AMERICAN UNIVERSITY OF BEIRUT

IMPACT OF REDUNDANT PARITY CHECKS ON LINEAR PROGRAMMING DECODING OF LDPC CODES

by

HANI THABET AUDAH

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
May 2014

AMERICAN UNIVERSITY OF BEIRUT

IMPACT OF REDUNDANT PARITY CHECKS ON LINEAR PROGRAMMING DECODING OF LDPC CODES

by

HANI THABET AUDAH

Approved by:

[illegible]

Dr. Louay Bazzi, Professor Electrical and Computer Engineering	Advisor

[illegible]

Dr. Fadi Zaraket, Professor Electrical and Computer Engineering	Committee Member

[illegible]

Dr. Ibrahim Abou Faycal, Professor Electrical and Computer Engineering	Committee Member

Date of thesis defense: May 9, 2014

AMERICAN UNIVERSITY OF BEIRUT

THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name:	Audah	Hani	Thabet
	Last	First	Middle

[✓ Master's Thesis] ○ Master's Project ○ Doctoral Dissertation

[✓] I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

☐ I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project,** to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

[illegible]	May 20, 2014
Signature	Date

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Professor Louay Bazzi, whose patience and guidance have been invaluable to me during the completion of this thesis. Working with him has certainly been a principal motivation for my graduate research, and I can say with confidence that I am a more proficient researcher now than I was when we began working together.

I would also like to thank my committee members, Professor Ibrahim Abou Faycal and Professor Fadi Zaraket, for their useful commentaries on the results that have been presented to them. They've helped me better appraise the value of my work. I have been fortunate enough to have a committee of accommodating and understanding faculty members.

Of course, special thanks are due to my family for their appreciation of the time and effort that has been spent working on this thesis. Surely it would not have been possible without your support. To my wonderful parents, Thabet and Mona, and my okay brothers, Nabeel and Mohammed, thank you for all the encouragement.

AN ABSTRACT OF THE THESIS OF

Hani Thabet Audah for Master of Engineering
Major: Electrical and Computer Engineering

Title:
Impact of Redundant Parity Checks on Linear Programming Decoding of LDPC Codes

The use of linear programming (LP) to decode low density parity check codes was first suggested by Feldman[1]. It is a relaxation of the maximum likelihood decoding problem into a linear optimization problem of polynomial complexity. The underlying polytope is obtained by adding all the local linear constraints associated with the individual check nodes in the Tanner graph of the code. This relaxation results in invalid codewords being included in the polytope of the LP solver. While it had been suggested by Feldman that the size of the polytope may be reduced through the addition of redundant parity checks – parity checks obtained through the combination of other parity checks – the issue was not further examined in that paper. The purpose of this thesis is to carry out such an analysis and determine what reduction in polytope size may be possible through the addition of these redundant checks.

It will be shown that in the case of good girth LDPC codes, and assuming reasonable conjectures on the fundamental polytope of the LP decoder to be true, the LDPC codes typically encountered in practice, such redundant checks do not improve the error-correction capability of these codes under LP decoding.

CONTENTS

TABLES

CHAPTER 1

INTRODUCTION

1.1. Background

Any transmission of data over non-ideal channels entails some uncertainty at the receiving end. Delay, distortion, and rearrangement are just three of the ways a channel may affect the transmitted data. Any of several models may be used to represent the effect of the channel as tractably as possible, taking into account the properties of the particular medium being used. Apart from some trivial cases, error-correction coding is needed if such communication is to achieve the theoretical bounds established by Shannon in his landmark 1948 paper. The premise behind error-correction is that the effect of the channel may be determined and reversed at the receiver if certain redundancy is added prior to transmission. The manner in which redundancy is to be added remains the principal focus of error-correction coding.

The k message bits to be transmitted are first mapped into a codeword X of n bits which is sent over the channel and received as another codeword Y (from a possibly different symbol alphabet). The receiver then must use knowledge of the channel model to recover the original transmitted codeword. The set of all codewords is called the code, referred to here as C. The decoder that returns a codeword X for any received Y with minimum probability of error is called the maximum-likelihood (ML) decoder. ML decoding is known to be NP-hard for general codes as well as the class of codes considered here, so a study of suboptimal decoding is warranted. Indeed, there has been a growing interest in suboptimal decoders and reducing the gap between the widely used iterative decoders and ML decoding, or at least a proper understanding of the theoretical limits of this gap.

The rate of a code C is defined as $\frac{1}{n}\log_{|F|}(M)$, M being the size $|C|$ and n the length of each codeword. Informally, the more redundancy is added to transmit a given length of message bits, the less the rate of the code. An information-theoretic bound introduced in Shannon's paper called channel capacity represents the maximum rate of communication possible given a probabilistic channel model. Until fairly recently, there had been no known good codes, that is, practical codes of rate close to capacity. The LDPC codes mentioned in the next section are a class of capacity-approaching codes that can be brought arbitrarily close to capacity (for symmetric memoryless channels) without requiring prohibitive complexity in the encoder/decoder. These codes were introduced by Gallager in his thesis dating back to 1960 [2] and did not receive much attention until researchers working on Turbo codes in the early 1990s revisited the area of iterative decoding (also discussed in the next section). Since that time, LDPC codes have been at the forefront of modern coding theory, with performance that is mostly unmatched by classical coding schemes.

Our interest is in the decoding of LDPC codes using linear programming (LP), an approach first proposed by Feldman[3]. Linear programming is a solution technique used to find the minimum of an objective linear function over a given domain under some specified linear constraints. We will see that ML decoding can be reformulated into a linear programming problem, so that a conventional LP algorithm effectively becomes our channel decoder. While this approach offers no computational advantage to the iterative algorithms that have been in use previously, they are more amenable to mathematical analysis. What Feldman actually proposes in his thesis is not exact ML decoding using LP but a relaxed form that avoids the computational complexity problems of ML decoding.

CHAPTER 2

LITERATURE REVIEW

2.1. Linear Codes

A code C of block length n is said to be linear over a field $\mathbb{F}$ if, when viewed as a set of n-tuples, it forms a subspace of the vector space $\mathbb{F}_n$. We will be concerned here exclusively with binary linear codes, where the field $\mathbb{F}$ is the binary field $\mathbb{F}_2$. For linear codes, it is easy to show that the minimum distance between any two codewords is the minimum of all non-zero codewords. Linear codes are commonly represented using a generator matrix G whose rows are the basis vectors of the subspace C. Each linear code has an associated dual code denoted $C^{\perp}$ and given by $C^{\perp} = \{y | \forall x \in C : \sum x_i y_i = 0\}$. The generator matrix H for this dual code is another description of the code C, and is referred to as the parity check matrix. Clearly, the matrix product HG results in a zero matrix, and the rows of H may be understood as a set of equations of the form $\{\sum_{i \in S} x_i = 0 \text{ in } \mathbb{F}_2\}$ where the set S is the set of indices of non-zero entries in a row of H. A code may thus alternatively be represented using a Tanner graph. In a Tanner graph, the codeword bits are represented as variable nodes, and each parity constraint is represented as a check node connected to the variable nodes to which the constraint applies.

Assume a codeword X is chosen from C with probability $p_x(X)$ and a symbol Y is received at the other end of the channel. If Y is decoded to $\hat{\mathbf{x}}(Y) \in C$, the probability of decoder error is $1 - p_{x|y}(X|Y)$. Maximum a-posteriori decoding minimizes this by taking $\hat{\mathbf{x}}(Y)$ to be $\arg\max_{x \in C} p_{x|y}(X|Y)$. For the case of uniform codeword probability $p_x(X)$, we end up with the maximum likelihood decoder:

$$\hat{\mathbf{x}}(Y) = \arg\max_{x \in C} p_{y|x}(Y|X) p_x(X) = \arg\max_{x \in C} p_{y|x}(Y|X)$$

For a memoryless channel, this expression may be reformulated as

$$\arg\max_{x\in C}\prod_{i=1}^{n} p_{y_i|x_i}(Y_i|X_i) = \arg\max_{x\in C}\frac{\prod_{i=1}^{n} p_{y_i|x_i}(Y_i|X_i)}{\prod_{i=1}^{n} p_{y_i|x_i}(Y_i|0)} = \arg\max_{x\in C} log(\prod_{i=1}^{n}\frac{p_{y_i|x_i}(Y_i|X_i)}{p_{y_i|x_i}(Y_i|0)})$$
$$= \arg\max_{x\in C}\sum_{i=1}^{n} log(\frac{p_{y_i|x_i}(Y_i|X_i)}{p_{y_i|x_i}(Y_i|0)}) = \arg\min_{x\in C}\sum_{i=1}^{n}\gamma_i x_i.$$

where the quantity $\gamma_i = \log\left(\frac{\Pr[Y_i|X_i=0]}{\Pr[Y_i|X_i=0]}\right)$ is known as the log-likelihood ratio (LLR). For the p-BSC this expression evaluates to $\log[(1-p)/p]$ when a 0 is received in error and $\log[p/(1-p)]$ when it is received correctly. We can rescale all γ_i values without affecting the ML decoder's decision, so we will take γ_i to be -1 for bits received in error and +1 for those received correctly.

Throughout this paper, the channel under consideration is the memoryless binary symmetric channel (BSC) where each bit is flipped with probability p and passes unaltered with probability 1-p, independently of all other bits.

2.2. Linear Programming

For the purpose of this thesis, only a brief mention of linear programming need be covered; we refer the reader to [4] for an introductory treatment to the subject. Linear programming searches for the minimum of some objective linear function f subject to some linear constraints on the independent variables. These constraints define a polytope containing some of the points in the domain of f. A linear program is generally expressed in **standard form**:

$$\text{Minimize } \sum_{j=1}^{n} c_j x_j \text{ subject to:} \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \text{ for i = 1, ... , m} \tag{2}$$

$$x_j \geq 0 \text{ for j = 1, ... , n} \tag{3}$$

It can be shown that for all feasible linear programs, the minimum must be at one or more of the vertices on the boundaries of this polytope. Linear programming algorithms are either vertex algorithms that can only iterate between vertices, or interior point algorithms that can visit interior points in the intermediate steps. There exist LP solvers that run in time polynomial in the number of constraints in the problem (e.g. ellipsoid algorithm).

