Closest point search in lattices. IEEE Trans Inf Theory

Article *in* IEEE Transactions on Information Theory · September 2002

DOI: 10.1109/TIT.2002.800499 · Source: IEEE Xplore

4 authors, including:

Erik Agrell
Chalmers University of Technology
297 PUBLICATIONS **6,887** CITATIONS

SEE PROFILE

Thomas Eriksson
Chalmers University of Technology
316 PUBLICATIONS **6,279** CITATIONS

SEE PROFILE

Alexander Vardy
University of California, San Diego
281 PUBLICATIONS **14,639** CITATIONS

SEE PROFILE

Closest Point Search in Lattices

Erik Agrell, *Member, IEEE*, Thomas Eriksson, *Member, IEEE*, Alexander Vardy, *Fellow, IEEE*, and Kenneth Zeger, *Fellow, IEEE*

***Abstract*—In this semitutorial paper, a comprehensive survey of closest point search methods for lattices without a regular structure is presented. The existing search strategies are described in a unified framework, and differences between them are elucidated. An efficient closest point search algorithm, based on the Schnorr–Euchner variation of the Pohst method, is implemented. Given an arbitrary point $\boldsymbol{x} \in \mathbb{R}^m$ and a generator matrix for a lattice Λ, the algorithm computes the point of Λ that is closest to $\boldsymbol{x}$. The algorithm is shown to be substantially faster than other known methods, by means of a theoretical comparison with the Kannan algorithm and an experimental comparison with the Pohst algorithm and its variants, such as the recent Viterbo–Boutros decoder. Modifications of the algorithm are developed to solve a number of related search problems for lattices, such as finding a shortest vector, determining the kissing number, computing the Voronoi-relevant vectors, and finding a Korkine–Zolotareff reduced basis.**

***Index Terms*—Closest point search, kissing number, Korkine–Zolotareff (KZ) reduction, lattice decoding, lattice quantization, nearest neighbor, shortest vector, Voronoi diagram.**

I. Introduction

In lattice theory, a *generator matrix* $\boldsymbol{G}$ is any matrix with real entries whose rows are linearly independent over $\mathbb{R}$. We let n and m denote the number of rows and columns of $\boldsymbol{G}$, respectively. Hence $n \leqslant m$. The *lattice* generated by $\boldsymbol{G}$ is

$$\Lambda(\boldsymbol{G}) \stackrel{\text{def}}{=} \{\boldsymbol{u}\boldsymbol{G}\colon \boldsymbol{u} \in \mathbb{Z}^n\}.$$

The rows of $\boldsymbol{G}$ are called *basis vectors* for Λ, and the number n of basis vectors is said to be the *dimension* of Λ.

The *closest point problem* is the problem of finding, for a given lattice Λ and a given input point $\boldsymbol{x} \in \mathbb{R}^m$, a vector $\hat{\boldsymbol{c}} \in \Lambda$ such that

$$||\boldsymbol{x} - \hat{\boldsymbol{c}}|| \leqslant ||\boldsymbol{x} - \boldsymbol{c}||, \qquad \text{for all } \boldsymbol{c} \in \Lambda$$

where $||\cdot||$ denotes the Euclidean norm. In channel coding, the closest point problem is referred to as *decoding*, and we adopt this terminology herein. Note, however, that in source coding, this problem is called *encoding* (see below).

The *Voronoi region* of a lattice point is the set of all vectors in $\mathbb{R}^m$ that can be decoded to this point, namely

$$\Omega(\Lambda, \boldsymbol{c}) \stackrel{\text{def}}{=} \{\boldsymbol{x} \in \mathbb{R}^m\colon ||\boldsymbol{x} - \boldsymbol{c}|| \leqslant ||\boldsymbol{x} - \boldsymbol{c}'||,\ \forall\, \boldsymbol{c}' \in \Lambda\}$$

Manuscript received December 4, 2000; revised October 5, 2001. This work was supported in part by the National Science Foundation, the David and Lucile Packard Foundation, and Stiftelsen ISS '90.

E. Agrell and T. Eriksson are with the Department of Signals and Systems, Chalmers University of Technology, S-412 96 Göteborg, Sweden. (e-mail: agrell@s2.chalmers.se; thomase@s2.chalmers.se).

A. Vardy and K. Zeger are with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA (e-mail: vardy@montblanc.ucsd.edu; zeger@ucsd.edu).

Communicated by P. Solé, Associate Editor for Coding Theory.

Publisher Item Identifier 10.1109/TIT.2002.800499.

where $\boldsymbol{c} \in \Lambda$. The *Voronoi diagram* of a lattice is the set of all its Voronoi regions. It is known [23] that the Voronoi regions $\Omega(\Lambda, \boldsymbol{c})$ are convex polytopes, that they are symmetrical with respect to reflection in $\boldsymbol{c}$, and that they are translations of $\Omega(\Lambda, \boldsymbol{0})$, where $\boldsymbol{0}$ is the origin of $\mathbb{R}^m$.

In communication theory, lattices are used for both modulation and quantization. If a lattice is used as a code for the Gaussian channel, maximum-likelihood decoding in the demodulator is a closest point search. The decoding of space–time codes is one example [16], [17], [25]. Analogously, if a lattice is used as a codebook for vector quantization and the mean-squared-error criterion is used, then the encoding of each input vector is also a closest point search. Furthermore, if the lattice is truncated into a so-called Voronoi code [21], another instance of the closest point problem arises at the opposite end of the communication system, in the source decoder and in the modulator. Typical for these applications in communications is that the same lattice is decoded numerous times for different input vectors.

Other applications where the closest point problem arises include lattice design [3] and Monte Carlo second-moment estimation [22]. In both cases, random vectors are generated uniformly in a Voronoi region of a lattice using closest point search.

The closely related *shortest vector problem* has been used in assessing the quality of noncryptographic random number generators [50, pp. 89–113] and in decoding of Chinese remainder codes [38], [40]. It also has important applications in cryptography [5], [7]. Another related problem of paramount importance in cryptography [13], [70] is that of lattice *basis reduction*. These search problems will be discussed in Section VI.

The choice of method for solving the closest point problem depends on the structure of the lattice. Intuitively, the more structure a lattice has, the faster can the closest point be found. For many classical lattices, efficient search methods are known [23, Ch. 20], [75]. A more general approach is to represent a lattice by a trellis [72] and use a trellis decoding algorithm such as the Viterbi algorithm [11], [33], [34], [76]. However, finite-state trellises exist if and only if the lattice contains n mutually orthogonal vectors, and even then decoding complexity quickly becomes prohibitive [73].

Herein, we address the problem of finding the closest point in a general lattice: we assume that it has no exploitable structure. One situation where this problem arises is when a generator matrix is continuously adjusted, e.g., in numerical lattice design [3]. Another important application of this problem is cryptanalysis [13], [68]. Yet another example is frequency estimation and phase unwrapping [19].

The complexity of the general closest point problem as a function of the dimension n was analyzed by van Emde

Boas [74] two decades ago, who showed that this problem is NP-hard. Micciancio gave a simpler proof in [55]. Thus, all known algorithms for solving the problem optimally have exponential complexity. It is known [9] that finding an approximate solution, such that the ratio between the distance found and the true distance is upper-bounded by a constant, is also NP-hard. Even finding a suboptimal solution within a factor $n^{c/\log\log n}$ for some constant $c > 0$ is NP-hard [27]. Nevertheless, algorithms that find a suboptimal solution are faster and can handle higher dimensions [52].

A common approach to the general closest point problem is to identify a certain region in $\mathbb{R}^m$ within which the optimal lattice point must lie, and then investigate all lattice points in this region, possibly reducing its size dynamically. The earliest work in the field was done for the shortest vector problem (see Section VI-A) in the context of assessing the quality of certain random number generators (cf. [24], [26] and [50, pp. 89–101, 110]). The finite region searched in these algorithms is a parallelepiped, with its axes parallel to the basis vectors.

In general, the development of closest point algorithms follows two main branches, inspired by two seminal papers: Pohst [63] in 1981 examined lattice points lying inside a hypersphere, whereas Kannan [46] in 1983 used a rectangular parallelepiped. Both papers later appeared in revised and extended versions, Pohst's as [30] and Kannan's (following the work of Helfrich [42]) as [47]. The Pohst and Kannan strategies are discussed in greater detail in Section III-A.

A crucial parameter for the performance of these algorithms is the initial size of the search region. Some suggestions to this point were given in [62], [78] for the Pohst strategy and in [12] for the Kannan strategy. The latter reference also includes an extensive complexity analysis. Applications are discussed in [15], [62], [78], [80].

Another, more subtle, difference between the two strategies is implicit in their presentation. Grossly generalizing, the Pohst method is intended as a practical tool while the method of Kannan is intended as a theoretical tool. Papers dealing with the Pohst strategy typically discuss issues of implementation, whereas papers dealing with the Kannan strategy usually focus on asymptotic complexity. This is probably the reason why the two strategies, despite having so much in common, have never been compared and evaluated against each other in the literature.

Recently, Schnorr and Euchner [67] suggested an important improvement of the Pohst strategy, based on examining the points inside the aforementioned hypersphere in a different order. In Sections V and VII-C, the strategies by Pohst, Kannan, and Schnorr–Euchner are compared to each other, and it is shown that the Schnorr–Euchner strategy is substantially faster than the other two.

While the preceding discussion is distilled from the existing literature, much of this literature is not directly accessible. Often, the results are buried in the context of specific applications. For example, the Schnorr–Euchner algorithm is described in [67] merely as a subroutine, called ENUM, in a function that computes the so-called block Korkine–Zolotareff (KZ) reduction, which itself serves as a tool for solving a certain type of subset-sum problems [67] and attacking the Chor–Rivest cryptosystem [68]. Thus, although the question "What is the best (fastest) algorithm currently available for decoding a general lattice?" frequently arises in communication practice, the answer to this question is not immediately clear.

In this paper, we first describe the two main decoding strategies, due to Pohst and to Kannan, in a unified framework, which makes it possible to elucidate the similarities and the differences between them. This is done in Section III-A, where we also discuss the Babai nearest plane algorithm [10] and the Schnorr–Euchner refinement of the Pohst strategy. In Section III-B, we present a stand-alone implementation of what we believe is the fastest closest point search algorithm currently available for general lattices. The algorithm is based on the Schnorr–Euchner [67] strategy, bootstrapped with the Babai [10] nearest point. It is described in sufficient detail to allow straightforward implementation, without knowledge of the underlying theory. One of the main contributions of this paper is a theoretical and experimental comparison of the various closest point search algorithms, presented in Sections V and VII, respectively. We also show in Section IV how a carefully selected preprocessing stage can reduce the complexity of the closest point search even further. Finally, we describe in Section VI several modifications to the algorithm of Section III-B designed to solve numerous related lattice-search problems, such as finding a shortest vector, determining the kissing number, computing the Voronoi-relevant vectors, and finding a Korkine–Zolotareff reduced basis.

II. PRELIMINARIES

We say that two lattices are *identical* if all lattice points are the same. Two generator matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ generate identical lattices $\Lambda(\boldsymbol{G}_1) = \Lambda(\boldsymbol{G}_2)$ if and only if

$$\boldsymbol{G}_1 = \boldsymbol{W}\boldsymbol{G}_2 \tag{1}$$

where $\boldsymbol{W}$ is a square matrix with integer entries such that $|\det \boldsymbol{W}| = 1$. A generator matrix $\boldsymbol{G}_2$ is a *rotated and reflected* representation of another generator matrix $\boldsymbol{G}_1$ if

$$\boldsymbol{G}_1 = \boldsymbol{G}_2\boldsymbol{Q} \tag{2}$$

where $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}$. This transformation can be regarded as a change of the coordinate system. If $\boldsymbol{G}_2$ is square and lower triangular, it is said to be a *lower-triangular representation* of $\boldsymbol{G}_1$. Any generator matrix has a lower-triangular representation, which is unique up to column negation. How to find a lower-triangular representation of a given generator matrix is discussed in Section IV.

