Decoding Turbo-Like Codes via Linear Programming

Jon Feldman David R. Karger*

MIT Laboratory for Computer Science
Cambridge, MA, 02139
{jonfeld,karger}@theory.lcs.mit.edu

Abstract

We introduce a novel algorithm for decoding turbo-like codes based on linear programming. We prove that for the case of Repeat-Accumulate (RA) codes, under the binary symmetric channel with a certain constant threshold bound on the noise, the error probability of our algorithm is bounded by an inverse polynomial in the code length.

Our linear program (LP) minimizes the distance between the received bits and binary variables representing the code bits. Our LP is based on a representation of the code where code words are paths through a graph. Consequently, the LP bears a strong resemblance to the min-cost flow LP. The error bounds are based on an analysis of the probability, over the random noise of the channel, that the optimum solution to the LP is the path corresponding to the original transmitted code word.

1 Introduction

The introduction of turbo codes [3] revolutionized the field of coding theory by achieving an error probability orders of magnitude smaller than any other code at the time. Since then, volumes of research has focused on design, implementation, and analysis of turbo codes and their variants and generalizations [17].

One of the main goals of this research has been to explain the somewhat mysterious good performance of turbo codes and turbo-like codes. Even though the distances of turbo-like codes are generally bad [10, 2, 5], when decoded using an iterative decoder, they seem to achieve very good error rates [6]. The drawback to an iterative decoder is that it not guaranteed to converge, nor does it have any guarantee on the quality of its output.

Some progress has been made assuming optimal maximum-likelihood (ML) decoding, for which no polynomial-time algorithm is known. It is known [6] that if the noise in the channel is under a certain constant threshold, ML decoding of randomly generated turbo codes has an error probability bounded by an inverse polynomial in the code length, as the code length goes to infinity. The first such result by Divsalar and McEliece[6] was for a type of turbo-like code called a *repeat-accumulate* (RA) code.

There are several drawbacks to these results. The fact that they do not apply to a specific constructible code forces the designer of the code to choose a random one, and thus be uncertain about its quality. Additionally, in many cases the asymptotic nature of the bound requires a large block length, whereas small fixed code lengths are desirable in practice to reduce latency in transmission. Most importantly, however, the given error probability is proven only under ML decoding, for which no efficient algorithm is known.

Our Results. In this paper we introduce a novel approach to decoding any turbo-like[1] code based on linear programming. We prove that for the case of Repeat-Accumulate (RA) codes, with a certain constant threshold bound on the noise in the channel, the error probability of our algorithm is bounded by an inverse polynomial in the code length. We improve upon previous results in three important respects. Our analysis holds for (i) a provably polynomial-time decoding algorithm, (ii) a specific, deterministically constructible code, and (iii), *any* code length.

More precisely, we show that for a particular RA code

*Research supported by NSF contract CCR-9624239 and a David and Lucille Packard Foundation Fellowship.

[1] We note that our definition for turbo-like codes is along the lines of [6], where the class includes any serial or parallel concatenated convolutional code.

with rate $1/2 - o(1)$ and length n, our LP makes an error with probability at most $n^{-\epsilon}$, for any $\epsilon > 0$, as long as $p < 2^{-4(\epsilon+(\log 24)/2)}$, where each code bit is flipped by the channel with probability p. As $\epsilon \to 0$, the threshold on p approaches $\approx 2^{-9.17}$. For lower rate RA codes, our bound on error rate is trivially applicable by simply decoding an embedded rate-$1/2$ RA code; however, we expect that a more general analysis will yield better bounds.

When applied to *any* turbo-like code, our decoder has the desirable property that when it outputs a code word, it is guaranteed to be the maximum-likelihood code word. As far as the authors are aware, no other efficient algorithm for decoding turbo-like codes is known to have this *ML-certificate* property. Additionally, the key structural theorem used to prove the error bound for RA codes easily generalizes to other turbo-like codes, and provides a good basis for proving better error bounds.

Previous Work. A breakthrough in the analysis of turbo codes came when McEliece, MacKay and Cheng[12] showed how the classic iterative decoding method for turbo codes is an instance of Pearl's *belief propagation* (BP) algorithm, a standard tool used in the artificial intelligence community. Now most work in the areas of turbo codes and low-density parity check (LDPC) codes is interpreted in this context (Richardson, Shokrollahi and Urbanke[13], for example).

The convergence of BP algorithms becomes difficult to prove when the underlying "belief network" contains cycles, as is the case for turbo codes. However, a lot of progress has been made by analyzing average codes (or "code ensembles"), giving various tradeoffs involving rate, probability thresholds, and iterations of the BP algorithm [14].

In follow-up work (with Wainwright [8]), we have shown that the the recent iterative tree-reweighted max-product (TRMP) algorithm for MAP estimation of graphical models of Wainwright, Jaakkola and Willsky [18], when applied to the problem of decoding turbo-like codes, has a fixed point equivalent to the solution of the LP we present here. Thus we have begun to connect our LP-based decoder with the world of iterative decoders.

The *minimum distance* of a code is the minimum Hamming distance between any two code words. The minimum distance of turbo-like codes has received some attention recently [10, 2, 5]. Most of the work has focused on the negative side, showing that the minimum distance of a turbo-like code is sub-linear. Kahale and Urbanke [10] give high-probability upper and lower bounds on the minimum distance of a random interleaver, as the block length goes to infinity, for any parallel or serially concatenated convolutional code. Bazzi, Mahdian, Miller and Spielman [2] give similar upper bounds over all interleavers for some of the same types of codes. They also give a construction of a rate-$1/2$ RA code and show its minimum distance is $\Omega(\log n)$. We will discuss this code in more detail later in the paper.

Coding Theory Background. An *error-correcting code* is used to build redundancy into a data stream for transmission over an unreliable channel. The sending party takes a binary vector x of length k (the *information word*), applies an *encoder* (a function $E : \{0,1\}^k \to \{0,1\}^n$), to obtain a *code word* of n bits, $n \geqslant k$ (the parameter n is referred to as the *code length* or *block length*), and transmits the code word $y = E(x)$ over the channel. The *rate* of the code is k/n. The code word is then subject to an unreliable channel that can be modeled in several ways. Here we will use the *binary symmetric channel* (BSC), where each bit of the code word is flipped independently with probability p (called the *crossover probability*). Our results in this paper also hold for the additive white Gaussian noise (AWGN) channel, where the channel adds an independent random Gaussian variable to each transmitted bit; for clarity, we discuss only the BSC.

The receiving party gets the corrupted code word v of length n, and must try to recover the original information word x using a *decoder*. The decoder is simply a function $D : \{0,1\}^n \to \{0,1\}^k$. The *word error probability* P_w of the decoder is the probability, over the random coin flips of the channel, that $D(v) \neq x$. The *maximum-likelihood* (ML) information word x is the one that maximizes $\Pr[E(x)$ was sent $\mid v$ was received $]$. Using Bayes' rule, and the conventional assumption that all information words have equal probability, this is the same information word x that maximizes $\Pr[v$ was received $\mid E(x)$ was sent $]$. Thus, under the BSC, x is the information word that minimizes the Hamming distance between $D(x)$ and v. Note that the ML information word is not necessarily the original encoded information word. An ML decoder is one that always finds the ML information word.