An important concept in linear programming is the notion of duality. Given a linear program expressed as in (1), what is the best (highest) lower bound we can achieve on the value returned by the LP solver? This question can be rephrased as yet another linear program. Without undue detail, the dual is obtained by replacing the above LP with:

$$\text{Maximize } \sum_{i=1}^{m} b_i y_i \text{ subject to:} \tag{4}$$

$$\sum_{j=1}^{m} a_{ij} y_i \leq b_i \text{ for j = 1, ... , n} \tag{5}$$

$$y_i \geq 0 \text{ for i = 1, ... , m} \tag{6}$$

By definition, the solution to this dual LP must be less than the solution of the original LP, called the **primal** LP. It can be shown that the two optima are actually equal. In fact, this equality follows directly from Farkas' Lemma[4].

Duality plays a very important role in linear programming and is referenced in much of the literature on LP decoding, discussed next.

2.3. Linear Programming Decoding

Maximum-likelihood decoding is optimal in that it returns the codeword most likely to have caused the observed channel output. The ML-decoding problem, however, is known to be NP-hard. The message-passing algorithms presented earlier are a widely used alternative approximation to ML decoding. But though much work has gone into better understanding the performance of message-passing decoders, results are still lacking. The LP decoding algorithm to be discussed here is more analytically tractable. Additionally, there are some known relations between LP decoding and min-sum decoding[1] over certain limited channels and code graphs (e.g. codes without cycles), and other less powerful relations that apply to more general codes [5].

2.3.1. Feldman Approach

The use of linear programming to decode linear codes was first suggested by Feldman[1]. Whereas the purpose in maximum likelihood decoding is to determine the codeword that minimizes $\sum_{i=1}^{n} x_i \gamma_i$, linear programming decoding makes use of the fact that the minimum value of a linear function of x will be on one (or more) of the vertices of the polytope defined by the LP constraints. The first step in LP decoding is to rephrase the ML decoding problem as a linear optimization problem. The function to be minimized here is the log-likelihood ratio over the set of possible codewords. We can expand the domain of definition of this function to include the convex hull of the region enclosed by these codewords by taking f = $\sum_{i=1}^{n} \gamma_i f_i$ where f_i must be in the convex hull of the i^{th} coordinate of all codewords in C. Linear programming can therefore be used on this polytope to minimize the objective function, and, since LP always returns a vertex of the polytope, the result will always be a valid codeword. The polytope defined by these codeword vertices is thus given by:

[1]Min-sum decoding is a form of iterative decoding that, in some cases, has a performance close to the BP algorithm mentioned earlier

$$\wp = \{\sum_{y\in C} \lambda_y y : \lambda_y \geq 0, \sum_{y\in C} \lambda_y = 1\}$$

Although LP polynomial-time algorithms do exist, they are polynomial in the number of constraints describing the polytope. As the number of constraints in formula increases exponentially with the codeword length, we end up again with an exponential-time algorithm for ML decoding. To reduce this complexity, Feldman proposes a relaxation of the polytope whereby every check node defines its own set of allowed codewords (local code) and the intersection of the polytopes of all local codes is considered instead. With each check node j having neighborhood $N(j)$, the subsets $S \subset N(j)$ with an even number of nodes defines a set $E_j = \{S \subset N(j) : |S|\ is\ even\}$ which corresponds clearly to a ***local codeword***, a codeword whose variable nodes in $N(j)$ satisfy the single check node constraint imposed by j. Each codeword x must correspond, then, to a single $S \subset E_j$ such that $x_i = 1 \leftrightarrow i \in S$. Let $w_{j,S}$ denote the indicator variable that is 1 for that particular set S chosen, so that $\sum_{S\in E_j} w_{j,S} = 1$ (as there is only one). Furthermore, let f_i denote the indicator variable for the i^{th} codeword bit, where $i \in N(j)$. we know that if $f_i = 1$ there is exactly one set S in E_j that contains i such that $w_{j,S} = 1$, and if $f_i = 0$ then there are none. Thus, $f_i = \sum_{\substack{S\in E_j \\ i\in S}} w_{j,S}$. The set of points (integral or real) satisfying these constraints form the relaxed polytope for Feldman's LP decoder.

The new linear program to be solved is thus:

$$\text{minimize} \sum_{i=1}^{n} \gamma_i f_i \text{ subject to } (f,w) \in \wp \tag{7}$$

where $\wp = \cap_j \wp_j$ and $\wp_j$ is the individual jth check node polytope defined by:

$$\forall i, 0 \leq f_i \leq 1 \tag{8}$$

$$\forall S \in E_j, 0 \leq w_{j,S} \leq 1 \tag{9}$$

$$\forall i \in N(j), f_i = \sum_{S\in E_j, i\in S} w_{j,S} \tag{10}$$

$$\sum_{S \in E_j} w_{j,S} = 1 \tag{11}$$

For LDPC codes, the above polytope would require a linear number of constraints, and LP decoding would therefore be linear in codeword length. Clearly, however, there are codewords in the relaxed polytope that are not included in the original polytope and the LP decoder fails if it returns any of these fractional codewords as the minimal codeword (or if ML decoding also fails). Any integral point in the relaxed polytope must clearly have been in the original polytope. Hence, the LP decoder has the property that any valid codeword returned by it must be the maximum-likelihood codeword. Moreover, as a result of the symmetry of the underlying polytope, the zero-codeword assumption remains valid for LP decoding with this relaxation. That is, the error-probability does not depend on which particular codeword was transmitted, and the all-zero codeword is always assumed to be transmitted.

2.3.2. Koetter-Vontobel Approach

Koetter and Vontobel arrived at the same polytope relaxation by analyzing universal graph covers[6]. A graph m-cover G' (or m-lifting) of a graph G consists of m copies of G with the adjacency between the nodes changed such that if $x, y \in G$ are neighbors then there exist two copies $x', y' \in G'$ of x and y such that x' and y' are neighbors. Any message-passing algorithm decides on the values of the variable nodes based on the local neighborhood of that node, so to any variable node any finite graph cover code "appears" the same as the original graph code under message-passing decoding. But it is not difficult to see that the graph cover code includes more than simply the liftings of the codewords from the original graph , and the additional codewords are therefore the ones responsible for decoding errors. The authors then proceed to analyze the code defined by the union of all codes of finite covers of the original code graph. To this end, the authors define pseudo-codewords $\omega_i(\hat{\mathbf{c}})$ in the original graph by averaging the values in all copies of x in the cover, and assigning that (real) value to x. The polytope of all pseudo-codewords in G turns out (after some derivation) to be the intersection of the polytopes obtained by considering check nodes individually, that is, it turns out to be identical to the relaxed polytope of Feldman.

The conditions that lead to LP decoder success have been studied in [7], and it is shown there that an equivalent characterization of LP success may be stated as: given an error pattern and the corresponding values $\gamma(v_i)$ for the variable nodes, the LP decoder succeeds if we can find a function (called the dual witness) defined on the edges of the Tanner graph satisfying

$$\forall v \in V, \sum_{c \in N(v): w(v,c) > 0} w(v,c) < \sum_{c \in N(v): w(v,c) \leq 0} (-w(v,c)) + \gamma(v) \tag{12}$$

$$\forall c \in C, \forall v, v' \in N(c), w(v,c) + w(v',c) \geq 0 \tag{13}$$

The second equation can be replaced with

$$\forall c \in C, \exists P_c \geq 0, \exists v \in N(c) : w(v,c) = -P_c \text{ and } \forall v' \in N(c)\ s.t.\ v' \neq v, w(v',c) = P_c \tag{14}$$

and the resulting function is then called a hyperflow[8]. Moreover, it has been shown that the existence of either of these functions for a given error pattern is in fact both sufficient and necessary for LP decoder success[9]. For the remainder of this thesis, we will choose to work with the hyperflow equations.

The following lemma follows from the definition of a hyperflow and will be used later. Informally, it states that, in a hyperflow, the total flow from the set of variable nodes to the set of check nodes cannot be greater than the total flow from the check nodes to the variable nodes by more than *n*.

Lemma 1. *Given a Tanner graph $G(V,C)$ with hyperflow w, define the quantities w_{LR}*

(L-to-R flow) and w_{RL} (R-to-L flow) as:

$$w_{LR} = \sum_{v\in V, c\in C : w(v,c)>0} w(v,c) \tag{15a}$$

$$w_{RL} = \sum_{v\in V, c\in C : w(v,c)<0} |w(v,c)| \tag{15b}$$

Then the summation w_{LR} is upper-bounded by $w_{RL}+n$.

Proof. We can rewrite (15a) as $w_{LR} = \sum_{v\in V}\left(\sum_{c\in N(v):w(v,c)>0} w(v,c)\right)$, and (15b) as $w_{RL} = \sum_{v\in V}\left(\sum_{c\in N(v):w(v,c)>0} w(v,c)\right)$. Now we can use equation (12) for each variable node v to obtain:

$$w_{LR} - w_{RL} < \sum_{v\in V} \gamma(v) < n$$

which completes the proof. □

2.4. LP Decoder Performance

We have already seen that the girth of the code graph plays an important role in the iterative decoding of LDPC codes, but as this paper deals with LP decoding, we should mention first a few results by Feldman et al. on the performance of LP decoders and its relation to properties of the underlying code graph. The results will also be of use in some of the proofs in later sections. We mention an important quantity that quantifies the error-correction capability of a code when the LP decoder is used. It is defined here over a sequence of codes rather than an ensemble, hence the definition may be somewhat at odds with others found in the literature.

