Résolution de systèmes linéaires d'équations diophantiennes

Evelyne Contejean et Hervé Devie

RESUME - Nous présentons ici un algorithme pour résoudre les systèmes linéaires homogènes d'équations diophantiennes de façon directe, à l'aide d'une interprétation géométrique.

SOLVING LINEAR SYSTEMS OF DIOPHANTINE EQUATIONS

ABSTRACT - An algorithm for solving directly homogeneous linear systems of Diophantine equations is presented, which is based on a geometrical interpretation.

Our aim is to get the set $\mathcal{S}$ of all solutions of a linear homogeneous system $Ax = 0$. q is the number of unknowns and p is the number of equations of this system. Since $\mathcal{S}$ is an additive monoïd, we actually need a set of generators of $\mathcal{S}$, for example the set of all minimal tuples of $\mathcal{S}$ with respect to the ordering $>^q$. (See the notations in the main text.)

The idea of our algorithm is to search the space $\mathbb{N}^q$ starting from the vectors in its canonical basis. Suppose that we have obtained a q-tuple $x = (x_1, \ldots, x_q)$. If x is equal to or greater than a solution, the search is stopped. Otherwise, x can be non deterministically incremented yielding q distinct new q-tuples $(x_1 + 1, x_2, \ldots, x_q)$, $(x_1, x_2 + 1, \ldots, x_q), \ldots, (x_1, x_2, \ldots, x_q + 1)$. The key point is that a constraint may be added without loosing completeness, which can drastically decrease the search space. Fortenbacher gave a constraint for the case of one equation [5]. By using a geometrical interpretation (see the figures in the main text) we generalize Fortenbacher's idea to arbitrary many equations. Formally, x can be incremented on its j^{th} component if and only if $Ax + Ae_j$ lies in the half-space containing the origin and limited by the affin hyperplan which is orthogonal to the vector Ax and contains the extremity of Ax. Mathematically, this constraint can be expressed by the following very simple condition :

$$Ax \cdot Ae_j < 0$$

This algorithm is correct and complete, i.e. it computes only elements in $\mathcal{S}$ and all the elements in $\mathcal{S}$. The termination proof is quite delicate and works by contradiction. Suppose that an infinite sequence $(v_n)_{n\geq 1}$ exists, such that each v_{n+1} is obtained from v_n by adding some e_{j_n} under the above condition $Av_n \cdot Ae_{j_n} < 0$. We consider the sequence $(v'_n)_{n\geq 1}$ defined by $v'_n = \frac{v_n}{n}$. Using some subsequences and compacity of the real interval $[0, 1]$, we can deduce that there exists a tuple l in $[0, 1]^q$ such that $Al = 0$. A continuity argument provides from l a tuple in $\mathbb{Q}^q$, hence a tuple l'' in $\mathbb{N}^q$ such that $Al'' = 0$. Hence, l'' is a solution. Moreover, l'' can be chosen in such a way that v_n is greater than l'' with respect to $>^q$ for some large enough n. Hence, this infinite sequence cannot be computed by the algorithm.

Since the termination proof is not constructive, it does not provide any complexity result for the algorithm. We explain how to minimize space computation and how

to improve time computation, and we give several benchmarks. As a conclusion, this method is, in most cases, much more efficient than the previously available ones.

INTRODUCTION ET NOTATIONS.- Les algorithmes connus jusqu'à présent s'attachent à résoudre une seule équation et procèdent par récurrence sur le nombre d'équations pour résoudre les systèmes [1][2][3][4]. Fortenbacher en particulier a décrit un algorithme pour une seule équation [5]. Nous nous proposons ici d'étendre cet algorithme aux systèmes, par le biais d'une interprétation géométrique. Les preuves de complétude et de correction de l'algorithme étendu sont calquées sur celles de Fortenbacher ; seule celle de terminaison est délicate.

