On Adaptive Linear Programming Decoding of Linear Codes Over GF(8)

Eirik Rosnes† and Michael Helmling‡

†Department of Informatics, University of Bergen, N-5020 Bergen, Norway, and the Simula Research Lab
Email: eirik@ii.uib.no
‡Fraunhofer Institute for Industrial Mathematics ITWM, 67663 Kaiserslautern, Germany
E-mail: michael.helmling@itwm.fraunhofer.de

Abstract—In this work, we consider adaptive linear programming (LP) decoding of linear codes over GF(8). In particular, we give explicit constructions of valid inequalities (using no auxiliary variables) for the codeword polytope (or the convex hull) of the so-called constant-weight embedding of a single parity-check code over GF(8) that all are facet-defining. We conjecture that these inequalities together with so-called simplex constraints give a complete and irredundant description of the embedded (under the constant-weight embedding) codeword polytope. Furthermore, these sets of inequalities are used to develop an efficient (as compared to a static approach) exact (assuming that the conjecture is true) adaptive LP decoder for linear codes over GF(8). Numerical results show that only a very small subset of these inequalities is necessary for achieving close to exact LP decoding performance.

I. INTRODUCTION

Linear programming (LP) decoding of binary linear codes was introduced by Feldman et al. in 2005 [1] as an efficient, but suboptimal decoding approach. In particular, LP decoding of low-density parity-check codes has attracted significant attention and several low-complexity approaches have been proposed during the last decade [2–6]. In 2009, Flanagan et al. [7] extended the LP decoding algorithm to nonbinary linear codes, and since then several low-complexity approaches have been proposed [8–10].

Adaptive LP (ALP) decoding, first introduced in [3] for binary codes, is an efficient approach to LP decoding in which only a small number of inequalities is added adaptively to the LP formulation. Initially, there are no constraints in the LP formulation. Then, the optimal LP solution is found after which a search for (in)equalities that violate the current LP solution (or so-called cuts) is performed. This is done by means of a tailor-made separation algorithm. The discovered inequalities are added to the LP formulation and then the program is solved again. This process continues iteratively until no more cuts can be added. ALP decoding is equivalent to a static approach in which all inequalities are added to the LP formulation from the beginning. However, with ALP decoding the average size (in terms of the number of variables and constraints) is much lower than with a static formulation, and despite the fact that several linear programs need to be solved with ALP decoding, the overall complexity is much lower. In particular, a plain static approach becomes infeasible for moderately dense codes, since the number of inequalities from a single single parity-check (SPC) code is exponential in its length. To a certain extent this can be overcome by using a cascaded static approach in which a parity-check equation of high degree is decomposed into a set of parity-check equations of degree at most 3 [7, Sec. IX]. However, ALP decoding is more efficient and known to be the fastest nonapproximate LP decoding method for binary codes.

In [11], ALP decoding of ternary linear codes was considered. In particular, an explicit description (in terms of linear (in)equalities and without using auxiliary variables) of the embedded codeword polytope of a ternary SPC code was provided and this description was used to develop the ALP decoding algorithm. Except for binary and ternary codes, no exact ALP decoding method is currently known for general codes. In this work, we extend some of the results from [11] to codes over GF(8). We remark here that for SPC codes over GF(4), it has been conjectured that its exact convex hull (under the constant-weight embedding) can be described by Feldman-type (or forbidden set) inequalities [10, Conj. 62] (see also [12]). Thus, exact (assuming that the conjecture is true) LP decoding can be achieved using the ALP decoding method of [3]. The conjecture can be verified to be true for small values of the SPC code length, using, for instance, the software package Polymake [13].

In this text, all proofs are omitted for brevity.

II. NOTATION AND BACKGROUND

Let $C$ denote a linear code of length $n$ and dimension $k$ over the finite field $\mathbb{F}_q$ with $q = p^m$, where $p$ is a prime and $m$ is a positive integer, elements. The code $C$ can be defined by an $r \times n$ parity-check matrix $H$, where $r \geq n-k$ and each matrix entry $h_{j,i} \in \mathbb{F}_q$, $i \in \mathcal{I}$ and $j \in \mathcal{J}$, where $\mathcal{I}$ is the column index set of $H$ and $\mathcal{J}$ is the row index set of $H$. Then, $C = \mathcal{C}(H) = \{ c = (c_1, \ldots, c_n)^T \in \mathbb{F}_q^n : Hc = 0 \}$, where $(\cdot)^T$ denotes the transpose of its vector argument. When represented by a factor graph, $\mathcal{I}$ is also the variable node index set and $\mathcal{J}$ is the check node index set. In the following, let $\mathcal{N}_v(i)$ (resp.
\( N_c(j) \) denote the set of neighboring nodes of variable node \( i \) (resp. check node \( j \)), and define \( [L] = \{1, 2, \ldots, L\} \) for any positive integer \( L \). Finally, call \( C \) an \((n, k)\) code (over some finite field) if its length is \( n \) and its dimension is \( k \).

For any field element \( \zeta \in F_q \) and any finite set \( A = \{\zeta_1, \ldots, \zeta_{|A|}\} \), \( \zeta_i \in F_q \), \( i \in [|A|] \), the summation \( \zeta + A \) is defined as

\[
\zeta + A = \{ \zeta + \zeta_1, \ldots, \zeta + \zeta_{|A|} \}.
\]

Furthermore, we use the short-hand notation \( \sum A = \sum_{a \in A} a \) for the sum (in \( F_q \)) of the elements in \( A \).

**Definition 1:** Any two finite sets \( A_1 = \{\zeta_1^{(1)}, \ldots, \zeta_{|A_1|}^{(1)}\} \), \( \zeta_i^{(1)} \in F_q \), \( i \in [|A_1|] \) and \( A_2 = \{\zeta_1^{(2)}, \ldots, \zeta_{|A_2|}^{(2)}\} \), \( \zeta_i^{(2)} \in F_q \), \( i \in [|A_2|] \) of equal size \( |A_1| = |A_2| \) are said to match perfectly if the sets, when appropriately ordered as tuples/arrays, add up to zero, i.e., there exists an order of the elements of \( A_1 \) and \( A_2 \) such that \( \zeta_i^{(1)} + \zeta_i^{(2)} = 0 \) (in \( F_q \)) for all \( i \in [|A_1|] \), where the order of the elements of \( A_1 \) (resp. \( A_2 \)) is according to \( \zeta_1^{(1)}, \ldots, \zeta_{|A_1|}^{(1)} \) (resp. \( \zeta_1^{(2)}, \ldots, \zeta_{|A_2|}^{(2)} \)).

Note that it follows from Definition 1 that for fields of characteristic 2, two sets match perfectly if and only if the sets are equal.

**Example 1:** Let \( A_1 = \{1, 1 + \alpha, 1 + \alpha + \alpha^2, 1\} \), \( A_2 = \{1, 1 + \alpha + \alpha^2, 1\} \), and \( A_3 = \{1, 1 + \alpha, 1 + \alpha + \alpha^2\} \) denote three sets of field elements of \( F_q \), where \( \alpha \) denotes a primitive element. Then, according to Definition 1, \( A_1 \) and \( A_2 \) match perfectly. However, \( A_1 \) and \( A_3 \) is not a perfect match, since \( \alpha + A_1 \) does not contain the zero element.

In this work, we will represent each element in \( F_{p^n} \) using an integer in \( \{0, \ldots, p^n - 1\} \). Without loss of generality, \( \zeta \in F_{p^n} \) can be represented by a polynomial \( \zeta(x) = \sum_{i=1}^{n} p_i x^{i-1} \), where \( p_i \in F_p \), and we use the integer representation \( \zeta(p) = \sum_{i=1}^{n} p_i p^{i-1} \).

