# Using Linear Programming to Decode Binary Linear Codes

Jon Feldman, Martin J. Wainwright, *Member, IEEE*, and David R. Karger, *Associate Member, IEEE*

***Abstract*—A new method is given for performing approximate maximum-likelihood (ML) decoding of an arbitrary binary linear code based on observations received from any discrete memoryless symmetric channel. The decoding algorithm is based on a linear programming (LP) relaxation that is defined by a factor graph or parity-check representation of the code. The resulting "LP decoder" generalizes our previous work on turbo-like codes.**

**A precise combinatorial characterization of when the LP decoder succeeds is provided, based on *pseudocodewords* associated with the factor graph. Our definition of a pseudocodeword unifies other such notions known for iterative algorithms, including "stopping sets," "irreducible closed walks," "trellis cycles," "deviation sets," and "graph covers."**

**The *fractional distance* $d_{\mathrm{frac}}$ of a code is introduced, which is a lower bound on the classical distance. It is shown that the efficient LP decoder will correct up to $\lceil d_{\mathrm{frac}}/2 \rceil - 1$ errors and that there are codes with $d_{\mathrm{frac}} = \Omega(n^{1-\epsilon})$. An efficient algorithm to compute the fractional distance is presented. Experimental evidence shows a similar performance on low-density parity-check (LDPC) codes between LP decoding and the min-sum and sum-product algorithms. Methods for tightening the LP relaxation to improve performance are also provided.**

***Index Terms*—Belief propagation (BP), iterative decoding, low-density parity-check (LDPC) codes, linear codes, linear programming (LP), LP decoding, minimum distance, pseudocodewords.**

## I. Introduction

LOW-density parity-check (LDPC) codes were first discovered by Gallager in 1962 [7]. In the 1990s, they were "rediscovered" by a number of researchers [8], [4], [9], and have since received a lot of attention. The error-correcting performance of these codes is unsurpassed; in fact, Chung *et al.* [10] have given a family of LDPC codes that come within 0.0045 dB of the capacity of the channel (as the block length goes to infinity). The decoders most often used for this family are based on the *belief-propagation* algorithm [11], where messages are iteratively sent across a *factor graph* modeling the structure of the code. While the performance of this decoder is quite good, analyzing its behavior is often difficult when the factor graph contains cycles.

In this paper, we introduce a new algorithm for decoding an arbitrary binary linear code based on the method of *linear programming* (LP) *relaxation*. We design a polytope that contains all valid codewords, and an objective function for which the maximum-likelihood (ML) codeword is the optimum point with integral coordinates. We use linear programming to find the polytope's (possibly fractional) optimum, and achieve success when that optimum is the transmitted codeword. Experiments on LDPC codes show that the performance of the resulting *LP decoder* is better than the iterative min-sum algorithm. In addition, the LP decoder has the *ML certificate* property; whenever it outputs a codeword, it is guaranteed to be the ML codeword. None of the standard iterative methods are known to have this desirable property.

A desirable feature of the LP decoder is its amenability to analysis. We introduce a variety of techniques for analyzing the performance of this algorithm. We give an exact combinatorial characterization of the conditions for LP decoding success, even in the presence of cycles in the factor graph. This characterization holds for any discrete memoryless symmetric channel; in such channels, a linear cost function can be defined on the code bits such that that the lowest cost codeword is the ML codeword. We define the set of *pseudocodewords*, which is a superset of the set of codewords, and we prove that the LP decoder always finds the lowest cost pseudocodeword. Thus, the LP decoder succeeds if and only if the lowest cost pseudocodeword is actually the transmitted codeword.

Next we define the notion of the *fractional distance* $d_{\mathrm{frac}}$ of a factor graph, which is essentially the minimum distance between a codeword and a pseudocodeword. In analogy to the performance guarantees of exact ML decoding with respect to classical distance, we prove that the LP decoder can correct up to $d_{\mathrm{frac}}/2 - 1$ errors in the binary-symmetric channel (BSC). We prove that the fractional distance of a linear code with check degree at least three is at least exponential in the girth of the graph associated with that code. Thus, given a graph with logarithmic girth, the fractional distance can be lower-bounded by $\Omega(n^{1-\epsilon})$, for some constant $\epsilon$, where $n$ is the code length.

For the case of LDPC codes, we show how to compute the fractional distance efficiently. This fractional distance is not only useful for evaluating the performance of the code under LP decoding, but it also serves as a lower bound on the true distance of the code.

Manuscript received May 6, 2003; revised December 8, 2004. The work of J. Feldman was conducted while the author was at the MIT Laboratory of Computer Science and supported in part by the National Science Foundation Postdoctoral Research Fellowship DMS-0303407. The work of D. Karger was supported in part by the National Science Foundation under Contract CCR-9624239 and a David and Lucille Packard Foundation Fellowship. The material in this paper was presented in part at the Conference on Information Sciences and Systems, Baltimore, MD, June 2003.

J. Feldman is with the Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027 USA (e-mail: jonfeld@ieor.columbia.edu).

M. J. Wainwright is with the Department of Electrical Engineering and Computer Science and the Department of Statistics, University of California, Berkeley, Berkeley, CA, 94720 USA (e-mail: wainwrig@eecs.berkeley.edu).

D. R. Karger is with the Computer Science and Artificial Intelligence Laboratory (CSAIL), Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: karger@mit.edu).

Communicated by R. L. Urbanke, Associate Editor for Coding Techniques.
Digital Object Identifier 10.1109/TIT.2004.842696

### A. Relation to Iterative Algorithms

The techniques used by Chung *et al.* [10] to analyze LDPC codes are based on those of Richardson and Urbanke [12] and Luby *et al.* [13], who give an algorithm to calculate the *threshold* of a randomly constructed LDPC code. This threshold acts as a limit on the channel noise; if the noise is below the threshold, then reliable decoding (using belief propagation (BP)) can be achieved as the block length goes to infinity.

The threshold analysis is based on the idea of considering an "ensemble" of codes for the purposes of analysis, then averaging the behavior of this ensemble as the block length of the code goes to infinity. For many ensembles, it is known [3] that for any constant $\epsilon$, the difference in error rate (under belief-propagation decoding) between a random code and the average code in the ensemble is less than $\epsilon$ with probability exponentially small in the block length.

Calculating the error rate of the ensemble average can become difficult when the factor graph contains cycles; because iterative algorithms can traverse cycles repeatedly, noise in the channel can affect the final decision in complicated, highly dependent ways. This complication is avoided by considering the limiting case of infinite block length, such that the probability of a message traversing a cycle converges to zero. However, for many practical block lengths, this leads to a poor approximation of the true error rate [3].

Therefore, it is valuable to examine the behavior of a code ensemble at fixed block lengths, and try to analyze the effect of cycles. Recently, Di *et al.* [3] took on the "finite length" analysis of LDPC codes under the binary erasure channel (BEC). Key to their results is the notion of a purely combinatorial structure known as a *stopping set*. BP fails if and only if a stopping set exists among the erased bits; therefore, the error rate of BP is reduced to a purely combinatorial question.

For the case of the BEC, we show that the pseudocodewords we define in this paper are exactly *stopping sets*. Thus, the performance of the LP decoder is equivalent to BP on the BEC. Our notion of a pseudocodeword also unifies other known results for particular cases of codes and channels. For tail-biting trellises, our pseudocodewords are equivalent to those introduced by Forney *et al.* [5]. Also, when applied to the analysis of computation trees for min-sum decoding, pseudocodewords have a connection to the *deviation sets* defined by Wiberg [4], and refined by Forney *et al.* [6] and Frey, Koetter, and Vardy [14].

### B. Previous Results

In previous work [1], [2], we introduced the approach of decoding any "turbo-like" code based on similar network flow and linear programming relaxation techniques. We gave a precise combinatorial characterization of the conditions under which this decoder succeeds. We used properties of this LP decoder to design a rate-$1/2$ repeat–accumulate (RA) code (a certain class of simple turbo codes), and proved an upper bound on the probability of decoding error. We also showed how to derive a more classical iterative algorithm whose performance is identical to that of our LP decoder.

### C. Outline

We begin the paper in Section II by giving background on factor graphs for binary linear codes, and the ML decoding problem. We present the LP relaxation of ML decoding in Section III. In Section IV, we discuss the basic analysis of LP decoding. We define pseudocodewords in Section V, and fractional distance in Section VI. In Section VII, we draw connections between various iterative decoding algorithms and our LP decoder, and present some experiments. In Section VIII, we discuss various methods for "tightening" the LP in order to get even better performance. We conclude and discuss future work in Section IX.

### D. Notes and Recent Developments

Preliminary forms of part of the work in this paper have appeared in the conference papers [15], [16], and in the thesis of one of the authors [17]. Since the submission of this work, it has been shown that the LP decoder defined here can correct a constant fraction of error in certain LDPC codes [18], and that a variant of the LP can achieve capacity using expander codes [19].

Additionally, relationships between LP decoding and iterative decoding have been further refined. Discovered independently of this work, Koetter and Vontobel's notion of a "graph cover" [20] is equivalent to the notion of a "pseudocodeword graph" defined here. More recent work by the same authors [21], [22] explores these notions in more detail, and gives new bounds for error performance.

## II. Background

A linear code $\mathcal{C}$ with parity-check matrix $A$ can be represented by a *Tanner* or *factor graph* $G$, which is defined in the following way. Let $\mathcal{I} = \{1, \ldots, n\}$ and $\mathcal{J} = \{1, \ldots, m\}$ be indices for the columns (respectively, rows) of the $n \times m$ parity-check matrix of the code. With this notation, $G$ is a bipartite graph with independent node sets $\mathcal{I}$ and $\mathcal{J}$. We refer to the nodes in $\mathcal{I}$ as *variable nodes*, and the nodes in $\mathcal{J}$ as *check nodes*. All edges in $G$ have one endpoint in $\mathcal{I}$ and the other in $\mathcal{J}$. For each $(i, j) \in \mathcal{I} \times \mathcal{J}$, the edge $(i, j)$ is included in $G$ if and only if $A_{ij} = 1$.

The *neighborhood* of a check node $j \in \mathcal{J}$, denoted by $N(j)$, is the set of nodes $i \in \mathcal{I}$ such that check node $j$ is incident to variable node $i$ in $G$. Similarly, we let $N(i)$ be the set of check nodes incident to a particular variable node $i \in \mathcal{I}$.

Imagine assigning to each variable node $i \in \mathcal{I}$ a value $y_i$ in $\{0, 1\}$, representing the value of a particular code bit. A parity-check node $j$ is "satisfied" if the collection of bits assigned to the variable nodes $i$ s.t. $i \in N(j)$ have even parity. The binary vector $y = (y_1, \ldots, y_n)$ is a codeword if and only if all check nodes are satisfied. Fig. 1 shows an example of a linear code and its associated factor graph. In this Hamming code, if we set $y_1 = y_3 = y_4 = y_7 = 1$, and $y_2 = y_5 = y_6 = 0$, then the neighborhood of every check node has even parity. Therefore, $y$ represents a codeword, which we can write as $1011001$. Other codewords include $1111111$, $0101001$, and $1010110$.

Let $\delta_\ell^+$ denote the maximum variable (left) degree of the factor graph; i.e., the maximum, among all nodes $i \in \mathcal{I}$, of the

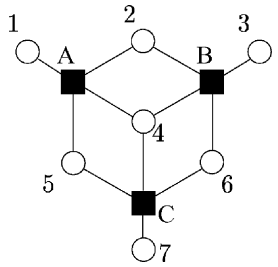


Fig. 1. A factor graph for the $(7, 4, 3)$ Hamming code. The nodes $\{1, 2, 3, 4, 5, 6, 7\}$ drawn in open circles correspond to variable nodes, whereas nodes $\{A, B, C\}$ in black squares correspond to check nodes.

degree of $i$. Let $\delta_\ell^-$ denote the minimum variable degree. Let $\delta_r^+$ and $\delta_r^-$ denote the maximum and minimum check (right) degree of the factor graph.

### A. Channel Assumptions

A binary-codeword $y$ of length $n$ is sent over a noisy channel, and a corrupted word $\hat{y}$ is received. In this paper, we assume an arbitrary discrete memoryless symmetric channel. We use the notation $\Pr[y \mid \hat{y}]$ to denote the probability that $y$ was the codeword sent over the channel, given that $\hat{y}$ was received. We assume that all information words are equally likely *a priori*. By Bayes' rule, this assumption implies that

$$\arg\max_{y \in \mathcal{C}} \Pr[y \mid \hat{y}] = \arg\max_{y \in \mathcal{C}} \Pr[\hat{y} \mid y]$$

for any code $\mathcal{C}$. Moreover, the memoryless property of the channel implies that

$$\Pr[\hat{y} \mid y] = \prod_{i=1}^{n} \Pr[\hat{y}_i \mid y_i].$$

Let $\Sigma$ be the space of possible received symbols. For example, in the BSC, $\Sigma = \{0, 1\}$, and in the additive white Gaussian noise (AWGN) channel, $\Sigma = \mathbb{R}$. By symmetry, the set $\Sigma$ can be partitioned into pairs $(\mathtt{a}, \mathtt{a}')$ such that

$$\Pr[\, \hat{y}_i = \mathtt{a} \mid y_i = 0 \,] = \Pr[\, \hat{y}_i = \mathtt{a}' \mid y_i = 1 \,] \tag{1}$$

and

$$\Pr[\, \hat{y}_i = \mathtt{a} \mid y_i = 1 \,] = \Pr[\, \hat{y}_i = \mathtt{a}' \mid y_i = 0 \,]. \tag{2}$$

### B. ML Decoding

Given the received word $\hat{y}$, the ML decoding problem is to find the codeword $y$ that maximizes $\Pr[y \mid \hat{y}]$. It is equivalent to minimizing the negative log-likelihood, which we will call our *cost function*. Using our assumptions on the channel, this cost function can be written as $\sum_{i=1}^{n} \gamma_i y_i$, where

$$\gamma_i = \log\left(\frac{\Pr[\hat{y}_i \mid y_i = 0]}{\Pr[\hat{y}_i \mid y_i = 1]}\right) \tag{3}$$

is the (known) negative log-likelihood ratio (LLR) at each variable node. For example, given a BSC with crossover probability $p$, we set $\gamma_i = \log[(p)/(1-p)]$ if the received bit $\hat{y}_i = 1$, and $\gamma_i = \log[(1-p)/p]$ if $\hat{y}_i = 0$. The interpretation of $\gamma_i$ is the "cost" of decoding $y_i = 1$. Note that this cost may be negative, if decoding to 1 is the "better choice."

We will frequently exploit the fact that the cost vector $\gamma$ can be uniformly rescaled without affecting the solution of the ML problem. In the BSC, for example, rescaling by $\log[(1-p)/p]$ allows us to assume that $\gamma_i = -1$ if $\hat{y}_i = 1$, and $\gamma_i = +1$ if $\hat{y}_i = 0$.

## III. Decoding With Linear Programming

In this section, we formulate the ML decoding problem for an arbitrary binary linear code, and show that it is equivalent to solving a linear program over the codeword polytope. We then define a modified linear program that represents a *relaxation* of the exact problem.

