Low Density Lattice Codes for the Relay Channel

Nuwan S. Ferdinand*, Matthew Nokleby†, Behnaam Aazhang‡
Centre for Wireless Communications, University of Oulu, Finland *
† Rice University, Texas, USA ‡
nuferdin@ee.oulu.fi*, {nokleby†, aazhang‡}@rice.edu

Abstract—We study practical, efficient codes for the Gaussian relay channel. It has been demonstrated that low-density lattice codes (LDLCs) can provide near-capacity performance for point-to-point Gaussian channels. We present an LDLC formulation that provides performance near the decode-and-forward inner bound of the relay channel capacity. We employ a superposition block Markov strategy tailored to LDLCs and design an appropriate iterative decoder. We characterize the error performance via simulations, showing that our scheme achieves performance only 2dB away from the decode-and-forward bound.

I. INTRODUCTION

The relay channel has emerged as a promising cooperative modality for wireless communications. The relay facilitates communication between the source and destination, providing increased robustness, higher transmission efficiency, and/or larger coverage range. The relay channel was studied by Cover and El Gamal [2], who proposed the decode-and-forward (DF) encoding strategy, in which the relay decodes the entirety of the source’s message in order to assist. While this approach has seen numerous applications [3], [4], it achieves capacity only in a few special cases. The bulk of these approaches is based on random Gaussian coding, which precludes practical implementation.

Lattice codes are the Euclidean-space analog of linear codes. It has been shown that lattice codes can achieve the capacity of AWGN channels [5]–[7]. Recently, the use of lattice codes in relay network has received significant interest [13]–[17], and it was shown in [15], [17] that lattice codes can achieve the DF rates for the relay channel. However, these achievable schemes rely on asymptotic code lengths, which again precludes practical implementation.

Low-density lattice codes (LDLC) [1] are a family of practical, low-complexity lattice codes inspired by low-density parity check (LDPC) codes. In addition to having linear encoding and decoding complexity, LDLCs have been shown to approach the capacity of the AWGN channel. A few other practical lattice schemes have been proposed, such as multilevel LDPC codes [9] or non-binary LDPCs [10], however, LDLC has become a viable solution due to the fact that LDLC uses same real algebra in both the encoder and channel, which is natural for the continuous-valued AWGN channel [1]. Power shaping methods for LDLC are proposed in [11] and efficient LDLC decoding is studied in [12].

In this work we propose practical lattice codes for the relay channel. Based on the scheme of [15], we construct an LDLC encoding based on superposition block Markov encoding. We develop a decode-and-forward style scheme, and adapt the LDLC decoder of [11] to our approach. Simulation results indicate that our approach is particularly effective, achieving rates within 2dB of the decode-and-forward inner bound using practical-length codes.

II. SYSTEM MODEL

We consider a three-terminal relay channel, as depicted in Figure 1. The source transmits a message to both relay and destination in the next time phase relay aids the destination by sending the part of the information of the previous time slot. We assume a full-duplex relay which can simultaneously transmit and receive. In the relay channel, the source and the relay transmit messages $x_S$ and $x_R$ and the relay and the destination receive $y_R$ and $y_D$ as

$$y_R = h_{SR} \sqrt{P_S} x_S + z_R$$

$$y_D = h_{SD} \sqrt{P_S} x_S + h_{RD} \sqrt{P_R} x_R + z_D$$

where $P_1 E[x_S^2]$ and $P_2 E[x_R^2]$ are the transmit powers at source and relay, and $z_R \sim \mathcal{N}(0, N_r)$ and $z_D \sim \mathcal{N}(0, N_d)$. Further, $h_{SR} = d_1^{-\alpha_1}$, $h_{SD} = 1$ and $h_{RD} = d_2^{-\alpha_2}$ are the path loss channel from source to relay, source to destination and relay to destination. Here we consider the distance between source to destination is 1 and $d_1$ and $d_2$ represent the distance between source to relay and relay to destination respectively and $\alpha_1$ and $\alpha_2$ are being corresponding path loss exponents. The capacity of this channel is unknown in general; however, in [2] the decode-and-forward scheme is proposed, which achieves the following inner bound:

$$R \leq \min \left\{ \frac{1}{2} \log \left( 1 + \frac{h_{SR}^2 P_1 E[x_S^2]}{N_r} \right), \right. \frac{1}{2} \log \left( 1 + \frac{h_{SD}^2 P_1 E[x_S^2] + h_{RD}^2 P_2 E[x_R^2]}{N_d} \right) \right\}$$

This rate is achieved via block Markov encoding. The sequel we adapt this approach to LDLCs to construct a practical coding scheme for the relay channel.

