Exact separation of forbidden-set cuts associated with redundant parity checks of binary linear codes

Christian Puchert and Andreas M. Tillmann

***Abstract*—In recent years, several integer programming (IP) approaches were developed for maximum-likelihood decoding and minimum distance computation for binary linear codes. Two aspects in particular have been demonstrated to improve the performance of IP solvers as well as adaptive linear programming decoders: the dynamic generation of forbidden-set (FS) inequalities, a family of valid cutting planes, and the utilization of so-called redundant parity-checks (RPCs). However, to date, it had remained unclear how to solve the exact RPC separation problem (i.e., to determine whether or not there exists any violated FS inequality w.r.t. any known or unknown parity-check). In this note, we prove NP-hardness of this problem. Moreover, we formulate an IP model that combines the search for most violated FS cuts with the generation of RPCs, and report on computational experiments. Empirically, for various instances of the minimum distance problem, it turns out that while utilizing the exact separation IP does not appear to provide a computational advantage, it can apparently be avoided altogether by combining heuristics to generate RPC-based cuts.**

***Index Terms*—computational complexity, linear codes, binary codes, integer linear programming, Hamming weight**

I. Introduction & Preliminaries

The well-known minimum distance computation problem for binary linear codes, defined by a parity-check matrix $\boldsymbol{H} \in \{0,1\}^{m\times n}$, can be stated as

$$\begin{aligned} \min \quad & \|\boldsymbol{x}\|_0 \\ \text{s.t.} \quad & \boldsymbol{H}\boldsymbol{x} = \boldsymbol{0} \mod 2,\ \boldsymbol{x} \neq \boldsymbol{0},\ \boldsymbol{x} \in \{0,1\}^n, \end{aligned} \tag{1}$$

where $\|\cdot\|_0$ denotes the Hamming weight (i.e., number of nonzeros). Similarly, the maximum-likelihood decoding (MLD) problem is obtained by replacing the objective function with $\boldsymbol{\gamma}^\top \boldsymbol{x}$, where $\boldsymbol{\gamma}$ is a (known) vector of negative log-likelihoods (see, e.g., [1]). Both problem (1) and MLD are NP-hard, see [2] and [3], respectively. For simplicity, we will focus on (1) throughout, but emphasize that as we will mainly deal with issues pertaining to the feasible set, the results are also applicable in the MLD context.

Manuscript received April 7, 2020.

This work was conducted in the RWTH ERS Start-Up project "Efficient exact maximum-likelihood decoding and minimum-distance computation for binary linear codes" (Tillmann), funded by the Excellence Initiative of the German federal and state governments.

C. Puchert was with the Chair of Operations Research at RWTH Aachen University, Kackertstr. 7, 52072 Aachen, Germany (e-mail: puchert@or.rwth-aachen.de), and is now with Ab Ovo Business and Software Solutions, Düsseldorf, Germany.

A. M. Tillmann was with the Chair of Operations Research and the Visual Computing Institute at RWTH Aachen University, Germany, and is now with the Institute for Mathematical Optimization, Technische Universität Braunschweig, Universitätsplatz 2, 38106 Braunschweig, Germany (e-mail: a.tillmann@tu-bs.de).

Problem (1) (and analogously, the MLD problem) can be rewritten in several ways (see, e.g., [4]–[8]); a fairly straight-forward, and the most common generic approach (cf. [9]–[11]) is to reformulate (1) as the integer program

$$\begin{aligned} \min \quad & \mathbb{1}^\top \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{H}\boldsymbol{x} - 2\boldsymbol{z} = \boldsymbol{0},\ \mathbb{1}^\top \boldsymbol{x} \geq 1,\ \boldsymbol{x} \in \{0,1\}^n,\ \boldsymbol{z} \in \mathbb{Z}^n_{\geq 0}, \end{aligned} \tag{2}$$

where $\mathbb{1}$ is the all-ones vector. Indeed, it is easily seen that the auxiliary nonnegative integer variables $\boldsymbol{z}$ enforce the required (even) codeword parity in all rows of the equality constraints.

