Preserving Sparsity for General Solution of Linear Diophantine Systems

Mostafa Khorramizadeh

Department of Mathematical Sciences
Shiraz University of Technology
Shiraz 71555-313, Iran
m.khorrami@sutech.ac.ir

Abstract

Here, we present an efficient algorithm for preserving sparsity in computing the general solution of linear Diophantine systems. In the $k$th iteration of the algorithm, the general solution $k - 1$ equations of the systems is at hand. Then, we present numerical results to justify the efficiency of the resulting algorithm.

Mathematics Subject Classification: 11D04, 65Y04

Keywords: linear Diophantine equations, Sparsity, Greatest common divisor, general solution

1 Introduction

Consider the linear Diophantine system

$$Ax = b, \quad A \in \mathbb{Z}^{m \times n}, \quad x \in \mathbb{Z}^n, \quad b \in \mathbb{Z}^m, \quad m \leq n, \quad \text{rank}(A) = m. \quad (1)$$

By solving (1), we mean finding an integer vector $v \in \mathbb{Z}^n$ satisfying (1) and an integer matrix $N \in \mathbb{Z}^{n \times (n-m)}$ with linearly independent columns so that every $x \in \mathbb{Z}^n$ satisfies (1) if and only if there exists some $y \in \mathbb{Z}^{n-m}$ so that $x = v + Ny$. $v$ is the particular solution and $N$ is a basis for the null space of $A$. Linear Diophantine systems has many applications in engineering and
mathematical sciences such as design of circuits, graph theory, integer rogram-ming anf etc. There are many algorithms for computing the solution of a linear Diophantine system [1, 2, 3, 4]. Many of these algorithms, including those of [2, 4], are generalization of algorithms for solving a single linear Diophantine equation such as the extended greatest common divisor [5] and the Rosser’s algorithm [7] and etc. This paper is concerned with preserving sparsity in computations of linear Diophantine systems. In section 2 we propose an efficient algorithm for computing the general solution of a linear diophantine equation which preserves sparsity. In section 3, we generaliza the proposed algorithm to obtain an algorthm for computing the general solution of a system of linear Diophantine equations which preserves sparsity. In section 4 we justify the efficiency of our proposed algotithm by presenting some numerical results. Finally, section 5 is devoted to concluding remarks.

2 Single linear Diophantine equation

Consider the following single linear Diophantine equation

\[ a_1^T x = a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n = b_1, \quad a_1, x \in \mathbb{Z}^n, \quad b_1 \in \mathbb{Z}. \]  

Let \( v = (v_1, \cdots, v_n)^T \in \mathbb{Z}^n \). Throughout the paper, by \( \text{gcd}(v^T) \) we mean the greatest common divisor of the elements of \( v \). Using the same notations as in [4], let \( I \) denote the \( n \times n \) identity matrix and

\[
C = \begin{bmatrix} a_1^T \\ U \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \quad B = \begin{bmatrix} -b_1 \\ 0 \\ 0 \end{bmatrix}.
\]

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 \( q \in \mathbb{Z} \). Below, \( \lfloor x \rfloor \) is the greatest integer number less than or equal to \( x \). Our proposed algorithm for solving a single linear Diophantine equation is as follows. Despite other algorithms, this algorithm selects the next two sparsest columns to perform the Euclidean algorithm.

**Algorithm 2.1 Sparse linear Diophantine Solver (SDS)**

**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 number of nonzero columns of \( U \) so that \( c_{11} > c_{12} \).
Step 2: While $c_{12} \neq 0$ do (B ← B − $\lfloor B_{1} / c_{12} \rfloor$C₁, C₁ ← C₁ − $\lfloor c_{11} / c_{12} \rfloor$C₂. Sort the columns C₁, C₂, ⋯, Cₙ in nonincreasing order with respect to number of nonzero columns of U so that $c_{11} > c_{12}$.)

