On the Complexity of Integer Programming in the Blum-Shub-Smale Computational Model

Valentin E. Brimkov[1] and Stefan S. Dantchev[2]

[1] Eastern Mediterranean University,
Famagusta, P.O. Box 95, TRNC, Via Mersin 10, Turkey
brimkov.as@mozart.emu.edu.tr
[2] BRICS - Basic Research in Computer Science,
Centre of the Danish National Research Foundation,
University of Aarhus, Department of Computer Science,
Ny Munkegade, Bldg. 540, DK-8000 Aarhus C., Denmark
dantchev@brics.dk

Abstract. In the framework of the Blum-Shub-Smale real number model [3], we study the *algebraic complexity* of the integer linear programming problem ($\mathrm{ILP}_{\mathbf{R}}$) : Given a matrix $A \in \mathbf{R}^{m\times n}$ and vectors $b \in \mathbf{R}^m$, $d \in \mathbf{R}^n$, decide whether there is $x \in \mathbf{Z}^n$ such that $Ax \leq b$, where $\mathbf{0} \leq x \leq d$. The main contributions of the paper are the following:
- An $O(m \log ||d||)$ algorithm for $\mathrm{ILP}_{\mathbf{R}}$, when the value of n is fixed. As a corollary, we obtain under the same restriction a tight algebraic complexity bound $\Theta\left(\log \frac{1}{a_{\min}}\right)$, $a_{\min} = \min\{a_1, \ldots, a_n\}$, for the knapsack problem ($\mathrm{KP}_{\mathbf{R}}$) : Given $a \in \mathbf{R}^n_+$, decide whether there is $x \in \mathbf{Z}^n$ such that $a^T x = 1$. We achieve these results in particular through a careful analysis of the algebraic complexity of the Lovász' basis reduction algorithm and the Kannan-Bachem's Hermite normal form algorithm, which may be of interest in its own.
- An $O\left(mn^5 \log n\,(n + \log ||d||)\right)$ depth *algebraic decision tree* for $\mathrm{ILP}_{\mathbf{R}}$, for every m and n.
- A new lower bound for 0/1 $\mathrm{KP}_{\mathbf{R}}$. More precisely, no algorithm can solve 0/1 $\mathrm{KP}_{\mathbf{R}}$ in $o(n \log n)\, f(a_1, \ldots, a_n)$ time, even if f is an *arbitrary continuous* function of n variables. This result appears as an alternative to the well-known Ben-Or's bound $\Omega(n^2)$ [1] and is independent upon it.
Keywords: *Algebraic complexity, Complexity bounds, Integer programming, Knapsack problem*

1 Introduction

We study the *algebraic complexity* of the following integer linear programming (ILP) problem:

($\mathrm{ILP}_{\mathbf{R}}$) Given a matrix $A \in \mathbf{R}^{m\times n}$ and vectors $b \in \mathbf{R}^m, d \in \mathbf{R}^n$, decide whether there is $x \in \mathbf{Z}^n$ such that $Ax \leq b$, where $\mathbf{0} \leq x \leq d$.

The input entries are arbitrary *real* numbers and, accordingly, the adopted model of computation is a *real number model*. This kind of model has been traditionally

J. van Leeuwen et al. (Eds.): IFIP TCS 2000, LNCS 1872, pp. 286–300, 2000.
© Springer-Verlag Berlin Heidelberg 2000

used in scientific computing, computational geometry, and (although not explicitly) numerical analysis (see, e.g., [18,19,22]). In our study we conform mainly to the model presented in [3], known as the BSS-model (named after its creators Blum, Shub and Smale). In the BSS-model, the assumption is that all the real numbers in the input have unit size, and the basic algebraic operations $+, -, *, /$ and the relation $\leq$ are executable at unit cost. Thus the algebraic complexity of a computation on a problem instance is the number of operations and branchings performed to solve the instance. For more details on the BSS-model and complexity theory over arbitrary rings, we refer to [3]. We notice that in this new theory, aimed at providing a complexity framework for disciplines like those mentioned above, an important issue is seen in the comparison of results over the reals with classical results over the integers, which may help elucidate some fundamental concepts, like computability and complexity.

At this point it is important to mention that the requirement for bounded domain (i.e., $\mathbf{0} \leq x \leq d$) is essential and dictated by the very nature of the problem, namely by the fact that the coefficients may be *irrational* numbers. In such a case, a problem with unbounded domain may be, in general, *undecidable*, as shown in [4].

In a classical setting, integer linear programming with integer or rational inputs is among the best-studied combinatorial problems. A substantial body of literature, impossible to report here, has been developed on the subject. In particular, it is well-known that ILP is NP-complete [8]. Comparatively less is known about the complexity of $\mathrm{ILP}_{\mathbf{R}}$ in the framework of the BSS-model. Some related results are reported in [3,1,17,7,4]. In [2] Blum et al. pose the problem of studying the complexity of an important special case of $\mathrm{ILP}_{\mathbf{R}}$, known as the "real" knapsack problem:

$(\mathrm{KP}_{\mathbf{R}})$ Given $a \in \mathbf{R}^n_+$, decide if there is $x \in \mathbf{Z}^n$ such that $a^T x = 1$.

With the present paper we take a step towards determining $\mathrm{ILP}_{\mathbf{R}}$'s and $\mathrm{KP}_{\mathbf{R}}$'s complexity. Our main contributions are the following.

1. An $O(m \log ||d||)$ algorithm for $\mathrm{ILP}_{\mathbf{R}}$ when the value of n is fixed (Section 2).
   A similar result is known for the integer case, namely, the well-known Lenstra's algorithm for ILP of a fixed dimension n [12]. Our algorithm consists of two stages: a reduction of the given real input to an integer input determining the same admissible set, followed by an application of Lenstra's algorithm. The first stage involves simultaneous Diophantine approximation techniques, while the second employs two well-known algorithms: the Lovász' basis reduction algorithm [13] and the Kannan-Bachem's Hermite normal form algorithm [10]. It is straightforward to obtain an upper time complexity bound that is *quadratic* in $\log ||d||$. Our more detailed analysis reveals that the actual complexity of the latter two algorithms (and, as a consequence, of the entire algorithm) is *linear* in $\log ||d||$.

