Minkowski's Convex Body Theorem and Integer Programming

Ravi Kannan

Abstract The paper presents an algorithm for solving Integer Programming problems whose running time depends on the number n of variables in the problem as $n^{O(n)}$. This is done by reducing an n variable problem to $n^{\frac{5}{2}i}$ problems in $n-i$ variables for some i greater than 1. The factor of $n^{5/2}$ "per variable" improves on the best previously known factor which is exponential in n. Minkowski's Convex Body theorem and other results from Geometry of Numbers play a crucial role in the algorithm; they are explained from first principles.

Supported by NSF grant ECS-8418392

Introduction

The Integer Programming (feasibility) Problem is the problem of determining whether there is a vector of integers satisfying a given system of linear inequalities. In settling an important open problem, H.W.Lenstra (1981,1983) showed in an elegant way that when n the number of variables is fixed, there is a polynomial time algorithm to solve this problem. He accomplishes this by giving a polynomial time algorithm that for any polytope P in $\mathcal{R}^n$ either finds an integer point (point with all integer coordinates) in P or finds an integer vector v so that the maximum value of (v,x) and the minimum value of (v,x) over the polytope P differ by less than c^{n^2} where c is a constant independent of n. Every integer point must lie on a hyperplane of the form $(v,x) = z$ for some integer z, and there are at most c^{n^2} such hyperplanes intersecting P. It obviously suffices to determine for each such hyperplane H, whether $H \cap P$ contains an integer point. Lenstra uses this to show that an n variable problem can be reduced to c^{n^2} problems each in $n-1$ variables. This raises two questions : Can we effectively reduce an n variable problem to polynomially many $n-1$ variable problems ? Can the reduction be done efficiently so as to achieve a better complexity for Integer Programming ? Both these questions are answered affirmatively in this paper.

If an n variable problem is reduced to polynomially many $n-1$ variable problems, the best complexity we can achieve is n^{cn} for some constant c, so we are at liberty to take this amount of time for the reduction to one less variable. Furthermore, the same result is obviously achieved if we reduce an n variable problem to n^{ci} problems in $n-i$ variables for *some* i between 1 and n. Indeed, the greater the i the better since then we reduce the number of variables by a larger amount. This paper presents an algorithm which either finds an integer point in the given polytope P in $\mathcal{R}^n$ or finds for some i, $1 \leq i \leq n$, an $n-i$ dimensional subspace V with the following property : the number of translates of V containing integer points that intersect P is at most $n^{\frac{5}{2}i}$. Each such translate leads to a $n-i$ dimensional problem. So, it can be shown that there is a factor of $O(n^{5/2})$ per variable in the running time. In this sense, it reduces an n variable problem effectively to $O(n^{5/2})$ problems in $n-1$ variables. The algorithm for finding the subspace V uses at most $O(n^n s)$ arithmetic operations where s is the length of description of the polytope. The dependence on n of the complete integer programming algorithm is shown to be $O(n^{cn})$.

This paper is the final journal version of the preliminary paper Kannan (1983). Since the appearance of the preliminary version, Hastad (1985) has observed using results of Lenstra and Schnorr (1984) that for any polytope P of positive volume in $\mathcal{R}^n$, if P does not contain an integer point, then, there *exists* an integer vector v such that the maximum and minimum of (v,x) over P differ by at most $O(n^{5/2})$. This is an interesting existence result. But, there is no finite algorithm known that with P as input either gives us an integer point in P or the vector v. If we relax the $O(n^{5/2})$ to $O(n^3)$, then we can get such an algorithm using the techniques of this paper ; it uses $O(n^n s)$ arithmetic operations. This gives a way of reducing an n variable Integer Program to $O(n^3)$ problems in $n-1$ variables.

However, the resulting algorithm for Integer Programming has, obviously, asymptotically worse complexity, so it is not presented here.

This paper uses several concepts and results from Geometry of Numbers, the most crucial of them being Minkowski's convex body theorem. This elegant classical theorem turns out to be crucial in effectively reducing an n variable problem to polynomially many $n-1$ variable problems rather than an exponential number of them. Section 1 contains a brief introduction to Geometry of Numbers to make the paper self-contained for Operations Researchers and Computer Scientists.

The integer programming algorithm will be presented after two other algorithms : one for finding the shortest (in Euclidean length) non-zero integer linear combination of a given set of vectors and the other for finding the integer linear combination of a set of vectors that is closest (in Euclidean distance) to another given vector. These are called the Shortest Vector Problem (SVP) and the Closest Vector Problem (CVP) respectively. The algorithms for both problems take $O(n^n s)$ arithmetic operations on n dimensional problems where s is the length of the input. The algorithm for the SVP is needed as a subroutine in the integer programming algorithm whereas the algorithm for the CVP is not directly needed, but has ideas that will be useful in integer programming.

It is well-known that Integer Programming is NP-hard. It has been shown recently that CVP is NP-hard. At present, it is not known whether SVP is NP-hard or admits a polynomial time algorithm (or both !). The last section of the paper provides another, more natural proof that CVP is NP-hard. Further, it relates the complexity of the SVP to an approximate version of the CVP. It is hoped that this is a beginning towards proving the NP-hardness of the SVP which remains an important open problem.

Summary of the paper

Operations Researchers are usually interested in solving the Integer Programming Optimality problem - i.e., the problem of maximizing a linear function over the set of integer solutions (solutions with all integer coordinates) to a system of linear inequalities. This question can be reduced by elementary means to the Integer Programming feasibility question which is the problem of determining whether there is an integer point inside a given polyhedron. This paper deals only with the feasibility question and this will be called the **Integer Programming Problem.** Computationally it can be stated as : Given $m \times n$ and $m \times 1$ matrices A and b respectively of integers, find whether there exists an $n \times 1$ vector x of integers satisfying the m inequalities $Ax \leq b$. The case of $n = 1$ can be trivially solved in polynomial time. For the case of $n = 2$, Hirschberg and Wong (1976), Kannan (1980) and Scarf (1981) gave polynomial time algorithms.

Central to H.W.Lenstra's algorithm for general n is an algorithm for finding a "reduced basis " of a "lattice"(both terms to be explained later). Lenstra's (1981) original basis reduction algorithm takes polynomial-time only when the number of dimensions is fixed. After his result, Lovász devised a basis reduction algorithm which runs in polynomial time even when n the number of dimensions varies . This algorithm combined with an earlier result of A.K.Lenstra's (1981) that reduced factoring a polynomial to finding a short vector

in a lattice yields a polynomial time algorithm for factoring polynomials over the rationals. All these ideas were first published in an important paper of Lenstra, Lenstra and Lovász (1983). This paper is referred to henceforth as the LLL paper. Here, the following result from the LLL paper is used : Given a set of vectors $b_1, b_2, \ldots b_n$, we can find in polynomial time a nonzero integer linear combination of them whose length is at most $2^{n/2}$ times the length of any (other) nonzero integer linear combination. In addition, we will need a technical result from H.W.Lenstra's paper which is due to Lovász. This result is stated in the section on integer programming.

Section 1 introduces lattices and proves Minkowski's theorem. Section 2 presents an algorithm for finding a "more reduced basis" [1] of a lattice than the LLL algorithm. While the end product of this algorithm is better because it is "more reduced", it also takes more time ($O(n^n s)$ arithmetic operations) than the LLL algorithm. The first vector of the "more reduced basis" will be a shortest nonzero vector in the lattice. This solves the SVP mentioned in the abstract. Section 2 closes with a proof of correctness and a bound on the number of arithmetic operations. Section 3, the most technical section of the paper, proves bounds on the size of numbers produced by the algorithm in section 2.

The second major algorithm in the paper is for solving the CVP and is given in section 4. It uses as a subroutine the algorithm for finding the "more reduced basis". After these, the algorithm for Integer Programming is given. It performs $O(n^{\frac{9}{2}n}s)$ arithmetic operations for an n variable problem and produces numbers with $O(n^{2n}s)$ bits where s is the length of the input. This is section 5. In a recent paper, Frank and Tardos (1985) show that all the numbers can be kept polynomially bounded in their number of bits. Their improvement also brings down the number of arithmetic operations of the algorithm to $O(n^{\frac{5}{2}n}s)$.

Here is a brief overview of the algorithms : The algorithm for the SVP first solves it approximately, then enumerates a bounded number of candidates for the shortest nonzero vector and chooses the best. Minkowski's theorem implies that this set of candidates suffices. In the algorithm for the CVP and integer programming, the original problem is transformed so that by appealing to the Minkowski's theorem, the transformed problem can be reduced to a bounded number of lower dimensional problems.

The last section of the paper contains some results on complexity. The Closest Vector Problem is shown to be NP-hard by reducing 3-dimensional matching to it. Then the Yes/No question that corresponds to the Shortest Vector Problem in a natural way is defined - it is namely the question of whether there is a nonzero integer linear combination of a set of given vectors of length less than or equal to a given number. The SVP is shown to be polynomial-time reducible to the Yes/No question. Then using a technique called "homogenization" from polyhedral theory, it is shown that the problem of solving the CVP to within a factor of $\sqrt{n/2}$ is polynomial-time reducible to the Yes/No question. I conjecture that this approximate version of the CVP is NP-hard. If the conjecture is proved , it would be the case that the Yes/No question is NP-complete in the sense of Cook (1971) and the reduction essentially is a Cook (Turing) reduction rather than a many-one

[1] (2.6) gives a definition of the "more reduced basis". It is a very natural concept.

reduction. At present, every language that is known to be NP-complete in the sense of Cook, is also NP-complete in the sense of Karp (1972), i.e., in all the known cases the reductions are many-one. Thus, the proof of NP-hardness of the approximate version of the CVP is an interesting open problem.

After the preliminary version of this paper appeared, Helfrich (1985) has made some improvements in the running time of some of the algorithms. I refer the reader to her paper for the improvements. Schnorr (1984) uses the algorithm presented here for solving the SVP to obtain *polynomial time* algorithms for finding better approximations to the shortest vector than the LLL paper. Lenstra and Schnorr (1984) prove very nice properties of the successive minima of lattices from the concept of "more reduced basis" used in this paper. They have traced this concept back to Korkhine and Zolotoreff (1873). Babai (1985) is an interesting related development to some of the algorithmic questions discussed in this paper.

Notation

$\mathcal{R}^n$- Euclidean $n-$space

$\mathcal{Z}^n$- the set of $n-$vectors with integer components

(a,b) is the dot product of the two vectors a,b

$|a| = |a|_2 =$ the Euclidean length of the vector a

$L(b_1,b_2,\ldots b_n) =$ the lattice generated by the vectors $b_1,b_2,\ldots b_n$ (i.e., the set of all integer linear combinations of these vectors).

For any set of vectors $b_1,b_2,\ldots b_n$, we reserve the notation $b_i(j)$ for the real numbers defined in (1.7) and $b(i,j)$ for the vectors defined in (1.7)'.

For any lattice L, (see definition in the next section) $\lambda_1(L)$ will denote the length of a shortest nonzero vector in the lattice.

Suppose $L(b_1,b_2,\ldots b_n)$ is a lattice. Then for $j = 1,2,\ldots n$, $L_j(b_1,b_2,\ldots b_n)$ will denote the projection of $L(b_1,b_2,\ldots b_n)$ orthogonal to the vector space spanned by $b_1,b_2,\ldots b_{j-1}$. By convention we take the space spanned by the empty set to be the singleton $\{0\}$ and hence the orthogonal complement of it is the whole space. Thus, $L_1(b_1,b_2,\ldots b_n) = L(b_1,b_2,\ldots b_n)$. Clearly $L_j(b_1,b_2,\ldots b_n)$ depends on the basis $b_1,b_2,\ldots b_n$ of the lattice we choose.

The programs in this paper will be written in "pidgin" ALGOL. The language is close enough to English that the reader should have no problem with it. I adopt the convention that the statement "Return x" means Stop execution and output x.

1 Basic definitions and facts about lattices

A *lattice* L in $\mathcal{R}^n$ is the set of all integer linear combinations of a set of linearly independent vectors in $\mathcal{R}^n$. The independent vectors are called a *basis* of the lattice.

If $b_1, b_2, \ldots, b_n$ are independent vectors in $\mathcal{R}^m, m \geq n$, the basis matrix of the lattice $L(b_1, b_2, \ldots, b_n)$ is the $n \times m$ matrix B with $b_1, b_2, \ldots, b_n$ as its n rows. Now suppose U is any $n \times n$ unimodular matrix (integer matrix with determinant ± 1). Clearly, the inverse of U exists and has integer entries. Then for any y in $\mathcal{R}^m$, y is in $L(b_1, b_2, \ldots, b_n)$ iff $\exists x \in Z^n$: $y = xB \iff \exists x' \in Z^n$: $x'(UB) = y$ (because U, U^{-1} have integer entries $) \iff y \in$ the lattice generated by the rows of UB.

Thus making a unimodular transformation of the basis leaves the lattice unchanged. Indeed the converse is also true.

Lemma (1.1)

Suppose B and B' are $n \times m$ and $k \times m$ matrices each with independent rows and suppose the rows of B and B' generate the same lattice. Then k equals n and there is a unimodular matrix U such that $UB = B'$.

Proof : Since the lattices generated by the rows of B and B' are the same, so are the subspaces. Hence $k = n =$ the dimension of the subspace. The rows of B' are integer combinations of the rows B and vice versa. Thus there are $n \times n$ matrices of integers U and U' such that $UB = B'$ and $U'B' = B$. So, $UU'B' = B'$, since B' has independent rows $UU' = I$. U and U' have integer determinants and are inverses of each other, so they must both have determinant ± 1.

