Solving Low-Density Subset Sum Problems

J. C. LAGARIAS AND A. M. ODLYZKO

AT&T Bell Laboratories, Murray Hill, New Jersey

Abstract. The subset sum problem is to decide whether or not the 0–1 integer programming problem

$$\sum_{i=1}^{n} a_i x_i = M, \qquad \forall i,\ x_i = 0 \quad \text{or} \quad 1,$$

has a solution, where the a_i and M are given positive integers. This problem is NP-complete, and the difficulty of solving it is the basis of public-key cryptosystems of knapsack type. An algorithm is proposed that searches for a solution when given an instance of the subset sum problem. This algorithm always halts in polynomial time but does not always find a solution when one exists. It converts the problem to one of finding a particular short vector $\mathbf{v}$ in a lattice, and then uses a lattice basis reduction algorithm due to A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász to attempt to find $\mathbf{v}$. The performance of the proposed algorithm is analyzed. Let the *density* d of a subset sum problem be defined by $d = n/\log_2(\max_i a_i)$. Then for "almost all" problems of density $d < 0.645$, the vector $\mathbf{v}$ we searched for is the shortest nonzero vector in the lattice. For "almost all" problems of density $d < 1/n$, it is proved that the lattice basis reduction algorithm locates $\mathbf{v}$. Extensive computational tests of the algorithm suggest that it works for densities $d < d_c(n)$, where $d_c(n)$ is a cutoff value that is substantially larger than $1/n$. This method gives a polynomial time attack on knapsack public-key cryptosystems that can be expected to break them if they transmit information at rates below $d_c(n)$, as $n \to \infty$.

Categories and Subject Descriptors: E.3 [**Data Encryption**]: Public Key Cryptosystems; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*number-theoretic computations*; G.2.0 [**Discrete Mathematics**]: General

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Knapsack public-key cryptosystem, subset sum problems, integer lattice, L^3 algorithm

1. *Introduction*

The *subset sum problem* is a well-known NP-complete set recognition problem [8, p. 226]. The problem is stated as follows: Given a set $A = \{a_i : 1 \le i \le n\}$ of positive integers and a positive integer M, recognize when some subset of A has sum equal to a given integer M. We consider the related NP-hard algorithmic problem: Find a feasible solution to the 0–1 integer programming problem

$$\sum_{i=1}^{n} a_i x_i = M; \qquad \forall i,\ x_i = 0 \quad \text{or} \quad 1, \tag{1.1}$$

when one exists.

A preliminary version of this paper appeared in *Proceedings of the 1983 IEEE 24th Annual Symposium on the Foundations of Computer Science*. IEEE, New York, 1983, pp. 1–10. © IEEE 1983.

Authors' address: AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1985 ACM 0004-5411/85/0100-0229 $00.75

Several proposed public-key cryptosystems, called *knapsack public-key cryptosystems*, are based on this problem [12, 15, 18]. Such cryptosystems give a set of weights $\{a_i: 1 \le i \le n\}$ as public information. A plaintext message consisting of a 0–1 vector $(e_1, \ldots, e_n)$ is encrypted using (1.1), the integer M being the ciphertext. The problem of decrypting an encrypted message M is, therefore, an instance of (1.1). In such cryptosystems the weights $\{a_i: 1 \le i \le n\}$ are chosen in such a way that (1.1) can be easily solved if certain secret information, called a *trapdoor*, is known. In particular, the sets of weights $\{a_i: 1 \le i \le n\}$ used in such cryptosystems form a very special subclass of subset sum problems (1.1). In 1982 Shamir [17] announced a method for breaking the simplest such public-key cryptosystem, the basic Merkle–Hellman cryptosystem. Since then several attacks on more complicated knapsack cryptosystems have been proposed [1, 11, 16]. These attacks are all based on the idea of recovering the trapdoor information concealed in the weights $\{a_i: 1 \le i \le n\}$.

In this paper we propose a simple method, which we call *Algorithm SV*, for directly locating a feasible solution to (1.1). Let $\mathbf{a} = (a_1, \ldots, a_n)$. The method consists of transforming (1.1) to the problem of finding a particular short vector $\mathbf{e}$ in an integer lattice $L = L(\mathbf{a}, M)$. Then we apply a lattice basis reduction algorithm to produce a reduced basis of the lattice. This algorithm is due to A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász [13]; we call it the L^3 *algorithm*. The method succeeds if $\pm\mathbf{e}$ appears in the reduced basis; a solution to (1.1) follows immediately from $\mathbf{e}$.

Since the problem (1.1) is NP-hard, Algorithm SV cannot always be expected to succeed. We analyze the circumstances under which it can be expected to work. We define the *density* $d(\mathbf{a})$ of a set of weights $\mathbf{a} = (a_1, \ldots, a_n)$ by

$$d(\mathbf{a}) = \frac{n}{\log_2(\max_i a_i)}.$$

In terms of knapsack public-key cryptosystems, $d(\mathbf{a})$ is an approximate measure of the *information rate* at which bits are transmitted; that is,

$$d(\mathbf{a}) \cong \frac{\text{number of bits in plaintext message}}{\text{average number of bits in ciphertext message}}.$$

Our main result is a performance analysis of Algorithm SV, which shows that it succeeds for "low-density" subset sum problems as follows:

(1) For "almost all" subset sum problems with $d(\mathbf{a}) < 0.645$, the vector $\mathbf{e}$ is the shortest nonzero vector in the lattice $L = L(\mathbf{a}, M)$. (See Theorem 3.3.)
(2) For "almost all" solvable subset sum problems with n weights having $d(\mathbf{a}) < (2 - \epsilon)(\log_2 4/3)^{-1} n^{-1}$, for any fixed $\epsilon > 0$, Algorithm SV finds a solution. (See Theorem 3.5 and the remark following its proof.)

We believe that the first result is essentially the best possible in the sense that it is no longer true when 0.645 is replaced by 0.646. (Our belief is based on heuristic arguments that we describe in Section 5.)

The second result is weaker than what we believe to be true. The reason for this is as follows. The L^3 algorithm is not guaranteed to produce the shortest nonzero vector $\mathbf{x}_{\min}$ in a lattice $L \subseteq \mathbf{Z}^n$, but only a relatively short vector. To prove that the algorithm succeeds on "almost all" problems with n weights having density $d(a) < (2 - \epsilon)(\log_2 (4/3))^{-1} n^{-1}$, we use a worst-case bound on the length of the shortest vector found by the L^3 algorithm (Proposition 2.1). Empirical experience with the

L^3 algorithm suggests that it usually finds considerably shorter vectors than those guaranteed by this bound. Computational evidence suggests that the algorithm succeeds on "almost all" problems with n items for which $d(a) < d_c(n)$ where $d_c(n)$ is a cutoff value that is substantially larger than $2(\log_2 (4/3))^{-1}n^{-1}$. We do not have enough data to make a reasonable guess on the behavior of $d_c(n)$, but it seems likely that $d_c(n) \to 0$ as $n \to \infty$. (See Section 4 for more details.)

The algorithm we present uses the L^3 algorithm because it is currently the only algorithm known for finding short vectors in a lattice that has been rigorously proved both to have a polynomial running time and to find reasonably short vectors in a lattice. Instead in Algorithm SV, one could use modifications of other algorithms for finding short vectors in a lattice or for finding good multidimensional Diophantine approximations such as those described in [2], [4], [6], and [7]; these might perform well in practice.

