POLYNOMIAL-TIME AGGREGATION OF INTEGER PROGRAMMING PROBLEMS

by

R. Kannan

Memorandum No. UCB/ERL M80/2

January 1980

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

POLYNOMIAL-TIME AGGREGATION OF INTEGER PROGRAMMING PROBLEMS*

R. Kannan

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley

Abstract

This paper shows that a set of linear diophantine equations in nonnegative variables with nonnegative coefficients can be reduced to a single equation with the same solution set in polynomial time. A weaker version of the above statement is shown to be true when the coefficients are allowed to be negative. Besides being polynomial-time bounded, the present aggregation scheme differs from existing ones in that the final equation is in variables that are not explicitly bounded and hence admits of a faster to dynamic programming algorithm. We finally present three applications of the aggregation technique of this paper: (i) it is proved that a certain type of knapsack problem cannot have a polynomial time approximation algorithm unless NP = P; (ii) an analog of Farkas' lemma for integer programming is proved and (iii) it is shown that a decision problem involving integer variables is NP-complete.

Keywords: integer programming, polynomial-time algorithm, aggregation, diophantine equations, knapsack problem.

*Research supported by National Science Foundation Grants ENG76-09936 and MCS77-09906-A02.

POLYNOMIAL-TIME AGGREGATION OF INTEGER PROGRAMMING PROBLEMS

Section 1. Introduction

By the integer programming problem, we will mean the question of feasibility of a set of simultaneous equations in nonnegative integers, i.e., a question of the form:

Is there a solution $x = (x_1, x_2, \ldots, x_n)$ to

(1.1) (1.1a) $\sum_{j=1}^{n} a_{ij} x_j = b_i$ for $i = 1,2,\ldots,m$

(1.1b) $x_j \geq 0$ integer for $j = 1,2,\ldots,n$

where all data are integers.

In case all a_{ij} and b_i are nonnegative in (1.1), we present a polynomial-time algorithm for obtaining a single equation in n variables that has the same set of nonnegative integer solutions as (1.1) This is later shown to be extendable to integer programming problems with a bounded feasible set. Since (1.1) is in *NP* (see e.g. Kannan and Monma [78]) and (1.2) below referred to as the knapsack problem throughout this paper is *NP*-complete (Lueker [75]), it is clear that (1.1) can be polynomial-time reduced to (1.2). But here, we present a stronger reduction--one that preserves the set of solutions. As one of the consequences, certain conclusions are drawn about the existence of polynomial-time approximation procedures for (1.2).

Is there a solution $x \in \mathbb{Z}_+^n$ to

(1.2) $$\sum_{j=1}^{n} a_j x_j = b_0$$

where all data are positive integers. Bradley [71], Padberg [72] and Chvatal and Hammer [77] have also dealt with aggregation. However, two qualifications have always been attached to methods of aggregation: (i) the coefficients of the "aggregated" equation may be very large in practice and (ii) it must be assumed that there is a given bound on each component of any integral solution vector for the given system.

The polynomial algorithm below establishes that "in theory" coefficients of the aggregated equation can be kept small. As for the second qualification, it is now known that if an integer programming problem does not already contain explicit bounds on the variables, they can be imposed so that the new problem encoding is of length polynomially bounded in the length of the encoding of the original problem and is feasible if and only if the original problem was (see e.g. Kannan and Monma [78]). These bounds, however, naturally curtail the set of all feasible solutions to the problem. If we then restrict the aim of aggregation to maintaining feasibility as invariant rather than maintaining the same set of feasible solutions, we show in Section 3 that any given integer programming problem can be aggregated in polynomial time to a problem of the form (1.2) (the knapsack problem). The final problem has (n+1) variables where the original integer programming problem had n variables. Further the final aggregated system obtained by Bradley [71] etc. consists of a single linear diophantine equation in nonnegative variables and inequalities giving upper bounds on the variables. Our final system will contain just one equation in nonnegative integer variables. Besides leading to an exact analog of Farkas' lemma for integer programming (Section 5) the latter system (with no explicit bounds on the variables) has an algorithm of lesser time-complexity as pointed in Garfinkel and Nemhauser [72]. We elaborate on thier argument in Section 3.

Hirschberg and Wong [76] have conjectured that the n-variable knapsack problem would have a polynomial-time algorithm when n is considered fixed. It is likely that such an algorithm (if it existed) would be of great practical value. The results of this paper show that if this conjecture were true, then the stronger conjecture obtained by replacing the words "knapsack problem" by "integer programming problem" would also be true.

The following notation will be used throughout.

