Towards Combinatorial LP Turbo Decoding

Michael Helmling and Stefan Ruzika
Mathematical Institute
University of Koblenz-Landau, Campus Koblenz
56070 Koblenz, Germany
Email: {helmling, ruzika}@uni-koblenz.de

***Abstract*—We present a novel algorithm that solves the turbo code LP decoding problem in a finite number of steps by Euclidean distance minimizations, which in turn rely on repeated shortest path computations in the trellis graph representing the turbo code. Previous attempts to exploit the combinatorial graph structure only led to algorithms which are either of heuristic nature or do not guarantee finite convergence. A numerical study shows that our algorithm clearly beats the running time, up to a factor of 100, of generic commercial LP solvers for medium-sized codes, especially for high SNR values.**

***Index Terms*—LP decoding, turbo codes, combinatorial optimization**

I. Introduction

Linear Programming based channel decoding [1] has gained tremendous interest because of its analytical power combined with noteworthy error-correcting performance. We consider LP decoding of turbo codes [2], a class of concatenated convolutional codes that, together with a simple iterative decoding algorithm, feature remarkable error-correcting performance.

Compared to LDPC codes, which the majority of publications in the area of LP decoding now focus on, turbo codes have some analytical advantages, most importantly the inherent combinatorial structure by means of the trellis graph representations of the underlying convolutional encoders. ML Decoding of turbo codes is closely related to shortest path and minimum network flow problems, both being well-studied topics in optimization theory. The hardness of ML decoding is caused by additional conditions on the path through the trellis graphs posed by the turbo code's interleaver. Thus ML (LP) decoding is equivalent to solving a (LP-relaxed) shortest path problem with additional linear side constraints.

So far, two methods for solving the LP have been proposed: General purpose LP solvers like CPLEX [3] are based on the matrix representation of the LP and do not exploit structural properties of the specific problem. Lagrangian relaxation with subgradient optimization [1], [4], on the other hand, utilizes this structure, but has practical limitations, most notably it usually converges rather slowly.

This paper presents a new approach to solve the LP decoding problem exactly by an algorithm that exploits its graphical substructure, thus combining the analytical power of the LP approach with the running-time benefits of a combinatorial method, which seems to be a necessary requirement for practical implementation. Our basic idea is to construct an alternative polytope in the space defined by the additional constraints (called *constraints space*) and show how the LP solution corresponds to a specific point z_{LP}^{Q} of that polytope. Then, we show how to computationally find z_{LP}^{Q} by a geometric algorithm that relies on a sequence of shortest path computations in the trellis graphs.

The reinterpretation of constrained optimization problems in constraints space was first developed in [5], where it is applied to minimum spanning tree and constrained shortest path problems with a single side constraint. In 2010, Tanatmis [6] applied this theory to the turbo decoding problem. His algorithm showed a drastic speedup compared to a general purpose LP solver, however it only works for up to two constraints, while in real-world turbo codes the number of constraints equals the information length.

By adapting an algorithm by Wolfe [7] to compute in a polytope the point with minimum Euclidean norm, we are able to overcome these limitations and decode turbo codes with lengths of practical interest. Compared to previous methods, the algorithm is advantageous not only in terms of running time, but also gives valuable information that can help to improve the error-correcting performance. Furthermore, branch-and-bound methods for integer programming-based ML decoding depend upon fast lower bound computations, mostly given by LP relaxations, and can often be significantly improved by dedicated methods that evaluate combinatorial properties of the LP solutions. Since our LP decoder contains such information, it could also be considered a step towards IP-based algorithms with the potential of practical implementation.

An extended version of this paper including all proofs and a more extensive numerical analysis has been published on arXiv [8].

II. Background and Notation

A. *Definition of Turbo Codes*

For the sake of clear notation, we focus on turbo codes as used in the LTE standard [9]—i. e., systematic, parallely concatenated turbo codes with two identical terminated rate-1 constituent encoders. An in-depth covering of coding theory basics and turbo code construction can be found in [10].

