A Protograph Construction for
LDPC Unequal Error Protection Codes

Seyed Jalal Etessami

Supervisor: Werner Henkel
Jacobs University Bremen

August 12, 2009
Abstract

We introduce a new algorithm for constructing LDPC codes based on a Protograph. With Protographs, we have a base graph (mother graph), which is duplicated a number of times followed by a special edge permutation obtaining the largest possible girth. This report presents an algorithm for choosing the permutations for girth maximization given a degree distribution and the corresponding mother graph.
Chapter 1

Introduction

There are lots of obstacles when transmitting data from one point to another. Regardless of the type of the communication channel, almost all of them cause errors in the received data.

One of the most prominent and basic methods of combating such errors is channel coding. In channel coding, we add redundant bits to ensure correct recovery at the receiver.

Many approaches have been proposed for channel coding. From the simplest one known as repetition code, to the complicated ones like low density parity check (LDPC) codes. In 1962, Gallager proposed LDPC codes in his PhD thesis [4]. Following the invention of Turbo codes in 1993 [5], Gallager’s work was rediscovered and especially, the so-called Tanner graphs [6] were used for representation and construction.

LDPC codes are linear codes which means that they can be encoded using their parity check matrix. The main focus of this work is construction approaches for LDPC codes.

One of the most important advantages of LDPC codes is its performance in error correction. Moreover, the decoding complexity of iterative decoding is linear in the codeword length.

The construction of LDPC codes is still an important research area. There are some known construction algorithm such as random construction, zigzag, PEG (progressive edge growth), ACE (approximate cycle extrinsic), combinatorial, and geometry construction.

In this report, we first describe linear block codes. In the following chapter, we discuss LDPC codes and the Tanner graph and after introducing a protograph, we state a new construction algorithm for protographs and compare the result with the PEG algorithm at the end.
Chapter 2

Linear Block Code

An \((N, K)\) linear block code is a subspace with dimension \(K\) of a vector space \(V\) with finite dimension \(N\) over some field \(F\). If \(F = GF(2)\), this is called a binary code. Each \(K\)-tuple of \(x = (x_1, ..., x_K)\) from \(F\) is called a binary information sequence. The codeword is computed by

\[
c = xG,
\]

where \(G\) is the generator matrix of the code. The rate of the code is \(R = K/N\), and the dual space of the code is given by

\[
C^\perp = \{y | y \cdot c^T = 0 \quad \forall c \in C\}.
\]

The dual space of a subspace is also a subspace and the generator matrix of the dual space is called a parity check matrix, where we have

\[
cH^T = 0 \quad \forall c \in C
\]

- The Hamming distance between two codewords \(u\) and \(v\) is defined by the \(l_0\)-norm of \(u - v\) which is defined by

\[
||e||_{l_0} := |\{i : e_i \neq 0\}|
\]

In the binary case, it is equal to the number of nonzero elements of \(u + v\) modulo 2. Similarly, we can define the weight of the codeword \(v\) by the distance between \(v\) and \(0\).
Chapter 3

LDPC code

LDPC codes are linear block codes with a sparse parity check matrix (H). This class of codes is generally described by their dual space. Basically, we can categorize LDPC code based on the structure of its parity check matrix into two cases, known as regular and irregular.

An LDPC code is called regular if it has an equal number of ones in each row and column of its parity check matrix. Here, we denote the number of ones in rows and columns by $\rho$ and $\lambda$, respectively.

The density of a code $D(C)$ which is defined as the ratio of the number of ones to the number of all elements of the parity check matrix, will be $D(C) = \rho/N = \lambda/(N - K)$, for the regular case, if the dimension of $H$ is $(N - K) \times N$. Recall that $(N - K)$ is the number of parity check equations and $N$ is the length of the codewords.

Irregular LDPC codes are known to approach the capacity bound more closely than regular LDPC codes.

