Adaptive Cut Generation for Improved Linear Programming Decoding of Binary Linear Codes

Xiaojie Zhang and Paul H. Siegel
University of California, San Diego, La Jolla, CA 92093, USA
Email:{ericzhang, psiegel}@ucsd.edu

***Abstract*—Linear programming (LP) decoding approximates optimal maximum-likelihood (ML) decoding of a linear block code by relaxing the equivalent ML integer programming (IP) problem into a more easily solved LP problem. The LP problem is defined by a set of linear inequalities derived from the constraints represented by the rows of a parity-check matrix of the code. Adaptive linear programming (ALP) decoding significantly reduces the complexity of LP decoding by iteratively and adaptively adding necessary constraints in a sequence of smaller LP problems. Adaptive introduction of constraints derived from certain additional redundant parity check (RPC) constraints can further improve ALP performance. In this paper, we propose a new and effective algorithm to identify RPCs that produce linear constraints, referred to as "cuts," that can eliminate non-ML solutions generated by the ALP decoder, often significantly improving the decoder error-rate performance. The cut-finding algorithm is based upon a specific transformation of an initial parity-check matrix of the linear block code. Simulation results for several low-density parity-check codes demonstrate that the modified ALP decoding algorithm significantly narrows the performance gap between LP decoding and ML decoding.**

I. Introduction

Linear programming (LP) decoding was first introduced by Feldman et al. [1] as an approximation to maximum-likelihood (ML) decoding. Many observations suggest similarities between the performance of LP and iterative belief propagation (BP) decoding methods [2]. However, there are some key differences that distinguish LP decoding from BP decoding. One of these differences is that the LP decoder has the *ML certificate property*, i.e., it is detectable if the decoding algorithm fails to find the ML codeword. When it fails to find a valid codeword, the LP decoder finds a non-integer solution, commonly called a *pseudocodeword*. Another difference is that while adding redundant parity checks satisfied by all the codewords can only improve LP decoding, it may have a negative effect on the BP decoder, especially in the waterfall region, due to the creation of short cycles in the Tanner graph. This property of LP decoding allows improvements by tightening the LP relaxation, i.e., reducing the feasible space of LP problem by adding more linear constraints.

In the original formulation of LP decoding proposed by Feldman *et al.*, the number of constraints in the LP problem is linear in the block-length but exponential in the maximum check node degree. In [3], Taghavi and Siegel introduced an adaptive linear programming (ALP) decoder in which these constraints are added in an adaptive and selective way. This approach also allows the adaptive incorporation of linear constraints generated by redundant parity checks (RPC) into the LP problem, making it possible to reduce the feasible space and improve the system performance. A linear inequality derived from an RPC that eliminates a pseudocodeword solution is referred to as a "cut." An algorithm proposed in [3] uses a random walk on a subset of the code factor graph to find these RPC cuts. However, the random nature of this algorithm limits its efficiency. Recently, authors in [4] proposed a separation algorithm which searches immediately for cuts that can be derived from an arbitrarily chosen dual codeword during each iteration of the LP decoding problem, and these cuts improve the error-correcting performance of the original LP decoder.

In this paper, we propose a novel adaptive cut-finding algorithm that greatly improves the error-correcting performance of LP decoding. First, we introduce an efficient approach to check whether a parity-check can generate a cut at nonintegral solution of the relaxed LP problem. We then propose a new, more efficient adaptive algorithm that identifies useful RPCs by performing specific elementary row operations on the original parity-check matrix of the binary linear code. By adding the corresponding linear constraints into the LP problem, we can significantly improve the error-rate performance of the LP decoder, even approaching the ML decoder performance in the high SNR region for some codes.

The remainder of the paper is organized as follows. In Section II, we review the original formulation of LP decoding and several adaptive LP decoding algorithms. In Section III, we describe our proposed algorithm for finding RPC cuts. Section IV presents our simulation results, and Section V concludes the paper.