For a matrix A of integers,

$\|A\|$ = maximum absolute value of any entry of A

A^i = the ith row of A

A_i = the ith column of A

A_{ij} or $A_{i,j}$ is the entry in the ith row and jth column of A

p-time $\equiv$ polynomial time

$1_n = 1 \times n$ vector of all 1's

$\mathbb{Z}^n$ ($\mathbb{Z}^n_+$) is the set of n vectors with (nonnegative) integer components

$\mathcal{R}^n$ ($\mathcal{R}^n_+$) is the set of n vectors with (nonnegative) real components

Section 2. Polynomial-Time Aggregation of Integer Programming Problems with Nonnegative Coefficients

In this section, we consider only integer programming problems with all nonnegative coefficients. (In Section 3, we explain how this restriction may be removed.) Theorem 2.1 states that we can "aggregate" these integer programming problems efficiently. Earliest results on aggregation are probably due to Matthews [1896]. He shows that given a system of two simultaneous Diophantine equations whose variables are restricted to be nonnegative and whose coefficients are nonnegative, there is an (easily obtainable) single Diophantine equation whose set of nonnegative solutions is identical to that of the given system. This can be used to aggregate m equations efficiently to one using the divide and conquer technique by recursively aggregating the first $(\frac{m}{2})$ equations and the last $(\frac{m}{2})$ equations and then combining the two resulting equations. We shall give a more direct method as proof of the following theorem.

Theorem 2.1. Let A and b be $m\times n$ and $m\times 1$ matrices, respectively, with all nonnegative integer entries where $\|A\|$ and $\|b\|$ are nonzero. Let $\lambda = 3nm\|A\|\|b\|$. Then

$$\{x\in\mathbb{Z}_+^n: \sum_{j=1}^{n}\left(\sum_{i=1}^{m}(\lambda^{m+1}+\lambda^i)A_{ij}\right)x_j = \sum_{i=1}^{m}(\lambda^{m+1}+\lambda^i)b_i\}$$
$$= \{x\in\mathbb{Z}_+^n: Ax=b\} .$$

Proof. Denote the two sets of the theorem by T' and T, respectively. Then it is clear that T' contains T. Suppose now that there is an x in T' that does not satisfy one of the equations of $Ax = b$. Let i_0 be the largest i for which $A^i\cdot x \neq b_i$. Since x is in T', we must have

$$\|x\| \leqq \frac{(m\lambda^{m+1} + (\lambda^{m+1}-1))\|b\|}{\lambda^{m+1}}$$

$$\leqq (m+1)\|b\| \leqq (\lambda-1)/(n\|A\|) \ . \qquad (2.1)$$

Thus we have

$$0 \leqq A^i \cdot x \leqq (\lambda-1) \quad \forall i \ .$$

Therefore

$$|A^i \cdot x - b_i| \leqq \max\{\|b\|, \lambda-1\} = (\lambda-1) \ . \qquad (2.2)$$

Since x is in T', we must have

$$\sum_{i=1}^{m} (\lambda^{m+1}+\lambda^i)(A^i \cdot x - b_i) = 0 \ ;$$

i.e.,

$$-\sum_{i=1}^{m} \lambda^i (A^i \cdot x - b_i) = \lambda^{m+1}\left(\sum_{i=1}^{m} A^i \cdot x - b_i\right) \ . \qquad (2.3)$$

By (2.2), the magnitude of the left hand side in the above equation is at most

$$\sum_{i=1}^{m} \lambda^i (\lambda-1) = \lambda^{m+1} - \lambda \leqq \lambda^{m+1} - 1 \ .$$

But λ^{m+1} divides the left hand side of (2.3). Thus both sides of the equation must be zero and hence,

$$\sum_{i=1}^{m} \lambda^i (A^i \cdot x - b_i) = 0 \ . \qquad (2.4)$$

By the definition of i_0, we have

$$\sum_{i=1}^{i_0-1} \lambda^i (A^i \cdot x - b_i) = -(A^{i_0} \cdot x - b_{i_0})\lambda^{i_0} \qquad (2.5)$$

and the right hand side of this equation is nonzero. Thus the left hand side is also nonzero and is a multiple of λ^{i_0}. But an argument similar to the one used to derive (2.4) shows that the magnitude of the left hand side is at most $\lambda^{i_0}-1$. Thus we arrive at a contradiction and must have that x belongs to T. $\square$

Since the sizes of $(\lambda^{m+1}+\lambda^{i})$ are all polynomially bounded, the aggregated equation can be arrived at in polynomial-time.
Hence, we have

Theorem 2.2. Given A and b $m\times n$ and $m\times 1$ nonzero matrices, respectively, with nonnegative integral entries, we can find in polynomial-time, a vector u in $\mathbb{Z}^m$ such that

$$\{x\in\mathbb{Z}_+^n: Ax=b\} = \{x\in\mathbb{Z}_+^n: (uA)\cdot x = u\cdot b\}$$

Section 3. Aggregation of General Integer Programming Problems to Knapsack Problems

The word aggregation is used here in a different sense than in the previous section. There it meant "given a system of linear Diophantine equations, find a single linear Diophantine equation which has the same set of nonnegative solutions as the given system." In this section, the final equation will be required to have a nonnegative solution if and only if the initial system had one. However, it will not, in general, have the same set of solutions as the given system.

We need the following result proved independently by various authors (Kannan and Monma [78], Cook [76] and Borosh and Treybig [76]).

Theorem 3.1. There exists a polynomial $p(\cdot)$ such that for all integral matrices A and b (of appropriate dimensions),

$$S(A,b) = \{x \in \mathbb{Z}_+^n : Ax = b\}$$

is nonempty iff

$$S(A,b) \cap \{x : \|x\| \leqq 2^{p(\text{length of encoding of } A,b)}\}$$

is nonempty.

