Vol. XVI, Nº 2, Diciembre (2008)
Matemáticas: 75–85

Matemáticas:
Enseñanza Universitaria

Un algoritmo para resolver el problema de Frobenius utilizando bases de Gröbner

Gilberto García-Pulgarín
Universidad de Antioquia

John Hermes Castillo Gómez
Universidad de Nariño

Recibido Feb. 20, 2007 Aceptado Feb. 12, 2008

Abstract

Let $A = \{a_1, a_2, \ldots, a_k\}$ be a set of relatively positive prime integers, a positive integer N is called representable by A if exists non-negatives integers $x_1, x_2, \ldots, x_k$, such that $N = \sum_{i=1}^{k} a_i x_i$. The *Frobenius Problem* consits in determining the largest integer, denoted with $g(A)$, that is not representable by A. In this work we present an algorithm to solve the Frobenius Problem using Gröbner Bases. In the Apendix we present the algorithms developed in this work, implemented in the computer system algebra MuPAD.

Keywords: Frobenius Problem, Gröbner Bases.

MSC(2000): Primary: 11D04, Secondary: 11D85, 11Y50, 13P10.

Resumen

Sea $A = \{a_1, a_2, \ldots, a_k\}$ un conjunto de enteros positivos primos relativos entre sí. Dado un entero positivo N, se dice que N es representable por A si existen enteros no negativos $x_1, x_2, \ldots, x_k$ tales que $N = \sum_{i=1}^{k} a_i x_i$. El *Problema de Frobenius* consiste en encontrar el mayor entero, denotado con $g(A)$, que no es representable por A. En este artículo se presenta un algoritmo para resolver el problema de Frobenius utilizando bases de Gröbner. Al final, en el Apéndice, se presentan los algoritmos desarrollados en este trabajo implementados en el sistema de álgebra computacional MuPAD.

Palabras y frases claves: Problema de Frobenius, Bases de Gröbner.

1 Introducción

Sea $A = \{a_1, a_2, \ldots, a_k\}$ un conjunto de enteros positivos primos relativos entre sí. Se dice que un entero N es representable por A (o simplemente representable) si existen enteros no negativos $m_1, \ldots, m_k$ tales que

$$N = m_1 a_1 + \cdots + m_k a_k$$

En [3], Brauer y Schockley demuestran que a partir de cierto entero, todos los demás, son representables por A.

El *Problema de Frobenius* consiste en encontrar el mayor entero que no es representable por A. A este número se le llama el número de Frobenius de A, y se le denota con $g(a_1, \ldots, a_k) = g(A)$. Este problema también se conoce como *el Problema del Cambio de Moneda*, pues se puede dar la siguiente interpretación: supóngase que en un país se tienen billetes de todas las denominaciones enteras, pero únicamente se tiene un número finito de monedas de denominaciones enteras $a_1, \ldots, a_k$, que son primos relativos entre sí, entonces el problema del cambio de

moneda consiste en encontrar el billete de mayor denominación que no puede cambiarse utilizando estas monedas.

El caso particular de un par de enteros positivos primos relativos a_1, a_2 fue resuelto por J.J Silvester en 1884, ver [6]. A medida que el número de enteros crece, encontrar el número de Frobenius se hace más difícil. Selmer y Beyer en 1978 resolvieron el caso de tres enteros primos relativos entre sí, ver [5], utilizando un algoritmo de fracciones continuas. Si los enteros no son primos relativos entre sí el problema no tiene sentido. En efecto, si $\gcd(a_1, a_2, \ldots, a_k) = d > 1$, entonces la ecuación $a_1x_1 + a_2x_2 + \cdots + a_kx_k = N$, tiene solución si y sólo si N es un múltiplo de d; por lo que los únicos enteros representables serán los múltiplos no negativos de d.