3.1 Tanner Graph

An LDPC code can be represented by a bipartite graph $G(V,C,E)$; $V$ and $C$ are the set of variable and check nodes, respectively, and $E$ is the set of edges. The Tanner graph for an LDPC code with parity check matrix $H_{(N-K) \times N}$ has $N$ nodes as variable nodes ($v_i, \ i = 1..N$), each variable node representing a column of $H$, and $(N - K)$ check nodes ($c_j, \ j = 1..(N - K)$), each check node representing a row of $H$. In other words, each variable node represents a code bit and each check node represents a check equation of the code.
Edges connecting check nodes to variable nodes are defined by
\[ e_{ij} = \begin{cases} 1 & \text{if } h_{ji} = 1 \\ 0 & \text{if } h_{ji} = 0 \end{cases}, \]  
(3.1)

where \( e_{ij} \) is the edge between \( v_i \) and \( c_j \).

Example 3.1 This is an example of an irregular low density parity check code:

\[
H = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}
\] 
(3.2)

- In a graph, the number of edges which are connected to a node \( v \) is called the \textit{degree} of that node, \( d_v \).

In this paper, we use the notation \( D_v = \{dv_i, i = 1..n\} \) as the degree of the variable nodes and \( D_c = \{dc_j, j = 1..(n - k)\} \) as the degree of the check nodes.

Corollary 3.1 For any bipartite graph, we must have
\[
\sum_{j=1}^{n} dv_j = \sum_{i=1}^{m} dc_i = |E|, \quad m = n - k.
\] 
(3.3)

- The irregular variable node and check node degree distributions may be defined from an edge perspective by the polynomials
\[
\rho(x) = \sum_{k=1}^{dc_{max}} \rho_k x^{k-1},
\] 
(3.4)
3.1. Tanner Graph

\[ \lambda(x) = \sum_{k=1}^{d_{v_{\max}}} \lambda_k x^{k-1}, \]  

(3.5)

where \( \lambda_k = \frac{N_c(k)}{|E|} \) and \( \rho_k = \frac{N_v(k)}{|E|} \). \( N_c(k) \) and \( N_v(k) \) are the number of check nodes and variable nodes with degree \( k \), respectively.

- A cycle \( g_e \) in a graph is a closed path where each edge is only visited once and the number of edges in a cycle is the length of that cycle, \( g_e \).

If \( \{g_1, ..., g_f\} \) are all the cycles in a graph with lengths \( \{g_1, ..., g_f\} \) then the Girth of the graph is determined by the cycle of minimum length, i.e.,

\[ g = \min\{g_1, ..., g_f\}. \]

(3.6)

Corollary 3.2 For a bipartite graph, the length of each cycle must be an even number. Hence, the girth of an LDPC code is at least 4.

Corollary 3.3 If we have a Tanner graph with \( d_{v_i} > 2 \) for \( i = 1, ..., n \), then for any code word with weight \( d \), there is a cycle with length at most \( 2d \).

Lemma 3.1 The number of vertices \( n \) in a general graph\(^1\) of girth \( g \) and average degree of at least \( d > 2 \), satisfies:

\[ n \geq n_0(d, g), \]

(3.7)

\[ n_0(d, 2r + 1) = 1 + d \sum_{j=0}^{r-1} (d - 1)^i, \]

(3.8)

\[ n_0(d, 2r) = 2 \sum_{j=0}^{r-1} (d - 1)^i. \]

(3.9)

Proof: The proof is in [2].

- A Tree is a connected graph, i.e., there exists a path between any two disjoint nodes, which does not have any cycle. In other words, in a Tree there exists exactly one path between each two nodes.

\(^1\)not necessarily bipartite
Lemma 3.2 For any integer numbers $d_1, \ldots, d_n, c_1, \ldots, c_m$ that fulfill
\[
\sum_{k=1}^{n} d_k = \sum_{i=1}^{m} c_i = |n + m - 1| ,
\] (3.10)
there is a Tanner graph with degree distribution $d_1, \ldots, d_n$ for the variable nodes and $c_1, \ldots, c_m$ for the check nodes, which is a Tree.

