ON SOLVING LINEAR DIOPHANTINE SYSTEMS USING GENERALIZED ROSSER'S ALGORITHM

M. KHORRAMIZADEH AND N. MAHDAVI-AMIRI*

Communicated by Mohammad Asadzadeh

ABSTRACT. A difficulty in solving linear Diophantine systems is the rapid growth of intermediate results. Rosser's algorithm for solving a single linear Diophatine equation is an efficient algorithm that effectively controls the growth of intermediate results. Here, we propose an approach to generalize Rosser's algorithm and present two algorithms for solving systems of linear Diophantine equations. Then, we show that the generalized approach provides us with a new formulation of the *LDSSBR* of Chou and Collins and a more efficient implementation of Rosser's approach. The new formulation also enables us to propose an efficient algorithm for solving rank one perturbed linear Diophantine systems based on the *LDSSBR*, and to improve and extend the class of integer *ABS* algorithms for solving linear Diophantine systems.

1. Introduction

Systems of linear Diophantine equations arise from many disciplines (e.g., integer programming [19], Frobenius problem [14], market split problem [22], complex chemical reactions [20]), and computing effective solutions of such systems is highly desired. Several algorithms exist

for solving systems of linear Diophantine equations. Blankinship [6] introduced a procedure for triangulation of a matrix with integer components, and used the procedure for solving simultaneous linear Diophantine equations [7]. Bradley [8] made some modifications to Blankinship's algorithm and introduced a new algorithm for solving systems of linear Diophantine equations. While algorithms introduced by Blankinship and Bradley were based on an explicit calculation of greatest common divisor (gcd), Barnette and Pace [5] presented a new algorithm based on an implicit calculation of gcd. Frumkin [15] showed that none of these algorithms is polynomial. Kannan and Bachem [16] introduced a polynomial algorithm for computing the Hermite and Smith normal forms of an integer matrix, and later Chou and Collins [9] used Kannan and Bachem's ideas to present a polynomial algorithm named *LDSMKB* for solving systems of linear Diophantine equations. They also developed another algorithm based on Rosser's idea [21], the so called *LDSSBR*, and showed numerically that the *LDSSBR* is more successful than *LDSMKB* in controlling the growth of intermediate results. In 1994, Contejean and Devie [12] generalized Fortenbacher's [11] algorithm for solving systems of linear Diophantine equations. Then in 2000, Aardal, Hurkens and Lenstra [3] gave an algorithm for solving systems of linear Diophantine equations with lower and upper bounds on variables. In 2001, Esmaeili, Mahdavi-Amiri and Spedicato [13] presented a class of algorithms for solving systems of linear Diophantine equations based on the *ABS* algorithms [1, 2] (the so called *EMAS* algorithms).

The main difficulty in solving systems of linear Diophantine equations is the rapid growth of intermediate results, called intermediate expression swell. One successful algorithm controlling this growth is the *LDSSBR*. The generalized Rosser's algorithm (*GRA*) to be described later can be considered as a new formulation of the *LDSSBR*, that is, it gives the same results as the *LDSSBR* does. Here, we also make use of the new formulation to propose a more efficient implementation of Rosser's approach (called the *modified GRA*) as compared to the *LDSSBR*. The generalized algorithm has this property that in the i–th iteration a particular solution of the first $i-1$ equations, and a null space generator of the first $i - 1$ rows of the coefficient matrix are at hand. The general solution (if any) of the first i equations, or redundancy of the i–th equation, or inconsistency of the first i equations is determined in the i–th iteration.

In the remainder of Section 1, we describe Rosser's algorithm (RA) and its generalization, the GRA. We will show in Section 2 the equivalence of the $LDSSBR$ of Chou and Collins with the GRA and show how to make use of the GRA to provide a more efficient implementation of Rosser's approach, compared with the $LDSSBR$. In Section 3, we explain that the new formulation can be used to propose a new class of algorithms for solving linear Diophantine systems and to present an efficient algorithm for solving rank one perturbed linear Diophantine systems, based on the $LDSSBR$. Conclusions are given in Section 4.

1.1. **Rosser's algorithm (RA)** [21].

Let $A = (a_1, \cdots, a_m)^T$, $a_j \in Z^n$, $1 \le j \le m$. Denote the range of A by $R(A)$ and the null space of A by $N(A)$.

Consider the following system of linear Diophantine equations:

$$Ax = b, \qquad A \in Z^{m \times n}, \ x \in Z^n, \ b \in Z^m. \qquad (1.1.1)$$

System (1.1.1) can be written as:

$$a_i^T x = b_i, \quad 1 \le i \le m,$$

where a_i^T is the i–th row of the coefficient matrix A, and b_i is the i–th component of the right hand side vector b. By the integer null space of A we mean the subspace containing integer vectors of the null space. By the null space of the first i rows of A or the first i equations of (1.1.1) we mean the null space of $(a_1, \cdots, a_i)^T$. We say that an integer matrix N spans the integer null space of an integer matrix A, if the space generated by integer combinations of the columns of N is the integer null space of A. A vector $x_p \in Z^n$ satisfying $a_i^T x_p = b_i$, $1 \le i \le m$, is called a particular solution of (1.1.1). Let $v \in Z^n$, $N \in Z^{n \times n_1}$, $y \in Z^{n_1}$, $n_1 \in Z$. Note that if v is a particular solution of (1.1.1) and N spans the integer null space of A then

$$x = v + Ny, \qquad (1.1.2)$$

is the general solution of (1.1.1). This means that x_p is a particular solution of (1.1.1) if and only if $x_p = v + Ny_p$, for some $y_p \in Z^{n_1}$. If the columns of N are linearly independent then N is called a basis for the integer null space of A. Let $v = (v_1, \cdots, v_n)^T \in Z^n$. Throughout the paper, by $gcd(v^T)$ we mean the greatest common divisor of the elements of v.

Now consider the following single linear Diophantine equation:

$$a_1^T x = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \quad a_1, x \in Z^n, \quad b_1 \in Z.$$
(1.1.3)

Using the same notations as in [9], let I denote the $n \times n$ identity matrix and

$$C = \begin{bmatrix} a_1^T \\ I \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} -b_1 \\ 0 \end{bmatrix} = \begin{bmatrix} -b_1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Assume $a_1 \neq 0$. Let c_{ij} be the element in the i–th row and j–th column of C, C_j be the j–th column of C and B_j be the j–th component of B. Moreover, let a and b be two integer numbers. We say a divides b, and write $a \mid b$ if and only if $b = aq$ for some $q \in Z$. Below, $\lfloor x \rfloor$ is the greatest integer number less than or equal to x. Rosser's algorithm (RA) for solving (1.1.3) follows next.