In the original work by Feldman et al. [1], the maximum-likelihood (ML) decoding problem was stated as an integer problem in the real space by using the obvious embedding of \( F_2 \cong \{0, 0\} \), \( F_2 \cong \{0, 1, 0\} \), and then relaxed into a linear program using vectors that live in \( [0, 1]^n \). In the nonbinary case, the obvious generalization that represents \( \zeta \in F_q \) into the reals by using its integer representation \( \zeta(p) \in \mathbb{R} \) does not work out for several reasons. Instead, the following mapping \( f(\cdot) \) (see [10, 12, 14]) embeds elements of \( F_q \) into the Euclidean space of dimension \( q \).

**Definition 2:** We define the constant-weight embedding of elements of \( F_q \) by

\[
f : F_q \rightarrow \{0, 1\}^q
\]

\[
\zeta \mapsto \mathbf{x} = (x_0, \ldots, x_{q-1})
\]

where \( x_0 = 1 \) if \( \delta = \zeta \) and \( x_0 = 0 \) otherwise.

**Example 2:** Consider \( q = 2^3 \). Then, the constant-weight embedding of Definition 2 is as follows:

\[
0 \mapsto (1, 0, 0, 0, 0, 0, 0, 0),
\]

\[
1 \mapsto (0, 1, 0, 0, 0, 0, 0, 0),
\]

\[
\alpha \mapsto (0, 0, 1, 0, 0, 0, 0, 0),
\]

\[
1 + \alpha \mapsto (0, 0, 0, 1, 0, 0, 0, 0),
\]

\[
\alpha^2 \mapsto (0, 0, 0, 0, 1, 0, 0, 0),
\]

\[
1 + \alpha^2 \mapsto (0, 0, 0, 0, 0, 1, 0, 0),
\]

\[
\alpha + \alpha^2 \mapsto (0, 0, 0, 0, 0, 0, 1, 0),
\]

\[
1 + \alpha + \alpha^2 \mapsto (0, 0, 0, 0, 0, 0, 0, 1),
\]

where \( \alpha \) is a primitive element.

Using the above definition, a column vector \( \mathbf{c} \in \mathbb{F}_{q^n} \) can be mapped to a binary vector \( \mathbf{f}(\mathbf{c}) = (f(c_1), \ldots, f(c_n))^T \) of length \( qn \), where \( (v_1, \ldots, v_n) \) denotes the concatenation of row vectors \( v_1, \ldots, v_n \). Finally, we denote the image of the set \( \mathcal{C} \) under the mapping \( \mathbf{f}_v \) by \( \mathbf{f}_v(\mathcal{C}) = \{\mathbf{f}_v(\mathbf{c}) : \mathbf{c} \in \mathcal{C} \} \).

**Remark 1:** Motivated by the definition of \( f \), we will use zero-based indexing for all (sub-)vectors that are defined as embeddings of (vectors of) field elements of \( F_q \). In fact, we will use field elements and their integer representation interchangeably for indexing such vectors.

Observe that \( f \) maps the elements of \( F_q \) to the vertices of the full-dimensional standard \((q-1)\)-simplex embedded in \( \mathbb{R}^q \), \( S_{q-1} : = \text{conv} \{e_i^q \}_{i=1}^{q} \), where \( \text{conv} \) denotes the convex hull in the real space of its argument and \( e_i \) is the \( i \)-th unit vector in \( \mathbb{R}^q \). Hence, \( \mathbf{f}_v \) maps \( F_q^n \) onto the vertices of \( S_{q-1} \times S_{q-1} \) (\( n \) times).

For any vector \( p \) of length \( qn \) that is related to the embedding of elements of \( F_q^n \) under \( \mathbf{f}_v \), we will mostly use double-indexing to emphasize on the \( q \)-blocks \( p_i \) of \( p \), as in

\[
p = (p_1, \ldots, p_n) = (p_{1,0}, \ldots, p_{1,q-1}, \ldots, p_{n,0}, \ldots, p_{n,q-1}).
\]

In line with Remark 1, the first index \( i \) of \( p_{i,j} \) (indicating a codeword coordinate) starts at 1, while the second (corresponding to an embedded element of \( F_q^n \)) starts with 0.

**Definition 3:** For the finite field \( F_q \), and \( d \geq 1 \), let \( \Delta^d \) denote the set of \( q \times d \) inequalities and \( d \) equations in \( \mathbb{R}^{qd} \) that define \( S_{q-1} \), i.e., the inequalities

\[
x_{i,j} \geq 0 \quad \text{for } i \in [d] \text{ and } j \in F_q \quad (3)
\]

and

\[
\sum_{j=0}^{q-1} x_{i,j} = 1 \quad \text{for } i \in [d];
\]

we call the (in)equalities in \( \Delta^d \) simplex constraints.

The simplex constraints \( \Delta^d \) can be interpreted as generalized box constraints that restrict, for \( i \in [d] \), the \( q \) variables representing \( f(c_i) \) to the simplex \( S_{q-1} \). where \( (c_1, \ldots, c_d)^T \) denotes a codeword of an SPC code of length \( d \). As they are independent of \( H \), an arbitrary code \( C \) of length \( n \) thus has only \( n(q+1) \) simplex constraints \( (n \) equations and \( qn \) inequalities \) in total. These will be denoted by \( \Delta^d_q \).

**A. Linear Programming Decoding of Nonbinary Codes**

In this subsection, we review the LP decoding formulation proposed by Flanagan et al. in [7], where in contrast to [7] we use constant-weight embedding. Let \( F_q \) and \( \Sigma \) respectively, denote the input and output alphabets of a memoryless channel
Furthermore, we define $\Lambda(y) = (\lambda(y_1), \ldots, \lambda(y_n))^T$ for $y = (y_1, \ldots, y_n)^T$. Now, the ML decoding problem can be written as [7]

$$
\hat{x}_{\text{ML}} = \arg \min_{x \in \mathcal{C}} \sum_{i=1}^{n} \log \left( \frac{\Pr(Y = y_i | X = 0)}{\Pr(Y = y_i | X = c_i)} \right)
$$

(5)

where $y_1, \ldots, y_n$ are the channel outputs. The problem in (5) can be relaxed into a linear program using the embedding from Definition 2 as follows [7]:

$$
\hat{x}_{\text{LP}} = \arg \min_{x \in \mathcal{F}} \Lambda_v(y)^T x
$$

s.t. $x^{(j)} = P_j x \in \conv(\mathcal{F}_v(C_j)), \forall j \in J$

(6)

where $x^{(j)} = (x_{1}^{(j)}, \ldots, x_{\mathcal{N}(j)}^{(j)})^T$, $x_{i}^{(j)} = (x_{i,0}^{(j)}, \ldots, x_{i,q-1}^{(j)})$ for all $i \in [\mathcal{N}(j)]$, and $P_j$ is a binary indicator matrix that selects the variables from $x$ that participate in the $j$-th check node. In (6), $C_j$ represents the SPC code defined by the $j$-th check node.

LP decoding, i.e., using the LP relaxation (6) as a decoder (which is defined to output a decoding failure if the optimal solution $\hat{x}_{\text{LP}}$ does not happen to be integral) preserves desirable properties. Of most importance is the so-called ML certificate property [1, 7] which says that, if $\hat{x}_{\text{LP}}$ is a codeword, then $\hat{x}_{\text{LP}} = \hat{x}_{\text{ML}}$, i.e., equal to the ML estimate.

Note that the ML certificate property remains to hold if $\conv(\mathcal{F}_v(C_j))$ is replaced by a relaxation $\mathcal{Q}_j \supset \conv(\mathcal{F}_v(C_j))$. We will use the term LP decoding also when such a further relaxation is used.

B. Background on Polyhedra

The convex hull of a finite number of points in $\mathbb{R}^n$ is called a polytope. It can be alternatively characterized as the (bounded) intersection of a finite number of half-spaces, i.e., the solution space of a finite number of linear inequalities.

Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a polytope. An inequality $\theta^T x \leq \kappa$ with $\theta \in \mathbb{R}^n$ and $\kappa \in \mathbb{R}$ is valid for $\mathcal{P}$ if it holds for any $x \in \mathcal{P}$. Every valid inequality defines a face $F = \{x \in \mathcal{P} : \theta^T x = \kappa\}$ of $\mathcal{P}$, which is itself a polytope. For notational convenience, we will identify a face $F$ with its defining inequality $\theta^T x \leq \kappa$ as long as there is no risk of ambiguity.

The dimension of a face (or polytope) $F$ is defined as the dimension of its affine hull $\text{aff}(F)$, which is calculated as one less than the maximum number of affinely independent vectors in $F$. A face $F$ with $\dim(F) = \dim(\mathcal{P}) - 1$ is called a facet, while a zero-dimensional face is a vertex of $\mathcal{P}$. It is a basic result of polyhedral theory that a face $F$ of dimension $\dim(F)$ actually contains $\dim(F)$ affinely independent vertices of $\mathcal{P}$.

Conversely, a face $F$ is uniquely determined by $\dim(F)$ affinely independent vertices of $\mathcal{P}$ that are contained in $F$.

Facets are important because every “minimal” representation of a polytope $\mathcal{P}$ is of the form

$$
\mathcal{P} = \{x \in \mathbb{R}^n : Ax = b, Cx \leq d\}
$$

where $A$ is an $r \times n$ matrix of rank $r = n - \dim(\mathcal{P})$ such that $\text{aff}(\mathcal{P}) = \{x : Ax = b\}$, and $C$ is an $s \times n$ matrix such that the rows of $Cx \leq d$ are in one-to-one correspondence with the $s$ facets of $\mathcal{P}$. For a more rigorous treatment of this topic, see e.g. [15].

III. CONSTRUCTION OF VALID INEQUALITIES FROM BUILDING BLOCKS

In this section, we establish constructions of valid inequalities for the polytope $\mathcal{P} = \conv(F_v(C))$, where $\mathcal{C}$ is an “all-ones” SPC code of length $d$ over the finite field $\mathbb{F}_8 = \text{GF}(2^3)$ (generated from the primitive polynomial $1 + D + D^2$), i.e., its parity-check matrix contains only ones. The symbols $\mathcal{P}$, $\mathcal{C}$, and $d$ will be used, with the above meaning, throughout the entire section. Also, $\alpha$ will denote a primitive element of $\mathbb{F}_8$, i.e., $1 + \alpha + \alpha^3 = 0$. In addition, define the following set (of size $7$) of (group) automorphisms of the additive group of $\mathbb{F}_8$: for $0 \neq \gamma \in \mathbb{F}_8$, let $\phi_{\gamma} : \mathbb{F}_8 \rightarrow \mathbb{F}_8$ be defined by $\phi_{\gamma}(\zeta) = \gamma \cdot \zeta$. For notational convenience, denote this set by $\text{Aut}_\alpha(\mathbb{F}_8)$ throughout the paper. It can easily be verified that any $\phi \in \text{Aut}_\alpha(\mathbb{F}_8)$ is an automorphism of $(\mathbb{F}_8, +)$ (the additive group of $\mathbb{F}_8$), i.e., $\phi$ is bijective and satisfies $\phi(\eta + \zeta) = \phi(\eta) + \phi(\zeta)$ for all $\eta, \zeta \in \mathbb{F}_8$ (which implies $\phi(0) = 0$), and that $\text{Aut}_\alpha(\mathbb{F}_8)$ is a subgroup of $\text{Aut}(\mathbb{F}_8, +)$, where $\text{Aut}(\mathbb{F}_8, +)$ denotes the full group of additive automorphisms of $\mathbb{F}_8$. Note, however, that there exist other automorphisms because in the general case an automorphism $\phi \in \text{Aut}(\mathbb{F}_8, +)$ is not uniquely determined by the image $\phi(1)$, and hence is not characterized as the multiplication with a nonzero field element. However, the particular subgroup $\text{Aut}_\alpha(\mathbb{F}_8)$ is sufficient for our purposes.

The constructions are based on classes of length-8 building blocks $T = \{t_k\}_{k \in K}$ that are assembled to form the left-hand side of an inequality according to several rules developed in the following. The individual length-8 building blocks $t_k$ are indexed by the finite set $K$ that may depend on the particular class, and each constructed inequality $\theta^T x \leq \kappa$, with $\theta \in \mathbb{R}^{8d}$ and $\kappa \in \mathbb{R}$, is of the form $\theta^T x = \kappa$ \quad (where each $t_k \in \mathcal{T}$.

In the following, we will identify, with some abuse of notation, such a $\theta$ with the sequence $(k_1, \ldots, k_d)^T$ where $k_i \in K$. Also, using automorphisms from $\text{Aut}(\mathbb{F}_8, +)$, a large set of inequalities can be derived from a single “prototype” one in the following way.

For any permutation $\pi \in S_8$ (where $S_8$ is the group of all permutations of size 8) and vector $a$ of length $8l$ for some $l \geq 1$, we will (with some abuse of notation) denote by $\pi(a)$ the vector obtained by permuting the entries of each 8-block $a_i$ of $a$ according to $\pi$, i.e.,

$$
\pi(a) = (a_{1,\pi(0)}, \ldots, a_{1,\pi(7)}, \ldots, a_{l,\pi(0)}, \ldots, a_{l,\pi(7)})^T
$$

(7)
where we adopt the double-index notation introduced in (2). Assume $\theta^T x \leq \kappa$ for $P = \text{conv}(F, C)$ and induces a face $F$ of dimension $\dim(F)$. Let $\varphi$ be an automorphism of $(F, +)$. It can be shown that

$$\varphi(\theta)^T x \leq \kappa$$

is also valid for $P$ and defines a face $F'$ with $\dim(F') = \dim(F)$. In particular, if $F$ is a facet, then $F'$ is also a facet. Equation (8) allows us to derive $q - 1 = 7$ classes of valid inequalities from a single one by permuting the entries of the building blocks involved. Note that while (8) is true for any automorphism $\varphi$ of $(F, +)$, it will be sufficient for our purposes to consider the subgroup $\text{Aut}_x(F, +)$ only, in the sense that for any automorphism $\varphi \in \text{Aut}(F, +)$, there will exist an automorphism $\bar{\varphi} \in \text{Aut}_x(F, +)$ that gives exactly the same class, i.e., $\{\varphi(\theta) : \varphi \in \text{Aut}(F, +)\} = \{\bar{\varphi}(\theta) : \varphi \in \text{Aut}_x(F, +)\}$.

Now, for any codeword $c = (c_1, \ldots, c_d)^T \in C$, the left-hand side of $\theta^T F_v(c) \leq \kappa$ is

$$\sum_{i=1}^d t_{k_i} f(c_i) = \sum_{i=1}^d t_{k_i, c_i}$$

because $f(c_i)$ is the $c_i$-th unit vector by Definition 2. Hence, for $k \in K$, the entries of $t_k = (t_{k,0}, \ldots, t_{k,7})$ immediately specify the values of the corresponding terms in $\theta^T F_v(c)$.

**Definition 4:** Define the functions $\zeta^\text{hi}, \zeta^\text{lo} : K \to 2^{F_8} \setminus \emptyset$, where $\emptyset$ denotes the empty set and $2^{F_8}$ the power set of $F_8$, i.e., the set of all subsets (including the empty set and $F_8$ itself) of $F_8$, by

$$\zeta^\text{hi}(k) = \left\{ \zeta \in F_8 : t_{k, \zeta} = \max_{n \in F_8} (t_{k, n}) \right\}$$

and

$$\zeta^\text{lo}(k) = \left\{ \zeta \in F_8 : t_{k, \zeta} = \min_{n \in F_8} (t_{k, n}) \right\}$$

for all $k \in K$, and let $k^{\text{hi}} : \zeta^\text{hi}(K) \to K$ and $k^{\text{lo}} : \zeta^\text{lo}(K) \to K$ be the inverses of $\zeta^\text{hi}$ and $\zeta^\text{lo}$, respectively, i.e.,

$$k^{\text{hi}}(A) = k \leftrightarrow A = \zeta^\text{hi}(k)$$

and

$$k^{\text{lo}}(A) = k \leftrightarrow A = \zeta^\text{lo}(k)$$

where $A$ denotes an arbitrary nonempty subset of $F_8$.

