Efficient Bounded Distance Decoders for Barnes-Wall Lattices*

Daniele Micciancio† Antonio Nicolosi‡

April 30, 2008

Abstract

We describe a new family of parallelizable bounded distance decoding algorithms for the Barnes-Wall lattices, and analyze their decoding complexity. The algorithms are parameterized by the number $p = 4^k \leq N^2$ of available processors, work for Barnes-Wall lattices in arbitrary dimension $N = 2^n$, correct any error up to squared unique decoding radius $d^2_{min}/4$, and run in worst-case time $O(N \log^2 N/\sqrt{p})$. Depending on the value of the parameter p, this yields efficient decoding algorithms ranging from a fast sequential algorithm with quasi-linear decoding complexity $O(N \log^2 N)$, to a fully parallel decoding circuit with polylogarithmic depth $O(\log^2 N)$ and polynomially many arithmetic gates.

1 Introduction: Barnes-Wall Lattices

Barnes-Wall lattices are an infinite sequence of full-rank lattices defined for every dimension N that is a power of 2. For their elegant simplicity and relevance to practical applications, Barnes-Wall lattices have been the subject of extensive investigations in coding theory [2, 1, 8, 9, 3, 4, 12, 11] and mathematics [7, 6]. We use the definition of Barnes-Wall lattice BW^n as $N = 2^n$ dimensional lattices over the Gaussian integers $\mathbb{G} = \mathbb{Z} + i\mathbb{Z}$.

Definition 1. *For any positive integer n, BW^n is the $N = 2^n$ dimensional lattice over $\mathbb{G}$ generated by the rows of the n-fold Kronecker product*

$$BW^n = \begin{bmatrix} 1 & 1 \\ 0 & \phi \end{bmatrix}^{\otimes n}.$$

where $\phi = 1 + i$ is the prime of least squared norm in $\mathbb{G}$, i.e., *the $N \times N$ matrix defined by the recurrence*

$$BW^n = \begin{bmatrix} BW^{n-1} & BW^{n-1} \\ \mathbf{O} & \phi \cdot BW^{n-1} \end{bmatrix}$$

with initial condition $BW^0 = [1]$.

*To appear in the proceedings of the IEEE International Symposium on Information Theory – ISIT 2008.

†Computer Science & Engineering Department, University of California, San Diego, daniele@cs.ucsd.edu

‡Computer Science Department, Stevens Institute of Technology. nicolosi@cs.stevens.edu

Algorithm 1 Parallel Bounded Distance Decoder (BDD) for Barnes-Wall Lattices

function $\textsc{ParBW}(p, \mathbf{s})$
    if $p < 4$ or $\mathbf{s} \in \mathbb{C}^1$ **then**
        return $\textsc{SeqBW}(0, \mathbf{s})$ ▷ Run the sequential decoder from Section 3
    else
        $[\mathbf{s}_0, \mathbf{s}_1] \leftarrow \mathbf{s}$ ▷ Split $\mathbf{s}$ into two halves
        $[\mathbf{s}_-, \mathbf{s}_+] = (\phi/2) \cdot [\mathbf{s}_0 - \mathbf{s}_1, \mathbf{s}_0 + \mathbf{s}_1]$ ▷ Compute $T(\mathbf{s})$
        $\begin{bmatrix} \mathbf{z}_0 \\ \mathbf{z}_1 \\ \mathbf{z}_- \\ \mathbf{z}_+ \end{bmatrix} \leftarrow \begin{bmatrix} \textsc{ParBW}(p/4, \mathbf{s}_0) \\ \textsc{ParBW}(p/4, \mathbf{s}_1) \\ \textsc{ParBW}(p/4, \mathbf{s}_-) \\ \textsc{ParBW}(p/4, \mathbf{s}_+) \end{bmatrix}$ ▷ Execute recursive calls in parallel
        $\mathbf{z}_0^- \leftarrow [\mathbf{z}_0, \mathbf{z}_0 - 2\phi^{-1}\,\mathbf{z}_-]$ ▷ Compute 4 candidate vectors
        $\mathbf{z}_0^+ \leftarrow [\mathbf{z}_0, 2\phi^{-1}\,\mathbf{z}_+ - \mathbf{z}_0]$
        $\mathbf{z}_1^- \leftarrow [2\phi^{-1}\,\mathbf{z}_- + \mathbf{z}_1, \mathbf{z}_1]$
        $\mathbf{z}_1^+ \leftarrow [(2\phi^{-1}\,\mathbf{z}_+ - \mathbf{z}_1, \mathbf{z}_1]$
        $\mathbf{z} = \underset{\mathbf{z}' \in \{\mathbf{z}_0^-, \mathbf{z}_0^+, \mathbf{z}_1^-, \mathbf{z}_1^+\}}{\operatorname{argmin}} \{\|\mathbf{s} - \mathbf{z}'\|\}$ ▷ Select the candidate closest to $\mathbf{s}$
        return $\mathbf{z}$
    end if
end function

