Fast Integer Programming in Fixed Dimension

Friedrich Eisenbrand

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, eisen@mpi-sb.mpg.de

Abstract. It is shown that the optimum of an integer program in fixed dimension, which is defined by a fixed number of constraints, can be computed with $O(s)$ basic arithmetic operations, where s is the binary encoding length of the input. This improves on the quadratic running time of previous algorithms which are based on Lenstra's algorithm and binary search.

It follows that an integer program in fixed dimension, which is defined by m constraints, each of binary encoding length at most s, can be solved with an expected number of $O(m + \log(m)\, s)$ arithmetic operations using Clarkson's random sampling algorithm.

1 Introduction

An *integer program* is a problem of the following kind. Given an integral matrix $A \in \mathbb{Z}^{m \times n}$ and integral vectors $b \in \mathbb{Z}^m$, $d \in \mathbb{Z}^n$, determine

$$\max\{d^T x \mid Ax \leqslant b,\, x \in \mathbb{Z}^n\}. \tag{1}$$

It is well known [6] that integer programming is NP-complete. The situation changes, if the number of variables or the *dimension* is fixed. For this case, Lenstra [13] showed that (1) can be solved in polynomial time. Lenstra's algorithm does not solve the integer programming problem directly. Instead, it is an algorithm for the *integer feasibility problem*. Here, the task is to find an integer point which satisfies all the constraints, or to assure that $Ax \leqslant b$ is integer infeasible. If $Ax \leqslant b$ consists of m constraints, each of binary encoding length $O(s)$, then Lenstra's algorithm requires $O(m + s)$ arithmetic operations on rational numbers of size $O(s)$. The actual integer programming problem (1) can then be solved via *binary search*. It is known [15, p. 239] that, if there exists an optimal solution, then there exists one with binary encoding length $O(s)$. Consequently, the integer programming problem can be solved with $O(m\, s + s^2)$ arithmetic operations on $O(s)$-bit numbers. Lenstra's algorithm was subsequently improved [9, 1] by reducing the dependence of the complexity on the dimension n. However, these improvements do not affect the asymptotic complexity of the integer programming problem in fixed dimension. Unless explicitely stated, we from now-on assume that the dimension n is fixed.

Clarkson [2] presented a random sampling algorithm to reduce the dependence of the complexity on the number of constraints.[1] His result is the following. An integer program which is defined by m constraints can be solved with $O(m)$ basic operations and $O(\log m)$ calls to an algorithm which solves an integer program defined by a fixed size subset of the constraints, see also [7].

In light of these results, we are motivated to find a faster algorithm for the integer programming problem in fixed dimension with a fixed number of constraints. It is known [4] that the 2-dimensional integer programming problem with a fixed number of constraints can be solved in linear time. We generalize this to any fixed dimension.

Theorem 1. *An integer program of binary encoding length s in fixed dimension, which is defined by a fixed number of constraints, can be solved with $O(s)$ arithmetic operations on rational numbers of binary encoding length $O(s)$.*

With Clarkson's result, Theorem 1 implies that an integer program which is defined by m constraints, each of binary encoding length $O(s)$ can be solved with an expected number of $O(m + \log(m)\, s)$ arithmetic operations on rational numbers of binary encoding length $O(s)$. Our result was also motivated by the following fact. The greatest common divisor of two integers can be formulated as an integer program in fixed dimension with a fixed number of constraints, see, e.g., [11]. Our result matches the complexity of the integer programming approach to the gcd with the complexity of the Euclidean algorithm.

Outline of our method. As in Lenstra's algorithm, we make use of the lattice width concept. Let $K \subseteq \mathbb{R}^n$ be a full-dimensional convex body. The *width* of K *along a direction* $c \in \mathbb{R}^n$ is the quantity $w_c(K) = \max\{c^T x \mid x \in K\} - \min\{c^T x \mid x \in K\}$. The *width* of K, $w(K)$, is the minimum of its widths along nonzero integral vectors $c \in \mathbb{Z}^n \setminus \{0\}$. If K does not include any lattice points, then K must be "flat". This fact is known as Khinchin's *flatness theorem* (see [10]).