□

The dimension of a lattice is the number of basis vectors that generate it. If a lattice is full dimensional, i.e., it is a lattice in $\mathcal{R}^n$ of dimension n and is generated by the rows of an $n \times n$ matrix B, the determinant of the lattice is defined to be the absolute value of the determinant of B (by the lemma above it is an invariant of the lattice) Geometrically, it is the volume of the parallelpiped spanned by $b_1, b_2, b_3, \ldots, b_n$. We also have to deal with lattices which are not full dimensional. Thus suppose $b_1, b_2, \ldots, b_n$ are independent vectors in $\mathcal{R}^m, m \geq n$ Then the determinant of $L(b_1, b_2, \ldots, b_n)$ is defined to be the n volume of the n-dimensional parallelpiped spanned by $b_1, b_2, \ldots, b_n$. To make this definition computationally more explicit as well for other purposes, we define unit vectors $u_1, u_2, \ldots, u_n$ which are mutually orthogonal as follows:

$$b_1^* = b_1; u_1 = b_1^*/|b_1^*| \tag{1.2}$$

$$b_{i+1}^* = b_{i+1} - \sum_{j=1}^{i} (b_{i+1}, u_j) u_j \text{ for } i = 1, 2, \ldots, n-1 \tag{1.3a}$$

$$u_{i+1} = b_{i+1}^*/|b_{i+1}^*| \tag{1.3b}$$

Thus b^*_{i+1} is the projection of b_{i+1} perpendicular to the space spanned $b_1, b_2, \ldots, b_i$. The procedure of finding the b^*_i is the familiar Gram-Schmidt orthogonalization process (for example see Strang(1980)). Rewriting (1.2) and (1.3) as

$$b^*_1 = b_1 \tag{1.4}$$

and

$$b^*_{i+1} = b_{i+1} - \sum_{j=1}^{i} \mu_{i+1,j} b^*_j \text{ for } i = 2, \ldots n-1 \tag{1.5}$$

where

$$\mu_{kl} = (b_k, b^*_l)/(b^*_l, b^*_l) \quad \text{for } 1 \leq l < k \leq n \tag{1.6}$$

we see that if $b_1, \ldots, b_n$ have rational coordinates, so do the b^*_i and they can be computed in polynomial time from $b_1, b_2, \ldots, b_n$. This is not obvious from (1.2) and (1.3) since $|b^*_j|$ may be irrational. The Gram-Schmidt procedure described by (1.4) and (1.5) is used repeatedly in this paper.

Clearly, there exist real numbers $b_i(j)$ such that

$$b_i = \sum_{j=1}^{n} b_i(j) u_j \tag{1.7}$$

In fact, from (1.5), we see that $b_i(j) = \mu_{ij}|b^*_j|$ for $1 \leq j < i \leq n$ and that $b_i(i) = |b^*_i|$ for all i. So with rational inputs, $b_i(j)^2$ is always rational (even though $b_i(j)$ may not be). This observation will be useful because I will have occasion to compare $|b_i(j)|$ with other real numbers. The definition of $b_i(j)$ in (1.7) will be used repeatedly and so it is part of the notation. I will also use occasionally the vectors $b(i,j)$ defined by (1.7)' below :

$$b(i,j) = \sum_{k=j}^{i} b_i(k) u_k \text{ for } 1 \leq j \leq i \leq n \tag{1.7}'$$

In many parts of the paper, it will be extremely useful to think of $b_1, b_2, \ldots, b_n$ as being represented in a coordinate system with $u_1, u_2, \ldots, u_n$ given by (1.2) and (1.3) as the axes vectors. In this coordinate system, the matrix with the basis vectors as its rows is lower triangular and has the $b_i(j)$ of (1.6) as its entries. The reader is reminded that $b_i(j)$ is the length of the projection of b_i onto the orthogonal complement of the space spanned by the vectors $b_1, b_2, \ldots, b_{j-1}$ for $1 \leq j \leq i \leq n$. Thus for example $b_i(i)$ is the length of b^*_i.

$$\begin{pmatrix} b_1(1) & 0 & 0 & . & . & . & . & . & 0 \\ b_2(1) & b_2(2) & 0 & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & b_i(i) & 0 & . & . \\ . & . & . & . & . & b_{i+1}(i) & b_{i+1}(i+1) & . & . \\ . & . & . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & 0 \\ . & . & . & . & . & . & . & . & 0 \\ b_n(1) & b_n(2) & . & . & . & . & . & . & b_n(n) \end{pmatrix}$$

The lower triangular representation of the basis matrix

I caution that these entries may be irrational and cannot be exactly computed in general. So, in the algorithms I do not change the coordinate system, but conceptually it is easier to think of the basis matrix being written in this form.

The determinant of $L(b_1, b_2, \ldots, b_n)$ denoted $d(L(b_1, b_2 \ldots, b_n))$ is defined to be the absolute value of the determinant of the lower triangular $n \times n$ matrix whose entries are $b_i(j)$. Clearly, this equals the product of the lengths of the b_i^*, $i = 1, 2, \ldots, n$. Thus while the determinant may not be rational even if the coordinates of $b_1, b_2, \ldots, b_n$ are, the square of the determinant is and it can be computed in polynomial-time.

We are often interested in "projecting" and "lifting" vectors. Projecting a vector b onto the hyperplane through the origin with v as the normal yields the vector $b - ((b,v)/(v,v))\, v$. The projection of b in the direction of v is the vector $((b,v)/(v,v))v$. To project perpendicular to a subspace we find an orthogonal basis of the subspace and project perpendicular to each basis vector successively - this is the Gram-Schmidt procedure described in (1.2) and (1.3). To project onto a subspace, means to project perpendicular to its orthogonal complement. The projection of a set is the set of projections of its elements. Suppose v is a nonzero element of a lattice L and $\hat{L}$ is the projection of L perpendicular to v. If $\hat{w}$ is any vector in the $\hat{L}$, we may "lift" it to a vector in L as follows : it is easy to see that there is a unique vector w in L such that w projects onto $\hat{w}$ and $(w,v) \in (-(v,v)/2, (v,v)/2]$. To see this note that we may take any vector u which projects onto $\hat{w}$ and add a suitable integer multiple of v into u to get a u' whose projection in the direction of v is at most $|v|/2$ which is exactly what the dot product condition above stipulates. Indeed, let $r = [(u,v)/(v,v)]$ where $[x]$ stands for the integer nearest to the real number x. Let $u' = u - rv$. Then $|(u',v)| \leq (1/2)(v,v)$. The process described here will be called *lifting* $\hat{w}$ to w.

A set S in $\mathcal{R}^n$ is said to be *discrete* if there is a positive real number δ so that $|v-u| \geq \delta \forall u, v \in S, u \neq v$. S is a Z-*module* if it forms a group under vector addition.

Proposition 1.8 : A set S in $\mathcal{R}^n$ is a lattice if and only if it is a discrete Z-module.

Proof : Suppose S is a lattice with basis $b_1, b_2, \ldots b_m$. Then clearly it is a Z-module. If it is not a discrete set, then it contains points arbitrarily close to and not equal to the origin (since it is a Z-module). This is impossible since the lattice does not have

any points inside the parallelpiped $\{x : x = \sum_{j=1}^{m} \lambda_j b_j; |\lambda_j| \leq \frac{1}{2}\}$ other than the origin because $b_1, b_2, \ldots b_m$ are independent. The converse will be proved by induction on the dimension of the vector space span of the set S. If this dimension is 1, S must be the set of integer multiples of a single vector. Suppose S is a discrete Z-module of dimension m greater than 1. Suppose $\delta > 0$ is the greatest lower bound on $|u - v|$ for $u, v \in S$. There must be a $v \in S$ such that $|v| = \delta$. (Otherwise, there is an infinite sequence of distinct elements $u_1, u_2, \ldots$ in S so that $|u_i|$ converges from above to δ, there is then a convergent subsequence of it. The distance between elements of the subsequence goes to zero, violating the fact that δ is positive.) I will construct a basis of S with v as the first basis vector. Towards this end, define $\hat{S}$ as the projection of S perpendicular to v. It is obvious that $\hat{S}$ is a Z-module. I claim that it is a discrete set : If not, there is a sequence of elements $u_1, u_2, \ldots$ in $\hat{S}$ such that they converge to the origin. By the paragraph preceding the proposition, they can be lifted to $v_1, v_2, \ldots$ in S so that the projections of the v_i in the direction of v is at most $|v|/2$. Hence, the v_i form a bounded sequence and so, there is a subsequence of the v_i that converges. This violates the discreteness of S. So, $\hat{S}$ is a discrete set and by induction on the dimension, $\hat{S}$ is a lattice. Suppose $\hat{b}_2, \hat{b}_3 \ldots \hat{b}_m$ is a basis of $\hat{S}$. Let $b_2, b_3, \ldots b_m$ be vectors in S whose projections are $\hat{b}_2, \hat{b}_3, \ldots \hat{b}_m$ respectively. Then I claim that $S = L(v, b_2, b_3, \ldots b_m)$. Clearly, $S \supseteq L(v, b_2, b_3, \ldots b_m)$. Suppose w is in S and $\hat{w} = w - ((w,v)/(v,v))v$. $\hat{w}$ is in $\hat{S}$ and hence equals $\sum_{i=2}^{m} z_i \hat{b}_i, z_i \in Z$. Let $w' = w - \sum_{i=2}^{m} z_i b_i$. Then $w' = \lambda v$ for some $\lambda \in \mathcal{R}$. Since λv and $\lfloor(\lambda)\rfloor v$ are both in S their difference is and if λ is not an integer, this would yield a shorter nonzero vector than v contradicting its definition. So λ must be an integer and by the definition of w' , we see that $w \in L(v, b_2, b_3, \ldots b_m)$. This completes the proof of the proposition.

□

Proposition 1.9 : Suppose v is a nonzero element of the lattice L, such that λv does not belong to the lattice for any λ in $(0,1)$. Then there is a basis of the lattice containing v. (Such a vector v is called *primitive*).

Proof Let $\hat{L}$ be the projection of L perpendicular to v. Then $\hat{L}$ is a lattice (from the proof of the last proposition) and has a basis $\hat{b}_2, \hat{b}_3, \ldots \hat{b}_m$. If we lift these vectors to $b_2, b_3, \ldots b_m$ they together with v form a basis of the lattice L - the proof is identical to the last part of the proof of the last proposition.

□

Proposition 1.10 : The following "algorithm" yields a basis $b_1, b_2, \ldots b_n$ of the lattice L.

Procedure Input lattice L of dimension n.

$b_0 = 0$

do for $i = 1$ to n by 1 :

Pick any nonzero v such that $(v, b_j) = 0$ for $j = 0, 1, \ldots, i-1$

Find the smallest positive real λ such that λv is in the lattice $\hat{L}$ obtained by projecting L onto the orthogonal complement of the span of $\{b_1, \ldots, b_{i-1}\}$

Find w in L such that w projects onto λv in $\hat{L}$.

$b_i = w$

end

return $(b_1, \ldots b_n)$

Proof : The proof of the last proposition actually proves this stronger result.

□

Of course the method presented here is not quite an algorithm - we do not know how the input is specified etc. I will later describe a more rigorous version of this algorithm called $SELECT - BASIS$ in section 2.

Theorem (1.11) (Minkowski)

If S is any convex set in $\mathcal{R}^n$ which is symmetric about the origin ($x \in S \Rightarrow -x \in S$) and has volume greater than 2^n, then S contains a nonzero point of Z^n.

Proof : Define $S/2 = \{x : x \in \mathcal{R}^n, 2x \in S\}$. Clearly, $S/2$ has volume greater than 1. Consider the convex bodies $v + S/2 = \{x : x \in \mathcal{R}^n, x = v + s \text{ for some } s \in S/2\}$ as v ranges over Z^n. There is one such body for each point of Z^n and their volumes are strictly greater than 1. Therefore, two of them must intersect. (I leave it to the reader to make this intuitive argument into a rigorous one.) Suppose $v + S/2$ and $u + S/2$ intersect, then so do $S/2$ and $(u - v) + S/2$. Let y be in their intersection. Then, y and $y - u + v$ both belong to $S/2$. So, $2y, 2(y - u + v)$ both belong to S. The symmetry of S implies that $-2y$ belongs to it, the convexity then implies that the average of $-2y$ and $2(y - u + v)$ which is $v - u$ belongs to S. Of course, $v - u$ is in Z^n proving the theorem.

□

I will generally only use the following direct consequence of Minkowski's theorem.

Theorem (1.12)

If L is an n-dimensional lattice with determinant $d(L)$, there is a nonzero element v of L with $|v| \leq \frac{1}{2}\sqrt{n}(d(L))^{1/n}$

Proof Let $L = Z^n B = \{x : x = yB \text{ for some } y \in Z^n\}$, B an $n \times n$ matrix with a basis of L as its rows. Consider the sphere T with the origin as center and radius $\frac{1}{2}\sqrt{n}(d(L))^{1/n}$. T has volume $\pi^{n/2}/\Gamma(n/2 + 1)R^n$ where R is its radius. So the volume of T is greater than $2^n d(L)$. T is convex and symmetric about the origin. Hence so is $TB^{-1} = \{x : \exists y \in T \text{ s.t. } x = yB^{-1}\}$. TB^{-1} has volume greater than $2^n d(L) \cdot det(B^{-1}) = 2^n$. Thus there is a y in $Z^n - \{0\} \cap TB^{-1}$. $v = yB$ is then a nonzero element of $T \cap L$. Clearly v is short enough to prove the theorem.

□

Remark (1.13) The factor $\sqrt{n}/2$ in the theorem can be improved by reckoning the volume of the $n-$ sphere more accurately. In fact, more sophisticated upper bounds on $\Lambda_1(L)/(d(L))^{1/n}$ are known. This is of course the ratio theorem (1.12) is bounding from above. The supremum value of the square of this ratio over all $n-$ dimensional lattices is called Hermite's constant and the best upper bound on it is $\frac{n}{e\pi}(1 + o(1))$ due to Blichfeldt (1929). See also Lekkerkerker (1969) - section 38.

The general references on the subject of Geometry of Numbers are Cassels (1959) and Lekkerkerker (1969). An expository survey of lattice basis reduction algorithms can be found in Kannan (1984).

2 The algorithm for finding the shortest vector

In this section , I describe an algorithm to find a shortest nonzero [3] vector in a lattice given by a basis $b_1, b_2, \ldots b_n$. This algorithm actually finds a "reduced basis"of the lattice of which the first vector will be the shortest vector in the lattice (the definition of a reduced basis used in this paper is found in (2.6)). The algorithm of this section will be used in the Integer Programming algorithm in two ways - it is needed as a subroutine there , perhaps more importantly, this section will develop a technique that is used both in the integer programming algorithm as well as the algorithm for the closest vector problem. I will describe an overview of this technique in the next paragraph.

The algorithm is a recursive procedure - it works by calling subroutines for lattices of dimension $n-1$ or less when n is the dimension of the given lattice. It will be shown that an "approximately" reduced basis can be found by these recursive calls. Then the shortest vector is found by enumerating all of a finite set of candidates. That this finite set is not too large will follow from Minkowski's theorem on convex bodies. The paper of Helfrich(1985) referred to earlier, improves the bound on the size of the finite set.

Here is how the algorithm works : Suppose we are given a basis $b_1, b_2, \ldots b_n$ of a lattice $L = L(b_1, b_2, \ldots b_n)$. Using polynomially (in n alone) calls to lower dimensional subroutines, the algorithm finds a basis $a_1, a_2, \ldots a_n$ for the lattice L which satisfies the following properties :

(2.1) For $j = 2, 3 \ldots n$, $a_j(j) = \Lambda_1(L_j(a_1, a_2, \ldots a_n))$ [4]

(2.2) $|a_1| \leq \frac{2}{\sqrt{3}}|a_2|$

(2.3) $|a_2(1)| \leq |a_1|/2$

This is the "approximately reduced " basis. Intuitively, these conditions can be understood by appealing to the representation of the basis $a_1, a_2, \ldots a_n$, as a lower triangular matrix. In such a representation, condition (2.1) says that for $j = 2, 3, \ldots n$, the j th diagonal entry is the length of the shortest vector in the lattice generated by the rows of the $(n-j+1) \times (n-j+1)$ matrix consisting of the last $n-j+1$ rows and columns of the basis matrix.

