THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 15, Number 3/2014, pp. 310–314

EFFICIENT COMPUTATION OF THE NUMBER OF SOLUTIONS OF THE LINEAR DIOPHANTINE EQUATION OF FROBENIUS WITH SMALL COEFFICIENTS

Mugurel Ionuț ANDREICA, Nicolae ȚĂPUȘ

"Politehnica" University of Bucharest, Computer Science Department
Splaiul Independenței 313, 060042, Bucharest, Romania
E-mail: mugurel.andreica@cs.pub.ro

In this paper we present a novel approach for computing the number of solutions of the linear diophantine equation of Frobenius $a_1 \cdot x_1 + ... + a_N \cdot x_N = T$ when the coefficients a_1, ..., a_N are small. The proposed algorithm has a time complexity of the order of $O(N \cdot S \cdot \log(T))$, where $S = a_1 + ... + a_N$. The algorithm can also be implemented to run in $O(S \cdot \log(S) \cdot \log(T) + N \cdot S)$ time, which is more efficient when $\log(S) < N$. Note that an elementary operation in these cases consists of an arithmetic operation (e.g. addition, multiplication) applied on numbers of the order of magnitude of the result, which may be exponential in both N and S. A common situation consists of computing the result modulo a given number P, in which case the arithmetic operation is applied on numbers of the same order of magnitude as P.

Key words: Frobenius equation, knapsack, convolution, discrete Fourier transform.

1. INTRODUCTION

The linear diophantine equation of Frobenius is defined as

$$a_1 \cdot x_1 + ... + a_N \cdot x_N = T, \tag{1}$$

where a_1, ..., a_N and T are positive integer numbers. The unknowns are x_1, ..., x_N, which must also be non-negative integer numbers. Computing the number of solutions of the equation (1) (i.e. the number of different tuples $(x_1, ..., x_N)$ which satisfy the equation) is important in several areas of Mathematics. For instance, [14] discusses their applications to the theory of invariant cubature formulas. Applications of linear Diophantine equations in cryptography are discussed in [3, 9]. Computing the number of solutions of equation (1) also has applications in solving a related problem, that of computing the so-called *Frobenius number*, defined as the largest number T for which equation (1) has no solutions [8, 12].

In this paper we present a novel algorithm for computing the number of solutions of equation (1) in the case when the sum of the coefficients a_1, …, a_N is not too large. T, however, can be as large as we want, because our algorithm's dependency on T is only logarithmic. In Chapter 2 we present the first version of our algorithm, which uses a knapsack-like approach for each bit of the number T. The algorithm presented there performs $O(N \cdot S \cdot \log(T))$ elementary operations. In Chapter 3 we improve the algorithm for the case when $N > \log(S)$, achieving $O(S \cdot \log(S) \cdot \log(T) + N \cdot S)$ elementary operations. Note that in this case we trade-off the knapsack-like simplicity of the core routine of the algorithm presented in Chapter 2 for a more complicated, yet well studied, discrete convolution routine. In Chapter 4 we discuss related work and in Chapter 5 we conclude.

2. A KNAPSACK-BASED ALGORITHM FOR COMPUTING THE NUMBER OF SOLUTIONS

We will represent each unknown x_i ($1 \leq I \leq N$) by introducing $K+1$ independent unknowns, as follows:

$$x_i = x_{i,0} \cdot 2^0 + x_{i,1} \cdot 2^1 + ... + x_{i,K} \cdot 2^K. \tag{2}$$

We choose $K = \log_2(T)$, rounded down to the nearest integer. The unknowns $x_{i,j}$ ($1 \le i \le N$, $0 \le j \le K$) can be either 0 or 1 and are independent from one another. $x_{i,j}$ can be imagined as the bit from position j of x_i. It is obvious that no x_i will have more than $K+1$ bits, as that is the number of bits in the binary representation of T. The algorithm for computing the number of solutions of the Frobenius equation will take advantage of this independence, in the sense that the value of each unknown $x_{i,j}$ (0 or 1) can be considered independently of other unknowns $x_{i,j'}$ (for $j' \ne j$).