II. LP Decoding and Adaptive Variants

A. LP Relaxation of ML Decoding

Consider a binary linear block code $\mathcal{C}$ of length n and a corresponding parity-check matrix $\mathbf{H}$. A codeword $\mathbf{y} \in \mathcal{C}$ is transmitted across a memoryless binary-input output-symmetric channel, resulting in a received vector $\mathbf{r}$. Assuming that the transmitted codewords are equiprobable, the ML decoder finds the solution to the following optimization problem

$$\begin{aligned} \text{minimize} \quad & \gamma^T \mathbf{u} \\ \text{subject to} \quad & \mathbf{u} \in \mathcal{C} \end{aligned} \tag{1}$$

where $u_i \in \{0,1\}$, and γ is the vector of log-likelihood ratios (LLR) defined as

$$\gamma_i = \log\left(\frac{\Pr(r_i|\, u_i = 0)}{\Pr(r_i|\, u_i = 1)}\right). \tag{2}$$

Since the ML decoding problem (1) is an integer programming problem, it is desirable to replace its nonlinear constraints with a set of linear constraints, transforming the IP problem into a more readily solved LP problem. As an approximation to ML decoding, Feldman *et al.* relaxed the codeword polytope onto the *fundamental polytope*, denoted as $\mathcal{P}$. This polytope has both nonintegral and integral vertices, with the latter corresponding precisely to the codewords in $\mathcal{C}$, yielding the ML-certificate property mentioned above. The fundamental polytope is described by a set of linear inequalities, obtained as follows. For each row $j = 1, \ldots, m$ of the parity-check matrix, corresponding to a check node in the associated Tanner graph, the linear inequalities used to form the fundamental polytope $\mathcal{P}$ are given by

$$\sum_{i\in\mathcal{V}}(1-u_i) + \sum_{i\in\mathcal{N}(j)\setminus\mathcal{V}} u_i \geq 1, \forall \mathcal{V} \subseteq \mathcal{N}(j) \text{ s.t. } |\mathcal{V}| \text{ is odd}. \tag{3}$$

These are equivalent to

$$\sum_{i\in\mathcal{V}} u_i - \sum_{i\in\mathcal{N}(j)\setminus\mathcal{V}} u_i \leq |\mathcal{V}| - 1, \forall \mathcal{V} \subseteq \mathcal{N}(j) \text{ s.t. } |\mathcal{V}| \text{ is odd} \tag{4}$$

where $\mathcal{N}(j) \subseteq \{1, 2, \ldots, n\}$ is the set of neighboring variable nodes of the check node j in the Tanner graph; that is, $\mathcal{N}(j) = \{i : H_{j,i} = 1\}$ where $H_{j,i}$ is the element in the jth row and ith column of the parity-check matrix, $\mathbf{H}$.

B. ALP Decoding

In the original formulation of LP decoding presented in [1], every check node j generates $2^{|\mathcal{N}(j)|-1}$ parity inequalities that are used as linear constraints in the LP problem described above. The total number of constraints and the complexity of the LP problem grows exponentially with the maximum check node degree. In [3], an adaptive approach, called adaptive linear programming (ALP) decoding, was proposed as an alternative to the direct implementation of the original LP decoding algorithm. The ALP decoder exploits the structure of the LP decoding problem, reflected in the statement of the following lemma.

Lemma 1 ([3]): If at any given point $u \in [0,1]^n$, one of the parity inequalities introduced by a check node j is violated, the rest of the parity inequalities from this check node are satisfied with strict inequality.

Definition 1: Given a parity-check node j, a set $\mathcal{V} \subseteq \mathcal{N}(j)$ of odd cardinality, and a vector $\mathbf{u} \in [0,1]^n$ such that the corresponding parity inequality of the form (3) or (4) does not hold, we say that the constraint is *violated* or, more succinctly, a *cut* at $\mathbf{u}$.

In [3], an efficient algorithm for finding cuts at a vector $\mathbf{u} \in [0,1]^n$ was presented. It relies on the observation that violation of a parity inequality (4) at $\mathbf{u}$ implies that

$$|\mathcal{V}| - 1 < \sum_{i\in\mathcal{V}} u_i \leq |\mathcal{V}| \tag{5}$$

and

$$0 \leq \sum_{i\in\mathcal{N}(j)\setminus\mathcal{V}} u_i < u_v, \forall v \in \mathcal{V}. \tag{6}$$

The algorithm first puts the entries of $\mathbf{u}$ in non-increasing order, i.e., $u_1 \geq \cdots \geq u_n$. It then successively considers subsets of odd cardinality having the form $\mathcal{V} = \{u_1, \ldots, u_{2k+1}\}$, increasing the size of $\mathcal{V}$ by two each step, until a cut (if one exists) is found. This algorithm can find a cut among the constraints corresponding to a check node j by examining at most $|\mathcal{N}(j)|/2$ inequalities, rather than exhaustively checking all $2^{|\mathcal{N}(j)|-1}$ inequalities in the original LP decoding formulation.