Algorithm 1. *Rosser's Algorithm (RA).*

    Step 1 :: *For $j = 1, \cdots, n$ **if** the leading component of C_j is negative **then** replace C_j by $-C_j$. Sort $C_1, C_2, \cdots, C_n$ in nonincreasing order with respect to their first component.*

    Step 2 :: ***While** $c_{12} \neq 0$ **do** ($B \leftarrow B - \lfloor \frac{B_1}{c_{11}} \rfloor C_1, C_1 \leftarrow C_1 - \lfloor \frac{c_{11}}{c_{12}} \rfloor C_2$. Sort the columns $C_1, C_2, \cdots, C_n$ in nonincreasing order with respect to their first component.)*

    Step 3 :: *Set $B \leftarrow B - \lfloor \frac{B_1}{c_{11}} \rfloor C_1$. At this point the matrix C and the vector B have the forms:*

$$C = \begin{bmatrix} \delta & 0 \\ p & U \end{bmatrix} = \begin{bmatrix} \delta & 0 & \cdots & 0 \\ p & u_1 & \cdots & u_{n-1} \end{bmatrix}, \qquad B = \begin{bmatrix} \theta \\ v \end{bmatrix},$$

*where v, p and $u_i \in Z^n$, $1 \leq i \leq n - 1$, and $\delta = gcd(a_1^T)$. **If** $\theta \neq 0$ **then** (1.1.3) has no integer solution **else** the general integer solution of (1.1.3) is:*

$$x = v + y_1u_1 + y_2u_2 + \cdots + y_{n-1}u_{n-1} = v + Uy, \qquad y \in Z^{n-1}.$$
(1.1.4)

Remark 1.1.1. It can be shown that $\theta = 0$ if and only if $\delta \mid b_1$. The matrix $[p, U]$ is a unimodular matrix (an integer matrix with determinant equal to $+1$ or -1), $\theta = a_1^T v - b_1$, $a_1^T p = \delta$ and $a_1^T U = 0$. Thus, U is an integer basis for the integer null space of a_1^T, and therefore the expression

$$x = (b_1/\delta)p + Uy, \qquad y \in Z^{n-1}, \qquad (1.1.5)$$

can also be considered as the general integer solution of (1.1.3), but the particular solution v in (1.1.4) usually contains components having smaller numbers of digits.

Remark 1.1.2. If $b_1 = 0$ then we will have $B_1 = 0$ at the start. Since upon the execution of RA, B remains unchanged, then in this case the particular solution obtained by RA is the zero vector.

Lemma 1.1.3. *Let $N \in Z^{n \times n}$ and $v \in Z^n$ be arbitrary integer matrix and vector, respectively. Let the matrix and vector obtained after application of Rosser's algorithm (RA) to*

$$\widetilde{C} = \left[\begin{array}{c} a_1^T \\ I \end{array}\right], \quad \widetilde{B} = \left[\begin{array}{c} -b_1 \\ 0 \end{array}\right],$$

be

$$\widetilde{C}_{fin} = \left[\begin{array}{cc} \delta_1 & 0 \\ p^{(1)} & U^{(1)} \end{array}\right], \quad \widetilde{B}_{fin} = \left[\begin{array}{c} \theta_1 \\ v^{(1)} \end{array}\right].$$

Then the same application of RA to the matrix and vector

$$\widehat{C} = \left[\begin{array}{c} A \\ N \end{array}\right], \quad \widehat{B} = \left[\begin{array}{c} -b \\ v \end{array}\right], \quad v \in Z^n,$$

results in:

$$\widehat{C}_{fin} = \left[\begin{array}{c} A \\ N \end{array}\right]\left[\begin{array}{cc} p^{(1)} & U^{(1)} \end{array}\right] = \left[\begin{array}{cc} \delta_1 & 0 \\ a_2^T p^{(1)} & a_2^T U^{(1)} \\ \vdots & \vdots \\ a_m^T p^{(1)} & a_m^T U^{(1)} \\ N p^{(1)} & N U^{(1)} \end{array}\right]$$

$$\widehat{B}_{fin} = \left[\begin{array}{c} A \\ N \end{array}\right] v^{(1)} + \left[\begin{array}{c} -b \\ v \end{array}\right] = \left[\begin{array}{c} \theta_1 \\ a_2^T v^{(1)} - b_2 \\ \vdots \\ a_m^T v^{(1)} - b_m \\ N v^{(1)} + v \end{array}\right].$$

Proof. Let $C \in Z^{n_1 \times n}$ be any matrix with the first row a_1^T, and $B \in Z^{n_1}$ be any vector with the first component B_1. For simplicity, assume that C is so that Step 1 of RA does not change the columns of C. Let K be the number of required iterations for performing Step 2 of RA. In the first iteration of Step 2 of RA we replace B with $B + \alpha_1 C e_1 = B + C(\alpha_1 e_1) = B + C v_1$, where $\alpha_1 = -\lfloor \frac{B_1}{c_{11}} \rfloor$, $v_1 = \alpha_1 e_1$. Replacement of the first column of C with $C_1 + \beta_1 C_2$, where $\beta_1 = -\lfloor \frac{c_{11}}{c_{12}} \rfloor$, and sorting the columns of C with respect to the first components of columns of C are equivalent to multiplying C by a matrix U_1. Repeating this procedure K times, after execution of iteration K we obtain,

$$B_{fin} = B + C v_1 + \cdots + C U_1 U_2 \cdots U_{K-1} v_K$$

as the vector B, and

$$C_{fin} = C U_1 U_2 \cdots U_K$$

as the matrix C. Let $v = v_1 + U_1 v_2 + \cdots + U_1 \cdots U_{K-1} v_k$ and $U = U_1 U_2 \cdots U_K$. Then, after an application of RA to the original C and B, the final matrix and vector are:

$$C_{fin} = CU, \qquad B_{fin} = B + Cv.$$

Specifically, for $C = \widetilde{C}$ and $B = \widetilde{B}$ we have,

$$\widetilde{C}_{fin} = \widetilde{C}U = \begin{bmatrix} a_1^T U \\ U \end{bmatrix} = \begin{bmatrix} a_1^T p^{(1)} & 0 \\ p^{(1)} & U^{(1)} \end{bmatrix} \Rightarrow U = [p^{(1)}, U^{(1)}]$$

$$\widetilde{B}_{fin} = \widetilde{B} + Cv = \begin{bmatrix} -b_1 + a_1^T v \\ v \end{bmatrix} = \begin{bmatrix} \theta_1 \\ v^{(1)} \end{bmatrix} \Rightarrow v = v^{(1)}, \quad \theta_1 = a_1^T v - b_1.$$

We also have $a_1^T p^{(1)} = \delta_1$ and $a_1^T U^{(1)} = 0$. Thus, an application of RA to $\widehat{C}$ and $\widehat{B}$ yields:

$$\widehat{C}_{fin} = \widehat{C}[p^{(1)}, U^{(1)}] = \begin{bmatrix} A \\ N \end{bmatrix} \begin{bmatrix} p^{(1)} & U^{(1)} \end{bmatrix} = \begin{bmatrix} \delta_1 & 0 \\ a_2^T p^{(1)} & a_2^T U^{(1)} \\ \vdots & \vdots \\ a_m^T p^{(1)} & a_m^T U^{(1)} \\ N p^{(1)} & N U^{(1)} \end{bmatrix},$$

$$\widehat{B}_{fin} = \widehat{B} + \widehat{C}v = \left[\begin{array}{c} A \\ N \end{array} \right] v^{(1)} + \left[\begin{array}{c} -b \\ v \end{array} \right] = \left[\begin{array}{c} a_1^T v^{(1)} - b_1 = \theta_1 \\ a_2^T v^{(1)} - b_2 \\ \vdots \\ a_m^T v^{(1)} - b_m \\ Nv^{(1)} + v \end{array} \right].$$

This completes the proof. □

1.2. Generalized Rosser's Algorithm (GRA).

We present a generalization of Rosser's algorithm for solving a single linear Diophantine equation to an algorithm for solving (1.1.1). We will do this by induction. But before doing so, we need the following two lemmas to state the main result characterizing the general solution of the linear Diophantine system (1.1.1).

Lemma 1.2.1. *Suppose that the general integer solution of the first i, $1 \leq i < m$, equations of (1.1.1) is:*

$$x^{(i)} = v_i + N_i y, \qquad v_i \in Z^n, \ N_i \in Z^{n \times n_1}, \ y \in Z^{n_1}, \ n_1 \in Z.$$

If the single linear Diophantine equation,

$$a_{i+1}^T N_i y = b_{i+1} - a_{i+1}^T v_i, \tag{1.2.1}$$

has no integer solution, then the system (1.1.1) has no integer solution.