Definition 1. *Let $\{C_n\}$ be a sequence of linear codes of codeword length n. The* ***LP-threshold*** *t for this sequence is defined as:*

$$t = \sup\{\varepsilon > 0 \mid \Pr_{\varepsilon\text{-}BSC}[\textit{LP error on } C_n] \to 0 \textit{ as } n \to \infty\} \tag{16}$$

To state some LP decoding performance bounds over graphs with good girth, we start with the following definition from [3]:

Definition 2. *Given a polytope P defined over some variables $\{f_i\}_{i \in I}$ where the variables $0 \leq f_i \leq 1$ correspond to the coordinates of the point f in the polytope, define the weight of a point $f \in P$ to be the sum $\sum_i f_i$. The* ***fractional distance*** *of a code is then defined to be the minimum weight of any vertex in the set of non-zero vertices of P.*

We can think of this fractional distance as an extension of the hamming distance for fractional codewords. In fact, as can be seen from the definition, the fractional distance serves as a lower bound for the hamming distance of the code since every actual codeword is also a vertex in the polytope P. Feldman goes even further by exhibiting the following relation between LP decoder performance and fractional distance:

Theorem 1. *For a code G with a fractional distance d_{frac}, the LP decoder is successful if at most $\lceil d_{frac}/2 \rceil - 1$ bits are flipped by the BSC.*

Proof. See theorem 9 from [3]. □

Just as girth may be used to derive a lower-bound on the hamming distance of a code, it can also be used to derive a lower-bound on its fractional distance. This is made precise by the following theorem[2] , also proved in [3].

Theorem 2. *Let G be a factor graph with $d_l \geq 3$ and $d_r \geq 2$. Let g be the girth of G, $g > 4$. Then the fractional distance is at least $d_{frac} \geq (d_l - 1)^{\lceil g/4 \rceil - 1}$.*

For graphs G with girth $\Omega(\log n)$, we thus have (from theorem 2) that G has a fractional distance at least on the order of $d_l^{\log n}$, and (from theorem 1) that the LP decoder

[2]Actually, the theorem references another quantity called the max-fractional distance, defined as

$$d_{frac}^{max} = \min_{f \in \wp} \left(\frac{\sum_i f_i}{\max_i f_i} \right) \tag{17}$$

but we will not be concerned with this distinction, as it can be shown that theorem 1 also applies for the max-fractional distance.

can correct up to $\Omega(n^{1-\varepsilon})$ errors, where ε can be chosen arbitrarily small. That is, we can correct any sublinear number of errors given code graphs of good girth.

A higher lower-bound on the minimum number of correctible errors can be derived for expander graphs, and these higher bounds will be used later in this paper.

Definition 3. *A (L,R) bipartite graph is called an (d,γ,α)-expander if every subset $S \subset L$ of size less than γn has at least $d\alpha|S|$ neighbors.*

We already know that such graphs exist. When they are used to construct code graphs, they can yield powerful codes with better provable lower bounds on correctable errors:

Theorem 3. *Let G be an $(\alpha n, \delta c)$-expander where $\delta > 2/3 + 1/(3c)$ and δc is an integer, and let C be an LDPC code with length n and rate at least $1 - \frac{m}{n}$ whose Tanner graph is G. Then the LP decoder succeeds, as long as at most $\frac{3\delta-2}{2\delta-1}(\alpha n - 1)$ bits are flipped by the channel.*

Proof. See section IV from [10]. □

Theorem 3 shows that LP decoding can be used to correct a linear fraction of the errors introduced by the BSC, when the code graph G is an expander. Equivalently, for the LP polytope of expander graphs, there are no fractional codewords of sublinear weight. This is in contrast to the case of good-girth graphs, which only guarantees a sublinear number of correctable errors (albeit for any exponent below 1). Efficient constructions of expander codes have already been derived, though it will suffice to know only that random d-regular bipartite graphs are known to be good expanders with high probability.

Our final observation concerning LP decoding performance is that LP decoding is useless for codes that do not contain low-degree check nodes, which will be the case for random high-density parity check codes (with high probability). This will explain the rationale behind restricting our discussion to LDPC codes to begin with. We will prove a lemma from which this claim follows trivially, and which will be of use later. A quantity that will need to be defined first is a variable node's ***in-flow***:

Definition 4. *Given a hyperflow w on some Tanner graph G containing a variable node v, define v's in-flow to be* $\sum_{c \in N(v), w(v,c)<0} |w(v,c)|$.

Lemma 2. *Let $\{C_n\}$ be a sequence of codes of length n, and assume all check nodes of code C_n's Tanner graph have degree at least $f(n)$, where $f(n)$ is a strictly increasing function of n. Then, for any pair of positive constants σ, δ there exists an integer N such that for all $n \geq N$, there are no hyperflows on code C_n's Tanner graph having δn variable nodes with in-flow greater than σ.*

Proof. Assume there is a pair of constants $\sigma > 0$, $\delta > 0$ that violates the conditions of the statement of the lemma. Then there is a subsequence $\{D_n\}$ of $\{C_n\}$ such that each code D_n has a Tanner graph with a hyperflow w such that δn variable nodes of D_n have in-flow greater than σ. Thus the R-to-L flow, as defined by lemma 1, is given by a function $\alpha(n) = \Omega(n)$, and the L-to-R is at least $\alpha(n)f(n)$. Their difference is therefore $w_{LR} - w_{RL} = \alpha(n)f(n) - \alpha(n) = \alpha(n)(f(n)-1) = \omega(n)$. By lemma 1, we know this difference can't be $\omega(n)$ for a valid hyperflow, so we conclude that there exist no σ, δ that violate the statement of the theorem. $\square$

Our claim regarding high-density codes follows:

Theorem 4. *Let $\{C_n\}$ be a sequence of codes of length n such that C_n's Tanner graph has a minimum check-node degree $f(n) \to \infty$. The LP-threshold for this sequence is then 0.*

Proof. Assume we are operating in the ε-BSC, where $\varepsilon > 0$. Take $0 < \varepsilon' < \varepsilon$. The probability that there are more than $\varepsilon' n$ incorrectly received variable nodes clearly goes to 1 as n increases (applying the Chernoff bound gives Pr[less than $\varepsilon' n$ errors] $< e^{-\frac{(1-\varepsilon'/\varepsilon)^2 \varepsilon n}{2}}$). If there are more than $\varepsilon' n$ errors, then these $\varepsilon' n$ variable nodes would each require an in-flow greater than 1. By lemma 2 we can find an integer N such that for all $n > N$, such a hyperflow cannot exist. That is, for all $n > N$, the decoder fails unless there are less than $\varepsilon' n$ errors, which occurs with probability going to zero. Since ε was chosen arbitrarily, the LP-threshold for the sequence is indeed 0. □

2.5. Redundant Parity Checks

The polytope defined above is just one relaxation that achieves a polynomial-in-n number of facets. It is not at all clear what relaxations are achievable while keeping the number of facets polynomial. One method of tightening the previous model, mentioned in [3], is the addition of redundant check nodes to the code graph. A redundant parity check is a parity check obtained through the addition of two or more parity checks from the original code graph. That is, a redundant parity check obtained by combining the check nodes $\{c_i\}$ contains an edge to variable node v if and only if an odd number of check nodes in S are connected to v. Since the LP polytope is the intersection of all individual check node polytopes, even a redundant check node (which does not alter the code) may strengthen the polytope. Some redundant checks are verifiably useless for this purpose; indeed, it will be shown first that adding a redundant check node over check nodes that do not form a cycle will not alter the polytope.

Of course, these joint polytopes represent one way to tighten Feldman's relaxation, and the simple proposition above makes it clear that only checks whose neighbors form cycles should be considered. We now know that these redundant checks must be placed on a cycle for any reduction in the polytope's size. It is of theoretical interest to know just what extent of tightening is possible through a similar approach. In this thesis, we investigate whether redundant checks might be useful for the LP decoding of LDPC codes with good girth (i.e. the LDPC codes encountered in practice).

Redundant cycles could also be added adaptively, once it is clear which cycles would help avoid LP decoder failure. Such an approach is taken in [11]. This form of adaptive decoding is possible due to the ML certificate property: as the LP solver must fail by returning an invalid codeword, it can be run again after modifying a region of the polytope containing the fractional codeword returned. Our interest, however, is in refining the polytope prior to running the LP solver, as there has been very little work done to this end.

In fact, a whole range of reductions could be possible using other techniques, and there is much theoretical interest in knowing if there are any that lie between the ML polytope and Feldman's relaxed polytope such that (i) there are polynomially many facets and (ii) the gap in error-probability to ML decoding is significantly reduced. A number of generic polytope reductions that apply to arbitrary 0-1 integer programming problems are explored in [12] and the original LP decoding paper by Feldman suggested using these reductions on the LP-decoding polytope to examine the resulting reduction in error probability.

CHAPTER 3

IMPACT OF REDUNDANT PARITY CHECKS

3.1. Redundant Parity Check Over Trees

The claim above that check nodes placed above other checks that do not form a cycle would not tighten polytope is easily proved.

