# Computational Experience with Lenstra's Algorithm*

Liyan Gao† Yin Zhang‡

September, 2002

**Abstract**

Integer programming is an important mathematical approach for many decision-making problems. In this field, a major theoretical breakthrough came in 1983 when H. W. Lenstra, Jr. proposed a polynomial-time algorithm for a general integer programming feasibility problem where the number of variables is fixed. Two key ingredients of Lenstra's algorithm are ellipsoidal approximation of polytopes and lattice basis reduction. However, the lack of practically efficient algorithms and software for the ellipsoidal approximation of polytopes had made it difficult to study the computational properties of Lenstra's algorithm. In this paper, using a newly developed ellipsoidal approximation algorithm as a subroutine, we have implemented a version of Lenstra's algorithm for the general integer programming feasibility problem. Our numerical results on small- to medium-sized test instances indicate that Lenstra's algorithm often examines far fewer nodes than the traditional branch-and-bound approach. Currently, the main bottle-neck in the performance of the algorithm lies in the step of lattice basis reduction.

## 1 Introduction

Many practical decision problems can be formulated as integer programming or mixed integer programming problems, such as transportation scheduling, production planning and telecommunications. Because integer programming (IP) has wide applicability, algorithms for IP are of great interest to researchers. The most common approach to solving integer programming problems is the branch-and-bound method. However, there is no guarantee that the branch-and-bound method will terminate in time polynomial in the size of input data. In this field, binary (or 0-1) IP problems have attracted most of the attention, while few theoretic results have been reported on general IP, and most general IP solvers rely on heuristic algorithms or random sampling to get a "good" feasible solution. Up to today, some small but hard general IP problems remain unsolved. A theoretical breakthrough in general IP came in 1983 when H. W. Lenstra, Jr. showed that general IP feasibility problems in a fixed dimension can be solved in polynomial time. However, the computational properties of Lenstra's algorithm using ellipsoidal approximation have not been fully explored [9]. In

*This research was supported in part by NSF Grants DMS-9973339, and DOE/LANL Contract 03891-99-23.

†The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA. E-mail: lgao@mathworks.com.

‡Department of Computational and Applied Mathematics, Rice University, Houston, TX 77005, USA. Email: yzhang@rice.edu.

this paper, we study issues of building a practical code based on Lenstra's algorithm using ellipsoidal approximation to solve general IP problems.

The general IP feasibility problem can be stated as follows.

> *Given a rational $m \times n$ matrix $A$ and a rational $m-$vector $b$, does the inequality system $Ax \leq b$ have an integral solution $x$?*

The IP feasibility problem is NP-complete, essentially due to S. A. Cook [7]. A question under this circumstance is: can this problem be solved in polynomial time when the number of variables $n$ is fixed?

In the cases of $n = 1$ or 2, the answer to the above question was known to be affirmative. For $n = 1$, the solution of the problem is trivial. For $n = 2$, Scarf [24, 25] gave a complete treatment after Hirschberg and Wong [14] and Kannan [15] had given polynomial algorithms for some special $n = 2$ cases.

For general cases, a breakthrough was due to H. W. Lenstra, Jr. [18] who proved that for a fixed number of variables $n$, there is a polynomial algorithm for the IP feasibility problems. The proof consists of two main techniques: the rounding of polytopes and basis reduction on lattices. Lenstra's algorithm introduced the geometry of numbers concept into integer programming and proposed the strategy of branching on hyperplanes instead of on variables as in traditional branch and bound methods.

Lovász and Scarf [21], using the same idea of branching on hyperplanes, developed a generalized basis reduction algorithm to solve the problem without ellipsoidal approximations. W. Cook *et al* [9] implemented the generalized basis reduction algorithm and solved some difficult integer network design instances. Wang [29] solved a set of linear and nonlinear integer programming test problems using the generalized basis reduction algorithm as a subroutine. Recently Aardal *et al* [1] developed an algorithm for solving a system of Diophantine equations with lower and upper bounds on the variables, based on the lattice basis reduction algorithm.

Since some small but hard problems still remain unsolved, it is worthwhile to build an implementation of Lenstra's algorithm and to investigate its computational properties. Since the generalized basis reduction algorithm [21] requires the solution of many linear programming problems at each node, instead of working directly with polytopes, in this paper we investigate the alternative of using ellipsoidal approximation of polytopes. We will discuss issues in implementing a version of Lenstra's algorithm using ellipsoidal approximations. In particular, we use a recently developed practical primal-dual interior point algorithm to solve the ellipsoidal approximation problem [30]. For lattice basis reduction, we use the software LiDIA library [19], while we use CPLEX [8] to solve linear programs involved.

The organization of this paper is as follows. Section 2 describes the framework of Lenstra's algorithm and its improvement. Implementation issues are presented in Section 3. Section 4 gives computational results along with discussions. We conclude the paper in Section 5.

We will use $\mathcal{Q}$ to denote the set of rational numbers, and $\mathcal{Z}$ the set of integers. For any real number $r$, $\lfloor r \rfloor$ is the largest integer that is less than or equal to $r$, and $\lceil r \rceil$ the smallest integer that is greater than or equal to $r$.

## 2 Lenstra's Algorithm and Variants

The intuition of Lenstra's algorithm is the observation that when a polytope is "thin" and extends arbitrarily far in some directions as in Figure 1, the traditional branch-and-bound search tree will grow exceedingly deep in order to find out whether there is a feasible solution or not. The problem can be fixed if a "thin" direction can be identified.

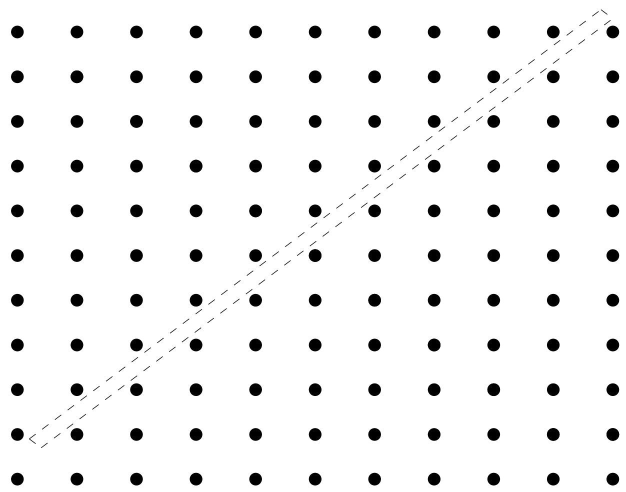

Figure 1: A Thin Polytope in 2-D

Assume that $\mathcal{P} = \{x \in \mathcal{R}^n : Ax \leq b\}$, where $A \in \mathcal{Q}^{m \times n}$ and $b \in \mathcal{Q}^m$, is a full-dimensional polytope. The *width* of $\mathcal{P} \in \mathcal{R}^n$ in a nonzero direction $c$ is defined as

$$F(c) = \max\{c^T x : x \in \mathcal{P}\} - \min\{c^T x : x \in \mathcal{P}\}.$$

Note that the width of a polytope $\mathcal{P} = \{Ax \leq b\}$ in a given direction $c$ can be computed by solving two associated LP problems.

