# Mathematical Programming Decoding of Binary Linear Codes: Theory and Algorithms 

Michael Helmling, Stefan Ruzika, Akın Tanatmis


#### Abstract

Mathematical programming is a branch of applied mathematics and has recently been used to derive new decoding approaches, challenging established but often heuristic algorithms based on iterative message passing. Concepts from mathematical programming used in the context of decoding include linear, integer, and nonlinear programming, network flows, notions of duality as well as matroid and polyhedral theory. This survey article reviews and categorizes decoding methods based on mathematical programming approaches for binary linear codes over binary-input memoryless symmetric channels.


Index Terms-Integer programming, LP decoding, Mathematical programming, ML decoding, Polyhedral theory.

## I. Introduction

Based on an integer programming (IP) ${ }^{1}$ formulation of the maximum likelihood decoding (MLD) problem for binary linear codes, linear programming decoding (LPD) was introduced by Feldman et al. [1], [2]. Since then, LPD has been intensively studied in a variety of articles especially dealing with low-density parity-check (LDPC) codes. LDPC codes are generally decoded by heuristic approaches called iterative message passing decoding (IMPD) subsuming sumproduct algorithm decoding (SPAD) [3], [4] and min-sum algorithm decoding (MSAD) [5]. In these algorithms, probabilistic information is iteratively exchanged and updated between component decoders. Initial messages are derived from the channel output. IMPD exploits the sparse structure of parity-check matrices of LDPC and turbo codes very well and achieves good performance. However, IMPD approaches are neither guaranteed to converge nor do they have the maximum likelihood (ML) certificate property, i.e., if the output is a codeword, it is not necessarily the ML codeword. Furthermore, performance of IMPD is poor for arbitrary linear block codes with a dense parity-check matrix. In contrast, LPD offers some advantages and thus has become an important alternative decoding technique. First, this approach is derived from the discipline of mathematical programming which provides analytical statements on convergence, complexity, and correctness of decoding algorithms. Second, LPD is not limited to sparse matrices.

This article is organized as follows. In Section II, notation is fixed and well-known but relevant results from coding theory and polyhedral theory are recalled. Complexity and

[^0]polyhedral properties of MLD are discussed in Section III. In Section IV a general description of LPD is given. Several linear programming (LP) formulations dedicated to codes with low-density parity-check matrices, codes with high-density parity-check matrices, and turbo-like codes are categorized and their commonalities and differences are emphasized in Section V. Based on these LP formulations, different streams of research on LPD have evolved. Methods focusing on efficient realization of LPD are summarized in Section VI, while approaches improving the error-correcting performance of LPD at the cost of increased complexity are reviewed in Section VII. Some concluding comments are made in Section VIII.

## II. Basics and Notation

This section briefly introduces a number of definitions and results from linear coding theory and polyhedral theory which are most fundamental for the subsequent text.

A binary linear block code $C$ with cardinality $2^{k}$ and block length $n$ is a $k$-dimensional subspace of the vector space $\{0,1\}^{n}$ defined over the binary field $\mathbb{F}_{2} . C \subseteq\{0,1\}^{n}$ is given by $k$ basis vectors of length $n$ which are arranged in a $k \times n$ matrix $G$, called the generator matrix of the code $C .^{2}$

The orthogonal subspace $C^{\perp}$ of $C$ is defined as

$$
C^{\perp}=\left\{y \in\{0,1\}^{n}: \sum_{j=1}^{n} x_{j} y_{j} \equiv 0(\bmod 2) \text { for all } x \in C\right\}
$$

and has dimension $n-k$. It can also be interpreted as a binary linear code of dimension $n-k$ which is referred to as the dual code of $C$. A matrix $H \in\{0,1\}^{m \times n}$ whose $m \geq n-k$ rows form a spanning set of $C^{\perp}$ is called a parity-check matrix of $C$. It follows from this definition that $C$ is the null space of $H$ and thus a vector $x \in\{0,1\}^{n}$ is contained in $C$ if and only if $H x \equiv 0(\bmod 2)$. Normally, $m=n-k$ and the rows of $H \in\{0,1\}^{(n-k) \times n}$ constitute a basis of $C^{\perp}$. It should be pointed out, however, that most LPD approaches (see Section VII) benefit from parity-check matrices being extended by redundant rows. Moreover, additional rows of $H$ never degrade the error-correcting performance of LPD. This is a major difference to IMPD which is generally weakened by redundant parity checks, since they introduce cycles to the Tanner graph.

[^1]Let $x, x^{\prime} \in\{0,1\}^{n}$. The Hamming distance between $x$ and $x^{\prime}$ is the number of entries (bits) with different values, i.e., $d\left(x, x^{\prime}\right)=\left|\left\{1 \leq j \leq n: x_{j} \neq x_{j}^{\prime}\right\}\right|$. The minimum (Hamming) distance of a code, $d(C)$, is given by $d(C)=$ $\min \left\{d\left(x, x^{\prime}\right): x, x^{\prime} \in C, x \neq x^{\prime}\right\}$. The Hamming weight of a codeword $x \in C$ is defined as $w(x)=d(x, 0)$, i.e., the number of ones in $x$. The minimum Hamming weight of $C$ is $w(C)=\min \{w(x): x \in C, x \neq 0\}$. For binary linear codes it holds that $d(C)=w(C)$. The error-correcting performance of a code is, at least at high signal-to-noise ratio (SNR), closely related to its minimum distance.

Let $A \in \mathbb{R}^{m \times n}$ denote an $m \times n$ matrix and $I=\{1, \ldots, m\}$, $J=\{1, \ldots, n\}$ be the row and column index sets of $A$, respectively. The entry in row $i \in I$ and column $j \in J$ of $A$ is given by $A_{i, j}$. The $i^{\text {th }}$ row and $j^{\text {th }}$ column of $A$ are denoted by $A_{i, .}$ and $A_{., j}$, respectively. A vector $e \in \mathbb{R}^{m}$ is called the $i^{\text {th }}$ unit column vector if $e_{i}=1, i \in I$, and $e_{h}=0$ for all $h \in I \backslash\{i\}$.

A parity-check matrix $H$ can be represented by a bipartite graph $G=(V, E)$, called its Tanner graph (Fig. 1). The vertex set $V$ of $G$ consists of the two disjoint node sets $I$ and $J$. The nodes in $I$ are referred to as check nodes and correspond to the rows of $H$ whereas the nodes in $J$ are referred to as variable nodes and correspond to columns of $H$. An edge $[i, j] \in E$ connects node $i$ and $j$ if and only if $H_{i, j}=1$. Let $N_{i}=$ $\left\{j \in J: H_{i j}=1\right\}$ denote the index set of variables incident to check node $i$, and analogously $N_{j}=\left\{i \in I: H_{i j}=1\right\}$ for $j \in J$. The degree of a check node $i$ is the number of

$$
H=\left(\begin{array}{llllllll}
1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{array}\right)
$$



Fig. 1. Parity-check matrix and Tanner graph of an $(8,4)$ code.
edges incident to node $i$ in the Tanner graph or, equivalently, $d_{c}(i)=\left|N_{i}\right|$. The maximum check node degree $d_{c}^{\max }$ is the degree of the check node $i \in I$ with the largest number of incident edges. The degree of a variable node $j, d_{v}(j)$, and the maximum variable node degree $d_{v}^{\max }$ are defined analogously.

Tanner graphs are an example of factor graphs, a general concept of graphical models which is prevalently used to describe probabilistic systems and related algorithms. The term stems from viewing the graph as the representation of some global function in several variables that factors into a product of subfunctions, each depending only on a subset of the variables. In case of Tanner graphs, the global function is the indicator function of the code, and the subfunctions are the parity-checks according to single rows of $H$. A different type of factor graphs will appear later in order to describe turbo codes. Far beyond these purely descriptive purpose, factor graphs have proven successful in modern coding theory
primarily in the context of describing and analyzing IMPD algorithms. See [6] for a more elaborate introduction.

Let $C$ be a binary linear code with parity-check matrix $H$ and $x \in C \subseteq\{0,1\}^{n}$. The index set $\operatorname{supp}(x)=\left\{j \in J: x_{j}=\right.$ $1\}$ is called the support of the codeword $x$. A codeword $0 \neq$ $x \in C$ is called a minimal codeword if there is no codeword $0 \neq y \in C$ such that $\operatorname{supp}(y) \subseteq \operatorname{supp}(x)$. Finally, $D$ is called a minor code of $C$ if $D$ can be obtained from $D$ by a series of shortening and puncturing operations.

The relationship between binary linear codes and polyhedral theory follows from the observation that a binary linear code can be considered a set of points in $\mathbb{R}^{n}$, i. e., $C \subseteq\{0,1\}^{n} \subseteq$ $\mathbb{R}^{n}$. In the following, some relevant results from polyhedral theory are recalled. For a comprehensive review on polyhedral theory the reader is referred to [7].

Definition II. 1 A subset $\mathcal{P}(A, b) \subseteq \mathbb{R}^{n}$ such that $\mathcal{P}(A, b)=$ $\left\{\nu \in \mathbb{R}^{n}: A \nu \leq b\right\}$ where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^{m}$ is called a polyhedron.

In this article, polyhedra are assumed to be rational, i. e., the entries of $A$ and $b$ are taken from $\mathbb{Q}$. The $i^{\text {th }}$ row vector of $A$ and the $i^{\text {th }}$ entry of $b$ together define a closed halfspace $\left\{\nu \in \mathbb{R}^{n}: A_{i, .} \nu \leq b_{i}\right\}$. In other words, a polyhedron is the intersection of a finite set of closed halfspaces. A bounded polyhedron is called a polytope. It is known from polyhedral theory that a polytope can equivalently be defined as the convex hull of a finite set of points. In this work, the convex hull of a binary linear code $C$ is denoted by $\operatorname{conv}(C)$ and referred to as the codeword polytope.

Some characteristics of a polyhedron are its dimension, faces, and facets. To define them, the notion of a valid inequality is needed.

Definition II. 2 An inequality $r^{T} \nu \leq t$, where $r \in \mathbb{R}^{n}$ and $t \in \mathbb{R}$, is valid for a set $\mathcal{P}(A, b) \subseteq \mathbb{R}^{n}$ if $\mathcal{P}(A, b) \subseteq\{\nu:$ $\left.r^{T} \nu \leq t\right\}$.

The following definition of an active inequality is used in several LPD algorithms.

Definition II. 3 An inequality $r^{T} \nu \leq t$, where $r, \nu \in \mathbb{R}^{n}$ and $t \in \mathbb{R}$, is active at $\nu^{*} \in \mathbb{R}^{n}$ if $r^{T} \nu^{*}=t$.

Valid inequalities which contain points of $\mathcal{P}(A, b)$ are of special interest.

Definition II. 4 Let $\mathcal{P}(A, b) \subseteq \mathbb{R}^{n}$ be a polyhedron, let $r^{T} \nu \leq$ $t$ be a valid inequality for $\mathcal{P}(A, b)$ and define $F=\{\nu \in$ $\left.\mathcal{P}(A, b): r^{T} \nu=t\right\}$. Then $F$ is called a face of $\mathcal{P}(A, b) . F$ is a proper face if $F \neq \emptyset$ and $F \neq \mathcal{P}(A, b)$.
The dimension $\operatorname{dim}(\mathcal{P}(A, b))$ of $\mathcal{P}(A, b) \subseteq \mathbb{R}^{n}$ is given by the maximum number of affinely independent points in $\mathcal{P}(A, b)$ minus one. Recall that a set of vectors $v^{1}, \ldots, v^{k}$ is affinely independent if the system $\left\{\sum_{i=1}^{k} \lambda_{k} v^{k}=0, \sum_{i=1}^{k} \lambda_{k}=0\right\}$ has no solution other than $\lambda_{i}=0$ for $i=1, \ldots, k$. If $\operatorname{dim}(\mathcal{P}(A, b))=n$, then the polyhedron is full-dimensional. It is a well-known result that if $\mathcal{P}(A, b)$ is not full-dimensional,
then there exists at least one inequality $A_{i, .} \nu \leq b_{i}$ such that $A_{i, .} \nu=b_{i}$ holds for all $\nu \in \mathcal{P}(A, b)$ (see e.g. [7]). Also, we have $\operatorname{dim}(F) \leq \operatorname{dim}(\mathcal{P}(A, b))-1$ for any proper face of $\mathcal{P}(A, b)$. A face $F \neq \emptyset$ of $\mathcal{P}(A, b)$ is called a facet of $\mathcal{P}(A, b)$ if $\operatorname{dim}(F)=\operatorname{dim}(\mathcal{P}(A, b))-1$.

In the set of inequalities defined by $(A, b)$, some inequalities $A_{i, .} \leq b_{i}$ may be redundant, i. e., dropping these inequalities does not change the solution set defined by $A \nu \leq b$. A standard result states that the facet-defining inequalities give a complete non-redundant description of a polyhedron $\mathcal{P}(A, b)$ [7].

A point $\nu \in \mathcal{P}(A, b)$ is called a vertex of $\mathcal{P}(A, b)$ if there exist no two other points $\nu^{1}, \nu^{2} \in \mathcal{P}(A, b)$ such that $\nu=$ $\mu_{1} \nu^{1}+\mu_{2} \nu^{2}$ with $0 \leq \mu_{1} \leq 1,0 \leq \mu_{2} \leq 1$, and $\mu_{1}+\mu_{2}=1$. Alternatively, vertices are zero dimensional faces of $\mathcal{P}(A, b)$. In an LP problem, a linear cost function is minimized on a polyhedron, i. e., $\min \left\{c^{T} x: x \in \mathcal{P}(A, b)\right\}, c \in \mathbb{R}^{n}$. Unless the LP is infeasible or unbounded, the minimum is attained on one of the vertices.

The number of constraints of an LP problem may be very large, e.g. Section V contains LPD formulations whose description complexity grows exponentially with the block length for general codes. In such a case it would be desirable to only include the constraints which are necessary to determine the optimal solution of the LP with respect to a given objective function. This can be accomplished by iteratively solving the associated separation problem, defined as follows.

