistent suggestion that the problems of resource allocation are extremely complex and that the firm is the economic entity organized to solve them.

I think that this point of view has much to recommend it. It is my own belief that an understanding of the complexity of economic problems will become an

[1] The research described in this paper and its predecessor was supported by a grant from the National Science Foundation. Some of the material was presented in the Fisher-Schultz lecture delivered at the 1978 European Meeting of the Econometric Society.

I am extremely grateful to Professors Sergiu Hart of Stanford University, Roger Howe of Yale University, and Michael Todd of Cornell University for the many interesting conversations I have had with them on the general topic of this paper.

important ingredient of any future theory of the firm, even though the specific lines of development of such a theory are far from clear at present.

During the last decade there has been a very active area of research in computer science and operations research which has enhanced our ability to describe and investigate the intrinsic difficulty of mathematical programming problems. The basic question asked in this theory of computational complexity (see Garey and Johnson [3] for an exposition of the theory) is whether a particular combinatorial or programming problem has a solution algorithm whose time of execution is in some way commensurate with the time required to describe the problem.

To be more specific let us consider the general integer programming problem:

$$\begin{aligned} &\max \sum_j a_{0j} h_j, \\ &\sum_j a_{ij} h_j \geqslant b_1, \\ &\qquad \vdots \\ &\sum_j a_{mj} h_j \geqslant b_m, \end{aligned}$$

with h_j integral. The problem is described by the $(m+1)\times n$ matrix A and the vector b. Let us imagine that the entries of A and b are all integral and are stored in a computer in terms of their binary expansion. Each of these $m+(m+1)\cdot n$ entries will require a certain number of binary bits to be represented. The total number of bits required to represent all of these numbers will be an integer, say I, which measures the size of the problem.

Let us consider an algorithm which solves a given set of integer programs in a finite number of basic steps, each of which consists of a single addition, subtraction, multiplication, division, or elementary logical operation. The algorithm is said to be *polynomial* if there is a polynomial function $f(I)$ of the size of the problem so that the algorithm terminates in no more than $f(I)$ steps, for all instances of the class of integer programs being studied whose size is no more than I.

An important example which illustrates the distinction between polynomial and nonpolynomial algorithms is the Euclidean algorithm for finding the greatest common divisor (a, b) of two positive integers a and b. Assume that $a > b$ and define the sequence $r_1, r_2, \ldots, r_{k+1}$ by

$$\begin{aligned} &a = t_1 b + r_1, \quad 0 < r_1 < b; \\ &b = t_2 r_1 + r_2, \quad 0 < r_2 < r_1; \\ &r_1 = t_3 r_2 + r_3, \quad 0 < r_3 < r_2; \\ &\qquad \vdots \\ &r_k = t_{k+2} r_{k+1}; \end{aligned}$$

with $t_1, \ldots, t_{k+2}$, and $r_1, \ldots, r_{k+1}$ integral. It is a trivial matter to verify that r_{k+1} is, in fact, the greatest common divisor of a and b.

To illustrate the process consider the two integers 141 and 15. We have

$$141 = 9 \cdot 15 + 6,$$

$$15 = 2 \cdot 6 + 3,$$

$$6 = 2 \cdot 3,$$

so that $(141, 15) = 3$.

The number of binary bits required to represent a positive integer a is

$$f(a) = [\log_2 (a+1)],$$

where $[x]$ is the smallest integer $\geq x$. The function $f(a)$ can easily be seen to satisfy $f(2a) = f(a) + 1$ for any positive argument.

The size of the problem is given by

$$S = f(a) + f(b),$$

the number of bits required to store the two integers a and b. From

$$r_{j-1} = t_{j+1} r_j + r_{j+1}, \quad \text{and}$$

$$r_{j-1} > r_j > r_{j+1},$$

we have $t_{j+1} \geq 1$ and, therefore, $r_{j-1} \geq r_j + r_{j+1} > 2r_{j+1}$. It follows that

$$f(r_{j-1}) + f(r_j) \geq f(r_j) + f(r_{j+1}) + 1,$$

and this measure of size decreases by at least one unit on every iteration of the Euclidean algorithm. The algorithm must therefore terminate in no more than S steps and is certainly polynomial.

There is an obvious variant of the Euclidean algorithm which is not polynomial in the data of the problem. Assuming that $a > b$, the Euclidean algorithm replaces these two integers by the pair

$$b, \quad r = a - tb, \quad \text{with} \quad 0 \leq r < b,$$

and continues. What might be called the slow Euclidean algorithm avoids division, replaces the pair by $b, a - b$ and continues. The slow algorithm is obviously not polynomial since it takes a steps to verify that the greatest common divisor $(a, 1) = 1$, even though the size of the problem is $[\log_2 (a+1)]$.

One way of formulating the central question of complexity theory is whether the general integer program has a polynomial algorithm. Integer programming has the property that if it were to have such an algorithm then a very large number of other combinatorial problems would have polynomial algorithms as well. And in particular virtually all problems of resource allocation involving indivisibilities could be solved with extreme rapidity.

Part I of this paper does describe a general algorithm for the solution of integer programs based on Sperner's Lemma. For this approach to be implemented it is necessary to be able to describe the full collection of primitive sets, or equivalently the neighborhood system, associated with a given technology. In this part of

the paper we shall provide a full characterization of the neighborhood systems for the general integer program with two activities, and an algorithm for determining the particular neighborhood system for a specific technology. The determination will be seen to be polynomial in the data of the problem.

In its immediate application Sperner's Lemma suffers from a difficulty similar to that experienced when it is used in fixed point problems. It proceeds through a large number of small steps from an initial position to the final answer. In the language of complexity theory, Sperner's Lemma will not provide a polynomial algorithm for the two variable integer program. But in much the same way as the slow Euclidean algorithm relates to the ordinary Euclidean algorithm, these small steps can be organized and telescoped so that the resulting algorithm is in fact polynomial in the data.

The existence of a polynomial algorithm for the two variable problem was first demonstrated for the particular example known as the knapsack problem by Hirschberg and Wong [1] and for a more general class of problems by Kannan [2]. In contrast to the arguments of the present paper, neither of these approaches seems to me to suggest a generalization when the number of variables is greater than two. The use of primitive sets and neighborhood systems does, however, point to a possible argument for the conjecture that there is a polynomial algorithm for integer programming when the number of variables is fixed in advance.

8. SOME REMARKS ON POLYGONS IN THE PLANE

In this section we shall review some elementary material on triangles and quadrilaterals in the plane.

Let us begin by considering a triangle in the plane (see Figure 19), whose three vertices h^0, h^1, h^2 are lattice points, and which contains no other lattice points in its interior or on the boundary. The area of such a triangle is given by $1/2$ of the absolute value of either of the following determinants:

$$\det \begin{vmatrix} 1 & 1 & 1 \\ h_1^0 & h_1^1 & h_1^2 \\ h_2^0 & h_2^1 & h_2^2 \end{vmatrix} \quad \text{or}$$

$$\det \begin{vmatrix} h_1^1 - h_1^0 & h_1^2 - h_1^0 \\ h_2^1 - h_2^0 & h_2^2 - h_2^0 \end{vmatrix}.$$

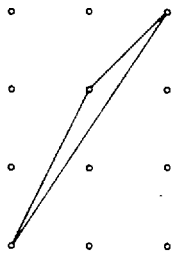


FIGURE 19

We have the following classical theorem.

THEOREM 8.1: *A necessary and sufficient condition that a triangle whose vertices are lattice points contain no other lattice points is that its area equal* $1/2$.