We now prove the main result of the section.

Theorem 3.2. Let A and b be $m \times n$ and $m \times 1$ all-integer matrices. Then there exists a p-time computable $1 \times n$ integral matrix u and integers s and t such that

$$\exists x \in \mathbb{Z}_+^n : Ax = b \Leftrightarrow \exists x \in \mathbb{Z}_+^n, y \in \mathbb{Z}_+ : ux + sy = t$$

and

$$ux + sy = t;\ x \in \mathbb{Z}_+^n,\ y \in \mathbb{Z}_+ \Rightarrow Ax = b\ .$$

Proof. From Theorem 3.1, there exists a p-time computable integer B such that

$$\exists x \in \mathbb{Z}_+^n: Ax = b \Leftrightarrow \exists x \in \mathbb{Z}_+^n: Ax = b;\ x_i \leqq B\ \forall i\ .$$

Thus,

$$\exists x \in \mathbb{Z}_+^n: Ax = b \Leftrightarrow \exists x \in \mathbb{Z}_+^n,\ y \in \mathbb{Z}_+: Ax = b;\ \sum_{i=1}^{n} x_i + y = nB$$

$$\Leftrightarrow \exists (x,y) \in \mathbb{Z}_+^{n+1}:$$

$$\begin{pmatrix} A & 0 \\ 1_n & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ nB \end{pmatrix}$$

We note that adding a nonzero multiple of an equation to another equation in a linear system does not change the set of solutions. By adding suitable multiples of the last equation to each of the others in the above system, we can make all the coefficients of the system nonnegative. Of course this process leaves the size of the encoding of the numbers bounded by a polynomial in the length of the input. Further, if any of the right hand side constants is negative at this stage, the system is inconsistent and an obviously inconsistent single equation may be returned as the aggregated equation. If all the coefficients and the right hand constants are nonnegative, then Theorem 2.2 implies the current theorem. □

The main differences between aggregation schemes found in the literature and Theorem 3.2 are

(1) In Theorem 3.2, the final equation can be written down in p-time.

(2) We do not reuqire that the set $\{x \in \mathbb{Z}_+^n: Ax = b\}$ be bounded.

(3) Whereas in known aggregation schemes, the final system is of the form:

$$\sum_{i=1}^{n} a_i x_i = b_0$$

$$x_i \geqq 0 \quad \text{integers} \quad \forall i,$$

and $$x_i \leqq B \quad \forall i$$

Our final system does not contain explicit upper bounds on the variables x_i. This gives a computational advantage as explained below.

It is not difficult to see that the following result about optimization problems follows from Theorem 3.1.

Corollary 3.3. There exists a polynomial $p(\cdot)$, such that for $m \times n$, $m \times 1$ and $1 \times n$ all integer matrices A, b and c,

$$\begin{array}{ccc} \text{maximum } c\cdot x & = & \text{maximum } c\cdot x \\ \text{such that } Ax = b & & \text{such that } Ax = b \\ x \in \mathbb{Z}_+^n & & x \in \mathbb{Z}_+^n;\ x_i \leqq 2^{p(\ell)} \end{array}$$

where ℓ = length of encoding of A, b and c.

Thus using the proof of Theorem 3.2, the integer programming optimization problem can be reduced to the knapsack optimization problem

(3.1) $$\text{maximize} \quad \sum_{i=1}^{n} c_i x_i$$

subject to (3.1a) $$\sum_{i=1}^{n} a_i x_i = b_0$$

(3.1b) $$x_i \geqq 0 \quad \text{integers}$$

Whereas, existing aggregation schemes would give rise to

$$\text{(3.2)} \qquad \text{maximize} \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{subject to} \quad \text{(3.2a)} \quad \sum_{i=1}^{n} a_i x_i = b_0$$

$$\text{(3.2b)} \quad x_i \geqq 0 \quad \text{integers}$$

$$\text{and} \quad \text{(3.2c)} \quad x_i \leqq B \quad \forall i$$

where all data are nonnegative integers. We elaborate here on a remark in Garfinkel and Nemhauser [72] that points out why (3.1) is easier to solve than (3.2).

For any $k \leqq n$ positive integer and y nonnegative integer define

$$f_k(y) = \text{maximum value of} \quad \sum_{i=1}^{k} c_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{k} a_i x_i = y$$

$$x_i \geqq 0 \quad \text{integers}$$

and

$$f'_k(y) = \text{maximum value of} \quad \sum_{i=1}^{k} c_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{k} a_i x_i = y$$

$$B \geqq x_i \geqq 0 \quad \text{integers}$$

where the maximum is taken to be $-\infty$ if the constraints are infeasible.