### A. Codeword Polytope

To motivate our LP relaxation, we first show how ML decoding can formulated as an equivalent LP. For a given code $\mathcal{C}$, we define the *codeword polytope* to be the convex hull of all possible codewords

$$\mathrm{poly}(\mathcal{C}) = \left\{ \sum_{y \in \mathcal{C}} \lambda_y y : \lambda_y \geqslant 0, \sum_{y \in \mathcal{C}} \lambda_y = 1 \right\}.$$

Note that $\mathrm{poly}(\mathcal{C})$ is a polytope contained within the $n$-hypercube $[0, 1]^n$, and includes exactly those vertices of the hypercube corresponding to codewords. Every point in $\mathrm{poly}(\mathcal{C})$ corresponds to a vector $f = (f_1, \ldots, f_n)$, where element $f_i$ is defined by the summation $f_i = \sum_y \lambda_y y_i$.

The *vertices* of a polytope are those points that cannot be expressed as convex combinations of other points in the polytope. A key fact is that any linear program attains its optimum at a vertex of the polytope [23]. Consequently, the optimum will always be attained at a vertex of $\mathrm{poly}(\mathcal{C})$, and these vertices are in one-to-one correspondence with codewords.

We can therefore define ML decoding as the problem of minimizing $\sum_{i=1}^{n} \gamma_i f_i$ subject to the constraint $f \in \mathrm{poly}(\mathcal{C})$. This formulation is a linear program, since it involves minimizing a linear cost function over the polytope $\mathrm{poly}(\mathcal{C})$.

### B. LP Relaxation

The most common practical method for solving a linear program is the simplex algorithm [23], which generally requires an explicit representation of the constraints. In the LP formulation of exact ML decoding we have just described, although $\mathrm{poly}(\mathcal{C})$ can be characterized by a finite number of linear constraints, the number of constraints is exponential in the code length $n$. Even the Ellipsoid algorithm [24], which does not require such an explicit representation, is not useful in this case, since ML decoding is NP-hard in general [25].

Therefore, our strategy will be to formulate a *relaxed* polytope, one that contains all the codewords, but has a more manageable representation. More concretely, we motivate our LP relaxation with the following observation. Each check node in a factor graph defines a *local code*; i.e., the set of binary vectors that have even weight on its neighborhood variables. The global code corresponds to the intersection of all the local codes. In LP

terminology, each check node defines a local codeword polytope (meaning the set of convex combinations of local codewords), and our global relaxation polytope will be the intersection of all of these polytopes.

We use the variables $(f_1, \ldots, f_n)$ to denote our code bits. Naturally, we have

$$\forall i \in \mathcal{I}, \qquad 0 \leqslant f_i \leqslant 1. \tag{4}$$

To define a local codeword polytope, we consider the set of variable nodes $N(j)$ that are neighbors of a given check node $j \in \mathcal{J}$. Of interest are subsets $S \subseteq N(j)$ that contain an even number of variable nodes; each such subset corresponds to a local codeword set, defined by setting $y_i = 1$ for each index $i \in S$, $y_i = 0$ for each $i \in N(j)$ but $i \notin S$, and setting all other $y_i$ arbitrarily.

For each $S$ in the set $E_j = \{S \subseteq N(j) : |S| \text{ even}\}$, we introduce an auxiliary LP variable $w_{j,S}$, which is an indicator for the local codeword set associated with $S$—notionally, setting $w_{j,S}$ equal to 1 indicates that $S$ is the set of bits of $N(j)$ that are set to 1. Note that the variable $w_{j,\emptyset}$ is also present for each parity check, and it represents setting all variables in $N(j)$ equal to zero.

As indicator variables, the variables $\{w_{j,S}\}$ must satisfy the constraints

$$\forall S \in E_j, \qquad 0 \leqslant w_{j,S} \leqslant 1. \tag{5}$$

The variable $w_{j,S}$ can also be seen as indicating that the codeword "satisfies" check $j$ using the configuration $S$. Since each parity check is satisfied with one particular even-sized subset of nodes in its neighborhood set to one, we may enforce

$$\sum_{S \in E_j} w_{j,S} = 1 \tag{6}$$

as a constraint that is satisfied by every codeword. Finally, the indicator $f_i$ at each variable node $i$ must belong to the local codeword polytope associated with check node $j$. This leads to the constraint

$$\forall\, i \in N(j), \qquad f_i = \sum_{\substack{S \in E_j \\ S \ni i}} w_{j,S}. \tag{7}$$

Let the polytope $Q_j$ be the set of points $(f, w)$ such that (4)–(7) hold for check node $j$. Let $Q = \cap_j Q_j$ be the intersection of these polytopes; i.e., the set of points $(f, w)$ such that (4)–(7) hold for *all* $j \in \mathcal{J}$. Overall, the Linear Code Linear Program (LCLP) corresponds to the problem

$$\text{minimize} \sum_{i=1}^{n} \gamma_i f_i \quad \text{s.t.} \quad (f, w) \in Q. \tag{8}$$

An *integral point* in a polytope (also referred to as an *integral solution* to a linear program) is a point in the polytope whose values are all integers. We begin by observing that there is a one-to-one correspondence between codewords and integral solutions to LCLP.

*Proposition 1:* For all integral points $(f, w) \in Q$, the sequence $(f_1, \ldots, f_n)$ represents a codeword. Furthermore, for all codewords $(y_1, \ldots, y_n)$, there exists a $w^y$ such that $(f, w^y)$ is an integral point in $Q$ where $f_i = y_i$ for all $i \in \mathcal{I}$.

*Proof:* Suppose $(f, w)$ is a point in $Q$ where all $f_i \in \{0, 1\}$ and all $w_{j,S} \in \{0, 1\}$. Now suppose $(f_1, \ldots, f_n)$ is *not* a codeword, and let $j$ be some parity check unsatisfied by setting $y_i = f_i$ for all $i \in \mathcal{I}$. By the constraints (6), and the fact that $w$ is integral, $w_{j,S} = 1$ for some $S \in E_j$, and $w_{j,S'} = 0$ for all other $S' \in E_j$ where $S' \neq S$. By the constraints (7), we have $f_i = 1$ for all $i \in S$, and $f_i = 0$ for all $i \in N(j)$, $i \notin S$. Since $|S|$ is even, $j$ is satisfied by setting $y = f$, a contradiction.

For the second part of the claim, let $(y_1, \ldots, y_n)$ be a codeword, and let $f_i = y_i$. For all $j \in \mathcal{J}$, let $S$ be the set of nodes $i$ in $N(j)$ where $f_i = y_i = 1$. Since $(y_1, \ldots, y_n)$ is a codeword, check $j$ is satisfied by $y$, so $|S|$ is even, and the variable $w_{j,S}$ is present. Set $w^y_{j,S} = 1$ and $w^y_{j,S'} = 0$ for all other $S' \in E_j$. All constraints are satisfied, and all variables are integral. □

Overall, the decoding algorithm based on LCLP consists of the following steps. We first solve the LP in (8) to obtain $(f^*, w^*)$. If $f^* \in \{0, 1\}^n$, we output it as the optimal codeword; otherwise, $f^*$ is fractional, and we output an "error." From Proposition 1, we get the following.

*Proposition 2:* LP decoding has the *ML certificate* property: if the algorithm outputs a codeword, it is guaranteed to be the ML codeword.

*Proof:* If the algorithm outputs a codeword $y$, then $(y, w^y)$ has cost less than or equal to all points in $Q$. For some codeword $y' \neq y$, we have that $(y', w^{y'})$ is a point in $Q$ by Proposition 1. Therefore, $y$ has cost less than or equal to $y'$. □

Given a cycle-free factor graph, it can be shown that any optimal solution to LCLP is integral [26]. Therefore, LCLP is an exact formulation of the ML decoding problem in the cycle-free case. In contrast, for a factor graph with cycles, the optimal solution to LCLP may *not* be integral. Take, for example, the Hamming code in Fig. 1. Suppose that we define a cost vector $\gamma$ as follows: for variable node 1, set $\gamma_1 = -7/4$, and for all other nodes $\{2, 3, 4, 5, 6, 7\}$, set $\gamma_i = +1$. It is not hard to verify that under this cost function, all codewords have nonnegative cost: any codeword with negative cost would have to set $y_1 = 1$, and therefore set at least two other $y_i = 1$, for a total cost of at least $+1/4$. Consider, however, the following fractional solution to LCLP: first, set $f = [1, 1/2, 0, 1/2, 0, 0, 1/2]$ and then for check node $A$, set $w_{A,\{1,2\}} = w_{A,\{1,4\}} = 1/2$; at check node $B$, assign $w_{B,\{2,4\}} = w_{B,\emptyset} = 1/2$; and lastly at check node $C$, set $w_{C,\{4,7\}} = w_{C,\emptyset} = 1/2$. It can be verified that $(f, w)$ satisfies all of the LCLP constraints. However, the cost of this solution is $-1/4$, which is strictly less than the cost of any codeword.

Note that this solution is not a convex combination of codewords, and so is not contained in $\text{poly}(\mathcal{C})$. This solution gets outside of $\text{poly}(\mathcal{C})$ by exploiting the local perspective of the relaxation: check node $B$ is satisfied by using the configuration $\{2, 4\}$, whereas in check node $A$, the configuration $\{2, 4\}$ is not used. The analysis to follow will provide further insight into the nature of such fractional (i.e., nonintegral) solutions to LCLP.

It is worthwhile noting that the local codeword constraints (7) are identical to those enforced in the Bethe free energy formulation of BP [27]. For this reason, it is not surprising that the performance of our LP decoder turns out to be closely related to that of the BP and min-sum algorithms.

### C. LP Solving and Polytope Representations

The efficiency of practical LP solvers depends on how the LP is represented. An LP is defined by a set of variables, a cost function, and a polytope (set of linear constraints). The ellipsoid algorithm is guaranteed to run in time polynomial in the size of the LP representation, which is proportional to the number of variables and constraints. The simplex algorithm, though not guaranteed to run efficiently in the worst case, still has a dependence on the representation size, and is usually much more efficient than the ellipsoid algorithm. For more details on solving linear programs, we refer the reader to standard texts [28], [23].

The polytope $Q$ described by (4)–(7) is the most intuitive form of the relaxation. For LDPC codes, $Q$ has size linear in $n$. Thus, the ellipsoid algorithm is provably efficient, and we can also reasonably expect the simplex algorithm to be even more efficient in practice. For arbitrary binary linear codes, the number of constraints in $Q$ is exponential in the degree of each check node. So, if some check node has degree $\Theta(n)$ (as one would expect in random codes, for example), the polytope has a number of constraints that is exponential in $n$. Therefore, to solve $Q$ efficiently, we need to define a smaller polytope that produces the same results.

Alternative representations of the LP are useful for analytical purposes as well. We will see this when we discuss fractional distance in Section VI.

*1) Polytope Equivalence:* All the polytopes we use in this paper have variables $f_i$ for all $i \in \mathcal{I}$. They may also involve auxiliary variables, such as the $\{w_{j,S}\}$ variables in the description of $Q$. However, if two polytopes share the same set of possible settings to the $\{f_i\}$ variables, then we may use either one. We formalize this notion.

*Definition 3:* Let $P$ be some polytope defined over variables $\{f_i\}_{i\in\mathcal{I}}$ where $0 \leqslant f_i \leqslant 1$, as well as some auxiliary variables $x$. We define

$$\overline{P} = \{f : \exists x \text{ s.t. } (f, x) \in P\}$$

as the *projection* of $P$ onto the $\{f_i\}$ variables. Given such a $P$, we say $P$ is *equivalent* to $Q$ if

$$\overline{P} = \overline{Q}.$$

In other words, we require that the projections of $P$ and $Q$ onto the $\{f_i\}$ variables are the same. Since the objective function of LCLP only involves the $\{f_i\}$ variables, optimizing over $P$ and $Q$ will produce the same result.

In the remainder of this section we define two new polytopes. The first is an explicit description of $\overline{Q}$ that will be useful for defining (and computing) the *fractional distance* of the code, which we cover in Section VI. The second polytope is equivalent to $Q$, but has a small overall representation, even for high-density codes. This equivalence shows that LCLP can be solved efficiently for any binary linear code.

*2) Projected Polytope:* In this subsection, we derive an explicit description of the polytope $\overline{Q}$. The following definition of $\overline{Q}$ in terms of constraints on $f$ was derived from the *parity polytope* of Jeroslow [29], [30]. We first enforce $0 \leqslant f_i \leqslant 1$ for all $i \in \mathcal{I}$. Then, for every check $j$, we explicitly forbid every *bad* configuration of the neighborhood of $j$. Specifically, for all $j \in \mathcal{J}$, $S \subseteq N(j)$, $|S|$ *odd*, we require

$$\sum_{i\in S} f_i + \sum_{i\in(N(j)\setminus S)} (1 - f_i) \leqslant |N(j)| - 1. \tag{9}$$

Note that the integral settings of the bits that satisfy these constraints for some check $j$ are exactly the local codewords for $j$, as before.

Let $\Omega_j$ be the set of points $f$ that satisfy (9) for a particular check $j$, and all $S \subseteq N(j)$ with $|S|$ odd. We can further understand the constraints in $\Omega_j$ by rewriting (9) as follows:

$$\sum_{i\in(N(j)\setminus S)} f_i + \sum_{i\in S}(1 - f_i) \geqslant 1. \tag{10}$$

In other words, the $l_1$ distance between (the relevant portion of) $f$ and and the incidence vector for each set $S$ is at least one. This constraint ensures that $f$ is separated by at least one bit flip from all illegal configurations. In three dimensions (i.e, $|N(j)| = 3$), it is easy to see that these constraints are equivalent to the convex hull of the even-sized subsets $S \in E_j$, as shown in Fig. 2. In fact, the following theorem states that in general, if we enforce (9) for all checks, we get an explicit description of $\overline{Q}$.

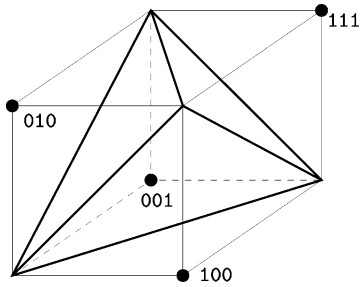


Fig. 2. The equivalence of the polytopes $\Omega_j$ and $\overline{Q}_j$ in three dimensions. The polytope $\Omega_j$ is defined as the set of points inside the unit hypercube with $l_1$ distance at least one from all odd-weight hypercube vertices. The polytope $\overline{Q}_j$ is the convex hull of even-weight hypercube vertices.

*Theorem 4:* Let the polytope $\Omega = \cap_j \Omega_j$. Then $\Omega$ and $Q$ are equivalent. In other words, the polytope

$$\overline{Q} = \{f : \exists w \text{ s.t. } (f, w) \in Q\}$$

is exactly the set of points that satisfy (9) for all checks $j$ and all $S \subseteq N(j)$ where $|S|$ odd.

