Electronic Colloquium on Computational Complexity, Report No. 15 (1998)

Lower Bounds, "Pseudopolynomial" and Approximation Algorithms for the Knapsack Problem with Real Coefficients

Valentin E. Brimkov[*] Stefan S. Dantchev[†]

Abstract

In this paper we study the Boolean Knapsack problem ($\mathrm{KP}_{\mathbf{R}}$) $a^T x = 1$, $x \in \{0,1\}^n$ with real coefficients, in the framework of the Blum-Shub-Smale real number computational model [4]. We obtain a new lower bound $\Omega(n \log n) \cdot f(1/a_{\min})$ for the time complexity of this problem, as well as an $\Omega(n \log n) \cdot f(b/a_{\min})$ lower time complexity bound for the classical Boolean Knapsack problem $a^T x = b$, $x \in \{0,1\}^n$ with positive integer coefficients. Here n is the dimension of the problem, $a_{\min}$ is the minimal coefficient in the input, and f is any continuous function of one variable. These lower bounds appear as alternatives to the well known lower bound $\Omega(n^2)$ [1] which applies to both cases.

We also construct certain algorithms for $\mathrm{KP}_{\mathbf{R}}$. We suggest a way of defining the concept of a pseudopolynomial algorithm for $\mathrm{KP}_{\mathbf{R}}$ and design such an algorithm. On integer inputs this algorithm is pseudopolynomial according to the classical complexity theory. Overall, it is seen as superior to the classical dynamic programming algorithm. Finally, we present two algorithms with running times $O(n^2/\varepsilon)$ and $O(n/(\varepsilon a_{\min}))$, respectively, with each finding an approximation solution with an absolute error ε.

1 Introduction

Blum *et al.* [4] establish the groundwork for a theory of real computation, with the goal of providing theoretical foundations to scientific computing and numerical analysis. They intro-

[*]Department of Mathematics, Eastern Mediterranean University, P.O. Box 95, Famagusta, TRNC, via Mersin 10, Turkey. On leave from Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, 1113 Sofia, Bulgaria. Part of this work has been done while the author was visiting IMC-CNR, Via S. Maria, 46, 56100 Pisa (Italy) and supported by a CNR fellowship.

[†]Centre of Biomedical Engineering, Bulgarian Academy of Sciences, Acad. G. Bonchev str., bl. 105, 1113 Sofia, Bulgaria.

ISSN 1433-8092

duce a real number model of computation in terms of a "real" Turing machine, able to operate with real numbers at unit cost by the four arithmetic operations. They also define the basic complexity classes and prove an analog of the well known Cook's theorem.

Recently an increasing number of authors have contributed to the progress in this new realm. Various problems have been studied under the Blum-Shub-Smale computational model or under some of its variations. A number of papers address the following version of the well known Boolean Knapsack problem.

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

In the classical complexity theory the Knapsack problem is among the best studied combinatorial problems. Regarding $\mathrm{KP}_{\mathbf{R}}$, a number of results are also known. An $\Omega(n^2)$ lower bound for this problem is provided by [1]. In [4] its topological complexity is found. [8] gives a parallel time lower bound. Some complexity properties of $\mathrm{KP}_{\mathbf{R}}$ are obtained in [6]. For related discussion on the matter, the reader is also referred to [3]. In a recent paper [5], we consider another formulation of the Knapsack problem for which we present some complexity results and algorithms.

With the present note, we take one more step towards the study of the Knapsack problem under the Blum-Shub-Smale computational model. We obtain a new lower bound $\Omega(n \log n) \cdot f(1/a_{\min})$ for the time complexity of this problem, as well as an $\Omega(n \log n) \cdot f(b/a_{\min})$ lower time complexity bound for the classical Boolean Knapsack problem $a^T x = b$, $x \in \{0,1\}^n$ with positive integer coefficients. Here n is the dimension of the problem, $a_{\min}$ is the minimal coefficient in the input, and f is any continuous function of one variable. These lower bounds appear as alternatives to the well known lower bound $\Omega(n^2)$ [1] which applies to both cases.