Proof: One can prove this by induction on $n + m$. If $n + m = 2$, this is a trivial case because this means just one variable and one check node and one edge between them. Suppose for any $n + m < N$ there exists a bipartite tree with given degrees. In order to prove $n + m = N$, assume, we remove a node with degree one, (every tree has at least two nodes with degree one). If the removed node was a check node, then we decrease the degree of a variable node (with degree greater than one) by one and vice versa. Then, we have new degrees with $\hat{m} + \hat{n} = N - 1$. One can infer from the induction assumption, by this degree distribution there exist a proper tree. In the end, if one adds the removed node and its edge to the graph, a new tree with $m + n = N$ is obtained. \hfill \Box

3.2 Encoding and Decoding

LDPC codes are defined by their dual space, i.e., their parity check matrix. One way for encoding these codes is systematic encoding using the parity check matrix in a systematic form.

\[
H = \begin{bmatrix} A_{M \times K} & I_{M \times M} \\ \end{bmatrix}, \begin{bmatrix} i_1 \\ \vdots \\ i_K \\ j_{K+1} \\ \vdots \\ c_{N} \\ \end{bmatrix} = 0 \Rightarrow \] (3.11)

\[
\sum_{j=1}^{K} a_{ij} j = -c_i , i = K + 1, \ldots, N .
\] (3.12)

Instead of the identity matrix $I_{M \times M}$, a triangular matrix may be used for a ‘systematic’ form, instead.
The general class of decoding algorithms for LDPC codes are message passing algorithms (MPA). The reason for their name is that at each step of the algorithm, some messages are passed from variable nodes to check nodes, and vice versa.
One of the MPA subclasses is Belief Propagation (BP). In this algorithm, messages which are passed along the edges are probabilities or beliefs. The BP is in general less powerful than ML decoding, in other words, BP is a suboptimum algorithm and calculates the corresponding beliefs based on observations correctly, if at every round the incoming messages are statistically independent.
The independency assumption is correct for the $l$ first rounds if the neighborhoods of a variable node up to depth $l$ are a tree. Hence, the length of the minimum cycle, the girth, becomes important.
Chapter 4

ProtoGraph

Let $G = (V, E)$ be a graph and $J > 0$ be an integer number. We can define a new graph $G' = (V', E')$ by copying each node of $G$ $J$ times and then connect nodes of $G'$ with the same structure as in $G$.

Let $S(k) = \{k_1, \ldots, k_J\}$ be the copies of node $k$. Each node in $S(k)$ is connected to a node of $S(l)$ in $G'$ if and only if $k$ is connected to node $l$ in $G$.

Note that there are $J!$ permutations for connecting $S(l)$ to $S(k)$.

$$|V'| = J|V| , \ |E'| = J|E| , \ D(G') = D(G)/J . \quad (4.1)$$

**Example 4.1** The following protograph is based on the previous example using $J = 3$.

![Figure 4.1: A protograph of Example 3.1 with $J = 3$](image)

If we look precisely to the $H$ matrix of the final graph, we can see that the parity check
matrix of $G'$ is similar to the parity check matrix of $G$, but each element of $H$, is substituted by $h_{i,j} P_{ij}^J$, where $P_{ij}^J$ is a $J$ by $J$ permutation matrix.

$$H' = \begin{bmatrix}
P_{11} & 0 & 0 & P_{14} & 0 & 0 & P_{17} \\
0 & P_{22} & 0 & 0 & P_{25} & 0 & P_{27} \\
P_{31} & 0 & P_{33} & 0 & 0 & P_{36} & 0 \\
0 & P_{42} & 0 & 0 & P_{45} & 0 & 0
\end{bmatrix}$$  \quad (4.2)
Chapter 5

Construction Algorithm

Since there is a relation between the girth of a Tanner graph and the minimum weight of the code $w_{min} > g$, and because of independency assumptions in the BP algorithm, the girth is one of the important parameters in a code. Hence, we are looking for a Tanner graph with a given degree distribution which has the biggest possible girth.

The main problem in the protograph construction is, how we chose the permutation matrices for all nonzero elements of the mother graph such that the girth of the final graph becomes greater than the mother graph.

In the following, the approach is briefly presented.

1. Find an underlying tree of the mother graph. Copy this tree $J$ times by means of the identity matrix, go to Step 2.

2. If the constructed graph has the given degree distribution, go to Step 4, if not, find the farthest possible check node and variable node (i.e. those variable and check nodes, that we can increase their degrees with respect to the given degree distribution.), go to Step 3.

3. Find a proper permutation matrix for connecting these variable and check node which were found in Step 2. Connect these two nodes and go to Step 2.