La búsqueda de solución para el problema de Frobenius se puede hacer por medio de dos caminos. El primero es encontrar una fórmula cerrada para el cálculo del número de Frobenius. Con respecto a este primer enfoque, no se conocen fórmulas precisas para el cálculo del número de Frobenius cuando $k \geq 3$, aunque se encuentran diversos trabajos en los que se dan fórmulas cuando el conjunto A tiene características especiales. El segundo es el de presentar algoritmos eficientes que calculen el número de Frobenius. J. L. Ramírez-Alfonsín en [7], demostró que el problema de Frobenius es NP-hard. En este artículo se presenta un algoritmo para resolver el problema de Frobenius utilizando bases de Gröbner. Luego de la culminación del presente trabajo fue publicado recientemente por B. H. Roune, ver [8], un artículo donde se presenta un algoritmo para resolver el problema de Frobenius cuando los enteros a_i tienen muchas cifras decimales con el uso de las bases de Gröbner. En [9] se hace una exposición más detallada del estado del arte del problema.

2 Resultados preliminares

J.J Silvester demostró, ver [6] y [4], que cuando $k = 2$ se tiene que $g(a_1, a_2) = a_1a_2 - a_1 - a_2 = (a_1 - 1)(a_2 - 1) - 1$. Además, en [3] se demuestra que si $a_1 \leq a_2 \leq \cdots \leq a_k$ entonces

$$-1 \leq g(a_1, a_2, \ldots, a_k) \leq (a_1 - 1)(a_k - 1) - 1 \tag{1}$$

En el estudio del problema de Frobenius se utiliza la siguiente función que sirve para obtener algunas propiedades del número de Frobenius. Sea $f(a_1, \ldots, a_k)$ el mayor entero que no se puede representar como combinación lineal de $a_1, a_2, \ldots, a_k$ con coeficientes enteros positivos. La primera relación que se puede demostrar, partiendo de sus definiciones, entre $f(a_1, \ldots, a_k)$ y $g(a_1, \ldots, a_k)$ es la siguiente

$$g(a_1, \ldots, a_k) = f(a_1, \ldots, a_k) - a_1 - \cdots - a_k \tag{2}$$

De esta forma, $f(a_1, a_2) = a_1a_2$ y si $a_1 \leq a_2 \leq \cdots \leq a_k$, de la desigualdad (1) se tiene que

$$a_1 + \cdots + a_k - 1 \leq f(a_1, \ldots, a_k) \leq a_2 + a_3 + \cdots + a_{k-1} + a_1a_k \tag{3}$$

Ahora se presenta el siguiente resultado que será utilizado más adelante.

Teorema 2.1. *Sean $a_1, a_2, \ldots, a_k$ enteros positivos primos relativos entre sí, entonces*

$$f(a_1, \ldots, a_k) = \sum_{i=2}^{k} a_i x_i$$

para ciertos enteros positivos $x_2, x_3, \ldots, x_k$.

Demostración. De la definición de $f(a_1, a_2, \ldots, a_k)$ se tiene que

$$f(a_1, \ldots, a_k) < a_1 + f(a_1, \ldots, a_k) = a_1 x_1 + \sum_{i=2}^{k} a_i x_i$$

con $x_i > 0$, luego $f(a_1, \ldots, a_k) = a_1(x_1 - 1) + \sum_{i=2}^{k} a_i x_i$ lo que contradice la definición de $f(a_1, \ldots, a_k)$, a menos que $x_1 = 1$. Esto establece el teorema. □

A continuación se presenta un resultado que proporciona los elementos necesarios para construir un algoritmo para calcular el número de Frobenius de un conjunto de enteros positivos primos relativos entre sí, $A = \{a_1, \ldots, a_k\}$. En [3], Brauer y Schockley demuestran el siguiente resultado

Teorema 2.2. *Sean $a_1, a_2, \ldots, a_k$ enteros positivos primos relativos entre sí. Para cada entero $l = 0, \ldots, a_1 - 1$, por t_l se denota el menor entero positivo congruente con l módulo a_1 que es representable por $a_2, \ldots, a_k$, entonces*

$$g(a_1, a_2, \ldots, a_k) = \max_{0 \leq l \leq a_1 - 1} \{t_l\} - a_1.$$

Para calcular el número de Frobenius de un conjunto dado, de acuerdo con el Teorema 2.2, basta con calcular los t_l. De esta forma si se tienen dos enteros primos relativos entre sí, a_1, a_2, el resultado obtenido por J.J. Sylvester es consecuencia del Teorema 2.2. En [2], Böcker y Lipták presentan un algoritmo para calcular los t_l.

