On the decoding of Barnes-Wall lattices

Vincent Corlay, Joseph Boutros, Philippe Ciblat, Loïc Brunel

On the decoding of Barnes-Wall lattices

Vincent Corlay[†,*], Joseph J. Boutros[‡], Philippe Ciblat[†], and Loïc Brunel[*]
[†] Telecom Paris,Institut Polytechnique de Paris, 91120 Palaiseau, France, v.corlay@fr.merce.mee.com
[‡] Texas A&M University, Doha, Qatar, [*]Mitsubishi Electric R&D Centre Europe, Rennes, France

***Abstract*—We present new efficient recursive decoders for the Barnes-Wall lattices based on their squaring construction. The analysis of the new decoders reveals a quasi-quadratic complexity in the lattice dimension. The error rate is shown to be close to the universal lower bound in dimensions 64 and 128.**

I. Introduction

Barnes-Wall (BW) lattices were one of the first series discovered with an infinitely increasing fundamental coding gain [2]. This series includes dense lattices in lower dimensions such as D_4, E_8, Λ_{16} [5], and is deeply related to Reed-Muller codes [8][16]: BW lattices admit a Construction D based on these codes. Multilevel constructions attracted the recent attention of researchers, mainly Construction C* [3], where lattice and non-lattice constellations are made out of binary codes. One of the important challenges is to develop a lattice with a reasonable-complexity decoding where a fraction of the fundamental coding gain is sacrificed in order to achieve a lower kissing number. BW lattices are attractive in this sense. For instance the lattice BW_{128}, with an equal fundamental coding gain as $Nebe_{72}$ [19], sacrifices 1.5dB of its coding gain with respect to MW_{128} [7] while the kissing number is reduced by a factor of 200.

Several algorithms have been proposed to decode BW lattices. Forney introduced an efficient maximum-likelihood decoding (MLD) algorithm in [8] for the low dimension instances of these lattices based on their trellis representation. Nevertheless, the complexity of this algorithm is exponential in the dimension and intractable for $n > 32$: e.g. decoding in BW_{64} involves $2 \cdot 2^{24} + 2 \cdot 2^{16}$ decoders of BW_{16} and decoding in BW_{128} involves $2 \cdot 2^{48} + 2 \cdot 2^{32}$ decoders of BW_{32} (using the two-level squaring construction to build the trellis, see [8, Section IV.B]). Later, [18] proposed the first bounded-distance decoders (BDD) running in polynomial time: a parallelisable decoder of complexity $O(n^2)$ and another sequential decoder of complexity $O(n \log^2(n))$. The parallel decoder was generalized in [13] to work beyond the packing radius, still in polynomial time. It is discussed later in the paper. The sequential decoder uses the BW multilevel construction to perform multistage decoding: each of the $\approx \log(n)$ levels is decoded with a Reed-Muller decoder of complexity $n \log(n)$. This decoder was also further studied, in [14], to design practical schemes for communication over the AWGN channel. The performance of this sequential decoder is far from MLD. A simple information-theoretic argument explains why multi-stage decoding of BW lattices cannot be efficient: the rates of some component Reed-Muller codes exceed the channel capacities of the corresponding levels [12][27]. As a result, no BW decoders, being both practical and quasi-optimal on the Gaussian channel, have been designed and executed for dimensions greater than 32.

We present new decoders for the BW lattices based on their $(u, u+v)$ construction [16]. We particularly consider this construction as a squaring construction [8] to establish a new recursive BDD (Algorithm 2, Section III-A), new recursive list decoders (Algorithms 3 and 5, Sections IV-B and IV-C), and their complexity analysis as stated by Theorems 2-4. As an example, Algorithm 5 decodes BW_{64} and BW_{128} with a performance close to the universal lower bound on the coding gain of any lattice and with a reasonable complexity almost quadratic in the lattice dimension.

II. Preliminaries

Lattice. A lattice Λ is a discrete additive subgroup of $\mathbb{R}^n$. For a rank-n lattice in $\mathbb{R}^n$, the rows of a $n \times n$ generator matrix G constitute a basis of Λ and any lattice point x is obtained via $x = zG$, where $z \in \mathbb{Z}^n$. The squared minimum Euclidean distance of Λ is $d(\Lambda) = (2\rho(\Lambda))^2$, where $\rho(\Lambda)$ is the packing radius. The number of lattice points located at a distance $\sqrt{d(\Lambda)}$ from the origin is the kissing number $\tau(\Lambda)$. The fundamental volume of Λ, i.e. the volume of its Voronoi cell and its fundamental parallelotope, is denoted by $\mathrm{Vol}(\Lambda)$. The fundamental coding gain $\gamma(\Lambda)$ is given by the ratio $\gamma(\Lambda) = d(\Lambda)/\mathrm{vol}(\Lambda)^{\frac{2}{n}}$. The squared Euclidean distance between a point $y \in \mathbb{R}^n$ and a lattice point $x \in \Lambda$ is denoted $d(x, y)$. Accordingly, the squared distance between $y \in \mathbb{R}^n$ and the closest lattice point of Λ is $d(y, \Lambda)$.
For lattices, the transmission rate used with finite constellations is meaningless. Poltyrev introduced the generalized capacity [21], the analog of Shannon capacity for lattices. The Poltyrev limit corresponds to a noise variance of $\sigma^2_{max} = \det(\Lambda)^{\frac{2}{n}}/(2\pi e)$ and the point error rate is evaluated with respect to the distance to Poltyrev limit, i.e. σ^2_{max}/σ^2.