4. End.
Figure 5.1: Copying a Tree by identity matrix

1: Since in a tree there is no cycle, permutations of edges between the nodes are not so important. In each permutation, just by changing the labels of nodes or equivalently by moving the columns or rows, we can obtain another permutation. Therefore, we chose the identity permutation for each edge of the tree.

\[
H = \begin{bmatrix}
0 & 0 & 0 & I \\
0 & 0 & I & I \\
0 & 0 & I & 0 \\
I & I & I & 0 \\
\end{bmatrix}.
\] (5.1)

One can find an underlying tree just by transforming the given degree distribution to another degree distribution which satisfies Lemma 3.2 constraints.

**Note.** One way is decreasing one of the maximum degrees of check and variable nodes by one and check the Lemma 3.2 constraints; if it is satisfied, the proper underlying tree is found, if not, apply this procedure again.

2: We define the proper set of variable and check nodes at round \( t + 1 \) by the following

\[
A_{v}^{t+1} = \{ v_j \in V^t | dv_j + 1 \in D_v \} \quad A_{v}^t \subset V,
\] (5.2)

\[
A_{c}^{t+1} = \{ c_j \in C^t | dc_j + 1 \in D_c \} \quad A_{c}^t \subset C,
\] (5.3)

where \( V^t \) and \( C^t \) are the degree distributions of the constructed graph until round \( t \). Consequently, the construction algorithm will be finished in round \( l \) if \( A_{v}^l = A_{c}^l = \emptyset \)

We define the *length set* as follows:

\[
L_{v_j}^t = \max_{c \in A_{c}^t} \{ l(c, v_j) \} , \quad \forall v_j \in A_{v}^t,
\] (5.4)
where \( l(c, v_j) \) is the minimum length of paths between \( c \) and \( v_j \). Then, we obtain the subset

\[
\tilde{V} = \arg\max_{v_j} \{L^t_{v_j}\}.
\]

(5.5)

We randomly choose one element form \( \tilde{V} \), say, \( \tilde{v}_k \) and connect it randomly to one element of

\[
\tilde{c}_k = \arg L^t_{v_k},
\]

(5.6)

**Example 5.1** Suppose the given degree distributions are \( D_c = [3, 4, 4] \) and \( D_v = [2, 2, 2, 3, 2] \) and after some steps, we have the graph in Fig. 5.2.

\[
V^i = \{2, 2, 1, 1, 1\} \Rightarrow A^{i+1}_v = \{v_1, v_2, v_3, v_4, v_5\}
\]

(5.7)

\[
C^i = \{2, 2, 3\} \Rightarrow A^{i+1}_c = \{c_1, c_2, c_3\}
\]

(5.8)

*The dashed line shows which check and variable node is chosen for round \( i + 1 \).*

Similarly, we can explain this by expanding a tree from each variable node as shown on the right side of Fig. 5.2.

3: The next step is finding the permutation matrix for the edge which we added to the mother graph in the previous step. We know that the permutation matrix \( P_{J \times J} \) with its powers, \( \{P, P^2, ..., P^r = I\} \) of order \( r \) are a cyclic group, and \( r \leq J \). So we choose a permutation matrix with biggest possible order, i.e., \( r = J \), like the cyclic permutation

\[
a_{J \times J} = \begin{bmatrix}
0 \\
: \\
I_{(J-1) \times (J-1)} \\
0 \\
1 & 0 & \ldots & 0
\end{bmatrix},
\]

(5.9)
Figure 5.3: Paths between $c_b$ and $v_a$ with length $l(c_b, v_a) = d$ before connecting to each other.

$\{q, q^2, \ldots, q^J = I_{J \times J}\}, \quad o(q^k) = k$

Let $l(c_b, v_a)$ be the minimum length of paths between $c_b$ and $v_a$. Suppose in the previous step, $v_a$ and $c_b$ were chosen to be connected to each other, and before connecting, $l(c_b, v_a)$ was equal to $d$. We define $p_{v_a,c_b}^t$, denoting all paths between $v_a$ and $c_b$ with length $d$ in the mother graph at round $t$ (see Fig. 5.3).

\[
\text{length}(i) = d, \quad \forall i \in p_{v_a,c_b}^t,
\]

\[
O^t_{v_a,c_b} = \{o(Q_i) \mid i \in p_{v_a,c_b}^t\}, \quad Q_i \propto i^{th}\text{path},
\]

where $Q_i$ is a permutation matrix which describes how $v_a$ and $c_b$ are connected to each other by the $i^{th}$ path.