Theorem 2 (Flatness theorem). *There exists a constant f_n depending only on the dimension n, such that each full-dimensional convex body $K \subseteq \mathbb{R}^n$, containing no integer points has width at most f_n.*

This fact is exploited in Lenstra's algorithm [13,8] for the integer feasibility problem as follows. If one has to decide, whether a full-dimensional polyhedron P is integer feasible or not, one computes a *flat direction* of P, which is an integral vector $c \in \mathbb{Z}^n \setminus \{0\}$ such that $w(P) \leqslant w_c(P) \leqslant \gamma\, w(P)$ holds for some constant γ depending on the dimension. If $w_c(P)$ is larger than $\gamma\, f_n$, then P must contain integer points by the flatness theorem. Otherwise, an integer point of P must lie in one of the constant number of $(n-1)$-dimensional polyhedra

$$P \cap (c^T x = \delta), \text{ where } \delta \in \mathbb{Z} \cap [\min\{c^T x \mid x \in P\}, \max\{c^T x \mid x \in P\}].$$

[1] Clarkson claims a complexity of $O(m + \log(m)\, s)$ because he mistakenly relied on algorithms from the literature [13,9,5] for the integer programming problem with a fixed number of constraints, which actually only solve the *integer feasibility problem.*

In this way one can reduce the integer feasibility problem in dimension n to a constant number of integer feasibility problems in dimension $n-1$.

Our approach is to let the objective function slide into the polyhedron until the with of the truncated polyhedron $P_\pi = P \cap (d^T x \geqslant \pi)$ is sandwiched between $f_n + 1$ and $\gamma\,(f_n+1)$. In this way, we assure that the optimum to the integer programming problem lies in the truncation P_π which is still flat along some integer vector c, thereby reducing the *integer programming problem* over an n-dimensional polyhedron to a constant number of *integer programming problems* over the $(n-1)$-dimensional polyhedra

$$P_\pi \cap (c^T x = \delta), \text{ where } \delta \in \mathbb{Z} \cap [\min\{c^T x \mid x \in P_\pi)\}, \max\{c^T x \mid x \in P_\pi\}].$$

The problem of determining the correct parameter π is referred to as the *approximate parametric lattice width problem.* The 2-dimensional integer programming algorithm of Eisenbrand and Rote [3] makes already use of this concept. In this paper we generalize this approach to any dimension.

1.1 Notation

A *polyhedron* P is a set of the form $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$, for some matrix $A \in \mathbb{R}^{m\times n}$ and some vector $b \in \mathbb{R}^m$. The polyhedron is *rational* if both A and b can be chosen to be rational. If P is bounded, then P is called a *polytope.* The *dimension* of P is the dimension of the affine hull of P. The polyhedron $P \subseteq \mathbb{R}^n$ is *full-dimensional*, if its dimension is n. An inequality $c^T x \leqslant \delta$ defines a *face* $F = \{x \in P \mid c^T x = \delta\}$ of P, if $\delta \geqslant \max\{c^T x \mid x \in P\}$. If $F \neq \emptyset$ is a face of dimension 0, then F is called a *vertex* of P. A *simplex* is full-dimensional polytope $\Sigma \subseteq \mathbb{R}^n$ with $n+1$ vertices. We refer to [14] and [15] for further basics of polyhedral theory.

The *size* of an integer z is the number $\mathrm{size}(z) = 1 + \lceil \log_2(|z|+1)\rceil$. The size of a rational is the sum of the sizes of its numerator and denominator. Likewise, the size of a matrix $A \in \mathbb{Z}^{m\times n}$ is the number of bits needed to encode A, i.e., $\mathrm{size}(A) = \sum_{i,j} \mathrm{size}(a_{i,j})$, see [15, p. 29]. If a polyhedron P is given as $P(A,b)$, then we denote $\mathrm{size}(A) + \mathrm{size}(b)$ by $\mathrm{size}(P)$. A polytope can be represented by a set of constraints, as well as by the set of its vertices. In this paper we concentrate on polyhedra in fixed dimension with a fixed number of constraints. In this case, if a rational polytope is given by a set of constraints $Ax \leqslant b$ of size s, then the vertex representation $\mathrm{conv}\{v_1, \ldots, v_k\}$ can be computed in constant time and the vertex representation has size $O(s)$. The same holds vice versa.

A rational *lattice* in $\mathbb{R}^n$ is a set of the form $\Lambda = \{Ax \mid x \in \mathbb{Z}^n\}$, where $A \in \mathbb{Q}^{n\times n}$ is a nonsingular matrix. This matrix is a *basis* of Λ and we say that Λ is generated by A and we also write $\Lambda(A)$ to denote a lattice generated by a matrix A. A *shortest vector* of Λ is a nonzero member $0 \neq v \in \Lambda$ of the lattice with minimal euclidean norm $\|v\|$. We denote the length of a shortest vector by $\mathrm{SV}(\Lambda)$.

2 Proof of Theorem 1

Suppose we are given an integer program (1) in fixed dimension with a fixed number of constraints of binary encoding length s. It is very well known that one can assume without loss of generality that the polyhedron $P = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$ is bounded and full-dimensional and that the objective is to find an integer vector with maximal first component. A transformation to such a *standard form problem* can essentially be done with a constant number of Hermite-Normal-Form computations and linear programming. Since the number of constraints is fixed, this can thus be done with $O(s)$ arithmetic operations on rational numbers of size $O(s)$.