What are the consequences of these results for breaking knapsack-type public-key cryptosystems? First, the empirical evidence implies that this method will very likely break nearly all knapsack cryptosystems for which $d(\mathbf{a}) < d_c(n)$ in polynomial time. In particular, it may well break "almost all" *ultimate knapsack cryptosystems* of Shamir [18], since these cryptosystems have $d(\mathbf{a}) < 1/\log_2 n$. Second, this method complements the existing attacks on knapsack cryptosystems that are based on recovering trapdoor information. When the information rate is low, the method described here should succeed. When the information rate is high, the trapdoor information is more difficult to conceal, and attacks based on finding the trapdoor are more likely to succeed, see [11].

Brickell [5] has developed another method to solve general subset sum problems, which can be expected to break most "low-density" problems. Although his method is superficially dissimilar to our method, its success seems to be based on the same basic principles. His method is more complicated and seems difficult to analyze in detail theoretically. Some further remarks on Brickell's algorithm are made in Section 5.

2. *The Method*

Before describing the method, we state the basic facts about integer lattices and the L^3 algorithm that we shall use.

We present the vector space $\mathbf{R}^n$ using row vectors, and define the *length* (i.e., *Euclidean norm*) $\| \mathbf{v} \|$ of a vector $\mathbf{v} = (v_1, \ldots, v_n)$ by

$$\| \mathbf{v} \|^2 = \sum_{i=1}^{n} v_i^2. \tag{2.1}$$

An *integer lattice* L is an additive subgroup of $\mathbf{Z}^n$ that contains n linearly independent vectors over $\mathbf{R}^n$. An (*ordered*) *basis* $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ of a lattice L is a set of elements of L such that $L = \mathbf{Z}\mathbf{v}_1 \oplus \mathbf{Z}\mathbf{v}_2 \oplus \cdots \oplus \mathbf{Z}\mathbf{v}_n$. We represent an ordered basis of a lattice L by the $n \times n$ *basis matrix*

$$V = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix}$$

whose rows are the basis vectors. If V_1 and V_2 are basis matrices of the same lattice L, then there is a unimodular matrix $U \in GL(n, \mathbf{Z})$ such that

$$UV_1 = V_2.$$

Conversely, if V is a basis matrix of L and $U \in GL(n, \mathbf{Z})$, then UV is a basis matrix of L. Lenstra et al. define the notion of a *reduced* (ordered) basis $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ of a lattice L. For the purpose of this paper, we do not need to know the precise definition of a reduced basis (it is given in the Appendix); we need only know that any reduced basis contains a relatively short vector [13, prop. 1.11].

PROPOSITION 2.1. *Let* $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ *be a reduced basis of a lattice* L. *Then*

$$\|\mathbf{v}_1\|^2 \leq 2^{n-1} \min_{\substack{\mathbf{x} \in L \\ \mathbf{x} \neq 0}} \|\mathbf{x}\|^2. \tag{2.2}$$

In fact, all of the basis vectors in a reduced basis tend to be short [13, prop. 1.12]; we take advantage of this in our method. Lenstra et al. [13] present an algorithm, which we call the L^3 algorithm, that, when given a basis $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ of a lattice L as input, produces a reduced basis $[\mathbf{w}_1, \ldots, \mathbf{w}_n]$ as output. They give the following polynomial worst-case running time bound for its performance [13, prop. 1.26].

PROPOSITION 2.2. *Let* $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ *be a basis of an integer lattice* L *such that* $\|\mathbf{v}_i\|^2 \leq B$ *for* $1 \leq i \leq n$. *Then the* L^3 *algorithm produces a reduced basis* $[\mathbf{w}_1, \ldots, \mathbf{w}_n]$ *for* L *using at most* $O(n^4 \log B)$ *arithmetic operations, and the integers on which these operations are performed have binary length at most* $O(n \log B)$.

If we use the classical algorithms for addition, subtraction, multiplication, and division, this algorithm has a guaranteed running time of $O(n^6(\log B)^3)$ bit operations. There are some practical speedups possible for this algorithm so that it seems possible in practice to find a reduced basis in $O(n(\log B)^3)$ bit operations. (See Section 4, [9], and [16].)

Now we can describe the method. We suppose we are given a vector $\mathbf{a} = (a_1, \ldots, a_n)$ of positive integers and an integer M. Our object is to find a feasible solution to

$$\sum_{i=1}^{n} a_i x_i = M; \qquad \forall i,\ x_i = 0 \quad \text{or} \quad 1. \tag{2.3}$$

We need only consider the case that $1 \leq M < \sum_{i=1}^{n} a_i$. We use the following algorithm.

Algorithm SV (SV = Short Vector)

(1) Take the following vectors as a basis $[\mathbf{b}_1, \ldots, \mathbf{b}_{n+1}]$ for an $n+1$-dimensional integer lattice $L = L(\mathbf{a}, M)$:

$$\begin{aligned} \mathbf{b}_1 &= (1, 0, \ldots, 0, -a_1) \\ \mathbf{b}_2 &= (0, 1, \ldots, 0, -a_2) \\ &\vdots \\ \mathbf{b}_n &= (0, 0, \ldots, 1, -a_n) \\ \mathbf{b}_{n+1} &= (0, 0, \ldots, 0, M). \end{aligned} \tag{2.4}$$

(2) Find a *reduced basis* $[\mathbf{b}_1^*, \ldots, \mathbf{b}_{n+1}^*]$ of L using the L^3 algorithm.

(3) Check if any $\mathbf{b}_i^* = (b_{i,1}^*, \ldots, b_{i,n+1}^*)$ has all $b_{ij}^* = 0$ or λ for some fixed λ for $1 \leq j \leq n$. For any such $\mathbf{b}_i^*$, check whether $x_j = \lambda^{-1} b_{i,j}^*$ for $1 \leq j \leq n$ gives a solution to (2.3), and if so, halt. Otherwise, continue.

(4) Repeat steps (1)–(3) with M replaced by $M' = \sum_{i=1}^{n} a_i - M$. Then halt.

If Algorithm SV produces a solution to (1) we say it *succeeds*; otherwise, it *fails*.

Since Algorithm SV is essentially two applications of the L^3 algorithm, we immediately obtain the following running time bound.

LEMMA 2.3. *Let* $\{a_i: 1 \le i \le n\}$ *and* $M < \sum_{i=1}^{n} a_i$ *be given as input to Algorithm SV, and suppose max* $a_i \le B$. *Then Algorithm SV halts after at most* $O(n^6(\log nB)^3)$ *bit operations.*

3. *Performance Analysis*

Our goal is to analyze the performance of Algorithm SV on a class of subset sum problems

$$\sum_{i=1}^{n} a_i x_i = M; \qquad \forall i,\ x_i = 0 \quad \text{or} \quad 1, \tag{3.1}$$

that are known to have a solution. To this end, we suppose that (3.1) has a particular distinguished 0–1 solution $(e_1, \ldots, e_n)$, which we treat as fixed, and that

$$1 \le \sum_{i=1}^{n} e_i \le n - 1;$$

that is, we exclude the trivial cases where $M = 0$ or $\sum_{i=1}^{n} a_i$. We set $\mathbf{e} = (e_1, \ldots, e_n, 0)$.