The key idea of Lenstra's algorithm is to find a nonzero integral vector $c$, at each node of a search tree, in polynomial time such that $F(c) \leq c(n)$ where $c(n)$ depends only on $n$ (the number of variables), but not on $m$ (the number of constraints) or anything else. Such a direction $c$ is called a *thin* (or flat) direction of the polytope.

A major contributation of Lenstra's algorithm is that he introduced the concept of lattice from the geometry of numbers into integer programming. Here we briefly describe the concept of lattice and lattice basis reduction, and refer to [5] for detailed information. A lattice in $R^n$ is generated by all integer combinations of a set of $n$ linearly independent vectors, which are called a lattice basis. A lattice has different bases, but some are nicer than others. We say that a basis is *reduced* if its elements are relatively short (without getting into technical details). The notion of a reduced basis is quite old, but no satisfactory algorithm was known for finding a reduced basis until Lenstra, Lenstra, and Lovász [17] proposed a polynomial-time lattice basis reduction algorithm (LLL-algorithm). The procedure of finding a reduced basis is called a basis reduction. The idea of LLL basis reduction is to find a basis, while consisting of relatively short vectors, whose Gram-Schmidt orthogonalization consists of sufficiently long vectors. Using Gram-Schmidt orthogonalization as a "reference", the LLL basis reduction algorithm applies a serial of size reduction and interchanges. The first basis vector in the reduced basis is an approximation of the shortest nonzero lattice vector.

In [18], Lenstra gave a constructive proof that for any fixed dimension $n$, the IP feasibility problem can be solved in time polynomial in the size of input $A$ and $b$. The structure of Lenstra's algorithm is a search tree branching on hyperplanes derived from lattice basis reduction (which will be described below).

Lenstra's algorithm was improved by Babai [3] and Lovász [20] who obtained thin directions bounded by $c(n) = 2(n+1)\sqrt{n}(3/\sqrt{2})^n$ which is better than Lenstra's original bound. The procedure of Lovasz's version [20] of Lenstra's algorithm can be described as the following three steps.

At each node, we first try to determine whether or not a corresponding polytope $\mathcal{P}$ contains an integral vector. We start with finding an ellipsoidal approximation, $\mathcal{E}$, of $\mathcal{P}$ and apply a linear transformation $\tau$ to $\mathcal{E}$ to transform $\mathcal{E}$ into a unit ball. Secondly, we use the basis reduction method to either find an integral feasible vector in $\mathcal{P}$, or find a thin direction $c$. Thirdly, in the former case above, the problem is solves and we stop; and in the latter case, we apply the same procedure, recursively, to each child polytope of the form $\mathcal{P} \bigcap \{x : c^T x = k\}$ for all integers $k$ such that

$$\lceil \min\{c^T x : x \in \mathcal{P}\} \rceil \leq k \leq \lfloor \max\{c^T x : x \in \mathcal{P}\} \rfloor.$$

We note that the dimension of every child polytope is at least one less than the dimension of its parent polytope because it is the intersection of the parent polytope with a hyperplane. Since the number of intersecting hyperplanes is bounded by $c(n)$ in a thin direction, and the search tree is at most $n$-level deep. For fixed $n$, this leads to a polynomial-time algorithm.

In this algorithm, on each node of the search tree, ellipsoid approximation and lattice basis reduction are the two main computational tasks. Another issue, not so obvious but important, is how to guarantee full-dimensionality of the child polytopes which is necessary for ellipsoidal approximation. As it is, a polytope of the form $\mathcal{P} \bigcap \{x : c^T x = k\}$ is not necessarily full-dimensional.

The ellipsoidal approximation problem is a convex optimization problem for which a number of polynomial-time algorithms have been developed (see [2, 16, 23], to list a few). However, until recently this problem had not been extensively studied from a computational point of view. In a recent paper [30], we have developed a reliable and practically efficient algorithm for computing the maximum volume ellipsoid inscribing a given polytope. This algorithm effectively exploits the problem-specific structures and utilizes a primal-dual type, interior point method. Extensive numerical experiments show that the algorithm performs considerably better than a number of other algorithms tested. This algorithm is used in our implementation of Lenstra's algorithm.

The lattice basis reduction algorithm was given by Lenstra, Lenstra and Lovász [17], or the LLL basis reduction algorithm. Several variants of the LLL basis reduction algorithm have been developed and a number of implementation variants have been suggested. The paper by Schnorr and Euchner [26] gives a detailed overview on lattice basis reduction. So far the available implementations of LLL lattice basis reduction algorithms are practical only for small problems. A bottleneck for the LLL-reduction algorithm is that it requires exact arithmetics on large integers. This limitation will be inherited by any implementation, including ours, that make use of the LLL-reduction algorithm as a subroutine on each node of the search tree.

Sevaral public-domain libraries for computational number theory implement the LLL basis reduction algorithm. One is the `LiDIA` library [19], developed at Darmstadt University of Technology (TU Darmstadt), and anther is `NTL` [28], developed by Shoup at University of Wisconsin-Madison. The `LiDIA` library is used to do the lattice basis reduction in our implementation of Lenstra's algorithm.

To our best knowledge, there has been no implementation of Lenstra's algorithm based on ellipsoidal approximation before our implementation. On the other hand, there has been at least one implementation (see [9]) of the generalized basis reduction algorithm proposed by Lovász and Scarf [21]. The generalized basis reduction algorithm shares the same idea of Lenstra's algorithm, i.e., constructing a thin direction of the polytope and branching on all possible hyperplanes intersecting the polytope in that direction. The term "generalized" indicates that this algorithm generalizes the basis reduction algorithm by Lenstra, Lenstra and Lovász [17]. The difference between Lenstra's algorithm and the generalized basis reduction algorithm lies in the different methods to capture a thin direction. Lenstra's algorithm applies ellipsoidal approximations to the polytope and uses the information of the ellipsoids to approximate the shortest lattice vector (i.e. a thin direction) through a lattice basis reduction procedure. On the other hand, the generalized basis reduction algorithm constructs a thin direction directly from the polytope by solving many linear programs. For more details on the generalized basis reduction algorithm and its implementation, we refer to Lovász and Scarf [21] and W. Cook *et al* [9].