*Proof:* Recall that $Q_j$ is the set of points $(f, w)$ that satisfy the local codeword polytope for check $j$. Consider the projection

$$\overline{Q}_j = \{f : \exists w \text{ s.t. } (f, w) \in Q_j\}.$$

In other words, $\overline{Q}_j$ is the convex hull of local codeword sets defined by sets $S \in E_j$. Note that $\overline{Q} = \cap_{j\in\mathcal{J}} \overline{Q}_j$, since each $\overline{Q}_j$ exactly expresses the constraints associated with check $j$. Recall that $\Omega_j$ is the set of points $f$ that satisfy the constraints (9) for a particular check $j$. Since $\Omega = \cap_{j\in\mathcal{J}} \Omega_j$, it suffices to show $\Omega_j = \overline{Q}_j$ for all $j \in \mathcal{J}$. This is shown by Jeroslow [29]. For completeness, we include a proof of this fact in Appendix I. □

*3) High-Density Code Polytope:* Recall that $\delta_r^+$ is the maximum degree of any check node in the graph. As stated, LCLP has $O(n+m2^{\delta_r^+})$ variables and constraints. For turbo and LDPC codes, this complexity is linear in $n$, since $\delta_r^+$ is constant.

For arbitrary binary linear codes, we give a characterization of LCLP with

$$O(mn + m(\delta_r^+)^2 + n\delta_\ell^+\delta_r^+) = O(n^3)$$

variables and constraints. To derive this characterization, we give a new polytope for each local codeword polytope, based on a construction of Yannakakis [30], whose size does not have an exponential dependence on the size of the check neighborhood. We refer to this representation of the polytope as $R$. The details of this representation, as well as a proof that $R$ and $Q$ are equivalent, can be found in Appendix II.

## IV. Analysis of LP Decoding

When using the LP decoding method, an error can arise in one of two ways. Either the LP optimum $f^*$ is not integral, in which case the algorithm outputs "error"; or, the LP optimum may be integral (and therefore corresponds to the ML codeword), but the ML codeword is not what was transmitted. In this latter case, the code itself has failed, so even exact ML decoding would make an error.

We use the notation $\Pr[\text{err} \mid y]$ to denote the probability that the LP decoder makes an error, given that $y$ was transmitted. By Proposition 1, there is some feasible solution $(y, w^y)$ to LCLP corresponding to the transmitted codeword $y$. We can characterize the conditions under which LP decoding will succeed as follows.

*Theorem 5:* Suppose the codeword $y$ is transmitted. If all feasible solutions to LCLP other than $(y, w^y)$ have cost more than the cost of $(y, w^y)$, the LCLP decoder succeeds. If some solution to LCLP has cost less than the cost of $(y, w^y)$, the decoder fails.

*Proof:* By Proposition 1, $(y, w^y)$ is a feasible solution to LCLP. If all feasible solutions to LCLP other than $(y, w^y)$ have cost more than the cost of $(y, w^y)$, then $(y, w^y)$ must be the unique optimal solution to LCLP. Therefore, the decoder will output $y$, which is the transmitted codeword.

If some solution $(f, w)$ to LCLP has cost less than the cost of $(y, w^y)$, then $(y, w^y)$ is not an optimal solution to LCLP. Since the $\{w\}$ variables do not affect the cost of the solution, it must be that $f \neq y$. Therefore, the decoder either outputs "error," or it outputs $f$, which is not the transmitted codeword. □

In the degenerate case where $(y, w^y)$ is one of multiple optima of LCLP, the decoder may or may not succeed. We will be conservative and consider this case to be decoding failure, and so by Theorem 5

$$\Pr[\text{err} \mid y] = \Pr\left[\exists (f, w) \in Q, f \neq y : \sum_i \gamma_i f_i \leqslant \sum_i \gamma_i y_i\right]. \quad (11)$$

We now proceed to provide combinatorial characterizations of decoding success and analyze the performance of LP decoding in various settings.

### A. The All-Zeros Assumption

When analyzing linear codes, it is common to assume that the codeword sent over the channel is the all-zeros vector (i.e., $y = 0^n$), since it tends to simplify analysis. In the context of our LP relaxation, however, the validity of this assumption is not immediately clear. In this section, we prove that one *can* make the all-zeros assumption when analyzing LCLP. Basically, this follows from the fact that the polytope $Q$ is highly symmetric; from any codeword, the polytope "looks" exactly the same.

*Theorem 6:* The probability that the LP decoder fails is independent of the codeword that was transmitted.

*Proof:* See Appendix III. □

From this point forward in our analysis of LP decoding, we assume that the all-zeros codeword was the transmitted codeword. Since the all-zeros codeword has zero cost, Theorem 5, along with our consideration of multiple LP optima as "failure," gives the following.

*Corollary 7:* Given that the all-zeros codeword was transmitted (which we may assume by Theorem 6), the LP decoder will fail if and only if there is some point in $Q$ other than $(0^n, w^0)$ with cost less than or equal to zero.

## V. Pseudocodewords

In this section, we introduce the concept of a pseudocodeword for LP decoding, which we will define as a scaled version of a solution to LCLP. As a consequence, Theorem 5 will hold for pseudocodewords in the same way that it holds for solutions to LCLP.

The following definition of a codeword motivates the notion of a pseudocodeword. Recall that $E_j$ is the set of even-sized subsets of the neighborhood of check node $j$. Let $E_j^- = E_j \setminus \{\emptyset\}$. Let $h$ be a vector in $\{0,1\}^n$, and let $u$ be a setting of nonnegative integer weights, one weight $u_{j,S}$ for each check $j$ and $S \in E_j^-$. We say that $(h, u)$ is a codeword if, for all edges $(i, j)$ in the factor graph $G$, $h_i = \sum_{S \in E_j^-, S \ni i} u_{j,S}$. This corresponds exactly to the consistency constraint (7) in LCLP. It is not difficult to see that this construction guarantees that the binary vector $h$ is always a codeword of the original code.

We obtain the definition of a *pseudocodeword* $(h, u)$ by removing the restriction $h_i \in \{0, 1\}$, and instead allowing each $h_i$ to take on arbitrary nonnegative integer values. In other words, a pseudocodeword is a vector $h = (h_1, \ldots, h_n)$ of nonnegative integers such that, for every parity check $j \in \mathcal{J}$, the neighborhood $\{h_i : i \in N(j)\}$ is a sum of local codewords (incidence vectors of even-sized sets in $E_j$).

With this definition, any codeword is (trivially) a pseudocodeword as well; moreover, any sum of codewords is a pseudocodeword. However, in general, there exist pseudocodewords that *cannot* be decomposed into a sum of codewords. As an illustration, consider the Hamming code of Fig. 1; earlier, we constructed a fractional LCLP solution for this code. If we simply scale this fractional solution by a factor of two, the result is a pseudocodeword $(h, u)$ of the following form. We begin by

setting $h = [2,1,0,1,0,0,1]$. To satisfy the constraints of a pseudocodeword, set

$$u_{A,\{1,2\}} = u_{A,\{1,4\}} = u_{B,\{2,4\}} = u_{B,\emptyset} = u_{C,\{4,7\}} = u_{C,\emptyset} = 1.$$

This pseudocodeword cannot be expressed as the sum of individual codewords.

In the following, we use the fact that all optimum points of a linear program with rational coefficients are themselves rational [23]. Using simple scaling arguments, and the all-zeros assumption, we can restate Corollary 7 in terms of pseudocodewords as follows.

*Theorem 8:* Given that the all-zeros codeword was transmitted (which we may assume by Theorem 6), the LP decoder will fail if and only if there is some pseudocodeword $(h,u)$, $h \neq 0^n$, where $\sum_i \gamma_i h_i \leqslant 0$.

*Proof:* Suppose the decoder fails. Let $(f,w)$ be the optimal point of the LP, the point in $Q$ that minimizes $\sum_i \gamma_i f_i$. By Corollary 7, $\sum_i \gamma_i f_i \leqslant 0$. Construct a pseudocodeword $(h,u)$ as follows. Let $\beta$ be the lowest common denominator of the $f_i$ and $w_{j,S}$, which exists because $(f,w)$ is the optimal point of the LP and all optimal points of the LP are rational. Then $\beta \cdot f_i$ is an integer for all $i \in \mathcal{I}$, and $\beta \cdot w_{j,S}$ is an integer for all for all $j \in \mathcal{J}$ and sets $S \in E_j^-$. For all bits $i$, set $h_i = \beta \cdot f_i$; for all checks $j$ and sets $S \in E_j^-$, set $u_{j,S} = \beta \cdot w_{j,S}$.

By the constraints (7) of $Q$, $(h,u)$ meets the definition of a pseudocodeword. The cost of $(h,u)$ is exactly $\beta \cdot \sum_i \gamma_i f_i$. Since $f \neq 0^n$, we have $\beta > 0$. This implies that $h \neq 0^n$. Since $\sum_i \gamma_i f_i \leqslant 0$ and $\beta > 0$, we see that $\sum_i \gamma_i h_i \leqslant 0$.

To establish the converse, suppose a pseudocodeword $(h,u)$, $h \neq 0^n$, has $\sum_i \gamma_i h_i \leqslant 0$. Let $\beta = \max_j(\sum_{S \in E_j} u_{j,S})$. We construct a point $(f,w) \in Q$ as follows: Set $f_i = h_i/\beta$ for all code bits $i$. For all checks $j$, do the following:

i) set $w_{j,S} = u_{j,S}/\beta$ for all sets $S \in E_j^-$;
ii) set $w_{j,\emptyset} = 1 - \sum_{S \in E_j^-} w_{j,S}$.

We must handle $w_{j,\emptyset}$ as a special case since $u_{j,\emptyset}$ does not exist. By construction, and the definition of a pseudocodeword, $(f,w)$ meets all the constraints of the polytope $Q$. Since $h \neq 0^n$, we have $f \neq 0^n$. The cost of $(f,w)$ is exactly $(1/\beta)\sum_i \gamma_i h_i$. Since $\sum_i \gamma_i h_i \leqslant 0$, the point $(f,w)$ has cost less than or equal to zero. Therefore, by Corollary 7, the LP decoder fails. □

This theorem will be essential in proving the equivalence to iterative decoding in the BEC in Section VII.

### A. Pseudocodeword Graphs

A codeword corresponds to a particular subgraph of the factor graph $G$. In particular, the vertex set of this subgraph consists of all the variable nodes $i \in \mathcal{I}$ for which $y_i = 1$, as well as all check nodes to which these variable nodes are incident.

Any pseudocodeword $(h,u)$ can be associated with a graph $H$ in an analogous way. The graph $H$ consists of the following vertices.

- For all $i \in \mathcal{I}$, the graph $H$ contains $h_i$ copies of each node $i$.
- For all $j$, $S \in E_j^-$, the graph $H$ contains $u_{j,S}$ copies of each check node $j$, with "label" $S$.

We refer to the set of copies of the variable node $i$ as

$$Y_i = \{[i,1],[i,2],\ldots,[i,h_i]\}$$

and the copies of the check node $j$ with label $S$ as

$$C_{j,S} = \{[j,S,1],[j,S,2],\ldots,[j,S,u_{j,S}]\}.$$

The edges of the graph are connected according to membership in the sets $S$. More precisely, consider an edge $(i,j)$ in $G$. There are $h_i$ copies of node $i$ in $H$, i.e., $|Y_i| = h_i$. Now consider the set $C^{i,j}$ of nodes in $H$ that are copies of check node $j$ labeled with sets $S$ that include $i$. In other words, the set

$$C^{i,j} = \bigcup_{S \in E_j^-, S \ni i} C_{j,S}.$$

By the definition of a pseudocodeword

$$h_i = \sum_{S \in E_j^-, S \ni i} u_{j,S} = \sum_{S \in E_j^-, S \ni i} |C_{j,S}|$$

and so

$$|C^{i,j}| = h_i = |Y_i|.$$

In the pseudocodeword graph $H$, connect the same-sized node sets $C^{i,j}$ and $Y_i$ using an arbitrary matching (one-to-one correspondence). This process is repeated for every edge $(i,j)$ in $G$. Note that every check node in $C_{j,S}$ appears in exactly $|S|$ sets $C^{i,j}$, one for each $i \in S$. Therefore, the neighbor set of any node in $C_{j,S}$ consists of exactly one copy of each variable node $i \in S$. Furthermore, every variable node in $Y_i$ will be connected to exactly one copy of each check node $j$ in $N(i)$. The cost of the pseudocodeword graph is the sum of the costs $\gamma_i$ of the variable nodes in the graph, and is equal to the cost of the pseudocodeword from which it was derived. Therefore, Theorem 8 holds for pseudocodeword graphs as well.

Fig. 3 gives the graph of the pseudocodeword example given earlier, and Fig. 4 gives the graph of a different more complex pseudocodeword.

This graphical characterization of a pseudocodeword is essential for proving our lower bound on the fractional distance. Additionally, the pseudocodeword graph is helpful in making connections with other notions of pseudocodewords in the literature. We discuss this further in Section VII.

## VI. Fractional Distance

A classical quantity associated with a code is its *distance*, which for a linear code is equal to the minimum weight of any nonzero codeword. In this section, we introduce a fractional analog of distance, and use it to prove additional results on the performance of LP decoding. Roughly speaking, the fractional distance is the minimum weight of any nonzero vertex of $Q$; since all codewords are nonzero vertices of $Q$, the fractional distance is a lower bound on the true distance. This fractional distance has connections to the minimum weight of a pseudocodeword, as defined by Wiberg [4], and also studied by Forney *et al.* [6].

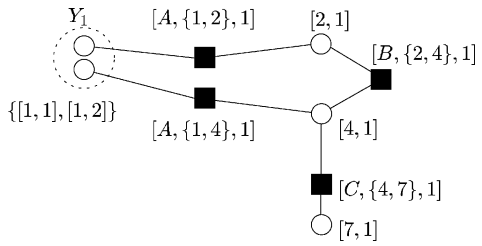


Fig. 3. The graph of a pseudocodeword for the $(7,4,3)$ Hamming code. In this particular pseudocodeword, there are two copies of node 1, and also two copies of check $A$.

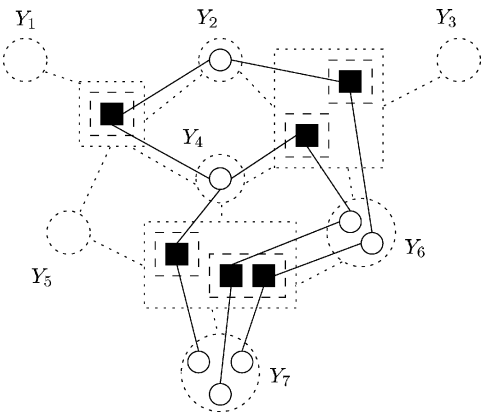


Fig. 4. A graph $H$ of the pseudocodeword $[0,1,0,1,0,2,3]$ of the $(7,4,3)$ Hamming code. The dotted circles show the original variable nodes $i$ of the factor graph $G$, which are now sets $Y_i$ of nodes in $H$. The dotted squares are original check nodes $j$ in $G$, and contain sets $C_{j,S}$ (shown with dashed lines) for each $S \in E_j$.