Furthermore, we can assume that P is a *two-layer simplex Σ*. A two-layer simplex is a simplex, whose vertices can be partitioned into two sets V and W, such that the first components of the elements in V and W agree, i.e., for all $v_1, v_2 \in V$ one has $v_1(1) = v_2(1)$ and for all $w_1, w_2 \in W$ one has $w_1(1) = w_2(1)$. An integer program over P can be reduced to the disjunction of integer programs over two-layer simplices as follows. First, compute the list of the first components $\alpha_1, \dots, \alpha_\ell$ of the vertices of P in decreasing order. The optimal solution of IP over P is the largest optimal solution of IP over polytopes

$$P_i = P \cap (x(1) \leqslant \alpha_i) \cap (x(1) \geqslant \alpha_{i+1}),\ i = 1, \dots, \ell - 1. \tag{2}$$

Carathéodory's theorem, see [15, p. 94], implies that each P_i is covered by the two-layer simplices, which are spanned by the vertices of P_i. Thus we assume that an integer program has the following form.

Problem 1 (IP). Given an integral matrix $A \in \mathbb{Z}^{n+1 \times n}$ and an integral vector $b \in \mathbb{Z}^{n+1}$ which define a two-layer simplex $\Sigma = \{x \in \mathbb{R}^n \mid Ax \leqslant b\}$, determine

$$\max\{x(1) \mid x \in P \cap \mathbb{Z}^n\}. \tag{3}$$

The *size* of an IP is the sum of the sizes of A and b.

Our main theorem is proved by induction on the dimension. We know that it holds for $n = 1, 2$ [4,17]. The induction step is by a series of reductions, for which we now give an overview.

(Step 1) We reduce IP over a two-layer simplex Σ to the problem of determining a parameter π, such that the width of the truncated simplex $\Sigma \cap (x(1) \geqslant \pi)$ is sandwiched between $f_n + 1$ and $(f_n + 1) \cdot \gamma$, where γ is a constant which depends on the dimension only. This problem is the *approximate parametric lattice width* problem.

(Step 2) We reduce the approximate parametric lattice width problem to an *approximate parametric shortest vector problem.* Here one is given a lattice basis A and parameters U and k. The task is to find a parameter p such that the length of the shortest vector of the lattice generated $A_{p,k}$ is sandwiched between U and $\gamma' U$, where γ' is a constant which depends on the dimension only. Here $A_{p,k}$ denotes the matrix, which evolves from A by scaling the first k rows with p.

(Step 3) We show that an approximate parametric shortest vector problem can be solved in linear time with a sequence of calls to the LLL-algorithm.

The linear complexity of the parametric shortest vector problem carries over to the integer programming problem with a fixed number of constraints, if we can ensure the following conditions for each reduction step.

(C-1) A problem of size s is reduced to a constant number of problems of size $O(s)$.
(C-2) The size of the rational numbers which are manipulated in the course of the reduction of a problem of size s, do not grow beyond $O(s)$.

At the end of each reduction step, we clarify that the conditions (C-1) and (C-2) are fulfilled.

2.1 Reduction to the Parametric Lattice Width Problem

The *parametric lattice width problem* for a two-layer simplex Σ is defined as follows.

Problem 2 (PLW). Given a two-layer simplex $\Sigma \subseteq \mathbb{R}^n$ and some $K \in \mathbb{N}$, find a parameter π such that the width of the truncated simplex $\Sigma_\pi = \Sigma \cap (x(1) \geqslant \pi)$ satisfies

$$K \leqslant w(\Sigma_\pi) \leqslant 2^{(n+1)/2+2} \cdot \lceil \sqrt{n} \rceil \cdot K, \tag{4}$$

or assert that $w(\Sigma) \leqslant 2^{(n+1)/2+2} \cdot \lceil \sqrt{n} \rceil \cdot K$.

Let us motivate this concept. Denote the constant $2^{(n+1)/2+2} \cdot \lceil \sqrt{n} \rceil$ by γ. Run an algorithm for PLW on input Σ and $f_n + 1$. If this returns a parameter π such that $f_n + 1 \leqslant w(\Sigma_\pi) \leqslant \gamma\,(f_n + 1)$, then the optimum solution of the IP over Σ must be in the truncated simplex Σ_π. This follows from the fact that we are searching an integer point with maximal first component, and that the truncated polytope has to contain integer points by the flatness theorem. On the other hand, this truncation Σ_π is flat along some integer vector c. Thus the optimum of IP is the largest optimum of the constant number of the $n-1$-dimensional integer programs

$$\max\{x(1) \mid x \in (\Sigma_\pi \cap (c^T x = \alpha)) \cap \mathbb{Z}^n\}, \tag{5}$$

where $\alpha \in \mathbb{Z} \cap [\min\{c^T x \mid x \in \Sigma_\pi\}, \max\{c^T x \mid x \in \Sigma_\pi\}]$. This means that we have reduced the integer programming problem over a two-layer simplex in dimension n to a constant number of integer programming problems in dimension $n-1$ with a fixed number of constraints.