Equivalently, BW^n can be defined as a $2N = 2^{n+1}$ dimensional lattice over the integers in the obvious way, but complex numbers make our definitions and algorithms easier to describe. It immediately follows from the definition that $\mathrm{BW}^0 = \mathbb{G}$ is the 1-dimensional lattice of all Gaussian integers, and

$$\mathrm{BW}^{n+1} = \{[\mathbf{u}, \mathbf{u} + \phi\,\mathbf{v}] \colon \mathbf{u}, \mathbf{v} \in \mathrm{BW}^n\}, \quad \text{for } n \geq 0.$$

The Barnes-Wall lattices have minimum squared distance $d^2_{min}(\mathrm{BW}^n) = N$, volume $V(\mathrm{BW}^n) = 2^{n2^{n-1}} = \sqrt{N^N}$, and nominal coding gain $\gamma_c(\mathrm{BW}^n) = 2^{n/2} = \sqrt{N}$.

Although much effort has been put in the design of efficient decoding algorithms for Barnes-Wall lattices in specific low dimensions (like BW^2 and BW^3, [8, 12]), not much is known about the asymptotic complexity of decoding BW^n. For arbitrary n, the only decoding algorithms explicitly discussed in the literature are those based on the four-section, $2^{N/2}$-state trellis realization of BW^n (*cf. e.g.* [3]), which accomplish maximum likelihood decoding but have exponential (in N) complexity.

In this paper, we give a family of efficient (polynomial time) algorithms to solve the bounded distance decoding problem for Barnes-Wall lattices: given a vector $\mathbf{s} \in \mathbb{C}^N$ within squared distance $d^2_{min}/4 = N/4$ from some lattice point $\mathbf{z}$ in BW^n, find $\mathbf{z}$. Our family of algorithms is parameterized by an integer $p = 4^k$, ranging from $1 = 4^0$ to $N^2 = 4^n$, that represents the number of available processors. The (parallel) running time of the algorithm (measured in terms of arithmetic operations) is $O(N \log^2 N/\sqrt{p})$. All arithmetic is performed using at most $n = \log_2 N$ bits of precisions, beyond the precision used to represent the target vector $\mathbf{s} \in \mathbb{C}^N$.

2 The Parallel Bounded Distance Decoder

The algorithm is based on the following easily verifiable observations:

Algorithm 2 Sequential Bounded Distance Decoder for Barnes-Wall Lattices and Their Principal Sublattices

```
function SeqBW(r, s)
    if s ∈ ℂ^N with N ≤ 2^r then
        return ⌈s⌋ ∈ 𝔾^N                    ▷ Round s component-wise to the closest Gaussian integer
    else
        b ← ⌈ℜ(s)⌋ + ⌈ℑ(s)⌋ mod 2                ▷ Compute binary target component-wise
        ρ = 1 − 2 max(|ℜ(s) − ⌈ℜ(s)⌋|, |ℑ(s) − ⌈ℑ(s)⌋|)   ▷ Compute the reliability information
        t ← (b, ρ)                               ▷ Component-wise pairing, i.e., t_j = (b_j, ρ_j)
        ψ(c) ← RMdec^ψ(r, t)                     ▷ Call the Reed-Muller soft-decision decoder
        v ← SeqBW(r + 1, (s − ψ(c))/φ)
        return ψ(c) + φ v
    end if
end function
```