### A. Definitions and Basic Properties

Since there is a one-to-one correspondence between codewords and integral vertices of $Q$, the (classical) distance of the code is equal to the minimum weight of a nonzero *integral* vertex in the polytope. However, the relaxed polytope $Q$ may have additional *nonintegral* vertices. In particular, our earlier example with the Hamming code involved constructing precisely such a fractional or nonintegral vertex.

As stated previously, any optimal solution $(f^*, w^*)$ to LCLP must be a vertex of $Q$. However, note that the objective function for LCLP only affects the variables $f_1, \ldots, f_n$; as a consequence, the point $f^*$ must *also* be a vertex of the projection $\overline{Q}$. (In general, not all vertices of $Q$ will be projected to vertices of $\overline{Q}$.)

Therefore, we use the projected polytope $\overline{Q}$ in our definition of the fractional distance, since its vertices are exactly the settings of $f$ that could be optimal solutions to LCLP. (Using $Q$ would introduce "false" vertices that would be optimal points only if the problem included costs on the $\{w_{j,s}\}$ variables.)

For a point $f$ in $\overline{Q}$, define the weight of $f$ to be $\sum_i f_i$, and let $\mathcal{V}_{\overline{Q}}^-$ be the set of nonzero vertices of $\overline{Q}$. We define the *fractional distance* of a code to be the minimum weight of any vertex in $\mathcal{V}_{\overline{Q}}^-$. Note that this fractional distance is always a lower bound on the classical distance of the code, since every nonzero codeword is contained in $\mathcal{V}_{\overline{Q}}^-$. Moreover, the performance of LP decoding is tied to this fractional distance, as we make precise in the following.

*Theorem 9:* For a code $G$ with fractional distance $d_{\text{frac}}$, the LP decoder is successful if at most $\lceil d_{\text{frac}}/2 \rceil - 1$ bits are flipped by the binary symmetric channel.

*Proof:* Suppose the LP decoder fails; i.e., the optimal solution $(f^*, w^*)$ to LCLP has $f^* \neq 0^n$. We know that $f^*$ must be a vertex of $\overline{Q}$. Since $f^* \neq 0^n$, we have $f^* \in \mathcal{V}_{\overline{Q}}^-$. This implies that $\sum_i f_i^* \geqslant d_{\text{frac}}$, since the fractional distance is at least $d_{\text{frac}}$.

Let $\mathcal{E} = \{i : \hat{y}_i \neq y_i\}$ be the set of bits flipped by the channel. Under the BSC, and the all-zeros assumption, we have $\gamma_i = -1$ if $i \in \mathcal{E}$, and $\gamma_i = +1$ if $i \notin \mathcal{E}$. Therefore, we can write the cost of $f^*$ as the following:

$$\sum_i \gamma_i f_i^* = \sum_{i \notin \mathcal{E}} f_i^* - \sum_{i \in \mathcal{E}} f_i^*. \tag{12}$$

Since at most $\lceil d_{\text{frac}}/2 \rceil - 1$ bits are flipped by the channel, we have that $|\mathcal{E}| \leqslant \lceil d_{\text{frac}}/2 \rceil - 1$, and so

$$\sum_{i \in \mathcal{E}} f_i^* \leqslant \lceil d_{\text{frac}}/2 \rceil - 1.$$

It follows that

$$\sum_{i \notin \mathcal{E}} f_i^* \geqslant \lfloor d_{\text{frac}}/2 \rfloor + 1$$

since $\sum_i f_i^* \geqslant d_{\text{frac}}$. Therefore, by (12), we have $\sum_i \gamma_i f_i^* > 0$. However, by Theorem 5 and the fact that the decoder failed, the optimal solution $(f^*, w^*)$ to LCLP must have cost less than or equal to zero; i.e., $\sum_i \gamma_i f_i \leqslant 0$. This is a contradiction. □

Note again the analogy to the classical case: just as exact ML decoding has a performance guarantee in terms of classical distance, Theorem 9 establishes that the LP decoder has a performance guarantee specified by the fractional distance of the code.

### B. Computing the Fractional Distance

In contrast to the classical distance, the fractional distance of an LDPC code can be computed efficiently. Since the fractional distance is a lower bound on the real distance, we thus have an efficient algorithm to give a nontrivial lower bound on the distance of an LDPC code.

To compute the fractional distance, we must compute the minimum-weight vertex in $\mathcal{V}_{\overline{Q}}^-$. We first consider a more general problem: given the $m$ facets of a polytope $P$ over vertices $(x_1, \ldots, x_n)$, a specified vertex $x^0$ of $P$, and a linear function $\ell(x)$, find the vertex $x^1$ in $P$ *other than* $x^0$ that minimizes $\ell(x)$. An efficient algorithm for this problem is the following: let $\mathcal{F}$ be the set of all facets of $P$ on which $x^0$ does not sit. Now for each facet in $\mathcal{F}$, intersect $P$ with the facet to obtain $P'$, and then optimize $\ell(x)$ over $P'$. The minimum value obtained over all facets in $\mathcal{F}$ is the minimum of $\ell(x)$ over all vertices $x^1$ other than $x^0$. The running time of this algorithm is equal to the time taken by $|\mathcal{F}| < m$ calls to an LP solver.

This algorithm is correct by the following argument. It is well known [23] that a vertex of a polytope of dimension $D$ is uniquely determined by giving $D$ linearly independent facets of the polytope on which the vertex sits. Using this fact, it is clear that the vertex $x^1$ we are looking for must sit on some facet in $\mathcal{F}$; otherwise, it would be the same point as $x^0$. Therefore, at some point in our procedure, each potential $x^1$ is considered. Furthermore, when we intersect $P$ with a facet of $P$ to obtain

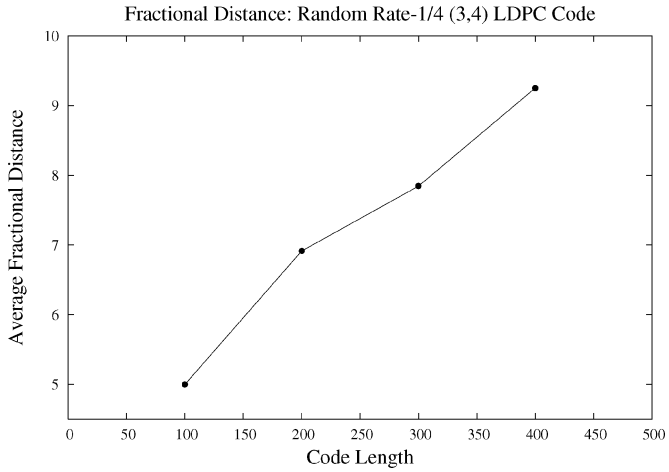


Fig. 5. The average fractional distance $d_{\text{frac}}$ as a function of length for a randomly generated LDPC code, with left degree 3, right degree 4, from an ensemble of Gallager [7].

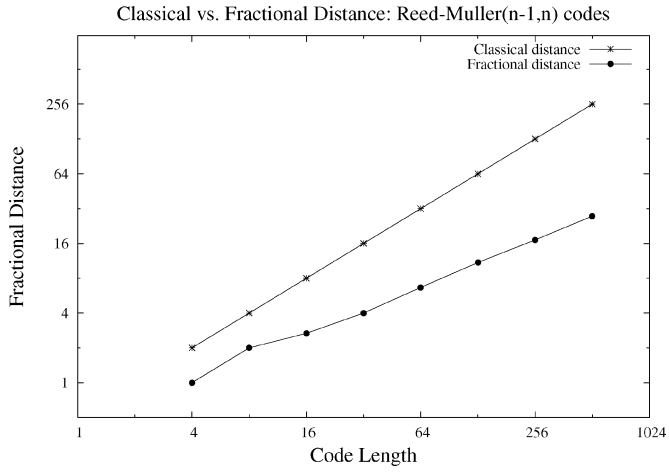


Fig. 6. The classical versus fractional distance of the "normal realizations" of the Reed–Muller$(n-1, n)$ codes [31]. The classical distance of these codes is exactly $n/2$. The upper part of the fractional distance curve follows roughly $\frac{5}{14}n^{0.7}$.

$P'$ we have that all vertices of $P'$ are vertices of $P$ not equal to $x^0$. Therefore, the true $x^1$ will obtain the minimum value over all facets in $\mathcal{F}$.

For our problem, we are interested in the polytope $\overline{Q}$, and the special vertex $0^n \in \overline{Q}$. In order to run the above procedure, we use the small explicit representation of $\overline{Q}$ given by (9) and Theorem 4. The number of facets in this representation of $\overline{Q}$ has an exponential dependence on the check degree of the code. For an LDPC code, the number of facets will be linear in $n$, so that we can compute the exact fractional distance efficiently. For arbitrary linear codes, we can still compute the minimum-weight nonzero vertex of $R$ (from Section III-C), which provides a (possibly weaker) lower bound on the fractional distance. However, this representation (given explicitly in Appendix II) introduces many auxiliary variables, and therefore may have many "false" vertices with low weight.

### C. Experiments

Fig. 5 gives the average fractional distance of a randomly chosen LDPC factor graph, computed using the algorithm we just described. The graph has left degree 3, right degree 4, and is randomly chosen from an ensemble of Gallager [7]. This data is insufficient to extrapolate the growth rate of the fractional distance; however, it certainly grows nontrivially with the block length. We had conjectured that this growth rate could be made linear in the block length [17]; for the case of graphs with regular degree, this conjecture has since been disproved by Koetter and Vontobel [20].

Fig. 6 gives the fractional distance of the "normal realizations" of the Reed–Muller$(n-1, n)$ codes [31].[1] These codes, well defined for lengths $n$ equal to a power of 2, have a classical distance of exactly $n/2$. The curve in the figure suggests that the fractional distance of these graphs is roughly $\frac{5}{14}n^{0.7}$. Note that for both these code families, there may be alternate realizations (factor graphs) with better fractional distance.

[1]We thank G. David Forney for suggesting the study of the normal realizations of the Reed–Muller codes.

### D. The Max-Fractional Distance

In this subsection, we define another notion of fractional distance, which we call the *max-fractional distance*. This is simply the fractional distance, normalized by the maximum $f_i$ value. We can also show that the LP decoder corrects up to half the max-fractional distance. Furthermore, we prove in the next section that the max-fractional distance grows exponentially in the girth of $G$.

We define the *max-fractional distance* $d_{\text{frac}}^{\max}$ of the code using polytope $Q$ as

$$d_{\text{frac}}^{\max} = \min_{\substack{(f,w)\in\mathcal{V}_Q^- \\ f\neq 0^n}} \left(\frac{\sum_i f_i}{\max_i f_i}\right).$$

Using essentially the same proof as for Theorem 9, we obtain the following.

*Theorem 10:* For a code $G$ with max-fractional distance $d_{\text{frac}}^{\max}$, the LP decoder is successful if at most $\lceil d_{\text{frac}}^{\max}/2 \rceil - 1$ bits are flipped by the binary-symmetric channel.

The exact relationship between $d_{\text{frac}}$ and $d_{\text{frac}}^{\max}$ is an interesting question. Clearly, $d_{\text{frac}}^{\max} \geqslant d_{\text{frac}}$ in general, since $\max_i f_i$ is always at most 1. In fact, we know that $d_{\text{frac}}^{\max} \leqslant d_{\text{frac}}(\delta_r^+/2)$, which follows from the fact that for all $f \in \mathcal{V}_Q^-$, there is some $i$ for which $f_i \geqslant 2/\delta_r^+$. (The proof of this fact comes from simple scaling arguments.) Therefore, for LDPC codes, the two quantities are the same up to a constant factor. We can compute the max-fractional distance efficiently using an algorithm similar to the one used for the fractional distance: we reduce the problem of finding the point with the minimum $d_{\text{frac}}^{\max}$ to finding the minimum-weight point in a polytope.

### E. A Lower Bound Using the Girth

The following theorem asserts that the max-fractional distance is exponential in the girth of $G$. It is analogous to an earlier result of Tanner [32], which provides a similar bound on the classical distance of a code in terms of the girth of the associated factor graph.

*Theorem 11:* Let $G$ be a factor graph with $\delta_\ell^- \geqslant 3$ and $\delta_r^- \geqslant 2$. Let $g$ be the girth of $G$, $g > 4$. Then the max-fractional distance is at least $d_{\text{frac}}^{\max} \geqslant (\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}$.

This theorem is proved in Appendix IV, and makes heavy use of the combinatorial properties of pseudocodewords. One consequence of Theorem 11 is that the max-fractional distance is at least $\Omega(n^{1-\epsilon})$ for some constant $\epsilon$ (where $0 < \epsilon < 1$), for any graph $G$ with girth $\Omega(\log n)$. Note that there are many known constructions of such graphs (e.g., [33]). Although Theorem 11 does not yield a bound on the word error rate (WER) for the BSC, it demonstrates that LP decoding can always correct $\Omega(n^{1-\epsilon})$ errors, for any code defined by a graph with logarithmic girth.

## VII. Comparison to Iterative Decoding

In this section, we draw several connections between LP decoding and iterative decoding for several code types and channel models. We show that many known combinatorial characterizations of decoding success are in fact special cases of our definition of a pseudocodeword. We discuss stopping sets in the BEC, cycle codes, tail-biting trellises, the tree-reweighted max-product algorithm of Wainwright *et al.* [26], and min-sum decoding.

At the end of the section, we give some experimental results comparing LP decoding with the min-sum and sum-product (BP) algorithms.

### A. Stopping Sets in the BEC

In the BEC, bits are not flipped but rather erased. Consequently, for each bit, the decoder receives either 0, 1, or an erasure. If either symbol 0 or 1 is received, then it must be correct. On the other hand, if an erasure (which we denote by $*$) is received, there is no information about that bit. It is well known [3] that in the BEC, the iterative BP decoder fails if and only if a "stopping set" exists among the erased bits. The main result of this section is that stopping sets are the special case of pseudocodewords on the BEC, and so LP decoding exhibits the same property.

We can model the BEC in LCLP with our cost function $\gamma$. As in the BSC, $\gamma_i = -1$ if the received bit $\hat{y}_i = 1$, and $\gamma_i = +1$ if $\hat{y}_i = 0$. If $\hat{y}_i = *$, we set $\gamma_i = 0$, since we have no information about that bit. Note that under the all-zeros assumption, all the costs are nonnegative, since no bits are flipped. Therefore, Theorem 8 implies that the LP decoder will fail only if there is a nonzero pseudocodeword with zero cost.

Let $\mathcal{E}$ be the set of code bits erased by the channel. A subset $\mathcal{S} \subseteq \mathcal{E}$ is a *stopping set* if all the checks in the neighborhood $\cup_{i \in \mathcal{S}} N(i)$ of $\mathcal{S}$ have degree at least two with respect to $\mathcal{S}$. In the following statement, we have assumed that both the iterative and the LCLP decoders fail when the answer is ambiguous. For the iterative algorithm, this ambiguity corresponds to the existence of a stopping set; for the LCLP algorithm, it corresponds to a nonzero pseudocodeword with zero cost, and hence multiple optima for the LP.

*Theorem 12:* Under the BEC, there is a nonzero pseudocodeword with zero cost if and only if there is a stopping set. Therefore, the performance of LP and BP decoding are equivalent for the BEC.

