International Symposium on Information Theory and its Applications, ISITA2008
Auckland, New Zealand, 7-10, December, 2008

A linear-time nearest point algorithm for the lattice A_n^*

Robby G. McKilliam[†], I. Vaughan L. Clarkson[†], Warren D. Smith and Barry G. Quinn[‡]

[†] School of Information Technology & Electrical Engineering
The University of Queensland
Brisbane, QLD 4072 AUSTRALIA
Email: [robertm, v.clarkson]@itee.uq.edu.au

[‡] Department of Statistics, EFS
Macquarie University
Sydney, NSW 2109 AUSTRALIA
Email: bquinn@efs.mq.edu.au

Abstract

The lattice A_n^* is an important lattice because of its covering properties in low dimensions. Two algorithms exist in the literature that compute the nearest point in the lattice A_n^* in $O(n \log n)$ arithmetic operations. In this paper we describe a new algorithm that requires only $O(n)$ operations. The new algorithm makes use of an approximate sorting procedure called a *bucket sort*. This is the fastest known nearest point algorithm for this lattice.

1. 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 quantisation and channel coding. They are also important in studying the sphere packing problem and the kissing number problem [1, 2].

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 matrix*.

The lattice A_n^* is an interesting lattice due to its covering properties in low dimensions. It gives the thinnest covering in all dimensions up to 8 [2]. A_n^* has also found application in a number of estimation problems including period estimation from sparse timing data [3, 4], frequency estimation [5] and direction of arrival estimation [6].

The nearest lattice point problem is: Given $\mathbf{y} \in \mathbb{R}^n$ and some lattice L whose lattice points lie in $\mathbb{R}^n$, find the lattice point $\mathbf{x} \in L$ such that the Euclidean distance between $\mathbf{y}$ and $\mathbf{x}$ is minimised. If the lattice is used for vector quantisation 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 [7].

Conway and Sloane [7] appear to have been the first to study the problem of computing the nearest lattice point in A_n^*. By decomposing A_n^* into a union of translations of its dual lattice A_n, they discovered an algorithm for computing the nearest lattice point to a given point in $O(n^2 \log n)$ arithmetic operations. Later [8], they were able to improve the execution time of the algorithm to $O(n^2)$ operations.

Clarkson [1] further improved upon the work of Conway and Sloane and described an algorithm to compute the nearest lattice point that requires only $O(n \log n)$ operations. Recently another algorithm that requires $O(n \log n)$ operations was described [9]. This algorithm proved to be simpler to describe, prove and implement than the algorithm proposed by Clarkson. The complexity of the algorithms from [1] and [9] are dominated by a sorting operation.

In this paper we describe a new algorithm that requires $O(n)$ arithmetic operations. The new algorithm makes use of an approximate sorting technique called a *bucket sort* [10]. Particular properties of the lattice A_n^* are used to show that the approximate sort is sufficient to find the nearest lattice point.

We now describe how the paper is organised. Section 2 introduces some properties of the lattice A_n^*. In Section 3 we derive results necessary to prove that our algorithm finds the nearest lattice point. In Section 4 the $O(n)$-time algorithm is described. Two pseudocode implementations are provided. The first being more instructive and easy to follow. The second being less intuitive but precisely indicating how to efficiently implement the algorithm practice. In Section 5 we provide a table displaying the practical run-time performance of the algorithms in [1, 9] and the new algorithm. Unsurprisingly, the new, linear-time algorithm is shown to be faster in practice.

Robby McKilliam is partly supported by the CSIRO ICT, Sydney, NSW, AUSTRALIA

2. Properties of A_n^*

Vectors and matrices are written in bold font. The ith element in a vector is denoted by a subscript: x_i. The transpose of a vector is indicated by $'$: $\mathbf{x}'$. We let $\mathbf{1}$ be a column vector of 1's and $\mathbf{e}_i$ be a column vector of zeros with a 1 in the ith position.

The *Voronoi region* or *nearest-neighbour region* $\mathrm{Vor}(\mathbf{x}, L)$ of a lattice point $\mathbf{x} \in L$ for a lattice L is the subset of $\mathbb{R}^n$ such that, with respect to a given norm, all points in $\mathrm{Vor}(\mathbf{x}, L)$ are nearer to $\mathbf{x}$ than to any other point in L. The Voronoi regions are n dimensional polytopes [2].