3 El problema de Frobenius y bases de Gröbner

Sea $A = \{a_1, \ldots, a_k\}$ un conjunto de enteros primos positivos relativos entre sí. Ahora se construye un algoritmo utilizando bases de Gröbner para determinar si un número es representable por A, que se basa en los siguientes resultados.

Sean $F = \mathbb{Q}$ y ϕ la aplicación polinomial definida por

$$\phi : F[y_1, \ldots, y_k] \longrightarrow F[x]$$

$$y_j \longmapsto \quad x^{a_j}$$

para cada $j = 1, \ldots, k$ (Se demuestra que ϕ es un homomorfismo entre F-álgebras; esto es, un homomorfismo de anillos que a la vez es una transformación lineal entre

espacios vectoriales sobre F.) . Sea $I = \langle y_1 - x^{a_1}, \ldots, y_k - x^{a_k} \rangle$ un ideal en el anillo de polinomios $F[x, y_1, \ldots, y_k]$. Sea $im(\phi) = \{f(x) \in F[x] : \phi(g(y_1, \ldots, y_k)) = f(x);\ g(y_1, \ldots, y_k) \in F[y_1, \ldots, y_k]\}$; es decir, con $im(\phi)$ se denota el conjunto de todas las imágenes de la aplicación polinomial ϕ. Sea G la base de Gröbner reducida de I con respecto a un orden de eliminación con la variable x mayor que las variables y (ver [1]). Dado $f \in k[x_1, \ldots, x_n]$, con $N_G(f)$ se denota la forma normal de f con respecto a G.

Teorema 3.1. *$N \in \mathbb{Z}^+$ es representable por A si y sólo si $x^N \in im(\phi)$. Además, si $x^N = \phi(y_1^{\sigma_1} y_2^{\sigma_2} \cdots y_k^{\sigma_k})$, entonces*

$$N = a_1\sigma_1 + a_2\sigma_2 + \cdots + a_k\sigma_k$$

Demostración. Si N es representable por A, existen enteros no negativos $\sigma_1, \ldots, \sigma_k$ tales que, $N = a_1\sigma_1 + \cdots + a_k\sigma_k$, luego $x^N = x^{a_1\sigma_1 + \cdots + a_k\sigma_k} = (x^{a_1})^{\sigma_1} \cdots (x^{a_k})^{\sigma_k} = \phi(y_1^{\sigma_1}) \cdots \phi(y_k^{\sigma_k}) = \phi(y_1^{\sigma_1} \cdots y_k^{\sigma_k})$; es decir $x^N \in im(\phi)$. Recíprocamente, si $x^N \in im(\phi)$, del *lema 2.8.2* en [1], p. 107, se tiene que es la imagen de un producto de potencias $y_1^{\sigma_1} \cdots y_k^{\sigma_k} \in k[y_1, \ldots, y_k]$, donde σ_i es un entero no negativo para cada i; en otras palabras $x^N = \phi(y_1^{\sigma_1} \cdots y_k^{\sigma_k}) = (x^{a_1})^{\sigma_1} \cdots (x^{a_k})^{\sigma_k}$, de donde se concluye que $N = a_1\sigma_1 + \cdots + a_k\sigma_k$. □

Teorema 3.2. *$N \in \mathbb{Z}^+$ es representable por A si y sólo si $N_G(x^N) \in F[y_1, \ldots, y_k]$.*

Demostración. El enunciado es una conclusión directa del Teorema anterior y el *Corolario 2.4.5* en [1, p. 82]. □

De los Teoremas 3.1 y 3.2 se concluye que para determinar si un entero positivo N es representable por A, basta con calcular la forma normal del polinomio x^N con respecto a la base de Gröbner reducida G del ideal $I = \langle y_1 - x^{a_1}, \ldots, y_k - x^{a_k} \rangle$. Si alguna potencia de x (diferente de x^0) aparece en esta forma normal, se tiene que N no es representable, de lo contrario N es representable. En la siguiente sección se presenta un algoritmo basado en los resultados anteriores. Además, con base en dicho algoritmo, se construirá otra rutina para el cálculo del número de Frobenius de un conjunto de tres enteros primos relativos entre sí, y luego se da una para el cálculo del número de Frobenius en el caso general.