*Proof:* If there is a zero-cost pseudocodeword, then there is a stopping set. Let $(h, u)$ be a pseudocodeword where $\sum_i \gamma_i h_i = 0$. Let $\mathcal{S} = \{i : h_i > 0\}$. Since all $\gamma_i \geqslant 0$, we must have $\gamma_i = 0$ for all $i \in \mathcal{S}$; therefore, $\mathcal{S} \subseteq \mathcal{E}$.

Suppose $\mathcal{S}$ is not a stopping set; then $\exists j' \in (\cup_{i \in \mathcal{S}} N(i))$ where check node $j'$ has only one neighbor $i'$ in $\mathcal{S}$. By the definition of a pseudocodeword, we have $h_{i'} = \sum_{S \in E_{j'}, S \ni i'} u_{j',S}$. Since $h_{i'} > 0$ (by the definition of $\mathcal{S}$), there must be some $S' \in E_{j'}$, $S' \ni i'$ such that $u_{j',S'} > 0$. Since $S'$ has even cardinality, there must be at least one other code bit $i''$ in $S'$, which is also a neighbor of check $j'$. We have $h_{i''} \geqslant u_{j',S'}$ by the definition of pseudocodeword, and so $h_{i''} > 0$, implying $i'' \in \mathcal{S}$. This contradicts the fact that $j'$ has only one neighbor in $\mathcal{S}$.

If there is a stopping set, then there is a zero-cost pseudocodeword. Let $\mathcal{S}$ be a stopping set. Construct pseudocodeword $(h, u)$ as follows. For all $i \in \mathcal{S}$, set $h_i = 2$; for all $i \notin \mathcal{S}$, set $h_i = 0$. Since $\mathcal{S} \subseteq \mathcal{E}$, we immediately have $\gamma h = 0$.

For a check $j$, let $M(j) = N(j) \cap \mathcal{S}$. For all $j \in (\cup_{i \in \mathcal{S}} N(i))$, where $|M(j)|$ even, set $u_{j,M(j)} = 2$. By the definition of a stopping set, $M(j) \geqslant 2$, so if $|M(j)|$ is odd, then $M(j) \geqslant 3$. For all $j \in (\cup_{i \in \mathcal{S}} N(i))$, where $|M(j)|$ odd, let $I = \{i_1, i_2, i_3\}$ be an arbitrary size-3 subset of $M(j)$. If $|M(j)| > 3$, set $u_{j,M(j)\setminus I} = 2$. Set $u_{j,\{i_1,i_2\}} = u_{j,\{i_1,i_3\}} = u_{j,\{i_2,i_3\}} = 1$. Set all other $u_{j,S} = 0$ that we have not set in this process. We have

$$\sum_{S \in E_j, S \ni i} u_{j,S} = 2 = h_i, \qquad \text{for all } i \in \mathcal{S},\ j \in N(i).$$

Additionally

$$\sum_{S \in E_j, S \ni i} u_{j,S} = 0 = h_i, \qquad \text{for all } i \notin \mathcal{S}, j \in N(i).$$

Therefore, $(h, u)$ is a pseudocodeword. □

### B. Cycle Codes

A cycle code is a binary linear code described by a factor graph whose variable nodes all have degree 2. In this case, pseudocodewords consist of a collection of cycle-like structures we call *promenades* [1]. This structure is a closed walk through the graph that is allowed to repeat nodes, and even traverse edges in different directions, as long as it makes no "U-turns;" i.e., it does not use the same edge twice in a row. Wiberg [4] calls these same structures irreducible closed walks. We may conclude from this connection that iterative and LP decoding have identical performance in the case of cycle codes.

We note that even though cycle codes are poor in general, they are an excellent example of when LP decoding can decode beyond the minimum distance. For cycle codes, the minimum distance is no better than logarithmic. However, we showed [1] that there are cycle codes for which LP decoding has a WER of $n^{-\alpha}$ for any $\alpha > 0$, requiring only that the crossover probability is bounded by a certain function of the constant $\alpha$ (independent of $n$).

### C. Tail-Biting Trellises

On tail-biting trellises, one can write down a linear program similar to the one we explored for turbo codes [1] such that pseudocodewords in this LP correspond to those analyzed by Forney *et al.* [5]. This linear program is, in fact, an instance of network flow, and therefore is solvable by a more efficient algorithm than a generic LP solver. (See [17] for a general treatment of LPs for trellis-based codes, including turbo-like codes.)

In this case, pseudocodewords correspond to cycles in a directed graph (a circular trellis). All cycles in this graph have length $\alpha n$ for some integer $\alpha \geqslant 1$. Codewords are simple cycles of length exactly $n$. Forney *et al.* [5] show that iterative decoding will find the pseudocodeword with minimum weight-per-symbol. Using basic network flow theory, it can be shown that the weight-per-symbol of a pseudocodeword is the same as the cost of the corresponding LP solution. Thus, these two algorithms have identical performance.

We note that to get this connection to tail-biting trellises, if the code has a factor graph representation, it is *not* sufficient simply to write down the factor graph for a code and plug in the polytope $Q$. This would be a weaker relaxation in general. One has to define a new linear program like the one we used for turbo-like codes [1]. With this setup, the problem reduces directly to min-cost flow.

### D. Tree-Reweighted Max-Product

In earlier work [2], we explored the connection between this LP-based approach applied to turbo codes, and the tree-reweighted max-product message-passing algorithm developed by Wainwright, Jaakkola, and Willsky [26]. Similar to the usual max-product (min-sum) algorithm, the algorithm is based on passing messages between nodes in the factor graph. It differs from the usual updates in that the messages are suitably reweighted according the structure of the factor graph. By drawing a connection to the dual of our linear program, we showed that whenever this algorithm converges to a codeword, it must be the ML codeword. Note that the usual min-sum algorithm does not have such a guarantee.

### E. Min-Sum Decoding

The *deviation sets* defined by Wiberg [4], and further refined by Forney *et al.* [6] can be compared to pseudocodeword graphs. The *computation tree* of the iterative min-sum algorithm is a map of the computations that lead to the decoding of a single bit at the root of the tree. This bit will be decoded correctly (assuming the all-zeros word is sent) unless there is a negative-cost locally consistent minimal configuration of the tree that sets this bit to 1. Such a configuration is called a *deviation set*, or a pseudocodeword.

All deviation sets have a *support*, which is the set of nodes in the configuration that are set to 1. All such supports $D$ are acyclic graphs of the following form. The nodes of $D$ are nodes from the factor graph, possibly with multiple copies of a node. Furthermore,

- all the leaves of $D$ are variable nodes;
- each nonleaf variable node $i \in \mathcal{I}$ is connected to one copy of each check node in $N(i)$; and
- each check node has even degree.

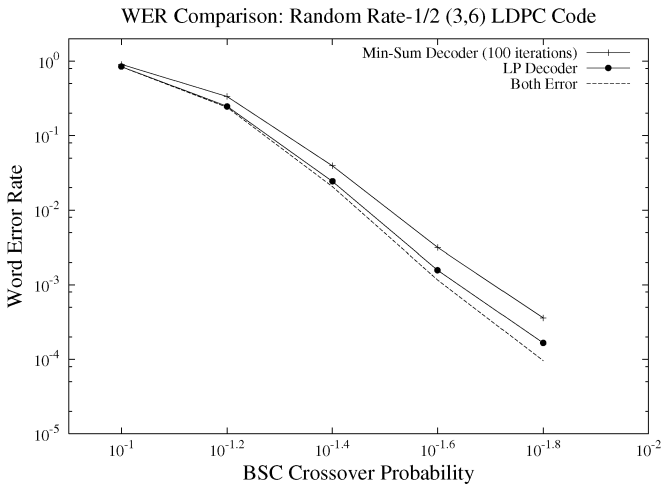


Fig. 7. A waterfall-region comparison between the performance of LP decoding and min-sum decoding (with 100 iterations) under the BSC using the same random rate-$1/2$ LDPC code with length $200$, left degree $3$, and right degree $6$. For each trial, both decoders were tested with the same channel output. The "Both Error" curve represents the trials where both decoders failed.

As is clear from the definition, deviation sets are quite similar to pseudocodeword graphs; essentially the only difference is that deviation sets are acyclic. In fact, if we removed the "nonleaf" condition above, the two would be equivalent. In his thesis, Wiberg states:

> *Since the (graph) is finite, an infinite deviation cannot behave completely irregularly; it must repeat itself somehow. … It appears natural to look for repeatable, or 'closed' structures (in the graph)…, with the property that any deviation can be decomposed into such structures.* [4]

Our definition of a pseudocodeword is the natural "closed" structure within a deviation set. However, an arbitrary deviation set *cannot* be decomposed into pseudocodewords, since it may be irregular near the leaves. Furthermore, as Wiberg points out, the cost of a deviation set is dominated by the cost near the leaves, since the number of nodes grows exponentially with the depth of the tree.

Thus, strictly speaking, min-sum decoding and LP decoding are incomparable. However, experiments suggest that it is rare for min-sum decoding to succeed and LP decoding to fail (see Fig. 7). We also conclude from our experiments that the irregular "unclosed" portions of the min-sum computation tree are not worth considering; they more often hurt the decoder than help it.

### F. New Iterative Algorithms and ML Certificates From the LP Dual

In earlier work [2], we described how the iterative subgradient ascent [34] algorithm can be used to solve the LP dual for RA codes. Thus, we have an iterative decoder whose error-correcting performance is identical to that of LP decoding in this case. This technique may also be applied in the general setting of LDPC codes [17]; thus, we have an iterative algorithm for any LDPC code with all the performance guarantees of LP decoding.

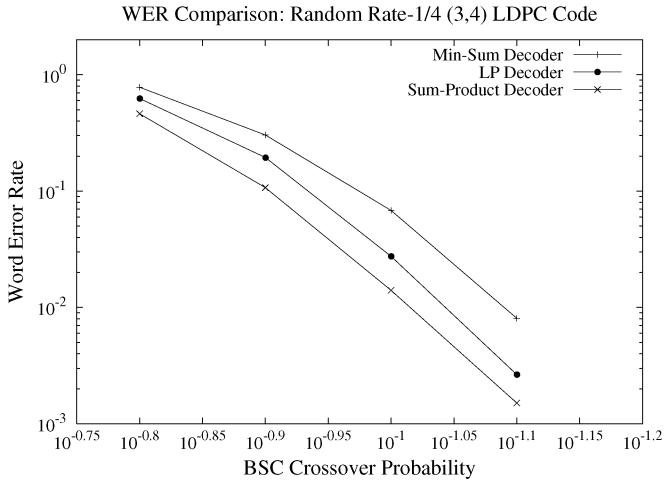


Fig. 8. A comparison between the performance of LP decoding, min-sum decoding (100 iterations) and BP (100 iterations) under the BSC using the same random rate-$1/4$ LDPC code with length 200, left degree 3, and right degree 4.

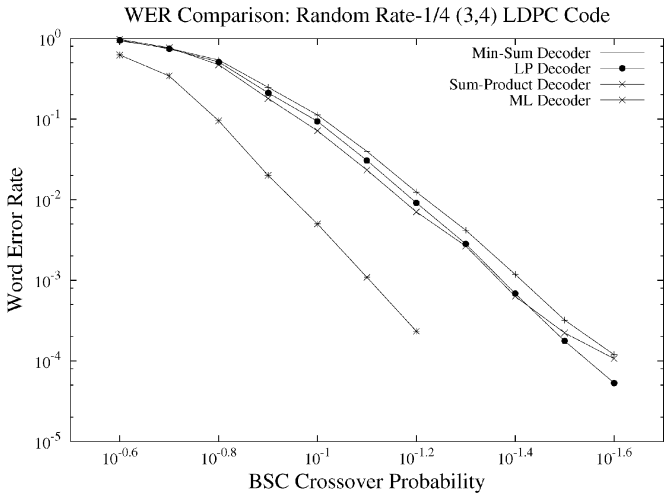


Fig. 9. A comparison between the performance of ML decoding, LP decoding, min-sum decoding (100 iterations), and BP (100 iterations) under the BSC using the same random rate-$1/4$ LDPC code with length 60, left degree 3, and right degree 4. The ML decoder is a mixed-integer programming decoder using the LP relaxation.

Furthermore, we show [17] that LP duality can be used to give *any* iterative algorithm the ML certificate property; that is, we derive conditions under which the output of a message-passing decoder is provably the ML codeword.

### G. Experimental Comparison

We have compared the performance of the LP decoder with the min-sum and sum-product decoders on the BSC. We used a randomly generated rate-$1/4$ LDPC code with left degree 3, and right degree 4. Fig. 8 shows an error-rate comparison in the waterfall region for a block length of 200. We see that LP decoding performs better than min-sum in this region, but not as well as sum-product.

However, when we compare all three algorithms to ML decoding, it seems that at least on random codes, all three have similar performance. This is shown in Fig. 9. In fact, we see that LP decoding slightly outperforms sum-product at very low noise levels. It would be interesting to see whether this is a general phenomenon, and whether it can be explained analytically.

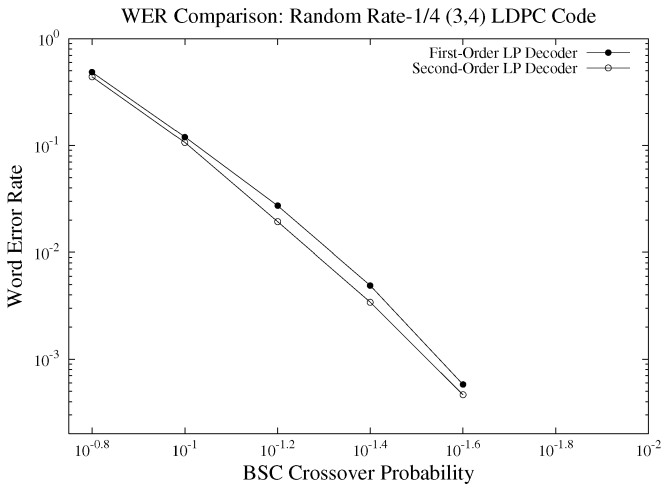


Fig. 10. Error-correcting performance gained by adding a set of (redundant) parity checks to the factor graph. The code is a randomly selected regular LDPC code, with length 40, left degree 3, and right degree 4, from an ensemble of Gallager [7]. The "First-Order Decoder" is the LP decoder using the polytope $Q$ defined on the original factor graph. The "Second-Order Decoder" uses the polytope $Q$ defined on the factor graph after adding a set of redundant parity checks; the set consists of all checks that are the sum $(\mathrm{mod} 2)$ of two original parity checks.

## VIII. Tighter Relaxations

It is important to observe that LCLP has been defined with respect to a specific factor graph. Since a given code has many such representations, there are many possible LP-based relaxations, and some may be better than others. Of particular significance is the fact that adding redundant parity checks to the factor graph, though it does not affect the code, provides new constraints for the LP relaxation, and will in general strengthen it. For example, returning to the $(7, 4, 3)$ Hamming code of Fig. 1, suppose we add a new check node whose neighborhood is $\{1, 3, 5, 6\}$. This parity check is redundant for the code, since it is simply the mod two sum of checks $A$ and $B$. However, the linear constraints added by this check tighten the relaxation; in fact, they render our example pseudocodeword $f = [1, 1/2, 0, 1/2, 0, 0, 1/2]$ infeasible. Whereas redundant constraints may degrade the performance of BP decoding (due to the creation of small cycles), adding new constraints can only improve LP performance.

