Linear-Time Nearest Point Algorithms for Coxeter Lattices

Robby G. McKilliam, *Student Member, IEEE*, Warren D. Smith, and I. Vaughan L. Clarkson, *Senior Member, IEEE*

***Abstract*—The Coxeter lattices are a family of lattices containing many of the important lattices in low dimensions. This includes A_n, E_7, E_8 and their duals A_n^*, E_7^*, and E_8^*. We consider the problem of finding a nearest point in a Coxeter lattice. We describe two new algorithms, one with worst case arithmetic complexity $O(n \log n)$ and the other with worst case complexity $O(n)$ where n is the dimension of the lattice. We show that for the particular lattices A_n and A_n^* the algorithms are equivalent to nearest point algorithms that already exist in the literature.**

***Index Terms*—Channel coding, lattice theory, nearest point algorithm, quantization.**

I. Introduction

THE study of point lattices is of great importance in several areas of number theory, particularly the studies of quadratic forms, the geometry of numbers and simultaneous Diophantine approximation, and also to the practical engineering problems of quantization and channel coding. They are also important in studying the sphere packing problem and the kissing number problem [1], [2]. Lattices have also found application in cryptography [3], [4], communications systems using multiple antannas [5], [6] and estimation theory [7]–[11].

A *lattice* L is a set of points in $\mathbb{R}^n$ such that

$$L = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{x} = \mathbf{B}\mathbf{w}, \mathbf{w} \in \mathbb{Z}^n\}$$

where $\mathbf{B}$ is termed the generator (or basis) matrix. We will write vectors and matrices in bold font. The ith element in a vector is denoted by a subscript: x_i. The generator matrix for a lattice is not unique. Let $\mathbf{M}$ be an $n \times n$ matrix with integer elements such that $\det(\mathbf{M}) = \pm 1$. $\mathbf{M}$ is called a *unimodular* matrix. Then both $\mathbf{B}$ and $\mathbf{BM}$ are generator matrices for the lattice L.

Lattices are equivalent under scaling, rotation and reflection. A lattice L with generator matrix $\mathbf{B}$ and a lattice $\hat{L}$ with generator lattice $\hat{\mathbf{B}}$ are equivalent, or *isomorphic*, if and only if

$$\mathbf{B} = \alpha \mathbf{R} \hat{\mathbf{B}} \mathbf{M}$$

where $\alpha > 0$ is real, $\mathbf{R}$ is an orthogonal matrix and $\mathbf{M}$ is unimodular. We write $L \simeq \hat{L}$.

The *Voronoi region* or *nearest-neighbour region*, denoted $\mathrm{Vor}(L)$ for a lattice L, is the subset of $\mathbb{R}^n$ such that, with respect to a given norm, all points in $\mathrm{Vor}(L)$ are *nearer* to the origin than to any other point in L. In this paper we will always assume that the Euclidean norm is used. The Voronoi region is an n-dimensional polytope [2]. Given some lattice point $\mathbf{x} \in L$ we will write $\mathrm{Vor}(L) + \mathbf{x}$ to denote the Voronoi region centered around the lattice point $\mathbf{x}$. It follows that $\mathrm{Vor}(L) + \mathbf{x}$ is the subset of $\mathbb{R}^n$ that is nearer to $\mathbf{x}$ than any other lattice point in L.

Given $\mathbf{y} \in \mathbb{R}^n$ and some lattice L whose lattice points lie in $\mathbb{R}^n$, the nearest lattice point problem is to find a lattice point $\mathbf{x} \in L$ such that the Euclidean distance between $\mathbf{y}$ and $\mathbf{x}$ is minimized. We use the notation $\mathrm{NearestPt}(\mathbf{y}, L)$ to denote the nearest point to $\mathbf{y}$ in the lattice L. It follows from the definition of the Voronoi region that[1]

$$\mathbf{x} = \mathrm{NearestPt}(\mathbf{y}, L) \Leftrightarrow \mathbf{y} \in \mathrm{Vor}(L) + \mathbf{x}.$$

The nearest lattice point problem has significant practical application. If the lattice is used for vector quantization then the nearest lattice point corresponds to the minimum-distortion point. If the lattice is used as a code for a Gaussian channel, then the nearest lattice point corresponds to maximum likelihood decoding [12].

In general the nearest lattice point problem is known to be NP-hard [13], [14]. Nevertheless, algorithms exist that can compute the nearest lattice point in reasonable time if the dimension is small [15]–[17]. One such algorithm introduced by Pohst [17] in 1981 was popularized in signal processing and communications fields by Viterbo and Boutros [16] and has since been called the *sphere decoder*. Kannan suggested a different approach that is known to be asymptotically faster than the sphere decoder [18]. For specific lattices the nearest point problem can be considerably easier and for many classical lattices fast nearest point algorithms are known [1], [2], [12], [19]–[21].

In this paper we discuss fast nearest point algorithms for a special family of lattices called the *Coxeter lattices*. Conway and Sloane [12], [19] have described simple nearest point algorithms for the Coxeter lattices that require $O(n^2)$ arithmetic operations in the worst case, where n is the dimension of the lattice. In this paper we describe two new algorithms, one that requires $O(n \log n)$ operations and another that requires $O(n)$ operations in the worst case.

[1]There is a slight technical deficiency here. We actually require to define half of the faces of $\mathrm{Vor}(L)$ to be closed and half to be open. Specifically, if $\mathbf{x}$ is on the boundary of $\mathrm{Vor}(L)$ then only one of $\mathbf{x}$ or $-\mathbf{x}$ is in $\mathrm{Vor}(L)$. Ties in $\mathrm{NearestPt}(\mathbf{y}, L)$ can then be broken accordingly.

Manuscript received March 04, 2009; revised October 02, 2009. Current version published March 10, 2010. The work of R. G. McKilliam was supported in part by a scholarship from the Wireless Technologies Laboratory, CSIRO ICT Centre, Sydney, Australia.

R. G. McKilliam and I. V. L. Clarkson are with the School of Information Technology and Electrical Engineering, The University of Queensland, 4072, Australia (e-mail: robertm@itee.uq.edu.au; v.clarkson@uq.edu.au).

W. D. Smith is with the Center for Range Voting, Stony Brook, NY 11790 USA (e-mail: warren.wds@gmail.com).

Communicated by E. Viterbo, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2009.2039090

The Coxeter lattices, denoted $A_{n/m}$, are a family of lattices first described by Coxeter [22], [23]

$$A_{n/m} = \left\{ \mathbf{Q}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^{n+1},\, \mathbf{x}'\mathbf{1} \bmod m = 0 \right\} \tag{1}$$

where $\mathbf{Q}$ is the orthogonal projection matrix

$$\mathbf{Q} = \left(\mathbf{I} - \frac{\mathbf{1}\mathbf{1}'}{n+1} \right) \tag{2}$$

$\mathbf{I}$ is the $(n+1)\times(n+1)$ identity matrix, $\mathbf{1} = [1,1,1,\ldots]'$ and $'$ indicates the vector or matrix transpose. If m does not divide $n+1$ then $A_{n/m} = A_{n/\gcd(m,n+1)}$ where $\gcd(\cdot,\cdot)$ denotes the greatest common divisor. Hence, in the sequel, we assume that m divides $n+1$.

