Group-Based Layered Scheduling of ADMM Decoding for LDPC Codes

Xing-Long Zhang(✉), Meng Niu, Luo-Hui Su, and Ke-Pu Song

AVIC Xi'an Flight Automatic Control Research Institute, Xi'an 710065, China
zxilong@163.com

Abstract. For low-density parity-check (LDPC) codes decoding by alternating direction method of multipliers (ADMM), the layered scheduling sequentially updates the messages of check nodes one by one. Though the layered scheduling can speed up the convergence rates, it may limit the throughput when implementing the ADMM decoder with multi-core systems due to its serial style. To circumvent this problem, a group-based layered scheduling is proposed by updating a group of check node messages at one time. Extensive simulation results for the proposed scheme over two typical LDPC codes with the ADMM penalized decoding algorithm are provided.

Keywords: ADMM decoding · LDPC codes · Message scheduling
Convergence rate

1 Introduction

It is well known that the decoding of low-density parity-check (LDPC) codes can be formalized by an optimization problem and solved by linear programming (LP) [1]. However, the computational complexity of LP decoding is high when using the classical LP solvers such as the simplex and the interior point method. Recently, an efficient LP decoding method for LDPC codes based on the alternating direction method of multipliers (ADMM) is proposed [2]. Simulations show that both the error rate performances and the computational complexity of the ADMM decoding are comparable with that of the classical belief propagation (BP) decoding [3, 4].

The Euclidean projection operation is the most complex part in the ADMM decoding [5]. To reduce the decoding complexity, Zhang and Siegel proposed a two-step method to simplify the operation [3]. Firstly, the hyperplane in the check polytope contains the projection results is determined. Then a simple optimization problem is solved to find the exact projection points. Later, the Euclidean projection is reduced by using a linear projection algorithm onto simplex which is effective when the dimension of the input is high [6]. Recently, Jiao et al. proposed a look-up table (LUT) based method to simplify the Euclidean projections [7]. In addition to simplify the Euclidean projection itself, the complexity of the ADMM decoding can also be reduced by saving the number of Euclidean projections [8].

Another way to reduce the complexity is to accelerate the convergence rate of the ADMM decoding by using message scheduling [9, 10]. In [9], the layered scheduling for the ADMM decoding parallel to that of the BP decoding is proposed. Simulation

B. Li et al. (Eds.): ChinaCom 2017, LNICST 237, pp. 207–214, 2018.
https://doi.org/10.1007/978-3-319-78139-6_22

results in [9] show that the decoding time of the ADMM decoding can be reduced significantly by combining the layered scheduling and the method proposed in [8]. A problem for the layered scheduling is its fully serial structure which may prevent it from being implemented with parallel architectures. In this paper, we propose a partial parallel scheduling method for ADMM decoding, named group-based layered scheduling. Simulation results for two typical LDPC codes with different parallel degrees are provided.

2 Preliminaries

Suppose an LDPC code C with information length K and block length N. The parity-check matrix $\boldsymbol{H}$ of C has M rows and N columns. The variable nodes in the Tanner graph of $\boldsymbol{H}$ are indexed by $\mathcal{I} = \{1, 2, \ldots, N\}$, and the check nodes are indexed by $\mathcal{J} = \{1, 2, \ldots, M\}$. Let the degree of i-th variable node v_i be d_i and the degree of j-th check node c_j be d_j. Let $\mathcal{N}(i) = \{j \in \mathcal{J} : H_{ji} = 1\}$ be the index set of neighbors of v_i. Similarly, let $\mathcal{M}(j) = \{i \in \mathcal{I} : H_{ji} = 1\}$ be the index set of neighbors of c_j. Let $\boldsymbol{x}$ be an LDPC codeword of length N and $\boldsymbol{r}$ be the received vector when transmitting $\boldsymbol{x}$. The LP decoding of LDPC codes can be formulated as

$$\begin{aligned} &\min \quad \boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{x} \\ &\text{s.t.}\, \mathbf{T}_j \boldsymbol{x} = \boldsymbol{z}_j, \boldsymbol{z}_j \in \mathbb{PP}_{d_j}, \forall j \in \mathcal{J} \end{aligned} \tag{1}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^N$ is the vector of log-likelihood ratios (LLRs) and the ith component of $\boldsymbol{\gamma}$ is defined as