Consider a triangle whose vertices are lattice points and such that

$$\det \begin{vmatrix} 1 & 1 & 1 \\ h_1^0 & h_1^1 & h_1^2 \\ h_2^0 & h_2^1 & h_2^2 \end{vmatrix} = \pm 1.$$

If a lattice point (h_1, h_2) is a convex combination of these three vertices, then

$$\begin{pmatrix} 1 \\ h_1 \\ h_2 \end{pmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ h_1^0 & h_1^1 & h_2^1 \\ h_2^0 & h_2^1 & h_2^2 \end{bmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix},$$

with $\alpha_i \geq 0$. But since the determinant is ± 1, the α_i's are integral. They sum to 1 and therefore h must be one of the three vertices.

In order to argue the converse let us consider a triangle whose vertices are lattice points and whose area is larger than $1/2$. We shall construct a fourth lattice point contained in the triangle.

Our argument will involve linear transformations $h' = Uh + k$ of the plane into itself, where k is an integral vector and U a unimodular matrix, i.e., a matrix with integral entries and determinant of ± 1. Such a transformation carries the lattice points in the plane onto themselves and preserves area.

By applying a translation we may assume that $h^0 = (0, 0)$. But then h_1^1 and h_2^1 must be relatively prime, since if they had a common factor there would be a lattice point on the line connecting h^0 and h^1. It follows that there are integers p and q such that

$$ph_1^1 - qh_2^1 = 1.$$

The linear transformation $h' = Uh$ where

$$U = \begin{bmatrix} p & -q \\ -h_2^1 & h_1^1 \end{bmatrix}$$

is unimodular, so that it is sufficient to consider the triangle (Figure 20) whose three vertices are Uh^0, Uh^1, Uh^2 or

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

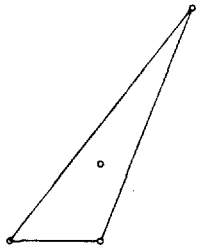


FIGURE 20

There is no loss in generality in assuming that $y > 0$, since this can be brought about by a unimodular reflection. We may also assume that $0 < x \leq y$ since this can be achieved by adding a suitable integral multiple of the second coordinate to the first coordinate of each of these vectors.

The area of this triangle is $y/2$; we therefore assume that $y \geq 2$. But the lattice point $(1, 1)'$ is contained in such a triangle since

$$\begin{pmatrix}1\\1\end{pmatrix} = \begin{bmatrix}0 & 1 & x\\ 0 & 0 & y\end{bmatrix}\begin{pmatrix}(x-1)/y\\ (y-x)/y\\ 1/y\end{pmatrix}.$$

This demonstrates Theorem 8.1.

In the course of this argument we have also verified that if the triangle (h^0, h^1, h^2) contains no other lattice point, then there is a unimodular transformation (including a translation) which brings this triangle to the form

$$\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}1\\1\end{pmatrix}.$$

If the second coordinate is subtracted from the first—again a unimodular transformation—we see that such a triangle can be brought into the canonical form

$$\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}.$$

Lemma 3.4 tells us that a convex polyhedron in the plane whose vertices are lattice points and which contains no other lattice points has either 3 or 4 vertices. We have, in the above argument, characterized such polyhedra with 3 vertices. Now let us consider one with 4 vertices h^0, h^1, h^2, h^3.

Since the triangle with vertices h^0, h^1, h^2 contains no other lattice points, there is a unimodular transformation which brings all four points to the form

$$\begin{pmatrix}0\\0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\begin{pmatrix}h_1\\h_2\end{pmatrix}.$$

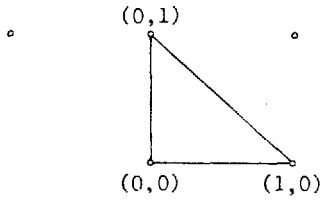


FIGURE 21

But the triangle formed by

$$\binom{0}{0}\binom{1}{0}\binom{h_1}{h_2}$$

contains no additional lattice points; therefore

$$\det \begin{vmatrix} 1 & h_1 \\ 0 & h_2 \end{vmatrix} = h_2 = \pm 1.$$

In a similar fashion $h_1 = \pm 1$, so that there are at most four possible locations for the point h, as illustrated in Figure 21. Of course, we cannot have $(h_1, h_2) = (-1, -1)$ since $(0, 0)$ would not be a vertex of such a quadrilateral. The three possible quadrilaterals appear in Figure 22.

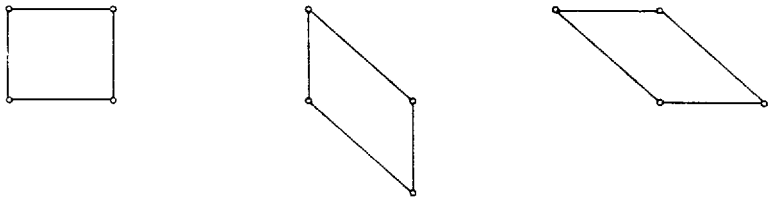


FIGURE 22

The first of these is the unit square, and the second, consisting of

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

can be brought to

$$\begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

by adding the first row to the second. The third figure can be transformed by a similar unimodular transformation into the unit square.

THEOREM 8.2: *A convex polyhedron whose four vertices are lattice points, and which contains no other lattice points is equivalent to the unit square under a unimodular transformation.*

Aside from translations the general example of such a quadrilateral is a regular parallelogram of unit area whose four vertices are given by

$$\binom{0}{0}\binom{p}{q}\binom{x-p}{y-q}\binom{x}{y},$$

with x, y, p, q integers satisfying $py - qx = \pm 1$. (See Figure 23.)

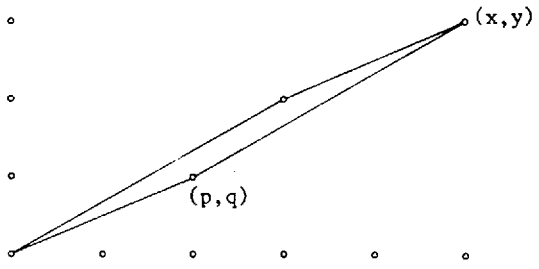


FIGURE 23

9. THE COLLECTION OF PRIMITIVE SETS FOR AN ACTIVITY ANALYSIS MODEL WITH TWO ACTIVITIES

Let the set X consist of all vectors $x = Ah$, with A an $(m+1)\times 2$ matrix of integers and $h = (h_1, h_2)$ ranging over the lattice points in the plane. Ties in the vectors x are assumed to be broken by the lexicographic rule of Section 1. Assuming that X contains some efficient vectors the primitive sets associated with this technology will contain either three or four non-slack vectors.

Each such primitive set will be described by a triangle in the plane or a parallelogram of the sort discussed in the previous section. According to Theorems 1.7 and 1.8, each lattice point in the plane will be contained in a finite number of primitive sets. This implies that, aside from translations, a finite number of triangles and parallelograms will appear as primitive sets. As we shall see this collections of polygons will have a definite structure.

Let us begin by examining the triangles which represent primitive sets in conjunction with a particular set of $m-2$ slack vectors. For notational convenience we shall assume that the non-slack vectors correspond to the coordinates 0, 1, 2. Let x^0, x^1, x^2 with $x^i = Ah^i$ be the non-slack vectors in a particular primitive set arranged so that the row minima of

$$\begin{bmatrix} \underline{x_0^0} & x_0^1 & x_0^2 \\ x_1^0 & \underline{x_1^1} & x_1^2 \\ x_2^0 & x_2^1 & \underline{x_2^2} \end{bmatrix}$$

lie on the main diagonal. There is no loss in generality in assuming that $h^0 = (1, 1)$, $h^1 = (1, 0)$, $h^2 = (0, 1)$ since this can be achieved by a unimodular transformation. Figure 24 displays this primitive set. The line through the vector h^i is given by

$$a_{i1}h_1 + a_{i2}h_2 = a_{i1}h_1^i + a_{i2}h_2^i.$$

In order to determine the replacement for h^0 we press in the line through h^0 until a second vector in the primitive set is reached; in this case h^1. We then relax the line through h^1 and reach the replacement $\hat{h} = (0, 0)$.