Soit le système

(1) $$Ax = 0,$$

où A est une matrice à p lignes et q colonnes d'entiers relatifs. On cherche toutes les solutions $x = (x_1, \ldots, x_q)$ à coefficients entiers naturels. On munit $\mathbb{N}^q$ de l'ordre partiel $>^q$ suivant, qui étend l'ordre naturel $>$ sur les entiers :

(2) $(s_1, \ldots, s_q) >^q (t_1, \ldots, t_q) \ \vdash$ pour tout j dans $\{1, ..., q\}, s_j \geq t_j$ et il existe j dans $\{1, ..., q\}$ tel que $s_j > t_j$.

L'ensemble $\mathcal{S}$ des solutions formant un monoïde additif, on s'intéresse uniquement aux solutions autres que la solution triviale et minimales pour l'ordre $>^q$, car elles constituent une partie génératrice minimale de $\mathcal{S}$ que l'on note $\mathcal{M}$.

Dans la suite, e_j ($j \in \{1, ..., q\}$) désigne le $j^{ième}$ vecteur de la base canonique de $\mathbb{N}^q$, '$\cdot$' le produit scalaire usuel et $\| \ \|$ la norme euclidienne. De plus, afin de faciliter l'exposition du fonctionnement de l'algorithme, nous identifions, pour les arbres, les forêts ou les graphes acycliques orientés, les notions de nœud et d'étiquette de nœud. En conséquence, nous confondons deux nœuds de même étiquette et nous parlons de successeur d'une étiquette.

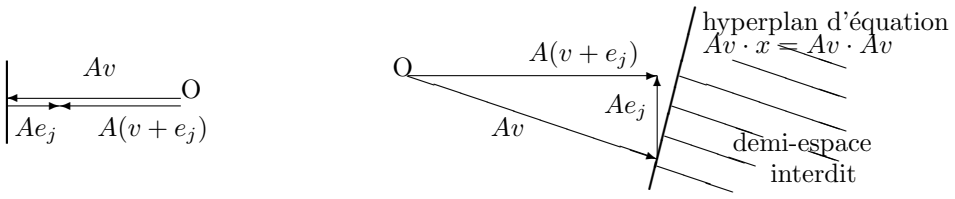
DESCRIPTION DE L'ALGORITHME.- L'algorithme consiste à développer en largeur d'abord un graphe acyclique orienté étiqueté par des vecteurs de $\mathbb{N}^q$. Pour j dans $\{1, ..., q\}$, le vecteur e_j étiquète la $j^{ième}$ racine du graphe. Pour une étiquette v, on distingue trois cas :

- v est une solution non nulle de $Ax = 0$; alors v est ajoutée à l'ensemble de solutions minimales déjà calculées et n'a pas de successeur.
- v est plus grande vis-à-vis de $>^q$ qu'une solution minimale déjà calculée ; alors v n'a pas de successeur.
- sinon, v n'est pas solution de $Ax = 0$, donc $Av \neq 0$; alors v a pour successeurs les seuls vecteurs $v + e_j$ tels que le point $A(v + e_j)$ soit situé dans le demi-espace ouvert contenant l'origine et délimité par l'hyperplan affine orthogonal au vecteur Av et passant par le point de coordonnées celles de Av, condition qui s'écrit :

  (3) $$Av \cdot Ae_j < 0.$$

Remarque. - Dans les deux premiers cas, on ne développe pas davantage le graphe

car on n'obtiendrait dans les parties tronquées que des solutions non minimales.



Cas particulier d'une équation (Fortenbacher)

Cas général

Pour $n \geq 1$, on désignera par $\mathcal{P}_n$ l'ensemble des étiquettes de la forêt situées à la profondeur $(n-1)$ et par $\mathcal{M}_n$ celui des étiquettes situées à une profondeur inférieure ou égale à $(n-1)$ qui sont solutions de S. On a donc formellement :