Example 5.2 If in the previous example, we supposed that $J = 3$ and the permutation matrices which are used for each edges be $I, q, I, q^2, q$. The equivalent matrix would be

Figure 5.4: One of paths between $v_a$ and $c_b$ with its permutation matrices
CONSTRUCTION ALGORITHM

$\mathbf{Q} = \mathbf{I} \cdot \mathbf{q}^T \cdot \mathbf{I} \cdot (\mathbf{q}^2)^T \cdot \mathbf{q} = \mathbf{q}.$  

(5.13)

The path which connected variable and check node in the crossed square has an equivalent permutation matrix $\mathbf{Q} = \mathbf{q}$.

Now, let us consider $O_{v_a,c_b}^t$, which represents all equivalent matrices between $v_a$ and $c_b$. The best permutation matrix for connecting these two nodes would be

$q^k, \ k \in \mathbb{Z}_J/O_{v_a,c_b}^t,$  

(5.14)

where $\mathbb{Z}_J = \{1, 2, \ldots, J = 0\}$ modulo $J$. For the previous example, $q^0 = \mathbf{I}$ would be one of the possible answers.

Now, there are $J$ variable nodes $v_a$ and $J$ check nodes $c_b$, which are connected to each other directly by matrix $q^k$ and also by other paths such as $p_{v_a,c_b}^t$ with length $d$. Hence, one can define a new $J \times J$ matrix named $U_{v_a,c_b}^t$ of a Tanner graph $\tilde{U}$ (Fig. 5.5), where $u_{ji}$ is one if there is a path with length $d$ or a single edge between an $i$th node among $v_a$ and a $j$th node among $c_b$.

**Lemma 5.1** We define a matrix $U_{v_a,c_b}^t$ at round $t$ by

$U_{v_a,c_b}^t = \bigvee_{j=1}^{\omega} \{q^{ij}\} \lor q^k, \ i_j \in O_{v_a,c_b}^t, \ k \in \mathbb{Z}_J/O_{v_a,c_b}^t,$  

(5.15)

where $\lor$ is the or operator, as a matrix of a Tanner graph, named $\tilde{U}$ with $J$ variable nodes and $J$ check nodes with girth $\tilde{g}$, then the length of the smallest cycle which contains $v_a$ and $c_b$ is at least $\frac{\tilde{g}}{2}(d + 1)$.  

**Figure 5.5:** Part of $\mathbf{H}$ matrix related to paths between $v_a$ and $c_b$ in Example 5.2
Proof: For each one in the matrix $U_{v_a,c_b}$ resulting from $\bigvee_{j=1}^{w}q^{j\nu}$, there is a path with length $d$ and for each one resulting from $q^k$ there is an edge in the Tanner graph, but at any cycle with length $l$ in $\tilde{U}$, there exist at most $l/2$ edges resulting from $q^k$ and the remaining ones contribute by the paths summarized in $\bigvee_{j=1}^{w}q^{j\nu}$, because if there were $l/2 + 1$ or more edges in a cycle of length $l$, then there would at least be one node in $\tilde{U}$ with two edges in this cycle and it is impossible, because $q^k$ has only one nonzero element in each row and column. Hence, if the girth of $\tilde{U}$ be $\tilde{g}$ then the smallest cycle would be $\tilde{g}/2$ edges and $\tilde{g}/2$ paths and the length of this cycle is $\frac{\tilde{g}}{2} + \frac{\tilde{g}}{2}d$.

\begin{figure}[h]
\centering
\includegraphics[width=0.3\textwidth]{tanner_graph}
\caption{Tanner graph $\tilde{U}$: dashed lines are for paths with length $d$ (In this figure we assume there is just one path with length $d$, $w = 1$), and the other lines are for $q^k$}
\end{figure}

**Theorem 5.1** The girth of final protograph is at least $2g_m$, where $g_m$ is the girth of the mother graph.

Proof: We know that the girth of a Tanner graph is at least 4. Hence, we just substitute $\tilde{g}$ in the previous Lemma by 4 and $d + 1$ by the girth $g_m$ of the mother graph.
Chapter 6

Simulation Results

We consider the UEP-LDPC code with length $N = 4096$ ensemble design proposed in [3]. The bits of the codeword are divided into 3 protection classes with rate $1/2$, $dv_{\text{max}} = 30$, and $\rho(x) = 0.00749x^7 + 0.99101x^8 + 0.00150x^9$, and $\lambda(x)$ is chosen according to Table 6.1.