The cubic lattice $\mathbb{Z}^n$ is the set of n dimensional vectors with integer elements. The Voronoi regions of $\mathbb{Z}^n$ are hypercubes of side length 1.

The lattice A_n^* can be defined as the projection of the cubic lattice $\mathbb{Z}^{n+1}$ onto the hyperplane orthogonal to $\mathbf{1}$. This is,

$$A_n^* = \left\{ \mathbf{Qx} \mid \mathbf{x} \in \mathbb{Z}^{n+1} \right\} \tag{1}$$

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

$$\mathbf{Q} = \left(\mathbf{I} - \frac{\mathbf{11}'}{n+1} \right) \tag{2}$$

where $\mathbf{I}$ is the $(n+1) \times (n+1)$ identity matrix.

3. The closest point

Lemma 1. *If $\mathbf{x} = \mathbf{Qk}$ is a closest point in A_n^* to $\mathbf{y} \in \mathbb{R}^{n+1}$ 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. See [9]. □

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

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

where $\lfloor \cdot \rceil$ applied to a vector denotes the vector in which each element is rounded to a nearest integer[1]. 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,

$$\mathbf{Qf}(\lambda + 1) = \mathbf{Qf}(\lambda). \tag{3}$$

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

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

then $\mathbf{Q}\mathscr{S}$ contains a closest point in A_n^* to $\mathbf{y}$. In fact only some of the vectors in $\mathbf{Q}\mathscr{S}$ are candidates for the nearest point. To show this we firstly require the *relevant vectors* for the lattice A_n^* and to prove a preliminary result in Lemma 2.

Consider the Voronoi region about the origin, $\mathrm{Vor}(\mathbf{0}, L)$. Its faces lie in hyperplanes that are on the midway point between the lines connecting nearby lattice points. The set of vectors that define the faces are called the *Voronoi relevant vectors* or simply *relevant vectors*.

Remark 1. *Let $\mathbf{r}$ be a relevant vector in the lattice L. If $\mathbf{y} \in \mathrm{Vor}(\mathbf{0}, L)$ then*

$$\mathbf{y} \cdot \mathbf{r} \leqslant \frac{\mathbf{r} \cdot \mathbf{r}}{2}.$$

Remark 2. *The relevant vectors for the lattice A_n^* are given by the vectors $\mathbf{Qu}$ where*

$$\mathbf{u} = \sum_{i \in P} \mathbf{e}_i$$

and where $P \subset \{1, 2, \ldots, n+1\}$.

A proof of Remark 2 is given in [1].

Lemma 2. *Let $\mathbf{Qf}(\lambda_0)$ be the closest point in A_n^* to $\mathbf{y} \in \mathbb{R}^{n+1}$. If $\mathscr{I}$ is defined as the interval containing λ_0 on which $\mathbf{f}(\lambda)$ is constant then the length of the interval is not less than $1/(n+1)$.*

Proof. Observe that $\mathbf{f}(\lambda)$ is piecewise constant, by virtue of the rounding operation. $\mathscr{I}$ will be open at one end and closed at the other. Which end is open and which is closed depends on the direction that half-integers are rounded. Let the endpoints of the interval be $\lambda_{\min} \leqslant \lambda_{\max}$.

Consider $\lambda \in \mathscr{I}$. For all such λ, $\mathbf{f}(\lambda)$ is constant. Let its value be $\mathbf{k}$ and let $\mathbf{x} = \mathbf{Qk}$. It is clear that $\mathbf{y} \in \mathrm{Vor}(\mathbf{x}, A_n^*)$ and so $\mathbf{y} + \lambda\mathbf{1} \in \mathrm{Vor}(\mathbf{x}, A_n^*)$. Also, $\mathbf{y} + \lambda\mathbf{1} \in \mathrm{Vor}(\mathbf{k}, \mathbb{Z}^{n+1})$. With $\mathbf{z} = \mathbf{y} - \mathbf{k}$, it follows that $\mathbf{z} + \lambda\mathbf{1} \in \mathrm{Vor}(\mathbf{0}, A_n^*) \cap \mathrm{Vor}(\mathbf{0}, \mathbb{Z}^{n+1})$.