We analyze the performance of Algorithm SV over a sample space of lattices. We define this *sample space* $\Lambda(B, \mathbf{e})$ to consist of all lattices $L(\mathbf{a}, M)$ defined by (2.4) such that

$$\mathbf{a} = (a_1, \ldots, a_n) \quad \text{has} \quad 1 \le a_i \le B \qquad \text{for all} \quad i, \tag{3.2}$$

$$M = M(\mathbf{a}, \mathbf{e}) = \sum_{i=1}^{n} a_i e_i. \tag{3.3}$$

In particular there is exactly one lattice $L(\mathbf{a}, M)$ in $\Lambda(B, \mathbf{e})$ for each $\mathbf{a}$ satisfying (3.2); hence $\Lambda(B, \mathbf{e})$ contains exactly B^n lattices. The distinguished vector $\mathbf{e}$ is in all the lattices in the sample space $\Lambda(B, \mathbf{e})$ since (2.4) and (3.2) give

$$\mathbf{e} = \sum_{i=1}^{n} e_i \mathbf{b}_i + \mathbf{b}_{n+1}. \tag{3.4}$$

The connection between the sample space $\Lambda(B, \mathbf{e})$ and the density $d(\mathbf{a})$ of its associated subset sum problems is as follows. All subset sum problems (3.1) associated with lattices in $\Lambda(B, \mathbf{e})$ have

$$d(\mathbf{a}) \ge \frac{n}{\log_2 B}, \tag{3.5}$$

and every $\mathbf{a}$ satisfying (3.5) contributes exactly one lattice to $\Lambda(B, \mathbf{e})$. Furthermore, for any $\epsilon > 0$ the fraction of lattices in $\Lambda(B, \mathbf{e})$ with

$$d(\mathbf{a}) \le \frac{n}{\log_2(B(1 - \epsilon))}$$

goes to 1 as $n \to \infty$ if $\log_2 B \sim cn$ for some $c > 0$. Consequently, the sample space $\Lambda(B, \mathbf{e})$ may be regarded as sampling subset sum problems of density $n/\log_2 B$.

We can now formulate the problem we want to solve as follows: Determine how often Algorithm SV finds the distinguished vector $\mathbf{e}$, when applied to all the lattices in the sample space $\Lambda(B, \mathbf{e})$. This problem is intimately tied to the question: How short is $\mathbf{e}$ relative to other short vectors in the lattices in $\Lambda(B, \mathbf{e})$? We consider this question first.

The expected length of other short vectors in lattices in $\Lambda(B, \mathbf{e})$ other than the distinguished vector $\mathbf{e}$ can be determined using Theorem 3.1 below. The bound

given by Theorem 3.1 involves the number of lattice points in spheres in n-dimensional space. We define $S_n(R)$ to be the number of integer solutions to the inequality

$$\sum_{i=1}^{n} x_i^2 \le R, \tag{3.6}$$

that is, the number of integer lattice points inside or on the n-dimensional sphere of radius $\sqrt{R}$ centered at the origin.

THEOREM 3.1. *The number of lattices $L(\mathbf{a})$ in the sample space $\Lambda(B, \mathbf{e})$ that contain a vector $\mathbf{w}$ such that*

(i) $\mathbf{w} \neq k\,\mathbf{e}$ *for all integers* k,

(ii) $\|\mathbf{w}\|^2 \le R$,

is

$$O(R\, S_n(R) B^{n-1} \log(BR)). \tag{3.7}$$

PROOF. Let $T = T(R, B, \mathbf{e})$ denote the number of such lattices. Let $\mathbf{w} = (w_1, \ldots, w_n, r) \in \mathbf{Z}^{n+1}$ be a fixed vector satisfying

$$\|\mathbf{w}\|^2 = \sum_{i=1}^{n} w_i^2 + r^2 \le R \tag{3.8}$$

and suppose that $\mathbf{w} \neq k\,\mathbf{e}$ for every integer k. We count how many lattices $L(\mathbf{a})$ in $\Lambda(B, \mathbf{e})$ contain $\mathbf{w}$. If $\mathbf{w} \in L(\mathbf{a})$, then expressing $\mathbf{w}$ in terms of the basis vectors (2.4) of $L(\mathbf{a})$ gives

$$\mathbf{w} = \sum_{i=1}^{n} w_i \mathbf{b}_i + \lambda \mathbf{b}_{n+1} \tag{3.9}$$

for some integer λ. In particular, evaluating the last coordinate of (3.9) gives

$$r = \sum_{i=1}^{n} w_i a_i - \lambda M(\mathbf{a}), \tag{3.10}$$

and using (3.3) gives

$$r = \sum_{i=1}^{n} (w_i - \lambda e_i) a_i. \tag{3.11}$$

We can easily bound λ using (3.10); we obtain

$$|\lambda|\, M \le |r| + \sum_{i=1}^{n} |w_i a_i| \le B(|r| + \sum_{i=1}^{n} |w_i|) \le RB \tag{3.12}$$

using (3.8), since r and the w_i are integers, so that $M \ge 1$ implies

$$|\lambda| \le RB. \tag{3.13}$$

Next we note that since $\mathbf{e} \neq 0$, it has a nonzero coordinate, which we suppose to be e_1 for convenience in subsequent calculations. Then

$$M = M(\mathbf{a}) = M(\mathbf{a}, \mathbf{e}) \ge a_1 e_1 = a_1, \tag{3.14}$$

so that (3.12) gives

$$a_1 \le \frac{RB}{|\lambda|}, \qquad \text{if} \quad \lambda \neq 0. \tag{3.15}$$

Also we note that (3.8) implies

$$|r| \leq R^{1/2}. \tag{3.16}$$

Now we commence counting. Let $N(\mathbf{w}, \lambda)$ denote the number of lattices $L(\mathbf{a})$ in $\Lambda(B, \mathbf{e})$ for which $\mathbf{w}$ is in $L(\mathbf{a})$ and for which λ satisfies (3.9). Then (3.13) gives

$$T \leq \sum_{\|\mathbf{w}\|^2 \leq R}{}' \left\{ \sum_{\lambda=-RB}^{RB} N(\mathbf{w}, \lambda) \right\}, \tag{3.17}$$

where the prime in the summation indicates that all $\mathbf{w}$ with

$$\mathbf{w} = k\mathbf{e}; \qquad k \quad \text{an integer}; \tag{3.18}$$

are excluded. To estimate this sum, we divide the sum on the right side of (3.17) into four sums, depending on the value of the auxiliary vector

$$\mathbf{z} = \mathbf{z}(\mathbf{w}, \lambda) = (w_1 - \lambda e_1, \ldots, w_n - \lambda e_n) \tag{3.19}$$

and the value of λ.

Case 1. $\mathbf{z} = 0$.

In this case (3.19) gives

$$\mathbf{w} = (\lambda e_1, \ldots, \lambda e_n, N) \tag{3.20}$$

for some $N \neq 0$. Then

$$\mathbf{w} - \lambda\mathbf{e} = (0, \ldots, 0, N)$$

is in $L(\mathbf{a})$, so that necessarily $N = kM(\mathbf{a})$ for some integer k. If $k = 0$, then $\mathbf{w} = \lambda\mathbf{e}$, which is ruled out by hypothesis (i). Hence $|k| \geq 1$ and

$$\|\mathbf{w}\| \geq |M(\mathbf{a})| \geq a_1,$$

using (3.14). The condition $\|\mathbf{w}\|^2 < R$ implies that

$$a_1 \leq R^{1/2}. \tag{3.21}$$

Consequently, we obtain the bound

$$N(\mathbf{w}, \lambda) \leq R^{1/2}B^{n-1}.$$

Now there are no more than $S_n(R)$ choices of $\mathbf{w}$, and each such $\mathbf{w}$ uniquely determines λ via (3.20), so that

$$\sum_{\text{Case 1}} N(\mathbf{w}, \lambda) = O(R^{1/2}S_n(R)B^{n-1}). \tag{3.22}$$

Case 2. $w_1 - \lambda e_1 \neq 0$ and $w_j - \lambda e_j = 0$ for $2 \leq j \leq n$.