Proof. Suppose that (1.1.1) has an integer solution x_p. Since x_p is also an integer solution of the first i equations, then there exists a vector $y_p \in Z^{n_1}$ such that

$$x_p = v_i + N_i y_p.$$

Since x_p satisfies the *(i+1)–th* equation as well, then we must have,

$$a_{i+1}^T x_p = a_{i+1}^T v_i + a_{i+1}^T N_i y_p = b_{i+1}$$

$$\Rightarrow a_{i+1}^T N_i y_p = b_{i+1} - a_{i+1}^T v_i,$$

contradicting the inconsistency of (1.2.1). □

Lemma 1.2.2. *Let the columns of N_i span the null space of a set of vectors $w_1^T, \cdots, w_i^T$ and w be any nonzero vector. Then $w^T N_i = 0$ if and only if w is linearly dependent on $w_1, \cdots, w_i$.*

Proof. Let $W_i = (w_1, \cdots, w_i)$. By definition, w is linearly dependent on $w_1, \cdots, w_i$ if and only if $w \in R(W_i)$. Since $R(W_i)$ is the orthogonal complement of $N(W_i^T)$ and by assumption we have $N(W_i^T) = R(N_i)$, then we conclude that $w \in R(W_i)$ if and only if w is in the orthogonal complement of $R(N_i)$. So $w \in R(W_i)$ if and only if $w^T N_i = 0$. Hence w is linearly dependent on $w_1, \cdots, w_i$ if and only if $w^T N_i = 0$. $\square$

The following corollary is now immediate.

Corollary 1.2.3. *Let $W_i = (w_1, \cdots, w_i)$, the columns of N_i span the null space of W_i^T and w be any nonzero vector. Then $w^T N_i \neq 0$ if and only if w is linearly independent of $w_1, \cdots, w_i$.*

We can now proceed with generalization of Rosser's algorithm. In the beginning, we solve the first equation of the system (1.1.1), that is $a_1^T x = b_1$, using RA, and decide the general solution of the first equation as follows:

$$x^{(1)} = v^{(1)} + U^{(1)} y^{(1)} \quad , v^{(1)} \in Z^n, U^{(1)} \in Z^{n \times (n-1)}, \ y^{(1)} \in Z^{n-1},$$
$$(1.2.2)$$

where $a_1^T v^{(1)} = b_1$, $a_1^T U^{(1)} = 0$, and the columns of the matrix $U^{(1)}$ are linearly independent. Note that if the first equation fails to have any integer solution then we conclude that the system has no integer solution. Now assume that the first equation has an integer solution. For $x^{(2)}$ to be an integer solution for the first two equations of the system, $x^{(2)}$ needs to be an integer solution of the first equation and hence there exists a vector $y^{(1)} \in Z^{n-1}$, such that

$$x^{(2)} = v^{(1)} + U^{(1)} y^{(1)}. \qquad (1.2.3)$$

For $x^{(2)}$ to be an integer solution of the second equation, we must have

$$a_2^T x^{(2)} = a_2^T v^{(1)} + a_2^T U^{(1)} y^{(1)} = b_2$$

$$\Rightarrow a_2^T U^{(1)} y^{(1)} = b_2 - a_2^T v^{(1)}. \qquad (1.2.4)$$

If (1.2.4) does not have an integer solution, then by Lemma 1.2.1 we conclude that the system (1.1.1) has no integer solution. If $a_2^T U^{(1)} = 0$ and $b_2 - a_2^T v^{(1)} \neq 0$, then we conclude that (1.2.4) lacks an integer solution and hence the system has no integer solution. If, however, $a_2^T U^{(1)} = 0$ and $b_2 - a_2^T v^{(1)} = 0$, then since every integer solution of the first equation satisfies the second equation, then the second equation

is redundant and (1.2.2) is the general integer solution of the first two equations of (1.1.1). If $a_2^T U^{(1)} \neq 0$ and (1.2.4) has an integer solution, then the general solution of (1.2.4) can be obtained by use of RA giving,

$$y^{(1)} = v^{(2)} + U^{(2)} y^{(2)} \quad , v^{(2)} \in Z^{n-1}, U^{(2)} \in Z^{(n-1) \times (n-2)}, \ y^{(2)} \in Z^{n-2}.$$
$$(1.2.5)$$

Substituting (1.2.5) in (1.2.3) we obtain,

$$x^{(2)} = v^{(1)} + U^{(1)} v^{(2)} + U^{(1)} U^{(2)} y^{(2)}, \quad y^{(2)} \in Z^{n-2}.$$
$$(1.2.6)$$

Now, according to properties of RA we have,

$$a_1^T (v^{(1)} + U^{(1)} v^{(2)}) = a_1^T v^{(1)} = b_1$$
$$a_2^T (v^{(1)} + U^{(1)} v^{(2)}) = a_2^T v^{(1)} + b_2 - a_2^T v^{(1)} = b_2$$
$$a_i^T U^{(1)} U^{(2)} = 0, \quad i = 1, 2.$$

Therefore, if $x^{(2)} = v^{(1)} + U^{(1)} v^{(2)} + U^{(1)} U^{(2)} y^{(2)}$, for some $y^{(2)} \in Z^{n-2}$, then we have,

$$a_1^T x^{(2)} = a_1^T (v^{(1)} + U^{(1)} v^{(2)} + U^{(1)} U^{(2)} y^{(2)}) = a_1^T v^{(1)} = b_1,$$

$$a_2^T x^{(2)} = a_2^T (v^{(1)} + U^{(1)} v^{(2)} + U^{(1)} U^{(2)} y^{(2)}) = a_2^T (v^{(1)} + U^{(1)} v^{(2)})$$
$$= a_2^T v^{(1)} + b_2 - a_2^T v^{(1)} = b_2.$$

So $x^{(2)}$ is a particular solution of the first two equations if and only if $x^{(2)} = v^{(1)} + U^{(1)} v^{(2)} + U^{(1)} U^{(2)} y^{(2)}$, for some $y^{(2)} \in Z^{n-2}$. Thus, (1.2.6) is the general solution of the first two equations of the system (1.1.1). Since the columns of matrices $U^{(1)}$ and $U^{(2)}$ are linearly independent, we conclude that, rank of $U^{(1)} U^{(2)}$ is equal to $n-2$, the column rank of $U^{(2)}$, and hence the columns of $U^{(1)} U^{(2)}$ are also linearly independent. Furthermore, the dimension of the integer null space of the first two rows of A is $n-2$ which is equal to the number of columns of $U^{(1)} U^{(2)}$, and so $U^{(1)} U^{(2)}$ forms a basis for the integer null space of the first two equations.

We can now proceed by induction. For notational simplicity, let

$$N^{(0)} = I \ , N^{(i)} = U^{(1)} \cdots U^{(i)} = N^{(i-1)} U^{(i)}, \quad 1 \leq i \leq m \quad (1.2.7)$$
$$s^{(0)} = 0 \ , s^{(i)} = s^{(i-1)} + N^{(i-1)} v^{(i)}, \quad 1 \leq i \leq m, \quad (1.2.8)$$

where $U^{(1)}$ is a basis for the integer null space of a_1^T and $v^{(1)}$ is particular solution of $a_1^T x = b_1$, both obtained by an application of RA to $a_1^T x = b_1$,

$U^{(i+1)}$ is the basis for the integer null space of $a_{i+1}^T N^{(i)}$, and $v^{(i+1)}$ is the particular solution of the single Diophantine equation,

$$a_{i+1}^T N^{(i)} y^{(i)} = b_{i+1} - a_{i+1}^T s^{(i)},$$

both obtained by an application of RA. We now state and prove the main result, characterizing the general solution of the Diophantine system (1.1.1).

