Electronic Colloquium on Computational Complexity, Report No. 17 (2000)

On the Algebraic Complexity of Integer Programming

Valentin E. Brimkov* Stefan S. Dantchev†

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 if there is $x \in \mathbf{Z}^n$ such that $Ax \le b$, where $\mathbf{0} \le x \le d$.

We show that there is an $O(m \log ||d||)$ algorithm for $\mathrm{ILP}_{\mathbf{R}}$, when the value of n is fixed. As a consequence, we obtain a tight algebraic complexity bound $\Theta\left(\log \frac{1}{a_{\min}}\right)$, $a_{\min} = \min\{a_1, \dots, a_n\}$ for the Knapsack problem $(\mathrm{KP}_{\mathbf{R}})$: Given $a \in \mathbf{R}^n_+$, decide if there is $x \in \mathbf{Z}^n$ such that $a^T x = 1$, when the dimension n is fixed. We achieve these results in particular through a careful analysis of the algebraic complexity of the Lovász' basis reduction algorithm and Kannan-Bachem's Hermite normal form algorithm, which may be of interest in its own.

We also obtain 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.

Keywords: *Algebraic complexity, Complexity bounds, Integer programming, Knapsack problem*

1 Introduction

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

$$(\mathrm{ILP}_{\mathbf{R}}) \qquad \begin{array}{l}\text{Given a matrix } A \in \mathbf{R}^{m\times n} \text{ and vectors } b \in \mathbf{R}^m, d \in \mathbf{R}^n,\\ \text{decide if there is } x \in \mathbf{Z}^n \text{ such that } Ax \le b, \text{where } \mathbf{0} \le x \le d.\end{array}$$

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 used in scientific computing, computational geometry, and (although not explicitly) numerical analysis (see, e.g., [15, 16, 19]). 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 that model, the assumption is that all the reals–elements of the input–have unit size, and the basic algebraic operations $+, -, *, /$ and the relation $\le$ 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 the ones mentioned above,

*Department of Mathematics, Eastern Mediterranean University, P.O. Box 95, Famagusta, TRNC, via Mersin 10, Turkey. E-mail: brimkov.as@mozart.emu.edu.tr. Part of this work has been done while this author was visiting Istituto di Matematica Computazionale, Consiglio Nazionale delle Ricerche, Pisa, Italy, and Dipartimento di Elettronica ed Informatica, Università di Padova, Italy.

†**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. E-mail: dantchev@brics.dk.

ISSN 1433-8092

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 [7]. 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, 14, 6, 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:

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

With the present paper we take a step towards determining $\mathrm{ILP}_{\mathbf{R}}$'s and $\mathrm{KP}_{\mathbf{R}}$'s complexity. As a main contribution we show in Section 3 that there is an $O\left(m \log ||d||\right)$ algorithm for $\mathrm{ILP}_{\mathbf{R}}$ when the value of n is fixed.

A similar result is known for the integer case, namely, the well-known Lenstra's algorithm for ILP of a fixed dimension n [10]. Ours 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 one employs two well-known algorithms: the Lovász' basis reduction algorithm [11] and the Kannan-Bachem's Hermite normal form algorithm [9]. It is straightforward to obtain an upper time complexity bound that is *quadratic* in $\log ||d||$. Our more detailed analysis (presented in Section 2) 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 Lovász' basis reduction algorithm and 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.

We also obtain an $O\left(mn^5 \log n \left(n + \log ||d||\right)\right)$ 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 $x \in \{0,1\}^n$) [12].

2 Analysis of the Basic Algorithms

In this section, we analyze Lovász lattice basis reduction algorithm [11] and Kannan and Bachem's Hermite normal form algorithm [9]. 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 more careful 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 very useful in analyzing the algorithms in the next sections.

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

2.2 Lovász Lattice Basis Reduction Algorithm

In the description and analysis of the algorithm we follow [8]. The input consists of linearly independent vectors $b_1, b_2, \dots b_n \in Q^n$, considered as a basis of a lattice L. The algorithm transforms them in iterations. At the end, they form a basis of L which is reduced in Lovázs sense.