We also construct certain algorithms for $\mathrm{KP}_{\mathbf{R}}$. We suggest a way of defining the concept of a pseudopolynomial algorithm for $\mathrm{KP}_{\mathbf{R}}$ and design such an algorithm. On integer inputs this algorithm is pseudopolynomial according to the classical complexity theory. Overall, it is seen as superior to the classical dynamic programming algorithm. Finally, we present two algorithms with running times $O(n^2/\varepsilon)$ and $O(n/(\varepsilon a_{\min}))$, respectively, with each finding approximation solution with an absolute error bounded by ε.

2 Model of Computation

Our results are valid in the *Blum-Shub-Smale computational model* [4]. As an overview, it suffices to give an idea about the notions of problem size, arithmetic, cost of arithmetic, algorithm and problem complexity.

In the model considered, a problem instance is a tuple of real numbers, with the number of coordinates counting as the *instance* (or *input*) *size*. Infinite precision real numbers can be stored and operated at unit space and time by the four *arithmetic operations* $+, -, *, /$ and *relation* $\leq$.

Let Π be a problem and A_Π be any algorithm solving Π. The *cost of computation* for an instance of Π is the number of arithmetic operations and branchings performed by A_Π to solve the instance. Then the (*time*) *complexity of algorithm* A_Π is the maximal cost to solve the problem, over all instances of size n. The (*computational*) *complexity of* Π is the minimal cost of solving Π over all possible algorithms.

To obtain rigorous definitions and a detailed presentation of complexity theory over $\mathbf{R}$ the reader is referred to [4] (see also [2]).

3 Lower Bound

As mentioned, an $\Omega(n^2)$ lower bound exists for $\mathrm{KP}_{\mathbf{R}}$'s complexity [1]. Next we obtain an alternative lower bound $\Omega(n \log n) \cdot f(1/a_{\min})$, where f is any continuous function of one variable.

Let us denote

$$B_n^k = \left\{ x \mid x \in \{0,1\}^n, \sum_{i=1}^n x_i = k \right\},$$

where $k \in \mathbf{N}$. Specifically, let us consider B_n^2. We say that a pair (S_1, S_2) of subsets of B_n^2 is a *linear separation* of B_n^2 if $B_n^2 = S_1 \cup S_2$, and $S_1 = \left\{ x \mid x \in B_n^2, w^T x < 0 \right\}$, $S_2 = \left\{ x \mid x \in B_n^2, w^T x > 0 \right\}$ for some $w \in \mathbf{R}^n$. We shall also say that the vector w *defines* the linear separation (S_1, S_2). Clearly, a linear separation (S_1, S_2) can be defined by uncountably many different vectors w. The following lemma can be proved along the lines of a similar claim in [6][1].

Lemma 1 *Let $\mathcal{E}_n = \{x \in \mathbf{R}^n \mid \forall y \in \{0,1\}^n \ x^T y \neq 0\}$. Then two points $u, v \in \mathbf{R}^n$ define the same linear separation of B_n^2 if and only if they belong to the same connected component of $\mathcal{E}_n$.*

Clearly, the connected components of $\mathcal{E}_n$ are open sets. In offering proof for our result, we shall use the following lemma.

[1]This work shows that the classes NTIME($O(n^d)$) and co-NTIME($O(n^d)$) are different. To this end, first a characterization is given for the set of hyperplanes that separate a given threshold subset of K^n, where $K = \{0, 1, \ldots, k-1\}$. Lemma 1 provides an analogous characterization for the set of hyperplanes that define a given linear separation.