Modern IP solvers are based on effective combination of the branch-and-bound paradigm (creating subproblems by fixing some variables and pruning parts of the resulting search tree, e.g., if local objective bounds prevent further improvements in the respective subtree) and the use of (linear) cutting planes, i.e., inequalities that are valid for integral feasible points but may be violated by fractional solutions of the commonly employed linear programming (LP) relaxations. Polyhedral investigations of the codeword polytope (i.e., the convex hull of integer feasible points of (1)) in [7], [12]–[18] (and other works) have identified, in particular, the following class of valid inequalities to strengthen the LP relaxation and to improve solver performance when tackling problems like (2):

$$\sum_{j\in S} x_j - \sum_{j\in \operatorname{supp}(\boldsymbol{h})\setminus S} x_j \leq |S| - 1, \tag{3}$$

where $\emptyset \neq S \subset \operatorname{supp}(\boldsymbol{h}) := \{j \in \{1,2,\dots,n\} : h_j \neq 0\}$ is odd and $\boldsymbol{h}$ is a parity-check. These inequalities became known as (odd) forbidden-set (FS) inequalities. It has been established that for any given parity-check, at most one of the associated FS inequalities can be violated at a time (by a given point) [19], and that if a violated one (called a *cut*) exists, it can be found in polynomial time; the state-of-the-art method for the latter is the sorting-based routine described in [20] (see also [21]).

Moreover, several papers discussing IP or adaptive LP approaches for (1) or MLD incorporate heuristics to find new redundant parity-checks (RPCs)—i.e., some linear combinations (over $\mathbb{F}_2$, i.e., in modulo-2 arithmetic) of rows of $\boldsymbol{H}$—that hopefully yield violated FS inequalities, see, for instance, [19]–[22]. While some results exist on when such heuristics will succeed in finding a cut that is violated by a given (fractional) point (cf., e.g., [10], [19]), there is no general guarantee for this. Indeed, it has often been claimed that the problem of finding cut-inducing RPCs is intractable (see, e.g., [10], [19], [21]), but a formal proof of this statement appears to be missing so far.

In the following, we close this gap by providing an NP-hardness proof of the RPC separation problem. Moreover, we will introduce an IP model to solve it, i.e., to find a maximally violated FS cut among all possible FS inequalities w.r.t. all parity-checks, known or unknown (also allowing to conclude that all FS inequalities are satisfied, should this be the case), and assess its potential practical applicability in some computational experiments with problem (2).

II. Exact RPC Cut Generation/Separation

Clearly, if for a given $\boldsymbol{x}^* \in [0,1]^n$, not all FS inequalities (3) are satisified, there exists at least one parity-check $\boldsymbol{h}$ and odd index set $S \subset \operatorname{supp}(\boldsymbol{h})$ such that the constraint violation

$$-|S| + 1 + \sum_{j\in S} x_j^* - \sum_{j\in\operatorname{supp}(\boldsymbol{h})\setminus S} x_j^* > 0.$$

Thus, the *RPC separation problem* can be stated as follows: *Given $\boldsymbol{x}^* \in [0,1]^n$, find a valid parity-check that yields a maximally violated FS cut, or conclude that all FS inequalities are satisfied.* This task can be formulated as the optimization problem

$$\begin{aligned} \max \quad & -|S| + 1 + \sum_{j\in S} x_j^* - \sum_{j\in\operatorname{supp}(\boldsymbol{h})\setminus S} x_j^* \qquad (4)\\ \text{s.t.} \quad & \boldsymbol{h} \text{ is a parity-check,}\\ & \emptyset \neq S \subseteq \operatorname{supp}(\boldsymbol{h}),\ |S| \text{ odd;} \end{aligned}$$

clearly, the optimal objective value is strictly larger than zero if and only if a cut-inducing parity-check $\boldsymbol{h}$ exists. The objective is the violation of the respective FS inequality associated with $\boldsymbol{h}$ and an odd subset S of its support at the given point $\boldsymbol{x}^*$, and the optimization problem searches for the largest such violation among all valid parity-checks.