We assume in the following that the inverses above are well-defined, or equivalently, that $\zeta^\text{hi}$ and $\zeta^\text{lo}$ are injective. Now, we will present a construction of building blocks from the subclass $\{1, 1, 1\}$ of $\zeta^\text{hi}(K)$ consisting of size 1. Similarly, when $k$ runs over the eight building blocks from $\zeta^\text{lo}(K)$ of size 1, and its image space is equal to $F_8$. Thus, Construction 1 is well-defined and can be used to build inequalities under the mentioned constraint. This will result in a total of $d \cdot 8^d$ inequalities (since there are $d$ possible positions for the nonbold building block) which

<table>
<thead>
<tr>
<th>$k$</th>
<th>$t_k$</th>
<th>$\zeta^\text{hi}(k)$</th>
<th>$\zeta^\text{lo}(k)$</th>
<th>$\sum \zeta^\text{hi}(k)$ and $\sum \zeta^\text{lo}(k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(0, 0, 0, -1, -1)</td>
<td>${0, 1, \alpha + 1}$</td>
<td>${0, 1, \alpha, \alpha + 1, \alpha + 2}$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>(0, 0, 0, 1, 1, 1)</td>
<td>${\alpha^2, 1 + \alpha^2, \alpha + \alpha^2, 1 + \alpha + \alpha^2}$</td>
<td>${0, 1, \alpha + 1}$</td>
<td>0</td>
</tr>
</tbody>
</table>
we denote by the set \( \Theta_2 \). Note that the nonbold (resp. bold) building blocks are indexed using \( K^{nb} = F_8 \) (resp. \( K^b = F_8 \)) in such a way that \( \sum \zeta^{hi}(k) \) (or equivalently, \( \sum \zeta^{lb}(k) \)) gives the nonbold (resp. bold) index from \( K^{nb} \) (resp. \( K^b \)), where \( k \) is the building block index from \( K \). By inspection (see Table II), it can be verified that this gives a well-defined indexing of both nonbold and bold building blocks. The overall set of building blocks \( T_2 \) is indexed by \( K = \{0, \ldots, 15\} \) and such that \( \zeta^{hi}(k) \) gives the bold index from \( K^b = F_8 \) if \( k \) corresponds to a bold building block and \(-1\) otherwise. Similarly, \( \zeta^{nb}(k) \) gives the nonbold index from \( K^{nb} = F_8 \) if \( k \) corresponds to a nonbold building block and \(-1\) otherwise.

**Example 3:** Choose \( d = 3 \) and consider the building block class \( T_2 \). Now, let \( k_1 = 4 \) (corresponding to the bold building block \((0, 1, 1, 0, 2, 1, 1, 0)) \) and \( k_2 = 7 \) (corresponding to the nonbold building block \((0, 1, 1, 0, 0, 1, 1, 2)) \). We have \( \zeta^{hi}(k_1) = \alpha^2 \), \( \zeta^{hi}(k_2) = 1 + \alpha + \alpha^2 \), and \( k_3 = 11 \) (corresponding to the nonbold building block \((0, -1, -1, 2, 0, -1, 0, -1)) \), since \( \zeta^{lb}(k_3) = 1 + \alpha \) and \( \zeta^{hi}(k_1) + \zeta^{hi}(k_2) + \zeta^{lb}(k_3) = 0 \).

We can prove the following proposition.

**Proposition 1:** Assume \( d \geq 2 \). An inequality \( \theta^T x \leq \kappa \) with \( \theta = (k_1, \ldots, k_d)^T \) is in \( \Theta_2 \) if and only if

\[
\begin{align*}
\sum_{k \in F_8} k \left( |\theta_k^b| + |\theta_k^{nb}| \right) &= 0, \quad (9a) \\
\sum_{k \in F_8} |\theta_k^{nb}| &= 1, \quad (9b)
\end{align*}
\]

and

\[
\kappa = \sum_{k \in F_8, \theta_k^{nb} \neq 0} |\theta_k^b| + \sum_{k \in \{1 + \alpha, 1 + \alpha^2, 1 + \alpha + \alpha^2\}} |\theta_k^b| + \\
+ \sum_{k \in \{0, 1, \alpha, 1 + \alpha, 1 + \alpha^2, 1 + \alpha + \alpha^2\}} |\theta_k^{nb}| + |\theta_k^{nb}^{0.1}| - 2 \quad (9c)
\]

where, for \( k \in F_8 \), \( V_k^{\theta^b} = \{i \in [d] : \xi^b(k_i) = k\} \) denotes the set of coordinates of \( \theta \) corresponding to bold building blocks with bold index \( k \). Similarly, \( V_k^{\theta^{nb}} = \{i \in [d] : \xi^{nb}(k_i) = k\} \) denotes the set of coordinates of \( \theta \) corresponding to nonbold building blocks with nonbold index \( k \).

Similarly to \( T_2 \), partition the class \( T_3 \) into two disjoint subclasses \( T_3^b \) and \( T_3^{nb} \) containing the bold (resp. nonbold) building blocks from \( T_3 \). Then, use Construction 1 repeatedly under the constraint of picking exactly one building block from the subclass \( T_3^{nb} \) and exactly \( d - 2 \) building blocks from the subclass \( T_3^b \), among \( d - 1 \) entries of \( \theta \). The final entry, say \( k_d \), is chosen to minimize \( t_{k_d, k} \) where \( k_d \) runs over all building blocks from the subclass \( T_3^{nb} \), but there is a catch. It can be seen from Table III that when \( k \) runs over building blocks from \( T_3^{b} \), \( \zeta^{lb}(k) \) is of size 1. However, its size is strictly larger than 1 when \( k \) runs over building blocks from \( T_3^{nb} \). Thus, when constructing the canonical codeword as part of Construction 1, we construct a set of codewords. Then, \( k_d \) is chosen (where \( k_d \) runs over the building blocks in the subclass \( T_3^{nb} \)) such that \( \zeta^{lb}(k_d) \) (which is a set) matches perfectly (see Definition 1) with the sum \( \sum_{i=1}^{d-1} \zeta^{hi}(k_i) \) (in \( F_8 \)) (which is also a set; one of the entries is a set, the remaining ones are single field elements, so the summation is well-defined according to (1)). One can show by inspection (see Table III) that with the building blocks that we have in \( T_3 \) there is always a unique \( k_d \) (corresponding to a building block from \( T_3^{nb} \)) with this property. In summary, this will result in a total of \( \binom{d}{2} \cdot 8^{d-2} = \binom{d}{2} \cdot 8^{d-1} \) inequalities, which we denote by the set \( \Theta_3 \), since any 2 positions among the \( d \) possible positions can be chosen for the nonbold building blocks.

We adopt the following indexing of the building blocks of \( T_3 \). The nonbold building blocks are indexed (using a nonbold index) with the elements of \( F_8 \) (\( K^{nb} = F_8 \)) in the order as they appear in Table III. The bold building blocks, however, are indexed, now using a bold index (as for the second class) in such a way that \( \sum \zeta^{hi}(k) \) (or equivalently, \( \sum \zeta^{lb}(k) \)) gives the bold index, where \( k \) is the index of a bold building block (see Table III). Thus, \( K^b = F_8 \) as
for the second building block class. The overall set of building blocks $T_3$ is indexed by $K = \{0, \ldots, 15\}$ and such that $\xi^b(k)$ gives the bold index from the corresponding bold index set $K^b = \mathbb{F}_8$ if $k$ corresponds to a bold building block and $-1$ otherwise. Similarly, $\xi^{nb}(k)$ gives the nonbold index from the corresponding nonbold index set $K^{nb} = \mathbb{F}_8$ if $k$ corresponds to a nonbold building block and $-1$ otherwise.