BDD, list decoding, and MLD. Given a lattice Λ, a radius $r > 0$, and any point $y \in \mathbb{R}^n$, the task of a BDD is to determine all points $x \in \Lambda$ satisfying $d(x, y) \leq r^2$. If $r < \rho(\Lambda)$, there is either no solution or a unique solution. Additionally, if $d(x, y) < \rho^2(\Lambda)$, we say that y is within the guaranteed error-correction radius of the lattice. If $r \geq \rho(\Lambda)$, there may be more than one solution. In this case, the process is called list decoding rather than BDD. When list decoding is used, lattice points within the sphere are enumerated and the decoded lattice point is the closest to y among them. MLD simply refers to finding the closest lattice point in Λ to any point $y \in \mathbb{R}^n$. If list decoding is used, it means choosing

a decoding radius equal to the covering radius of Λ.

Coset decomposition of a lattice. Let Λ and Λ' be two lattices such that $\Lambda' \subseteq \Lambda$. If the order of the quotient group Λ/Λ' is q, then Λ can be expressed as the union of q cosets of Λ'. We denote by $[\Lambda/\Lambda']$ a system of coset representatives for this partition. It follows that $\Lambda = \bigcup_{x_i \in [\Lambda/\Lambda']} \Lambda' + x_i = \Lambda' + [\Lambda/\Lambda']$.

The BW lattices. Let the scaling-rotation operator $R(2n)$ in dimension $2n$ be defined by the application of the 2×2 matrix

$$R(2) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

on each pair of components. I.e. the scaling-rotation operator is $R(2n) = I_n \otimes R(2)$, where I_n is the $n \times n$ identity matrix and $\otimes$ the Kronecker product. For $\Lambda \subset \mathbb{R}^{2n}$ with generator matrix G, the lattice generated by $G \cdot R(2n)$ is denoted $R\Lambda$.

Definition 1 (The squaring construction of BW_{2n} [8])**.** *The BW lattices in dimension $2n$ are obtained by the following recursion:*

$$BW_{2n} = \{(\underbrace{v_1' + m}_{u_1 \in BW_n}, \underbrace{v_2' + m}_{u_2 \in BW_n}), v_i' \in RBW_n, m \in [BW_n/RBW_n]\},$$

with initial condition $BW_2 = \mathbb{Z}^2$.

Using this construction, it is easily seen that $d(BW_{2n}) = d(RBW_n) = 2d(BW_n)$ and the fundamental coding gain increases infinitely as $\gamma(BW_n) = \sqrt{2} \cdot \gamma(BW_n) = \sqrt{n/2}$ [8]. Note that the squaring construction can be expressed under the form of the Plotkin $(u, u+v)$ construction [20]:

$$\begin{aligned} BW_{2n} &= \{(v_1' + m, v_2' + m),\ v_i' \in RBW_n, m \in [BW_n/RBW_n]\}, \\ &= \{(v_1' + m, \underbrace{v_1' + v_2}_{v_2'} + m)\} = \{(u_1, u_1 + v_2)\}. \end{aligned}$$

III. Bounded-distance BW decoding

A. The new BDD

Given a point $y = (y_1, y_2) \in \mathbb{R}^{2n}$ to be decoded, a well-known algorithm [25][6] for a code obtained via the $(u, u+v)$ construction is to first decode y_1 as u_1, and then decode $y_2 - u_1$ as v_2 [1]. Our lattice decoder, Algorithm 1, is double-sided since we also decode y_2 as u_2 and then $y_1 - u_2$ as v_2: the decoder is based on the squaring construction. The main idea exploided by the algorithm is that if there is too much noise on one side, then there is less noise on the other side, and vice versa.

Algorithm 1 Double-sided $(u, u+v)$ decoder of BW_{2n}

Input: $y = (y_1, y_2) \in \mathbb{R}^{2n}$.
1: Decode (MLD) y_1, y_2 in BW_n as u_1, u_2.
2: Decode (MLD) $y_2 - u_1$ in RBW_n as v_2. Store $\hat{x} \leftarrow (u_1, u_1 + v_2)$.
3: Dec. (MLD) $y_1 - u_2$ in RBW_n as v_1. Store $\hat{x}' \leftarrow (u_2 + v_1, u_2)$.
4: **Return** $x_{dec} = \underset{x \in \{\hat{x}, \hat{x}'\}}{\operatorname{argmin}} ||y - x||$

Theorem 1. *Let y be a point in $\mathbb{R}^{2n}$ such that $d(y, BW_{2n})$ is less than $\rho^2(BW_{2n})$. Then, Algorithm 1 outputs the closest lattice point $x \in BW_{2n}$ to y.*

[1]The standard decoder for $(u, u+v)$ has a second round: once v_2 is decoded u_1 is re-decoded based on the two estimates y_1 and $y_2 - v_2$.