# 3 Implementation of Lenstra's Algorithm

Utilizing our algorithm for ellipsoidal approximation developed in [30] and the availability of public-domain lattice basis reduction software, we have implemented a Lovász version [20] of Lenstra's algorithm suitable for solving some small- to medium-sized IP feasibility problems. In this section, we discuss a number of implementation issues.

## 3.1 Non-full-dimensional Polytopes

Lenstra's algorithm operates under the assumption that the polyhedron defined by linear inequalities $Ax \leq b$ is bounded (hence it is a polytope) and *full-dimensional* (i.e. it has a positive volume).

In the branching process of Lenstra's algorithm, all the child polyhedra are bounded if the root polyhedron is bounded. However, it is not difficult to see that a child polytope can be non-full-dimensional even if its parent is. Hence, how to derive a full-dimensional polytope from a non-full-dimensional one is an important issue to be addressed in our implementation.

In Lenstra's paper [18], the assumption of full-dimensionality is justified by constructing vertices of the polytope and employing the Hermite normal form algorithm. We would like to avoid constructing vertices in our implementation since it is not computationally viable and unnecessary for our purpose.

In a more general setting, the problem becomes much harder when the polytope is given by a strong oracle or a strong violation oracle. Grötschel, Lovász and Schrijver [13] and Edmonds, Lovász and Pulleyblank [11] discussed this question of full-dimensionality and proposed solution methods. Since we consider a polytope given by a system of linear inequalities, their methods are not efficient for our purpose.

Our solution to the non-full-dimension problem can be summarized into the following three steps.

The first step is to detect non-full-dimensionality and identify implicit equalities. The necessary and sufficient condition for a polytope $\{x : Ax \leq b\}$ to be non-full-dimensional is that the system of linear inequalities contains at least one implicit equality. To detect implicit equalities, we construct an auxiliary linear program $\max\{t : A\,x + t\,e \leq b, t \geq 0\}$, where $e$ is the $m$-vector of all ones, and solve it using the CPLEX barrier solver. If the maximum value $t^* > 0$, then the corresponding polytope is full-dimensional and the optimal solution $x^*$ is an interior point such that $Ax^* < b$. If $t^* = 0$, then the polytope is non-full-dimensional and we can identify the implicit equalities. In our implementation, we consider the $i$-th inequality in the linear system $Ax \leq b$ to be an implicit equality if $b_i - (Ax^*)_i \leq 1.0e - 12$.

The second step is to reduce the implicit equalities, if any, to its Hermite normal form. Given any full row-rank, rational matrix $B$, there exists an unimodular matrix $U$ such that $BU = [H\ 0]$ where the entries of $H$ satisfy: (1) $h_{ij} = 0$ for all $j > i$, (2) $h_{ii} > 0$ for all $i$, and (3) $h_{ij} < 0$ and $|h_{ij}| < h_{ii}$ for $j < i$. Clearly, $H$ is a square, nonsingular, low triangular matrix. For more details and related results on the Hermite normal form, we refer the reader to [27] and [10]. In our implementation, we use the Hermite normal form reduction function in the `LiDIA` library. One potential drawback of Hermite normal form is that the elements in the unimodular matrix may become excessively large, though we did not seem to have encountered this difficulty on our test problems.

The third step, if necessary, is to project the non-full-dimensional polytope into a lower full-dimensional polytope. Let us split the polytope-defining inequalities $Ax \leq b$ into implicit equalities $Bx = d$ and true inequalities $Cx \leq g$; that is,

$$Ax \leq b \Longleftrightarrow \begin{cases} Bx = d, \\ Cx \leq g, \end{cases}$$

where $B$ has full row-rank (which can be achieved by some row operations if necessary). If we use the change of variable $y = U^{-1}x$, where $U$ is unimodular such that $BU = [H\ 0]$, then the equality $Bx = d$ is equivalent to $Hy_\alpha + 0y_\beta = d$, or $y_\alpha = H^{-1}d$ where $y_\alpha$ is the sub-vector of $y$ with the number of components equal to the number of rows in $B$. Using the same change of variables, we can rewrite the true inequalities $Cx \leq g$ into

$$[CU]\begin{bmatrix} y_\alpha \\ y_\beta \end{bmatrix} \leq g.$$

Upon the substitution of $y_\alpha = H^{-1}d$ into the above, we obtain a full-dimensional polytope in the lower dimensional space of $y_\beta$.

We note that a usual projection would not work here because we have to keep the integrality property so that integer points in the original system corresponds to integer points in the transformed system. In the Hermite normal form approach, because of the unimodularity of the transformation matrix $U$, we can project the non-full-dimensional polytope into a lower full-dimensional polytope without losing integrality.

## 3.2 Choices of the Branching Direction

The key idea of Lenstra's algorithm is to choose a thin direction in which there are only a bounded number of hyperplanes intersecting the polytope at each level of the search tree. Such a thin direction, also called a flat direction, is chosen among the candidates in the dual basis of a reduced basis of the lattice. A direct computation on the number of hyperplanes intersecting the polytope corresponding to all the candidates in the dual basis involves solving $2n$ linear programming problems on each node of dimension $n$, which is too costly. Therefore we want to avoid a direct computation and use an easy-to-compute approximation to serve this purpose. In this section we discuss three different choices of branching directions: Babai's direction, the choice of the smallest number of intersecting hyperplanes intersecting the approximating ellipsoid, and the choice of the direction closest to the smallest eigenvector of the matrix $E$ that defines the ellipsoid approximating the polytope.

### 3.2.1 Babai's direction

We recall that $d(\mathcal{Z}, w^T s)/\|w\|$, where $d(\cdot,\cdot)$ is the Euclidean distance of two sets, is the shorter distance from a given point $s$ to two adjacent hyperplanes $w^T x = \lceil w^T s \rceil$ and $w^T x = \lfloor w^T s \rfloor$, which also provides a lower bound on the the distance from $s$ to the closest lattice point in the lattice $\mathcal{L}$ produced by the LLL lattice basis reduction algorithm [3].

Let the dual basis of the lattice $\mathcal{L}$ be $D = (d_1, d_2, \cdots, d_n)$. In Babai's close lattice point procedure [3], a dual basis direction $d_j$ is chosen such that it has the largest shorter distance from the given point $s$ to the two adjacent hyperplanes ${d_i}^T x = \lceil {d_i}^T s \rceil$ and ${d_i}^T x = \lfloor {d_i}^T s \rfloor$; that is,

$$d_j = \arg\max_i \{ d(\mathcal{Z}, d_i^T s)/\|d_i\| \}.$$

The idea behind this choice is to choose the direction in which the distance between two adjacent lattice hyperplanes from the point $s$ is the largest among the candidates in the dual basis. Since the ellipsoidal approximation of the original polytope is transformed into a unit ball centered at the point $s$, this choice picks an approximation of the direction that has fewer lattice hyperplanes intersecting the transformed unit ball. Figure 2 gives an example on $2-$dimensional case. This choice of branching direction outperforms the other on our test problems.