Lemma 2 *There are exactly $n!$ distinct linear separations of B_n^2, $n \geq 2$.*

Proof. We use induction on n. Let us consider an arbitrary linear separation (S_1, S_2) of B_{n-1}^2, defined by some vector $w \in \mathbf{R}^n$. Without loss of generality assume that $w_1 < w_2 < \cdots < w_{n-1}$. If we choose w_n such that $-w_1 < w_n$, $-w_{i+1} < w_n < -w_i$ for $1 \leq i \leq n-2$, and $w_n < -w_{n-1}$, we obtain the following n linear separations $\left(S_1^j, S_2^j\right)$ of B_n^2 for $0 \leq j \leq n-1$:

$$S_1^j = \{(x,0) \mid x \in S_1\} \cup \left\{(x,1) \mid x \in B_{n-1}^1, \sum_{i=1}^{j} x_i = 1\right\},$$
$$S_2^j = \{(x,0) \mid x \in S_2\} \cup \left\{(x,1) \mid x \in B_{n-1}^1, \sum_{i=j+1}^{n-1} x_i = 1\right\}.$$

Thus every linear separation of B_{n-1}^2 generates exactly n distinct linear separations of B_n^2.

Since there are two linear separations of B_2^2, namely $(\emptyset, \{(1,1)\})$ and $(\{(1,1), \emptyset\})$, we find that the number of all distinct linear separations of B_n^2 is $n!$ □

Now we are in a position to prove the following theorem.

Theorem 1 *Any algorithm solving* $\mathrm{KP}_{\mathbf{R}}$ *has lower time complexity bound* $\Omega(n \log n) \cdot f(\frac{1}{a_{\min}})$, *where f is an arbitrary continuous function of one variable.*

Proof. Consider the class of Knapsack problems $\mathrm{KP}_{\mathbf{R}}$ which satisfies the condition $\frac{2}{3} \geq a_i \geq \frac{1}{3}$ for $1 \leq i \leq n$, and denote

$$U = \left\{a \mid \tfrac{2}{3} \geq a_i \geq \tfrac{1}{3}, 1 \leq i \leq n\right\},$$
$$A = \left\{a \mid \tfrac{2}{3} \geq a_i \geq \tfrac{1}{3}, \exists x \in \{0,1\}^n : a^T x = 1\right\},$$
$$W = U \setminus A.$$

We define a correspondence between the set of linear separations of B_n^2 and the set of connected components of W: if a vector $w \in \mathbf{R}^n$ defines a linear separation of B_n^2, then the vector $a = (a_1, a_2, \ldots, a_n)$ which specifies the corresponding connected component of W is computed by

$$a_i = \frac{1}{2} + \frac{w_i}{6 \max\limits_{1 \leq i \leq n} |w_i|}.$$

This correspondence can be seen as a continuous mapping. Consequently, for any two different linear separations of B_n^2 there will be at least two different corresponding connected components of W. Then, keeping in mind Lemmas 1 and 2, we can conclude that W has at least $n!$ connected components.

From [1] we have a lower bound $\Omega(\log c(W))$ for the complexity of any algorithm solving the considered subclass of Knapsack problems, where $c(W)$ is the number of connected components of W. Therefore any algorithm solving $\mathrm{KP}_{\mathbf{R}}$ must have time complexity $\Omega(\log n!) = \Omega(n \log n)$. On the other hand, if k is a constant, f is any continuous function of one variable, and if $a_{\min} > 1/3$, then $f(1/a_{\min}) = O(1)$. Thus, if one assumes that there is an algorithm for $\mathrm{KP}_{\mathbf{R}}$ with time complexity $o(n \log n) \cdot f(1/a_{\min})$ for certain continuous function f, then, in particular, for $a_{\min} > 1/3$ this algorithm will have time complexity $o(n \log n)$, which contradicts the lower bound $\Omega(n \log n)$. $\square$

Clearly, an analogous lower bound holds for the complexity of the classical Boolean Knapsack problem $a^T x = b$ with integer coefficients. This problem is equivalent to the equation $\bar{a}^T x = 1$, where $\bar{a} = \frac{1}{b} a$. If $\bar{a}_{\min} = \frac{a_{\min}}{b}$ is the minimal entry in $\bar{a}$, we obtain the lower bound $\Omega(n \log n) \cdot f(1/\bar{a}_{\min}) = \Omega(n \log n) \cdot f(b/a_{\min})$. 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 on the known lower bound $\Omega(n^2)$.