Two lattices are congruent, or *equivalent*, if one can be obtained from the other through scaling, rotation, and reflection. Two generator matrices $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$ generate equivalent lattices if and only if

$$\boldsymbol{G}_1 = c\boldsymbol{W}\boldsymbol{G}_2\boldsymbol{Q} \tag{3}$$

where $c > 0$ is a real constant, while $\boldsymbol{W}$ and $\boldsymbol{Q}$ obey the same conditions as in (1) and (2), respectively. The equivalence relation is denoted $\Lambda(\boldsymbol{G}_2) \cong \Lambda(\boldsymbol{G}_1)$.

The process of selecting a good basis for a given lattice, given some criterion, is called *reduction*. In many applications, it is advantageous if the basis vectors are as short as possible and

"reasonably" orthogonal to each other (for lattice-search problems, this was first noted by Coveyou and MacPherson [24]). This property of the basis vectors can be formalized in a number of ways, giving rise to several types of reduction. Simply selecting the n shortest nonzero vectors in the lattice is, however, not a practicable approach, since these vectors do not in general form a basis.

The problem was studied by Hermite in 1850, who suggested [44, pp. 301–303] that a generator matrix $\boldsymbol{G}$ with rows $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ is reduced if the following holds for all $i = 1, \ldots, n$: $\|\boldsymbol{v}_i\| \leqslant \|\boldsymbol{v}'_i\|$, for all generator matrices $\boldsymbol{G}'$ with rows $\boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_n$ such that $\Lambda(\boldsymbol{G}') = \Lambda(\boldsymbol{G})$ and $\|\boldsymbol{v}'_j\| = \|\boldsymbol{v}_j\|$ for $j = 1, \ldots, i-1$. In other words, a generator matrix $\boldsymbol{G}$ is reduced in this sense if the sequence $(\|\boldsymbol{v}_1\|, \ldots, \|\boldsymbol{v}_n\|)$ comes first in a lexicographically ordered list of the corresponding sequences for all generator matrices of the same lattice. The first basis vector $\boldsymbol{v}_1$ is always a shortest nonzero lattice vector. There exists at least one reduced basis in this sense for every lattice, but Hermite gave no algorithm to compute it. Note that this reduction criterion is usually *not* referred to as the "Hermite reduction" in recent literature (see footnote 2).

Minkowski made extensive use of the above reduction criterion in his earlier work [56] [57], [58]. In 1905, he suggested a subtle but significant modification [61], defining the criterion now known as the *Minkowski reduction*. A generator matrix $\boldsymbol{G}$ with rows $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ is Minkowski-reduced if the following holds for all $i = 1, \ldots, n$: $\|\boldsymbol{v}_i\| \leqslant \|\boldsymbol{v}'_i\|$ for all $\boldsymbol{G}'$ with rows $\boldsymbol{v}'_1, \ldots, \boldsymbol{v}'_n$ such that $\Lambda(\boldsymbol{G}') = \Lambda(\boldsymbol{G})$ and $\boldsymbol{v}'_j = \boldsymbol{v}_j$ for $j = 1, \ldots, i-1$.[1] This is in essence a "greedy" version of the stricter criterion by Hermite. Suppose that a set of vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_i$ have been found that satisfy Minkowski's criterion up to a certain value of i. Then there is always a Minkowski-reduced basis that contains these vectors, and the search can be focused on finding the next vector $\boldsymbol{v}_{i+1}$ in the basis. This is not necessarily the case with the aforementioned criterion by Hermite. In particular, if there is more than one inequivalent shortest nonzero vector, it may well be that only one of them can be included in a reduced basis in the sense of Hermite, whereas there is always at least one Minkowski-reduced basis for each of them.

Minkowski reduction has received much attention, particularly in number theory [18, pp. 27–28], [28, pp. 83–84]. Algorithms to compute a Minkowski-reduced basis of an arbitrary lattice may be found in [1], [42].

Two types of reduction that are more widely used in practice are Korkine–Zolotareff (KZ) reduction and Lenstra–Lenstra–Lovász (LLL) reduction. One reason for their popularity is that with both of those criteria, the n-dimensional reduction problem can be recursively reduced to an $(n-1)$-dimensional reduction problem, which is not feasible with Minkowski reduction.

The *KZ reduction* is named after the authors of [51], who defined this reduction criterion in 1873. To determine whether a given generator matrix is a KZ-reduced basis, it is convenient to study its lower-triangular representation. A lower-triangular square generator matrix

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_n \end{bmatrix} = \begin{bmatrix} v_{11} & 0 & \cdots & 0 \\ v_{21} & v_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} \tag{4}$$

is defined, recursively, to be KZ-reduced if $n = 1$, or else each of the following three conditions holds:[2]

$$\boldsymbol{v}_1 \text{ is a shortest nonzero vector in } \Lambda(\boldsymbol{G}) \tag{5}$$

$$|v_{k1}| \leqslant \frac{|v_{11}|}{2}, \qquad \text{for } k = 2, \ldots, n \tag{6}$$

and the submatrix

$$\begin{bmatrix} v_{22} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ v_{n2} & \cdots & v_{nn} \end{bmatrix} \tag{7}$$

is KZ-reduced. An arbitrary generator matrix is KZ-reduced if and only if its lower-triangular representation is KZ-reduced. It is known [64], that every lattice has at least one KZ-reduced generator matrix.

The *LLL reduction* is named after Lenstra, Lenstra, and Lovász, who suggested the corresponding reduction criteria in [53]. The LLL reduction is often used in situations where the KZ reduction would be too time-consuming. A lower-triangular generator matrix (4) is LLL-reduced if either $n = 1$, or else each of the following three conditions holds:

$$\|\boldsymbol{v}_1\| \leqslant \frac{2}{\sqrt{3}} \|\boldsymbol{v}_2\| \tag{8}$$

$$|v_{k1}| \leqslant \frac{|v_{11}|}{2}, \qquad \text{for } k = 2, \ldots, n \tag{9}$$

and the submatrix (7) is LLL-reduced. As before, an arbitrary generator matrix is LLL-reduced if its lower-triangular representation is LLL-reduced.

Any KZ-reduced matrix is clearly also LLL-reduced. The motivation for the latter reduction is that there exists an efficient algorithm [53] to convert any $n \times m$ generator matrix into an LLL-reduced one. This algorithm, which operates in polynomial time in n and m, has become very popular. It was improved upon in [69] and [66].

The LLL reduction algorithm has been modified in a number of ways, see [20, pp. 78–104]. Hybrids between KZ and LLL reductions have also been proposed [65].

III. Closest Point Search Algorithms

We start with a conceptual description of various lattice search algorithms in Section III-A. In this framework, we introduce the Babai nearest plane algorithm, the Kannan strategy, the Pohst strategy, and the Schnorr–Euchner refinement of the Pohst strategy. In Section III-B, we present a detailed pseudocode implementation of a closest point search algorithm based on the Schnorr–Euchner strategy.

[1] We disregard, as is commonly done in recent literature, that Minkowski also required the scalar product between $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ to be nonnegative for all $i = 1, \ldots, n-1$.

[2] Because the condition (6) was proposed by Hermite in his first and second letters to Jacobi [44, pp. 269–271, 280–282], KZ reduction is sometimes called "Hermite reduction" (cf. [42]). The terminology is further complicated by the fact that in some contexts "Hermite reduction" refers to a criterion for so-called indefinite quadratic forms, not immediately applicable to lattices [18, p. 29].

A. Conceptual Description

To understand lattice search algorithms, a recursive characterization of lattices is useful. Let $\boldsymbol{G}$ be an $n \times m$ generator matrix for a lattice Λ, and let us write $\boldsymbol{G}$ as

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{G}^* \\ \boldsymbol{v}_n \end{bmatrix}$$

where $\boldsymbol{G}^*$ is an $(n-1) \times m$ matrix consisting of the top $n-1$ rows of $\boldsymbol{G}$. Furthermore, let us write $\boldsymbol{v}_n$ as $\boldsymbol{v}_n = \boldsymbol{v}_{\|} + \boldsymbol{v}_{\perp}$, with $\boldsymbol{v}_{\|}$ in the row space of $\boldsymbol{G}^*$ and $\boldsymbol{v}_{\perp}$ in the null space. If $\boldsymbol{G}$ is lower triangular, as in (4), then this decomposition is particularly simple, namely, $\boldsymbol{v}_{\|} = (v_{n1}, \ldots, v_{n,n-1}, 0)$ and $\boldsymbol{v}_{\perp} = (0, \ldots, 0, v_{nn})$.

With this terminology, any n-dimensional lattice can be decomposed as follows:

$$\Lambda(\boldsymbol{G}) = \bigcup_{u_n=-\infty}^{+\infty} \left\{ \boldsymbol{c} + u_n \boldsymbol{v}_{\|} + u_n \boldsymbol{v}_{\perp} \colon \boldsymbol{c} \in \Lambda(\boldsymbol{G}^*) \right\} \tag{10}$$

which is basically a stack of $(n-1)$-dimensional translated sublattices. The $(n-1)$-dimensional hyperplanes that contain these sublattices will be called $(n-1)$-dimensional *layers*. Thus, the index u_n denotes which layer a certain lattice point belongs to. The vector $\boldsymbol{v}_{\|}$ is the offset by which one sublattice is translated within its layer, with respect to an adjacent sublattice. The vector $\boldsymbol{v}_{\perp}$ is normal to the layers, and the distance between two adjacent layers is $\|\boldsymbol{v}_{\perp}\|$. For lower-triangular generator matrices, we have $\|\boldsymbol{v}_{\perp}\| = |v_{nn}|$. Recalling that any generator matrix can be rotated into a lower-triangular form with $v_{nn} > 0$, we let v_{kk} denote the distance between the $(k-1)$-dimensional layers, even when the triangular constraint is not explicitly imposed.

Now, all search algorithms for an n-dimensional lattice will be described recursively as a finite number of $(n-1)$-dimensional search operations. Let $\boldsymbol{x} \in \mathbb{R}^m$ be a vector to decode in the lattice $\Lambda(\boldsymbol{G})$, which is decomposed into layers according to (10). The orthogonal distance from $\boldsymbol{x}$ to the layer with index u_n is given by

$$y_n \stackrel{\text{def}}{=} |u_n - \hat{u}_n| \cdot \|\boldsymbol{v}_{\perp}\| \tag{11}$$

where

$$\hat{u}_n \stackrel{\text{def}}{=} \frac{\boldsymbol{x}\boldsymbol{v}_{\perp}^t}{\|\boldsymbol{v}_{\perp}\|^2}. \tag{12}$$

Let $\hat{\boldsymbol{x}}$ denote the closest lattice point to $\boldsymbol{x}$, and suppose that an upper bound ρ_n on $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|$ is known. Then, in order to ensure that $\hat{\boldsymbol{x}}$ will be found, it suffices to consider a finite number of layers in (10). The indices of these layers are

$$u_n = \left\lceil \hat{u}_n - \frac{\rho_n}{\|\boldsymbol{v}_{\perp}\|} \right\rceil, \ldots, \left\lfloor \hat{u}_n + \frac{\rho_n}{\|\boldsymbol{v}_{\perp}\|} \right\rfloor \tag{13}$$

since layers for which $y_n > \rho_n$ are not relevant. Of these, the layer with $u_n = \lfloor \hat{u}_n \rceil$ has the shortest orthogonal distance to $\boldsymbol{x}$, where $\lfloor z \rceil$ denotes the closest integer to $z \in \mathbb{R}$.

Four types of search methods will now be identified. They each search the layers indexed in (13), but they differ in the order in which these layers are examined and in the choice of the upper bound ρ_{n-1} to be used, recursively, in the $(n-1)$-dimensional search problems.

If only $u_n = \lfloor \hat{u}_n \rceil$ is considered, the n-dimensional search problem is reduced to just one $(n-1)$-dimensional problem, and no upper bound ρ_n is needed. Recursive application of this strategy [10] yields the *Babai nearest plane algorithm*, and we call the returned lattice point the *Babai point*. The Babai nearest plane algorithm is a fast method to find a nearby lattice point, in time polynomial in the number of rows and columns of $\boldsymbol{G}$. In general, the Babai point depends not only on $\boldsymbol{x}$ and the lattice, but also on the basis used to represent the lattice. It is not necessarily the closest point, but the error can be bounded. A probabilistic variant of the Babai nearest plane algorithm was proposed by Klein [49].