Applied to the knapsack problem $\mathrm{KP}_{\mathbf{R}}$ of fixed dimension n, our algorithm has complexity $O\left(\log \frac{1}{a_{\min}}\right)$, $a_{\min} = \min\{a_1, \ldots, a_n\}$, and turns out to be *optimal*.
In view of the fact that the Lovász' basis reduction algorithm and the Kannan-Bachem's Hermite normal form algorithm are fundamental and very important combinatorial algorithms, we believe that their algebraic complexity analysis within the BSS-model may be of interest in its own.

2. An $O\left(mn^5 \log n\,(n + \log ||d||)\right)$ depth *algebraic decision tree* for $\mathrm{ILP}_{\mathbf{R}}$, for every m and n (i.e., in a model which is nonuniform with respect to them) (Section 3).
This result is in the spirit of the well-known Meyer auf der Heide's $n^4 \log n + O\left(n^3\right)$ depth *linear decision tree* for 0/1 $\mathrm{KP}_{\mathbf{R}}$ (i.e., $\mathrm{KP}_{\mathbf{R}}$ with $x \in \{0,1\}^n$) [14].
3. A new lower bound for 0/1 $\mathrm{KP}_{\mathbf{R}}$. More precisely, no algorithm can solve 0/1 $\mathrm{KP}_{\mathbf{R}}$ in $o\left(n \log n\right) f\left(a_1, \ldots, a_n\right)$ time, even if f is an *arbitrary continuous* function of n variables (Section 4).
This result appears as an alternative to the well-known Ben-Or's bound $\Omega(n^2)$ [1] and is independent upon it, in the sense that neither of both results is superior to or implies the other.

2 Analysis of the Basic Algorithms

In this section, we analyze the Lovász lattice basis reduction algorithm [13] and the Kannan and Bachem's Hermite normal form algorithm [10]. It is well-known that these are polynomial within the classical computational model. This implies that, within the BSS model, they are polynomial with respect to the dimensions m and n of the input matrices and the maximal bit-size S of their integer (or rational) entries. Our deeper analysis shows that they are *linear* in S.

2.1 Some Useful Facts

In this section, we state some simple facts about vectors and matrices with rational entries of bit-size at most S. Although trivial, these facts will be instrumental in analyzing the algorithms in the next sections.

1. Let a be a non-zero rational number. Then $1/2^S \leq |a| \leq 2^S$.
2. Let b_1, b_2 be non-orthogonal n-dimensional rational vectors. Then $1/2^{2nS} \leq |\langle b_1, b_2 \rangle| \leq n2^{2S}$.
3. Let B be a non-singular $n \times n$ rational matrix. Then $1 \Big/ 2^{n^2 S} \leq |\det(B)| \leq n!2^{nS}$.
4. Let B_i be an $n \times i$ rational matrix of rank i, $i \leq n$. Then $1 \Big/ 2^{2n^2 S} \leq \left|\det\left(B^T B\right)\right| \leq n!2^{n(\log n + 2S)}$.

2.2 Lovász Lattice Basis Reduction Algorithm

In the description and analysis of the algorithm we follow [9]. The input consists of linearly independent vectors $b_1, b_2, \ldots b_n \in \mathbf{Q}^n$, considered as a basis for a lattice L. The algorithm transforms them iteratively. At the end, they form a basis for L which is reduced in the Lovázs sense.

First we recall some definitions, then describe the Lovász lattice basis reduction algorithm, itself. With a basis $b_1, b_2, \ldots b_n$, we associate the orthogonal system $b_1^*, b_2^*, \ldots b_n^*$, where b_i^* is the component of b_i which is orthogonal to $b_1, b_2, \ldots b_{i-1}$. The vectors $b_1^*, b_2^*, \ldots b_n^*$ can be computed by Gram-Schmidt orthogonalization:

$b_1^* = b_1$,
$b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \ \ 2 \le i \le n$,
where $\mu_{i,j} = \langle b_i, b_j^* \rangle \Big/ \left\| b_j^* \right\|^2$.

The basis $b_1, b_2, \ldots b_n$ is *size-reduced* if all $|\mu_{i,j}| \le \frac{1}{2}$. Given an arbitrary basis $b_1, b_2, \ldots b_n$, we can transform it into a size-reduced basis with the same Gram-Schmidt orthogonal system, as follows:

For every i from 2 to n; For every j from $i-1$ to 1;
Set $b_i := b_i - \lceil \mu_{i,j} \rfloor \, b_j$ and update $\mu_{i,k}$ for $1 \le k \le i-1$, by setting $\mu_{i,k} = \mu_{i,k} - \lceil \mu_{i,j} \rfloor \, \mu_{j,k}$.

Now, we can describe a variant of the Lovász lattice basis reduction algorithm.

1. *Initiation.* Compute the Gram-Schmidt quantities $\mu_{i,j}$ and b_i^* for $1 \le j < i \le n$. Size-reduce the basis.
2. *Termination condition.* If $\|b_i^*\|^2 \le 2 \left\| b_{i+1}^* \right\|^2$ for $1 \le i \le n-1$, then stop.
3. *Exchange step.* Choose the smallest i such that $\|b_i^*\|^2 > 2 \left\| b_{i+1}^* \right\|^2$. Exchange b_i and b_{i+1}. Update the Gram-Schmidt quantities. Size-reduce the basis. Go to 2.