### 3.2.2 Direction of the smallest number of intersecting hyperplanes

At each node in the search tree, we have at hand the maximum volume ellipsoid approximating the polytope and the reduced basis of the lattice generated from the matrix defining the ellipsoid. Using this information, we propose another branching direction choice: a dual basis direction in which there is a smallest number of intersecting hyperplanes with a circumscribing ellipsoid of the polytope.

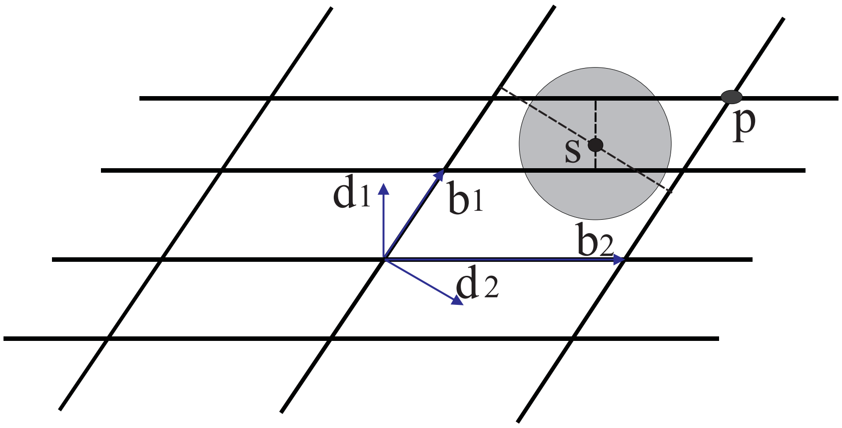


Figure 2: Babai's Choice of Branching Direction in 2-D

At a given search node with a polytope of dimension $n$, the maximum volume inscribing ellipsoid $\mathcal{E}$ is transformed into a unit ball centered at, say $s$. The concentric circumscribing ellipsoid $n\mathcal{E}$ is transformed into a ball of radius $n$ centered at $s$ as well. Let the reduced basis of the lattice generated from the matrix defining $\mathcal{E}$ have a dual basis $D = (d_1, d_2, \cdots, d_n)$.

The number of hyperplanes intersecting the larger ball in a given unit direction $w$ is

$$\lfloor n - d(\mathcal{Z}, w^T s) \rfloor + \lfloor n - 1 + d(\mathcal{Z}, w^T s) \rfloor$$

We propose to choose a dual basis vector $d_j$ such that the above number is the smallest among all dual basis vectors. With this choice, we obtain a direction in which there is the smallest number of hyperplanes intersecting with the circumscribing ball concentric to the unit ball transformed from the maximum volume ellipsoid inscribing the polytope.

#### 3.2.3 Direction closest to the smallest Eigenvector

Ellipsoids have good geometric and computational properties. Recall that an ellipsoid centered at $x$ can be written as

$$\mathcal{E}(x, E) = \{y \in \mathcal{R}^n : y = x + Es \text{ and } \|s\| \leq 1\},$$

for some symmetric positive definite matrix $E$. The width of the ellipsoid is the length of the shortest axis, denoted by $2\lambda$, where $\lambda$ is the smallest eigenvalue of the matrix $E$ The direction of the shortest axis of $\mathcal{E}$ is the eigenvector $v$ corresponding to the smallest eigenvalue $\lambda$.

In our implementation, we compute the maximum volume ellipsoidal approximation of the given polytope at each node. We may use the the width of the ellipsoid to approximate that of the polytope. Since the eigenvector $v$ corresponding to the smallest eigenvalue of $E$ is the thin direction of the approximating ellipsoid, we propose to choose a dual basis vector $d_j$ of the lattice that is closest to $v$ among all the dual basis vectors $(d_1, d_2, \cdots, d_n)$ to be the branching direction; that is, $d_j = \arg\max_i \{v^T d_i / \|v\| \|d_i\|\}$.

### 3.3 Implementation Details

Our implementation of Lenstra's algorithm is for solving the IP feasibility problem and uses the ellipsoidal approximation of polytopes and the lattice basis reduction as described. The input data is $(A, b)$ for defining the linear inequality system $Ax \leq b$, which must be a bounded polyhedron (i.e., a polytope). The output is either that an integer feasible solution is found or that there is no integer feasible solution for the given system. The input data are read by a CPLEX function and must be in a file format readable by CPLEX.

#### 3.3.1 The search tree

The overall picture of the search procedure is a depth-first search tree as is shown in Figure 3. We use a *stack* to store the nodes in the tree. The root of the search tree is the original polytope. This search tree is different from the branch-and-bound search tree in two ways: one is that this search tree grows only as deep as $n$ while the branch-and-bound tree can grow much deeper depending on the input data; the other is that this search tree branches on hyperplanes instead of on variables as in the branch-and-bound search tree. This novel way of searching makes Lenstra's algorithm polynomial for a fixed number of variables $n$ whereas the branch-and-bound method is not a polynomial algorithm even for a fixed number of variables.

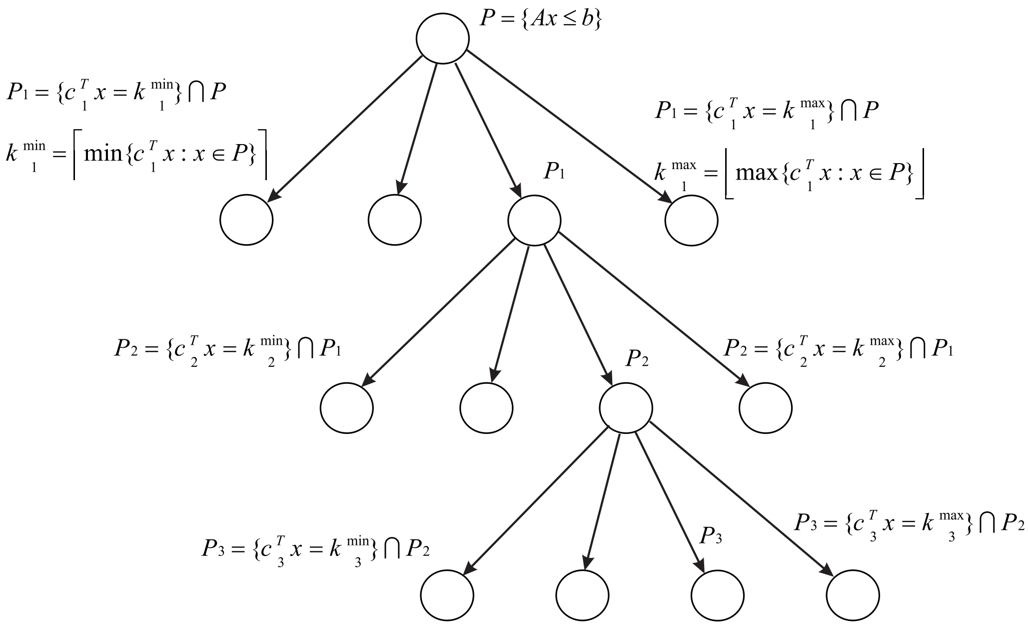


Figure 3: Search Tree Branching on Hyperplanes

#### 3.3.2 On each node of the search tree

In our implementation, we construct a structure called `node` to store the relevant information for each node inn the search tree. The structure `node` consists of components for the matrix

$A$ and the vector $b$, branching bounds $klo$ and $kup$, a transformation matrix $T$ to bring the solution from the transformed space into the original space. The path from the root node to the current node is recorded in a vector variable, and another vector is used to remember the associated swap operations.