A simple geometric description of $A_{n/m}$ is to consider the subset consisting of the points of $\mathbb{Z}^{n+1}$ whose coordinate-sum is divisible by m. This subset consists of points that lie in "layers" parallel to the hyperplane orthogonal to $\mathbf{1}$. By orthogonally projecting the subset along $\mathbf{1}$ we obtain a set of points equivalent to the n-dimensional lattice $A_{n/m}$. The family of Coxeter lattices contains many of the important lattices in low dimension. The family is related to the well-studied root lattice A_n and its dual lattice A_n^*. When $m = 1$

$$A_{n/1} = A_n^* = \{ \mathbf{Q}\mathbf{x} \mid \mathbf{x} \in \mathbb{Z}^{n+1} \} \tag{3}$$

and when $m = n+1$

$$A_{n/n+1} = A_n = \left\{ \mathbf{x} \in \mathbb{Z}^{n+1} \mid \mathbf{x}'\mathbf{1} = 0 \right\}. \tag{4}$$

It follows that $A_n \subseteq A_{n/m} \subseteq A_n^*$ [2], [23]. Note that $A_{n/m} \subset A_{n/k}$ whenever k divides m and, therefore

$$\mathrm{Vor}(A_{n/k}) \subset \mathrm{Vor}(A_{n/m}). \tag{5}$$

Other isomorphisms exist: $A_{8/3} \simeq E_8 \simeq E_8^*$, $A_{7/4} \simeq E_7$ and $A_{7/2} \simeq E_7^*$. Of significant practical interest is the lattice $E_8 \simeq A_{8/3}$. Due to its excellent packing and quatisizing properties E_8 has found applications to trellis codes [24]–[27] and vector quantization [2], [28], [29]. The particular representation of E_8 as $A_{8/3}$ was used by Secord and deBuda to create a code with a power spectrum that is zero at dc [30].

The lattice $A_n^* \simeq A_{n/1}$ is also of practical interest. It gives the thinnest sphere-covering in all dimensions up to 8 [2] and has found application in statistical estimation problems including period estimation from sparse timing data [9], [11], frequency estimation [8], [31], direction of arrival estimation [32], and noncoherent detection [10].

The paper is organized as follows. Section II describes a log-linear-time nearest point algorithm for $A_{n/m}$. This algorithm is a generalization of a nearest point algorithm for A_n^* that was derived in [20]. Section III improves this to worst case linear-time. The speedup employs both a partial sorting procedure called a bucket sort [33] and also the linear-time Rivest-Tarjan selection algorithm [34]–[37]. In Section IV, we show that for the specfic cases of A_n and A_n^* the new nearest point algorithms are equivalent to nearest point algorithms that already exist in the literature [2], [20], [21]. In Section V, we review a simple nearest point algorithm for $A_{n/m}$ based on translates of the lattice A_n. This algorithm was previously described by Conway and Sloane [12], [19]. The algorithm requires $O(n^2)$ arithmetic operations in the worst case. In Section VI, we evaluate the practical computational performance of the algorithms.

II. Log-Linear-Time Algorithm

In this section, we describe a nearest point algorithm for $A_{n/m}$ that requires $O(n \log n)$ operations in the worst case. This algorithm is a generalization of the nearest point algorithm for A_n^* described in [20]. To describe the algorithm we first require to derive some properties of the Voronoi region of $A_{n/m}$. This is done in Lemmata 1 and 2. We first require the follow definitions.

Let H be the hyperplane in $\mathbb{R}^{n+1}$ orthogonal to $\mathbf{1}$. H is known as the *zero-sum-plane*. For some lattice L, we will use the notation $\mathrm{Vor}_H(L)$ to denote the region $\mathrm{Vor}(L) \cap H$. For example, $\mathrm{Vor}_H(A_n)$ is the cross section of $\mathrm{Vor}(A_n)$ lying in the hyperplane H. Given some region $R \subset H$ we define the n-volume of R as $\mathrm{vol}_H(R)$. For example, the n-volume of $\mathrm{Vor}_H(A_n)$ is denoted by $\mathrm{vol}_H(\mathrm{Vor}_H(A_n))$.

Given a set of n-dimensional vectors S and suitable matrix $\mathbf{M}$ we will write $\mathbf{M}S$ to denote the set with elements $\mathbf{M}\mathbf{s}$ for all $\mathbf{s} \in S$. For example, $\mathbf{Q}\,\mathrm{Vor}(\mathbb{Z}^{n+1})$ denotes the region of space that results from projecting $\mathrm{Vor}(\mathbb{Z}^{n+1})$ onto the hyperplane H.

Lemma 1: The projection of $\mathrm{Vor}(\mathbb{Z}^{n+1})$ into H is a subset of $\mathrm{Vor}(A_n) \cap H$. That is

$$\mathbf{Q}\,\mathrm{Vor}(\mathbb{Z}^{n+1}) \subseteq \mathrm{Vor}_H(A_n).$$

Proof: Let $\mathbf{y} \in \mathrm{Vor}(\mathbb{Z}^{n+1})$. Decompose $\mathbf{y}$ into orthogonal components so that $\mathbf{y} = \mathbf{Q}\mathbf{y} + t\mathbf{1}$ for some $t \in \mathbb{R}$. Then $\mathbf{Q}\mathbf{y} \in \mathbf{Q}\,\mathrm{Vor}(\mathbb{Z}^{n+1})$. Assume that $\mathbf{Q}\mathbf{y} \notin \mathrm{Vor}_H(A_n)$. Then there exists some $\mathbf{x} \in A_n$ such that

$$\begin{aligned} \|\mathbf{x} - \mathbf{Q}\mathbf{y}\|^2 < \|\mathbf{0} - \mathbf{Q}\mathbf{y}\|^2 &\Rightarrow \|\mathbf{x} - \mathbf{y} + t\mathbf{1}\|^2 < \|\mathbf{y} - t\mathbf{1}\|^2 \\ &\Rightarrow \|\mathbf{x} - \mathbf{y}\|^2 + 2t\mathbf{x}'\mathbf{1} < \|\mathbf{y}\|^2. \end{aligned}$$

By (4) $\mathbf{x}'\mathbf{1} = 0$ and so $\|\mathbf{x} - \mathbf{y}\|^2 < \|\mathbf{y}\|^2$. This violates that $\mathbf{y} \in \mathrm{Vor}(\mathbb{Z}^{n+1})$ and, hence, $\mathbf{Q}\mathbf{y} \in \mathrm{Vor}_H(A_n)$.[2] □

Lemma 2: The projection of $\mathrm{Vor}(\mathbb{Z}^{n+1})$ into H is a superset of $\mathrm{Vor}(A_{n/m}) \cap H$. The sets are equal for $\mathrm{Vor}(A_n) \cap H$. That is

$$\mathrm{Vor}_H(A_{n/m}) \subseteq \mathbf{Q}\mathrm{Vor}(\mathbb{Z}^{n+1})$$

with equality only when $m = n+1$.