The fact that $\mathbf{z} + \lambda\mathbf{1} \in \mathrm{Vor}(\mathbf{0}, A_n^*)$ does not immediately yield any information on the length of the interval $\mathscr{I}$ since this Voronoi region is an infinite cylinder whose central axis is in the direction of the vector $\mathbf{1}$. On the other hand, $\mathbf{z} + \lambda\mathbf{1} \in \mathrm{Vor}(\mathbf{0}, \mathbb{Z}^{n+1})$ implies that $|z_i + \lambda| \leqslant \frac{1}{2}$. If we set $\ell = \arg\max_i z_i$ and $m = \arg\min_i z_i$, it is clear that $\lambda_{\max} = \frac{1}{2} - z_\ell$ and $\lambda_{\min} = -\frac{1}{2} - z_m$. Hence, the length of the interval is

$$\lambda_{\max} - \lambda_{\min} = 1 - z_\ell + z_m. \tag{4}$$

Now, from Remark 1, it follows that

$$\mathbf{z} \cdot (\mathbf{Qe}_\ell) \leqslant \frac{1}{2} \|\mathbf{Qe}_\ell\|^2$$

[1]The direction of rounding for half-integers is not important. However, the authors have chosen to round up half-integers in their own implementation.

which implies that

$$z_\ell - \bar{z} \leqslant \frac{n}{2(n+1)} \tag{5}$$

where $\bar{z} = {}^{\mathbf{1}\cdot\mathbf{z}}/_{n+1}$. On the other hand, we must also have that

$$\mathbf{z}\cdot(-\mathbf{Q}\mathbf{e}_m) \leqslant \frac{1}{2}\|\mathbf{Q}\mathbf{e}_m\|^2$$

which implies that

$$z_m - \bar{z} \geqslant -\frac{n}{2(n+1)}. \tag{6}$$

Combining (4), (5) and (6), we find that the length of the interval $\mathscr{I}$ conforms to the lower bound

$$\lambda_{\max} - \lambda_{\min} \geqslant \frac{1}{n+1}.$$

$\square$

Lemma 3. *If* $\mathbf{Qk}$ *is the nearest point in* A_n^* *to* $\mathbf{y} \in \mathbb{R}^{n+1}$ *then*

$$\mathbf{k} = \mathbf{f}\left(\frac{i-1}{n+1}\right)$$

for some $i \in \{1, \cdots, n+1\}$.

Proof. Assume that the lemma is false. Then $\mathbf{k} = \mathbf{f}(\lambda)$ for some $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ such that

$$\frac{i-1}{n+1} < \lambda < \frac{i}{n+1}$$

for some $i \in \{1, \cdots, n+1\}$. However then

$$\lambda_{\max} - \lambda_{\min} < \frac{i}{n+1} - \frac{i-1}{n+1} = \frac{1}{n+1}$$

contradicting Lemma 2. $\square$

From Lemma 3 we see that only the lattice points

$$\mathbf{Qf}\left(\frac{i-1}{n+1}\right)$$

for $i \in \{1, \cdots, n+1\}$ are candidates for the nearest point. An algorithm suggests itself: test each point in turn and find the closest to $\mathbf{y}$. This is exactly the principal of the algorithm we propose here. It only remains to show that this can done in linear time.

4. The linear-time algorithm

We define $n+1$ sets

$$S_i = \left\{ j \mid 0.5 - y_j + \lfloor y_j \rceil \in \left(\frac{i-1}{n+1}, \frac{i}{n+1}\right] \right\}$$

for $i \in \{1, \cdots, n+1\}$. Then it follows that

$$\mathbf{f}\left(\frac{0}{n+1}\right) = \lfloor \mathbf{y} \rceil$$
$$\mathbf{f}\left(\frac{1}{n+1}\right) = \lfloor \mathbf{y} \rceil + \sum_{j\in S_1} \mathbf{e}_j$$
$$\mathbf{f}\left(\frac{2}{n+1}\right) = \lfloor \mathbf{y} \rceil + \sum_{j\in S_1} \mathbf{e}_j + \sum_{j\in S_2} \mathbf{e}_j$$

and in general

$$\mathbf{f}\left(\frac{i}{n+1}\right) = \lfloor \mathbf{y} \rceil + \sum_{j\in K_i} \mathbf{e}_j \tag{7}$$