Example 4: Choose $d = 3$ and consider the building block class $T_3$. Now, let $k_1 = 11$ corresponding to the nonbold building block $(0, -1, -1, -1, 0, 0, 0, 0)$ and $k_2 = 7$ (corresponding to the bold building block $(0, 0, 0, 0, 0, 0, 0, 0)$). We have $\xi^b(k_1) = 0, 1 + \alpha^2, \alpha + \alpha^2, 1 + \alpha + \alpha^2$, $\xi^{nb}(k_2) = 1 + \alpha + \alpha^2$, and $k_3 = 8$ (corresponding to the nonbold building block $(0, 0, 0, 0, 0, 0, 0, 1)$). Since $\xi^{o}(k_2) = 0, 1, \alpha, 1 + \alpha + \alpha^2$ and $\xi^{nh}(k_2) = 1 + \alpha + \alpha^2, \alpha, 1 + \alpha + \alpha^2$ matches perfectly (see Definition 1) with $\xi^{o}(k_3)$ $(1 + \alpha + \alpha^2 + 1 + \alpha + \alpha^2 = \alpha + \alpha = 1 + 1 = 0 + 0 = 0)$. Note that there is no other building block with this property in $T_3$.

We can prove the following proposition.

Proposition 2: Assume $d \geq 2$. An inequality $\theta^T x \leq \kappa$ with $\theta = (k_1, \ldots, k_d)^T$ is in $T_3$ if and only if

$$\sum_{k \in \mathbb{F}_8} \left| V_{k}^b \right| + \left| V_{k}^{nb} \right| = \alpha^2,$$

$$\sum_{k \in \mathbb{F}_8} \left| V_{k}^{nh} \right| = 2,$$

and $\kappa = \sum_{k \in \mathbb{F}_8 \setminus \{0\}} \left| V_{k}^b \right| + \sum_{k \in \{0, 1, 1 + \alpha, 1 + \alpha^2\}} \left| V_{k}^{nh} \right| - 1$

where, for $k \in \mathbb{F}_8$, $V_{k}^b = \{i \in [d] : \xi^b(k_i) = k\}$ denotes the set of coordinates of $\theta$ corresponding to bold building blocks with bold index $k$. Similarly, $V_{k}^{nh} = \{i \in [d] : \xi^{nh}(k_i) = k\}$ denotes the set of coordinates of $\theta$ corresponding to nonbold building blocks with nonbold index $k$.

Proposition 3: Let $C$ be a length-$d$ “all-ones” SPC code over $GF(8)$ and $P = \text{conv}(F_8(C))$. For $d \geq 3$,

1. $\dim(P) = 7d$,
2. the affine hull of $P$ is $\text{aff}(P) = \{x : (4) \text{ holds for } i \in [d]\}$, and
3. (3) defines a facet of $P$ for $i \in [d]$ and $j \in \mathbb{F}_8$.

In fact, Proposition 3 can be shown to hold (with $\dim(P) = (2^m - 1)d$) for any SPC code over $GF(2^m)$ when $d \geq 3$ and $m \geq 2$ (if $m = 1$, for $d \geq 4$), but in this work the results for $GF(8)$ are the ones that are needed. The results for $m = 1$ are already known; see e.g. [16, Thm. III.2].

**Lemma 1:** The inequalities from $\Theta_1 \cup \Theta_2 \cup \Theta_3$ are all valid for $P = \text{conv}(F_8(C))$, where $C$ is an “all-ones” SPC code over $GF(8)$ of length $d \geq 2$.

**Theorem 1:** Every inequality $\theta^T x \leq \kappa$ in $\Theta_1 \cup \Theta_2 \cup \Theta_3$ defines a facet of $P = \text{conv}(F_8(C))$, where $C$ is an “all-ones” SPC code over $GF(8)$ of length $d \geq 2$.

For the set $\Theta_i, i \in [3]$, of inequalities and an automorphism $\varphi$ of $(\mathbb{F}_8, +)$, we will denote by $\varphi(\Theta_i)$ the set of all inequalities derived by (8) from those in $\Theta_i$. We make the following conjecture.

**Conjecture 1:** Let $C$ be the “all-ones” SPC code over $GF(8)$ of length $d \geq 3$ and $P = \text{conv}(F_8(C))$. For notational convenience, let $\Phi = \text{Aut}_x(F_8, +)$. Then,

$$\Theta = (\cup_{i=1}^{3} (\cup_{\varphi \in \Phi} \varphi(\Theta_i))) \cup \Delta_8^d = (\cup_{i=1}^{3} \Phi(\Theta_i)) \cup \Delta_8^d$$

with $|\Theta| = 7 \cdot \left( (\binom{8}{2} + d) \cdot 8^{d-1} + 7 \cdot 2^{d-1} + 9d \right)$, and where we have implicitly defined $\Phi(\Theta_i) = \cup_{\varphi \in \Phi} \varphi(\Theta_i)$, $i \in [3]$, gives a complete and irredundant description of $P$.

We have verified numerically using the software package **Polymake** [13] that the conjecture is true for $d = 3$ and 4. Note that since $\cup_{\varphi \in \text{Aut}_x(F_8, +)} \varphi(\Theta_i) = \cup_{\varphi \in \text{Aut}_x(F_8, +)} \varphi(\Theta_i)$, for all $i \in [3]$, it is sufficient to consider the subgroup $\text{Aut}_x(F_8, +)$. When we talk about exact ALP decoding in the following, it is under the assumption that the conjecture above is true.
IV. Adaptive Linear Programming Decoding

In this section, we show how to overcome the exponential number of facets of an embedded SPC code over GF(8) by giving an efficient separation algorithm for the convex hull of its constant-weight embedding which allows for efficient ALP decoding of general codes over GF(8). It thus generalizes the well-known Adaptive LP Decoder for binary codes [3].

The main loop of our ALP decoder (Algorithm 1) is similar to [3, Alg. 2], except that the generalized box constraints from $\Delta_8^C$, where $C$ is a general code over GF(8), are present from start; due to their small number it does not pay off to separate them adaptively.

Algorithm 1 ALP Decoder for Codes Over GF(8)

**Input:** Code $C$ of length $n$, channel output $\Lambda_v(y)$

**Output:** Solution $x$ of (6)

1. Initialize a linear program with variables $x \in \mathbb{R}^{8n}$, constraints from $\Delta_8^C$, and objective function $\min \Lambda_v(y)^T x$
2. while True do
3.  $x^{LP} \leftarrow$ optimal LP solution
4.  for all $j \in J$ do
5.   for all $\varphi \in \text{Aut}_x(F_8, +)$ do
6.      $\theta \leftarrow \text{SEPARATE}\varphi(\Theta_1)(C_j, x^{LP})$
7.      if $\theta \neq \text{Null}$ then
8.         add ineq. from $\varphi(\Theta_1)$ defined by $\theta$
9.      if no cut was added then
10.     for all $\varphi \in \text{Aut}_x(F_8, +)$ do
11.       $\theta \leftarrow \text{SEPARATE}\varphi(\Theta_2)(C_j, x^{LP})$
12.       if $\theta \neq \text{Null}$ then
13.          add ineq. from $\varphi(\Theta_2)$ defined by $\theta$
14.     if no cut was added then
15.       for all $\varphi \in \text{Aut}_x(F_8, +)$ do
16.          $\theta \leftarrow \text{SEPARATE}\varphi(\Theta_3)(C_j, x^{LP})$
17.          if $\theta \neq \text{Null}$ then
18.             add ineq. from $\varphi(\Theta_3)$ defined by $\theta$
19.     if no cut was added then
20.        return $x^{LP}$