Proof: When $m = n+1$, $A_{n/n+1} = A_n$. The n-volume $\mathrm{vol}_H(\mathrm{Vor}_H(A_n)) = \sqrt{n+1}$ [2]. From Burger *et al.* [38] we find that the n-volume of the projected polytope $\mathrm{vol}_H(\mathbf{Q}\,\mathrm{Vor}(\mathbb{Z}^{n+1})) = \sqrt{n+1}$ also. As $\mathrm{Vor}_H(A_n)$ and $\mathbf{Q}\,\mathrm{Vor}(\mathbb{Z}^{n+1})$ are convex polytopes it follows from Lemma 1 that

$$\mathrm{Vor}_H(A_n) = \mathbf{Q}\mathrm{Vor}(\mathbb{Z}^{n+1}).$$

The proof follows from (5). □

[2]We can generalize this proof to show that for any lattice L and hyperplane P it is true that $\mathrm{project}_P(\mathrm{Vor}(L)) \subseteq \mathrm{Vor}(L \cap P) \cap P$ where $\mathrm{project}_P(\cdot)$ indicates the orthogonal projection onto P.

We will now prove Lemma 3, from which our algorithm is derived. We firstly need the following definition. Given two sets A and B we let $A+B$ be their Minkowski sum. That is, $x \in A+B$ if and only if $x = a+b$ where $a \in A$ and $b \in B$. We will also write $\mathbf{1}\mathbb{R}$ to denote the line of points $\mathbf{1}r$ for all $r \in \mathbb{R}$. Then $\mathrm{Vor}_H(A_{n/m}) + \mathbf{1}\mathbb{R}$ is an infinite cylinder with cross-section $\mathrm{Vor}_H(A_{n/m})$. It follows that $\mathrm{Vor}_H(A_{n/m}) + \mathbf{1}\mathbb{R} = \mathrm{Vor}(A_{n/m})$

Lemma 3: If $\mathbf{x} = \mathbf{Q}\mathbf{k}$ is a closest point in $A_{n/m}$ to $\mathbf{y} \in \mathbb{R}^{n+1}$ for some $\mathbf{k}$ in $\mathbb{Z}^{n+1}$, then there exists some $\lambda \in \mathbb{R}$ for which $\mathbf{k}$ is a closest point in $\mathbb{Z}^{n+1}$ to $\mathbf{y} + \lambda\mathbf{1}$.

Proof: As $\mathbf{Q}\mathbf{k}$ is the nearest point to $\mathbf{y}$ then for all $\mu \in \mathbb{R}$

$$\mathbf{y} + \mathbf{1}\mu \in \mathrm{Vor}(A_{n/m}) + \mathbf{Q}\mathbf{k} = \mathrm{Vor}_H(A_{n/m}) + \mathbf{k} + \mathbf{1}\mathbb{R}.$$

It follows from Lemma 2 that

$$\mathrm{Vor}_H(A_{n/m}) + \mathbf{k} + \mathbf{1}\mathbb{R} \subseteq \mathbf{Q}\mathrm{Vor}(\mathbb{Z}^{n+1}) + \mathbf{k} + \mathbf{1}\mathbb{R}.$$

Then $\mathbf{y} + \mathbf{1}\mu \in \mathbf{Q}\,\mathrm{Vor}(\mathbb{Z}^{n+1}) + \mathbf{k} + \mathbf{1}\mathbb{R}$ and for some $\lambda \in \mathbb{R}$

$$\mathbf{y} + \mathbf{1}\lambda \in \mathrm{Vor}(\mathbb{Z}^{n+1}) + \mathbf{k}.$$

□

Now consider the function $\mathbf{f} : \mathbb{R} \mapsto \mathbb{Z}^{n+1}$ defined so that

$$\mathbf{f}(\lambda) = \lfloor \mathbf{y} + \lambda\mathbf{1} \rceil \tag{6}$$

where $\lfloor \cdot \rceil$ applied to a vector denotes the vector in which each element is rounded to a nearest integer[3]. That is, $\mathbf{f}(\lambda)$ gives a nearest point in $\mathbb{Z}^{n+1}$ to $\mathbf{y} + \lambda\mathbf{1}$ as a function of λ. Observe that $\mathbf{f}(\lambda + 1) = \mathbf{f}(\lambda) + \mathbf{1}$. Hence

$$\mathbf{Q}\mathbf{f}(\lambda + 1) = \mathbf{Q}\mathbf{f}(\lambda). \tag{7}$$

Lemma 3 implies there exists some $\lambda \in \mathbb{R}$ such that $\mathbf{x} = \mathbf{Q}\mathbf{f}(\lambda)$ is a closest point to $\mathbf{y}$. Furthermore, we see from (7) that λ can be found within an interval of length 1. Hence, if we define the set

$$S = \{\mathbf{f}(\lambda) \mid \lambda \in [0, 1)\}$$

then $\mathbf{Q}S$ contains a closest point in $A_{n/m}$ to $\mathbf{y}$. In order to evaluate the elements in S we require the following function.

Definition 1. (Sort Indices): We define the function

$$\mathbf{s} = \mathrm{sortindices}(\mathbf{z})$$

to take a vector $\mathbf{z}$ of length $n+1$ and return a vector $\mathbf{s}$ of indices such that

$$z_{s_1} \geq z_{s_2} \geq z_{s_3} \geq \cdots \geq z_{s_{n+1}}.$$

Let

$$\mathbf{s} = \mathrm{sortindices}(\{\mathbf{y}\})$$

[3]The direction of rounding for half-integers is not important so long as it's consistent. The authors have chosen to round up half-integers in their own implementation.

where $\{g\} = g - \lfloor g \rceil$ denotes the centered fractional part of $g \in \mathbb{R}$ and we define $\{\cdot\}$ to operate on vectors by taking the centered fractional part of each element in the vector. It is clear that S contains at most $n+2$ vectors, i.e.

$$S \subseteq \{\lfloor \mathbf{y} \rceil, \lfloor \mathbf{y} \rceil + \mathbf{e}_{s_1}, \lfloor \mathbf{y} \rceil + \mathbf{e}_{s_1} + \mathbf{e}_{s_2}, \ldots, \lfloor \mathbf{y} \rceil + \mathbf{e}_{s_1} + \ldots + \mathbf{e}_{s_{n+1}}\} \tag{8}$$

where $\mathbf{e}_i$ is a vector of 0's with a 1 in the ith position. It can be seen that the last vector listed in the set is simply $\lfloor \mathbf{y} \rceil + \mathbf{1}$ and so, once multiplied by $\mathbf{Q}$, the first and the last vector are identical.

We can define the set $W \subseteq S$ such that

$$W = \{\mathbf{x} \in S \mid \mathbf{x}'\mathbf{1} \bmod m = 0\}. \tag{9}$$

Noting (1) then $\mathbf{Q}W$ contains the nearest point in $A_{n/m}$ to $\mathbf{y}$.

An algorithm suggests itself: test each of the distinct vectors in $\mathbf{Q}W$ and find the closest one to $\mathbf{y}$. This is the principle of the algorithm we propose in this section. It remains to show that this can be done in $O(n \log n)$ arithmetic operations.