First we recall some definitions, then describe the Lovász lattice basis reduction algorithm, itself. With a basis $b_1, b_2, \dots b_n$, we associate the orthogonal system $b_1^*, b_2^*, \dots b_n^*$, where b_i^* is the component of b_i which is orthogonal to $b_1, b_2, \dots b_{i-1}$. The vectors $b_1^*, b_2^*, \dots b_n^*$ 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 $\mu_{i,j} = \left\langle b_i, b_j^* \right\rangle \Big/ \left\| b_j^* \right\|^2$.

The basis $b_1, b_2, \dots b_n$ is *size-reduced* if all $|\mu_{i,j}| \le \frac{1}{2}$. Given an arbitrary basis $b_1, b_2, \dots b_n$, we can transform it into a size-reduced basis having 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 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 $\|b_i^*\|^2 \leq 2\left\|b_{i+1}^*\right\|^2$, $1 \leq i \leq n-1$. We call such a basis *reduced in 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

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

and

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

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 $\|b_i^*\|$s. After an exchange step, we obtain

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

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 $\lceil . \rfloor$ operations 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

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

Thus, the time complexity of one iteration is

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

and then the time complexity of all iterations is

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

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 $O\left(n^2 \log n \log\frac{F_{start}}{F_{end}}\right)$. 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 of Section 2.1, it is not hard to conclude that before the size-reduction phase of step 1, $\|\mu_{i,j}\| \leq 2^{O(Sn^2)}$. The size-reduction itself takes $O(n^2)$ $\lceil . \rfloor$ operations on these numbers, so that the time complexity of step 1 is clearly $O(Sn^4)$.

Lastly, we need to estimate F_{start} and F_{end}. F_{start} is a product of the determinants of $n-1$ matrices $B_i^T B_i$ for $1 \leq i \leq n-1$, so that $|F_{start}| \leq 2^{O(n^2(\log n + S))}$. To esitmate F_{end}, let us observe that any of the vectors $b_1^{end}, b_2^{end}, \dots b_n^{end}$ is an *integer* linear combination of $b_1^{start}, b_2^{start}, \dots b_n^{start}$. 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 of Section 2.1, and consequently $F_{end} \geq 1 \big/ 2^{O(n^3 S)}$. Thus $\log \frac{F_{start}}{F_{end}} = O(n^3 S)$ and the overall complexity of the Lovász basis reduction algorithm is $O(Sn^5 \log n)$.

Finally, we will prove that the bit-size of the entries of the reduced basis is $O(Sn^3)$. Let us remember 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 by a the least common multiple of all entries of B_n^{start}. Note that the bit-size of a is $O(Sn^2)$. Since b_i^{end}'s are integer linear combinations of b_i^{start}'s, the vectors ab_i^{end} are integer. Therefore, we have

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

Thus, every entry of aB_n^{end} is of bit-size $O(Sn^3)$ and so is every entry of B_n^{end}. In terms of the adopted denotations, we have proven 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 [18]. The input for the algorithm is an $m \times n$ ($m \leq n$) integer matrix A of full rank. The algorithm uses the matrix

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

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 $\left\lfloor a'_{k,i} \big/ a'_{k,j} \right\rfloor$, and 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

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

After one iteration of step 3, we have

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

which implies both $F_{new}/F < 1/2$ and $F_{new}/F < a'_{k,j} \big/ a'_{k,i}$. It is not hard to see that one iteration of step 3 can be performed in $O\left(m \log(a'_{k,i} \big/ a'_{k,j})\right) = O\left(m \log(F/F_{new})\right)$ time. So, step 3 takes $O\left(m \log(F_{start}/F_{end})\right)$ 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 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 proven 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 Kannan-Bachem's algorithm that reduces A into Hermite normal form, is $O(m^2 n(\log m + S))$.*

3 Basic Results about $\mathbf{ILP_R}$

In this section we use the algorithms' analysis 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]. We notice that there the complexity of that method is 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.