In this case, (3.11) gives

$$r = (w_1 - \lambda e_1)a_1. \tag{3.23}$$

Together with (3.16), this gives

$$1 \leq a_1 \leq R^{1/2}, \tag{3.24}$$

so that

$$N(\mathbf{w}, \lambda) \leq R^{1/2}B^{n-1} \tag{3.25}$$

for such pairs $(\mathbf{w}, \lambda)$.

How many such pairs $(\mathbf{w}, \lambda)$ can occur? We have the bound

$$|w_1| < R^{1/2} \tag{3.26}$$

from (3.8), while (3.23) and (3.16) yield

$$|w_1 - \lambda e_1| \le \frac{r}{a_1} \le R^{1/2}. \tag{3.27}$$

Combining (3.26) and (3.27) and using $e_1 = 1$ give

$$|\lambda| \le 2R^{1/2}. \tag{3.28}$$

The values of $(w_2, \ldots, w_n)$ are all determined by

$$w_j = \lambda e_j,$$

so that there are $O(R)$ choices of pairs $(\mathbf{w}, \lambda)$ in Case 2. Hence

$$\sum_{\text{Case 2}} N(\mathbf{w}, \lambda) = O(R^{3/2}B^{n-1}). \tag{3.29}$$

Case 3. $w_j - \lambda e_j \neq 0$ for some $j \ge 2$, and $\lambda \neq 0$.

Consider $\mathbf{w}$ and λ as fixed. Now by (3.15) there are at most RB/λ choices for a_1. Now choose all the other a_i arbitrarily, except for $i = j$. There are B^{n-2} such choices. For each such choice, there is at most one possible choice for a_j, since a_j is determined by eq. (3.11), since $w_j - \lambda e_j \neq 0$. Hence in this case

$$N(\mathbf{w}, \lambda) \le \frac{RB^{n-1}}{\lambda}. \tag{3.30}$$

Hence

$$\begin{aligned}\sum_{\text{Case 3}} N(\mathbf{w}, \lambda) &\le \sum_{\|\mathbf{w}\|^2 \le R} \sum_{\substack{\lambda = -RB \\ \lambda \neq 0}}^{RB} \frac{RB^{n-1}}{\lambda} \\ &\le 2RB^{n-1}S_n(R) \sum_{\lambda=1}^{RB} \frac{1}{\lambda}.\end{aligned}$$

Since

$$\sum_{i=1}^{RB} \frac{1}{\lambda} = O(\log(RB)),$$

this yields

$$\sum_{\text{Case 3}} N(\mathbf{w}, \lambda) = O(RS_n(R)B^{n-1}\log(RB)). \tag{3.31}$$

Case 4. Some $w_j - \lambda e_j \neq 0$ for $j \ge 2$ and $\lambda = 0$.

Consider $\mathbf{w}$ as fixed. In this case we can pick all a_i except a_j arbitrarily, and there are B^{n-1} such choices. There are at most $2R^{1/2} + 1$ choices for a_j, since it must satisfy (3.11) and there are at most $2R^{1/2} + 1$ choices of r by (3.16). Hence in this case

$$N(\mathbf{w}, 0) \le (2R^{1/2} + 1)B^{n-1}.$$

Consequently, summing over all $\mathbf{w}$ gives

$$\sum_{\text{Case 4}} N(\mathbf{w}, \lambda) \le (2R^{1/2} + 1)S_n(R)B^{n-1}. \tag{3.32}$$

Theorem 3.1 follows on combining the bounds (3.22), (3.29), (3.31), and (3.32) and the trivial inequality $S_n(R) \geq R$. □

We remark that the dependence on B in Theorem 3.1 cannot be much improved, since all $L(\mathbf{a}, M)$ for which $a_1 = a_2$ contain the short vector $\mathbf{w} = (1, -1, 0, 0, \ldots, 0)$ which satisfies the conditions of Theorem 3.1, and there are B^{n-1} such lattices in $\Lambda(B, \mathbf{e})$. It is an interesting question as to whether or not substantial improvement is possible in the dependence on R in (3.7).

To apply Theorem 3.1, we need explicit estimates for the number of lattice points in spheres. A general principle here is that $S_n(R)$ should be equal to the volume $V_n(R)$ of a sphere of radius $R^{1/2}$, with an error proportional to the surface area $A_n(R)$ of such a sphere. Now

$$\begin{aligned} V_n(R) &= c_n R^{n/2}, \\ A_n(R) &= n c_n R^{(n-1)/2}, \end{aligned} \tag{3.33}$$

where

$$c_n = \frac{\pi^{n/2}}{\Gamma(n/2 + 1)} \tag{3.34}$$

is the volume of an n-dimensional sphere of radius 1. For large R one has $V_n(R)$ much larger than $A_n(R)$, but for R small enough, say $R = \alpha n$, this is not true, and spheres of this radius centered at the origin contain many more lattice points than their volume would suggest. It turns out, furthermore, that for n-dimensional spheres of such small radius $(\alpha n)^{1/2}$, the number of lattice points in the sphere depends strongly on the location of the center of the sphere [14]. For our application we need a good upper bound for $S_n(\frac{1}{2}n)$, and to obtain it we use the following simplified version of the proof in [14].

THEOREM 3.2. *For all* $n \geq 1$, $S_n(\frac{1}{2}n) \leq 2^{1.54725n}$.

PROOF. Let $\theta(z) = 1 + 2\sum_{i=1}^{\infty} z^{i^2}$. Let $r_n(k)$ count the number of solutions to

$$\sum_{i=1}^{n} x_i^2 = k.$$

Then

$$[\theta(z)]^n = \sum_{k=0}^{\infty} r_n(k) z^k.$$

Now for $x \geq 0$ we have

$$\begin{aligned} S_n(\alpha n) &= \sum_{k \leq \alpha n} r_n(k) \\ &\leq e^{n\alpha x} \sum_{k=0}^{\infty} r_n(k) e^{-kx} \\ &= e^{n\alpha x}[\theta(e^{-x})]^n, \end{aligned} \tag{3.35}$$

since for $x \geq 0$ we have

$$e^{n\alpha x} e^{-kx} \geq 1 \qquad \text{when} \quad k \leq n\alpha.$$

Now set

$$\delta(\alpha, x) = \alpha x + \ln \theta(e^{-x})$$

and observe that (3.35) gives

$$S_n(\alpha n) \le e^{n\delta} = 2^{(\log_2 e)\delta(\alpha,x)n}. \tag{3.36}$$

We are interested in $\alpha = \frac{1}{2}$ and choose $x \ge 0$ to optimize (3.36); the value $x = x_0 = 0.997994$ is a nearly optimal choice. Then

$$\delta(\tfrac{1}{2}, x_0) \le 1.07247$$

and

$$(\log_2 e)\delta(\tfrac{1}{2}, x_0) \le 1.54725. \qquad \square$$

We remark that the constant 1.54725 in Theorem 3.2 is the best possible to within one unit in the last decimal place (see [14]).