4 Algorithms for $\mathrm{KP}_{\mathbf{R}}$

In this section, we present and analyze the complexity of algorithms for $\mathrm{KP}_{\mathbf{R}}$ under the Blum-Shub-Smale computational model.

4.1 "Pseudopolynomial" Algorithm

One direction of research in complexity theory over $\mathbf{R}$ can be seen as seeking appropriate definitions for some basic notions, corresponding to the traditional terminology of the classical complexity theory. It is difficult to discover universal definitions which would be expedient for all possible numerical problems. As Smale notes in a related discussion [9], this only can be done by "taking into account the particular numerical analysis setting, its limitations, and just what a good algorithm can be expected to accomplish."

In this type of attempt, we look for algorithms for $\mathrm{KP}_{\mathbf{R}}$ to be similar to the pseudopolynomial algorithms for the Knapsack problem over $\mathbf{Z}$. We shall call an algorithm for $\mathrm{KP}_{\mathbf{R}}$ *pseudopolynomial* if its time complexity is bounded by a polynomial in the input size n and the number $1/\delta(a)$, where $\delta(a) = \min\left\{|a^T z| : a^T z \neq 0,\ z \in \{-1, 0, 1\}^n\right\}$. Note that if an algorithm for $\mathrm{KP}_{\mathbf{R}}$ is pseudopolynomial according to this definition, then for integer inputs it will also be pseudopolynomial according to classical complexity theory. The following result holds.

Theorem 2 *There is a pseudopolynomial* $O\left(\frac{n}{\delta(a)}\right)$ *time and space algorithm for* $\mathrm{KP}_{\mathbf{R}}$.

Proof Let us suppose first that the value of $\delta(a)$ is known in advance. We partition the interval $[0,1]$ into $\lfloor 1/\delta(a)\rfloor + 1$ "cells" $C_j = [(j-1)\,\delta(a), j\,\delta(a))$, $1 \le j \le \lfloor 1/\delta(a)\rfloor + 1$. In particular, $0 \in C_1 = [0, \delta(a))$ and $1 \in C_{\lfloor 1/\delta(a)\rfloor+1} = [\lfloor 1/\delta(a)\rfloor\,\delta(a), \lfloor 1/\delta(a)\rfloor\,\delta(a)+1)$.

According to the definition of $\delta(a)$ above, there are no $x, y \in \{0,1\}^n$ for which $a^T x \neq a^T y$, and such that $a^T x$ and $a^T y$ fall into the same cell. Therefore $\mathrm{KP}_{\mathbf{R}}$ can be solved by the dynamic programming algorithm with time complexity and memory space $O(n/\delta(a))$.

Consider now the general case when the value of $\delta(a)$ is unknown. Initially we set $\delta = 1$, so that at the beginning we have two cells $C_1 = [0,1)$ and $C_2 = [1,2)$. Then we start the dynamic programming algorithm. At the first step, the variable x_1 is included. We obtain the values 0 (for $x_1 = 0$) and a_1 (for $x_1 = 1$). We change the value of δ to a_1 and then continue with the next step of the algorithm.

After performing the jth step, for a current δ and a corresponding set of cells, we have a set V of different values of the form

$$\sum_{i=1}^{k} a_i x_i, \text{ where } x_i \in \{0,1\},\ 1 \le k \le j, \tag{1}$$

obtained at steps number $1, 2, \ldots, j$. If all of these values belong to different cells, then we proceed with the next step of dynamic programming algorithm. Otherwise, there is at least one cell C which contains two or three different values from V. This may happen in the following two cases.