For completeness, we give formulae for updating the Gram-Schmidt quantities in step 3:

$$\|b_i^*\|_{new}^2 = \left\| b_{i+1}^* \right\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2$$
$$\left\| b_{i+1}^* \right\|_{new}^2 = \|b_i^*\|^2 \left\| b_{i+1}^* \right\|^2 \Big/ \|b_i^*\|_{new}^2$$
$$\mu_{i+1,i}^{new} = \mu_{i+1,i} \|b_i^*\|^2 \Big/ \|b_i^*\|_{new}^2$$
$$\begin{pmatrix} \mu_{i,j}^{new} \\ \mu_{i+1,j}^{new} \end{pmatrix} = \begin{pmatrix} \mu_{i+1,j} \\ \mu_{i,j} \end{pmatrix} \text{ for } 1 \le j \le i-1$$
$$\begin{pmatrix} \mu_{j,i}^{new} \\ \mu_{j,i+1}^{new} \end{pmatrix} = \begin{pmatrix} 1 & \mu_{i+1,i}^{new} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -\mu_{i+1,i} \end{pmatrix} \begin{pmatrix} \mu_{j,i} \\ \mu_{j,i+1} \end{pmatrix} \text{ for } i+2 \le j \le n.$$

The other $\|b_i^*\|^2$'s and $\mu_{i,j}$'s do not change.

After termination of the above algorithm, we have a size-reduced basis for which $\|b_i^*\|^2 \le 2 \left\| b_{i+1}^* \right\|^2$, $1 \le i \le n-1$. We call such a basis *reduced in the Lovász sense.* (There are other definitions of this concept in the literature, but

for our purposes they are essentially equivalent.) Important properties of such a basis are

$$\|b_1\| \leq 2^{\frac{n-1}{2}} \|(shortest\ vector\ in\ L)\|,$$

and

$$\prod_{i=1}^{n} \|b_i\| \leq 2^{\frac{n(n-1)}{4}} \det(L). \tag{1}$$

Let us analyze the running time of steps 2 and 3. Consider the function

$$F(b_1^*, b_2^*, \ldots b_n^*) := \prod_{i=1}^{n} \|b_i^*\|^{2(n-i)} = \prod_{i=1}^{n-1} det\left(B_i^T B_i\right),$$

where B_i is a matrix having $b_1, b_2, \ldots b_i$ as column vectors. No size-reduction operation changes F, as it does not change the $\|b_i^*\|$'s. After an exchange step, we obtain

$$\frac{F_{new}}{F} = \frac{\|b_i^*\|_{new}^2}{\|b_i^*\|^2} = \frac{\left\|b_{i+1}^*\right\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2}{\|b_i^*\|^2} < \frac{3}{4}. \tag{2}$$

It is not hard to see that every iteration of steps (2-3) consists of $O\left(n^2\right)$ basic arithmetic operations (because of the size-reduction, an updating needs only $O(n)$ such operations). The only problem might be the rounding operations $\lceil . \rfloor$ performed during the size-reductions. We observe that the absolute values of their arguments are at most $O\left(n\mu_{i+1,i}^{new}\right)$. Then the time needed for one such an operation is

$$O\left(\log n + \log \mu_{i+1,i}^{new}\right) = O\left(\log n + \log\left(\|b_i^*\|^2 \Big/ \|b_i^*\|_{new}^2\right)\right).$$

Thus, the time complexity of one iteration is

$$O\left(n^2\left(\log n + \log\left(\|b_i^*\|^2 \Big/ \|b_i^*\|_{new}^2\right)\right)\right) = O\left(n^2\left(\log n + \log \frac{F}{F_{new}}\right)\right).$$

Then the time complexity of all iterations is

$$O\left((\#iterations)\, n^2 \log n + n^2 \log \frac{F_{start}}{F_{end}}\right). \tag{3}$$

Because of (2), the number of iterations is $O\left(\log \frac{F_{start}}{F_{end}}\right)$, so that the overall complexity of steps 2 and 3 is

$$O\left(n^2 \log n \log \frac{F_{start}}{F_{end}}\right). \tag{4}$$

What remains is to estimate the running time of step 1 and the ratio $\frac{F_{start}}{F_{end}}$.

By definition, $\mu_{i,j}$ ($1 \leq j \leq i-1$) can be considered as a solution of the following linear system

$$\begin{pmatrix} \langle b_1, b_1 \rangle & & \langle b_1, b_{i-1} \rangle \\ & \ddots & \\ \langle b_{i-1}, b_1 \rangle & & \langle b_{i-1}, b_{i-1} \rangle \end{pmatrix} \begin{pmatrix} \mu_{i,1} \\ \vdots \\ \mu_{i,i-1} \end{pmatrix} = \begin{pmatrix} \langle b_1, b_i \rangle \\ \vdots \\ \langle b_{i-1}, b_i \rangle \end{pmatrix},$$

for $2 \leq i \leq n$. From here and fact 4 from Section 2.1, it is not difficult to deduce that before the size-reduction phase of step 1, $\|\mu_{i,j}\| \leq 2^{O(Sn^2)}$. The size-reduction itself takes $O\left(n^2\right)$ $\lceil . \rfloor$ operations on these numbers, so that the time complexity of step 1 is clearly $O\left(Sn^4\right)$.

Lastly, we need to estimate F_{start} and F_{end}. F_{start} is a product of the determinants of $n-1$ matrices $B_i^T B_i$ where $1 \leq i \leq n-1$, so that

$$|F_{start}| \leq 2^{O\left(n^2(\log n + S)\right)}.$$

To estimate F_{end}, let us observe that any of the vectors $b_1^{end}, b_2^{end}, \ldots b_n^{end}$ is an *integer* linear combination of $b_1^{start}, b_2^{start}, \ldots b_n^{start}$. Therefore, for $1 \leq i \leq n$, we have $B_i^{end} = B_n^{start} A_i$, where A_i is an $n \times i$ integer matrix. This implies

$$\det\left(B_i^{end^T} B_i^{end}\right) = \det\left(A_i^T A_i\right) \det\left(B_i^{start^T} B_i^{start}\right) \geq 1 \Big/ 2^{2n^2 S},$$

by fact 4 from Section 2.1. Consequently, $F_{end} \geq 1 \Big/ 2^{O\left(n^3 S\right)}$. Thus we obtain $\log \frac{F_{start}}{F_{end}} = O\left(n^3 S\right)$. Hence, the overall complexity of the Lovász basis reduction algorithm is $O\left(Sn^5 \log n\right)$.