Whereas in the reduced basis of LLL (1982), the length of the first vector is guaranteed to be at most $2^{n/2} d(L)^{1/n}$, for the basis here , one can prove (2.4) below using Minkowski's theorem, conditions (2.1), (2.2) and (2.3) and the fact that $d(L) = |a_1| d(L_2(a_1, a_2, \ldots a_n))$.

$$|a_1| \leq \sqrt{2n}\ \ d(L)^{1/n} \tag{2.4}$$

In other words, our a_1 is much a shorter vector than theirs - but of course we will spend more time finding it.

---

[3] Henceforth, I will use the phrase "shortest vector" for "shortest nonzero vector" when the meaning is clear.

[4] See notation in section 1 for the definition of $L_j(a_1, a_2, \ldots a_n)$ and Λ_1(a lattice)

Having obtained such a basis $a_1, a_2, \ldots a_n$, I show that the shortest vector in the lattice must be of the form

$y = \sum_{i=1}^{n} \alpha_i a_i$ where $(\alpha_1, \alpha_2, \ldots \alpha_n) \in T$

where T is a subset of Z^n. I show that it is enough to consider a set T of cardinality at most

$$\frac{2^n |a_1|^n}{d(L)} \tag{2.5}$$

.

(2.4) is used to bound the expression (2.5) in terms of n alone. We enumerate all elements of T, find the corresponding y and take the shortest of these which must then be the shortest vector in the lattice. Intuitively, here is how (2.5) is derived. Let A be the basis matrix with rows $a_1, a_2, \ldots, a_n$ and $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be a row vector of integers so that λA is a shortest nonzero vector of the lattice. Since a_1 is obviously a nonzero vector in the lattice, the shortest vector must have length at most $|a_1|$, thus λA must belong to a cube of side $2|a_1|$ with the origin as center and edges parallel to the axes. This cube has volume $2^n |a_1|^n$. Applying the linear transformation A^{-1} to the cube we get a parallelpiped P of volume $2^n |a_1|^n det(A^{-1})$ which equals the expression (2.5) and the integer vector λ must belong to P for λA to belong to the cube. So, we can enumerate all the integer vectors in P and find the one that leads to the shortest nonzero vector of the lattice. We would expect the number of integer vectors in P to be equal to the volume of P. This describes the idea behind (2.5), I caution that a proper argument involves several delicate points which will be dealt with later.

We find an entire reduced basis instead of just the shortest vector to facilitate the recursion. First, here is the definition of **reduced basis** with which we will work.

Definition 2.6 : A basis $v_1, v_2, \ldots v_n$ of the lattice $L(v_1, v_2, \ldots v_n)$ is called a *reduced basis* if (2.7) and (2.8) below are satisfied.

$$\text{For } j = 1, 2, \ldots n, \qquad v_j(j) = \Lambda_1(L_j(v_1, v_2, \ldots v_n)) \quad [5] \tag{2.7}$$

$$|v_i(j)| \leq v_j(j)/2 \quad \text{for} \quad i \geq j+1 \geq 2 \tag{2.8}$$

Note the difference between (2.1) and (2.7) is that (2.7) includes $j = 1$ also whereas (2.1) does not. Thus in the lower triangular representation, every diagonal entry is the length of the shortest vector in the lattice generated by the rows of the square submatrix of which it is the top left entry . The essential feature of the LLL reduced basis is that in the lower triangular representation, the j th diagonal entry is the length of the shortest vector in the 2-dimensional lattice generated by the rows of the submatrix containing the rows $j, j+1$ and columns $j, j+1$ of the basis matrix. Here instead the submatrix is not 2×2, but $(n-j+1) \times (n-j+1)$. Schnorr (1984) generalizes the LLL reduced basis to allow $k \times k$ submatrices for any fixed k. Schnorr's algorithm uses the algorithm $SHORTEST$ of

[5] Thus v_1 is a shortest vector in the whole lattice

this section as a subroutine to make the $k \times k$ matrices reduced in the sense defined here. He arranges the reduction of the various $k \times k$ matrices so as to make only polynomially many calls on the subroutine.

A detailed description of the algorithm $SHORTEST$ is given below followed by a proof of correctness and bounds on the running time.

Procedure $SHORTEST(n; b_1, b_2, \ldots b_n)$

Comment The preceding paragraphs explain what the algorithm accomplishes. $L = L(b_1, b_2, \ldots b_n)$

1. If $n = 1$ then return $\{b_1\}$
2. Use the basis reduction algorithm from Lenstra, Lenstra and Lovász to make the basis reduced in their sense.
3. $\overline{b_i}' \leftarrow$ projection of b_i perpendicular to b_1 for $i = 2, 3, \ldots n$
4. $\overline{b_2}, \overline{b_3}, \ldots \overline{b_n} \leftarrow SHORTEST(n-1; \overline{b_2}', \overline{b_3}', \ldots \overline{b_n}')$
5. For $i = 2$ to n do :
   6. Lift $\overline{b_i}$ to b_i in L. (cf: Paragraph preceding proposition 1.8. The proof of theorem 3.8 contains the mundane details of how the lifting is done.)
7. end

   Comment We now have a basis of L satisfying conditions (2.1) and (2.3)

8.If $|b_2| < \frac{\sqrt{3}}{2}|b_1|$ then do :

   Swap b_1 and b_2

   GOto 3

9.end

   Comment We now satisfy condition (2.2) also. Caution : $|b_1|, |b_2|$ may be irrational, but their squares are not. So we use them instead.

10. If $|b_j(j)| \geq |b_1|$ for some j then $j_0 \leftarrow$ minimum such j, else $j_0 \leftarrow n+1$.
11. $BASIS \leftarrow \{b_1, b_2, \ldots b_{j_0-1}\}$

    Comment I show later that some nonzero shortest vector of L is in $L(BASIS)$

12. Call $ENUMERATE(BASIS)$ to obtain a shortest vector in $L(BASIS)$. Call this vector v_1.

    Comment This procedure is explained later.

13.$\{b_1, b_2 \ldots b_n\} \leftarrow SELECT - BASIS(n; v_1, b_1, b_2 \ldots b_n)$

    Comment Procedure explained later. It returns a basis of L containing v_1 as the first vector - cf proposition 1.9.

14.Execute steps 3,4 ,5,6 and 7.

15. Return $\{b_1, b_2, \ldots b_n\}$

end $SHORTEST$

Procedure $SELECT - BASIS(n; b_1, b_2, \ldots b_{n+1})$

Comment $b_1, b_2, \ldots b_{n+1}$ are vectors in $\mathcal{Q}^k$ for some $k \geq n$ and span an $n-$dimensional subspace of $\mathcal{R}^k$. The procedure returns a basis $a_1, a_2, \ldots a_n$ of $L = L(b_1, b_2, \ldots b_{n+1})$. It first finds a shortest lattice vector in the direction of b_1 - call this a_1. Then it projects L orthogonal to a_1 to get a lattice $\overline{L}$. It works by recursively finding a basis of $\overline{L}$. (cf. proposition 1.10) - See proposition 2.9 for more documentation on the procedure.

1. If $n = 0$ then return the empty set.
2. If $b_1 = 0$ then return $\{b_2, b_3, \ldots b_{n+1}\}$.
3. If b_1 is linearly independent of $b_2, \ldots b_{n+1}$
   then $a_1 \leftarrow b_1$
   else do:
   4. Find $\alpha_2, \ldots \alpha_{n+1}$ (rationals - these are unique) such that $\sum_{j=2}^{n+1} \alpha_j b_j = b_1$
   5. $M \leftarrow$ least common multiples of denominators of $\alpha_2, \ldots \alpha_{n+1}$
   6. $\gamma \leftarrow GCD(M\alpha_2, M\alpha_3, \ldots M\alpha_{n+1})$
   7. If M/γ is an integer then $a_1 \leftarrow b_1$
      else do
      8. find $p, q \in Z$ relatively prime so that $p/q = (M/\gamma)$
      9. $a_1 \leftarrow (1/q)b_1$
      end
   end
10. $\overline{b}_i \leftarrow$ projection of b_i perpendicular to a_1 for $i = 2, 3, \ldots n+1$
11. $\{c_2, c_3, \ldots c_n\} \leftarrow SELECT - BASIS(n-1; \overline{b}_2, \ldots \overline{b}_{n+1})$
12. Lift c_i to a_i in L for $i = 2, 3, \ldots n$
13. **Return** $\{a_1, a_2, \ldots a_n\}$

end $SELECT - BASIS$

Proposition 2.9: The basis $a_1, a_2, \ldots a_n$ returned by the above procedure is a basis of $L = L(b_1, b_2, \ldots b_{n+1})$ assuming that $b_1, b_2, \ldots b_{n+1}$ span an $n-$dimensional subspace.

Proof By proposition (1.10), it suffices to show that a_1 is a shortest vector of L in the direction of b_1 provided b_1 is not equal to zero. We have

$Mb_1 = \gamma\left(\frac{M\alpha_2}{\gamma}b_2 + \frac{M\alpha_3}{\gamma}b_3 + \ldots \frac{M\alpha_{n+1}}{\gamma}b_{n+1}\right)$

and $\frac{M\alpha_2}{\gamma}, \frac{M\alpha_3}{\gamma}, \ldots \frac{M\alpha_{n+1}}{\gamma}$ are relatively prime integers. Any other vector in $L(b_2, b_3, \ldots b_{n+1})$ in the direction of b_1 must thus be an integer multiple of $(M/\gamma)b_1 = (p/q)b_1$. Thus any vector of $L(b_1, b_2, \ldots b_n, b_{n+1})$ in the direction of b_1 is an integer multiple of $(1/q)b_1$. Conversely, $(p/q)b_1$ and $(q/q)b_1$ belong to L and p, q are relatively prime implies that $(1/q)b_1$ does too.

□

Proposition 2.10 : The vectors returned by the procedure $SHORTEST(n; b_1, b_2, \ldots b_n)$ form a basis of $L(b_1, b_2, \ldots b_n)$.

Proof : For $n = 1$, the proof is clear. I proceed by induction on n. At the end of step 4 of the procedure, $\overline{b}_2, \ldots \overline{b}_n$ form a basis of the lattice $L_2(b_1, b_2, \ldots b_n)$ and thus by proposition 1.10, $b_1, b_2, \ldots b_n$ form a basis of $L(b_1, b_2, \ldots b_n)$ at the end of step 7. By

repeating the argument, they form a basis of L at the end of step 9. By proposition 2.9, procedure $SELECT-BASIS$ works correctly to produce a basis of the lattice. Hence, the current proposition is proved.

□

Proposition 2.11: Let j_0 be as defined in step 10 of $SHORTEST$. Then a shortest vector of $L(b_1, b_2, \ldots b_{j_0-1})$ is also a shortest vector of $L(b_1, b_2, \ldots b_n)$.

Proof : Suppose $v = \sum_{i=1}^{n} \alpha_i b_i$ is a shortest nonzero vector of $L(b_1, b_2, \ldots b_n)$ and one of $\alpha_{j_0}, \alpha_{j_0+1}, \ldots \alpha_n$ is nonzero. Then the projection v' of v onto the vector space V =the orthogonal complement of Span $(b_1, b_2, \ldots b_{j_0-1})$ is nonzero. Therefore we must have

$|v'| \geq \Lambda_1(L_{j_0}(b_1, b_2, \ldots b_n))$.

Then clearly, $|v| \geq |v'| \geq b_{j_0}(j_0) \geq |b_1|$. Thus b_1 is a shortest vector of L.

□

Proposition 2.12 : The procedure $SHORTEST$ executes recursive call of step 4 at most $(5/2)n$ times when started on an $n-$dimensional lattice.

Proof : By Lenstra, Lenstra and Lovász, the execution of their basis reduction algorithm in step 2 of procedure $SHORTEST$ yields a basis of L with $|b_1| \leq 2^{n/2}\Lambda_1(L)$. Each execution but the first of the loop steps 3-9 of $SHORTEST$ cuts down $|b_1|$ by a factor of at least $\sqrt{3}/2$. Thus each 5 iterations of the loop cuts it down by a factor of 2. We cannot reduce $|b_1|$ further once it reaches $\Lambda_1(L)$. Thus at most $5n/2$ executions of the loop suffice.

□

Description of procedure *ENUMERATE*

The crucial reason that we can complete the recursion is that we can enumerate relatively few candidates to determine the shortest vector. This fact is proved now. Suppose $j_0 - 1 = m$ in step 10 of procedure $SHORTEST$ and suppose a shortest vector of $L(b_1, b_2, \ldots, b_m)$ is $y = \sum_{i=1}^{m} \alpha_i b_i$. Then since y must be of length at most $|b_1|$, the projection of y onto V_m, the orthogonal complement in $\mathbf{R}^n$ of the span of $\{b_1, b_2, \ldots, b_{m-1}\}$ must be of length at most $|b_1|$. This projection has length $|\alpha_m b_m(m)|$, so we must have

$$|\alpha_m| \leq |b_1| \Big/ |b_m(m)|$$

More generally, we have the following proposition. The reader might want to use the lower triangular representation of the basis matrix to understand the proposition.

Proposition 2.13 With the above notation, suppose $\beta_{i+1}, \beta_{i+2}, \ldots, \beta_m$ are fixed integers. Then there is an easily computed integer β_i^0 such that for all integers $\alpha_1, \alpha_2 \ldots, \alpha_{i-1}$ and β_i,

$$\left| \sum_{j=1}^{i-1} \alpha_j b_j + \beta_i b_i + \sum_{j=i+1}^{m} \beta_j b_j \right| \leq |b_1|$$

$$\Longrightarrow \beta_i \in [\beta_i^0 \qquad \beta_i^0 + 2\frac{|b_1|}{|b_i(i)|}]$$

Proof : For any vector v, I denote by $\hat{v}$, the projection of v along the direction of b_i^* in this proof. Let $u = \sum_{j=i+1}^{m} \beta_j b_j$ and $w = \sum_{j=1}^{i-1} \alpha_j b_j + \beta_i b_i + u$. Clearly, $\hat{w} = \beta_i b_i^* + \hat{u}$

$= \beta_i b_i^* + t b_i^* (say)$ where t is some fixed real number (since $\beta_{i+1}, \beta_{i+2}, \ldots, \beta_m$ and hence u are fixed.)

