# Grammar–Based Algorithms for Solving Certain Classes of Linear Diophantine Systems in Non–negative Integers

Dmitri G. Korzoun

Department of Computer Science, University of Petrozavodsk

Lenin St., 33, Petrozavodsk, Republic of Karelia, 185640, Russia

E-mail: dkorzun@cs.karelia.ru

**Abstract**

Searching the Hilbert basis of an arbitrary system of linear Diophantine equations in non-negative integers is more complex in general than problems in the complexity class NP. We consider a special class of such systems—systems associated with context-free grammars, which was introduced in our previous work. Some of its subclasses can be efficiently solved (polynomial or pseudo-polynomial time complexity). The paper presents several new algorithms for searching the Hilbert basis of systems belonging to these subclasses.

**Contents**

# 1 Introduction

Let $\mathbb{Z}$ be a set of integers and $\mathbb{Z}_+$ be a set of non-negative integers. A system of $n$ non-negative linear Diophantine equations (NLDE) in $m$ unknowns can be written as follows:

$$Ax = b\ , \tag{1.1}$$

where $A$ is an integer $(n \times m)$-matrix, $x \in \mathbb{Z}_+^m$ is a vector of unknowns, and $b$ is a vector in $\mathbb{Z}^n$. Solutions of system (1.1) is restricted with non-negative integer vectors that is a reason for the letter "N" to be in the introduced notation "NLDE".

As a rule, a complete solution for system (1.1) means searching its Hilbert basis—a pair $(\mathcal{N}, \mathcal{H})$ of finite sets such that a set $\mathcal{S}$ of all solutions can be represented as (see [1, 2])

$$\mathcal{S} = \mathcal{N} + \mathcal{H}^*,$$

where $\mathcal{H}^*$ is a set of all non-negative linear combinations of elements of $\mathcal{H}$. In other words, any solution $x \in \mathcal{S}$ can be expressed as

$$x = h^{(0)} + \sum_{s=1}^{q} \alpha_s h^{(s)}\ , \qquad h^{(0)} \in \mathcal{N}\ , \quad \mathcal{H} = \left\{h^{(1)}, h^{(2)}, \ldots, h^{(q)}\right\}$$

for arbitrary non-negative integers $\alpha_s$, $s = 1, 2, \ldots, q$. Elements of $\mathcal{N}$ are all minimal[1] solutions of system (1.1). Elements of $\mathcal{H}$ are all minimal solutions of the corresponding homogenous system:

$$Ax = \mathbb{O}\ . \tag{1.2}$$

A problem of finding any solution for system (1.1) is known to be NP-complete [1]. Searching of the whole basis is even more complex, because the cardinality of the basis increases exponentially with $n$, $m$ and $||(A, b)||$, where $||(A, b)||$ is a norm of coefficients. The study of special classes of NLDE systems is therefore concentrated in constructing algorithms dedicated for these particular classes. An interesting case is NLDE systems associated with context-free (CF) grammars. They were firstly introduced by M. Filgueiras and A. Tomás [3] (some particular type of these systems).

[1] A solution $x$ is minimal if there is no other solution $y$ such that $x \geq y$ (the component-wise partial order $x_i \geq y_i$, $i = 1, 2, \ldots, m$). It is easy to see that if $x$ is not minimal then it can be decomposed as $x = y + h$, where $h$ is a non-zero solution of homogenous system (1.2).

In this paper we introduce a class of the associated NLDE system (ANLDE) in a more general form than in [3] and present new algorithms for solving some of its subclasses. These algorithms are characterized with polynomial or pseudo-polynomial time complexity and are significantly more efficient than existing "universal" algorithms for solving arbitrary NLDE systems.

We use the terminology and notation of the formal languages theory similar to monographs of A. Aho and J. Ullman [4], and S. Sippu and E. Soisalon-Soininen [5]. A CF-grammar $G$ is defined as a quadruple $(N, \Sigma, P, S)$, where $N = \{A_1, A_2, \ldots, A_n\}$ is a nonterminal alphabet, $\Sigma = \{a_1, a_2, \ldots, a_t\}$ is a terminal alphabet, $P = \{r_1, r_2, \ldots, r_m\}$ is a set of grammar rules in the form $r_i = (A_k \to p)$ for some nonterminal $A_k \in N$ and string $p \in (N \cup \Sigma)^*$, $S$ is a start symbol. For the study a start symbol is not important and we denote a grammar as $G = (N, \Sigma, P, \cdot)$.

The rest of the paper is organized as follows. In Section 2 we introduce a class of ANLDE system, consider its important subclasses and discuss some properties of these systems related to CF-grammars. Based on these properties we present in Section 3 several new efficient algorithms for searching Hilbert basis of the systems belonging to the subclasses.

# 2 Systems Associated with Context–Free Grammars

In this section we introduce a wide class of NLDE systems. The systems are constructed according to a given CF-grammar and two arbitrary strings over the grammar alphabet. The solutions of these systems are strongly connected with certain derivations in the grammar and this property is intensively used to construct several solving algorithms presented in Section 3. The detailed theoretical background of the discussed issues can be found in [6–9].