Now we prove a result about short vectors in lattices in the class $\Lambda(B, \mathbf{e})$ where $\mathbf{e}$ satisfies

$$\sum_{i=1}^{n} e_i \le \tfrac{1}{2}n. \tag{3.37}$$

The reason we consider this extra condition is that Algorithm SV examines two lattice problems, one of which is a lattice $L(\mathbf{a}, \mathbf{e})$ and the other $L(\mathbf{a}, \mathbf{e}^*)$ where $\mathbf{e}^* = (e_1^*, \ldots, e_n^*)$ is the 0–1 vector complementary to $\mathbf{e}$; that is, $e_i^* = 1 - e_i$ for all i. Since

$$\min\left(\sum_{i=1}^{n} e_i, \sum_{i=1}^{n} e_i^*\right) \le \frac{1}{2}n,$$

the hypothesis (3.37) applies to at least one of these lattice problems.

THEOREM 3.3. *Let* $\mathbf{e}$ *be a* 0–1 *vector for which* $\sum_{i=1}^{n} e_i \le n/2$. *Then if* $B = 2^{\beta n}$ *for any constant* $\beta > 1.54725$, *the number of lattices* L *in* $\Lambda(B, \mathbf{e})$ *for which* $\mathbf{e}$ *is the nonzero vector of shortest Euclidean norm in* L *is*

$$B^n + O(B^{n-c_1(\beta)}(\log B)^2)$$

where $c_1(\beta) = 1 - 1.54725/\beta > 0$.

This theorem asserts that, under the stated hypotheses, "almost all" the lattices in $\Lambda(B, \mathbf{e})$ have $\mathbf{e}$ as the shortest vector. In particular, for $B = 2^{\beta n}$ the density $d(\mathbf{a})$ of lattices in $\Lambda(B, \mathbf{e})$ is β^{-1}, so that this theorem applies to sets of lattices with density less than $(1.54725)^{-1} \cong 0.645$.

PROOF OF THEOREM 3.3. Theorem 3.1 estimates the number of such lattices by

$$B^n + O(nS_n(\tfrac{1}{2}n)B^{n-1}\log(B_n)).$$

Applying Theorem 3.2 gives

$$S_n(\tfrac{1}{2}n) \le 2^{1.54725n} \le B^{1-c_1(\beta)},$$

where $B \ge 2^{\beta n}$. Finally $n \log B_n = O((\log B)^2)$ for $B \ge 2^{\beta n}$, and the theorem follows. $\square$

Theorem 3.3 gave a result when the vector $\mathbf{e}$ is fixed. We can immediately derive a result where $\mathbf{e}$ varies.

THEOREM 3.4. *Let* $B = 2^{\beta n}$ *for any* $\beta > 2.54725$. *The number of vectors* $\mathbf{a} = (a_1, \ldots, a_n)$ *with* $1 \le a_i \le B$ *for* $1 \le i \le n$ *for which* $\mathbf{e}$ *is the shortest vector in* $L(\mathbf{a}, \mathbf{e})$

for all 0–1 *vectors* **e** *for which*

$$1 \le \sum_{i=1}^{n} e_i \le \frac{n}{2} \tag{3.38}$$

is

$$B^n + O(B^{n-c_2(\beta)}(\log B)^2), \tag{3.39}$$

where $c_2(\beta) = 1 - 2.54725/\beta > 0$.

PROOF. Sum the result of Theorem 3.1 over all $2^{n-1} - 1$ vectors **e** satisfying (3.38). The resulting bound is

$$O(n2^n S_n \left(\frac{n}{2}\right) B^{n-1} \log(nB)).$$

This is certainly an upper bound for the error term in (3.39). Now use

$$2^n S_n \left(\frac{n}{2}\right) \le 2^{2.54725n} \le B^{1-c_2(\beta)},$$

and the result follows. □

Theorem 3.4 makes an assertion about lattices of density $d(\mathbf{a}) \le 0.393 < (2.54725)^{-1}$. Now we prove the main result on the performance of Algorithm SV.

THEOREM 3.5. Let $B > 2^{(1/2+\beta)n^2}$ *for some fixed* $\beta > 0$. *Then the number of vectors* $\mathbf{a} = (a_1, \ldots, a_n)$ *with* $1 \le a_i < B$ *for all* i *for which Algorithm SV will succeed for all* 0–1 *vectors* **e** *is*

$$B^n + O(B^{n-c_3(\beta)+4(\log n)/n}),$$

where $c_3(\beta) = 2\beta/(1 + 2\beta) > 0$.

This theorem asserts then that for any fixed $\beta > 0$ one can solve the subset sum problem for "almost all" $\mathbf{a} = (a_1, \ldots, a_n)$ for which $d(\mathbf{a}) < (\frac{1}{2} + \beta)^{-1} n^{-1}$, provided $n \ge n_0(\beta)$.

PROOF OF THEOREM 3.5. At least one of the two lattice problems that Algorithm SV considers has an associated **e** satisfying

$$\sum_{i=1}^{n} e_i \le \tfrac{1}{2}n. \tag{3.40}$$

Now suppose, for this lattice problem, that the lattice $L(\mathbf{a}, \mathbf{e})$ has the property that all vectors **w** in $L(\mathbf{a}, \mathbf{e})$ that are not a scalar multiple of **e** satisfy

$$\|\mathbf{w}\| > n2^{n-2} \ge 2^{n-1}\|\mathbf{e}\|$$

using (3.40). Then Proposition 2.1 guarantees that some vector $\lambda\mathbf{e}$ must appear in the reduced basis produced by the L^3 algorithm applied to $L(\mathbf{a}, \mathbf{e})$. Hence Algorithm SV succeeds in this case. (We remark that if $\lambda\mathbf{e}$ appears in a reduced basis, then necessarily $\lambda = \pm 1$.)

It remains to bound the exceptional cases in which this does not occur. We use the bound of Theorem 3.1 with $R = n2^{n-2}$, summing over all **e** satisfying (3.40), to obtain the upper bound

$$O(n2^n S_n(n2^{n-2})B^{n-1} \log(n2^{n-1}B)), \tag{3.41}$$

for the exceptional cases. Then, using the trivial bound

$$S_n(R) \le (2R^{1/2} + 1)^n \le 3^n R^{n/2},$$

we can easily obtain an upper bound for (3.41) of

$$O(2^{n^2/2+n\log_2 n} B^{n-1} \log(n2^n B)).$$

Since $B \ge 2^{(1/2+\beta)n^2}$, we find that for large n,

$$\log(n2^n B) \le B^{1/n},$$

and

$$n^2 2^{n^2/2+n\log n} = O(B^{1-c_3(\beta)+3(\log n)/n}),$$

where $c_3(\beta) = 2\beta(1 + 2\beta)$. □

Theorem 3.5 can be sharpened by using an improved form of the L^3 algorithm. Lenstra et al. [13] actually defined a notion of *y-reduced basis*, which depends on a parameter y satisfying $\frac{1}{4} \le y < 1$. The notion of reduced basis corresponds to choosing $y = \frac{3}{4}$; the general definition is given in the Appendix. For a y-reduced basis the bound (2.2) of Proposition 2.1 is replaced by

$$\| \mathbf{v}_i \|^2 \le \left(\frac{4}{4y - 1} \right)^{n-1} \min_{\substack{\mathbf{x} \in L \\ \mathbf{x} \ne 0}} \| \mathbf{x} \|^2.$$

They gave an algorithm, which we may call the $L^3(y)$ *algorithm*, that produces a y-reduced basis. An analog of Proposition 2.2 holds for this algorithm, in which the constants implied by the O symbols depend on the choice of y. We can modify Algorithm SV to use the $L^3(y)$ algorithm and obtain Algorithm SV(y). Then we may prove Theorem 3.5 for Algorithm SV(y), obtaining a similar bound for $B = 2^{(1+\beta)c(y)n^2}$ where $c(y) = \frac{1}{2}\log_2(4/(4y - 1))$. With this bound, letting $y \to 1$, we get a result that asserts that we can solve the subset sum problem for "almost all" problems of density $d(\mathbf{a}) < (2 - \epsilon)(\log_2(4/3))^{-1} n^{-1}$.