If the algorithm for PLW asserts that $w(\Sigma) \leqslant \gamma\, K$, then Σ itself is already flat along an integral direction c. Similarly in this case, the optimization problem can be reduced to a constant number of optimization problems in lower dimension.

Analysis. If the size of Σ and K is at most s and PLW can be solved in $O(s)$ steps with rational numbers of size $O(s)$, then the parameter π which is returned has size $O(s)$. A flat direction of Σ_π can be computed with $O(s)$ arithmetic operations on rationals of size $O(s)$. In fact, a flat direction is a by-product of our algorithm for the approximate parametric shortest vector problem below. It follows that the constant number of $n-1$-dimensional IP's (5) have size $O(s)$. These can then be transformed into IP's in standard form with $n-1$ variables and a constant number of constraints, in $O(s)$ steps. Consequently we have the following lemma.

Lemma 1. *Suppose that PLW for a two-layer simplex Σ and parameter K with* $\text{size}(\Sigma) + \text{size}(K) = s$ *can be solved with $O(s)$ operations on rational numbers of size $O(s)$, then IP over Σ can also be solved with $O(s)$ operations with rational numbers of size $O(s)$.*

2.2 Reduction to the Approximate Parametric Shortest Vector Problem

In this section we show how to reduce PLW for a two-layer simplex $\Sigma = \text{conv}(V \cup W)$ and parameter K to an *approximate parametric shortest vector* problem. The width of a polyhedron is invariant under translation. Thus we can assume that $0 \in V$ and that the first component of the vertices in W is negative.

Before we formally describe our approach, let us explain the idea with the help of Figure 1. Here we have a two-layer simplex Σ in 3-space. The set V

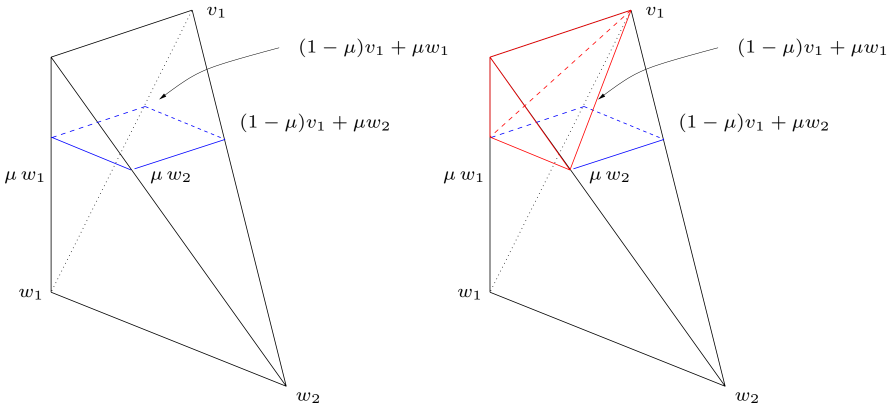

Fig. 1. Solving PLW.

consists of the points 0 and v_1 and W consists of w_1 and w_2. The picture on the left describes a particular point in time, where the objective function slid into Σ. So we consider the truncation $\Sigma_\pi = \Sigma \cap (x(1) \geqslant \pi)$ for some $\pi \geqslant w_1(1)$. This truncation is the convex hull of the points

$$0, v_1, \mu w_1, \mu w_2, (1-\mu)v_1 + \mu w_1, (1-\mu)v_1 + \mu w_2, \tag{6}$$

where $\mu = \pi / w_1(1)$. Now consider the simplex $\Sigma_{V,\mu W}$, which is spanned by the points $0, v_1, \mu w_1, \mu w_2$. This simplex is depicted on the right in Figure 1. If this simplex is scaled by 2, then it contains the truncation Σ_π. This is easy to see, since the scaled simplex contains the points $2(1-\mu)\,v_1$, $2\,\mu\,w_1$ and $2\,\mu\,w_2$. So we have the condition $\Sigma_{V,\mu W} \subseteq \Sigma_\pi \subseteq 2\,\Sigma_{V,\mu W}$. From this we can infer the important observation

$$w(\Sigma_{V,\mu W}) \leqslant w(\Sigma_\pi) \leqslant 2\,w(\Sigma_{V,\mu W}). \tag{7}$$

This means that we can solve PLW for Σ, if we can determine a $\mu \geqslant 0$, such that the width of the simplex $\Sigma_{V,\mu W}$ is sandwiched between K and $(\gamma/2)\,K$, where γ denotes the constant $2^{(n+1)/2+2} \cdot \lceil \sqrt{n} \rceil$. We now generalize this observation with the following lemma. A proof is straightforward.