Finally, we will prove that the bit-size of the entries of the reduced basis is $O\left(Sn^3\right)$. Let us recall inequality (1):

$$\prod_{i=1}^{n} \left\|b_i^{end}\right\| \leq 2^{\frac{n(n-1)}{4}} \det(L) = 2^{\frac{n(n-1)}{4}} \left|\det\left(B_n^{start}\right)\right|,$$

and denote with a the least common multiple of all entries of B_n^{start}. Note that the bit-size of a is $O\left(Sn^2\right)$. Since b_i^{end}'s are integer linear combinations of b_i^{start}'s, the vectors ab_i^{end} are integer. Therefore, we have

$$\prod_{i=1}^{n} \left\|ab_i^{end}\right\| \leq 2^{\frac{n(n-1)}{4}} a^n \left|\det\left(B_n^{start}\right)\right| \leq 2^{\frac{n(n-1)}{4}} 2^{Sn^3} n! 2^S = 2^{O\left(Sn^3\right)}.$$

Thus, every entry of aB_n^{end} is of bit-size $O\left(Sn^3\right)$ and so is every entry of B_n^{end}. In terms of the adopted denotations, we have proved the following lemma.

Lemma 1. *The algebraic complexity of Lovász' basis reduction algorithm is $O(Sn^5 \log n)$, and the bit-size of the entries in the reduced basis is $O(Sn^3)$.*

2.3 Kannan and Bachem's Hermite Normal Form Algorithm

In our description we follow [21]. The input for the algorithm is an $m \times n$ ($m \leq n$) integer matrix A of full rank. The algorithm uses the matrix

$$A' = \left(A \left| \begin{array}{ccc} M & & \\ & \ddots & \\ & & M \end{array} \right. \right),$$

where M is the absolute value of some nonsingular $m \times m$ minor of A. A' has the same Hermite normal form as A. The algorithm consists of the following five steps:

1. Cause all the entries of the matrix A to fall into the interval $[0, M)$, by adding to the first n columns of A' proper integer multiples of the last n columns;
2. For k from 1 to m do 3-4;
3. If there are $i \neq j$, $k \leq i, j \leq n + k$, such that $a'_{k,i} \geq a'_{k,j} > 0$, then subtract from the ith column the jth one multiplied by $\left\lfloor \frac{a'_{k,i}}{a'_{k,j}} \right\rfloor$. Then reduce the ith column modulo M. Go to 3;
4. Exchange the kth column and the only column with $a'_{k,i} > 0$;
5. For every i from 2 to n; for every j from 1 to $i - 1$, add an integer multiple of the ith column to the jth one, to get $a'_{i,i} > a'_{i,j} \geq 0$.

In order to show that the time complexity is polynomial in m, n and linear in S, we need to analyze step 3. For this, we introduce the function

$$F\left(a'_{k,k}, a'_{k,k+1}, \ldots a'_{k,n+k}\right) := \prod_{\substack{k \leq i \leq n+k \\ a'_{k,i} > 0}} a'_{k,i}.$$

After one iteration of step 3, we have

$$\frac{F_{new}}{F} = \frac{a'_{k,i} - \left\lfloor \frac{a'_{k,i}}{a'_{k,j}} \right\rfloor a'_{k,j}}{a'_{k,i}},$$

which implies both $\frac{F_{new}}{F} < \frac{1}{2}$ and $\frac{F_{new}}{F} < \frac{a'_{k,j}}{a'_{k,i}}$. It is not hard to see that one iteration of step 3 can be performed in $O\left(m \log \frac{a'_{k,i}}{a'_{k,j}}\right) = O\left(m \log \frac{F}{F_{new}}\right)$ time. So, step 3 takes $O\left(m \log \frac{F_{start}}{F_{end}}\right)$ time. Since $F_{start} < M^{n+1}$, $F_{end} \geq 1$, and $M = O\left(m! 2^{mS}\right)$ by fact 3 of Section 2.1, the overall running time of step 3 is $O\left(nm\left(\log m + S\right)\right)$. Then the complexity of the Kannan-Bachem's algorithm is $O\left(nm^2\left(\log m + S\right)\right)$.

Since all the resulting integers are smaller than M, their bit-size is $O\left(Smn\right)$. Thus we have proved the following lemma.

Lemma 2. *Let A be an $m \times n$ ($m \leq n$) integer matrix of full rank. Then the algebraic complexity of the Kannan-Bachem's algorithm that reduces A into its Hermite normal form, is $O(m^2 n(\log m + S))$.*

3 Basic Results about $\mathrm{ILP}_{\mathbf{R}}$

In this section we use the analysis of the algorithms from the previous section to obtain the first two of the results announced in the Introduction.

To solve $\mathrm{ILP}_{\mathbf{R}}$ algorithmically within the BSS-model, we follow the idea of our method developed in [4]. There its complexity was analyzed within a strengthened version of the BSS-model, in which the floor operation $\lfloor . \rfloor$ is considered as a basic one, executable at unit cost. Here we will apply and analyze it within the standard BSS-model.

The algorithm employs in one of its stages the well-known algorithm for finding a simultaneous Diophantine approximation to a given rational vector. In particular, we will use the following lemma.

Lemma 3. *(see, e.g., [21, Corollary 6.4c]) There exists a polynomial algorithm which, given a vector $a \in \mathbf{Q}^n$ and a rational number ε, $0 < \varepsilon < 1$, finds an integral vector p and an integer q such that $||a - (1/q)p|| < \varepsilon/q$, and $1 \leq q \leq 2^{n(n+1)/4}\varepsilon^{-n}$.*

Now we pass to the description of our algorithm for $\mathrm{ILP}_{\mathbf{R}}$. As mentioned before, it consists of two main stages. In the first stage, the algorithm reduces the constraints with real coefficients to constraints with integer coefficients determining the same admissible set. The first step of this reduction is the substitution of a given real vector with an appropriate rational vector, justified by the following lemma[1].

Lemma 4. *Given a vector $\alpha \in \mathbf{R}^n$ with $|\alpha_j| \leq 1, j = 1, 2, \ldots, n$, and $D \in \mathbf{Z}_+$, there exists an $O(n^4 \log n(n + \log D))$ algorithm that finds $p \in \mathbf{Z}^n$ and $q \in \mathbf{Z}_+$ such that $|\alpha_j - p_j/q| < 1/(qD)$, $j = 1, 2, \ldots, n$, and $1 \leq q \leq \lceil 2^{n(n+5)/4} D^n \rceil$.*

Proof First we describe the algorithm finding $p \in \mathbf{Z}^n$ and $q \in \mathbf{Z}_+$ with the required properties. It consists of two basic steps.