We label the elements of S according to the order given in (8). That is, we set $\mathbf{u}_0 = \lfloor \mathbf{y} \rceil$ and, for $i = 1, \ldots, n$

$$\mathbf{u}_i = \mathbf{u}_{i-1} + \mathbf{e}_{s_i}. \tag{10}$$

Let $\mathbf{z}_i = \mathbf{y} - \mathbf{u}_i$. Clearly, $\mathbf{z}_0 = \{\mathbf{y}\}$. Decompose $\mathbf{y}$ into orthogonal components $\mathbf{Q}\mathbf{y}$ and $t\mathbf{1}$ for some $t \in \mathbb{R}$. The squared distance between $\mathbf{Q}\mathbf{u}_i$ and $\mathbf{y}$ is

$$\|\mathbf{y} - \mathbf{Q}\mathbf{u}_i\|^2 = d_i + t^2(n+1) \tag{11}$$

where we define d_i as

$$d_i = \|\mathbf{Q}\mathbf{z}_i\|^2 = \left\|\mathbf{z}_i - \frac{\mathbf{z}_i'\mathbf{1}}{n+1}\mathbf{1}\right\|^2 = \mathbf{z}_i'\mathbf{z}_i - \frac{(\mathbf{z}_i'\mathbf{1})^2}{n+1}. \tag{12}$$

We know that the nearest point to $\mathbf{y}$ is that $\mathbf{Q}\mathbf{u}_i$ with $\mathbf{u}_i \in W$ which minimizes (11). Since the term $t^2(n+1)$ is independent of the index i, we can ignore it. That is, it is sufficient to minimize d_i, $i = 0, \ldots, n$.

We now show that d_i can be calculated inexpensively in a recursive fashion. We define two new quantities, $\alpha_i = \mathbf{z}_i'\mathbf{1}$ and $\beta_i = \mathbf{z}_i'\mathbf{z}_i$. Clearly, $d_i = \beta_i - \alpha_i^2/(n+1)$. From (10)

$$\alpha_i = \mathbf{z}_i'\mathbf{1} = (\mathbf{z}_{i-1} - \mathbf{e}_{s_i})'\mathbf{1} = \alpha_{i-1} - 1 \tag{13}$$

and

$$\begin{aligned}\beta_i &= \mathbf{z}_i'\mathbf{z}_i = (\mathbf{z}_{i-1} - \mathbf{e}_{s_i})'(\mathbf{z}_{i-1} - \mathbf{e}_{s_i}) \\ &= \beta_{i-1} - 2\{y_{s_i}\} + 1.\end{aligned} \tag{14}$$

Algorithm 1 now follows. The main loop beginning at line 8 calculates the α_i and β_i recursively. There is no need to retain their previous values, so the subscripts are dropped. The variable D maintains the minimum value of the (implicitly calculated values of) d_i so far encountered, and k the corresponding index. The variable γ maintains the value of $\mathbf{u}_i'\mathbf{1} \bmod m$ which must equal 0 in order for $\mathbf{u}_i \in W$.

Input: $\mathbf{y} \in \mathbb{R}^{n+1}$
1 $\mathbf{u} = \lfloor \mathbf{y} \rceil$
2 $\mathbf{z} = \mathbf{y} - \mathbf{u}$
3 $\alpha = \mathbf{z}'\mathbf{1}$
4 $\beta = \mathbf{z}'\mathbf{z}$
5 $\gamma = \mathbf{u}'\mathbf{1} \bmod m$
6 $\mathbf{s} = \text{sortindices}(\mathbf{z})$
7 $D = \infty$
8 **for** $i = 1$ **to** $n+1$ **do**
9 **if** $\beta - \alpha^2/(n+1) < D$ **and** $\gamma = 0$ **then**
10 $D = \beta - \alpha^2/(n+1)$
11 $k = i - 1$
12 $\alpha = \alpha - 1$
13 $\beta = \beta - 2z_{s_i} + 1$
14 $\gamma = (\gamma + 1) \bmod m$
15 **for** $i = 1$ **to** k **do**
16 $u_{s_i} = u_{s_i} + 1$
17 $\mathbf{x} = \mathbf{Q}\mathbf{u}$
18 **return** $\mathbf{x}$

Algorithm 1: Algorithm to find a nearest lattice point in $A_{n/m}$ to $\mathbf{y} \in \mathbb{R}^{n+1}$ that requires $O(n \log n)$ arithmetic operations.

Each line of the main loop requires $O(1)$ arithmetic computations so the loop (and that on line 15) requires $O(n)$ in total. The function $\text{sortindices}(\mathbf{z})$ requires sorting $n+1$ elements. This requires $O(n \log n)$ arithmetic operations. The vector operations on lines 2–5 all require $O(n)$ operations and the matrix multiplication on line 17 can be performed in $O(n)$ operations as

$$\mathbf{Q}\mathbf{u} = \mathbf{u} - \frac{\mathbf{1}'\mathbf{u}}{n+1}\mathbf{1}.$$

It can be seen, then, that the computational cost of the algorithm is dominated by the $\text{sortindices}(\cdot)$ function and is therefore $O(n \log n)$.

This algorithm is similar to the nearest point algorithm for A_n^* described in [20]. The significant difference is the addition of $\gamma = 0$ on line 1. This ensures that the lattice points considered are elements of $A_{n/m}$, i.e., they satisfy (1). We further discuss the relationship between the algorithms in Section IV.

III. Linear-Time Algorithm

In the previous Section we showed that the nearest point to $\mathbf{y}$ in $A_{n/m}$ lies in the set $\mathbf{Q}W$ with W defined in (9). We will show that some of the elements of $\mathbf{Q}W$ can be immediately excluded from consideration. This property leads to a nearest point algorithm that requires at most $O(n)$ arithmetic operations.

Lemma 4: Suppose, for some integers $i, m > 0$, $k \geq 2$, that

$$\{y_{s_i}\} - \{y_{s_{i+km}}\} \leq \frac{m}{n+1}. \tag{15}$$

Then the minimum of the d_{i+cm}, $c = 0, \ldots, k$, occurs at $c = 0$ or $c = k$.