An (n, k) turbo code $TC = TC(C, \pi)$ is defined by a rate-1 convolutional (n_C, k) code C with constraint length d and a permutation $\pi \in \mathbb{S}_k$ such that $n = k + 2 \cdot n_C$. Here C is a terminated code where the "tail" of the information sequence is a part of the output; thus $n_C = k + 2 \cdot d$ and the code rate is slightly below 1. Let $e_C : \mathbb{F}_2^k \longrightarrow \mathbb{F}_2^{n_C}$ be the associated encoding function. Then, the encoding function of TC is defined by $e_{TC}(x) = (x \mid e_C(x) \mid e_C(\pi(x)))$ where $\pi(x) = (x_{\pi(1)}, \ldots, x_{\pi(k)})$.

B. Trellis Graphs of Convolutional Codes

A convolutional code with a specific length is represented naturally by its trellis graph, which is obtained by unfolding the code-defining finite state machine in the time domain. The following description of convolutional codes loosely follows [11, Section V.C].

We denote a trellis by $T = (V, E)$ with vertex set V and edge set E. Vertices are indexed by time step and state; i. e., $v_{i,s}$ denotes the vertex corresponding to state $s \in \{0, \ldots, 2^d - 1\}$ at time $i \in \{1, \ldots, k + d + 1\}$. An edge in turn is identified by the time and state of its tail vertex plus its input label, so $e_{i,s,b}$ denotes the edge emerging from $v_{i,s}$ with input bit $b \in \{0, 1\}$. The trellis *segment* at time i is $S_i = (V_i, E_i)$ where $V_i = \{v_{j,s} \in V : j \in \{i, i + 1\}\}$ and $E_i = \{e_{j,s,b} \in E : j = i\}$.

By construction, the paths from the starting node to the end node in a trellis of a convolutional code C are in one-to-one correspondence with the codewords of C. Moreover, for each such path $P = (e_1, \ldots, e_{k+d})$ and each $e \in E$ an index set $J_C(e)$ can be computed with the property that $e \in P \Rightarrow y_j = 1$ for all $j \in J_C(e)$.

The path-codeword relation can be exploited for maximum likelihood (ML) decoding, if the codewords are transmitted through a memoryless binary-input output-symmetric (MBIOS) channel: Let $\lambda \in \mathbb{R}^{k+2\cdot d}$ be the vector of LLR values of the received signal. If we assign to each edge $e \in E$ the cost $c(e) = \sum_{j \in J_C(e)} \lambda_j$, it can be shown [12] that the shortest (or minimum cost) path in T corresponds to the ML codeword.

C. Trellis Representation of Turbo Codes

For turbo codes, we have two isomorphic trellis graphs, T^1 and T^2, according to the two component convolutional encoders. Let formally $T = (G^1 \cup G^2, E^1 \cup E^2)$, and by $P = P^1 \circ P^2$ denote the path that consists of P^1 in T^1 and P^2 in T^2. An *agreeable* path $P_1 \circ P_2 = (e^1_1, \ldots, e^1_{k+d}, e^2_1, \ldots, e^2_{k+d})$ must obey the k *consistency constraints*

$$e^1_i \in I^1_i \;\Leftrightarrow\; e^2_{\pi(i)} \in I^2_{\pi(i)} \quad \text{for } i = 1, \ldots, k$$

because both encoders operate on the same information word, only that it is permuted for the second encoder. Consequently, ML decoding for turbo codes can be formulated as finding the shortest agreeable path in T. If an agreeable path contains $e^1_i \in I^1_i$, it must also contain $e^2_{\pi(i)} \in I^2_{\pi(i)}$, and thus $i \in J_C(e)$ for both e^1_i and $e^2_{\pi(i)}$. To avoid counting the LLR value λ_i twice in the objective function, we use the modified cost

$$c(e) = \sum_{j \in J_C(e)} \hat{\lambda}_j \text{ with } \hat{\lambda}_j = \begin{cases} \frac{\lambda_j}{2} & \text{if } 1 \le j \le k, \\ \lambda_j & \text{otherwise.} \end{cases} \tag{1}$$