Proof. If $(x_1, x_2) \in BW_{2n}$, then $x_1, x_2 \in BW_n$. Also, we have $||(y_1, y_2)||^2 = ||y_1||^2 + ||y_2||^2$. So if $d(y, BW_{2n}) < \rho^2(BW_{2n})$, then at least one among the two y_i is at a distance smaller than $\frac{\rho^2(BW_{2n})}{2} = \rho^2(BW_n)$ from BW_n and thus no further than $\rho(BW_n)$ from BW_n. Therefore, at least one of the two u_i is correct.
Assume (without loss of generality) that u_1 is correct. We have $d(y_2 - u_1, RBW_n) < \rho^2(BW_{2n}) = \rho^2(RBW_n)$. Therefore, $y_2 - u_1$ is also correctly decoded.
As a result, among the two lattice points stored, at least one is the closest lattice point to y. □

Note that the BW_n decoder in the previous proof got exploited up to $\rho^2(BW_n)$ only. Consequently, Algorithm 1 should exceed the performance predicted by Theorem 1 given that step 1 is MLD.
Algorithm 1 can be generalized into the recursive Algorithm 2, where operations 4, 5, and 6 of the latter algorithm correspond respectively to operations 1, 2, and 3 of Algorithm 1. This algorithm is similar to the parallel decoder of [18]. The main difference is that [18] uses the automorphism group of BW_{2n} to get four candidates at each step of the recursion whereas we use the squaring construction to generate only two candidates at each step. Nevertheless, both our algorithm and [18] use four recursive calls at each recursive step and have the same asymptotic complexity.

Algorithm 2 Recursive BDD of BW_{2n} (where $2n = 2^t$)

Function $RecBW(y, t)$
Input: $y = (y_1, y_2) \in \mathbb{R}^{2^t}$, $1 \le t$.
1: **if** t = 1 **then**
2: $x_{dec} \leftarrow (\lfloor y_1 \rceil, \lfloor y_2 \rceil)$ // Decoding in $\mathbb{Z}^2$
3: **else**
4: $u_1 \leftarrow RecBW(y_1, t-1)$, $u_2 \leftarrow RecBW(y_2, t-1)$
// $y_2 - u_1$ (and $y_1 - u_2$) should be decoded in RBW_n:
// this is equivalent to decoding $(y_2 - u_1) \cdot R(2^{t-1})^{-1}$ in BW_n
// and then rotate the output lattice point by $R(2^{t-1})$.
5: $v_2 \leftarrow RecBW((y_2 - u_1) \cdot R(2^{t-1})^T/2, t-1) \cdot R(2^{t-1})$.
Store $\hat{x} \leftarrow (u_1, u_1 + v_2)$.
6: $v_1 \leftarrow RecBW((y_1 - u_2) \cdot R(2^{t-1})^T/2, t-1) \cdot R(2^{t-1})$.
Store $\hat{x}' \leftarrow (u_2 + v_1, u_2)$.
7: $x_{dec} = \underset{x \in \{\hat{x}, \hat{x}'\}}{\operatorname{argmin}} ||y - x||$
8: **end if**
9: **Return** x_{dec}

Theorem 2. *Let n be the dimension the lattice BW_n to be decoded. The complexity of Algorithm 2 is $O(n^2)$.*

Proof. Let $\mathfrak{C}(n)$ be the complexity of the algorithm for $n = 2^t$. We have $\mathfrak{C}(n) = 4\mathfrak{C}(n/2) + O(n) = O(n^2)$. □

B. Performance on the Gaussian channel

In the Appendix (see [28, Section VII-A]), we show via an analysis of the effective error coefficient of Algorithm 2 that the loss in performance compared to the MLD (in dB) is expected to grow with n.
Our simulations show that there is a loss of ≈0.25dB for $n =$

16, ≈0.5dB for $n = 32$, ≈1.25dB for $n = 64$ (compare $\aleph = 1$ and $\aleph = 20$ on Figure 1) and ≈2.25dB for $n = 128$. As a result, this BDD is not suited for effective decoding of BW lattices on the Gaussian channel. However, it is essential for building efficient decoders as shown in the next section.

IV. List decoding of the BW lattices beyond the packing radius

Let $L(\Lambda, r^2)$ be the maximum number of lattice points of Λ within a sphere of radius r around any $y \in \mathbb{R}^n$. If $\Lambda = BW_n$ we write $L(n, r^2)$. The following lemma is proved in [13].

Lemma 1. *The list size of the BW_n lattices is bounded as [13]:*

- $L(n, r^2) \leq \frac{1}{4\epsilon}$ *if* $r^2 \leq d(BW_n)(1/2 - \epsilon)$, $0 < \epsilon \leq 1/4$.
- $L(n, r^2) = 2n$ *if* $r^2 = d(BW_n)/2$,
- $L(n, r^2) \leq 4n^{16 \log_2(1/\epsilon)}$ *if* $r^2 \leq d(BW_n)(1 - \epsilon)$, $0 < \epsilon \leq 1/2$.

[13] also shows that the parallel BDD of [18], which uses the automorphism group of BW_n, can be slightly modified to output a list of all lattice points lying at a squared distance $r^2 = d(BW_n)(1 - \epsilon)$ from any $y \in \mathbb{R}^n$ in time $O(n^2) \cdot l(n, r^2)^2$. With Lemma 1, this becomes $n^{O(\log(1/\epsilon))}$ for any $r^2 = d(BW_n)(1 - \epsilon), \forall \epsilon > 0$. This result is of theoretical interest: it shows that there exists a polynomial time algorithm in the dimension for any radius bounded away from the minimum distance. However, due to the quadratic dependence in the list size, the complexity of this list decoder rapidely becomes intractable: for $\epsilon = 1/2$, we get a complexity of $O(n^4)$ and for any ϵ greater than $1/2$ it is $O(n^{66})$. Finding an algorithm with quasi-linear dependence in the list-size is stated as an open problem in [13].

