ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414728

ADMM-BASED ML DECODING: FROM THEORY TO PRACTICE

Kira Kraft and Norbert Wehn

Microelectronic Systems Design Research Group
Technische Universität Kaiserslautern
Kaiserslautern, Germany
{kraft, wehn}@eit.uni-kl.de

ABSTRACT

Integer Linear Programming (ILP) is a general method to solve the Maximum-Likelihood (ML) decoding problem for all kinds of binary linear codes. To this end, state-of-the-art techniques use a Branch-and-Bound (B&B) framework to partition the underlying integer linear problem into several relaxed linear problems. These linear problems then have to be solved in reasonable time by an efficient Linear Programming (LP) solver. Recently, the Alternating Direction Method of Multipliers (ADMM) has been proposed for efficient software and hardware LP decoding of sparse codes, hence, an ADMM-based ML decoder seems to be a promising approach. In this paper, we investigate this approach with respect to its algorithmic and implementation-specific challenges.

***Index Terms*—** ML Decoding, Branch and Bound, LP Decoding, ADMM

1. INTRODUCTION

The efficient transmission of digital data is a key component of modern communication systems. In particular in wireless communications, the reliability of data transmission is a major challenge. Channel coding ensures the reliable transmission of data via noisy channels by adding redundant information to the actual data to be transmitted. In his fundamental work, Shannon [10] derived theoretical limits for an error-free transmission under certain assumptions and proved the existence of codes that reach these limits.

In addition to the coding gain through the code itself, the used decoding algorithm is of central importance. Maximum-Likelihood (ML) decoding is an optimal approach that belongs to the class of NP-hard problems [2]. Therefore, in nowaday's decoding applications, which mostly require real-time performance, heuristic decoding algorithms are used. But knowing the performance of a code under ML decoding is very valuable for various reasons:

1. Evaluation of Code Potential: ML decoding allows to compare different codes in a fair way, i.e., without being biased by the performance of their heuristic decoding algorithms. Let us consider Fig. 1 [9], which shows the performance of a non-binary Low-Density Parity-Check (LDPC) code over $\mathrm{GF}(64)$ of dimension $k_b = 48$ and of a binary LDPC code of dimension $k = 50$, both of rate $1/2$, under heuristic Belief-Propagation (BP) and ML decoding [8]. Considering only the heuristic decoding results, the non-binary LDPC code seems to perform much better than the binary LDPC code. However, with the help of ML decoding, it was shown in [9] that the benefit of non-binary LDPC codes over binary LDPC codes only lies in the superiority of the non-binary heuristic decoding algorithm, but not the code itself.

A similar phenomenon can be observed when comparing LDPC and Polar codes. Fig. 2 depicts the heuristic and optimal decoding performance of a $(128, 64)$ CCSDS LDPC code and a Polar code of the same parameters [8]. Under their heuristic decoding algorithms (Sum-Product Algorithm (SPA) with 40 iterations for the LDPC code, Successive Cancellation List (SCL) with list size 2 for the Polar code), the performance of the Polar code is around 0.5 dB better than that of the LDPC code. However, when comparing their performances under ML decoding, the LDPC code yields a superior error-correcting performance of around 1 dB at a Frame Error Rate (FER) of 10^{-3}.

These two examples show the importance of ML decoding for making a fair comparison between different codes.

2. Evaluation of Decoding Potential: ML decoding allows to evaluate the potential of improvement for heuristic decoding algorithms by looking at the gap between their error-correcting performance compared to that of the ML decoder. Considering again Fig. 2, it can be observed that there is a lot of potential for the LDPC code decoding algorithm, as the heuristic decoding algorithm (SPA) leaves a large gap towards ML decoding. On the other hand, the heuristic Polar code decoding algorithms is already very close to the ML decoding performance. Indeed, for larger list sizes around $L = 32$, SCL decoding already hits the ML decoding bound, so there is no further improvement possible [8].

This work was supported by the InnoProm program (projekt no. 84003923) of the EU and the state Rhineland-Palatinate, Germany.