## 2.1 Construction

Let $G = (N, \Sigma, P, \cdot)$ be a CF-grammar and strings $w', w'' \in (N \cup \Sigma)^*$. One can construct the following NLDE system:

$$\begin{cases} \sum\limits_{i \in H_A} x_i + \beta_A = \sum\limits_{i=1}^{n} \gamma_{iA}\, x_i + \alpha_A \quad \text{for all nonterminals } A \in N \\ \sum\limits_{i=1}^{n} \gamma_{iA}\, x_i + \alpha_a = \beta_a \quad \text{for all terminals } a \in \Sigma\,, \end{cases} \tag{2.1}$$

where $H_A$ is a set[2] of indices of those grammar rules whose left-hand side is equal to the nonterminal $A$; the values $\alpha_A$ and $\alpha_a$ are the numbers of occurrences in the string $w'$ respectively the nonterminal $A$ and the terminal $a$; the values $\beta_A$ and $\beta_a$ are the same as $\alpha_A$ and $\alpha_a$ but in the string $w''$; the values $\gamma_{iA}$ and $\gamma_{ia}$ are the numbers of occurrences in a right-hand side of the rule $r_i$ respectively $A$ and $a$.

The system has $m+t$ equations—one for each symbol (nonterminal or terminal) and $n$ unknowns—one for each grammar rule. Each unknown $x_i$ is allowed to take non-negative integers only. This system is called an *associated NLDE system* (ANLDE), and the corresponding grammar—a *generative grammar.*

Let a derivation $w' \Rightarrow^+ w''$ exist in $G$. We consider a vector $\xi \in \mathbb{Z}_+^n$ such that each component $\xi_i$ is equal to the number of times the rule $r_i$ is applied during the fixed derivation $w' \Rightarrow^+ w''$.

**Theorem 1** *The vector $\xi$ is a solution of system (2.1).*

This fact[3] is explained as follows. At the beginning of any successful derivation $w' \Rightarrow^+ w''$ there is $\alpha_A$ occurrences of each nonterminal $A$, because we start the derivation with the string $w'$. At the end it is equal to $\beta_A$, because we finish the derivation with the string $w''$. The sum $\sum\limits_{i \in H_A} \xi_i$ is the number of times the nonterminal $A$ is expanded during the derivation, the second sum $\sum\limits_{i=1}^{n} \gamma_{iA}\xi_i$ is the number of times it is produced: each expansion of $A$ with a rule $r_i$ decrements by one the number of occurrences of $A$ in the current sentential form and increments by $\gamma_{iB}$ the number of occurrences of each nonterminal $B \in N$. Thus, there were $\alpha_A$ nonterminals $A$ in $w'$, then $\sum\limits_{i=1}^{n} \gamma_{iA}\xi_i$ occurrences of the nonterminal $A$ had appeared and $\sum\limits_{i \in H_A} \xi_i$ occurrences simultaneously had disappeared, but at the end of the derivations $\beta_A$ occurrences are preserved. This is a reason for $\xi$ to satisfy all nonterminal equations of the system. Similarly, the number of times a terminal occurs in $w''$ is equal to a sum of the number of times it occurs in $w'$ and the number of times it is produced[4].

[2] Formally $H_A = \{i \mid r_i = (A \to p)\}$, where $r_i$ is the $i$th grammar rule, $A$ is a left-hand side and $p$ is a right-hand side of the rule.

[3] See the proof in [6].

[4] Terminals can not be expanded in contrast to nonterminal symbols.

## 2.2 Subclasses of the Associated Systems

Restricting $w'$ and $w''$ one can derive some particular subclasses of ANLDE systems (2.1). Let us introduce several of them.

**Derivation of a sentence starting from a nonterminal**
In this case the string $w'$ is a single nonterminal $A_k$, $w''$ is a sentence derived from $A_k$ in $G$: $w'' \in L_G(A_k)$, where $L_G(A_k)$ is a set of all sentential forms of $A_k$ in $G$, or in other words the language generated by $A_k$.

$$\begin{cases} \sum\limits_{i \in H_{A_k}} x_i = \sum\limits_{i=1}^{n} \gamma_{iA_k}\, x_k + 1 \quad \text{for the start symbol } A_k \\ \sum\limits_{i \in H_A} x_i = \sum\limits_{i=1}^{n} \gamma_{iA}\, x_k \quad \text{for all } A \in N \setminus \{A_k\} \\ \sum\limits_{i=1}^{n} \gamma_{ka}\, x_i = \beta_a \quad \text{for all } a \in \Sigma\,, \end{cases} \tag{2.2}$$

The nonterminal $A_k$ is considered as a start symbol of $G$. This subclass of ANLDE systems was introduced by M. Filgueiras and A. Tomás [3].

**Derivation of empty string starting from a nonterminal**
A particular case of previous one when $w' = A_k$, $w'' = \varepsilon$. It is evident that there must be no terminal symbols in any derivation $A_k \Rightarrow^+ \varepsilon$.

$$\begin{cases} \sum\limits_{i \in H_{A_k}} x_i = \sum\limits_{i=1}^{n} \gamma_{iA_k}\, x_i + 1 \quad \text{for the start symbol } A_k \\ \sum\limits_{i \in H_A} x_i = \sum\limits_{i=1}^{n} \gamma_{iA}\, x_i \quad \text{for all } A \in N \setminus \{A_k\}\,, \end{cases} \tag{2.3}$$

We denote such a system as $\varepsilon$-ANLDE system.