Definition II. 5 Let $\mathcal{P}(A, b) \subset \mathbb{R}^{n}$ be a rational polyhedron and $\nu^{*} \in \mathbb{R}^{n}$ a rational vector. The separation problem is to either conclude that $\nu^{*} \in \mathcal{P}(A, b)$ or, if not, find a rational vector $(r, t) \in \mathbb{R}^{n} \times \mathbb{R}$ such that $r^{T} \nu \leq t$ for all $\nu \in \mathcal{P}(A, b)$ and $r^{T} \nu^{*}>t$. In the latter case, $(r, t)$ is called a valid cut.

We will see applications of this approach in Sections VI and VII.

There is a famous result about the equivalence of optimization and separation by Grötschel et al. [8].

Theorem II. 6 Let $\mathcal{P}$ be a proper class of polyhedra (see e.g. [7] for a definition). The optimization problem for $\mathcal{P}$ is polynomial time solvable if and only if the separation problem is polynomial time solvable.

## III. Complexity and Polyhedral Properties

In this section, after referencing important NP-hardness results for the decoding problem, we state useful properties of the codeword polytope, exploiting a close relation between coding and matroid theory.

Integer programming provides powerful means for modeling several real-world problems. MLD for binary linear codes is modeled as an IP problem in [2], [9]. Let $y \in \mathbb{R}^{n}$ be the channel output. In MLD the probability (or, in case of a continuous-output channel, the probability density) $P(y \mid x)$ is maximized over all codewords $x \in C$. Let $x^{*}$ denote the ML codeword. It is shown in [1] that for a symmetric memoryless channel the calculation of $x^{*}$ amounts to the minimization of
a linear cost function, namely

$$
\begin{equation*}
x^{*}=\underset{x \in C}{\arg \max } P(y \mid x)=\underset{x \in C}{\arg \min } \sum_{j=1}^{n} \lambda_{j} x_{j}, \tag{1}
\end{equation*}
$$

where the values $\lambda_{j}=\log \frac{P\left(y_{j} \mid x_{j}=0\right)}{P\left(y_{j} \mid x_{j}=1\right)}$ are the so-called loglikelihood ratios (LLR). Consequently the IP formulation of MLD is implicitly given as

$$
\begin{equation*}
\min \left\{\lambda^{T} x: x \in C\right\} \tag{2}
\end{equation*}
$$

Berlekamp et al. have shown that MLD is NP-hard in [10] by a polynomial-time reduction of the three-dimensional matching problem to the decision version of MLD. An alternative proof is via matroid theory: as shall be exposed shortly, there is a one-to-one correspondence between binary matroids and binary linear codes. In virtue of this analogy, MLD is equivalent to the minimum-weight cycle problem on binary matroids. Since the latter contains the max-cut problem, which is known to be NP-hard [11], as a special case, the NPhardness of MLD follows.

Another problem of interest in the framework of coding theory is the computation of the minimum distance of a given code. Berlekamp et al. [10] conjectured that computing the distance of a binary linear code is NP-hard as well, which was proved by Vardy [12] about two decades later. The minimum distance problem can again be reformulated in a matroid theoretic setting. In 1969 Welsh [13] formulated it as the problem of finding a minimum cardinality circuit in linear matroids.

In the following, we assume $C \subseteq\{0,1\}^{n}$ to be canonically embedded in $\mathbb{R}^{n}$ when referring to conv $(C)$ (see Fig. 2 for an example). Replacing $C$ by $\operatorname{conv}(C)$ in (2) leads to a linear programming problem over a polytope with integer vertices. In general, computing an explicit representation of $\operatorname{conv}(C)$ is intractable. Nevertheless, some properties of $\operatorname{conv}(C)$ are known from matroid theory due to the equivalence of binary linear codes and binary matroids. In the following, some definitions and results from matroid theory are presented. An extensive investigation of matroids can be found in [14] or [15]. The definition of a matroid in general is rather technical.

Definition III. 1 A matroid $\mathcal{M}$ is an ordered pair $\mathcal{M}=(J, \mathcal{U})$ where $J$ is a finite ground set and $\mathcal{U}$ is a collection of subsets of $J$, called the independent sets, such that $(a)-(c)$ hold.
(a) $\emptyset \in \mathcal{U}$.
(b) If $u \in \mathcal{U}$ and $v \subset u$, then $v \in \mathcal{U}$.
(c) If $u_{1}, u_{2} \in \mathcal{U}$ and $\left|u_{1}\right|<\left|u_{2}\right|$ then there exists $j \in u_{2} \backslash u_{1}$ such that $u_{1} \cup\{j\} \in \mathcal{U}$.

In this work, the class of $\mathbb{F}_{2}$-representable (i.e., binary) matroids is of interest. A binary $m \times n$ matrix $H$ defines an $\mathbb{F}_{2}$-representable matroid $\mathcal{M}[H]$ as follows. The ground set $J=\{1, \ldots, n\}$ is defined to be the index set of the columns of $H$. A subset $U \subseteq J$ is independent if and only if the column vectors $H_{., u}, u \in U$ are linearly independent in the vector space defined over the field $\mathbb{F}_{2}$. A minimal dependent set, i. e., a set $\mathcal{V} \in 2^{J} \backslash \mathcal{U}$ such that all proper subsets of $\mathcal{V}$


Fig. 2. The codewords of the single parity-check code $C=\left\{x \in \mathbb{F}_{2}^{3}\right.$ : $\left.x_{1}+x_{2}+x_{3} \equiv 0(\bmod 2)\right\}$ and the polytope $\operatorname{conv}(C)$ in $\mathbb{R}^{3}$.
are in $\mathcal{U}$, is called a circuit of $\mathcal{M}[H]$. If a subset of $J$ is a disjoint union of circuits then it is called a cycle.

The incidence vector $x^{\mathcal{C}} \in \mathbb{R}^{n}$ corresponding to a cycle $\mathcal{C} \subseteq J$ is defined by

$$
x_{j}^{\mathcal{C}}= \begin{cases}1 & \text { if } j \in C \\ 0 & \text { if } j \notin C\end{cases}
$$

The cycle polytope is the convex hull of the incidence vectors corresponding to all cycles of a binary matroid.

Some more relationships between coding theory and matroid theory (see also [16]) can be listed: a binary linear code corresponds to a binary matroid, the support of a codeword corresponds to a cycle (therefore, each codeword corresponds to the incidence vector of a cycle), the support of a minimal codeword corresponds to a circuit, and the codeword polytope $\operatorname{conv}(C)$ corresponds to the cycle polytope. Let $H$ be a binary matrix, $\mathcal{M}[H]$ be the binary matroid defined by $H$ ( $H$ is a representation matrix of $\mathcal{M}[H]$ ) and $C$ be the binary linear code defined by $H$ ( $H$ is a parity-check matrix of $C$ ). It can easily be shown that the dual $C^{\perp}$ of $C$ is the same object as the dual of the binary matroid $\mathcal{M}[H]$. We denote the dual matroid by $\mathcal{M}[G]$, where $G$ is the generator matrix of $C$. Usually the matroid related terms are dualized by the prefix "co". For example, the circuits and cycles of a dual matroid are called cocircuits and cocycles, respectively. The supports of minimal codewords and the supports of codewords in $C^{\perp}$ are associated with cocircuits and cocycles of $\mathcal{M}[H]$, respectively.

A minor of a parent matroid $\mathcal{M}=(J, \mathcal{U})$ is the submatroid obtained from $\mathcal{M}$ after any combination of contraction and restriction operations (see e.g. [14]). In the context of coding theory, contraction corresponds to puncturing, i. e., the deletion of one or more columns from the generator matrix of a parent code, and restriction corresponds to shortening, i.e., the deletion of one or more columns from the parity-check matrix of a parent code.

Next, some results from Barahona and Grötschel [17] which are related to the structure of the cycle polytope are rewritten in terms of coding theory. Kashyap provides a similar transfer in [18]. Several results are collected in Theorem III.2.

Theorem III. 2 Let $C$ be a binary linear code.
(a) If $d\left(C^{\perp}\right) \geq 3$ then the codeword polytope is fulldimensional.
(b) The box inequalities

$$
\begin{equation*}
0 \leq x_{j} \leq 1 \quad \text { for all } j \in J \tag{3}
\end{equation*}
$$

and the cocircuit inequalities

$$
\begin{align*}
& \sum_{j \in \mathcal{F}} x_{j}-\sum_{j \in \operatorname{supp}(q) \backslash \mathcal{F}} x_{j} \leq|\mathcal{F}|-1  \tag{4}\\
& \quad \text { for all } \mathcal{F} \subseteq \operatorname{supp}(q) \text { with }|\mathcal{F}| \text { odd, }
\end{align*}
$$

where $\operatorname{supp}(q)$ is the support of a dual minimal codeword $q$, are valid for the codeword polytope.
(c) The box inequalities $x_{j} \geq 0, x_{j} \leq 1$ define facets of the codeword polytope if $d\left(C^{\perp}\right) \geq 3$ and $j \in J$ is not contained in the support of a codeword in $C^{\perp}$ with weight three.
(d) If $d\left(C^{\perp}\right) \geq 3$ and $C$ does not contain $H_{7}^{\perp}((7,3,4)$ simplex code) as a minor, and if there exists a dual minimal codeword $q$ of weight 3 , then the cocircuit inequalities derived from $\operatorname{supp}(q)$ are facets of $\operatorname{conv}(C)$.

Part (b) of Theorem III. 2 implies that the set of cocircuit inequalities derived from the supports of all dual minimal codewords provide a relaxation of the codeword polytope. In the polyhedral analysis of the codeword polytope the symmetry property stated below plays an important role.

Theorem III. 3 [17] If $a^{T} x \leq \alpha$ defines a face of $\operatorname{conv}(C)$ of dimension $d$, and $y$ is a codeword, then the inequality $\bar{a}^{T} x \leq \bar{\alpha}$ also defines a face of $\operatorname{conv}(C)$ of dimension $d$, where

$$
\bar{a}_{j}=\left\{\begin{aligned}
a_{j} & \text { if } j \notin \operatorname{supp}(y) \\
-a_{j} & \text { if } j \in \operatorname{supp}(y)
\end{aligned}\right.
$$

and $\bar{\alpha}=\alpha-a^{T} y$.
Using this theorem, a complete description of $\operatorname{conv}(C)$ can be derived from all facets containing a single codeword [17].

Let $q$ be a dual minimal codeword. To identify if the cocircuit inequalities derived from $\operatorname{supp}(q)$ are facet-defining it should be checked if $\operatorname{supp}(q)$ has a chord. For the formal definition of chord, the symmetric difference $\triangle$ which operates on two finite sets is used, defined by $A \triangle B=(A \backslash B) \cup(B \backslash A)$. Note that if $A=\operatorname{supp}\left(q_{1}\right), B=\operatorname{supp}\left(q_{2}\right)$ and $\operatorname{supp}\left(q_{0}\right)=$ $A \triangle B$, then $q_{0} \equiv q_{1}+q_{2}(\bmod 2)$.

Definition III. 4 Let $q_{0}, q_{1}, q_{2} \in C^{\perp}$ be dual minimal codewords. If $\operatorname{supp}\left(q_{0}\right)=\operatorname{supp}\left(q_{1}\right) \triangle \operatorname{supp}\left(q_{2}\right)$ and $\operatorname{supp}\left(q_{1}\right) \cap$ $\operatorname{supp}\left(q_{2}\right)=\{j\}$, then $j$ is called a chord of $\operatorname{supp}\left(q_{0}\right)$.

Theorem III. 5 [17] Let $C$ be a binary linear code without the $(7,3,4)$ simplex code as a minor and let $\operatorname{supp}(q)$ be the support of a dual minimal codeword with Hamming weight at least 3 and without chord. Then for all $\mathcal{F} \subseteq \operatorname{supp}(q)$ with $|\mathcal{F}|$ odd, the inequality

$$
\sum_{j \in \mathcal{F}} x_{j}-\sum_{j \in \operatorname{supp}(q) \backslash \mathcal{F}} x_{j} \leq|\mathcal{F}|-1
$$

defines a facet of $\operatorname{conv}(C)$.
Optimizing a linear cost function over the cycle polytope, known as the cycle problem in terms of matroid theory, is investigated by Grötschel and Truemper [19]. The work of Feldman et al. [2] enables to use the matroid theoretic results in the coding theory context. As shown above, solving the

MLD problem for a binary linear code is equivalent to solving the cycle problem on a binary matroid. In [19], binary matroids for which the cycle problem can be solved in polynomial time are classified, based on Seymour's matroid decomposition theory [20]. Kashyap [16] shows that results from [19] are directly applicable to binary linear codes. The MLD problem as well as the minimum distance problem can be solved in polynomial time for the code families for which the cycle problem on the associated binary matroid can be solved in polynomial time. This code family is called polynomially almost-graphic codes [16].

An interesting subclass of polynomially almost-graphic codes are geometrically perfect codes. Kashyap translates the sum of circuits property (see [19]) to the realm of binary linear codes. If the binary matroid associated with code $C$ has the sum of circuits property then $\operatorname{conv}(C)$ can be described completely and non-redundantly by the box inequalities (3) and the cocircuit inequalities (4). These codes are referred to as geometrically perfect codes in [16]. The associated binary matroids of geometrically perfect codes can be decomposed in polynomial time into its minors which are either graphic (see [14]) or contained in a finite list of matroids.

From a coding theoretic point of view, a family of errorcorrecting codes is asymptotically bad if either dimension or minimum distance grows only sublinearly with the code length. Kashyap proves that the family of geometrically perfect codes unfortunately fulfills this property. We refer to [16] for the generalizations of this result.

## IV. Basics of LPD

LPD was first introduced in [2]. This decoding method is, in principle, applicable to any binary linear code over any binary input memoryless channel. ${ }^{3}$ In this section, we review the basics of the LPD approach based on [1].