Lemma 2. *Let $\Sigma = \mathrm{conv}(V \cup W) \subseteq \mathbb{R}^n$ be a two-layer simplex, where $0 \in V$, $w(1) < 0$ for all $w \in W$ and let π be a number with $0 \geqslant \pi \geqslant w(1)$, $w \in W$. The truncated simplex $\Sigma_\pi = \Sigma \cap (x(1) \geqslant \pi)$ is contained in the simplex $2\,\Sigma_{V,\mu W}$, where $\Sigma_{V,\mu W} = \mathrm{conv}(V \cup \mu W)$, where $\mu = \pi / w(1)$, $w \in W$. Furthermore, the following relation holds true*

$$w(\Sigma_{V,\mu W}) \leqslant w(\Sigma_\pi) \leqslant 2\,w(\Sigma_{V,\mu W}). \tag{8}$$

Before we inspect the with of $\Sigma_{V,\mu\,W}$, let us introduce some notation. We define for an $n \times n$-matrix A, the matrix $A_{\mu,k}$, as

$$A_{\mu,k}(i,j) = \begin{cases} \mu \cdot A(i,j), & \text{if } i \leqslant k, \\ A(i,j), & \text{otherwise.} \end{cases} \tag{9}$$

In other words, the matrix $A_{\mu,k}$ results from A by scaling the first k rows with μ.

Suppose that $V = \{0, v_1, \ldots, v_{n-k}\}$ and $W = \{w_1, \ldots, w_k\}$. Let $A \in \mathbb{R}^{n \times n}$ be the matrix, whose rows are the vectors $w_1^T, \ldots, w_k^T, v_1^T, \ldots, v_{n-k}^T$ in this order. The width of $\Sigma_{V,\mu\,W}$ along the vector c can be bounded as

$$\|A_{\mu,k}\,c\|_\infty \leqslant w_c(\Sigma_{V,\mu\,W}) \leqslant 2\,\|A_{\mu,k}\,c\|_\infty, \tag{10}$$

and consequently as

$$(1/\sqrt{n})\,\|A_{\mu,k}\,c\| \leqslant w_c(\Sigma_{V,\mu\,W}) \leqslant 2\,\|A_{\mu,k}\,c\|. \tag{11}$$

The width of $\Sigma_{V,\mu\,W}$ is the minimum width along a nonzero vector $c \in \mathbb{Z}^n - \{0\}$. Thus we can solve PLW for a two-layer simplex with parameter K if we can determine a parameter $\mu \in \mathbb{Q}_{>0}$ with

$$\lceil \sqrt{n} \rceil \cdot K \leqslant \mathrm{SV}(\Lambda(A_{\mu,k})) \leqslant \gamma/4 \cdot K. \tag{12}$$

By substituting $U = \lceil \sqrt{n} \rceil \cdot K$ this reads as follows. Determine a $\mu \in \mathbb{Q}_{>0}$ such that

$$U \leqslant \mathrm{SV}(\Lambda(A_{\mu,k})) \leqslant 2^{(n+1)/2} \cdot U. \tag{13}$$

If such a $\mu > 0$ exists, we distinguish two cases. In the first case one has $0 < \mu < 1$. Then $\pi = w(1) \cdot \mu$ is a solution to PLW. In the second case, one has $1 < \mu$ and it follows that $w(\Sigma) \leqslant \gamma K$. If such a $\mu \in \mathbb{Q}_{>0}$ does not exist, then $\mathrm{SV}(\Lambda(A_{\mu,k})) < U$ for each $\mu > 0$. Also then we assert that $w(\Sigma) \leqslant \gamma K$.

Thus we can solve PLW for a two-layer simplex $\Sigma = \mathrm{conv}(V \cup W)$ with an algorithm which solves the *approximate parametric shortest vector problem*, which is defined as follows: Given a nonsingular matrix $A \in \mathbb{Q}^{n \times n}$, an integer $1 \leqslant k \leqslant n$, and some $U \in \mathbb{N}$, find a parameter $p \in \mathbb{Q}_{>0}$ such that $U \leqslant \mathrm{SV}(\Lambda(A_{p,k})) \leqslant 2^{(n+1)/2} \cdot U$ or assert that $\mathrm{SV}(\Lambda(A_{p,k})) \leqslant 2^{(n+1)/2} \cdot U$ for all $p \in \mathbb{Q}_{>0}$.

