EE229B PROJECT REPORT

STUDY OF PERMUTATION MATRICES BASED LDPC CODE CONSTRUCTION

Zhengya Zhang
SID: 16827455
zyzhang@eecs.berkeley.edu

1 MOTIVATION

Permutation matrices refer to the square matrices with one 1 in each row and one 1 in each column. Permutation matrices based LDPC parity check matrix has a regularity that facilitates hardware implementation of the decoder. Each permutation matrix can be realized in a super block that consolidates the interconnects for message passing from/to the bit nodes within the super block. In this way, the number of long interconnects can be greatly reduced and serialization of message processing within the super block allows the efficient reuse of hardware components. These result in a globally parallel, locally serial hardware architecture with a low interconnection overhead.

Some existing LDPC constructions produce patterns in the H matrices that can be represented using permutation matrices. Examples are Ramanujan graph based and Reed-Solomon codes based constructions. In these cases, large H matrices can be partitioned into smaller “patches” of permutation matrices and zero matrices, so the decoder for such constructions can take advantage of this pattern and be realized in a partially parallel architecture described above.

2 PROJECT SCOPE

The project used a small 6×9 H matrix constructed using 3×3 permutation matrices as an example, and focused the analysis on minimum distance, code rate, and pseudocodewords. Part of the project was devoted to the study of pseudocodewords using graph covers and the new approach using LP relaxation. 6×9 H matrices were studied and verified by simulation. Some results could be easily extended to larger H matrices. However, pseudocodeword analysis becomes exponentially more complex as the size of the H matrix grows. Even the LP codeword polytope solver becomes infeasible when block length exceeds 16. So the pseudocodeword findings were limited to 6×9 H matrices. In the end, a few systematic approaches were surveyed as ways to construct larger H matrices with such regular structures.

3 INTERPRETATION OF PSEUDOCODEWORDS USING GRAPH COVERS AND CODEWORD POLYTOPES USING LP RELAXATION

One potential setback of some LDPC codes is the existence of error floors [1]. Error floors cause concerns in low BER applications. In message-passing decoding, the error floor is manifested as a failure to converge to the correct codeword. It is believed that the error floors are caused by low-weight pseudocodewords. The existence of pseudocodewords is due to the cycles in the graph producing covers, and iterative, locally operating decoding algorithms that cannot distinguish the underlying graph from any covering graph [2], which sometimes results in inconsistencies among lifted copies of the same bit in the covering graph.

A more amenable way to analyze the codeword space is to use LP over a relaxed codeword polytope. Each parity check defines a local codeword polytope and the entire codeword polytope is defined by the intersection of all the local polytopes [3] [4]. There is a relationship between the combined codeword polytope and graph covers. The local codeword polytope corresponds to a lifted local parity check, with vertices of polytope as the codewords that satisfy the lifted local parity check. When the local polytopes are combined, fractional codewords may appear at the vertices of the global polytope, in so called relaxed codeword polytope. In iterative decoding, fractional codeword bits can be interpreted as indecisive bits that wonder around locally satisfied integral values, but these integral bit values always cause inconsistencies in some other checks in the original graph and thus fail to converge to definite values.

If a graph does not contain a cycle, it can be unwrapped in a tree. The graph would have no covers and the combined codeword polytope contains only integral codewords. Then there is a guarantee of no pseudocodewords. If the graph contains cycles, the graph will be unwrapped into covers and the combined local codeword polytope may contain fractional vertices. It should be noted that a vertex cannot be expressed as the convex combination of other points in the polytope. Some fractional codewords can be expressed as convex combination of valid integral codewords, but they are not considered pseudocodewords since they are not the vertices.

In the covers of graphs with cycles, fractional codewords can be formed. If the fractional codeword is a convex combination of integral codewords that satisfy all other checks in the original graph, this fractional codeword reside inside the codeword polytope. However if the fractional codeword is not a convex combination of valid integral codewords, it translates to a fractional vertex in the relaxed codeword polytope.