So we have,

$|w| \leq |b_1| \Longrightarrow |\hat{w}| \leq |b_1| \Longrightarrow |(\beta_i + t)| \leq |b_1| \Big/ |b_i^*| \Longrightarrow -t - \frac{|b_1|}{|b_i^*|} \leq \beta_i \leq -t + \frac{|b_1|}{|b_i^*|}$

So the proposition follows with

$\beta_i^0 = -t - \frac{|b_1|}{|b_i^*|}$

□

Note: I needed $|b_1| \geq b_i(i)$ to state the proposition as it is. If $|b_1| < b_i(i)$, we need to try at most 2 values of α_i- **not** $2(|b_1|/b_i(i))$.

The proposition bounds the number of candidates for β_i and leads directly to the following procedure $ENUMERATE$.

Procedure $ENUMERATE(b_1, b_2, \ldots, b_m)$

Comment $b_1, b_2, \ldots, b_m$ are vectors satisfying (2.1),(2.2) and (2.3). The procedure finds the shortest vector of $L(b_1, b_2 \ldots, b_m)$.

if $m = 1$ then return b_1.

for each integer α_m in the range $[-\frac{|b_1|}{b_m(m)} \qquad + \frac{|b_1|}{b_m(m)}]$ do :

call $LIST(m-1)$

end

Comment $LIST(m-1)$ returns T - a list of candidates $\{\alpha_1, \alpha_2 \ldots, \alpha_m\}$ as per proposition 2.13.

Return the shortest nonzero vector in the set $\{\sum_{j=1}^m \alpha_j b_j : (\alpha_1, \alpha_2 \ldots, \alpha_m) \in T\}$

end $ENUMERATE$.

Procedure $LIST(k)$

Comment: When this procedure is called, $\alpha_{k+1}, \alpha_{k+2}, \ldots, \alpha_m$ are already known integers .

if $k = 0$ then do:

$T \leftarrow T \bigcup \Big\{ (\alpha_1, \alpha_2 \ldots, \alpha_m) \Big\}$

Return

end

Compute β_k^0 based on proposition 2.13.

for each integer α_k in the range $[\beta_k^0 \qquad \beta_k^0 + 2\frac{|b_1|}{b_k(k)}]$ do:

Call $LIST(k-1)$.

end

end $LIST$

Lemma (2.14)

At the end of procedure $ENUMERATE$, $|T|$ *is at most* $2^m \prod_{i=2}^m \Big(|b_1|/b_i(i) \Big)$ *which is at most* $(2n)^{n/2}$

Proof : The first part follows from the last proposition. The denominator $\prod_{i=2}^{m} b_i(i)$ is of course the determinant of the lattice $L_2(b_1, b_2, \ldots, b_m)$- call it L_2 for short in the rest of the proof.

Since $b_2(2) = \Lambda_1(L_2)$,

$$\frac{b_2(2)^{m-1}}{d(L_2)} \leq (\sqrt{m-1})^{m-1}/2^{m-1} \leq n^{(n-1)/2}/2^{m-1}$$

(by Minkowski's theorem (1.11)).

Further, $|b_1| \leq \frac{2}{\sqrt{3}}|b_2|$ and $|b_2(1)| \leq \frac{|b_1|}{2}$ imply that $\sqrt{2}b_2(2) \geq |b_1|$. Thus we need to enumerate at most

$$2^m(\sqrt{2})^{m-1}n^{(n-1)/2}/2^{m-1} \leq (2n)^{n/2}$$

possibilities.

□

Proposition 2.15 : Procedure *Enumerate* correctly finds the shortest nonzero vector of $L(b_1, b_2, \ldots b_m)$.

Proof : Obvious

Proposition 2.16 The basis $b_1, b_2, \ldots b_n$ returned by $SHORTEST$ satisfies the conditions (2.7) and (2.8).

Proof : Induction on n. $n = 1$ is obvious. By proposition 2.11, the shortest vector of $L(BASIS)$ in step 11 is also the shortest vector of the whole $n-$ dimensional lattice. By proposition 2.15, the vector v_1 at the end of step 12 is indeed a shortest vector of the lattice. Using proposition 2.9 and the inductive assumption on step 14, the current lemma follows.

□

This completes the proof of correctness. As for the time bound, I will split it into two parts : a bound on the number of arithmetic operations - additions, subtractions, multiplications, divisions and comparisons with operands that are rational numbers, and a bound on the operand sizes. The number of arithmetic operations will depend on the dimension n of the problem as well as the length s of the input. However, going through the procedure $SHORTEST$ step by step, we see that the total number of arithmetic operations performed *while the procedure is not inside a call to LLL basis reduction algorithm* is bounded by a function of n alone - it does not depend on s. This is seen by an inductive proof using proposition 2.12. Unfortunately, the same does not hold for LLL. In the next section (proposition 3.8), I show that the total number of arithmetic operations performed by $SHORTEST$ in all the calls to LLL is $n^n s$. For now, I will assume this proposition.

Theorem (2.17)

$SHORTEST(n; \ldots)$ finds a reduced basis satisfying (2.7) and (2.8) in $O(n^n s)$ arithmetic operations where s is the length of the input.

Note: In common usage, we might call this a $0(n^n s)$ - algorithm. This, however, counts only the number of arithmetic operations, and ignores the size of the operands. In

an algorithm such as this one which manipulates numbers and keeps them all precisely, it is important to prove bounds on the size of the numbers. I do so in the next section.

Proof : Let $T(n)$ be the maximum number of arithmetic operations performed by $SHORTEST(n;\ldots)$ while not inside a call to LLL. It is easily seen that all steps of the algorithm except recursive calls to shortest, the enumeration and calls to LLL call for a number of arithmetic operations bounded by a polynomial in n alone. Thus we have (by proposition 2.12 and lemma 2.14),

$$T(n) \leq \frac{5n}{2}T(n-1) + (2n)^{n/2}.q(n)$$

(q -a polynomial). I will derive the bound $T(n) \in O(n^n)$. $Lim_{n\to\infty}\ (1-\frac{1}{n})^{n-1} = 1/e$ and $\frac{5}{2e}$ is less than .93, so there exists an N_1 such that for all $n > N_1$, we have $\frac{5}{2}(1-\frac{1}{n})^{n-1} \leq .95$. Further, let N_2 be a natural number so that $\forall n \geq N_2$, $(2n)^{n/2}q(n) \leq .05n^n$. Let N be the maximum of N_1, N_2. Choose a constant $c \geq 1$ such that $T(n) \leq cn^n \forall n \leq N$. Now, I argue by induction on n that $T(n) \leq cn^n$ for all n. For $n \leq N$, this is true by definition. So, assume $n > N$ and suppose it is true for $n-1$. Then $T(n)/(cn^n) \leq \frac{5}{2}(1-\frac{1}{n})^{n-1} + (2n)^{n/2}q(n)/(cn^n) \leq .95 + .05 = 1$. This completes the inductive proof. The total number of arithmetic operations performed by the algorithm is $T(n)+$ the number of operations performed while executing calls to LLL. From proposition 3.8, then the current theorem follows.

□

Remark (2.18) : Here, I considered the shortest vector in the Euclidean (L_2) norm. We can also define the shortest vector according to other norms in the obvious fashion. To find the L_1 shortest vector in a lattice, we proceed as follows : We apply $SHORTEST$ to the basis. Then, analogous to proposition 2.11, I claim now that if we choose $j_0 = Min\{j : b_j(j) \geq \sqrt{n}b_1(1)\}$, then a L_1 shortest vector of $L(b_1, b_2, \ldots b_{j_0-1})$ is also an L_1 shortest vector of the whole lattice. This is because any vector in $L(b_1, b_2, \ldots b_n) \setminus L(b_1, b_2, \ldots b_{j_0-1})$ must have L_2 norm at least $\sqrt{n}b_1(1)$ and therefore L_1 norm at least $\sqrt{n}b_1(1)$ which is clearly at least the L_1 norm of b_1. Let $m = j_0 - 1$. In any candidate, $\sum_{j=1}^{m}\lambda_j b_j$ for the L_1 shortest vector, we must have $\lambda_m b_m(m) \leq |b_1|_1 \leq \sqrt{n}b_1(1)$. Thus there are at most $2\sqrt{n}b_1(1)/b_m(m)$ candidates for λ_m. Arguing in this vein, the total number of candidates is at most $2^m n^{\frac{1}{2}n}\prod_{j=1}^{m}\frac{b_1(1)}{b_j(j)}$ which is at most n^n by Minkowski's theorem. This will give an algorithm for finding the L_1 shortest vector in $O(n^n s)$ arithmetic operations. Similar ideas work for the L_∞ or for any other $L_p$norms. van Emde Boas (1981) has shown that the shortest vector problem for the L_1 and L_∞ norms is NP-complete.

3 Size of the numbers involved in the algorithm

We assume that the original input consists of integers. It is easy to see then that all the numbers produced by the algorithm are rational numbers. In what follows, I will derive bounds on the size of the numerators and denominators of all these numbers. The numerator of a rational is of course bounded in absolute value by its magnitude, so really the bounds will be on the magnitude and the denominator of each rational .

First,we will observe that even though the algorithm works on various projected lattices, there is always an implicit "current basis" of the original input $n-$dimensional lattice. This is true of step 2 (of $SHORTEST$) from the Lenstra, Lenstra and Lovász algorithm. In step 4, we work on the projected lattice $L_2(b_1, b_2, \ldots b_n)$, but since there is a natural way to "lift" any element of $L_2(b_1, b_2, \ldots b_n)$ to $L(b_1, b_2, \ldots b_n)$ (in section 1), we can assume that implicitly we have a basis of the whole lattice $L(b_1, b_2, \ldots b_n)$ provided we can assume that during step 4, while the algorithm is working on $L_2(b_1, b_2, \ldots b_n)$, it has a basis of $L_2(b_1, b_2, \ldots b_n)$. By induction, we may indeed assume this and thus there is always an implicit basis of the whole lattice during step 4. Step 5 explicitly computes this implicit basis. By the definition of lifting, note that the basis constructed in step 5,6,7 satisfies (2.8) - we will refer to any such basis as "proper". The LLL algorithm always explicitly maintains a basis of the input lattice. Unfortunately, however this basis is not proper at all times. However, when the LLL algorithm terminates, the basis will be proper.

Running through the algorithm $SHORTEST$, we see that at the end of step 9, there is a "current basis"of the whole lattice which is not disturbed until $SELECT-BASIS$ is executed in step 13. At the end of step 13, we have a different current basis , but it is easy to see by induction and the definition of "lifting" applied to step 12 of $SELECT-BASIS$, that the basis at the end of step 13 of $SHORTEST$ is proper. The argument for step 14 (of $SHORTEST$) is similar to steps 3,4 and 5. In summary, I have argued the following :

Proposition 3.1 : There is an (implicit) "current basis" of the whole lattice at all times during the execution of the algorithm $SHORTEST$. This basis is proper (satisfies (2.8)) except possibly in the middle of the execution of the LLL algorithm.

□

In what follows I talk about certain properties of the "current basis" which I will refer to as $b_1, b_2, \ldots b_n$. With this we can associate the quantities $b_i(j)$ as defined in (1.7).

Proposition 3.2: $\mathrm{Max}_{i=1}^{n} b_i(i)$ never increases during the execution of $SHORTEST$.

Proof : We consider the algorithm step by step. The proof is by induction on n. For $n = 1$, the proof is trivial. So assume $n \geq 2$. For step 2, the LLL algorithm never increases the quantity as seen from their proof of their proposition (1.26). For step 4, the inductive hypothesis suffices. In step 8, $b_1(1)$ strictly decreases, the new $b_2(2)$ is at most the old $|b_1|$ and $b_3(3), \ldots, b_n(n)$ remain the same. For steps 11 through 13, the enumeration and basis selection processes, the proof is a little harder and is dealt with in proposition 3.3. For step 14 again, we invoke the inductive hypothesis, completing the proof of this proposition.

□

Proposition 3.3: Steps 11 through 13 of the algorithm $SHORTEST(n; b_1, \ldots, b_n)$ do not increase $\mathrm{max}_i b_i(i)$.

Proof : Suppose $b_1, b_2, \ldots, b_n$ is the basis of the lattice at the beginning of step 11. Let $b_i(j), 1 \leq j \leq i \leq n$ be defined as in (1.7). Suppose v_1 is found to be shortest nonzero vector of $L(b_1, b_2, \ldots, b_n)$ by enumeration. Define $u_1 = v_1, u_2 = b_1, u_3 = b_2, \ldots, u_{n+1} = b_n$. Let $u_i(j), 1 \leq j \leq i \leq n+1$ be defined again as in (1.7), i.e., by performing Grahm-Schmidt on $u_1, u_2, \ldots u_{n+1}$. Clearly, precisely one of the $u_i(i)$'s is zero. Let this be $u_j(j)$.

Let $v_1, v_2, \ldots, v_n$ be the basis returned by SELECT-BASIS in step 13. Again define $v_i(j)$ by (1.7). Then for $l = 2, 3, \ldots, j-1$, v_l 's projection onto the orthogonal complement of span of $\{v_1, v_2, \ldots, v_{l-1}\}$ must be a scalar multiple of u_l 's projection onto the same space. Thus by induction on l, span $\{v_1, v_2, \ldots, v_{l-1}\}$ equals span $\{u_1, u_2, \ldots, u_{l-1}\}$ and hence we must have

$$v_l(l) \leq u_l(l) \text{ for } l = 2, 3, \ldots, j-1.$$

$u_l(l)$ is the length of the projection of b_{l-1} orthogonal to the span of $\{v_1, b_1, b_2, \ldots b_{l-2}\}$. $b_{l-1}(l-1)$ is the length of the projection of b_{l-1} orthogonal to the span of $\{b_1, b_2, \ldots b_{l-2}\}$. So, we must have $u_l(l) \leq b_{l-1}(l-1)$. So,

$$v_l(l) \leq b_{l-1}(l-1) \quad \text{for} \quad l = 2, 3 \ldots, j-1$$

Further, since $u_l(l) \neq 0$ for $l = j+1, j+2, \ldots, n+1$, $u(l,l)$ (see $(1.7)'$) is independent of $u(l+1,l), u(l+2,l), \ldots u(n+1,l)$ and so $SELECT-BASIS$ (in its step 3) will make $v_l(l) = u_{l+1}(l+1)$ for $l = j, j+1, \ldots, n$. So we have

$$v_l(l) = u_{l+1}(l+1) \leq b_l(l) \text{ for } l = j, j+1, \ldots, n$$

The two inequalities together establish the proposition.