The main issue that needs to be addressed is that of efficient separation of the sets of inequalities $\varphi(\Theta_1)$, $\varphi(\Theta_2)$, and $\varphi(\Theta_3)$ (for some fixed automorphism $\varphi \in \text{Aut}_x(F_8, +)$), i.e., to develop an efficient search algorithm for cuts, i.e., inequalities (from either $\varphi(\Theta_1)$, $\varphi(\Theta_2)$, or $\varphi(\Theta_3)$) violated by the current solution $x^{LP}$. This issue is addressed by the subprocedure $\text{SEPARATE}\varphi(\Theta_i)$ ($i \in [3]$ and $\varphi \in \text{Aut}_x(F_8, +)$) called in Lines 6, 11, and 16, for $i = 1, 2, 3$, respectively. Below, we first describe $\text{SEPARATE}\varphi(\Theta_1)$ (for any $i \in [3]$) when $\varphi \in \text{Aut}_x(F_8, +)$ is the identity mapping and $C_j$ is assumed to be an “all-ones” SPC code. Then, we will consider a general automorphism $\varphi$ from $\text{Aut}_x(F_8, +)$ in Section IV-D, and finally, general SPC codes in Section IV-E.

A. Separation for $\Theta_1$

The class $T_j$ in Table I gives Feldman-type (or forbidden set) inequalities and these are exactly those of [11, Sec. V] or [10, Def. 38] (the relaxed polytope $\mathcal{U}'$) when considering all (additive) automorphisms in $\text{Aut}_x(F_8, +)$ according to (8). Thus, the inequalities in $\Theta_1$ can be separated in the same manner as for binary SPC codes. For details, see [5, Alg. 2].

B. Separation for $\Theta_2$

Consider an inequality in $\Theta_2$ specified by $\theta = (k_1, \ldots, k_d)^T$ and $\kappa$ as in (9a)–(9b) and (9c), respectively. For $k \in F_8$, let (as before) $V_k^{\theta, b} = \{i \in [d]: \xi^b(k_i) = k\}$ and $V_k^{\theta, nb} = \{i \in [d]: \xi^{nb}(k_i) = k\}$. Then, the facet-defining inequality $\theta^T x \leq \kappa$ (where $x = x(j)$ for some $j \in J$ with $d = |N_c(j)|$) can be rewritten as

$$\Psi(\theta, x) = \sum_{k \in F_8} \sum_{i \in V_k^{\theta, b}} v^{k, b}(x_i) + \sum_{k \in F_8} \sum_{i \in V_k^{\theta, nb}} v^{k, nb}(x_i) \geq 2$$

where $x = (x_1, \ldots, x_d)^T$, $x_i = (x_{i,0}, \ldots, x_{i,7})$ for all $i \in [d]$, $v^{k, b}(x_i) = \left\{\begin{array}{ll} -t_k^b x_i^T & \text{if } k = 0, \\
 2 - t_k^b x_i^T & \text{if } k = 1 + \alpha, \alpha^2, 1 + \alpha + \alpha^2, \\
 1 - t_k^b x_i^T & \text{otherwise}
\end{array}\right.$

and

$$v^{k, nb}(x_i) = \left\{\begin{array}{ll} -t_k^{nb} x_i^T & \text{if } k = 1 + \alpha, \alpha^2, 1 + \alpha + \alpha^2, \\
 2 - t_k^{nb} x_i^T & \text{if } k = 0, \\
 1 - t_k^{nb} x_i^T & \text{otherwise}
\end{array}\right.$$
It follows that \( \psi^* = \psi(x, d, 1, 0) \) and it can be computed using the following recursion

\[
\psi(x, s, 1, \zeta) = \min_{\beta \in \mathbb{F}_8} \left\{ v^{\beta, b}(x_s) + \psi(x, s - 1, 1, \zeta - \beta), \right. \\
\left. v^{\beta, nb}(x_s) + \psi(x, s - 1, 0, \zeta - \beta) \right\}
\]

(13)

and

\[
\psi(x, s, 0, \zeta) = \min_{\beta \in \mathbb{F}_8} \left\{ v^{\beta, b}(x_s) + \psi(x, s - 1, 0, \zeta - \beta) \right\}
\]

(14)

for \( s \geq 2 \) and \( \zeta \in \mathbb{F}_8 \). The recursion allows to compute a \( d \times 2 \times 8 \) array \( T \) with entries \( T[s, t, \zeta] = \psi(x, s, t, \zeta) \); initialize \( T \) with

\[
T[1, t, \zeta] = \psi(x, 1, t, \zeta) = \begin{cases} \psi^{c, b}(x_1) & \text{if } t = 0, \\
\psi^{c, nb}(x_1) & \text{otherwise} \end{cases}
\]

for \( t \in \{0, 1\} \) and \( \zeta \in \mathbb{F}_8 \). Then, use (13) and (14) to proceed from top to bottom until reaching row \( d \), where only the entry \( T[d, 1, 0] \) is needed. Because the expressions in (13) and (14) both can be calculated in time \( O(8 \cdot 3) \), the overall time needed to obtain \( \psi^* \) is \( O(d \cdot 8^2 \cdot 3) \) (the field size is fixed to eight).

Note that the actual minimizing solution \( \theta^* \) of (12) can be obtained within the same asymptotic running time by storing the minimizing \( \beta \)'s from (13) and (14) in a second \( d \times 2 \times 8 \) array \( S \). The complete algorithm is outlined in Algorithm 2.

Note that in Line 35, we regard with some abuse of notation \( k[d] \mod 8 \) as a field element, i.e., it is regarded as the field element whose integer representation is \( k[d] \mod 8 \).

Remark 2: Algorithm 2 can be tweaked in several ways:

- At row \( d \) of \( T \), only the single value \( T[d, 1, 0] \) has to be computed.
- If \( d \) is large, one could first minimize \( \Psi(\theta, x) \) without the constraints (12c) and (12d) (which is possible in time \( O(d \cdot 8) \)). If the result satisfies (12c) and (12d) (optimum found) or fulfills \( \psi^* \geq 2 \) (no cut can be included), we are done.
- Because \( v^{k, b}(x_i) \) and \( v^{k, nb}(x_i) \) are both nonnegative for all \( k \in \mathbb{F}_8 \) and \( i \in [d] \), \( \psi^* \geq \min_{\zeta \in \mathbb{F}_8} \psi(x, i, t, \zeta) \) holds for any \( i \in [d] \). Hence, the search can be stopped as soon as all entries in a single row of \( T \) are at least 2.