The replacement may be written as $\hat{h} = h^1 + h^2 - h^0$. In this form the replacement operation is invariant under unimodular transformations. This permits us to

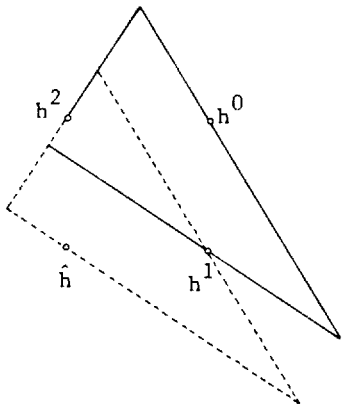


FIGURE 24

conclude that, in general, the replacement for a given vertex in a triangle is equal to the sum of the other two vertices minus the vertex itself. When all possible replacements are carried out we see that we never leave the collection of triangles described in Figure 25. Since any two primitive sets with the same slack vectors can be obtained, one from the other, by a sequence of replacement operations, we have therefore discovered the full collection of primitive sets.

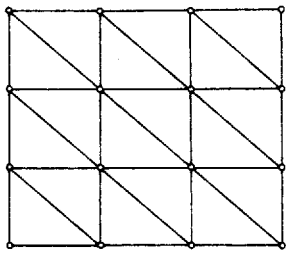


FIGURE 25

THEOREM 9.1: *The collection of triangles which form primitive sets in conjunction with a particular set of* $m-2$ *slack vectors form a simplicial subdivision of the plane which can be brought to Figure* 25 *by a unimodular transformation.*

Now let us turn to the determination of the primitive sets which are associated with a particular collection of $m-3$ slack vectors. Again we assume for convenience that the non-slack vectors correspond to the coordinates 0, 1, 2, 3. Some of these primitive sets will be described in the plane, by parallelograms of unit area, and others by triangles in conjunction with one of the first four slack vectors.

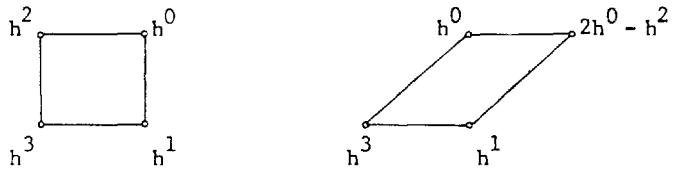


FIGURE 26

Consider a parallelogram of unit area whose vertices h^0, h^1, h^2, h^3 are lattice points. To be specific we assume that $h^3 = (0, 0)$, that h^1 and h^2 are adjacent to h^3, and that h^0 is opposite to h^3. The new parallelogram obtained by replacing any one of these vertices by its reflection through an adjacent vertex will also be of unit area. Figure 26 represents a pair of parallelograms the second of which is obtained from the first by replacing h^2 by its reflection through h^0. A different, and noncongruent, parallelogram would be obtained if we replace h^1 by its reflection through h^0.

We shall be concerned with a sequence of parallelograms $P_1, \ldots, P_k$ with the properties given in the following definition.

DEFINITION 9.2: A finite sequence of parallelograms, each of which has integral vertices and unit area will be called a *chain* of parallelograms if (i) the origin is a vertex of each P_j, and (ii) P_{j+1} is obtained from P_j by replacing one of the vertices adjacent to the origin by its reflection through the vertex opposite to the origin.

Without loss of generality the first parallelogram in a chain can be taken to be the unit square since an arbitrary such parallelogram can be brought into this form by a unimodular transformation. At each stage in the sequence of parallelograms a choice is made as to which of the two vertices adjacent to the origin is to be replaced in order to determine the next parallelogram in the chain. One of these two vertices will be *above* the diagonal through the origin; we shall use the letter a to represent this choice, and the letter b to represent the choice of the vertex *below* the diagonal. Given the first parallelogram the chain is determined by a sequence of a's and b's. Figure 27 illustrates the particular chain of parallelograms associated with the sequence aab.

Figure 27 also includes two pairs of triangles, associated with the first and last parallelograms in the chain. The first pair is obtained by drawing the diagonal through the two vertices adjacent to the origin in the unit square. The second pair is obtained by drawing the diagonal through the origin in the final parallelogram.

The concept of a chain of parallelograms permits us to characterize the primitive sets which are associated with a specific set of $m-3$ slack vectors.

THEOREM 9.3: *The quadrilateral primitive sets associated with a specific set of $m-3$ slack vectors are represented, up to translations and unimodular transformations, by a chain of parallelograms. The primitive sets which are triangles, in*

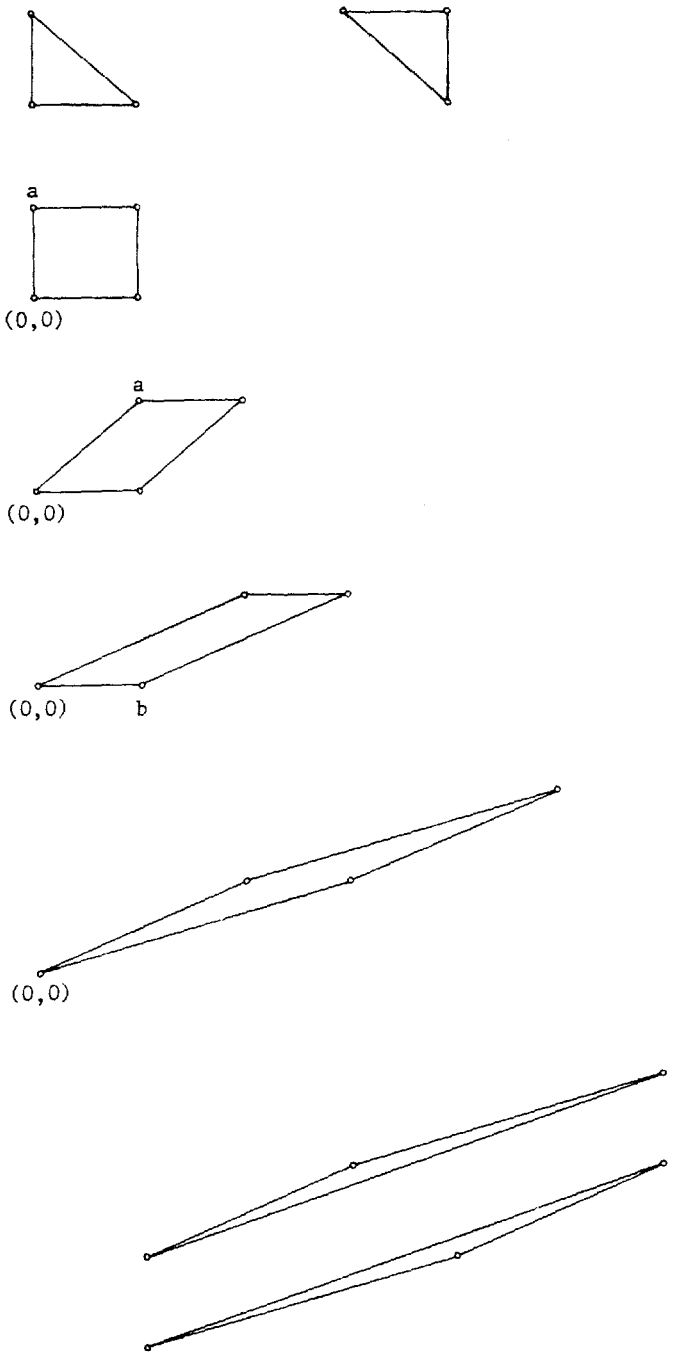


FIGURE 27—The chain *aab*.

conjunction with an additional slack vector, are given by the pairs of triangles at either end of the chain.