**Nonterminal–to–nonterminal derivation**
Here $w' = A_k$, $w'' = A_j$. If $k \neq j$ then ANLDE system has the form:

$$\begin{cases} \sum\limits_{i \in H_{A_k}} x_i = \sum\limits_{i=1}^{n} \gamma_{iA_k} x_i + 1 \\ \sum\limits_{i \in H_{A_j}} x_i + 1 = \sum\limits_{i=1}^{n} \gamma_{iA_j} x_i \\ \sum\limits_{i \in H_A} x_i = \sum\limits_{i=1}^{n} \gamma_{iA}\, x_i \quad \text{for all } A \in N \setminus \{A_k, A_j\}\,, \end{cases} \tag{2.4}$$

We denote such a system as ($A$,$B$)-ANLDE system. In the case of $k = j$ the system is homogenous:

$$\sum_{i \in H_A} x_i = \sum_{i=1}^{n} \gamma_{iA}\, x_i \quad \text{for all } A \in N\ . \tag{2.5}$$

It is called a *homogenous ANLDE system.*

## 2.3 Solutions, derivations and cycles

A *cycle* is a derivation $A \Rightarrow^+ \alpha A \beta$ for some strings $\alpha, \beta \in (N \cup \Sigma)^*$. A cycle is *empty* if $\alpha\beta = \varepsilon$. A derivation $\alpha \Rightarrow^+ \beta$ is *minimal* if it does not contain empty cycles. An empty cycle is minimal if it does not contain another empty cycle.

Theorem 1 states that for any derivation $w' \Rightarrow^+ w''$ there is the corresponding solution $\xi$ of system (2.1). However there may exist such solutions that correspond to another derivation $v' \Rightarrow^+ v''$ or such ones that do not correspond to any successful derivation in the generative grammar $G$. Moreover, it is possible that $w''$ is not derivable from $w'$ in $G$. The first case appears because the associated system does not contain information on the order of symbols in sentential forms. The reason of the second case is the existence of cycles in the grammar.

A minimal (basis) solution of ANLDE system does not always correspond to a standard derivation $w' \Rightarrow^+ w''$. For this reason we introduce a *generalized derivation* as a set of standard derivations $\{w'_1 \Rightarrow^+ w''_1, w'_2 \Rightarrow^+ w''_2, \ldots, w'_l \Rightarrow^+ w''_l\}$ such that for each grammar symbol $X \in N \cup \Sigma$ the equality

$$\mathrm{occ}(X, w''_1 w''_2 \ldots w''_l) - \mathrm{occ}(X, w'_1 w'_2 \ldots w'_l) = \mathrm{occ}(X, w'') - \mathrm{occ}(X, w')$$

is satisfied, where $\mathrm{occ}(X, w)$ is the number of occurrences of the symbol $X$ in the string $w$.

It can be proved that any solution of ANLDE-system corresponds to a minimal generalized derivation plus a non-negative linear combination of all minimal empty cycles of the grammar.

**Theorem 2** *Any solution of ANLDE system (2.1) can be expressed as*[5]*:*

$$x = y^{w' \Rightarrow^+ w''} + y^{\varepsilon}\ , \tag{2.6}$$

[5]The proof can be found in [6]

*where* $y^{w' \Rightarrow^+ w''}$ *is a solution component corresponding to a minimal generalized derivation* $w' \Rightarrow^+ w''$, $y^{\varepsilon}$ *is a component corresponding to a multiset*[6] *of minimal empty cycles.*

Theorem 2 reduces a problem of solving ANLDE system to searching some minimal derivations and empty cycles in $G$. These minimal derivations form the set $\mathcal{N}$ of all minimal solutions of system (2.1), and the minimal empty cycles form the set $\mathcal{H}$ of all minimal solutions of the homogenous system for (2.1).

**Example 1** Let $G$ be a CF-grammar with $N = \{A_1, A_2\}$, $\Sigma = \varnothing$, and $P = \{\ A_1 \to A_1A_1A_2\ ,\ \ A_1 \to A_2A_2\ ,\ A_2 \to A_1A_1A_1A_2\ ,\ \ A_2 \to \varepsilon\ \}$. $G$ and $(A_1, \varepsilon)$ generate the $\varepsilon$-ANLDE system:

$$\begin{cases} x_1 + x_2 = 2x_1 + 3x_3 + 1 \\ x_3 + x_4 = x_1 + 2x_2 + x_3 \end{cases} \quad \Longrightarrow \quad \begin{cases} x_2 = x_1 + 3x_3 + 1 \\ x_4 = x_1 + 2x_2 \end{cases}$$

The derivation $A_1 \overset{2}{\Rightarrow} A_2A_2 \overset{3}{\Rightarrow} A_2A_1A_1A_1A_2 \overset{4}{\Rightarrow} A_2A_1A_1A_1A_1 \overset{4}{\Rightarrow} A_1A_1A_1A_1 \overset{2,2,2}{\Longrightarrow}{}^+ A_2A_2A_2A_2A_2A_2 \overset{4}{\Rightarrow} A_2A_2A_2A_2A_2 \overset{4,4,4,4}{\Longrightarrow}{}^+ A_2 \overset{4}{\Rightarrow} \varepsilon$ corresponds to the nonminimal solution $x = (0,4,1,8)$. It can be decomposed as $x = y^{A_1 \Rightarrow^+ \varepsilon} + y^{\varepsilon} = (0,1,0,2) + (0,3,1,6)$, where $y^{A_1 \Rightarrow^+ \varepsilon}$ corresponds to the minimal derivation $A_1 \overset{2}{\Rightarrow} A_2A_2 \overset{4}{\Rightarrow} A_2 \overset{4}{\Rightarrow} \varepsilon$, and $y^{\varepsilon}$ corresponds to the minimal empty cycle $A_2 \overset{3}{\Rightarrow} A_1A_1A_1A_2 \overset{2,2,2}{\Longrightarrow}{}^+ A_2A_2A_2A_2A_2A_2A_2 \overset{4,4,4,4,4,4,4}{\Longrightarrow}{}^+ A_2$ or the similar minimal empty cycle $A_1 \overset{2}{\Rightarrow} A_2A_2 \overset{3}{\Rightarrow} A_1A_1A_1A_2A_2 \overset{2,2}{\Longrightarrow}{}^+ A_1A_2A_2A_2A_2A_2A_2 \overset{4,4,4,4,4,4,4}{\Longrightarrow}{}^+ A_1$.