To sum up the above arguments, iterative, locally operated decoding algorithms operate on graph covers. It would fail to converge if it oscillates between locally satisfied set of values without satisfying all other checks in the original graph. This can be intuitively interpreted as fractional vertices in the relaxed codeword polytope in LP decoding. Thus we can use LP to study the pseudocodeword performances.

LP decoding always converge to a vertex of the codeword polytope. Using LP, we can formulate the relaxed codeword polytope and identify all the pseudocodewords. In this project, the PORTA software [5] was used to solve the codeword polytope. Each parity

check in the H matrix is translated to a set of inequalities. For each check node j, let $N(j)$ be the set of nodes such that check node j is connected to. For all $S \subseteq N(j)$ with $|S|$ odd. We can write a set of constraints [4] as $\sum_{i\in(N(j)\setminus S)} f_i + \sum_{i\in S}(1-f_i) \geq 1$.
For larger dimensions, we can constrain the solver to only look for pseudocodewords around the all-0 point that correspond to the most likely errors that would happen when the all-0 codeword is transmitted. This shortcut reduces the complexity of the solver when the dimension of the polytope grows.

After pseudocodewords are identified, we can characterize the minimum fractional distance and the number of pseudocodewords. In high SNR conditions, minimum fractional distance is of relevance. Intuitively, low-weight pseudocodewords cause the decoder to get trapped. In low SNR conditions, the target function, which is the log likelihood ratio of the received bits, fluctuates widely and the number of pseudocodewords determines how easily the decoder gets trapped.

4 ANALYSIS OF 6×9 H MATRICES

As shown below, 6×9 H matrices are constructed using 3×3 submatrices M, which are either permutation matrices or zero matrices. Enumerating all the possibilities, there are a total of 7^6 = 117,649 such constructions. However, only a small fraction of these possible constructions would yield "good" LDPC code. Several cases are considered in the following sections.

M	M	M
M	M	M

4.1 ALL PERMUTATION MATRICES BASED CONSTRUCTIONS

P	P	P
P	P	P

There are six possible 3×3 submatrices, which results in a total of 6^6 = 46,656 choices. To filter through these possibilities for good LDPC code, the first criterion is to eliminate short cycles, meaning that there should be no four 1's at the corners of a rectangle. A cycle of length 4 is illustrated as follows.

1	0	0	1	0	0	1	0	0
0	**1**	0	0	**1**	0	0	1	0
0	0	1	0	0	1	0	0	1
1	0	0	0	0	1	0	0	1
0	**1**	0	0	**1**	0	1	0	0
0	0	1	1	0	0	0	1	0

4.1.1 MINIMUM DISTANCE AND CODE RATE

Cycles of length 4 give $d_{min} = 2$, because the two columns involved in the cycles add up to zero. With 4-cycles eliminated, only 2,592 out of the 46,656 choices are left. An example of the code construction with 4-cycles eliminated is shown below.

1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1
0	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	0

This is a (2, 3)-regular code. It has the property that no two rows or columns have more than one 1 in common positions. Since each bit node has a degree of 2, it connects to 2 check nodes. If there is one 1 in the codeword, this bit node would require two more 1's to satisfy both checks. So there should be a total of at least 3 1's, i.e., $d_{min} \geq 3$. Furthermore, no odd number of columns can add up to zero. Therefore, $d_{min} = 4$. It can also be observed that if four columns add up to zero, the bits and checks involved in the four columns form a cycle of length 8. This is the minimum cycle length, i.e., girth = 8. So the girth is directly related to d_{min} in this case.

In this construction, 6 rows always add to zero. So the rank of this matrix is 5 and the code rate is (9-5)/9 = 4/9.