$$\gamma_i = \log\left(\frac{\Pr(r_i|0)}{\Pr(r_i|1)}\right), \tag{2}$$

the $d_j \times N$ binary matrix $\mathbf{T}_j$ selects out the d_j components of $\boldsymbol{x}$ that participate in the j-th check node, $\boldsymbol{z}_j \in \mathbb{R}^{d_j}$ are "replica" variables, and the check polytope $\mathbb{PP}_{d_j}$ is the convex hull of all binary vectors of length d_j with an even number of 1s [3]. The augmented Lagrangian of LP problem (1) with scaled dual variables is [3]

$$L_\rho(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}) = \boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{x} + \frac{\rho}{2} \sum\nolimits_{j \in \mathcal{J}} \left\| \mathbf{T}_j \boldsymbol{x} - \boldsymbol{z}_j + \boldsymbol{y}_j \right\|_2^2 - \frac{\rho}{2} \sum\nolimits_{j \in \mathcal{J}} \left\| \boldsymbol{y}_j \right\|_2^2, \tag{3}$$

where $\boldsymbol{y}_j \in \mathbb{R}^{d_j}$ is the scaled dual variable and $\rho > 0$ is the penalty parameter. Based on (3), the $(k + 1)$-th iteration of ADMM decoding is described as follows

$$\boldsymbol{x}^{k+1} := \operatorname{argmin}_{\boldsymbol{x}} \left(\boldsymbol{\gamma}^{\mathrm{T}} \boldsymbol{x} + \frac{\rho}{2} \sum\nolimits_{j \in \mathcal{J}} \left\| \mathbf{T}_j \boldsymbol{x} - \boldsymbol{z}_j^k + \boldsymbol{y}_j^k \right\|_2^2 \right), \tag{4}$$

$$\boldsymbol{z}_j^{k+1} := \Pi_{\mathbb{PP}_{d_j}} \left(\mathbf{T}_j \boldsymbol{x}^{k+1} + \boldsymbol{y}_j^k \right), \tag{5}$$

$$\boldsymbol{y}_j^{k+1} := \boldsymbol{y}_j^k + \mathbf{T}_j \boldsymbol{x}^{k+1} - \boldsymbol{z}_j^{k+1}, \tag{6}$$

where $\Pi_{\mathbb{PP}_{d_j}}(\mathbf{u})$ is the Euclidean projection of vector $\mathbf{u}$ onto $\mathbb{PP}_{d_j}$. The minimization of (4) can be calculated in component-wise as follows [2]

$$x_i = \Pi_{[0,1]}\left(\frac{1}{d_i}\left(\sum\nolimits_{j\in N_v(i)}\left(z_j^{(i)} - y_j^{(i)}\right)\right) - \frac{1}{\rho}\gamma_i\right) \tag{7}$$

where $\Pi_{[0,1]}(\cdot)$ denotes the projection onto the interval [0, 1], and the superscript (i) denotes the entries in $\boldsymbol{z}_j$ and $\boldsymbol{y}_j$ that correspond to the variable node i. The over-relaxed ADMM decoder can be implemented by replacing $\mathbf{P}_j\boldsymbol{x}^{k+1}$ in the $\boldsymbol{z}$- and $\boldsymbol{y}$-updates in (5) and (6) with

$$\theta\mathbf{P}_j\boldsymbol{x}^{k+1} + (1 - \theta)\boldsymbol{z}_j^k \tag{8}$$

where θ is an over-relaxation parameter and we choose $\theta = 1.9$ in our simulations as in [2].

It has been observed that the performance of ADMM decoder is worse than the BP decoding in the waterfall region. To address this problem, a penalty term is added to the objective function of (1) to penalize pseudocodewords. Two frequently used penalty terms are l_1 and l_2 penalty functions [4]. For more information about penalty functions, please refer to [11, 12]. In our simulations, the ADMM penalized decoder with l_1 penalty function is used.

3 Group-Based Layered Scheduling Algorithm

In the original ADMM decoding, the flooding scheduling simultaneously updates all the variable-to-check messages followed by all the check-to-variable messages. The advantage of the flooding scheduling is its fully parallel structure. While for the layered scheduling, the message corresponds to the check nodes are updated sequentially. For the j-th check node, the layered scheduling updates the variable-to-check messages associated with c_j firstly and then update the messages from c_j to its neighbor variable nodes. After that, the layered scheduling deals with the $(j + 1)$-th check node. The layered scheduling converges faster than the flooding scheduling. But the serial message updates may hinder it from being implemented in parallel. In the following, we propose a group-based layered scheduling which shares the advantages of the flooding scheduling and the layered scheduling.