The ALP decoding algorithm starts by solving the optimization problem with the following constraints

$$\begin{cases} 0 \leq x_i & \text{if} \quad \gamma_i \geq 0 \\ x_i \leq 1 & \text{if} \quad \gamma_i < 0. \end{cases} \tag{7}$$

The solution of this initial problem can be obtained simply by making a hard decision on the components of a received vector. The ALP decoding algorithm starts with this point, searches every check node for cuts, adds all the cuts found during the search into the LP problem, and solves it again. This procedure is repeated until an optimal integer solution is generated or no more cuts can be found. (See [3] for more details). The adaptive LP decoding algorithm has exactly the same error-correcting performance as the original LP decoder.

III. Adaptive Cut-Generating Algorithm

In this section, we provide an efficient method to search for violated parity inequalities (or cuts) corresponding to an existing parity-check. This result serves as the basis for a new adaptive approach to generating RPCs that provide cuts. Computer simulation results, presented in Section IV, indicate that the new cut-generating algorithm can significantly improve the error-rate performance of LP decoding.

A. Finding Cuts from Parity-check Nodes

Consider the original parity inequalities in (3) given by Feldman *et al.* in [1]. If a parity inequality derived from check node j induces a cut at $\mathbf{u}$, the cut can be written as

$$\sum_{i\in\mathcal{V}}(1-u_i) + \sum_{i\in\mathcal{N}(j)\setminus\mathcal{V}} u_i < 1, \quad \text{for some } \mathcal{V} \subseteq \mathcal{N}(j) \text{ and } |\mathcal{V}| \text{ is odd} \tag{8}$$

From (8) and Lemma 1, we can derive the following necessary condition for a parity-check constraint to induce a cut.

Theorem 1: Given a vector $\mathbf{u}$, let $\mathcal{S} = \{i \in \mathcal{N}(j) | 0 < u_i < 1\}$ be the set of nonintegral neighbors of parity-check node j, and let $\mathcal{T} = \{i \in \mathcal{S} | u_i > \frac{1}{2}\}$. A necessary condition for parity-check constraint j to induce a cut at $\mathbf{u}$ is

$$\sum_{i\in\mathcal{T}}(1-u_i) + \sum_{i\in\mathcal{S}\setminus\mathcal{T}} u_i < 1. \tag{9}$$

Algorithm 1 Cut Search Algorithm

Input: parity-check node j and vector $\mathbf{u}$

Output: variable node set $\mathcal{V}$

$\mathcal{V} \leftarrow \{i \in \mathcal{N}(j) | u_i > \frac{1}{2}\}$
if $|\mathcal{V}|$ is even **then**
  $i^* \leftarrow \arg\min_{i \in \mathcal{N}(j)} |\frac{1}{2} - u_i|$
  if $i^* \in \mathcal{V}$ **then**
    $\mathcal{V} \leftarrow \mathcal{V} \setminus \{i^*\}$
  else
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{i^*\}$
  end if
end if
if $\sum_{i \in \mathcal{V}} (1 - u_i) + \sum_{i \in \mathcal{N}(j) \setminus \mathcal{V}} u_i < 1$ **then**
  Find a violated parity inequality on check node j
else
  There is no violated parity inequality on check node j
  $\mathcal{V} \leftarrow \emptyset$
end if
return $\mathcal{V}$

This is equivalent to

$$\sum_{i \in \mathcal{S}} |\frac{1}{2} - u_i| > \frac{1}{2} \cdot |\mathcal{S}| - 1 \tag{10}$$

where, for $x \in \mathbb{R}$, $|x|$ denotes the absolute value, and for a set $\mathcal{X}$, $|\mathcal{X}|$ denotes its cardinality.