□

We now define, for any basis $b_1, \ldots, b_n$ of the n-dimensional lattice

$$d_i = d(L(b_1, b_2, \ldots, b_i))^2.$$

It is not difficult to see that d_i is the determinant of the $i \times i$ matrix with entries (b_j, b_l) for $1 \leq j, l \leq i$. Since our original basis vectors had integer coordinates, this is also true of any other basis. Thus the d_i are all integers. Clearly,

$$d_i = \prod_{j=1}^{i} |b_j(j)|^2 \tag{3.4}$$

The following proposition resembles a similar one in the LLL paper.

Proposition 3.5 All numbers produced by the algorithm are rationals of the form p/q, p, q in Z where q is one of the d_i's corresponding to the current basis .

Proof : Let $b(j,i)$ be the projection of b_j orthogonal to $b_1, \ldots, b_{i-1}$(for $j \geq i \geq 2$). (See $(1.7)'$) Then $b(j,i) = b_j - \sum_{k=1}^{i-1} \delta_{jk} b_k$ where δ_{jk} are some real numbers. Taking a dot product with $b_l (1 \leq l \leq i-1)$ and noting that $(b_l, b(j,i)) = 0$, we have

$$(b_j, b_l) = \sum_{k=1}^{i-1} \delta_{jk} (b_k, b_l) \text{ for } l = 1, 2, \ldots, i-1.$$

These are $(i-1)$ independent equations in the $(i-1)$ variables δ_{jk} with a coefficient matrix whose determinant is d_{i-1}. Thus $d_{i-1}\delta_{jk}$ are all integers. Hence $d_{i-1}b(j,i)$ is an

integral vector. Now, the algorithm *SHORTEST* keeps these vectors $b(j,i)$ for some i as it works on projected lattices. In addition, it has to keep some auxiliary quantities at various times - the μ_{ij}'s during LLL, certain other quantities during the execution of the enumeration and select-basis steps. The proof of the proposition for other quantities is similar and I omit it.

□

Lemma (3.6)

Except while executing the Lenstra, Lenstra and Lovasz algorithm , the current basis contains vectors of length at most $(nB)^{1/2}$ *if the original input* $b_1, \ldots, b_n$ *consisted of integral vectors each of length at most* $\sqrt{B}$.

Proof : By proposition 3.1, the current basis is always proper in these situations which implies of course that if we did Grahm-Schmidt on the current basis, the μ_{kl} of (1.6) are all at most $1/2$ in magnitude. Further, the initial $b_i(i)$ is at most $\sqrt{B}$ by hypothesis and so by proposition 3.2, they are all always bounded by this quantity. Thus using (1.7), and properness, we observe that the length of b_i in the current basis is at most $\left(|b_i^*|^2 + (1/4)\sum_{j=1}^{i-1}|b_j^*|^2\right)^{1/2}$ which is at most $(nB)^{1/2}$.

□

Lemma (3.7)

In $SHORTEST(n; b_1, \ldots, b_n)$ *during every execution of LLL algorithm all numbers produced are bounded in magnitude by* $(\sqrt{n}B)^{cn^2}$ *for some fixed constant c.*

Proof: Whenever the LLL algorithm is called, all the input vectors to it -say - $a_1, a_2, \ldots, a_i$ have rational components with common denominator d where by Proposition 3.5, d is one of the d_i and hence by (3.4) and by proposition (3.2), is bounded by B^i. Also, by the previous lemma, the lengths of the vectors are all bounded by $(\sqrt{nB})$. Further, it is easily seen that the *LLL* algorithm behaves identically on input $(da_1, da_2 \ldots, da_i)$ as it does on input $a_1, a_2, \ldots, a_1$ except that in the second case all vectors are divided by d. $(da_1, \ldots, da_i)$ are integral vectors and thus the bounds proved in the *LLL* paper apply to them. For these input, we have (from their Proposition (1.26)) that all numbers produced by their algorithm are bounded in magnitude by

$(\max|da_i|)_{i=1}^n)^{cn}$ which is at most $(B^{n-1}\sqrt{nB})^{cn} \leq (\sqrt{n}B)^{cn^2}$

□

Proposition 3.8 : The total number of arithmetic operations performed by *SHORTEST* while executing calls to the LLL algorithm is $O(n^n \log B)$.

Proof First, let us bound the total number of times LLL is called. By proposition 2.12, this is at most $(\frac{5}{2})^n n!$. Using the argument in lemma 3.7, each call to LLL performs at most as many arithmetic operations as a call to LLL with integer input vectors each of length at most $B^{n-1}\sqrt{nB}$ which is at most $\sqrt{n}B^n$. Using their proposition 1.26 then, we have that each call to LLL performs at most $O(n^4 \log(\sqrt{n}B^n)) = O(n^5 \log B + n^4 \log n)$ arithmetic operations. So the total number of operations performed by all calls to LLL is

$(\frac{5}{2})^n n!(n^5 \log B + n^4 \log n)$. Using Stirling's approximation and the fact that $\frac{5}{2}$ is strictly less than e, the base of the natural logarithm, we see that this is asymptotically $O(n^n \log B)$.

□

Theorem (3.9)

On input $b_1, \ldots, b_n$ which are independent vectors with integer components each of length at most $\sqrt{B}$, all numbers produced by the algorithm $SHORTEST(n; b_1, \ldots, b_n)$ can be represented in $0(n^2(\log n + \log B))$ bits.

Proof : The proof will be based on lemma 3.6. It is not by induction on n - I will actually consider the execution of the recursive calls in detail. Let us consider any call to the procedure $SHORTEST(i; u_1, u_2, \ldots u_i)$ (where i is less than n) occurring inside the main call to $SHORTEST(n; ....)$. For each such call, I will consider the execution of steps 1 through 3 and steps 5 through 13 . (In other words, I do not consider the steps invoking the recursive calls since I have in the first place picked any arbitrary call to the procedure inside of the main program.) Step 1 is trivial, step 2 is covered by lemma 3.7. In step three we have to project vectors - $u_2, u_3, \ldots u_i$ perpendicular to a vector u_1 where of course, these u_j form the basis of some projected lattice. Arguing as in lemma 3.6, we see that the u_j are all bounded in length by $\sqrt{nB}$. By (3.5), their denominators are bounded by B^{n-1}. Since projecting perpendicular to a vector involves taking certain dot products and simple arithmetic operations, it is easy to see that step 3 never involves more than $O(n(\log n + \log B))$ bit integers. Step 6 is a little harder to analyze partly because I have not specified exactly how the lifting is done. I will do so presently. Suppose $u_1, u_2, \ldots u_i$ is the basis of the lattice in step 3 and suppose $\overline{u}'_j, \quad j = 2, 3, \ldots i$ are the projections perpendicular to $u_1$in step 3, let $\overline{U}'$ be a matrix with these $i-1$ vectors as its $i-1$ rows. Further, let $\overline{u}_2, \overline{u}_3, \ldots \overline{u}_i$ is the basis returned in step 4 after the call to $SHORTEST(i-1; ....)$ and let $\overline{U}$ be the matrix with these $i-1$ vectors as its $i-1$ rows. . To lift these vectors, we do the following : We solve a linear system of equations in $(i-1)^2$ variables to find a $(i-1) \times (i-1)$ matrix T so that

$$\overline{U} = T\overline{U}\prime$$

Clearly, T so found will have integer entries and the determinant of it will be 1 in absolute value. Now let V equal TU where U is the matrix with $u_2, u_3, \ldots u_i$ as its $i-1$ rows. Then the rows of V are nearly what we want. We need to ensure that for each row of V, the projection of the row onto u_1 is at most $(1/2)|u_1|$ in length. This is done without much difficulty. The solution of the simultaneous equations with a coefficient matrix with entries of $O(n(\log n + \log B))$ bits does not produce any numbers larger than $O(n^2(\log n + \log B))$ bits. (Edmonds 1967)

All other steps are easily handled. In fact the only other step in which the size of numbers exceeds $O(n(\log n + \log B))$ bits is in the $SELECT - BASIS$ step when we have to solve equations with the coefficient matrix entries with $O(n \log B)$ bits - in this case the number of bits still remains $O(n^2 \log B)$.

□

4 Finding the closest vector

In this section, I consider the following *closest vector problem* :

(4.1) Given $b_1, b_2, \ldots b_n$ independent vectors in $\mathcal{Q}^n$ and b_0 in $\mathcal{Q}^n$, find b in $L(b_1, b_2, \ldots b_n)$ such that $|b_0 - b|$ is as small as possible.

This is called the inhomogeneous problem (corresponding to the homogeneous problem called the Shortest Vector Problem earlier). The reason for this terminology is that in the SVP we had to find the closest lattice point to 0, excluding itself whereas here we have to find the closest lattice point to an arbitrary b_0. Note however that here if b_0 itself belongs to the lattice, then the answer to be returned is b_0 - in other words, here we do not exclude b_0 as an answer. We can test in polynomial time whether b_0 in fact belongs to the lattice by using the algorithm of von zur Gathen and Sieveking (1976) or Kannan and Bachem (1979) to solve simultaneous diophantine equations , so I assume this is done at the outset and in what follows b_0 does not belong to the lattice.

Whereas as I remarked in the introduction, the complexity of the SVP is unknown at present, the CVP (closest vector problem) is easily shown to be NP-hard. I will argue this in section 6. So it is the case that the CVP is at least as hard as the SVP (since the latter obviously is in NP when properly coded as a language as was done in the introduction). I give an algorithm here to solve the CVP. This serves two purposes - it of course gives a solution to the problem on hand and secondly, it introduces an idea that will be useful in the integer programming algorithm.

The CVP algorithm functions as follows : It first uses the procedure $SHORTEST$ to make the basis $b_1, b_2, \ldots b_n$ reduced - i.e., the basis then satisfies (2.7) and (2.8). Next, we use an upper bound

$\frac{1}{2}\left(\sum_{j=1}^{n}(b_j(j))^2\right)^{1/2} = M$ (say)

on the distance between any b_0 and its closest lattice point. (This bound will be proved in proposition (4.2).) Because of this I can argue that there are not too many values of $(\alpha_1, \alpha_2, \ldots \alpha_n)$ integers such that $|\sum_{j=1}^{n} \alpha_j b_j - b_0|$ is within the upper bound. Arguing as in the case of shortest vector problem, (lemma (2.14)), this gives us a bound of $M^n/d(L)$ on the number of possible $n-$tuples $(\alpha_1, \alpha_2, \ldots \alpha_n)$ to enumerate. Unfortunately, this will not in general be bounded by a function of n alone. So we have to use another idea : If $b_i(i)$ is the largest among all the $b_j(j)$, then I will show that not too many values of $(\alpha_i, \alpha_{i+1}, \ldots \alpha_n)$ are candidates to be tried. The bound on the number of candidates will be $n^{(n-i+1)}$. For each such candidate, we project to a $(i-1)$ dimensional problem and solve these recursively. The details are explained after the algorithm.

Procedure $CLP(n; b_0, b_1, b_2, \ldots b_n)$

Comment : This procedure returns the vector in $L(b_1, b_2, \ldots b_n)$ that is closest in Euclidean norm to b_0. We assume that $b_1, b_2, \ldots b_n$ are independent vectors with integer coordinates.

$\{b_1, b_2, \ldots b_n\} \leftarrow SHORTEST(n; b_1, b_2, \ldots b_n)$

Return $CLP'(n; b_0, b_1, b_2, \ldots b_n)$
end CLP

Procedure $CLP'(n; b_0, b_1, b_2, \ldots b_n)$

Comment : Does the same as CLP, but assumes that the input basis is reduced in the sense of (2.7) and (2.8).

If $n = 1$ then return the easily computed closest lattice point.
Find i such that $b_i(i) = \max_{j=1}^{n} b_j(j)$
$CANDIDTES \leftarrow \emptyset$
For each "possible" $\lambda_i, \lambda_{i+1}, \ldots \lambda_n$ integers do :

Comment : This is the enumeration step. I will later explain what the word "possible" here means.

If $i = 1$ then $CANDIDATES \leftarrow CANDIDATES \bigcup \{\sum_{j=1}^{n} \lambda_j b_j\}$
else do
$v \leftarrow \sum_{j=i}^{n} \lambda_j b_j$
$v' \leftarrow CLP'(i-1; b_0 - v, b_1, b_2, \ldots b_{i-1})$
$CANDIDATES \leftarrow CANDIDATES \bigcup \{v + v'\}$
end
end
Return the element of $CANDIDATES$ that is closest to b_0.
end CLP'

The following proposition is used to show that the number of "possible" $\lambda_i, \lambda_{i+1}, \ldots \lambda_n$ is small.

Proposition 4.2 : Suppose $L = L(b_1, b_2, \ldots b_n)$ is a lattice in $\mathcal{R}^k$, $k \geq n$ with $b_1, b_2, \ldots b_n$ independent and suppose b_0 is any point in $\mathcal{R}^k$. Let $\overline{b}_0$ be the projection of b_0 onto the span of $\{b_1, b_2, \ldots b_n\}$. Then there exists a point b in L such that

$|b - \overline{b}_0| \leq \frac{1}{2} \left(\sum_{j=1}^{n} (b_j(j))^2\right)^{\frac{1}{2}}$

Further if i is such that $b_i(i) = \max b_j(j)$, then clearly, $|b_0 - b| \leq \frac{\sqrt{n}}{2} b_i(i)$.

Proof : It is not difficult to see that we can successively choose integers $\alpha_n, \alpha_{n-1}, \ldots \alpha_1$ (in that order) such that

$\left|\left((\sum_{l=j}^{n} \alpha_l b_l - \overline{b}_0), b(j,j)\right)\right| \leq b_j(j)/2$

for all j. This is so because the choice of α_j does not affect the inequalities that were earlier ensured. Since $b(1,1), b(2,2), \ldots b(n,n)$ form an orthogonal basis for the vector space they span, and $\overline{b}_0$ by definition lies in that space, the proposition follows.

□

Proposition 4.3 : With the notation set up in the last proposition, there exists an easily determined set $T \subseteq Z^{n-i+1}$ with $|T| \leq n^{(n-i+1)}$ such that if $\sum_{j=1}^{n} \lambda_j b_j$ is the (a) closest point to b_0 in the lattice then, $(\lambda_i, \lambda_{i+1}, \ldots \lambda_n)$ belongs to T.