With a few observations, we can turn the above abstract optimization problem into an equivalent concrete IP: First, recall that parity-checks correspond to dual codewords, cf. [19]; thus, we can express the first constraint in terms of a generator matrix $\boldsymbol{G}$ of the code (which can be obtained as a basis matrix for the $\mathbb{F}_2$-nullspace of the given parity-check matrix $\boldsymbol{H}$ by Gaussian elimination) as $\boldsymbol{G}^\top\boldsymbol{h} = \boldsymbol{0} \mod 2$ with $\boldsymbol{h} \in \{0,1\}^n$. To model the odd subsets S of $\operatorname{supp}(\boldsymbol{h})$, we introduce binary variables $\boldsymbol{s} \leq \boldsymbol{h}$ with the constraint that $\mathbb{1}^\top\boldsymbol{s}$ is odd, i.e., $\mathbb{1}^\top\boldsymbol{s} = 1 \mod 2$. The condition $S \neq \emptyset$ is then reflected as $\mathbb{1}^\top s \geq 1$, and is automatically ensured by the previous odd-sum constraint. The objective of (4) can be rewritten as

$$\begin{aligned} &-|S| + 1 - \sum_{j\in\operatorname{supp}(\boldsymbol{h})} x_j^* + 2\sum_{j\in S} x_j^*\\ =& -\mathbb{1}^\top\boldsymbol{s} + 1 - (\boldsymbol{x}^*)^\top\boldsymbol{h} + 2(\boldsymbol{x}^*)^\top(\boldsymbol{s})\\ =&(2\boldsymbol{x}^* - \mathbb{1})^\top\boldsymbol{s} - (\boldsymbol{x}^*)^\top\boldsymbol{h} + 1. \end{aligned}$$

Assuming w.l.o.g. that $\operatorname{rank}(\boldsymbol{H}) = m < n$ (so $\boldsymbol{G} \in \{0,1\}^{n\times(n-m)}$ with rank $n-m$), with auxiliary integer variables $\boldsymbol{z}^h \in \mathbb{Z}^{n-m}_{\geq 0}$ and $z^s \in \mathbb{Z}_{\geq 0}$ to linearize the modulo-2 equations, we thus arrive at the *RPC separation IP* (equivalent to (4))

$$\begin{aligned} \max \quad & (2\boldsymbol{x}^* - \mathbb{1})^\top\boldsymbol{s} - (\boldsymbol{x}^*)^\top\boldsymbol{h} + 1 \qquad (5)\\ \text{s.t.} \quad & \boldsymbol{G}^\top\boldsymbol{h} - 2\boldsymbol{z}^h = \boldsymbol{0}\\ & \mathbb{1}^\top\boldsymbol{s} - 2z^s = 1\\ & \boldsymbol{s} \leq \boldsymbol{h}\\ & \boldsymbol{h}, \boldsymbol{s} \in \{0,1\}^n,\ \boldsymbol{z}^h \in \mathbb{Z}^{n-m}_{\geq 0},\ z^s \in \mathbb{Z}_{\geq 0}. \end{aligned}$$

(An alternative formulation that does not require knowledge, or construction, of the generator matrix $\boldsymbol{G}$ can be obtained by replacing the first equality constraint by $\boldsymbol{h} = \boldsymbol{H}^\top\boldsymbol{w} - 2\boldsymbol{z}^h$ with $\boldsymbol{w} \in \{0,1\}^m$ and $\boldsymbol{z}^h \in \mathbb{Z}^n_{\geq 0}$; however, this variant has m additional binary variables and m more integer variables.)