In order to demonstrate Theorem 9.3 we assume for definiteness that 0, 1, 2, 3 are the coordinates corresponding to the non-slack vectors. Let x^0, x^1, x^2, x^3 with $x^i = Ah^i$ be a primitive set arranged so that the row minima of $[x_j^i]$ lie on the main diagonal. There is no loss in generality in assuming that h^i are the vertices of the unit square drawn as in Figure 28. The line through h^i is given by $a_{i1}h_1 + a_{i2}h_2 = a_{i1}h_1^i + a_{i2}h_2^i$.

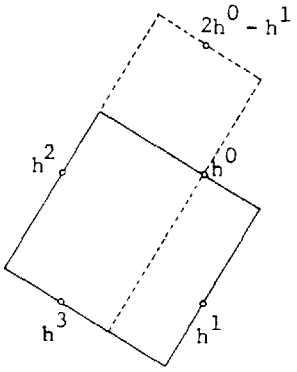


FIGURE 28

LEMMA 9.4: *The replacements for h^1 and h^2 are either the same slack vector, or they result in a pair of congruent parallelograms which are a successor to the unit square in a chain of parallelograms.*

In order to determine the replacement for h^1 we press in the line through h^1 until we reach a second vector in the primitive set—in Figure 28 this is h^0. We then relax the line through h^0. If the point $2h^0 - h^1$ does not satisfy the inequality through h^2 then this line may be relaxed indefinitely and the replacement for h^1 is the slack vector ξ^0. In this case it will also be true that the replacement for h^2 is the same slack vector ξ^0. If, on the other hand, the point $2h^0 - h^1$ does satisfy the inequality through h^2 then it is the replacement for h^1 and the new parallelogram is the successor to the unit square obtained by using the letter *b*. Moreover, the replacement for h^2 can be seen to be $2h^3 - h^2$, resulting in a congruent parallelogram. And finally, if h^3 is reached before h^0 in pressing in the line through h^1, the new parallelogram is the successor obtained by using the letter *a*.

In the same fashion the replacements for h^0 and h^3 will either lead to triangular primitive sets with the same additional slack vector or to a parallelogram which preceeds the unit square in a chain of parallelograms. Repeated applications of the replacement operation will result in a chain of parallelograms with a pair of

triangular primitive sets at either end. In order to complete the proof of Theorem 9.3 we must simply argue that there are no primitive sets other than those produced by this construction. But this is clear since the replacement operations (other than eliminating a slack vector ξ^i with $i > 3$) keep us within this class of primitive sets.

One final observation is appropriate before completing this section. The chain of parallelograms has a structure which will be very useful in accelerating the small steps of Sperner's Lemma to obtain a polynomial algorithm for integer programs with two variables. The chain will begin, say, with the unit square and be described by a sequence of letters *a* or *b*. The parallelograms have a particularly simple form if the same letter persists for a number of steps. Figure 29, for example, describes the initial segment of a chain given by the sequence *aaa*. For such a segment the sequence of new vertices are consecutive lattice points on a line.

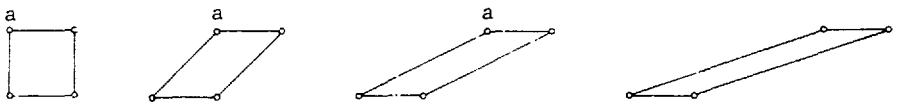


FIGURE 29

DEFINITION 9.5: By a *link* in the chain of parallelograms we mean a maximal set of consecutive parallelograms in the chain, each of which is obtained from its predecessor by using the same letter *a* or *b*.

The chain can therefore be decomposed into links, and as we shall see each such link can be transversed very rapidly in the application of Sperner's Lemma.

10. THE APPLICATION OF SPERNER'S LEMMA

In this section we shall discuss the application of Sperner's Lemma to the solution of the general integer program in two variables

$$\begin{aligned} \max\ & a_{01}h_1 + a_{02}h_2\,, \\ & a_{11}h_1 + a_{12}h_2 \geq b_1\,, \\ & \quad\vdots \\ & a_{m1}h_1 + a_{m2}h_2 \geq b_m\,, \end{aligned}$$

with $h = (h_1, h_2)$ integral. One possible approach is to define X to consist of all vectors $x = Ah$ and to apply that form of Sperner's Lemma discussed in Section 5. If the specific structure of the collection of primitive sets could be determined easily, we would be able to carry out the required replacements leading to a completely labeled simplex.

There is no difficulty in proceeding in this fashion and in showing that the sequence of steps may be accelerated so as to produce an algorithm which is polynomial in the data of the problem. In the interest of simplicity, however, I shall take a slightly different approach. Theorem 3.3 states that there is a subproblem obtained by selecting three of the m inequalities which has the same solution as the original problem, assuming that the original problem is feasible and has a finite maximum. Since there are $\binom{m}{3}$ such subproblems at most, an argument that the problem with two variables and three inequalities has a polynomial algorithm will yield a corresponding conclusion for the general problem as well.

For this reason we shall restrict our attention to matrices

$$A = \begin{bmatrix} a_{01} & a_{02} \\ a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix},$$

whose corresponding set X contains some efficient vectors. The subsequent section will provide an algorithm to determine the chain of parallelograms associated with this technology. At present we shall assume that the chain is known to us in the form given by Theorem 9.3.

The typical primitive set in the chain is given by four vectors x^0, x^1, x^2, x^3 with $x^i = Ah^i$ and organized so that the row minima of $[x^i_j]$ lie on the main diagonal. This particular convention gives specific names to the four vertices of the parallelogram, names which change as the sequence of replacement operations is carried out. In Figure 28 the vector h^0, associated with row 0 of the matrix, is opposite to h^3, associated with row 3. In the same way h^1 and h^2 are opposite to each other. The careful reader of the proof of Lemma 9.4 will realize that this same relationship persists for all parallelograms in the chain. This observation gives meaning to the following notational convention.

CONVENTION 10.1: We assume that for all parallelograms in the chain the vertex associated with row 0 of A is opposite to that associated with row 3, and similarly for rows 1 and 2.

In applying Sperner's Lemma the vectors in X are given an integer label l corresponding to a violated inequality, and the label 0 if all of the inequalities are satisfied. When the notational Convention 10.1 holds we adopt the more specific Rule 5.3 that x is labeled with the subscript of the *last* violated inequality. This innocuous assumption is important in demonstrating the major result of the present section.

The algorithm for finding a completely labeled primitive set moves through a sequence of adjacent primitive sets each one of which bears the labels 1, 2, 3 with one of the labels appearing on two vertices. The arguments of Section 6 based on orientation theory permit us to say which of the vertices with doubled labels is to be removed in the path towards the solution.

The primitive sets which are encountered will be triangles with associated slack vectors or parallelograms. The transition from a triangle to a parallelogram must be at the first or last parallelogram of the chain. It is entirely conceivable, however, that we enter and depart from the chain of parallelograms any number of times. The following theorem, which is primarily responsible for the simplicity of the algorithm, tells us that this possibility will, in fact, never occur. It says that if we ever enter the chain of parallelograms, we proceed through it—never visiting two congruent parallelograms—until a completely labeled primitive set is found, or we exit at the other end of the chain and never return to a parallelogram.

THEOREM 10.2: *The algorithm never encounters a parallelogram congruent to one previously visited.*

We shall demonstrate this theorem by showing that the algorithm always takes us from an almost completely labeled parallelogram to its successor in the chain of parallelograms under our notational convention and the particular labeling rule.