Let's denote the bits of T by $t_0, t_1, ..., t_K$, i.e.

$$T = t_0 \cdot 2^0 + t_1 \cdot 2^1 + ... + t_K \cdot 2^K. \tag{3}$$

The proposed algorithm will compute the following values:

$CNT(j,C)$ = the number of ways of choosing the values $x_{i,p}$ ($1 \le i \le N$, $0 \le p \le j$), such that the result of the equation matches the bits 0, ..., j of the number T and we have a „carry" equal to C (i.e. C is a number which influences the result for the next bits, starting from $j+1$). To be more precise, $CNT(j,C)$ = the number of ways of choosing the values $x_{i,p}$ ($1 \le i \le N$, $0 \le p \le j$), such that the obtained number is $t_0 \cdot 2^0 + t_1 \cdot 2^1 + ... + t_j \cdot 2^j + C \cdot 2^{j+1}$.

C will never exceed $S = a_1 + ... + a_N$.

Initially we will have $CNT(-1,0)=1$ and $CNT(-1,1 \le C \le S) = 0$.

Let's assume that we computed all the values $CNT(j,C)$ ($0 \le C \le S$) and we will see how to compute the values $CNT(j+1,C)$ ($0 \le C \le S$). We will use an auxiliary (multidimensional) array $CNTAUX$, where $CNTAUX(i,C)$= the number of ways of obtaining a „sum" equal to C ($0 \le C \le 2 \cdot S$) if we started from the values $CNT(j,*)$ and we considered only the unknowns $x_{p,j+1}$ so far ($1 \le p \le i$).

We will initialize $CNTAUX(0,C)=CNT(j,C)$ for $0 \le C \le S$ and $CNTAUX(0,C) = 0$ for $S+1 \le C \le 2 \cdot S$.

Then, we will consider the unknowns $x_{i,j+1}$, in increasing order of i ($1 \le i \le N$). We have:

$$CNTAUX(i,0 \le C \le a_i - 1) = CNTAUX(i-1,C). \tag{4}$$

For $a_i \le C \le 2 \cdot S$ we have:

$$CNTAUX(i,C) = CNTAUX(i-1,C) + CNTAUX(i-1, C-a_i). \tag{5}$$

The right-hand side terms from equation (5) correspond to the following two cases: $x_{i,j+1}=0$ and $x_{i,j+1}=1$. Note how equation (5) is similar to the equations used in 0-1 knapsack dynamic programming algorithms [15].

Finally, we will compute the values $CNT(j+1,*)$ from the values $CNTAUX(N,*)$. We have:

$$CNT(j+1,0 \le C \le S) = CNTAUX(N,2 \cdot C + t_{j+1}). \tag{6}$$

We consider $CNTAUX(N,C) = 0$ for $C > 2 \cdot S$.

The justification of equation (6) is as follows. So far we matched the bits $t_0, ..., t_j$ of T and now we must match the bit t_{j+1}. Thus, we can only consider values $CNTAUX(N,C)$ where the least significant bit of C is equal to t_{j+1}. After matching the bit $j+1$ of T with a number C, the remaining „carry" (for the bits $j+2, \ldots, K$) is equal to $C/2$ (integer division). Thus, we have $CNT(j+1,C/2)=CNTAUX(N,C)$, where bit 0 of C is equal to t_{j+1}.

The final result can be found in $CNT(K,0)$.

In order to analyze the time complexity of our algorithm, let's notice that a knapsack-like algorithm is run for each bit j. The knapsack-like algorithm takes $O(N \cdot S)$ time. Since there are $\log_2(T)$ bits, we obtain a time complexity of $O(N \cdot S \cdot \log(T))$. An elementary operation in our time complexity analysis consists of adding together two numbers which are of the order of magnitude of the final result. When we care about the exact number of solutions, then these numbers can be exponential in N and S. However, adding together two such numbers is proportional to the number of their digits, which is polynomial in N and S. There are cases when the numbers always remain small, such as when we want to compute the number of solutions modulo a given number P. In these cases the numbers involved are always of the order of magnitude of P (because the additions are performed modulo P). If we can consider the number of digits of P to be a constant value, then an elementary operation truly takes $O(1)$ time.