The purpose of an error-correcting code is to be robust against noise, so the word error probability P_w is the metric that should be used to measure the quality of a code. Codes are often measured for quality in terms of their *minimum distance* d. This is the minimum, over all pairs of valid code words, of the Hamming distance between the pair. It is not hard to see that an ML decoder will always correct up to $\lceil d/2 \rceil - 1$ errors in the channel, so a large minimum distance is desirable. However, the minimum distance is a "worst-case" measure, so considering it as the only measure of quality ignores other important attributes of the code that affect the word error probability. In fact, turbo codes are a perfect example of codes whose minimum distance is considered bad, but whose word error probability is good.

We will use the definitions given here for k, n, x, y, v and p throughout the paper. For more background on error-correcting codes, we refer the reader to textbooks written on the subject [19, 11].

Our Techniques. We show that the problem of finding the ML code word (referred to as *ML decoding*) can be solved by finding an optimal *integral* point inside a polytope with linear constraints. Binary variables represent the information word x, and the objective function is to minimize $\Pr[v \text{ was received} \mid x \text{ was sent }]$, where v is the corrupted code word received from the channel. We relax the integral constraints to obtain a linear program (LP). The algorithm solves the LP, and if the solution is integral, outputs the information word, which it knows is the ML information word. If the solution is not integral, the algorithm outputs "error." This approach guarantees the ML-certificate property discussed earlier.

In the setting of decoding algorithms, the rounding scheme for an LP relaxation is not as important as it is in other more conventional optimization problems. Because the LP is only guaranteed to output the ML information word if it finds an *integral* solution, even if we provide a rounding scheme with a provably small approximation ratio, it does not help bound the probability that the decoder returns the ML information word.

Therefore, instead of analyzing the approximation ratio as is normally done, we assume that the algorithm is always wrong when the solution is fractional, and bound the probability that the LP returns the *real* information word that was transmitted (not the ML word). By doing so, not only do we bound the error probability of our polynomial-time algorithm, we also bound the error probability of ML decoding; this is because whenever our LP finds the correct solution, this solution must also be the optimal point in the integral polytope, and an ML decoder would have found it as well.

Our LP can be used to decode *any* set of concatenated codes, with interleavers between them. However, the LP has polynomial size only if the component codes can be expressed using a polynomial-sized "trellis." The simplest such codes are the class of *convolutional codes* (see [19]). Thus we say that the LP decoder is a decoder intended for "turbo-like" [6] codes: serial or parallel concatenated convolutional codes.

Outline. In Section 2, we define repeat-accumulate (RA) codes, and present our linear program as applied to RA codes. In Section 3 we state our main structural theorem (proven in Section 4), and show our bound on the probability that the LP returns the original information word. We discuss the generalization of the linear program to any set of concatenated codes in Section 5. In Section 6 we give several interesting open questions that arise from this work.

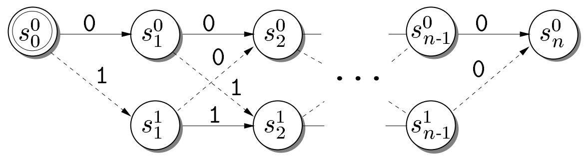

Figure 1. The trellis for an accumulator, used in RA codes. The dashed-line *switch-edges* correspond to information bit `1`, the solid-line *remain-edges* to information bit `0`. The edges are labeled with their associated output code bit.

2 Repeat-Accumulate Codes

Repeat-Accumulate codes are perhaps the simplest nontrivial example of a turbo-like code. They were introduced by Divsalar and McEliece [6] in order to show the first bounds for error probabilities under ML decoding. Their simple structure and highly efficient encoding scheme make them both practical and simpler to analyze than other more complex turbo-like codes. They have also been shown experimentally to have excellent error-correcting ability under iterative decoding [6], on par with classic turbo codes.

Encoding. The encoder for an RA code takes the input word, repeats every bit ℓ times, then sends it through an *interleaver* (known permutation), and then through an *accumulator*. The accumulator maintains a partial sum mod 2 of the input seen so far, and outputs the new sum at each step.

More formally, an RA(ℓ) code of length $n = \ell k$ has an interleaver (permutation) $\pi : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$. The encoder is given an information word x of length k. Let x' be the length-n repeated and permuted information word, i.e., for all $t \in \{0, \ldots, n-1\}$, $x'_t = x_{\lfloor \pi^{-1}(t)/\ell \rfloor}$. The RA encoder outputs a code word y of length n, where for all $j \in \{0, \ldots, n-1\}$, $y_j = \sum_{t=0}^{j} x'_t \mod 2$.

For all $i \in \{0, \ldots, k-1\}$, let X_i be the set of indices to which information bit x_i was repeated and permuted, i.e., $X_i = \{\pi(\ell i), \pi(\ell i + 1), \ldots, \pi(\ell i + \ell - 1)\}$. Let $\mathcal{X} = \{X_i : i \in \{0, \ldots, k-1\}\}$. To keep the proofs in this paper simpler, we assume that the input contains an even number of `1`s. This can be achieved by padding the information sequence with an extra parity bit. Thus the rate of this code is $(k-1)/\ell k$, or $1/\ell - o(1)$. The $o(1)$ can be avoided by a more technical proof, which we leave out for clarity.

The Accumulator Trellis. A *trellis* is a simple layered graph that models the actions of a finite-state encoder over time, as it encodes a binary string passed to it. The trellis is the basis for the classic Viterbi decoding algorithm [16, 9].

Figure 1 shows the trellis T for an accumulator.

To derive the trellis, we view the accumulator as a simple finite state machine, receiving the n-bit binary input string x' one bit at a time. The accumulator has two possible states per time step, depending on the parity of the input bits seen so far. We refer to the two states of the accumulator as 0 and 1; state 0 represents even parity, 1 represents odd parity. We use $\{s_0^0, \ldots, s_n^0\}$ and $\{s_1^1, \ldots, s_{n-1}^1\}$ to refer to the sets of even and odd parity nodes, respectively.

An encoder using this trellis begins in state s_0^0. At each time step, if it receives a `1` as input, it follows the dashed-line transition to the next layer of nodes, switching states. We call this a *switch-edge*. If it receives a `0`, it follows the solid-line transition to the next layer of nodes, staying in the same state (see Figure 1). We call this type of edge a *remain-edge*. The labels on the transition edges represent the code bit output by the encoder taking that transition; label `0` for edges entering state 0, label `1` for edges entering state 1.

A path from s_0^0 across the trellis to s_n^0 corresponds to an entire n-bit input string. Since the accumulator begins and ends in state 0 (the input string has even parity by assumption), we do not need the states s_0^1 and s_n^1. Looking at the edge labels on the path, one can read off the code word corresponding to that input string. Let P_x be the path taken by the encoder through the trellis T while encoding x'.