It is straightforward to formulate ML decoding as an integer linear program by introducing a binary flow variable $f_e \in \{0, 1\}$ for each $e \in E^1 \cup E^2$ with the usual flow conservation and capacity constraints [13] that define the path polytopes $\mathcal{P}^1_{\text{path}}$ and $\mathcal{P}^2_{\text{path}}$, respectively. We obtain

$$\begin{aligned} \text{(TC-IP)} \qquad \min \; & \sum_{e \in E^1 \cup E^2} c(e) \cdot f_e \\ \text{s. t.} \; & f^1 \in \mathcal{P}^1_{\text{path}},\ f^2 \in \mathcal{P}^2_{\text{path}} && (2) \\ & \sum_{e \in I^1_i} f_e = \sum_{e \in I^2_{\pi(i)}} f_e \qquad i = 1, \ldots, k && (3) \\ & f_e \in \{0, 1\},\ e \in E. && (4) \end{aligned}$$

D. Polyhedral Theory Background

Besides coding theory, this paper requires some bits of polyhedral theory. A *polytope* is the convex hull of a finite number of points: $\mathcal{P} = \operatorname{conv}(v_1, \ldots, v_n)$. It can be described either by its *vertices* (or *extreme points*), i. e., the unique minimal set fulfilling this defining property, or as the intersection of a finite number of halfspaces: $\mathcal{P} = \bigcap_{i=1}^m \{x : a_i^T x \le b_i\}$. An inequality $a^T x \le b$ is called *valid* for $\mathcal{P}$ if it is true for all $x \in \mathcal{P}$. In that case, the set $F_{a,b} = \{x \in \mathcal{P} : a^T x = b\}$ is called the *face induced by* the inequality. For any r satisfying $a^T r \ge b$ ($a^T r > b$) we say that the inequality *separates* (*strongly separates*) r from $\mathcal{P}$.

III. The LP Relaxation and Conventional Solution Methods

ML decoding of general linear block codes is known to be **NP**-complete [14]. The computational complexity of ML turbo decoding is still open, but it is widely believed that TC-IP is **NP**-complete as well. By relaxing (4) to $f_e \in [0, 1]$, we get the LP relaxation TC-LP of TC-IP, which in contrast can be solved efficiently by standard Linear Programming algorithms [15]. Feldman *et al.* [1] were the first to analyze this relaxation and attested it reasonable decoding performance.

A general purpose LP solver, however, does not make use of the combinatorial substructure contained in TC-IP via (2) and thus wastes some potential of solving the problem more efficiently—while LPs are solvable in polynomial time, they do not scale too well, and the number of variables and constraints in TC-LP becomes very large for codes of practical interest.

Note that without the consistency constraints (3), we could solve TC-LP by simply computing shortest paths in both trellis graphs, which is possible in time $\mathcal{O}(k+d)$. A popular approach for solving optimization problems that comprise "easy" subproblem plus some "complicating" additional constraints is to solve the Lagrangian dual by subgradient optimization [16]. With g_i defined suitably, (3) can be compactly rewritten as $g_i(f) = 0$ for $i = 1, \ldots, k$. The *Lagrangian relaxation with* *multiplier* $\mu \in \mathbb{R}^k$ is defined as

$$\begin{aligned}\text{(TC-LR)} \quad z(\mu) = \min \; & \sum_{e \in E^1 \cup E^2} c(e) \cdot f_e + \sum_{i=1}^{k} \mu_k \cdot g_i(f) \\ \text{s.t.} \quad & f^1 \in \mathcal{P}^1_{\text{path}} \\ & f^2 \in \mathcal{P}^2_{\text{path}} \\ & f_e \in \{0,1\},\, e \in E.\end{aligned}$$

For all $\mu \in \mathbb{R}^k$, the objective value of TC-LR is smaller or equal to that of TC-LP. The *Lagrangian dual problem* is to find multipliers μ that maximize this objective, thus minimizing the gap to the LP solution. It can be shown that in the optimal case both values coincide. Note that the feasible region of TC-LR is the combined path polytope of both T_1 and T_2, so it can be solved by a shortest path routine in both trellises with modified costs. Applying Lagrangian relaxation to turbo decoding was already proposed in [1] and further elaborated in [4].