We argue now that we can assume that A is an integral matrix and that 1 is a lower bound on the parameter p we are looking for. Clearly we can scale the matrix A and U with the product of the denominators of the components of A. In this way we can already assume that A is integral. If A is integral, then $(|\det(A)|, 0, \ldots, 0)$ is an element of $\Lambda(A)$. This implies that we can bound p from below by $1/|\det(A)|$. Thus by scaling U and the last $n-k$ rows of A with $|\det(A)|$, we can assume that $p \geqslant 1$. Therefore we formulate the approximate parametric shortest vector problem in its integral version.

Problem 3 (PSV). Given a nonsingular matrix $A \in \mathbb{Z}^{n \times n}$, an integer $1 \leqslant k \leqslant n$, and some $U \in \mathbb{N}$, find a parameter $p \in \mathbb{Q}_{\geqslant 1}$ such that $U \leqslant \mathrm{SV}(\Lambda(A_{p,k})) \leqslant 2^{(n+1)/2} \cdot U$ or assert that $\mathrm{SV}(\Lambda(A_{p,k})) \leqslant 2^{(n+1)/2} \cdot U$ for all $p \in \mathbb{Q}_{\geqslant 1}$ or assert that $\mathrm{SV}(\Lambda(A)) > U$.

By virtue of our reduction to the integral problem, the assertion $\mathrm{SV}(\Lambda(A)) > U$ can never be met in our case. It is only a technicality for the description and analysis of our algorithm below.

Analysis. The conditions (C-1) and (C-2) are straightforward since the binary encoding lengths of the determinant and the products of the denominators are linear in the encoding length of the input in fixed dimension.

Lemma 3. *Suppose that a PSV of size s can be solved with $O(s)$ arithmetic operations on rational numbers of size $O(s)$, then a PLW of size s for a two-layer simplex Σ and parameter K can also be solved with $O(s)$ arithmetic operations on rational numbers of size $O(s)$.*

2.3 Solving the Approximate Parametric Shortest Vector Problem

In the following, we do not treat the dimension as a constant.

The LLL Algorithm

First, we briefly review the *LLL-algorithm* for lattice-basis reduction [12]. We refer the reader to the book of Grötschel, Lovász and Schrijver [8] or von zur Gathen and Gerhard [16] for a more detailed account.

Intuitively, a lattice basis is reduced, if it is "almost orthogonal". Reduction algorithms apply unimodular transformations of a lattice basis from the right, to obtain a basis whose vectors are more and more orthogonal.

The *Gram-Schmidt orthogonalization* $(b_1^*, \dots, b_n^*)$ of a basis $(b_1, \dots, b_n)$ of $\mathbb{R}^n$ satisfies

$$b_j = \sum_{i=1}^{j} \mu_{ji} b_i^*, \; j = 1, \dots, n, \tag{14}$$

where each $\mu_{jj} = 1$. A lattice basis $B \in \mathbb{Z}^{n \times n}$ is *LLL-reduced*, if the following conditions hold for its Gram-Schmidt orthogonalization.

(i) $|\mu_{i,j}| \leqslant 1/2$, for every $1 \leqslant i < j \leqslant n$;
(ii) $\|b_{j+1}^* + \mu_{j+1,j}\, b_j^*\|^2 \geqslant 3/4\, \|b_j^*\|^2$, for $j = 1, \dots, n-1$.

The LLL-algorithm iteratively *normalizes* the basis, which means that the basis is unimodularly transformed into a basis which meets condition (i), and *swaps* two columns if these violate condition (ii). These two steps are repeated until the basis is LLL-reduced. The first column of an LLL-reduced basis is a $2^{(n-1)/2}$-factor approximation to the shortest vector of the lattice.

```
Algorithm 1: LLL
Input:  Lattice basis $A \in \mathbb{Z}^{n \times n}$.
Output: Lattice basis $B \in \mathbb{Z}^{n \times n}$ with $\Lambda(A) = \Lambda(B)$ and
        $\|b_1\| \leqslant 2^{(n-1)/2} \mathrm{SV}(\Lambda(A))$.
(1)   $B \leftarrow A$
(2)   Compute GSO $b_j^*$, $\mu_{ji}$ of $B$ as in equation (14).
(3)   repeat
(4)       foreach $j = 1, \dots, n$
(5)           foreach $i = 1, \dots, j-1$
(6)               $b_j \leftarrow b_j - \lceil \mu_{ji} \rfloor b_i$
(7)       if There is a subscript $j$ which violates condition (ii)
(8)           Swap columns $b_j$ and $b_{j+1}$ of $B$
(9)       Update GSO $b_j^*$, $\mu_{ji}$
(10)  until $B$ is LLL-reduced
(11)  return $B$
```