4. *Computational Results*

We performed extensive computational tests using Algorithm SV. We tested several variants of Algorithm SV, obtained by modifying the L^3 algorithm in ways designed to improve its chance of finding the shortest vector in a lattice. We considered two such modifications.

(1) Running the $L^3(y)$ algorithm for $y = 1$ instead of $y = \frac{3}{4}$. In the case $y = 1$, the algorithm is not proved to run in polynomial time; however, in practice, it takes about three times as long as the algorithm with $y = \frac{3}{4}$. It seems to find much shorter vectors than the algorithm with $y = \frac{3}{4}$.

(2) Running the L^3 algorithm several times for the same lattice, starting with different initial ordered bases for the lattice. The rationale for this is that running the L^3 algorithm with different initial ordered bases will tend to produce different reduced bases; finding several reduced bases improves our chance of finding one containing the shortest vector in the lattice. We obtained different ordered bases by randomly permuting the given ordered basis.

Our initial computations were done on a VAX 11/780 using the Vaxima symbolic manipulation system on lattices of dimension ≤21. Then we made

TABLE I. TEST RESULTS USING THE $L^3(y)$ ALGORITHM WITH $y = 1$

Dimension n	$\log_2 B$	Density d	Number of Tests	Number of Successes		Success Rate after 5 Trials
				1 trial	5 trials	
14	16	0.875	50	42	48	0.96
	20	0.700	50	50	50	1.00
20	24	0.833	50	16	36	0.72
	30	0.667	50	47	50	1.00
	36	0.555	50	50	50	1.00
26	30	0.867	50	10	23	0.46
	36	0.722	50	29	49	0.98
	44	0.591	50	42	50	1.00
30	36	0.833	50	10	19	0.38
	44	0.682	50	17	38	0.76
	45	0.667	10	5	8	0.80
	51	0.588	10	7	10	1.00
	60	0.500	10	8	10	1.00
40	60	0.667	9	2	6	0.67
	68	0.588	9	4	8	0.89
	80	0.500	10	8	10	1.00
50	85	0.588	5	1	1	0.20
	100	0.500	3	1	2	0.67

extensive computations on a CRAY-1, using standard FORTRAN double-precision arithmetic when possible and otherwise the Brent MP (multiprecision) package [3]. The version of the $L^3(y)$ algorithm we used differed from that described in [13] in the use of floating-point approximations to the μ_{ij}. This version of the algorithm is not guaranteed to produce a reduced basis in polynomial time, but in practice it does. The running times of our implementation on the CRAY-1 using the Brent MP package were proportional to $n(\log B)^3$ over a wide range of values of n and B, up to $n = 80$. The $n = 50$, $\log_2 B = 100$ runs of the $L^3(1)$ algorithm described below took on the order of 14 minutes of CPU time on a CRAY-1.

The results of the most systematic of our tests are presented in Table I. In these tests we fixed the dimension of the subset sum problem n and fixed $\log_2 B$; and then we generated at random a vector $\mathbf{a} = (a_1, \ldots, a_n)$ with $1 \leq a_i \leq B$, and defined a vector $\mathbf{e} = (e_1, \ldots, e_n)$ by $e_i = 1$ for $1 \leq i \leq n/2$ and $e_i = 0$ otherwise. We ran the $L^3(y)$ algorithm with $y = 1$ on a basis of the lattice $L(\mathbf{a}, M(\mathbf{a}, \mathbf{e}))$ obtained by randomly permuting the rows of the basis (2.4). The run was a success if $\pm\mathbf{e}$ appeared in the reduced basis. In cases of failure, the L^3 algorithm was applied again to another permutation of the basis (2.4), and this was repeated for up to five trials.

Table I defines the density d of a problem as $n/\log_2 B$. It reports the number of successes both on the first trial and after five trials of permuting the basis. The final column gives the success rate after five permutations of the basis. We observed that when the run was a success and $\pm\mathbf{e}$ appeared in the reduced basis, it was not always the first vector in the reduced basis. In most cases in which $\pm\mathbf{e}$ appeared in the reduced basis, it was the shortest vector in the reduced basis. If we use an observed success rate of 1.00 as an estimate for an upper bound for the cutoff value $d_c(n)$, we have $d_c(30) \leq 0.60$, $d_c(40) \leq 0.50$. This suggests to us that $d_c(n) \to 0$ as $n \to \infty$.

We also examined the effect of varying the parameter y in the $L^3(y)$ algorithm. The effect of setting $y = 1$ is to increase the running time (by roughly a factor of 3 compared with the usual variant with $y = \frac{3}{4}$) but also to produce considerably shorter vectors in the reduced basis, and thereby to increase the chances of success of Algorithm SV. For example, although the 50 runs with $n = 26$, $\log_2 B = 36$ described in Table I led to 49 successes for $y = 1$ with 29 of them on the first try, the same algorithm with $y = \frac{3}{4}$ achieved only 19 successes, 2 of them on the first try. We encountered a few cases where values of $y < 1$ led to success, whereas $y = 1$ did not, but such cases were rare.

Lenstra et al. [13] prove that if $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ is a reduced basis of a lattice L, and $d(L)$ denotes the determinant of the lattice (which equals the determinant of the matrix formed by the vectors $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$), then

$$r = \frac{\prod_{i=1}^{n} \| \mathbf{v}_i^* \|}{d(L)}$$

satisfies

$$r \leq \left(\frac{4}{4y - 1} \right)^{n(n-1)/4}. \tag{4.1}$$

Our runs for $n \leq 50$ indicate that for most L-reduced lattices $\ln r$ is on the order of $n^2/50$ for $y = 1$, and on the order of $n^2/33$ for $y = 3/4$.

5. *Discussion*

(1) Theorem 3.3 implies that "almost all" subset sum problems (1.1) with $d(\mathbf{a}) < 0.645$ can be solved in polynomial time if one can find a polynomial time algorithm that "almost always" finds the shortest Euclidean norm vector in an integer lattice L. This possibility may seem unlikely in view of the result that the corresponding sup norm minimum vector problem is NP-hard (see [10]). On the other hand, there are natural NP-complete problems for which polynomial-time algorithms exist that solve "almost all" instances of the problem, (Wilf [19]).