- If $[\mathbf{z}_0, \mathbf{z}_1] \in \mathrm{BW}^{n+1}$, then $\mathbf{z}_0, \mathbf{z}_1 \in \mathrm{BW}^n$.
- $\|[\mathbf{s}_0, \mathbf{s}_1]\|^2 = \|\mathbf{s}_0\|^2 + \|\mathbf{s}_1\|^2$, so if $[\mathbf{s}_0, \mathbf{s}_1]$ is within the squared unique decoding radius of BW^{n+1} ($d^2_{min}(\mathrm{BW}^{n+1})/4 = N/2$), then at least one among $\mathbf{s}_0$ and $\mathbf{s}_1$ is within the squared unique decoding radius $d^2_{min}(\mathrm{BW}^n)/4 = N/4$ of BW^n.
- The function:
$$T\colon [\mathbf{z}_0, \mathbf{z}_1] \mapsto (\phi/2) \cdot [\mathbf{z}_0 - \mathbf{z}_1, \mathbf{z}_0 + \mathbf{z}_1]$$
is an automorphism of BW^n, *i.e.*, a distance preserving linear transformation that maps BW^n to itself.
- The vectors $\mathbf{z}_0$ and $\mathbf{z}_1$ can be recovered from any of the following pairs: $(\mathbf{z}_0, \mathbf{z}_-)$, $(\mathbf{z}_0, \mathbf{z}_+)$, $(\mathbf{z}_1, \mathbf{z}_-)$, $(\mathbf{z}_1, \mathbf{z}_+)$, where $[\mathbf{z}_-, \mathbf{z}_+] = T([\mathbf{z}_0, \mathbf{z}_1])$.

These observations translate pretty much directly into Algorithm 1, reported above.

Theorem 1. *For any $N = 2^n$, $1 \le p \le N^2$, and $\mathbf{s} \in \mathbb{C}^N$ such that $dist^2(\mathbf{s}, BW^n) < N/4$, Algorithm 1 computes the (unique) lattice vector $z \in BW^n$ within squared distance $N/4$ from the target vector $\mathbf{s}$.*

Proof. The proof easily follows from the previous observations and from the correctness of the sequential decoder SeqBW given in Section 3. Let $[\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1]$ be the lattice point within squared distance $N/4$ from the target $[\mathbf{s}_0, \mathbf{s}_1]$. Since T is an automorphism of BW^n, also the target $[\mathbf{s}_-, \mathbf{s}_+] = T([\mathbf{s}_0, \mathbf{s}_1])$ is within squared distance from BW^n, and the closest lattice point to it is $[\tilde{\mathbf{z}}_-, \tilde{\mathbf{z}}_+] = T([\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1])$.

The algorithm recursively computes four $N/2$-dimensional vectors $\mathbf{z}_\star$ (for $\star \in \{0, 1, +, -\}$) with the property that if $\mathbf{s}_\star$ is within squared distance $N/2$ from BW^{n-1}, then $\mathbf{z}_\star = \tilde{\mathbf{z}}_\star$. Next, for each $b \in \{0, 1\}$ and $s \in \{-, +\}$, the algorithm computes a candidate vector $\mathbf{z}^s_b$ from $[\mathbf{z}_b, \mathbf{z}_s]$ by inverting the linear transformation that maps $[\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1]$ to $[\tilde{\mathbf{z}}_b, \tilde{\mathbf{z}}_s]$.

Since at least one vector from each pair $(\mathbf{s}_0, \mathbf{s}_1)$ and $(\mathbf{s}_-, \mathbf{s}_+)$ is within the unique decoding radius from the lattice, the algorithm correctly recovers $[\mathbf{z}_b, \mathbf{z}_s] = [\tilde{\mathbf{z}}_b, \tilde{\mathbf{z}}_s]$ for some $b \in \{0, 1\}$ and $s \in \{-, +\}$, and $\mathbf{z}^s_b = [\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1]$. Selecting the vector among $\mathbf{z}^-_0, \mathbf{z}^+_0, \mathbf{z}^-_1, \mathbf{z}^+_1$ closest to the target correctly identifies $[\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1]$ because $[\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_1]$ is the only lattice vector within the unique decoding radius from the lattice. □

Theorem 2. *For any $N = 2^n$, $1 \le p \le N^2$, and $\mathbf{s} \in \mathbb{C}^N$, the execution of Algorithm 1 on p processors terminates after $O(N \log^2 N/\sqrt{p})$ steps on each processor.*