The other three methods all find the optimal (closest) point. Scanning all the layers in (13), and supplying each $(n-1)$-dimensional search problem with the same value of ρ_{n-1} regardless of u_n, yields the *Kannan strategy*.[3] Variants of this strategy [12], [42], [46], [47] differ mainly in how the bounds ρ_k are chosen for $k = 1, \ldots, n$. In this context, a recent improvement by Blömer [14] seems particularly promising. Geometrically, the Kannan strategy amounts to generating and examining all lattice points within a given rectangular parallelepiped.

The n-dimensional decoding error vector $\hat{\boldsymbol{x}} - \boldsymbol{x}$ consists, in the given recursive framework, of two orthogonal components: one in the row space of $\boldsymbol{G}^*$ and one parallel to $\boldsymbol{v}_{\perp}$. The former is the $(n-1)$-dimensional decoding error while the length of the latter is y_n. Since y_n varies with u_n, the upper bound ρ_{n-1} can be chosen as

$$\rho_{n-1} = \sqrt{\rho_n^2 - y_n^2} \tag{14}$$

which is different for different layers in (13). The idea of letting ρ_{n-1} depend on u_n is the *Pohst strategy* [30], [62], [63], [78], [80]. In geometrical terms, points inside a hypersphere, not a parallelepiped, are investigated. When any lattice point $\boldsymbol{x}'$ inside the sphere is found, the bound ρ_n can be immediately updated to $\|\boldsymbol{x}' - \boldsymbol{x}\|$, since $\|\boldsymbol{x}' - \boldsymbol{x}\|$ is an obvious upper bound on $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|$ and $\|\boldsymbol{x}' - \boldsymbol{x}\| \leqslant \rho_n$.

The *Schnorr–Euchner strategy*, proposed in [67], combines the advantages of the Babai nearest plane algorithm and the Pohst strategy. Assume that $\hat{u}_n \leqslant \lfloor \hat{u}_n \rceil$. Then the sequence

$$u_n = \lfloor \hat{u}_n \rceil, \lfloor \hat{u}_n \rceil - 1, \lfloor \hat{u}_n \rceil + 1, \lfloor \hat{u}_n \rceil - 2, \ldots \tag{15}$$

orders the layers in (13) according to nondecreasing distance from $\boldsymbol{x}$. A trivial counterpart holds when $\hat{u}_n > \lfloor \hat{u}_n \rceil$. The advantages of examining the layers in this order are subtle but significant. Since the volume of a layer decreases with increasing y_n, the chance of finding the correct layer early is maximized. Another advantage of the nondecreasing distance y_n is that the search can safely be terminated as soon as y_n exceeds the distance to the best lattice point found so far. Notice that the very first lattice point generated will, by definition, be the Babai point. Furthermore, since the ordering in (15) does not depend on ρ_n, no initial bound ρ_n is needed. Instead, this

[3] In its original form [46], [47], Kannan's strategy is described recursively as a set of $(i-1)$-dimensional search problems, where i is the index of the largest element in $(v_{11}, \ldots, v_{nn})$. This viewpoint may be useful for a complexity analysis, but because $u_n, u_{n-1}, \ldots, u_i$ can be selected sequentially, the strategy is computationally equivalent to recursively eliminating just one dimension at a time.

bound can be updated dynamically during the search, with the first finite value of ρ_n being equal to the distance to the Babai point.

B. Detailed Description

This subsection contains a stand-alone presentation of an efficient closest point search algorithm, based on the Schnorr–Euchner strategy. It is intended to be sufficiently detailed to allow a straightforward implementation, even without knowledge of the underlying theory.

For efficiency, the recursive operations discussed in the previous subsection have been restructured into a loop. The variables $\boldsymbol{H}$ and $\hat{\boldsymbol{u}}$ are used as input and output parameters, instead of the more natural $\boldsymbol{G} = \boldsymbol{H}^{-1}$ and $\hat{\boldsymbol{x}} = \hat{\boldsymbol{u}}\boldsymbol{G}$. As discussed in Section IV, this is motivated by the typical communication application, where numerous input vectors are decoded in the same lattice.

First, some notation needs to be defined. Matrix and vector elements are named according to the following conventions:

$$\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$$

$$\boldsymbol{e}_k = (e_{k1}, e_{k2}, \ldots, e_{kk}), \qquad \text{for } k = 1, \ldots, n$$

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & 0 & \cdots & 0 \\ h_{21} & h_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ h_{n1} & h_{n2} & \cdots & h_{nn} \end{bmatrix}.$$

The operation $\mathtt{sgn}*(z)$ returns -1 if $z \leqslant 0$ and 1 if $z > 0$ (which may deviate from most built-in sign functions). Ties in the rounding operation $\lfloor z \rceil$ are broken arbitrarily.

Algorithm DECODE($\boldsymbol{H}, \boldsymbol{x}$)

Input: an $n \times n$ lower-triangular matrix $\boldsymbol{H}$ with positive diagonal elements, and an n-dimensional vector $\boldsymbol{x} \in \mathbb{R}^n$ to decode in the lattice $\Lambda(\boldsymbol{H}^{-1})$.

Output: an n-dimensional vector $\hat{\boldsymbol{u}} \in \mathbb{Z}^n$ such that $\hat{\boldsymbol{u}}\boldsymbol{H}^{-1}$ is a lattice point that is closest to $\boldsymbol{x}$.

```
 1  n := the size of H                          /* dimension */
 2  bestdist := ∞                               /* current distance record */
 3  k := n                                      /* dimension of examined layer */
 4  dist_k := 0                                 /* distance to examined layer */
 5  e_k := xH                                   /* used to compute û_n, see (12) */
 6  u_k := ⌊e_kk⌉                               /* examined lattice point */
 7  y := (e_kk − u_k)/h_kk                      /* see (11) */
 8  step_k := sgn*(y)                           /* offset to next layer in (15) */
 9  ⟨loop⟩
10  newdist := dist_k + y^2
11  if newdist < bestdist then {
                                                          Case A
12    if k ≠ 1 then {
13      e_{k−1,i} := e_ki − y h_ki for i = 1, ..., k − 1
14      k := k − 1                                /* move down */
15      dist_k := newdist
16      u_k := ⌊e_kk⌉                             /* closest layer */
17      y := (e_kk − u_k)/h_kk
18      step_k := sgn*(y)
                                                          Case B
19    } else {
20      û := u                                    /* best lattice point so far */
21      bestdist := newdist                       /* update record */
22      k := k + 1                                /* move up */
23      u_k := u_k + step_k                       /* next layer */
24      y := (e_kk − u_k)/h_kk
25      step_k := −step_k − sgn*(step_k)
26    }
                                                          Case C
27  } else {
28    if k = n then return û (and exit)
29    else {
30      k := k + 1                                /* move up */
31      u_k := u_k + step_k                       /* next layer */
32      y := (e_kk − u_k)/h_kk
33      step_k := −step_k − sgn*(step_k)
34    }
35  }
36 goto ⟨loop⟩
```

In this algorithm, k is the dimension of the sublayer structure that is currently being investigated. Each time the algorithm finds a k-dimensional layer, the distance to which is less than the currently smallest distance, this layer is expanded into $(k-1)$-dimensional sublayers. This is done in Case A. Conversely, as soon as the distance to the examined layer is greater than the lowest distance, the algorithm moves up one step in the hierarchy of layers. This is done in Case C. Case B is invoked when the algorithm has successfully moved down all the way to the zero-dimensional layer (that is, a lattice point) without exceeding the lowest distance. Then this lattice point is stored as a potential output point, the lowest distance is updated, and the algorithm moves back up again, without restarting.

IV. PREPROCESSING AND POSTPROCESSING

The algorithm DECODE of the previous section requires a representation of the lattice at hand by a lower-triangular generator matrix, whose diagonal elements are all positive. Such a representation exists for any lattice, so this requirement does not impose any constraints on the kind of lattices that can be searched. Moreover, for any given lattice, a representation with the required properties can be found in infinitely many ways, which leaves the user with the freedom of choosing one of them. The algorithm computes a closest vector regardless of the representation choice, but the speed with which it reaches this result varies considerably between different representations.

This is the topic of this section: How should a given search problem be preprocessed, in order to make the most efficient use of DECODE?

To address this question, we now present a general lattice search algorithm. This algorithm can be regarded as a "front-end" to DECODE, where explicit preprocessing and postprocessing is performed to allow generator matrices that are not lower triangular, possibly not even square. As with DECODE, we first describe this algorithm conceptually, and then suggest how to implement it.

Assume that a generator matrix $\boldsymbol{G}$ and an input vector $\boldsymbol{x}$ are given. By linear integer row operations, we first transform $\boldsymbol{G}$ into another matrix, say $\boldsymbol{G}_2$, which generates an identical lattice. The purpose of this transformation is to speed up DECODE; see below. Next, we rotate and reflect $\boldsymbol{G}_2$ into a lower-triangular form $\boldsymbol{G}_3$, so that

$$\Lambda(\boldsymbol{G}_3) \cong \Lambda(\boldsymbol{G}_2) = \Lambda(\boldsymbol{G}).$$

It is essential to rotate and reflect the input vector $\boldsymbol{x}$ in the same way, so that the transformed input vector, say $\boldsymbol{x}_3$, is in the same relation to $\Lambda(\boldsymbol{G}_3)$ as $\boldsymbol{x}$ is to $\Lambda(\boldsymbol{G})$. All this can be regarded as a change of the coordinate system. Now the search problem has a form that is suitable for DECODE, which will find the closest lattice point $\hat{\boldsymbol{x}}_3$ in this coordinate system. Reversing the operations of rotation and reflection produces $\hat{\boldsymbol{x}}$, the lattice point closest to $\boldsymbol{x}$ in $\Lambda(\boldsymbol{G})$. Following these steps, the algorithm is detailed as follows.

Algorithm CLOSESTPOINT($\boldsymbol{G}$, $\boldsymbol{x}$)

Input: an $n \times m$ generator matrix $\boldsymbol{G}$, and an m-element vector $\boldsymbol{x} \in \mathbb{R}^m$ to decode in $\Lambda(\boldsymbol{G})$.

Output: a lattice point $\hat{\boldsymbol{x}} \in \Lambda(\boldsymbol{G})$ that is closest to $\boldsymbol{x}$.

Step 1. Let $\boldsymbol{G}_2 := \boldsymbol{W}\boldsymbol{G}$, where $\boldsymbol{W}$ is an $n \times n$ matrix with integer entries and determinant ± 1.

Step 2. Compute an $n \times m$ orthonormal matrix $\boldsymbol{Q}$ such that $\boldsymbol{G}_2 = \boldsymbol{G}_3\boldsymbol{Q}$, where $\boldsymbol{G}_3$ is an $n \times n$ lower-triangular matrix with positive diagonal elements.

Step 3. Let $\boldsymbol{H}_3 := \boldsymbol{G}_3^{-1}$.

Step 4. Let $\boldsymbol{x}_3 := \boldsymbol{x}\boldsymbol{Q}^T$.

Step 5. Let $\hat{\boldsymbol{u}}_3 :=$ DECODE$(\boldsymbol{H}_3, \boldsymbol{x}_3)$.

Step 6. Return $\hat{\boldsymbol{x}} := \hat{\boldsymbol{u}}_3\boldsymbol{G}_2$.

Step 1 is a basis reduction. This step is optional: it is possible to select $\boldsymbol{W}$ as the identity matrix, which amounts to no reduction at all. This works well for low-dimensional and not too ill-conditioned generator matrices, as will be shown in Section VII. However, the speed and the numerical stability of the search can be improved significantly by an appropriate reduction, as discussed later in this section.