A trivial implementation of the presented algorithm would use $O(K \cdot S)$ memory. However, we can easily notice that the recurrences for the values $CNT(j,C)$ and $CNTAUX(j,C)$ never need values of the form $CNT(j',*)$ and $CNTAUX(j',*)$ with $j' < j-1$. Thus, it is always sufficient to maintain the last two rows of these two 2D arrays, reducing the required memory to $O(S)$.

3. OPTIMIZING THE ALGORITHM TO O(S·LOG(S)·LOG(T)+N·S)

The algorithm proposed in Chapter 2 can be optimized for the situation when $N>log_2(S)$. We will compute the same values. The values $CNT(0,*)$ and $CNT(1,*)$ will be computed as before. When computing the values $CNTAUX(N,*)$ ($j \geq 1$) we will use a different approach. Let's notice that

$$CNTAUX(N,C) = CNT(j,0)\cdot CNT(1,C) + CNT(j,1)\cdot CNT(1,C-1) + ... + CNT(j,C)\cdot CNT(1,0) = \sum_{x=0}^{C} CNT(j,x)\cdot CNT(1,C-x), \quad (7)$$

where we consider $CNT(*, S+1 \leq C \leq 2\cdot S)=0$.

Equation (7) defines a discrete convolution between the sequences $CNT(j,*)$ and $CNT(1,*)$. $CNTAUX(N,*)$ is computed by applying the convolution operation on these two sequences of size $S+1$. The convolution of two sequences of size $O(S)$ can be computed efficiently, by using the Discrete Fourier Transform, in $O(S\cdot\log(S))$ time [23]. After computing the values $CNTAUX(N,*)$ we obtain the values $CNT(j+1,*)$ from them as before. Note that in this case we do not need to compute the values $CNTAUX(i,*)$ for $i<N$, because we can compute $CNTAUX(N,*)$ directly by using the discrete convolution.

Thus, instead of using a knapsack-like algorithm for computing the values $CNTAUX(N,*)$ at each step, we use an algorithm for computing the convolution of two sequences. The overall time complexity of the modified algorithm is $O(S\cdot\log(S)\cdot\log(T)+N\cdot S)$ (the $O(N\cdot S)$ term comes from computing the values $CNT(1,*)$). The amount of memory used by the algorithm can be as low as $O(S)$, by using the same argument presented in the previous chapter.

When computing the convolution of two discrete sequences we will need to add and multiply numbers which are of the same order of magnitude as the final result (which we will call „large" numbers), or multiply and divide such numbers with „small" numbers (of the order of magnitude of N). We consider the addition of two „large" numbers or the multiplication and division of a „large" number with a „small" number to be equally complex and we will denote them as „simple" operations. The multiplication of two „large" numbers is more complex than the other operations and, thus, we will denote it as a „complex" operation. In the DFT-based discrete convolution algorithm applied for two sequences of length $O(S)$, there are $O(S\cdot\log(S))$ „simple" operations performed and only $O(S)$ „complex" operations. Because of this, we can conclude that an elementary operation in this second version of the algorithm is equivalent to an elementary operation in the first version of the algorithm.

4. RELATED WORK

The problem of computing the number of solutions for the Frobenius equation has been studied before in several papers. A common approach [4, 20, 22] is to find a closed-form function which computes the number of solutions of the equation when given the argument T. This approach uses the fact that this function is the sum of a polynomial of degree $N-1$ and a periodic function with period equal to $LCM(a_1, ..., a_N)$ (LCM = lowest common multiple). The coefficients of the polynomial are rational numbers, but the periodic part has a more complicated structure. The theory of generating functions is usually employed in order to explicitly find the function. In [6] the authors tackle only the polynomial part of the function, which can be computed in time exponential in N. The general case (both the polynomial and periodic parts) has been considered in [16], where an exact representation of the „number of solutions" function is given, which can be computed in a reasonable amount of time for moderately large values of the numbers a_i ($1 \leq i \leq N$) and N.