We will refer to the group of four edges at one time step of the trellis as a *segment*. Define the *cost* $c[e]$ of an edge e in the trellis at segment t to be the Hamming distance between its label and the received bit v_t. The cost of a path is the sum of the costs of the edges on the path.

Decoding with the Trellis. Assume for the moment that the accumulator was the entire encoder (i.e., the information bits were fed directly into the accumulator without repeating and permuting). Then, all paths through the trellis from s_0^0 to s_n^0 represent valid information words, and the labels along the path represent valid code words. Furthermore, the cost of the path is the Hamming distance between the code word and the received word. A simple shortest-path computation thus yields the ML information word. The decoding algorithm we just described is exactly the Viterbi algorithm [16, 9, 19].

However, if we tried to apply the Viterbi algorithm to RA codes, we would run into problems. For example, suppose $\ell = 2$, and let $x_i = 1$ be some arbitrary information bit, where $X_i = \{t, \hat{t}\}$. Since x_i is input into the accumulator at time t *and* time $\hat{t}$, any path through the trellis T that represents a valid encoding would use a switch-edge at time step t, *and* at time step $\hat{t}$. In general, any path representing a valid encoding would do the same thing (as in switch levels or remain on the same level) at every time step $t \in X_i$. We say a path is *agreeable* for x_i if it has this property for x_i. An *agreeable path* is a path that is agreeable for all x_i. Any path that is *not* agreeable does not represent a valid encoding, and thus finding the lowest cost path is not guaranteed to return a valid encoder path.

What we would like to find is the lowest cost *agreeable* path from s_0^0 to s_n^0. We give a simple integer program based on min-cost flow that solves this problem.

RALP: Repeat-Accumulate Linear Program. For each node s in the trellis, define $\delta(s)$ to be the set of outgoing edges from s, and $\gamma(s)$ to be the set of incoming edges. Our integer program contains variables $f(e) \in \{0, 1\}$ for every edge e in the trellis, and free variables z_i for every information bit x_i, $i \in \{0, \ldots, n-1\}$. The relaxation RALP of the integer program simply relaxes the flow variables such that $0 \leqslant f(e) \leqslant 1$. RALP is defined as follows:

$$\text{RALP:} \quad \min \sum_{e \in T} c[e] f(e) \quad \text{s.t.}$$

$$\sum_{e \in \gamma(s_n^0)} f(e) = 1 \tag{1}$$

$$\sum_{e \in \gamma(s)} f(e) = \sum_{e \in \delta(s)} f(e) \quad \forall\, s \in T \setminus \{s_0^0, s_n^0\} \tag{2}$$

$$f(s_t^0, s_{t+1}^1) + f(s_t^1, s_{t+1}^0) = z_i \quad \forall\, X_i \in \mathcal{X},\ t \in X_i \tag{3}$$

$$0 \leqslant f(e) \leqslant 1 \quad \forall\, e \in T$$

RALP is very close to being a simple min-cost flow LP: Equation (1) gives a demand of one unit of flow at the sink node s_n^0, and equation (2) is a flow conservation constraint at each node. Unique to RALP are the *agreeability constraints* (3). These constraints say that a feasible flow must have, for all $X_i \in \mathcal{X}$, the same amount z_i of total flow on switch-edges at every segment $t \in X_i$. Note that these constraints also imply a total flow of $1 - z_i$ on remain-edges at every segment $t \in X_i$. We will refer to the flow values f of a feasible solution (f, z) to RALP as an *agreeable flow*. The free variables z_i do not play a role in the objective function, but rather enforce constraints among the flow values.

Using RALP as a decoder. A decoding algorithm based on RALP is as follows. Run an LP-solver to find the optimal solution (f^*, z) to RALP, setting the costs on the edges according to the received word v. If f^* is integral, output z as the decoded information word x. If not, output "error." We will refer to this algorithm as the *RALP decoder*.

All integral solutions to RALP represent agreeable paths, and thus valid encodings of some information word. This implies that if the optimal solution f^* to RALP is in fact integral, then f^* is the lowest cost agreeable path, and z represents the ML code word. Thus the RALP decoder has the

ML certificate property: whenever it outputs an information word, it is guaranteed to be the ML information word. No standard iterative decoding techniques are known to have this property.

3 A Coding Theorem for RA(2) Codes

In this section we state our main structural theorem (proven in section 4). We then show how this theorem suggests a design for an interleaver for RA(2) codes, and prove an inverse polynomial upper bound on the RALP decoder's word error probability when we use this interleaver.

Our main structural theorem states that the RALP decoder succeeds (returns the original information word) if and only if a particular graph does not contain a certain negative-cost subgraph, generalizing the role of a negative-cost cycle in min-cost flow. The graph has a structure that depends on the interleaver π and weights that depend on the errors made by the channel. We note that an analogous theorem holds for any RA(ℓ) code, or any turbo-like code for that matter. This is discussed further in Section 5.

For the remainder of this section we deal exclusively with RA(2) codes. This means that each set $X_i \in \mathcal{X}$ has two elements, and the agreeability constraints may be expressed as, $\forall X_i \in \mathcal{X}$, $X_i = \{t, \hat{t}\}$,

$$z_i = f(s^0_t, s^1_{t+1}) + f(s^1_t, s^0_{t+1}) = f(s^0_{\hat{t}}, s^1_{\hat{t}+1}) + f(s^1_{\hat{t}}, s^0_{\hat{t}+1}).$$

The Promenade. Let $G = (V_G, E_G)$ be a weighted undirected graph with n nodes $(g_0, \ldots, g_{n-1})$ connected in a line, where the cost $c[g_t, g_{t+1}]$ of edge (g_t, g_{t+1}) is -1 if the t^{th} bit of the transmitted code word is flipped by the channel ($v_t \neq y_t$), and $+1$ otherwise. Call these edges the Hamiltonian edges, since they make a Hamiltonian path. Note that these costs are not known to the decoder, since they depend on the transmitted code word. For each $\{t, \hat{t}\} \in \mathcal{X}$, add an edge between node g_t and node $g_{\hat{t}}$ with cost 0. Call these edges the matching edges. Note that G is a line plus a matching.

Define a *promenade* to be a circuit in G that begins and ends in the same node, and may repeat edges as long as it does not travel along the same edge twice in a row. The cost of a promenade is the total cost of the edges visited during the path, including repeats (i.e., repeats are not free). Formally, a promenade is a path $M = (p_0, p_1, \ldots, p_{|M|} = p_0)$ in G that begins and ends at the same node p_0, where for all $i \in \{0, \ldots, |M|-1\}$, $p_i \neq p_{i+2 \mod |M|}$. The cost of a promenade M is $c[M] = \sum_{i=0}^{|M|-1} c[p_i, p_{i+1}]$. We are now ready to state our main structural theorem.