Proof : Suppose $\sum_{j=1}^{n} \lambda_j b_j = v$ is a closest point in L to b_0. Then clearly, v must be the closest point in L to $\overline{b}_0$. By the last proposition we must have $|v - \overline{b}_0| \leq \frac{\sqrt{n}}{2} b_i(i)$. But,

$|v - \overline{b}_0| \geq \left|\left((v - \overline{b}_0), b(n,n)\right)\right| / b_n(n) = \left|\lambda_n b_n(n) - \left((\overline{b}_0, b(n,n))/b_n(n)\right)\right| = |\lambda_n - t| b_n(n)$

for some fixed real number t. Thus there are at most $\frac{\sqrt{n}}{2}b_i(i)/b_n(n)$ candidates for λ_n. Now, one can show a similar bound for $\lambda_i, \lambda_{i+1}, \ldots \lambda_n$ using an argument similar to proposition 2.13. So suppose $\lambda_{j+1}, \ldots \lambda_n$ are fixed integers, for some $j \geq i+1$. Then arguing as in that proposition there are at most

$\max(2, \quad 2\frac{\sqrt{n}}{2}b_i(i)/b_j(j)) = \sqrt{n}b_i(i)/b_j(j)$

possible values of λ_j such that the length of $v - \bar{b}_0$ in the direction of $b_j(j)$ remains bounded by $\frac{\sqrt{n}}{2}b_i(i)$. Note that I have used the fact that $b_i(i) \geq b_j(j)$. Thus we have to consider a set T of candidates $\lambda_i, \lambda_{i+1} \ldots \lambda_n$ where

$$|T| \leq \prod_{j=i}^{n} \sqrt{n}b_i(i)/b_j(j) \tag{4.4}$$

Since the basis was reduced in the sense of (2.7) and (2.8), $b_i(i)$ is the length of the shortest vector in the lattice $L_i(b_1, b_2, \ldots b_n)$. Further, the denominator of the expression in (4.4) is obviously the determinant of the lattice $L_i(b_1, b_2, \ldots b_n)$. Thus by Minkowski's theorem,

$|T| \leq (\sqrt{n})^{n-i+1}(n-i+1)^{\frac{1}{2}(n-i+1)} = n^{(n-i+1)}$.

□

Theorem (4.5)

The algorithm $CLP(n;\ldots)$ solves the closest vector problem in $O(n^n s)$ arithmetic operations where s is the length of the input. Further all numbers produced by the algorithm are rationals with numerator and denominator expressible in $O(n^2(s + \log n))$ bits each.

Proof : Let $T(n)$ be the number of arithmetic operations performed by $CLP'(n;\ldots)$. Then,

$T(n) \quad \leq n^{(n-i+1)}T(i-1) + q(n)$

where $q(n)$ is a polynomial (Note that this does not depend upon s). Using the fact that the maximum of $(\frac{i-1}{n})^{(i-1)}$ for $1 \leq i \leq n$ is attained at $i = n$ and that the limit of $(\frac{n-1}{n})^{(n-1)}$ is e, we can establish by induction on n that $T(n)$ is $O(n^n)$. The proof is similar to that of theorem (2.17) and I omit the details. So, the number of arithmetic operations performed by $CLP(n;\ldots)$ is $O(n^n)$ plus the number performed by $SHORTEST$. Applying theorem 2.17, we get the current theorem. The bound on the number of bits of all numbers is similar to the proof in section 3.

□

One can also find the L_1 closest and the L_∞ closest vectors. See remark (2.18). The number of candidates will have to be suitably adjusted.

5 Integer Programming

Integer programming again is the following problem:

(5.1) Given $m \times n$ and $m \times 1$ matrices A and b of integers, determine whether there is a x in $\mathbf{Z}^n$ such that $Ax \leq b$.

We will do some "preprocessing" on the problem. First, we will modify the problem so that the set $\{x : Ax \leq b\}$ is bounded, i.e., is a polytope. Second, we ensure that the polytope has positive volume by projecting down to some lower dimensional set if necessary. Then, we will apply an invertible linear transformation to both the polytope and the lattice simultaneously so that the polytope becomes "well-rounded". I will define "well-rounded" more rigorously in (5.2) below. Intuitively, it means that there are two concentric spheres with the smaller one contained in the polytope and the larger one containing the polytope so that the ratio of their radii is bounded above by a function of the dimension alone. Lovász has devised an ingenious polynomial time algorithm to make the polytope "well-rounded". This and the rest of the preprocessing are also part of Lenstra's algorithm. He gives a complete description of this in his paper, so I will say nothing more here except to state precisely the problem at the end of the preprocessing :

(5.2) Given independent vectors $b_1, b_2, \ldots, b_n$ in $\mathbf{Z}^n$, an $m \times n$ integer matrix A and an $m \times 1$ integer matrix b, determine whether there is an x in $L(b_1, b_2, \ldots, b_n)$ such that $Ax \leq b$, where the following additional conditions are satisfied by the input:
$\exists\, p \in \mathcal{R}^n$, r and R reals such that
$R/r \leq 2n^{3/2}$ .....(5.2a)
$B(p,r) \subset \{x \in \mathcal{R}^n, Ax \leq b\} \subset B(p,R)$ .....(5.2b)
where $(B(q,s)$ is the ball of radius s with q as center).

We proceed as follows: We apply $SHORTEST$ to $b_1, \ldots, b_n$. Let now $|b_i(i)| = \max_j |b_j(j)|$. Then there is clearly a point b of $L(b_1, \ldots, b_n)$ (by proposition 4.2) such that $|b - p| \leq \frac{\sqrt{n}}{2}|b_i(i)|$. We consider two cases: (as in Lenstra)

Case 1: $r \geq \frac{\sqrt{n}}{2}|b_i(i)|$. Then the answer to question (5.2) is Yes since the inner sphere itself contains a lattice point. So we can return Yes and stop the algorithm. It is easy to see that in this case, we can in fact find the lattice point.

Case 2: $r < \frac{\sqrt{n}}{2}|b_i(i)|$ whence $R < n^2|b_i(i)|$. In this case, we argue as in the last section that there are not too many integer values of $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ for which there exist integers $\lambda_1, \lambda_2, \ldots \lambda_{i-1}$ so that $\sum_{j=1}^{n} \lambda_j b_j$ belongs to $B(p,R)$. We then enumerate all these values of $\lambda_i, \ldots, \lambda_n$ and for each, solve a $(i-1)$ dimensional problem. So the algorithm is going to be a recursive procedure.

procedure. $ILP(n; A, b)$.

Comment. See description of problem (5.1) above. A is an $m \times n$ matrix of integers and b an $m \times 1$ matrix of integers. The procedure returns Yes or No to the question (5.1)

1.Ensure boundedness of the feasible set in $\mathcal{R}^n$. Then ensure positive volume. Use Lovász's algorithm which applies a suitable linear transformation on the space and ensures conditions (5.2a) and (5.2b). Apply the same linear transformation to the lattice. So now we have independent vectors $b_1, b_2, \ldots b_n$, an $m \times n$ matrix A and an $m \times 1$ matrix b satisfying the conditions of problem (5.2) and we must solve this problem. (Of course, n, m may not be the same as in the original input.)

2. $\{b_1, b_2, \ldots, b_n\} \leftarrow SHORTEST\ \{b_1, b_2, \ldots, b_n\}$

3. Let $b_i(i) = \max_{j=1}^{n} b_j(j)$.

4. if $r \geq \frac{\sqrt{n}}{2}|b_i(i)|$ then return Yes

Comment. We may now assume that $r < \frac{n}{2}|b_i(i)|$ and $R < n^{5/2}|b_i(i)|$.

5. if $i = 1$ then do.

6.for each candidate $\lambda_1, \lambda_2, \ldots, \lambda_n$ integers do.

Comment. Enumeration is explained later.

7.if $\sum_{j=1}^{n} \lambda_j b_j = x$ satisfies $Ax \leq b$ then return Yes.

end

8.Return No

end

9.for each candidate $\{\lambda_i, \lambda_{i+1}, \ldots, \lambda_n\} \in Z^{n-i+1}$ do:

Comment. We explain later what the candidates are.

10. $b_0 \leftarrow \sum_{j=i}^{n} \lambda_j b_j$

Now, a candidate $\{\lambda_i, \lambda_{i+1}, \ldots, \lambda_n\} \in Z^{n-i+1}$ is fixed. We want to determine whether there is a point z in $L(b_1, b_2, \ldots b_{i-1})$ such that $z + b_0$ satisfies $Ax \leq b$, equivalently z satisfies $Az \leq b - Ab_0$. Letting $z = \sum_{j=1}^{i-1} \alpha_j b_j$, and B to be the $n \times (i-1)$ matrix with $b_1, b_2, \ldots b_{i-1}$ as its columns, we want $AB\alpha \leq (b - Ab_0)$ where α is required to be a $i-1$ vector of integers.

11. if $ILP(i-1; AB, b - Ab_0)$ returns yes then return yes.

end

Return No

end ILP

As usual, we first explain the enumeration process. At the beginning of Step 5, we may assume that R is less than $b_i(i)n^2$. Thus any vector a, in $L(b_1, \ldots, b_n)$ which could belong to the polytope $\{x : Ax \leq b\}$ must have the property that $|a - p| \leq b_i(i)n^2$. Hence the projection of $a - p$ in the direction of $b(n, n)$ must be less than $n^2 b_i(i)$. Thus, we need to try at most

$$2n^2 \frac{b_i(i)}{b_n(n)} \text{ values of } \lambda_n$$

Arguing in a similar vein to Proposition 2.13 and Proposition 4.3, the number of candidates for $\lambda_i, \lambda_{i+1}, \ldots, \lambda_n$ is at most

$$\left|2^{n-i+1}n^{2(n-i+1)}\prod_{j=i}^{n}|b_i(i)|/|b_j(j)|\right| \tag{5.3}$$

$b_i(i)$ equals $\Lambda_1(L_i(b_1, b_2, \ldots b_n)$ since we applied $SHORTEST$. The denominator of (5.3) is, of course, $d(L_i(b_1, b_2, \ldots b_n))$. Thus using Minkowski's theorem, the whole expression is bounded by $n^{\frac{5}{2}(n-i+1)}$.

The lengthy comments already argue the correctness of the algorithm. In what follows , I prove bounds on the number of arithmetic operations and the bits needed to represent intermediate numbers. The latter bound is not a polynomial, which is undesirable. Recently, Frank and Tardos (1985) have used an ingenious method of approximating linear equations to make the number of bits in this algorithm polynomially bounded. The interested reader is referred to their paper.

Theorem (5.4)

The algorithm $ILP(n; \ldots)$ on an input of length s, correctly solves the n-variable integer programming problem in $0(n^{\frac{5}{2}n}s)$ arithmetic operations. Each integer produced by the algorithm is $0(n^{2n}s)$ bits in size.

Proof The second part is proved first. There are two steps that dominate the production of large numbers - the "rounding out" step (step 1) and the execution of $SHORTEST$. In what follows, I will restrict attention to these steps, leaving it to the reader to check the other ones. The ILP algorithm takes various sections of the polytope and works on each of these sections. Since "taking a section" reduces the dimension by 1, there can be at most n nestings of the "rounding out" and $SHORTEST$ steps. I will argue that one pair of executions of these two steps does not increase the size of numbers by more than a factor of $O(n^2)$, thus yielding an overall factor of at most $O(n^{2n})$.

By going through the construction to "round out" a polytope due to Lovász , one finds that this increases the number of bits by at most a factor of n^2. This is because the algorithm obtains the affine transformation that rounds out the polytope $\{x : Ax \leq b\}$ by mapping $(n+1)$ of its vertices (in n dimensions) to $(0, 0, 0, \ldots, 0), (1, 0, \ldots 0), (0, 1, \ldots, 0) \ldots$
Let S be the $n \times n$ matrix whose i th row equals the $i+1$ st of these $n+1$ vertices minus the first. Then the linear transformation corresponding to the affine transformation has as its matrix S^{-1}. The number of bits of S^{-1} is at most $O(n^2)$ times the number of bits needed to define the polytope; we loose at most a factor of $O(n)$ to get S - see for example Gács and Lovász(1979) and a factor of $O(n)$ for the inverse . Thus the transformation does not increase the number of bits by more than a factor of $O(n^2)$. The algorithm $SHORTEST$ then increases the sizes by at most a factor of $0(n^2)$ by theorem 3.9. (There is an *additive* $n^2 \log n$ term in the theorem, but this is subsumed by the other terms.) But when the algorithm $SHORTEST$ is finished, the sizes are much smaller by propositions 3.1 and 3.2, in fact, they are at most $O(\log n)$ *plus* what it used to be at the start of $SHORTEST$. Thus I have proved what I promised at the end of the last paragraph.

For the number of arithmetic operations, we use the recursion
$T(n, s) \leq n^{\frac{5}{2}(n-i+1)}T(i-1, n^2s) + cn^n s$

By induction, we can now show that $T(n,s)$ is $O(n^{\frac{5}{2}n}s)$.

□

If we use the algorithm of Frank and Tardos (1985) and keep numbers polynomially bounded in size, the number of arithmetic operations will be reduced to $O(n^{\frac{5}{2}n}s)$.

I will now briefly discuss the structural result underlying the algorithm. Consider the subspace V spanned by the vectors $b_1, b_2, \ldots b_{i-1}$ where i is defined in step 3 of the procedure ILP. Every lattice point of $L(b_1, b_2, \ldots b_n)$ belongs to a translate of V of the form $V+z$ where z is in $L(b_i, b_{i+1}, \ldots b_n)$. What I have shown is that at most $n^{\frac{5}{2}(n-i+1)}$ such translates interest the polytope if the polytope contains no lattice points. The proof applies to only "well-rounded " polytopes. But, it can be easily extended to all polytopes with positive volume : Suppose P is any polytope with positive volume (i.e., is full-dimensional). Then Lovász's algorithm finds a linear transformation τ so that τP is well-rounded. Then clearly, the number of translates of V intersecting τP equals the number of translates of $(\tau^{-1})V$ intersecting P.

If we only want an existential result and are not interested in finding the subspace V, we can do better than $\frac{5}{2}$ in the exponent. The argument is as follows : A result of John (1948) says that for any convex body K in $\mathcal{R}^n$ (the word "body" is used to denote a set of positive volume) there are two similar ellipsoids E_1, E_2 such that $E_1 \subseteq K \subseteq E_2$ and E_2 is obtained by dilating E_1 about its center by a factor of n. Suppose τ is the invertible linear transformation that sends E_1 into a sphere of radius 1 (and hence E_2 into a sphere of radius n). Suppose also that $K \cap Z^n$ is empty. Let $b_1, b_2, \ldots b_n$ be a reduced basis of the lattice $L = \tau Z^n$ (in the sense of (2.6)). Let $b_i(i)$ be the maximum of the $b_j(j)$'s and let V be the space spanned by $b_1, b_2, \ldots b_{i-1}$. Since $E_1 \cap Z^n$ is empty, we must have that $\tau E_1 \cap L$ is empty and hence $\frac{\sqrt{n}}{2} b_i(i) > 1$ by proposition 4.2. This and the fact that τE_2 has radius n can be used to show that the number of $\lambda_i, \lambda_{i+1} \ldots \lambda_n$ for which there exists $\lambda_1, \lambda_2, \ldots \lambda_{i-1}$ (all integers) so that $\sum_{j=1}^{n} \lambda_j b_j$ belongs to τE_2 is at most $n^{\frac{3}{2}(n-i+1)} \prod_{j=i}^{n} \frac{b_i(i)}{b_j(j)}$ and by using Minkowski's theorem (1.12) we get that this quantity is at most $n^{2(n-i+1)}$. This bounds the number of translates of V intersecting the τK and hence the number of translates of $\tau^{-1}V$ intersecting K. The case when i was equal to 1 was the "best" case for the algorithm, because, then the problem is solved by simple enumeration, no recursive calls were needed in this case. However, for the existential result, it is not interesting because then $V = \{0\}$ and the relevant translates of V are just the singleton sets consisting of lattice points. Since $K \cap Z^n = \emptyset$, we already know that *none* of these translates intersects K. But, going back to the enumeration argument, since $b_i(i)/b_i(i) = 1$, we could argue exactly the same bound on the number of candidates of $\lambda_{i+1}, \lambda_{i+2} \ldots \lambda_n$. This ensures that the subspace V is always of dimension at least 1. I have proved (albeit sketchily) the following theorem.