Let the unit square represent a typical parallelogram in the chain, as in Figure 30. We shall argue that the labeling rule requires us to remove either h^1 or h^2.

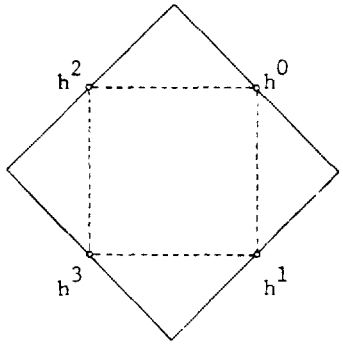


FIGURE 30

The labels associated with the four vertices depend on the particular placement of the inequalities $a_{i1}h_1 + a_{i2}h_2 = b_i$. The reader may easily verify that Figure 31 captures all possible dispositions of these labels.

Let us consider each case in turn.

I. In this case we eliminate either h^3 or h^2. According to 6.5, h^2 will be eliminated because the permutation $(l(h^0), l(h^1), 0, l(h^3))$ or $(2, 1, 0, 3)$ is odd.

II. In this case h^1 will be eliminated since $(l(h^0), 0, l(h^2), l(h^3)) = (1, 0, 2, 3)$ is odd.

III. Here h^2 will be eliminated since $(l(h^0), l(h^1), 0, l(h^3)) = (2, 1, 0, 3)$.

IV. $(l(h^0), 0, l(h^2), l(h^3)) = (1, 0, 2, 3)$, again an odd permutation.

This demonstrates Theorem 10.2. The removal of h^1 or h^2 will lead to congruent parallelograms, but the particular placement of these parallelograms,

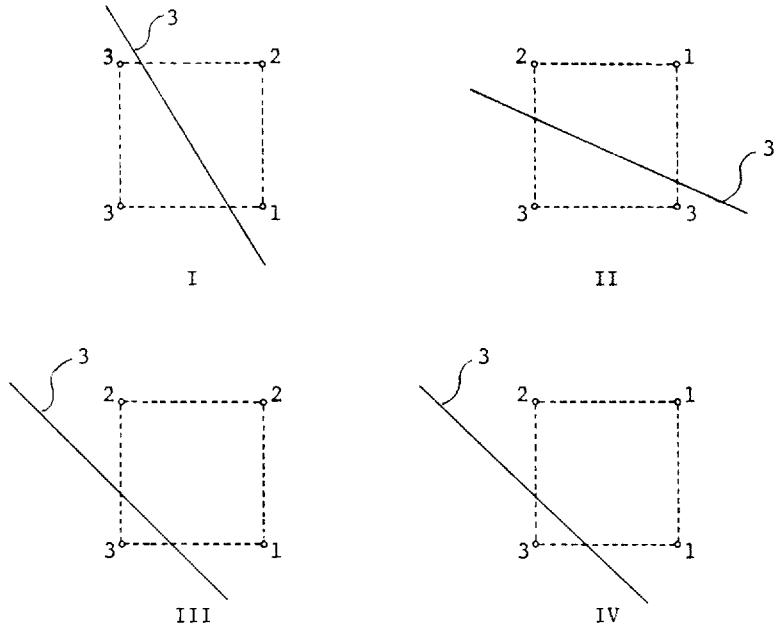


FIGURE 31

and the subsequent sequence of labels, will depend on the particular choice of h^1 or h^2 to be removed. This would seem to require a calculation for each parallelogram in the chain, and the algorithm might then require at least as many steps as there are distinct parallelograms. We shall see, in a subsequent section, that the number of parallelograms is not polynomial in the data of the problem. The number of links will, however, be polynomial in the data, and this raises the question of whether the passage of the algorithm through all of the parallelograms in a given link can be accomplished rapidly.

In order to see how this can be done, let us assume that the unit square of Figure 30 is the initial parallelogram in the link illustrated in Figure 32. Aside from translations the parallelograms in the link are given by

$$\begin{bmatrix} 0 & 1 & j-1 & j \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

for $j = 1, 2, \ldots, t$. Let us consider Case I in detail in order to see the particular sequence of parallelograms which appear as we progress through the link.

Case I: In this case h^2 is initially removed and replaced by (2, 1) which cannot bear the label 3. If $l(2, 1) = 2$ we remove (1, 1) and continue along the line $(j, 1)$. If $l(j, 1) = 2$ for $j = 1, 2, \ldots, t-1$, then we immediately move to the final primitive

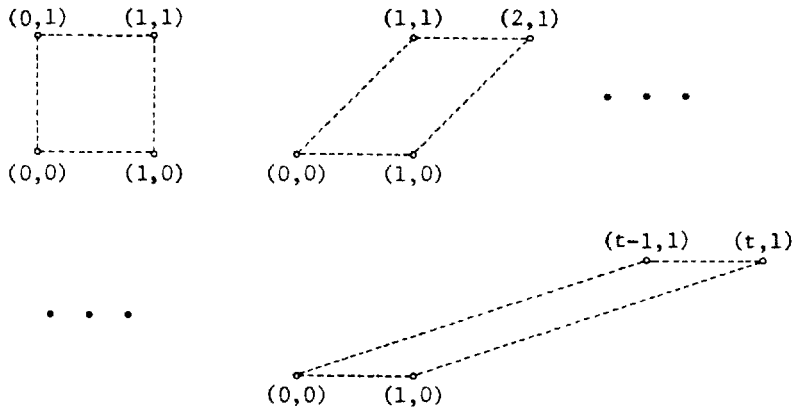


FIGURE 32

set in the link, located at

$$\begin{bmatrix} 0 & 1 & t-1 & t \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

On the other hand there may be some point $(j^*, 1)$ where the label changes from 2 to either 0 or 1. The first such point may, of course, be found by a simple computation. If $l(j^*, 1) = 0$, then the primitive set

$$\begin{bmatrix} 0 & 1 & j^*-1 & j^* \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

bears all four labels and $(j^*, 1)$ is the global solution to the integer programming problem.

The label for $(j^*, 1)$ may be 1, as in Figure 33. At this point the vector $(1, 0)$ is removed and replaced by $(-1, 0)$ which bears the label 3, as do all points on the line to the left of the origin. But, without any additional computation, this locates the final primitive set in the link as

$$\begin{bmatrix} j^*-t & j^*-t+1 & j^*-1 & j^* \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

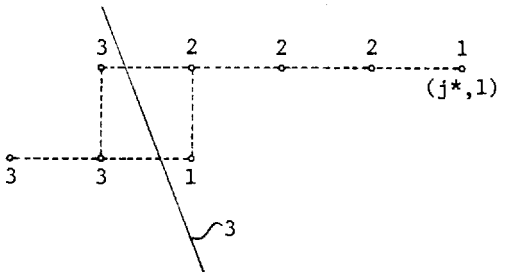


FIGURE 33

The remaining cases may be examined in a similar fashion. In two of them no computation at all is required to determine the precise location of the final primitive set in the link. In the remaining case, a single computation of the first lattice point on a line for which the label changes is adequate. We have the following theorem.

THEOREM 10.3: *The passage of the algorithm through the entire chain of parallelograms can be accomplished by a number of computations equal to the number of links in the chain.*

Two items are missing from a complete description of the algorithm. First of all we must determine the precise location of the almost completely labeled parallelogram at the beginning of the chain. Theorem 10.3 would then tell us that the passage through the chain could be accomplished rapidly. But the completely labeled primitive set may not be a parallelogram; it may, in fact, be a triangle associated with one of the two simplicial subdivisions at either end of the chain, in conjunction with a particular slack vector. A second missing item is, therefore, a description of the algorithm which solves the two inequality problems obtained by discarding the inequality corresponding to that particular slack vector. We shall concentrate on the latter problem first.