Theorem 1 *The RALP decoder succeeds if all promenades in G have positive cost. The RALP decoder fails if there is a promenade in G with negative cost.*

When there is a zero-cost promenade, the RALP decoder may or may not decode correctly (this is a degenerate case when the LP has multiple optima). We will prove Theorem 1 in Section 4, but first we show what it suggests about interleaver design, and how it can used to prove a bound on the probability of error of our algorithm.

Recall that every Hamiltonian edge of G has cost -1 with some small constant probability p, so promenades with many edges are less likely to have a total negaitve cost (at least every other edge of a promenade is Hamiltonian). The *girth* of a graph is the length of its shortest simple cycle. It is not hard to see that every promenade contains at least one simple cycle, and so graphs with high girth will have promenades with many edges. This suggests that what we want out of an interleaver, if we are to use the RALP decoder, is one that produces a graph G with high girth.

We use a result of Erdös and Sachs [7] and Sauer [15] (see also [4]) to make a graph that is a line plus a matching, and has high girth. Their construction allows us, in cubic time, to start with an n-node cycle and build a 3-regular graph G with girth $\lfloor \log n \rfloor$ that contains the original cycle. We remove an edge from the original cycle to obtain a line plus a matching with girth at least as high. To derive the interleaver, we simply examine the edges added by the construction. We will refer to this interleaver as π_E. This is the same construction used by Bazzi *et al.* [2] to show a $1/2 \log n$ lower bound on the minimum distance of an RA code using this interleaver.

Theorem 2 **[2]** *The rate 1/2 - $o(1)$ RA code with block length n, using π_E as an interleaver, has minimum distance of at least $\frac{1}{2}\log n$.*

Error Bound. We would like to bound the probability that G contains a promenade with cost less than or equal to zero, thus bounding the probability that the RALP decoder fails. However, there are many promenades in G, so even a tight bound on the probability that a particular one is negative is insufficient. Furthermore, the fact that promenades may repeat edges creates dependencies that interfere with using standard Chernoff bound analysis. Consequently, we need to find a simpler structure that is present when there is a promenade with cost less than or equal to zero. In the following, a simple path or cycle means a path or cycle that does not repeat edges. For clarity (to avoid floors and ceilings), we will assume n is a power of 16, though our arguments do not rely on this assumption.

Lemma 3 *If $n \geqslant 2^4$, and there exists a promenade M in G, $c[M] \leqslant 0$, then there exists a simple path or cycle Y in G, $c[Y] \leqslant 0$, that contains $\frac{1}{2}\log n$ Hamiltonian edges.*

Proof: Let $M = (p_0, p_1, \ldots, p_{|M|-1}, p_{|M|} = p_0)$ be a promenade in G, $c[M] \leqslant 0$. Since M contains a cycle,

$|M| \geqslant \log n$. Contract every matching edge in M to obtain a tour $H = (h_0, h_1, \ldots, h_{|H|} = h_0\}$. No two matching edges share an endpoint, so at worst every other edge of the tour is a matching edge. Thus, $|H| \geqslant \frac{1}{2}|M| \geqslant \frac{1}{2}\log n$. Let $c[H] = \sum_{i=0}^{|H|-1} c[h_i, h_{i+1}]$. We know that $h_i \neq h_{i+2}$, $0 \leqslant i < |H|$ (operations on indices of $h \in H$ are performed mod $|H|$); if this was not the case, then there would be a cycle in G of length 3, violating our assumption that $n \geqslant 2^4$, since the girth of G is at least $\log n$. Let $H_i = (h_i, \ldots, h_{i+\frac{1}{2}\log n})$ be the subsequence of H whose edge set contains $\frac{1}{2}\log n$ edges, and let $c[H_i] = \sum_{j=i}^{i+\frac{1}{2}\log n - 1} c[h_j, h_{j+1}]$. We know that H_i has no repeated edges (i.e., it is a simple path or a simple cycle); if it were not, then if we added the matching edges back into H we would get a path in G with at most $\log n$ edges that repeated an edge, which would imply a cycle in G of length less than $\log n$.

Since all matching edges have zero cost, $c[M] = c[H]$. Note that $c[H] = (\sum_{i=0}^{|H|-1} c[H_i])/(\frac{1}{2}\log n)$, since every edge of H is counted in the sum exactly $(\frac{1}{2}\log n)$ times. Since $c[H] = c[M] \leqslant 0$, at least one simple path or cycle H_{i^*} must have $c[H_{i^*}] \leqslant 0$. Set Y to be the simple path or cycle in G obtained by expanding the zero-cost matching edges in the path or cycle H_{i^*}. ■

Theorem 1, along with the above construction of G and a union bound over the paths of length $\frac{1}{2}\log n$ of G, gives us an analytical bound on the probability of error when decoding with the RALP decoder:

Theorem 4 *For any $\epsilon > 0$, the rate $1/2 - o(1)$ RA code with block length n using π_E as an interleaver decoded with the RALP decoder under the binary symmetric channel with crossover probability $p < 2^{-4(\epsilon+(\log 24)/2)}$ has a word error probability P_w of at most $n^{-\epsilon}$.*

Proof: By Theorem 1, and Lemma 3, the RALP decoder succeeds if all simple paths or cycles in G with $\frac{1}{2}\log n$ Hamiltonian edges have positive cost. We claim that there are at most $n \cdot 3^{\frac{1}{2}\log n}$ different simple paths that have $\frac{1}{2}\log n$ Hamiltonian edges. To see this, consider building a path by choosing a node, and building a simple path from that node. At each point, there are at most three choices for the next Hamiltonian edge.

The remainder of the proof is a union bound over the paths of G with $\frac{1}{2}\log n$ Hamiltonian edges. Let Y be a particular path with $\frac{1}{2}\log n$ Hamiltonian edges. Each Hamiltonian edge has cost -1 or $+1$, so at least half of the Hamiltonian edges must have cost -1 in order for $c[Y] \leqslant 0$. Each Hamiltonian edge had cost -1 with probability p, so the probability that $c[Y] \leqslant 0$ is at most $\binom{\frac{1}{2}\log n}{\frac{1}{4}\log n} p^{\frac{1}{4}\log n}$. By the union bound,

$$\begin{aligned} P_w &\leqslant n 3^{\frac{1}{2}\log n}\binom{\frac{1}{2}\log n}{\frac{1}{4}\log n} p^{\frac{1}{4}\log n} \\ &\leqslant n^{1+\frac{1}{2}\log 3} n^{\frac{1}{2}} 2^{-(\epsilon+\frac{1}{2}\log 24)\log n} \\ &\leqslant n^{1+\frac{1}{2}\log 3+\frac{1}{2}-(\epsilon+\frac{1}{2}\log 24)} \\ &\leqslant n^{-\epsilon} \end{aligned}$$

■

4 Proof of Theorem 1

Theorem 1 *The RALP decoder succeeds if all promenades in G have positive cost. The RALP decoder fails if there is a promenade in G with negative cost.*

In this section we prove Theorem 1. We assume a working knowledge of the *min-cost flow* problem, specifically the notions of a *residual graph*, a *circulation* and a *path decomposition* [1].