Proof. Performing steps 5–6 and 8–11 of Algorithm 1 clearly takes $O(\max\{1, N/p\})$ parallel time. Computing the distance between the four candidates and the target vector (step 12) entails the evaluation of summations with N terms, each requiring $\log N$ sequential rounds, and overall $O(\log N + N/p)$ parallel time. As a result, the running time $T_1(p, N)$ of Algorithm 1 on p processors for inputs of size N satisfies the recurrence:

$$T_1(p, N) = \begin{cases} T_2(0, N) & \text{if } p < 4 \text{ or } N = 1 \\ O(\log N + N/p) + T_1(p/4, N/2) & \text{o/w} \end{cases}$$

where $T_2(r, N) = (\log N - r)(N \log N)$ is the running time of the sequential decoder SeqBW (*cf.* Section 3 of Algorithm 1). When $p = N^2 = 4^n$, the recursion unfolds exactly n times and terminates with $p = N = 1$, yielding $T_1(N^2, N) = O(\log^2 N)$. When $p = 4^k$, $k < n$, the running time is dominated by the sequential decoding (*cf.* step 3) of a vector of residual length $N/2^k = N/\sqrt{p}$, yielding $T_1(p, N) = O(T_2(0, N/\sqrt{p})) = O(N \log^2 N/\sqrt{p})$. □

3 The Sequential Bounded Distance Decoder

In this section we present a sequential algorithm for decoding Barnes-Wall lattices up to their squared unique decoding radius. The algorithm is based on the multilevel construction [4, 1] of Barnes-Wall lattices from Reed-Muller codes, and employs the soft decision decoder of [10, 5].

Definition 2. *For any $r \le n$, the Reed-Muller code RM_r^n is the $N = 2^n$ dimensional binary linear code defined by*

$$RM_r^n = \{[p(\mathbf{x}) : \mathbf{x} \in \mathbb{F}_2^n] : p \in \mathbb{F}_2[\mathbf{x}], \deg(p) \le r\}.$$

It follows from the definition that RM_r^n satisfies $\mathrm{RM}_0^n = \{\mathbf{0}, \mathbf{1}\}$, $\mathrm{RM}_n^n = \mathbb{F}_2^N$ and, for $0 < r < n$, $\mathrm{RM}_r^n = \{[\mathbf{u}, \mathbf{u} \oplus \mathbf{v}] : \mathbf{u} \in \mathrm{RM}_r^{n-1}, \mathbf{v} \in \mathrm{RM}_{r-1}^{n-1}\}$. The binary code RM_r^n has block length $N = 2^n$, dimension $k = \sum_{s \le r} \binom{n}{s}$ and minimum distance $d = 2^{n-r}$.

Notice that Reed-Muller codewords are vectors in $\mathbb{F}_2^N$, but for the purposes of our decoding algorithms we need to interpret them as vectors in $\mathrm{BW}^n \subset \mathbb{G}^N$. This can be done via the following linear transformation $\boldsymbol{\psi} : \mathbb{F}_2^N \to \mathbb{G}^N$:

$$\begin{cases} \boldsymbol{\psi}(\mathbf{0}) = \mathbf{0} \\ \boldsymbol{\psi}(\mathbf{1}) = \mathbf{1} \\ \boldsymbol{\psi}([\mathbf{u}, \mathbf{u} \oplus \mathbf{v}]) = [\boldsymbol{\psi}(\mathbf{u}), \boldsymbol{\psi}(\mathbf{u}) + \boldsymbol{\psi}(\mathbf{v})] \end{cases}$$

The relation between Barnes-Wall lattices and Reed-Muller codes can then be described as follows (*cf.* also [3], Section IV.B):

Theorem 3. *Each lattice vector* $\mathbf{v} \in BW^n$ *can be uniquely expressed as*

$$\mathbf{v} = \sum_{r=0}^{n-1} \phi^r \boldsymbol{\psi}(\mathbf{c}_r) + \phi^n \mathbf{c}_n$$

where $\mathbf{c}_n \in \mathbb{G}^N$ *and* $\mathbf{c}_r \in RM_r^n$ *for* $r = 0, \ldots, n-1$.