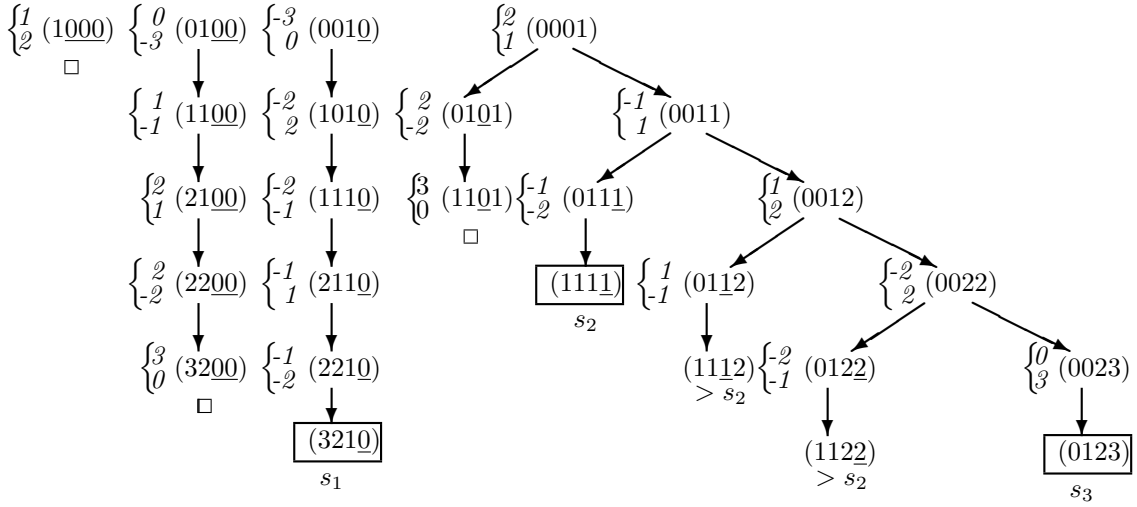
- $\mathcal{P}_1 = \{e_j \mid j \in \{1, ..., q\}\}$;
  $\mathcal{M}_1 = \{v \in \mathcal{P}_1 \mid Av = 0\}$;
- $\forall n \geq 1$, $\mathcal{P}_{n+1} = \{v + e_j \mid v \in \mathcal{P}_n \backslash \mathcal{M}_n,\ \forall u \in \mathcal{M}_n \ \ v \not>^q u,\ Av \cdot Ae_j < 0\}$;
  $\mathcal{M}_{n+1} = \mathcal{M}_n \cup \{v \in \mathcal{P}_{n+1} \mid Av = 0\}$.

L'algorithme retourne le premier $\mathcal{M}_n$ tel que $\mathcal{P}_{n+1}$ est vide.

Remarque. - Il est en outre possible de réduire les ensembles $\mathcal{P}_n$ en imposant une contrainte supplémentaire (C) de non redondance portant sur les indices j : la $j^{ième}$ composante d'un vecteur est dite *gelée* si l'on s'interdit de l'incrémenter lorsque l'on développe tous les descendants de ce vecteur. Si v a au moins deux successeurs $v + e_{j_1}$ et $v + e_{j_2}$, avec $j_1 < j_2$, on gèle la $j_2^{ième}$ composante de $v + e_{j_1}$. Cette contrainte réduit le graphe à une forêt.

Exemple. - L'algorithme développe l'arbre ci-dessous pour résoudre le système suivant :

$$\begin{cases} x & - & 3z & + & 2t & = & 0 \\ 2x & - & 3y & & + & t & = & 0 \end{cases}$$



A gauche de chaque vecteur v figure le vecteur Av. Les composantes soulignées sont gelées à cause de la contrainte (C). Le signe □ signifie qu'un nœud n'a pas de successeurs

à cause des contraintes (C) et (3).