where

$$K_i = \bigcup_{j=1}^{i} S_j$$

Let

$$\mathbf{u}_i = \mathbf{f}\left(\frac{i}{n+1}\right) \tag{8}$$

and let

$$\mathbf{z}_i = \mathbf{y} - \mathbf{u}_i \tag{9}$$

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

$$\|\mathbf{y} - \mathbf{Qu}_i\|^2 = d_i + t^2(n+1) \tag{10}$$

where we define d_i as

$$\begin{aligned} d_i &= \|\mathbf{Qy} - \mathbf{Qu}_i\|^2 = \|\mathbf{Qz}_i\|^2 \\ &= \left\| \mathbf{z}_i - \frac{\mathbf{z}_i'\mathbf{1}}{n+1}\mathbf{1} \right\|^2 \\ &= \mathbf{z}_i'\mathbf{z}_i - \frac{(\mathbf{z}_i'\mathbf{1})^2}{n+1}. \end{aligned} \tag{11}$$

We know that the nearest point to $\mathbf{y}$ is that $\mathbf{Qu}_i$ which minimises (11). Since the term $t^2(n+1)$ is independent of the index i, we can ignore it. That is, it is sufficient to minimise 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 $\beta_i = \mathbf{z}_i'\mathbf{z}_i$. Using (7), (8) and (9),

$$\begin{aligned} \alpha_i = \mathbf{z}_i'\mathbf{1} &= \left(\mathbf{z}_{i-1} - \sum_{j\in S_i} \mathbf{e}_j \right) \cdot \mathbf{1} \\ &= \alpha_{i-1} - |S_i| \end{aligned} \tag{12}$$

and

$$\beta_i = \mathbf{z}_i'\mathbf{z}_i = \left\| \mathbf{z}_{i-1} - \sum_{j \in S_i} \mathbf{e}_j \right\|^2$$
$$= \beta_{i-1} + |S_i| - 2\sum_{j \in S_i} y_j - \lfloor y_j \rceil. \qquad (13)$$

Algorithm 1 now follows. Lines 4–7 calculate the sets S_i. This is the bucket sort operation [10]. The main loop beginning at line 10 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 m the corresponding index.

The vector operations on lines 1–3 and 21 all require $O(n)$ operations. Provided that the set operations on lines 4, 7, 11 and 19 can be performed in constant time the loops on lines 4, 5, 10 and 18 require only $O(n)$ operations. The overall computational complexity of the algorithm is then $O(n)$.

Naive implementation of the set operations may lead to poor performance. For this reason we have provided a second version of the pseudocode (Algorithm 2) that hides the set notation but demonstrates how to efficiently implement the algorithm in practice. The sets, S_i, are replaced by two arrays ***bucket*** and ***link***, both of length $n + 1$.

```
   Input: $\mathbf{y} \in \mathbb{R}^{n+1}$
1  $\mathbf{z} = \mathbf{y} - \lfloor \mathbf{y} \rceil$
2  $\alpha = \mathbf{z}'\mathbf{1}$
3  $\beta = \mathbf{z}'\mathbf{z}$
4  for $i = 1$ to $n + 1$ do $S_i = \emptyset$
5  for $t = 1$ to $n + 1$ do
6      $i = n + 1 - (n + 1)\lfloor z_t + 0.5 \rfloor$
7      $S_i = S_i \cup \{t\}$
8  $D = \beta - \frac{\alpha^2}{n+1}$
9  $m = 0$
10 for $i = 1$ to $n + 1$ do
11     forall $t \in S_i$ do
12         $\alpha = \alpha - 1$
13         $\beta = \beta - 2z_t + 1$
14     if $\beta - \frac{\alpha^2}{n+1} < D$ then
15         $D = \beta - \frac{\alpha^2}{n+1}$
16         $m = i$
17 $\mathbf{k} = \lfloor \mathbf{y} \rceil$
18 for $i = 1$ to $m$ do
19     forall $t \in S_i$ do
20         $k_t = k_t + 1$
21 $\mathbf{x} = \mathbf{k} - \frac{\mathbf{1}'\mathbf{k}}{n+1}\mathbf{1}$
22 return $\mathbf{x}$
```

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

5. Run-time analysis

Table 1 shows the practical computation times achieved with the algorithms from [1] and [9] and with the new algorithm. The Clarkson algorithm is that described in [1] and the MCQ algorithm is that described in [9]. Not surprisingly, the linear-time algorithm is the fastest. The computer used is an Intel Core2 running at 2.4GHz. The software is written in Java.