For any $0 \leq r \leq n$, let

$$\mathrm{BW}_r^n = \Big\{ \sum_{k=r}^{n-1} \phi^{k-r} \boldsymbol{\psi}(\mathbf{c}_k) + \phi^{n-r} \mathbf{c}_n \quad : \quad \mathbf{c}_k \in \mathrm{RM}_k^n, \mathbf{c}_n \in \mathbb{G}^{2^n} \Big\}.$$

be the so-called *principal sublattices* of BW^n (*cf.* [3], Section IV.B). In other words, BW_r^n is the set of all lattice vectors in BW^n such that $\mathbf{c}_0 = \ldots = \mathbf{c}_{r-1} = \mathbf{0}$, scaled by a factor ϕ^r. It is clear that each set BW_r^n is itself a lattice, *i.e.*, it is closed under addition and subtraction.

Algorithm 2 above defines a sequential decoder $\textsc{SeqBW}(r, \mathbf{s})$ for this family of lattices. When $r = 0$, $\textsc{SeqBW}(0, \mathbf{s})$ gives a decoder for $\mathrm{BW}_0^n = \mathrm{BW}^n$.

Theorem 4. *For any* $N = 2^n$, $r \leq n$, *and* $\mathbf{s} \in \mathbb{C}^N$ *such that* $dist^2(\mathbf{s}, BW_r^n) < N/2^{r+2}$, *Algorithm 2 computes the (unique) lattice vector* $\mathbf{z} \in BW_r^n$ *within squared distance* $N/2^{r+2}$ *from* $\mathbf{s}$.

In order to complete the description of the sequential decoding algorithm, we need to give a soft decision decoder for Reed-Muller codes in Euclidean space. Algorithm 3 is essentially the soft-decision decoder of [10], with the following differences: 1) our algorithm uses additive $0, 1$ notation for vectors, whereas [10] represents codewords as vectors in $\{-1, +1\}^N$; 2) we combine the soft-decision decoding of the Reed-Muller code with the linear embedding $\boldsymbol{\psi} \colon \mathrm{RM}_r^n \to \mathrm{BW}_r^n$. We remark that the image of $\{0, 1\}^N$ under $\boldsymbol{\psi}$ is a subset of $\mathbb{Z}^N$. So, on input a vector $\mathbf{t} \in (\{0, 1\} \times [0, 1])^N$, Algorithm 3 outputs a vector $\textsc{RMdec}^{\boldsymbol{\psi}}(r, \mathbf{t}) \in \mathbb{Z}^N \cap \mathrm{BW}_r^n$.

For any $N = 2^n$, $0 \leq r \leq n$, and $\mathbf{t} \in (\{0, 1\} \times [0, 1])^N$, the running time $T_3(r, N)$ of Algorithm 3 is described by the recurrence:

$$T_3(r, N) = \begin{cases} O(N) & \text{if } r = 0 \text{ or } N = 2^r \\ O(N) + T_3(r-1, N/2) + T_3(r, N/2) & \text{o/w} \end{cases}$$

which is easily seen to satisfy:

$$T_3(r, N) = O(N \log N).$$

Since Algorithm 2 essentially amounts to iterative decoding of length-N Reed-Muller codewords of order ranging from r to $(\log N - 1)$, it follows that its running time grows asymptotically as:

$$T_2(r, N) = O(\log N - r)(N \log N).$$

4 Open Problems

Our investigation on efficient bounded distance decoding for Barnes-Wall lattices brings up several questions and directions for future work: Is it possible to improve the efficiency of the BDD