Let the initial parallelogram in the chain be the unit square as in Figure 30, and assume for definiteness that h^1 is reached first when the line through h^0 is pressed in. This implies that the triangles of Figure 34 are primitive sets in conjunction with the first slack vector ξ^1.

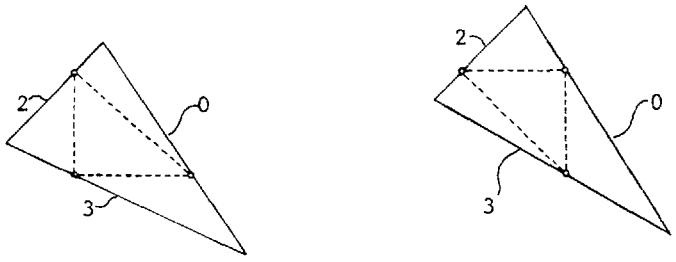


FIGURE 34

We solve the problem

$$\max a_{01}h_1 + a_{02}h_2 ,$$
$$a_{21}h_1 + a_{22}h_2 \geq b_2 ,$$
$$a_{31}h_1 + a_{32}h_2 \geq b_3 ,$$

and (h_1, h_2) integral, by first solving the equations $\Sigma_{j=1}^{2} a_{ij}\hat{h}_j = b_i$, $i = 2, 3$. The solution $(\hat{h}_1, \hat{h}_2)$ is typically fractional and we let h_2^* be the greatest integer less than or equal to h_2. We have the following conclusion.

THEOREM 10.4: *Let the structure of primitive sets be as in Figure* 34. *The optimal solution to the above integer program will be on one of the two lines* $h_2 = h_2^*$ *or* $h_2 = h_2^* + 1$.

In order to demonstrate 10.4, we let (h_1^*, h_2^*) be the rightmost lattice point on the line $h_2 = h_2^*$ which violates inequality number 3. The label associated with (h_1^*, h_2^*) and any point on the line to its left is 3. The label associated with $(h_1^* + 1, h_2^*)$ is 0, since if the second inequality were violated at this point, the intersection of the two lines would have its second coordinate less than h_2^*. We have therefore found a completely labeled primitive set for the problem with two inequalities, if $l(h_1^*, h_2^* + 1) = 2$.

If, on the other hand, $l(h_1^*, h_2^* + 1) = 0$, we replace $(h_1^* + 1, h_2^*)$ by $(h_1^* - 1, h_2^* + 1)$ and continue moving to the left until we first find a point on the line $h_2 = h_2^* + 1$ whose label is different from 0. By construction this point will violate inequality number 2. If, as in Figure 35, the point satisfies the third inequality, we have found

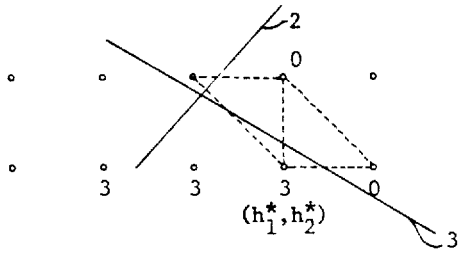


FIGURE 35

a completely labeled primitive set. On the other hand, as in Figure 36 the point may violate the third inequality as well, and one additional replacement is required to find the completely labeled primitive set. In either case a single computation on each of the two lines is sufficient to solve the problem.

The solution will, of course, satisfy inequalities 2 and 3. If it happens to satisfy inequality number 1 as well, it is the optimal solution to the problem with three

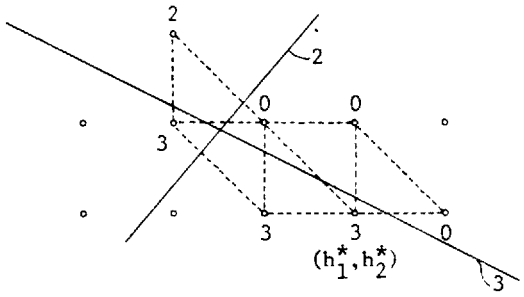


FIGURE 36

inequalities. If, to the contrary, the solution violates inequality number 1, it will receive the label 1 in the context of the problem involving all four rows. But then this triangle, in conjunction with the first slack vector ξ^1, will be an almost completely labeled primitive set. When ξ^1 is eliminated we enter the chain of parallelograms with an almost completely labeled parallelogram. This concludes our description of the algorithm, which can be carried out in a number of computations bounded by a linear function of the number of links in the chain of parallelograms.

11. THE DETERMINATION OF PRIMITIVE SETS

Let A be an integral matrix with two columns and four rows, possibly selected from an activity analysis matrix with a larger number of rows. In order for Theorem 3.3 to be valid it is necessary that the tie-breaking rule used for the submatrix be consistent with the rule used in the larger matrix. In the interest of simplicity, however, we shall restrict our attention to the rule which breaks ties in the ith coordinates of two vectors $(x_0, x_1, x_2, x_3)'$ and $(\hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3)'$ by the lexicographic ordering starting with the 0th coordinate. Other tie-breaking rules may be treated in a similar fashion.

We assume that the production set X, consisting of the vectors $x = Ah$, with h ranging over the lattice points in the plane, contains some efficient vectors. Our purpose will be to describe an algorithm to determine the chain of parallelograms and two pairs of triangles associated with this technology.

Making use of unimodular transformations the collection of quadrilateral primitive sets can be put in the form given by Theorem 9.3: a chain of parallelograms whose first member is the unit square.

Let us begin by observing that there is a very simple test to determine whether the four vertices of the unit square, oriented as in Figure 30, form a primitive set for the matrix A. It is clear from an examination of this figure that a necessary and sufficient condition is that A have the sign pattern:

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ + & + \end{bmatrix}.$$

(Zero entries are, of course, resolved by the lexicographic tie-breaking rule.)

The four vectors in X which describe this primitive set are given by the columns of

$$\begin{array}{cccc} h^0 & h^1 & h^2 & h^3 \\ \begin{bmatrix} \underline{a_{01}+a_{02}} & a_{01} & a_{02} & 0 \\ a_{11}+a_{12} & \underline{a_{11}} & a_{12} & 0 \\ a_{21}+a_{22} & a_{21} & \underline{a_{22}} & 0 \\ a_{31}+a_{32} & a_{31} & a_{32} & \underline{0} \end{bmatrix} & & & \end{array}.$$

The sign pattern of the matrix A tells us that the row minima lie on the main diagonal. Let us assume that the next parallelogram in the chain is obtained by the operation we have called a: the replacement of the vector h^2 above the main diagonal by its reflection through h^0. In the new primitive set h^0 will have the smallest second coordinate, and the replacement $2h^0 - h^2$ will have the smallest zeroth coordinate.

The columns of the following matrix represent the vectors in the new primitive set

$$\begin{bmatrix} \underline{2a_{01}+a_{02}} & a_{01} & a_{01}+a_{02} & 0 \\ 2a_{11}+a_{12} & \underline{a_{11}} & a_{11}+a_{12} & 0 \\ 2a_{21}+a_{22} & a_{21} & \underline{a_{21}+a_{22}} & 0 \\ 2a_{31}+a_{32} & a_{31} & a_{31}+a_{32} & \underline{0} \end{bmatrix}.$$

We see immediately that a necessary and sufficient condition for the new primitive set to be obtained by using operation a is that

$$\begin{bmatrix} a_{01} & a_{01}+a_{02} \\ a_{11} & a_{11}+a_{12} \\ a_{21} & a_{21}+a_{22} \\ a_{31} & a_{31}+a_{32} \end{bmatrix}$$

have the same sign pattern as the original matrix. An identical argument tells us that the successor to the unit square is obtained by operation b if and only if the matrix

$$\begin{bmatrix} a_{01}+a_{02} & a_{02} \\ a_{11}+a_{12} & a_{12} \\ a_{21}+a_{22} & a_{22} \\ a_{31}+a_{32} & a_{32} \end{bmatrix}$$

has the same sign pattern as the original matrix.