(a) A new value $v = \sum_{i=1}^{j} a_i x_i$, where $x_j = 1$, falls into a cell C which already contains a value of the form (1) for some $k \le j-1$.

(b) Two new values $v' = \sum_{i=1}^{j} a_i x'_i$ with $x'_j = 1$ and $v'' = \sum_{i=1}^{j} a_i x''_i$ with $x''_j = 1$, fall into the same cell C. Clearly, this can happen only if v' and v'' originate from two values lying in neighbouring cells. Note that if C also contains a value obtained at a prior step, then after the jth step C may contain three different values.

After we identify a cell wich has been established as having more than one value, we recompute δ. To this end, we compute the minimal difference between values of the form (1), corresponding to pairs of neighbouring points in the interval $[0,1]$. The minimum is taken over all such pairs. Then we assign the obtained minimum as a new value of δ, next partition the interval $[0,1]$ into a set of cells according to this value, and then perform the next step of dynamic programming algorithm.

Clearly, each of the cells of this new cell partition will contain not more than one value from the set V. Keeping in mind the definition of $\delta(a)$, it is obvious that throughout the work of the algorithm the value of δ is maintained as greater than $\delta(a)$. It is also easy to realize that a new value of δ and a corresponding new cell partition of the interval $[0,1]$ can be found in time $O(1/\delta) = O(1/\delta(a))$. Thus, the overall time complexity of the algorithm is $O(n/\delta(a))$. Moreover, it can be seen as using the same order of memory space. $\square$

It is not difficult to see that the described algorithm is pseudopolynomial according to the classical complexity theory, when it is applied to the equation $a^T x = b$ with integer coefficients. In this case the above equation can be represented in the form $\bar{a}^T x = 1$ with $\bar{a} = \frac{1}{b}a$, for which the algorithm applies. Clearly, for such an input, the last value of δ will satisfy the condition $\delta \geq \delta(a) \geq 1/b$. Hence, in the extreme (worst) case when $\delta = \delta(a) = 1/b$, the complexity of our algorithm will equal the complexity of the dynamic programming algorithm for the Boolean Knapsack problem. Otherwise, it will be superior to the latter algorithm.

A similar situation takes place if we consider $\mathrm{KP}_{\mathbf{R}}$ with rational coefficients $a_i = p_i/q_i$, $p_i, q_i \in \mathbf{Z}_+$. This problem can be directly solved by using the algorithm of Theorem 2 in time $O(n/\delta(a))$. In order to apply the classical dynamic programming algorithm, we first obtain an equivalent formulation $\sum_{i=1}^n c_i x_i = M$ with integer coefficients $c_i = a_i M$, where $M = LCM(q_1, q_2, \ldots, q_n)$ is the least common multiple of the integers $q_1, q_2, \ldots, q_n$. Then this problem can be solved by the dynamic programming algorithm in time $O(nM)$. It is easy to see that the last value of δ satisfies the condition $\delta \geq \delta(a) \geq 1/M$. Therefore, our algorithm will be faster than the dynamic programmic algorithm, except in the case where $\delta = \delta(a) = 1/M$, when both algorithms have the same time complexity.

4.2 Approximation Algorithms

Let us consider the following optimization version of $\mathrm{KP}_{\mathbf{R}}$.

$$\text{Given } a \in \mathbf{R}^n_+, \text{ find } x_0 \in \{0,1\}^n \text{ that minimizes } \left| a^T x - 1 \right|.$$

Our goal is to find approximation algorithms which solve this problem within a given error ε. In other words, we look for a solution to the following approximation Knapsack problem.

$$(\varepsilon{-}\mathrm{KP}_{\mathbf{R}}) \qquad \begin{array}{l} \text{Given } a \in \mathbf{R}^n_+ \text{ and } \varepsilon \in \mathbf{R}_+, \text{ find } y \in \{0,1\}^n \\ \text{such that } \left| a^T y - 1 \right| \leq \min\limits_{x \in \{0,1\}^n} \left| a^T x - 1 \right| + \varepsilon. \end{array}$$

Our strategy is to reduce this problem defined over the real numbers to an equivalent integer programming problem, which can be solved by applying a known integer algorithm. We notice that in the first algorithm we follow the idea of Ibarra and Kim [7] for scaling the equation coefficients.