Then obviously to solve (3.1), we can use the dynamic programming recursion:

$$f_{k+1}(y) = \max\{f_k(y), f_{k+1}(y-a_{k+1}) + c_{k+1}\} .$$

Thus if f is computed in the order

$$f_1(0), f_1(1), \ldots, f_1(b), f_2(0), \ldots, f_2(b), \ldots$$

since the computation of each $f_k(y)$ requires $O(1)$ operations, $f_n(b)$, the solution value can be computed in $O(n\cdot b)$ operations. However, a similar recursion does not apply to $f'_k(y)$. One is forced to use some other dynamic programming equation and the usual one is

$$f'_{k+1}(y) = \max_{x_{k+1}=0,1,\ldots,B} \{f_k(y-a_{k+1}x_{k+1}) + c_{k+1}x_{k+1}\} .$$

Each $f'_k(y)$ thus requires $O(B)$ operations and hence $f'_n(b)$ takes $O(nbB)$ steps instead of $O(nb)$ steps.

Section 4. Aggregation and Approximative Algorithms

The following theorem due to Luekar follows as a corollary to Theorem 3.2.

Theorem 4.1. The knapsack problem (1.2) is *NP*-complete.

Proof. From Karp [75] we know that the integer programming problem (1.1) is *NP*-complete. Theorem 3.2 gives a polynomial-time reduction of this problem to (1.2). □

The fact that the set of solutions is preserved in the reduction of Theorem 2.2 gives us a stronger result. It establishes that unless $NP = P$, there is no polynomial-time algorithm that finds even an approximate solution to the optimization problem (3.1). To make this more precise, we briefly define various types of approximation algorithms. These definitions have been widely used in the literature (Sahni and Gonsalves [76], Garey and Johnson [76], Ibarra and Kim [75]).

By an optimization problem here, we will mean a problem of the form

$$\text{(4.1)} \qquad \begin{aligned} &\text{maximize} \quad c \cdot x \\ &\text{subject to} \quad x \in s \end{aligned}$$

where $c \cdot x$ is a vector dot product and s is a set suitably defined. $c \cdot x$ is called the objective function.

Definition 4.1. For $e > 0$, an e-approximation algorithm for the problem is an algorithm that produces a solution x with objective function value P such that if the optimal objective function value P^* is nonzero, then

$$\text{(4.2)} \qquad \left|\frac{P^*-P}{P^*}\right| \leqq \epsilon \ .$$

Definition 4.2. An approximation scheme for the problem is an algorithm which given an $\epsilon > 0$ and an instance of the problem produces a solution x with value P satisfying (4.2).

Definition 4.3. A p-time approximation scheme (PAS) is an approximation scheme whose running time is bounded by a function $f(\frac{1}{\epsilon}, \ell)$ where ℓ is the length of the encoding of the instance presented such that $f(\frac{1}{\epsilon_0}, \ell)$ is a polynomial in ℓ for each fixed value ϵ_0 of ϵ.

Definition 4.4. A fully-polynomial approximation scheme (FPAS) is a PAS whose running time is bounded by a polynomial in ℓ and $1/\epsilon$.

Ibarra and Kim [75] have given a FPAS for solving the problem (4.3) below where x_i are further constrained to be 0 or 1. As Lawler [79] points out, this can be extended to a FPAS for (4.3):

$$\text{(4.3)} \qquad \text{maximize} \sum_{j=1}^{n} c_j x_j$$
$$\text{subject to} \sum_{j=1}^{n} a_j x_j \leqq b_0; \quad x \in \mathbb{Z}_+^n$$

where c_j, a_j and b_0 are nonnegative integers. In spite of the similarity of this problem to (3.1), here we show that replacing the inequality by an equation makes the problem more difficult in the following sense:

Theorem 4.2. (3.1) has a p-time e-approximation algorithm for at least one fixed $\epsilon > 0$ if and only if $NP = P$.

Proof. If $NP = P$, it is clear that (1.1) is in P. Thus an exact solution of (3.1) may be found in polynomial time by solving polynomially many instances of (1.1).

Conversely, Theorem 2.2 implies that (3.1) has a p-time e-approximation algorithm iff the integer programming optimization problem (4.4) below with nonnegative coefficients has a p-time e-approximation algorithm

$$\text{(4.4)} \qquad \begin{aligned} \text{maximize} \quad & c\cdot x \\ \text{subject to} \quad & Ax = b \\ & x \in \mathbb{Z}_+^n \end{aligned}$$

We use a proof technique of Sahni and Gonsalves to show that (4.4) has a p-time ϵ-approximation algorithm for some $\epsilon > 0$ iff $NP = P$.

It is known that the following problem known as the subset sum problem is NP-complete:

(4.5) Given (n+1) positive integers $w_1, w_2, \ldots, w_n$ and w_{n+1}, does there exist an $I \subseteq \{1,2,\ldots,n\}$ such that

$$\sum_{i\in I} w_i = w_{n+1} \ ?$$

Let k be a positive integer and consider

$$\text{(4.6)} \qquad \begin{aligned} \text{maximize} \quad & -ky + z \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i x_i + y = w_{n+1} \\ & x_i + x_i' = 1 \\ & z = 1 \\ x_i, x_i', y, z \geqq 0 \quad & \text{integers.} \end{aligned}$$

If the answer to (4.5) is yes, then the optimal solution value of (4.6) is 1. Otherwise, the optimal solution value is at most (1-k). Suppose (4.4) has a p-time ϵ-approximation algorithm for some fixed $\epsilon > 0$. Then given an instance of (4.5), we do the following:

If $n < 2e$, solve (4.5) by enumeration. Otherwise, write down the corresponding instance of (4.6) with $k = n$ and noting that all coefficients are nonnegative, apply the e-approximation algorithm to get a solution.