Theorem 1.2.4. *Let $N^{(i)}$ and $s^{(i)}$ be given by (1.2.7) and (1.2.8), respectively, and suppose that A has full row rank and the Diophantine system (1.1.1) has an integer solution. Then $s^{(i)}$ is a particular solution and $N^{(i)}$ is a basis for the integer null space of the first i equations.*

Proof. We will prove the theorem by induction. The theorem is trivially true for $i = 1$, by properties of Rosser's algorithm for a single equation. Now suppose that the theorem is true for $i = k$, that is, $s^{(k)}$ is a particular solution and $N^{(k)}$ is a basis for the integer null space of the first k equations. Then, for every j, $1 \le j \le k$, we have,

$$a_j^T s^{(k)} = b_j \tag{1.2.9}$$

$$a_j^T N^{(k)} = 0. \tag{1.2.10}$$

Consider the case $i = k + 1$. If $x^{(k+1)}$ is an integer solution of the first $k + 1$ equations (and hence the first k equations as well), then there exists $y^{(k)} \in Z^{n-k}$ such that

$$x^{(k+1)} = s^{(k)} + N^{(k)} y^{(k)}. \tag{1.2.11}$$

Since $x^{(k+1)}$ satisfies the *(k+1)–th* equation of the system, then we must have:

$$b_{k+1} = a_{k+1}^T x^{(k+1)} = a_{k+1}^T s^{(k)} + a_{k+1}^T N^{(k)} y^{(k)}.$$

This implies:

$$a_{k+1}^T N^{(k)} y^{(k)} = b_{k+1} - a_{k+1}^T s^{(k)}. \tag{1.2.12}$$

Note that (1.2.12) has integer solutions, because if (1.2.12) does not have any integer solution then by Lemma 1.2.1 we conclude that (1.1.1) has no integer solution, contradicting the hypothesis of the theorem. Since

we assumed that A has full row rank, then by Corollary 1.2.3 we have $a_{k+1}^T N^{(k)} \neq 0$. So an application of RA to (1.2.12) gives

$$y^{(k)} = v^{(k+1)} + U^{(k+1)} y^{(k+1)} \quad , y^{(k+1)} \in Z^{n-k-1}$$

(1.2.13)

as the general solution of (1.2.12). Indeed, we have,

$$a_{k+1}^T N^{(k)} v^{(k+1)} = b_{k+1} - a_{k+1}^T s^{(k)}$$

$$a_{k+1}^T N^{(k)} U^{(k+1)} = 0,$$

and the columns of the matrix $U^{(k+1)}$ are linearly independent, as implied by RA. Now, substituting (1.2.13) in (1.2.11) yields:

$$x^{(k+1)} = s^{(k)} + N^{(k)} v^{(k+1)} + N^{(k)} U^{(k+1)} y^{(k+1)}$$

$$\Rightarrow x^{(k+1)} = s^{(k+1)} + N^{(k+1)} y^{(k+1)}.$$

(1.2.14)

So every integer solution of the first $k+1$ equations can be written in the form (1.2.14) for some $y^{(k+1)} \in Z^{n-k-1}$. To complete the proof, we need to show that $s^{(k+1)}$ is a particular solution and $N^{(k+1)}$ is a basis for the integer null space of the first $k+1$ equations. Note that from (1.2.12), (1.2.8), (1.2.9) and (1.2.10) we can write:
$a_j^T s^{(k+1)} = a_j^T s^{(k)} + a_j^T N^{(k)} v^{k+1} = b_j + 0 = b_j \quad , 1 \leq j \leq k$
$a_{k+1}^T s^{(k+1)} = a_{k+1}^T s^{(k)} + a_{k+1}^T N^{(k)} v^{(k+1)} = a_{k+1}^T s^{(k)} + b_{k+1} - a_{k+1}^T s^{(k)}$
$= b_{k+1}.$

Moreover, from (1.2.7) and the induction hypothesis (1.2.10) we have,

$$a_j^T N^{(k+1)} = a_j^T N^{(k)} U^{(k+1)} = 0, \qquad 1 \leq j \leq k,$$

and for $j = k+1$, by properties of RA we also have,

$$a_{k+1}^T N^{(k+1)} = a_{k+1}^T N^{(k)} U^{(k+1)} = 0.$$

So, we have,

$$a_j^T s^{(k+1)} = b_j$$

$$a_j^T N^{(k+1)} = 0$$

(1.2.15)

for every j, $1 \leq j \leq k+1$. It remains to show that $N^{k+1)}$ is a basis for the integer null space of the first $k+1$ rows of A. First, by (1.2.14), every particular solution of the first $k+1$ equations can be written as an integer linear combinations of the columns of $N^{(k+1)}$. Using this fact and considering (1.2.15), $N^{(k+1)}$ spans the integer null space of the first k rows of A. Now, since by the induction hypothesis and properties

of RA, the columns of $N^{(k)}$ and $U^{(k+1)}$ are linearly independent, and $N^{(k+1)} = N^{(k)}U^{(k+1)}$, then the columns of the matrix $N^{(k+1)}$ are also linearly independent. Moreover, the dimension of the integer null space of the first $k + 1$ rows of A is $n - k - 1$, which is equal to the number of columns of $N^{(k+1)}$. Hence, $N^{(k+1)}$ is a basis for the integer null space of the first $k + 1$ rows of A. $\qquad\square$

The following corollary characterizes the general solution of (1.1.1) and it follows Theorem 1.2.4 immediately.

Corollary 1.2.5. *The general solution of the first i equations of the system (1.1.1) is $s^{(i)} + N^{(i)}y^{(i)}$, $y^{(i)} \in Z^{n-i}$. Indeed, the general solution of (1.1.1) is $s^{(m)} + N^{(m)}y^{(m)}$, $y^{(m)} \in Z^{n-m}$.*

Corollary 1.2.6. *For every i, $1 \le i \le m$, $a_i^T N^{(i-1)} \ne 0$ if and only if a_i is linearly independent of $a_1, a_2, \cdots, a_{i-1}$ (equivalently, $a_i^T N^{(i-1)} = 0$ if and only if a_i is linearly dependent on $a_1, a_2, \cdots, a_{i-1}$).*

Proof. Since by Theorem 1.2.4 the columns of $N^{(i-1)}$ form a basis for the integer null space of the first $i - 1$ rows of A, the result follows from Lemma 1.2.2. $\qquad\square$

Remark 1.2.7. If A does not have full row rank and the *(i+1)–th* row of A is redundant, then according to Lemma 1.2.2 we have $a_{i+1}^T N^{(i)} = 0$. Now, if $b_{i+1} - a_{i+1}^T s^{(i)} \ne 0$ then the linear Diophantine equation $a_{i+1}^T N^{(i)}y = b_{i+1} - a_{i+1}^T s^{(i)}$ and hence the linear system (1.1.1) has no integer solution. Otherwise, if $b_{i+1} - a_{i+1}^T s^{(i)} = 0$ then every solution of the first i equations satisfies the *(i+1)–th* equation also, and we conclude that the *(i+1)–th* equation is redundant.

Remark 1.2.8. If for every i, $1 \le i \le m$, the linear Diophantine equation $a_{i+1}^T N^{(i)}y = b_{i+1} - a_{i+1}^T s^{(i)}$ has an integer solution, then the Diophantine system (1.1.1) has a solution. So, if the system (1.1.1) lacks solution, then there exists an i such that $a_{i+1}^T N^{(i)}y = b_{i+1} - a_{i+1}^T s^{(i)}$ has no integer solution.