The Lagrangian dual is typically solved by a subgradient algorithm that iteratively adjusts the multipliers μ, converging (under some mild conditions) to the optimal value [16]. However, the convergence is often slow in practice and the limit is not guaranteed to be ever reached exactly, leaving it an approximate method. Additionally, the dual only informs us about the objective value; recovering the actual solution of the problem requires additional work. The main result of this paper is an alternative algorithm which exhibits none of these drawbacks.

IV. An Equivalent Problem in Constraints Space

Like Lagrangian dualization, our algorithm also uses a relaxation of TC-IP with modified objective function that resembles TC-LR. However, via geometric interpretation of the image of the path polytope in the "constraints space", as defined below, the exact LP solution is found in finitely many steps.

A. The Image Polytope $\mathcal{Q}$

Let $\mathcal{P}_{\text{path}} = \mathcal{P}^1_{\text{path}} \times \mathcal{P}^2_{\text{path}}$ be the feasible region of TC-LR. We define the map

$$\begin{aligned}\mathfrak{D} : \mathcal{P}_{\text{path}} &\to \mathbb{R}^{k+1} \\ f &\mapsto (g_1(f), \dots, g_k(f), c(f))^T\end{aligned}$$

where $c(f) = \sum_{e \in E^1 \cup E^2} c(e) \cdot f_e$ is a short hand for the objective function value of TC-LP. For a path f, the first k coordinates $v_i, i = 1, \dots, k$, of $v = \mathfrak{D}(f)$ tell if and how the condition $g_i(f) = 0$ is violated, while the last coordinate v_{k+1} equals the cost of f. Let $\mathcal{Q} = \mathfrak{D}(\mathcal{P}_{\text{path}})$ be the image of the path polytope under $\mathfrak{D}$. The following results are immediate:

Lemma 1:

1) $\mathcal{Q}$ is a polytope.
2) If f is an agreeable path in T, then $\mathfrak{D}(f)$ lies on the $(k+1)$st axis (henceforth called c-axis or A_c).
3) If v is a vertex of $\mathcal{Q}$ and $v = \mathfrak{D}(f)$ for some $f \in \mathcal{P}_{\text{path}}$, then f is also a vertex of $\mathcal{P}_{\text{path}}$.

In the situation that $v = \mathfrak{D}(f)$ we will also write $f = \mathfrak{D}^{-1}(v)$ with the meaning that f is *any* preimage of v.

We consider the auxiliary problem

$$\begin{aligned}\text{(TC-LP}_{\mathcal{Q}}) \quad z^{\mathcal{Q}}_{\text{LP}} = \; & \min v_{k+1} \\ \text{s.t.} \quad & v \in \mathcal{Q} \\ & v \in A_c \qquad (5)\end{aligned}$$

the solution of which is the lower "piercing point" of the axis A_c through $\mathcal{Q}$. Note that due to (5), k of the $k+1$ variables in TC-LP($\mathcal{Q}$) are fixed to zero, thus the feasible region is the (one-dimensional) projection of $\mathcal{Q}$ onto A_c. Nevertheless, the following theroem shows that TC-LP$_{\mathcal{Q}}$ and TC-LP are essentially equivalent.

Theorem 1: Let v_{LP} be an optimal solution of TC-LP$_{\mathcal{Q}}$ with objective value $z^{\mathcal{Q}}_{\text{LP}}$ and $f_{\text{LP}} = \mathfrak{D}^{-1}(v_{\text{LP}}) \in \mathcal{P}_{\text{path}}$ the corresponding flow. Then $z^{\mathcal{Q}}_{\text{LP}} = z_{\text{LP}}$, the optimal objective value of TC-LP, and f_{LP} is an optimal solution of TC-LP.