(2) We conjecture that Theorem 3.3 is sharp in the sense that there is a *critical density* β_0 below which Theorem 3.3 is true for "almost all" subset sum problems of density $d(\mathbf{a}) \leq \beta$ for any $\beta < \beta_0$ and true for "almost no" subset sum problems with $d(\mathbf{a}) \geq \beta$ for any fixed $\beta > \beta_0$; here we average not only over all sets $a_1, \ldots, a_n$ of the given density, but also over all binary vectors $\mathbf{e}$, so that $\sum e_i \sim n/2$ holds for almost all of them. Furthermore, we conjecture that this critical density is

$$\beta_0 = (C_0)^{-1} = 0.645 \cdots, \tag{5.1}$$

where

$$C_0 = \lim_{n \to \infty} \left(\log_2 S_n \left(\frac{n}{2} \right) \right) = 1.54725 \cdots. \tag{5.2}$$

We are unable to prove these conjectures rigorously. It is possible to use arguments similar to those used in the proof of Theorem 3.1 to obtain a lower bound of $\Omega(S_n(n/2)B^{n-1})$ for the number of pairs $(L, \mathbf{x})$, where L is a lattice in $\Lambda(B, \mathbf{e})$ containing the vector $\mathbf{x}$, where $\| \mathbf{x} \|^2 \leq n/2$. Using this result one can show that the *expected* number of lattice points $\mathbf{x}$ with $\| \mathbf{x} \|^2 \leq n/2$ in a lattice drawn from $\Lambda(B, \mathbf{e})$ with $\sum_{i=1}^{n} e_i = [n/2]$ and $B = 2^{\beta n}$ is at least $2^{(\beta - \beta_0 - \epsilon)n}$ for any fixed $\epsilon > 0$, as $n \to \infty$. The conjectures above would follow from a proof that "almost all" lattices

in $\Lambda(B, \mathbf{e})$ contain around the expected number of short vectors $\mathbf{x}$ with $\|\mathbf{x}\|^2 \leq n/2$.

(3) Algorithm SV can be expected to solve even relatively dense subset sum problems when the solution has $\sum e_i$ either very small or very large, since then the $\mathbf{e}$ vector in one of the lattices considered by the algorithm will be very short. For example, if $\sum e_i \sim n/4$, the critical density changes from 0.645 to 0.94. For a random problem, we expect $\sum e_i \sim n/2$ by the law of large numbers. In data communication one often encounters a preponderance of zeros in the message $\mathbf{e}$, in which case Algorithm SV should succeed on much higher density problems for those particular $\mathbf{e}$. We have verified this experimentally for $n \leq 40$.

(4) For what kinds of low-density vectors $\mathbf{a}$ does Algorithm SV fail? This question is important if we want to design a knapsack-type public-key cryptosystem that is immune to this attack.

It is easy to see that Theorem 3.3 does not hold for vectors $\mathbf{a} = (a_1, \ldots, a_n)$, which satisfy a small linear dependency

$$\sum_{i=1}^{n} a_i \lambda_i = 0$$

with $\sum \lambda_i^2$ small. In this case every lattice $L(\mathbf{a}, M)$ contains the short vector $\mathbf{v}_1 = (\lambda_1, \ldots, \lambda_n, 0)$. However Algorithm SV may still succeed in this case, since it may find both $\mathbf{v}_0 = (e_1, \ldots, e_n, 0)$ and $\mathbf{v}_1$ in the reduced basis it produces. Indeed this has occurred on numerical examples. It appears that for this attack to fail $\mathbf{a}$ must have *many* small linear dependencies. Two problems immediately face the would-be cipher designer:

(i) A fast decryption algorithm for such $\mathbf{a} = (a_1, \ldots, a_n)$ is needed.
(ii) The existence of a large number of linear dependencies in the a_i's forms a point of attack for cryptanalysis of such problems.

(5) How are the results of Theorem 3.3 and 3.6 to be interpreted as applying to knapsack-type public-key cryptosystems? Indeed, although these results apply to "almost all" subset sum problems in the appropriate range, the particular subset sum problems arising from the knapsack cryptosystems proposed so far form only an infinitesimal fraction of "subset sum" problems in this range. Therefore it could be argued that perhaps Algorithm SV will not succeed on them.

The simplest way to address this criticism is by actually testing Algorithm SV on such subset sum problems. We have performed such tests on three subset sum problems, with $n = 20$, $B \sim 2^{40}$, that were constructed using a doubly iterated Merkle–Hellman knapsack [15]. All three tests were successful. In fact, we expect that Algorithm SV will tend to be *more* successful on vectors $\mathbf{a}$ arising from knapsack cryptosystems than for random vectors, because these $\mathbf{a}$ automatically have all small linear dependencies of the form

$$\sum_{i=1}^{n} a_i x_i = 0; \qquad \forall i,\ x_i = 1, 0, \quad \text{or} \quad -1,$$

ruled out by the need for unique decipherability of encrypted messages.

(6) E. Brickell (personal communication) has suggested that "dense" subset sum problems might be solved by converting them to "low-density" subset sum problems through one or more modular multiplications. To do this, let

$$\sum_{i=1}^{n} a_i x_i = M \tag{5.3}$$

be the original problem, and select a pair (W, Y) subject to

$$Y > \sum_{i=1}^{n} a_i; \qquad (W, Y) = 1,$$

and form

$$b_i \equiv Wa_i \pmod{Y}; \qquad 0 \le b_i < Y. \tag{5.4}$$

Algorithm SV presumably does not work on the original problem because the $\{a_i\}$ satisfy many small linear dependencies

$$\sum_{i=1}^{n} a_i x_i = 0. \tag{5.5}$$

Indeed, we indicated in (2) of this section that for subset sum problems of density $\beta > \beta_0$, where β_0 is the critical density, we expect that there are many such small linear dependencies (5.5), on the order of $2^{(\beta-\beta_0+\epsilon)n}$. Now a single modular multiplication will destroy some of these dependencies. Indeed, if we write $b_i = Wa_i - k_i Y$, then we see that

$$\sum_{i=1}^{n} b_i x_i = 0 \tag{5.6}$$

holds if and only if

$$\sum_{i=1}^{n} k_i x_i = 0. \tag{5.7}$$

This last condition is equivalent to

$$\sum_{i=1}^{n} \left\{ \frac{Wa_i}{Y} \right\} x_i = 0, \tag{5.8}$$

where $\{t\}$ denotes the fractional part of t. We expect a given small linear dependency (5.5) to hold after a single modular multiplication with probability on the order of $n^{-1/2}$. The total number of such small linear dependencies is thus cut down by at most a factor of $n^{1/2}$. (Note that new small linear dependencies may be introduced; the probability of this is quite low, however.) Therefore, we expect that it requires on the order of n successive modular multiplications to arrive at a subset sum problem in which all of the small linear dependencies have been eliminated and for which Algorithm SV can be expected to succeed.

We expect that repeated modular multiplication used in this way are of little help in solving the original subset sum problem (5.3). This is because a single modular multiplication transforms the original subset sum problem into one of n different subset sum problems, for if

$$M' \equiv WM \pmod{Y}, \qquad 0 \le M' \le Y,$$

then we only know that

$$M' = \sum_{i=1}^{n} b_i X_i - kY$$

holds for some k with $0 \le k \le n$. A probabilistic argument suggests that usually k is on the order of $n^{1/2}$, so that about $n^{1/2}$ subset sum problems after the modular multiplication by (W, Y) have to be examined to solve the original problem (5.3). This blowup by a factor of $n^{1/2}$ essentially cancels the $n^{-1/2}$ reduction in small

linear dependencies; after n successive modular multiplications, we would have exponentially many subset sum problems to search, exactly one of which corresponds to the original problem (5.3).

(7) The success of Brickell's method [5] appears to be based on similar principles to our method. However, it seems difficult to analyze. His method has two practical advantages over our method:

(i) It has an initial preprocessing step with the same asymptotic running time as our method. If it succeeds, however, it can subsequently solve all subset sum problems with the same $\mathbf{a}$ for each M in time $O(n^2(\log(M + \|\mathbf{a}\|))^2)$ or less.