4.1.2 PSEUDOCODEWORD ANALYSIS

The PORTA polytope solver returns no pseudocodewords for all permutation matrices based 6×9 constructions void of 4-cycles. Therefore the relaxed codeword polytope contains only integral vertices. This is an interesting observation because the 6×9 matrix contains cycles of length 8, but it still did not produce any pseudocodewords. As an example, the polytope vertices for the following H matrix are listed below.

v0	v1	v2	v3	v4	v5	v6	v7	v8	
1	0	0	1	0	0	1	0	0	c0
0	1	0	0	1	0	0	1	0	c1
0	0	1	0	0	1	0	0	1	c2
1	0	0	0	1	0	0	0	1	c3
0	1	0	0	0	1	1	0	0	c4
0	0	1	1	0	0	0	1	0	c5

```
(  1) 0 0 0 0 0 0 0 0 0
(  2) 0 0 1 0 1 0 0 1 1
(  3) 0 0 1 1 0 1 1 0 0
(  4) 0 1 0 0 1 1 0 0 1
(  5) 0 1 0 1 0 0 1 1 0
(  6) 0 1 1 0 0 1 0 1 0
(  7) 1 0 0 0 0 1 1 0 1
(  8) 1 0 0 1 1 0 0 1 0
(  9) 1 0 1 1 0 0 0 0 1
( 10) 1 1 0 0 1 0 1 0 0
( 11) 0 0 0 1 1 1 1 1 1
( 12) 0 1 1 1 1 0 1 0 1
( 13) 1 0 1 0 1 1 1 1 0
( 14) 1 1 0 1 0 1 0 1 1
( 15) 1 1 1 0 0 0 1 1 1
( 16) 1 1 1 1 1 1 0 0 0
```

One way to look for pseudocodewords by hand is to initialize a few bit nodes in different checks, unwrap the Tanner graph, and check whether it would result in inconsistencies when the local checks are being satisfied. For example, pick $v_0 = 0$, $v_1 = 0$, and $v_2 = 1$, then try to satisfy c_0, c_1, and c_2. We get the following table of bit node values. Note that a, b, c are either 0 or 1. If a is 0, a' is 1, vice versa.

v0	v1	v2	v3	v4	v5	v6	v7	v8
0	0	1	a	b	c	a	b	c'

Substitute the above bit values to checks c_3 to c_5 and get,

$$c3 : b + c' = 0$$
$$c4 : c + a = 0$$
$$c5 : a + b = 1$$

From c_4 and c_5, we get b + c = 1, which is consistent with c_3. All permutation matrices based 6×9 H matrices have a regular structure, so you can start with any parity check with assigned initial values, and traverse through all other checks and end up with consistencies among all other checks. This means, despite the fact that the graph has covers, it does not result in pseudocodewords. This is confirmed by the disappearance of fractional vertices in the relaxed codeword polytope.

Another way to explain the disappearance of pseudocodewords is that they have been pruned by a redundant check. In this H matrix, only 5 rows are independent, the 6th row is redundant. However, the 6th row tightens the first-order relaxation. To see this, we can remove the 6th row, which can be any of the 6 rows because it can be expressed as a linear combination of all the other rows. So the first-order relaxation looks like follows.