The Hilbert basis is $\mathcal{N} = \{(0,1,0,2\}$ and $\mathcal{H} = \{(0,3,1,6),(1,1,0,3)\}$, where the minimal solution $(1,1,0,3)$ of the homogenous system corresponds to the minimal empty cycle $A_1 \overset{1}{\Rightarrow} A_1A_1A_2 \overset{2}{\Rightarrow} A_1A_2A_2A_2 \overset{4,4,4}{\Longrightarrow}{}^+ A_1$.

# 3 Solving the Associated Systems

A large number of various algorithms have been proposed by numerous authors for solving linear Diophantine equations: G. Huet [10], M. Clausen and A. Fortenbacher [11], E. Contejean and H. Devie [12], A. Tomás and M. Filgueiras [13], L. Pottier [14], A. Boudet and H. Comon [15], E. Domenjoud and A. Tomás [2], E. Contejean [16], and F. Ajili and E. Contejean [17]. The most of them solve homogenous systems, because any NLDE-system can be transformed into an equivalent homogenous

[6] Multiset is a set in which elements may be repeated.

one. These algorithms are based on some enumeration methods and it makes them applicable only if absolute values of coefficients $||(A,b)||$ and dimension $n \times m$ are small.

In this section we introduce several new efficient algorithms for solving i) $\varepsilon$-ANLDE systems (2.3), ii) ($A$,$B$)-ANLDE system (2.4), and iii) homogenous ANLDE systems (2.5).

## 3.1 Strings and vectors of $\mathbb{Z}_+^{n+t}$

ANLDE system (2.1) does not take into account the order of symbols in strings involved in derivations. It means that there is no need to preserve the order during a derivation. As a result one can get more efficient ways to store strings over $N \cup \Sigma$.

Any string in $(N \cup \Sigma)^*$ can be described with a vector $\sigma \in \mathbb{Z}_+^{n+t}$. Each component $\sigma_l$ $(l = 1, 2, \ldots, n+t)$ is equal to the number of occurrences of each symbol of $N \cup \Sigma = \{X_1, X_2, \ldots, X_{n+t}\}$ in this string. As a result, the derivation $w' \Rightarrow^+ w''$ can be presented as a path in $\mathbb{Z}_+^{n+t}$. This description is significantly more practical than to store a string directly as it can have an arbitrary length.

**Example 2** Let $G$ be a CF-grammar from Example 1. It can be stored as the following matrix:

$$\left(\begin{array}{c|cc} 1 & 2 & 1 \\ 1 & 0 & 2 \\ 2 & 3 & 1 \\ 2 & 0 & 0 \end{array}\right)$$

The derivation $A_1 \overset{2}{\Rightarrow} A_2A_2 \overset{3}{\Rightarrow} A_2A_1A_1A_1A_2 \overset{4}{\Rightarrow} A_2A_1A_1A_1A_1 \overset{4}{\Rightarrow} A_1A_1A_1A_1 \overset{2,2,2}{\Longrightarrow}{}^+ A_2A_2A_2A_2A_2A_2 \overset{4}{\Rightarrow} A_2A_2A_2A_2A_2 \overset{4,4,4,4}{\Longrightarrow}{}^+ A_2 \overset{4}{\Rightarrow} \varepsilon$ can be presented as the path in $\mathbb{Z}_+^2$: $(1,0) \to (0,2) \to (3,2) \to (4,1) \to (4,0) \to^+ (0,6) \to (0,5) \to^+ (0,1) \to (0,0)$.

## 3.2 Algorithm for solving $\varepsilon$-ANLDE systems

Let $\Pi_k$ be a set of all minimal solutions of $\varepsilon$-ANLDE system (2.3). In this case any solution $\pi \in \Pi_k$ corresponds to a standard derivation $A_k \Rightarrow^+ \varepsilon$. The simplest elements of $\Pi_k$ corresponds to one-step derivations according to a grammar rule $A_k \to \varepsilon$, i.e. if there exists a rule $r_i = (A_k \to \varepsilon)$, then a vector $\pi^{(k)} = (0, \ldots, 0, 1, 0, \ldots, 0) \in \mathbb{Z}^m$ is a solution (the only 1 is

on the $i$th place—the only application of the rule $r_i$). To find all other elements of $\Pi_k$ one can use the well-known Dijkstra algorithm—a very similar algorithm is known to decide whether or not empty string $\varepsilon$ belongs to the language $L(A_k)$, see [4, 5].