Recall the general idea behind the LP relaxation of Feldman: rather than taking the convex hull of all valid codewords (vertices satisfying all the parity constraints), we take the convex hull of all codewords satisfying some parity constraint j to obtain an intermediate polytope $\wp_j$, then intersect all the polytopes $\wp_j$ to obtain the final polytope $\wp$. The effect of adding a redundant parity check r on the LP decoder for some Tanner graph can therefore be visualized as intersecting this polytope with an additional intermediate polytope $\wp_r$: that corresponding to the new redundant parity constraint. Consider the polytope $\bigcap\limits_{i \in A} \wp_i$ for some set of check nodes A in the original graph. We can form a ***joint polytope*** from these $|A|$ checks as follows:

Definition 5. *The joint polytope for the parity checks in a set A is the convex hull of codewords satisfying all parity constraints in A*[1]*. The* ***code node*** *j_A corresponding to the checks in A is a check node connected to the variable nodes in $N(A)$ such that a code satisfies j_A if it satisfies all the parity checks in A.*

With the previous definition, we can relate the code node j_A with the joint polytope as follows:

Theorem 5. *Given a set A of parity checks corresponding to code node j_A, let E_{j_A} be the set of sets $S \subset N(A)$ such that if the variable nodes in S are all 1 and the variable nodes*

[1]Note in particular that the joint polytope of all the check nodes is thus the original ML decoding polytope.

in $N(A)/S$ are 0, then the parity checks in A are all satisfied. Then the convex hull of the set of codewords satisfying the constraints in A is given by the set of all f for which there exists variables $w_{j_A,S}$ such that:

$$\forall i, 0 \leq f_i \leq 1 \tag{18}$$

$$\forall S \in E_{j_A}, 0 \leq w_{j_A,S} \leq 1 \tag{19}$$

$$\forall i \in N(j_A), f_i = \sum_{S \in E_{j_A}, i \in S} w_{j_A,S} \tag{20}$$

$$\sum_{S \in E_{j_A}} w_{j_A,S} = 1 \tag{21}$$

If the joint polytope of the parity constraints in A is the same as the polytope $\bigcap\limits_{i \in A} \wp_i$, then the intersection of the polytope $\wp_r$ (where r is now the redundant parity constraint formed from the checks in A) will not lead to a reduction in polytope size. To see why this is the case, compare the joint polytope with the added polytope $\wp_r$: a codeword that satisfied all parity constraints in $|A|$ would clearly satisfy the redundant constraint formed by r, hence the convex hull $\wp_A$ of these codewords would be a subset of the convex hull of the codewords satisfying r.

The purpose of this section will be to show that if the parity checks in A are such that they do not form a cycle, then adding a redundant parity check over A will not alter the LP decoding polytope described by equations (18) - (21).

Theorem 6. *Let A be a set of parity checks in some Tanner graph $G(V,C)$ such that there exists no cycle in the subgraph formed by the variable and check nodes in $\bigcup_{c \in C}(c \cup N(c))$. Let r_A be a redundant parity check over the check nodes in A. Then, the LP fundamental polytope for the Tanner graph $G(V,C)$ is equivalent to the fundamental polytope of $G(V, C \cup r_A)$.*

We restrict our attention first to two check nodes j_1 and j_2 that share a single variable node. Both nodes are assumed to be code nodes as defined in Definition 5; a parity check j is therefore a special case of code node j_A where the set A consists of a single parity check (j itself). We will show that the joint polytope of these two code nodes is equivalent to the intersection of the two polytopes $\wp_{j_1} \bigcap \wp_{j_2}$. We know that $\wp \subset \wp_{j_1} \bigcap \wp_{j_2}$ so we need only show the opposite inclusion. Let f be a fractional codeword in both polytopes. Then by (9) - (11) we can find variables w_{j_1,S_1} and w_{j_1,S_2} such that:

$$\forall S \in E_j, 0 \leq w_{j,S} \leq 1 \tag{22}$$

$$\forall i \in N(j), f_i = \sum_{S \in E_j, i \in S} w_{j,S} \tag{23}$$

$$\sum_{S \in E_j} w_{j,S} = 1 \tag{24}$$

We need to form variables $\{w_{j_A,S}\}$ where $A = \{1,2\}$ and $S \in E_{j_A}$ such that, if equations (22) - (24) are satisfied by (f, w_{j_1,S_1}) and (f, w_{j_2,S_2}), those same equations would still be satisfied for (f, w).

Since this is equivalent to the statement that any f in , we conclude that we gain nothing by joining these two nodes. Before furnishing a proof, it should be noted that this single case is sufficient, since any tightening of the polytope obtained by considering the joint polytopes of multiple check nodes whose variable nodes do not form a cycle could be obtained by repeatedly considering two check nodes with some shared neighbor[2].

Lemma 3. *Let j_1 and j_2 be two code nodes sharing a single variable node v. Suppose there exist two sets $\{w_{j_1,S}\}$ and $\{w_{j_2,S}\}$ satisfying equations (18) - (21). Then there exists a third set $\{w_{j_A,S}\}$ satisfying those equations for $A = \{1,2\}$.*

Proof. If S is the subset of variable nodes connected to $j_1 \cup j_2 \setminus v$, let $S' = S \cap N(j_1)$ and $S'' = S \cap N(j_2)$. We define $w_{j_A,S}$ to be

[2]Trivially, if two check nodes share more than two variable nodes, they would form a cycle. So we are justified in assuming only a single shared variable node.

$$\begin{cases} w_{j_1,S'\cup v}w_{j_2,S''\cup v}/f_v & \text{if } v\in S, f_v\neq 0 \\ w_{j_1,S'}w_{j_2,S''}/(1-f_v) & \text{if } v\notin S, f_v\neq 1 \\ 0 & \text{if } f_v=0 \\ 1 & \text{if } f_v=1 \end{cases} \tag{25}$$

where f_v denotes the (fractional) value of f on variable node v. The proof is complete if we can show that equations (18) - (21) are satisfied for $(f, w_{j,S})$. Equations (18) and (19) are clearly satisfied. To show that equation 20 is also satisfied, we consider first the case where the i^{th} node is the common variable node v. Then

$$\begin{aligned}\sum_{S\in E_j, i\in S} w_{j,S} &= \sum_{S\in E_j}(w_{j_1,(S'\cup v)}w_{j_2,(S''\cup v)})/f_v \\ &= \frac{1}{f_v}\sum_{(S'\cup v)\in E_{j_1}} w_{j_1,S'\cup v}\sum_{(S''\cup v)\in E_{j_2}} w_{j_2,S''\cup v} \\ &= f_v\end{aligned}$$

As for the case $i\in N_{j_1}, i\neq v$ we get

$$\begin{aligned}\sum_{S\in E_j, i\in S} w_{j,S} &= \sum_{\substack{S\in E_j, i\in S \\ v\in S}} w_{j,S'}w_{j_2,S''}/f_v \\ &+ \sum_{\substack{S\in Ej, i\in S \\ v\notin S}} (w_{j_1,S'}w_{j_2,S''})/(1-f_v) \\ &= \sum_{(S'\cup v)\in E_{j_1}, i\in S'} \frac{w_{j_1,(S'\cup v)}}{f_v} \sum_{S''\cup v\in E_{j_2}, i\in S''} w_{j_2,S''} \\ &+ \sum_{S'\in E_{j_1}, i\in S'} \frac{w_{j_1,S'}}{f_v} \sum_{S''\in E_{j_2}, i\in S''} w_{j_2,S''} \\ &= f_i\end{aligned}$$

with the case $i\in N_{j_2}, i\neq v$ handled identically.

And (21) is satisfied since

$$\begin{aligned}\sum_{S\in E_j} w_{j,S} &= \frac{1}{1-f_v}\sum_{S'\in E_{j_1}}\sum_{S''\in E_{j_2}} w_{j_1,S'}w_{j_2,S''}\\ &+\frac{1}{f_v}\sum_{(S'\cup v)\in E_{j_1}}\sum_{(S''\cup v)\in E_{j_2}} w_{j_1,(S'\cup v)}w_{j_2,(S''\cup v)}\\ &=1\end{aligned}$$

For the case $f_v = 0$, the verification is also trivial:

$$\begin{aligned}\sum_{S\in E_j, i\in S} w_{j,S} = 0\\ = f_v\end{aligned}$$

since $w_{j,S} = 0$ for all S that contain v when $f_v = 0$.

Moreover, we have:

$$\begin{aligned}\sum_{S\in E_j, i\in S} w_{j,S} &= \sum_{\substack{S\in E_j, i\in S\\ v\in S}} 0\\ &+\sum_{\substack{S\in Ej, i\in S\\ v\notin S}} (w_{j_1,S'}w_{j_2,S''})/(1-f_v)\\ &=\sum_{S'\in E_{j_1}, i\in S'} w_{j_1,S'}\sum_{S''\in E_{j_2}} w_{j_2,S''}\\ &= f_i\end{aligned}$$

since $\sum_{S''\in E_{j_2}} w_{j_2,S''} = 1$ and $\sum_{S'\in E_{j_1}, i\in S'} w_{j_1,S'} = f_i$.

And finally for summation (21) we have:

$$\sum_{S\in E_j} w_{j,S} = \sum_{S\in E_j, v\in S} w_{j,S} + \sum_{S\in E_j, v\notin S} w_{j,S} = 0+1 = 1$$

The proof when $f_v = 1$ follows similarly. □

Any redundant check that does not form a cycle with its neighboring variable nodes can be handled using the above lemma, by sequentially merging two of its check nodes that share a single variable node. An immediate corollary for this lemma (which will also be needed) is now given.

Corollary 1. *Let j_1 and j_2 be two code nodes sharing a single variable node v. Suppose there exist two sets $\{w_{j_1,S}\}$ and $\{w_{j_2,S}\}$ satisfying equations (18) - (21). Then there exists a third set $\{w_{j_A,S}\}$ satisfying those equations for $A = \{1,2\}$.*

Proof. Let j_3 be a check node of degree 2 attached a one variable node from each of $N(j_1), N(j_2)$. Clearly, the joint polytope $\wp_B$ where $B = 1,2,3$ is a subset of $\wp_A$. From the previous lemma, we know that the joint polytope is in fact equivalent to the intersection of the polytopes $\cap_{i \in B} \wp_i$ (simply apply the lemma to $B = 1,3$ and then to $B = 1,2,3$). Thus, the joint polytope for two code nodes that do not share any variable nodes is equivalent to the intersection of the individual polytopes corresponding to the code nodes. $\square$

Thus we can now give a proof for theorem 6.