Theorem (5.5)

Suppose K is any bounded convex body in $\mathcal{R}^n$ with $K \cap Z^n = \emptyset$. Then there is a $i, 1 \leq i \leq n$ and an i dimensional space V which has a basis of integer vectors such that the number of translates of V containing lattice points that intersect K is at most $n^{2(n-i+1)}$.

The result of H.W.Lenstra's mentioned in the abstract can be restated as : If K is any bounded convex body in $\mathcal{R}^n$ with $K \cap Z^n = \emptyset$, then there is a $n-1$ dimensional

subspace V, spanned by integer vectors such that the number of translates of V containing integer points that intersect K is at most c^{n^2}. The bound was improved to c^n by Babai (1985). Based on the results of Lenstra and Schnorr (1984), Hastad (1985) improved it to a polynomial - $C(n^{5/2})$. Grötschel, Lovász and Schrijver (1982) have extended this to unbounded convex bodies. Cook, Collurd and Turan (1985) use this to derive bounds on the number of cutting planes needed to prove the infeasibility of integer programs. By not restricting only to $n-1$ dimensional subsapces, theorem (5.5) is able to get a 2 in the exponent. It is likely that both Hastad's result and theorem (5.5) can be improved giving us further improvement in the running time of the integer programming algorithm.

6 Complexity Issues.

It was conjectured in Lenstra (1981) that the problem of finding a shortest vector in a lattice $L = L(b_1, \ldots, b_n)$ given $b_1, \ldots, b_n$ is NP-hard.

The conjecture is still open. Van Emde Boas has proved the language $L_2 - CLOSEST$ defined below (which is the natural language corresponding to the Closest Vector Problem) to be NP-complete. Van Emde Boas's proof is complicated and technical. It is also not published. So I will give here a more natural NP-completeness proof of this language. The reduction will be from 3-dimensional matching (3DM) described below which is known to be NP-complete. (Karp 1972)

(6.1) Given a set $T \subseteq \{1, 2, \ldots n\}^3$, determine whether there is a subset M of T such that for each $i \in \{1, 2, \ldots n\}$, M has precisely one 3-tuple containing i in the first coordinate, precisely one 3-tuple containing i in the second coordinate and precisely one 3-tuple containing it in the third coordinate. (These 3-tuples do not have to be distinct.)

Theorem (6.2)

The language $L_2 - CLOSEST = \{(b_0, b_1, \ldots, b_n; K) | \exists b \in L(b_1, \ldots, b_n)$ *such that* $|b - b_0| \leq K\}$ *is NP-complete.*

Proof We can easily reduce the 3DM problem to an integer program as follows : We set up one variable x_t for each 3-tuple t in T. This variable will be forced to take on only the values 0 or 1. The interpretation is that $x_t = 1$ iff t is included in M. Then the 3DM problem is equivalent to the following problem. I leave the proof of this to the reader.

(6.3) Does there exist a feasible solution to the following integer program :

$$\sum_{\{(j,k):(i,j,k)\in T\}} x_{(i,j,k)} = 1 \text{ for } i = 1, 2, \ldots n \qquad (6.3a)$$

$$\sum_{\{(j,k):(j,i,k)\in T\}} x_{(j,i,k)} = 1 \text{ for } i = 1, 2, \ldots n \qquad (6.3b)$$

$$\sum_{\{(j,k):(j,k,i)\in T\}} x_{(j,k,i)} = 1 \text{ for } i = 1, 2, \ldots n \tag{6.3c}$$

$$x_t \in \{0,1\} \text{ for all } t \in T \tag{6.3d}$$

Proposition (6.4) : In the above integer program, (6.3d) can be replaced by the following conditions :

$$\sum_{t\in T} x_t^2 \leq n; \tag{6.5a}$$

$$x_t \in Z \forall t \in T \tag{6.5b}$$

Proof : Suppose x (considered as a vector with $|T|$ components) is a solution to (6.3a),(6.3b),(6.3c) and (6.3d). If x has less than n nonzero components (which are each of course one), then one of the equations (6.3a) will be violated because (6.3a) comprises of n different equations in disjoint sets of variables; also if x has more than n components with value 1, one of the left hand sides in (6.3a) will be at least 2. Thus x must have precisely n 1's and so it satisfies (6.5). Conversely, suppose x satisfies (6.3a),(6.3b),(6.3c) and (6.5). To satisfy (6.3a) for example, x must have at least n nonzero components . Each of the nonzero components is of course an integer, so to satisfy the inequality in (6.5), there must be precisely n nonzero components in x and each of these must be ± 1. But if even one of them is -1, there is no way to satisfy (6.3a) say. So they must all be $+1$ and we have proved the proposition.

With the proposition, I have shown the following problem to be NP-complete : (by reducing 3DM to it)

(6.6) Given $m \times n$ and $m \times 1$ matrices of integers A and b respectively and an integer K, determine whether there is a $n-$ vector x satisfying :

$$Ax = b \quad x \in Z^n \quad |x| \leq \sqrt{K} \tag{6.7}$$

where $|x|$ is the Euclidean length. I will now show that this problem is polynomial time many-one reducible to the Closest Vector Problem (CVP). By using the Hermite Normal form algorithm of Kannan and Bachem (1979), one can find the general integer solution of a system of linear equations in polynomial time. We use this to obtain $b_0, b_1, \ldots b_r$ belonging to Z^n so that $Ab_0 = b$ and $L(b_1, b_2, \ldots b_r) = \{x : x \in Z^n; Ax = 0\}$ whence we have $\{x : x \in Z^n, Ax = b\} = b_0 + L(b_1, b_2, \ldots b_n)$. Thus to solve the problem (6.7),, it suffices to find whether there is an element of $L(b_1, b_2, \ldots b_r)$ within distance $\sqrt{K}$ of $-b_0$. This then completes the proof of theorem (6.2)

□

The inequality (6.5a) may be replaced by $\sum |x_t| \leq n$. This proves that the corresponding language for the L_1 norm is also NP-complete. A similar proof works for the L_∞ norm too.

Now, let us turn our attention to the Shortest Vector Problem (SVP). First, it is convenient to define a language corresponding to the SVP. I will call this language $L_2 SHORTEST$

.

$$L_2 - SHORTEST = \{(b_1, \ldots, b_n, ; K) | \exists b \in L(b_1, \ldots, b_n) \qquad b \neq 0 \text{ such that } |b| \leq K\}$$

Theorem (6.8)

Given $b_0, b_1, \ldots, b_n$ in $\mathcal{R}^n$, $n \geq 2$ with polynomially many calls to a subroutine for deciding membership in $L_2 - SHORTEST$ and polynomial additional time we can find a vector y in $L(b_1, \ldots, b_n)$ such that for all y' in $L(b_1, \ldots, b_n)$,

$$|y - b_0| \leq \sqrt{n/2}.|y' - b_0|$$

Remark. The theorem asserts that the problem of finding an approximate closest vector to within a factor of $\sqrt{n/2}$ is polynomial-time Turing(Cook) reducible to $L_2 - SHORTEST$. The reduction given is essentially a Cook reduction - it invokes more than one call to the subroutine.

We show first that given a subroutine that accepts L_2-shortest, we can actually find a shortest vector in a lattice. Suppose $L = L(b_1, \ldots, b_n)$, $b_i \in Z^n$ independent is the lattice in which we want to find a shortest nonzero vector. Define

$$\ell = \left| (\sqrt{n}\,(d(L))^{\frac{1}{n}})^4 \right| \tag{6.9}$$

Let τ be the linear transformation given by the $n \times n$ diagonal matrix containing entries $\ell^{3n} + \ell^{n+1-i}$ in the (i,i) th position for $i = 1, 2, \ldots n$. . . . . (6.10)

(τ multiplies the i th coordinate by $(\ell^{n+1-i} + \ell^{3n})$.

Lemma (6.11)

Suppose $L = L(b_1, \ldots, b_n)$ where $b_i \in Z^n$ and are independent and define ℓ and τ as in (6.9) and (6.10). Then for $L^* = \tau L$, any shortest nonzero vector of L^* must be of the form

$$Y = ((\ell^{3n} + \ell^n)y_1, (\ell^{3n} + \ell^{(n-1)})y_2, \ldots, (\ell^{3n} + \ell)y_n) \quad \ldots \ldots \ldots \ldots \quad (6.12)$$

where $(y_1, y_2, \ldots, y_n)$ is a shortest vector L and for any other shortest vector $(y'_1, y'_2, \ldots, y'_n)$ of L, we have

$$|y_{i_0}| < |y'_{i_0}| \text{ for } i_0 = \min_{i=1}^{n} \{i : |y_i| \neq |y'_i|\} \tag{6.13}$$

(In other words $(|y_1|, \ldots, |y_n|)$ is the lexicographically least among the shortest vectors of L).

Proof : Clearly, $\Lambda_1(L^*) \leq (\ell^{3n} + \ell^n)\,\Lambda_1(L)$. . . . . (6.14)

Suppose now Y is a shortest vector in L^* and the corresponding $(y_1, y_2, \ldots, y_n)$ (according to (6.12)) is not a shortest vector of L. Then noting that the y_j are all integers,

$$|Y|^2 \geq \ell^{6n}(|(y_1, y_2, \ldots, y_n)|^2) \geq \ell^{6n}(\Lambda_1(L)^2 + 1)$$

$$= \ell^{6n}\Lambda_1^2(L) + \ell^{6n} > (\ell^{3n} + \ell^n)^2\Lambda_1(L)^2$$

(from (6.9) and Minkowski (1.12))

This contradicts (6.14) and hence $y = (y_1, y_2, \ldots, y_n)$ must be a shortest nonzero vector of L. Further, $|Y|^2 = (\ell^{3n} + \ell^n)^2 y_1^2 + \ldots + (\ell^{3n} + \ell)^2 y_n^2$. Suppose $y' = (y_1', \ldots, y_n')$ is another shortest vector of L and (6.13) is violated. Then $|y_{i_0}| \geq |y_{i_0}'| + 1$. It can be seen easily that this together with the fact that $|y_j'| \leq \ell$ for all j (by the definition of ℓ in (6.9) and Minkowski) implies that $Y' = \tau y'$ is shorter than Y in $L^* - a$ contradiction. Thus $(|y_1|, \ldots, |y_n|)$ must be lexicographically least among all the shortest vectors of L.

□

From (6.12) and the fact that a shortest nonzero vector $y = (y_1, \ldots, y_n)$ of L must satisfy $|y_j| \leq \ell^{1/4}$ for all j, we see easily that if $|Y|^2$ is given, then $(|y_1|, |y_2|, \ldots, |y_n|)$ can be determined: Expand the integer $|Y|^2$ to the base ℓ to write

$$|Y|^2 = \sum_{j=0}^{6n} \alpha_j \ell^j;$$

then $y_1^2 = \alpha_{2n}, y_2^2 = \alpha_{2(n-1)}, \ldots, y_n^2 = \alpha_2$. Now further, given a subroutine for $L_2 SHORTEST$, we can find $|Y|^2$ using binary search in polynomial time. Thus we can find $(|y_1|, |y_2|, \ldots, |y_n|)$ using the subroutine and polynomial additional time . Using this, of course, we could also find $|Y_1|, \ldots |Y_n|$.

We still need the signs of the components of y. Towards this end, first note that L^* has the property that if Y is a shortest vector of L^*, then for any other shortest vector Y' of $L^*, |Y_i| = |Y_i'|$ (by (6.13)). Let $(|Y_1|, |Y_2|, \ldots, |Y_n|)$ be the magnitudes of the coordinates of a shortest vector Y of L^* already found as described above. Consider the $(n-1)$ dimensional lattice:

$$L' = L^* \cap \{x : \ x_1|Y_2| - x_2|Y_1| \ = \ 0\}$$

Clearly, $\Lambda_1(L') = \Lambda_1(L^*)$ iff there is a shortest vector of L^* with the first two coordinates positive. Let $L'' = L^* \cap \{x : \ x_1|Y_2| + x_2|Y_1| = 0\}$. Then $\Lambda_1(L'') = \Lambda_1(L^*)$ iff there is a shortest vector of L^* with the first two coordinates of opposite signs. So, we do the following: using our subroutine for L_2-Shortest, we check if $\Lambda_1(L') = \Lambda_1(L^*)$. If so we find (recursively) a shortest vector in L' and hence figure out a shortest vector of L^*, then of L. If not, we find (recursively) a shortest vector of L'' and do like-wise. Note that to solve the problem of finding a shortest vector in n-dimensions, we solve one instance of the corresponding $(n-1)$ dimensional problem plus polynomially many calls to L_2-shortest.

Lemma (6.15) *With polynomially many calls to a subroutine accepting the language $L_2 - SHORTEST$ and polynomial additional time, we can find a shortest nonzero vector in a lattice.*

Remark: A lemma similar to the one above holds for most known NP-complete languages and several other ones-like linear programming. For example, it is easy to see by using self-reducibility that given an algorithm to test whether a given Boolean formula is satisfiable, we may use it to find a satisfying assignment. This speaks for the versatility of the language SAT. (the set of satisfiable Boolean formulas). It is interesting that the language $L_2 - SHORTEST$ not yet known to be NP-complete has this versatility.

We now study the relationship between the problem of finding a closest vector of a lattice in $\mathcal{R}^n$, to a given point in $\mathcal{R}^n$ (called the "inhomogeneous problem") to that of finding a shortest nonzero vector of a lattice (called the "homogeneous problem"). The device we use to relate these two may be called the process of "homogenization". The technique is used in polyhedral theory. The idea is to relate the inhomogeneous problem for a lattice L in n dimensions to a homogeneous problem for a lattice L' constructed from L in $(n+1)$ dimensions.