Let f^0 be the unit flow along the path P_x taken by the encoder. The RALP decoder will succeed if f^0 is the unique optimal solution to RALP. The RALP decoder will fail if f^0 is not an optimal solution to RALP. To prove the theorem, we will first establish conditions under which f^0 is optimal that are very similar to the conditions under which a normal flow is optimal; specifically, we will show that f^0 is optimal if and only if the residual graph T_{f^0} does not contain a certain negative-cost subgraph. We will then show a cost-preserving correspondence between these subgraphs in T_{f^0} and promenades in G, the Hamiltonian line graph on which Theorem 1 is based.

Agreeable Circulations. Suppose that f is some feasible flow for a normal min-cost flow problem (without agreeability constraints) on an arbitrary graph R. Let $c[f]$ be the cost of f, and R_f be the residual graph of f. A solution f is optimal iff R_f does not contain a negative-cost circulation.

Now consider the agreeable flow problem (RALP) on the trellis T, and the solution f^0 to RALP. Define the residual graph T_{f^0} in the same manner as in normal min-cost flow. Now consider some other feasible agreeable flow $f' \neq f^0$ in T. The circulation $f' - f^0$ in T_{f^0} has cost $c[f'] - c[f^0]$. Since both f^0 and f' obey the agreeability constraints, the circulation $f' - f$ also obeys the agreeability constraints. We call such a circulation an *agreeable circulation*.

Lemma 5 *The RALP decoder succeeds if all agreeable circulations in T_{f^0} have positive cost.*

Proof: Suppose the RALP decoder fails, then f^0 is not the unique optimal solution to RALP. Let $f^* \neq f^0$ be some optimal solution to RALP. Consider the circulation $f' = f^* - f^0$. Since $c[f'] = c[f^*] - c[f^0]$, and $c[f^*] \leqslant c[f^0]$, we know that $c[f'] \leqslant 0$. It is clear that the sum or difference of two agreeable flows is agreeable, so f' is agreeable. ■

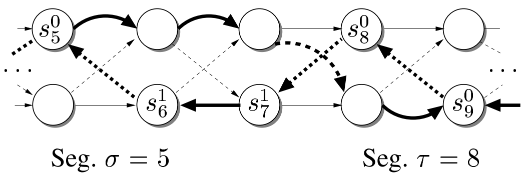

Figure 2. An example of a cycle $w(\sigma = 5, \tau = 8)$ in the residual graph T_{f^0}. The cycle (in bold) contains a subpath (straight bold lines) backwards along residual edges of P_x, where $P_x = (\ldots, s_5^0, s_6^1, s_7^1, s_8^0, s_9^0, \ldots)$. It also contains the complimentary forwards path from s_5^0 to s_9^0 (curved bold lines).

Lemma 6 *The RALP decoder fails if there exists an agreeable circulation in T_{f^0} with negative cost.*

Proof: Let f' be the circulation in T_{f^0} with negative cost. Let $f^* = f^0 + f'$. Since $c[f'] < 0$, $c[f^*] < c[f^0]$. By a similar argument as in Lemma 5, f^* is an agreeable flow. Since f^* is a feasible solution to RALP with cost less than f^0, f^0 is not optimal, thus the RALP decoder fails. ■

Agreeable Circulations and Promenades. In the remainder of the proof, we show a correspondence between agreeable circulations in T_{f^0} and promenades in G. We first define a special class of cycles in T_{f^0} and show that the cost of a cycle in this class is the same as that of a corresponding subpath in G. Then we show that every simple cycle in T_{f^0} is from this class. We conclude the proof by arguing that agreeable circulations always contain sets of these cycles that correspond to promenades in G with the same cost.

Let $g(\sigma, \tau)$, $\sigma < \tau$, be the path $(g_\sigma, g_{\sigma+1}, \ldots, g_\tau)$ of Hamiltonian edges in G. Now let $w(\sigma, \tau)$, $\sigma < \tau$, be a certain cycle in T_{f^0}, as depicted in Figure 2. We define $w(\sigma, \tau)$ formally as follows. Let $(\alpha[0] = 0, \alpha[1], \ldots, \alpha[n] = 0)$ be the sequence of states of the accumulator when encoding x (note that $y_t = \alpha[t+1]$). For example, in Figure 2, $\alpha[\sigma] = 0, \alpha[\sigma+1] = 1, \alpha[\sigma+2] = 1, \alpha[\tau] = 0, \alpha[\tau+1] = 1$. We have that $P_x = (s_0^{\alpha[0]}, s_1^{\alpha[1]}, \ldots, s_{n-1}^{\alpha[n-1]}, s_n^{\alpha[n]})$, the path in T taken by the encoder. In T_{f^0}, all edges along P_x have a unit residual capacity going backwards towards s_0^0, since f^0 is the unit flow across path P_x. The cycle $w(\sigma, \tau)$ in T consists of the backward subpath $(s_{(\tau+1)}^{\alpha[\tau+1]}, s_\tau^{\alpha[\tau]}, \ldots, s_{(\sigma+1)}^{\alpha[\sigma+1]}, s_\sigma^{\alpha[\sigma]})$, and the forward subpath $(s_\sigma^{\alpha[\sigma]}, s_{\sigma+1}^{1-\alpha[\sigma+1]}, \ldots, s_\tau^{1-\alpha[\tau]}, s_{\tau+1}^{\alpha[\tau+1]})$.

For $\tau > \sigma$, we let $g(\tau, \sigma) = g(\sigma, \tau)$ and $w(\tau, \sigma) = w(\sigma, \tau)$. We will also use the notation λw to represent a circulation in T_f that sends λ units of flow around a residual cycle w. For a set $\mathcal{W}$ of cycles w in T_{f^0}, we say that $\lambda\mathcal{W}$ is the circulation $\sum_{w \in \mathcal{W}} \lambda w$.

Claim 7 $c[g(\sigma, \tau)] = c[w(\sigma, \tau)]$, $0 \leqslant \sigma < \tau < n$.

Proof: The cost of the cycle $w(\sigma, \tau)$ is the sum of the costs of its residual edges:

$$\begin{aligned} c[w(\sigma,\tau)] = \quad & c[s_\sigma^{\alpha[\sigma]}, s_{\sigma+1}^{1-\alpha[\sigma+1]}] \\ &+ \sum_{t=\sigma+1}^{\tau-1} c[s_t^{1-\alpha[t]}, s_{t+1}^{1-\alpha[t+1]}] \\ &+ c[s_\tau^{1-\alpha[\tau]}, s_{\tau+1}^{\alpha[\tau+1]}] \\ &- \sum_{t=\sigma}^{\tau} c[s_t^{\alpha[t]}, s_{t+1}^{\alpha[t+1]}] \end{aligned}$$