**Note:** For constructing the $H$ matrix based on a protograph, we are not so flexible in choosing any arbitrary degree distribution because in the protograph construction algorithm, the final Tanner graph is obtained by copying the mother graph $J$ (here $J = 2^5$) times. The greatest common divisor (GCD) of the number of different degrees must be divisible by $J$.

The degree distribution which is used for constructing the $H$ matrix by the protograph algorithm is given in Table 6.2 and $\rho(x)$ is chosen as

<table>
<thead>
<tr>
<th></th>
<th>$P^1$</th>
<th>$P^2$</th>
<th>$P^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_{18}^{P^1}$</td>
<td>0.2521</td>
<td>$\lambda_3^{P^2}$</td>
<td>0.0786</td>
</tr>
<tr>
<td>$\lambda_{19}^{P^1}$</td>
<td>0.0965</td>
<td>$\lambda_4^{P^2}$</td>
<td>0.2511</td>
</tr>
<tr>
<td>$\lambda_{30}^{P^1}$</td>
<td>0.0946</td>
<td>$\lambda_5^{P^2}$</td>
<td>0.0141</td>
</tr>
</tbody>
</table>

$\lambda_{18}^{P^1} = 0.2521$, $\lambda_3^{P^2} = 0.0786$, $\lambda_4^{P^2} = 0.2511$, $\lambda_5^{P^2} = 0.0141$.

**Table 6.1:** Degree distribution generated by UEP for $\rho(x) = 0.00749x^7 + 0.99101x^8 + 0.00150x^9$. 
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
$P^1$ & $P^2$ & $P^3$ \\
\hline
$\lambda_{18}^{P^1} = 0.2509$ & $\lambda_3^{P^2} = 0.0886$ & $\lambda_2^{P^3} = 0.1951$ \\
$\lambda_{19}^{P^1} = 0.0662$ & $\lambda_4^{P^2} = 0.2787$ & $\lambda_3^{P^3} = 0.0159$ \\
$\lambda_{30}^{P^1} = 0.1045$ & & \\
\hline
\end{tabular}
\caption{Degree distribution near to the UEP for construction with protograph}
\end{table}

\[ \rho(x) = x^8. \]  
\hfill (6.1)

**Note:** By using Lemma 3.1, one can obtain an upper bound for the girth with a given degree distribution as

\[ g \leq \frac{2 \log\left(\frac{(n + m)(\bar{d} - 2)}{2} + 1\right)}{\log(\bar{d} - 1)}. \]  
\hfill (6.2)

For the above degree distribution, we have $n + m = 6144, \bar{d} \approx 5.99$, hence $g \leq 11.71$, which means $g \in \{4, 6, 8, 10\}$.

We compare the bit error probabilities of the code constructed by the PEG algorithm and the Protograph algorithm.
As mentioned above, the degree distribution for the protograph is just near to the one proposed in [3]. Figure 6.1 shows better error probabilities for protection classes 1 and 2. Since the degrees are very close to [3], we obtained even better error probabilities, but for protection class 3, the variable-node degrees are smaller than in [3]. Hence, the number of connections to variable nodes are reduced, hence we have a weaker protection, there.
Chapter 7

Conclusions

In this work, we introduced a new protograph construction algorithm. This algorithm can be used for expanding any mother graph. We are able to construct the mother graph for a given degree distribution by a progressive edge growth method at each step of the algorithm and simultaneously find a proper permutation matrix for permuting edges in its protograph.

If a specific mother graph is not given and a Tanner graph is to be constructed solely based on a given degree distribution, degree restrictions have to be obeyed, which result from the copying principle of the Protograph construction. Hence, the next problem at hand will be to determine a proper degree distribution suited for constructing a protograph for unequal error protection.
Bibliography