In the following, we show that if we use the squaring construction rather than the automorphism group of BW_n we get a quasi-linear complexity in the list size. This enables to get a practical list decoding algorithm up to $n = 128$.

A. Some notations

Notice that $L(n, r^2) = L(RBW_n, 2r^2)$, e.g. both are equal to $2n$ if $r^2 = d(BW_n)/2$. It is therefore convenient to consider the relative squared distance as in [13]: $\delta(x, y) = \frac{2d(x,y)}{n}$. Then, if we define $l(\Lambda, r^2/d(\Lambda)) = L(\Lambda, r^2)$ this yields for instance $l(BW_n, 1/2) = l(RBW_n, 1/2) = 2n$. We call $\delta = r^2/d(\Lambda)$ the relative squared radius. Let $y = (y_1, y_2) \in \mathbb{R}^{2n}$. Then, $x = (u_1, u_1 + v_2) = (u_2 + v_1, u_2) \in BW_{2n}$ is any lattice point where $\delta(x, y) \leq \delta$. We recall that for BDD of BW_n we have $\delta = 1/4$.
The following lemma is trivial, but convenient to manipulate distances.

Lemma 2. *(Lemma 2.1 in [13])*
Let $y = (y_1, y_2) \in \mathbb{R}^{2n}$ and $x = (u_1, u_1 + v_2) \in BW_{2n}$. Then,

$$\delta(x, y) = \delta(u_1, y_1)/2 + \delta(v_2, y_2 - u_1). \quad (1)$$

Finally, let us choose an integer $a = 2/3\delta$ (i.e. such that $\delta = a/2 + a$, see (1)).

B. List-decoding with $r^2 < 3/4 d(BW_n)$

Consider $d(x, y) = d(x_1, y_1) + d(x_2, y_2)$. We split the possible situations into four cases and establish a decoding strategy to recover the lattice points x accordingly.

- $a \leq \delta(u_1, y_1) \leq \delta$ and $\delta(v_2, y_2 - u_1) < a$:
  then, y_1 should be list-decoded in BW_n with a relative squared radius δ and $y_2 - u_1$ list-decoded in RBW_n with a relative squared radius a.
- $\delta(u_1, y_1) < a$ and $a \leq \delta(v_2, y_2 - u_1) \leq \delta$:
  then, y_1 should be list-decoded in BW_n with a relative squared radius a and $y_2 - u_1$ list decoded in RBW_n with a relative squared radius δ.
- The two other cases are the symmetric cases using $x = (u_2 + v_1, u_2)$ instead of $x = (u_1, v_2 + u_1)$.

This analysis yields Algorithm 3 listed below. The "removing step" (11 in bold) is added to ensure that a list with no more than $l(n, \delta)$ elements is returned by each recursive call. The maximum number of points to process by this removing step is $4l(n/2, \delta)l(n/2, a)$. Regarding step 12, using the classical Merge Sort algorithm, it can be done in $O(n \cdot l(n, \delta) \log(l(n, \delta)))$ operations.

Theorem 3. *Let $f(\delta) = -\log_2(1 - \frac{4}{3}\delta)$. Given any point $y \in \mathbb{R}^n$ and $1/4 \leq \delta < 3/4$, Algorithm 3 outputs the list of all lattice points in BW_n lying within a sphere of relative squared radius δ around y in time:*

- $O(n^2 \cdot \log(n))$ *if* $1/4 \leq \delta \leq 3/8$,
- $O(n^2 \cdot \log^2(n))$ *if* $3/8 < \delta \leq 1/2$,
- $O(n^{2+f(\delta)} \log^2(n))$ *if* $1/2 < \delta < 3/4$.

Note that if $\delta < 1/4$, then one should simply use Algorithm 2 of complexity $O(n^2)$.

Proof. Let $\mathfrak{C}(n, \delta)$ be the complexity of Algorithm 3. We have

$$\mathfrak{C}(n, \delta) \leq \underbrace{4\mathfrak{C}(n/2, \delta)}_{\text{Four recursive calls with } \delta} + \underbrace{4\mathfrak{C}(n/2, a)}_{\text{Four recursive calls with } a} + \underbrace{4 \cdot l(n/2, \delta) l(n/2, a) O(n)}_{\text{removing}} + \underbrace{O(n \cdot l(n/2, \delta) \log(l(n/2, \delta)))}_{\text{Merge Sort}}.$$

If $\delta \leq 3/8$, then $l(n, \delta) \leq 2$, $l(n, a) \leq 1$, $4\mathfrak{C}(n/2, a) \leq 4\mathfrak{C}(n/2, 1/4) = O((n/2)^2)$ (the complexity of Algorithm 2). Hence, the complexity becomes $\mathfrak{C}(n, \delta) \leq 4\mathfrak{C}(\frac{n}{2}, 3/8) + O(n^2) = O(n^2 \log(n))$.

If $\delta \leq 1/2$, then $l(n, \delta) \leq 2n$, $l(n, a) \leq 2$, $4\mathfrak{C}(n/2, a) \leq 4\mathfrak{C}(n/2, 3/8) = O(n^2 \log(n))$. Hence, the complexity becomes $\mathfrak{C}(n, \delta) \leq 4\mathfrak{C}(\frac{n}{2}, 1/2) + O(n^2 \log(n)) = O(n^2 \log^2(n))$.