Let e be the indicator vector for errors occurring in the channel. In other words, for all t, $0 \leqslant t < n$, let $e_t = 1$ if $y_t \neq v_t$, and $e_t = 0$ otherwise. Note that every edge $(\,\cdot\,, s_{t+1}^{\alpha[t+1]})$ in T entering a node on P_x has $c[\,\cdot\,, s_{t+1}^{\alpha[t+1]}] = e_t$. Similarly, every edge $(\,\cdot\,, s_{t+1}^{1-\alpha[t+1]})$ has $c[\,\cdot\,, s_{t+1}^{1-\alpha[t+1]}] = 1 - e_t$. Thus we may conclude

$$\begin{aligned} c[w(\sigma,\tau)] &= (1-e_\sigma) + \left(\sum_{t=\sigma+1}^{\tau-1} 1 - e_t\right) + e_\tau - \sum_{t=\sigma}^{\tau} e_t \\ &= \sum_{t=\sigma}^{\tau-1} 1 - 2e_t \\ &= c[g(\sigma,\tau)]. \end{aligned} \tag{4}$$

Equation (4) follows from the definition of G. ■

Claim 8 *For every simple cycle C in T_{f^0}, $C = w(\sigma, \tau)$ for some $0 \leqslant \sigma < \tau < n$.*

Proof: Let s_τ^α be the node on the cycle C that is the "closest" to s_n^0, i.e., τ is maximum among all nodes on C. The node s_τ^α must be on P_x; otherwise, the outgoing edge from s_τ^α on the cycle C would go forwards in the trellis to some node $s_{\tau+1}^\beta$. Follow C backwards along P_x until it diverges from P_x at some node s_σ^γ. The only remaining simple path from s_σ^γ to s_τ^α (that does not reuse an edge) is forward in the trellis along the edges that complete the cycle $w(\sigma, \tau)$. ■

Lemma 9 *If there is a promenade in G with negative cost, there is an agreeable circulation in T_{f^0} with negative cost.*

Proof: Let M be a promenade in G with negative cost. Let $1/\lambda$ be the maximum number of occurrences of any single Hamiltonian edge in M. If the matching edges along M are removed, what remains is a multiset of subpaths of the Hamiltonian path of G, of the form $g(\sigma, \tau)$. Let $\mathcal{U} = \{g(\sigma_0, \tau_0), \ldots, g(\sigma_{|\mathcal{U}|}, \tau_{|\mathcal{U}|})\}$ be this set of subpaths, ordered by their occurrence during a traversal of M. In other words, one may traverse M by following the path

$g(\sigma_0, \tau_0)$, matching edge $(g_{\tau_0}, g_{\sigma_1})$, path $g(\sigma_1, \tau_1)$, ..., path $g(\sigma_{|\mathcal{U}|}, \tau_{|\mathcal{U}|})$, and finally matching edge $(g_{\tau_{|\mathcal{U}|}}, g_{\sigma_0})$

Let $\mathcal{W} = \{w(\sigma_0, \tau_0), \ldots, w(\sigma_{|\mathcal{U}|}, \tau_{|\mathcal{U}|})\}$ be the set of corresponding cycles in T_{f^0}. No edge in $\lambda\mathcal{W}$ has more than one unit of flow sent across it (by the definition of λ), so $\lambda\mathcal{W}$ is a feasible circulation in T_{f^0}.

Since all the matching edges of G have zero cost, and M has negative cost, we have $c[M] = \sum_{j=0}^{|\mathcal{U}|} c[g(\sigma_j, \tau_j)] < 0$. Therefore by Claim 7, $\sum_{j=0}^{|\mathcal{W}|} c[w(\sigma_j, \tau_j)] < 0$. It follows that $c[\lambda\mathcal{W}] < 0$.

It remains to show that $\lambda\mathcal{W}$ is agreeable. Consider the first cycle in $\mathcal{W}$, namely $w(\sigma_0, \tau_0)$. As Figure 2 shows, the only agreeability constraints violated by sending flow around this cycle are at trellis segments σ_0 and τ_0; all segments in-between have the same amount of flow on switch-edges as they did before.

Assume wlog that P_x has a remain-edge at segment τ_0, so $\lambda w(\sigma_0, \tau_0)$ increases the flow on switch-edges by λ at segment τ_0. Since $(g_{\tau_0}, g_{\sigma_1})$ is a matching edge (by definition of $\mathcal{U}$), $\{\tau_0, \sigma_1\} \in \mathcal{X}$. Therefore P_x must obey the agreeability constraint at segments τ_0 and σ_1; in other words, it must also have a remain-edge at segment σ_1. It follows that $\lambda w(\sigma_1, \tau_1)$ increases the flow on switch-edges by λ at segment σ_1. Therefore the agreeability constraint for information bit x_i is preserved by the circulation $w(\sigma_0, \tau_0) + w(\sigma_1, \tau_1)$, and the only violated agreeability constraints are at segments σ_0 and τ_1. A symmetric argument holds if P_x has a switch-edge at segment τ_0.

By repeatedly applying the same argument along $\mathcal{W}$, we may conclude that the circulation $\lambda\mathcal{W}$ only violates agreeability constraints at σ_0 and $\tau_{|\mathcal{W}|}$. Since $\{\sigma_0, \tau_{|\mathcal{W}|}\} \in \mathcal{X}$, we have that $\lambda\mathcal{W}$ does not violate any agreeability constraints. ■

Lemma 10 *If all promenades in G have positive cost, all agreeable circulations in T_{f^0} have positive cost.*

Proof: Suppose f' is an agreeable circulation in T_{f^0} where $c[f'] < 0$. Let λ be the gcd of all the values of flow on edges of f'. Consider the cycle decomposition of f' where every cycle has λ units of flow around it. Let $\mathcal{W}$ be the collection of cycles in this decomposition. By claim 8, every cycle in T_{f^0} is of the form $w(\sigma, \tau)$. For all t, $0 \leqslant t < n$, let $B_t = \{w(\sigma, \tau) \in \mathcal{W} : \sigma = t \text{ or } \tau = t\}$.

For some set A of cycles $w(\sigma, \tau)$, we use the notation $f \in A$ to denote the flow $\lambda w(\sigma, \tau)$ for some arbitrary cycle $w(\sigma, \tau) \in A$.

Consider some $X_i = \{t, \hat{t}\}$. Let $z_t = f'(s_t^0, s_{t+1}^1) + f'(s_t^1, s_{t+1}^0)$. Because each cycle $w(\sigma, \tau) \in \mathcal{W}$ only affects the agreeability constraints at segments σ and τ, $z_t = \sum_{f \in B_t} \lambda f(s_t^0, s_{t+1}^1) + \lambda f(s_t^1, s_{t+1}^0)$. If $x_i = 0$, then for all $f \in B_t$, f has λ units of flow on one of the edges (s_t^0, s_{t+1}^1) or (s_t^0, s_{t+1}^1), and zero units on the other one. Thus f will add λ to z_t in the above sum. If $x_i = 1$, then f has $-\lambda$ flow on either (s_t^0, s_{t+1}^1) or (s_t^0, s_{t+1}^1), and contributes $-\lambda$ to the sum. Therefore,

$$z_t = \begin{cases} \lambda|B_t| & \text{if } x_i = 0 \\ -\lambda|B_t| & \text{if } x_i = 1 \end{cases}$$

Define $z_{\hat{t}}$ similarly. By the same logic,

$$z_{\hat{t}} = \begin{cases} \lambda|B_{\hat{t}}| & \text{if } x_i = 0 \\ -\lambda|B_{\hat{t}}| & \text{if } x_i = 1 \end{cases}$$

Since f' is agreeable, $z_t = z_{\hat{t}}$. Thus we may conclude that $|B_t| = |B_{\hat{t}}|$. For all $\{t, \hat{t}\} \in \mathcal{X}$, create a one-to-one correspondence between the members of B_t and $B_{\hat{t}}$ (we can do this because the sets are the same size).