Our following result establishes that, in general, exact RPC separation is computationally challenging:

Theorem 1. *The RPC separation problem is NP-hard.*

Proof. The NP-hardness is due to the fact that the separation problem contains as a special case the task of finding the girth of a binary matroid (i.e., the minimum distance problem (1)), which is well-known to be NP-hard [2]: Let $\boldsymbol{A} \in \{0,1\}^{m\times n}$ be an arbitrary instance of the binary matroid girth problem; w.l.o.g., we may assume $\operatorname{rank}(\boldsymbol{A}) = m < n$ (over $\mathbb{F}_2$). With $\boldsymbol{x}^* := (1/2)\mathbb{1}$ and $\boldsymbol{G}^\top := \boldsymbol{A}$, the corresponding instance of the RPC separation problem (in IP form (5)) collapses to

$$\begin{aligned} &\max_{\boldsymbol{s},\boldsymbol{h}\in\{0,1\}^n} -\tfrac{1}{2}\mathbb{1}^\top\boldsymbol{h} + 1\\ &\quad\text{s.t.}\quad \boldsymbol{A}\boldsymbol{h} = \boldsymbol{0} \mod 2,\ \boldsymbol{s} \leq \boldsymbol{h},\ \mathbb{1}^\top\boldsymbol{s} = 1 \mod 2\\ \Leftrightarrow\quad & 1 - \tfrac{1}{2}\min_{\boldsymbol{h}\in\{0,1\}^n}\left\{\mathbb{1}^\top\boldsymbol{h} : \boldsymbol{A}\boldsymbol{h} = \boldsymbol{0} \mod 2,\ \boldsymbol{h} \neq \boldsymbol{0}\right\}. \end{aligned}$$

The latter minimization problem is precisely the girth problem for the instance given by $\boldsymbol{A}$, so unless P = NP, there cannot exist a polynomial-time algorithm for the RPC separation problem. □

Remark 2. *Note that, although the above proof shows hardness of a separation problem for a large class of valid inequalities for the minimum distance problem* (1) *(and others) by exhibiting that very same problem as a special case, the result is* not *implied by the general so-called "equivalence of optimization and separation" (cf. [23]). Indeed, even if* all *FS inequalities for* all *parity-checks (given and redundant ones) were added to the IP model* (2)*, its LP relaxation is not guaranteed to yield an integral (binary) optimum point [24].*

Moreover, using results from [25], NP-hardness of the RPC separation problem can be seen to persist even if $\boldsymbol{A}$ is restricted to be the parity-check matrix of an LDPC code.

Remark 3. *The problem obtained by minimizing the RPC separation IP's objective (instead of maximizing it) is, in fact, also NP-hard, as it contains the well-known max-cut problem as a special case (similar to the problem of finding a cycle of maximum weight in a binary matroid, cf. [13]); we omit the proof for brevity. The fact that both maximizing and minimizing the objective function over the feasible set of the RPC separation problem are NP-hard indicates that the hardness is inherent in the combinatorial constraints.*

(Similarly, the paper [26] gave a hardness-of-maximization result complementing the intractability of the minimization problem (1) *shown later in [2].)*

Naturally, given its inherent intractability as established by Theorem 1, solving the RPC separation IP (5) will be more challenging than running the heuristic schemes from, e.g., [20], [21]. Ideally, the additional computational effort to gain stronger cuts (and corresponding "optimal" new parity-checks) could lead to a sufficient benefit in terms of search space reduction (particularly, LP relaxation tightening) to compensate for the increased runtime overhead that can be expected. It is also worth pointing out that the IP (5) may also be used in a heuristic fashion: any feasible solution with positive objective value yields a violated FS inequality, so the solving process may be terminated early as soon as the current objective value turns positive, still providing a cut (although not necessarily a most violated one).

III. Computational Experiments

We now turn to some numerical experiments, based on the minimum distance IP (2), to assess the influence of separating FS inequalities and redundant parity-checks in different ways. We employ the open-source MIP solver SCIP [27] (with the LP solver SoPlex that comes with it) and use a variety of test instances from the Channel Code Database [28]. Our test set consists of 27 parity-check matrices, of which 5 belong to array codes, 3 to BCH/Hamming codes, 6 to LDPC codes, 2 to polar codes, one is the Tanner(3,5) code, 5 belong to WiMax and 3 to WiMax-like codes, and 2 to WRAN codes. The average matrix size is about 106×372, ranging from 8 to 588 rows (parity-checks) and from 32 to 1344 variables. For brevity, we forego instance-specific details.