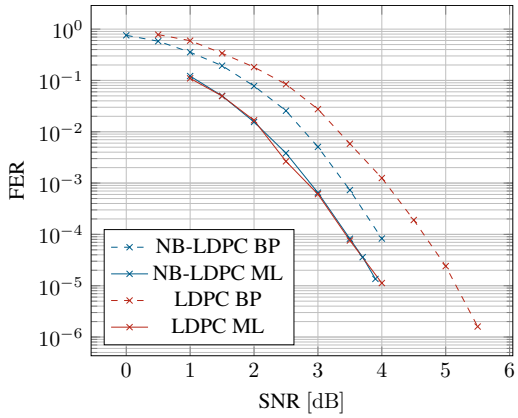


Fig. 1. Non-Binary vs. Binary LDPC Code from [9]

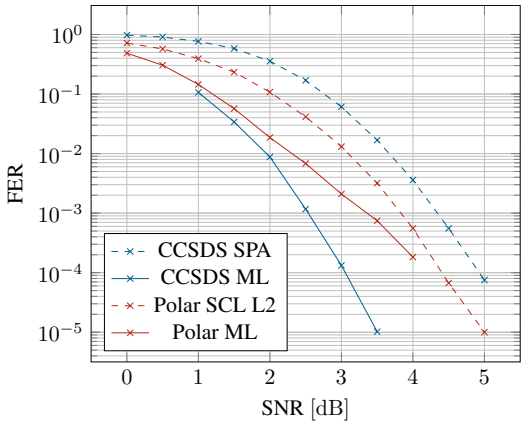


Fig. 2. LDPC Code vs. Polar Code

3. Decoding for Real Applications: Although ML decoding involves solving an NP-hard problem, i.e., broadly speaking, the decoding complexity grows exponentially in the blocklength of the code, there are some applications that can tolerate such a high decoding complexity. These applications include, e.g., deep space communication, where a transmission of data is very costly in terms of latency and needed power for the transmission. Since these applications do not require real-time decoding, it can be more efficient to spend the efforts for best-possible error-correction on receiver-side instead of requesting a retransmission.

These three use-cases show the importance and value of ML decoding. As of today, ML decoding is possible for blocklengths of up to $500 - 1000$ bits [8].

There are different approaches to decrease the runtime of ML decoding. First, improvements can be achieved on algorithmic level. For instance, [7] proposes a sophisticated software algorithm for Integer Linear Programming (ILP)-based ML decoding of binary linear block codes that uses a Branch-and-Bound (B&B) framework tailored to ML decoding, which partitions the underlying integer program into several relaxed linear programs. Thus, the main effort of ILP-based ML decoding lies in repeatedly solving the LP decoding problem. Recently, using the Alternating Direction Method of Multipliers (ADMM) [3], a method from convex optimization that combines decomposability with strong convergence, has been proposed for LP decoding [15]. It has shown to be less complex than other Simplex-based LP solvers for low-density codes [6] with respect to the number of needed arithmetic operations.

In addition to improvements on algorithmic level, further speedup can be achieved by using dedicated hardware accelerators. A first step towards hardware-based ML decoding has been made in [12], where the ADMM as LP decoder has been implemented on an FPGA. However, this was not considered in the context of ML decoding.

In this paper, we will highlight algorithmic and implementation-specific challenges that have to be tackled for ADMM-based ML decoding and show different ways to overcome them.

2. BACKGROUND

Feldman et al. [4] showed that, for memoryless channels and binary linear block codes, the ML decoding problem can be written as an integer program

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

where λ_i are the log-likelihood ratios (LLRs) and H is the parity-check matrix of the code $\mathcal{C}$. In the remainder of this paper, n will denote the blocklength and k the dimension of the code.

In optimization theory, integer programs are oftentimes solved by B&B algorithms. These algorithms repeatedly calculate upper and lower bounds $z_{\mathrm{LB}} \leq z \leq z_{\mathrm{UB}}$ for the optimal objective value z. Upper bounds $z_{\mathrm{UB}} = \lambda^\top \widetilde{x}$ are induced by valid codewords $\widetilde{x} \in \mathcal{C}$, while lower bounds z_{LB} are computed solving a relaxation of (1). In the case of ML decoding, this relaxation leads to the LP decoding problem

$$\begin{aligned} z_{\mathrm{LB}} = \min \quad & \lambda^\top x \\ \text{s.t.} \quad & H \cdot x = 0 \pmod 2, \end{aligned} \tag{2}$$

where the integrality condition is omitted.