1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1
0	1	0	0	0	1	1	0	0

Solve the codeword polytope, we get the following list of vertices.

```
(  1)   0   0 0 0   0   0   0 0   0
(  2)   0 1/2 0 0 1/2 1/2   0 1 1/2
(  3)   0 1/2 1 0 1/2 1/2   0 0 1/2
(  4) 1/2   0 0 1   0 1/2 1/2 0 1/2
(  5) 1/2   0 1 0   0 1/2 1/2 0 1/2
(  6) 1/2 1/2 0 0 1/2   0 1/2 1   0
(  7) 1/2 1/2 0 1 1/2   0 1/2 0   0
(  8)   0 1/2 0 1 1/2 1/2   1 0 1/2
(  9) 1/2   0 0 0   1 1/2 1/2 1 1/2
( 10) 1/2 1/2 1 0 1/2   0 1/2 0   1
( 11) 1/2 1/2 1 0 1/2   1 1/2 0   0
( 12) 1/2   1 0 0   0 1/2 1/2 1 1/2
( 13)   1 1/2 0 1 1/2 1/2   0 0 1/2
( 14) 1/2 1/2 0 0 1/2   1 1/2 1   1
( 15) 1/2 1/2 0 1 1/2   1 1/2 0   1
( 16) 1/2   1 0 1   1 1/2 1/2 0 1/2
( 17) 1/2   1 1 0   1 1/2 1/2 0 1/2
( 18)   1 1/2 0 0 1/2 1/2   1 1 1/2
( 19)   1 1/2 1 0 1/2 1/2   1 0 1/2
( 20)   0   0 1 0   1   0   0 1   1
( 21)   0   0 1 1   0   1   1 0   0
( 22)   0 1/2 1 1 1/2 1/2   1 1 1/2
( 23)   0   1 0 0   1   1   0 0   1
( 24)   0   1 0 1   0   0   1 1   0
( 25)   0   1 1 0   0   1   0 1   0
( 26) 1/2   0 1 1   1 1/2 1/2 1 1/2
( 27) 1/2 1/2 1 1 1/2   0 1/2 1   1
( 28) 1/2 1/2 1 1 1/2   1 1/2 1   0
( 29) 1/2   1 1 1   0 1/2 1/2 1 1/2
( 30)   1   0 0 0   0   1   1 0   1
( 31)   1   0 0 1   1   0   0 1   0
( 32)   1   0 1 1   0   0   0 0   1
( 33)   1 1/2 1 1 1/2 1/2   0 1 1/2
( 34)   1   1 0 0   1   0   1 0   0
( 35)   0   0 0 1   1   1   1 1   1
( 36)   0   1 1 1   1   0   1 0   1
( 37)   1   0 1 0   1   1   1 1   0
( 38)   1   1 0 1   0   1   0 1   1
( 39)   1   1 1 0   0   0   1 1   1
( 40)   1   1 1 1   1   1   0 0   0
```

It produced 40 vertices, which includes 16 codewords and 24 pseudocodewords with minimum fractional distance $d_{\text{frac}} = 3$. Adding one more check ($c0 \oplus c1 \oplus c2 \oplus c3 \oplus c4$) and all the pseudocodewords are removed. Let S be the set of rows in the first-order relaxation of the H matrix. Let b be the linear combinations of rows in S. Adding constraints induced by b can tighten the relaxation only if the factor graph induced by S contains at least one cycle [6]. We already know that this construction contains cycles of length 8, and cycles introduce ambiguities, thus adding additional constraints can always do better. In the extreme, we can add a set of constraints for all possible dual codewords (linear combinations of rows in S), thus forming a metric polytope. This is also called the canonical full relaxation. The metric polytope is equal to the exact codeword polytope if and only if the metric polytope does not contain the three forbidden code minors (for details, see [6]), which is true in this case. To achieve canonical full relaxation, we need $\binom{r}{2} + \binom{r}{3} + \ldots + \binom{r}{r} = 2^r - 1 - r$ (where $r = rank(H)$) additional checks.

We can also view nth-order relaxation as the checksum for the bits involved in the n checks. If the Tanner graph contains a cycle, ambiguity maybe introduced. A checksum can be added to enclose the entire cycle, so the ambiguity might be reduced. If the girth of the Tanner graph is g, the shortest cycle contains g/2 check nodes. So a minimum of (g/2)th-order relaxation can produce any effect in reducing pseudocodewords. The interesting problem would be to find the minimum number of higher-order relaxations that eliminate pseudocodewords, or reduce the low-weight pseudocodewords to increase d_{frac} of the code. It should be noted that redundant constraints may degrade the performance of Belief Propagation decoding due to the creation of small cycles, while adding new constraints can only improve LP performance [4].