On each node in the search tree, the first step is to determine whether the polytope is full-dimensional or not. This is done by solving an auxiliary linear program $\max\{t : Ax+te \leq b\}$ using the CPLEX barrier solver. If the polytope is not full-dimensional, we use the Hermite normal form to project it onto a lower but full-dimensional polytope while maintaining integrality. If the polytope is full-dimensional, the optimal solution to the auxiliary problem gives a strictly feasible starting point for the primal-dual interior point algorithm employed in the CPLEX barrier solver. Once a polytope is found, or made, to be full-dimensional, a preprocessing procedure is applied to delete redundant constraints from the linear inequality system defining the polytope. This preprocessing procedure will reduce the computational costs of solving the relevant linear programs.

The algorithm is a recursive search procedure through the nodes. Either an integer feasible solution is found on some node and the algorithm stops, or the entire tree is searched to ensure that there exists no feasible solution for the problem. A main recursive subroutine does operations on the current node. It starts with classifying the current node into one of the following four cases.

In the first case, if the current node is a leaf node (i.e. 1-dimensional), solve all of its leaf siblings and pop up the top of the stack as the current node. In the second case, if all of the children of the current node have been searched, then check whether the current node has siblings. In the third case, if it has other siblings, move to its immediate sibling and update the current node; otherwise pop up its parent node from the stack as the updated current node.

In the fourth case, if the current node has not been explored, major steps are taken on it. First, do preprocessing to eliminate redundant inequalities in the linear system and solve an auxiliary linear program $\max\{t : Ax + te \leq b, t \geq 0\}$ to determine whether the polytope is full dimensional. If the polytope is not full-dimensional, apply the Hermite normal form procedure to project the polytope onto a lower but full-dimensional polytope, and remember the transformation matrix $U$ as explained in an earlier section. If the polytope $\mathcal{P} = \{x : Ax \leq b\}$ associated with the current node is already full dimensional, then proceed with the following three steps as described in Section 2 (see [20] and also [12] for more details): (1) ellipsoidal approximation and linear transformation, (2) lattice basis reduction and close lattice point procedure, (3) if an integer point is found, then stop, else branch on hyperplanes.

The flowchart of our implementation of Lenstra's algorithm is given in Figure 4 which presents a concise picture of the whole algorithm. We next explain how to generate child nodes.

### 3.3.3 Generating child nodes

We generate a child node from the polytope $\{x : Ax \leq b, c^T x = k\}$ as follows. Notice that the branching direction is $c = \tau^T(d)$ where $d$ is the chosen dual basis vector and $\tau$ is the linear transformation that transforms the maximum volume ellipsoid inscribing the

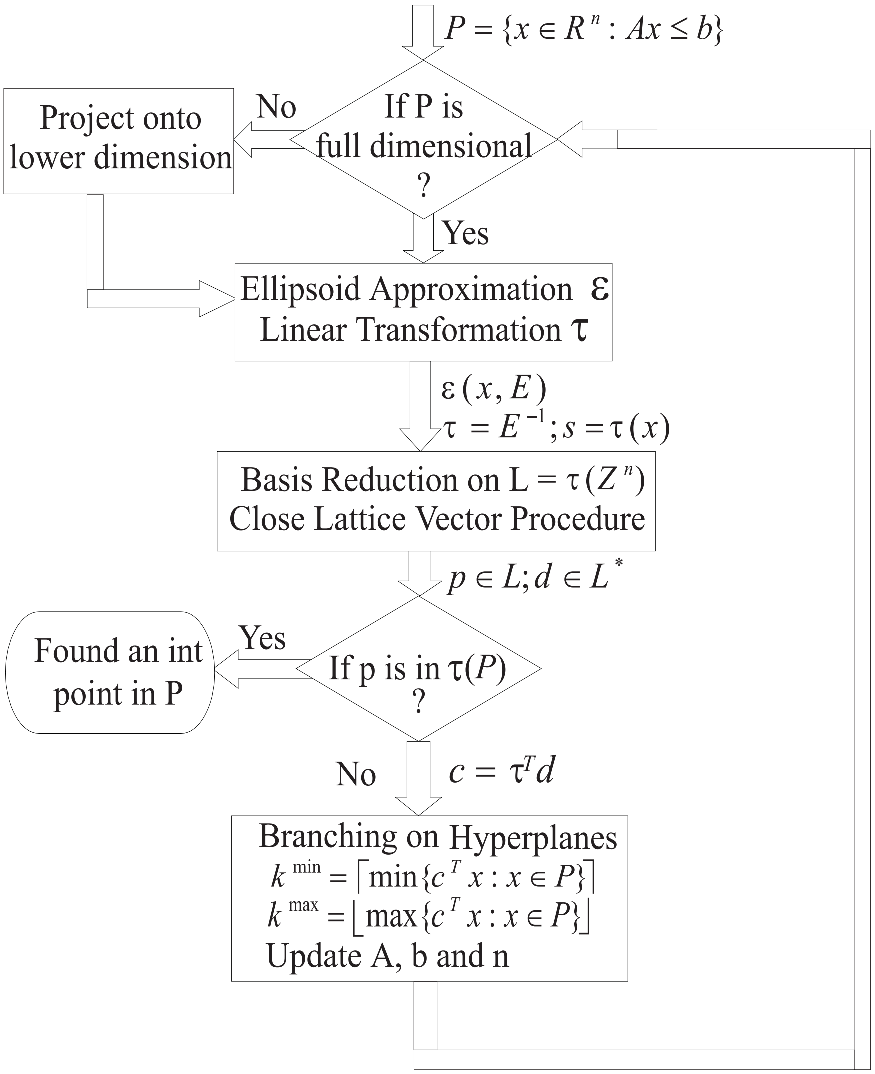


Figure 4: Flowchart of Lenstra's Algorithm

polytope into a unit ball. Hence the hyperplane $c^T x = k$ can be written as $d^T y = k$, where $y = \tau(x)$ is in the transformed space. Making use of the relation between a lattice basis and its dual basis, we have that the component of $y$ corresponding to dual vector $d$ has a coordinate $k$ on the lattice basis $B$ since $d^T y = k$. Thus we can apply the same transformation to $Ax \le b$ and fix the relevant component of $y$ to $k$, which corresponding to a hyperplane in the original space.