Step 2 implies rotation and reflection of $\boldsymbol{G}_2$ into a lower-triangular form, as in (2). The standard method to achieve this is by QR decomposition. Given an arbitrary $m \times n$ matrix $\boldsymbol{M}$, its QR decomposition is a factorization of $\boldsymbol{M}$ of the form $\boldsymbol{M} = \boldsymbol{Q}\boldsymbol{R}$, where $\boldsymbol{R}$ is an $n \times n$ upper-triangular matrix, and $\boldsymbol{Q}$ is an $m \times n$ orthonormal matrix, that is, one satisfying $\boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}$. It is well known that a QR decomposition exists for any matrix; efficient algorithms to compute it may be found in [41, pp. 208–236] and [71, pp. 166–176], for example. In our context, QR decomposition of $\boldsymbol{G}_2^T$ gives both $\boldsymbol{Q}^T$ and $\boldsymbol{G}_3$, with $\boldsymbol{G}_3$ being equal to $\boldsymbol{R}^T$. As an alternative to QR decomposition, $\boldsymbol{G}_3$ can be obtained by Cholesky decomposition of $\boldsymbol{G}_2\boldsymbol{G}_2^T$. Given an $n \times n$ positive-definite matrix $\boldsymbol{A}$, its Cholesky decomposition is a factorization of the form $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{U}^T$ where $\boldsymbol{U}$ is an $n \times n$ upper-triangular matrix. In our context, $\boldsymbol{G}_3$ is equal to $\boldsymbol{U}^T$, and the rotation matrix is given by $\boldsymbol{Q} = \boldsymbol{G}_3^{-1}\boldsymbol{G}_2$. Algorithms for computing the Cholesky decomposition may be found in [20, pp. 102–104], [41, pp. 84–93], and [71, pp. 332–334].

All these transformations can be thought of as a change of the coordinate system. Measure the first coordinate along $\boldsymbol{v}_1$ (the first row of $\boldsymbol{G}_2$), the second in the plane spanned by $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and so on. The generator matrix in this coordinate system will be square and lower triangular.

For DECODE to work, all diagonal elements of $\boldsymbol{G}_3$ must be positive. Some implementations of QR factorization do not do this automatically; if this is the case, we multiply by -1 all columns of $\boldsymbol{G}_3$ that contain a negative diagonal element, as well as the corresponding rows of $\boldsymbol{Q}$.

In Steps 4–6, the input vectors are processed. They are transformed into the coordinate system of $\boldsymbol{G}_3$, decoded, and transformed back again.

If a large set of vectors is to be decoded for the same lattice, Steps 1–3 are, of course, carried out only once for the whole set. In this case, the overall execution time may benefit substantially from an effective but time-consuming reduction method applied in Step 1. To understand precisely what kind of preprocessing would improve the performance of the search algorithm, recall the recursive representation of lattices in (10). An n-dimensional lattice consists of parallel $(n-1)$-dimensional sublattices, translated and stacked on top of each other. This decomposition into sublattices is controlled by the reduction method. Two properties of the decomposition are desirable for a given lattice.

a) The $(n-1)$-dimensional layers should be as far apart as possible. This minimizes the number of layers to be investigated, as only the layers within a certain distance range need to be scanned. As an extreme case, suppose that the spacing between $(n-1)$-dimensional layers is much larger than any other k-dimensional layer spacing in the lattice. Then the closest point will always lie in the closest $(n-1)$-dimensional layer, and the dimensionality of the problem is essentially reduced by one.

b) The zero-dimensional layers (lattice points) should be as densely spaced as possible in the one-dimensional layers (lines). The denser they are, the higher is the probability that the closest lattice point will belong to the closest lattice line. If the one-dimensional spacing is much smaller than all other interlayer distances, then the closest point will always lie in the closest line, so the dimensionality of the problem is essentially reduced by one.

Both observations can, of course, be applied recursively. Thus, high-dimensional layer spacing should be large, while low-dimensional spacing should be small. This suggests two greedy

algorithms: a) sequentially maximizing the distances between k-dimensional layers, starting at $k = n - 1$, and b) minimizing the same distances, starting at $k = 0$.

These two goals are each other's duals in a fairly strict sense. Even though they may appear contradictory, they are, in fact, very similar (cf. [50, pp. 94–98]). To see this, observe that a reduction algorithm can choose the numbers $\{v_{kk}\}$ in many ways for a given lattice, but their product is invariant: it equals the volume of the Voronoi region. Now, a) is solved by maximizing first v_{nn}, then $v_{n-1,\,n-1}$, and so on. Because of the constant product, this procedure forces low values for v_{11}, v_{22}, etc. Thus, a good solution of a) is in general good for b) too. Conversely, b) is solved by first minimizing v_{11}, then v_{22}, and so on, which automatically produces a good basis in the sense of a) as well.

The smallest possible value of v_{11} that can be selected for a given lattice equals the length of the shortest vector in the lattice. (Shortest vector problems can be solved by a variant of the CLOSESTPOINT algorithm, as described in Section VI-A.) On the other hand, the largest possible v_{nn} is the reciprocal of the length of the shortest vector in the dual lattice $\Lambda^{\perp}$, since $(\boldsymbol{G}^{-1})^T$ is a generator matrix for $\Lambda^{\perp}$, provided that $\boldsymbol{G}$ is square. Applying these shortest vector criteria recursively, we conclude that b) is solved optimally by KZ reduction of any basis for the lattice. This follows immediately from the recursive definition of KZ reduction in Section II. Similarly, a) is solved optimally by KZ reduction of a basis for the dual lattice, followed by reversing the order of the rows and transposing the inverse of the resulting matrix (hereafter, we refer to this procedure as *KZ reduction of the dual*). Finally, the LLL reduction yields an approximate (but faster) solution to both a) and b), because of its inherent sorting mechanism.

Our recommendation is to use KZ reduction in applications where the same lattice is to be searched many times, otherwise use LLL. This recommendation is supported by the experimental results in Section VII.

V. COMPLEXITY ANALYSIS

Banihashemi and Khandani [12] observed that the average complexity of a search method for uniformly distributed input vectors[4] is proportional to the volume of the region being searched. They used this observation to assess the complexity of the Kannan algorithm. We adopt the same approach here to analyze the CLOSESTPOINT algorithm and compare it with the Kannan algorithm. A comparison between CLOSESTPOINT and an algorithm based on the Pohst strategy is carried out experimentally in Section VII.

For a given lattice, let $V_k(\rho)$ denote the volume searched in a k-dimensional layer, when ρ is the given upper bound on the attainable distance. Since the CLOSESTPOINT algorithm does not require an initial value for ρ, the desired complexity measure is $V_n(\infty)$.

[4]In this context, a "uniform distribution" is assumed to be uniform over a region large enough to make boundary effects negligible. This is equivalent to a uniform distribution over just one Voronoi region.

Theorem 1: Let $\beta_k = \sqrt{v_{11}^2 + \cdots + v_{kk}^2}$ for $k = 1, \ldots, n$. Then

$$V_n(\infty) \leqslant \prod_{k=1}^{n} \beta_k \tag{16}$$

$$V_n(\infty) \leqslant \left(\frac{2n}{\pi e}\right)^{-n/2} \beta_n^n. \tag{17}$$

Proof: As before, we let ρ_k denote the upper bound used by the CLOSESTPOINT algorithm when searching a k-dimensional layer. In view of (14), we have

$$\rho_{k-1} \leqslant \sqrt{\rho_k^2 - y_k^2}, \qquad \text{for } k = 2, \ldots, n \tag{18}$$

where y_k is the distance accumulated within the k-dimensional layer, as in (11). Combining (11) and (13), we see that y_k varies from at least $-\rho_k$ to at most $+\rho_k$. Thus, expressing $V_k(\rho_k)$ as an integral over $V_{k-1}(\rho_{k-1})$, we obtain the following recursive bound:

$$V_k(\rho_k) \leqslant \int_{-\rho_k}^{\rho_k} V_{k-1}(\rho_{k-1})\, dy, \qquad \text{for } k = 2, \ldots, n. \tag{19}$$

The bounds (16) and (17) follow from this recursion in conjunction with two different bounds on $\rho_1, \ldots, \rho_n$. In either case, we use the initial condition

$$V_1(\rho_1) = 2\rho_1 \tag{20}$$

which is the volume of a line extending from $-\rho_1$ to $+\rho_1$. To derive (16), we first use (18) to transform (19) into the form

$$V_k(\rho) \leqslant \int_{-\rho}^{\rho} V_{k-1}\left(\sqrt{\rho^2 - y^2}\right) dy$$

where the index of ρ has been dropped. Solving this recursion with the initial condition (20) yields

$$V_k(\rho) \leqslant \frac{\pi^{k/2}}{\Gamma(k/2+1)}\, \rho^k, \qquad \text{for } k = 1, \ldots, n. \tag{21}$$

Notice that the right-hand side of (21) is the volume of a k-dimensional sphere of radius ρ.

It is known [10] that for any input vector $\boldsymbol{x}$, the distance to the Babai point in k dimensions is at most $\beta_k/2$, where $\beta_k = (v_{11}^2 + \cdots + v_{kk}^2)^{1/2}$. Since the Babai point is the first lattice point generated by the CLOSESTPOINT algorithm, we have

$$V_n(\infty) = V_n(\beta_n/2) \tag{22}$$

and $\rho_k \leqslant \beta_k/2$ for $k = 1, \ldots, n$. Using this bound on ρ_k in conjunction with the recursion (19), we obtain

$$V_k(\rho_k) \leqslant \prod_{j=1}^{k} \beta_k, \qquad \text{for } k = 1, \ldots, n \tag{23}$$

regardless of the value of ρ_k. This proves (16). Notice that the right-hand side of (23) is the volume of a k-dimensional parallelepiped with sides $\beta_1, \ldots, \beta_k$.

To complete the proof of (17), we observe that by (21) and (22), we have

$$V_n(\infty) \leqslant \frac{\pi^{n/2}\beta_n^n}{2^n \Gamma(n/2+1)} \leqslant \left(\frac{2n}{\pi e}\right)^{-n/2} \beta_n^n \tag{24}$$

where the last inequality follows from $\Gamma(x+1) \geqslant (x/e)^x$, which is the well-known Stirling inequality [29, p. 54]. □

Let K_n denote the volume of the region being searched in the Kannan algorithm for an n-dimensional lattice. Since Kannan [47] focused on proving the existence of an algorithm within a certain complexity bound rather than presenting a single immediately implementable algorithm, there is some ambiguity regarding what exactly is to be meant by "Kannan's algorithm." We here adopt the same interpretation as in [12]. It is shown in [12] that for every lattice, K_n is in the range

$$\prod_{k=1}^{n} \beta_k \leqslant K_n \leqslant \beta_n^n \tag{25}$$

where the lower bound is exact if the sequence $v_{11}, \ldots, v_{nn}$ is increasing and the upper bound is exact if it is decreasing. For a "good" lattice (say, one of the first 48 laminated lattices [23, p. 158]), this sequence generally displays a decreasing trend, although the decrease is not necessarily monotonic [48]. Thus, K_n is often close to the upper bound. On the other hand, the *recursive cube search* algorithm [12], an improved variant of Kannan's algorithm, attains the lower bound in (25) with equality (cf. [12, eq. (19)]).

The CLOSESTPOINT algorithm is faster than the Kannan algorithm for all dimensions and all lattices, since the upper bound (16) coincides with the lower bound (25) for the Kannan algorithm. The magnitude of the gain is suggested by (17). For lattices such that the upper bound in (25) is exact, the CLOSESTPOINT algorithm is faster by at least a factor of $(2n/\pi e)^{n/2}$. Notice that this factor is meant to indicate the asymptotic relation for large n. For low and moderate values of n, the first inequality in (24) yields a significantly better bound.

Also notice that in assessing the volume searched by the CLOSESTPOINT algorithm, the general bound

$$V_n(\infty) \leqslant \frac{\pi^{k/2}\beta_k^k}{2^k\Gamma(k/2+1)} \prod_{j=k+1}^{n} \beta_j, \qquad \text{for } k = 0, 1, \ldots, n$$

may be useful. This bound includes (16) and (17) as two extreme special cases. It follows straightforwardly from (19), (21), and the fact that $\rho_k \leqslant \beta_k/2$ for $k = 1, \ldots, n$.