Remark 1.2.9. Assume that $rank(A) = r$ and the linear system (1.1.1) has an integer solution. Then, after m iterations of RA to (1.1.1), deleting redundant equations, we will obtain,

$$x^{(m)} = s^{(r)} + N^{(r)}y^{(r)}, \quad y^{(r)} \in Z^{n-r}$$

as the general solution of the system.

Based on the above results, we now present a generalization of Rosser's algorithm for solving systems of linear Diophantine equations (1.1.1). In the following algorithm if (1.1.1) has an integer solution, then r denotes the rank of the coefficient matrix A.

Algorithm 2. *Generalized Rosser's Algorithm (GRA) for solving linear Diophantine systems.*

    Step 1:: *Set $i = 1$, $v = 0$, $r = 0$, $U = I_n$.*

    Step 2:: *Compute $s_i = U^T a_i$ and $\tau_i = b_i - a_i^T v$.*

        (a): *If $s_i = 0$ and $\tau_i \neq 0$ then stop {the system has no integer solution}*

        (b): *If $s_i = 0$ and $\tau_i = 0$ {the ith equation is redundant} then go to Step 4.*

        (c): *If $s_i \neq 0$ then apply RA to the linear Diophantine equation, $s_i^T y = b_i - a_i^T v$. If the equation has no integer solution then stop {the system (1.1.1) has no integer solution} else let $v^{(i)}$ be the particular solution and $U^{(i)}$ be the basis for the integer null space of s_i^T both obtained by an application of RA.*

    Step 3:: *Set $v = v + Uv^{(i)}$, $U = UU^{(i)}$ and $r = r + 1$.*

    Step 4:: *If $i = m$ then stop {v is a particular solution for (1.1.1) and U is a basis for the integer null space of A and r is the rank of A} else set $i = i + 1$ and go to Step 2.*

Next, we explain how we can use Lemma 1.1.3, to combine steps 2 and 3 of the GRA, making the algorithm more efficient for large scale linear Diophantine systems. We observe that in part (c) of Step 2 of

Algorithm 2, application of RA makes use of

$$C = \begin{pmatrix} s_i^T \\ I_{n-i+1} \end{pmatrix}, \quad B = \begin{pmatrix} -b_i + a_i^T v \\ 0 \end{pmatrix},$$

where, I_{n-i+1} is the identity matrix in R^{n-i+1}. By properties of RA, the resulting integer matrix and integer vector have the following forms:

$$C_{fin} = \begin{pmatrix} \delta_i & 0 \\ p^{(i)} & U^{(i)} \end{pmatrix}, \quad B_{fin} = \begin{pmatrix} \theta_i \\ v^{(i)} \end{pmatrix}.$$

If $\theta_i \neq 0$ then we conclude that the system $s_i^T y = b_i - a_i^T v$ has no integer solution. Otherwise, $v^{(i)}$ is a particular solution of $s_i^T y = b_i - a_i^T v$, $U^{(i)}$ is the basis for the integer null space of s_i^T and $\delta_i = s_i^T p^{(i)} = gcd(s_i^T)$.

Now, we note that in the ith iteration of the GRA, if we instead apply RA to

$$C = \begin{pmatrix} s_i^T \\ U \end{pmatrix}, \quad B = \begin{pmatrix} -b_i + a_i^T v \\ v \end{pmatrix},$$

then, by Lemma 1.1.3, the resulting matrix and vector have the forms:

$$C_{fin} = \begin{pmatrix} \delta_i & 0 \\ p^{(i)} & UU^{(i)} \end{pmatrix}, \quad B_{fin} = \begin{pmatrix} \theta_i \\ v + Uv^{(i)} \end{pmatrix}.$$

If $\theta_i \neq 0$ then we conclude that the system $s_i^T y = b_i - a_i^T v$ has no integer solution. By the above observation, in Algorithm 2 we may combine steps 2 and 3 of the GRA. We will see that this change results in a more efficient algorithm for large scale systems. We call the resulting algorithm the *modified GRA*.

Algorithm 3. *Modified Generalized Rosser's Algorithm (MGRA) for solving linear Diophantine systems.*

Step 1:: *Set $i = 1$, $v = 0$, $r = 0$, $U = I_n$.*

Step 2:: *Compute $s_i = U^T a_i$ and $\tau_i = b_i - a_i^T v$.*

    (a): If $s_i = 0$ **and** $\tau_i \neq 0$ **then stop** {*the system has no integer solution*}

    (b): If $s_i = 0$ **and** $\tau_i = 0$ {*the ith equation is redundant*} **then go to Step 4.**

    (c): If $s_i \neq 0$ **then** let

$$C = \begin{pmatrix} s_i^T \\ U \end{pmatrix}, \quad B = \begin{pmatrix} -b_i + a_i^T v \\ v \end{pmatrix}.$$

Apply RA to the matrix C and the vector B above. The resulting matrix C_{fin} and the resulting integer vector B_{fin} have the forms,

$$C_{fin} = \begin{pmatrix} \delta & 0 \\ p & N \end{pmatrix}, \quad B_{fin} = \begin{pmatrix} \theta \\ s \end{pmatrix}.$$

If $\theta \neq 0$ **then stop** {the linear Diophantine system $Ax = b$ has no integer solution} **else** set $v = s$, $U = N$ and $r = r+1$.

Step 4:: *If* $i = m$ ***then stop*** {*v is a particular solution for (1.1.1) and U is a basis for the integer null space of A and r is the rank of A*} ***else*** *set* $i = i+1$ *and* ***go to Step 2****.*

In the next section, we show the equivalence of the *GRA* and the *LDSSBR* of Chou and Collins [9] and compare the computational work of the *GRA*, the *modified GRA* and the *LDSSBR*.

2. ***GRA* and *LDSSBR***

2.1. **The *GRA* as a formulation of the *LDSSBR*.** The main difficulty in solving systems of linear Diophantine equations is the rapid growth of intermediate results making some algorithms impractical even for large computers. In 1982, Chou and Collins [9] presented two algorithms for solving systems of linear Diophantine equations, named *LDSMKB* and *LDSSBR*, to control the growth of intermediate results. The basic ideas for the *LDSMKB* algorithm come from Kannan and Bachem [16]. They used these ideas in computing the Smith and Hermite normal forms for a nonsingular integer matrix A and showed that the length of the digits of intermediate results are bounded by a polynomial function of the components of A. The *LDSSBR* (Linear Diophantine System Solver Based on Rosser's idea) is based on Rosser's idea [21], for finding a general solution of a single linear Diophantine equation with smaller norms of the particular solution and the columns of the basis of the integer null space. The numerical experiments in [9] show that in all cases the *LDSSBR* finds smaller solutions than the *LDSMKB* (the norms of particular solutions obtained by *LDSSBR* are smaller than the norms of those obtained by *LDSMKB*). In this section, we show that the *GRA* can be considered as a formulation of the *LDSSBR*, in the sense

that it generates the same particular solution and basis for the integer null space of A as the $LDSSBR$ does. We further show that the new modified GRA algorithm is more efficient than the $LDSSBR$ for large scale problems. Assume that the system (1) has an integer solution and $rank(A) = m$, so that the m rows of A are linearly independent. The $LDSSBR$ for solving (1.1.1), given in [9], can be written as follows.