Proof: Omitted due to space limitations. ∎

Remark 1: Given a nonintegral vector $\mathbf{u}$, to see whether a parity check node could provide a cut at $\mathbf{u}$, we only need to check its fractional neighbors.

We can further extend Lemma 1 to get a sufficient condition for a parity-check node to give a cut at $\mathbf{u}$.

Theorem 2: Given a vector $\mathbf{u}$, let $\mathcal{S} = \{i \in \mathcal{N}(j) | 0 < u_i < 1\}$ and $\mathcal{T} = \{i \in \mathcal{S} | u_i > \frac{1}{2}\}$. If the inequality

$$\sum_{i \in \mathcal{T}} (1 - u_i) + \sum_{i \in \mathcal{S} \setminus \mathcal{T}} u_i + 2 \cdot \min_{i \in \mathcal{N}(j)} |\frac{1}{2} - u_i| < 1 \tag{11}$$

holds, there must be a violated parity inequality derived from parity-check j. This sufficient condition can be written as

$$\sum_{i \in \mathcal{S}} |\frac{1}{2} - u_i| - 2 \cdot \min_{i \in \mathcal{N}(j)} |\frac{1}{2} - u_i| > \frac{1}{2} \cdot |\mathcal{S}| - 1. \tag{12}$$

Proof: Omitted due to space limitations. ∎

Theorem 1 and Theorem 2 provide a necessary condition and a sufficient condition, respectively, for a parity-check node to produce a cut at any given vector $\mathbf{u}$. Together, they form the basis for a highly efficient technique for finding cuts, the Cut Search Algorithm described in Algorithm 1. If there is a violated parity inequality, the Cut Search Algorithm returns the set $\mathcal{V}$ corresponding to the cut; otherwise, it returns an empty set.

B. Generating Cut-Inducing RPCs

Although the addition of a redundant row to a parity-check matrix does not affect the null-space and, therefore, the linear code it defines, different parity-check matrix representations of a linear code may give different fundamental polytopes underlying the corresponding LP relaxation of the ML decoding problem. This fact inspires the use of cutting-plane techniques to improve the error-correcting performance of LP and ALP decoders. Specifically, when the LP decoder gives a nonintegral solution (i.e., a pseudocodeword), we try to find the RPCs that introduce cuts at that point. The cuts obtained in this manner are called *RPC cuts*. The effectiveness of this method depends on how closely the new relaxation approximates the ML decoding problem, as well as on the efficiency of the technique used to search for the cut-generating RPCs.

An RPC can be obtained by modulo-2 addition of some of the rows of the original parity-check matrix, and this new check introduces a number of linear constraints that may give a cut. In [3], a random walk on a cycle within the subgraph defined by the fractional-valued entries in a pseudocodeword served as the basis for a search for RPC cuts. However, there is no guarantee that this method will find a cut (if one exists) within a finite number of iterations. In fact, the average number of random trials needed to find an RPC cut grows exponentially with the code length. The separation algorithm in [4] provides another way to search for RPC cuts, but in many cases the RPC cuts found by this algorithm do not lead to an integral solution, and therefore it provides only limited performance improvement. Motivated by the Cut Search Algorithm introduced in Section III-A, we next propose a new, efficient RPC cut-generating algorithm that has been found empirically to provide useful RPC cuts.

Given a nonintegral solution of the LP problem, we can see from Theorem 1 and Theorem 2 that an RPC with a small number of nonintegral neighboring variable nodes may be more likely to satisfy the necessary condition for generating a cut at the pseudocodeword. Moreover, the nonintegral neighbors should have values either close to 0 or close to 1; in other words, they should be as far from $\frac{1}{2}$ as possible.