For the case $1/2 < \delta < 3/4$, we first need to compute $4l(n/2, a) \cdot l(n/2, \delta)$, the maximum number of points to be processed at each recursive step of the algorithm (the removing step 11).

$$4l(n/2, a) \cdot l(n/2, \delta) \leq \frac{2}{1 - \frac{4}{3}\delta} \cdot l(n/2, \delta),$$

$$= \left(\frac{2}{1 - \frac{4}{3}\delta}\right)^t \cdot 4 = n^{1 - \log_2(1 - \frac{4}{3}\delta)} \cdot 4 = O(n^{1 - \log_2(1 - \frac{4}{3}\delta)}).$$

Let us define $f(\delta) = -\log_2(1-\frac{4}{3}\delta)$. Then, we have $4\mathfrak{C}(n/2,a) \leq 4\mathfrak{C}(n/2,1/2) = O(n^2\log^2(n))$. Hence, the complexity becomes $\mathfrak{C}(n,\delta) \leq 4\mathfrak{C}(\frac{n}{2},3/4) + O(n^{2+f(\delta)} \cdot f(\delta)\log(n)) = O(n^{2+f(\delta)}\log^2(n))$.

□

Algorithm 3 First recursive list decoding of BW_{2n} ($2n = 2^t$).

Function $ListRecBW(y,t,\delta)$
Input: $y = (y_1,y_2) \in \mathbb{R}^{2^t}$, $1 \leq t$, $1/4 \leq \delta < 3/4$.

$a \leftarrow 2/3 \cdot \delta$
$r \leftarrow \sqrt{2^{t-1}\cdot\delta}$
if t = 1 **then**
  $\hat{x} \leftarrow Enum_{\mathbb{Z}_2}(y,r)$ // Enum. in $\mathbb{Z}^2$ with radius $r = \sqrt{\delta}$.
else
  $\hat{x}_1 \leftarrow SubRoutine(y_1,y_2,t,a,\delta,0)$
  $\hat{x}_2 \leftarrow SubRoutine(y_1,y_2,t,\delta,a,0)$
  $\hat{x}_3 \leftarrow SubRoutine(y_2,y_1,t,\delta,a,1)$
  $\hat{x}_4 \leftarrow SubRoutine(y_2,y_1,t,a,\delta,1)$
  Remove all candidates at a distance $> r$ **from** y.
  Sort the remaining list of candidates in a lexicographic order and remove all duplicates.
end if
Return the list of all the candidates remaining.

Algorithm 4 Subroutine of Algorithms 3 & 5

Function $SubRoutine(y_1,y_2,t,\delta_1,\delta_2,reverse)$
Input: $y_1,y_2 \in \mathbb{R}^{2^{t-1}}$, $1 \leq t$, $0 < \delta_1,\delta_2 \leq 3/4$, $rev. \in \{0,1\}$.

if $\delta_1 \leq 1/4$ **then**
  $u_1\_List \leftarrow RecBW(y_1,t-1)$
else
  $u_1\_List \leftarrow ListRecBW(y_1,t-1,\delta_1)$
end if
for $u_1 \in u_1\_List$ **do**
  if $\delta_2 \leq 1/4$ **then**
    $v_2\_List(u_1) \leftarrow RecBW((y_2-t_1)\cdot R(2^{t-1})^T/2, t-1)\cdot R(2^{t-1})$
  else
    $v_2\_List(u_1) \leftarrow ListRecBW((y_2-t_1)\cdot R(2^{t-1})^T/2, t-1,\delta_2)\cdot R(2^{t-1})$
  end if
end for
for $u_1 \in u_1\_List$ **do**
  for $v_2 \in v_2\_List(u_1)$ **do**
    if $reverse = 0$ **then**
      Compute and store $\hat{x} \leftarrow (u_1, v_2+u_1)$.
    else
      Compute and store $\hat{x} \leftarrow (v_2+u_1, u_1)$.
    end if
  end for
end for
Return the list of all candidates $\hat{x}$.

Unfortunately, the performance of Algorithm 3 on the Gaussian channel is disappointing. This is not surprising: notice that due to the "removing step" (in bold), some points that are correctly decoded by Algorithm 2 (the BDD) are not in the list outputted by Algorithm 3! Therefore, instead of removing all candidates at a distance greater than r, it is tempting to keep $\aleph$ candidates at each step.

C. An efficient list decoder on the Gaussian channel

Algorithm 5 Second rec. list decoding of BW_{2n} ($2n = 2^t$)

Function $ListRecBW(y,t,\delta)$
Input: $y = (y_1,y_2) \in \mathbb{R}^{2^t}$, $1 \leq t$, $1/4 \leq \delta < 3/4$.
Global variables: $\{\aleph(\delta)\}$.

$a \leftarrow 2/3 \cdot \delta$
$r \leftarrow \sqrt{2^{t-1}\cdot\delta}$
if t = 1 **then**
  $\hat{x} \leftarrow Enum_{\mathbb{Z}_2}(y,r)$ // Enum. in $\mathbb{Z}^2$ with radius $r = \sqrt{\delta}$.
else
  $\hat{x}_1 \leftarrow SubRoutine(y_1,y_2,t,a,\delta,0)$
  $\hat{x}_2 \leftarrow SubRoutine(y_1,y_2,t,\delta,a,0)$
  $\hat{x}_3 \leftarrow SubRoutine(y_2,y_1,t,\delta,a,1)$
  $\hat{x}_4 \leftarrow SubRoutine(y_2,y_1,t,a,\delta,1)$
  Sort the candidates from the closest to the furthest to y and remove all duplicates.
  Keep the $\aleph(\delta)$ **closest candidates to** y.
end if
Return the list of all the candidates remaining.