Algorithm 4. *The LDSSBR.*

  Step 1:: {*Initialize*} *n = Number of Columns of A, m = Number of Rows of A.*
  Step 2:: {*Adjoin identity matrix to A and zero vector to* $-b$}

$$C \leftarrow \left[\begin{array}{c} A \\ I \end{array} \right], \quad B \leftarrow \left[\begin{array}{c} -b \\ 0 \end{array} \right].$$

  Step 3:: {*Make pivot row zero*} **If** *the first row of C is the zero vector* **then go to Step 4 else** *apply RA to the first row of C, computing the new C and B (see Lemma 1.1.3).*
  Step 4:: {*Checking for inconsistency and elimination of the pivot row*} **If** $B_1 \neq 0$ **then stop** {*the system is inconsistent*} **else** *replace C with the matrix obtained by deleting the first row and the first column of C and replace B by the vector obtained by deleting the first component of B and set* $m \leftarrow m - 1$.
  Step 5:: **If** $m > 0$ **then go to Step 3 else** *let* $x^* \leftarrow B$, $N \leftarrow C$, *and* **stop**.

Theorem 2.1.1. *GRA generates the same particular solution and basis for the integer null space of A, as the LDSSBR does.*

Proof. We prove the theorem by induction. In the first iteration of the $LDSSBR$, RA is applied to

$$C = \left[\begin{array}{c} A \\ I \end{array} \right], \quad B = \left[\begin{array}{c} -b \\ 0 \end{array} \right],$$

resulting into the new matrix and vector,

$$
\widehat{C} = \begin{bmatrix} \delta_1 & 0 \\ a_2^T p^{(1)} & a_2^T U^{(1)} \\ \vdots & \vdots \\ a_m^T p^{(1)} & a_m^T U^{(1)} \\ p^{(1)} & U^{(1)} \end{bmatrix}, \quad \widehat{B} = \begin{bmatrix} \theta_1 \\ a_2^T v^{(1)} - b_2 \\ \vdots \\ a_m^T v^{(1)} - b_m \\ v^{(1)} \end{bmatrix},
$$

where $\delta_1 = gcd(a_1^T)$. If $\theta_1 \neq 0$ then the system has no integer solution; otherwise the algorithm does the replacement,

$$
C \leftarrow \begin{bmatrix} a_2^T U^{(1)} \\ \vdots \\ a_m^T U^{(1)} \\ U^{(1)} \end{bmatrix}, \quad B \leftarrow \begin{bmatrix} a_2^T v^{(1)} - b_2 \\ \vdots \\ a_m^T v^{(1)} - b_m \\ v^{(1)} \end{bmatrix}
$$

and then applies RA to the new C and B (we understand that C and B are used symbolically here to be replaced by the smaller matrix and vector, respectively).

We have shown that, for $i = 1$, the matrix C and the vector B of $LDSSBR$ have the forms,

$$
C = \begin{bmatrix} a_{i+1}^T N^{(i)} \\ \vdots \\ a_m^T N^{(i)} \\ N^{(i)} \end{bmatrix}, \quad B = \begin{bmatrix} a_{i+1}^T s^{(i)} - b_{i+1} \\ \vdots \\ a_m^T s^{(i)} - b_m \\ s^{(i)} \end{bmatrix} \tag{2.1.1}
$$

with $N^{(i)}$ and $s^{(i)}$ as defined by (1.2.7) and (1.2.8). Now, suppose that for $k \geq 1$, after execution of iteration $i = k$, the matrix C and the vector B are given by (2.1.1) and consider the case $i = k + 1$. According to Lemma 1.1.3, after step $k + 1$ of $LDSSBR$ we have,

$$
C = \begin{bmatrix} \delta_{k+1} & 0 \\ a_{k+2}^T N^{(k)} p^{(k+1)} & a_{k+2}^T N^{(k)} U^{(k+1)} \\ \vdots & \vdots \\ a_m^T N^{(k)} p^{(k+1)} & a_m^T N^{(k)} U^{(k+1)} \\ N^{(k)} p^{(k+1)} & N^{(k)} U^{(k+1)} \end{bmatrix}
$$

$$= \begin{bmatrix} \delta_{k+1} & 0 \\ a_{k+2}^T N^{(k)} p^{(k+1)} & a_{k+2}^T N^{(k+1)} \\ \vdots & \vdots \\ a_m^T N^{(k)} p^{(k+1)} & a_m^T N^{(k+1)} \\ N^{(k)} p^{(k+1)} & N^{(k+1)} \end{bmatrix},$$

$$B = \begin{bmatrix} \theta_{k+1} \\ a_{k+2}^T s^{(k)} + a_{k+2}^T N^{(k)} v^{(k+1)} - b_{k+2} \\ \vdots \\ a_m^T s^{(k)} + a_m^T N^{(k)} v^{(k+1)} - b_m \\ s^{(k)} + N^{(k)} v^{(k+1)} \end{bmatrix} = \begin{bmatrix} \theta_{k+1} \\ a_{k+2}^T s^{(k+1)} - b_{k+2} \\ \vdots \\ a_m^T s^{(k+1)} - b_m \\ s^{(k+1)} \end{bmatrix},$$

where $\delta_{k+1} = gcd(a_{k+1}^T N^{(k)})$; that is, δ_{k+1} is the greatest common divisor of the components of the integer vector $a_{k+1}^T N^{(k)}$. If $\theta_{k+1} \neq 0$ then the system has no integer solution. Otherwise, by deleting the first row and the first column of C and the first component of B we obtain:

$$C = \begin{bmatrix} a_{k+2}^T N^{(k+1)} \\ \vdots \\ a_m^T N^{(k+1)} \\ N^{(k+1)} \end{bmatrix}, \quad B = \begin{bmatrix} a_{k+2}^T s^{(k+1)} - b_{k+2} \\ \vdots \\ a_m^T s^{(k+1)} - b_m \\ s^{(k+1)} \end{bmatrix}.$$

This proves the theorem for $i = k + 1$. Thus, for $i = m$ we have,

$$C = N^{(m)}, \qquad B = s^{(m)},$$

as the basis for the integer null space of A and a particular solution of the system (1.1.1), respectively. Indeed, the general solution of (1.1.1) is:

$$x = s^{(m)} + N^{(m)} y^{(m)}, \quad y^{(m)} \in Z^{(n-m)},$$

completing the proof of the Theorem. $\qquad \qquad \square$

2.2. **Computational comparisons.** In Theorem 2.1.1, we stated that the *GRA* and the *modified GRA* and the *LDSSBR* give the same results. We now compare these approaches with respect to their computational work. We assume that $rank(A) = m$ and the linear Diophantine system (1.1.1) has integer solutions.

In the *(k+1)–th* iteration, for the *LDSSBR*, as discussed in the proof of Theorem 2.1.1, we apply *RA* to

$$
C = \begin{bmatrix} a_{k+1}^T N^{(k)} \\ \vdots \\ a_m^T N^{(k)} \\ N^{(k)} \end{bmatrix}, \quad B = \begin{bmatrix} a_{k+1}^T s^{(k)} - b_{k+1} \\ \vdots \\ a_m^T s^{(k)} - b_m \\ s^{(k)} \end{bmatrix},
$$

and for the *GRA*, in part (c) of Step 2, we need to solve (1.2.12) and thus apply *RA* to

$$
C = \begin{pmatrix} a_{k+1}^T N^{(k)} \\ I_{n-i+1} \end{pmatrix}, \quad B = \begin{pmatrix} a_{k+1}^T s^{(k)} - b_{k+1} \\ 0 \end{pmatrix},
$$

and for the *modified GRA*, knowing that $U = N^{(k)}$ and $v = s^{(k)}$, we apply *RA* to

$$
C = \begin{bmatrix} a_{k+1}^T N^{(k)} \\ N^{(k)} \end{bmatrix}, \quad B = \begin{bmatrix} a_{k+1}^T s^{(k)} - b_{k+1} \\ s^{(k)} \end{bmatrix}.
$$

Therefore, in the *(k+1)–th* iteration of *GRA* and *modified GRA* we save by not needing the application of *RA* on $a_j^T N^{(k)}$ and $a_j^T s^{(k)} - b_j$, $k + 2 \leq j \leq m$. In return, in the *(k+1)–th* iteration of the *GRA* we compute $N^{(k+1)} = N^{(k)} U^{(k+1)}$, $s^{(k+1)} = s^{(k)} + N^{(k)} v^{(k+1)}$, $a_{k+1}^T N^{(k)}$, $b_{k+1} - a_{k+1}^T s^{(k)}$ and in the *(k+1)–th* iteration of the *modified GRA* we compute $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$. The difference between the *GRA* and the *modified GRA* is that, in the *(k+1)–th* iteration of the *GRA*, the computations of *RA* are done on vectors in Z^{n-k+1}, while, in the *(k+1)–th* iteration of the *modified GRA*, the computations of *RA* are done on vectors in Z^{n+1}. In return, in the *(k+1)–th* iteration of the *modified GRA*, we do not need to compute $N^{(k+1)} = N^{(k)} U^{(k+1)}$ and $s^{(k+1)} = s^{(k)} + N^{(k)} v^{(k+1)}$.