Although several structural properties of $\operatorname{conv}(C)$ are known, it is in general infeasible to compute a concise description of conv $(C)$ by means of linear inequalities. In LPD, the linear cost function of the IP formulation is minimized on a relaxed polytope $\mathcal{P}$ where $\operatorname{conv}(C) \subseteq \mathcal{P} \subseteq \mathbb{R}^{n}$. Such a relaxed polytope $\mathcal{P}$ should have the following desirable properties:

- $\mathcal{P}$ should be easy to describe, and
- integral vertices of $\mathcal{P}$ should correspond to codewords.

Together with the linear representation (1) of the likelihood function, this leads to one of the major benefits of LPD, the so-called ML certificate property: If the LP decoder outputs an integral optimal solution, it is guaranteed to be the ML codeword. This is a remarkable difference to IMPD: If no general optimality condition applies (see e. g. [23, Sec. 10.3]), there is no method to provably decide the optimality of a solution obtained by IMPD.

Each row (check node) $i \in I$ of a parity-check matrix $H$ defines the local code

$$
C_{i}=\left\{x \in\{0,1\}^{n}: \sum_{j=1}^{n} H_{i j} x_{j} \equiv 0 \quad(\bmod 2)\right\}
$$

[^2]that consists of the bit sequences which satisfy the $i^{\text {th }}$ paritycheck constraint; these are called local codewords. A particularly interesting relaxation of $\operatorname{conv}(C)$ is
$$
\mathcal{P}=\operatorname{conv}\left(C_{1}\right) \cap \cdots \cap \operatorname{conv}\left(C_{m}\right) \subseteq[0,1]^{n},
$$
known as the fundamental polytope [24]. The vertices of the fundamental polytope, the so-called pseudocodewords, are a superset of $C$, where the difference consists only of nonintegral vertices. Consequently, optimizing over $\mathcal{P}$ implies the ML certificate property. These observations are formally stated in the following result (note that $C=C_{1} \cap \cdots \cap C_{m}$ ).

Lemma IV. 1 [24] Let $\mathcal{P}=\operatorname{conv}\left(C_{1}\right) \cap \cdots \cap \operatorname{conv}\left(C_{m}\right)$. If $C=C_{1} \cap \cdots \cap C_{m}$ then $\operatorname{conv}(C) \subseteq \mathcal{P}$ and $C=\mathcal{P} \cap\{0,1\}^{n}$.

The description complexity of the convex hull of any local code $\operatorname{conv}\left(C_{i}\right)$ and thus $\mathcal{P}$ is usually much smaller than the description complexity of the codeword polytope $\operatorname{conv}(C)$.

LPD can be written as optimizing the linear objective function on the fundamental polytope $\mathcal{P}$, i.e.,

$$
\begin{equation*}
\min \left\{\lambda^{T} x: x \in \mathcal{P}\right\} \tag{5}
\end{equation*}
$$

Based on (5), the LPD algorithm which we refer to as bare linear programming decoding (BLPD) is derived.

```
Bare LP decoding (BLPD)
Input: \(\lambda \in \mathbb{R}^{n}, \mathcal{P} \subseteq[0,1]^{n}\).
Output: ML codeword or ERROR.
    solve the LP given in (5)
    if LP solution \(x^{*}\) is integral then
        output \(x^{*}\)
    else
        output ERROR
    end if
```

Because of the ML certificate property, if BLPD outputs a codeword, then it is the ML codeword. BLPD succeeds if the transmitted codeword is the unique optimum of the LP given in (5). BLPD fails if the optimal solution is non-integral or the ML codeword is not the same as the transmitted codeword. Note that the difference between the performance of BLPD and MLD is caused by the decoding failures for which BLPD finds a non-integral optimal solution. It should be emphasized that in case of multiple optima it is assumed that BLPD fails.
In some special cases, the fundamental polytope $\mathcal{P}$ is equivalent to $\operatorname{conv}(C)$, e.g., if the underlying Tanner graph is a tree or forest [24]. In these cases MLD can be achieved by BLPD. Note that in those cases also MSAD achieves MLD performance [5].
Observe that the minimum distance of a code can be understood as the minimum $\ell_{1}$ distance between any two different codewords of $C$. Likewise the fractional distance of the fundamental polytope $\mathcal{P}$ can be defined as follows.

Definition IV. 2 [2] Let $V(\mathcal{P})$ be the set of vertices (pseudocodewords) of $\mathcal{P}$. The fractional distance $d_{\text {frac }}(\mathcal{P})$ is the
minimum $\ell_{1}$ distance between a codeword and any other vertex of $V(\mathcal{P})$, i.e.
$d_{\text {frac }}(\mathcal{P})=\min \left\{\sum_{j=1}^{n}\left|x_{j}-v_{j}\right|: x \in C, v \in V(\mathcal{P}), x \neq v\right\}$.
It follows that the fractional distance is a lower bound for the minimum distance of a code: $d(C) \geq d_{\text {frac }}(\mathcal{P})$. Moreover, both definitions are related as follows. Recall that on the binary symmetric channel (BSC), MLD corrects at least $\lceil d(C) / 2\rceil-1$ bit flips. As shown in [1], LPD succeeds if at most $\left\lceil d_{\text {frac }}(\mathcal{P}) / 2\right\rceil-1$ errors occur on the BSC.

Analogously to the minimum distance, the fractional distance is equivalent to the minimum $\ell_{1}$ weight of a non-zero vertex of $\mathcal{P}$. This property is used by the fractional distance algorithm (FDA) to compute the fractional distance of a binary linear code [1]. If $\mathcal{M}$ is the set of inequalities describing $\mathcal{P}$, let $\mathcal{M}_{I}$ be the subset of those inequalities which are not active at the all-zero codeword. Note that these are exactly the inequalities with a non-zero right hand side. In FDA the weight function $\sum_{j \in J} x_{j}$ is subsequently minimized on $\mathcal{P} \cap f$ for all $f \in \mathcal{M}_{I}$ in order to find the minimum-weight non-zero vertex of $\mathcal{P}$.

```
Fractional distance algorithm (FDA)
Input: \(\mathcal{P} \subseteq[0,1]^{n}\).
Output: Minimum-weight non-zero vertex of \(\mathcal{P}\).
    for all \(f \in \mathcal{M}_{I}\) do
        Set \(\mathcal{P}^{\prime}=\mathcal{P} \cap f\).
        Solve \(\min \left\{\sum_{j \in J} x_{j}: x \in \mathcal{P}^{\prime}\right\}\).
    end for
    Choose the minimum value obtained over all \(\mathcal{P}^{\prime}\).
```

A more siginficant distance measure than $d_{\text {frac }}$ is the socalled pseudo-distance which quantifies the probability that the optimal solution under LPD changes from one vertex of $\mathcal{P}$ to another [25], [24]. Likewise, the minimum pseudoweight is defined as the minimum pseudo-distance from 0 to any other vertex of $\mathcal{P}$ and therefor identifies the vertex (pseudocodeword) which is most likely to cause a decoding failure. Note that the pseudo-distance takes the channel's probability measure into account and thus depends on the chosen channel model.

Albeit no efficient algorithms are known to compute the exact minimum pseudo-weight of the fundamental polytope of a code, promising heuristics as well as analytical bounds have been proposed [24], [25], [26].

## V. LPD Formulations for Various Code Classes

This section reviews various formulations of the polytope $\mathcal{P}$ from (5), leading to optimized versions of the general BLPD algorithm for different classes of codes.

In Step 1 of BLPD the LP problem is solved by a general purpose LP solver. These solvers usually employ the simplex method since it performs well in practice. The simplex method iteratively examines vertices of the underlying polytope until the vertex corresponding to the optimal solution is reached.

If there exists a neighboring vertex for which the objective function can be improved in the current step, the simplex method moves to this vertex. Otherwise it stops. The procedure of moving from one vertex to an other is called a simplex iteration. Details on the simplex algorithm can be found in classical books about linear programming (see e.g. [27]).

The efficiency of the simplex method depends on the complexity of the constraint set describing the underlying polytope. Several such explicit descriptions of the fundamental polytope $\mathcal{P}$ have been proposed in the LPD literature. Some can be used for any binary linear code whereas others are specialized for a specific code class. Using alternative descriptions of $\mathcal{P}$, alternative LP decoders are obtained. In the following, we are going to present different LP formulations.

## A. LP formulations for LDPC codes

The solution algorithm referred to as BLPD in Section IV was introduced by Feldman et al. [2]. In order to describe $\mathcal{P}$ explicitly, three alternative constraint sets are suggested by the authors by the formulations BLPD1, BLPD2, and BLPD3. In the following, some abbreviations are used to denote both the formulation and the associated solution (decoding) algorithm, e.g., solving an LP, subgradient optimization, neighborhood search. The meaning will be clear from the context.

The first LP formulation, BLPD1, of [2] is applicable to LDPC codes.

$$
\begin{align*}
& \min \lambda^{T} x \quad \text { (BLPD1) } \\
& \text { s.t. } \sum_{S \in E_{i}} w_{i, S}=1 \quad i=1, \ldots, m  \tag{6}\\
& x_{j}=\sum_{S \in F_{i}} w_{i, S} \quad \forall j \in N_{i}, i=1, \ldots, m  \tag{7}\\
& \text { with } j \in S \\
& 0 \leq x_{j} \leq 1 \quad j=1, \ldots, n \\
& 0 \leq w_{i, S} \leq 1 \quad \forall S \in E_{i}, i=1, \ldots, m
\end{align*}
$$

Here, $E_{i}=\left\{S \subseteq N_{i}:|S|\right.$ even $\}$ is the set of valid bit configurations within $N_{i}$. The auxiliary variables $w_{i, S}$ used in this formulation indicate which bit configuration $S \in E_{i}$ is taken at parity check $i$. In case of an integral solution, (6) ensures that exactly one such configuration is attained at every checknode, while (7) connects the actual code bits, modeled by the variables $x_{j}$, to the auxiliary variables: $x_{j}=1$ if and only if the set $S \in E_{i}$ contains $j$ for every check node $i$. Note that here we consider the LP relaxation, so it is not guaranteed that a solution of the above program is indeed integral.

A second linear programming formulation for LDPC codes, BLPD2, is obtained by employing the so-called forbidden set (FS) inequalities [28]. The FS inequalities are motivated by the observation that one can explicitly forbid those value assignments to variables where $|S|$ is odd. For all local codewords in $C_{i}$ it holds that

$$
\sum_{j \in S} x_{j}-\sum_{j \in N_{i} \backslash S} x_{j} \leq|S|-1 \quad \forall S \in \Sigma_{i}
$$

where $\Sigma_{i}=\left\{S \subseteq N_{i}:|S|\right.$ odd $\}$. Feldman et al. show in [2] that for each single parity-check code $C_{i}$, the FS inequalities
together with the box inequalities $0 \leq x_{j} \leq 1, j \in J$ completely and non-redundantly describe $\operatorname{conv}\left(C_{i}\right)$ (the case $\left|N_{i}\right|=3$ as depicted in Fig. 2 is the only exception where the box inequalities are not needed). In a more general setting, Grötschel proved this result for the cardinality homogeneous set systems [29].

If the rows of $H$ are considered as dual codewords, the set of FS inequalities is a reinvention of cocircuit inequalities explained in Section III. BLPD2 is given below.

$$
\begin{array}{ll} 
& \min \lambda^{T} x \quad \text { (BLPD2) } \\
\text { s.t. } & \sum_{j \in S} x_{j}-\sum_{j \in N_{i} \backslash S} x_{j} \leq|S|-1 \\
0 \leq x_{j} \leq 1 & \forall S \in \Sigma_{i}, i=1, \ldots, m \\
& j=1, \ldots, n
\end{array}
$$

Feldman et al. [2] apply BLPD using formulations BLPD1 or BLPD2 to LDPC codes. Under the BSC, the errorcorrecting performance of BLPD is compared with the MSAD on an random rate- $\frac{1}{2}$ LDPC code with $n=200, d_{v}=3$, $d_{c}=6$; with the MSAD, SPAD on the random rate- $\frac{1}{4}$ LDPC code with $n=200, d_{v}=3, d_{c}=4$; with the MSAD, SPAD, MLD on the random rate $-\frac{1}{4}$ LDPC code with $n=60, d_{v}=3$, $d_{c}=4$. On these codes, BLPD performs better than MSAD but worse than SPAD. Using BLPD2, the FDA is applied to random rate $-\frac{1}{4}$ LDPC codes with $n=100,200,300,400$, $d_{v}=3$ and $d_{c}=4$ from an ensemble of Gallager [30]. For ( $n-1, n$ ) Reed-Muller codes [31] with $4 \leq n \leq 512$ they compare the classical distance with the fractional distance. The numerical results suggest that the gap between both distances grows with increasing block length.

Another formulation for LDPC codes is given in Section VI-B in the context of efficient implementations.

In a remarkable work, Feldman and Stein [32] have shown that the Shannon capacity of a channel can be achieved with LP decoding, which implies a polynomial-time decoder and the availability of an ML certificate. To this end, they use a slightly modified version of BLPD1 restricted to expander codes, which are a subclass of LDPC codes. See [32] for a formal definition of expander codes as well as the details of the corresponding decoder.