Banihashemi and Khandani [12] point out that the covering radii of the lattice and its sublattices, if known, can be exploited to reduce the complexity of the Kannan algorithm. This option can be incorporated into the CLOSESTPOINT algorithm as well. However, it is difficult to determine the covering radius of a general lattice. The only known method is the "diamond-cutting" algorithm of [79], which, as detailed in Section VI-C, is confined by memory limitations to low dimensions. If an upper bound on the covering radius for the particular lattice is known, it can be used as well, as proposed in [78]. Unfortunately, even though there exist upper bounds on the *minimal possible* covering radius for packings in a given dimension [23, pp. 39–40], [39, p. 241], no method to upper-bound the covering radius of an arbitrary *given* lattice is known.

VI. MORE LATTICE SEARCH PROBLEMS

Other search problems involving lattices can be solved using modifications and extensions of the CLOSESTPOINT algorithm. These include computing lattice parameters such as the shortest vector, the kissing number, and the Voronoi-relevant vectors. The CLOSESTPOINT algorithm can be also used to perform the key step in the KZ basis reduction.

A. Shortest Vector

Given a lattice $\Lambda \subset \mathbb{R}^m$, the *shortest vector problem* is to find a vector in $\Lambda - \{\mathbf{0}\}$ that has the smallest Euclidean norm. The history of the shortest vector problem is closely interlinked with that of the closest point problem. It has been conjectured in [74] that the shortest vector problem (with $\Lambda \subseteq \mathbb{Z}^m$) is NP-hard, but, in contrast to the closest point problem, this is still not proved. The conjecture of [74] is supported by the result of Ajtai [6], who showed that the shortest vector problem is NP-hard under randomized reductions. Micciancio [54] furthermore proved that finding an approximate solution within any constant factor less than $\sqrt{2}$ is also NP-hard for randomized reductions. It is known [37], [43], however, that the shortest vector problem is not harder than the closest vector problem.

The CLOSESTPOINT algorithm can be straightforwardly modified to solve the shortest vector problem. The idea is to submit $\boldsymbol{x} = \mathbf{0}$ as the input and exclude $\hat{\boldsymbol{x}} = \mathbf{0}$ as a potential output. Algorithmically, the changes needed to convert CLOSESTPOINT into SHORTESTVECTOR are as follows.

1. Omit $\boldsymbol{x}$ as an input to DECODE and CLOSESTPOINT.
2. In CLOSESTPOINT, skip Step 4.
3. In DECODE, replace line 5 with "$\boldsymbol{e}_k := \mathbf{0}$."
4. In DECODE, replace lines 20–22 with:

```
if newdist ≠ 0 then {
    û := u
    bestdist := newdist
    k := k + 1
}
```

In any lattice, there is an even number of shortest vectors, because the lattice is symmetrical with respect to reflection in $\mathbf{0}$. Hence, if $\hat{\boldsymbol{x}}$ is a shortest vector, then so is $-\hat{\boldsymbol{x}}$. A factor of two in computation time can be gained by exploiting this symmetry. This is achieved by rewriting DECODE to scan only half of the candidates $\boldsymbol{u}$ (say, the ones for which the first nonzero component is positive).

Of course, when a KZ-reduced basis is used for the lattice at hand, a shortest vector is directly available as the first basis element, and the SHORTESTVECTOR algorithm becomes trivial. However, one of the main applications of the SHORTESTVECTOR algorithm, at least in our context, is precisely to compute a KZ-reduced basis.

B. Kissing Number

The *kissing number* of a lattice Λ is defined as the number of shortest nonzero vectors in Λ. If the lattice has no regular structure (say, if the basis vectors are drawn randomly from a

continuous distribution), there are typically exactly two shortest nonzero lattice vectors, and the kissing number is 2. In general, to compute the kissing number (say, for a structured lattice), it is essential to use infinite precision: an arbitrarily small perturbation of a generator matrix has the potential of reducing the kissing number to 2, regardless of the original value. However, we do not recommend implementing DECODE using exact arithmetic. The same goal can be achieved far more efficiently by implementing the time-consuming operations, as before, using finite-precision real numbers, followed by an infinite-precision postprocessing stage, whereby a finite set of candidates is evaluated.

The new version of DECODE needs to keep track of a set of potential shortest vectors, not just the single best candidate. A margin of accuracy must be included in the comparisons, to avoid missing some of the shortest vectors due to numerical errors. Thus, the changes needed to convert CLOSESTPOINT into KISSINGNUMBER are as follows.

1. Apply the changes 1–3 of Section VI-A.
2. In DECODE, include "$\widehat{\mathcal{U}} := \varnothing$" among the initial assignments.
3. In DECODE, replace line 11 with:
   if `newdist` $< (1+\epsilon)$`bestdist` **then {**
   where ϵ is a small positive number.
4. In DECODE, replace lines 20 and 21 with:
   if `newdist` $\neq 0$ **then {**
   $\widehat{\mathcal{U}} := \widehat{\mathcal{U}} \cup \{\boldsymbol{u}\}$
   `bestdist` $:= \min($`bestdist`, `newdist`$)$
   }
5. In DECODE, remove line 22.
6. In DECODE, replace $\hat{\boldsymbol{u}}$ in line 28 with $\widehat{\mathcal{U}}$. In CLOSESTPOINT, replace $\hat{\boldsymbol{u}}_3$ in Step 5 with $\widehat{\mathcal{U}}_3$.
7. In CLOSESTPOINT, replace Step 6 with:
   Step 6. Compute the exact value of $||\boldsymbol{u}\boldsymbol{G}_2||$ for all $\boldsymbol{u} \in \widehat{\mathcal{U}}_3$ and return the number of occurrences of the lowest value.

As for the shortest vector problem, a variant of the closest point problem can be formulated that, in case of a tie, returns all the lattice points that have minimum distance to a given input vector, not just one of them. Specifically, CLOSESTPOINT can be converted into ALLCLOSESTPOINTS through the following modifications.

- Apply the changes 2–6 above.
- In CLOSESTPOINT, replace Step 6 with:
  Step 6. Compute the exact value of $||\boldsymbol{u}\boldsymbol{G}_2 - \boldsymbol{x}||$ for all $\boldsymbol{u} \in \widehat{\mathcal{U}}_3$ and call the lowest value γ. Return
  $$\widehat{\mathcal{X}} := \left\{\boldsymbol{u}\boldsymbol{G}_2 : \boldsymbol{u} \in \widehat{\mathcal{U}}_3,\ ||\boldsymbol{u}\boldsymbol{G}_2 - \boldsymbol{x}|| = \gamma\right\}$$

The main application of this algorithm lies in the solution of the next problem.

C. Voronoi-Relevant Vectors

A *facet* is an $(m-1)$-dimensional face of an m-dimensional polytope. The *relevant-vector problem* is to find the facets of the Voronoi region $\Omega(\Lambda, \boldsymbol{0})$ or, in other words, to find a minimal set $\mathcal{N}(\Lambda) \subseteq \Lambda$ for which

$$\Omega(\Lambda, \boldsymbol{0}) = \{\boldsymbol{x} \in \mathbb{R}^m : ||\boldsymbol{x}|| \leqslant ||\boldsymbol{x} - \boldsymbol{c}|| \quad \forall\, \boldsymbol{c} \in \mathcal{N}(\Lambda)\}.$$

The vectors in $\mathcal{N}(\Lambda)$ are called *Voronoi-relevant*, or simply *relevant*. Our method to solve the relevant-vector problem is based upon the following proposition.

Proposition 2: The Voronoi regions of any two distinct lattice points $\boldsymbol{c}_1 \in \Lambda$ and $\boldsymbol{c}_2 \in \Lambda$ share a facet if and only if

$$||\boldsymbol{s} - \boldsymbol{c}_1|| = ||\boldsymbol{s} - \boldsymbol{c}_2|| < ||\boldsymbol{s} - \boldsymbol{c}'|| \tag{26}$$

for all $\boldsymbol{c}' \in \Lambda - \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$, where

$$\boldsymbol{s} \stackrel{\text{def}}{=} \frac{\boldsymbol{c}_1 + \boldsymbol{c}_2}{2}. \tag{27}$$

Proof: It follows from (26) that $\boldsymbol{s} \in \Omega(\Lambda, \boldsymbol{c}_1) \cap \Omega(\Lambda, \boldsymbol{c}_2)$, and $\boldsymbol{s} \notin \Omega(\Lambda, \boldsymbol{c}')$ for all $\boldsymbol{c}' \in \Lambda - \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$. It is known (cf. [23, p. 33]) that if two Voronoi regions Ω_1 and Ω_2 intersect but do not share a facet, then all points in $\Omega_1 \cap \Omega_2$ also belong to some other Voronoi region Ω_3. Hence, the above property of the point $\boldsymbol{s} = (\boldsymbol{c}_1 + \boldsymbol{c}_2)/2$ suffices to establish that $\Omega(\Lambda, \boldsymbol{c}_1)$ and $\Omega(\Lambda, \boldsymbol{c}_2)$ share a facet.

To prove the "only if" part of the proposition, assume that $\Omega(\Lambda, \boldsymbol{c}_1)$ and $\Omega(\Lambda, \boldsymbol{c}_2)$ have a common facet. Let $\boldsymbol{x}$ be any point in the interior of this facet, so that

$$||\boldsymbol{x} - \boldsymbol{c}_1|| = ||\boldsymbol{x} - \boldsymbol{c}_2|| < ||\boldsymbol{x} - \boldsymbol{c}'|| \tag{28}$$

for all $\boldsymbol{c}' \in \Lambda - \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$. In addition to (28), we will make use of the following identity:

$$||\boldsymbol{s} - \boldsymbol{c}||^2 = \frac{||\boldsymbol{x} - \boldsymbol{c}||^2}{2} + \frac{||\boldsymbol{x} - 2\boldsymbol{s} + \boldsymbol{c}||^2}{2} - ||\boldsymbol{x} - \boldsymbol{s}||^2 \tag{29}$$

which holds for any three points $\boldsymbol{s}, \boldsymbol{c}, \boldsymbol{x} \in \mathbb{R}^m$. Now, for all $\boldsymbol{c}' \in \Lambda - \{\boldsymbol{c}_1, \boldsymbol{c}_2\}$ we have

$$\begin{aligned}||\boldsymbol{s} - \boldsymbol{c}_1||^2 - ||\boldsymbol{s} - \boldsymbol{c}'||^2 &= \frac{||\boldsymbol{x} - \boldsymbol{c}_1||^2 - ||\boldsymbol{x} - \boldsymbol{c}'||^2}{2} \\ &\quad + \frac{||\boldsymbol{x} - \boldsymbol{c}_2||^2 - ||\boldsymbol{x} - (\boldsymbol{c}_1 + \boldsymbol{c}_2 - \boldsymbol{c}')||^2}{2} < 0\end{aligned}$$

where the equality follows from (29), while the inequality follows by applying (28) twice. This establishes (26). □

This proposition was proved by Voronoï in a slightly different context [81, vol. 134, pp. 277–278], [23, p. 475], based on a theory by Minkowski [59, pp. 81–85], [60]. Similar properties have been established for the Voronoi regions of binary linear codes [2] and of parallelepipeds [4].

In order to compute $\mathcal{N}(\Lambda)$ for a lattice $\Lambda(\boldsymbol{G})$, we now proceed as follows. Consider a vector $\boldsymbol{z} \in (\mathbb{Z}/2)^n$, and let $\boldsymbol{s} = \boldsymbol{z}\boldsymbol{G}$. It is obvious that any vector $\boldsymbol{s}$ in (27) is of this form. Notice that $\Lambda(\boldsymbol{G})$ is symmetric with respect to reflection in $\boldsymbol{s}$. That is, if $\boldsymbol{s} + \boldsymbol{x}$ is a lattice point, then so is $\boldsymbol{s} - \boldsymbol{x}$.

Although there are infinitely many pairs of lattice points $(\boldsymbol{c}_1, \boldsymbol{c}_2)$ that have $\boldsymbol{s}$ as their midpoint, Proposition 2 implies that at most one such pair can share a facet. A closest point search in the lattice $\Lambda(\boldsymbol{G})$, with $\boldsymbol{s}$ as the input vector, will find the pair, if it exists. Therefore, we evaluate ALLCLOSESTPOINTS$(\boldsymbol{G}, \boldsymbol{s})$, while distinguishing between the following three cases.