While we do not have an explicit representation of $\mathcal{Q}$—by means of either vertices or inequalities—at hand, we can easily minimize linear functionals over $\mathcal{Q}$: The problem

$$\begin{aligned}\text{(LP}_{\mathcal{Q}}) \qquad & \min \gamma^T v \\ \text{s.t.} \quad & v \in \mathcal{Q}\end{aligned}$$

can be solved by first computing an optimal solution f^* of

$$\begin{aligned}\text{(TC-WS)} \qquad & \min \sum_{i=1}^{k} \gamma_i \cdot g_i(f) + \gamma_{k+1} \cdot c(f) \qquad (6) \\ \text{s.t.} \quad & f \in \mathcal{P}_{\text{path}}\end{aligned}$$

and then taking the image of f^* under $\mathfrak{D}$. As noted before, this can be achieved within running time $\mathcal{O}(n)$.

B. Solving TC-LP$_{\mathcal{Q}}$ with Nearest Point Calculations

Our algorithm solves TC-LP$_{\mathcal{Q}}$ by a series of nearest point computations between $\mathcal{Q}$ and reference points r^i on A_c, the last of which gives a face of $\mathcal{Q}$ containing the optimum v_{LP}.

For each $r \in \mathbb{R}^{k+1}$, we denote the nearest point to r in $\mathcal{Q}$ by $\text{NP}(r) = \arg\min_{v \in \mathcal{Q}} \|v - r\|_2$ the nearest point to r in $\mathcal{Q}$ and define $a(r) = r - \text{NP}(r)$ and $b(r) = a(r)^T \text{NP}(r)$. Our proof relies on the following basic result.

Lemma 2: The inequality $a(r)^T v \le b(r)$ is valid for $\mathcal{Q}$ and induces a face containing $\text{NP}(r)$, which we call $\text{NF}(r)$.

Theorem 2: There exists an $\varepsilon > 0$ such that for all r inside the open line segment $\left(v_{\text{LP}}, v_{\text{LP}} - (0, \dots, 0, \varepsilon)^T\right)$ the condition $v_{\text{LP}} \in \text{NF}(r)$ holds.

The constructive proof of this theorem immediately leads to our algorithm. A shortened proof is shown below; for the complete version see the extended paper [8].

The idea is as follows: Start with a reference point $r \in A_c$ that is guaranteed to be located below v_{LP}. Then, iteratively compute $\text{NF}(r)$ and update r to be the intersection of A_c with the hyperplane defining $\text{NF}(r)$, until the stopping criterion $r = \text{NP}(r)$ is reached. We show that the sequence of reference points r is strictly increasing and that no face is repeated in the sequence of faces $\text{NF}(r)$, which guarantees termination.

The first result is that the hyperplane defining $\mathrm{NF}(r)$ is always oriented "downwards".

Lemma 3: Let $r = (0, \dots, 0, \rho)^T$ with $\rho < z_{\mathrm{LP}}^{\mathcal{Q}}$. Then $a(r)$ as defined above fulfills $a(r)_{k+1} < 0$.

Next we show that updating r always leads to a different face, unless we have arrived at the optimal solution.

Lemma 4: Under the same assumptions as in Lemma 3, let $s = (0, \dots, 0, \frac{b(r)}{a(r)_{k+1}})$ be the point where the separating hyperplane and A_c intersect. If $\mathrm{NF}(r) = \mathrm{NF}(s)$, then $s = v_{\mathrm{LP}}$.

The following lemma states that the part of A_c that lies below v_{LP} dissects into intervals such that reference points within one interval yield the same face of $\mathcal{Q}$.

Lemma 5: If $r^1 < r^2 < x_d^*$ and $\mathrm{NF}(r^1) = \mathrm{NF}(r^2)$, then $\mathrm{NF}(r) = \mathrm{NF}(r^1)$ for all $r \in [r^1, r^2]$.

Proof of Theorem 2: First we show that there exists at least one r with the claimed properties.