## B. LP formulations for codes with high-density parity-check matrices

The number of variables and constraints in BLPD1 as well as the number of constraints in BLPD2 increase exponentially in the check node degree. Thus, for codes with high-density parity-check matrices, BLPD1 and BLPD2 are computationally inefficient. A polynomial-sized formulation, BLPD3, is based on the parity polytope of Yannakakis [33]. There are two types of auxiliary variables in BLPD3. The variable $p_{i, k}$ is set to one if $k$ variable nodes are set to one in the neighborhood of parity-check $i$, for $k$ in the index set $K_{i}=\left\{0,2, \ldots, 2\left\lfloor\frac{\left|N_{i}\right|}{2}\right\rfloor\right\}$. Furthermore, the variable $q_{j, i, k}$ is set to one if variable node $j$ is one of the $k$ variable nodes set
to one in the neighborhood of check node $i$.

$$
\begin{array}{ll}
\min \lambda^{T} x \quad \text { (BLPD3) } & \\
\text { s.t. } & x_{j}=\sum_{k \in K_{i}} q_{j, i, k} \\
\sum_{k \in K_{i}} p_{i, k}=1 & i \in N_{j}, j=1, \ldots, n \\
\sum_{j \in N_{i}} q_{j, i, k}=k p_{i, k} & k \in 1, \ldots m \\
0 \leq x_{j} \leq 1 & \\
0 \leq K_{i, k}, i=1, \ldots m \\
0 \leq q_{j, i, k} \leq p_{i, k} & k \in 1, \ldots, n \\
0 \in K_{i}, i=1, \ldots, m \\
& k \in 1, \ldots, n, i \in N_{j}
\end{array}
$$

Feldman et al. [2] show that BLPD1, BLPD2, and BLPD3 are equivalent in the sense that the $x$-variables of the optimal solutions in all three formulations take the same values.

The number of variables and constraints in BLPD3 increases as $O\left(n^{3}\right)$. By applying a decomposition approach, Yang et al. [34] show that an alternative LP formulation which has size linear in the length and check node degrees can be obtained (it should be noted that independently from [34] a similar decomposition approach was also proposed in [35]). In the LP formulation of [34] a high degree check node is decomposed into several low degree check nodes. Thus, the resulting Tanner graph contains auxiliary check and variable nodes. Fig. 3 illustrates this decomposition technique: a check node with degree 4 is decomposed into 2 parity checks each with degree at most 3 . The parity-check nodes are illustrated


Fig. 3. Check node decomposition.
by squares. In the example, original variables are denoted by $\nu_{1}, \ldots, \nu_{4}$ while the auxiliary variable node is named $\nu_{5}$. In general, this decomposition technique is iteratively applied until every check node has degree less than 4 . The authors show that the total number of variables in the formulation is less than doubled by the decomposition. For the details of the decomposition [34] is referred.

For the ease of notation, suppose $K$ is the set of paritycheck nodes after decomposition. If $d_{c}(k)=3, k \in K$, then the parity-check constraint $k$ is of the form $\nu_{1}^{k}+\nu_{2}^{k}+\nu_{3}^{k} \equiv 0$ $(\bmod 2)$. Note that with our notation some of these variables $\nu_{s}^{k}$ might represent the same variable node $\nu_{j}$, e.g. $\nu_{5}$ from Fig. 3 would appear in two constraints of the above form, as $\nu_{s}^{1}$ and $\nu_{s^{\prime}}^{2}$, respectively. Yang et al. show that the parity-check constraint $\nu_{1}^{k}+\nu_{2}^{k}+\nu_{3}^{k} \equiv 0(\bmod 2)$ can be replaced by the linear constraints $\nu_{1}^{k}+\nu_{2}^{k}+\nu_{3}^{k} \leq 2, \nu_{1}^{k}-\nu_{2}^{k}-\nu_{3}^{k} \leq 0, \nu_{2}^{k}-\nu_{1}^{k}-$ $\nu_{3}^{k} \leq 0, \nu_{3}^{k}-\nu_{1}^{k}-\nu_{2}^{k} \leq 0$ (for a single check node of degree 3 the box inequalities are not needed). If $d_{c}(k)=2$ then $\nu_{1}^{k}=$ $\nu_{2}^{k}$ along with the box constraints models the parity-check. The constraint set of the resulting LP formulation, which we call cascaded linear programming decoding (CLPD), is the union
of all constraints modeling the $|K|$ parity checks.

$$
\begin{array}{ll} 
& \min \bar{\lambda}^{T} \nu \\
\text { s.t. } & \sum_{j \in S} \nu_{j}^{k}-\sum_{j \in N_{k} \backslash S} \nu_{j}^{k} \leq|S|-1 \\
0 \leq \nu_{j} \leq 1 & \forall S \in \Sigma_{k}, k=1, \ldots,|K| \\
0 & \text { if } d_{c}(i) \leq 2 \forall i: j \in N_{i}
\end{array}
$$

In the objective function only the $\nu$ variables corresponding to the original $x$ variables have non-zero coefficients. Thus, the objective function of CLPD is the same as of BLPD1. The constraints in CLPD are the FS inequalities used in BLPD2 with the property that the degree of the check node is less than 4.

Yang et al. prove that the formulations introduced in [2] and CLPD are equivalent. Again, equivalence is used in the sense that in an optimal solution, the $x$-variables of BLPD1, BLPD2, BLPD3, and the variables of the CLPD formulation which correspond to original $x$-variables take the same values. Moreover, it is shown that CLPD can be used in FDA. As a result, the computation of the fractional distance for codes with high-density parity-check matrices is also facilitated. Note that using BLPD2, the FDA algorithm has polynomial running time only for LDPC codes. If $\mathcal{P}$ is described by the constraint set of CLPD, then in the first step of the FDA, it is sufficient to choose the set $\mathcal{F}$ from the facets formed by cutting planes of type $\nu_{1}^{k}+\nu_{2}^{k}+\nu_{3}^{k}=2$ where $\nu_{1}^{k}, \nu_{2}^{k}$, and $\nu_{3}^{k}$ are variables of the CLPD formulation. Additionally, an adaptive branch \& bound method is suggested in [36] to find better bounds for the minimum distance of a code. On a random rate- $\frac{1}{4}$ LDPC code with $n=60, d_{v}=3, d_{c}=4$, it is demonstrated that this yields a better lower bound than the fractional distance does.

## C. LP formulations for turbo-like codes

The various LP formulations outlined so far have in common that they are derived from a parity-check matrix which defines a specific code. A different approach is to describe the encoder by means of a finite state machine, which is the usual way to define so-called convolutional codes. The bits of the information word are subsequently fed into the machine, each causing a state change that emits a fixed number of output bits depending on both the current state and the input. In a systematic code, the output always contains the input bit. The codeword, consisting of the concatenation of all outputs, can thus be partitioned into the systematic part which is a copy of the input and the remaining bits, being refered to as the parity output.

A convolutional code is naturally represented by a trellis graph (Fig. 4), which is obtained by unfolding the state diagram in the time domain. Each vertex of the trellis represents the state at a specific point in time, while edges correspond to valid transitions between two subsequent states and are labelled by the according input and output bits. Each path from the starting node to the end node corresponds to a codeword. ${ }^{4}$ The cost of a codeword is derived from the received LLR values and the edge labels on the path associated with this codeword. See [23] for an in-depth survey of these concepts.

[^3]

Fig. 4. Excerpt from a trellis graph with four states and initial state 0 . The style of an edge indicates the according information bit, while the labels refer to the single parity bit.

Convolutional codes are the building blocks of turbo codes, which revolutionized coding theory because of their near Shannon limit error-correcting performance [37]. An $(n, k)$ turbo code consists of two convolutional codes $C_{a}$ and $C_{b}$, each of input length $k$, which are linked by a so-called interleaver that requires the information bits of $C_{a}$ to match those of $C_{b}$ after being scrambled by some permutation $\pi \in \mathbb{S}_{k}$ which is fixed for a given code. ${ }^{5}$ It is this coupling of rather weak individual codes and the increase of complexity arising therefrom that entails the vast performance gain of turbo codes. A typical turbo code (and only this case is covered here; it is straightforward to generalize) consists of two identical systematic encoders of rate $\frac{1}{2}$ each. Only one of the encoders $C_{a}$ and $C_{b}$, however, contributes its systematic part to the resulting codeword, yielding an overall rate of $\frac{2}{3}$, i. e. $n=3 k$ (since their systematic parts differ only by a permutation, including both would imply an embedded repetition code). We thus partition a codeword $x$ into the systematic part $x^{s}$ and the parity outputs $x^{a}$ and $x^{b}$ of $C_{a}$ and $C_{b}$, respectively.

A turbo code can be compactly represented by a so-called Forney-style factor graph (FFG) as shown in Fig. 5. As opposed to Tanner graphs, in an FFG all nodes are functional nodes, whereas the (half-)edges correspond to variables. In our case, there are variables of two types, namely state variables $s_{j}^{\nu}(\nu \in\{a, b\})$, reflecting the state of $C_{\nu}$ at time step $j$, and a variable for each bit of the codeword $x$. Each node $T_{j}^{\nu}$ represents the indicator function for a valid state transition in $C_{\nu}$ at time $j$ and is thus incident to one systematic and one parity variable as well as the "before" and "after" state $s_{j-1}^{\nu}$ and $s_{j}^{\nu}$, respectively. Note that such a node $T_{j}^{\nu}$ corresponds to a vertical "slice" (often called a segment) of the trellis graph of $C_{\nu}$, and each valid configuration of $T_{j}^{\nu}$ is represented by exactly one edge in the respective segment.

Turbo codes are typically decoded by IMPD techniques operating on the factor graph. Feldman [1] in contrast introduced an LP formulation, turbo code linear programming decoding (TCLPD), for this purpose. This serves as an example that mathematical programming is a promising approach in decoding even beyond formulations based on parity-check matrices.

In TCLPD, the trellis graph of each constituent encoder $C^{\nu}$ is modeled by flow conservation and capacity constraints [39], along with side constraints appropriately connecting the flow

[^4]

Fig. 5. The factor graph of a turbo code. The interleaver links the systematic bits $x^{s}$ of both encoders $C_{a}$ (upper part) and $C_{b}$ (lower part).
variables $f^{\nu}$ to auxiliary variables $x^{s}$ and $x^{\nu}$, respectively, which embody the codeword bits.

For $\nu \in\{a, b\}$, let $G_{\nu}=\left(S_{\nu}, E_{\nu}\right)$ be the trellis according to $C_{\nu}$, where $S_{\nu}$ is the index set of nodes (states) and $E_{\nu}$ is the set of edges (state transitions) $e$ in $G_{\nu}$. Let $s^{\text {start, } \nu}$ and $s^{\text {end }, \nu}$ denote the unique start and end node, respectively, of $G_{\nu}$. We can now define a feasible flow $f^{\nu}$ in the trellis $G_{\nu}$ by the system

$$
\begin{gather*}
\sum_{e \in \operatorname{out}\left(s^{\mathrm{start}, \nu}\right)} f_{e}^{\nu}=1, \quad \sum_{e \in \operatorname{in}\left(s^{\mathrm{end}, \nu}\right)} f_{e}^{\nu}=1  \tag{8}\\
\sum_{e \in \operatorname{out}(s)} f_{e}^{\nu}=\sum_{e \in \operatorname{in}(s)} f_{e}^{\nu} \forall s \in S_{\nu} \backslash\left\{s^{\mathrm{start}, \nu}, s^{\mathrm{end}, \nu}\right\}  \tag{9}\\
f_{e}^{\nu} \geq 0 \quad \forall e \in E_{\nu} \tag{10}
\end{gather*}
$$

Let $I_{j}^{\nu}$ and $O_{j}^{\nu}$ denote the set of edges in $G_{\nu}$ whose corresponding input and output bit, respectively, is a 1 (both being subsets of the $j$-th segment of $G_{\nu}$ ), the following constraints relate the codeword bits to the flow variables:

$$
\begin{array}{rr}
x_{j}^{\nu}=\sum_{e \in O_{j}^{\nu}} f_{e}^{\nu} & \text { for } j=1, \ldots, k \text { and } \nu \in\{a, b\}, \\
x_{j}^{s}=\sum_{e \in I_{j}^{a}} f_{e}^{a} & \text { for } j=1, \ldots, k, \\
x_{\pi(j)}^{s}=\sum_{e \in I_{j}^{b}} f_{e}^{b} & \text { for } j=1, \ldots, k . \tag{13}
\end{array}
$$

We can now state TCLPD as

$$
\min \sum_{\nu \in\{a, b\}}\left(\lambda^{\nu}\right)^{T} x^{\nu}+\left(\lambda^{s}\right)^{T} x^{s}
$$

(TCLPD)
s.t. (8)-(13) hold.
where $\lambda$ is split in the same way as $x$.
The formulation straightforwardly generalizes to all sorts of "turbo-like" codes, i. e., codes built by convolutional codes plus interleaver conditions. In particular, Feldman and Karger have applied TCLPD to repeat-accumulate (RA $(l)$ ) codes [40]. The encoder of an $\mathrm{RA}(l)$ repeats the information bits $l$ times, and then sends them to an interleaver followed by an accumulator, which is a two-state convolutional encoder. The authors derive bounds on the error rate of TCLPD for RA codes which were later improved and extended by Halabi and Even [41] as well as by Goldenberg and Burshtein [42].

Note that all $x$ variables in TCLPD are auxiliary: we could replace each occurence by the sum of flow variables defining it. In doing so, (12) and (13) break down to the condition

$$
\begin{equation*}
\sum_{e \in I_{\pi(j)}^{a}} f_{e}^{a}=\sum_{e \in I_{j}^{b}} f_{e}^{b} \quad \text { for } j=1, \ldots, k \tag{14}
\end{equation*}
$$

Because the rest of the constraints defines a standard network flow, TCLPD models a minimum cost flow problem plus the $k$ additional side constraints (14). Using a general purpose LP solver does not exploit this combinatorial substructure. As was suggested already in [1], in [43] Lagrangian relaxation is applied to (14) in order to recover the underlying shortestpath problem. Additionally, the authors of [43] use a heuristic based on computing the $K$ shortest paths in a trellis to improve the decoding performance. Via the parameter $K$ the trade-off between algorithmic complexity and error-correcting performance can be controlled.

## VI. Efficient LP Solvers for BLPD

A successful realization of BLPD requires an efficient LP solver. To this end, several ideas have been suggested in the literature. CLPD (cf. Section V) can be considered an efficient LPD approach since the number of variables and constraints are significantly reduced. We review several others in this section.