Proof: The proof proceeds by contradiction. Suppose, to the contrary, that

$$d_{i+cm} < d_i \text{ and } d_{i+cm} < d_{i+km}.$$

Observe that

$$d_{i+cm} - d_i = \frac{2\alpha_i cm - (cm)^2}{n+1} + \sum_{j=1}^{cm}(1 - 2\{y_{s_{i+j}}\}).$$

Now, since $\{y_{s_{i+j}}\} \leq \{y_{s_i}\}$, it follows that

$$d_{i+cm} - d_i \geq \frac{2\alpha_i cm - (cm)^2}{n+1} + cm(1 - 2\{y_{s_i}\}).$$

With the assumption that $d_{i+cm} - d_i < 0$, we have that

$$\frac{2\alpha_i - cm}{n+1} < 2\{y_{s_i}\} - 1. \tag{16}$$

Similarly, observe that

$$d_{i+km} - d_{i+cm} = \frac{2\alpha_i(k-c)m - (k^2 - c^2)m^2}{n+1} + \sum_{j=cm+1}^{km}(1 - 2\{y_{s_{i+j}}\}).$$

Since $\{y_{s_{i+j}}\} \geq \{y_{s_{i+km}}\}$, it follows that

$$d_{i+km} - d_{i+cm} \leq \frac{2\alpha_i(k-c)m - (k^2 - c^2)m^2}{n+1} + (k-c)m\big(1 - 2\{y_{s_{i+km}}\}\big).$$

With the assumption that $d_{i+km} - d_{i+cm} > 0$, we have that

$$\frac{2\alpha_i - cm}{n+1} > \frac{km}{n+1} - 1 + 2\{y_{s_{i+km}}\}. \tag{17}$$

Equations (16) and (17) together imply that

$$\{y_{s_i}\} - \{y_{s_{i+km}}\} > \frac{km}{2(n+1)},$$

which contradicts (15) because $k \geq 2$. □

From S we can construct the following $q = (n+1)/m$ subsets

$$U_j = \left\{ \mathbf{u}_i \mid 0.5 - \{y_{s_i}\} \in \left(\frac{m(j-1)}{n+1}, \frac{mj}{n+1} \right] \right\} \tag{18}$$

where $j = 1, \ldots, q$. Note that $\mathbf{Q}S = \mathbf{Q}\bigcup_{j=1}^{q} U_j$. We are interested in the $\mathbf{u}_i \in U_j$ such that $\mathbf{u}_i'\mathbf{1}$ is a multiple of m i.e., the elements in $U_j \cap W$. Let g be the smallest integer such that $\mathbf{u}_g \in U_j \cap W$. Let p be the largest integer such that $\mathbf{u}_p \in U_j \cap W$. It follows that $p = g + km$ for some $k \in \mathbb{Z}$. Also, from (18)

$$\{y_{s_g}\} - \{y_{s_p}\} \leq \frac{m}{n+1}.$$

It then follows from Lemma 4 that (11) is minimized either by $\mathbf{u}_g$ or $\mathbf{u}_p$ and not by any $\mathbf{u}_i \in U_j \cap W$ where $g < i < p$. We see that for each set $\mathbf{Q}U_j$ there are at most two elements that are candidates for the nearest point. An algorithm can be constructed as follows: test the (at most two) candidates in each set $\mathbf{Q}U_j$ and return the closest one to $\mathbf{y}$. We will now show how this can be achieved in linear time.

We construct q sets

$$B_j = \left\{ i \mid 0.5 - \{y_i\} \in \left(\frac{m(j-1)}{n+1}, \frac{mj}{n+1} \right] \right\} \tag{19}$$

and the related sets

$$K_j = \bigcup_{t=1}^{j} B_t.$$

It follows that

$$\mathbf{u}_{|K_j|} = \lfloor \mathbf{y} \rceil + \sum_{t \in K_j} \mathbf{e}_t.$$

Definition 2. (Quick Partition): We define the function

$$\mathbf{b} = \text{quickpartition}(\mathbf{z}, B_j, c)$$

to take a vector $\mathbf{z}$ and integer $c \in \{1, \ldots, |B_j|\}$ and return a vector $\mathbf{B}$ of length $|B_j|$ such that for $i = 1, \ldots, c-1$ and $t = c+1, \ldots, |B_j|$

$$z_{b_i} \geq Z_{b_c} \geq Z_{b_t}.$$

Somewhat surprisingly, $\text{quickpartition}(\mathbf{z}, B_j, c)$ can be implemented such that the required number of operations is $O(|B_j|)$. This is facilitated by the Rivest-Tarjan selection algorithm [34]–[37]. We can compute

$$\mathbf{b} = \text{quickpartition}(\mathbf{z}, B_j, c) \tag{20}$$

for some integer $1 \leq c \leq |B_j|$. Then

$$\mathbf{u}_{|K_{j-1}|+c} = \mathbf{u}_{|K_{j-1}|} + \sum_{t \in 1}^{c} \mathbf{e}_{b_t}. \tag{21}$$

Let g be the smallest integer such that $1 \leq g \leq |B_j|$ and

$$\mathbf{1} \cdot \mathbf{u}_{|K_{j-1}|+g} \bmod m = 0 \tag{22}$$

and let p be the largest integer such that $1 \leq p \leq |B_j|$ and

$$\mathbf{1} \cdot \mathbf{u}_{|K_{j-1}|+p} \bmod m = 0. \tag{23}$$

From the previous discussion the only candidates for the nearest point out of the elements

$$\mathbf{Q}\left\{ \mathbf{u}_{|K_{j-1}|+1}, \ldots, \mathbf{u}_{|K_{j-1}|+|B_j|} \right\} = \mathbf{Q}U_j$$

are $\mathbf{Q}\mathbf{u}_{|K_{j-1}|+g}$ and $\mathbf{Q}\mathbf{u}_{|K_{j-1}|+p}$. We can compute these quickly using the $\text{quickpartition}(\cdot)$ function as in (20) and (21).

Input: $\mathbf{y} \in \mathbb{R}^{n+1}$
1 $\mathbf{z} = \mathbf{y} - \lfloor \mathbf{y} \rceil$
2 **for** $j = 1$ **to** q **do** $B_j = \emptyset$
3 **for** $i = 1$ **to** $n+1$ **do**
4 $\quad j = \lceil q(1/2 - z_i) \rceil$
5 $\quad B_j = B_j \cup i$
6 $\mathbf{u} = \lfloor \mathbf{y} \rceil$
7 $\alpha = \mathbf{z}'\mathbf{1}$
8 $\beta = \mathbf{z}'\mathbf{z}$
9 $\gamma = \mathbf{u}'\mathbf{1} \bmod m$
10 $k = 1$
11 $D = \infty$
12 **for** $j = 1$ **to** q **do**
13 $\quad g = m - \gamma$
14 $\quad p = |B_j| - (|B_j| + \gamma) \bmod m$
15 $\quad \mathbf{b} = \text{quickpartition2}(\mathbf{z}, B_j, g, p)$
16 $\quad$ **for** $i = 1$ **to** $|B_j|$ **do**
17 $\quad\quad \alpha = \alpha - 1$
18 $\quad\quad \beta = \beta - 2z_{b_i} + 1$
19 $\quad\quad \gamma = (\gamma + 1) \bmod m$
20 $\quad\quad$ **if** $(i = g$ **or** $i = p)$ **and** $\beta - \alpha^2/(n+1)$
21 $\quad\quad\quad D = \beta - \alpha^2/(n+1)$
22 $\quad\quad\quad k^* = k$
23 $\quad\quad k = k + 1$
24 $\quad \text{concatenate}(\mathbf{w}, \mathbf{b})$
25 **for** $i = 1$ **to** k^* **do**
26 $\quad u_{w_i} = u_{w_i} + 1$
27 $\mathbf{x} = \mathbf{Q}\mathbf{u}$
28 **return** $\mathbf{x}$

Algorithm 2: Algorithm to find a nearest lattice point in $A_{n/m}$ to $\mathbf{y} \in \mathbb{R}^{n+1}$ that requires $O(n)$ arithmetic operations.