Algorithm 1 sums up these ideas. It uses the following notations:

$p\Pi$ is a set of all possible sums of $p$ elements from $\Pi$ (these elements may be repeated). Formally $p\Pi = \{\pi^{i_1}+\pi^{i_2}+\ldots+\pi^{i_p} \mid \pi^{i_1}, \pi^{i_2}, \ldots, \pi^{i_p} \in \Pi\}$. For example, for $p = 3$ and $\Pi = \{(1,2,0),(0,0,3)\}$ the set $p\Pi$ is equal to $\{(3,6,0),(2,4,3),(1,2,6),(0,0,9)\}$.

**Algorithm 1** All minimal solutions of $\varepsilon$-ANLDE system (for all nonterminal symbols $A_k$, $k = 1, 2, \ldots, n$)

**Require:** Sets $N$ and $P$,
$N = \{A_1, A_2, \ldots, A_n\}$ —nonterminal alphabet,
$P = \{r_1, r_2, \ldots, r_m\}$ —grammar rules in the form $r = (A_k \to p) \in P$, ($p$ is presented as $(p_1, p_2, \ldots, p_n) \in \mathbb{Z}_+^n$).
**Ensure:** Sets $\Pi_k$ for each $k = 1, 2, \ldots, n$.

$\Pi_k \leftarrow \varnothing \quad k = 1, 2, \ldots, n$; {At the beginning $\Pi_k$ are empty sets}
{Initialization of $\Pi_k$ with the simplest empty rules $A_k \to \varepsilon$}
**for all** $r_i \in P$ such that $r_i = (A_k \to \varepsilon)$ **do**
  $\pi \leftarrow (0, 0, \ldots, 1, \ldots, 0)$; {the only 1 is on the $i$th position}
  $\Pi_k \leftarrow \Pi_k \cup \{\pi\}$;
**end for**
`modif_flag` $\leftarrow$ TRUE
**while** `modif_flag` = FALSE **do** {Iteration of all $\Pi_k$}
  `modif_flag` $\leftarrow$ FALSE
  **for all** $r_i = (A_k \to p) \in P$ **do**
    $U \leftarrow \min(e_i + p_1\Pi_1 + p_2\Pi_2 + \ldots + p_n\Pi_n, \Pi_k)$;
    **if** $U \neq \varnothing$ **then**
      `modif_flag` $\leftarrow$ TRUE;
      $\Pi_k \leftarrow \Pi_k \cup U$;
    **end if**
  **end for**
**end while**

$\Pi' + \Pi''$ is a set of all possible sums of an element form $\Pi'$ and an element from $\Pi''$. Formally $\Pi' + \Pi'' = \{\pi' + \pi'' \mid \pi' \in \Pi', \pi'' \in \Pi''\}$. For example, for $\Pi' = \{(3,6,0),(2,4,3),(1,2,6),(0,0,9)\}$ and $\Pi'' = \{(0,5,2)\}$ the sum $\Pi' + \Pi'' = \{(3,11,2),(2,9,5),(1,7,8),(0,5,11)\}$.

$\min(\Pi', \Pi)$ is a set of all minimal elements from $\Pi' \cup \Pi$. Formally $\min(\Pi', \Pi) = \{\pi \in \Pi' \cup \Pi \mid \nexists \pi' \in \Pi' \cup \Pi, \pi' \leq \pi, \pi' \neq \pi\}$. For example, for $\Pi' = \{(1,2,0), (3,0,1), (0,0,2)\}$ and $\Pi = \{(0,2,0), (3,0,1)\}$ the set $\min(\Pi', \Pi)$ contains the only vector $(0,0,2)$.

At each step of the iteration the algorithm constructs a set $U = \min(e_i + p_1\Pi_1 + p_2\Pi_2 + \ldots + p_n\Pi_n, \Pi_k)$. It means that it tries to use a rule $r_i = (A_k \to p)$ as the first rule of a derivation $A_k \Rightarrow \varepsilon$. The application of the rule results in a sentential form that is equal to its right-hand side—there are $p_1$ occupancies of $A_1$, $p_2$ occupancies of $A_2$, ..., and $p_n$ occupancies of $A_n$. Thus, to reduce it to $\varepsilon$ one should use $p_1$ derivations $A_1 \Rightarrow^+ \varepsilon$, $p_2$ derivations $A_2 \Rightarrow^+ \varepsilon$, ..., and $p_n$ derivations $A_n \Rightarrow^+ \varepsilon$.

Algorithms searching the Hilbert basis can not be considered as NP problems, because in general their output $(\mathcal{N}, \mathcal{H})$ is sized exponentially on the input dimensions $n \times m$. Thus we introduce an additional parameter $M$ that limits the size of the output and it will be used to describe the complexity among with standard $n$ and $m$. We call the complexity *polynomial* if it is $O(n^\alpha m^\beta M^\gamma)$ for some $\alpha, \beta, \gamma \geq 0$. The complexity is *pseudo-polynomial* if a bound on absolute values of input ($||(A,b)|| \leq$ const) makes it polynomial.

The total time complexity of Algorithm 1 is determined by the iteration stage (Dijkstra algorithm). Let the cardinality of each $\Pi_k$ be limited with a constant $M$: $|\Pi_k| \leq M$ for all $k = 1, 2, \ldots n$. In the worst case at each iteration there is only one element added to some $\Pi_k$, i.e. $|U| = 1$. Thus, the number of the iterations is limited by $Mnm$.