Proof of theorem 6. For a check node j, let $N'(j)$ denote the set of check nodes that share a variable node with j. Since $\bigcup_{c \in A}(c \cup N(c))$ contains no cycles, the check nodes connected to r_A must form a forest. Assume for now that there is a single tree in this forest. Recall that each parity check node may be viewed as a code node j_B where B has a single parity check. We can therefore apply lemma 3 to obtain a proof by induction. The theorem holds when the tree is of size 1 since, in that case, both the fundamental and joint polytope are given by $\wp_A$. Assume now that the theorem holds for all sets of size less than $|A|$. Pick a check node j_1 from the tree. The joint polytope for $j_1 \cap N'(j_1)$ (the polytope corresponding to the code node j_B where $B = j_1 \cup N'(j_1)$) is equal to the intersection of polytopes $\bigcap_{i \in \{j_1 \cup N'(j_1)\}} \wp_i$. We may thus merge the code nodes in $j_1 \cup N'(j_1)$ without a reduction in size of the fundamental polytope. The remaining code nodes clearly form a tree of size less than $|A|$. Hence, the proof follows by induction, under the previous assumption of a single tree in the forest. If the forest had more than one tree, we would end up with several check nodes that do not share any variable nodes. By refdisjoint-corollary, the joint polytope for these code nodes is also equal to the intersection of the individual code nodes' polytopes, and we conclude that the joint polytope for all the check nodes in A is equal to the intersection of the individual polytopes for each of the check nodes. □

3.2. Redundant Parity Checks Over Random LDPC Codes

Having considered the case of redundant checks containing no cycles, we move now to those that contain at least one cycle. The purpose here is not to show that these checks do not alter the polytope (by returning to the defining equations 8 - 11), but to show that adding all such checks will not improve the LP-threshold of the LP decoder. So though such checks may reduce the polytope size, the reduction is insignificant in our present context of error correction. Essential to this part of the proof is our assumption of good girth for the underlying code graph. That is, we prove that for almost all LDPC code graphs with girth $\Omega(\log n)$, adding all redundant parity checks does not increase the LP threshold. We now state the main theorem of this thesis:

Theorem 7. *Let $\{C_n\}$ be a sequence of LDPC codes and assume there exist values for N, ξ, and Λ and a function $f(n) \to \infty$ such that the following conditions are satisfied for $n > N$.*

(a) If there exists a function w on C_n's Tanner graph G_n that satisfies equation (14) for all check nodes and equation (12) for all but ξn variable nodes, then there exists a hyperflow on G_n.

(b) The Tanner graph for code C_n has a girth of at least $\Lambda \log n$.

(c) Any sum of at least $\Lambda \log n$ parity checks results in a parity check of degree at least $f(n)$.

Then, for $n > N$, the LP thresholds for the sequence $\{C_n\}$ and the corresponding sequence $\{C'_n\}$ obtained by adding all redundant parity checks to each C_n are equal.

Before we provide a proof of this theorem we will need an additional lemma. The proof will follow the general idea behind our proof of theorem 4 for random high-density codes. To show that redundant check nodes will not help increase the LP-threshold, it is necessary to show that for there to be any non-zero difference Δ in threshold resulting from the addition of redundant checks there must exist a code graph (possibly different from the original graph) and a hyperflow on that graph with a linear-sized subset U of variable nodes such that each variable node $v \in U$ receives "non-negligible" flow from redundant nodes. We will make use of the following definition throughout the rest of the section:

Definition 6. *Let v be a variable node in a Tanner graph G that includes some redundant parity checks, and let w be a hyperflow for G under some error-pattern. Then we refer to the sum of all the edge weights for edges between a variable node v and the redundant check nodes in the hyperflow as node v's redundant-in-flow (flow coming from redundant nodes).*

To prove the above statement, we need in turn a result from [9] that will allow us to find a hyperflow with a constant lower-bound on this redundant-in-flow.

Lemma 4. *Let ζ be a binary linear code with Tanner graph (V, C, E) where $V = v1, , vn$. Let $\varepsilon, \delta > 0$ and $\varepsilon' = \varepsilon + (1-\varepsilon)\delta$. Assume that $\varepsilon, \varepsilon', \delta < 1$. Let $q_{\varepsilon'}$ be the probability of LP decoding error on the ε'-BSC. For every error pattern $x \in \{0,1\}^n$, if $G = (V, C, E, w, \gamma)$ is a WDAG corresponding to a dual witness for x, let $f(w) \in \mathbb{R}^n$ be defined by*

$$f_i(w) = \sum_{c \in N(v_i): w(v_i,c) > 0} w(v_i, c) - \sum_{c \in N(v_i): w(v_i,c) \leq 0} (-w(v_i, c)) = \sum_{c \in N(v_i)} w(v_i, c)$$

for all $i \in [n]$. Then,

$$\Pr_{x\sim Ber(\varepsilon,n)}\{\exists \textit{ a dual witness } w \textit{ for } x \textit{ s.t. } f_i(w) < \gamma(v_i) - \frac{\delta}{2}, \forall i \in [n]\} \geq 1 - \frac{2q_{\varepsilon'}}{\delta}$$

The paper fittingly calls the difference $\gamma(v_i) - f_i(w)$ the ***LP excess***. The theorem therefore establishes a lower-bound of $1 - \frac{2q_{\varepsilon'}}{\delta}$ on the probability that there is a hyperflow with an excess $\geq \frac{\delta}{2}$ on all variable nodes.

Proof of theorem 7. Denote the Tanner graph for code C_n by G_n and that for C_n' by G_n'. Assume $\varepsilon_1 < \varepsilon_2$ and take $\sigma = \frac{\varepsilon_2 - \varepsilon_1}{1-\varepsilon_1}$. Furthermore, assume we are operating on an ε_2-BSC. From lemma 4, there exists (with probability $1 - \frac{2q_{\varepsilon_2}}{\sigma} \to 1$) a hyperflow for G_n' with an excess of σ on all variable nodes; denote this hyperflow by w'. We have shown (theorem 6) that redundant parity checks over checks that do not form cycles in the original graph do not alter the LP decoding polytope. We may thus consider only redundant checks placed over checks that form a cycle in G_n'. Now, since (by hypothesis) graph G_n' has no cycles of length less than $\Lambda \log n$, all such redundant checks must be placed over at least $\Lambda \log n$ many checks in G_n'. By condition (c), all such checks thus have degree greater than $f(n)$ where $f(n) \to \infty$ for increasing n. We can now use lemma 2 to show that the number of nodes with redundant-in-flow greater than σ must be sublinear. If the number of variable nodes with redundant-in-flow greater than σ was larger than δn for all n sufficiently large, the constants σ, δ would violate lemma 2. Thus, there are only sublinearly many variable nodes in the resulting graph with redundant-in-flow greater than σ. Since the hyperflow w' was chosen to have an excess of σ on all its variable nodes, we can conclude that there exists a function w on the edges of G_n such that w has at most sublinearly many variable nodes that do not satisfy the hyperflow equation (12). Namely, we can just take w to be the function w' restricted to the edges in G_n; the variable nodes in this hyperflow all have at most σ less flow from the same variable nodes in w' and only a sublinear number of variable nodes had less than σ LP-excess in w'. From (a), we know that, for n sufficiently large, if there exists a function w on the edges of G_n that has less than ξn variable nodes that violate equation 12 (and all of whose check nodes satisfy hyperflow equation 14), then there exists a valid hyperflow on G_n for that same error pattern. This implies that for n sufficiently large, the existence of a hyperflow on G_2 implies the existence of a hyperflow on G_1 under the same error pattern. But this in turn implies that $\Pr\{\text{LP decoder failure on } G_1\} \leq \Pr\{\text{LP decoder failure on } G_2\}$, where the latter term goes to zero as $n \to \infty$ since we are operating at $\varepsilon < \varepsilon_2$. Hence, the initial assumption of $\varepsilon_1 < \varepsilon_2$ must have been incorrect, and the two codes have the same threshold, completing the proof. □

Thus, we have shown that, under the conditions stated in the hypothesis of theorem 7, redundant parity checks offer no advantage to LP decoding of LDPC codes, in that they cause no increase in LP-threshold. We have already seen why condition (b) is a valid assumption: codes with girth $\Omega(\log n)$ are the ones used in practice. Our reasoning behind the first condition comes from the fact that most graphs are expanders (the probability of a random regular-graph not being an expander decreases exponentially) and, as explained in section 3, the LP polytope over such graphs cannot return the non-zero codeword if the received codeword has weight less than δn for some constant $\delta > 0$. The conjecture that condition (a) follows from this will be briefly discussed later. In what follows, we will show why the second condition is in fact also a valid assumption for almost all graphs.

3.3. Redundant Parity Checks Over Long Cycles

For the remainder of this section, we assume the LDPC code graph to be constructed randomly such that each check node is connected to d_r variable nodes chosen at random from the n total variable nodes, for increasing values of codeword length n.

Note that this will not result in a (d_l, d_r)-regular LDPC code, and we will have more to say about this in section 4.1.. Moreover, and perhaps more importantly, the resulting code graph may not have good girth, and this issue is breifly addressed in section 4.2.. We will use G to refer to both a sequence of codegraphs for increasing codeword length and a specific graph in the sequence interchangeably. H will refer to the $m \times n$ matrix (m being the number of check nodes) such that

$$H_{i,j} = \begin{cases} 1, & \text{if check node i is connected to variable node j} \\ 0, & \text{otherwise} \end{cases}$$

H is also seen to be the parity check matrix for code G's dual, since the rows of a parity check matrix are a basis for the orthogonal space for G. For ease of notation we will call this model of randomness in the matrix ***row-randomness***, since the rows are chosen independently at random from the set of weight-d_r binary vectors of length n.