As mentioned before, the algorithm consists of two main stages. In the first stage, it reduces the constraints with real coefficients to ones 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 3 *Given a vector $\alpha \in \mathbf{R}^n$ with $|\alpha_j| \leq 1, j = 1, 2, \dots, n$, and $L \in \mathbf{Z}_+$, there exists an $O(n^4 \log n(n + \log L))$ algorithm that finds $p \in \mathbf{Z}^n$ and $q \in \mathbf{Z}_+$ such that $|\alpha_j - p_j/q| < 1/(qL)$, $j = 1, 2, \dots, n$, and $1 \leq q \leq \lceil 2^{n(n+5)/4} L^n \rceil$.*

[1]To reduce the given real constraints to an equivalent set of integer constraints, one can also use the approach from [17].

Proof The proof is analogous to the one of [4, Lemma 4.4], therefore we omit some technical details. Let us sketch the algorithm finding a vector $p \in \mathbf{Z}^n$ and an integer $q \in \mathbf{Z}_+$ with the required properties. For each α_j we find the closest rational fraction a_j with denominator $G = \lceil 2^{n(n+5)/4} L^{n+1} \rceil$. This can be done in time $O(n \log G) = O(n^3 + n^2 \log L)$ in the BSS-model. (Note that in the BSS-model extended with a unit cost floor operation–the case handled in [4, Lemma 4.4]–this requires $O(n)$ operations.) Next we use the algorithm for finding simultaneous Diophantine approximations [12, Corollary 6.4c]). For a given *rational* vector $v \in \mathbf{Q}^n$ and a *rational* number $0 < \varepsilon < 1$, this algorithm finds an *integer* vector p and an *integer* number q with $||v - (1/q)p|| < \varepsilon/q$ and $1 \le q \le 2^{n(n+1)/4}\varepsilon^{-n}$. We apply this algorithm to the constructed *rational* vector $a = (a_1, a_2, \ldots, a_n)$ and $\varepsilon = 1/(2L)$, and obtain a vector $p \in \mathbf{Z}^n$ and an integer $q \in \mathbf{Z}_+$ satisfying the desired conditions.

In order to evaluate the algorithm's complexity, we have to evaluate the complexity of the simultaneous Diophantine approximation algorithm. Note that the latter is, in fact, a specialization of the Lovász basis reduction algorithm, applied to a matrix of the form

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

(see, e.g., [12, Corollary 6.4c]). This matrix entries are rational numbers, all with the same denominator $G = 2^{O(n(n+\log L))}$. It is shown in [4, Lemma 4.4] that in this special case the number of iterations of the algorithm is $O(\log \frac{F_{start}}{F_{end}}) = O(n^3 + n^2 \log L)$. Combining this fact with (3), we obtain the result stated. □

It is shown in [4] how the algorithm of the above lemma can be used to substitute any *real* linear constraint $ax \le b$ with an *integer* one, preserving the same admissible integer points x with $0 \le x \le d$, $d \in \mathbf{R}^n$. For this, at most n applications of the algorithm are needed, with $L = ||d||$ (see [4, Lemma 5.1]). Hence, the overall time complexity of the reduction stage turns out to be $O(mn^5 \log n(n + \log ||d||))$. Also, 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, one of the announced results follows immediately. First we unfold the applications of the Lovász basis reduction algorithm in an algebraic 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 can state 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 our other results, we continue with the second stage of the algorithm, which is an application of Lenstra's algorithm [10] to the integer data problem obtained as an 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 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 on matrices of dimension depending only on n and with entries of bit-size $O(\log ||d||)$, when the value of n is fixed. Then by Lemmas 1 and 2, their complexity and the bit-size of the integers they generate are bounded by $O(\log ||d||)$. The linear programming problem can be solved in time $O(m+n)$ (that is, *linear* in m) using the well-known Megiddo's algorithm [13]. 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 $\Theta(\log \frac{1}{a_{\min}})$ was proved for the algebraic complexity within the BSS-model of the two-dimensional Knapsack problem with real coefficients. □

4 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 computation model. A further task would be to show that this complexity bound is tight.