Case 1. ALLCLOSESTPOINTS returns one point $\boldsymbol{s} + \boldsymbol{x} \in \Lambda$. Since $\boldsymbol{s} - \boldsymbol{x}$ is also a lattice point at the same distance from $\boldsymbol{s}$, we conclude that $\boldsymbol{x} = \boldsymbol{0}$ and $\boldsymbol{s}$ is itself

a lattice point. Obviously, this happens if and only if $\boldsymbol{z} \in \mathbb{Z}^n$, and no pair of lattice points can satisfy (26) with respect to $\boldsymbol{s}$ in this case.

Case 2. ALLCLOSESTPOINTS returns exactly two lattice points $\boldsymbol{c}_1 = \boldsymbol{s} + \boldsymbol{x}$ and $\boldsymbol{c}_2 = \boldsymbol{s} - \boldsymbol{x}$. Then these points share a facet by Proposition 2. Notice that if $\boldsymbol{c}_1, \boldsymbol{c}_2 \in \Lambda$ share a facet, then so do $\boldsymbol{c}_1 + \boldsymbol{c}'$ and $\boldsymbol{c}_2 + \boldsymbol{c}'$ for all $\boldsymbol{c}' \in \Lambda$. This establishes an equivalence class of pairs of points of Λ that share a facet, whose midpoint is of the form $(\boldsymbol{z} + \boldsymbol{u})\boldsymbol{G}$ for some $\boldsymbol{u} \in \mathbb{Z}^n$. We are interested in only two pairs in this class, namely

$$(\boldsymbol{c}_1 - \boldsymbol{c}_1, \boldsymbol{c}_2 - \boldsymbol{c}_1) = (\boldsymbol{0}, 2\boldsymbol{c}_2 - 2\boldsymbol{s})$$

$$(\boldsymbol{c}_2 - \boldsymbol{c}_2, \boldsymbol{c}_1 - \boldsymbol{c}_2) = (\boldsymbol{0}, 2\boldsymbol{c}_1 - 2\boldsymbol{s}).$$

In other words, the points $2\boldsymbol{c}_1 - 2\boldsymbol{s}$ and $2\boldsymbol{c}_2 - 2\boldsymbol{s}$ are the only Voronoi-relevant points derived from this equivalence class.

Case 3. ALLCLOSESTPOINTS returns four or more lattice points. Then no pair of points can satisfy (26).

The discussion in Cases 1 and 2 shows that in order to determine $\mathcal{N}(\Lambda)$ for a given lattice $\Lambda(\boldsymbol{G})$, it suffices to investigate potential midpoints $\boldsymbol{s}$ in the finite set

$$\mathcal{M}(\boldsymbol{G}) \stackrel{\text{def}}{=} \{\boldsymbol{s} = \boldsymbol{z}\boldsymbol{G} : \boldsymbol{z} \in \{0, 1/2\}^n - \{\boldsymbol{0}\}\}.$$

For each such vector $\boldsymbol{s}$, we can use the ALLCLOSESTPOINTS algorithm to check whether condition (26) of Proposition 2 is satisfied. This leads to the following algorithm.

Algorithm RELEVANTVECTORS($\boldsymbol{G}$)

Input: an $n \times m$ generator matrix $\boldsymbol{G}$.
Output: the set $\mathcal{N}$ of the Voronoi-relevant vectors of $\Lambda(\boldsymbol{G})$.

Step 1. Let $\mathcal{N} := \varnothing$.
Step 2. For all vectors $\boldsymbol{s} \in \mathcal{M}(\boldsymbol{G})$, do:
a) Let $\widehat{\mathcal{X}} :=$ ALLCLOSESTPOINTS($\boldsymbol{G}, \boldsymbol{s}$);
b) If $|\widehat{\mathcal{X}}| = 2$, let $\mathcal{N} := \mathcal{N} \cup \{2\hat{\boldsymbol{x}} - 2\boldsymbol{s} : \hat{\boldsymbol{x}} \in \widehat{\mathcal{X}}\}$.
Step 3. Return $\mathcal{N}$.

Optional optimization includes moving Steps 1–3 of the ALLCLOSESTPOINTS algorithm out of the loop, since all the calls to ALLCLOSESTPOINTS concern the same lattice. Since for each $\boldsymbol{s} \in \mathcal{M}(\boldsymbol{G})$, the lattice is symmetric with respect to reflection in $\boldsymbol{s}$, a factor of two in complexity can be gained through the same symmetry argument as for SHORTESTVECTOR in Section VI-A.

It follows from the preceding discussion that the maximum number of facets that a Voronoi region can have in any n-dimensional lattice is $2|\mathcal{M}(\boldsymbol{G})| = 2^{n+1} - 2$, which was proved by Minkowski in 1897 [60]. Voronoï showed that this number is attained with probability 1 by a lattice whose basis is chosen at random from a continuous distribution [81, vol. 134, pp. 198–211 and vol. 136, pp. 67–70].

Relevant vectors have been determined for many classical lattices [23, Chs. 4 and 21], but we believe that the RELEVANTVECTORS algorithm proposed here is the fastest known in the general case. The only alternative algorithm known to the authors is the "diamond-cutting" algorithm of Viterbo and Biglieri [79], which computes a complete geometrical description of the Voronoi region of any lattice. This description includes all vertices, edges, etc., which evidently includes the information about the relevant vectors. However, using the diamond-cutting algorithm for the sole purpose of determining the relevant vectors is inefficient. Voronoï showed in his classical work [81] that the number of $(n-k)$-dimensional faces of a Voronoi region of an n-dimensional lattice is upper-bounded by

$$(k+1)\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i+1)^n \tag{30}$$

and that there exist lattices whose Voronoi regions attain this number for every k [81, vol. 136, pp. 74–82, 137–143]. One example of such a lattice, given by Voronoï, is the lattice usually denoted by A_n^*, which is the dual of the root lattice A_n [23, p. 115]. Furthermore, the number of $(n-k)$-dimensional faces is lower-bounded by

$$2^k\binom{n}{k}. \tag{31}$$

This can be proved by induction, keeping in mind that the Voronoi region, as well as all its k-faces, are symmetric polytopes. The lower bound (31) is attained for every k by the cubic lattice $\mathbb{Z}^n$. Evaluating (30) and (31) for $k = n, n-1, \ldots$ shows that the number of vertices is between 2^n and $(n+1)!$, inclusively, the number of edges is between $n2^{n-1}$ and $(n/2)(n+1)!$, and so on. This implies that the memory requirements for the diamond-cutting algorithm grow very rapidly with dimension. This property limits the use of the diamond-cutting algorithm to low dimensions, as the authors of [79] themselves point out.

The RELEVANTVECTORS algorithm, on the other hand, uses negligible memory but does not fully determine the Voronoi regions. In those cases where a complete description (vertices, edges, etc.) is desired, we suggest preceding the diamond-cutting algorithm with RELEVANTVECTORS, since the complexity (both time and memory) of the diamond-cutting algorithm can be reduced by incorporating knowledge of the relevant vectors.

D. KZ Reduction

The last problem we deal with here is the *reduction problem*. This is the problem of finding a KZ-reduced basis, which has been already mentioned in Sections II and IV. Theoretical results are available for specific lattices in [48]. Algorithms for general lattices have been proposed by Kannan [47] and by Schnorr [65]. Since KZ reduction essentially consists of solving n shortest vector problems, a closest point algorithm can be used in this context too. In our experiments (see the next section), we have computed KZ-reduced bases using this method.

The general strategy is to find a shortest vector in the lattice, project the lattice onto the hyperplane orthogonal to this vector, and find a KZ-reduced basis of the resulting $(n-1)$-dimensional lattice, recursively. In this application of the SHORTESTVECTOR algorithm, Step 1 is performed using the LLL reduction, since a

KZ reduction is obviously not a usable prerequisite for KZ reduction. The implementation details, which we omit, follow straightforwardly from the definition of KZ reduction in Section II.

E. Closest Point in a Lattice Code

The primary focus of this paper is search problems for lattices viewed as infinite point sets. Under some circumstances, the methods discussed earlier in the paper can be modified to solve search problems for finite subsets of lattices. This has important applications in communications. Specifically, demodulation and quantization both involve finding the closest vector to a given input in a finite point set. One popular method to design such a point set is to form a *lattice code*, which is the intersection of a lattice and a bounded region in $\mathbb{R}^m$. This bounded region is usually called the *support* of the lattice code [35, pp. 470–479], [36].

If a general closest point algorithm for lattices is applied to such a problem, there is a risk that the returned lattice point lies outside the support and hence does not belong to the lattice code. This typically happens when the input vector lies outside the support, but it may also happen in some cases when it lies slightly inside the support boundary.

Several ways to handle this problem have been proposed. If a lattice point outside the support is returned by the closest point algorithm, an obvious option is to declare a failure or erasure, if the application permits this. Otherwise, the algorithm can be modified to disregard such points and output the closest point found in the support, or if no such point is found, to increase the size of the initial search region and try again [78], [80]. Increasing the size repeatedly ensures that the closest point in the lattice code will eventually be found.

Alternatively, the input vector may be projected onto the boundary of the support before the closest point search algorithm is invoked [8], [31], [32], [45]. Quite often, the closest lattice point to the projected input vector belongs to the lattice code and is its closest point to the original input, but this is not always the case. Hence, it might be advantageous to combine this method with increasing the size of the search region, or to project the vector onto a surface slightly inside the support boundary instead. If the input vector is far outside the support region, a much smaller search region needs to be considered around the projected vector in order to find the closest point in the lattice code, compared to the size of the search region without projection.

The previously described methods are applicable for the Kannan, Pohst, and Schnorr–Euchner strategies alike. It can be argued that increasing the size of the initial search region is useless for the Schnorr–Euchner strategy, because its initial value of ρ_n is unbounded. However, we recommend giving ρ_n an explicit finite value in the context of lattice codes, because if for a certain input vector the Babai point lies outside the support (and if the line through the Babai point in the direction of $\boldsymbol{v}_1$ does not pass through any point in the lattice code), then the unmodified version of DECODE will never terminate. To avoid this, line 2 of DECODE should be appropriately modified.

VII. EXPERIMENTS

In this section, we report on experiments with the CLOSESTPOINT algorithm of Section III-B. We evaluate its performance for both low- and high-dimensional lattices. We also compare it with other similar algorithms, and show how the basis for the lattice at hand should be preprocessed in order to achieve the best performance.

A. The Setup

To evaluate the performance of the CLOSESTPOINT algorithm, we must decide what class of lattices to investigate. The closest point search methods studied here are general. Thus, they do not compete well with algorithms specially designed for searching a particular lattice; such algorithms can exploit structure in the lattice and are generally faster (see Section I). Here, we concentrate on experiments with random lattices without any apparent structure that can be exploited in their decoding. However, for comparison, we also include several experiments where the algorithms were applied to classical, highly structured, lattices, such as the Leech lattice in 24 dimensions and the cubic lattice $\mathbb{Z}^n$.

Following the discussion above, we use generator matrices with random elements, drawn from independent and identically distributed zero-mean, unit variance Gaussian distributions. For each point in Figs. 1–3, 50 random matrices are generated, and the mean search time for each matrix is computed by averaging over a large number of random vectors. The exact number of input vectors is dependent on dimension: for large dimensions with long search times the average is computed over 200 vectors for each of the 50 matrices, while for small dimensions the number of vectors is much larger.

Then the median of the average search times for the 50 matrices is computed. Occasionally, a random matrix with very long search times is drawn. Computing the median rather than the mean guarantees that these rare matrices do not totally dominate the average search times. The search times for all the algorithms are averaged using the same matrices and the same set of input vectors. The results are given as average time (in seconds), using a DELL computer based upon a 733-MHz Pentium III processor, with Visual C++ running under Windows XP.

The random vectors were drawn according to a uniform distribution. Conway and Sloane [22] report on a method to generate uniform data within a Voronoi region, which is equivalent to generating data uniformly distributed over a infinite-sized region. Uniform data is a reasonable assumption for applications such as source coding and cryptography. In channel coding applications, a more reasonable assumption is a Gaussian distribution around a lattice point, but such experiments have not been performed here.