For computational convenience, we do a swap operation to exchange the fixed component and the last component and record the information in a vector variable accordingly. After fixing one component of $y$ and eliminating the equality $c^T x = k$, the dimension of the resulting polytope is at least one less than its parent. In the non-full-dimensional case, the dimension of its child node decreases more than 1.

In this algorithm, there is a strong similarity between sibling nodes. From one node to its sibling, only the right hand $k$ in the hyperplane $c^T x = k$ needs to be updated. Therefore we only need to store one child node for each parent node since it is easy to modify the child node to its siblings once it has been searched. Only the path and the transformation matrix need to be remembered. Overall speaking, however, compared with the traditional branch-and-bound method for the IP feasibility problems, Lenstra's algorithm requires more computation on branching.

Even though the floating point computation is applied in the computation of ellipsoidal approximation and lattice basis reduction, the construction of Lenstra's algorithm has the nice property of preserving the integrality.

# 4 Results and Discussion

In this section we present some numerical results for our implementation of Lenstra's algorithm on a set of test problems, along with a short discussion on the results.

Written in C language, our code is developed for the purpose of experimentation, and takes advantage of a number of existing libraries to perform various tasks. The ellipsoidal approximation subroutine makes use of CLAPACK/CBLAS [6] linear algebra library. For lattice basis reduction and Hermite normal form, we use the `LiDIA` [19] library for computational number theory. We use the CPLEX callable library [8] to solve linear programs involved. All the experiments were run on Sun ULTRA60 workstation with two 450-MHz UltraSPARC-II modules. We use only a single processor at a time.

We stress that the Lenstra's algorithm has potential advantages in handling severely skewed polytopes, and is not likely to be efficient for regular problems that can be well solved by the traditional branch-and-bound approach. Since the computational tasks involved in Lenstra's algorithm have relatively high complexity, at present our experimental code is not suitable for solving IP feasibility problems with more than a few hundred variables.

## 4.1 Test Problems

We conducted experiments on a test set of twelve IP feasibility problems constructed from six IP optimization problems. Three of these optimization problems are from the MIPLIB [22] library and the other three from a private collection of test problems for CPLEX [4]. For each optimization problem, we create two feasibility problems, one feasible and another infeasible, using the following procedure. Given an original optimization problem:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \le b, \\ & l \le x \le u, \\ & x \in \mathcal{Z}^n, \end{array}$$

with a known optimal solution $x^*$, we create two corresponding feasibility problems (though still written as optimization problems with a dummy objective function) as follows:

$$\begin{array}{rc} \min & 0^T x \\ \text{s.t.} & c^T x \leq c^T x^* + \delta_i, \\ & Ax \leq b, \\ & l \leq x \leq u, \\ & x \in \mathcal{Z}^n, \end{array}$$

with two chosen scalar $\delta_i$, $i = 1, 2$, such that $\delta_1 < 0$ and $\delta_2 > 0$. Therefore, the former is infeasible while the latter is feasible.

The problem statistics are given in Table 1 for the six optimization problems. The first three problems are pure binary integer programming problems, the fourth and the sixth problem are pure general integer programming problems, and the fifth problem has 64 binary variables and 17 general integer variables. The first six columns of the table are self-explanatory where by LP we refer to the linear programming relaxation of the integer programming (IP) problems. The last column of the table gives the values we used to create the two versions of the feasibility problems, where a slash separates the $c^T x^* + \delta$ value for the infeasible problem with that for the feasible one.

Table 1: Test Problem Statistics

| Problem | Rows of $A$ | Variables | Min LP | Max LP | Min IP | $c^T x^* + \delta$ |
|---|---|---|---|---|---|---|
| stein15 | 36 | 15 | 7.00 | 15.00 | 9 | 8/9 |
| bm23 | 20 | 27 | 20.57 | 97.03 | 34 | 21/40 |
| p0033 | 16 | 33 | 2520.57 | 5299.70 | 3089 | 2996/3090 |
| afix | 44 | 51 | 1295.00 | 470672.00 | 2136 | 2135/2140 |
| dp8rint | 128 | 81 | 8929.00 | 43239.00 | 12360 | 12350/34023 |
| small | 24 | 96 | 189.65 | 217.63 | 191 | 190/195 |

### 4.2 Numerical Results

We tested three variations of our implementation of Lenstra's algorithm using the three different choices of branching direction as described in the last section on the twelve test problems, i.e., Lenstra($i$), $i = 1, 2, 3$, corresponding to Babai's branching direction, the direction based on the number of intersecting hyperplanes with the circumscribing ellipsoid, and the direction closest to the smallest eigenvector of the ellipsoid-defining matrix, respectively. For comparison, we also ran the ILOG CPLEX mixed integer program (MIP) solver (version 6.5.2) on the set of test problems with its default setting.

The computational results are summarized in Tables 2 and 3, where Table 2 gives the time (in CPU seconds) used for the test runs and Table 3 gives the number of nodes examined. In the tables, the first six instances are infeasible and the last six feasible, where the two groups are separated by double lines. In addition, we use "+" to denote the situation in which the program was terminated at a prescribed maximum CPU time or at a maximum number of nodes while the problem was still unsolved. To save experimentation time, we set the maximum allowable time or number of nodes on a problem-to-problem basis, which are shown in the two tables.

CPLEX is widely recognized as a leading commercial software for solving MIP problems. While as most MIP solvers it uses the branch-and-bound method as the basic approach, the CPLEX MIP solver also utilizes a number of cutting plane techniques. We emphasize that our inclusion of CPLEX results is for the purpose of providing a reference point. There is little comparability between the performance of our rather rudimentary implementation of Lenstra's algorithm and that of the CPLEX MIP solver which, as a leading commercial code, has been under continued research and development for many years. The purpose of our experiments is to test the concepts and viability of Lenstra's algorithm, not to compare its practical performance with a well-established and comprehensive commercial solver.

We stress that (i) the computational results for CPLEX were obtained under the CPLEX version 6.5.2 default setting which is not necessarily the optimal for any given individual problem; and (ii) the newer versions of CPLEX perform considerably better on some of the test problems. In particular, we mention that CPLEX version 7.0 can solve the problem *smallm* in a very short amount of time under a well-chosen parameter setting [4].