Suppose we are given $b_1, b_2, \ldots, b_n, b_0$ in Z^n and are asked to find a point b of $L = L(b_1, \ldots, b_n)$ which is approximately (to be defined later) closest (in Euclidean distance) point of L to b_0. We first check whether b_0 is in L by using a polynomial-time algorithm to solve linear diophantine equations. If so, we may stop. Otherwise we find (using the subroutines for the homogeneous problem) $\Lambda_1(L)$ (the length of a shortest nonzero vector of L: Caution: this may be irrational, so we will only find an approximation to it in the actual algorithm, but to simplify the current discussion, assume we know $\Lambda_1(L)$ exactly). We then consider the lattice L' in $\mathcal{Q}^{n+1}$ generated by $b'_i = (b_i, 0)$ for $i = 1, 2, \ldots, n$ and $b'_{n+1} = (b_0, (.51)|\Lambda_1(L)|)$. We find a shortest nonzero vector $v = (v_1, \ldots, v_{n+1})$ of L' (Lemma 6.15). This gives us information about the vector closest to b_0 in L as summarized by the following lemma:

Lemma (6.16)

Suppose $L = L(b_1, \ldots, b_n)$ is a lattice in Z^n and b_0 in $\mathbf{Z}^n$ is not in L. Let L' be as defined in the last paragraph and let $v = (v_1, \ldots, v_{n+1})$ be a shortest nonzero vector of L' with $v_{n+1} \leq 0$. If $v_{n+1} = 0$, then, $|b_0 - b| \geq .8|\Lambda_1(L)|$ for all b in L. If $v_{n+1} \neq 0$ then $v_{n+1} = -(.51)|\Lambda_1(L)|$ and $(v_1, v_2, \ldots, v_n) + b_0$ is the closest vector in L to b_0.

Proof The shortest vector $v = (v_1, v_2, \ldots, v_{n+1})$ of L' must clearly satisfy $|v_{n+1}| \leq |\Lambda_1(L)|$ because there is a vector of length $|\Lambda_1(L)|$ in L and hence in L'. Thus $v_{n+1} = 0$ or $\pm .51|\Lambda_1(L)|$. Without loss of generality, we assumed $v_{n+1} \leq 0$ and hence is 0 or $-(.51)\ |\Lambda_1(L)|$. Let b be a closest point of L to b_0 and $b = \sum_{j=1}^{n} \alpha_j b_j$. If $|b - b_0| < .8|\Lambda_1(L)|$, then

$$|\sum_{j=1}^{n} \alpha_j b'_j - b'_{n+1}| \leq |\Lambda_1(L)|\ ((.8)^2 + (.51)^2)^{\frac{1}{2}} < |\Lambda_1(L)|$$

and hence the shortest vector v of L' must have $v_{n+1} = -(.51)\Lambda_1(L)$. This proves the first statement.

To prove the second statement, assume that $v_{n+1} = -.51|\Lambda_1(L)|$. Then v equals $(-b'_{n+1} + \sum_{j=1}^{n} \beta_j b'_j)$ for some integers β_j and since the last component of v is fixed at

absolute value .51 ($|\Lambda_1(L)|$, v will be shortest when $\sum_1^n \beta_j b_j$ is closest to b_0. This proves the lemma.

□

The lemma leads to the following recursive algorithm for approximating the closest vector. The recursion will be on the dimension of the lattice. The factor of approximation will be $\sqrt{n/2}$ as asserted in theorem 6.8. For $n = 2$, the algorithm is obvious. So assume we are given a lattice L of dimension $n > 2$ and a point b_0. First, we find the shortest vector v in the lattice L' used in the lemma. If $v_{n+1} \neq 0$, then we have already found the closest vector and we may stop. In the other case, the distance of b_0 to L (henceforth denoted $d(b_0, L)$) is at least $.8\Lambda_1(L)$. We obtain a basis $b_1, b_2 \ldots b_n$ of L with b_1 as a shortest vector using the subroutine for $L_2 - SHORTEST$ (cf. Lemma 6.15 and the procedure $SELECT - BASIS$ of section 2). In the rest of this proof, we let the superscript $\hat{}$ denote the projection perpendicular to b_1. Recursively we find an element $\hat{b} \in \hat{L}$ so that $|\hat{b} - \hat{b}_0| \leq \sqrt{\frac{n-1}{2}} d(\hat{b}_0, \hat{L})$. Now, find b in L so that b projects to $\hat{b}$ and $b - b_0$ has a projection along the direction of b_1 of length at most $|b_1|/2$. Then,

$$|b - b_0|^2 \leq |b_1|^2/4 + \frac{(n-1)}{2}(d(\hat{b}_0, \hat{L}))^2 \tag{6.17}$$

We know that $|b_1|^2/4 \leq (4 \times .64)^{-1} d(b_0, L)^2 \leq \frac{1}{2}(d(b_0, L))^2$. Further, we must of course have $d(\hat{b}_0, \hat{L}) \leq d(b_0, L)$. Using these two inequalities in (6.17), we get $|b - b_0|^2 \leq \frac{n}{2}(d(b_0, L))^2$ proving theorem (6.8) .

7 Remarks

The most important open problem in the area is of course the complexity of the shortest vector problem which has been discussed in the body of the paper. It is conjectured that this problem is NP-hard at least under Cook (Turing) reductions. One approach to proving this is to prove that the approximate version of the Closest vector problem is NP-hard. Approximate versions of Integer Programming, Traveling Salesman problem etc. are known to be NP-hard. The difficulty with the CVP is that it is asking for an integer point within a sphere - a very special object. This also raises another interesting question - in proving NP-completeness of the CVP in section 4, I reduced the 3-dimensional matching problem to it. Suppose now, we wish to reduce Integer Programming to the CVP. If the IP has n variables and has a total description of length s (the number of bits), then the reduction to the 3DM in general will lead to a problem where the number of variables will depend on n *as well as* s polynomially. The question is : Can we reduce integer programming in polynomial time to CVP so that the number of dimensions of the CVP is a small function of n, the number of variables of the IP ? Geometrically, can the question of whether a polytope in $\mathcal{R}^n$ has an integer point be reduced in polynomial time to questions of whether certain spheres in $\mathcal{R}^m$ have integer points for some m close to n. It seems possible that we can achieve a polynomial bound on m in terms of n alone. For the reasons stated earlier in the paragraph, the answer to this question should shed some light

on the NP-hardness of the SVP.

Another interesting open problem is to devise polynomial time algorithms that come within a subexponential factor of the shortest vector. In this connection, it is also interesting to consider lattices over other rings than the integers. Lattices over $GF(2)$ which are of course just vector spaces are of particular interest in coding theory and cryptography, so, I state the "Shortest Vector Problem " for such lattices below : The *length* of a vector with 0,1 components is defined to be the number of 1's in it for this discussion. This is also called the "Hamming length". The question is : Given n 0,1 vectors $b_1, b_2, \ldots, b_n$ find the (Hamming) shortest nonzero linear combination of them where all operations are done modulo 2. We can also define an analogous "closest vector problem" for these . The CVP is easily shown to be NP-hard (Berlekamp, McElicee and van Tilborg (1978)) , however the complexity of the SVP is still open. The CVP is equivalent to the question of finding the shortest circuit containing a particular edge in a binary matroid (Tutte (1959)). In very special cases when the binary matroid is graphic, the problem is the shortest path problem for graphs, which is , of course, polynomial time solvable. A complicated and clever argument of Seymour's (1980) gives a polynomial time algorithm for a broader class of binary matroids. The SVP is equivalent to the problem of finding the shortest circuit in a binary matroid. It is trivial to solve the SVP in 2^n steps where n is the dimension of the lattice. A slightly better algorithm is possible when we wish to determine whether there is a nonzero vector in the lattice of (Hamming) length at most k where k is small compared to n. In this case, we can do with $\binom{n}{k}$ steps as follows : we do Gaussian elimination on the basis vectors (since we are in a field) to ensure that there are n distinct components $i_1, i_2, \ldots i_n$ such that the j th basis vector in the new basis is 1 in the i_j th position and zero in the other $n-1$ positions of the set $\{i_1, i_2, \ldots i_n\}$. Then it is clear that any vector in the lattice of length at most k must be the mod 2 sum of at most k of the new basis vectors. Obviously, this does better than the naive algorithm when $k \leq n/2$. This case is of interest in certain situations in cryptography. However, to my knowledge, no subexponential algorithm is known for the problem in general. It is not clear prima facie that any of the techniques for integer lattices will carry over to these lattices.

One of the essential ideas for all the three algorithms in this paper is the argument bounding the number of candidates for the enumeration. It seems possible that this argument will be of more general use. There is a context other than those in this paper where it has been shown to be useful. (Furst and Kannan 1985). I mention this briefly : Suppose we are given a basis $b_1, b_2, \ldots b_n$ of a lattice with $\mathrm{Min}_{i=1}^{n} b_i(i) = t$. Then for any vector v, we can determine in polynomial time whether there is a point u in the lattice such that $|u-v| < t/2$. To see this, let $u = \sum \lambda_i b_i$ satisfy $|u-v| < t/2$. It is not difficult to see that there is at most one candidate for λ_n, since $b_n(n) \geq t$. Similarly, if $\lambda_n . \lambda_{n-1}, \ldots \lambda_{i+1}$ are fixed, there is at most one candidate for λ_i. This helps us determine quickly whether or not there is such a u. This is one of the ideas used by Furst and me to develop a proof system that yields polynomial length proofs of the *infeasibility* of subset sum problems in almost all instances.

Acknowledgment: I wish to thank Alan Frieze, Merrick Furst, Bettina Helfrich, Gary Miller, and Claus Schnorr for helpful discussions and pointing out errors in earlier drafts.

8 References

L.Babai, *On Lovśaz's lattice reduction and the nearest lattice point problem* , to appear in Combinatorica (1985)

E.R.Berlekamp, R.J.McElicee and H.C.van Tilborg, *On the inherent intractability of certain coding problems* IEEE Transactions on Information Theory Vol. 24 , May (1978) pp 384-386

H.F.Blichfeldt, *The minimum value of quadratic forms and the closest packing of spheres* Math. Annalen 101 pp 605-608 (1929)

J.W.S.Cassels, *An introduction to the geometry of numbers* Springer Verlag (1971)

S.A.Cook, *The complexity of theorem proving procedures* Proc. 3rd Ann. ACM Symposium on Theory of Computing, Assoc. Comput. Mach., New York (1971) pp 151-158

W.Cook, C.Cuilard and Gy.Turan, *Complexity of cutting planes* , Technical report , Institut für Operations Research , University of Bonn (1985).

J.Edmonds, *Systems of distinct representatives and Linear Algebra*, J. Res. Nat. Bur. Standards, Sect. B 71B (1967) pp 241-245

A.Frank and E.Tardos, *An application of simultaneous approximation in combinatorial optimization* , Report Institut für Ökonometrie und Operations Research , Uni. Bonn, W.Germany (1985) to appear in Combinatorica.

M.L.Furst and R.Kannan, *Proofs of infeasibility for almost all subset sum problems* , In preparation (1985)

P.Gács and L.Lovász, *Khachian's algorithm for linear programming*, Mathematical Programming Studies (1979).

M.Grötschel, L.Lovász and A.Schrijver, *Geometric methods in combinatorial optimization* , in : Progress in Combinatorial Optimization (W.R.Pulleyblank, ed.), Proc. Silver Jubilee Conference on Comb. Opt. , Univ. of Waterloo, Vol. 1, (1982), Academic Press , N.Y. (1984)

J.Hasted Private Communication (1985)

B.Helfrich, *Algorithms to construct Minkowski reduced Hermite reduced lattice bases*, Uni. Frankfurt Technical report, to appear in Theoretical Computer Science. (1985)

D.Hirschberg and C.K.Wong, *A polynomial-time algorithm for the knapsack problem with two variables*, J. Assoc. Comput. Mach. 23, (1976) pp 147-154

F.John, *Extremum problems with inequalities as subsidiary conditions*, Studies and Essays presented to R.Courant (1948)

R.Kannan, *A polynomial algorithm for the two variable integer programming problem* , J. Assoc. Comput. Mach. 27, (1980) pp 118-122

R.Kannan, *Lattices, basis reduction and the shortest vector problem*, Coll. Math. Soc. J.Bolyai , 44, Theory of Algorithms , Pécs (Hungary) (1984)

R.Kannan, *Improved algorithms for integer programming and related lattice problems* 15 th Annual ACM symposium on theory of computing (1983) pp193-206

R.Kannan and A.Bachem, *Polynomial time algorithms for computing the Smith and Hermite normal forms of an integer matrix*, SIAM Journal on Computing, 8 (1979) pp 499-507

R.M.Karp, *Reducibility among combinatorial problems* in R.E.Miller and J.W.Thatcher (eds.) Complexity of Computer Computations , Plenum Press, New York pp 85-103 (1972)

A.Korkine and G.Zolotareff, *Sur les formes quadratiques*, Math. Annalen 6, (1873) pp 366-389

C.G.Lekkerkerker, *Geometry of Numbers* North Holland , Amsterdam, (1969)

A.K.Lenstra, *Lattices and factorization of polynomials* , Report IW 190/81, Mathematisch Centrum, Amsterdam (1981)

A.K.Lenstra, H.W.Lenstra and L.Lovász, *Factoring polynomials with rational coefficients* Mathematische Annalen 261 (1982), pp513-534

H.W.Lenstra, *Integer programming with a fixed number of variables* First announcement (1979) Mathematics of Operations research, Volume 8, Number 4 Nov (1983) pp 538-548

H.W.Lenstra and C.P.Schnorr, *On the successive minima of a pair of polar lattices* Technical report, Uni. Frankfurt, Oct (1984)

H.Minkowski, *Geometrie der Zahlen* Leipzig, Tuebner (1910)

H.E.Scarf, *Production sets with indivisibilities* Part I : Generalities, Econometrica 49 pp1-32 , Part II :The case of two activities , ibid, pp 395-423 (1981)

C.P.Schnorr, *A hierarchy of polynomial time basis reduction algorithms* in : Coll. Math. Soc. Janos Bolyai, 44 , Theory of Algorithms Pécs (hungary) (1984)

P.D.Seymour, *Decomposition of regular matroids* Journal of Combinatorial Theory (B) 28 (1980) pp 305-359

G.Strang *Linear algebra and its applications* Academic press (1980)

W.T.Tutte, *Matroids and graphs* Trans. Amer. Math. Soc., 90 (1959) pp 527-552

P.van Emde Boas, *Another NP-complete problem and the complexity of computing short vectors in a lattice* Report 81-04, Mathematische Instituut, Uni. Amsterdam (1981)

J. von zur Gathen and M.Sieveking, *Weitere zum Erfullungsprobleme polynomial äquivalente kombinatorische Aufgaben* in : Lecture Notes in Computer Science 43 (Springer , Berlin 1976)