As an example, Fig. 10 shows the performance improvement achieved by adding all "second-order" parity checks to a factor graph $G$. By second-order we mean all parity checks that are the sum of two original parity checks. A natural question is whether adding *all* redundant parity checks results in the codeword polytope $\mathrm{poly}(\mathcal{C})$. This turns out not to be the case; the dual of the $(7, 4, 3)$ Hamming code provides one counterexample.

In addition to redundant parity checks, there are various generic ways in which an LP relaxation can be strengthened (e.g., [35], [36]). Such "lifting" techniques provide a nested sequence of relaxations increasing in both tightness and complexity, the last of which is equivalent to $\mathrm{poly}(\mathcal{C})$ (albeit with exponential complexity). Therefore, we obtain a sequence of decoders, increasing in both performance and complexity, the last of which is an (intractable) ML decoder. It would be interesting to analyze the rate of performance improvement along

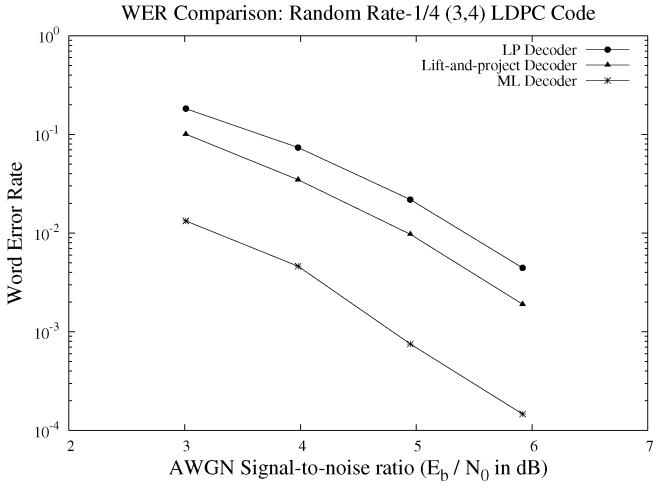


Fig. 11. The WER of the lift-and-project relaxation compared with LP decoding and ML decoding.

this sequence. Fig. 11 shows the performance gained by one application of the "lift-and-project" [35] method on a random $(3,4)$ LDPC code of length 36. Another interesting question is how complex a decoder is needed in order to surpass the performance of BP.

Finally, the fractional distance of a code, as defined here, is also a function of the factor graph representation of the code. Fractional distance yields a lower bound on the true distance, and the quality of this bound could also be improved by adding redundant constraints, or other methods of tightening the LP.

### A. ML Decoding Using Integer Programming

Another interesting application of LP decoding is to use the polytope $Q$ to perform true ML decoding. An *integer program* (IP) is an optimization problem that allows integer constraints; that is, we may force variables to be integers. If we add the constraint $f_i \in \{0,1\}$ to our linear program, then we get an exact formulation of ML decoding. In general, integer programming is NP-hard, but there are various methods for solving an IP that far outperform the naive exhaustive search routines for ML decoding. Using the program CPLEX [37], we were able to perform ML decoding on LDPC codes with moderate block lengths (up to about 100) in a "reasonable" amount of time. Fig. 9 includes an error curve for ML decoding an LDPC code with a block length of 60. Each trial took no more than a few seconds (and often much less) on a Pentium IV (2-GHz) processor.

Drawing this curve allows us to see the gap between various suboptimal algorithms and the optimal ML decoder. This gap further motivates the search for tighter LP relaxations to approach ML decoding. The running time of this decoder becomes prohibitive at longer block lengths; however, ML decoding at small block lengths can be very useful in evaluating algorithms, and determining whether decoding errors are the fault of the decoder or the code.

## IX. Discussion

We have described an LP-based decoding method, and proved a number of results on its error-correcting performance. Central to this characterization is the notion of a *pseudocodeword*, which corresponds to a rescaled solution of the LP relaxation. Our definition of pseudocodeword unifies previous work on iterative decoding (e.g., [5], [6], [4], [3]). We also introduced the *fractional distance* of a code, a quantity which shares the worst case error-correcting guarantees with the classical notion, but with an efficient algorithm to realize those guarantees.

There are a number of open questions and future directions suggested by the work presented in this paper, some of which we detail here. In an earlier version of this work [15], we had suggested using graph expansion to improve the performance bounds given here. This has been accomplished [18] to some degree, and we now know that LP decoding can correct a constant fraction of error. However, there is still work to be done in order to improve the constant to the level of the best known (e.g., on the Zyablov bound or beyond). We also know that LP decoding can achieve the capacity of many commonly considered channels [19]. It would be interesting to see if these methods can be extended to achieve capacity without an exponential dependence on the gap to capacity (all known capacity-achieving decoders also have this dependence).

Since turbo codes and "turbo-like" codes have much more efficient encoders than LDPC codes, it would be interesting to see if LP decoding can be used to obtain good performance bounds in this setting as well. In previous work on RA codes [1], we were able to prove a bound on the error rate of LP decoding stronger than that implied by the minimum distance. Analogous LP decoders for general "turbo-like" codes have also been given [17], but it remains to provide satisfying analysis of their performance.

## Appendix I
## Proving Theorem 4

Recall that $\Omega_j$ is the set of points $f$ such that $0 \leqslant f_i \leqslant 1$ for all $i \in \mathcal{I}$, and for all $S \subseteq N(j)$, $|S|$ *odd*

$$\sum_{i \in S} f_i + \sum_{i \in (N(j) \setminus S)} (1 - f_i) \leqslant |N(j)| - 1. \tag{13}$$

To prove Theorem 4, it remains to show the following.

*Theorem 13:* [29] The polytope

$$\Omega_j = \overline{Q}_j = \{F : \exists w \text{ s.t. } (f, w) \in Q_j\}.$$

*Proof:* For all $i \notin N(j)$, the variable $f_i$ is unconstrained in both $\Omega_j$ and $\overline{Q}_j$ (aside from the constraints $0 \leqslant f_i \leqslant 1$); thus, we may ignore those indices, and assume that $N(j) = \mathcal{I}$. (We henceforth use $n$ to denote $|N(j)|$.)

Let $f$ be a point in $\overline{Q}_j$. By the constraints (6) and (7), $\overline{Q}_j$ is the convex hull of the incidence vectors of even-sized sets $S \in E_j$. Since all such vectors satisfy the constraints (13) for check node $j$, then $f$ must also satisfy these constraints. Therefore, $f \in \Omega_j$.

For the other direction, suppose some point $f' \in \Omega_j$ is not contained in $\overline{Q}_j$. Then some facet $F$ of $\overline{Q}_j$ cuts $f'$ (makes it infeasible). Since $0 \leqslant f'_i \leqslant 1$ for all $i \in \mathcal{I}$, it must be the case that $F$ passes through the hypercube $[0,1]^n$, and so it must cut off some vertex of the hypercube; i.e., some $x \in \{0,1\}^n$. Since all incidence vectors of even-sized sets are feasible for $\overline{Q}_j$, the vertex $x$ must be the incidence vector for some odd-sized set $S \notin E_j$.

For a particular $f \in [0,1]^n$, let $[f]_i = 1 - f_i$ if $x_i = 1$, and $[f]_i = f_i$ if $x_i = 0$. We specify the facet $F$ in terms of the variables $[f]$, using the equation

$$F: \quad a_1[f]_1 + a_2[f]_2 + \cdots + a_n[f]_n \geqslant b.$$

Since $x$ is infeasible for $F$, it must be that $\sum_i a_i[x]_i < b$. Since $[x]_i = 0$ for all $i \in \mathcal{I}$, we have $\sum_i a_i[x]_i = 0$, so we may conclude that $b > 0$.

For some $i' \in \mathcal{I}$, let $x^{\oplus i'}$ denote vector $x$ with bit $i'$ flipped; i.e., $x_{i'}^{\oplus i'} = 1 - x_{i'}$ and $x_i^{\oplus i'} = x_i$ for all $i \neq i'$. Since $x$ has odd parity, we have that for all $i'$, $x^{\oplus i'}$ has even parity, so $x^{\oplus i'} \in \overline{Q}_j$, and $x^{\oplus i'}$ is not cut by $F$. This implies that for all $i' \in \mathcal{I}$

$$\sum_i a_i[x^{\oplus i'}]_i \geqslant b.$$

Note that $[x^{\oplus i'}]_i = [x]_i = 0$ for all $i \neq i'$, and

$$[x^{\oplus i'}]_{i'} = 1 - [x]_{i'} = 1.$$

So, we may conclude $a_{i'} \geqslant b > 0$, for all $i' \in \mathcal{I}$.

Except for the case $n = 2$, the polytope $\overline{Q}_j$ is full-dimensional. For example, the set of points with exactly two (cyclically) consecutive 1's is a full-dimensional set. (For a full proof, see [29]). Therefore, $F$ must pass through $n$ vertices of $\overline{Q}_j$; i.e., it must pass through at least $n$ even-parity binary vectors. This claim is still true for the case $n = 2$, since both faces in this case pass through both points.

We claim that those $n$ vectors must be the points $\{x^{\oplus i} : i \in \mathcal{I}\}$. Suppose this is not the case. Then some vertex $x'$ of $\overline{Q}_j$ is on the facet $F$, and differs from $x$ in more than one place. Suppose without loss of generality (w.l.o.g.) that $x'_1 \neq x_1$ and $x'_2 \neq x_2$, and so $[x']_1 = 1$, $[x']_2 = 1$. Since $x'$ is on $F$, we have $\sum_i a_i[x']_i = b$. Therefore,

$$a_1 + a_2 + \sum_{i=3}^{n} a_i[x']_i = b.$$

Since $a_i > 0 \ \forall i$, we have $\sum_{i=3}^{n} a_i[x']_i \geqslant 0$, and so

$$a_1 + a_2 \leqslant b.$$

This contradicts the fact that $a_1, a_2 \geqslant b > 0$.

Thus, $F$ passes through the vertices $\{x^{\oplus i} : i \in \mathcal{I}\}$. It is not hard to see that $F$ is exactly the odd-set constraint (13) corresponding to the set $S$ for which $x$ is the incidence vector. Since $F$ cuts $f'$, and $F$ is a facet of $\Omega_j$, we have $f' \notin \Omega_j$, a contradiction. □

## APPENDIX II
## HIGH-DENSITY POLYTOPE REPRESENTATION

In this appendix, we give a polytope $R$ of for use in LCLP with $O(mn + m(\delta_r^+)^2 + n\delta_\ell^+\delta_r^+) = O(n^3)$ variables and constraints. This polytope provides an efficient way to perform LP decoding for any binary linear code. This polytope was derived from the "parity polytope" of Yannakakis [30].

For a check node $j \in \mathcal{J}$, let $T_j = \{0, 2, 4, \ldots, 2\lfloor |N(j)|/2 \rfloor\}$ be the set of even numbers between 0 and $|N(j)|$. Our new representation has three sets of variables.

- For all $i \in \mathcal{I}$, we have a variable $f_i$, where $0 \leqslant f_i \leqslant 1$. This variable represents the code bit $y_i$, as before.
- For all $j \in \mathcal{J}$ and $k \in T_j$, we have a variable $\alpha_{j,k}$, $0 \leqslant \alpha_{j,k} \leqslant 1$. This variable indicates the contribution of weight-$k$ local codewords.
- For all $j \in \mathcal{J}$, $k \in T_j$, and $i \in N(j)$, we have a variable $z_{i,j,k}$, $0 \leqslant z_{i,j,k} \leqslant \alpha_{j,k}$, indicating the portion of $f_i$ locally assigned to local codewords of weight $k$.

Using these variables, we have the following constraint set:

$$\forall i \in \mathcal{I}, j \in N(i), \qquad f_i = \sum_{k \in T_j} z_{i,j,k} \quad (14)$$

$$\forall j \in \mathcal{J}, \qquad \sum_{k \in T_j} \alpha_{j,k} = 1 \quad (15)$$

$$\forall j \in \mathcal{J}, k \in T_j, \qquad \sum_{i \in N(j)} z_{i,j,k} = k \cdot \alpha_{j,k} \quad (16)$$

$$\forall i \in \mathcal{I}, \qquad 0 \leqslant f_i \leqslant 1 \quad (17)$$

$$\forall j \in \mathcal{J}, k \in T_j, \qquad 0 \leqslant \alpha_{j,k} \leqslant 1 \quad (18)$$

$$\forall i \in \mathcal{I}, j \in N(i), k \in T_j, \qquad 0 \leqslant z_{i,j,k} \leqslant \alpha_{j,k}. \quad (19)$$

Let $R$ be the set of points $(f, \alpha, z)$ such that the above constraints hold. This polytope $R$ has only $O((\delta_r^+)^2)$ variables per check node $j$, plus the $\{f\}$ variables, for a total of $O(n + m(\delta_r^+)^2)$ variables. The number of constraints is at most $O(mn + n\delta_\ell^+\delta_r^+)$. In total, this representation has at most $O(n^3)$ variables and constraints. We must now show that optimizing over $R$ is equivalent to optimizing over $Q$. Since the cost function only affects the $\{f\}$ variables, it suffices to show that the two polytopes have the same projection onto the $\{f\}$ variables. Before proving this, we need the following fact.

*Lemma 14:* Let $X = \{x_1, \ldots, x_N\}$, $x_i \leqslant M$, and $\sum_i x_i = kM$, where $k$, $N$, $M$, and all $x_i$ are nonnegative integers. Then, $X$ can be expressed as the sum of sets of size $k$. Specifically, there exists a setting of the variables

$$\{w_S : S \subseteq \{1, \ldots, N\}, |S| = k\}$$

to nonnegative integers such that $\sum_S w_S = M$, and for all $i \in \{1, \ldots, N\}$, $x_i = \sum_{S \ni i} w_S$.

*Proof:* By induction on $M$.[2] The base case $(M = 1)$ is simple; all $x_i$ are equal to either 0 or 1, and so exactly $k$ of them are equal to 1. Set $w_{\{i:x_i=1\}} = 1$.

For the induction step, assume w.l.o.g. that $x_1 \geqslant x_2 \geqslant \cdots \geqslant x_N$. Set $X' = (x'_1, \ldots, x'_N)$, where $x'_i = x_i - 1$ if $i \leqslant k$, and $x'_i = x_i$ otherwise. The fact that $\sum_i x_i = kM$ and $x_i \leqslant M$ for all $i$ implies that $x_i \geqslant 1$ for all $i \leqslant k$, and $x_i \leqslant M-1$ for all $i > k$. Therefore, $0 \leqslant x'_i \leqslant M-1$ for all $i$. We also have

$$\sum_i x'_i = \sum_i x_i - k = (M-1)k.$$

Therefore, by induction, $X'$ can be expressed as the sum of $w'_S$, where $S$ has size $k$. Set $w = w'$, then increase $w_{\{1,\ldots,k\}}$ by 1. This setting of $w$ expresses $X$. □

*Proposition 15:* The set $\{f : \exists \alpha, z, \text{ s.t. } (f, \alpha, z) \in R\}$ is equal to the set $\overline{Q} = \{f : \exists w, \text{ s.t. } (f, w) \in Q\}$. Therefore, optimizing over $R$ is equivalent to optimizing over $Q$.