Create an auxiliary multigraph H with a node $v[w(\sigma, \tau)]$ for each $w(\sigma, \tau) \in \mathcal{W}$. Add edges according to the correspondence we just created for each $\{t, \hat{t}\} \in \mathcal{X}$. Note that if $w(t, \hat{t}) \in \mathcal{W}$, it would be in both B_t and $B_{\hat{t}}$. In this case, the correspondence may assign $v[w(t, \hat{t})]$ to itself; we represent this by a self-loop on $v[w(t, \hat{t})]$.

Each $w(\sigma, \tau) \in \mathcal{W}$ is in exactly two sets: B_σ and B_τ. Therefore, H is 2-regular, a collection $\mathcal{Y}$ of simple cycles (where a node with a self-loop is considered a cycle). For a cycle $Y \in \mathcal{Y}$, let $\mathcal{W}_Y = \{w(\sigma, \tau) \in \mathcal{W} : v[w(\sigma, \tau)] \in Y\}$. The set $\{\mathcal{W}_Y : Y \in \mathcal{Y}\}$ constitutes a partition of $\mathcal{W}$ into subsets, so $f' = \sum_{f \in \mathcal{W}} f = \sum_{Y \in \mathcal{Y}} \sum_{f \in \mathcal{W}_Y} f$. Since $c[f'] < 0$, there must be some $Y^* \in \mathcal{Y}$ such that $\sum_{f \in \mathcal{W}_{Y^*}} c[f] < 0$. It follows that $\sum_{w(\sigma,\tau) \in \mathcal{W}_{Y^*}} c[w(\sigma, \tau)] < 0$.

We build a promenade M in G by following the cycle Y. We begin with an arbitrary node $v[w(\sigma_0, \tau_0)]$, and add the edges of the path $g(\sigma_0, \tau_0)$ to M. We then follow an edge in H to a node $v[w(\sigma_1, \tau_1)]$, where $\{\tau_0, \sigma_1\} \in \mathcal{X}$, by definition of H. When we follow this edge, we add the matching edge (τ_0, σ_1) to M, then the path $g(\sigma_1, \tau_1)$. We continue this way until we complete the cycle Y, and thus close the promenade M.

Let $\mathcal{U}$ be the set of subpaths $g(\sigma, \tau)$ we added to M while following Y. Matching edges have zero cost, so $c[M] = \sum_{g(\sigma,\tau) \in \mathcal{U}} c[g(\sigma, \tau)]$. Since $\sum_{w(\sigma,\tau) \in \mathcal{W}_{Y^*}} c[w(\sigma, \tau)] < 0$, we have $c[M] = \sum_{g(\sigma,\tau) \in \mathcal{U}} c[g(\sigma, \tau)] < 0$ by Claim 7. Thus M is a negative-cost promenade, and we have a contradiction. ■

Theorem 1 is implied by Lemmas 5, 6, 9 and 10. ■

5 Generalization to Concatenated Codes

In this section we give a generic LP to decode a repetition code concatenated with any other inner rate-1 code, with an interleaver between them. The LP can be generalized further to apply to any set of parallel or serially concatenated codes of any rate, connected by interleavers. We defer this general form to a later version, though it is not difficult to derive.

We model the inner code using a simple graph (trellis) that represents each input string by a path from a start node to an end node. Any rate-1 binary code can be modeled this way, though for some codes, the trellis could require exponential size. So, the LP is only of polynomial size if the code can be described by a polynomial-sized trellis. Any convolutional code (see [19]) has a simple linear-size trellis.

Let trellis T be a directed graph with the following properties: **(i)** There is a specified start node s_0. For all other nodes s in T, all paths from s_0 to s have equal length. Let L_t be the set of nodes distance t from s_0. **(ii)** Nodes in L_n have no outgoing edges. All other nodes have two outgoing edges: an "input-0" edge, and an "input-1" edge. Let S_t be the set of input-1 edges leaving nodes in L_t. **(iii)** Each edge is labeled with a code bit `0` or `1`.

Let π be a permutation $\pi_j : \{0, \ldots, n-1\} \rightarrow \{0, \ldots, n-1\}$. The code (T, π) encodes an information word x of length k as follows. **(i)** Let x' be a binary string of length n, where $x'_t = x_{\lfloor \pi^{-1}(t)/\ell \rfloor}$. **(ii)** From the start node s_0 of T, follow a path in T using bits from x': on step t of the path, follow the "input-0" edge if $x'_t = 0$, and follow the "input-1" edge if $x'_t = 1$. Concatenate the labels on the edges of the path to obtain a code word y of length n.

We define the linear program TCLP as follows. As in RALP, we have a flow varaible $f(e)$ for each edge in the trellis T. We also have free variables z_i for each information bit x_i. The cost $c[e]$ of an edge e entering a node from L_t is the Hamming distance between the label on the edge and the received bit v_t. For each node s in T, define $\delta(s)$ to be the set of outgoing edges from s, and $\gamma(s)$ to be the set of incoming edges. For all $i \in \{0, \ldots, k-1\}$, let X_i be the set of indicies to which information bit x_i was repeated and permuted, i.e., $X_i = \{\pi(\ell i), \pi(\ell i + 1), \ldots, \pi(\ell i + \ell - 1)\}$. Let $\mathcal{X} = \{X_i : i \in \{0, \ldots, k-1\}\}$.

$$\text{TCLP:} \quad \min \sum_{e \in T} c[e] f(e) \quad \text{s.t.}$$

$$\sum_{e \in \delta(s_0)} f(e) = 1$$

$$\sum_{e \in \gamma(s)} f(e) = \sum_{e \in \delta(s)} f(e) \quad \forall\, s \in T, s \neq s_0, s \notin L_n$$

$$\textstyle\sum_{e \in S_t} f(e) = z_i \quad \forall\, X_i \in \mathcal{X}, t \in X_i$$

$$0 \leqslant f(e) \leqslant 1 \quad \forall\, e \in T$$

A decoder based on TCLP has the ML certificate property for the same reasons the RALP decoder did. It is also not hard to derive a generalized "promenade" structure for TCLP, and prove a theorem analogous to Theorem 1 for this class of codes. The challenge then is to prove that for some interleaver, it is unlikely that there will be a "promenade" with cost less than or equal to zero. It would be interesting to see a general construction to derive interleavers for this class of codes.