Branching is performed to subdivide the problem into several subproblems by partitioning the feasible set. A natural branching strategy for binary ML decoding is to fix one component x_i to zero or one, thus inducing two subproblems and spanning a binary tree. For every node in this tree, new lower and upper bounds are computed to repeatedly improve the bounds of the original problem, called bounding. These

branching and bounding steps are repeated until $z_{\mathrm{LB}} = z = z_{\mathrm{UB}}$. The effectiveness of the B&B algorithm relies on the fact that good bounds allow to prune large parts of the B&B tree, i.e., the set of feasible solutions.

The main computational effort of a B&B ML decoder is the computation of the lower bounds, i. e., solving the relaxed LP decoding problems (cf. Eq. (2)). Recently, the ADMM [3] has been proposed for efficient LP decoding [15]. The ADMM-based LP decoder uses the formulation

$$\begin{aligned} \min \quad & \lambda^\top x \\ \text{s. t.} \quad & T_j x = z_j \quad \forall j \in J \\ & z_j \in \mathcal{P}_{d_j} \quad \forall j \in J, \end{aligned} \tag{3}$$

where J is the set of parity rows, d_j the check degree of row j, and the rows of $T_j \in \{0,1\}^{d_j \times n}$ equal the unit vectors e_i where $H_{j,i} = 1$.

The set $\mathcal{P}_d$ (here we call it $\mathcal{P}_{d,\mathrm{even}}$) denotes the (even) parity polytope

$$\mathcal{P}_d := \mathcal{P}_{d,\mathrm{even}} := \mathrm{conv}\{x \in \{0,1\}^d : \sum_{i=1}^{d} x_i \text{ is even}\}.$$

In the following, we will use the notion of the odd parity polytope [5]

$$\mathcal{P}_{d,\mathrm{odd}} := \mathrm{conv}\{x \in \{0,1\}^d : \sum_{i=1}^{d} x_i \text{ is odd}\}.$$

The main effort of ADMM-based LP decoding are projections onto the parity polytopes $\mathcal{P}_{d_j,\mathrm{even}}$ for every parity row j. The first projection algorithm for this purpose was presented by Barman et al. [1]. Several other projection algorithms are proposed in the literature [15, 5, 11, 13, 14].

3. CHALLENGES FOR ADMM-BASED ML DECODING

In this section, we will identify some major challenges that arise for ADMM-based ML decoding and give different possibilities how to overcome them.

3.1. Challenge I: Interface

When a component of the input vector is set to 0 or 1 by the outer B&B framework, this information has to be incorporated into the LP solver. The subproblem that arises after a fixing of some components should be of smaller dimension than the original problem. For the Simplex algorithm, which works directly on the LP formulation in Eq. (2), the strategy is straight-forward: Fixing a component translates into a deletion of the fixed variable and its corresponding column of the H-matrix. Additionally, the right-hand-side of Eq. (2) has to be adjusted depending on the value the component is fixed to and the entries of the removed column. In this way, fixing one component reduces the dimension of the problem by one.

As the ADMM does not use the LP formulation in Eq. (2), but the formulation in Eq. (3), the problem is slightly more complex. We will use Theorem 6 of [5] to incorporate the information about fixed components directly into the ADMM's projection algorithm. The theorem states that whenever a component is fixed, the remaining problem can be considered as a projection onto a smaller-dimensional even or odd parity polytope:

Theorem 1 ([5]) *Let $d \geq 2$ and $i \in \{1, \ldots, d\}$. Then it holds:*

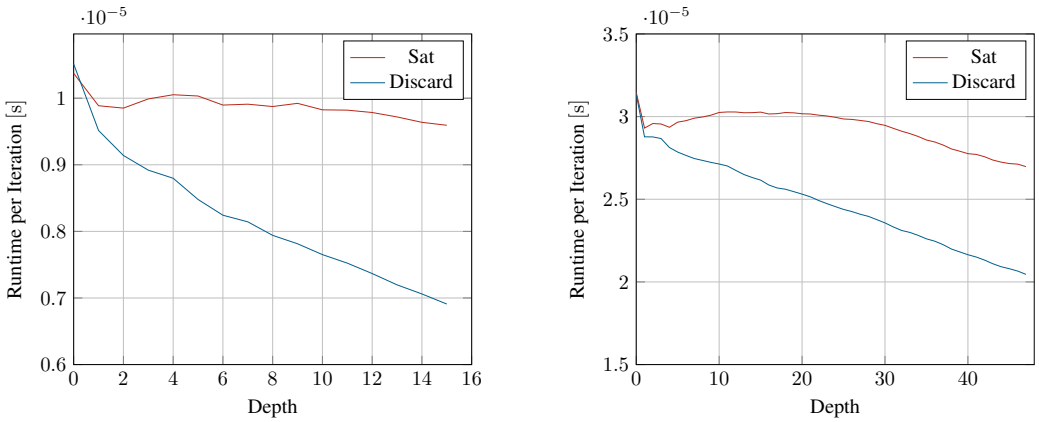
1) $\{z \in \mathcal{P}_{d,\mathrm{even}} : z_i = 1\} =$
$\{(z'_1, \ldots, z'_{i-1}, 1, z'_i, \ldots, z'_{d-1})^\top \in \mathbb{R}^d : z' \in \mathcal{P}_{d-1,\mathrm{odd}}\}$
2) $\{z \in \mathcal{P}_{d,\mathrm{even}} : z_i = 0\} =$
$\{(z'_1, \ldots, z'_{i-1}, 0, z'_i, \ldots, z'_{d-1})^\top \in \mathbb{R}^d : z' \in \mathcal{P}_{d-1,\mathrm{even}}\}$
3) $\{z \in \mathcal{P}_{d,\mathrm{odd}} : z_i = 1\} =$
$\{(z'_1, \ldots, z'_{i-1}, 1, z'_i, \ldots, z'_{d-1})^\top \in \mathbb{R}^d : z' \in \mathcal{P}_{d-1,\mathrm{even}}\}$
4) $\{z \in \mathcal{P}_{d,\mathrm{odd}} : z_i = 0\} =$
$\{(z'_1, \ldots, z'_{i-1}, 0, z'_i, \ldots, z'_{d-1})^\top \in \mathbb{R}^d : z' \in \mathcal{P}_{d-1,\mathrm{odd}}\}$

Following this theorem, fixed components can be discarded from the calculations in the ADMM by adjusting the projection algorithm, thereby reducing the dimension of the subproblem. In the projection algorithm from [5], projections onto the even or odd parity polytope only differ in the cut-search algorithm, where either an odd- or an even-sized (potential) cutting set has to be found.

Alternatively, the information about fixed components can be incorporated into the LP solver by saturating the corresponding LLR values to the highest or smallest possible values. With this strategy, fixing a component does no longer reduce the dimension of the LP decoding problem.

The runtime per ADMM iteration of the two approaches depending on the depth of the binary B&B tree, i.e., the number of fixed components, is depicted in Fig. 3 for two different LDPC codes [8]. The measurement was carried out on one core of an Intel® Core™ i7-6700 CPU 3.4 GHz with 16 GB memory. It can clearly be seen that discarding fixed components and thereby reducing the dimension of the problem results in a large reduction in runtime compared to saturating the LLRs of fixed components, making it a good choice for efficient software implementations.

However, from a hardware implementation perspective, saturating LLR values of fixed components has a large benefit, because the dimension and structure of the ADMM are constant. Thus, the same hardware instance of the ADMM can be used for all nodes of the B&B tree. On the other hand, discarding fixed components would result in different hardware instances of the ADMM for every node of the B&B tree, therefore, this approach is not applicable for pure hardware implementations.



(a) Runtime of CCSDS(32,16)

(b) Runtime of LDPC(96,48)

Fig. 3. Runtime Per ADMM Iteration of the Two Different Approaches

3.2. Challenge II: Infeasibility of Subproblems

When moving deeper in the B&B tree, fixing more and more components of the input will eventually end up in an invalid configuration, i.e., an infeasible subproblem as input to the LP solver. In contrast to the Simplex method, the ADMM does not indicate whether the received subproblem is feasible, since the ADMM formally solves some particular unconstrained dual problem of Eq. (3) that is never infeasible. We observed that, for a subproblem without feasible solutions, the ADMM reaches the maximum number of iterations, which also happens if the bounds of the stopping criteria are set too strict. If infeasible subproblems are not detected correctly, the result of the ML decoder can be falsified.

One possibility to solve this problem is to check the validity of the subproblem before it is passed to the ADMM, i.e., by checking whether the system of linear equations (cf. Eq. (1)) of the current subproblem is solvable. To this end, a transformation into reduced row echelon form has to be performed, which results in a huge computational overhead.