Table 1: Computation time in seconds for 10^5 trials

Algorithm	n=20	n=50	n=100	n=500
Clarkson $O(n \log n)$	2.72	5.73	10.99	58.98
MCQ $O(n \log n)$	2.28	4.60	8.61	32.73
$O(n)$	1.83	3.33	5.86	25.91

6. Conclusion

In this paper we have derived a linear-time algorithm to compute the nearest lattice point in the lattice A_n^*. Two $O(n \log n)$-time algorithms have previously been described in the literature [1, 9]. The computational complexity of these algorithms was dominated by a sorting operation. By exploiting properties of the Voronoi region of the lattice A_n^* the new algorithm only requires to perform a partial sort that can be computed in $O(n)$-time using a bucket sort [10]. This results in the new algorithm requiring only $O(n)$ arithmetic operations. The new algorithm is in practice faster than previous algorithms as observed in Table 1.

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*, Marc Fossorier, Hideki Imai, Shu Lin, and Alain Poli, Eds., vol. 1719 of *Lecture Notes in Computer Science*, pp. 104–120. Springer, 1999.

[2] J. H. Conway and N. J. A. Sloane, *Sphere packings, lattices and groups*, Springer, 3rd edition, 1998.

[3] 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.

```
Input: $\mathbf{y} \in \mathbb{R}^{n+1}$
$\mathbf{z} = \mathbf{y} - \lfloor \mathbf{y} \rceil$
$\alpha = \mathbf{z}'\mathbf{1}$
$\beta = \mathbf{z}'\mathbf{z}$
$\boldsymbol{bucket} = \mathbf{0}$
for $t = 1$ to $n+1$ do
    $i = n + 1 - (n+1)\lfloor z_t + 0.5 \rfloor$
    $link_t = bucket_i$
    $bucket_i = t$
$D = \beta - \frac{\alpha^2}{n+1}$
$m = 0$
for $i = 1$ to $n+1$ do
    $t = bucket_i$
    while $t \neq 0$ do
        $\alpha = \alpha - 1$
        $\beta = \beta - 2z_t + 1$
        $t = link_t$
    if $\beta - \frac{\alpha^2}{n+1} < D$ then
        $D = \beta - \frac{\alpha^2}{n+1}$
        $m = i$
$\mathbf{k} = \lfloor \mathbf{y} \rceil$
for $i = 1$ to $m$ do
    $t = bucket_i$
    while $t \neq 0$ do
        $k_t = k_t + 1$
        $t = link_t$
$\mathbf{x} = \mathbf{k} - \frac{\mathbf{1}'\mathbf{k}}{n+1}\mathbf{1}$
return $\mathbf{x}$
```

Algorithm 2: Algorithm to find a nearest lattice point in A_n^* to $\mathbf{y} \in \mathbb{R}^{n+1}$ that requires $O(n)$ arithmetic operations. This pseudocode indicates how to implement the algorithm in practice using two arrays.

[4] R. G. McKilliam and I. V. L. Clarkson, "Maximum-likelihood period estimation from sparse, noisy timing data," *Proc. Internat. Conf. Acoust. Speech Signal Process.*, pp. 3697–3700, Mar. 2008.

[5] I. V. L. Clarkson, "Frequency estimation, phase unwrapping and the nearest lattice point problem," *Proc. Internat. Conf. Acoust. Speech Signal Process.*, vol. 3, pp. 1609–1612, Mar. 1999.

[6] B. G. Quinn, "Estimating the mode of a phase distribution," *Asilomar Conference on Signals, Systems and Computers*, pp. 587–591, Nov 2007.

[7] J. H. Conway and N. J. A. Sloane, "Fast quantizing and decoding and algorithms for lattice quantizers and codes," *IEEE Trans. Inform. Theory*, vol. 28, no. 2, pp. 227–232, Mar. 1982.

[8] 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. Inform. Theory*, vol. 32, no. 1, pp. 41–50, Jan. 1986.

[9] 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. Inform. Theory*, vol. 54, no. 9, pp. 4378–4381, Sep. 2008.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press. and McGraw-Hill, 2nd edition, 2001.