Step 3: Set B ← B − $\lfloor B_{1} / c_{11} \rfloor$C₁. At this point the matrix C and the vector B have the forms:

$$C = \begin{bmatrix} \delta & 0 \\ p & U \end{bmatrix}, \quad B = \begin{bmatrix} \theta \\ v \end{bmatrix},$$

where $v$, $p$ and $u_i \in \mathbb{Z}^n$, $1 \leq i \leq n - 1$, and $\delta = \gcd(a_{1T})$. If $\theta \neq 0$ then (2) has no integer solution else the general integer solution of (2) is:

$$x = v + y_1 u_1 + y_2 u_2 + \cdots + y_{n-1} u_{n-1} = v + U y, \quad y \in \mathbb{Z}^{n-1}. \quad (3)$$

3 Sparsity in linear Diophantine systems

In this section we develop an efficient algorithm for computing the general solution of a system of linear Diophantine system (1). Note that we assume that $\text{rank}(A) = m$. Let

$$N^{(0)} = I, \quad N^{(i)} = U^{(1)} \cdots U^{(i)} = N^{(k-1)} U^{(i)}, \quad 1 \leq k \leq m \quad \text{(4)}$$

$$s^{(0)} = 0, \quad s^{(i)} = s^{(i-1)} + N^{(i-1)} v^{(i)}, \quad 1 \leq i \leq m, \quad \text{(5)}$$

where $U^{(1)}$ is a basis for the integer null space of $a_{1T}$ and $v^{(1)}$ is an integer solution of $a_{1T} x = b_1$, both obtained by an application of SDS to $a_{1T} x = b_1$, $U^{(i+1)}$ is the basis for the integer null space of $a_{i+1T} N^{(i)}$, and $v^{(i+1)}$ is the particular integer solution of the single Diophantine equation,

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

both obtained by an application of SDS. Suppose that $A$ has full row rank and the Diophantine system (1) has an integer solution. It can be verified that $s^{(i)}$ is a particular solution and $N^{(i)}$ is a basis for the integer null space of the first $i$ equations, see [6] for the proof. Here, to consider the sparsity issues, in every iteration, among all rows not considered so far, we choose the row with smallest number of nonzeros for computing $s_i$. The resulting algorithm is as follows.

Algorithm 3.1 Sparse linear Diophantine System Solver (SDSS)

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

Step 2: Among all rows of $A$ not considered so far, let $a_i$ be the row with minimum number of nonzero components. 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 $i$th 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 SDS to the matrix $C$ and the vector $B$ above. The resulting matrix $C_{\text{fin}}$ and the resulting integer vector $B_{\text{fin}}$ have the forms,

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

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 3: Set $v = v + U v^{(i)}$, $U = U U^{(i)}$.

Step 4: If $i = m$ then stop \{$v$ is an integer solution for (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.

4 Numerical results

In this section we provide with a numerical example, showing that our proposed algorithm is efficient. We implemented the algorithm in the Matlab environment, using an Intel Core 2 Duo cpu, 1.8 GHz processor of 1GB RAM. We applied the resulting algorithm to a sparse test problem which is made as follows. We first considered a sparse integer matrix which is given by Mark Giesbrecht and can be obtained from sparse matrix collection of the university of Florida. The matrix has 18 rows and 30 columns. Moreover, the density of the given sparse matrix is 0.12. We then let $x$ be an arbitrary sparse integer vector and compute $b = Ax$. This way we are sure that the linear Diophantine system $Ax = b$ has sparse integer solutions. We then applied our implemented algorithm to the resulting linear Diophantine system to obtain a null space basis and a particular solution. We obtained an sparse null space basis with 12 rows and 30 columns and a sparse integer solution. The density of the
resulting null space basis was 0.14 which is close to the density of the original coefficient matrix. To compare our proposed algorithm with an efficient algorithm of the literature we also applied the LLL-based Hermite normal form algorithm [5] to the same linear Diophantine system and obtained a null space basis with sparsity of 0.21. In a similar way the sparsity of the particular solution obtained by using our approach is more than that of the LLL–based Hermite normal form algorithm. This justifies the efficiency of our proposed algorithm.

5 Conclusions

In this paper we proposed an efficient algorithm for computing the general solution of a sparse linear Diophantine system and presented numerical results to justify its efficiency. We also compared our proposed algorithm with an algorithm which is based on LLL–reduction algorithms.

ACKNOWLEDGEMENTS. The authors would like to thank the research council of Shiraz university of technology for its support.

References


Received: June 1, 2013