## A. Solving the separation problem

The approach of Taghavi and Siegel [44] tackles the large number of constraints in BLPD2. In their separation approach called adaptive linear programming decoding (ALPD), not all FS inequalities are included in the LP formulation as in BLPD2. Instead, they are iteratively added when needed. As in Definition II.5, the general idea is to start with a crude LP formulation and then improve it. Note that this idea can also be used to improve the error-correcting performance (see Section VII). In the initialization step, the trivial LP $\min \left\{\lambda^{T} x: x \in[0,1]^{n}\right\}$ is solved. Let $\left(x^{*}\right)^{k}$ be the optimal solution in iteration $k$. Taghavi and Siegel show that it can be checked in $O\left(m d_{c}^{\max }+n \log n\right)$ time if $\left(x^{*}\right)^{k}$ violates any FS inequality derived from $H_{i, .} x=0(\bmod 2)$ for all $i \in I$ (recall that $m \times n$ is the dimension of $H$ and $d_{c}^{\max }$ is the maximum maximum check-node degree). This check can be considered as a special case of the greedy separation algorithm (GSA) introduced in [29]. If some of the FS inequalities are violated then these inequalities are added to the formulation and the modified LP is solved again with the new inequalities. ALPD stops if the current optimal solution $\left(x^{*}\right)^{k}$ satisfies all FS inequalities. If $\left(x^{*}\right)^{k}$ is integral then it is the ML codeword, otherwise an error is output. ALPD does not yield an improvement in terms of frame error rate since the same solutions are found as in the formulations in the previous section. However, the computational complexity is reduced.

An important algorithmic result of [44] is that ALPD converges to the same optimal solution as BLPD2 with significantly fewer constraints. It is shown empirically that in the last iteration of ALPD, less constraints than in the formulations BLPD2, BLPD3, and CLPD are used. Taghavi and Siegel [44]
prove that their algorithm converges to the optimal solution on the fundamental polytope after at most $n$ iterations with at most $n(m+2)$ constraints.

Under the binary-input additive white Gaussian noise channel (BIAWGNC), [44] uses various random $\left(d_{v}, d_{c}\right)$-regular codes to test the effect of changing the check node degree, the block length, and the code rate on the number of FS inequalities generated and the convergence of their algorithm. Setting $n=360$ and rate $R=\frac{1}{2}$, the authors vary the check node degree in the range of 4 to 40 in their computational testing. It is observed that the average and the maximum number of FS inequalities remain below 270. The effect of changing block length $n$ between 30 and 1920 under $R=\frac{1}{2}$ is demonstrated on a $(3,6)$-regular LDPC code. For these codes, it is demonstrated that the number of FS inequalities used in the final iteration is generally between $0.6 n$ and $0.7 n$. Moreover, it is reported that the number of iterations remain below 16. The authors also investigate the effect of the rate on the number of FS inequalities created. Simulations are performed on codes with $n=120$ and $d_{v}=3$ where the number of parity checks $m$ vary between 15 and 90 . For most values of $m$ it is observed that the average number of FS inequalities ranges between 1.1 m and 1.2 m . For ALPD, BLPD2, and SPAD (50 iterations), the average decoding time is testet for $(3,6)$-regular and $(4,8)$-regular LDPC codes with various block lengths. It is shown that ALPD outperforms BLPD with respect to computation time, whil still being slower than SPAD. Furthermore, increasing the check node degree does not increase the computation time of ALPD as much as the computation time of BLPD. The behavior of ALPD, in terms of the number of iterations and the FS inequalities used, under increasing SNR is tested on a (3, 6)-regular LDPC code with $n=240$. It is concluded that ALPD performs more iterations and uses more FS inequalities for the instances it fails. Thus, decoding time decreases with increasing SNR.

In [45] ALPD is improved further in terms of complexity. The authors use some structural properties of the fundamental polytope. Let $\left(x^{*}\right)^{k}$ be an optimal solution in iteration $k$. In [44] it is shown that, if $\left(x^{*}\right)^{k}$ does not satisfy an FS inequality derived from check node $i$, then $\left(x^{*}\right)^{k}$ satisfies all other FS inequalities derived from $i$ with strict inequality. Based on this result, Taghavi et al. [45] modify ALPD and propose the decoding approach we refer to as modified adaptive linear programming decoding (MALPD). In the $(k+1)^{\text {th }}$ iteration of MALPD, it is checked in $O\left(m d_{c}^{\max }\right)$ time if $\left(x^{*}\right)^{k}$ violates any FS inequality derived from $H_{i, .} x=0(\bmod 2)$ for some $i \in I$. This check is performed only for those parity checks $i \in I$ which do not induce any active FS inequality at $\left(x^{*}\right)^{k}$. Moreover, it is proved that inactive FS inequalities at iteration $k$ can be dropped. In any iteration of MALPD, there are at most $m$ FS inequalities. However, the dropped inequalities might be inserted again in a later iteration; therefore the number of iterations for MALPD can be higher than for ALPD.

## B. Message passing-like algorithms

An approach towards low complexity LPD of LDPC codes was proposed by Vontobel and Kötter in [46]. Based on an

FFG representation of an LDPC code, they derive an LP, called primal linear programming decoding (PLPD), which is based on BLPD1. The FFG, shown in Fig. 6, and the Tanner graph are related as follows.


Fig. 6. A Forney-style factor graph for PLPD.

For each parity check, the FFG exhibits a node $C_{i}$ which is incident to a variable-edge $v_{i, j}$ for each $j \in N_{i}$ and demands those adjacent variables to form a configuration that is valid for the local code $C_{i}$, i.e., their sum must be even. This corresponds to a check node in the Tanner graph and thus to (6) and (7) except that now there are, for the moment, independent local variables $v_{i, j}$ for each $C_{i}$. Additionally, the FFG generalizes the concept of row-wise local codes $C_{i}$ to the columns of $H$, in such a way that the $j^{\text {th }}$ column is considered a local repetition code $A_{j}$ that requires the auxiliary variables $u_{j, i}$ for each $i \in N_{j} \cup\{0\}$ to be either all 1 or all 0 . By this, the variable nodes of the Tanner graph are replaced by check nodes $A_{j}$-recall that in an FFG all nodes have to be check nodes. There is a third type of factor nodes, labelled by " $=$ ", which simply require all incident variables to take on the same value. These are used to establish consistency between the row-wise variables $v_{i, j}$ and the column-wise variables $u_{j, i}$ as well as connecting the codeword variables $x_{j}$ to the configurations of the $A_{j}$.

From this discussion it is easily seen that the FFG indeed ensures that any configuration of the $x_{j}$ is a valid codeword. The outcome of writing down the constraints for each node and relaxing integrality conditions on all variables is the LP

$$
\begin{array}{ll} 
& \min \lambda^{T} x \quad(\mathrm{PLPD}) \\
\text { s.t. } & x_{j}=u_{j, 0} \\
& \\
u_{j, i}=v_{i, j} & j=1, \ldots, n, \\
& \forall(i, j) \in I \times J: H_{i, j}=1, \\
u_{j, i}=\sum_{S \in A_{j}, S \ni j} \alpha_{j, S} & \forall i \in N_{j}, j=1, \ldots, n, \\
& \sum_{S \in A_{j}} \alpha_{j, S}=1 \\
v_{i, j}=\sum_{S \in E_{i}, S \ni j} w_{i, S} & \forall j \in N_{i}, i=1, \ldots, m \\
& \\
\sum_{S \in E_{i}} w_{i, S}=1 & \text { for all } j=1, \ldots, n \\
\alpha_{j, S} \geq 0 & \forall S \in A_{j}, j=1, \ldots, n \\
w_{i, S} \geq 0 & \forall S \in E_{i}, i=1, \ldots, m
\end{array}
$$

where the sets $E_{i}$ are defined as in (BLPD1).
While bloating BLPD1 in this manner seems inefficient at first glance, the reason behind is that the LP dual of

PLPD, leads to an FFG which is topologically equivalent to the one of the primal LP, which allows to use the graphical structure for solving the dual. After manipulating constraints of the dual problem to obtain a closely related, "softened" dual linear programming decoding (SDLPD) formulation, the authors propose a coordinate-ascent-type algorithm resembling the min-sum algorithm and show convergence under certain assumptions. In this algorithm, all the edges of FFG are updated according to some schedule. It is shown that the update calculations required during each iteration can be efficiently performed by the SPAD. The coordinate-ascent-type algorithm for SDLPD is guaranteed to converge if all the edges of the FFG are updated cyclically.

Under the BIAWGNC, the authors compare the errorcorrecting performance of the coordinate-ascent-type algorithm (max iterations: 64, 256) against the performance of the MSAD (max iterations: 64, 256) on the (3,6)-regular LDPC code with $n=1000$ and rate $R=\frac{1}{2}$. MSAD performs slightly better than the coordinate-ascent-type algorithm. In summary, Vontobel and Kötter [46] show that it is possible to develop LP based algorithms with complexities similar to IMPD.

The convergence and the complexity of the coordinate-ascent-type algorithm proposed in [46] are studied further in [47] by Burshtein. His algorithm has a new scheduling scheme and its convergence rate and computational complexity are analyzed under this scheduling. With this new scheduling scheme, the decoding algorithm from [46] yields an iterative approximate LPD algorithm for LDPC codes with complexity in $O(n)$. The main difference between the two algorithms is the selection and update of edges of the FFG. In [46] all edges are updated cyclically during one iteration, whereas in [47], only few selected edges are updated during one particular iteration. The edges are chosen according to the variable values obtained during previous iterations.

## C. Nonlinear programming approach

As an approximation of BLPD for LDPC codes, Yang et al. [36] introduce the box constraint quadratic programming decoding (BCQPD) whose time complexity is linear in the code length. BCQPD is a nonlinear programming approach derived from the Lagrangian relaxation (see [7] for an introduction to Lagrangian relaxation) of BLPD1. To achieve BCQPD, a subset of the set of the constraints are incorporated into the objective function. To simplify notation, Yang et al. rewrite the constraint blocks (6) and (7) in the general form $A y=b$ by defining a single variable vector $y=(x, w)^{T} \in\{0,1\}^{K}$ (so $K$ is the total number of variables in BLPD1) and choosing $A$ and $b$ appropriately. Likewise, the objective function coefficients are rewritten in a vector $c$, wich equals $\lambda$ followed by the appropriate number of zeros. The resulting formulation is $\min \left\{c^{T} y: A y=b, y \in\{0,1\}^{K}\right\}$. Using a multiplier $\alpha>0$, the Lagrangian of this problem is

$$
\begin{aligned}
& \quad \min c^{T} y+\alpha(A y-b)^{T}(A y-b) \\
& \text { s.t. } 0 \leq y_{k} \leq 1 \quad \text { for } k=1, \ldots, K
\end{aligned}
$$

If $A y=b$ is violated then a positive value is added to the original objective function $c^{T} y$, i.e., the solution $y$ is
penalized. Setting $Q=2 \alpha A^{T} A$ and $r=c-2 \alpha A^{T} b$ the BCQPD problem

$$
\begin{gathered}
\min y^{T} Q y+2 r^{T} y \quad(\mathrm{BCQPD}) \\
\text { s.t. } 0 \leq y_{k} \leq 1 \quad \text { for } k=1, \ldots, K
\end{gathered}
$$

is obtained. Since $Q$ is a positive semi-definite matrix, i. e., the objective function is convex, and since the set of constraints constitutes a box, each $y_{k}$ can be minimized separately. This leads to efficient serial and parallel decoding algorithms. Two methods are proposed in [36] to solve the BCQPD problem, the projected successive overrelaxation method (PSORM) and the parallel gradient projection method (PGPM). These methods are generalizations of Gauss-Seidel and Jacobi methods [48] with the benefit of faster convergence if proper weight factors are chosen. PSORM and PGPM benefit from the lowdensity structure of the underlying parity-check matrix.

One of the disadvantages of IMPD is the difficulty of analyzing the convergence behavior of such algorithms. Yang et al. showed both theoretically and empirically that BCQPD converges under some assumptions if PSORM or PGPM is used to solve the quadratic programming problem. Moreover, the complexity of BCQPD is smaller than the complexity of SPAD. For numerical tests, the authors use a product code with block length $4^{5}=1024$ and rate $\left(\frac{3}{4}\right)^{5}=0.237$. The BIAWGNC is used. It is observed that the PSORM method converges faster than PGPM. The error-correcting performance of SPAD is poor for product codes due to their regular structure. For the chosen product code, Yang et al. demonstrate that PSORM outperforms SPAD in computational complexity as well as in error-correcting performance.

## D. Efficient LPD of SPC product codes

The class of single parity-check (SPC) product codes is of special interest in [34]. The authors prove that for SPC product codes the fractional distance is equal to the minimum Hamming distance. Due to this observation, the minimum distance of SPC product codes can be computed in polynomial time using FDA. Furthermore, they propose a low complexity algorithm which approximately computes the CLPD optimum for SPC product codes. This approach is based on the observation that the parity-check matrix of an SPC product code can be decomposed into component SPC codes. A Lagrangian relaxation of CLPD is obtained by keeping the constraints from only one component code in the formulation and moving all other constraints to the objective function with a penalty vector. The resulting Lagrangian dual problem is solved by subgradient algorithms (see [7]). Two alternatives, subgradient decoding (SD) and joint subgradient decoding (JSD) are proposed. It can be proved that subgradient decoders converge under certain assumptions.