The computation of $U$ consumes additional time. Let $N = \max\{p_j \mid r_i = (A_k \to p) \in R\}$. Any set $p_j\Pi_k$ contains no more then $C^N_{M+N-1}$ elements. Therefore $|p_1\Pi_1 + p_2\Pi_2 + \ldots + p_n\Pi_n| \leq \left(C^N_{M+N-1}\right)^n$. On this assumption the complexity of the iteration stage is equal to $(Mnm)\left(C^N_{M+N-1}\right)^n m$. Fortunately, it is possible to reduce the complexity of $U$ computation, constructing only those elements that are really necessary for $\Pi_k$ and not to spend time for extra solutions (nonmini-

mal or previously found). It means $U$ can be constructed in time $Mnm$. Therefore, we have

**Theorem 3** *Sets* $\Pi_k$ *can be constructed by Algorithm 1 in time*[7]

$$mn + (Mnm)(Mnm) = O(M^2n^2m^2) = O(M^2m^4) .$$

The theorem gives an upper bound of the time complexity of Algorithm 1. In practice, this algorithm works faster, because the number of the iterations and the cardinality of $U$ compensate each other.

The most important disadvantage of the algorithm is that it solves $n$ $\varepsilon$-ANLDE systems simultaneously, but it is not likely to be a satisfactory way to search $\Pi_k$ separately.

## 3.3 Algorithm for solving ($A$,$B$)-ANLDE systems

For solving the ($A_k$,$A_j$)-ANLDE system one can use the sets $\Pi_k$ constructed by Algorithm 1. Let $C_{kj}$ be a set of all minimal solutions of the ($A_k$,$A_j$)-ANLDE system. Let $r_i = (A_k \to p)$ be a grammar rule. The simplest solutions corresponding to $A_k \Rightarrow^+ A_j$ belong to the set:

$$p_1\Pi_1 + p_2\Pi_2 + \ldots + p_{j-1}\Pi_{j-1} + (p_j - 1)\Pi_j + p_{j+1}\Pi_{j+1} + \ldots + p_n\Pi_n .$$

It means that the algorithm tries to use the rule $r_i$ as the first rule of a derivation. Reducing the sentential form $p$ to $A_j$ requires $p_1$ derivations $A_1 \Rightarrow^+ \varepsilon$, $p_2$ derivations $A_2 \Rightarrow^+ \varepsilon$, ..., $p_j - 1$ derivations $A_j \Rightarrow^+ \varepsilon$, ..., and $p_n$ derivations $A_n \Rightarrow^+ \varepsilon$, This is performed in the initialization stage of Algorithm 2.

Unfortunately, this does not result in all possible minimal solutions, but it is a complete base for the next stage that iteratively constructs the remaining solutions; if there are two derivations $A_k \Rightarrow^+ A_s$ and $A_s \Rightarrow^+ A_j$ then they can be combined in the derivation $A_k \Rightarrow^+ A_j$.

Let $M$ be a constant limiting the cardinality of all sets $\Pi_k$ and $C_{kj}$. The initialization stage of Algorithm 2 works in time $Mnm^2$. The iteration stage (transitive closure) works in time $n^3M^3m$.

**Theorem 4** *Sets* $C_{kj}$ *can be constructed by Algorithm 2 in time*

$$Mnm^2 + M^3n^3m = O(M^3m^4) .$$

[7] For any ANLDE system $n \leq m$.

**Algorithm 2** All minimal solutions of $(A_k,A_j)$-ANLDE system $A_k, A_j \in N$ for all $k, j = 1, 2, \ldots, n$

**Require:** Sets $N$, $P$ and $\Pi_k$,
$N = \{A_1, A_2, \ldots, A_n\}$ is a nonterminal alphabet of the grammar,
$P = \{r_1, r_2, \ldots, r_m\}$ is the grammar rules in the form $r = (A_k \to p) \in P$,
$\Pi_k$ is the set of all minimal solutions of $\varepsilon$-ANLDE system for the nonterminal $A_k \in N$, $k, j = 1, 2, \ldots, n$.
**Ensure:** Sets $C_{kj}$ for all $k, j = 1, 2, \ldots, n$.

$C_{kj} \leftarrow \varnothing \quad k, j = 1, 2, \ldots, n$; {At the beginning $C_{kj}$ are empty sets}
**for all** $r_i = (A_k \to p) \in P$ **do** {Initialization of the sets $C_{kj}$}
  **for** $j = 1$ to $n$ **do**
    **if** $p_j > 0$ **then**
      $U \leftarrow \min\Big(e_i + p_1\Pi_1 + p_2\Pi_2 + \ldots + p_{j-1}\Pi_{j-1} + (p_j - 1)\Pi_j +$
      $+ p_{j+1}\Pi_{j+1} + \ldots + p_n\Pi_n, C_{kj}\Big)$;
      $C_{kj} \leftarrow C_{kj} \cup U$;
    **end if**
  **end for**
**end for**
**for** $s = 1$ to $n$ **do** {Transitive closure of the sets $C_{kj}$}
  **for** $k = 1$ to $n$, $k \neq s$ **do**
    **for** $j = 1$ to $n$, $j \neq s$ **do**
      $U \leftarrow \min(C_{ks} + C_{sj}, C_{kj})$;
      $C_{kj} \leftarrow C_{kj} \cup U$;
    **end for**
  **end for**
**end for**

This algorithm can be used for solving the most important ANLDE subclass—homogenous ANLDE. In this case $\bigcup_{k=1}^{n} C_{kk}$ is a set of all minimal solutions of homogenous ANLDE system (2.5) (all minimal empty cycles).