4 Un algoritmo para el caso de tres enteros primos relativos entre sí

El siguiente algoritmo, recoge las ideas anteriores y, determina si un entero es o no representable por un conjunto de enteros primos relativos entre sí. Esto se consigue mediante el algoritmo que se presenta a continuación

Algoritmo 1 (**Algoritmo represg**).

Entradas: $n \in \mathbb{Z}^+$ y $A = \{a_1, \ldots, a_k\}$ *conjunto de enteros primos relativos entre sí.*

Salida: *Un vector de* $k+1$ *componentes,* $v = [\sigma_0, \sigma_1, \ldots, \sigma_k]$.

Paso 1. $I := \langle y_1 - x^{a_1}, \ldots, y_k - x^{a_k} \rangle$.

Paso 2. *Se calcula la base de Gröbner reducida de* I.

Paso 3. *Se calcula la forma normal de* x^N *con respecto a la base de Gröbner* G, $t := N_G(x^n)$.

Paso 4. *Se determina el vector de los exponentes del término lider de* t. *Éste es el vector de salida,* $v = [\sigma_0, \sigma_1, \ldots, \sigma_k]$.

Este algoritmo recibe un entero positivo n, y una lista de k enteros positivos primos relativos entre sí. Retorna un vector de $k+1$ componentes. Si la primera componente del vector es igual a 0, entonces n es representable por los elementos de la lista; mientras que, si la primera componente es mayor que 0, el entero no es representable. Además, $N = \sigma_0 + a_1\sigma_1 + \cdots + a_k\sigma_k$. En el siguiente ejemplo se da una aplicación de este algoritmo.

Ejemplo 4.1. *Si el algoritmo se aplica a* 13 *con* $\{5, 7, 9\}$*; se obtiene*

```
represg(13,[5,7,9])
                                [1, 1, 1, 0]
```

lo que indica que 13 *no es representable por* $\{5, 7, 9\}$. *Pero, por ejemplo si se aplica a* 29

```
represg(29,[5,7,9])
                                [0, 3, 2, 0]
```

lo que indica que 29 *es representable por* $\{5, 7, 9\}$. *Es más,* $29 = (3)\cdot 5 + (2)\cdot 7 + (0)\cdot 9$.

La rutina **grob**, recibe cuatro enteros positivos a, b, n, k, y encuentra el conjunto de todos los enteros en el intervalo $M = [n, k]$ que se pueden escribir como combinación lineal de a y b con coeficientes positivos.

Algoritmo 2 (**Algoritmo grob**).

Entradas: $a, b, n, k \in \mathbb{Z}^+$.

Salida: *El conjunto de enteros en el intervalo* $M = [n, k]$ *que se pueden escribir como combinación lineal de* a *y* b *con coeficientes positivos.*

Paso 1. $I := \langle y_1 - x^a, y_2 - x^b \rangle$.

Paso 2. *Se calcula la base de Gröbner de* I.

Paso 3. *Se determinan los elementos del intervalo* M *que son combinación lineal de* a *y* b *con coeficentes positivos. Entonces para cada* $n \in M$*, se encuentra* $N_G(x^n)$.