The number of iterations performed against the convergence behavior of SD is tested on the $(4,4)$ SPC product code, which has length $n=256$, rate $R=\left(\frac{3}{4}\right)^{4} \approx 0.32$ and is defined as the product of four SPC codes of length 4 each. All variants tested (obtained by keeping the constraints from component code $j=1,2,3,4$ in the formulation) converge in less than 20 iterations. For demonstrating the error-correcting performance
of SD if the number of iterations are set to $5,10,20,100$, the $(5,2)$ SPC product code $\left(n=25\right.$, rate $\left.R=\left(\frac{4}{5}\right)^{2}=0.64\right)$ is used. The error-correcting performance is improved by increasing the number of iterations. Under the BIAWGNC, this code and the $(4,4)$ SPC product code are used to compare the error-correcting performance of SD and JSD with the performance of BLPD and MLD. It should be noted that for increasing SNR values, the error-correcting performance of BLPD converges to that of MLD for SPC codes. JSD and SD approach the BLPD curve for the code with $n=25$. For the SPC product code with $n=256$ the subgradient algorithms perform worse than BLPD. For both codes, the error-correcting performance of JSD is superior to SD. Finally, the $(10,3)$ SPC product code with $n=1000$ and rate $R=\left(\frac{9}{10}\right)^{3} \approx 0.729$ is used to compare the error-correcting performance of SD and JSD with the SPAD. Again the BIAWGNC is used. It is observed that SD performs slightly better than the SPAD with a similar computational complexity. JSD improves the error-correcting performance of the SD at the cost of increased complexity.

## E. Interior point algorithms

Efficient LPD approaches based on interior point algorithms are studied by Vontobel [49], Wadayama [50], and Taghavi et al. [45]. The use of interior point algorithms to solve LP problems as an alternative to the simplex method was initiated by Karmarkar [51]. In these algorithms, a starting point in the interior of the feasible set is chosen. This starting point is iteratively improved by moving through the interior of the polyhedron in some descent direction until the optimal solution or an approximation is found. There are various interior point algorithms and for some, polynomial time convergence can be proved. This is an advantage over the simplex method which has exponential worst case complexity.

The proposed interior point algorithms aim at using the special structure of the LP problem. The resulting running time is a low-degree polynomial function on the block length. Thus, fast decoding algorithms based on interior point algorithms may be developed for codes with large block lengths. In particular affine scaling algorithms [49], primal-dual interior point algorithms [45], [49] and primal path following interior point algorithm [50] are considered. The bottleneck operation in interior point methods is to solve a system of linear equations depending on the current iteration of the algorithm. Efficient approaches to solve this system of equations are proposed in [49], [45], the latter containing an extensive study, including investigation of appropriate preconditioners for the often illconditioned equation system. The speed of convergence to the optimal vertex of the algorithms in [50] and [45] under the BIAWGNC are demonstrated on a nearly $(3,6)$-regular LDPC code with $n=1008, R=\frac{1}{2}$ and a randomly-generated $(3,6)$ regular LDPC code with $n=2000$, respectively.

## VII. Improving the Error-Correcting Performance of BLPD

The error-correcting performance of BLPD can be improved by techniques from integer programming. Most of the
improvement techniques can be grouped into cutting plane or branch \& bound approaches. In this section, we review the improved LPD approaches mainly with respect to this categorization.

## A. Cutting plane approaches

The fundamental polytope $\mathcal{P}$ can be tightened by cutting plane approaches. In the following, we refer to valid inequalities as inequalities satisfied by all points in $\operatorname{conv}(C)$. Valid cuts are valid inequalities which are violated by some non-integral vertex of the LP relaxation. Feldman et al. [2] already address this concept; besides applying the "Lift and project" technique which is a generic tightening method for integer programs [52], they also strengthen the relaxation by introducing redundant rows into the parity-check matrix (or, equivalently, redundant parity-checks into the Tanner graph) of the given code (cf. Section II). When using the BLPD2 formulation, we derive additional FS inequalities from the redundant parity-checks without increasing the number of variables. We refer to such inequalities as redundant parity-check (RPC) inequalities. RPC inequalities may include valid cuts which increase the possibility that LPD outputs a codeword. An interesting question relates to the types of inequalities required to describe the codeword polytope $\operatorname{conv}(C)$ exactly. It turns out that conv $(C)$ cannot be described completely by using only FS and box inequalities; the $(7,3,4)$ simplex code (dual of the $(7,4,3)$ Hamming code) is given as a counterexample in [2]. More generally, it can be concluded from [53] that these types of inequalities do not suffice to describe all facets of a simplex code.

RPCs can also be interpreted as dual codewords. As such, for interesting codes there are exponentially many RPC inequalities. The RPC inequalities cutting off the non-integral optimal solutions are called RPC cuts [44]. An analytical study under which circumstances RPCs can induce cuts is carried out in [24]. Most notably, it is shown that RPCs obtained by adding no more than $\frac{g-2}{2}$ dual codewords, where $g$ is the length of a shortest cycle in the Tanner graph, never change the fundamental polytope.

There are several heuristic approaches in the LPD literature to find cut inducing RPCs [2], [54], [44], [55]. In [2], RPCs which result from adding any two rows of $H$ are appended to the original parity-check matrix. The authors of [44] find RPCs by randomly choosing cycles in the fractional subgraph of the Tanner graph, which is obtained by choosing only the fractional variable nodes and the check nodes directly connected to them. They give a theorem which states that every possible RPC cut must be generated by such a cycle. Their approach is a heuristic one since the converse of that theorem does not hold. In [54] the column index set corresponding to an optimal LP solution is sorted. By re-arranging $H$ and bringing it to row echelon form, RPC cuts are searched. In [55], the parity-check matrix is reformulated such that unit vectors are obtained in the columns of the parity-check matrix which correspond to fractional valued bits in the optimal solution of the current LP. RPC cuts are derived from the rows of the modified parity-check matrix.

The approaches in [28], [44], and [55] rely on a noteworthy structural property of the fundamental ploytope. Namely, it can be shown that no check node of the associated Tanner graph (regardless of the existence of redundant parity-checks) can be adjacent to only one non-integral valued variable node.

Feldman et al. [2] test the lift and project technique on a random rate- $\frac{1}{4}$ LDPC code with $n=36, d_{v}=3$ and $d_{v}=4$ under the BIAWGNC. Moreover, a random rate $-\frac{1}{4}$ LDPC code with $n=40, d_{v}=3$, and $d_{c}=4$ is used to demonstrate the error-correcting performance of BLPD when the original parity-check matrix is extended by all those RPCs obtained by adding any two rows of the original matrix. Both tightening techniques improve the error-correcting performance of BLPD, though the benefit of the latter is rather poor, due to the abovementioned condition on cycle lengths.

The idea of tightening the fundamental polytope is usually implemented as a cutting plane algorithm, i.e., the separation problem is solved (see Definition II. 5 and Section VI-A). In cutting plane algorithms, an LP is solved which contains only a subset of the constraints of the corresponding optimization problem. If the optimal LP solution is a codeword then the cutting plane algorithm terminates and outputs the ML codeword. Otherwise, valid cuts from a predetermined family of valid inequalities are searched. If some valid cuts are found, they are added to the LP formulation and the LP is resolved. In [44], [54], [55] the family of valid cuts is FS inequalities derived from RPCs.

In [54] the main motivation for the greedy cutting plane algorithm is to improve the fractional distance. This is demonstrated for the $(7,4,3)$ Hamming code, the $(24,12,8)$ Golay code and a $(204,102)$ LDPC code. As a byproduct under the BSC it is shown on the $(24,12,8)$ Golay code and a $(204,102)$ LDPC code that the RPC based approach of [54] improves the error-correcting performance of BLPD.

In the improved LPD approach of [44], first ALPD (see Section VI) is applied. If the solution is non-integral, an RPC cut search algorithm is employed. This algorithm can be briefly outlined as follows:

1) Given a non-integral optimal LP solution $x^{*}$, remove all variable nodes $j$ for which $x_{j}^{*}$ is integral from the Tanner graph.
2) Find a cycle by randomly walking through the pruned Tanner graph.
3) Sum up (in $\mathbb{F}_{2}$ ) the rows $H$ which correspond to the check nodes in the cycle.
4) Check if the resulting RPC introduces a cut.

The improved decoder of [44] performs noticeably better than BLPD and SPAD. This is shown under the BIAWGNC on (3,4)-regular LDPC codes with $n=32,100,240$.

The cutting plane approach of [55] is based on an IP formulation of MLD, which is referred to as IPD. (Note that this formulation was already mentioned in [9].) Auxiliary variables $z \in \mathbb{Z}^{m}$ model the binary constraints $H x=0$ over
$\mathbb{F}_{2}$ in the real number field $\mathbb{R}^{n}$.

$$
\begin{array}{ll} 
& \min \lambda^{T} x \\
\text { s.t. } & H x-2 z=0 \\
& x \in\{0,1\}^{n}, z \in \mathbb{Z}^{m}
\end{array}
$$

In [55], the LP relaxation of IPD is the initial LP problem which is solved by a cutting plane algorithm. Note that the LP relaxation of IPD is not equivalent to the LP relaxations given in Section V. In almost all improved (in the errorcorrecting performance sense) LPD approaches reviewed in this article first the BLPD is run. If BLPD fails, some technique to improve BLPD is used with the goal of finding the ML codeword at the cost of increased complexity. In contrast, the approach by Tanatmis et al. in [55] does not elaborate on the solution of BLPD, but immediately searches for cuts which can be derived from arbitrary dual codewords. To this end, the parity-check matrix is modified and the conditions under which certain RPCs define cuts are checked. The average number of iterations performed and the average number of cuts generated in the separation algorithm decoding (SAD) of [55] are presented for the $(3,6)$ random regular codes with $n=40,80,160,200,400$ and for the $(31,10),(63,39),(127,99),(255,223) \mathrm{BCH}$ codes. Both performance measures seem to be directly proportional to the block length. The error-correcting performance of SAD is measured on the random regular $(3,4)$ LDPC codes with block length 100 and 200, and Tanner's $(155,64)$ group structured LDPC code [56]. It is demonstrated that the improved LPD approach of [55] performs better than BLPD applied in the adaptive setting [44] and better than SPAD. One significant numerical result is that SAD proposed in [55] performs much better than BLPD for the $(63,39)$ and $(127,99) \mathrm{BCH}$ codes, which have high-density parity check matrices. In all numerical simulations the BIAWGNC is used.

Yufit et al. [57] improve SAD [55] and ALPD [44] by employing several techniques. The authors propose to improve the error-correcting performance of these decoding methods by using RPC cuts derived from alternative parity-check matrices selected from the automorphism group of $C, \operatorname{Aut}(C)$. In the alternative parity-check matrices, the columns of the original parity-check matrix are permuted according to some scheme. At the first stage of Algorithm 1 of [57], SAD is used to solve the MLD problem. If the ML codeword is found then Algorithm 1 terminates, otherwise an alternative parity-check matrix from $\operatorname{Aut}(C)$ is randomly chosen and the SAD is applied again. In the worst case this procedure is repeated $N$ times where $N$ denotes a predetermined constant. A similar approach is also used to improve ALPD in Algorithm 2 of [57]. Yufit et al. enhance Algorithm 1 with two techniques to improve the error-correcting performance and complexity. The first technique, called parity-check matrix adaptation, is to alter the parity-check matrix prior to decoding such that at the columns of the parity-check matrix which correspond to least reliable bits, i.e., bits with the smallest absolute LLR values, unit vectors are obtained. The second technique, which is motivated by MALPD of [45], is to drop the inactive inequalities at each iteration of SAD, in order to avoid that the
problem size increases from iteration to iteration. Under the BIAWGNC, it is demonstrated on the $(63,36,11) \mathrm{BCH}$ code and the $(63,39,9) \mathrm{BCH}$ code that SAD can be improved both in terms of error-correcting performance and computational complexity.

## B. Facet guessing approaches

Based on BLPD2, Dimakis et al. [28] improve the errorcorrecting performance of BLPD with an approach similar to FDA (see Section IV). They introduce facet guessing algorithms which iteratively solve a sequence of related LP problems. Let $x^{*}$ be a non-integral optimal solution of BLPD, $x^{\mathrm{ML}}$ be the ML codeword, and $\mathcal{F}$ be a set of faces of $\mathcal{P}$ which do not contain $x^{*}$. This set $\mathcal{F}$ is given by the set of inequalities which are not active at $x^{*}$.

The set of active inequalities of a pseudocodeword $v$ is denoted by $\mathbb{A}(v)$. In facet guessing algorithms, the objective function $\lambda^{T} x$ is minimized over $f \cap \mathcal{P}$ for all $f \in \mathcal{K} \subseteq \mathcal{F}$ where $\mathcal{K}$ is an arbitrary subset of $\mathcal{F}$. The optimal solutions are stored in a list. In random facet guessing decoding (RFGD), $|\mathcal{K}|$ of the faces $f \in \mathcal{F}$ are chosen randomly. If $\mathcal{K}=\mathcal{F}$ then exhaustive facet guessing decoding (EFGD) is obtained. From the list of optimal solutions, the facet guessing algorithms output the integer solution with minimum objective function value. It is shown that EFGD fails if there exists a pseudocodeword $v \in f$ such that $\lambda^{T} v<\lambda^{T} x^{\mathrm{ML}}$ for all $f \in \mathbb{A}\left(x^{\mathrm{ML}}\right)$. For suitable expander codes this result is combined with the following structural property of expanderbased codes also proven by the authors. The number of active inequalities at some codeword is much higher than at a non-integral pseudocodeword. Consequently, theoretical bounds on the decoding success conditions of the polynomial time algorithms EFGD and RFGD for expander codes are derived. The numerical experiments are performed under the BIAWGNC, on Tanner's $(155,64)$ group-structured LDPC code and on a random LDPC code with $n=200, d_{v}=3$, $d_{c}=4$. For these codes the RFG algorithm performs better than the SPAD.

## C. Branch \& bound approaches

Linear programming based branch \& bound is an implicit enumeration technique in which a difficult optimization problem is divided into multiple, but easier subproblems by fixing the values of certain discrete variables. We refer to [7] for a detailed description. Several authors improved LPD using the branch \& bound approach.