B. The Preprocessing

An important question for a closest point algorithm is whether the performance can be improved by preprocessing the generator matrix. Since the preprocessing needs to be performed only once, while the processed basis is typically used many times (in most communication applications), it is usually worthwhile to invoke a good preprocessing procedure. In Section IV, three dif-

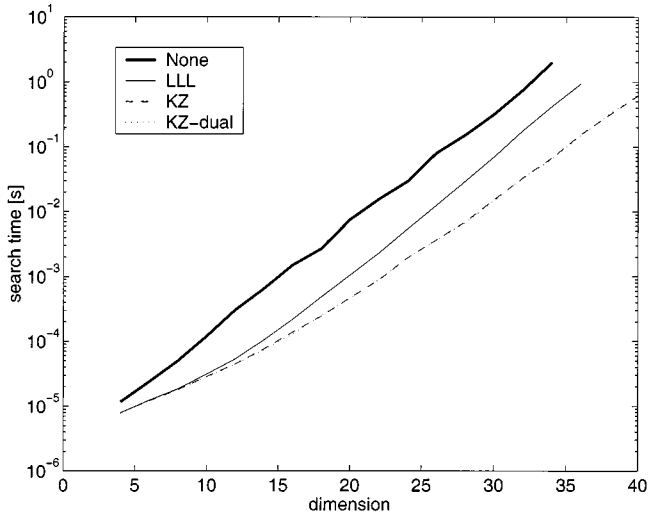

Fig. 1. Comparison of average search times among different reduction methods for preprocessing of the generator matrix.

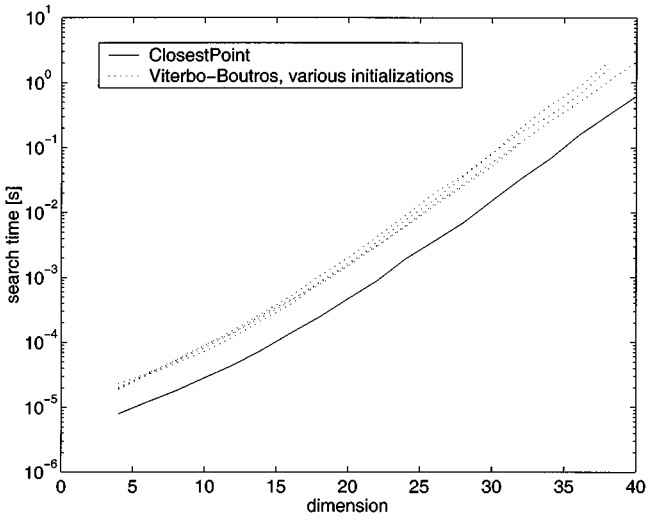

Fig. 2. Comparison of the average search times for the CLOSESTPOINT algorithm and the Viterbo–Boutros algorithm.

ferent preprocessing strategies were discussed: LLL reduction, KZ reduction, and KZ reduction of the dual. All of these strategies basically aim to find as short and as orthogonal basis vectors as possible. Here, we present experiments designed to find the best of these reduction methods.

In Fig. 1, the simulation results for the three reduction methods are given (the time needed for the reduction itself is not included in these results). We see that performance can be improved significantly by selecting a good preprocessor. The best methods in our study are the ones based on the two KZ reductions; as expected, there is almost no difference between the KZ reduction and the KZ reduction of the dual. For high dimensions ($30+$), the KZ reductions lower the average search times by almost two orders of magnitude, as compared to unreduced bases, and by about one order of magnitude as compared to the LLL reduction. On the other hand, up to about 10–15 dimensions, the polynomial-time LLL reduction gives good results.

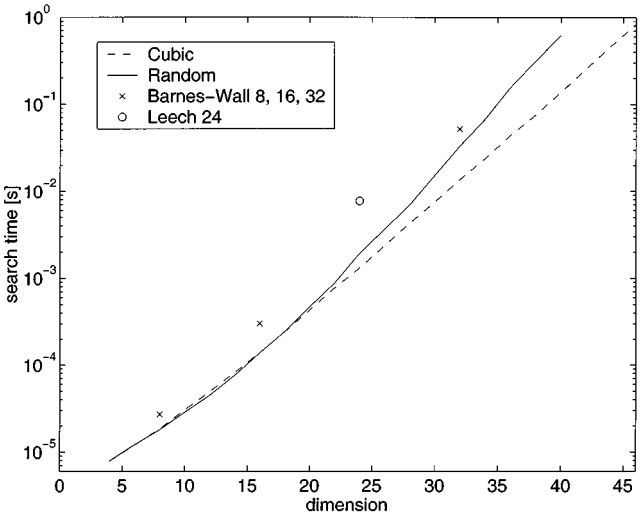

Fig. 3. Average search times for classical and random lattices.

C. Comparison With Other Algorithms

To assess the performance of the CLOSESTPOINT algorithm, we have also implemented an algorithm described by Viterbo and Boutros in [80], which is based on the Pohst strategy. The Viterbo–Boutros algorithm requires an initial bound on the attainable distance (see Section III-A). A natural choice is the covering radius of the lattice, but it is not clear how to compute the covering radius for random lattices. Viterbo [77] suggests to use the length of the shortest basis vector as an initial guess. If no lattice point is found within this distance from the input vector, the distance is multiplied by some factor greater than 1, and the search is repeated. We have performed some experiments using factors between 1.1 and 1.6. We have also used the distance to the Babai point as an initial distance bound, thereby ensuring that at least one point is found within the distance. The CLOSESTPOINT algorithm needs no initial bound for the distance; the Babai point is by default the first point examined by this algorithm.

In Fig. 2, the average time for a single closest point search operation is plotted as a function of dimension for the CLOSESTPOINT and the Viterbo–Boutros algorithms (with several values for the initial distance bound). For both algorithms, KZ reduction was first applied to the generator matrices. We see that the CLOSESTPOINT algorithm is faster for all tested dimensions, by a factor of 2.5–3 in our implementation.

D. Comparison With Classical Lattices

To further illustrate the performance of the CLOSESTPOINT algorithm, we evaluate its performance for classical lattices, and compare it with the performance for random matrices (chosen from an independent and identically distributed Gaussian source). In Fig. 3, the average search times for random lattices and for the cubic lattice $\mathbb{Z}^n$ are plotted as a function of dimension, together with the search times for the Leech lattice in 24 dimensions, and for the Barnes–Wall lattices in dimensions 8, 16, and 32. For the classical lattices just as for

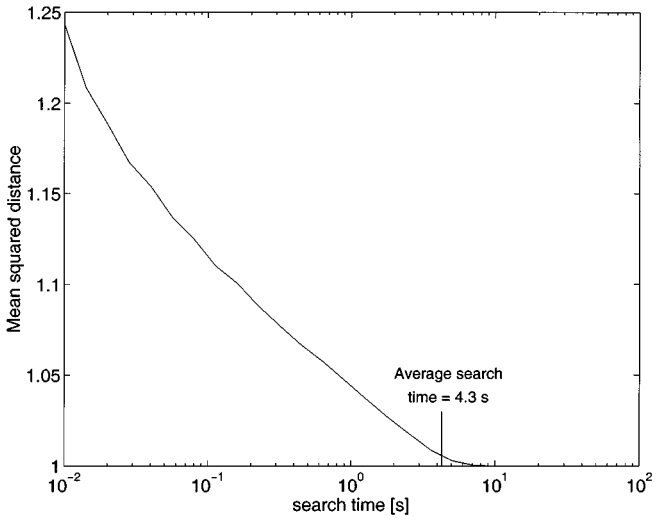

Fig. 4. Normalized mean squared distance as a function of allowed search time, when the search is aborted before the optimal point is found. The Babai point had a normalized mean squared distance of 1.49 for this 45-dimensional example.

random lattices, KZ reduction leads to faster search times, and is therefore applied before the experiments.

We see that although the search times for the classical, highly structured, lattices are slightly higher, the general curve is about the same as that for random lattices. This is the strength as well as the weakness of search algorithms of this type: they do not rely on any particular structure.

E. Suboptimal Search

The search algorithms studied here always return a lattice point that is closest to the input point. However, in certain applications (e.g., source coding), it may be necessary to abort the search before the closest point has been found. Therefore, we have included experiments where the CLOSESTPOINT algorithm is aborted after a given time. The measure of performance in these experiments is the *mean squared distance* to the point produced by an aborted algorithm.

In Fig. 4, the ratio between the suboptimal and the optimal mean squared distances is given for a 45-dimensional example, as a function of the time allotted for the search. From this figure, we see that the CLOSESTPOINT algorithm quickly finds lattice points fairly close to the optimal one.

We see that if a 10% higher mean squared distance than the optimal can be tolerated, then the CLOSESTPOINT algorithm is approximately 40 times faster than if the optimal point is required. We only report results for a single 45-dimensional example, but the general conclusion is the same for all tested dimensions and lattices. If the search is aborted before the optimal point is found, considerable time savings can be achieved at the cost of a slightly increased mean squared distance. Note that the good result relies on the layers being searched according to (13); if the layers are searched according to (15), the convergence is considerably slower.

ACKNOWLEDGMENT

The authors gratefully acknowledge helpful comments by Daniele Micciancio, who brought several recent references to their attention. They also thank Emanuele Viterbo and Joseph Boutrous for valuable suggestions, especially regarding optimization of the Viterbo–Boutros algorithm.

REFERENCES

[1] L. Afflerbach and H. Grothe, "Calculation of Minkowski-reduced lattice bases," *Computing*, vol. 35, no. 3–4, pp. 269–276, 1985.
[2] E. Agrell, "On the Voronoi neighbor ratio for binary linear block codes," *IEEE Trans. Inform. Theory*, vol. 44, pp. 3064–3072, Nov. 1998.
[3] E. Agrell and T. Eriksson, "Optimization of lattices for quantization," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1814–1828, Sept. 1998.
[4] E. Agrell and T. Ottosson, "ML optimal CDMA multiuser receiver," *Electron. Lett.*, vol. 31, pp. 1554–1555, Aug. 1995.
[5] M. Ajtai, "Generating hard instances of lattice problems," in *Proc. 28th Annu. ACM Symp. Theory of Computing*, Philadelphia, PA, May 1996, pp. 99–108.
[6] ——, "The shortest vector problem in L_2 is NP-hard for randomized reductions," in *Proc. 30th Annu. ACM Symp. Theory of Computing*, Dallas, TX, May 1998, pp. 193–203.
[7] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. 29th Annu. ACM Symp. Theory of Computing*, El Paso, TX, 1997, pp. 284–293.
[8] M. Antonini, M. Barlaud, and T. Gaidon, "Adaptive entropy constrained lattice vector quantization for multiresolution image coding," *Proc. SPIE*, pt. 2, vol. 1818, pp. 441–457, Nov. 1992.
[9] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, "The hardness of approximate optima in lattices, codes, and systems of linear equations," *J. Comput. Syst. Sci.*, vol. 54, pp. 317–331, Apr. 1997.
[10] L. Babai, "On Lovász' lattice reduction and the nearest lattice point problem," *Combinatorica*, vol. 6, no. 1, pp. 1–13, 1986.
[11] A. H. Banihashemi and I. F. Blake, "Trellis complexity and minimal trellis diagrams of lattices," *IEEE Trans. Inform. Theory*, vol. 44, pp. 1829–1847, Sept. 1998.
[12] A. H. Banihashemi and A. K. Khandani, "On the complexity of decoding lattices using the Korkine–Zolotarev reduced basis," *IEEE Trans. Inform. Theory*, vol. 44, pp. 162–171, Jan. 1998.
[13] I. F. Blake, "Lattices and cryptography," in *Codes, Graphs and Systems*, R. E. Blahut and R. Kötter, Eds. Norwell, MA: Kluwer, 2002, pp. 317–332.
[14] J. Blömer, "Closest vectors, successive minima, and dual HKZ-bases of lattices," in *Proc. Int. Colloq. Automata, Languages and Programming*, U. Montanari, J. D. P. Rolim, and E. Welzl, Eds. Geneva, Switzerland, July 2000, pp. 248–259.
[15] J. Boutros, E. Viterbo, C. Rastello, and J.-C. Belfiore, "Good lattice constellations for both Rayleigh fading and Gaussian channels," *IEEE Trans. Inform. Theory*, vol. 42, pp. 502–518, Mar. 1996.
[16] L. Brunel and J. Boutros, "Euclidean space lattice decoding for joint detection in CDMA systems," in *Proc. Int. Workshop Information Theory*, Kruger Park, South Africa, June 1999, p. 129.
[17] ——, "Lattice decoding for joint detection in direct sequence CDMA systems," *IEEE Trans. Inform. Theory*, 2002, to be published.
[18] J. W. S. Cassels, *An Introduction to the Geometry of Numbers*. Berlin, Germany: Springer, 1959.
[19] I. V. L. Clarkson, "Frequency estimation, phase unwrapping, and the nearest lattice point problem," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, Phoenix, AZ, Mar. 1999, pp. 1609–1612.
[20] H. Cohen, *A Course in Computational Algebraic Number Theory*. Berlin, Germany: Springer-Verlag, 1993.
[21] J. H. Conway and N. J. A. Sloane, "A fast encoding method for lattice codes and quantizers," *IEEE Trans. Inform. Theory*, vol. IT-29, pp. 820–824, Nov. 1983.
[22] ——, "On the Voronoi regions of certain lattices," *SIAM J. Algebraic Discr. Methods*, vol. 5, pp. 294–305, Sept. 1984.
[23] ——, *Sphere Packings, Lattices and Groups*, 3rd ed. New York: Springer-Verlag, 1999.
[24] R. R. Coveyou and R. D. MacPherson, "Fourier analysis of uniform random number generators," *J. Assoc. Comput. Mach.*, vol. 14, pp. 100–119, Jan. 1967.
[25] O. Damen, A. Chkeif, and J.-C. Belfiore, "Lattice code decoder for space-time codes," *IEEE Commun. Lett.*, vol. 4, pp. 161–163, May 2000.
[26] U. Dieter, "How to calculate shortest vectors in a lattice," *Math. of Comput.*, vol. 29, pp. 827–833, July 1975.