Note: $\aleph(\delta)$ means that the number of candidates kept depends on δ.

Algorithm 5 is a modified version of Algorithm 3 where $\aleph(\delta)$ candidates are kept at each recursive step. The size of the list $\aleph(\delta)$, for a given δ, is a parameter to be fine tuned: e.g. for $\delta = 1/2$, one needs to chose $\aleph(1/2)$ and $\aleph(2/3\cdot 1/2 = 1/3)$. The following theorem follows from Theorem 3.

Theorem 4. *The complexity of Algorithm 5 is:*

- $O\left(\max\left(n^2\aleph(\delta)\log(\aleph(\delta)), n^2\log(n)\right)\right)$ *with* $\delta \leq 3/8$.
- $O\left(\max\left(n^2\aleph'\log(\aleph'), n^2\log^2(n)\right)\right)$ *with* $3/8 < \delta \leq 1/2$ *(where* $\aleph' = \aleph(2/3\cdot\delta)\cdot\aleph(\delta)$*).*

V. Numerical results

A. Performance of Algorithm 5

Figure 1 shows the influence of the list size when decoding BW_{64} using Algorithm 5 with $\delta = 3/8$. On this figure we also plotted an estimate of the MLD performance of BW_{64}, obtained as $\tau(BW_{64})/2\cdot\text{erfc}(\gamma/(8\sigma^2_{max}/\sigma^2))$ [5, Chap. 3].

Figure 2 depicts the performance of Algorithm 5 for the BW lattices up to $n = 128$ and the universal bounds provided in [26] (see also [12] or [15], where it is called the sphere lower bound). This universal bound is a limit on the highest possible coding gain using *any* lattice in n dimensions. For each BW_n we tried to reduce as much as possible the list size while keeping quasi-MLD performance. The choice of $\delta = 3/8$ yields quasi-MLD performance up to $n = 64$ with small list size and thus reasonable complexity. This shows that BW_{64}, with Algorithm 5, is a good candidate to design finite constellations in dimension 64. However, for $n = 128$ one needs to set $\delta = 1/2$ and choose $\aleph(\delta) = 1000$. Nevertheless, $\aleph(2/3\cdot\delta)$ can be as small as 4, which is still tractable.

In the litterature, several constructions have been proposed for block-lengths around $n = 100$. For fair comparison between the dimensions, P_e is either the normalized error probability, which is equal to the point error rate divided by the dimension (as done in e.g. [26]), or the symbol error rate.

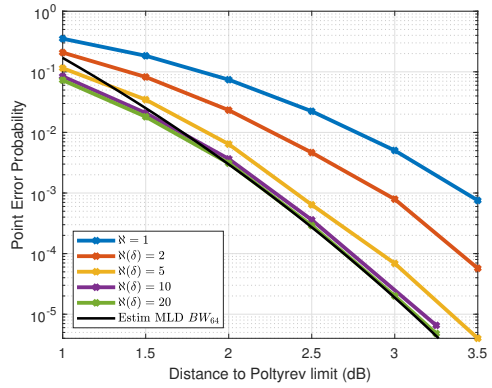

Fig. 1. Influence of the list size when decoding BW_{64} using Alg. 5, $\delta = 3/8$.

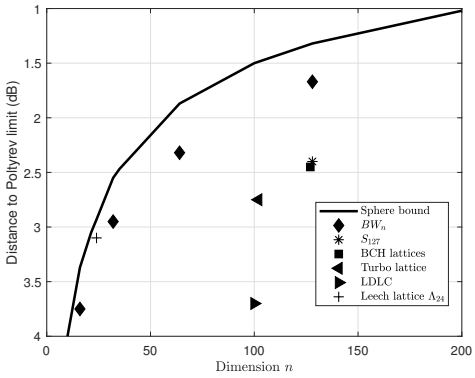

Fig. 3. Perf. of different lattices for normalized error probability $P_e = 10^{-5}$.

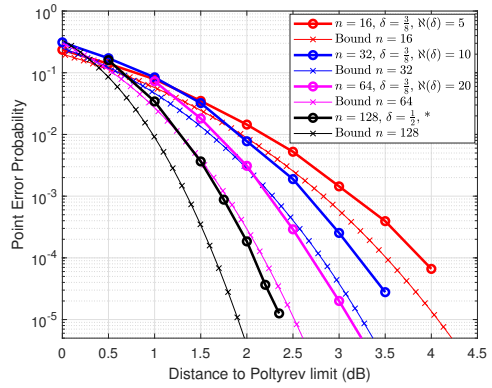

Fig. 2. Algorithm 5 for the BW lattices up to $n = 128$ and the universal bounds of [26]. *For $n = 128$, $\aleph(\delta) = 1000$ and $\aleph(2/3\delta) = 4$.