Breitbach et al. [9] solved IPD by a branch \& bound approach. Depth-first and breadth-first search techniques are suggested for exploring the search tree. The authors point out the necessity of finding good bounds in the branch \& bound algorithm and suggest a neighborhood search heuristic as a means of computing upper bounds. In the heuristic, a formulation is used which is slightly different to IPD. We refer to this formulation as alternative integer programming decoding (AIPD). AIPD can be obtained by using error vectors. Let $\bar{y}=\frac{1}{2}(1-\operatorname{sign}(\lambda))$ be the hard decision for the LLR vector $\lambda$ obtained from the BIAWGNC. Comparing
$\bar{y} \in\{0,1\}^{n}$ with a codeword $x \in C$ results in an error vector $e \in\{0,1\}^{n}$, i.e., $e=\bar{y}+x(\bmod 2)$. Let $s=H \bar{y}$, and define $\bar{\lambda}$ by $\bar{\lambda}_{i}=\left|\lambda_{i}\right|$. IPD can be reformulated as

$$
\begin{array}{ll} 
& \min \bar{\lambda}^{T} e \quad(\mathrm{AIPD}) \\
\text { s.t. } & H e-2 z=s \\
e \in\{0,1\}^{n}, z \in \mathbb{Z}^{m} .
\end{array}
$$

In the neighborhood search heuristic of [9], first a feasible starting solution $e^{0}$ is calculated by setting the coordinates of $e^{0}$ corresponding to the $n-m$ most reliable bits (i. e., those $j \in J$ such that $\left|y_{j}\right|$ are largest) to 0 . These are the non-basic variables while the $m$ basic variables are found from the vector $s \in\{0,1\}^{m}$. Starting from this solution a neighborhood search is performed by exchanging basic and non-basic variables. The tuple of variables yielding a locally best improvement in the objective function is selected for iterating to the next feasible solution.

In [9], numerical experiments are performed under the BIAWGNC, on the $(31,21,5) \mathrm{BCH}$ code, the $(64,42,8)$ Reed-Muller code, the $(127,85,13) \mathrm{BCH}$ code and the $(255,173,23) \mathrm{BCH}$ code. The neighborhood search with single position exchanges performs very similar to MLD for the $(31,21,5) \mathrm{BCH}$ code. As the block length increases the error-correcting performance of the neighborhood search with single position exchanges gets worse. An extension of this heuristic allowing two position exchanges is applied to the $(64,42,8)$ Reed-Muller code, the $(127,85,13) \mathrm{BCH}$ code, and the $(255,173,23) \mathrm{BCH}$ code. The extended neighborhood search heuristic improves the error-correcting performance at the cost of increased complexity. A branch \& bound algorithm is simulated on the $(31,21,5) \mathrm{BCH}$ code and different search tree exploration schemes are investigated. The authors suggest a combination of depth-first and breadth-first search.

In [58], Draper et al. improve the ALPD approach of [44] with a branch \& bound technique. Branching is done on the least certain variable, i.e., $x_{j}$ such that $\left|x_{j}^{*}-0.5\right|$ is smallest for $j \in J$. Under the BSC, it is observed on Tanner's $(155,64,20)$ code that the ML codeword is found after few iterations in many cases.

In [36] two branch \& bound approaches for LDPC codes are introduced. In ordered constant depth decoding (OCDD) and ordered variable depth decoding (OVDD), first BLPD1 is solved. If the optimal solution $x^{*}$ is non-integral, a subset $\mathcal{T} \subseteq \mathcal{E}$ of the set of all non-integral bits $\mathcal{E}$ is chosen. Let $g=|\mathcal{T}|$. The subset $\mathcal{T}$ is constituted from the least certain bits. The term "ordered" in OCDD and OVDD is motivated by this construction. It is experimentally shown in [36] that choosing the least certain bits is advantageous in comparison to a random choice of bits. OVDD is a breadth first branch \& bound algorithm where the depth of the search tree is restricted to $g$. Since this approach is common in integer programming, we do not give the details of OVDD and refer to [7] instead. For OVDD, the number of LPs solved in the worst case is $2^{g+1}-1$

In OCDD, $m$-element subsets $\mathcal{M}$ of $\mathcal{T}$, i.e., $\mathcal{M} \subseteq \mathcal{T}$ and $m=|\mathcal{M}|$, are chosen. Let $b \in\{0,1\}^{m}$. For any $\mathcal{M} \subseteq \mathcal{T}, 2^{m}$

LPs are solved, each time adding a constraint block

$$
x_{k}=b_{k} \text { for all } k \in \mathcal{M}
$$

to BLPD1, thus fixing $m$ bits. Let $\hat{x}$ be the solution with the minimum objective function value among the $2^{m}$ LPs solved. If $\hat{x}$ is integral, OCDD outputs $\hat{x}$; otherwise another subset $\mathcal{M} \subseteq \mathcal{T}$ is chosen. Since OCDD exhausts all $m$-element subsets of $\mathcal{T}$, in the worst case $\binom{g}{m} 2^{m}+1$ LPs are solved.

The branch \& bound based improved LPD of Yang et al. [36] can be applied to LDPC codes with short block length. For the following numerical tests, the BIAWGNC is used. Under various settings of $m$ and $g$ it is shown on a random LDPC code with $n=60, R=\frac{1}{4}, d_{c}=4$, and $d_{v}=3$ that OCDD has a better error-correcting performance than BLPD and SPAD. Several simulations are done to analyze the trade-off between complexity and error-correcting performance of OCDD and OVDD. For the test instances and parameter settings ${ }^{6}$ used in [36] it has been observed on the above-mentioned code that OVDD outperforms OCDD. This behavior is explained by the observation that OVDD applies the branch \& bound approach on the most unreliable bits. On a longer random LDPC code with $n=1024, R=\frac{1}{4}, d_{c}=4$, and $d_{v}=3$, it is demonstrated that the OVDD performs better than BLPD and SPAD.

Another improved LPD technique which can be interpreted as a branch \& bound approach is randomized bit guessing decoding (RBGD) of Dimakis et al. [28]. RBGD is inspired from the special case that all facets chosen by RFGD (see Section VII-B) correspond to constraints of type $x_{j} \geq 0$ or $x_{j} \leq 1$. In RBGD, $k=c \log n$ variables, where $c>0$ is a constant, are chosen randomly. Because there are $2^{k}$ different possibile configurations of these $k$ variables, BLPD2 is run $2^{k}$ times with associated constraints for each assignment. The best integer valued solution in terms of the objective function $\lambda$ is the output of RBGD. Note that by setting $k$ to $c \log n$, a polynomial complexity in $n$ is ensured. Under the assumption that there exists a unique ML codeword, exactly one of the $2^{k}$ bit settings matches the bit configuration in the ML codeword. Thus, RBGD fails if a non-integral pseudocodeword with a better objective function value coincides with the ML codeword in all $k$ components. For some expander codes, the probablilty that the RBGD finds the ML codeword is given in [28]. To find this probability expression, the authors first prove that, for some expander-based codes, the number of nonintegral components in any pseudocodeword scales linearly in block length.

Chertkov and Chernyak [59] apply the loop calculus approach [60], [61] to improve BLPD. Loop calculus is an approach from statistical physics and related to cycles in the Tanner graph representation of a code. In the context of improved LPD, it is used to either modify objective function coefficients [59] or to find branching rules for branch and bound [62]. Given a parity-check matrix and a channel output, linear programming erasure decoding (LPED) [59] first solves BLPD. If a codeword is found then the algorithm terminates. If a non-integral pseudocodeword is found then a so-called critical loop is searched by employing loop calculus. The indices

[^5]of the variable nodes along the critical loop form an index set $\mathcal{M} \subseteq J$. LPED lowers the objective function coefficients $\lambda_{j}$ of the variables $x_{j}, j \in \mathcal{M}$, by multiplying $\lambda_{j}$ with $\epsilon$, where $0 \leq \epsilon<1$. After updating the objective function coefficients, BLPD is solved again. If BLPD does not find a codeword then the selection criterion for the critical loop is improved. LPED is tested on the list of pseudocodewords found in [35] for Tanner's $(155,64,20)$ code. It is demonstrated that LPED corrects the decoding errors of BLPD for this code.

In [62], Chertkov combines the loop calculus approach used in LPED [59] with RFGD [28]. We refer to the combined algorithm as loop guided guessing decoding (LGGD). LGGD differs from RFGD in the sense that the constraints chosen are of type $x_{j} \geq 0$ or $x_{j} \leq 1$ where $j$ is in the index set $M$, the index set of the variable nodes in the critical loop. LGGD starts with solving BLPD. If the optimal solution is nonintegral then the critical loop is found with the loop calculus approach. Next, a variable $x_{j}, j \in M$, is selected randomly and two partial LPD problems are deduced. These differ from the original problem by only one equality constraint $x_{j}=0$ or $x_{j}=1$. LGGD chooses the minimum of the objective values of the two subproblems. If the corresponding pseudocodeword is integral then the algorithm terminates. Otherwise the equality constraints are dropped, a new $j \in M$ along the critical loop is chosen, and two new subproblems are constructed. If the set $M$ is exhausted, the selection criterion of the critical loop is improved. LGGD is very similar to OCDD of [36] for the case that $g=|M|$ and $m=1$. In LGGD branching is done on the bits in the critical loop whereas in OCDD branching is done on the least reliable bits. As in [59], LGGD is tested on the list of pseudocodewords generated in [35] for Tanner's $(155,64,20)$ code. It is shown that LGGD improves BLPD under the BIAWGNC.

SAD of [55] is improved in terms of error-correcting performance by a branch \& bound approach in [57]. In Algorithm 3 of [57], first SAD is employed. If the solution is non-integral then a depth-first branch \& bound is applied. The non-integral valued variable with smallest LLR value is chosen as the branching variable. Algorithm 3 terminates as soon as the search tree reaches the maximally allowed depth $D_{p}$. Under the BIAWGNC, on the $(63,36,11)$ BCH code and the $(63,39,9) \mathrm{BCH}$ code Yufit et al. [57] demonstrate that the decoding performance of Algorithm 3 (enhanced with paritycheck matrix adaptation) approaches MLD.

## VIII. Conclusion

In this survey we have shown how the decoding of binary linear block codes benefits from a wide range of concepts which originate from mathematical optimization-mostly linear programming, but also quadratic (nonlinear) and integer programming, duality theory, branch \& bound methods, Lagrangian relaxation, network flows, and matroid theory. Bringing together both fields of research does lead to promising new algorithmic decoding approaches as well as deeper structural understanding of linear block codes in general and special classes of codes-like LDPC and turbo-like codes-in particular. The most important reason for the success of this
connection is the formulation of MLD as the minimization of a linear function over the codeword polytope $\operatorname{conv}(C)$. We have reviewed a variety of techniques of how to approximate this polytope, whose description complexity in general is too large to be computed efficiently.

For further research on LPD of binary linear codes, two general directions can be distinguished. One is to decrease the algorithmic complexity of LPD towards reducing the gap between LPD and IMPD, the latter of which still outperforms LPD in practice. The other direction aims at increasing errorcorrecting performance, tightening up to MLD performance. This includes a continued study of RPCs as well as the characterization of other, non-RPC facet-defining inequalities of the codeword polytope.

There are other lines of research related to LPD and IMPD which are not covered in this article. Flanagan et al. [21] have generalized LP decoding, along with several related concepts, to nonbinary linear codes. Another possible generalization is to extend to different channel models [22]. Connecting two seemingly different decoding approaches, structural relationship between LPD and IMPD has been discussed in [63]. Moreover, the discovery that both decoding methods are closely related to the Bethe free energy approximation, a tool from statistical physics, has initiated vital research [64]. Also, of course, research on IMPD itself, independent of LPD, is still ongoing with high activity. A promising direction of research is certainly the application of message passing techniques to mathematical programming problems beyond LPD [65].

## Acknowledgment

We would like to thank Pascal O. Vontobel and Frank Kienle for their comments and suggestions. We also thank the anonymous referees for the helpful reviews.

## REFERENCES

[1] J. Feldman, "Decoding Error-Correcting Codes via Linear Programming," Ph.D. dissertation, Massachusetts Institute of Technology, 2003.
[2] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," IEEE Trans. Inf. Theory, vol. 51, pp. 954-972, 2005.
[3] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," IEEE Trans. Inf. Theory, vol. 47, pp. 498-519, 2001.
[4] S. M. Aji and R. J. McEliece, "The generalized distributive law," IEEE Trans. Inf. Theory, vol. 46, pp. 325-343, 2000.
[5] N. Wiberg, "Codes and decoding on general graphs," Ph.D. dissertation, Linköping University, 1996.
[6] H.-A. Loeliger, "An introduction to factor graphs," IEEE Signal Process. Mag., vol. 21, no. 1, pp. 28-41, Jan. 2004.
[7] G. L. Nemhauser and L. A. Wolsey, Integer and Combinatorial Optimization. Wiley-Interscience series in discrete mathematics and optimization, John Wiley \& Sons, 1988.
[8] M. Grötschel, L. Lovász, and A. Schrijver, Geometric algorithms and combinatorial optimization. Berlin Heidelberg: Springer-Verlag, 1988.
[9] M. Breitbach, M. Bossert, R. Lucas, and C. Kempter, "Soft-decision decoding of linear block codes as optimization problem," Eur. Trans. Telecommun., vol. 9, pp. 289-293, 1998.
[10] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems," IEEE Trans. Inf. Theory, pp. 954-972, 1978.
[11] R. M. Karp, "Reducibility among combinatorial problems," in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, 1972, pp. 85-103.
[12] A. Vardy, "The intractability of computing the minimum distance of a code," IEEE Trans. Inf. Theory, vol. 43, pp. 1757-1766, 1997.