So, if for a linear Diophantine system the total number of required computations for computing $N^{(k+1)} = N^{(k)} U^{(k+1)}$, $s^{(k+1)} = s^{(k)} + N^{(k)} v^{(k+1)}$, $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$, $0 \leq k \leq m-1$, is less than the saved amount of computations of *RA* on $a_j^T N^{(k)}$ and $a_j^T s^{(k)} - b_j$, $k+2 \leq j \leq m$, $0 \leq k \leq m-1$, then the *GRA* provides us with a more efficient implementation of Rosser's approach than the *LDSSBR*. Similarly, if for a linear Diophantine system the total number of required computations for computing $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$, $0 \leq k \leq m-1$, is less than the saved amount of computations of *RA* on $a_j^T N^{(k)}$ and

$a_j^T s^{(k)} - b_j$, $k + 2 \leq j \leq m$, $0 \leq k \leq m - 1$, then the *modified GRA* provides us with a more efficient implementation of Rosser's approach than the *LDSSBR*. In the following, we consider the number of multiplications (since this comprises most of the work) for comparing the computational work of the *GRA*, the *modified GRA* and the *LDSSBR*.

Let t_{k+1} be the number of required iterations for performing Step 2 of *RA*, including one additional iteration at the beginning of Step 3. Note that, since the number of iterations of Step 2 of *RA* only depends on the first row of C and the first element of B, then in the *GRA* and in the *modified GRA*, the values of $t_1, \cdots, t_m$ are the same as those of the *LDSSBR*. For the *LDSSBR*, in every iteration of Step 2 of *RA* we multiply a vector in Z^{n+m-k} by a scalar and subtract the resulting vector from another vector in Z^{n+m-k}. By the above observation, in the *(k+1)–th* iteration of the *LDSSBR* we need to perform $(n + m - k)t_{k+1}$ multiplications. Therefore, solving the linear Diophantine system (1.1.1) by the *LDSSBR* needs

$$\sum_{k=0}^{m-1}(n + m - k)t_{k+1} = \sum_{k=0}^{m-1}(n - k + 1)t_{k+1} + (m - 1)\sum_{k=0}^{m-1} t_{k+1}$$
$$= (n + 1)\sum_{k=0}^{m-1} t_{k+1} + \sum_{k=0}^{m-1}(m - k - 1)t_{k+1}$$

$$(2.2.1)$$

multiplications.

On the other hand, in the *(k+1)–th* iteration of *GRA*, as discussed in the proof of Theorem 1.2.4, we first compute $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$ and then apply *RA* to

$$C = \begin{pmatrix} a_{k+1}^T N^{(k)} \\ I_{n-i+1} \end{pmatrix}, \quad B = \begin{pmatrix} a_{k+1}^T s^{(k)} - b_{k+1} \\ 0 \end{pmatrix}.$$

Computing $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$ needs $n(n-k)$ and n multiplications, respectively, and since in every iteration of Step 2 we multiply a vector in Z^{n-k+1} by a scalar and subtract the resulting vector from another vector in Z^{n-k+1}, then, applying *RA* needs $(n - k + 1)t_{k+1}$ multiplications. After the application of *RA*, we obtain $U^{(k+1)}$ and $v^{(k+1)}$. Then, in Step 3 of the *GRA* we compute $N^{(k+1)} = N^{(k)}U^{(k+1)}$ and $s^{(k+1)} = s^{(k)} + N^{(k)}v^{(k+1)}$, by performing $n(n-k)(n-k-1)$ and $n(n-k)$ multiplications, respectively. Therefore, solving the linear Diophantine

system (1.1.1) by the *GRA* needs

$$\sum_{k=0}^{m-1} (n-k+1)t_{k+1} + \sum_{k=0}^{m-1} \{n + n(n-k)(n-k+1)\} \tag{2.2.2}$$

multiplications.

In the *(k+1)–th* iteration of the *modified GRA*, we first compute $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$ and then apply *RA* to

$$C = \begin{bmatrix} a_{k+1}^T N^{(k)} \\ N^{(k)} \end{bmatrix}, \quad B = \begin{bmatrix} a_{k+1}^T s^{(k)} - b_{k+1} \\ s^{(k)} \end{bmatrix}.$$

Computing $a_{k+1}^T N^{(k)}$ and $b_{k+1} - a_{k+1}^T s^{(k)}$ needs $n(n-k)$ and n multiplications, respectively, and applying *RA* to the integer matrix C and the integer vector B, needs $(n+1)t_{k+1}$ multiplications. Hence, solving the system (1.1.1) by use of the *modified GRA* needs

$$(n+1)\sum_{k=0}^{m-1} t_{k+1} + \sum_{k=0}^{m-1} \{n + n(n-k)\} \tag{2.2.3}$$

multiplications.

From (2.2.1), (2.2.2) and (2.2.3) we conclude that if for a linear Diophantine system we have,

$$(m-1)\sum_{k=0}^{m-1} t_{k+1} > \sum_{k=0}^{m-1} \{n + n(n-k)(n-k+1)\}, \tag{2.2.4}$$

then, the required number of multiplications by the *GRA* is less than that of the *LDSSBR*. Likewise, if for a linear Diophantine system we have,

$$\sum_{k=0}^{m-1} (m-k-1)t_{k+1} > \sum_{k=0}^{m-1} \{n + n(n-k)\}, \tag{2.2.5}$$

then, the required number of multiplications of the *modified GRA* is less than that of the *LDSSBR*. To see whether the inequalities (2.2.4) and (2.2.5) hold, we need to know about the values of t_k, $1 \le k \le m$, that is, the number of iterations of *RA* for solving a linear Diophantine equation. Although Rosser's algorithm is shown to be efficient in practice [9], no analysis of Rosser's algorithm for $n > 2$ have been given to date. Not having any analytical result about the values of t_k, $1 \le k \le m$, we considered numerical experiments to see if the inequalities (2.2.4) and (2.2.5) hold for linear Diophantine systems, in practice. We generated

linear Diophantine systems with random coefficient matrices and right hand sides and computed the values of t_k, $1 \leq k \leq m$, for each consistent system and then checked to see wether the inequalities (2.2.4) and (2.2.5) hold. As the numerical results of tables 1 and 2 show, in all cases, the inequality (2.2.4) does not hold and hence the $LDSSBR$ is more efficient than the GRA. The inequality (2.2.5) does not hold for a few small values of m and n either, but for larger values of m and n the inequality (2.2.5) always hold and hence for large scale linear Diophantine systems, the *modified GRA* is more efficient than the $LDSSBR$. In tables 1 and 2, f_L^1 denotes $(m-1)\sum_{k=0}^{m-1} t_{k+1}$, f_G^1 denotes $\sum_{k=0}^{m-1}\{n+n(n-k)(n-k+1)\}$, f_L^2 denotes $\sum_{k=0}^{m-1}(m-k-1)t_{k+1}$ and f_{mG}^2 denotes $\sum_{k=0}^{m-1}\{n+n(n-k)\}$.