PROPRIÉTÉS DE L'ALGORITHME.- PROPOSITION 1. - (Complétude) *L'algorithme calcule l'ensemble de toutes les solutions minimales $\mathcal{M}$, c'est à dire que $\mathcal{M}$ est inclus dans l'union des $\mathcal{M}_n$, pour $n \geq 1$.*

PROPOSITION 2. - (Correction) *L'algorithme ne calcule que des solutions minimales, c'est à dire que $\mathcal{M}_n$ est inclus dans $\mathcal{M}$ pour tout $n \geq 1$.*

THEOREME 1. - (Terminaison) *L'algorithme termine, c'est à dire qu'il existe $n_0 \geq 1$ tel que $\mathcal{P}_{n_0}$ est vide.*

Démonstration. - Si l'algorithme termine sans la contrainte (C), alors il termine également si l'on en tient compte. On va donc prouver que l'algorithme termine sans imposer la contrainte (C). La démonstration repose sur le lemme suivant :

LEMME 1. - *Soit v_n un vecteur de $\mathcal{P}_n$. Alors $\|Av_n\| \leq C\sqrt{n}$,*
où $C = Max\{\|Ae_j\| \mid j \in \{1, ..., q\}\}$.

Démonstration. - La démonstration se fait par récurrence sur n. Le cas $n = 1$ est trivial. Supposons le résultat vrai pour $n \geq 1$ et soit $v_{n+1} \in \mathcal{P}_{n+1}$. v_{n+1} s'écrit $v_n + e_j$ avec $v_n \in \mathcal{P}_n$ et $j \in \{1, ..., q\}$ tel que $Av_n \cdot Ae_j < 0$. On a donc :
$\|Av_{n+1}\|^2 = \|Av_n\|^2 + \|Ae_j\|^2 + 2\,Av_n \cdot Ae_j \leq (n+1)\,C^2$. □

Supposons que l'algorithme ne termine pas. Alors il existe une suite $(v_n)_{n\geq 1}$ infinie telle que, pour tout $n \geq 1$, v_n appartient à $\mathcal{P}_n$ et $v_{n+1} = v_n + e_j$ pour un certain j dans $\{1, ..., q\}$. Le vecteur v_n s'écrit $(v_{1n}, \ldots, v_{qn})$ où, pour tout j dans $\{1, ..., q\}$, v_{jn} est un entier naturel, et $\sum_{j=1}^{q} v_{jn} = n$. Considérons alors la suite $(v'_n)_{n\geq 1}$ définie par : $v'_n = \frac{v_n}{n}$. La suite $(v'_n)_{n\geq 1}$ prend ses valeurs dans l'ensemble $[0,1]^q$ (où $[0,1]$ est l'intervalle réel). Cet ensemble étant compact, elle admet une valeur d'adhérence $(l_1, \ldots, l_q)$. Soit $(v'_{\phi(n)})_{n\geq 1}$ une suite extraite de (v'_n) de limite $(l_1, \ldots, l_q)$ et $(v_{\phi(n)})_{n\geq 1}$ la suite extraite de (v_n) correspondante :

$$\sum_{j=1}^{q} l_j\, Ae_j = \sum_{j=1}^{q} (\lim_{n\to\infty} v'_{j\phi(n)})\, Ae_j = \lim_{n\to\infty} \frac{1}{\phi(n)} (\sum_{j=1}^{q} v_{j\phi(n)}\, Ae_j),$$

et donc, d'après le lemme précédent, il vient :

$$\|\sum_{j=1}^{q} l_j\, Ae_j\| = \lim_{n\to\infty} \frac{1}{\phi(n)} \|\sum_{j=1}^{q} v_{j\phi(n)}\, Ae_j\| = \lim_{n\to\infty} \frac{1}{\phi(n)} \|Av_{\phi(n)}\| \leq \lim_{n\to\infty} \frac{C}{\sqrt{\phi(n)}} = 0.$$