Algorithm 3 Soft Decision Decoder for Reed-Muller Codes

```
function RMDEC^ψ(r, t)                                   ▷ Input: r ≥ 0, t ∈ ({0,1} × [0,1])^N
    if r = 0 then
        if Σ_{b_j=0} ρ_j > Σ_{b_j=1} ρ_j then
            return [0, ..., 0]
        else
            return [1, ..., 1]
        end if
    else if N = 2^r then
        return [b_1, ..., b_N]                            ▷ where (b_j, ρ_j) = t_j
    else
        [t^0, t^1] ← t                                    ▷ Split t into halves
        for j = 1, ..., N/2 do
            t_j^+ ← (b_j^0 ⊕ b_j^1, min(ρ_j^0, ρ_j^1))     ▷ where (b_j^0, ρ_j^0) = t_j^0 and (b_j^1, ρ_j^1) = t_j^1
        end for
        v ← RMDEC^ψ(r − 1, t^+)
        for j = 1, ..., n/2 do
            if b_j^0 ⊕ b_j^1 = v_j mod 2 then
                t_j^- ← (b_j^0, (ρ_j^0 + ρ_j^1)/2)
            else
                t_j^- = (b_j^0 ⊕ EVAL(ρ_j^0 < ρ_j^1), |ρ_j^0 − ρ_j^1|/2)   ▷ where EVAL(φ) = 1 iff formula φ holds
            end if
        end for
        u ← RMDEC^ψ(r, t^-)
        return [u, u + v].
    end if
end function
```

algorithm given in this paper? In particular, is it possible to reduce the sequential running time from $O(N \log^2 N)$ to $O(N \log N)$? Is it possible to reduce the circuit depth of the parallel algorithm from $O(\log^2 N)$ to $O(\log N)$, without increasing the circuit size beyond polynomial? Is it possible to reduce the circuit size from $O(N^2)$ to $O(N \log N)$, while maintaining poly-logarithmic circuit depth? More generally, can the complexity of the generic algorithm (for arbitrary p) be improved from $O(N \log^2 N/\sqrt{p})$ to $O(N \log N/p)$?

On a different front, is it possible to efficiently decode Barnes-Wall lattices beyond the squared unique decoding radius? Can the maximum-likelihood decoding problem (*i.e.*, the closest vector problem) be solved in polynomial time? Is it possible to list decode BW^n, and up to what radius?

Acknowledgments

The authors thank Alex Vardy, Ilya Dumer and the anonymous referees for interesting conversations and pointers to the literature. The first author was supported in part by NSF grant CCF-0634909. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] D. Agrawal and A. Vardy. Generalized minimum-distance decoding in Euclidean space: performance analysis. *IEEE Transactions on Information Theory*, 46(1):60–83, 2000.

[2] A. Banihashemi and I. Blake. Trellis complexity and minimal trellis diagrams of lattices. *IEEE Transactions on Information Theory*, 44(5):1829–1847, 1998.

[3] G.D. Forney. Coset codes. II. Binary lattices and related codes. *IEEE Transactions on Information Theory*, 34(5):1152–1187, 1988.

[4] G.D. Forney and A. Vardy. Generalized minimum-distance decoding of Euclidean-space codes and lattices. *IEEE Transactions on Information Theory*, 42(6):1992–2026, 1996.

[5] G. Kabatyanskii. On decoding of Reed-Muller codes in semicontinuous channels. In *Intl. Workshop on Algebra and Combinatorial Coding Theory*, pages 87–91, 1990.

[6] G. Nebe, E. Rains, and N. Sloane. The invariants of the Clifford groups. *Designs, Codes and Cryptography*, 24(1):99–122, 2001.

[7] G. Nebe, E. Rains, and N. Sloane. A simple construction for the Barnes-Wall lattices. Presented at the Forney Festschrift at MIT, based on [6], 2002.

[8] M. Ran and J. Snyders. Efficient decoding of the Gosset, Coxeter-Todd and the Barnes-Wall lattices. In *IEEE International Symposium on Information Theory*, page 92, 1998.

[9] A. Salomon and O. Amrani. Augmented product codes and lattices: Reed-Muller codes and Barnes-Wall lattices. *IEEE Transactions on Information Theory*, 51(11):3918–3930, 2005.

[10] G. Schnabl and M. Bossert. Soft-decision decoding of Reed-Muller codes as generalized multiple concatenated codes. *IEEE Transactions on Information Theory*, 41(1):304–308, 1995.

[11] K. Wahlgren and Z.-X. Wan. Iterated squaring construction of bi-infinite group partition chain. In *IEEE International Symposium on Information Theory*, page 442, 1997.

[12] C. Wang, B. Shen, and K. Tzeng. Generalised minimum distance decoding of Reed-Muller codes and Barnes-Wall lattices. In *IEEE International Symposium on Information Theory*, page 186, 1995.