4.2 PERMUTATION AND ZERO MATRICES BASED CONSTRUCTIONS

Some sparse low density H matrices contain patches of zeros. These can be represented as zero matrices. We expect a large number of the regular patterns be represented as a combination of permutation matrices and zero matrices. The following experiments study the effect of zero matrices on the 6×9 H matrices.

4.2.1 SINGLE ZERO MATRIX

P	P	Z
P	P	P

An example is shown above. In this case, there is only one zero matrix. It is no longer a regular code. First eliminate the 4-cycles in the H matrix. 7,776 out of the possible 46,656 choices are left.

Minimum distance of such a construction remains 4. To see this, you can always pick two degree-2 columns that have their upper 1's in the same row. Then you can pick two degree-1 columns that have 1's at the same positions as the lower 1's of the degree-2 columns. These four columns add up to zero.

The 6 rows of such a construction are independent, so the rank of such an H matrix is 6. Code rate is reduced to (9 – 6)/9 = 1/3.

Cycles can only appear among the 6 degree-2 bit nodes. These nodes can be visualized as positioned in a 6×6 submatrix as highlighted below. The shortest cycle in this 6×6 submatrix is of length 12. So the girth improves to 12 for such a construction.

1	0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	0	0
0	0	1	0	0	1	0	0	0
1	0	0	0	1	0	0	0	1
0	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	0

To see the pseudocodeword performance, formulate the constraints based on the parity check matrix and solve the codeword polytope for all the possible vertices using LP relaxation. A listing of the results is shown below. There are 8 codewords and 4 pseudocodewords with a minimum fractional distance $d_{frac} = 4$.

```
(  1)   0   0   0   0   0   0 0 0 0
(  2) 1/2 1/2 1/2 1/2 1/2 1/2 0 0 1
(  3) 1/2 1/2 1/2 1/2 1/2 1/2 0 1 0
(  4) 1/2 1/2 1/2 1/2 1/2 1/2 1 0 0
(  5) 1/2 1/2 1/2 1/2 1/2 1/2 1 1 1
(  6)   0   0   1   0   0   1 1 1 0
(  7)   0   1   0   0   1   0 1 0 1
(  8)   1   0   0   1   0   0 0 1 1
(  9)   0   1   1   0   1   1 0 1 1
( 10)   1   0   1   1   0   1 1 0 1
( 11)   1   1   0   1   1   0 1 1 0
( 12)   1   1   1   1   1   1 0 0 0
```

Similar to the approach in the previous section, the pseudocodewords can be discovered using some simple hand calculations. Initialize three bits that belong to three different checks, i.e., $(v_6, v_7, v_8) = (0, 0, 1)$. Follow the constraints in c_3, c_4, and c_5, we find the relationships between v_0 and v_4, v_2 and v_5, as well as v_2 and v_3.

v0	v1	v2	v3	v4	v5	v6	v7	v8
a	b	c	c	a’	b	0	0	1

Substitute the node values to checks c_0, c_1, and c_2 give the following set of constraints.

$$c0 : a + c = 0$$
$$c1 : b + a' = 0$$
$$c2 : c + b = 0$$

From c_1 and c_2, we get a’+c = 0, which directly conflicts with c_0: a+c = 0. For any set of bit values that satisfies a+c = 0, we get a’+c = 1. So it appears that values of a, b, and c oscillate between sets of incorrect values that fail to satisfy all the constraints. These show up as 1/2 fractional values.

In summary, introducing one zero matrix in the 6×9 construction results in a code with the same d_{min}, lower code rate, but higher girth compared to the all permutation matrices based construction. It also produces a set of locally satisfied but globally unsatisfied pseudocodewords.