On a donc :

$$\sum_{j=1}^{q} l_j\, Ae_j = 0.$$

Tous les l_j ne sont pas nuls (car, par passage à la limite, $\sum_{j=1}^{q} l_j = 1$) et, sans perte de généralité, on peut supposer qu'aucun d'entre eux n'est nul (sinon, on ne considère que ceux qui ne le sont pas). Remarquons qu'alors, la suite $(v_{j\phi(n)})_{n\geq 1}$ prend pour tout j dans $\{1, ..., q\}$ des valeurs arbitrairement grandes.
La famille de réels $(l_j)_{1\leq j\leq q}$ engendre un espace vectoriel sur $\mathbb{Q}$ de dimension m, où $1 \leq m \leq q$. Soit $(l_1, \ldots, l_m)$ (à renumérotation des l_j près) une base de cet espace vectoriel. Les vecteurs l_k, $m+1 \leq k \leq q$, se décomposent dans cette base :

$$\forall k \in \{m+1, ..., q\},\ \forall j \in \{1, ..., m\},\ \exists \alpha_{jk} \in \mathbb{Q},\ l_k = \sum_{j=1}^{m} \alpha_{jk} l_j.$$

Il vient alors :

$$0 = \sum_{j=1}^{q} l_j\, Ae_j = \sum_{j=1}^{m} l_j\, (Ae_j + \sum_{k=m+1}^{q} \alpha_{jk}\, Ae_k).$$

Pour tout j dans $\{1..., m\}$, $Ae_j + \sum_{k=m+1}^{q} \alpha_{jk}\, Ae_k$ est un vecteur à coefficients rationnels. La famille $(l_j)_{1 \leq j \leq m}$ étant libre sur $\mathbb{Q}$, on a donc nécessairement :

$$\forall j \in \{1, ..., m\},\ Ae_j + \sum_{k=m+1}^{q} \alpha_{jk}\, Ae_k = 0.$$

Par conséquent :

$$\forall (z_1, \ldots, z_m) \in \mathbb{R}^m,\ 0 = \sum_{j=1}^{m} z_j\, (Ae_j + \sum_{k=m+1}^{q} \alpha_{jk}\, Ae_k) = \sum_{j=1}^{m} z_j\, Ae_j + \sum_{k=m+1}^{q} (\sum_{j=1}^{m} \alpha_{jk} z_j)\, Ae_k.$$

Or, pour tout k dans $\{m+1, ..., q\}$, on a $\sum_{j=1}^{m} \alpha_{jk} l_j = l_k > 0$; donc, par un argument de continuité, il existe des nombres r_j, $(1 \leq j \leq m)$ rationnels et strictement positifs, suffisamment voisins des l_j respectifs de façon à avoir :

$$\forall k \in \{m+1, ..., q\},\ \sum_{j=1}^{m} \alpha_{jk} r_j > 0.$$

Pour de tels r_j, on a donc exhibé une combinaison linéaire nulle à coefficients rationnels positifs des vecteurs Ae_j $(1 \leq j \leq q)$.
En multipliant cette combinaison linéaire par le plus petit commun multiple des dénominateurs de ses coefficients, on obtient une combinaison linéaire nulle à coefficients entiers naturels des vecteurs Ae_j $(1 \leq j \leq q)$, c'est-à-dire une solution s (peut-être pas minimale) de S. D'après une remarque faite précédemment, chacune des suites $(v_{j\phi(n)})_{n \geq 1}$ pour tout j dans $\{1, ..., q\}$ tend vers $+\infty$; donc il existe n_0 plus petit entier de $\mathbb{N}$ tel que $v_{n_0} >^q s$. L'algorithme n'engendrera donc pas de vecteur v_{n_0+1} : la suite $(v_n)_{n \geq 1}$ est finie. □

Corollaire 1. - *Il existe n_0 tel que $\mathcal{M} = \mathcal{M}_{n_0}$.*

Performances.- La preuve de terminaison n'est pas constructive, puisqu'elle utilise de l'analyse réelle : elle ne permet pas de déterminer la complexité en temps de l'algorithme.

De façon pratique, la forêt développée par l'algorithme peut être parcourue en profondeur d'abord. On ne perd pas la propriété de terminaison, car les solutions situées à droite d'un q-uplet dans la forêt ne sont jamais plus petites que ce dernier en raison de la condition (C). En conséquence, on peut implanter l'algorithme à l'aide d'une pile de vecteurs de $\mathbb{N}^q$, de hauteur bornée par le nombre q de variables. Ainsi, l'algorithme est quadratique en espace dans le nombre de variables du système, à la place occupée par les solutions près.