(ii) If it succeeds, it can afterwards test for infeasibility of (1.1) for a given M in time $O(n^2(\log(M + \|\mathbf{a}\|))^2)$ or less.

The cutoff density $d(\mathbf{a})$ below which Brickell's method can be expected to be successful may turn out to be smaller than that for Algorithm SV. This is because it must succeed in solving all problems (1.1) for a given $\mathbf{a}$ when it succeeds (compare Theorems 3.3 and 3.4).

(8) When trying to solve many subset sum problems with the same $\mathbf{a}$, it might be possible to speed up Algorithm SV by first reducing a lattice $L = L(\mathbf{a}, M)$ with $M = M^*$ very large, say $M^* = (\max a_i)^2$. The reduced basis would consist of n vectors, call them $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$, that are linear combinations of $\mathbf{b}_1, \ldots, \mathbf{b}_n$ only, and of one vector, call it $\mathbf{b}_{n+1}^*$, that is long and depends on $\mathbf{b}_{n+1}$ also. Then, to solve a subset sum problem with a given M, we can reduce a basis consisting of $\mathbf{b}_1^*, \ldots, \mathbf{b}_n^*$ and $(0, \ldots, 0, M)$. Since the $\mathbf{b}_i^*$are short, this reduction ought to be very rapid.

Appendix. Reduced Basis of a Lattice

Let $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ be an ordered basis of a lattice L. Associated to this basis is the set of vectors $[\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*]$ obtained by applying the Gram–Schmidt process to $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$, namely,

$$\begin{aligned}
\mathbf{v}_1^* &= \mathbf{v}_1 \\
\mathbf{v}_2^* &= \mathbf{v}_2 - \mu_{2,1}\mathbf{v}_1^* \\
&\vdots \\
\mathbf{v}_n^* &= \mathbf{v}_n - \mu_{n,1}\mathbf{v}_1^* - \cdots - \mu_{n,n-1}\mathbf{v}_{n-1}^*
\end{aligned}$$

where

$$\mu_{ij} = \frac{(\mathbf{v}_i, \mathbf{v}_j^*)}{(\mathbf{v}_j^*, \mathbf{v}_j^*)}, \qquad \text{for} \quad j < i.$$

Let y be a parameter satisfying $\frac{1}{4} \le y < 1$. Lenstra et al. define $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ to be *y-reduced* provided that:

(1) $\|\mathbf{v}_i^* + \mu_{i,i-1}\mathbf{v}_{i-1}^*\|^2 \ge y\|\mathbf{v}_{i-1}^*\|^2$ for $2 \le i \le n$,
(2) $|\mu_{ij}| \le \frac{1}{2}$ for all $i > j$,

both hold. The definition of reduced basis used in [13] (and in this paper) takes $y = \frac{3}{4}$.

This definition of y-reduced basis can be reformulated in another way. Let V_j denote the $\mathbf{R}$-subspace of $\mathbf{R}^n$ spanned by $[\mathbf{v}_1, \ldots, \mathbf{v}_{j-1}]$ (equivalently, $[\mathbf{v}_1^*, \ldots, \mathbf{v}_{j-1}^*]$). Let $\mathbf{v}_i(j)$ for $i \ge j$ denote the projection of $\mathbf{v}_i$ onto the orthogonal complement V_j^* of V_j. In particular $\mathbf{v}_i(i) = \mathbf{v}_i^*$. Then the conditions above for a basis to be *y-reduced* can be written as

(1) $\|\mathbf{v}_i(i-1)\|^2 \ge y\|\mathbf{v}_{i-1}(i-1)\|^2$,
(2) $\|\mathbf{v}_i(j+1) - \mathbf{v}_i(j)\| \le \frac{1}{2}\|\mathbf{v}_j(j)\|$, for all $i > j$.

REFERENCES

1. ADLEMAN, L.M. On breaking generalized knapsack public key cryptosystems. In *Proceedings of the 15th ACM Symposium on Theory of Computing* (Boston, Mass., Apr. 25–27). ACM, New York, 1983, 402–412.
2. AFFLERBACH, L. Minkowskische Reduktionsbedingungen für positiv definite quadratische Formen in 5 Variabeln. *Monatsh. Math. 94* (1982), 1–8.
3. BRENT, R.P. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Softw. 4,* 1 (Mar. 1978), 57–70.
4. BRENTJES, A.J. Multi-dimensional continued fraction algorithms. Mathematical Centre Tract No. 145, Matematisch Centrum, Amsterdam, The Netherlands, 1981.
5. BRICKELL, E. Are most low density knapsacks solvable in polynomial time? In *Proceedings of the 14th Southeastern Conference on Combinatorics, Graph Theory, and Computing, 1983.* Congressus Numerantium, Vol. 39, 1983, pp. 145–156.
6. DIETER, U. How to calculate shortest vectors in a lattice. *Math. Comput. 29* (1975), 827–833.
7. FERGUSON, H.R.P., AND FORCADE, R.W. Multidimensional Euclidean algorithms. *J. reine angew. Math. 344* (1982), 171–181.
8. GAREY, M.R., AND JOHNSON, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.
9. KALTOFEN, E. On the complexity of finding short vectors in integer lattices. In *Computer Algebra: Proceedings of EUROCAL '83, European Computer Algebra Conference,* J. A. VanHulzen, Ed. Lecture Notes in Computer Science, vol. 162. Springer-Verlag, New York, 1983, pp. 236–244.
10. LAGARIAS, J.C. The computational complexity of simultaneous Diophantine approximation problems. *SIAM J Comput. 14* (1985), to be published.
11. LAGARIAS, J.C. Knapsack-type public key cryptosystems and Diophantine approximation, (Extended Abstract). In *Advances in Cryptology, Proceedings of CRYPTO-83* (Santa Barbara, Aug.), D. Chaum, Ed. Plenum, New York, 1984, pp. 3–24.
12. LEMPEL, A. Cryptography in transition: A survey. *Comput. Surv. 11* (1979), 285–304.
13. LENSTRA, A.K., LENSTRA, H.W., JR., AND LOVÁSZ, L. Factoring polynomials with rational coefficients. *Math. Annalen 261* (1982), 515–534.
14. MAZO, J.E., AND ODLYZKO, A.M. Lattice points in high-dimensional spheres, paper in preparation.
15. MERKLE, R.C., AND HELLMAN, M.E. Hiding information and signatures in trap-door knapsacks. *IEEE Trans. Inf. Theory IT-24* (1978), 525–530.
16. ODLYZKO, A.M. Cryptanalytic attacks on the multiplicative knapsack cryptosystem and on Shamir's fast signature scheme. *IEEE Trans. Inf. Theory IT-30,* 4 (July 1984), 584–601.
17. SHAMIR, A. A polynomial time algorithm for breaking the Merkle–Hellman cryptosystem. In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science.* IEEE, New York, 1982, pp. 145–152.
18. SHAMIR, A. Embedding cryptographic trapdoors in arbitrary knapsack systems. *Inf. Proc. Lett. 17* (1983), 77–79.
19. WILF, H.S. Backtrack: An $O(1)$ expected time algorithm for the graph coloring problem. *Inf. Proc. Lett. 18* (1984), 119–121.

RECEIVED AUGUST 1983; REVISED MARCH 1984; ACCEPTED AUGUST 1984