TABLE I
List of Abbreviations

| AIPD | alternative integer programming decoding |
| :--- | :--- |
| ALPD | adaptive linear programming decoding |
| BCQPD | box constrained quadratic programming decoding |
| BIAWGNC | binary input additive white Gaussian noise channel |
| BLPD | bare linear programming decoding |
| BSC | binary symmetric channel |
| CLPD | cascaded linear programming decoding |
| EFGD | exhaustive facet guessing decoding |
| FDA | fractional distance algorithm |
| FFG | Forney style factor graph |
| FS | forbidden set |
| GSA | greedy separation algorithm |
| IMPD | iterative message passing decoding |
| IP | integer programming |
| IPD | integer programming decoding |
| JSD | joint subgradient decoding |
| LLR | log likelihood ratio |
| LDPC | low-density parity-check |
| LGGD | loop guided guessing decoding |
| LP | linear programming |
| LPD | linear programming decoding |
| LPED | linear programming erasure decoding |
| MALPD | modified adaptive linear programming decoding |
| ML | maximum likelihood |
| MLD | maximum likelihood decoding |
| MSAD | min-sum algorithm decoding |
| OCDD | ordered constant depth decoding |
| OVDD | ordered variable depth decoding |
| PGPM | parallel gradient projection method |
| PLPD | primal linear programming decoding |
| PSORM | projected successive overrelaxation method |
| RA | repeat accumulate |
| RBGD | randomized bit guessing decoding |
| RFGD | randomized facet guessing decoding |
| RPC | redundant parity-check |
| SAD | separation algorithm decoding |
| SD | subgradient decoding |
| SDLPD | softened dual linear programming decoding |
| SNR | signal-to-noise ratio |
| SPAD | sum-product algorithm decoding |
| SPC | single parity-check |
| TCLPD | turbo code linear programming decoding |
|  |  |

$[13]$ D. J. A. Welsh, "Combinatorial problems in matroid theory"" in Combinatorial Mathematics and its Applications, D. J. A. Welsh, Ed. London, U.K.: Academic Press, 1971, pp. 291-307.
[14] J. G. Oxley, Matroid Theory. Oxford University Press, 1992.
[15] D. J. A. Welsh, Matroid Theory. L. M. S. Monographs. Academic Press, 1976.
[16] N. Kashyap, "A decomposition theory for binary linear codes," IEEE Trans. Inf. Theory, vol. 54, pp. 3035-3058, 2008.
[17] F. Barahona and M. Grötschel, "On the cycle polytope of a binary matroid," J. Comb. Theory Ser. B, vol. 40, pp. 40-62, 1986.
[18] N. Kashyap, "On the convex geometry of binary linear codes," in Proc. Inaugural UC San Diego Workshop Inf. Theory Appl., La Jolla, CA, Feb. 2006. [Online]. Available: http://ita.ucsd.edu/workshop/06/talks
[19] M. Grötschel and K. Truemper, "Decomposition and optimization over cycles in binary matroids," J. Comb. Theory Ser. B, vol. 46, pp. 306-337, 1989.
[20] P. D. Seymour, "Decomposition of regular matroids," J. Comb. Theory Ser. B, vol. 28, pp. 305-359, 1980.
[21] M. Flanagan, V. Skachek, E. Byrne, and M. Greferath, "Linearprogramming decoding of nonbinary linear codes," IEEE Trans. Inf. Theory, vol. 55, no. 9, pp. 4134-4154, Sep. 2009.
[22] A. Cohen, F. Alajaji, N. Kashyap, and G. Takahara, "LP decoding for joint source-channel codes and for the non-ergodic Polya channel," IEEE Comтип. Lett., vol. 12, no. 9, pp. 678-680, 2008.
[23] S. Lin and D. Costello, Jr., Error Control Coding, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, Inc., 2004.
[24] P. O. Vontobel and R. Koetter, "Graph-cover decoding and finite-
length analysis of message-passing iterative decoding of LDPC codes," arXiv:cs/0512078v1 [cs.IT], 2005.
[25] G. D. Forney, Jr., R. Koetter, F. R. Kschischang, and A. Reznik, "On the effective weights of pseudocodewords for codes defined on graphs with cycles," in Codes, systems, and graphical models, ser. IMA Vol. Math. Appl., B. Marcus and J. Rosenthal, Eds., vol. 123. Springer Verlag, New York, Inc., 2001, pp. 101-112.
[26] M. Chertkov and M. Stepanov, "Polytope of correct (linear programming) decoding and low-weight pseudo-codewords," in Proc. IEEE Int. Symp. Inform. Theory, St. Petersburg, Russia, Jul. / Aug. 2011, pp. 16481652.
[27] A. Schrijver, Theory of linear and integer programming. John Wiley \& Sons, 1986.
[28] A. G. Dimakis, A. A. Gohari, and M. J. Wainwright, "Guessing facets: Polytope structure and improved LP decoding," IEEE Trans. Inf. Theory, vol. 55, pp. 4134-4154, 2009.
[29] M. Grötschel, "Cardinality homogeneous set systems, cycles in matroids, and associated polytopes," in The Sharpest Cut: The Impact of Manfred Padberg and His Work, MPS-SIAM, pp. 99-120, 2004.
[30] R. G. Gallager, "Low-density parity-check codes," IRE Trans. Inf. Theory, vol. 8, no. 1, pp. 21-28, 1962.
[31] G. D. Forney Jr., "Codes on graphs: Normal realizations," IEEE Trans. Inf. Theory, vol. 47, pp. 529-548, 2001.
[32] J. Feldman and C. Stein, "LP decoding achieves capacity"" in Proc. 16th annual ACM-SIAM symp. on discrete algorithms. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, Jan. 2005, pp. 460-469.
[33] M. Yannakakis, "Expressing combinatorial optimization problems by linear programs," J. Comput. Syst. Sci., vol. 43, pp. 441-466, 1991.
[34] K. Yang, X. Wang, and J. Feldman, "A new linear programming approach to decoding linear block codes," IEEE Trans. Inf. Theory, vol. 54, pp. 1061-1072, 2008.
[35] M. Chertkov and M. Stepanov, "Pseudo-codeword landscape," in Proc. IEEE Int. Symp. Inform. Theory, Nice, France, Jun. 2007, pp. 15461550.
[36] K. Yang, J. Feldman, and X. Wang, "Nonlinear programming approaches to decoding low-density parity-check codes," IEEE J. Sel. Areas Commun., vol. 24, pp. 1603-1613, 2006.
[37] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," IEEE Trans. Commun., vol. 44, no. 10, pp. 1261-1271, Oct. 1996.
[38] C. Berrou, A. Graell i Amat, Y. Ould-Cheikh-Mouhamedou, and Y. Saouter, "Improving the distance properties of turbo codes using a third component code: 3D turbo codes," IEEE Trans. Commun., vol. 57, no. 9, pp. 2505-2509, Sep. 2009.
[39] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, Network Flows. PrenticeHall, 1993.
[40] J. Feldman and D. R. Karger, "Decoding turbo-like codes via linear programming," J. Comput. Syst. Sci., vol. 68, pp. 733-752, Jun. 2004.
[41] N. Halabi and G. Even, "Improved bounds on the word error probability of RA(2) codes with linear-programming-based decoding," IEEE Trans. Inf. Theory, vol. 51, no. 1, pp. 265-280, Jan. 2005.
[42] I. Goldenberg and D. Burshtein, "Error bounds for repeat-accumulate codes decoded via linear programming," in Proc. 6th Int. Symp. Turbo Codes and Iterative Inf.. Proc., Brest, France, Sep. 2010, pp. 487-491.
[43] A. Tanatmis, S. Ruzika, and F. Kienle, "A Lagrangian relaxation based decoding algorithm for LTE turbo codes," in Proc. 6th Int. Symp. Turbo Codes and Iterative Inf.. Proc., Brest, France, Sep. 2010, pp. 369-373.
[44] M. H. Taghavi and P. H. Siegel, "Adaptive methods for linear programming decoding," IEEE Trans. Inf. Theory, vol. 54, pp. 5396-5410, 2008.
[45] M. H. Taghavi, A. Shokrollahi, and P. H. Siegel, "Efficient implementation of linear programming decoding," IEEE Trans. Inf. Theory, vol. 57, no. 9, pp. 5960-5982, 2011.
[46] P. O. Vontobel and R. Kötter, "On low-complexity linear-programming decoding of LDPC codes," Eur. Trans. Telecommun., vol. 18, pp. 509517, 2007.
[47] D. Burshtein, "Iterative approximate linear programming decoding of LDPC codes with linear complexity," IEEE Trans. Inf. Theory, vol. 55, no. 11, pp. 4835-4859, Nov. 2009.
[48] D. Bertsimas and D. J. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods. Athena Scientific, 1997.
[49] P. O. Vontobel, "Interior-point algorithms for linear-programming decoding," in Proc. Inf. Theory and Applications Workshop. La Jolla, CA: UC San Diego, Jan. 2008, pp. 433-437.
[50] T. Wadayama, "An LP decoding algorithm based on primal pathfollowing interior point method," in Proc. IEEE Int. Symp. Inform. Theory, Seoul, Korea, Jun./Jul. 2009, pp. 389-393.
[51] N. Karmarkar, "A new polynomial-time algorithm for linear programming," Combinatorica, vol. 4, pp. 373-395, 1984.
[52] L. Lovasz and A. Schrijver, "Cones of matrices and set-functions and 0-1 optimization," SIAM Journal on Optimization, vol. 1, pp. 166-190, 1991.
[53] M. Grötschel and K. Truemper, "Master polytopes for cycles in binary matroids," Linear Algebra and its Applications, vol. 114/115, pp. 523540, 1989.
[54] M. Miwa, T. Wadayama, and I. Takumi, "A cutting-plane method based on redundant rows for improving fractional distance," IEEE J. Sel. Areas Comтип., vol. 27, no. 6, pp. 1005-1012, Aug. 2009.
[55] 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. Inf. Theory, vol. 56, no. 7, pp. 3277-3289, 2010.
[56] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja, and D. Costello, Jr., "LDPC block and convolutional codes based on circulant matrices," IEEE Trans. Inf. Theory, vol. 50, no. 12, pp. 2966-2984, 2004.
[57] A. Yufit, A. Lifshitz, and Y. Be'ery, "Efficient linear programming decoding of HDPC codes," IEEE Trans. Commun., vol. 59, no. 3, pp. 758-766, Mar. 2011.
[58] S. C. Draper, J. S. Yedidia, and Y. Wang, "ML decoding via mixedinteger adaptive linear programming," in Proc. IEEE Int. Symp. Inform. Theory, Nice, France, Jun. 2007, pp. 1656-1660.
[59] M. Chertkov and V. Y. Chernyak, "Loop calculus helps to improve belief propagation and linear programming decodings of low-density-parity-check codes," in Proc. 44th Allerton Conf. Commun., Control, Computing, Monticello, IL, Sep. 2006.
[60] - , "Loop calculus in statistical physics and information science," Phys. Rev. E, vol. 73, no. 6, p. 065102, Jun. 2006. [Online]. Available: arXiv.org:cond-mat/0601487
[61] -, "Loop series for discrete statistical models on graphs," J. Stat. Mech. Theor. Exp., vol. 2006, p. P06009, 2006. [Online]. Available: arXiv.org:cond-mat/0603189
[62] M. Chertkov, "Reducing the error floor," in IEEE Information Theory Workshop, Sep. 2007, pp. 230-235.
[63] P. O. Vontobel and R. Koetter, "On the relationship between linear programming decoding and min-sum algorithm decoding," in Proc. Int. Symp. Inform. Theory, Parma, Italy, Oct. 2004, pp. 991-996.
[64] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Constructing free-energy approximations and generalized belief propagation algorithms," IEEE Trans. Inf. Theory, vol. 51, no. 7, pp. 2282-2312, 2005.
[65] M. Bayati, D. Shah, and M. Sharma, "Max-product for maximum weight matching: Convergence, correctness, and LP duality," IEEE Trans. Inf. Theory, vol. 54, no. 3, pp. 1241-1251, 2008.

Michael Helmling Michael Helmling received the Diploma in Mathematics from the University of Kaiserslautern, Germany, in 2011.

Since 2011, he has been a Ph. D. student with the Optimization Research Group, Department of Mathematics, University of Kaiserslautern.

Stefan Ruzika Stefan Ruzika received a M. S. degree in mathematics from Clemson University, SC, in 2002 and the M. S. and Ph. D. degrees in mathematics from the University of Kaiserslautern in 2003 and 2007, respectively.

Since 2008 he is assistant professor at the University of Kaiserslautern. His research interests include coding theory, combinatorial optimization and multiple objective programming.

Akin Tanatmis Akin Tanatmis received the B. Sc. degree in Industrial Engineering from Bilkent University, Turkey, in 2002 and the Diploma in Mathematics and Ph. D. degrees from the University of Kaiserslautern in 2006 and 2011, respectively.


[^0]:    This work was supported in part by the Center of Mathematical and Computational Modeling of the University of Kaiserslautern.
    M. Helmling, S. Ruzika and A. Tanatmis are with Department of Mathematics, University of Kaiserslautern, Erwin-Schroedinger-Strasse, 67663 Kaiserslautern, Germany. Email: \{helmling, ruzika\}@mathematik.uni-kl.de
    ${ }^{1}$ See the table on page 16 for a list of the acronyms used.

[^1]:    ${ }^{2}$ Note that single vectors in this paper are generally column vectors; however, in coding theory they are often used as rows of matrices. The transposition of column vector $a$ makes it a row vector, denoted by $a^{T}$.

[^2]:    ${ }^{3}$ In fact, Flanagan et al. [21] have recently generalized a substantial portion of the LPD theory to the nonbinary case. Similarly, work has been done to include channels with memory; see e.g. [22].

[^3]:    ${ }^{4}$ We intentionally do not discuss trellis termination here and assume that the encoder always ends in a fixed terminal state; cf. [23] for details.

[^4]:    ${ }^{5}$ Using exactly two constituent convolutional encoders eases notation and is the most common case, albeit not being essential for the concept-in fact, recent development suggest that the error-correcting performance benefits from adding a third encoder [38].

[^5]:    ${ }^{6}$ The parameters $m$ and $g$ are chosen such that OVDD and OCDD have similar worst case complexity.