The key to the termination argument of the LLL-algorithm is the following potential function $\phi(B)$ of a lattice basis $B \in \mathbb{Z}^{n \times n}$:

$$\phi(B) = \|b_1^*\|^{2n} \|b_2^*\|^{2(n-1)} \cdots \|b_1^*\|^2. \tag{15}$$

The potential of an integral lattice basis is always an integer. Furthermore, if B_1 and B_2 are two subsequent bases at the end of the repeat-loop of Algorithm 1, then

$$\phi(B_2) \leqslant \frac{3}{4}\phi(B_1). \tag{16}$$

The potential of the input A can be bounded by $\phi(A) \leqslant (\|a_1\| \cdots \|a_n\|)^{2n}$. The number of iterations can thus be bounded by $O(n(\log \|a_1\| + \ldots + \|a_n\|))$. Step (2) is executed only once and costs $O(n^3)$ operations. The number of operations performed in one iteration of the repeat-loop can be bounded by $O(n^3)$. The rational numbers during the course of the algorithm have polynomial binary encoding length. This implies that the LLL-algorithm has polynomial complexity.

Theorem 3 (Lenstra, Lenstra and Lovász). *Let $A \in \mathbb{Z}^{n\times n}$ be a lattice basis and let A_0 be the number $A_0 = \max\{\|a_j\| \mid j = 1, \ldots, n\}$. The LLL-algorithm performs $O(n^4 \log A_0)$ arithmetic operations on rational numbers, whose binary encoding length is $O(n \log A_0)$.*

An Algorithm for PSV

Suppose we want to solve PSV on input $A \in \mathbb{Z}^{n\times n}$, $U \in \mathbb{N}$ and $1 \leqslant k \leqslant n$. The following approach is very natural. We use the LLL-algorithm to compute approximate shortest vectors of the lattices $\Lambda(A_{p,k})$ for parameters $p = 2^{\lceil \log U \rceil - i}$ with increasing i, until the approximation of the shortest vector, returned by the LLL-algorithm for $\Lambda(A_{p,k})$ is at most $2^{(n-1)/2} \cdot U$.

Before this is done, we try to assert that $\mathrm{SV}(\Lambda(A_{p,k})) \leqslant 2^{(n+1)/2} \cdot U$ holds for all $p \in \mathbb{Q}_{\geqslant 1}$. This is the case if and only if the sub-lattice Λ' of $\Lambda(A)$, which is defined by $\Lambda' = \{v \in \Lambda \mid v(1) = \ldots = v(k) = 0\}$ contains already a nonzero vector of at most this length. A basis B' of Λ' can be read off the Hermite-Normal-Form of A. The first step of the algorithm checks whether the LLL-approximation of the shortest vector of Λ' has length at most $2^{(n-1)/2} \cdot U$. If this is not the case, then there must be a $p \geqslant 1$ such that $\mathrm{SV}(\Lambda(A_{p,k})) > U$.

As the algorithm enters the repeat-loop, we can then be sure that the length of the shortest vector of $\Lambda(B)$ is at least U. In the first iteration, this is ensured by the choice of the initial p and the fact that the length of the shortest vector of Λ' is at least U. In the following iterations, this follows, since the shortest vector of $\Lambda(B)$ has length at least $\|b_1\|/2^{(n-1)/2} > U$. Consider now the iteration where the condition $\|b_1\| \leqslant 2^{(n-1)/2} \cdot U$ is met. If we scale the first k components of b_1 by 2, we obtain a vector $b' \in \Lambda(A_{2\,p,k})$. The length of b' satisfies $\|b'\| \leqslant 2 \cdot \|b_1\| \leqslant 2^{(n+1)/2} \cdot U$. On the other hand, we argued above that $\mathrm{SV}(\Lambda_{2\,p,k}) \geqslant U$. Last, if the condition in step (6) is satisfied, then we can assure that $\mathrm{SV}(\Lambda(A)) > U$. This implies the correctness of the algorithm.

Analysis. Let $B^{(0)}, B^{(1)}, \ldots, B^{(s)}$ be the values of B in the course of the algorithm at the beginning of the repeat-loop (step (5)) and consider two consecutive bases $B^{(k)}$ and $B^{(k+1)}$ of this sequence. Step (8) decreases the potential of $B^{(k)}$.