Some of the obtained results (e.g., Corollary 1) show that the integer programming problems are, in general, intractable in the framework of 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 [20], 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 dependence on the problem specificity and used to measure the difficulty of a problem instance. Under such a convention, let us define the weight of $\text{ILP}_{\mathbf{R}}$ as a number $||d||$ which bounds the norms of the admissible solutions. Then the results of Section 3 imply that $\text{ILP}_{\mathbf{R}}$ and $\text{KP}_{\mathbf{R}}$ are efficiently solvable in the above sense.

Acknowledgments

The authors thank Bruno Codenotti for reading an early version of this paper.

References

[1] Ben-Or, M., Lower bounds for algebraic computation tree, in: Proc. 15th ACM STOC (1983) 80-86.

[2] Blum, L., F. Cucker, M. Shub, and S. Smale, *Complexity and Real Computation* (Springer, New York, 1995).

[3] Blum, L., M. Shub, and S. Smale, 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., and S.S. Danchev, Real data - integer solution problems within the Blum-Shub-Smale computational model, *J. of Complexity* **13** (1997) 279-300.

[5] Brimkov, V.E., S.S. Danchev, and M. Leoncini, Tight complexity bounds for the two-dimensional real knapsack problem, *Calcolo* **36** (1999) 123-128.

[6] Cucker, F., and M. Shub, Generalized Knapsack problem and fixed degree separations, *Theoret. Comput. Sci.* **161** (1996) 301-306.

[7] Garey, M.S., and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness* (Freeman & Co., San Francisco, 1979).

[8] Hastad, J., B. Just, J.C. Lagarias, C.P. Schnoor, Polynomial time algorithms for finding integer relations among real numbers, *SIAM J. Comput.* **18** (1989) 859-881

[9] Kannan, R., and A. Bachem, Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix, *SIAM J. Comput.* **8** (1979) 499-507.

[10] Lenstra, H.W., Jr., Integer programming with a fixed number of variables, *Math. Oper. Res.* **8** (1983) 538-548.

[11] Lenstra, A.K., H.W. Lenstra, Jr., and L. Lovász, Factoring polynomials with rational coefficients, *Math. Ann.* **261** (1982) 515-534.

[12] Meyer Auf Der Heide, F., A Polynomial Linear Search Algorithm for the n-Dimensional K-napsack Problem, *J. of ACM* **31**(3)(1984) 668-676.

[13] Megiddo, N., Linear programming in linear time when the dimension is fixed, *J. of ACM*, **31** 1 (1984) 114-127.

[14] Montaña, J.L., and L.M. Pardo, Lower bounds for arithmetic networks, *Appl. Algebra Eng. Comm. Comput.* **4** (1993) 1-24.

[15] Novak, E., The real number model in Numerical Analysis, *J. of Complexity*, **11** (1994) 57-73.

[16] Preparata, F.P., and M.I. Shamos, *Computational Geometry* (Springer, Berlin, 1985).

[17] Rössner, C., C.P. Schnorr, An optimal, stable continued fraction algorithm for arbitrary dimension, in *Integer Programming and Combinatorial Optimization*, LNCS, 1084, Springer, Berlin (1996) 31-43.

[18] Schrijver, A., *Theory of Linear and Integer Programming*, (Wiley, Chichester/New York/Brisbane/Toronto/Singapore, 1986)

[19] Strassen, V., Algebraic complexity theory, in: van Leeuwen, J. (Ed.), *Handbook of Theoretical Computer Science, Vol. A* (Elsevier, Amsterdam, 1990) 633-672.

[20] Smale, S., Some remarks on the foundations of numerical analysis, *SIAM Rev.* **32** (2) (1990) 211-220.