Let $\mathbf{p} = (p_1, p_2, \ldots, p_n) \in [0, 1]^n$ be a pseudocodeword solution to LP decoding, with a nonintegral positions, b zeros, and $n - a - b$ ones. We first group entries of $\mathbf{p}$ according to whether their values are nonintegral, zero, or one. Then, we sort the nonintegral positions in ascending order according to the value of $|\frac{1}{2} - p_i|$ and define the permutated vector $\mathbf{p}' = \Pi(\mathbf{p})$ satisfying $|\frac{1}{2} - p'_1| \leq \cdots \leq |\frac{1}{2} - p'_a|$, $p'_{a+1} = \cdots = p'_{a+b} = 0$, and $p'_{a+b+1} = \cdots = p'_n = 1$. By applying the same permutation Π to the columns of the original parity-check matrix $\mathbf{H}$, we get

$$\mathbf{H}' \triangleq \Pi(\mathbf{H}) = \left(\mathbf{H}^{(f)} | \mathbf{H}^{(0)} | \mathbf{H}^{(1)}\right) \tag{13}$$

where $\mathbf{H}^{(f)}$, $\mathbf{H}^{(0)}$, and $\mathbf{H}^{(1)}$ consist of columns of $\mathbf{H}$ corresponding to positions of $\mathbf{p}'$ with nonintegral values, zeroes, and ones, respectively.

The following familiar definition from matrix theory will be useful [5, p. 10].

Definition 2: A matrix is in *reduced row echelon form* if its nonzero rows (i.e., rows with at least one nonzero element) are above any rows of all zeroes, and the leading entry (i.e., the first nonzero number from the left) of a nonzero row is the only nonzero entry in its column and is always strictly to the right of the leading coefficient of the row above it.

By applying a suitable sequence of elementary row operations Φ (over $\mathbb{F}_2$) to $\mathbf{H}'$, we get

$$\bar{\mathbf{H}} \triangleq \Phi(\mathbf{H}') = \left(\bar{\mathbf{H}}^{(f)} | \bar{\mathbf{H}}^{(0)} | \bar{\mathbf{H}}^{(1)}\right), \tag{14}$$

where $\bar{\mathbf{H}}^{(f)}$ is in reduced row echelon form. Applying the inverse permutation Π^{-1} to the columns of $\bar{\mathbf{H}}$, we get an equivalent parity-check matrix $\tilde{\mathbf{H}} = \Pi^{-1}(\bar{\mathbf{H}})$, whose rows are likely to be cut-generating RPCs.

Theorem 3: If there exists a weight-one row in submatrix $\bar{\mathbf{H}}^{(f)}$, the corresponding row of the equivalent parity-check matrix $\tilde{\mathbf{H}}$ is a cut-generating RPC.

Proof: Given a pseudocodeword $\mathbf{p}$, suppose the jth row of submatrix $\bar{\mathbf{H}}^{(f)}$ has weight one and the corresponding nonintegral position in $\mathbf{p}$ is p_i. Since it is the only nonintegral neighbor of RPC j, the left-hand side of (12) is equal to $-|\frac{1}{2} - p_i|$. Since $0 < p_i < 1$, this is larger than $-\frac{1}{2}$, the right-hand side. Hence, according to Theorem 2, RPC j satisfies the sufficient condition for providing a cut. In other words, there must be a violated parity inequality induced by RPC j. ■

For other rows of $\tilde{\mathbf{H}}$, we can apply the Cut Search Algorithm to find any further potential cuts. Algorithm 2 describes an improved LP decoding algorithm which includes the adaptive cut generation techniques just described.

Algorithm 2 LP Decoding with Adaptive Cut Generation

Input: cost vector $\mathbf{L}$, parity-check matrix $\mathbf{H}$
Output: Optimal solution of current LP problem

1: Initialize LP problem with the constraints in (7).
2: Solve the current LP problem; get optimal solution x^*.
3: Apply **Algorithm 1** on each row of $\mathbf{H}$.
4: **if** No cut is found **and** x^* is nonintegral **then**
5: Construct $\tilde{\mathbf{H}}$ according to x^*
6: Apply **Algorithm 1** on each row of $\tilde{\mathbf{H}}$.
7: **end if**
8: **if** No cut is found **then**
9: Terminate.
10: **else**
11: Add found cuts into the LP problem; go to line 2.
12: **end if**

IV. Numerical Results

To demonstrate the improvement offered by the proposed RPC search algorithm, we compare its error-correcting performance to that of LP/ALP decoding , BP decoding (sum-product algorithm with a maximum of 1000 iterations), the Separation Algorithm (SA) [4], and ML decoding for two