Choose some arbitrary $r^0 \in A_c$ with $r_{k+1}^0 < z_{\mathrm{LP}}^{\mathcal{Q}}$. If $v_{\mathrm{LP}} \in \mathrm{NF}(r^0)$, we are done. Otherwise, by Lemma 4 we find r^1 with $r_{k+1}^1 > r_{k+1}^0$ such that $\mathrm{NF}(r^1) \neq \mathrm{NF}(r^0)$. Iterating this argument and assuming that v_{LP} is never contained in the induced face results in a sequence $(r^i)_i$ with $r_{k+1}^{i+1} > r_{k+1}^i$ for all i. Because of Lemma 5, $\mathrm{NF}(r^{i+1}) \neq \mathrm{NF}(r^i)$ implies $\mathrm{NF}(r^{i+1}) \neq \mathrm{NF}(r^l)$ for all $0 \leq l < i+1$, so that all $\mathrm{NF}(r^i)$ are distinct, in contradiction to the fact that the number of faces of a polytope is finite.Thus eventually there must be an i^* such that $v_{\mathrm{LP}} \in \mathrm{NF}(r^{i^*})$.

Now let $r^* \in A_c$ be any such point whose existence we have just proven, $v_N = \mathrm{NP}(r^*)$ and $\lambda \in (0, 1]$. Let $\bar{r} = \lambda r^* + (1-\lambda) v_{\mathrm{LP}}$ and $\bar{v} = \lambda v_N + (1-\lambda) v_{\mathrm{LP}}$. It can be shown [8] that $(\bar{r} - \bar{v})^T v \leq (\bar{r} - \bar{v})^T \bar{v}$ induces $\mathrm{NF}(r^*)$. Because the above holds for any $0 < \lambda \leq 1$, we can choose $\bar{r}$ arbitrarily close to v_{LP} on A_c, which completes the proof. ∎

Algorithm 1 formalizes the procedure developed above in pseudocode. The initial reference point r^0 is generated by

Algorithm 1 Combinatorial Turbo LP Decoder (CTLP)

```
1: Initialize edge cost $c(f)$ by (1)
2: $f^0 \leftarrow \arg\min\{c(f) : f \in \mathcal{P}_{\text{path}}\}$.
3: $v^0 \leftarrow \mathfrak{D}(f^0)$.
4: $r^0 \leftarrow (0, \dots, 0, v_{k+1}^0)^T$
5: $i \leftarrow 0$
6: while $v^i \neq r^i$ do
7:     $v^{i+1} \leftarrow \mathrm{NP}(r^i) = \arg\min_{v \in \mathcal{Q}} \left\|v - r^i\right\|_2$
8:     $f^{i+1} = \mathfrak{D}^{-1}(v^{i+1})$
9:     $a^{i+1} \leftarrow v^{i+1} - r^i$
10:    $b^{i+1} \leftarrow a^{(i+1)T} v^{i+1}$
11:    $r^{i+1} \leftarrow \left(0, \dots, 0, \frac{b^{i+1}}{a_{k+1}^{i+1}}\right)^T$
12:    $i \leftarrow i + 1$
13: return $f^i$
```

first minimizing $c(f)$ on $\mathcal{P}_{\text{path}}$. Thereby we ensure that either $r^0 \notin \mathcal{Q}$ or it is located on the boundary of $\mathcal{Q}$, in which case it already is the optimal LP solution. The solution of the nearest point problem and the recovery of the original flow are encapsulated in Lines 7 and 8.

Table I: Average decoding CPU time per instance

SNR	0	1	2	3	4	5
(132,40) LTE Turbo code (values in seconds $\times 10^{-2}$)						
CPLEX	9.1	9.5	9.6	9.6	9.6	9.8
CTLP	1.4	0.9	0.5	0.29	0.24	0.22
ratio	6.5	10.6	19	33	40	45
(396,128) LTE Turbo code (values in seconds $\times 10^{-1}$)						
CPLEX	4.4	4.2	3.6	3.3	3.2	
CTLP	6.3	4.1	0.6	0.09	0.08	
ratio	0.7	1	6	37	40	
(384,128) 3-D turbo code (values in seconds)						
CPLEX	1.4	1.2	0.9	0.72	0.57	
CTLP	4.5	3.1	0.8	0.04	0.014	
ratio	0.31	0.39	1.1	18	41	

C. *Solving the Nearest Point Problems*

To solve the nearest point problems we utilize an algorithm by Wolfe [7] that finds in a polytope the point with minimum Euclidean norm. It elaborates on a set of vertices of the polytope that are obtained via minimization of linear objective functions. In our situation, this means that $\mathrm{LP}_{\mathcal{Q}}$, which is equivalent to TC-WS (6), has to be solved repeatedly. It also requires matrix operations that introduce an undesirable numerical component, though they can be implemented in a very efficient way. The algorithm is explicitly described in [8].