Paso 4. *Se retorna el conjunto C de los elementos encontrados en el paso anterior.*

Ejemplo 4.2. *Siguiendo con el ejemplo anterior. Los números que son representables con coeficientes positivos en el intervalo* $[20, 52]$ *por las parejas* $\{5, 7\}$, $\{5, 9\}$ *y* $\{7, 9\}$*; respectivamente son*

```
>> grob(5,7,20,52)
{22, 24, 26, 27, 29, 31, 32, 33, 34, 36, 37, 38, 39, 40,
 41, 43, 44, 45, 46, 47, 48, 50, 51, 52}
>> grob(5,9,20,52)
{23, 24, 28, 29, 32, 33, 34, 37, 38, 39, 41, 42, 43, 44,
 46, 47, 48, 49, 50, 51, 52}
>> grob(7,9,20,52)
{23, 25, 30, 32, 34, 37, 39, 41, 43, 44, 46, 48, 50, 51, 52}
```

Según el Teorema 2.1, $f(a_1, a_2, a_3)$ se puede representar como combinación lineal de $\{a_1, a_2\}, \{a_2, a_3\}$ y $\{a_1, a_3\}$. Así, que para calcular f basta calcular los enteros que se puedan representar como combinación lineal de los tres conjuntos con coeficientes enteros positivos. Además, si se supone que $a_1 \leq a_2 \leq a_3$, de la desigualdad (3), está busqueda se puede restringir al intervalo $[a_1 + a_2 + a_3 - 1, a_2 + a_1a_3]$. Estas ideas se plasman en el siguiente algoritmo.

Algoritmo 3 (**Algoritmo g**).

Entradas: *a, b, c enteros positivos primos relativos entre sí.*

Salida: *El número de Frobenius $g(a, b, c)$.*

Paso 1. *$T := [a, b, c]$, esto organiza el conjunto de menor a mayor.*

Paso 2. *$k := T[2] + T[1] * T[3]$ y $w = a + b + c - 1$*

Paso 3. *Se calculan los elementos en el intervalo $[w, k]$ que son combinación lineal de las parejas $\{a, b\}, \{a, c\}$ y $\{b, c\}$ con coeficientes positivos; es decir se calcula*

$$D := grob(a, b, w, k) \cap grob(a, c, w, k) \cap grob(b, c, w, k).$$

Paso 4. *El número de Frobenius de a, b, c es el mayor entero $n - a - b - c$ que no sea representable por a, b, c, donde $n \in D$.*

Este algoritmo recibe tres enteros primos relativos entre sí y retorna el número de Frobenius.

Ejemplo 4.3. *En el ejemplo anterior*

```
>> g(5,7,9)
            13
```

Por lo tanto el número de Frobenius de $A = \{5, 7, 9\}$ es 13.

5 Algoritmo para el cálculo del número de Frobenius en el caso general

En esta sección se da un algoritmo para solucionar el caso general del problema de Frobenius.

Sea $A = \{a_1, a_2, \ldots, a_k\}$ un conjunto de enteros positivos primos relativos entre sí. Sea t_l el menor entero positivo congruente con l módulo a_1 representable por $a_2, a_3, \ldots, a_k$. En el Teorema 2.2 se tiene que

$$g(a_1, a_2, \ldots, a_k) = \max_{0 \leq l \leq a_1 - 1} t_l - a_1.$$

Entonces, calcular el número de Frobenius se reduce a determinar los números t_l y luego calcular $r = \max t_l$ para cada $l = 0, \ldots, a_1 - 1$. Note que a_1 puede ser cualquier elemento del conjunto A, en lo que sigue, para agilizar los cálculos, se supone que $a_1 = m = \min A$, ya que así se debe calcular un menor número de clases residuales. Como los t_l deben ser representables por $a_2, \ldots, a_k$ en esta situación se pueden utilizar las ideas vistas, en la sección anterior, para dar un algoritmo para encontrar los t_l. Sea $M = \max A$; entonces se tiene que

$$-1 \leq g(a_1, \ldots, a_k) \leq (m-1)(M-1) - 1;$$

lo que implica que

$$m - 1 \leq \max_{0 \leq l \leq a_1 - 1} \leq M(m-1).$$

Por lo tanto, para encontrar el número de Frobenius de A se deben encontrar los t_l en el intervalo $[m-1, mM-M]$. Para hacer esto se pueden utilizar las bases de Gröbner. En consecuencia se obtiene el algoritmo **grobfrob**:

Algoritmo 4 (**Algoritmo grobfrob**)**.**

Entradas: $A = \{a_1, \ldots, a_k\}$ *un conjunto de enteros positivos primos relativos entre sí.*

Salida: *El número de Frobenius de* A, $g(A)$.