C. Separation for \( \Theta_3 \)

Due to the similarity of Propositions 1 and 2, it follows that the inequalities in \( \Theta_3 \) can be separated in more or less the same way as the inequalities in \( \Theta_2 \). In particular, a function \( \Psi(\theta, x) \) is defined as in (11). However, the right-hand side of (11) becomes 1 instead of 2, and

\[
v^{k, b}(x_i) = \begin{cases} -t^{k, b}_i x_i^T & \text{if } k = 0, \\
1 - t^{k, b}_i x_i^T & \text{otherwise} \end{cases}
\]

and

\[
v^{k, nb}(x_i) = \begin{cases} -t^{k, nb}_i x_i^T & \text{if } k = 1 + \alpha, \alpha^2, 1 + \alpha^2, \alpha + \alpha^2, \\
1 - t^{k, nb}_i x_i^T & \text{otherwise} \end{cases}
\]

Algorithm 2 \textsc{Separate}\( \Theta_2 \)

Input: “All-ones” SPC code \( C \) over GF(8) of length \( d = |N_c(j)| \) and LP solution \( x_{\text{LP}} \in S^d_+ \)

Output: Solution \( \theta^* \) of (12), if \( \psi^* < 2 \); Null otherwise

1: Let \( T, v, \) and \( S \) be \( d \times 2 \times 8 \) arrays, and let \( k \) be a length-\( d \) array
2: \( x = (x_1, \ldots, x_d) \leftarrow P_j x_{\text{LP}} \)
3: for \( \zeta \in \mathbb{F}_8 \) do
4: for \( t \in \{0, 1\} \) do
5: if \( t = 0 \) then
6: for \( i \in [d] \) do
7: \( v[i, 0, \zeta] \leftarrow v^{c, b}(x_i) \) \( \triangleright \) initialize \( v \)
8: else
9: for \( i \in [d] \) do
10: \( v[i, 1, \zeta] \leftarrow v^{c, nb}(x_i) \) \( \triangleright \) initialize \( v \)
11: \( T[1, t, \zeta] \leftarrow v[1, t, \zeta] \) \( \triangleright \) initialize \( T[1, : , : ] \)
12: \( S[1, t, \zeta] \leftarrow (2 + t) \cdot 8 \) \( \triangleright \) \( (2) \) is int. rep. of \( \zeta \)
13: for \( i = 2, \ldots, d \) do
14: for \( \zeta \in \mathbb{F}_8 \) do
15: \( S[i, 0, \zeta] \leftarrow -1, S[i, 1, \zeta] \leftarrow -1 \)
16: \( T[i, 0, \zeta] \leftarrow \infty, T[i, 1, \zeta] \leftarrow \infty \)
17: for \( \beta \in \mathbb{F}_8 \) do
18: \( \text{val} \leftarrow v[i, 0, \beta] + T[i - 1, 1, \zeta - \beta] \)
19: if \( \text{val} < T[i, 1, \zeta] \) then
20: \( T[i, 1, \zeta] \leftarrow \text{val} \)
21: \( S[i, 1, \zeta] \leftarrow \beta(2) \) \( \triangleright \) \( \beta(2) \) is int. rep. of \( \beta \)
22: \( \text{val} \leftarrow v[i, 1, \beta] + T[i - 1, 0, \zeta - \beta] \)
23: if \( \text{val} < T[i, 1, \zeta] \) then
24: \( T[i, 1, \zeta] \leftarrow \text{val} \)
25: \( S[i, 1, \zeta] \leftarrow (2 + \beta) + 8 \) \( \triangleright \) See Line 21
26: \( \text{val} \leftarrow v[i, 0, \beta] + T[i - 1, 0, \zeta - \beta] \)
27: if \( \text{val} < T[i, 0, \zeta] \) then
28: \( T[i, 0, \zeta] \leftarrow \text{val} \)
29: \( S[i, 0, \zeta] \leftarrow \beta(2) \) \( \triangleright \) See Line 21
30: \( \text{tmp} \leftarrow 1 \)
31: if \( T[d, \text{tmp}, 0] < 2 \) then
32: \( k[d] \leftarrow S[d, \text{tmp}, 0] \)
33: if \( k[d] \geq 8 \) then
34: \( \text{tmp} \leftarrow 0 \)
35: \( \text{next} \leftarrow (0 - (k[d] \mod 8)) \) \( \triangleright \) \( k[d] \mod 8 \) in \( \mathbb{F}_8 \)
36: for \( i = d - 1, \ldots, 1 \) do
37: \( k[i] \leftarrow S[i, \text{tmp}, \text{next}] \)
38: if \( k[i] \geq 8 \) then
39: \( \text{tmp} \leftarrow 0 \)
40: \( \text{next} \leftarrow (\text{next} - (k[i] \mod 8)) \) \( \triangleright \) See Line 35
41: return \( k \)
42: return Null
In particular, in the separation algorithm, the vector $x$ case of having of the SPC code. (3)

\[ \Theta \phi \]

for the class $D$. Separation for the Class $\Theta$. The error-rate performance of the

\[ \mathbf{H} \]

parity-check matrix is replaced by $1$, $\text{erfc}(255 \cdot N)$, $1$.

\[ \mathbf{H} \]

second, third, fourth, and fifth nonzero entry in each row of the parity-check matrix is replaced by $1$, $\alpha^2$, $\alpha^4$, $\alpha^6$, and $1$, respectively. The error-rate performance of the $\Theta_{\phi}$ code over $GF(8)$ (with the same modified parity-check matrix) was also studied in [9] under sum-product, min-sum, and low-complexity LP decoding. In Fig. 1, the frame error-rate (FER) performance over an additive white Gaussian noise channel using 8-phase-shift keying modulation versus the signal-to-noise ratio (SNR) $E_b/N_0$ is depicted for both codes when using inequalities from the sets $\Phi(\Theta_1)$ and $\Phi(\Theta_1) \cup \Phi(\Theta_2)$ separately. The symbol $\zeta \in F_8$ is mapped to a constellation point according to $\zeta \mapsto \exp(i(2\mu(\zeta) + 1)/\pi)$, where $\mu(\zeta) = \log_\alpha(\zeta) + 1$ (log$_\alpha$·) denotes discrete logarithm with respect to the primitive element $\alpha$ of $GF(8)$ when $\zeta$ is nonzero and $-1$ for $\zeta = 0$ (the same modulation method as in [9]). The decoder is implemented as described in Algorithm 1 using the dual simplex method of the LP solver GLPK [18]. As we can observe from the figure, there is almost no difference in FER performance when using the

\[ \mathbf{H} \]

An algorithm similar to Algorithm 2 can be used to separate the inequalities in $\Theta_{\phi}$ with running time $O(d \cdot 8^2 \cdot 5)$. Details are omitted for brevity.

D. Separation for the Class $\varphi(\Theta_i), i \in [3], \varphi \in \text{Aut}_x(F_8, +)$

There is no need to make an explicit separation algorithm for the class $\varphi(\Theta_i), i \in [3]$ and $\varphi \in \text{Aut}_x(F_8, +)$ (assuming $\varphi$ is not the identity mapping), since the separation algorithm for $\Theta_i$ can be applied to the parity-check matrix $\varphi(H) = (\varphi(h_{1,1}), \ldots, \varphi(h_{1,5}))$ instead of $H$ as parity-check matrix of the SPC code.

E. Separation for General Single Parity-Check Codes

So far, we have only considered “all-ones” SPC codes. The case of having $h_{1,i} > 1$ is handled using automorphisms. In particular, in the separation algorithm, the vector $x_i = (x_i,0, \ldots, x_i,7)$ is permuted according to the specific automorphism $\varphi$ from $\text{Aut}_x(F_8, +)$ which permutes the (multiplicative) identity $1$ to $h_{1,i}$, i.e., $\varphi$ is chosen from $\text{Aut}_x(F_8, +)$ such that $\varphi(1) = h_{1,i}$, or equivalently, $\varphi = \varphi_{h_{1,i}}$, since $\varphi(1) = \varphi_{h_{1,i}}(1) = h_{1,i} \cdot 1 = h_{1,1}$. The components of $x_i$ are permuted accordingly using $\varphi$ (see the example below).

Example 5: If $h_{1,i} = \alpha$, for some $i \in [d]$, then $\varphi = \varphi_\alpha$ since $\varphi(1) = \varphi_\alpha(1) = \alpha = 1 = \alpha = h_{1,1}$. In particular, $\varphi_\alpha = (0)(1, \alpha, \alpha^2, 1+\alpha, \alpha+\alpha^2, 1+\alpha+\alpha^2, 1+\alpha^2)$, which means that $x_i = (x_i,0, x_i,1, x_i,2, x_i,3, x_i,4, x_i,5, x_i,6, x_i,7)$ is permuted to $x_i = (x_i,0, x_i,5, x_i,1, x_i,2, x_i,4, x_i,3, x_i,6, x_i,7)$, or equivalently, $x_i$ is permuted to $\varphi_\alpha^{-1}(x_i)$ (see (7)).