*Proof:* Suppose $(f, w) \in Q$. Set

$$\alpha_{j,k} = \sum_{\substack{S \in E_j, \\ |S| = k}} w_{j,S} \qquad (\forall j \in \mathcal{J}, k \in T_j) \quad (20)$$

[2]We thank Ryan O'Donnell for showing us this proof.

$$z_{i,j,k} = \sum_{\substack{S \in E_j, \\ |S|=k, \\ S \ni i}} w_{j,S} \quad (\forall i \in \mathcal{I}, j \in N(i), k \in T_j). \quad (21)$$

It is clear that the constraints (17)–(19) are satisfied by this setting. Constraint (14) is implied by (7) and (21). Constraint (15) is implied by (6) and (20). Finally, we have, for all $\forall j \in \mathcal{J}$, $k \in T_j$

$$\begin{aligned} \sum_{i \in N(j)} z_{i,j,k} &= \sum_{i \in N(j)} \sum_{\substack{S \in E_j, \\ |S|=k, \\ S \ni i}} w_{j,S} && \text{(by (21))} \\ &= \sum_{\substack{S \in E_j, \\ |S|=k}} |S| w_{j,S} = k \sum_{\substack{S \in E_j, \\ |S|=k}} w_{j,S} \\ &= k \cdot \alpha_{j,k} && \text{(by (20))} \end{aligned}$$

giving constraint (16).

Now suppose $(f, \alpha, z)$ is a vertex of the polytope $R$, and so all variables are rational. For all $j \in \mathcal{J}, k \in T_j$, consider the set

$$X_1 = \left\{ \frac{z_{i,j,k}}{\alpha_{j,k}} : i \in N(j) \right\}.$$

By (19), all members of $X_1$ are between 0 and 1. Let $1/\beta$ be a common divisor of the numbers in $X_1$ such that $\beta$ is an integer. Let

$$X_2 = \left\{ \beta \frac{z_{i,j,k}}{\alpha_{j,k}} : i \in N(j) \right\}.$$

The set $X_2$ consists of integers between 0 and $\beta$. By (16), we have that the sum of the elements in $X_2$ is equal to $k\beta$. So, by Lemma 14, the set $X_2$ can be expressed as the sum of sets $S$ of size $k$. Set the variables $\{w_S : S \in N(j), |S| = k\}$ according to Lemma 14. Now set $w_{j,S} = \frac{\alpha_{j,k}}{\beta} w_S$, for all $S \in N(j), |S| = k$. We immediately satisfy (5). By Lemma 14 we get

$$z_{i,j,k} = \sum_{\substack{S \in E_j, \\ S \ni i, \\ |S|=k}} w_{j,S} \quad (22)$$

and

$$\alpha_{j,k} = \sum_{\substack{S \in E_j, \\ |S|=k}} w_{j,S}. \quad (23)$$

By (14), we have

$$\begin{aligned} f_i = \sum_{k \in T_j} z_{i,j,k} &= \sum_{k \in T_j} \sum_{\substack{S \in E_j, \\ S \ni i, \\ |S|=k}} w_{j,S} && \text{(by (22))} \\ &= \sum_{\substack{S \in E_j, \\ S \ni i}} w_{j,S} \end{aligned}$$

giving (7). By (15), we have

$$\begin{aligned} 1 = \sum_{k \in T_j} \alpha_{j,k} &= \sum_{k \in T_j} \sum_{\substack{S \in E_j, \\ |S|=k}} w_{j,S} && \text{(by (23))} \\ &= \sum_{S \in E_j} w_{j,S} \end{aligned}$$

giving (6). Since this construction works for all vertices of $R$, the projection of any point in $R$ onto the $\{f\}$ variables must be in $\overline{Q} = \{f : \exists w, \text{ s.t. } (f, w) \in Q\}$. □

## Appendix III
## Proving Theorem 6

In this appendix, we show that the all-zeros assumption is valid when analyzing LP decoders defined on factor graphs. Specifically, we prove the following theorem.

*Theorem 6:* The probability that the LP decoder fails is independent of the codeword that was transmitted.

*Proof:* Recall that $\Pr[\text{err} \mid y]$ is the probability that the LP decoder makes an error, given that $y$ was transmitted. For an arbitrary transmitted word $y$, we need to show that

$$\Pr[\text{err} \mid y] = \Pr[\text{err} \mid 0^n].$$

Define $\mathrm{B}(y) \subseteq \Sigma^n$ to be the set of received words $\hat{y}$ that would cause decoding failure, assuming $y$ was transmitted. By Theorem 5

$$\mathrm{B}(y) = \left\{ \hat{y} : \exists (f, w) \in Q, f \neq y : \sum_i \gamma_i f_i \leqslant \sum_i \gamma_i y_i \right\}.$$

Note that in the above, the cost vector $\gamma$ is a function of the received word $\hat{y}$. Rewriting (11), we have for all codewords $y$

$$\Pr[\text{err} \mid y] = \sum_{\substack{\hat{y} \in \Sigma^n, \\ \hat{y} \in \mathrm{B}(y)}} \Pr[\hat{y} \mid y]. \quad (24)$$

Applying this to the codeword $0^n$, we get

$$\Pr[\text{err} \mid 0^n] = \sum_{\substack{\hat{y} \in \Sigma^n, \\ \hat{y} \in \mathrm{B}(0^n)}} \Pr[\hat{y} \mid 0^n]. \quad (25)$$

We will show that the space $\Sigma^n$ of possible received vectors can be partitioned into pairs $(\hat{y}, \hat{y}^0)$ such that

$$\Pr[\hat{y} \mid y] = \Pr[\hat{y}^0 \mid 0^n]$$

and $\hat{y} \in \mathrm{B}(y)$ if and only if $\hat{y}^0 \in \mathrm{B}(0^n)$. This, along with (24) and (25), gives

$$\Pr[\text{err} \mid y] = \Pr[\text{err} \mid 0^n].$$

The partition is performed according to the symmetry of the channel. Fix some received vector $\hat{y}$. Define $\hat{y}^0$ as follows: let $\hat{y}^0_i = \hat{y}_i$ if $y_i = 0$, and $\hat{y}^0_i = \hat{y}'_i$ if $y = 1$, where $\hat{y}'_i$ is the symmetric symbol to $\hat{y}_i$ in the channel. Note that this operation is its own inverse and therefore gives a valid partition of $\Sigma^n$ into pairs.

First, we show that $\Pr[\hat{y} \mid y] = \Pr[\hat{y}^0 \mid 0^n]$. From the channel being memoryless, we have

$$\begin{aligned} \Pr[\hat{y} \mid y] &= \prod_{i=1}^{n} \Pr[\hat{y}_i \mid y_i] \\ &= \prod_{i:y_i=0} \Pr[\hat{y}_i \mid 0] \prod_{i:y_i=1} \Pr[\hat{y}_i \mid 1] \\ &= \prod_{i:y_i=0} \Pr[\hat{y}^0_i \mid 0] \prod_{i:y_i=1} \Pr[\hat{y}_i \mid 1] \quad (26) \\ &= \prod_{i:y_i=0} \Pr[\hat{y}^0_i \mid 0] \prod_{i:y_i=1} \Pr[\hat{y}'_i \mid 0] \quad (27) \\ &= \prod_{i:y_i=0} \Pr[\hat{y}^0_i \mid 0] \prod_{i:y_i=1} \Pr[\hat{y}^0_i \mid 0] \\ &= \Pr[\hat{y}^0 \mid 0^n]. \quad (28) \end{aligned}$$

Equations (26) and (28) follow from the definition of $\hat{y}^0$, and (27) follows from the symmetry of the channel.

Now it remains to show that $\hat{y} \in \mathrm{B}(y)$ if and only if $\hat{y}^0 \in \mathrm{B}(0^n)$. Let $\gamma$ be the cost vector when $\hat{y}$ is received, and let $\gamma^0$ be the cost vector when $\hat{y}^0$ is received, as defined in (3).

Suppose $y_i = 0$. Then, $\hat{y}_i = \hat{y}_i^0$, and so $\gamma_i = \gamma_i^0$. Now suppose $y_i = 1$; then $\hat{y}_i^0 = \hat{y}_i'$, and so

$$\begin{aligned}\gamma_i^0 &= \log\left(\frac{\Pr[\hat{y}_i' \mid y_i = 0]}{\Pr[\hat{y}_i' \mid y_i = 1]}\right) \\ &= \log\left(\frac{\Pr[\hat{y}_i \mid y_i = 1]}{\Pr[\hat{y}_i \mid y_i = 0]}\right) \\ &= -\gamma_i. \end{aligned} \tag{29}$$

Equation (29) follows from the symmetry of the channel. We conclude that

$$\gamma_i = \gamma_i^0 \text{ if } y_i = 0 \quad \text{and} \quad \gamma_i = -\gamma_i^0, \qquad \text{if } y_i = 1. \tag{30}$$

The following lemma shows a correspondence between the points of $Q$ under cost function $\gamma$, and the points of $Q$ under cost function $\gamma^0$.

*Lemma 16:* Fix some codeword $y$. For every $(f, w) \in Q$, $f \neq y$, there is some $(f^r, w^r) \in Q$, $f^r \neq 0^n$, such that

$$\sum_i \gamma_i f_i - \sum_i \gamma_i y_i = \sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n.$$

Furthermore, for every $(f^r, w^r) \in Q$, $f^r \neq 0^n$, there is some $(f, w) \in Q$, $f \neq y$, such that

$$\sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n = \sum_i \gamma_i f_i - \sum_i \gamma_i y_i.$$

We prove this lemma later in this section. Now suppose $\hat{y} \in \mathrm{B}(y)$, and so by the definition of B there is some $(f, w) \in Q$, $f \neq y$ where

$$\sum_i \gamma_i f_i - \sum_i \gamma_i y_i \leqslant 0.$$

By Lemma 16, there is some $(f^r, w^r) \in Q$, $f^r \neq 0^n$, such that

$$\sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n \leqslant 0.$$

Therefore, $\hat{y}^0 \in \mathrm{B}(0^n)$. A symmetric argument (using the other half of the lemma) shows that if $\hat{y}^0 \in \mathrm{B}(0^n)$ then $\hat{y} \in \mathrm{B}(y)$. □

Before proving Lemma 16, we need to define the notion of a *relative solution* in $Q$, and prove results about its feasibility and cost. For two sets $A$ and $B$, let $A \oplus B$ denote the symmetric difference of $A$ and $B$, i.e., $A \oplus B = (A \cup B) \setminus (A \cap B)$. Let $(y, w^y)$ be the point in LCLP corresponding to the codeword sent over the channel. For a particular feasible solution $(f, w)$ to LCLP, set $(f^r, w^r)$ to be the relative solution with respect to $(y, w^y)$ as follows: For all bits $i$, set $f_i^r = |f_i - y_i|$. For all checks $j$, let $S_j^*$ be the member of $E_j$ where $(w_{j,S}^y = 1)$. For all $S \in E_j$, set $w_{j,S \oplus S_j^*}^r = w_{j,S}$.

Note that for a fixed $(y, w^y)$, the operation of making a relative solution is its own inverse; i.e., the relative solution to $(f^r, w^r)$ is $(f, w)$.

*Lemma 17:* For a feasible solution $(f, w)$ to LCLP, the relative solution $(f^r, w^r)$ with respect to $(y, w^y)$ is also a feasible solution to LCLP.

*Proof:* First consider the bounds on the variables (see (4) and (5)). These are satisfied by definition of $(f^r, w^r)$, and the fact that both $(f, w)$ and $(y, w^y)$ are feasible. Now consider the distribution constraints (6). From the feasibility of $(f, w)$ and the definition of $w^r$ we have, for all checks $j$

$$1 = \sum_{S \in E_j} w_{j,S} = \sum_{S \in E_j} w_{j,S \oplus S_j^*}^r \tag{31}$$

where $S_j^*$ is defined as in the definition of $w^r$. Note that

$$\{S : S \in E_j\} = \{S \oplus S_j^* : S \in E_j\}$$

so we get $\sum_{S \in E_j} w_{j,S}^r = 1$, satisfying the distribution constraints.

It remains to show that $(f^r, w^r)$ satisfies the consistency constraints (7). In the following, we assume that sets $S$ are contained within the appropriate set $E_j$, which will be clear from context. For all edges $(i, j)$ in $G$, we have

$$f_i^r = |f_i - y_i| = \left|\sum_{S \ni i} w_{j,S} - y_i\right|. \tag{32}$$

- Case 1: $y_i = 0$. In this case, from (32) we have

$$f_i^r = \sum_{S \ni i} w_{j,S} = \sum_{S \ni i} w_{j,S \oplus S_j^*}^r = \sum_{S \ni i} w_{j,S}^r.$$

The last step follows from the fact that

$$\{S : S \ni i\} = \{S \oplus S_j^* : S \ni i\}$$

as long as $S_j^* \not\ni i$.

- Case 2: $y_i = 1$. From (32), we have

$$\begin{aligned} f_i^r &= \left|\left(\sum_{S \ni i} w_{j,S}\right) - 1\right| = 1 - \sum_{S \ni i} w_{j,S \oplus S_j^*}^r \quad \text{(by (31))} \\ &= 1 - \sum_{S \not\ni i} w_{j,S}^r . \end{aligned} \tag{33}$$

The last step follows from the fact that

$$\{S : S \not\ni i\} = \{S \oplus S_j^* : S \ni i\}$$

as long as $S_j^* \ni i$. Finally, from the distribution constraints on $w^r$, we get $\sum_S w_{j,S}^r = 1$, and therefore, by (33),

$$f_i^r = \sum_{S \ni i} w_{j,S}^r.$$ □

*Lemma 18:* Given a point $(f, w) \in Q$, and its relative solution $(f^r, w^r)$, we have

$$\sum_i \gamma_i f_i - \sum_i \gamma_i y_i = \sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n.$$

*Proof:* From the definition of $f^r$, we have

$$\begin{aligned} \sum_i \gamma_i^0 f_i^r &= \sum_i \gamma_i^0 |f_i - y_i| \\ &= \sum_{i:y_i=1} \gamma_i^0 (1 - f_i) + \sum_{i:y_i=0} \gamma_i^0 f_i. \end{aligned}$$

Using (30), we get

$$\begin{aligned} &= \sum_{i:y_i=1} \gamma_i(f_i - 1) + \sum_{i:y_i=0} \gamma_i f_i \\ &= \sum_{i:y_i=1} \gamma_i f_i + \sum_{i:y_i=0} \gamma_i f_i - \sum_{i:y_i=1} \gamma_i \\ &= \sum_i \gamma_i f_i - \sum_i \gamma_i y_i. \end{aligned}$$

The lemma follows from the fact that $\sum_i \gamma_i^0 0_i^n = 0$. □

We are now ready to prove Lemma 16, which completes the proof of Theorem 6. We restate the lemma here for reference.