4.2.2 TWO ZERO MATRICES

P1	P2	Z
P3	Z	P4

An example is shown above. Two zero matrices can be allowed in different rows and different columns. There is a total of $6\times6^4 = 7{,}776$ such constructions. This is not a regular code. 3 bit nodes are of degree-2 and 6 bit nodes are of degree 1. The degree-1 nodes break up all the possible cycles, so this construction does not contain any cycle. The Tanner graph of this construction can be unwrapped into a tree. There are no graph covers, and no pseudocodewords are expected.

Formulate the constraints and perform LP relaxation to solve the relaxed codeword polytope. A list of vertices is shown below. There is no fractional vertex and it confirmed that there are no pseudocodewords in this type of construction.

```
(  1) 0 0 0 0 0 0 0 0 0
(  2) 0 0 1 0 0 1 0 1 0
(  3) 0 1 0 0 1 0 1 0 0
(  4) 1 0 0 1 0 0 0 0 1
(  5) 0 1 1 0 1 1 1 1 0
(  6) 1 0 1 1 0 1 0 1 1
(  7) 1 1 0 1 1 0 1 0 1
(  8) 1 1 1 1 1 1 1 1 1
```

Minimum distance of this construction is reduced to 3. It can be observed that you can always take 1 degree-2 column and 2 degree-1 columns, and they add up to zero. With this construction, the number of independent rows is 6, i.e., rank = 6 and code rate is reduced to (9-6)/9 = 1/3.

4.3 Summary of findings on 6×9 H matrices

All permutation matrices based H matrices void of 4-cycles yield good LDPC constructions. The properties are $d_{min} = 4$, rank =5, girth = 8, and no pseudocodewords is observed. The disappearance of pseudocodewords can be explained using the concept of metric polytope. The redundant 6th row in the H matrix tightens the first-order LP relaxation and reduces the ambiguity. The resulting codeword polytope is exact. Such a property may be leveraged to construct exact polytopes hierarchically for LP decoding.

When a single zero matrix is introduced to the H matrix, the properties of the H matrix are $d_{min} = 3$, rank = 6, girth = 12, and pseudocodewords were observed. This is due to the existence of a set of bit values that satisfy local checks without global consistency. In practice, such a construction is inferior to the all permutation matrices based construction since d_{min} is lower, code rate is lower, and the existence of pseudocodewords.

When two zero matrices are introduced to the H matrix, we require they are located in different rows and different columns to make a meaningful construction. The resulting H matrix has the following properties, $d_{min} = 3$, rank = 6, and there is no cycle in the Tanner graph. There is no pseudocodewords, which proved the fact that if there is no cycle in the Tanner graph, the first-order LP relaxation produces an exact codeword polytope. This serves as a good example though it does not have much practical value since large practical H matrices would always contain cycles. It is hopeless to use zero matrices to break up cycles without severely sacrificing the code rate.

5 SURVEY OF METHODS THAT SYSTEMATICALLY CONSTRUCT PERMUTATION MATRICES BASED LDPC CODES

In the above, we analyzed 6×9 H matrix composed of 3×3 permutation matrices void of 4-cycles. We can generalize the structure as H matrices composed of q×q permutation submatrices in a $\gamma \times \rho$ formation, such that no two rows or two columns have more than one 1 in common. Such an H matrix has a block length $n = q\rho$ and parity length $n\text{-}k < q\gamma$.

5.1 ARRAY CODES

The first example is the array code [7][8][9]. Pick a prime number q, the H matrix is constructed using cyclic shifts of the identity matrix. As illustrated below, σ corresponds to the permutation matrix that consists of a single cyclic shift of the q×q identity matrix to the right (or to the left). The H matrix can be constructed as below.