Then it is clear that the answer to the instance of (4.5) is Yes iff the e-approximate solution is $(1 - \frac{n}{2})$ or greater.

We note that since $k = n$, (4.6) can be obtained from (4.5) in p-time and thus we have proved the theorem. $\square$

Section 5. Analog of Farkas' Lemma for Bounded Integer Programming

Theorem 5.1 (Farkas' Lemma). For any $m \times n$ and $m \times 1$ rational matrices A and b, precisely one of the following alternatives holds:

(5.1a) $\exists x$ such that $Ax = b$; $x \geqq 0$

(5.1b) $\exists u \in \mathbb{Z}^m$ such that $(uA) \geqq 0$ and $ub < 0$.

The usual statement of Farkas' Lemma (see, for example, Gale [60]) does not assume rationality of A and b and does not assert that u is in $\mathbb{Z}^m$. It has been stated here in the form most useful to us. Noting that (5.1b) is equivalent to (5.2b) below, we rewrite the theorem as:

Theorem 5.2 (Farkas' Lemma restated). For any $m \times n$ and $m \times 1$ rational matrices A and b, precisely one of the following alternatives holds:

(5.2a) $\exists x$: $Ax = b$; $x \geqq 0$

(5.2b) $\exists u \in \mathbb{Z}^m$: $(uA)x = ub$; $x \geqq 0$ has no solution x.

An analogous result has been stated as a classical result in Edmonds and Giles [77]. We restate this theorem in a form similar to Theorem 5.2.

Theorem 5.3. For any $m \times n$ and $m \times 1$ rational matrices A and b, precisely one of the following alternatives holds:

(5.3a) $\exists x$: $Ax = b$; $x \in \mathbb{Z}^n$

(5.3b) $\exists u \in \mathbb{Z}^m$: $(uA)x = ub$; $x \in \mathbb{Z}^n$ has no solution x.

We prove the following theorem here.

Theorem 5.4. For any $m\times n$ and $m\times 1$ rational matrices A and b such that $\{x\in \mathbb{R}^n_+ : Ax=b\}$ is bounded, precisely one of the following alternatives holds:

(5.4a) $\exists x$: $Ax = b;\ x\in \mathbb{Z}^n_+$

(5.4b) $\exists u\in \mathbb{Z}^m$: $(uA)x = ub;\ x\in \mathbb{Z}^n_+$ has no solution x.

Before proving the theorem, we give an example to show that the hypothesis "$\{x\in \mathbb{R}^n_+ : Ax=b\}$ is bounded" is essential. Consider

$$A = \begin{pmatrix} 2 & -2 & -1 & 0 \\ -2 & 2 & 0 & -1 \end{pmatrix}$$

$$b = \begin{pmatrix} 3 \\ -3 \end{pmatrix}$$

For any solution $x\in \mathbb{Z}^n_+$ to the system, we must have $x_3+x_4 = 0$ (by adding the equations), i.e., we must have $x_3 = x_4 = 0$. But since 2 does not divide 3, no solution is possible. Thus (5.4a) fails. Suppose the alternative (5.4b) holds for $u = (u_1\ u_2)$. By dividing through, if necessary, we can assume that u_1 and u_2 are relatively prime. Thus either u_1 or u_2 is not divisible by 2. We will only treat the case when 2 does not divide u_1. The other case is similar. $2 \nmid u_1$ implies that $\pm 2(u_1-u_2)$ and u_1 are relatively prime. Further, u_1 and u_2 are nonzero, since each individual equation has a solution in nonnegative integers. If $u_1 = u_2$, then $x = 0$ solves $(uA)x = ub$. Thus, $u_1 \neq u_2$. Thus, both u_1 and $\pm 2(u_1-u_2)$ are nonzero. uA therefore has two nonzero components of opposite signs that are relatively prime. By Proposition 5.5 below, $(uA)x = 1$ has a solution x in $\mathbb{Z}^n_+$. Further, since one of $\pm 2(u_1-u_2)$ is negative, $(uA)x = -M$ has a solution x in $\mathbb{Z}^n_+$ for arbitrarily large positive M. Thus $(uA)x = ub$ has a solution x in $\mathbb{Z}^n_+$.

Note that here $\{x: Ax = b;\ x \geqq 0\}$ is unbounded.

<u>Proposition 5.5</u>. For any two nonzero integers r and s with $\mathrm{GCD}(r,s) = 1$ and $rs < 0$, there exist nonnegative integers p and q such that $pr + qs = 1$.

<u>Proof</u>. From basic arithmetic, there are integers p and q such that $pr + qs = 1$. If p and q are both nonzero and of different signs, then pr and qs are both of the same sign and each at least 1 in magnitude. Thus, we cannot have this case. Hence for any p, q such that $pr + qs = 1$, adding a suitable multiple of $|s|$ to p and the same multiple of $|r|$ to q establishes the proposition.