Paso 1. *Se calculan* $m = \min A$ *y* $M = \max A$. *Considere a* $J = [m-1, M(m-1)]$ *un intervalo de enteros positivos.*

Paso 2. *Sea* $I = \langle y_i - x^{a_i} : a_i \neq m \rangle$ *y se calcula* G *la base de Gröbner reducida de* I.

Paso 3. *Para cada* $n \in J$ *se calcula la forma normal de* x^n *con respecto a* G. *Entonces se determina si* n *es representable por* $A - \{m\}$.

Paso 4. *Luego para cada* $1 \leq l \leq m-1$, *el menor* n *en el intervalo* J *que sea congruente con* l *módulo* a_1 *y representable por* $A - \{m\}$ *es* t_l.

Paso 5. *Por lo tanto* $g(A) = \max t_l - m$.

Así, el algoritmo **grobfrob** recibe un conjunto de enteros primos relativos entre sí y retorna el número de Frobenius de A.

6 Apéndice: Algoritmos

Los algoritmos que se presentan a continuación fueron utilizados en el desarrollo de este artículo. Estos algoritmos están implementados en el sistema de álgebra computacional MuPAD.

6.1 Algoritmo represg

Este algoritmo recibe un entero positivo n y una lista $L = [a_1, a_2,$
$\ldots, a_k]$ de enteros primos relativos entre sí. El algoritmo retorna un vector $v = [\sigma_0, \sigma_1, \ldots, \sigma_k]$, de $k+1$ componentes tal que, $n = a_1\sigma_1 + a_2\sigma_2 + \cdots + a_k\sigma_k + \sigma_0$, donde cada $\sigma_i \geq 0$. Entonces, si $\sigma_0 = 0$ se tiene que n es representable por $A = a_1, a_2, \ldots, a_k$ y si $\sigma_0 > 0$, entonces n no es representable.

```
represg:=proc(n,L) local i,j,lispol,G,t,k; begin
 k:=nops(L);
 lispol:=[];
 for j from 1 to k do
  lispol:=[op(lispol),poly(y.j-x^L[j],[x,y.i$ i = 1..k])]
 end_for;
 G:=groebner::gbasis(lispol,LexOrder);
 t:=degreevec(groebner::normalf(poly(x^n,[x,y.i $ i =
 1..k]),G,LexOrder));
end_proc;
```

6.2 Algoritmo grob

Dados a, b, n, k enteros positivos, el algoritmo *grob* retorna el conjunto de enteros que se pueden escribir como combinación lineal de a y b con coeficientes enteros positivos en el intervalo $I = [n, k]$.

```
grob:=proc(a,b,k,n) local i,j,lista,l,G; begin s:=k; C:={};
l:={a,b}; lista:=[op(l)]; k:=nops(lista);
G:=grobner(lista); t:=[];
for i from n  to s do
 t:=degreevec(groebner::normalf(poly(x^i,[x,y.i $ i =
 1..k]),G,LexOrder));
  if  t[1]=0 and t[2]*t[3]>0 then
       C:=_union({i},C);
  end_if
end_for;
C
end_proc;
```

6.3 Algoritmo g

Este algoritmo recibe tres enteros positivos primos relativos entre sí, a, b, c y retorna el número de Frobenius $g(a, b, c)$.

```
g:=proc(a,b,c) local i,j,D,L,n,f,M,T,k1,k,t,m;
begin M:={a,b,c};
T:=[op(M)]; k:=T[2]+T[1]*T[3]; w:=a+b+c-1;
D:=_intersect(grob(a,b,k,w),grob(a,c,k,w),grob(b,c,k,w));
L:=[op(D)]; n:=nops(L); k1:=nops(T); G:=grobner(T); t:=[];
 for i from 1 to n do
 m:=L[i]-a-b-c;
 t:=degreevec(groebner::normalf(poly(x^m,[x,y.i $ i =
 1..k1]),G,LexOrder));
 if t[1]>0 then
  f:=i;
 end_if
end_for;
L[f]-a-b-c;
end_proc;
```