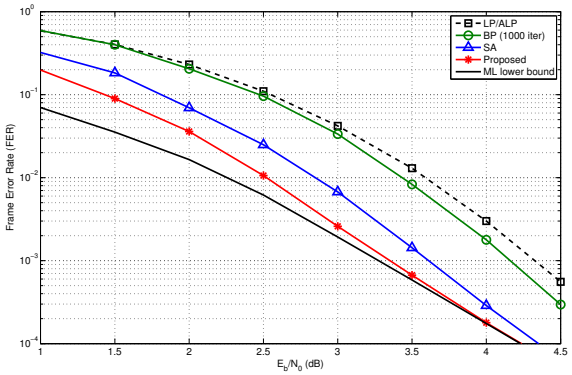

Fig. 1. FER versus E_b/N_0 for MacKay's random (3,6)-regular LDPC code of length 96.

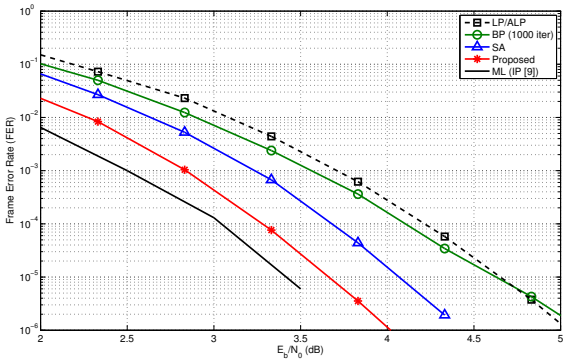

Fig. 2. FER versus E_b/N_0 for the (3,5)-regular (155,64) Tanner LDPC code on the AWGN channel.

TABLE I
Frame Errors Using Algorithm 2

E_b/N_0 (dB)	Transmitted Frames	Error Frames	Pseudo-codewords	Incorrect Codewords
3.0	386,445	1,000	255	745
3.5	1,493,801	1,000	126	874
4.0	5,589,843	1,000	23	977
4.5	19,128,086	1,000	4	996
5.0	61,198,400	1,000	0	1,000
5.5	225,764,513	1,000	0	1,000

LDPC codes on the additive white Gaussian noise (AWGN) channel. We use the Simplex algorithm from the open-source GNU Linear Programming Kit (GLPK [6]) as the LP solver. The LDPC codes we evaluated are a length-96, (3,6)-regular LDPC code [7] and the (155,64) Tanner LDPC code [8].

In Fig. 1, we show the results for MacKay's length-96, (3,6)-regular LDPC code (the 96.33.964 code from [7]). As a benchmark, we also plot a lower bound on ML decoding performance. In order to obtain this ML lower bound, we counted the number of times that our proposed algorithm converged to an incorrect codeword and then divided that by the total number of transmitted codewords, as shown in Table I. The ML certificate property of LP decoding implies that ML decoding would also fail in these cases, and it would also probably fail on some other cases where LP decoding outputs pseudocodewords. Therefore, this estimate gives us a

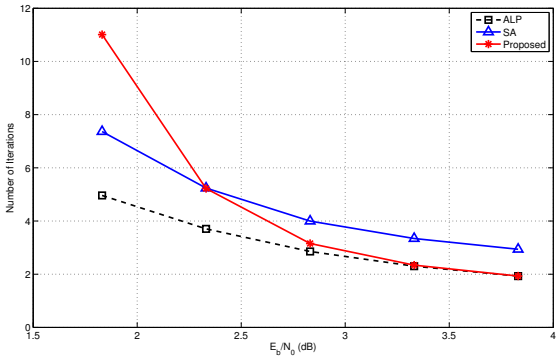

Fig. 3. Average number of iterations for decoding one codeword of (155,64) Tanner LDPC code.