The process can obviously be continued to determine all of the parallelograms in the chain. Each such parallelogram has associated with it a matrix with sign pattern

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ + & + \end{bmatrix}.$$

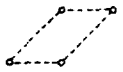
The matrix associated with the successor in the chain is obtained by replacing one of the two columns by the sum of the two columns. Operation a is used if the second column is replaced by the sum and operation b if the first column is replaced.

If the sign pattern of the row sums is such that neither replacement preserves the sign pattern of the matrix, the chain of parallelograms terminates with a pair of

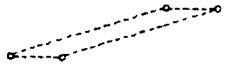
triangles, each of which forms a primitive set in conjunction with one of the slack vectors. We should also remark that a link in the chain is obtained by successively replacing the same column by the sum. The following list of matrices and associated parallelograms illustrates this sequence of steps.

$$\begin{bmatrix} -4 & -1 \\ -2 & 5 \\ 1 & -2 \\ 1 & 1 \end{bmatrix}$$

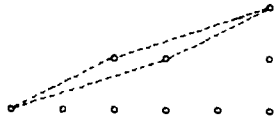
$$\begin{bmatrix} -4 & -5 \\ -2 & 3 \\ 1 & -1 \\ 1 & 2 \end{bmatrix}$$



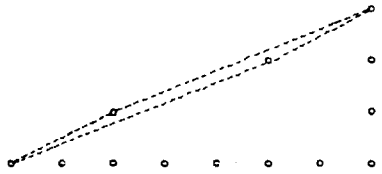
$$\begin{bmatrix} -4 & -9 \\ -2 & 1 \\ 1 & 0 \\ 1 & 3 \end{bmatrix}$$



$$\begin{bmatrix} -13 & -9 \\ -1 & 1 \\ 1 & 0 \\ 4 & 3 \end{bmatrix}$$



$$\begin{bmatrix} -22 & -9 \\ 0 & 1 \\ 1 & 0 \\ 7 & 3 \end{bmatrix}$$



At either end of the chain of parallelograms is a pair of triangles which represent primitive sets in conjunction with a specific slack vector. The chain is entered by removing that slack vector and replacing it by a non-slack vector in X. It is possible, however, that the particular matrix A has no quadrilateral primitive sets associated with it at all. This would be revealed by a triangular primitive set with the property that the replacement for the associated slack vector is, itself, another slack vector.

Let us argue that there are no quadrilateral primitive sets if a unimodular transformation of A can be found which has the sign pattern

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ - & - \end{bmatrix}$$

with $a_{11}+a_{12}>0$ and $a_{21}+a_{22}>0$. Of course, the particular rows in which the + signs appear are arbitrary.

Our argument proceeds by showing that the three columns of

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$
$$h^0 \quad h^1 \quad h^2$$

form a primitive set in conjunction with the third slack vector ξ^3. It would then follow from the replacement operation that the triangles of Figure 37 are primitive sets in conjunction with ξ^3. But the above sign pattern makes no

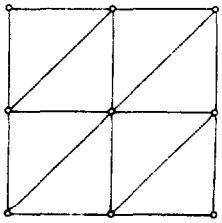


FIGURE 37

distinction between rows 0 and 3. This implies that the same triangles are primitive sets in conjunction with ξ^0. Since all replacements stay within this class we have determined the full collection of primitive sets.

The three vectors in X associated with h^0, h^1, h^2 are given by the columns of

$$\begin{bmatrix} \underline{a_{01}+a_{02}} & a_{01} & 0 \\ a_{11}+a_{12} & \underline{a_{11}} & 0 \\ a_{21}+a_{22} & \underline{a_{21}} & \underline{0} \end{bmatrix},$$

with row minima determined by the sign pattern of A. In asking whether there is a vector $x=Ah$ with $(x_0, x_1, x_2)'$ greater than these row minima we can restrict our attention to vectors h so that the quadrilateral

$$\begin{bmatrix} 1 & 1 & 0 & h_1 \\ 1 & 0 & 0 & h_2 \end{bmatrix}$$

contains no lattice points other than these four vectors themselves. It follows that h must be one of the following three vectors

$$\binom{0}{-1}\binom{0}{1}\binom{2}{1},$$

whose associated vectors in X are

$$\begin{pmatrix} -a_{02} & a_{02} & 2a_{01}+a_{02} \\ -a_{12} & a_{12} & 2a_{11}+a_{12} \\ -a_{22} & a_{22} & 2a_{21}+a_{22} \end{pmatrix}.$$

But $-a_{12} < a_{11}$, $a_{22} < 0$, and $2a_{01} + a_{02} < a_{01} + a_{02}$. This demonstrates that the three vectors h^0, h^1, h^2 do indeed form a primitive set in conjunction with ξ^3.

The remainder of this section will exhibit an algorithm which yields one of the two forms of the following theorem.

THEOREM 11.1: *Let A be a matrix with four rows and two columns whose associated set X contains some efficient vectors. Then there is a unimodular transformation which brings A to one of the two forms*

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ + & + \end{bmatrix}$$

or

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ - & - \end{bmatrix},$$

with the two middle row sums positive. In the first case there are some quadrilateral primitive sets and in the second case there are none. The signs are to be interpreted according to the lexicographic rule and the sign patterns can appear in rows other than those indicated above.

The algorithm has two parts. In the first part we use the Euclidean algorithm to find a unimodular transformation which brings A into one of the following two sign patterns:

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ + & - \end{bmatrix}, \quad \begin{bmatrix} - & - \\ - & + \\ + & - \\ - & - \end{bmatrix},$$

I II

or one of the variants where the sign patterns appear in different rows. In the second part we achieve one or another of the specific forms of Theorem 11.1.

Part 1. The Euclidean algorithm for finding the greatest common divisor of a_{01} and a_{02} can be carried out by a sequence of elementary column operations. Each such operation, and their composition, is equivalent to multiplying A on the right by a unimodular matrix, and results in a new matrix whose zeroth row has the sign

pattern, say, $(-, 0)$. Let us examine the second column

$$\begin{pmatrix} 0 \\ a_{12} \\ a_{22} \\ a_{32} \end{pmatrix}$$

in this matrix.

The nonzero entries in this column cannot all have the same sign if the technology is to have efficient vectors. Assume, first of all, that all three entries are different from 0. Then, by multiplying by -1 if necessary, column two has one of the sign patterns

$$\begin{pmatrix} 0 \\ - \\ + \\ + \end{pmatrix}, \quad \begin{pmatrix} 0 \\ - \\ + \\ - \end{pmatrix}, \quad \begin{pmatrix} 0 \\ - \\ - \\ + \end{pmatrix}.$$

If we then subtract a large multiple of this column from column one, the pattern I is reached.

If, on the other hand, there is another zero entry in column two, we can obtain one of the following patterns:

$$\begin{pmatrix} 0 \\ 0 \\ - \\ + \end{pmatrix}, \quad \begin{pmatrix} 0 \\ - \\ 0 \\ + \end{pmatrix}, \quad \begin{pmatrix} 0 \\ - \\ + \\ 0 \end{pmatrix},$$

and by subtracting a large multiple of this column from column one, one of the two patterns I or II is reached.

Part 2. Let us assume, first of all, that we begin with a matrix of sign pattern II. We proceed through a series of steps, each of which consists of adding a positive integral multiple of one column to the other, so as to preserve the sign pattern. If such an addition is possible, let us add the largest such possible multiple. If no such addition is possible, then the sign pattern of the sum of the columns must be different from either column, and must therefore be

$$\begin{pmatrix} - \\ + \\ + \\ - \end{pmatrix}.$$

Since the process terminates after a finite number of iterations, we reach a matrix of the type described in Theorem 11.1, which reveals that there are no quadrilateral primitive sets.