The argument for the case of redundant checks including cycles is as follows: In a graph with good girth, a redundant parity check must include a large number of parity checks from the original graph if it is to contain a cycle[3]. If we can show that such redundant checks must consequently have a large degree, then, noting that in a hyperflow a check node with degree d and an out-flow of α requires an in-flow of exactly $d\alpha$, for these nodes to supply a non-negligible flow to some variable node would require a large in-flow from the other variable nodes. Indeed, it will be shown that the in-flow required from all added redundant parity checks to cause a reduction in the LP threshold cannot be met given only n variable nodes. It would follow, then, that the addition of all redundant parity checks connected to parity checks forming cycles in the original graph could not improve the error-correction of the LP decoder.

The main difficulty, however, is in showing that a redundant parity check containing a large number of checks, each of degree d_r, cannot lead to a parity check of low degree. It is conceivable (in a graph with cycles) for a large set of parity checks to share variable nodes such that the degrees of most of these variable nodes is even, making the weight of the sum of the parity constraints (and the degree of the corresponding redundant parity check) small. It will be shown that for code graphs sampled from the uniform en-

[3]'large' is on the order of $\log n$, the girth of the graphs considered here

semble of (d_v, d_c) codes, the probability of this occurring for any redundant parity check will converge to zero exponentially as the codeword length is increased. This is the reason for our use of the random ensemble as opposed to a single deterministic code matrix. Note that it is certainly possible to construct a matrix that violates this property, but the result is sufficient as it shows that this will not be the case for almost all codes we encounter.

In the theorem below, note that the term $c \log n$ arises from the girth of the code graphs we are working with. Under this assumption, any subset of less than $c \log n$ rows contains no cycles, and does not affect the LP threshold by Theorem 3.

Theorem 8. *Let $H_{m,n}$ denote an $m \times n$ matrix constructed by randomly selecting each of its m rows from the set of binary vectors of length n and weight d_c. There exists a threshold β such that if $m < \beta n$, the probability that $H_{m,n}$ has a combination of at least $c \log n$ rows that sum to a vector of weight less than any fixed constant N goes to zero as $n \to \infty$.*

The proof of theorem 8 follows an argument by Calkin[13], though that paper deals with the probability of a row-randomly constructed matrix H having less than full rank. That is, Calkin proves the existence of a threshold β as in theorem 8 above such that a row-random matrix of uniform row weight and $\frac{\text{\# of rows}}{\text{\# of columns}} < \beta$ $(resp. > \beta)$ will have full rank with a probability that converges to 1 (resp. 0) as $n \to \infty$.

In what follows, we will view the addition of rows of the matrix H as a Markov process, and attempt to upper-bound the probability that such a process will lead to a sum of weight less than N. We recall first the definition of a Markov chain, and its associated transition matrix.

Definition 7. *If we let the random variables $X_1, X_2, \ldots, X_n$ denote the state in a stochastic process, the sequence is said to form a Markov chain if knowledge of the current state makes future states independent of past ones, that is, if $\Pr\{X_n | X_1, X_2, \ldots, X_{n-1}\} = \Pr\{X_n | X_{n-1}\}$. If the transition probabilities are independent of time (the index of the current state), the process is said to be a stationary Markov process. We can then form a matrix P, called the transition matrix, whose ij^{th} entry is given by the probability*

$\Pr\{X_n = j | X_{n-1} = i\}$ *of transitioning from the* i^{th} *to the* j^{th} *state in any given index. Furthermore, the transition matrix for a sequence of m consecutive transitions in the process is given by the matrix product* P^m.

We will modify the proof to show we can get no low-weight vectors from summing a large number of rows. We start with a zero vector and iteratively add rows chosen uniformly at random from the set of weight-d_c vectors in $\{0,1\}^n$. The process clearly forms a Markov chain whose states are the weight of the summation, and whose transition matrix is given by $A = \{a_{pq}\}$ where a_{pq} is the state transition probability

$$a_{pq} = \frac{\binom{q}{\frac{k-p+q}{2}}\binom{n-q}{\frac{k+p-q}{2}}}{\binom{n}{k}},$$

and the numerator is interpreted to be zero if $k+q+p$ is even. This expression can be motivated as follows: starting with a vector of weight q, there are $\binom{n}{k}$ different ways to choose the weight-k binary vector to be added. This vector will flip k bits from the original vector and must lead to a weight difference of $p-q$. The original vector has q ones. Let k_1 and k_2 denote the number of ones and zeros flipped, respectively, from the original vector. We must have that $k_1 + k_2 = k$ and $k_2 - k_1 = p - q$; the only solution is thus $k_1 = \frac{k-p+q}{2}$, $k_2 = \frac{k+p-q}{2}$. There are obviously $\binom{n}{\frac{k-p+q}{2}}\binom{n-q}{\frac{k+p-q}{2}}$ ways to do this.

The resulting matrix A must of course be symmetric. Simply consider the probabilities a_{pq} and a_{qp}: adding any vector v_2 twice to some vector v_1 will result in v_1, and we can use this to obtain a one-to-one correspondence between vectors that cause a weight transition $p \to q$ and those that cause a weight transition $q \to p$. Hence, there are as many vectors that cause either weight transition and $a_{pq} = a_{qp}$ for any p,q. It follows from linear algebra then that we may diagonalize A as $A = P^{-1}AP$ where P is an invertible matrix. More specifically, we may diagonalize A using its eigenvectors by taking P to be the matrix whose *ith* column vector is the i^{th} eigenvector. The theorem below involves tedious algebraic manipulations and we will omit the proof. Refer to (2.1) and (2.2) in

Calkin's paper for a detailed derivation.

Theorem 9. *Let λ_i and e_i denote the eigenvalues of A and their corresponding eigenvectors, respectively. Let U be a matrix whose ith column is e_i and let Λ be a diagonal matrix with ith diagonal entry λ_i.*

(a) The eigenvalues λ_i of A are given by

$$\lambda_i = \sum_{t=0}^{k} (-1)^t \frac{\binom{i}{t}\binom{n-i}{k-t}}{\binom{n}{k}} \tag{26}$$

(b) The jth component of e_i is given by

$$e_i[j] = \sum_{t=0}^{j} (-1)^t \binom{i}{t}\binom{n-i}{j-t} \tag{27}$$

(c) $U^2 = 2^n I$

(d) $A = (1/2^n)U\Lambda U$

We are interested in the probability of returning to any of states {0, ..., N} after t steps. That is, we are interested in the sum $\sum_{p=0}^{N} a_{p0}^{(t)}$, where $a_{pq}^{(t)}$ denotes the pq^{th} entry of the transition matrix for t iterations of the above Markov process, that is, the pq^{th} entry for matrix A^t.

From parts (c) and (d) of Theorem 9, $a_{p0}^{(t)}$ is given by $\sum_{i=0}^{n} \frac{1}{2^n} e_i[p]\lambda_i^t \binom{n}{i}$. Throughout the proof, we will make use of the following naive bound on $e_i[p]$: $e_i[p] = \sum_{t=0}^{p} (-1)^t \binom{i}{t}\binom{n-i}{p-t} \leq (N+1)\binom{n}{N}^2$ for all $p \in [0,N]$ and n sufficiently large. Hence, $\sum_{p=0}^{N} a_{p0}^{(t)}$ is bounded by $\frac{(N+1)\binom{n}{N}^2}{2^n} \sum_{i=0}^{n} \lambda_i^t \binom{n}{i}$.

Definition 8. *Let $H_{m,n}$ be the matrix constructed in theorem 8. We define M to be the number of combinations of rows of $H_{m,n}$ that sum to a binary vector of weight less than N.*

In what follows, we will attempt to upper-bound the expected value $E(M)$ with a bound that converges to zero for increasing n. From the previous discussion, we have that

$$E(M) \leq \sum_{t=c\log n}^{m} \binom{m}{t} \sum_{i=0}^{n} \frac{N+1}{2^n} \lambda_i^t \binom{n}{N}^2 \binom{n}{i} = \sum_{i=0}^{n} \left(\sum_{t=c\log n}^{m} \frac{N+1}{2^n} \lambda_i^t \binom{n}{N}^2 \binom{n}{i} \right) \tag{28}$$

However, as will become apparent in the proof of lemma 6 below, the fact that t is restricted to be above $c\log n$ will only be needed for $i \in \left(\frac{n}{2} - n^{4/7}, \frac{n}{2} + n^{4/7}\right)$. That is, for $i \notin \left(\frac{n}{2} - n^{4/7}, \frac{n}{2} + n^{4/7}\right)$, we will only need:

$$\sum_{t=c\log n}^{m} \binom{m}{t} \frac{N+1}{2^n} \lambda_i^t \binom{n}{N}^2 \binom{n}{i} \leq \sum_{t=0}^{m} \binom{m}{t} \frac{N+1}{2^n} \lambda_i^t \binom{n}{N}^2 \binom{n}{i} \tag{29}$$

$$= \frac{N+1}{2^n} \binom{n}{N}^2 \binom{n}{i} (1+\lambda_i)^m \tag{30}$$

We will show that the added factor of $(N+1)\binom{n}{N}^2$ doesn't affect the asymptotic behaviour of $E(M)$. The following bounds on λ_i will also be needed:

Lemma 5. *(a)* $|\lambda_i| < 1$ *for all* $0 \leq i \leq n$.

(b) If $i > \frac{n}{2}$ *then* $\lambda_i = (-1)^k \lambda_{n-i}$.

(c) If $0 < \xi < \frac{1}{2}$ *and* $i = \xi n$ *then*

$$\lambda_i = \left(1 - \frac{2i}{n}\right)^k - \frac{4\binom{k}{2}}{n}\left(1 - \frac{2i}{n}\right)^{k-2} \frac{i}{n}\left(1 - \frac{i}{n}\right) + O\left(\frac{k^3}{\xi^2 n^2}\right)$$

(d) If $\theta < 1 - \frac{1}{k}$ *and* $\frac{n}{2} - i = \frac{n^\theta}{2}$ *then* $\lambda_i = o\left(\frac{1}{n}\right)$.