Theorem 3 *There exist* $O\left(\frac{n^2}{\varepsilon}\right)$ *and* $O\left(\frac{n}{\varepsilon\, a_{\min}}\right)$ *time algorithms for* $\varepsilon-\mathrm{KP}_{\mathbf{R}}$.

Proof Let $x_0 \in \operatorname{Argmin}_{x\in\{0,1\}^n} \left|a^T x - 1\right|$ and $y \in \operatorname{Argmin}_{x\in\{0,1\}^n} \left|b^T x - 1\right|$ for some $b \in \mathbf{Q}^n$. We have

$$\left|a^T y - 1\right| \le \left|b^T y - 1\right| + \left|(a-b)^T y\right| \le \left|b^T x_0 - 1\right| + \left|(a-b)^T y\right| \le \\ \left|a^T x_0 - 1\right| + \left|(b-a)^T x_0\right| + \left|(a-b)^T y\right|.$$

We shall consider two ways to choose b.

1. $b = \frac{\lfloor Ma\rceil}{M}$, where $M = \left\lceil \frac{n}{\varepsilon}\right\rceil$, and $\lfloor Ma\rceil$ denotes the vector whose components are the closest integers to the corresponding components of Ma. For arbitrary $x \in \{0,1\}^n$ we have

$$\left|(a-b)^T x\right| = \frac{1}{M}\left|(Ma - \lfloor Ma\rceil)^T\right| x \le \frac{\varepsilon}{n}\cdot\frac{1}{2}\cdot\mathbf{1}^T\cdot\mathbf{1} = \frac{\varepsilon}{2}.$$

(Here $\mathbf{1}^T$ denotes the n-dimensional vector with all components equal to 1.) Then it follows that

$$\left|a^T y - 1\right| \le \left|a^T x_0 - 1\right| + \varepsilon.$$

The problem of finding $\min_{x\in\{0,1\}^n} \left|b^T x - 1\right|$ is equivalent to the problem of finding $\min_{x\in\{0,1\}^n} \left|\lfloor Ma\rceil^T x - M\right|$. We have that if $y \in \operatorname{Argmin}_{x\in\{0,1\}^n} \left|b^T x - 1\right|$ then $b^T y < 2$. Then the latter optimization problem can be solved in $2Mn$ steps and memory space by using dynamic programming. Thus, we obtain that the complexity of this algorithm is $O\left(\frac{n^2}{\varepsilon}\right)$.

2. $b = \frac{\lfloor Ma\rceil}{M}$, where $M = \left\lceil \frac{1}{a_{\min}\,\delta}\right\rceil$, $\delta = \min\left\{1, \frac{\varepsilon}{3}\right\}$. We have

$$b \in \left[a - \tfrac{1}{2M}, a + \tfrac{1}{2M}\right] \subseteq \left[a - \tfrac{a_{\min}\delta}{2}, a + \tfrac{a_{\min}\delta}{2}\right] \subseteq \left[a\left(1 - \tfrac{\delta}{2}\right), a\left(1+\tfrac{\delta}{2}\right)\right].$$

If $x_0 \in \operatorname{Argmin}_{x\in\{0,1\}^n} \left|a^T x - 1\right|$, then $a^T x_0 < 2$, so that

$$\left|(b-a)^T x_0\right| \le \frac{\delta}{2} a^T x_0 \le \delta \le \frac{\varepsilon}{3}.$$

If $y \in \operatorname{Argmin}_{x\in\{0,1\}^n} \left|b^T x - 1\right|$, then $\left(1 - \frac{\delta}{2}\right) a^T y \le b^T y \le 2$, so that

$$\left|(a-b)^T y\right| \le \frac{\delta}{2} a^T y \le \frac{\delta}{1-\delta/2} \le \frac{\varepsilon/3}{1/2} = \frac{2\varepsilon}{3}.$$