La seule méthode connue jusqu'à présent pour résoudre un système à p équations consiste en gros à injecter les solutions de la première équation dans l'équation suivante pour obtenir les solutions des deux premières équations et à itérer la méthode. Bien que l'on puisse déterminer la complexité d'un tel algorithme, les inconvénients de cette méthode sont non négligeables. En effet, au cours des transformations successives, le nombre de variables risque de croître de façon importante : ainsi, le nombre de variables du premier système transformé est égal au nombre de solutions minimales de la première équation. De plus, les solutions obtenues ne sont pas toutes minimales.

Dans le tableau ci-dessous qui regroupe quelques exemples, la première colonne contient le système sous forme matricielle. Chaque système, composé d'équations e_i, est résolu de façon globale par notre algorithme (voir la dernière colonne). Les colonnes intermédiaires sont relatives à la résolution équation par équation : on doit résoudre e_1, puis les équations e'_{i+1}, chaque solution s' de e'_{i+1} donnant lieu à une solution s du sous-système composé de $e_1, \ldots, e_{i+1}$. Il faut noter que les solutions s ne sont pas toutes nécessairement distinctes ni minimales (comparer les colonnes 3 et 4). Le dernier exemple a donné lieu à un dépassement de capacité mémoire *(core)*. Les temps qui sont donnés correspondent à notre algorithme programmé en langage C sur une machine SUN 3/75. Le nombre d'étapes indique le nombre de nœuds de la forêt.

Système	algorithme équation par équation			nouvel algorithme
	Nb var	Nb sol	Nb sol min	
e_1 : -4 1 1 1 1	e_1 : 5	e_1 : 35	e_1 : 35	
e_2 : 1 -4 1 1 1	e'_2 : 35	e'_2 :215	$e_1 + e_2$: 10	1 solution
e_3 : 1 1 -4 1 1	e'_3 : 10	e'_3 : 20	$e_1 + e_2 + e_3$: 3	en 0.017s
e_4 : 1 1 1 -4 1	e'_4 : 3	e'_4 : 2	$e_1 + e_2 + e_3 + e_4$: 1	en 30 étapes
e_1 : 0 0 0 0 -2 1 1	e_1 : 7	e_1 : 7	e_1 : 7	95 solutions
e_2 : 0 0 -4 1 1 1 1	e'_2 : 7	e'_2 :21	$e_1 + e_2$: 15	en 5.817s
e_3 : -6 1 1 1 1 1 1	e'_3 : 15	e'_3 : 323	$e_1 + e_2 + e_3$: 95	en 14800 étapes
e_1 : 0 0 0 0 -3 1 1 1	e_1 : 8	e_1 : 14	e_1 : 14	11942 solutions
e_2 : 0 0 -5 1 1 1 1 1	e'_2 : 14	e'_2 :2015	$e_1 + e_2$: 149	en 760s
e_3 : -7 1 1 1 1 1 1 1	e'_3 : 149	e'_3 :*core*	$e_1 + e_2 + e_3$: 11942	en 1994343 étapes

References

[1] G. HUET : An algorithm to generate the basis of solutions to homogenous linear Diophantine equations.
Information Processing Letters, vol. 3, n^o7,1978.

[2] J.-L. LAMBERT : Une borne pour les générateurs des solutions entières positives d'une équation diophantienne linéaire.
Comptes Rendus de l'Académie des Sciences de Paris, t.305, Série I, 39-40, 1987.

[3] J.-F. ROMEUF : Solutions of a linear Diophantine system.
Rapport L.I.T.P. 88-76, Université Paris 7.

[4] E. CONTEJEAN : Unification associative-commutative.
Mémoire de D.E.A. sept 1988, Université Paris 11.

[5] A. FORTENBACHER & M. CLAUSEN : Efficient solution of linear Diophantine equations.
Interner Bericht Nr.32/87 (nov 1987), Universität Karslruhe Fakultät für Informatik.