1. For each α_j, $1 \leq j \leq n$, find the closest rational fraction a_j with denominator $G = \lceil 2^{n(n+5)/4} D^{n+1} \rceil$.
2. Apply the algorithm from Lemma 3 with input $a = (a_1, \ldots, a_n) \in \mathbf{Q}^n$ and $\varepsilon = 1/(2D)$. The output is a vector $p \in \mathbf{Z}^n$ and an integer $q \in \mathbf{Z}_+$ with $||a - (1/q)p|| < 1/(2qD)$ and $1 \leq q \leq \lceil 2^{n(n+5)/4} D^n \rceil$.

Clearly, $|\alpha_j - a_j| \leq 1/(2G)$. Then we have

$$\left|\alpha_j - \frac{p_j}{q}\right| \leq |\alpha_j - a_j| + \left|a_j - \frac{1}{q}p_j\right| \leq |\alpha_j - a_j| + \left\|a - \frac{1}{q}p\right\| <$$

$$< \frac{1}{2G} + \frac{1}{2qD} \leq \frac{1}{2.\lceil 2^{n(n+5)/4} D^n \rceil . D} + \frac{1}{2qD} \leq \frac{1}{qD},$$

i.e., the obtained vector p and integer q are as desired.

[1] To reduce the given real constraints to an equivalent set of integer constraints, one can also use the approach from [20].

Now we evaluate the algorithm's complexity. For a given real number α_j, the closest rational fraction with denominator $G = \lceil 2^{n(n+5)/4} D^{n+1} \rceil$ can be found in time $O(\log G) = O(n^2 + n \log D)$. Thus the overall time complexity of Step 1 is $O(n^3 + n^2 \log D)$.[2]

Step 2 involves the simultaneous Diophantine approximation algorithm applied to the particular class of inputs $a \in \mathbf{Q}^n$, $\varepsilon = 1/(2D)$ obtained in Step 1. As a matter of fact, this algorithm is a specialization of the Lovász basis reduction algorithm, applied to a matrix of the form

$$\begin{pmatrix} 1 & & & a_1 \\ & 1 & & a_2 \\ & & \ddots & \vdots \\ & & 1 & a_n \\ & & & 1/G \end{pmatrix}$$

where $a_1, a_2, \ldots, a_n$ are rational numbers, all of them with the same denominator $G = 2^{O(n(n+\log D))}$. The following bound on the number of iterations holds.

Lemma 5. *(see [4, Lemma 4.4]) In Step 2 of the algorithm,* $O(\log \frac{F_{start}}{F_{end}}) = O(n^3 + n^2 \log D)$ *iterations are performed.*

From (4) and Lemma 5, we obtain that the time complexity of Step 2 is

$$O\left(n^2 \log n \log \frac{F_{start}}{F_{end}}\right) = O(n^2 \log n(n^3 + n^2 \log D)) =$$

$$= O(n^4 \log n(n + \log D)).$$

Thus the overall time complexity of the algorithm is $O(n^4 \log n(n + \log D))$. □

The algorithm of Lemma 4 can be used to substitute any *real* constraint $ax \leq b$ with an *integer* one, preserving the same admissible integer points x with $\mathbf{0} \leq x \leq d$, $d \in \mathbf{R}^n$. More precisely, we have the following lemma.

Lemma 6. *Let* $T = \{x \in \mathbf{Z}^n : ax \leq b; \mathbf{0} \leq d\}$, *where* $a \in \mathbf{R}^n$, $b \in \mathbf{R}$, $d \in \mathbf{Z}^n_+$. *Then there exists an algorithm which finds a vector* $r \in \mathbf{Z}^n$ *and a number* $r_0 \in \mathbf{Z}$ *such that* $T = \{x \in \mathbf{Z}^n : rx \leq r_0; \mathbf{0} \leq x \leq d\}$. *The algorithm involves at most n applications of the algorithm from Lemma 4, with* $D = ||d||$.

Proof of the above fact is available in [4, Lemma 5.1].

From Lemmas 4 and 6, we obtain that the overall time complexity of the reduction stage is $O(mn^5 \log n(n + \log ||d||))$. Furthermore, the bit-size of the generated integers is $O(n^2(n + \log ||d||))$. Therefore, the overall bit-size of the reduced problem is $O(mn^3(n + \log ||d||))$.

At this point, the second of the announced results follows immediately. First we unfold the applications of the Lovász basis reduction algorithm in an algebraic

[2] Note that in the BSS-model extended with a unit cost floor operation (the case handled in [4]) this requires $O(n)$ operations.

decision tree with depth $O(mn^5 \log n(n + \log ||d||))$. After that, we branch on every bit of the obtained integer data problem, which adds $O(mn^3(n+\log ||d||))$ to the depth of the tree. Thus we obtain the following theorem.

Theorem 1. *There is an* $O(mn^5 \log n(n + \log ||d||))$ *algebraic decision tree for* $ILP_{\mathbf{R}}$.

To obtain the other result of ours, we continue with the second stage of the algorithm. That stage is an application of the Lenstra's algorithm to the integer data problem obtained as output of the first stage. A recursive step of this algorithm reduces an n-dimensional problem to a set of subproblems of dimension $n-1$, whose number is exponential but depending only on n. The basic algorithms used in this reduction are the Lovász basis reduction algorithm and the Kannan-Bachem's Hermite normal form algorithm. In addition, a linear programming problem of dimension $(m+2n) \times n$ is to be solved.

The two algorithms are applied to matrices of dimension depending only on n and with entries of bit-size $O(\log ||d||)$, as the value of n is fixed. Then, by Lemmas 1 and 2, their complexity as well as the bit-size of the integers they generate, are bounded by $O(\log ||d||)$. The linear programming problem can be solved in time $O(m+n)$ (i.e., *linear* in m) using the well-known Megiddo's algorithm [16]. Hence, if n is fixed, the overall complexity of this stage is $O(m \log ||d||)$. Thus we have obtained the following theorem.