Proof. See Lemma 3.1 in Calkin's paper and the comments following it. □

Define $f(\alpha,\beta) = -log(2) - \alpha(\alpha) - (1-\alpha)\log(1-\alpha) + \beta\log(1+(1-2\alpha)^k)$ and let (α_k, β_k) be the root of $f(\alpha,\beta) = 0$ and $\frac{\partial f(\alpha,\beta)}{\partial \alpha} = 0$.

Lemma 6. *There exists a positive constant* β_k *such that if* $\beta < \beta_k$ *and* $m \leq \beta n$ *then* $E(M) \to 0$ *as* $n \to \infty$.

Proof. The summation is divided into the same parts as in Calkin's paper, and each is seen to be unaffected (asymptotically) by the added factor. The total range is $i \in [0,n]$ and the expression to be evaluated is (28).

For the tail $i \in [0, \varepsilon n]$, we use the bound $(1+\lambda_i)^m < 2^m$:

$$\begin{aligned}\sum_{i=0}^{\varepsilon n} \frac{N+1}{2^n} \binom{n}{N}^2 \binom{n}{i} (1+\lambda_i)^m &< \sum_{i=0}^{\varepsilon n} \frac{N+1}{2^{n-m}} \binom{n}{N}^2 \binom{n}{i} \\ &< \varepsilon n \binom{n}{N}^2 \frac{N+1}{2^{n-m}} \binom{n}{\varepsilon n} \to 0\end{aligned}$$

for ε sufficiently small. Note that the convergence is due to the fact that $2^{n-m} = 2^{\Delta n}$ for some $\Delta > 0$ (since m is assumed to be a fraction of n) and $\binom{n}{\varepsilon n} < \left(\frac{en}{\varepsilon n}\right)^{\varepsilon n} = \left(\frac{e}{\varepsilon}\right)^{\varepsilon n} = \left(\left(\frac{e}{\varepsilon}\right)^{\varepsilon/\Delta}\right)^{\Delta n}$. Now,

$$\frac{\left(\left(\frac{e}{\varepsilon}\right)^{\varepsilon/\Delta}\right)^{\Delta n}}{2^{\Delta n}} = \left(\frac{\left(\frac{e}{\varepsilon}\right)^{\varepsilon/\Delta}}{2}\right)^{\Delta n}$$

and we can select $\varepsilon < \Delta$ small enough so that this last expression goes to zero (this in turn follows trivially when we recall that $n^{1/n} \to 1$ as $n \to \infty$). Similarly for the range $(1-\varepsilon)n < i < n$,

$$\begin{aligned}\sum_{i=(1-\varepsilon)n}^{n} \frac{N^2}{2^n} \binom{n}{N}^2 \binom{n}{i} (1+\lambda_i)^m &< \sum_{i=(1-\varepsilon)n}^{n} \frac{N^2}{2^{n-m}} \binom{n}{N}^2 \binom{n}{i} \\ &< \varepsilon n \binom{n}{N}^2 2^{m-n} \binom{n}{\varepsilon n} \to 0\end{aligned}$$

For the range $\frac{n}{2} - n^{4/7} < i < \frac{n}{2} + n^{4/7}$, where $\frac{4}{7}$ was chosen so that θ in (d) of lemma (5) satisfies the condition $\theta < 1 - \frac{1}{d_c}$ required for $d_c \geq 3$, the naive bound above in (29) will not suffice, since, for example, at $i = \frac{n}{2}$ the Stirling approximation on $\binom{n}{i}$ gives $\frac{2^n}{(\pi\frac{n}{2})^{1/2}}$. Instead, we return to the original expression for $E(M)$:

$$E(M) \leq \sum_{t=c\log n}^{m} \binom{m}{t} \sum_{i=0}^{n} \frac{N^2}{2^n} \lambda_i^t \binom{n}{N}^2 \binom{n}{i} = \sum_{i=0}^{n} \left(\sum_{t=c\log n}^{m} \binom{m}{t} \lambda_i^t \right) \frac{N^2}{2^n} \binom{n}{N}^2 \binom{n}{i}$$

Consider the term $\left(\sum_{t=c\log n}^{m}\binom{m}{t}\lambda_i^t\right)$. As will be seen, lemma 5 above guarantees a decreasing λ_i in the interval in question. Hence, we expect there to be a small tail in the summation for low weight codewords $(t < c\log n)$, and indeed the summation can be shown to converge to zero.

Using the bound $\binom{m}{t} \leq \left(\frac{em}{t}\right)^t$ and the bound (by (d) of Theorem 5) $\lambda_i \leq \left(\frac{1}{\omega(n)}\right)$, we get:

$$\begin{aligned}\sum_{t=c\log n}^{m}\binom{m}{t}\lambda_i^t &\leq \sum_{t=c\log n}^{m}\left(\frac{em}{t}\right)^t\left(\frac{1}{\omega(n)}\right)^t\\ &\leq \max_{t\geq c\log n} m\left(\frac{1}{t^t}\right)\left(\frac{em}{\omega(n)}\right)^t\\ &= O\left(\frac{1}{(\log n)^{\log n}}\right)\end{aligned}$$

where in the last equality we used the fact that $m \leq n$ so that $\frac{em}{\omega(n)} < 1$ for n sufficiently large, and the expression is maximized by minimizing t.

We thus have:

$$\sum_{i=\frac{n}{2}-n^{4/7}}^{\frac{n}{2}+n^{4/7}}\left(\sum_{t=c\log n}^{m}\binom{m}{t}\lambda_i^t\right)\frac{N+1}{2^n}\binom{n}{N}^2\binom{n}{i} = O\left(\frac{1}{(\log n)^{\log n}}(N+1)\binom{n}{N}^2\right)$$
$$\to 0$$

since $\sum_{i}^{n}\binom{n}{i} \leq 2^n$.

For $\frac{n}{2}(1-\varepsilon) < i < \frac{n}{2} - n^{4/7}$, we have $\lambda_i < \varepsilon^k - \frac{\binom{k}{2}}{n}\varepsilon^{k-2} + O(\frac{k^3}{n^2})$ and $(1+\lambda_i)^m < e^{n\varepsilon^k - \binom{k}{2}\varepsilon^{k-2}}$, so:

$$\sum_{i=\frac{n}{2}(1-\varepsilon)}^{\frac{n}{2}-n^{4/7}}\frac{N^2}{2^n}\binom{n}{N}^2\binom{n}{i}(1+\lambda_i)^m < \sum_{i=\frac{n}{2}(1-\varepsilon)}^{\frac{n}{2}-n^{4/7}}\frac{N^2}{2^n}\binom{n}{N}^2\binom{n}{i}e^{n\varepsilon^k - \binom{k}{2}\varepsilon^{k-2}} \to 0$$

At this point, we need to explain the choice of function $f(\alpha,\beta)$ defined earlier.

The value $f(\frac{i}{n},\frac{m}{n})$ in fact corresponds to the value f such that $\exp nf(\frac{i}{n},\frac{m}{n})$ is equal to the i^{th} term in equation (30) excluding the factor $(N+1)\binom{n}{N}^2$. Hence, for negative values of f, the term would decrease exponentially to zero and if the function f is negative for all $\left(\frac{i}{n},\frac{m}{n}\right)$ then the sum would also decrease exponentially to zero. Moreover, the added factor $(N+1)\binom{n}{N}^2$ clearly does not affect this convergence.

$$\begin{aligned} 2^{nf(\frac{i}{n},\frac{m}{n})} &= \frac{\left(1+(1-2\alpha)^k\right)^m}{2^n} 2^{-\alpha\log\alpha-(1-\alpha)\log(1-\alpha)} \\ &\approx \frac{1}{2^n}(1+\lambda_i)^m \binom{n}{i} \end{aligned}$$

where we used (c) from lemma (5) to approximate the numerator and the following relation between binary entropy and binomial coefficients:

$$H(\alpha) \approx \frac{1}{n}\log\binom{n}{pn}$$

to approximate the denominator.

Thus, if $f(\frac{i}{n},\frac{m}{n}) < \gamma < 0$ for all $\alpha \in (\varepsilon, 1-\varepsilon)$, then

$$\sum_{i=\varepsilon n}^{\frac{n}{2}(1-\varepsilon)} \frac{N^2}{2^n}\binom{n}{N}^2\binom{n}{i}(1+\lambda_i)^m < \binom{n}{N}^2 ne^{\gamma n+o(n)} \to 0$$

Note that this holds for the same (α_k,β_k) threshold parameters as for the 0-weight case. Hence, as long as $\beta < \beta_k$ we have that $E(M) \to 0$.

□

Though we are working exclusively with low-density codes of fixed weight, it should be noted that the threshold parameter varies with the weight k of each row according to the (approximate) relationship $\beta_k \sim 1 - \frac{e^-k}{\log 2}$. The thresholds can be solved numerically, and below is a table of bounds on the threshold from Darling[14]. For the codes under consideration, the values are clearly seen to be sufficient, that is, for all prac-

Row weight (k)	β_k
3	0.917935
4	0.976770
5	0.992438
6	0.997380
7	0.999064
8	0.999660

Table 1: Threshold β_k for values of k from 3 to 8

tically encountered code rates ($\frac{n-m}{n}$) theorem 8 holds.

Thus, we have shown that the third condition mentioned in theorem 7 is in fact true with probability going to one for randomly selected LDPC codes, that is, for almost all LDPC codes. Theorem 7 may therefore be interpreted as a statement that the addition of all redundant parity checks to almost any LDPC code of good girth will not lead to an increase in the LP threshold, as long as it satisfies the first condition.