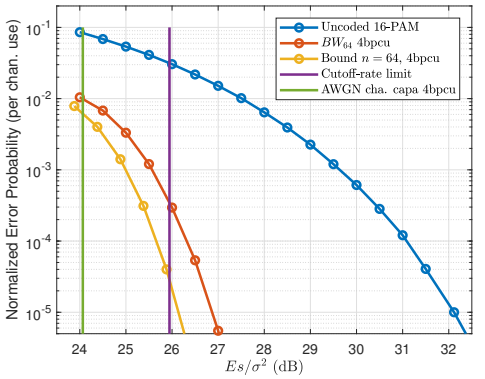

Fig. 4. Performance of a Voronoi constellation based on the partition $BW_{64}/2^4 BW_{64}$ where Algorithm 5, with $\delta = 3/8$ and $\aleph(\delta) = 20$, is used for encoding and decoding. The cutoff-rate limit is 1.7+0.179dB right to Shannon limit (coding + shaping loss for $n = 64$)[11].

We compare the constructions for $P_e = 10^{-5}$.
In [17] a two-level construction based on BCH codes with $n = 128$ achieves this error-rate at 2.4 dB. The decoding involves an OSD of order 4 with 1505883 candidates. In [1] the multilevel (non-lattice packing) $\mathcal{S}_{127}$ ($n = 127$) has similar performance but with much lower decoding complexity via generalized minimum distance decoding. In [22] and [24] a turbo lattice with $n = 102$ and a LDLC with $n = 100$ achieve the error-rate at respectively 2.75 dB and 3.7 dB (unsurprisingly, these two shemes are efficient for larger block-lengths). All these schemes are outperformed by BW_{64}, were $P_e = 10^{-5}$ is reached at 2.3 dB. Moreover, BW_{128} has $P_e = 10^{-5}$ at 1.7 dB, which is similar to many schemes with block-length $n = 1000$ such as the LDLC with $n = 1000$ (1.3 dB) [24], the polar lattices with $n = 1000$ (2.2 dB) [27], the turbo lattices (1.2 dB) [22]. This benchmark is summarized in Figure 3.

B. Performance BW finite constellations

We uncover the performance of a Voronoi constellation [4][9] based on the partition $BW_{64}/2^{\eta} BW_{64}$ via Monte Carlo simulation, where η is the desired rate in bits per channel use (bpcu): i.e. both the coding lattice and the shaping lattice are based on BW_{64}. It follows that the encoding complexity is the same as the decoding complexity: the complexity of Algorithm 5 with $\delta = 3/8$ and $\aleph(\delta) = 20$. Figure 4 exhibits the performance of our scheme for $\eta = 4$bpcu. In our simulation, the errors are counted on the uncoded symbols. The error-rate also includes potential errors due to incomplete encoding, which seem to be negligible compared to decoding errors. Again, we plotted the best possible performance of *any* lattice-based constellation in dimension 64 (obtained from [26]). The scheme performs within 0.7dB of the bound.

VI. Conclusions

Our recursive paradigm can be seen as a tree search algorithm and our decoders fall therefore in the class of sequential decoders. While the complexity of Algorithm 5 remains stable and low for $n \leq 64$, there is a significant increase for $n = 128$ and it becomes intractable for $n = 256$ due to larger lists. This is not surprising from the cut-off rate perspective [11]; For $n = 64$ the MLD is still at a distance of 1dB from this limit (Figure 4), but it is very close to the limit for $n = 128$ and potentially better at larger n. One should not expect to perform quasi-MLD of these lattices with any sequential decoder. This raises the following open problem: can we decode lattices beyond the cut-off rate in non-asymptotic dimensions, i.e. $n < 500$, where classical capacity-approaching decoding techniques (e.g. BP) cannot be used?
As final words, note that our scheme offers performance/complexity trade-off similar to that of trellis-coded modulations [11] (the best non-capacity-approaching scheme) but without the need to use large block lengths.

REFERENCES

[1] D. Agrawal and A. Vardy, "Generalized minimum distance decoding in euclidean space: Performance analysis," *IEEE Trans. Inform. Theory*, vol. 46, pp. 60–83, 2000.

[2] E. S. Barnes and G. E. Wall, "Some extreme forms defined in terms of Abelian groups," J. Australian Math. SOC., vol. 1, pp. 47-63, 1959.

[3] M. F. Bollauf, R. Zamir, S. I. R. Costa, "Multilevel Constructions: Coding, Packing and Geometric Uniformity," *IEEE Trans. on Inform. Theory*, Vol. 65, 2019.

[4] J. Conway and N. Sloane, "A fast encoding method for lattice codes and quantizers," *IEEE Trans. Inform. Theory*, vol. 19, pp. 820-824, 1983.

[5] J. Conway and N. J. A. Sloane. *Sphere packings, lattices and groups*. Springer-Verlag, New York, 3rd edition, 1999.

[6] I. Dumer and K. Shabunov, "Soft-decision decoding of Reed-Muller codes: recursive lists," *IEEE Trans. Inform. Theory*, vol. 52, pp. 1260-1266, 2006.

[7] N. D. Elkies, "Mordell-Weil lattices in characteristic 2: III. A Mordell-Weil lattice of rank 128," *Experimental Math.*, vol. 3, pp. 467-473, 2001.

[8] G. D. Forney, Jr., "Coset codes II: Binary lattices and related codes," *IEEE Trans. Inform. Theory*, vol. 34, pp. 1152-1187, 1988.

[9] G. D. Forney, Jr., "Multidimensional Constellations - part II: Voronoi Constellations," *IEEE J. Select. Areas Com.*, vol. 7, pp. 941-958, 1989.

[10] G. D. Forney, Jr. and A. Vardy, "Generalized Minimum-Distance decoding of Euclidean-Space Codes and Lattices," *IEEE J. Select. Areas Com.*, vol. 42, pp. 1992-2026, 1996.