Algorithm 2 now follows. Lines 2–5 construct the sets B_j. We use $\lceil x \rceil$ to denote the smallest integer greater than or equal to x. The main loop on line 12 then computes the values of g and p for each B_j. We define the function

$$\mathbf{b} = \text{quickpartition2}(\mathbf{z}, B_j, g, p)$$

to return $\mathbf{b}$ so that for $i = 1, \ldots, g-1$ and $t = g+1, \ldots, p-1$ and $c = p+1, \ldots, |B_j|$

$$z_{b_i} \geq z_{b_g} \geq z_{b_t} \geq z_{b_p} \geq z_{b_c}.$$

Notice that $\text{quickpartition2}(\cdot)$ can be performed by two consecutive iterations of the Rivest-Tarjan algorithm and therefore requires $O(|B_j|)$ operations. The $d_{|K_j|+g}$ and $d_{|K_j|+p}$ are computed within the loop on line 16 and the index of the nearest lattice point is stored using the variable k^*. The $\text{concatenate}(\mathbf{w}, \mathbf{b})$ function on line 24 adds the elements of $\mathbf{b}$ to the end of the array $\mathbf{w}$. This can be performed in $O(|B_j|)$ operations. Lines 25–27 recovers the nearest lattice point using $\mathbf{w}$ and k^*.

In practice the B_j can be implemented as a list so that the set insertion operation on line 5 can be performed in constant time.

Input: $\mathbf{y} \in \mathbb{R}^{n+1}$

1 $k = (n+1-\lfloor \mathbf{y} \rceil' \mathbf{1}) \bmod n+1$

2 $\mathbf{s} = \text{sortindices}(\{\mathbf{y}\})$

3 $\mathbf{u} = \lfloor \mathbf{y} \rceil$

4 **foreach** $i = 1$ **to** k **do**

5 $\quad u_{s_i} = u_{s_i} + 1$

6 $\mathbf{x} = \mathbf{Q}\mathbf{u}$

7 **return** $\mathbf{x}$

Algorithm 3: Algorithm to find a nearest lattice point in A_n to $y \in \mathbb{R}^n$ that requires $O(n \log n)$ operations.

Then the loops on lines 2 and 3 require $O(n)$ arithmetic operations. The operations inside the main loop on line 12 require $O(|B_j|)$ operations. The complexity of these loops is then

$$\sum_{j=1}^{n+1/m} O(|B_j|) = O(n).$$

The remaining lines require $O(n)$ or less operations. The algorithm therefore requires $O(n)$ arithmetic operations in total.

IV. Specific Algorithms for A_n and A_n^*

For the lattices $A_n = A_{n/n+1}$ and $A_n^* = A_{n/1}$ Algorithms 1 and 2 are equivalent to algorithms that have previously been described in the literature. For A_n a log-linear time algorithm similar to that of Conway and Sloane [12], [19] is derived from Algorithm 1 by noting that only one iteration in the main loop on line 8 will satisfy $\gamma = 0$. Algorithm 3 now follows.

A simple linear-time algorithm for A_n can be constructed from Algorithm 3 by replacing the $\text{sortindices}(\cdot)$ function on line 2 with

$$\mathbf{s} = \text{quickpartition}(\{\mathbf{y}\}, \{1, 2, \ldots, n+1\}, k).$$

In effect this is a modification of Algorithm 2 where the sets from (19) are replaced by the single set $\{1, 2, \ldots, n+1\}$. This algorithm has previously been suggested by A. M. Odlyzko [2, page 448].

For A_n^* a log-linear time algorithm identical to that described in [20] can be derived from Algorithm 1 by noting that $\gamma \bmod 1 = 0$ for all γ. A linear-time algorithm for A_n^* can be constructed from Algorithm 2 by noting that $g = 1$ (22) and $p = |B_j|$ (23) for all B_j where $j = 1, 2, \ldots, n+1$. This removes the need for using the $\text{quickpartition2}(\cdot)$ function. A further simplification is noted in [21] where it was shown that the nearest point is one of the $\mathbf{Q}\mathbf{u}_{|K_j|}$ where $j = 0, \ldots, n$. The reader is referred to [21] for further details. The proofs used in [21] are different to those in this paper and are only applicable to A_n^*.

V. Algorithm Based on Glue Vectors

In this section, we describe a simple nearest point algorithm for $A_{n/m}$. This algorithm was described by Conway and Sloane [12], [19] but not directly applied to the Coxeter lattices. The algorithm has complexity $O(nq)$ where $q = (n+1)/m$.

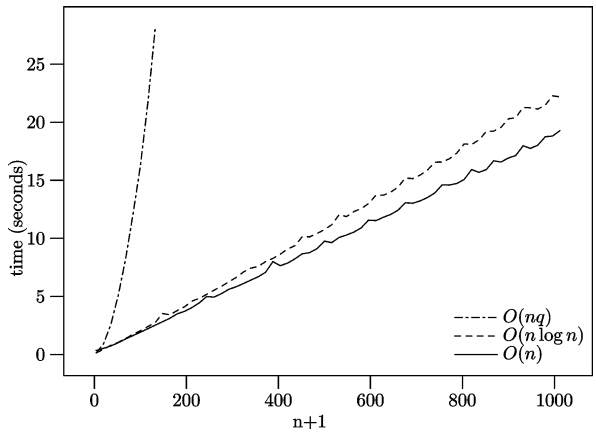

Fig. 1. Computation time in seconds for $A_{n/4}$.

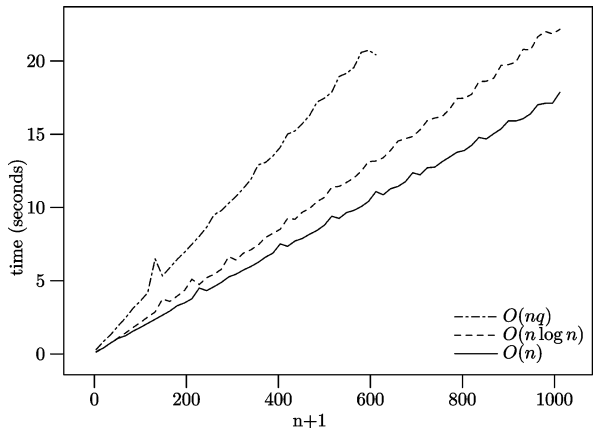

Fig. 2. Computation time in seconds for $A_{n/m}$ with $m = \frac{n+1}{4}$.

$A_{n/m}$ can be constructed by *gluing* translates of the lattice A_n [2]. That is

$$A_{n/m} = \bigcup_{i=0}^{q-1} ([im] + A_n) \tag{24}$$

where $q = (n+1)/m$ and $[i]$ are called *glue vectors* and are defined as

$$[i] = \mathbf{Q}[\underbrace{1, \ldots, 1}_{j \text{ times}}, \underbrace{0, \ldots, 0}_{i \text{ times}}]' \tag{25}$$

for $i \in \{0, \ldots, n\}$ with $i + j = n + 1$. Following the notation of Conway and Sloane the glue vectors will not be written in boldface. Instead they are indicated by square brackets.