Algorithm 2 simultaneously constructs all sets $C_{kj}$. However, there is an algorithm based on Algorithm 1 that simultaneously constructs

all $\Pi_l$ and $C_{kj}$ for fixed $k$ and $j$. This algorithm uses a fact that[8] $c^{(kj)} + \pi^{(j)} = \pi^{(k)}$, where $c^{(kj)} \in C_{kj}$, $\pi^{(j)} \in \Pi_j$, and $\pi^{(k)}$ corresponds to $A_k \Rightarrow^+ \varepsilon$. We omit to present this algorithm in the paper because its understanding requires additional theoretical background. This algorithm is more efficient than Algorithm 2 both in time and space, and its theoretical time complexity is $O(M^2 m^4)$.

## 3.4 Tests

We have tested the algorithms on specially generated homogenous ANLDE systems. The dimensions of the generated systems were in range: $n \in [1, 1000]$, $m \in [n, n + 200]$, $p_j \in [0, 500]$. The systems were generated in such a way that each of them has at least one solution but not more than 200. Figures 1 and 2 shows the results on 7267 sample.

To test the implementation we use an integer linear programming solver[9]. It can find only one solution in accordance with some cost function. For each ANLDE system we run our algorithm to construct $\mathcal{H}$, then run the ILP solver several times and tested that the found solution could be represented with the Hilbert basis. The ILP solver used significantly more time than our grammar-based algorithm—several hours or days[10], but our algorithm nearly always worked in less than a minute.

The experiment results approve nonlinear increasing of the complexity (see Figure 1). In Figure 2 we can see that the complexity is approximately linear if one considers it as a function of the number of minimal solutions. It allows to conclude that the algorithm works approximately in time $\Theta(Mm^2)$ in the average case.

The angle of the complexity in $m = 1000$ in Figure 1 is a result of our generating strategy: $n$ has an upper limit 1000, but $m$ can exceed it on 200. Thus, in the case of $m \in (1000, 1200]$ there is always $n < m$ and as a result the system always has more than one solution: the more the difference $m-n$ is then a system is less constrained the more solutions it has.

[8] It is easy to see that two derivations $A_k \Rightarrow^+ A_j$ and $A_j \Rightarrow^+ \varepsilon$ can be combined into the derivation $A_k \Rightarrow^+ \varepsilon$.

[9] M. Berkelaar, lp_solver, `ftp://ftp.ics.ele.tue.nl/pub/lp_solve/`

[10] Sometimes we had to stop the ILP solver because it worked too long. ILP is satisfactory for very small dimensions like $m \in [1, 30]$ and its performance decreases rapidly (exponentially) with the growth of $m$.