III. LOW DENSITY LATTICE CODES

A lattice $\Lambda \subset \mathbb{R}^n$ is a discrete additive subgroup of the Euclidean space $\mathbb{R}^n$. For any two lattice points $\lambda_1, \lambda_2 \subset \Lambda$, then $\lambda_1 + \lambda_2 \subset \Lambda$. An n-dimensional lattice $\Lambda$ is defined by

\[ \Lambda = \{ \lambda_1 + \cdots + \lambda_n : \lambda_1, \cdots, \lambda_n \in \mathbb{Z} \} \]
the low degree columns of \( H \) are related to the low degree columns of \( H \) where \( \chi \) is a vector whose elements are chosen randomly from the set \{1, 2, ..., \( n \)\} with \( \chi_i \neq \chi_j, \forall i \neq j \). Let \( b_r \in \mathbb{Z}^n \) be the vestigial component vector and the \( i^\text{th} \) element of \( b_v \) can be given as

\[
b_{ri} = \begin{cases} b_i, & i \in \chi \\ 0, & i \notin \chi \end{cases}
\]

Then we define the resolution lattice codeword \( x_r \) and the vestigial lattice codeword \( x_v \) as,

\[
x_r = Gb_r,
\]

and similarly,

\[
x_v = Gb_v
\]

It is straightforward to verify that the original lattice codeword \( x = Gb \) is the sum of its resolution and vestigial components:

\[
x = Gb = G[b_r + b_v]
\]

\[
x = x_r + x_v
\]

where the first equality follows from the definitions of \( b_r \) and \( b_v \).

A. Power constraint for decomposition

It is observed from the above that the codewords \( i.e., x, x_r \) and \( x_v \) have unconstrained power due to the fact that we still have not enforced shaping for the lattice. Although linear decomposition is possible in the unconstrained power case, it is not trivial with constrained power scenario. We propose the following method to enforce the power constraint.

First, we consider the hypercube shaping in which the elements of lattice codeword are uniformly distributed over finite length, hence, the power constraint of the lattice code is preserved. In order to obtain hypercube shaping, we first

1Alternatively, we could choose \( \chi = \{1, \ldots, k\} \). However, in this case the lattice codeword generated from \( b_v \) contains zeros in first \( k \) elements due the lower-triangular structure of \( H \).
map the information integer vector \( b \) to \( b' \) such that the \( i^{th} \) element of \( b' \) is given as,

\[
b'_i = b_i - L_i s_i \tag{11}\]

The special case of hypercube shaping \( s_i \) is given by [11],

\[
s_i = \left[ \frac{1}{L_i} \left( b_i - \sum_{j=1}^{i-1} H_{i,j} x'_j \right) \right] \tag{12}\]

Then the \( i^{th} \) element of the lattice codeword \( x' \) is given as,

\[
x'_i = b'_i - \sum_{j=1}^{i-1} H_{i,j} x'_j. \tag{13}\]

Now we decompose the original integer information vector \( b \) as in (6) and (7). Then we map the resolution component \( b_r \) to \( b'_r \) such that the new integer vector results in a power-constrained codeword:

\[
b'_{ri} = b_{ri} - L_i s_{ri} = \begin{cases} 
    b_i - L_i s_{ri}, & i \in \chi \\
    -L_i s_{ri}, & i \notin \chi 
\end{cases} \tag{14}\]

where \( b_{ri} \) and \( b'_{ri} \) are the \( i^{th} \) elements of \( b_r \) and \( b'_r \) respectively. For hypercube shaping, \( s_{ri} \) can be written as,

\[
s_{ri} = \begin{cases} 
    \left[ \frac{1}{L_i} \left( b_i - \sum_{j=1}^{i-1} H_{i,j} x'_j \right) \right] & i \in \chi \\
    \left[ \frac{1}{L_i} \left( -\sum_{j=1}^{i-1} H_{i,j} x'_j \right) \right] & i \notin \chi 
\end{cases} \tag{15}\]

Then the \( i^{th} \) element of mapped lattice codeword \( x'_r \) is given as,

\[
x'_{ri} = b'_{ri} - \sum_{j=1}^{i-1} H_{i,j} x'_{rj} \tag{16}\]