Theorem 2. *There is an* $O(m \log ||d||)$ *algorithm for* $ILP_{\mathbf{R}}$ *of fixed dimension* n.

Theorem 2 implies a tight bound for the algebraic complexity of the knapsack problem.

Corollary 1. *The algebraic complexity of the knapsack problem* $KP_{\mathbf{R}}$ *of fixed dimension* n *is* $\Theta(\log \frac{1}{a_{\min}})$.

Proof An upper bound $O(\log \frac{1}{a_{\min}})$ follows from Theorem 2. A lower bound $\Omega(\log \frac{1}{a_{\min}})$ follows from [5], where a tight bound $\Theta(\log \frac{1}{a_{\min}})$ is proved for the algebraic complexity of the two-dimensional knapsack problem with real coefficients. □

Regarding possible practical applications, one can expect that the proposed algorithm for $\mathrm{ILP}_{\mathbf{R}}$ may be useful for problems with a small number of variables and a large number of constraints.

4 Lower Bound for the Knapsack Problem

In this section we study the complexity of the Boolean knapsack problem

$(0/1-\mathrm{KP}_{\mathbf{R}})$ Given $a \in \mathbf{R}_+^n$, decide if there is $x \in \{0,1\}^n$ such that $a^T x = 1$.

In the classical setting, the knapsack problem has been studied intensively (see [15] and the bibliography therein). In particular, the problem is NP-complete

[8]. Regarding "real" knapsacks, a number of results have been proved. Notable among them are the lower bounds $\Omega(n^2 \log \frac{1}{a_{\min}})$ and $\Omega(n^2)$ for $\mathrm{KP}_{\mathbf{R}}$'s and 0/1-$\mathrm{KP}_{\mathbf{R}}$'s complexity, respectively (see [1]). In [3] the topological complexity of the latter problem is found. [17] provides a parallel time lower bound. Some other results are presented in [7,4,5]. For a related discussion the reader is also referred to [2].

In this section, we take one more step towards determining the algebraic complexity of the knapsack problem. We obtain a result which complements the above mentioned Ben-Or's lower bounds. More precisely, we have the following theorem.

Theorem 3. *No algorithm solving* $0/1{-}\mathrm{KP}_{\mathbf{R}}$ *can achieve a time complexity* $o(n \log n) \cdot f(a_1, \ldots, a_n)$*, where* f *is an arbitrary continuous function of* n *variables.*

Remark 1. The requirement for f to be a continuous function is essential, as follows from [6]. More precisely, it has been shown that there is an $O(n^{\frac{1}{\delta(a)}})$ algorithm for $0/1{-}\mathrm{KP}_{\mathbf{R}}$, where $\delta(a) = \min\{|a^T z| : a^T z \neq 0, z \in \{-1, 0, 1\}^n\}$.

Proof Assume the opposite, i.e., that there is an algorithm $\mathcal{A}$ that solves 0/1-$\mathrm{KP}_{\mathbf{R}}$ in $o(n \log n) \cdot f(a_1, \ldots, a_n)$ time, for a continuous function f. We consider a subclass C of inputs of 0/1-$\mathrm{KP}_{\mathbf{R}}$ determined by the constraints

$$\frac{1}{3} < a_i < \frac{2}{3} \quad \text{for } 1 \leq i \leq n.$$

Let us denote

$$\begin{aligned} C &= \left\{a \mid \tfrac{1}{3} < a_i < \tfrac{2}{3},\ 1 \leq i \leq n\right\}, \\ C_{yes} &= \left\{a \mid a \in C,\ \exists x \in \{0,1\}^n : a^T x = 1\right\}, \\ C_{no} &= C \setminus C_{yes}. \end{aligned}$$

For any problem input from the considered subclass the value $f(a_1, \ldots, a_n)$ is bounded by a constant, and thus the time complexity of the algorithm $\mathcal{A}$ reduces to $o(n \log n)$.

On the other hand, according to the Ben-Or's theorem [1], in the considered computational model a lower bound $\Omega(\log \#c.c.(C_{no}))$ holds for the complexity of any algorithm solving 0/1-$\mathrm{KP}_{\mathbf{R}}$ for inputs from C_{yes}, where $\#c.c.(C_{no})$ is the number of connected components of C_{no}. We will show that $\#c.c.(C_{no}) = n!$, i.e., $\log \#c.c.(C_{no}) = \log n! = \Theta(n \log n)$. This will contradict the $o(n \log n)$ time complexity of the algorithm $\mathcal{A}$ on inputs from C. Thus, to complete the proof it suffices to prove the following lemma.

Lemma 7. *The set* C_{no} *has exactly* $n!$ *connected components.*

Proof First of all, let us observe that $a \in C_{yes}$ if and only if $\exists i, j\ i \neq j$ such that $a_i + a_j = 1$. For every $a \in C_{no}$ we denote

$$S_a = \{\{a_i, a_j\} \mid i \neq j,\ a_i + a_j < 1\}.$$

As a first step of the proof we will show that C_{no} has as many connected components as the number of all distinct sets S_a, $a \in C_{no}$.

First we show that if for some $a', a'' \in C_{no}$ the condition $S_{a'} = S_{a''}$ holds, then a' and a'' belong to the same connected component of C_{no}. Clearly, $C_{no} = \left\{a \mid a \in C,\ \forall x \in \{0,1\}^n : a^T x \neq 1\right\}$. Since $S_{a'} = S_{a''}$, for every i, j $i \neq j$ both $a'_i + a'_j - 1$ and $a''_i + a''_j - 1$ have the same sign, either positive or negative but not zero. The same is the sign of $a_i + a_j - 1$, where $a = (a_1, \ldots, a_n)$ is any affine combination of a' and a''. Thus $a \in C_{no}$, so that the whole segment with endpoints a' and a'' lies in C_{no}. This implies that these two points belong to the same connected component.