Table 2: Computational Time

| Problem | Lenstra(1) | Lenstra(2) | Lenstra(3) | CPLEX v.6.5.2 |
|---|---:|---:|---:|---:|
| stein15m | 11 | 31 | 37 | 0.05 |
| bm23m | 25 | 52 | 125 | 0.13 |
| p0033m | 384.41 | 483.35 | 676.09 | 0.06 |
| afixm | 51213 | 41927 | 50000+ | 500.81 |
| dp8rintm | 50000+ | 50000+ | 50000+ | 189793+ |
| smallm | 2246.77 | 10488.54 | 10473.53 | 286867+ |
| stein15m2 | 13 | 14.84 | 29.41 | 0.01 |
| bm23m2 | 49.13 | 84.52 | 351.99 | 0.74 |
| p0033m2 | 112 | 340 | 140 | 0.02 |
| afixmb2 | 7741.57 | 50000+ | 50000+ | 131.09 |
| dp8rintm2 | 40196 | 44473 | 41438 | 0.09 |
| smallm2 | 60000+ | 59205 | 60000+ | 0.12 |

### 4.3 Discussion

Based on our computational results for Lenstra's algorithm, we have the following observations for this algorithm's behavior on the set of tested problems.

**i. Lenstra's algorithm examines fewer nodes.** Our numerical results indicate that the search tree for Lenstra's algorithm may contain much fewer nodes than a branch-and-bound search tree does. One good example that shows potential benefits of hyperplane branching directions is the problem *smallm*. For this problem, the variations of Lenstra's algorithm examined only one to three nodes to detect the infeasibility, while CPLEX v6.5.2,

Table 3: Node Counts

| Problem | Lenstra(1) | Lenstra(2) | Lenstra(3) | CPLEX v.6.5.2 |
|---|---|---|---|---|
| stein15m | 56 | 138 | 181 | 112 |
| bm23m | 4 | 23 | 55 | 58 |
| p0033m | 96 | 234 | 165 | 6 |
| afixmb | 665 | 336 | 700+ | 979,236 |
| dp8rintm | 500+ | 500+ | 500+ | 306,148,430+ |
| smallm | 1 | 3 | 3 | 734,100,288+ |
| stein15m2 | 84 | 97 | 193 | 0 |
| bm23m2 | 35 | 137 | 398 | 49 |
| p0033m2 | 33 | 82 | 32 | 8 |
| afixmb2 | 56 | 100+ | 100+ | 248,296 |
| dp8rintm2 | 73 | 67 | 68 | 43 |
| smallm2 | 178+ | 174 | 200+ | 132 |

under its default setting, took tens of thousand nodes to draw the same conclusion. This example suggests the possibility that Lenstra's algorithm can be extremely effective in some, perhaps unusual, cases where the branch-and-bound approach encounters great difficulties.

**ii. Lattice basis reduction is the bottle neck.** In terms of timing, the current implementation of Lenstra's algorithm is still in general non-competitive with good commercial codes like CPLEX. Currently, the computational burden in Lenstra's algorithm lies mainly in the part of lattice basis reduction, which becomes increasingly costly as the number of inequality constraints increases. On our test problem set, the ratio between the time spent on ellipsoidal approximation and that on lattice basis reduction is approximately 1 to 10 on average. In the extreme instance of *smallm2*, where $m = 121$ and $n = 96$, the ratio between the ellipsoidal approximation time and basis reduction time is 1 to 147, and the lattice basis reduction time took more than 99% of the total computing time. Some sample data are given in Table 4 in the appendix to illustrate this point. This evidence clearly indicates that lattice basis reduction subroutine dominates the computational cost in the current implementation of Lenstra's algorithm. Therefore, a speedup on the lattice basis reduction procedure is crucial to the applicability of Lenstra's algorithm in practice.

**iii. Babai's is the best among the three branching policies.** Branching policies have a significant impact on the number of nodes examined in the search trees. Amongst the 11 test problems listed in table 3, excluding problem *dp8rintm2* because the run was incomplete, Babai's choice examined fewest nodes for 9 problems. The policy based on the lease number of hyperplanes intersecting the circumscribing ellipsoid performed better than the other two in problems *afixmb* and *dp8rintm2*. For this set of problems, the policy based on the closest direction to the smallest eigenvector of the maximum-volume ellipsoid has not exhibited competitiveness with the other two approaches.

**iv. Lenstra's algorithm need help from other techniques.** In advanced codes for integer programming, the traditional branch-and-bound method is often combined with other techniques. To be competitive, Lenstra's algorithm also need to be combined with other techniques, which may include, for example, the warm-start techniques, cutting-plane methods and preprocessing techniques.

# 5 Conclusion

Using our ellipsoidal approximation algorithm as a subroutine, we have implemented a version of Lenstra's algorithm for general integer programming feasibility problems. At each node, the method uses ellipsoidal approximation and lattice basis reduction to find a "thin" direction of the polytope, and branches on hyperplanes rather than on variables as in the traditional branch-and-bound method. In this procedure, it is guaranteed that the number of branches at each node is independent of the number of constraints. Our numerical results on a set of rather small test instances suggest that Lenstra's algorithm examines many fewer nodes than the traditional branch-and-bound method. However, there is a tradeoff between many nodes and fast re-optimization as in the traditional branch-and-bound method and fewer nodes but more time-consuming decisions on branching as in Lenstra's algorithm. Currently, the main bottle-neck in the performance of the algorithm lies at the step of lattice basis reduction.

At present, our implementation of Lenstra's algorithm should not be considered as a competitor to the more mature methods such as branch-and-bound and branch-and-cut for solving commonplace integer programming problems. However, if the performance of lattice basis reduction is significantly improved, then we believe that Lenstra's algorithm can be a useful alternative, especially when combined with other available techniques, for solving some small but hard general integer programming problems for which the traditional branch-and-bound algorithms have difficulties.

# Acknowledgment

The author would like to thank Bill Cook for his insight in integer programming and many valuable suggestions that motivated and guided this research effort.

# References

[1] K. Aardal, C. Hurkens, and A. K. Lenstra. Solving a linear diophantine equation with lower and upper bounds on the variables. In R.E. Bixby, E.A. Boyd, and R.Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Lecture Notes in Computer Science 1412*, pages 229–242, Berlin, Heidelberg, 1998. Springer-Verlag.

[2] K. M. Anstreicher. Improved Complexity for Maximum Volume Inscribed Ellipsoids. Manuscript. June 2001.

[3] L. Babai. On Lovász' lattice reduction and nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[4] R. E. Bixby. Private communication. Rice University, 2001.

[5] J. W. S. Cassels. An Introduction to the Geometry of Numbers. Springer-Verlag, Berlin, 1959.

[6] *CBLAS*. http://www.netlib.org/clapack/cblas, lapack 2.0 edition.