In order to preserve the linearity of the lattice decomposition, we map the vestigial information integer vector \( b_v \) to \( b'_v \) such that the \( i^{th} \) element of \( b'_v \) is given as,

\[
b'_{vi} = b'_v - L_i s_{vi} = \begin{cases} 
    b_i - L_i (s_i - s_{ri}), & i \in \chi \\
    L_i (s_i - s_{ri}), & i \notin \chi 
\end{cases} \tag{17}\]

where \( s_i \) and \( s_{ri} \) is given in (12) and (15) respectively. Then the \( i^{th} \) element of the vestigial codeword \( x'_v \) can be written as

\[
x'_{vi} = b'_{vi} - \sum_{j=1}^{i-1} H_{i,j} x'_{vj} \tag{18}\]

From (11), (14) and (17), we observe that the original information integers \( b_i, b_{ri} \) and \( b_{vi} \) can be recovered from \( b'_i, b'_{ri} \) and \( b'_{vi} \) by modulo \( L_i \) operation. Note that the lattice codeword \( x \) and the resolution lattice codeword \( x_r \) obey the power constraint. For the case of hypercube shaping, magnitude of each element of \( x \) and \( x_r \) are less that \( L_i/2 \) (i.e. \( |x_i|, |x_r| \leq L_i/2, \forall i \)). Further, it is noticed that the vestigial lattice codeword \( x_v \) does not obey the power constraint generally. However, this is not a problem for the considered relay network which is explained in later section. Although, hypercube shaping has low complexity, further shaping gain of 1.53dB [19] can be achieved by using hypersphere shaping or in other words nested lattice shaping, however, mapping to hypersphere domain is complex and we use the approximation method proposed in [11]. Here, we find the vectors \( s \) and \( s_r \) such that

\[
s^* = \arg \min_{s \in \mathbb{Z}^n} ||x'||^2 \tag{19}\]

\[
s_r^* = \arg \min_{s_r \in \mathbb{Z}^n} ||x'_r||^2 \tag{20}\]

where \( x = G(b - Ls) \) and \( x'_r = G(b_r - Ls_r) \). We use the triangularity of party check matrix \( H \) to find the suboptimal solution of \( M-algorithm \) [18], where we start from first row of \( H \) and sequentially goes down with tree search by keeping up to \( M \) number of sequences [11], [18]. After finding \( s^* \) and \( s_r^* \), the vestigial codeword can be found as \( x'_v = G[b_v + L(s^* - s_r^*)] \).

B. Encoding

The source transmits its signal to both relay and destination and relay also transmits its own signal to destination. The destination therefore receives the superposition of signals from the source and relay. Our encoding scheme is block Markov, meaning that we will encode the message over \( T + 1 \) blocks of \( n \) symbol times each. We first show the encoding for the first two blocks, after which we will generalize to the rest of

\[\text{Fig. 2. Lattice subspace decomposition. Full (x), vestigial (x_r) and resolution (x_v) lattices are shown in subfigures respectively. We used G} = H^{-1} = \left( \begin{smallmatrix} 1 & 0.5 \\ 0 & 0.5 \end{smallmatrix} \right)^{-1} \text{ for full lattice then first and second columns of G are used for resolution and vestigial lattices respectively. Shaping regions are shown in shaded area.}\]
Now relay performs iterative LDLC decoding to obtain the destination only receives the resolution information and the power constraint. At the last block the source has no fresh information to send, hence the destination only receives the resolution information and the power constraint.

C. Decoding

Decoding occurs in three stages. At first stage, the relay decodes \( x' \); in the next stage, the destination decodes \( \hat{x}' \); finally, the destination uses \( \hat{x}' \) to decode \( x' \). First we focus on the decoding at the relay. The received signal at the \( i \)th block at relay is

\[
y_R(i) = h_{SR} \sqrt{P_1} x'(i) + z_R(i) \quad (23)
\]

Now relay performs iterative LDLC decoding to obtain the decoded information vector \( \hat{x}' \) as

\[
\hat{x}' = LDLC_{decod} \left( \frac{y_R(i)}{\sqrt{P_1 h_{SR}}} \right) \quad (24)
\]

where \( LDLC_{decod} \) is the LDLC iterative algorithm which is described in the next section. Let \( e_r(i) \) be the decoding error, which can be written as

\[
e_r(i) = x'(i) - \hat{x}'(i). \quad (25)
\]

The received signal at the destination of block \((i+1)^{th}\) can be written as

\[
y_D(i+1) = h_{SD} x_S(i+1) + h_{RD} x_R(i+1) + z_D(i+1) \quad (26)
\]

Now we substitute (21) and (22) in (26) to obtain

\[
y_D(i+1) = h_{SD} \sqrt{P_1} x'(i+1) + h_{RD} \sqrt{P_2} \hat{x}'(i) + z_D(i+1) \quad (27)
\]