Table 1 : $LDSSBR$ and GRA			
m	n	f_L^1	f_G^1
4	5	168	360
10	15	4392	19500
16	21	19380	73248
30	37	156861	671180
33	42	223968	1100022
45	50	488972	2208750
60	75	1462020	10875000
70	85	2229873	17909500

Table 2 : $LDSSBR$ and $MGRA$			
m	n	f_L^2	f_{mG}^2
4	5	85	90
10	15	1973	1725
16	21	8233	4872
30	37	69504	26085
33	42	98792	37422
45	50	238224	65250
60	75	715887	209250
70	85	1129059	306425

3. Some applications

Here, we discuss some applications of the GRA.

3.1. Integer extended *ABS* algorithms.
The new formulation can be used to show how to modify the class of integer ABS algorithms, the so called $EMAS$ algorithms given in [13], so that the parameters of the new algorithms can be chosen to generate the same solution iterates as the GRA, while having different null space generators. This leads to presenting a new class of integer extended ABS ($IEABS$) algorithms, based on extended ABS algorithms ($EABS$) of the real case [10], to solve systems of linear Diophantine equations, improving upon both the efficiency and the effectiveness of the $EMAS$ algorithms by generating

Abaffians with independent rows and controlling the growth of intermediate results. It can also be shown that both the *EMAS* algorithms and the *GRA* (and hence the *LDSSBR*) belong to the new class of *IEABS* algorithms by specifying the parameters of the *IEABS* algorithms so that both the same solution iterates and the same null space generators for the *EMAS* algorithms and *GRA* are produced. The new *IEABS* class of algorithms, with its new parameters of choice, includes reliable and competitive algorithms for solving systems of linear Diophantine equations; see [18] for details.

3.2. **Rank one perturbed linear Diophantine systems.** After solving a linear Diophantine system by the *LDSSBR*, suppose that it is desired to delete a variable or a constraint. If we know how to make use of the information obtained during the application of the *LDSSBR* to the original system effectively, then we can avoid solving a new linear Diophantine system from scratch that may be so time consuming and costly. In 2006, Amini and Mahdavi-Amiri [4] showed how one could obtain the general solution of a rank one perturbed linear Diophantine system using the information obtained after an application of a member of the *EMAS* class of algorithms. But, *EMAS* algorithms are not shown to be effective in practice. The new formulation can be successively used to propose an efficient algorithm for solving rank one perturbed linear Diophantine systems based on the *LDSSBR* of Chou and Collins. The new algorithm is shown to be efficient in practice; see [17] for details.

4. **Conclusions**

We presented a generalization of Rosser's algorithm for solving a single linear Diophantine equation to an algorithm for solving systems of linear Diophantine equations. Then, we proved that the generalized algorithm presented a formulation of the *LDSSBR* of Chou and Collins, and showed how we could use the formulation to provide a more efficient implementation, the *modified GRA* of Rosser's approach. Comparative computational results showed that the *modified GRA* provided a more efficient implementation, specially for large scale problems. Finally, we

pointed out some applications of the new formulation, developed to improve upon the recently proposed integer *ABS* (the *EMAS*) algorithms for Diophantine systems and their rank one perturbations.

Acknowledgments

REFERENCES

[1] J. Abaffy, C. G. Broyden and E. Spedicato, A class of direct methods for linear equations, *Numerische Mathematik* **45** (1984) 361-376.

[2] J. Abaffy and E. Spedicato, *ABS* Projection Algorithms: Mathematical Techniques for Linear and Nonlinear Equations, Ellis Horwood, Chichester, 1989.

[3] K. Aardal, C. A. J. Hurkans and A. K. Lenstra, Solving a systems of linear Diophantine equations with lower and upper bounds on the variables, *Mathematics of Operations Research* **3** (2000) 427-442.

[4] K. Amini and N. Mahdavi-Amiri, Solving rank one perturbed linear Diophantine systems by the ABS method, *Optimization Methods and Software* **21** (2006) 819-831.

[5] S. Barnette and I. S. Pace, Efficient algorithms for linear systems calculations: Part I- Smith form and common divisor of polynomial matrices, *Internat. J. of Systems Sci.* **5** (1974) 403-411.

[6] W. A. Blankinship, Algorithm 287, matrix triangulation with integer arithmetic [F1], *Comm. ACM* **9** (1966) 513.

[7] W. A. Blankinship, Algorithm 288, solution of simultaneous linear Diophantine equations, *Comm. ACM* **9** (1966) 514.

[8] G. H. Bradley, Algorithms for Hermite and Smith normal matrices and linear Diophantine equations, *Math. Comp.* **25** (1971) 897-907.

[9] T. J. Chou and E. E. Collins, Algorithms for the solutions of systems of linear Diophantine equations, *SIAM J. Comput.* **11** (1982) 686-708.

[10] Z. Chen, N. G. Deng and Y. Xue, A general algorithm for underdetermined linear systems, *The Proceedings of the first International Conference on ABS Algorithms*, Luoyang, China, September 2-6, (1991) 1-13.

[11] M. Clausen and A. Fortenbacher, Efficient solution of Diophantine equations, *Journal of Symbolic Computation* **8(1&2)** (1989) 201-216.

[12] E. Contejean and H. Devie, An efficient incremental algorithm for solving systems of Diophantine equations, *Information and Computation* **113(1)** (1994) 143-172.

[13] H. Esmaeili , N. Mahdavi-Amiri and E. Spedicato, A class of ABS algorithms for Diophantine linear systems, *Numeriche Mathematik* **90** (2001) 101-115.

[14] P. Erdos and R. L. Graham, On a linear Diophantine problem of Frobenius, *Acta Arith.* **21** (1972) 399-408.

[15] M. A. Frumkin, Polynomial Time Algorithms in the Theory of Linear Diophantine Equations, M. Karpinski, (ed.), Fundamentals of Computation Theory, Lecture Notes in Computer Science, Vol. 56, Springer, New York, 1977, pp. 386-392.

[16] R. Kannan and A. Bachem, Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. Comput.* **8** (1979) 499-507.

[17] M. Khorramizadeh and N. Mahdavi-Amiri, Solving rank one perturbed linear Diophantine systems by use of the *LDSSBR* of Chou and Collins, Submitted.

[18] M. Khorramizadeh and N. Mahdavi-Amiri , Integer extended *ABS* algorithms and possible control of intermediate results for linear Diophantine systems, to appear in *4OR*, DOI 10-1007/s 10288-008-0082-8, 23 pages.

[19] H. W. Lenstra, Jr., Integer programming with a fixed number of variables, *Mathematics of Operations Research* **8** (1983) 538-548.

[20] D. Papp and B. Vizvari, Effective solution of linear Diophantine equation systems with an application in chemistry, *Journal of Mathematical Chemistry* **39** (2005) 15-31.

[21] J. B. Rosser, A note on the linear Diophantine equation, *Amer. Math. Monthly* **48** (1941) 662-666.

[22] A. Wassermann, Attacking the market split problem with lattice point enumeration, *Journal of Combinatorial Optimization* **6** (2002) 5-16.

M. Khorramizadeh

N. Mahdavi-Amiri

Faculty of Mathematical Sciences, Sharif University of Technology, Tehran, Iran.

Email: mkhorrami@math.sharif.edu

Email: nezamm@sharif.edu