Algorithm 2: Iterated LLL

```
Input: Lattice basis $A \in \mathbb{Z}^{n\times n}$, parameters $k, U \in \mathbb{N}$, $1 \leqslant k \leqslant n$.
(1)   Compute basis $B'$ of $\Lambda'$, $B' \leftarrow LLL(B')$
(2)   if $\|b'\| \leqslant 2^{(n-1)/2} \cdot U$
(3)       return $\mathrm{SV}(\Lambda(A_{p,k})) \leqslant 2^{(n+1)/2} \cdot U$ for all $p \in \mathbb{Q}_{\geqslant 1}$
(4)   $p \leftarrow 2^{\lceil \log U \rceil + 1}$, $B \leftarrow A_{p,k}$
(5)   repeat
(6)       if $p = 1$
(7)           return $\mathrm{SV}(\Lambda) > U$
(8)       $B \leftarrow B_{1/2,k}$
(9)       $p \leftarrow p/2$
(10)      $B \leftarrow \mathrm{LLL}(B)$
(11)  until $\|b_1\| \leqslant 2^{(n-1)/2} \cdot U$
(12)  return $2p$
```

Thus by (16), we conclude that the number ℓ of iterations performed by the LLL-algorithm in step (10) satisfies

$$\left(\frac{3}{4}\right)^{\ell} \phi(B^{(k)}) \geqslant \phi(B^{(k+1)}). \tag{17}$$

From this we conclude that the overall amount of iterations through the repeat-loop of the calls to the LLL-algorithm in step (10) can be bounded by

$$O(\log \phi(B^{(0)})) = O(\log \phi(A_{U,k})). \tag{18}$$

The potential $\phi(A_{U,k})$ can be bounded by $\phi(A_{U,k}) \leqslant U^{2\,n^2}(\|a_1\| \cdots \|a_n\|)^{2n}$. As in the analysis of the LLL-algorithm, let A_0 be the number $A_0 = \max\{\|a_j\| \mid i = 1, \ldots, n\}$. The overall number of iterations through the repeat-loop of the LLL-algorithm can be bounded by

$$O(n^2(\log U + \log A_0)). \tag{19}$$

Each iteration performs $O(n^3)$ operations. As far as the binary encoding length of the numbers is concerned, we can directly apply Theorem 3 to obtain the next result.

Theorem 4. *Let $A \in \mathbb{Z}^{n\times n}$ be a lattice basis, $U \in \mathbb{N}$ and $1 \leqslant k \leqslant n$ be positive integers. Furthermore let $A_0 = \max\{\|a_j\| \mid j = 1, \ldots, n\}$. The parametric shortest vector problem for A, U and k can be solved with $O(n^5(\log U + \log A_0))$ basic arithmetic operations with rational numbers of binary encoding length $O(n(\log A_0 + \log U))$.*

This shows that the complexity of *PSV* in fixed dimension n is linear in the input size and operates on rationals whose size is also linear in the input. This concludes the proof of Theorem 1. □

As a consequence, we obtain the following result using Clarkson's [2] random sampling algorithm.

Theorem 5. *An integer program* (1) *in fixed dimension n, where the objective vector and each of the m constraints of $Ax \leqslant b$ have binary encoding length at most s, can be solved with an expected amount of $O(m + \log(m)\, s)$ arithmetic operations on rational numbers of size $O(s)$.*

Acknowledgement. Many thanks are due to Günter Rote and to an ESA-referee for many helpful comments and suggestions.

References

1. M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 601–610. ACM Press, 2001.
2. K. L. Clarkson. Las vegas algorithms for linear and integer programming when the dimension is small. *Journal of the Association for Computing Machinery*, 42:488–499, 1995.
3. F. Eisenbrand and G. Rote. Fast 2-variable integer programming. In K. Aardal and B. Gerards, editors, *Integer Programming and Combinatorial Optimization, IPCO 2001*, volume 2081 of *LNCS*, pages 78–89. Springer, 2001.
4. S. D. Feit. A fast algorithm for the two-variable integer programming problem. *Journal of the Association for Computing Machinery*, 31(1):99–113, 1984.
5. A. Frank and É. Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7:49–65, 1987.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* Freemann, 1979.
7. B. Gärtner and E. Welzl. Linear programming—randomization and abstract frameworks. In *STACS 96 (Grenoble, 1996)*, volume 1046 of *Lecture Notes in Comput. Sci.*, pages 669–687. Springer, Berlin, 1996.
8. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer, 1988.
9. R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.
10. R. Kannan and L. Lovász. Covering minima and lattice-point-free convex bodies. *Annals of Mathematics*, 128:577–602, 1988.
11. D. Knuth. *The art of computer programming*, volume 2. Addison-Wesley, 1969.
12. A. K. Lenstra, H. W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Annalen*, 261:515 – 534, 1982.
13. H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538 – 548, 1983.
14. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley, 1988.
15. A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley, 1986.
16. J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, 1999.
17. L. Y. Zamanskij and V. D. Cherkasskij. A formula for determining the number of integral points on a straight line and its application. *Ehkon. Mat. Metody*, 20:1132–1138, 1984.