We rewrite the received signal \( y_D(i+1) \) as,

\[
y_D(i+1) = h_{RD} \sqrt{P_2} \hat{x}'(i) + h_{SD} \sqrt{P_1} x'(i+1) - h_{RD} \sqrt{P_2} e_r(i) + z_D(i+1) \quad (28)
\]

where \( e_r(i) \) is given in (25). Now we treat \( h_{SD} \sqrt{P_1} x'(i+1) - h_{RD} \sqrt{P_2} e_r(i) + z_D(i+1) \) as Gaussian noise and decode the \( \hat{x}'(i) = LDLC_{decod} \left( \frac{y_D(i+1)}{\sqrt{P_2 h_{RD}}} \right) \) \((29)\) where \( LDLC_{decod} \) as we will show in Section V, exploits the resolution information to help decode the desired codeword. The decoding error of the resolution information at destination is given by

\[
e_{d1}(i) = x'(i) - \hat{x}'(i) \quad (30)
\]

D. Decoding vestigial information

Now the destination knows \( \hat{x}'(i) \) from \( y_D(i+1) \) and \( \hat{x}'(i-1) \) from \( y_D(i) \). The received signal at \( i \)th block given as,

\[
y_D(i) = h_{SD} \sqrt{P_1} x'(i) + h_{RD} \sqrt{P_2} \hat{x}'(i-1) - e_r(i-1) + z_D(i) \quad (31)
\]

Now we use the linearity property as given in (10) to rewrite (31) as

\[
y_D(i) = h_{SD} \sqrt{P_1} [x'(i) + \hat{x}'(i)] + h_{RD} \sqrt{P_2} [\hat{x}'(i-1) - e_r(i-1)] + z_D(i) \quad (32)
\]

Now we subtract the decoded resolution information to obtain

\[
y_D(i) = h_{SD} \sqrt{P_1} x'(i) + e_{d2}(i) + z_D(i), \quad (33)
\]

where

\[
e_{d2}(i) = h_{SD} \sqrt{P_1} e_{d1}(i) + h_{RD} \sqrt{P_2} [e_{d1}(i-1) - e_r(i-1)] \quad (34)
\]

Now we use \( y_D(i) \) in (33) to decode the vestigial information:

\[
\hat{x}'(i) = LDLC_{decod} \left( \frac{y_D(i)}{\sqrt{P_1 h_{SD}}} \right) \quad (35)
\]

Once we have decoded both the resolution and vestigial lattice codeword, the destination can find the desired lattice codeword by

\[
\hat{x}' = \hat{x}' + \hat{x}' \quad (36)
\]

Then the integer information vector can be recovered from \( \hat{y}' = \lfloor \mathbf{H} \hat{x}' \rfloor \). Then the desired information integers can be
obtained by taking modulo: \( \hat{b} = \bar{b}' \mod L_i \). Error occurs when the destination is not able to decode the information vector correctly. We define a symbol error occurs when,
\[
\hat{b}_i \neq b_i, \quad \forall i = 1, \ldots n \tag{37}
\]
The probability of symbol error is obtained by averaging the number of error over block length of \( n \) and repeating for \( T \) number of blocks.

V. LOW COMPLEXITY LDLC DECODER ALGORITHM

We adapt the iterative LDLC decoder proposed in [1], [12], which has linear complexity, to our block Markov scheme. As analogue to LDPC, here the LDLC decoder is a message passing scheme over bipartite graph and the difference is that in LDPC the messages are scalar values, where in LDLC the messages are real functions over the entire interval. Basically LDLC has two phases; passing messages to the check nodes where check nodes represent the parity check equations (row of parity check matrix \( \mathbf{H} \)) and message passing to the variable nodes which represent the received codeword. Variable nodes send probability density functions (pdfs) and check nodes send periodic extension of pdfs. At \( i^{th} \) check node it convolves all the pdfs received from variable nodes except \( j^{th} \) node and then stretch it by \((-H_{i,j})\) then the result is periodically extended with period \( 1/|H_{i,j}| \) and send it to \( j^{th} \) variable node. At \( j^{th} \) variable node, it multiplies all the periodically extended pdfs received from check nodes and the original received pdf except from \( i^{th} \) check node and then normalize the pdf and sends it to \( i^{th} \) check node. After finishing all the iteration, the \( j^{th} \) variable node multiplies all the received pdfs with original pdf to find the final pdf of the decoded codeword and obtain the \( j^{th} \) element of estimated codeword by finding the peak of final pdf. We specifically use the efficient parametric LDLC decoder given in [12] for LDLCdecoder1, however, for LDLCdecoder2 we make some changes to utilize the known information at the decoder and describe as follows.