Noting that $A_{n/m}$ can be constructed as a union of q translates of the lattice A_n we can use a nearest point algorithm for A_n to find the nearest point in each of the translates. The translate containing the closest point yields the nearest point in $A_{n/m}$. A pseudocode implementation is provided in Algorithm 4. The function $\text{NearestPt}(\mathbf{y}, A_n)$ can be implemented by the algorithms discussed in Section IV.

Input: $\mathbf{y} \in \mathbb{R}^n$
$D = \infty$
for $i = 0$ **to** $q - 1$ **do**
  $\mathbf{x} = \text{NearestPt}(\mathbf{y} - [im], A_n) + [im]$
  if $\|\mathbf{x} - \mathbf{y}\| < D$ **then**
    $\mathbf{x}_{\text{NP}} = \mathbf{x}$
    $D = \|\mathbf{x} - \mathbf{y}\|$
return $\mathbf{x}_{\text{NP}}$

Algorithm 4: Nearest point algorithm for $A_{n/m}$ using glue vectors.

The algorithm requires iterating $\text{NearestPt}(\mathbf{y}, A_n)$ q times. Assuming that $\text{NearestPt}(\mathbf{y}, A_n)$ is implemented using the linear time algorithm discussed in Section IV then the algorithm requires $O(nq)$ operations.

VI. Run-Time Analysis

In this section, we tabulate some practical computation times attained with the nearest point algorithms described in Sections II–V. Simulations were run by setting n to be $16i + 3$ for $i \in \{0, 1, \ldots, 62\}$. Ten thousand trials were run for each value of n with randomly generated input vectors. The average computation time in seconds was calculated. The algorithms were written in Java and the computer used is a 900 MHz Intel Celeron M.

Fig. 1 shows the computation times for the three algorithms from Sections V, II and III for the lattice $A_{n/4}$ and $q = (n+1)/4$. It is evident that the linear-time algorithm is the fastest. The glue vector algorithm is significantly slower for large n. By comparison, Fig. 2 shows the computation times for the algorithms with $A_{n/m}$ for $m = (n+1)/4$ and $q = 4$. The glue vector algorithm now performs similarly to the other algorithms. This behavior is expected as the glue vector algorithms complexity is $O(nq)$.

Figs. 3 and 4 show the performance of the linear-time Coxeter lattice algorithm compared to the specialized algorithms for the lattices A_n^* and A_n discussed in Section IV. It is evident that the specialized algorithms are faster. This behavior is expected as the specialized algorithms have less computational overhead.

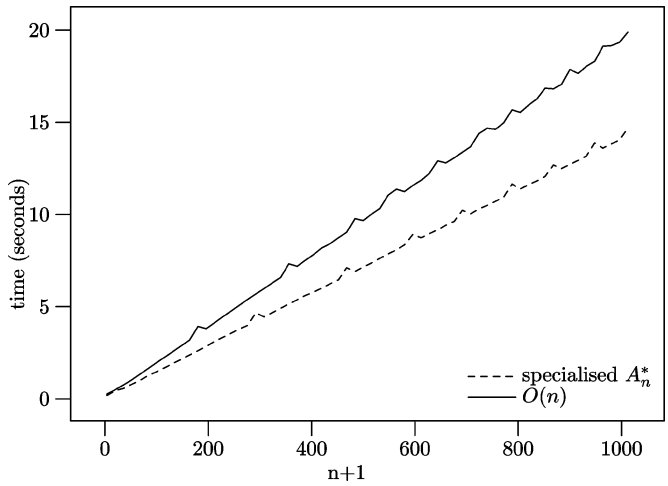

Fig. 3. Computation time in seconds for linear-time A_n^* [21] and $A_{n/1}$ (Algorithm 2).

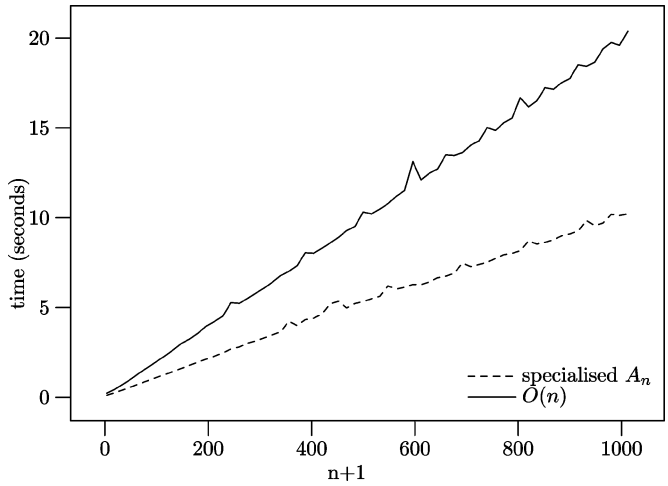

Fig. 4. Computation time in seconds for linear-time A_n algorithm (Section IV) and $A_{n/n+1}$ (Algorithm 2).

VII. Conclusion

In this paper we have described two new nearest point algorithms for the Coxeter lattices. The first algorithm is a generalization of the nearest point algorithm for A_n^* described in [20] and requires $O(n \log n)$ arithmetic operations. The second algorithm requires $O(n)$ operations in the worst case. The second algorithm makes use of a partial sorting procedure called a bucket sort [33] and also the linear-time Rivest-Tarjan selection algorithm [34]–[37]. In Section IV we showed how for the specific lattices A_n and A_n^* the log-linear and linear-time algorithms for the Coxeter lattices are equivalent to nearest point algorithms that already exist in [2], [20], [21]. The practical computational performance of the algorithms was evaluated in Section VI. It was found that the linear-time algorithm outperforms the log-linear-time algorithm in practice. The linear-time algorithm is not as fast as specialized algorithms for the specific lattices A_n and A_n^*.

References

[1] I. V. L. Clarkson, "An algorithm to compute a nearest point in the lattice A_n^*," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, M. Fossorier, H. Imai, S. Lin, and A. Poli, Eds. Karlsruhe, Germany: Springer, 1999, vol. 1719, Lecture Notes in Computer Science, pp. 104–120.

[2] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, 3rd ed. New York: Springer, 1998.

[3] M. Ajtai, "Generating hard instances of lattice problems," in *Proc. 28th ACM Symp. Theory Computing*, May 1996, pp. 99–108.

[4] M. Ajtai and C. Dwork, "A public-key cryptosystem with worst-case/average-case equivalence," in *Proc. 29th ACM Symp. Theory of Computing*, May 1997, pp. 284–293.

[5] L. Brunel and J. J. Boutros, "Lattice decoding for joint detection in direct-sequence CDMA systems," *IEEE Trans. Inf. Theory*, vol. 49, pp. 1030–1037, 2003.

[6] D. J. Ryan, I. V. L. Clarkson, I. B. Collings, and W. Heath, Jr, "Performance of vector perturbation multiuser MIMO systems with limited feedback," *IEEE Trans. Commun.*, Sep. 2008.