Even assuming that the "number of solutions" function is fully known, computing the exact number of solutions is not always easy, because computations involving real numbers will be performed. By comparison, our algorithm uses only integer numbers (at least in the first version of the algorithm).

Easily computable closed-form expressions of the „number of solutions" function for simple cases (e.g. $N=2$) have also been derived [19].

A very simple, yet inefficient, algorithm for computing the number of solutions of equation (1) is presented in [18]. The algorithm tries every possible value for x_1, ..., x_{N-1} (the value of x_N is then uniquely determined, if it exists). Obviously, this algorithm is exponential in T, while our algorithm's dependency on T is only logarithmic.

A more popular version of the problem studied in this paper [20] is to compute the so-called Frobenius number, i.e. the largest number T for which the Frobenius equation (1) does not have any solutions. Several algorithms have been proposed for this problem [7, 8, 12]. Reference[6] connects the Frobenius number problem and the problem of computing the number of solutions of equation (1) in a natural manner: the authors are interested in computing the largest number T for which equation (1) has a given number of solutions. Extensions to other types of equations have also been considered [11], as well as extensions to multiple dimensions [2].

Because of its popularity, the problem of finding the Frobenius number has also been considered as a useful educational tool in teaching discrete mathematics (particulary for the well studied case N=2) [5]. Moreover, generalizations of this problem have also been considered [10].

The convolution operation is a very useful tool in a wide variety of domains, both continuous and discrete [1, 17, 24]. A common method of efficiently computing the convolution of two discrete sequences is to apply the Discrete Fourier Transform on each sequence, multiply the obtained coefficients together (position by position), and then apply the Inverse Discrete Fourier Transform in order to obtain the result. Other fast convolution algorithms use the Discrete Fourier transform in other spaces than the space of complex numbers (e.g. in rings) [21].

5. CONCLUSIONS

In this paper we presented a novel algorithm for computing the number of solutions of the Frobenius equation $a_1 \cdot x_1 + ... + a_N \cdot x_N = T$ when the coefficients a_1, ..., a_N are small (so that their sum is not too large). Our algorithm works even when T is very large. N can also be moderately large, as long as the sum S of the coefficients is not too large and the product $N \cdot S$ is also not too large. An elementary operation in our algorithm consists of an arithmetic operation applied on two numbers which are of the same order of magnitude as the final result. When computing the result modulo a given number P, the numbers involved in the computations are of the same order of magnitude as P. However, in this case, the standard Discrete Fourier Transform cannot be used. Instead, the number-theoretic transform must be used, which imposes some constraints on the value of P; of course, these constraints can be overcome, at the expense of more complex implementations [13, 17].