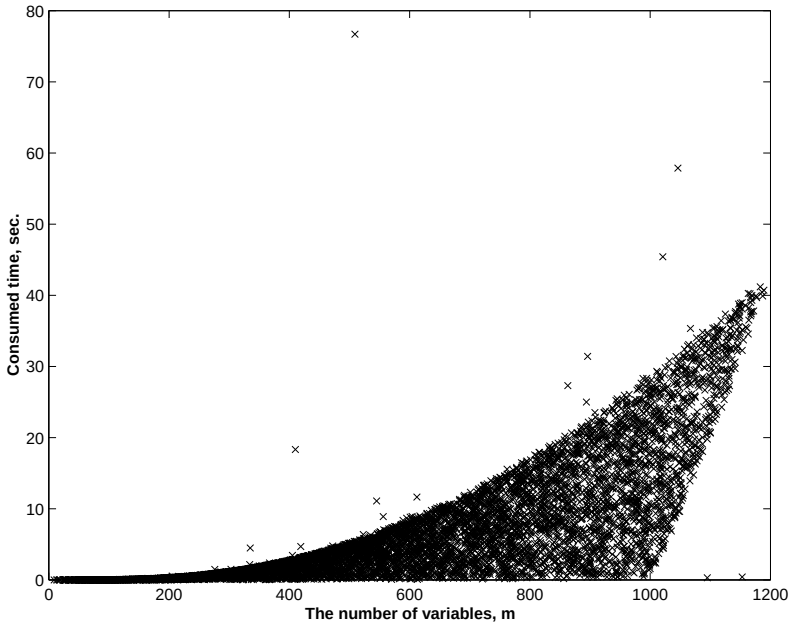


***Figure 1:*** *Experiment total time complexity as a function on the number of unknowns*

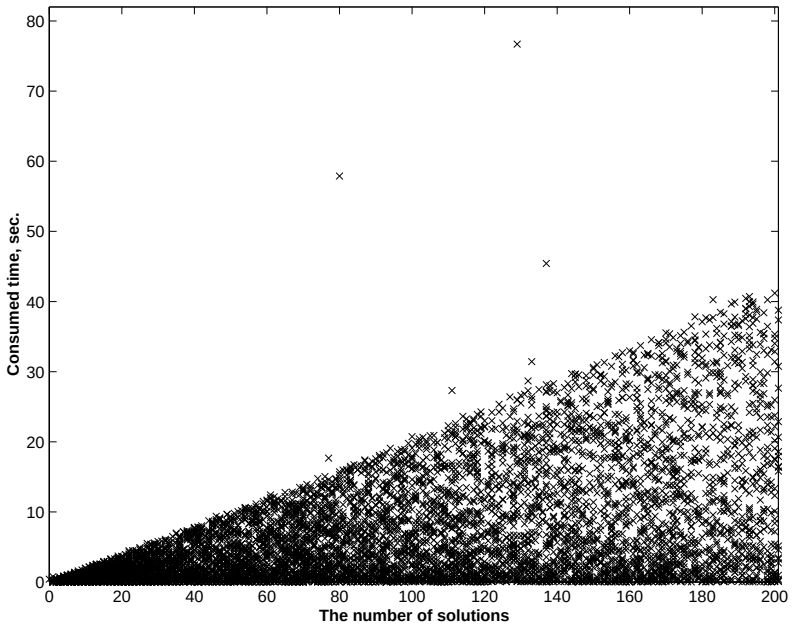


***Figure 2:*** *Experiment total time complexity as a function on the number of solutions*

# 4 Conclusion

We gave several algorithms for solving special subclasses of non-negative linear Diophantine systems. These subclasses belong to a class of associated with context-free grammars systems. The problem of solving these systems is reduced to computing some derivations in generative grammars. This allows using the well-known parsing methods.

The attractive property of our algorithms is their polynomial and pseudo-polynomial time complexity. They are significantly more efficient than "universal" algorithms for solving arbitrary non-negative linear Diophantine systems and can be used even for very large systems.

# References

[1] Schrijver A., *Theory of Linear and Integer Programming.* John Willey & Sons Ltd., 1986.

[2] Domenjoud E., Tomás A. *From Elliott-MacMahon to an algorithm for general constraints on naturals.* In U. Montanari, F. Rossi (eds.), Principles and Practice of Constraint Programming — CP'95. Lecture Notes in Computer Science 976. Springer-Verlag, 1995. pp. 18–35.

[3] M. Filgueiras, A. Tomás, *Solving Linear Constraints on Finite Domains through Parsing.* Proc. of EPTA'91, 1991. pp. 1–16.

[4] A. Aho, J. Ullman, *The Theory of Parsing, Translation and Compiling. Volume I: Parsing.* Prentice-Hall, Inc. 1972.

[5] S. Sippu, E. Soisalon-Soininen, *Parsing Theory. Volume I: Languages and Parsing.* EATCS Monographs on Theoretical Computer Science, W. Brauer, G.Rozenberg, A. Salomaa (Eds.), Springer-Verlag, 1988.

[6] I. A. Bogoiavlenski[11], D. G. Korzoun, *On Solutions of Linear Diophantine Equations System, Associated with a Context-Free Grammar.* Transactions of the University of Petrozavodsk on "Applied mathematics and computer science". Vol. 6. Petrozavodsk, 1998, pp. 79–94. (in Russian)

[7] D. G. Korzoun, *Solution of One Class of Linear Diophantine Equations in Non-negative Integers through Parsing.* Transactions of the University of Petrozavodsk on "Applied mathematics and computer science". Vol. 7. Petrozavodsk, 1999, pp. 93–116. (in Russian)

[11] The name "Yury A. Bogoyavlenskiy" was spelled as "Iouri A. Bogoiavlenski" in accordance with the old passport for travelling abroad, which was renewed at 2000.

[8] D. G. Korzoun, *On a Certain Interdependence of Formal Grammars and Systems of Linear Diophantine Equations.* Bulletin of Young Scientists. 2'99. Sant-Peterburg: St. Petersburg State Technical University (StPSTU), 1999. (in Russian)

[9] D. G. Korzoun, *Applications of Formal Languages Theory to Development of Algorithms for Analysis and Solving of Linear Diophantine Systems.* Proceedings of the IV Assembly of Young Scientists and Specialists. Sant-Peterburg: St. Petersburg State Technical University (StPSTU), 1999, pp. 38–38. (in Russian)

[10] Huet G. *An algorithm to generate the basis of solutions to homogenous linear Diophantine equations.* Information Processing Letters. Vol. 3. No. 7. 1978. pp. 144–147.

[11] Clausen M., Fortenbacher A. *Efficient Solution of Linear Diophantine Equations.* Journal of Symbolic Computation. Vol. 8. No. 1&2. 1989. pp. 201–216.

[12] Contejean E., Devie H. *An efficient algorithm for solving systems of Diophantine equations.* Information and Computation. Vol. 113. No. 1. 1994. pp. 143–172.

[13] A. Tomás, M. Filgueiras *A New Method for Solving Linear Constraints on the Natural Numbers.* Proc. of EPTA'91, 1991. pp. 30–44.

[14] Pottier L. *Minimal solutions of linear Diophantine systems: bounds and algorithms.* Proceedings of the 4th International Conference on Rewriting Techniques and Applications (RTA'91). Como, Italy. 1991. pp. 162–173.

[15] Boudet A., Comon H. *Diophantine Equations, Presburger Arithmetic and Finite Automata.* Proceedings of the 21st International Colloquium on Trees in Algebra and Programming (CAAP'96), 1996. pp. 30–43.

[16] Contejean E. *Solving linear Diophantine constraints incrementally.* Proceedings of the 10th International Conference on Logic programming, Budapest, Hungary, 1993. The MIT Press.

[17] Ajili F., Contejean E. *Avoiding slack variables in the solving linear Diophantine equations and inequations.* Theoretical Computer Science. Vol. 173. No. 1. 1997. pp. 183–208.