<u>Proof of Theorem 5.4</u>. We multiply each equation in $Ax = b$ by an integer, if necessary, to make all data integral. It is clear that both (5.4a) and (5.4b) cannot hold. We assume that (5.4a) does not hold and prove (5.4b) holds. If $\{x: Ax = b;\ x \geqq 0\}$ is empty, (5.2b) and hence (5.4b) hold. Thus, assume that this set is nonempty. Then the boundedness hypothesis of the theorem implies that

$$\nexists x:\quad Ax = 0;\ x \geqq 0;\ x \neq 0.$$

Equivalently,

$$\nexists x:\quad Ax = 0;\ x \geqq 0;\ \sum_{i=1}^{n} x_i = 1.$$

Using Farkas' lemma (Theorem 5.1) with the $(m+1) \times n$ matrix $\bar{A}$ obtained by augmenting A with an $(m+1)^{st}$ row of 1's we have

$$\exists \bar{u}' \in Z^{m+1}:\quad \bar{u}'\bar{A} \geqq 0$$

and

$$\bar{u}'_{m+1} < 0\ .$$

Thus if u' denotes the vector of the first m components of $\bar{u}'$, then $u'A > 0$. Hence, there exists a positive integer k such that

$$A^i + ku'A \geqq 0 \quad \forall i \ .$$

Clearly,

$$\begin{aligned}\{x \in \mathbb{R}^n_+ : Ax = b\} &= \{x \in \mathbb{R}^n_+ : Ax = b;\ ku'Ax = ku' \cdot b\} \\ &= \{x \in \mathbb{R}^n_+ : (A^i + ku'A) \cdot x = b_i + ku' \cdot b \text{ for } i = 1,2,\dots,m \\ &\qquad \text{and } ku'Ax = ku' \cdot b\} \ .\end{aligned}$$

Noting that the last equation is implied by the first m and that the above set is nonempty,

$$\{x \in \mathbb{Z}^n_+ : Ax = b\} = \{x \in \mathbb{Z}^n_+ : (A^i + ku'A) \cdot x = b_i + ku' \cdot b \ \forall i\}$$

where all coefficients are nonnegative in the last system. Note that the validity of (5.4a) and (5.4b) are invariant under this change. Thus, A and b can be assumed to be nonnegative integral matrices and hence Theorem 2.1 implies Theorem 5.4. □

We now contrast the three apparently similar Theorems 5.2, 5.3, and 5.4. In the second theorem given A and b we can assume that A is of full row rank and that A and b are integral without loss of generality. Using the results of Kannan and Bachem [79] we can compute in polynomial-time unimodular matrices (i.e., matrices with integral entries and a determinant of ±1) U and K such that $UAK = S(A)$ has the form:

$$\begin{bmatrix} * & 0 & & & 0 & 0 & \cdots & 0 \\ 0 & * & & & & & & \\ & & \ddots & & & & & \\ & & & \ddots & 0 & & & \\ 0 & \cdots & \cdots & 0 & * & 0 & \cdots & 0 \end{bmatrix}$$

where $*$ is a nonzero entry. By the unimodularity of K,

$\exists x \in \mathbb{Z}^n$ such that $Ax = b$

$\Leftrightarrow$ $\exists x \in \mathbb{Z}^n$ such that $S(A)x = Ub$

$\Leftrightarrow$ each nonzero diagonal entry of $S(A)$ divides the entry of Ub in its row.

Thus for any given integral A and b, (5.3a) fails

$\Leftrightarrow$ $\exists i$ such that $(S(A))_{ii} \nmid (Ub)_i$

i.e., $(S(A))^i \cdot x = (Ub)^i$ has no solution $x \in \mathbb{Z}^n$, i.e.,

$(U^i AK)x = U^i \cdot b$ has no solution x in $\mathbb{Z}^n$

$\Leftrightarrow$ $(U^i A)x = U^i \cdot b$ has no solution x in $\mathbb{Z}^n$
(since K is unimodular)

Thus, given A and b, we can compute $S(A)$, U and K in polynomial time, check if (5.3a) holds and, if not, return $u = U^i$ above such that the other alternative of the theorem holds.

In Theorem 5.2, for any fixed u, the validity of (5.2b) can be easily checked. It is valid if and only if at least one of $(uA)_1, (uA)_2, \ldots, (uA)_n$ has the same sign as (ub). Recently, Khacian [79] gave a polynomial-time algorithm to solve linear programming which can be used to decide which of the alternatives of Theorem 5.2 hold.

In Theorem 5.4, even for a fixed u, the decision as to whether (5.4b) is valid is *NP*-complete (Lueker [75]). We note that when the given A and b have all nonnegative entries a u can be computed in polynomial time using Theorem 2.2 such that one of the alternatives (5.4a) or (5.4b) must hold for this A, b and u. However, this of course does not lead to a polynomial-time bounded algorithm for deciding which of the alternatives holds in Theorem 5.4 because of the fact that there is no known good algorithm to decide (5.4b) even for a fixed u.