Now we prove that if $a', a'' \in C_{no}$ are in the same connected component, then $S_{a'} = S_{a''}$. Assume the opposite, i.e., that there are $a', a'' \in C_{no}$ belonging to the same connected component D, while $S_{a'} \neq S_{a''}$. Then there are i, j $i \neq j$ such that $a'_i + a'_j < 1$ and $a''_i + a''_j > 1$. Let $\mathcal{L}$ be a continuous curve with end-points a' and a'', which is contained in D. We define a function $h(x) = x_i + x_j$, where x_i, x_j are, respectively, the ith and jth component of $x \in \mathbf{R}^n$. Let us consider the restriction of $h(x)$ on the curve $\mathcal{L}$. We have $h(a') = a'_i + a'_j < 1 < a''_i + a''_j = h(a'')$. Since h is continuous, there must be a point a on the curve $\mathcal{L}$ for which $h(a) = a_i + a_j = 1$. But $a \in \mathcal{L} \subseteq D \subseteq C_{no}$ and therefore $a_i + a_j < 1$, which is a contradiction.

Thus it only remains to count all distinct sets S_a, $a \in C_{no}$. We will use induction on the dimension n. As the basis of the induction, for $n = 2$ we have two such distinct sets, namely $\emptyset$ and $\{\{a_1, a_2\}\}$. Suppose that the thesis is true for dimension $n - 1$, and take an arbitrary set S_a, where $a = (a_1, \ldots, a_{n-1})$ is an $(n-1)$-dimensional vector. Without loss of generality we assume that the coordinates of a are all distinct, and consider them in an increasing order:

$$\frac{1}{3} < a_{i_1} < a_{i_2} < \ldots < a_{i_{n-1}} < \frac{2}{3}.$$

To pass to dimension n, we need to add one more coordinate a_n to the vector a. One can choose a_n in n different ways:

$$\begin{array}{l} 1 - a_{i_1} < a_n < \frac{2}{3}, \\ 1 - a_{i_j} < a_n < 1 - a_{i_{j-1}} \text{ for } 2 \leq j \leq n-1, \\ \frac{1}{3} < a_n < 1 - a_{i_{n-1}}. \end{array}$$

Thus we get n distinct sets "generated" by S_a:

$$S_a \cup \left\{\{a_{i_k}, a_n\} \mid 1 \leq k \leq j\right\}, \quad \text{for } 0 \leq j \leq n-1.$$

Note also that two sets generated by sets $S_{a'} \neq S_{a''}$ are distinct because their maximum subsets of (unordered) pairs not containing a_n are (i.e., the sets $S_{a'}$ and $S_{a''}$, themselves).

Thus we have proved that the number of connected components of C_{no} increases by a factor of n when passing from dimension $n - 1$ to dimension n. Hence, this number is exactly $n!$, as claimed. □

Remark 2. The Ben-Or's theorem states a lower bound $\Omega(\log \#c.c.(C_{no}))$ on the complexity of any algorithm solving 0/1-KP$_{\mathbf{R}}$. Note that, in the general case, when no restrictions on the coefficients $a_1, a_2, \ldots, a_n$ are imposed, the number of connected components of C_{no} is $2^{\Omega(n^2)}$. Hence, the lower bound $\Omega(n^2)$ holds. This larger number $2^{\Omega(n^2)}$ (compared to the number $n!$ of connected components of the class C) is due to the inputs in which the a_i's are "close to 0 or 1." These inputs are excluded from C. Therefore, the Ben-Or's bound $\Omega(n^2)$ does not imply directly ours, although we have used his theorem, together with Lemma 7, to obtain ours.

Remark 3. To obtain the complexity result of Theorem 3, we have used the class of instances $C = \left\{a \mid \frac{1}{3} < a_i < \frac{2}{3},\ 1 \leq i \leq n\right\}$. It is easy to see that an $O(n \log n)$ algorithm exists for this particular class. To show this, first we sort in $O(n \log n)$ time the coefficients of $a^T x = 1$. After an appropriate substitution and enumeration of the variables we obtain an equation with coefficients $a_1 < a_2 < \ldots < a_n$. As already observed, a solution to $a^T x = 1$ exists if and only if $\exists i, j\ i \neq j$ such that $a_i + a_j = 1$. In order to check whether this condition is met, we search the sorted array of coefficients in linear time, as follows. Set $i = 1$, $j = n$. If $a_i + a_j < 1$, then set $i := i + 1$; If $a_i + a_j > 1$, then set $j := j - 1$. If $a_i + a_j = 1$ or $i = j$, then stop. In the former case the equation has a solution (namely, $x_i = 1$, $x_j = 1$, $x_k = 0$ for $k \neq i, j$). In the latter case a solution does not exist. The complexity of this procedure is $O(n)$, and thus the overall complexity of the algorithm is $O(n \log n)$. The proof of Theorem 3 demonstrates that for the class C this algorithm is, in fact, optimal.

Clearly, an analogous result holds for the complexity of the classical Boolean knapsack problem (0/1-KP) $a^T x = b$ with integer coefficients. This problem is equivalent to the equation $\bar{a}^T x = 1$ with $\bar{a} = \frac{1}{b} a$, to which Theorem 3 applies. Thus, we have lower time complexity bounds, both for the Boolean knapsack problem with real coefficients and the classical formulation, and these bounds are independent of the known lower bound $\Omega(n^2)$.

One can show that the result of Theorem 3 is still valid for the integer knapsack problem KP$_{\mathbf{R}}$. Unlike the Boolean case, here an input a belongs to C_{yes} not only if $a_i + a_j = 1$ for some indices i, j, but also if $a_k = \frac{1}{2}$ for some index k. Accordingly, the set S_a is modified as

$$S_a = \{\{a_i, a_j\} \mid\ a_i + a_j < 1\}.$$

Note in the definition above that we allow $i = j$. This adds to the set S_a every singleton $\{a_k\}$ such that $a_k = 1/2$. One can show that C_{no} has *at least* as many connected components as the number of the distinct sets S_a, $a \in C_{no}$. Then by induction on n one obtains that the set C_{no} has *at least* $n!$ connected components, from where the result follows. The proof is a straightforward modification of the one of Theorem 3 and therefore is omitted.