CHAPTER 4

FUTURE WORK

4.1. Extension to (d_l, d_r)-regular Codes

Theorem 6 above applies in the case of equal weight rows chosen randomly and independently. This leads to column weights also being random, meaning the code is actually not an (d_v, d_c) regular LDPC code. Fixing the column weights as well would make the rows dependent, and the previous analysis would therefore no longer apply. To remedy this, consider modifying the previous code graph to satisfy both the row and column constraints. Specifically, the following process is applied to the random graph constructed previously:

Algorithm 1 Convert d_l code to (d_l, d_r) code.

```
Require: $H$ is a matrix whose rows are random $k$-weight binary vectors
Ensure: A matrix $H'$ whose rows are random $k$-weight binary vectors and whose
  columns are random $l$-weight binary vectors where $ln = km$
  for i = 1 to n do
      j = 1;
      if ith column of $H$ has weight $< l$ then
          Pick any column $i < j < n$ (at random) of weight $> l$
          Pick any row $r$ (at random) such that $r(1) = 1$ and $r(j) = 0$
      else
          if ith column of $H$ has weight $> l$ then
              Pick any column $i < j < n$ (at random) of weight $< l$
              Pick any row $r$ (at random) such that $r(1) = 0$ and $r(j) = 1$
          end if
      end ifSwap $r(1)$ and $r(j)$
  end for
```

We need to show that this algorithm, when applied to the previous code matrices, will result in (i) H' having a uniform distribution over the set of all (d_l, d_r) code matrices, and (ii) $\Pr\{\exists x \in 0,1^m : w(xH') < N\} < \Pr\{\exists x \in 0,1^m : w(xH) < N\}$. The proof of (i) is trivial. A valid swap in the ith iteration of the loop above is between a random entry in the first column of the submatrix $H_{i \to n}$ and an entry on the same row of a randomly selected second column. Thus we have that, after the ith iteration, the first column of $H_{i \to n}$ has a uniform distribution over the vectors of weight l and the matrix $H_{i+1 \to n}$ has a uniform distribution over all matrices such that H has rows of weight k and $H_{1 \to i}$ has columns of weight l. At the end of the nth iteration, H' will therefore have a uniform distribution over all (l,k)-regular matrices.

For (ii) we need show that any such swap that leads to more balanced columns will reduce the probability that there exists a combination that sums to a vector of weight less than N. Let c be some combination of rows $\{r_S\}$ of H and assume we are swapping the i^{th} and j^{th} columns of some row $r \in \{r_S\}$, where $w(i) > w(j)$, $r(i) = 1$ and $r(j) = 0$. The only values affected by the swap are $c(i)$ and $c(j)$, and they are both inverted. Hence the weight of c would have increased if both were 0, decreased if both were 1, and remained unchanged otherwise. If we can show that $\Pr\{c(j) = 0\} > \Pr\{c(i) = 1\}$ then the new combination c' (after the swap) is more likely to have weight greater than c. This would show that any individual swap in the above procedure reduces the chances of a combination having weight below N. Since the proof of theorem 8 was done through an application of the union bound, this would imply that $\Pr\{\exists x \in 0,1^m : w(xH') < N\} < \Pr\{\exists x \in 0,1^m : w(xH) < N\}$, as claimed.

Now $\Pr\{c(i) = 1\}$ is the probability of an even number of 1s in the set C of i^{th} column entries in $\{r_S\}$. Intuitively, we expect this to be higher for the column with more 1s already in it. This can be verified when the columns are populated using i.i.d. Bernoulli trials, that is, when the entries are 1 with fixed probability p regardless of the other entries. For this case, the above probability is equivalent to $\frac{1}{2}\left[1 + (1-2p_i)^{|C|}\right]$ where p_i is the probability of an entry in the column being 1. Similarly for the i^{th} column: $\Pr\{c(j) = 1\} = \frac{1}{2}\left[1 + (1-2p)^{|C|}\right]$. Since there are more 1 entries in the ith column than the jth column, we have $p_i > p_j$ and $\Pr\{c(j) = 0\} > \Pr\{c(i) = 1\}$. We would need to verify this for the distribution of entries in this matrix for the extension to (d_v, d_c) random codes to be valid.

4.2. Effect of Good Girth

The matrices H considered so far in this thesis have been random – whether row-random or random (d_l, d_r)-regular. Such a construction does not ensure the resulting codes will have good girth, yet girth was a central part of the proof. For the results to be valid, theorem 7 must be shown to be true when H is chosen randomly from the ensemble of (d_l, d_r) codes having good girth. That is, we must show that the exclusion of matrices H having small cycles does not increase the probability of the sum of $O(\log n)$ its rows having a weight less than some constant N. Intuitively, exclusion of cycles should actually increase the weight. Specifically, consider the sum of two randomly chosen rows of H. For these rows to form a cycle they must share two of their adjacent variable nodes, meaning the bits corresponding to these two column positions would be zero, decreasing the weight of the sum. Things are less obvious when we consider a larger number of rows.

We will state what is required and leave the problem open for future work. Ideally, we would want a mapping ϕ from a graph with cycles smaller than $c \log n$ to one with none, such that (i) the distribution on $\phi(H)$ remains uniform over the matrices with no cycles smaller than $c \log n$ and (ii) the sum of the rows of $\phi(H)$ is at least as large as that of $\phi(H)$. Moreover, ϕ should maintain regularity of the resulting code graph so that we are still dealing with (d_l, d_r)-regular codes. The most trivial examples – breaking a cycle by removing a random edge from any small cycle – results in random matrices of good girth, but the process of removing edges does not preserve regularity and may decrease the weight of some combinations, as variables nodes that previously had odd degree might now have even degree.

There is a whole range of manipulations to the graph that might be considered for this purpose. Unfortunately, none of the techniques we considered was shown to satisfy all the above conditions, and this problem thus remains unsolved.

4.3. Hyperflows and Fractional Weight

Finally, and perhaps most importantly, we require a proof of the following statement: In a sequence of codes that can correct a linear fraction of errors introduced by the BSC channel, any edge function on the Tanner graph G_n of C_n implies the existence a hyperflow on G_n, for n sufficiently large. More informally, we are looking for a relation between the number of bit errors in the codeword returned by the LP decoder, and the minimum number of variable nodes in an edge function on G_n that violate the hyperflow equation. If it can be shown that a linear number in the former minimum implies a linear number in the latter, the proof can be considered complete. Unfortunately, this is something that was not addressed in this thesis, and represents the biggest gap in the above proofs, though the problem appears to suggest a more simple proof.

CHAPTER 5

CONCLUSION

Our work has shown that, under a reasonable assumption, redundant parity checks may be excluded as a method to tighten the polytope when it comes to the LDPC codes that are typically encountered: those of good girth. The assumption used was a relation between the error correction of the LP decoder and the number of variable nodes in a Tanner graph that satisfy the hyperflow equations, namely, that for LP decoder that correct a constant fraction of errors, the existence of an edge function that leaves at most sublinearly many variable nodes that don't satisfy their hyperflow equations implies the existence of a valid hyperflow on the same graph. By showing that (i) redundant checks over non-cycle-forming parity checks from the original code do not alter the polytope and (ii) redundant checks over cycle-forming parity checks will almost surely have high degree, we arrived at the conclusion that any "useful" parity checks require too much flow from a valid hyperflow of the code graph. This answers a question initially posed by Feldman concerning the extent of error-correction improvement possible through redundant parity checks. In his paper, Feldman presents a random code graph from the LDPC ensemble and demonstrates that the addition of all sum-of-two redundant checks causes an appreciable reduction in the error rate. According to the results presented in this thesis, any such improvement will not cause a reduction in the LP threshold for the ensemble of LDPC codes of girth $O(\log n)$. We have shown that the probability that there exists a "useful" redundant parity check for a fundamental polytope that satisfies the conjecture falls exponentially with codeword length, implying that for almost all regular LDPC codes of good girth, these checks offer no improvement.

BIBLIOGRAPHY

[1] J. Feldman. *Decoding Error-Correcting Codes via Linear Programming*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, September 2003.

[2] R. G. Gallager. *Low Density Parity Check Codes*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, September 1960.

[3] J. Feldman, M.J. Wainwright, and D.R. Karger. Using linear programming to decode binary linear codes. *Information Theory, IEEE Transactions on*, 51(3):954–972, 2005.

[4] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[5] S. Arora, C. Daskalakis, and D. Steurer. Message-passing algorithms and improved lp decoding. *Information Theory, IEEE Transactions on*, 58(12):7260–7271, 2012.

[6] Ralf Koetter and Pascal O. Vontobel. Graph-covers and iterative decoding of finite length codes, 2003.

[7] J. Feldman, T. Malkin, R.A. Servedio, C. Stein, and M.J. Wainwright. Lp decoding corrects a constant fraction of errors. *Information Theory, IEEE Transactions on*, 53(1):82–89, 2007.

[8] C. Daskalakis, A.G. Dimakis, Richard M. Karp, and M.J. Wainwright. Probabilistic

analysis of linear programming decoding. *Information Theory, IEEE Transactions on*, 54(8):3565–3578, 2008.

[9] Louay Bazzi, Badih Ghazi, and Rüdiger L. Urbanke. Linear programming decoding of spatially coupled codes. *CoRR*, abs/1301.6410, 2013.

[10] J. Feldman, T. Malkin, R.A. Servedio, C. Stein, and M.J. Wainwright. Lp decoding corrects a constant fraction of errors. *Information Theory, IEEE Transactions on*, 53(1):82–89, Jan 2007.

[11] Alexandros G. Dimakis, Amin A. Gohari, and Martin J. Wainwright. Guessing facets: Polytope structure and improved lp decoding. *CoRR*, abs/0709.3915, 2007.

[12] Monique Laurent. A comparison of the sherali-adams, lovász-schrijver, and lasserre relaxations for 0-1 programming. *Mathematics of Operations Research*, 28(3):pp. 470–496, 2003.

[13] Neil J. Calkin. Dependent sets of constant weight binary vectors. *Combinatorics, Probability and Computing*, 6(3):263–271, 1997.

[14] R. W. R. Darling, M. D. Penrose, A. R. Wade, and S. L. Zabell. Rank deficiency in sparse random GF[2] matrices. *ArXiv e-prints*, November 2012.