Section 6. An *NP*-Completeness Result Using Aggregation

As a final application of polynomial-time aggregation, we wish to establish that question (6.1) below is *NP*-complete.

(6.1) Do there exist integers $x_1, x_2, \ldots, x_n$ such that

$$\sum_{i=1}^{n} a_i x_i = b \quad \text{and} \quad \sum_{i=1}^{n} |x_i| \leqq k$$

where a_i, b and k are given integers.

Note that this would prove that the optimization problem (6.2) has a polynomial-time algorithm if and only if $P = NP$.

(6.2)
$$\begin{aligned} &\min \sum_{i=1}^{n} |x_i| \\ &\text{subject to } \sum_{i=1}^{n} a_i x_i = b \\ &x_i \text{ integers} \end{aligned}$$

Theorem 6.1. Question (6.1) is *NP*-complete.

Before giving the formal proof of the theorem, we give an idea of the proof. We wish to reduce the knapsack problem (1.2) (repeated here for convenience as (6.3)) to (6.1).

(6.3) Do there exist $x_1, x_2, \ldots, x_n \geqq 0$ integers such that $\sum_{i=1}^{n} a_i x_i = b$?

where a_i and b are given nonnegative integers. Consider the system

(6.4)
$$\begin{gathered} a_i x_i = y_i \quad \text{for } i = 1,2,\ldots,n \\ \sum_{i=1}^{n} y_i = b \\ \sum_{i=1}^{n} |y_i| = b \; ; \quad x_i,\, y_i \text{ integers} \end{gathered}$$

If (6.3) has a feasible solution, then with $y_i = a_i x_i$, (6.4) would have a feasible solution, because $a_i \geqq 0$, $x_i \geqq 0 \Rightarrow y_i \geqq 0 \Rightarrow y_i = |y_i| \ \forall i$. Conversely, suppose (6.4) had a feasbile solution. Then,

$$\sum_{i=1}^{n} y_i = \sum_{i:y_i>0} y_i + \sum_{i:y_i<0} y_i = b$$

$$\sum_{i=1}^{n} |y_i| = \sum_{i:y_i>0} y_i - \sum_{i:y_i<0} y_i = b$$

Thus the second sum must be zero. Hence all y_i are nonnegative, and therefore so are the x_i. Thus the x_i are a feasible solution to (6.3). The idea of the proof of Theorem 6.1 is to aggregate in polynomial-time the first (n+1) equations of (6.4) to get a form like (6.1). However in (6.4) the last equation involves only the absolute values of the y_i's whereas in (6.1) we take the sum of the absolute values of all variables. This and certain other details are taken care of in the implicit aggregation of the proof that follows.

Proof of Theorem 6.1. The following lemma establishes the theorem.

Lemma 6.2. A given instance of (6.3) has a Yes answer iff (6.5) below has a Yes answer.

(6.5) Do there exist $x_1, x_2, \ldots, x_{2n}$ integers such that

$$\sum_{i=1}^{2n} |x_i| \leqq s^2$$

and

$$(6.6) \quad a_1(1+\lambda s)x_1 + a_2(1+\lambda^2 s)x_2 + \cdots + a_n(1+\lambda^n s)x_n + \lambda x_{n+1} + \lambda^2 x_{n+2} + \cdots + \lambda^n x_{2n} = b$$

where a_i and b are as in the given instance of (6.3) and

$$s = b+1 \quad \text{and} \quad \lambda = 3ns^3b+b+1 .$$

Proof. If the answer to (6.3) is Yes, let $x_1, x_2, \dots, x_n \geqq 0$ integers satisfy $\sum_{i=1}^{n} a_i x_i = b$. Put $x_{n+i} = -sa_i x_i$ for $i = 1,2,\dots,n$. Then,

$$\sum_{i=1}^{2n} |x_i| \leqq b+bs \leqq s^2$$

and equation (6.6) is satisfied. Thus the answer to (6.5) is Yes. We now set out to prove the converse.

Suppose for integers $x_1, x_2, \dots, x_{2n}$, $\sum_{i=1}^{2n} |x_i| \leqq s^2$ and (6.6) is satisfied. Rewrite (6.6) as

$$(6.7) \quad a_1x_1 + a_2x_2 + \cdots + a_nx_n + \lambda(sa_1x_1+x_{n+1}) + \lambda^2(sa_2x_2+x_{n+2}) + \cdots + \lambda^n(sa_nx_n+x_{2n}) = b .$$

Claim 6.3. $sa_ix_i + x_{n+i} = 0 \ \forall i, 1 \leqq i \leqq n$

Proof. Suppose not. Let j be the largest value of i such that the equation is violated. Then

$$\begin{aligned}|a_1x_1 + a_2x_2 + \cdots + a_nx_n + \lambda(sa_1x_1+x_{n+1}) + \lambda^2(sa_2x_2+x_{n+2}) + \lambda^{j-1}(sa_{j-1}x_{j-1}+x_{n+j-1})| \\ \geqq |\lambda^j(sa_jx_j + x_{n+j})| - |b| \\ \geqq \lambda^j - b\end{aligned}$$