[27] I. Dinur, G. Kindler, R. Raz, and S. Safra. (2002) An improved lower bound for approximating CVP. Preprint. [Online]. Available: http://www.math.ias.edu/~iritd

[28] P. Erdös, P. M. Gruber, and J. Hammer, *Lattice Points*. Harlow, U.K./ New York: Longman/Wiley, 1989.

[29] W. Feller, *An Introduction to Probability Theory and its Applications*, 3rd ed. New York: Wiley, 1968, vol. 1.

[30] U. Fincke and M. Pohst, "Improved methods for calculating vectors of short length in a lattice, including a complexity analysis," *Math. of Comput.*, vol. 44, pp. 463–471, Apr. 1985.

[31] T. R. Fischer, "A pyramid vector quantizer," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 568–583, July 1986.

[32] ——, "Geometric source coding and vector quantization," *IEEE Trans. Inform. Theory*, vol. 35, pp. 137–145, Jan. 1989.

[33] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268–278, Mar. 1973.

[34] ——, "Coset codes—Part II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152–1187, Sept. 1988.

[35] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Boston, MA: Kluwer, 1992.

[36] J. D. Gibson and K. Sayood, "Lattice quantization," in *Advances in Electronics and Electron Physics*, P. W. Hawkes, Ed. Boston, MA: Academic, 1988, vol. 72, pp. 259–330.

[37] O. Goldreich, D. Micciancio, S. Safra, and J.-P. Seifert, "Approximating shortest lattice vectors is not harder than approximating closest lattice vectors," *Inform. Processing Lett.*, vol. 71, pp. 55–61, July 1999.

[38] O. Goldreich, D. Ron, and M. Sudan, "Chinese remaindering with errors," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1330–1338, July 2000.

[39] P. M. Gruber and C. G. Lekkerkerker, *Geometry of Numbers*. Amsterdam, The Netherlands: North-Holland, 1987.

[40] V. Guruswami, A. Sahai, and M. Sudan, "'Soft-decision' decoding of Chinese remainder codes," in *Proc. 41st Annu. Symp. Found. Computer Science*, Redondo Beach, CA, Nov. 2000, pp. 159–168.

[41] W. W. Hager, *Applied Numerical Linear Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[42] B. Helfrich, "Algorithms to construct Minkowski reduced and Hermite reduced lattice bases," *Theor. Comput. Sci.*, vol. 41, no. 2–3, pp. 125–139, 1985.

[43] M. Henk, "Note on shortest and nearest lattice vectors," *Inform. Processing Lett.*, vol. 61, pp. 183–188, 1997.

[44] C. Hermite, "Extraits de lettres à M. Jacobi sur différents objets de la théorie des nombres" (in French), *J. Reine und Angewandte Math.*, vol. 40, no. 3–4, pp. 261–315, 1850.

[45] D. G. Jeong and J. D. Gibson, "Uniform and piecewise uniform lattice vector quantization for memoryless Gaussian and Laplacian sources," *IEEE Trans. Inform. Theory*, vol. 39, pp. 786–804, May 1993.

[46] R. Kannan, "Improved algorithms for integer programming and related lattice problems," in *Proc. ACM Symp. Theory of Computing*, Boston, MA, Apr. 1983, pp. 193–206.

[47] ——, "Minkowski's convex body theorem and integer programming," *Math. Oper. Res.*, vol. 12, pp. 415–440, Aug. 1987.

[48] A. K. Khandani and M. Esmaeili, "Successive minimization of the state complexity of the self-dual lattices using Korkine–Zolotarev reduced basis," Dept. Elec. Comput. Eng., Univ. of Waterloo, Waterloo, ON, Canada, Tech. Rep. UW-E&CE#97-01, Jan. 1997.

[49] P. Klein, "Finding the closest lattice vector when it's unusually close," in *Proc. 11th ACM-SIAM Symp. Discrete Algorithms*, San Francisco, CA, Jan. 2000, pp. 937–941.

[50] D. E. Knuth, *The Art of Computer Programming*, 2nd ed. Reading, MA: Addison-Wesley, 1981, vol. 2.

[51] A. Korkine and G. Zolotareff, "Sur les formes quadratiques" (in French), *Math. Annalen*, vol. 6, pp. 366–389, 1873.

[52] C. Lamy and J. Boutros, "On random rotations diversity and minimum MSE decoding of lattices," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1584–1589, July 2000.

[53] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Math. Annalen*, vol. 261, pp. 515–534, 1982.

[54] D. Micciancio, "The shortest vector in a lattice is hard to approximate to within some constant," in *Proc. 39th Annu. Symp. Foundations of Computer Science*, Palo Alto, CA, Nov. 1998, pp. 92–98.

[55] ——, "The hardness of the closest vector problem with preprocessing," *IEEE Trans. Inform. Theory*, vol. 47, pp. 1212–1215, Mar. 2001.

[56] H. Minkowski, "Sur la réduction des formes quadratiques positives quaternaires" (in French), *C. R. Académie des Sciences*, vol. 96, pp. 1205–1210, 1883. Also in *Gesammelte Abhandlungen von Hermann Minkowski* (D. Hilbert, Ed.). Leipzig, Berlin, Germany: Teubner, vol. 1, 1911, pp. 145–148 (in French).

[57] ——, "Über positive quadratische Formen" (in German), *J. Reine und Angewandte Math.*, vol. 99, no. 1, pp. 1–9, 1886. Also in *Gesammelte Abhandlungen von Hermann Minkowski* (D. Hilbert, Ed.). Leipzig, Berlin, Germany: Teubner, vol. 1, 1911, pp. 149–156 (in German).

[58] ——, "Zur Theorie der positiven quadratischen Formen" (in German), *J. Reine und Angewandte Math.*, vol. 101, no. 3, pp. 196–202, 1887. Also in *Gesammelte Abhandlungen von Hermann Minkowski* (D. Hilbert, Ed.). Leipzig, Berlin, Germany: Teubner, vol. 1, 1911, pp. 212–218 (in German).

[59] ——, *Geometrie der Zahlen* (in German), Leipzig, Germany, 1896.

[60] ——, "Allgemeine Lehrsätze über die konvexen Polyeder" (in German), *Nachrichten der K. Gesellschaft der Wissenschaften zu Göttingen. Mathematisch-physikalische Klasse*, pp. 198–219, 1897. Also in *Gesammelte Abhandlungen von Hermann Minkowski* (D. Hilbert, Ed.). Leipzig, Berlin, Germany: Teubner, vol. 2, 1911, pp. 103–121 (in German).

[61] ——, "Diskontinuitätsbereich für arithmetische Äquivalenz" (in German), *J. Reine und Angewandte Math.*, vol. 129, no. 3–4, pp. 220–274, 1905. Also in *Gesammelte Abhandlungen von Hermann Minkowski* (D. Hilbert, Ed.). Leipzig, Berlin, Germany: Teubner, vol. 2, 1911, pp. 53–100 (in German).

[62] W. H. Mow, "Maximum likelihood sequence estimation from the lattice viewpoint," *IEEE Trans. Inform. Theory*, vol. 40, pp. 1591–1600, Sept. 1994.

[63] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *ACM SIGSAM Bull.*, vol. 15, pp. 37–44, Feb. 1981.

[64] S. S. Ryshkov and E. P. Baranovskii, "Classical methods in the theory of lattice packings" (in Russian), *Usp. Mat. Nauk*, vol. 34, pp. 3–64, July–Aug. 1979. Translated into English in *Russ. Math. Surv.*, vol. 34, no. 4, pp. 1–68, 1979.

[65] C. P. Schnorr, "A hierarchy of polynomial time lattice basis reduction algorithms," *Theor. Comput. Sci.*, vol. 53, no. 2–3, pp. 201–224, 1987.

[66] ——, "A more efficient algorithm for lattice basis reduction," *J. Algorithms*, vol. 9, pp. 47–62, Mar. 1988.

[67] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Programming*, vol. 66, pp. 181–191, 1994.

[68] C. P. Schnorr and H. H. Hörner, "Attacking the Chor–Rivest cryptosystem by improved lattice reduction," in *Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 1995, vol. 921, pp. 1–12.

[69] A. Schönhage, "Factorization of univariate integer polynomials by diophantine approximation and an improved basis reduction algorithm," in *Proc. Colloq. Automata, Languages and Programming*, J. Paredaens, Ed. Antwerp, Belgium, July 1984, pp. 436–447.

[70] J. Stern, "Lattices and cryptography: An overview," in *Public Key Cryptography*, H. Imai and Y. Zheng, Eds. Yokohama, Japan, Feb. 1998, pp. 50–54.

[71] G. Strang, *Linear Algebra and Its Applications*, 3rd ed. San Diego, CA: Harcourt Brace Jovanovich, 1988.

[72] V. Tarokh and I. F. Blake, "Trellis complexity versus the coding gain of lattices, Parts I and II," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1796–1816, Nov. 1996.

[73] V. Tarokh and A. Vardy, "Upper bounds on trellis complexity of lattices," *IEEE Trans. Inform. Theory*, vol. 43, pp. 1294–1300, July 1997.

[74] P. van Emde Boas, "Another NP-complete partition problem and the complexity of computing short vectors in a lattice," Mathematisch Instituut, Amsterdam, The Netherlands, Rep. 81-04, Apr. 1981.

[75] A. Vardy and Y. Be'ery, "Maximum-likelihood decoding of the Leech lattice," *IEEE Trans. Inform. Theory*, vol. 39, pp. 1435–1444, July 1993.

[76] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inform. Theory*, vol. IT-13, pp. 260–269, Apr. 1967.

[77] E. Viterbo, private communication, Jan. 2002.

[78] E. Viterbo and E. Biglieri, "A universal decoding algorithm for lattice codes," in *Proc. GRETSI*, Juan-les-Pins, France, Sept. 1993, pp. 611–614.

[79] ——, "Computing the Voronoi cell of a lattice: The diamond-cutting algorithm," *IEEE Trans. Inform. Theory*, vol. 42, pp. 161–171, Jan. 1996.

[80] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1639–1642, July 1999.

[81] G. Voronoï, "Nouvelles applications des paramètres continus à la théorie des formes quadratiques," *J. Reine und Angewandte Math.*, vol. 133, pp. 97–178, 1908. Also, vol. 134, pp. 198–287, 1908; and vol. 136, pp. 67–181, 1909 (in French).