Thus

$$\left|a^T y - 1\right| \le \left|a^T x_0 - 1\right| + \varepsilon.$$

As in Case 1, the problem of finding $\min_{x\in\{0,1\}^n} \left|b^T x - 1\right|$ is equivalent to the problem of finding $\min_{x\in\{0,1\}^n} \left|\lfloor Ma\rceil^T x - M\right|$, which can be solved in $2Mn$ steps and memory space by using dynamic programming. Hence the complexity of this algorithm is $O\left(\frac{n}{\varepsilon\, a_{\min}}\right)$. □

5 Concluding Remarks

In this paper we have presented some results related to the Boolean Knapsack problem with real coefficients; in particular, we have obtained a lower bound $\Omega(n \log n) \cdot f(1/a_{\min})$ for $\mathrm{KP}_{\mathbf{R}}$. Bearing in mind the other lower bound $\Omega(n^2)$, we can ask the question: is there an algorithm for $\mathrm{KP}_{\mathbf{R}}$ with time complexity $O(n^{2-\delta} f(1/a_{\min}))$, $\delta \geq 0$?

We have suggested a definition for pseudopolynomial algorithm and have designed such an algorithm for $\mathrm{KP}_{\mathbf{R}}$. We think that this definition could be a starting point for defining this theoretical construct for larger classes of numerical problems.

We notice also that all the obtained results can be adapted for the case of an *integer* Knapsack problem with real coefficients ($\mathrm{IKP}_{\mathbf{R}}$), when looking for a vector $x \in \mathbf{Z}_+^n$ such that $a^T x = 1$. It is easy to see that the lower bound of Theorem 1 is still valid in this case. Note that another lower bound for the complexity of $\mathrm{IKP}_{\mathbf{R}}$ is $\Omega(g(n))$, where $g(n)$ is an arbitrary function [5]. For the classical Knapsack problem $a^T x = b$ with integer coefficients we get a lower bound $\Omega(n \log n) \cdot f(b/a_{\min})$. Additionally, the result of Theorem 2 holds, provided that the parameter $\delta(a)$ in the definition of a pseudopolynomial algorithm is determined as $\delta(a) = \min\left\{|a^T z| : a^T z \neq 0,\ z \in \mathbf{Z}_+^n, z_i \in [-\lfloor 1/a_i \rfloor, \lfloor 1/a_i \rfloor]\right\}$. The two approximation algorithms of Theorem 3 can be straightforwardly modified for solving the corresponding approximation version of $\mathrm{IKP}_{\mathbf{R}}$. The complexity of the first algorithm becomes $O(n^2(-\log a_{\min})^2/\varepsilon)$, while the complexity of the second algorithm remains the same.

Acknowledgments

The authors would like to thank Bruno Codenotti and Mauro Leoncini for a number of useful discussions, and also Marylin Mell for reading an early version of this paper.

References

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

[2] Blum, L. (1989), Lectures on a theory of computation and complexity over the reals (or an arbitrary ring), TR-89-065, Internat. Comput. Sci. Inst., Berkeley, California.

[3] Blum, L., F. Cucker, M. Shub, and S. Smale (1995), Complexity and Real Computation: a Manifesto, Manuscript.

[4] Blum, L., M. Shub, and S. Smale (1989), On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines, *Bull. Amer. Math. Soc. (NS)* **21**, 1-46.

[5] Brimkov, V.E., and S.S. Danchev (1997), Real data - integer solution problems within the Blum-Shub-Smale computational model, *J. Complexity* **13**, 279-300.

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

[7] Ibarra, O, and C.E. Kim (1975), Fast approximation algorithms for the knapsack and sum of subset problems, *J. ACM* **22**, 463-468.

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

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