6.4 Algoritmo grobfrob

Este algoritmo recibe una lista de enteros primos relativos entre sí y retorna el número de Frobenius de la lista.

```
grobfrob:=proc(L) local i,j,k,a1,ak,m,t,C,K,G,k1,s,jk;
begin
k:=nops(L); sort(L); a1:=L[1]; ak:=L[k]; delete L[1];
 G:=grobner(L);
k1:=nops(L); for i from 1 to (a1-1) do
  T:=[op(T),infinity];
end_for;
T[1]:=0;s:=0; K:={}; K:={i $ i = 1..a1-1};
 while nops(K)>0 do
  for m from a1-1 to (a1*ak-ak) do
   s:=m mod a1;
   if contains(K,s) then
    t:=degreevec(groebner::normalf(poly(x^m,[x,y.i $ i
    = 1..k1]),G, LexOrder));
    if t[1]=0 then
      T[s+1]:=m;
      K:=K minus {s};
      if K={} then
       m:=a1*ak;
```

```
      end_if;
    end_if;
   end_if;
 end_for;
end_while;
jk:=max(T)-a1;delete T;
jk;
end_proc;
```

Agradecimientos Este artículo hace parte del trabajo de investigación realizado por John Hermes Castillo para optar al título de Mágister en Matemáticas en la Universidad de Antioquia en mayo de 2006, ver [10]. Los autores queremos agradecer al grupo de investigación *Álgebra, Teoría de Números y Aplicaciones, ERM* por el apoyo brindado durante la elaboración de este trabajo.

Referencias

[1] William W. Adams and Philippe Loustaunau, An introduction to Gröbner bases, Graduate Studies in Mathematics, vol. 3, American Mathematical Society, Providence, RI, 1994. MR 1287608 (95g:13025)

[2] Sebastian Böcker and Zsuzsanna Lipták, The money changing problem revisited: Computing the Frobenius number in time o(k a_1), COCOON, 2005, pp. 965–974.

[3] Brauer, A. and Shockley, J.: On a problem of Frobenius, J. Reine Angew. Math. 211 (1962), 215–220. MR 0148606 (26 #6113)

[4] Selmer, E.: On the linear Diophantine problem of Frobenius, J. Reine Angew. Math. 293/294 (1977), 1–17.

[5] Selmer, E. and Beyer, Ö.: On the linear Diophantine problem of Frobenius in three variables, J. Reine Angew. Math. 301 (1978), 161–170. MR 0557015 (58 #27740)

[6] Sylvester, J.: Mathematical questions, with their solutions, Educational Times 41 (1884), 21.

[7] Ramírez-Alfonsín, J.: Complexity of the Frobenius problem, Combinatorica 16 (1996), 143-147.

[8] Roune, B.: Solving Thousand Digit Frobenius Problems Using Gröbner Bases, J. Symb. Comput. 43, 1 (Jan. 2008), 1-7.

[9] Ramírez-Alfonsín, J.: The Diophantine Frobenius Problem, Oxford Lectures Series in Mathematics and its Applications 30. Oxford University Press (2005).

[10] Castillo, J.: Los problemas de intercambio de moneda y de las estampillas de correo, Tesis de Maestría en Matemáticas, Universidad de Antioquia (2006).

Dirección de los autores
Gilberto García-Pulgarín — Universidad de Antioquia, Grupo de Investigación Álgebra, Teoría de Números y Aplicaciones
e-mail: gigarcia@member.ams.org
John Hermes Castillo Gómez — Universidad de Nariño, Departamento de Matemáticas y Estadística, Grupo de Investigación Álgebra, Teoría de Números y Aplicaciones
e-mail: jhcastillo@gmail.com