In the case of a systematic code, we propose a more efficient solution by only fixing within the components belonging to the systematic information. Thereby, it is guaranteed that the configuration always yields a feasible subproblem. As soon as k bits are fixed by the B&B, these k bits are reencoded to obtain the only valid codeword for the remaining subtree. However, this has an impact on efficient search strategies in the B&B tree [7], which can result in a larger number of node visits. Thus, a trade-off exists between the increased complexity of checking for an infeasible subproblem and an increased number of node visits.

3.3. Challenge III: ML Certificate

The ML certificate guarantees that whenever an LP decoder outputs an integer solution, this solution is indeed the ML solution. In contrast to pure LP decoding, this ML certificate is very important for the correctness of the B&B ML decoder. Thus, the last challenge lies in maintaining the ML certificate for the ADMM als LP decoder.

There are various problems that can cause a loss of the ML certificate: First, as pointed out in [12], using a penalized version of the ADMM removes the ML certificate property, although a penalization term can largely improve the error-correcting performance of pure LP decoding. Second, terminating the ADMM after a maximum number of iterations removes the ML certificate property, as the algorithm might be stopped before it has converged to its final solution. Although setting a maximum number of iterations largely reduces the execution time with negligible loss in error-correcting performance for pure LP decoding, special care has to be taken for ML decoding. Third, an implementation of the ADMM with a fixed-point number representation might falsely return integer values due to quantization effects. To overcome this problem, the fixed-point number representation has to be carefully selected for every code.

4. CONCLUSION AND FUTURE WORK

In this paper, we bridged the gap between theory and practice for ILP-based ML decoding. In theory, the B&B requires solving an LP relaxation (cf. Eq. (2)) that can be done by any LP solver. Here, we focused on the ADMM, since it has shown to be less complex compared to other LP solvers for low-density codes. However, in practice, major algorithmic and implementation-specific challenges have to be met when inserting the ADMM as LP solver. We have highlighted three of these challenges and presented different ways how to overcome them.

As future work, we will focus on the hardware implementation of an ADMM-based ML decoder. To achieve this, not only the challenges highlighted in this paper have to be met, but also additional studies regarding implementation-specific questions (e.g., quantization, parallelization) have to be carried out.

5. REFERENCES

[1] S. Barman, X. Liu, S. C. Draper, and B. Recht. Decomposition methods for large scale LP decoding. *IEEE Transactions on Information Theory*, 59(12):7870–7886, Dec 2013.

[2] E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Transactions on Information Theory*, 24(3):384–386, 1978.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[4] J. Feldman, M. J. Wainwright, and D. R. Karger. Using linear programming to decode binary linear codes. *IEEE Transactions on Information Theory*, 51(3):954–972, March 2005.

[5] F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn. A reduced-complexity projection algorithm for ADMM-based LP decoding, 2019.

[6] F. Gensheimer, S. Ruzika, S. Scholl, and N. Wehn. ADMM versus simplex algorithm for LP decoding. In *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 211–215, 2016.

[7] M. Helmling, E. Rosnes, S. Ruzika, and S. Scholl. Efficient maximum-likelihood decoding of linear block codes on binary memoryless channels. In *IEEE International Symposium on Information Theory (ISIT), 2014*, pages 2589–2593. IEEE, 2014.

[8] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn. Database of Channel Codes and ML Simulation Results. `www.uni-kl.de/channel-codes`, 2019.

[9] S. Scholl, F. Kienle, M. Helmling, and S. Ruzika. ML vs. BP decoding of binary and non-binary LDPC codes. In *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 71–75, 2012.

[10] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 1948.

[11] M. Wasson and S. C. Draper. Hardware based projection onto the parity polytope and probability simplex. In *2015 49th Asilomar Conference on Signals, Systems and Computers*, pages 1015–1020, Nov 2015.

[12] M. Wasson, M. Milicevic, S. C. Draper, and G. Gulak. Hardware-based linear program decoding with the alternating direction method of multipliers. *IEEE Transactions on Signal Processing*, 67(19):4976–4991, 2019.

[13] H. Wei and A. H. Banihashemi. An iterative check polytope projection algorithm for ADMM-based LP decoding of LDPC codes. *IEEE Communications Letters*, 22(1):29–32, Jan 2018.

[14] G. Zhang, R. Heusdens, and W. B. Kleijn. Large scale LP decoding with low complexity. *IEEE Communications Letters*, 17(11):2152–2155, November 2013.

[15] X. Zhang and P. H. Siegel. Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers. In *2013 IEEE International Symposium on Information Theory*, pages 1501–1505, July 2013.