6 Future Work

Improving the Running Time The obvious drawback of our approach to decoding is the complexity of solving a linear program. Even though interior point methods run in polynomial-time (and the simplex algorithm often faster), most applications of error-correcting codes require a more efficient decoding algorithm. There are two possible solutions to this problem, and both have some important unanswered questions.

The first option is to try and solve RALP or TCLP combinatorially. For the case of RA(2) codes, the resulting agreeable flow problem can be reduced to an instance of normal min-cost flow, and thus yields a more efficient combinatorial algorithm. We leave the details for a later version. It is an interesting open question as to whether combinatorial solutions exist for RA(ℓ), $\ell \geqslant 3$, or other codes.

The agreeable flow problem has a more general formulation that could apply to areas outside of coding theory. We define the *min-cost agreeable flow problem* as follows:

Min-Cost Agreeable Flow: Given a directed network $G = (V, E)$, a source $s \in V$ and a sink $t \in V$ with a demand α, capacities $u : E \rightarrow \mathbb{R}^+$ and costs $c : E \rightarrow \mathbb{R}$ on the edges, and a sequence of edge sets $\mathcal{A} = \{A_1, \hat{A}_1, A_2, \hat{A}_2, \ldots, A_m, \hat{A}_m\}$, find a minimum-cost α-unit flow f from s to t, where $\forall (A_i, \hat{A}_i)$, the total flow going through arcs in A_i is equal to the total flow going through arcs in $\hat{A}_i$.

Any LP with a constraint matrix made up of $\{+1, -1\}$ can be expressed using only the agreeability constraints of the above formulation, so we would not expect to be able to solve *Min-Cost Agreeable Flow* combinatorially in its full generality. However, the specialized structure of RALP or TCLP may allow a combinatorial solution. In general, it is an interesting question to determine how the min-cost agreeable flow problem must be restricted in order to make it solvable combinatorially.

The second option for improving efficiency is to use an iterative decoder. We showed (with Wainwright [8]) that the iterative TRMP algorithm of Wainwright, Jaakkola and Willsky [18], when applied to the problem of decoding turbo-like codes, outputs exactly the solution to TCLP. We have done some testing of this algorithm applied to various codes, and it seems to converge reasonably well. However, it is not known whether this algorithm converges in general, or how long it takes to converge when it does.

Improving the Error Bounds. The RA code with rate $1/2 - o(1)$ that we were able to analyze completely is not the best code experimentally in the literature. We are currently trying to understand the combinatorics behind more complicated codes such as a rate 1/3 RA code, and the classic turbo code (parallel concatenated convolutional code). In order to provide better bounds for these codes, we need to prove that negative-cost "promenade"-like subgraphs are unlikely. Theorem 1 suggested a design for an interleaver for the rate $1/2 - o(1)$ RA code. It would be interesting to see if other design suggestions can be derived for more complex turbo-like codes.

Interpreting Iterative Decoders as Optimization Algorithms. We have already begun the work of connecting the world of iterative decoders to our LP decoder [8]. However, preliminary testing suggests that BP algorithms outperform our LP decoder. The reason for this difference is not well understood. If we were able to interpret standard BP decoders as global optimization routines, perhaps we would be able to bound their error rate using the same techniques.

Using the Polytope for ML Decoding. We observed through experimentation that the integrality gap of TCLP applied to RA codes was usually very good, and so the solution to TCLP provided a good lower bound for use in a branch-and-bound search for the ML code word. Although not guaranteed to run in polynomial time, our branch-and-bound search finished much sooner than a naive ML decoding algorithm would have. More details of these experiments will appear in a later version.

More applications of LP-based decoding. Our analysis of the error probability is over the random coin flips of the channel; in the context of TCLP, this translates to the probability, in a fixed polytope, over a random *direction* of the objective function, that the LP solution was integral. It would be interesting to write LPs for other codes (such as low-density parity check codes or expander-based codes) and apply the same sort of analysis. Perhaps this sort of analysis could have other applications outside of coding theory.

Acknowledgements

We would like to thank Piotr Indyk, Martin Wainwright and Matthias Ruhl for helpful discussions. We also thank Dan Spielman, David Forney, Ibrahim Abou-Faycal, Matteo Frigo, Vanu Bose, John Chapin, Andrew Russell, Madhu Sudan, Ryan O'Donnell, Muriel Medard and Matt Levine for their comments, suggestions, and support.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice-Hall, 1993.

[2] L. Bazzi, M. Mahdian, S. Miller, and D. Spielman. The minimum distance of turbo-like codes. *manuscript*, 2001.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon limit error-correcting coding and decoding: turbo-codes. *Proc. IEEE International Conference on Communication (ICC), Geneva, Switzerland*, pages 1064–1070, May 1993.

[4] N. Biggs. Constructions for cubic graphs with large girth. *Electronic Journal of Combinatorics*, 5(A1), 1998.

[5] M. Breiling and J. Huber. Upper bound on the minimum distance of turbo codes using a combinatorial approach, 2001.

[6] D. Divsalar, H. Jin, and R. McEliece. Coding theorems for 'turbo-like' codes. *Proc. 36th Annual Allerton Conference on Comm., Control, and Computing*, pages 201–210, Sept 1998.

[7] P. Erdös and H. Sachs. Regulare graphen gegebene taillenweite mit minimaler knotenzahl. *Wiss. Z. Univ. Hall Martin Luther Univ. Halle–Wittenberg Math.– Natur.Reine*, 12:251–257, 1963.

[8] J. Feldman, D. Karger, and M. Wainwright. Linear programming based decoding of turbo-like codes and its relation to iterative approaches. *Proc. 40th Annual Allerton Conference on Comm., Control, and Computing*, October 2002.

[9] G. Forney. Convolutional codes II: Maximum likelihood decoding. *Inform. Control*, 25:222–266, 1974.

[10] N. Kahale and R. Urbanke. On the minimum distance of parallel and serially concatenated codes. *IEEE International Symposium on Information Theory*, 1998.

[11] F. MacWilliams and N. Sloane. *The Theory of Error Correcting Codes*. North-Holland, 1981.

[12] R. McEliece, D.MacKay, and J. Cheng. Turbo decoding as an instance of Pearl's belief propagation algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):140–152, 1998.

[13] T. Richardson, M. A. Shokrollahi, and R. L. Urbanke. Design of capacity-approaching irregular low-density parity-check codes. *Information Theory, IEEE Transactions on*, 47(2):619–637, 2001.

[14] T. Richardson and R. Urbanke. The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory*, 47(2), February 2001.

[15] N. Sauer. Extremaleigenschaften regularer graphen gegebener taillenweite, iandii. *Sitzungsberichte Osterreich. Acad. Wiss. Math. Natur. Kl.*, S-BII(176):27–43, 1967.

[16] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, IT-13:260–269, Apr. 1967.

[17] B. Vucetic and J. Yuan. *Turbo Codes*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[18] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. MAP estimation via agreement on (hyper)trees: Message-passing and linear programming approaches. MIT LIDS Tech. report, July 2002.

[19] S. Wicker. *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, Englewood Cliffs, NJ, 1995.