V. Numerical Results

A. *Running Time Comparison*

To evaluate the computational performance of our algorithm, we compare its running time with the commercial general purpose LP solver CPLEX [3] (which is said to be one of the most competitive implementations available) on LTE turbo codes with blocklengths 132 and 396, respectively, and a three-dimensional turbo code with blocklength 384 (taken from [17]). The results are shown Table I.

As one can see, the benefit of using the new algorithm is larger for high SNR values. This becomes most eminent for the 3-D code for which the dimension of $\mathcal{Q}$ is the highest. The reason for this behavior can be explained by analyzing statistical information about various internal parameters of the algorithm when run with different SNR values. Most notably, the average number of main loops (Line 6 of Algorithm 1) as well as the average dimension of the optimal nearest face, found in the last iteration, drop substantially with increasing SNR values. Confer [8] for a more detailed evaluation.

For comparison to Lagrangian dualization (LD), we let the subgradient algorithm (with parameters taken from [4]) terminate when the objective value changes by less than 10^{-6} in one step. Table II shows the relative (compared to CTLP) values for CPU time and the total number of shortest path computations for the $(132, 40)$ LTE code. In the low SNR region, LD needs to solve much more shortest path problems, which manifests also in the CPU time. For higher SNR values, the differences fade out, and the time spent by CTLP on the nearest point problems becomes significant. Note however that LD does not always find a codeword even if the LP solution

Table II: Relative performance of LD compared to CTSP

SNR	0	1	2	3	4	5
CPU time	16.4	15.1	7.7	4.1	1.3	1.1
SP problems	13.4	13.3	8.7	9.4	3.1	2.2

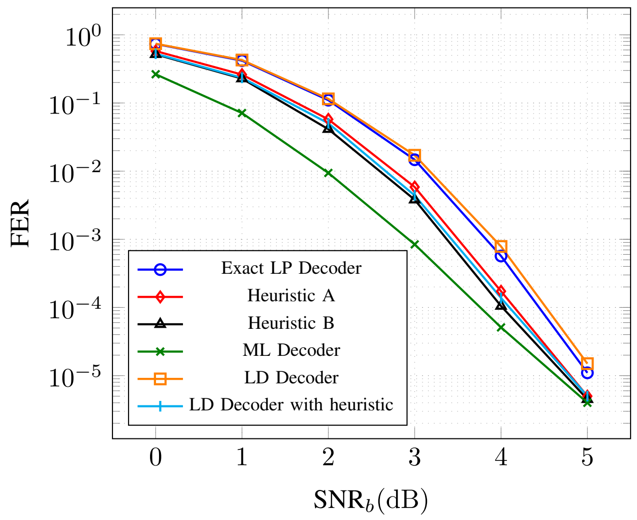

Figure 1: Decoding performance for the $(132, 40)$ LTE code

is optimal, leading to a slightly decreased FER (cf. Figure 1).

VI. Improving Error-Correcting Performance

Algorithm 1 can be modified to return a list of paths f_i, $i = 1, \dots, t$, such that the LP solution is a convex combination of that paths. Each f_i can be split into a paths f_i^1 and f_i^2 through trellis T^1 and T^2, respectively. A path in a single trellis, in turn, can uniquely be extended to a codeword. By selecting among the $2t$ candidate codewords the one with minimum objective value, we obtain a heuristic decoder (*Heuristic A* in the following) that always outputs a valid codeword.

A slightly better decoding performance, at the cost of increased running time, is reached if we consider not only the paths that constitute the final LP solution but rather *all* intermediate solutions of TC-WS (*Heuristic B*). A similar approach was proposed in [4] for LD. It exhibts a slightly decreased performance, despite the much larger number of shortest path computations.