All experiments were carried out in single-thread mode on a Linux machine with Intel Core i7-7700T CPUs (2.9 GHz, 8 MB cache) and 16 GB memory, using SCIP 6.0.2. We set a time limit of 1 hour (3600 s) per instance.

The state-of-the-art separation routine from [20] for FS inequalities associated with *given* parity checks is, in fact, implemented in SCIP already (as part of the XOR constraint handler). We extended the implementation by the best-known RPC cut generation heuristics from [20] and from [21], respectively, as well as by an exact separation routine based on the RPC separation IP (5).

First, we used SCIP as a black-box, applied to the model (2); by default, SCIP does not separate FS inequalities. Then, with separation activated in the root node only, in every node, or on every 5-th level of the branch and bound tree, respectively, we tested the following solver variations:

1) exact separation of FS inequalities for existing parity-checks according to [20] ("ZS", for short),
2) ZS, followed by exact RPC cut separation via (5), if ZS found no violated FS inequality ("ZSe"),
3) ZS, followed (if unsuccessful) by the RPC cut heuristic from [20], followed (if also unsuccessful) by the exact separation routine ("ZS+e"),
4) ZS, followed (if unsuccessful) by the RPC cut heuristic from [20], then (if still unsuccessful) by that from [21], and finally (if still unsuccessful) by the exact separation routine ("ZS++e").

We leave all of SCIP's many other solver settings (w.r.t. general-purpose heuristics, cutting planes, branching rules etc.) at their respective default values. The solver also automatically takes care of cutting plane management, i.e., we neither force inequalities into the LP nor do we remove them by hand, but let the solver decide how to handle and store cuts. (Nevertheless, FS inequalities are given the highest separation priority, so that the solver first tries to find cuts of this type.)

Table I
Overview of experimental results.

solver variant	gap (%)	# solved	nodes	time (s)
SCIP default	15.92	21	74464.3	112.9
ZS (sepa. root only)	12.37	21	81829.1	116.9
ZS (sepa. always)	1.45	26	4394.1	84.1
ZS (sepa. freq. 5)	1.28	26	14518.6	70.9
ZSe (sepa. root only)	12.37	21	81833.0	117.1
ZSe (sepa. always)	1526.64	6	145.2	1562.4
ZSe (sepa. freq. 5)	557.50	10	836.8	837.7
ZS+e (sepa. root only)	12.37	21	81816.9	117.1
ZS+e (sepa. always)	12.63	19	1517.0	468.6
ZS+e (sepa. freq. 5)	12.25	21	7751.2	293.4
ZS++e (sepa. root only)	17.60	21	150529.4	182.8
ZS++e (sepa. always)	5.43	24	2564.8	128.5
ZS++e (sepa. freq. 5)	7.76	23	9256.5	108.9

Table I presents average results over all instances, namely the arithmetic mean optimality gap ((upper bound − lower bound)/(lower bound)·100%), the number of instances (out of 27) that were solved to optimality within the given time limit, and the average number of processed branch-and-bound (search tree) nodes as well as the average running times (in seconds) of the different solver variants. For nodes and times, we state shifted geometric mean values with shifts 100 and 10, respectively, to mitigate the influence of easy instances.

First of all, the overview in Table I clearly confirms previous studies: adding FS cuts during the solving process can significantly help solve IPs like (2). While adding cuts only in the root node can be seen to (adaptively) strengthen the original LP relaxation, the effect becomes much more prominent when parity-checks are also used to separate cuts in deeper levels of the search tree. Comparing the "SCIP default" results with, in particular, the "ZS" results, using (exact) separation of FS cuts based on the *known*, original parity-checks allows for more instances to be solved (thus also yielding a smaller average optimality gap) in significantly shorter time.