REFERENCES

1. F. ABTAHI, R. N. ISFAHANI, A. REJALI, *Convolution on Weighted L^p-spaces of Locally Compact Groups*, Proceedings of the Romanian Academy, Series A, **13**, *2*, pp. 97–102, 2012.
2. J. AMOS, I. PASCU, V. PONOMARENKO, E. TREVINO, Y. ZHANG, *The Multidimensional Frobenius Problem*, Involve – a Journal of Mathematics, **4**, *2*, 2011.
3. M. R. K. ARIFFIN, N. A. ABU, *Linear Diophantine Equation Discrete Log Problem, Matrix Decomposition Problem and the AA_β-Cryptosystem*, IACR Cryptology ePrint Archive, 2011.
4. A. BADRA, *Frobenius Number of a Linear Diophantine Equation*, Lecture Notes in Pure and Applied Mathematics, **231**, pp. 23–36, 2003.
5. M. BECK, *How to Change Coins, M&M's, or Chicken Nuggets: The Linear Diophantine Problem of Frobenius*, in *Resources* for Teaching Discrete Mathematics: Classroom Projects, History Modules, and Articles (B. Hopkins, ed.), pp. 65–74. Mathematical Association of America, 2009.
6. M. BECK, I. M. GESSEL, T. KOMATSU, *The Polynomial Part of a Restricted Partition Function related to the Frobenius Problem*, The Electronic Journal of Combinatorics, **8**, *1*, 2001.
7. M. BECK, S. ROBBINS, *Computing the Continuous Discretely*, Springer, 2009.
8. D. BEIHOFFER, J. HENDRY, A. NIJENHUIS, S. WAGON, *Faster Algorithms for Frobenius Numbers*, The Electronic Journal of Combinatorics, **12**, 2005.
9. D. BISHOP, *Introduction to Cryptography with Java Applets*, Jones and Bartlett Publishers, 2003.
10. A. BROWN, E. DANNENBERG, J. FOX, J. HANNA, K. KECK, A. MOORE, Z. ROBBINS, B. SAMPLES, J. STANKEWICZ, *On a Generalization of the Frobenius Number*, Journal of Integer Sequences, **13**, 2010.
11. M. DELGADO, J. C. ROSALES, *On the Frobenius Number of a Proportionally Modular Diophantine Inequality*, Portugaliae Mathematica - Nova Serie, **63**, *4*, pp. 415–425, 2006.
12. D. EINSTEIN, D. LICHTBLAU, A. STRZEBONSKI, S. WAGON, *Frobenius Numbers by Lattice Point Enumeration*, Integers, **7**, *1*, 2007.

13. W. B. HART, G. TORNARIA, M. WATKINS, *Congruent Number Theta Coefficients to 10^{12}*, Algorithmic Number Theory: 9th International Symposium ANTS-IX, Springer-Verlag, 2010, pp. 186–200.
14. M. I. ISRAILOV, *Numbers of Solutions of Linear Diophantine Equations and Their Applications in the Theory of Invariant Cubature Formulas*, Siberian Mathematical Journal, **22**, *2*, pp. 260–273, 1981.
15. H. KELLERER, U. PFERSCHY, D. PISINGER, *Knapsack Problems*, Springer-Verlag, 2004.
16. T. KOMATSU, *On the Number of Solutions of the Diophantine Equation of Frobenius – General Case*, Mathematical Communications, **8**, pp. 195-206, 2003.
17. L. M. LEIBOWITZ, *Fast Convolution by Number Theoretic Transforms*, Naval Research Laboratory, Washington D.C., 1975.
18. R. MAHMOUDVAND, H. HASSANI, A. FARZANEH, G. HOWELL, *The Exact Number of Nonnegative Integer Solutions for a Linear Diophantine Inequality*, IAENG International Journal of Applied Mathematics, **40**, *1*, 2010.
19. T. POPOVICIU, *Asupra unei Probleme de Partiție a Numerelor*, Studii şi Cercetări Ştiințifice, **4**, pp. 7–58, 1953.
20. J. L. RAMIREZ ALFONSIN, *The Diophantine Frobenius Problem*, Oxford Lecture Series in Math. and its Appl., **30**, Oxford Univ. Press, Oxford, 2005.
21. A. SCHÖNHAGE, V. STRASSEN, *Schnelle Multiplikation großer Zahlen*, Computing, **7**, pp. 281–292, 1971.
22. S. SERTÖZ, *On the Number of Solutions of a Diophantine Equation of Frobenius*, Discrete Mathematics and Applications, **8**, *2*, pp. 153–162, 1998.
23. D. SUNDARARAJAN, *The Discrete Fourier Transform: Theory, Algorithms and Applications*, World Scientific Publishing, 2001.
24. G. ZBĂGANU, *On Infinite Bernoulli Convolutions*, Proceedings of the Romanian Academy, Series A, **7**, *3*, 2006.

Received May 20, 2013