Simulation results for the $(132, 40)$ LTE code are shown in Figure 1. The frame error rate indeed drops notably when using the heuristics, and at $5\,\text{dB}$, both heuristics come very close to ML performance.

VII. Conclusion and Outlook

We have shown how the inherent combinatorial network-flow structure of turbo codes in form of the trellis graphs can be utilized to construct a highly efficient LP solver, specialized for that class of codes. The decrease in running time, compared to a general purpose solver, is dramatic.

It is still an open question, however, if and how the LP can be solved in a *completely* combinatorial manner. The nearest point algorithm introduces a numerical component, which is necessary at this time but rather undesirable since it can lead to numerical problems if the dimension becomes too large.

Another direction for further research is to examine the usefulness of our decoder as a building block of branch-and-bound methods that solve the integer programming problem, i. e., ML decoders. Several properties of the decoder suggest that this might be a valuable task.

Finally, the concepts presented here might be of inner-mathematical interest as well. Problems that comprise an easy subproblem augmented by complicating constraints are very common in optimization. Being able to efficiently solve their LP relaxation is a key component of virtually all fast solution algorithms.

Acknowledgements

We would like to acknowledge the German Research Council (DFG), the German Academic Exchange Service (DAAD), and the Center for Mathematical and Computational Modelling of the University of Kaiserslautern for financial support.

References

[1] J. Feldman, D. R. Karger, and M. Wainwright, “Linear programming-based decoding of turbo-like codes and its relation to iterative approaches,” in *Proc. 40th Allerton Conf. Commun. Control Computing*, 2002.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding: Turbo-codes,” in *IEEE Int. Conf. Commun.*, May 1993, pp. 1064–1070.

[3] “IBM ILOG CPLEX optimization studio,” Software Package, 2011, version 12.4.

[4] A. Tanatmis, S. Ruzika, and F. Kienle, “A Lagrangian relaxation based decoding algorithm for LTE turbo codes,” in *Proc. Int. Symp. Turbo Codes and Iterative Inf. Proc.*, Brest, France, Sep. 2010, pp. 369–373.

[5] S. Ruzika, “On multiple objective combinatorial optimization,” Ph.D. dissertation, University of Kaiserslautern, 2007.

[6] A. Tanatmis, “Mathematical programming approaches for decoding of binary linear codes,” Ph.D. dissertation, University of Kaiserslautern, Kaiserslautern, Germany, Aug. 2010.

[7] P. Wolfe, “Finding the nearest point in a polytope,” *Math. Program.*, vol. 11, pp. 128–149, 1976.

[8] M. Helmling and S. Ruzika, “Towards an exact combinatorial algorithm for LP decoding of turbo codes,” arXiv:1301.6363 [cs.IT], Jan. 2013.

[9] *TS 36.212 v11.0.0: LTE E-UTRA Mutliplexing and Channel Coding*, 3rd Generation Partnership Project (3GPP) Std., Oct. 2012. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/136200_136299/136212/11.00.00_60/ts_136212v110000p.pdf

[10] S. Lin and D. Costello, Jr., *Error Control Coding*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 2004.

[11] M. Helmling, S. Ruzika, and A. Tanatmis, “Mathematical programming decoding of binary linear codes: Theory and algorithms,” *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4753–4769, Jul. 2012.

[12] J. Feldman, “Decoding error-correcting codes via linear programming,” Ph.D. dissertation, Massachusetts Institute of Technology, 2003.

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

[14] E. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, “On the inherent intractability of certain coding problems,” *IEEE Trans. Inf. Theory*, vol. 24, no. 3, pp. 954–972, 1978.

[15] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1986.

[16] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Sons, 1988.

[17] E. Rosnes, M. Helmling, and A. Graell i Amat, “Pseudocodewords of linear programming decoding of 3-dimensional turbo codes,” in *Proc. IEEE Int. Symp. Inform. Theory*, St. Petersburg, Russia, Jul. / Aug. 2011, pp. 1643–1647.