$$\boxed{\begin{array}{lllllll}
I & I & I & I & .. & .. & I \\
I & \sigma & \sigma^{2} & \sigma^{3} & .. & .. & \sigma^{q-1} \\
I & \sigma^{2} & \sigma^{4} & \sigma^{6} & .. & .. & \sigma^{2(q-1)} \\
I & \sigma^{3} & \sigma^{6} & \sigma^{9} & .. & .. & \sigma^{3(q-1)} \\
: & : & : & : & & & : \\
: & : & : & : & & & : \\
I & \sigma^{\gamma-1} & \sigma^{2(\gamma-1)} & \sigma^{3(\gamma-1)} & .. & .. & \sigma^{(\gamma-1)(q-1)}
\end{array}}$$

In this construction, $\rho = q$ and $\gamma \le q$. Let c_0 be the first q rows of the H matrix, c_1 be the second q rows of the H matrix, and so on. We observe that q rows in c_i always add up to [1 1 1 … 1]. To find the set of independent rows, we can pick q rows from any c_i, $0 \le i \le \gamma\text{-}1$, and any q-1 rows from each of c_j, $0 \le j \le \gamma\text{-}1$, $j \neq i$. The remaining row in each of c_j can be formed by the linear combination of rows in c_i and the q-1 rows in c_j. Therefore, there are a total number of $q + (q - 1)(\gamma - 1) = q\gamma - \gamma + 1$ independent rows. The code rate is $\frac{q^2 - (q\gamma - \gamma + 1)}{q^2} = \frac{q-\gamma}{q} + \frac{\gamma - 1}{q^2}$. In cases where $q >> \gamma$, the code rate is high. There are $q\gamma - (q\gamma - \gamma + 1) = \gamma - 1$ redundant rows.

Array code constructions do not contain any 4-cycles. To see this, a 4-cycle can only exist if you can find four cyclic shift submatrices at four corners of a rectangle, shown below, such that $(jy - iy) - (jx - ix)$ is a multiple of q. $(jy - iy) - (jx - ix) = (j - i)(y - x)$, where $0 \le x < y \le \gamma\text{-}1 < q$, and $0 \le i < j \le q\text{-}1$. Thus $0 < (j - i) < q$ and $0 < (y - x) < q$. Because q is a prime, $(j - i)(y - x)$ does divide q. Therefore, 4-cycles do not exist.

$$\boxed{\begin{matrix} \sigma^{ix} & .. & \sigma^{jx} \\ : & & : \\ \sigma^{iy} & .. & \sigma^{jy} \end{matrix}}$$

Minimum distance is usually lower bounded by $\gamma+1$ for such constructions because if one bit is 1, it is connected to γ checks and there have to be at least γ bits being 1 to satisfy all γ checks. If γ is even, $d_{min} = \gamma + 1$ is odd. But it is impossible to pick an odd number of columns that add up to zero. Therefore, when γ is even, $d_{min} \geq \gamma + 2$.

As we can see, lower γ gives higher data rate but lower bound on d_{min}. Higher γ gives lower code rate and possible higher d_{min}. In this construction, q and γ are the two free parameters that can be used to produce codes of desired properties. Usually lower γ is of practical interest [8].

5.2 REED-SOLOMON CODE BASED CONSTRUCTION

Permutation matrices based LDPC code can be constructed from shortened Reed-Solomon code [10]. It produces an H matrix that contains $\gamma\times\rho$ of $p^s\times p^s$ permutation submatrices, where p is a prime, s is a positive integer, $2 \leq \rho < p^s$, and $1 \leq \gamma \leq p^s$. This is a regular (γ, ρ) code. We can divide the H matrix into γ sections of p^s rows. Based on the structural property of the shortened Reed-Solomon code, there are no two rows or two columns have more than one 1 in common positions. So there are no four 1's at the corner of a rectangle and the Tanner graph has no cycles of length 4.

Let $q = p^s$. Following the argument in the previous section, there are $q + (q - 1)(\gamma - 1) = q\gamma - \gamma + 1$ independent rows. The code rate is $\frac{q\rho - (q\gamma - \gamma + 1)}{q\rho} = \frac{\rho - \gamma}{\rho} + \frac{\gamma - 1}{q\rho}$. There are $\gamma - 1$ redundant rows. Similar to array codes, d_{min} is lower bounded by $\gamma + 1$ when γ is odd, and $\gamma + 2$ when γ is even.