[11] G. D. Forney and G. Ungerboeck, "Modulation and coding for linear Gaussian channels," *IEEE Trans. Inform. Theory*, vol. 44, pp. 2384-2415, 1998.

[12] G. D. Forney, M. D. Trott, and S. Chung, "Sphere-Bound-Achieving Coset Codes and Multilevel Coset Codes," *IEEE Trans. Inform. Theory*, vol. 46, pp. 820-850, 2000.

[13] E. Grigorescu and C. Peikert, "List-decoding Barnes-Wall lattices," *Computational Complexity*, vol. 26, pp 365-39, 2017.

[14] J. Harshan, E. Viterbo, and J.-C. . Belfiore, "Practical Encoders and Decoders for Euclidean Codes from Barnes-Wall Lattices," *IEEE Trans. Communications*, vol. 61, pp. 4417-4427, 2013.

[15] A. Ingber, R. Zamir, and M. Feder, "Finite-dimensional infinite constellations," *IEEE Trans. Inform. Theory*, vol. 59, pp. 1630-1656, 2013.

[16] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.

[17] T. Matsumine, B. M. Kurkoski, and H. Ochiai, "Construction D Lattice Decoding and Its Application to BCH Code Lattices," *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018.

[18] D. Micciancio and A. Nicolosi, "Efficient Bounded Distance Decoders for Barnes-Wall Lattices," *2008 IEEE Int. Symp. Inform. Theory*, 2008.

[19] G. Nebe, "An even unimodular 72-dimensional lattice of minimum 8," *J. Reine Angew. Math.*, vol. 673, pp. 237-247, 2012.

[20] M. Plotkin, "Binary codes with specified minimum distances," *IEEE Trans. Inform. Theory*, vol. 6, pp. 445-450, 1960.

[21] G. Poltyrev, "On coding without restrictions for the AWGN channel," *IEEE Trans. Inform. Theory*, vol. 40, pp. 409-417, 1994.

[22] A. Sakzad, M. Sadeghi, and D. Panario, "Turbo Lattices: Construction and Performance Analysis," arXiv preprint arXiv:1108.1873, 2011. *48th Annual Allerton Conf. on Communication, Control, and Computing* , 2011.

[23] A. J. Salomon and O. Amrani, "Encoding and Decoding Binary Product Lattices," *IEEE Trans. Inform. Theory*, vol 52, pp 5485-5495, 2006.

[24] N. Sommer, M. Feder, and O. Shalvi, "Low-density lattice codes," *IEEE Trans. Inform. Theory*, vol. 54, pp. 1561-1585, 2008.

[25] G. Schnabl and M. Bossert, "Soft-decision decoding of Reed-Muller codes as generalized multiple concatenated codes," *IEEE Trans. on Inform. Theory*, vol. 41, pp. 304-308, 1995.

[26] V. Tarokh, A. Vardy, and K. Zeger, "Universal bound on the performance of lattice codes," *IEEE Trans. Inform. Theory*, vol. 45, pp. 670-681, 1999.

[27] Y. Yan, C. Ling, X. Wu, "Polar lattices: Where Arıkan meets Forney," *2013 IEEE Int. Symp. Inform. Theory*, 20013.

[28] The long version of this paper including an appendix is available at: arXiv preprint arXiv:XXX.

VII. Appendix

A. Analysis of the effective error coefficient

Let us define the decision region of a BDD algorithm $\mathbb{R}_{BDD}(\mathbf{0})$ as the set of all points of the space that are decoded to $\mathbf{0}$ by the algorithm. The number of points at distance $\rho(\Lambda)$ from the origin that are not necessarily decoded to $\mathbf{0}$ are called boundary point of $\mathbb{R}_{BDD}(\mathbf{0})$ The number of such points is called effective error coefficient of the algorithm. The performance of BDD algorithms are usually estimated via this effective error coefficient [10][23]. Indeed, BDD up to the packing radius achieves the best possible error exponent on the Gaussian channel, but the performance might be significantly degraded, compared to MLD, due to a high effective error coefficient.

In [18], the error coefficient of the parallel decoder is not computed and the performance of the algorithm is not assessed on the Gaussian channel. The following analysis of Algorithm 2 is also valid for the parallel decoder [18]. Let us express the point to be decoded as $y = x + \eta$, where $x \in BW_n$ and η is a noise pattern. Scale BW_n such that its packing radius is 1. It is easily seen that any η of the form $(\pm\frac{1}{\sqrt{2^t}}^{2^t}) = (\pm\frac{1}{\sqrt{2^t}}, ..., \pm\frac{1}{\sqrt{2^t}})$, $t = \log_2(n)$, is on the boundary of $\mathbb{R}_{BDD}(\mathbf{0})$. The number of such noise patterns is $2^{2^t} = 2^n$. According to Forney's rule of thumb, every factor-of-two increase in the number of nearest neighbor results in a 0.2dB loss in effective coding gain [11]. Since the kissing number of BW_n is $\prod_{i=1}^{t}(2^i+2) \approx 4.768...\cdot 2^{0.5log_2 n(log_2 n+1)}$ [5], to be compared to the above number of noise patterns 2^n, we see that the loss in performance compared to the MLD (in dB) is expected to grow as $\approx 0.2n$. However, this rule holds only if the effective error coefficient is not too large and the performance of Algorithm 2 is not as bad in practice. Nevertheless, this analysis hints that one should expect the performance of this BDD to degrade as n increases.