[7] S.A. Cook. The complexity of theorem-proving procedures. *Proceedings of Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[8] CPLEX Optimization. *Using the CPLEX Callable Library*. Incline Village, NV, http://www.cplex.com, 6.0.1 edition, 1998.

[9] W. Cook, T. Rutherford, H. E. Scarf, and D. Schallcross. An implementation of the generalized basis reduction algorithm for integer programming. *Operations Research Society of America Journal on Computing*, 5:206–212, 1993.

[10] P. Domich, R. Kannan, and L. Trotter, Jr. Hermite Normal Form Computation Using Modulo Determinant Arithmetic. Math. Operations Research, 12:50-59, 1987.

[11] J. Edmonds, L. Lovász, and W. R. Pulleyblank. Brick decompositions and the matching rank of graphs. *Combinatorica*, 2:247–274, 1982.

[12] L. Gao. *Ellipsoidal Approximation to Polytopes and Computational St udy of Lenstra's Algorithm*. PhD thesis, Rice University, 2001.

[13] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.

[14] D. S. Hirschberg and C. K Wong. A polynomial-time algorithm for the knapsack problem with two variables. *Journal of the Association for Computing Machinery*, 23:147–154, 1976.

[15] R. Kannan. A polynomial algorithm for the two-variable integer programming problems. *Journal of the Association of Computing Machinery*, 27:118–122, 1980.

[16] L. Khachiyan and M. Todd. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, 61:137–159, 1993.

[17] A. K. Lenstra, H. W. Lenstra, Jr, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[18] H. W. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8:538–548, 1983.

[19] LiDIA. *LiDIA – A Library for Computational Number Theory*. TH Darmstadt Universität des Saarlandes, Fachbereich Informatik, Institute für Theoretische Informatik, http://www.informatik.th-darmstadt.de/pub/TI/LiDIA, 1.3.4 edition, 1999.

[20] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*. The Society for Industrial and Applied Mathematics, Philadelphia, 1986.

[21] L. Lovász and H.E. Scarf. The generalized basis reduction algorithm. *Mathematics of Operations Research*, 17:751–764, 1992.

[22] *MIPLIB*. http://www.caam.rice.edu/~bixby/miplib/miplib.html, 3.0 edition, 1996.

[23] Y. E. Nesterov and A. S. Nemirovskii. *Interior Point Methods in Convex Programming –Theory and Applications*. The Society for Industrial and Applied Mathematics, Philadelphia, 1994.

[24] H.E. Scarf. Production sets with indivisibilities– part I: Generalities. *Econometrica*, 49:1–32, 1981.

[25] H.E. Scarf. Production sets with indivisibilities– part II: the case of two activities. *Econometrica*, 49:395–423, 1981.

[26] C. P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical Programming*, 66:181–199, 1994.

[27] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons Ltd, Chichester, 1986.

[28] V. Shoup. *NTL: A Library for Doing Number Theory*. Department of Computer Science, University of Wisconsin-Madison, http://www.shoup.net.

[29] X. Wang. *A New Implementation of the Generalized Basis Reduction Algorithm for Convex Integer Programming*. PhD thesis, Yale University, 1997.

[30] Y. Zhang and L. Gao. On numerical solution of the maximum volume ellipsoid problem. To appear in *SIAM Journal on Optimization*.

## Appendix

In Table 4, we give some statistics, gathered on 40 nodes from two search trees indexed as 1 to 40, that compare the computing time in seconds spent on the ellipsoidal approximation and on the lattice basis reduction, respectively, for each node. The last column gives the ratio of the latter vs. and former.

Table 4: Time for Ellipsoidal Approximation and Lattice Basis Reduction

| Index | Constraints | Variables | Ellipsoidal | Basis reduction | Ratio |
|---|---|---|---|---|---|
| 1 | 282 | 77 | 56.27 | 2084.40 | 37 |
| 2 | 280 | 76 | 54.76 | 1890.06 | 35 |
| 3 | 278 | 75 | 53.61 | 1710.66 | 32 |
| 4 | 276 | 74 | 52.49 | 1623.96 | 31 |
| 5 | 274 | 73 | 47.29 | 1507.25 | 32 |
| 6 | 272 | 72 | 46.43 | 1377.78 | 30 |
| 7 | 270 | 71 | 48.97 | 1313.96 | 27 |
| 8 | 268 | 70 | 43.82 | 1165.54 | 27 |
| 9 | 266 | 69 | 46.35 | 1110.90 | 24 |
| 10 | 264 | 68 | 49.97 | 1031.70 | 21 |
| 11 | 262 | 67 | 45.54 | 943.33 | 21 |
| 12 | 260 | 66 | 43.77 | 884.03 | 20 |
| 13 | 258 | 65 | 42.46 | 742.97 | 17 |
| 14 | 256 | 64 | 41.86 | 660.12 | 16 |
| 15 | 254 | 63 | 43.95 | 622.03 | 14 |
| 16 | 252 | 62 | 69.78 | 569.17 | 8 |
| 17 | 250 | 61 | 71.25 | 522.26 | 7 |
| 18 | 248 | 60 | 72.50 | 470.91 | 6 |
| 19 | 250 | 61 | 71.12 | 535.74 | 8 |
| 20 | 248 | 60 | 67.31 | 451.50 | 7 |
| 21 | 147 | 51 | 10.90 | 198.66 | 18 |
| 22 | 147 | 50 | 10.02 | 192.50 | 19 |
| 23 | 147 | 49 | 9.79 | 177.02 | 18 |
| 24 | 147 | 50 | 11.37 | 186.18 | 16 |
| 25 | 147 | 49 | 9.80 | 174.89 | 18 |
| 26 | 147 | 49 | 11.16 | 167.58 | 15 |
| 27 | 147 | 48 | 9.60 | 148.03 | 15 |
| 28 | 147 | 50 | 11.42 | 182.11 | 16 |
| 29 | 147 | 49 | 11.16 | 196.64 | 18 |
| 30 | 147 | 49 | 9.78 | 169.53 | 17 |
| 31 | 147 | 48 | 9.65 | 164.16 | 17 |
| 32 | 147 | 49 | 11.16 | 165.99 | 15 |
| 33 | 147 | 48 | 9.64 | 154.77 | 16 |
| 34 | 147 | 48 | 10.99 | 143.78 | 13 |
| 35 | 147 | 47 | 11.48 | 156.83 | 14 |
| 36 | 147 | 47 | 9.52 | 138.16 | 15 |
| 37 | 147 | 46 | 9.30 | 128.55 | 14 |
| 38 | 147 | 50 | 11.37 | 181.25 | 16 |
| 39 | 147 | 49 | 11.13 | 196.29 | 18 |
| 40 | 147 | 49 | 9.81 | 172.30 | 18 |