The results also show that more is not necessarily better: adding cuts at every search node does lead to the smallest average number of nodes that are explored before certifying optimality (or reaching the time limit), but apparently also significantly increases the time needed to solve the many (LP) subproblems during the branch-and-bound process. Separating FS cuts only at every 5-th level of the search tree typically yields results of comparable quality in a shorter average time, despite requiring more nodes to be explored.

The results incorporating the generation of redundant parity-checks in order to find further violated FS inequalities are somewhat ambiguous. Recall that we only ever used RPC-based schemes if the basic FS cut separation failed. Thus, the longer runtimes and less impressive gains in solution quality of the variants "ZSe", "ZS+e" and "ZS++e" compared to "ZS" indicate that, on average, it does not pay off to search for FS cuts by exploring RPCs if the originally given parity-checks do not already yield such cuts. Nevertheless, it is important to emphasize that this is an empirical average statement only – indeed, e.g., for the Tanner(3,5) instance, the fastest "ZS" version (in this case, the one separating FS cuts in all nodes) solved the minimum distance problem in about 2029 s, whereas the "ZS++e" variant (also with separation in every node) only took roughly 1417 s. Thus, our results do not directly contradict earlier claims that RPC cuts help, although their benefit does appear to be less pronounced when integrating the corresponding separation schemes into a full MIP framework rather than more basic branch-and-cut or adaptive LP decoding procedures that had been explored before. (Recall also that [19] already noted that the relative improvement provided by RPC cuts decreases as the number of variables increases, which might have played a role here as well.)

Finally, the results for "ZSe" and "ZS+e" show that, unfortunately (though not unexpectedly, given Theorem 1), exact RPC cut separation does not pay off, as the IP separation subproblems are too hard in practice to be routinely solved within a branch-and-cut framework for the original problem. On the positive side, we note that in the variant "ZS++e", i.e., where *exact* RPC separation would only be used if both state-of-the-art RPC cut *heuristics* failed, the separation IP (5) was actually *never* called at all. Thus, although one cannot be sure that the most violated cut was used, some violated cut was always found without having to resort to exact separation. Overall, however, the best results were still achieved by variant "ZS" that did not use RPC cuts at all.

IV. Concluding Remarks

In this note, we formally proved computational intractability of the RPC cut separation problem for binary linear codes. Moreover, our numerical experiments demonstrated that an associated IP formulation is indeed too hard to solve in practice to become useful during branch-and-cut methods. While our empirical results further showed that by combining the existing state-of-the-art RPC separation heuristics, the exact problem never needed to be resorted to, they also suggested that, on average, RPC cuts do not help improve the solution process of a current powerful MIP solver compared to using only FS cuts associated with the initial, known parity-checks.

References

[1] S. C. Draper, J. S. Yedidia, and Y. Wang, "ML decoding via mixed-integer adaptive linear programming," in *Proc. ISIT 2007*, 2007, pp. 1656–1660.

[2] A. Vardy, "The Intractability of Computing the Minimum Distance of a Code," *IEEE Trans. Inf. Theory*, vol. 43, no. 6, pp. 1757–1766, 1997.

[3] E. R. Berlekamp, R. J. McEliece, and H. C. A. van Tilborg, "On the inherent intractability of certain coding problems," *IEEE Trans. Inf. Theory*, vol. 24, no. 3, pp. 384–386, 1978.

[4] A. B. Keha and T. M. Duman, "Minimum Distance Computation of LDPC Codes Using a Branch and Cut Algorithm," *IEEE Trans. Commun.*, vol. 58, no. 4, pp. 1072–1079, 2010.

[5] M. Punekar, F. Kienle, N. Wehn, A. Tanatmis, S. Ruzika, and H. W. Hamacher, "Calculating the Minimum Distance of Linear Block Codes via Integer Programming," in *Proc. ISTC 2010*, 2010, pp. 329–333.

[6] A. Tanatmis, S. Ruzika, H. W. Hamacher, M. Punekar, F. Kienle, and N. Wehn, "Valid Inequalities for Binary Linear Codes," in *Proc. ISIT 2009*, 2009, pp. 2216–2220.