In RS based construction, we have an extra degree of freedom in choosing ρ. In array based construction, the check degree must be q.

5.3 FINITE GEOMETRY BASED CONSTRUCTION

Finite geometry is an approach that constructs LDPC codes based on the structure of finite and projective geometries over finite fields [11]. The properties of finite geometries guarantee a regular (γ, ρ) construction with two rows in the H matrix having at most one 1 in common. This construction does not necessarily have a permutation matrix structure. Similar to the previous two cases, $d_{min} \geq \gamma + 1$ and girth is at least 6.

In summary various ways exist in producing regular (γ, ρ) H matrices with the property that no two rows or two columns have more than one 1 in the common positions. This property guarantees a girth of 6, $d_{min} \geq \gamma + 1$, and code rate at least $\frac{\rho - \gamma}{\rho}$. This construction also has a good expansion property (meaning that you can start from one bit and get to all other bits in a short distance), which is good for iterative decoding. Some systematic ways, either structurally (such as array codes), or algebraically (such as shortened Reed-Solomon construction), produce H matrices that can be decomposed into permutation matrices. These constructions are especially amenable to efficient hardware implementation. Pseudocodeword performance is still unknown for larger code due to the limitation on the simulating larger codes. The insight gained in 6×9 matrices suggest that, by properly introducing some redundant checks in LP decoding, some ambiguities created by cycles can be overcome and the number pseudocodewords can be reduced or even eliminated.

Acknowledgements

I would like to thank Lara Dolecek for weekly discussions on pseudocodewords and regular code constructions. I would also like to thank Alex Dimakis and Professor Martin Wainwright for helpful discussions on LP relaxation and codeword polytopes.

REFERENCES

[1] Richardson, T., "Error floors of LDPC codes," in *Proceedings of Allerton Conference on Communication, Control, and Computing*, pp.1426-1435, October 2003.
[2] Koetter, R., and Vontobel, P., "Graph-covers and iterative decoding of finite-length codes," in *Proceedings of the 3rd International Symposium on Turbo Codes and Related Topics*, pp.75-82, September 2003.
[3] Feldman, J., Wainwright, M., and Karger, D., "LP decoding," in *Proceedings of Allerton Conference on Communication, Control, and Computing*, October 2003.
[4] Feldman, J., Wainwright, M., and Karger, D., "Using linear programming to decode binary linear codes," in *IEEE Transactions on Information Theory*, pp.954-972, March 2005.
[5] PORTA, http://www.iwr.uni-heidelberg.de/groups/comopt/software/PORTA/.
[6] Wainwright, M., "Notes on codeword polytopes and the sum-of-circuits property," February 2005.
[7] Fan, J., "Array codes as low-density parity-check codes," in *Proceedings of the 2nd International Symposium on Turbo Codes and related Topics*, pp.543-546, September 2000.
[8] Eleftheriou, E. and Olcer, S., "Low-density parity-check codes for digital subscriber lines," in *IEEE International Conference on Communiations*, pp.1752-1757, April 2002.

[9] Mitelholzer, T., “Efficient encoding and minimum distance bounds of Reed-Solomon-type array codes,” in *Proceedings of IEEE International Symposium on Information Theory*, pp.282, June 2002.
[10]Djurdjevic, I., Xu, J., Abdel-Ghaffar, K. Lin, S., “A class of low-density parity-check codes constructed based on Reed-Solomon codes with two information symbols,” in *IEEE Communications Letters*, pp.317-319, July 2003.
[11]Kou, Y., Shu, L., Fossorier, M., “Low-density parity-check codes based on finite geometries: a rediscovery and new results,” in IEEE Transactions on Information Theory, pp.2711-2736, November 2001.