*Lemma 16:* Fix some codeword $y$. For every $(f, w) \in Q$, $f \neq y$, there is some $(f^r, w^r) \in Q$, $f^r \neq 0^n$, such that

$$\sum_i \gamma_i f_i - \sum_i \gamma_i y_i = \sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n.$$

Furthermore, for every $(f^r, w^r) \in Q$, $f^r \neq 0^n$, there is some $(f, w) \in Q$, $f \neq y$, such that

$$\sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n = \sum_i \gamma_i f_i - \sum_i \gamma_i y_i.$$

*Proof:* Consider some $(f, w) \in Q$, and let $(f^r, w^r)$ be the relative solution with respect to $(y, w^y)$. By Lemma 17, $(f^r, w^r) \in Q$. By definition, $f^r \neq 0^n$, since $f \neq y$. By Lemma 18

$$\sum_i \gamma_i f_i - \sum_i \gamma_i y_i = \sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n.$$

For the second part of the lemma, consider some $(f^r, w^r) \in Q$, and let $(f, w)$ be the relative solution with respect to $y$. By Lemma 17, $(f, w) \in Q$. By definition, $f \neq y$, since $f^r \neq 0^n$.

Because the operation of making a relative solution is its own inverse, the relative solution to $(f, w)$ is $(f^r, w^r)$. Therefore, by Lemma 18

$$\sum_i \gamma_i^0 f_i^r - \sum_i \gamma_i^0 0_i^n = \sum_i \gamma_i f_i - \sum_i \gamma_i y_i.$$ □

## Appendix IV
## Proving Theorem 11

Before proving Theorem 11, we will prove a few useful facts about pseudocodewords and pseudocodeword graphs. For all the theorems in this section, let $G$ be a factor graph with all variable nodes having degree at least $\delta_\ell^-$, where $\delta_\ell^- \geqslant 3$ and all check nodes having degree at least $\delta_r^-$, where $\delta_r^- \geqslant 2$. Let $g$ be the girth of $G$, $g > 4$. Let $H$ be the graph of some arbitrary pseudocodeword $(h, u)$ of $G$, $h \neq 0^n$.

We define a *promenade* $\Phi$ to be a path $\Phi = (\phi_1, \phi_2, \ldots, \phi_{|\Phi|})$ in $H$ that may repeat nodes and edges, but takes no U-turns; i.e., for all $i$, $0 \leqslant i \leqslant |\Phi| - 2$, $\phi_i \neq \phi_{i+2}$. We will also use $\Phi$ to represent the set of nodes on the path $\Phi$ (the particular use will be clear from context). Note that each $\phi_i$ could be a variable or a check node. These paths are similar to the notion of promenade in [1], and to the irreducible closed walk of Wiberg [4]. A *simple* path of a graph is one that does not repeat nodes.

Recall that $Y_i$ and $C_{j,S}$ represent the sets of variable and check nodes in $H$ that are copies of the same node in the underlying factor graph $G$. For a variable node $\phi \in H$, let $G(\phi)$ be the corresponding node in $G$; i.e., $(G(\phi) = i : \phi \in Y_i)$ if $\phi$ is a variable node, and $(G(\phi) = j : \phi \in C_{j,S}$ for some $S \in E_j)$ if $\phi$ is a check node.

*Claim 19:* For all promenades $\Phi$ of length less than the girth of $G$, $\Phi$ is a simple path in $H$, and also represents a simple path in $G$. More precisely, for all promenades

$$\Phi = (\phi_1, \phi_2, \ldots, \phi_{|\Phi|}), \qquad |\Phi| < g$$

$\Phi$ is a simple path in $H$, and that

$$G(\Phi) = (G(\phi_1), \ldots, G(\phi_{|\Phi|}))$$

is a simple path in $G$.

*Proof:* First note that $G(\Phi)$ is a valid path. By construction, if there is an edge $(\phi_i, \phi_{i+1})$ in $H$, there must be an edge $(G(\phi_i), G(\phi_{i+1}))$ in $G$. If $G(\Phi)$ is simple, then $\Phi$ must be simple, so we only need to show that $G(\Phi)$ is simple. This is true since the length of $G(\Phi)$ is less than the girth of the graph. □

For the remainder of this appendix, suppose w.l.o.g. that $h_1 = \max_i h_i$. Thus, $|Y_1| = h_1$. Note that $g$ is even, since $G$ is bipartite. For all $i \in \{1, \ldots, h_1\}$, let $\mathcal{T}_i$ be the set of nodes in $H$ within distance $(g/2) - 1$ of $[1, i]$; i.e., $\mathcal{T}_i$ is the set of nodes with a path in $H$ of length at most $(g/2) - 1$ from $[1, i]$.

*Claim 20:* The subgraph induced by the node set $\mathcal{T}_i$ is a tree.

*Proof:* Suppose this is not the case. Then, for some node in $\phi \in H$, there are at least two different paths from $[1, i]$ to $\phi$, each with length at most $(g/2) - 1$. This implies a cycle in $H$ of length less than $g$; a contradiction to Claim 19. □

*Claim 21:* The node subsets $(\mathcal{T}_1, \ldots, \mathcal{T}_{h_1})$ in $H$ are all mutually disjoint.

*Proof:* Suppose this is not the case; then, for some $i \neq i'$, $\mathcal{T}_i$ and $\mathcal{T}_{i'}$ share at least one vertex. Let $\phi$ be the vertex in $\mathcal{T}_i$ closest to the root $[1, i]$ that also appears in $\mathcal{T}_{i'}$. Now consider the promenade $\Phi = ([1, i], \ldots, \phi', \phi, \phi'', \ldots, [1, i'])$, where the subpath from $[1, i]$ to $\phi$ is the unique such path in the tree $\mathcal{T}_i$, and the subpath from $\phi$ to $[1, i']$ is the unique such path in the tree $\mathcal{T}_{i'}$. We must show that $\Phi$ has no U-turns. The subpaths $([1, i], \ldots, \phi', \phi)$ and $(\phi, \phi'', \ldots, [1, i'])$ are simple, so we must show only that $\phi' \neq \phi''$. Since we chose $\phi$ to be the node closest to $[1, i]$ that appears in $\mathcal{T}_{i'}$, $\phi'$ must not appear in $\mathcal{T}_{i'}$, and so $\phi' \neq \phi''$. Since $|\Phi| < g$, $G(\Phi)$ must be simple path by Claim 19. However, it is not, since node 1 appears twice in $G(\Phi)$, once at the beginning and once at the end. This is a contradiction. □

*Claim 22:* The number of variable nodes in $H$ is at least $h_1(\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}$.

*Proof:* Take one node set $\mathcal{T}_i$. We will count the number of nodes on each "level" of the tree induced by $\mathcal{T}_i$. Each level $\ell$ consists of all the nodes at distance $\ell$ from $[1, i]$. Note that even levels contain variable nodes, and odd levels contain check nodes.

Consider a variable node $\phi$ on an even level. All variable nodes in $H$ are incident to at least $\delta_\ell^-$ other nodes, by the construction of $H$. Therefore, $\phi$ has at least $\delta_\ell^- - 1 \geqslant 2$ children in the tree on the next level. Now consider a check node on an odd level; check nodes are each incident to at least two nodes, so this check node has at least one child on the next level.

Thus, the tree expands by a factor of at least $\delta_\ell^- - 1 \geqslant 2$ from an even to an odd level. From an odd to an even level, it may not expand, but it does not contract. The final level of the tree is level $(g/2) - 1$, and thus, the final even level is level $2(\lceil g/4 \rceil - 1)$. By the expansion properties we showed, this level (and, therefore, the tree $\mathcal{T}_i$) must contain at least $(\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}$ variable nodes.

By Claim 21, each tree is independent, so the number of variable nodes in $H$ is at least $h_1(\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}$. □

*Theorem 11:* Let $G$ be a factor graph with $\delta_\ell^- \geqslant 3$ and $\delta_r^- \geqslant 2$. Let $g$ be the girth of $G$, $g > 4$. Then the max-fractional distance is at least $d_{\text{frac}}^{\max} \geqslant (\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}$.

*Proof:* Let $(f, w)$ be an arbitrary vertex in $\mathcal{V}_Q^-$. Construct a pseudocodeword $(h, u)$ from $(f, w)$ as in Lemma 8; i.e., let $\beta$ be an integer such that $\beta f_i$ is an integer for all bits $i$, and $\beta w_{j,S}$ is an integer for all for all checks $j$ and sets $S \in E_j^-$. Such an integer exists because $(f, w)$ is a vertex of $Q$, and therefore rational [23]. For all bits $i$, set $h_i = \beta f_i$; for all checks $j$ and sets $S \in E_j^-$, set $u_{j,S} = \beta w_{j,S}$.

Let $H$ be a graph of the pseudocodeword $(h, u)$, as defined in Section V-A. By Lemma 22, $H$ has at least

$$(\max_i h_i)(\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}$$

variable nodes. Since the number of variable nodes is equal to $\sum_i h_i$, we have

$$\sum_i h_i \geqslant (\max_i h_i)(\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}. \tag{34}$$

Recall that $h_i = \beta f_i$. Substituting into (34), we have

$$\beta \sum_i f_i \geqslant \beta(\max_i f_i)(\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}.$$

It follows that

$$\left( \frac{\sum_i f_i}{\max_i f_i} \right) \geqslant (\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}.$$

This argument holds for an arbitrary $(f, w) \in \mathcal{V}_Q^-$ where $f \neq 0^n$. Therefore.

$$d_{\text{frac}}^{\max} = \min_{\substack{(f,w) \in \mathcal{V}_Q^- \\ f \neq 0^n}} \left( \frac{\sum_i f_i}{\max_i f_i} \right) \geqslant (\delta_\ell^- - 1)^{\lceil g/4 \rceil - 1}. \quad \square$$

## REFERENCES


[1] J. Feldman and D. R. Karger, "Decoding turbo-like codes via linear programming," in *Proc. 43rd Annu. IEEE Symp. Foundations of Computer Science (FOCS)*, Vancouver, BC, Canada, Nov. 2002, pp. 251–260.

[2] J. Feldman, M. J. Wainwright, and D. R. Karger, "Linear programming-based decoding of turbo-like codes and its relation to iterative approaches," in *Proc. Allerton Conf. Communications, Control and Computing*, Monticello, IL, Oct. 2002.

[3] C. Di, D. Proietti, I. E. Telatar, T. J. Richardson, and R. L. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channels," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1570–1579, Jun. 2002.

[4] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, Linköping, Sweden, 1996.

[5] G. D. Forney, F. R. Kschischang, B. Marcus, and S. Tuncel, "Iterative decoding of tail-biting trellises and connections with symbolic dynamics," in *Codes, Systems and Graphical Models*. New York: Springer-Verlag, 2001, pp. 239–241.

[6] G. D. Forney, R. Koetter, F. R. Kschischang, and A. Reznik, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," in *Codes, Systems and Graphical Models*. New York: Springer-Verlag, 2001, pp. 101–112.

[7] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, vol. IT-8, no. 1, pp. 21–28, Jan. 1962.

[8] D. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[9] M. Sipser and D. Spielman, "Expander codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1710–1722, Nov. 1996.

[10] S.-Y. Chung, G. D. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 58–60, Feb. 2001.

[11] R. McEliece, D. MacKay, and J. Cheng, "Turbo decoding as an instance of Pearl's belief propagation algorithm," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 140–152, Feb. 1998.

[12] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[13] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," in *Proc. IEEE Int. Symp. Information Theory*, Cambridge, MA, Oct. 1998, p. 117.

[14] B. J. Frey, R. Koetter, and A. Vardy, "Signal-space characterization of iterative decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 766–781, Feb. 2001.

[15] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode linear codes," presented at the 37th Annu. Conf. on Information Sciences and Systems (CISS '03), Baltimore, MD, Mar. 2003.

[16] J. Feldman, D. R. Karger, and M. J. Wainwright, "LP decoding," in *Proc. 41st Annu. Allerton Conf. Communications, Control, and Computing*, Monticello, IL, Oct. 2003.

[17] J. Feldman, "Decoding error-correcting codes via linear programming," Ph.D. dissertation, MIT, Cambridge, MA, 2003.

[18] J. Feldman, T. Malkin, R. A. Servedio, C. Stein, and M. J. Wainwright, "LP decoding corrects a constant fraction of errors," in *Proc. IEEE Int. Symp. Information Theory*, Chicago, IL, Jun./Jul. 2004, p. 68.

[19] J. Feldman and C. Stein, "LP decoding achieves capacity," in *Proc. Symp. Discrete Algorithms (SODA '05)*, Vancouver, BC, Canada, Jan. 2005.

[20] R. Koetter and P. O. Vontobel, "Graph-covers and iterative decoding of finite length codes," in *Proc. 3rd Int. Symp. Turbo Codes*, Brest, France, Sep. 2003, pp. 75–82.

[21] P. Vontobel and R. Koetter, "On the relationship between linear programming decoding and max-product decoding," paper submitted to Int. Symp. Information Theory and its Applications, Parma, Italy, Oct. 2004.

[22] ——, "Lower bounds on the minimum pseudo-weight of linear codes," in *Proc. IEEE Int. Symp. Information Theory*, Chicago, IL, Jun./Jul. 2004, p. 70.

[23] A. Schrijver, *Theory of Linear and Integer Programming*. New York: Wiley, 1987.

[24] M. Grotschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, no. 2, pp. 169–197, 1981.

[25] E. Berlekamp, R. J. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 3, pp. 384–386, May 1978.

[26] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches," in *Proc. 40th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2002.

[27] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Understanding Belief Propagation and its Generalizations," Mitsubishi Electric Res. Labs, Tech. Rep. TR2001-22, 2002.

[28] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*. Belmont, MA: Athena Scientific, 1997.

[29] R. G. Jeroslow, "On defining sets of vertices of the hypercube by linear inequalities," *Discr. Math.*, vol. 11, pp. 119–124, 1975.

[30] M. Yannakakis, "Expressing combinatorial optimization problems by linear programs," *J. Comp. Syst. Sci.*, vol. 43, no. 3, pp. 441–466, 1991.

[31] G. D. Forney Jr., "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 529–548, Feb. 2001.

[32] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sep. 1981.

[33] J. Rosenthal and P. O. Vontobel, "Constructions of LDPC codes using Ramanujan graphs and ideas from Margulis," in *Proc. 38th Annu. Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2000, pp. 248–257.

[34] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena Scientific, 1995.

[35] L. Lovász and A. Schrijver, "Cones of matrices and set-functions and 0–1 optimization," *SIAM J. Optimiz.*, vol. 1, no. 2, pp. 166–190, 1991.

[36] H. D. Sherali and W. P. Adams, "A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems," *SIAM J. Optimiz.*, vol. 3, pp. 411–430, 1990.

[37] *User's Manual for ILOG CPLEX*, 7.1 ed., ILOG, Inc., Mountain View, CA, 2001.