[7] K. Yang, X. Wang, and J. Feldman, "Cascaded Formulation of the Fundamental Polytope of General Linear Block Codes," in *Proc. ISIT 2007*, 2007, pp. 1361–1365.

[8] B. Kabakulak, Z. C. Taşkın, and A. E. Pusane, "A Branch-Price-and-Cut Algorithm for Optimal Decoding of LDPC Codes," arXiv:1803.04798 [cs.IT], 2018.

[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, no. 3, pp. 289–293, 1998.

[10] 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.

[11] S. Scholl, F. Kienle, M. Helmling, and S. Ruzika, "Integer Programming as a Tool for Analysis of Channel Codes," in *Proc. SCC 2013*, 2013, pp. 1–6.

[12] F. Barahona and M. Grötschel, "On the cycle polytope of a binary matroid," *J. Comb. Theory, Ser. B*, vol. 40, no. 1, pp. 40–62, 1986.

[13] M. Grötschel and K. Truemper, "Decomposition and Optimization over Cycles in Binary Matroids," *J. Comb. Theory, Ser. B*, vol. 46, no. 3, pp. 306–337, 1989.

[14] A. Tanatmis, "Mathematical Programming Approaches for Decoding of Binary Linear Codes," Ph.D. dissertation, Technische Universität Kaiserslautern, 2011, doctoral dissertation.

[15] R. H. Jeroslow, "On defining sets of vertices of the hypercube by linear inequalities," *Discrete Math.*, vol. 11, no. 2, pp. 119–124, 1975.

[16] J. Feldman, M. J. Wainwright, and D. R. Krager, "Using Linear Programming to Decode Binary Linear Codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 954–972, 2005.

[17] M. Helmling, S. Ruzika, and A. Tanatmis, "Mathematical Programming Decoding of Binary Linear Codes: Theory and Algorithms," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4753–4769, 2012.

[18] G. Lancia and P. Serafini, "The Parity Polytope," in *Compact Extended Linear Programming Models*, ser. EURO Advanced Tutorials on Operational Research. Springer, 2018, pp. 113–121.

[19] M. H. Taghavi and P. H. Siegel, "Adaptive Methods for Linear Programming Decoding," *IEEE Trans. Inf. Theory*, vol. 54, no. 12, pp. 5396–5410, 2008.

[20] X. Zhang and P. H. Siegel, "Adaptive Cut Generation Algorithm for Improved Linear Programming Decoding of Binary Linear Codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 10, pp. 6581–6594, 2012.

[21] H. Falsafein and S. R. Mousavi, "A Generator-Matrix-Based Approach for Adaptively Generating Cut-Inducing Redundant Parity-Checks," *IEEE Commun. Lett.*, vol. 20, no. 4, pp. 640–643, 2016.

[22] M. Miwa, T. Wadayama, and I. Takumi, "A Cutting-Plane Method Based on Redundant Rows for Improving Fractional Distance," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 6, pp. 1005–1012, 2009.

[23] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, 2nd ed., ser. Algorithms and Combinatorics. Heidelberg, Germany: Springer, 1993, vol. 2.

[24] J. Zumbrägel, V. Skachek, and M. F. Flanagan, "On the Pseudocodeword Redundancy of Binary Linear Codes," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4848–4861, 2012.

[25] A. McGregor and O. Milenkovic, "On the Hardness of Approximating Stopping and Trapping Sets," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1640–1650, 2010.

[26] S. C. Ntafos and S. L. Hakimi, "On the Complexity of Some Coding Problems," *IEEE Trans. Inf. Theory*, vol. 27, no. 6, pp. 794–796, 1981.

[27] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig, "The SCIP Optimization Suite 6.0," Zuse Institute Berlin, ZIB-Report 18-26, 2018.

[28] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, "Database of channel codes and ML simulation results," 2019, www.uni-kl.de/channel-codes.