Actually, both an automorphism $\varphi \in \text{Aut}_x(F_8, +)$ and $h_{1,i} > 1$ can be handled together by setting $x_i \leftarrow (\varphi \circ \varphi_{h_{1,i}})^{-1}(P_j x_{LP})$, where $\circ$ denotes function composition and $(P_j x_{LP})_i$ denotes the $i$-th block of $P_j x_{LP}$, i.e., appropriately rotating $x_i$, and then using the algorithm for the identity mapping and $h_{1,1} = 1$.

V. NUMERICAL RESULTS

In this section, we provide some simulation results for the proposed ALP decoding algorithm for two examples codes; the (155, 64) and the (755, 334) Tanner code over $GF(8)$, denoted by $C^{(1)}$ and $C^{(2)}$, respectively, from [17], in which the first, second, third, fourth, and fifth nonzero entry in each row of the parity-check matrix is replaced by $1$, $\alpha^2$, $\alpha^4$, $\alpha^6$, and $1$, respectively. The error-rate performance of the (755, 334) Tanner code over $GF(8)$ (with the same modified parity-check matrix) is shown in Fig. 1.

![Fig. 1. FER performance of the (155, 64) and the (755, 334) Tanner code over $GF(8)$, denoted by $C^{(1)}$ and $C^{(2)}$, respectively, as a function of $E_b/N_0$.](image-url)

<table>
<thead>
<tr>
<th>$E_b/N_0$ (dB)</th>
<th>7.5</th>
<th>8</th>
<th>8.5</th>
<th>9</th>
<th>9.5</th>
<th>10</th>
<th>10.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALP simplex</td>
<td>13780</td>
<td>10510</td>
<td>6460</td>
<td>3980</td>
<td>2930</td>
<td>2370</td>
<td>2160</td>
</tr>
<tr>
<td>LP probs solved</td>
<td>7.1</td>
<td>5.7</td>
<td>4.1</td>
<td>3.0</td>
<td>2.6</td>
<td>2.3</td>
<td>2.2</td>
</tr>
<tr>
<td>$\Phi(\Theta_1)$-cuts added</td>
<td>843</td>
<td>797</td>
<td>694</td>
<td>564</td>
<td>456</td>
<td>382</td>
<td>354</td>
</tr>
<tr>
<td>$\Phi(\Theta_2)$-cuts added</td>
<td>18.0</td>
<td>11.9</td>
<td>4.7</td>
<td>0.9</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$\Phi(\Theta_3)$-cuts added</td>
<td>0.02</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E_b/N_0$ (dB)</th>
<th>7.5</th>
<th>8</th>
<th>8.25</th>
<th>8.5</th>
<th>8.625</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALP simplex</td>
<td>157500</td>
<td>74200</td>
<td>47800</td>
<td>39800</td>
<td>38200</td>
</tr>
<tr>
<td>LP probs solved</td>
<td>8.7</td>
<td>4.8</td>
<td>3.6</td>
<td>3.4</td>
<td>3.4</td>
</tr>
<tr>
<td>$\Phi(\Theta_1)$-cuts added</td>
<td>4240</td>
<td>2040</td>
<td>3570</td>
<td>3120</td>
<td>2890</td>
</tr>
<tr>
<td>$\Phi(\Theta_2)$-cuts added</td>
<td>96</td>
<td>31</td>
<td>7.7</td>
<td>1.2</td>
<td>0.5</td>
</tr>
<tr>
<td>$\Phi(\Theta_3)$-cuts added</td>
<td>0.1</td>
<td>0.03</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
inequalities in the set $\Phi(\Theta_1)$ as compared to using the full set $\Phi(\Theta_1) \cup \Phi(\Theta_2) \cup \Phi(\Theta_3)$ of inequalities. For both codes, we also recorded the average number of simplex iterations per decoding instance, the average number of LP problems solved until optimality, and the average number of cuts added from the different sets $\Phi(\Theta_1)$ through $\Phi(\Theta_3)$, for different values of the SNR. The results are presented in Tables IV and V for the codes $C^{(1)}$ and $C^{(2)}$, respectively.

Note that implementing exact LP decoding with both the plain and cascaded static approaches from [7] is not practical unless the codes are small, since the number of LP constraints and variables becomes very large. In the comparisons below, when counting constraints, “$x \geq 0$”-constraints ($x$ denotes a generic LP variable) are not included, since variable bounds are handled implicitly by all serious LP solvers and the extra effort is only a small constant not affecting asymptotic running time. Thus, when talking about LP constraints we refer to all constraints except “$x \geq 0$”-constraints. For a right-regular code with $d$ ones in each row, the number of LP constraints with the plain approach is $r(d(q-1)+1)$ [7, Eqs. (9)–(10)] ($r$ is the number of parity-check equations), which is equal to 3348 for the code $C^{(1)}$. Although this number is much lower than the total number of LP constraints from $\Theta$, which is 40008011 for $C^{(1)}$ (computed from (10)), it is much higher than what is observed with ALP decoding for this code (see Table IV). For a fair comparison though, $n = 155$ of the simplex constraints (according to (4)) are also needed with ALP decoding. For the cascaded approach, the number of LP constraints is upper-bounded by $r(d-2)(3q-2)$, $d \geq 3$ [7, Sec. IX], with equality if there are no 4-cycles in the corresponding Tanner graph. For the code $C^{(1)}$ (which do not contain any 4-cycles in its Tanner graph), this formula evaluates to 6138, which is higher than for the plain approach.

However, the number of variables is much lower. The number of variables with the plain static approach is $n(q-1) + q^d - 1$, while it is upper-bounded by $(n+r(d-3))(q-1)+r(d-2)q^2$, $d \geq 3$ for the cascaded approach [7, Sec. IX] (again with equality if there are no 4-cycles in the corresponding Tanner graph). For the code $C^{(1)}$, these formulas evaluate to 382013 and 20243, respectively, while the number of variables with ALP decoding is only $n(q-1) = 1085$. Note that the reason why it is not valid is that all building blocks start with a zero and that the objective function is independent of $x_i, i \in [n]$. Thus, the number of variables can be reduced by $n$. In summary, the number of LP constraints and variables is much lower for ALP decoding than with a static approach, which translates directly into a lower decoding complexity.

Finally, we remark that almost no difference in error-rate performance when using the inequalities in the set $\Phi(\Theta_1)$ as compared to using the full set $\Phi(\Theta_1) \cup \Phi(\Theta_2) \cup \Phi(\Theta_3)$ of inequalities has been observed for other low-density and high-density codes as well, e.g., for Reed-Muller codes over GF(8), which from a decoding complexity point of view is very interesting. This can only mean that the facets induced by the inequality sets $\Phi(\Theta_2)$ and $\Phi(\Theta_3)$ somehow cut off only very “small” parts of the decoding polytope.

VI. CONCLUSION

We have given explicit constructions of valid facet-defining inequalities (using no auxiliary variables) for SPC codes over GF(8) and conjectured based on numerical observations that these sets of inequalities together with the simplex constraints give a complete and irredundant description of the convex hull of the constant-weight embedding of the given code. Based on the explicit form of the inequalities, an efficient separation algorithm for ALP decoding of general codes over GF(8) was proposed using the principle of dynamic programming. The proposed approach is much more efficient than the plain or cascaded static approaches. Finally, we presented numerical results showing that typically only a small set of inequalities is necessary for achieving close to exact ALP decoding.

REFERENCES

[7] M. F. Flanagan, V. Skachek, E. Byrne, and M. Grefenath, “Linear-
programming decoding of nonbinary linear codes,” IEEE Trans. Inf. The-
[16] M. Helmling, S. Ruzika, and A. Tatamis, “Mathematical program-