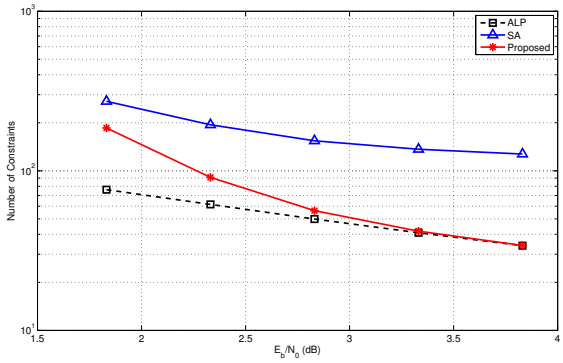

Fig. 4. Average number of constraints in final iteration for decoding one codeword of (155,64) Tanner LDPC code.

lower bound on the frame error rate (FER) of ML decoding. We can see that the proposed search algorithm approaches the ML lower bound in the high SNR region. Actually, from the simulation results, reflected in Table I, we observed that, when E_b/N_0 is greater than 4.5 dB, all decoding errors correspond to incorrect codewords, which means that the proposed algorithm basically achieved ML decoding performance. In Fig. 2, which shows results for the (3,5)-regular (155,64) Tanner code, we plot the ML performance curve from [4]. It can be seen that the proposed algorithm closes the 1.25 dB gap between LP decoding and ML decoding to approximately 0.25 dB.

Since the improvement in error-rate performance comes from the additional RPC cuts found in each iteration, our algorithm generally requires more iterations and the solution of larger LP problems in comparison to ALP decoding. In the remaining part of this section, we investigate the relative complexity of our proposed algorithm in comparison to ALP decoding and the Separation Algorithm [4]. The simulation statistics are averaged over the number of transmitted codewords required for the decoder to fail on 200 codewords. In Fig. 3, we compare the average number of iterations needed to decode one codeword, i.e., the average number of LP problems solved to decode one codeword of the Tanner (155,64) code on the AWGN channel. Fig. 4 shows the average number of constraints in the LP problem of the final iteration that either outputs a valid codeword or, when no more cuts can be found, a pseudocodeword. The decoding time depends on both the number of decoding iterations and the size of the LP problem in each iteration.

The results show that the improvement in performance achieved by the proposed algorithm does come at the cost of increased decoding time. As the SNR increases, however, the average numbers of iterations and constraints required to decode one codeword using the proposed algorithm approach those of the ALP decoder. This is because, at higher SNR values, the ALP decoder can often successfully decode the received frames without requiring any RPC cuts. In contrast, the Separation Algorithm becomes less efficient as the SNR increases because it can not find cuts efficiently and therefore requires more iterations to complete the decoding.

V. Conclusion

In this paper, we derived a new sufficient condition and a new necessary condition for a parity-check constraint in an LDPC code definition to give a cut at a pseudocodeword produced by LP decoding. Using these results, we developed an efficient algorithm to search for cuts and proposed an effective RPC cut generating algorithm. The key innovation in the cut generating algorithm is a particular transformation of the parity-check matrix used to define the LP decoding problem. By properly re-ordering the columns of the original parity-check matrix and transforming it to a partial reduced row echelon form, we could efficiently identify RPC cuts that were found empirically to significantly improve the LP decoder performance. Error-rate simulation results for two moderate-length codes show that the proposed algorithm outperforms previously proposed decoding algorithms based upon linear programming, closing the performance gap to ML decoding to less than 0.3 dB when the FER is less than 10^{-2}.

Acknowledgment

This work was supported in part by the National Science Foundation under Grant CCF-0829865.

References

[1] J. Feldman, M. Wainwright, and D. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inform. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.

[2] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-length analysis of message-passing iterative decoding of LDPC codes," *IEEE Trans. Inform. Theory*, accepted for publication.

[3] M. H. Taghavi and P. H. Siegel, "Adaptive methods for linear programming decoding," *IEEE Trans. Inform. Theory*, vol. 54, no. 12, pp. 5396–5410, Dec. 2008.

[4] A. Tanatmis, S. Ruzika, H. W. Hamacher, M. Punekar, F. Kienle, and N. Wehn, "A separation algorithm for improved LP-decoding of linear block codes," *IEEE Trans. Inform. Theory*, vol. 56, no. 7, pp. 3277–3289, Jul. 2010.

[5] R.A. Horn and C.R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1990.

[6] GNU Linear Programming Kit, http://www.gnu.org/software/glpk

[7] D. J. C. MacKay, *Encyclopedia of Sparse Graph Codes*. [Online]. Available: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html

[8] R. M. Tanner, D. Sridhara, and T. Fuja, "A class of group-structured LDPC codes," in *Proc. Int. Symp. Communication Theory and Applications (ISCTA)*, Ambleside, U.K., Jul. 2001.