[7] R. G. McKilliam and I. V. L. Clarkson, "Identifiability and aliasing in polynomial-phase signals, accepted for," *IEEE Trans. Signal Process.*, 2009.
[8] R. G. McKilliam, B. G. Quinn, I. V. L. Clarkson, and B. Moran, "Frequency estimation by phase unwrapping," *IEEE Trans. Signal Process.*, 2009, submitted for publication.
[9] I. V. L. Clarkson, "Approximate maximum-likelihood period estimation from sparse, noisy timing data," *IEEE Trans. Signal Process.*, vol. 56, no. 5, pp. 1779–1787, May 2008.
[10] R. G. McKilliam, I. V. L. Clarkson, D. J. Ryan, and I. B. Collings, "Linear-time block noncoherent detection of PSK," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, Apr. 2009, pp. 2465–2468.
[11] R. G. McKilliam and I. V. L. Clarkson, "Maximum-likelihood period estimation from sparse, noisy timing data," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, Mar. 2008, pp. 3697–3700.
[12] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," *IEEE Trans. Inf. Theory*, vol. 28, pp. 227–232, Mar. 1982.
[13] D. Micciancio, "The hardness of the closest vector problem with preprocessing," *IEEE Trans. Inf. Theory*, vol. 47, pp. 1212–1215, 2001.
[14] P. van Emde Boas, Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice Mathematisch Instituut, Amsterdam, The Netherlands, 1981, Tech. Rep..
[15] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, pp. 2201–2214, Aug. 2002.
[16] E. Viterbo and J. Boutros, "A universal lattice code decoder for fading channels," *IEEE Trans. Inf. Theory*, vol. 45, pp. 1639–1642, Jul. 1999.
[17] M. Pohst, "On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications," *SIGSAM Bull.*, vol. 15, no. 1, pp. 37–44, 1981.
[18] R. Kannan, "Minkowski's convex body theorem and integer programming," *Math. Oper. Res.*, vol. 12, no. 3, pp. 415–440, 1987.
[19] J. H. Conway and N. J. A. Sloane, "Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice," *IEEE Trans. Inf. Theory*, vol. 32, pp. 41–50, Jan. 1986.
[20] R. G. McKilliam, I. V. L. Clarkson, and B. G. Quinn, "An algorithm to compute the nearest point in the lattice A_n^*," *IEEE Trans. Inf. Theory*, vol. 54, pp. 4378–4381, Sep. 2008.
[21] R. G. McKilliam, I. V. L. Clarkson, W. D. Smith, and B. G. Quinn, "A linear-time nearest point algorithm for the lattice A_n^*," in *Int. Symp. Inf. Theory and its Appl.*, Dec. 2008.
[22] H. S. M. Coxeter, "Extreme forms," *Canad. J. Math.*, vol. 3, pp. 391–441, 1951.
[23] J. Martinet, *Perfect Lattices in Euclidean Spaces*. New York: Springer, 2003.
[24] A. R. Calderbank and N. J. A. Sloane, "An eight-dimensional trellis code," *Proc. IEEE*, vol. 74, pp. 757–759, May 1986.
[25] L.-F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Trans. Inf. Theory*, vol. 33, pp. 483–501, Jul. 1987.
[26] G. D. Forney, Jr, "Coset codes I: Introduction and geometrical classification," *IEEE Trans. Inf. Theory*, vol. 34, pp. 1123–1151, Sep. 1988.
[27] G. D. Forney, Jr, "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inf. Theory*, vol. 34, pp. 1152–1187, Sep. 1988.
[28] J. H. Conway and N. J. A. Sloane, "Voronoi regions of lattices, second moments of polytopes, and quantization," *IEEE Trans. Inf. Theory*, vol. 28, pp. 211–226, Mar. 1982.
[29] M. S. Postol, "Some new lattice quantization algorithms for video compression coding," *IEEE Trans. Circuits Syst.*, vol. 12, no. 1, pp. 53–60, Jan. 2002.
[30] N. Secord and R. de Buda, "Demodulation of a Gosset lattice code having a spectral null at DC," *IEEE Trans. Inf. Theory*, vol. 35, pp. 472–477, Mar. 1989.
[31] I. V. L. Clarkson, "Frequency estimation, phase unwrapping and the nearest lattice point problem," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, Mar. 1999, vol. 3, pp. 1609–1612.
[32] B. G. Quinn, "Estimating the mode of a phase distribution," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, Nov. 2007, pp. 587–591.
[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, U.K.: MIT Press, 2001.
[34] M. Blum, R. W. Floyd, V. R. Pratt, R. L. Rivest, and R. E. Tarjan, "Time bounds for selection," *J. Comput. Syst. Sci.*, vol. 7, no. 4, pp. 448–461, 1973.
[35] R. W. Floyd and R. L. Rivest, "The algorithm SELECT—For finding the ith smallest of n elements," *Commun. ACM*, vol. 18, no. 3, pp. 173–173, 1975.
[36] R. W. Floyd and R. L. Rivest, "Expected time bounds for selection," *Commun. ACM*, vol. 18, pp. 165–172, Mar. 1975.
[37] D. E. Knuth, *The Art of Computer Programming*, 3rd ed. Reading, MA: Addison-Wesley, 1997, vol. 2, Seminumerical Algorithms.
[38] T. Burger, P. Gritzmann, and V. Klee, "Polytope projection and projection polytopes," *The Amer. Math. Monthly*, vol. 103, no. 9, pp. 742–755, Nov. 1996.

Robby G. McKilliam (S'07) was born in Brisbane, Queensland, Australia, in 1983. He received the B.Sc. degree in mathematics and the B.E. degree (Hons. I) in computer systems engineering from the University of Queensland, Brisbane, in 2006, where he is currently working toward the Ph.D. degree.

His fields of interest are lattice theory, number theory and signal processing and communications.

Mr. McKilliam currently receives funding from an Australian postgraduate award and the CSIRO ICT Centre, Sydney, Australia

Warren D. Smith completed his undergraduate studies in mathematics and physics at the Massachusetts Institute of Technology, Cambridge, in 1984. He received the Ph.D. degree in applied mathematics from Princeton University, Princeton, NJ, in 1988.

He has since worked with Bell Labs, NEC, and Temple University, Philadelphia, PA. In 2005, he founded the Center for Range Voting, Stony Brook, NY.

I. Vaughan L. Clarkson (S'93–M'98–SM'04) was born in Brisbane, Queensland, Australia, in 1968. He received the B.Sc. degree in mathematics and the B.E. degree (Hons. I) in computer systems engineering from the University of Queensland, Brisbane, Australia, in 1989 and 1990, respectively, and the Ph.D. degree in systems engineering from the Australian National University, Canberra, Australia, in 1997.

Since 1988, he has been with the Defence Science and Technology Organisation, Adelaide, Australia, first as a Cadet, later as a Professional Officer, and finally as a Research Scientist. From 1998 to 2000, he was a Lecturer with at the University of Melbourne, Melbourne, Australia. From 2000 to 2008, he was a Senior Lecturer with the School of Information Technology and Electrical Engineering, The University of Queensland. In 2008, he was promoted to Reader. In 2005, he was a Visiting Professor with the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada. His research interests include statistical signal processing for communications and defence, image processing, information theory, and lattice theory.