If, on the other hand, the initial matrix is of sign pattern I, we also add the largest positive multiple of one column to the other so as to preserve this sign pattern.

This can be carried out unless the sign pattern of the sum of the two columns differs from either of them. There are 8 possible sign patterns for the sum, one of which—all signs negative—is ruled out if there are to be efficient vectors in X. Two of the sign patterns duplicate the columns. The remaining 5 patterns are:

$$\begin{array}{ccccc} - & - & - & - & - \\ - & - & + & + & + \\ + & - & + & + & - \\ - & + & + & - & + \\ 1 & 2 & 3 & 4 & 5. \end{array}$$

In the first two cases a matrix with pattern II is reached if the first column is replaced by the sum; similarly for case 3 if the first column is replaced by the negative of the sum. In cases 4 and 5 if the second column is replaced by the sum we reach a matrix of the sort described in Theorem 11.1, revealing the existence of quadrilateral primitive sets. The algorithm terminates in a finite number of steps with the desired conclusion.

12. THE COMPLEXITY OF THE ALGORITHM

In this section we return to the ideas of complexity theory and demonstrate that our accelerated version of Sperner's Lemma is a polynomial algorithm in the size of the problem. Since the algorithm requires a fixed amount of computation for each link in the chain of parallelograms, we shall show that the number of links is bounded by a polynomial function—in fact a linear function—of the problem size, if the matrix has the sign pattern which permits us to recognize a parallelogram in the chain.

In general the matrix A is not given to us in a form which permits an immediate recognition of the primitive sets, and the procedures of Section 11 are required. But identical arguments can be used to show that these procedures terminate in a number of steps which is polynomial in the size of the original problem. Moreover the resulting matrix has a size which is also polynomial in the data. In the interest of simplicity these later arguments will be omitted in the present paper.

Let A be an integral matrix with sign pattern

$$\begin{bmatrix} - & - \\ - & + \\ + & - \\ + & + \end{bmatrix},$$

so that the vertices of the unit square form a primitive set. The number of bits required to represent the integer a_{ij} is $[\log_2 (|a_{ij}|+1)]$, where $[x]$ is the smallest integer $\geq x$. The matrix A therefore has a size given by

$$S = \sum \sum [\log_2 (|a_{ij}|+1)].$$

THEOREM 12.1: *The number of links in the chain of parallelograms associated with A is no larger than $S/2$.*

In order to find the successors to the unit square in the chain of parallelograms we add one of the columns of A to the other in such a way as to preserve the sign pattern. When this can no longer be done the chain of primitive sets terminates.

A link in the chain is obtained by repeated addition of the same column to the other. If it is the first column which is being added to the second this results in a sequence

$$\begin{bmatrix} a_{01} & a_{02} \\ a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \to \begin{bmatrix} a_{01} & a_{01}+a_{02} \\ a_{11} & a_{11}+a_{12} \\ a_{21} & a_{21}+a_{22} \\ a_{31} & a_{31}+a_{32} \end{bmatrix} \to \cdots \to \begin{bmatrix} a_{01} & ta_{01}+a_{02} \\ a_{11} & ta_{11}+a_{12} \\ a_{21} & ta_{21}+a_{22} \\ a_{31} & ta_{31}+a_{32} \end{bmatrix},$$

with the last matrix representing the final primitive set in the link. For the chain to proceed to a subsequent link it must be that the sign pattern of the sum of the two columns in the final matrix is the same as the sign pattern of the first column, or

$$ta_{11}+a_{12} > 0 \geqslant (t+1)a_{11}+a_{12}$$

and

$$ta_{21}+a_{22} \leqslant 0 < (t+1)a_{21}+a_{22}.$$

Let us examine the change in the expression

$$[\log_2 (|a_{11}|+1)]+[\log_2 (|a_{12}|+1)]$$

evaluated at the first primitive set and the last primitive set in the link. The change is given by

$$\Delta = [\log_2 (|a_{12}|+1)]-[\log_2 (|ta_{11}+a_{12}|+1)].$$

But

$$\begin{aligned} a_{12} &= (ta_{11}+a_{12})-ta_{11} \\ &\geqslant (ta_{11}+a_{12})-a_{11} \\ &\geqslant 2(ta_{11}+a_{12}) > 0, \end{aligned}$$

and using the obvious fact that

$$[\log_2 (|2a|+1)] = [\log_2 (|a|+1)]+1,$$

for any nonzero integer a, we see that

$$\Delta \geqslant 1.$$

In the same fashion

$$[\log_2 (|a_{21}|+1)]+[\log_2 (|a_{22}|+1)]$$

decreases by at least one unit as we move through each link in the chain, other than the last link. We see that the number of links which follow the unit square in the chain is not larger than

$$\tfrac{1}{2}\sum_{i=1}^{2}\sum_{j=1}^{2}[\log_2(|a_{ij}|+1)].$$

If the unit square is not the first primitive set in the chain, the matrices representing its predecessors are obtained by repeated subtraction rather than addition. Using the same arguments the number of links which precede the unit square is not larger than

$$\tfrac{1}{2}\sum_{i=0,3}\sum_{j}[\log_2(|a_{ij}|+1)].$$

This demonstrates Theorem 12.1, which is the basic idea of the argument that the accelerated algorithm is polynomial.

The number of parallelograms in the chain of primitive sets is definitely not polynomial in the data. To see this consider the matrix

$$\begin{bmatrix} -1 & -1 \\ -1 & k \\ 1 & -k \\ 1 & 1 \end{bmatrix}$$

whose size is $S = 6 + 2[\log_2(k+1)]$. If the first column is added to the second j times, we obtain

$$\begin{bmatrix} -1 & -1-j \\ -1 & k-j \\ 1 & -k+j \\ 1 & 1+j \end{bmatrix}$$

with sign pattern sufficient to recognize a primitive set for $j = 0, 1, \ldots, k-1$. It follows that there are at least k distinct primitive sets, a number which is exponential in S.

For this series of examples the number of lattice points in the minimal neighborhood of the origin for which a local maximum is global is also exponential in the data. But in spite of the fact that a nonpolynomial number of neighbors of a given lattice point must be examined to verify optimality, there is sufficient structure in the problem so that this can be accomplished in polynomial time. By an analytical investigation the apparently unmanageable complexity of these integer programs receives a structure and coherence leading to a rapid method of solution.

It is my belief that these results are not special to the two variable problem. It is entirely possible that there is a polynomial algorithm for integer programming whenever the number of variables is fixed in advance, but that the degree of the polynomial is exponential in the number of variables. The resolution of this and

related questions seems to me to be of the greatest significance to the problem of economic decision making in the presence of indivisibilities, and ultimately to the construction of a realistic theory of the firm.

In a subsequent paper we shall discuss this approach for the 3-variable integer program.

Yale University

Manuscript received June, 1979; revision received February, 1980.

REFERENCES

[1] HIRSCHBERG, D. S., AND C. K. WONG: "A Polynomial-Time Algorithm for the Knapsack Problem in Two Variables," *Jour. Ass. Comp. Math.*, 23 (1976), 147–154.

[2] KANNAN, ROVINDRAN: "A Polynomial Algorithm for the Two Variable Integer Programming Problem," Technical Report No. 348, School of Operations Research, Cornell University, Ithaca, New York, 1977.

[3] GAREY, MICHAEL R., AND DAVID S. JOHNSON: *Computers and Intractability*. San Francisco: W. H. Freeman, 1979.