As a last comment of this section we mention that the Ben-Or's lower bound, as well as the negative complexity result of Theorem 3, suggest seeking certain

approximation solutions to the knapsack problem. For instance, one can look for a Boolean vector that is "enough close" to the hyperplane determined by $a^T x = 1$ (e.g., minimizing $|a^T x - 1|$ within a given tolerance $\varepsilon > 0$). We notice that the best of the existing algorithms (see, e.g., [11]) for this approximation problem are indeed *linear* with respect to the dimension n.

5 Concluding Remarks

We have presented an $O(m \log ||d||)$ algorithm for integer linear programming with real coefficients and fixed number of variables, within the Blum-Shub-Smale computational model. A further task would be to show that this complexity bound is tight.

We have also obtained a lower bound for the complexity of the real knapsack problem. In view of this result, it would be interesting to know if there is an algorithm for 0/1-$\mathrm{KP}_{\mathbf{R}}$ with time complexity $O(n^{2-\delta} f(a))$, $\delta \geq 0$.

Some of the obtained results (e.g., Corollary 1) show that the integer programming problems are, in general, intractable in the framework of the complexity theory over the reals, since their complexity cannot be bounded by any polynomial in the input size, the latter being a polynomial only in m and n. Some further refinements of the theory suggest, however, that these problems can be considered as efficiently solvable. Following Smale [23], a numerical algorithm can be considered as efficient only if its complexity is bounded by a polynomial in the problem dimensions and the logarithm of its *weight*. The weight function is defined in accordance with the problem specificity and used to measure the difficulty of a problem instance. Under such a convention, let us define the weight of $\mathrm{ILP}_{\mathbf{R}}$ as a number $||d||$ which bounds the norms of the admissible solutions. Then the results of Section 3 imply that $\mathrm{ILP}_{\mathbf{R}}$ and $\mathrm{KP}_{\mathbf{R}}$ are efficiently solvable in the above sense.

Acknowledgments

The authors thank Bruno Codenotti for reading an early version of this paper, and the two referees for their valuable suggestions.

References

1. Ben-Or, M.: Lower Bounds for Algebraic Computation Tree. Proc. Of the 15th ACM STOC (1983) 80–86
2. Blum, L., Cucker, F., Shub, M., Smale, S.: Complexity and Real Computation. Springer-Verlag, Berlin Heidelberg New York (1995)
3. Blum, L., Shub, M., Smale, S.: On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness, Recursive Functions and Universal Machines. Bull. Amer. Math. Soc. (NS), **21** (1989) 1–46
4. Brimkov, V.E., Danchev, S.S.: Real Data – Integer Solution Problems within the Blum-Shub-Smale Computational Model: J. of Complexity, **13** (1997) 279–300

5. Brimkov, V.E., Danchev, S.S., Leoncini, M.: Tight Complexity Bounds for the Two-Dimensional Real Knapsack Problem. Calcolo, **36** (1999) 123–128
6. Brimkov, V.E., Dantchev, S.S.: Lower Bounds, "Pseudopolynomial" and Approximation Algorithms for the Knapsack Problem with Real Coefficients. Electronic Colloquium on Computational Complexity, TR98-015 (1998). http://www.eccc.uni-trier.de/eccc/
7. Cucker, F., Shub, M.: Generalized Knapsack Problem and Fixed Degree Separations. Theoret. Comput. Sci., **161** (1996) 301–306
8. Garey, M.S., Johnson, D.S.: Computers and Intractability: a Guide to the Theory of NP-Completeness. Freeman & Co., San Francisco (1979)
9. Hastad, J., Just, B., Lagarias, J.C., Schnoor, C.P.: Polynomial Time Algorithms for Finding Integer Relations among Real Numbers. SIAM J. Comput., **18** (1989) 859–881
10. Kannan, R., Bachem, A.: Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix. SIAM J. Comput., **8** (1979) 499–507
11. Kellerer, H., Mansini, R., Pferschy, U., Speranza, M.G.: An Efficient Fully Polynomial Approximation Scheme for the Subset-Sum Problem. Proc. 8th ISAAC Symposium, Lecture Notes in Computer Science, Vol. 1350. Springer-Verlag, Berlin Heidelberg New York (1997) 394–403
12. Lenstra, H.W., Jr.: Integer Programming with a Fixed Number of Variables. Math. Oper. Res., **8** (1983) 538–548
13. Lenstra, A.K., Lenstra, H.W., Jr., Lovász, L.: Factoring Polynomials with Rational Coefficients. Math. Ann., **261** (1982) 515–534
14. Meyer Auf Der Heide, F.: A Polynomial Linear Search Algorithm for the n-Dimensional Knapsack Problem. J. of ACM, **31** (3) (1984) 668–676
15. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, Chichester New York Brisbane Toronto Singapore (1990)
16. Megiddo, N.: Linear Programming in Linear Time when the Dimension is Fixed. J. of ACM, **31** (1) (1984) 114–127
17. Montaña, J.L., Pardo, L.M.: Lower Bounds for Arithmetic Networks. Appl. Algebra Eng. Comm. Comput., **4** (1993) 1–24
18. Novak, E., The Real Number Model in Numerical Analysis. J. of Complexity, **11** (1994) 57–73
19. Preparata, F.P., Shamos, M.I.: Computational Geometry. Springer-Verlag, Berlin Heidelberg New York (1985)
20. Rössner, C., Schnorr, C.P.: An Optimal, Stable Continued Fraction Algorithm for Arbitrary Dimension. In: Integer Programming and Combinatorial Optimization, Lecture Notes in Computer Science, Vol. 1084. Springer-Verlag, Berlin Heidelberg New York (1996) 31–43
21. Schrijver, A.: Theory of Linear and Integer Programming, Wiley, Chichester New York Brisbane Toronto Singapore (1986)
22. Strassen, V.: Algebraic Complexity Theory. In: van Leeuwen, J. (Ed.): Handbook of Theoretical Computer Science, Vol. A, Elsevier, Amsterdam (1990) 633–672
23. Smale, S.: Some Remarks on the Foundations of Numerical Analysis. SIAM Rev., **32** (2) (1990) 211–220