When we decode the resolution and vestigial information, many of the elements are zeros; we need to adapt the decoder to exploit this information and improve performance. However, once we perform shaping in order to constraint the power, these integers may not necessarily be zeros. Fortunately, however, both encoder and decoder\(^4\) knows the locations of zeros elements in resolution and vestigial information. While the decoder does not know the exact value of these integers, it is evident from (14) and (17) that these integers are multiples of \( L_i \). We exploit these information at the check node equation of the known zero locations resolution or vestigial information as
\[
x'_i = L_is_i - \sum_{j=1,j\neq i}^{m_i} H_{i,j}x'_j \tag{38}
\]
where \( m_i \) is the number of non zero elements of \( i^{th} \) row of \( \mathbf{H} \). Hence, at the periodic extension step, the decoder extends the pdfs only for the integers that are multiplication of \( L_i \) and it results in better decoding with higher performance.

VI. NUMERICAL RESULTS

In this section we provide a numerical analysis of LDLC for relay channel. For the illustration of the SER curves, we assume \( 50\% \) of the information integers are zero for resolution and vestigial information [i.e. \( k = n/2 \) in (6)] without loss of generality. It is important to note that when performing hypercube shaping, all the elements of whole lattice codeword are uniformly distributed over \((-L/2, L/2)\), hence the average power of \( x_i \) is \( E[x_i^2] = L^2/12 \). However, for the case of resolution or vestigial case the average power is less than \( L^2/12 \) due to the fact that these information vectors contain more zeros. For numerical illustration we use the code lengths of \( n = 100, 1000 \) and \( 10000 \). We perform both hypercube \((M = 1)\) and nested lattice shaping and fix \( M = 21 \) for tree search in nested lattice case. Average power variation in dB for different cases in given in Table II. Since we use low-triangular parity check matrix, it has different degree for each row and TABLE III shows the degrees of each row which we used in the simulation. Moreover, it is observed that the integers which are related to higher rows of the low-triangular parity check matrix are less protected as discussed earlier, hence, we use different integer constellations and it is given in TABLE IV.

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>AVERAGE POWER VARIATION.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( n = 100 )</td>
</tr>
<tr>
<td>Hypercube</td>
<td>7.2073</td>
</tr>
<tr>
<td>Hypercube (resolution)</td>
<td>5.3656</td>
</tr>
<tr>
<td>Nested lattice</td>
<td>6.6598</td>
</tr>
<tr>
<td>Nested lattice (resolution)</td>
<td>4.8144</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>VARIATION OF ROW DEGREE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>degree</td>
<td>( n = 100 )</td>
</tr>
<tr>
<td>1</td>
<td>0 - 5</td>
</tr>
<tr>
<td>2</td>
<td>6 - 10</td>
</tr>
<tr>
<td>3</td>
<td>11 - 20</td>
</tr>
<tr>
<td>4</td>
<td>21 - 35</td>
</tr>
<tr>
<td>5</td>
<td>36 - 100</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
</tr>
</tbody>
</table>

\(^4\)We fix the decomposition of location of zero elements in resolution and vestigial information vectors and are known globally.
(only several points are given here), for $n = 1000$, $P_1 = [10.8, 14.4, 15.2, 16.7, 18.7, 19.7, 22]$ and corresponding $P_2 = 10^{-2} \times [6.3, 11.2, 12.5, 15, 18.9, 20.9, 26]$ are used. Finally, for $n = 10000$, $P_1 = [10.8, 13.9, 15.1, 16]$ and corresponding $P_2 = 10^{-2} \times [6.3, 10.4, 12.4, 13.9]$. Both hypercube and nested lattice shaping ($M = 21$) methods are plotted and one can clearly noticed that nested lattice shaping has approximately $0.5dB$ gain over hypercube shaping.

Fig. 4 shows the comparison of SER performance of with/without relay case. It is clearly evident that presence of relay enhances the performance, infact the improvement of performance increases for higher $n$. For the case of $n = 10000$ we have $0.9dB$ performance improvement by having relay.

One can notice that, $n = 1000$ with relay outperforms $n = 10000$ without the relay case.

VII. CONCLUSION

We have proposed a low complexity encoder/decoder design for decode-and-forward relaying using low density lattice codes. We have studied the symbol error rate performance for our system and it is clearly evident that employment of relay enhances the performance. We have used low complexity hypercube shaping and nested lattice shaping and SER performance is given for both cases. Our encoding/decoding scheme using the combination of superposition, block Markov encoding and LDLC is viable solution for implementation of relay network with lattice codes. As future work these methods can be utilized for other relay scheme such as compress and forward, compute and forward, and multi-user networks.

REFERENCES