But $|x_i| \leqq s^2 \ \forall i$. Substituting this into the above inequality, we get (using $\max a_i \leqq b$)

$$ns^2b + [sbs^2+s^2][\lambda+\lambda^2 + \cdots + \lambda^{j-1}] \geqq \lambda^j - b$$

i.e.,

$$b + ns^2b + 2s^3b(\frac{\lambda^j-1}{\lambda-1}) \geqq \lambda^j$$

i.e.,

$$(\lambda-1)ns^2b + 2s^3b\lambda^j + b(\lambda-1) \geqq \lambda^j(\lambda-1)$$

i.e.,

$$\frac{b(\lambda-1)}{\lambda^j} + \frac{\lambda-1}{\lambda^j}ns^2b + 2s^3b \geqq \lambda - 1 .$$

Since $j \geqq 1$, this implies that

$$b + ns^2b + 2s^3b + 1 \geqq \lambda \Rightarrow b + 3ns^3b \geqq \lambda$$

contradicting the definition of λ. Hence Claim 6.3 follows. □

<u>Claim 6.4</u>. $\{x_i\colon i = 1,2,\ldots,n\}$ and hence $\{x_i\colon i = n+1,\ldots,2n\}$ are of the same sign.

<u>Proof</u>. By (6.7) and Claim 6.3, $a_1x_1 + \cdots + a_nx_n = b$. Suppose $\{x_i\}_{i=1}^n$ are not all of the same sign. Then

$$\sum_{i=1}^n |a_ix_i| \geqq b + 1 .$$

Thus

$$\begin{aligned} \sum_{i=1}^n |x_{n+i}| &= \sum_{i=1}^n s|a_ix_i| \quad \text{(by Claim 6.3)} \\ &\geqq sb + s \\ &= s^2 \end{aligned}$$

Thus $\sum_{i=1}^{2n} |x_i| \geqq s^2 + 1$, a contradiction. □

Since $\{x_i\}_{i=1}^n$ are all of the same sign and $a_1x_1 + \cdots + a_nx_n = b$, we must have $x_i \geqq 0 \ \forall i$. Hence Theorem 6.10 is proved. □

Acknowledgments

I wish to thank my Ph.D. thesis advisor Professor L. E. Trotter for encouragement during the research and for critically reading the thesis where several sections of this paper first appeared. Thanks are due to Professor R. M. Karp for financial support and several discussions during the writing of the paper.

References

I. Borosh and L. B. Treykig, "Bounds on positive solutions of linear Diophantine equations," Proceedings of the American Mathematical Society 55 (1976), 299.

G. H. Bradley, "Transformation of integer programs to knapsack problems," Discrete Mathematics 1:1 (1971) 29-45.

V. Chvátal and P. L. Hammer, "Aggregation of inequalities in integer programming," Annals of Discrete Mathematics 1 (1977).

S. A. Cook, "A short proof that the linear diophantine problem is in NP," private communication (Oct. 1976).

J. Edmonds and F. R. Giles, "A min-max relation for submodular functions on graphs," Annals of Discrete Mathematics 1 (1977) 185-204.

D. Gale, The Theory of Linear Economic Models, McGraw-Hill Book Company, New York, 1960.

M. R. Garey and D. S. Johnson, "Approximate algorithms for combinatorial problems: an annotated bibliography," in J. F. Traub (ed.), 1976.

R. Garfinkel and G. L. Nemhauser, Integer Programming, J. Wiley & Sons, New York, 1972.

Hirschberg and Wong, "A polynomial algorithm for the knapsack problem in two varibles," Journal of the Association for Computing Machinery 23:1 (January 1976) 147-154.

O. H. Ibarra and Ch. E. Kim, "Fast approximation algorithms for the knapsack and subset sum problems," Journal of the Association for Computing Machinery 22:4 (1975) 463-468.

R. Kannan and A. Bachem, "Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix," Technical Report No. 7895-OR, Institut für Operations Research, University of Bonn. To appear in SIAM Journal on Computing.

R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in R. Henn, B. Korte and W. Oettli (eds.), Lecture Notes in Economics and Mathematical Systems 157, Springer-Verlag, 1978.

R. M. Karp, "On the computational complexity of combinatorial problems," Networks 5 (1975) 44-68.

L. G. Khacian, "A polynomial-time algorithm for the linear programming problem," Doklady Akademii Nauk SSSR 244:5 (1979) 1093-1096.

E. L. Lawler, "Fast approximation algorithms for knapsack problems," Mathematics of Operations Research 4:4 (Nov. 1979) 339-356.

G. S. Lueker, "Two polynomial complete problems in nonnegative integer programming," TR-178, Department of Computer Science, Princeton University, March 1975.

G. B. Matthews, "On partition of numbers," Proceedings of the London Mathematical Society 28 (1896) 486-490.

M. W. Padberg, "Equivalent knapsack-type formulations of bounded integer linear prohrams," Naval Research Logistics Quarterly 19 (1972) 699-708.

S. Sahni and T. Gonzales, "P-complete approximation problems," Journal of the Association for Computing Machinery 23 (1976) 555-565.