Assuming the M check nodes are divided into G groups, and each group contains $M/G = M_G$ check nodes. Let g be the index of the check-node groups. The group-based layered scheduling algorithm is described in Algorithm 1. Three remarks about Algorithm 1 need to be clarified:

(1) When the ADMM penalized decoding is used, line 8 in Algorithm 1 should be calculated based on (12) in [4] for l_1 penalty function and (14) in [4] for l_2 penalty function;
(2) For the over-relaxed ADMM, calculating z_j in line 12 of Algorithm 1 should be based on (5) and (8);
(3) We do not use the traditional stopping condition for the ADMM framework such as the stopping condition in [2]. In Algorithm 1, we use the structural property of LDPC codes to terminate the iterative decoding. In particular, we test whether the hard decision bits satisfy all the parity checks in line 20. This is known as the early termination (ET) scheme in the literature. Simulations show that ADMM decoding with the ET scheme converges faster than the stopping condition in [2]. Therefore, we use the ET scheme throughout our simulations in the next section.

Algorithm 1: The group-based layered scheduling algorithm
1: **Kernel 1:** Initialization
2: Calculate $\boldsymbol{\gamma}$ based on (2);
3: $\boldsymbol{z}_j \leftarrow \mathbf{0.5}$, $\boldsymbol{y}_j \leftarrow \mathbf{0}$.
4: **For** *Iter* from 1 to *iter_max* **do**
5: **For** g from 1 to G **do**
6: **Kernel 2:** variable node update
7: **For** all check nodes c_j with $(\mathrm{g}-1)M_G+1 \le j \le \mathrm{g}M_G$ **do**
8: Calculate x_s for all $\mathrm{s} \in \mathcal{M}(j)$ based on (7)
9: **End for**
10: **Kernel 3:** check node update
11: **For** all check nodes c_j with $(\mathrm{g}-1)M_G+1 \le j \le \mathrm{g}M_G$ **do**
12: Calculate $\boldsymbol{z}_j$ based on (5)
13: Calculate $\boldsymbol{y}_j$ based on (6)
14: **End for**
15: **End for**
16: **Kernel 4:** Hard decisions from soft values
17: **For** all varialbe nodes v_i with $1 \le i \le N$ **do**
18: **If** $(x_i > 0.5)$ $\hat{x}_i$=1; else $\hat{x}_i$=0.
19: **End for**
20: **If** ($\mathbf{H}\hat{\boldsymbol{x}}^T$=**0**) break;
21: **End for**

In general, the flooding scheduling is used in each group of check nodes in Algorithm 1. However, for different groups of check nodes, we use the layered scheduling. Therefore, when $G = 1$, Algorithm 1 is indeed the flooding scheduling method; when $G = M$, Algorithm 1 is the layered scheduling method in [9].

In the following, we give a simple discussion on the parallelism of Algorithm 1 with different group size M_G. The "**For**" loop in line 5 of Algorithm 1 cannot expanded in parallel since there exists data dependent in the loop. In particular, the update of kernel 2 and kernel 3 at $g = t + 1$ needs the information obtained at $g = t$. When $M_G = 1$ or $G = M$, the number of check nodes in line 7 and 11 is only one since

$(g-1)M_G+1 \le j \le gM_G$ becomes $g \le j \le g$. In such a case, there is no parallelism can be utilized. When $M_G > 1$ or $G < M$, the "**For**" loop in kernel 2 and kernel 3 of Algorithm 1 can be expanded with a factor of M_G. Thus the degree of parallelism is M_G for kernel 2 and kernel 3 in the group-based layered scheduling. In addition, the hard decisions of kernel 4 in Algorithm 1 can also be implemented in parallel since the degree of parallelism for $\mathbf{H}\hat{\boldsymbol{x}}^T = \mathbf{0}$ in line 20 is M.

4 Simulation Results

In this section, we illustrate the frame error rate (FER) performances and the convergence rates of the proposed group-based layered scheduling for two LDPC codes C_1 and C_2, where C_1 is the Margulis (N = 2640, M = 1320) regular LDPC code with rate 0.5 [13] and C_2 is the IEEE 802.16e (N = 576, M = 288) irregular LDPC code with rate 0.5 [14]. All the simulations are performed over additive white Gaussian noise (AWGN) channels with binary phase shift key (BPSK) modulation. The simulation environment we used is Intel Core i5 CPU, 4.0 GB memory and Visual C++ 6.0 software development tool. The l_1 penalty function $f(x) = -\eta|x - 0.5|$ is used in our simulations. The parameters ρ and η are selected by simulations. For C_1, we choose $\rho = 3.0$ and $\eta = 0.8$. For C_2, $\rho = 4.0$ and $\eta = 0.8$ are used.

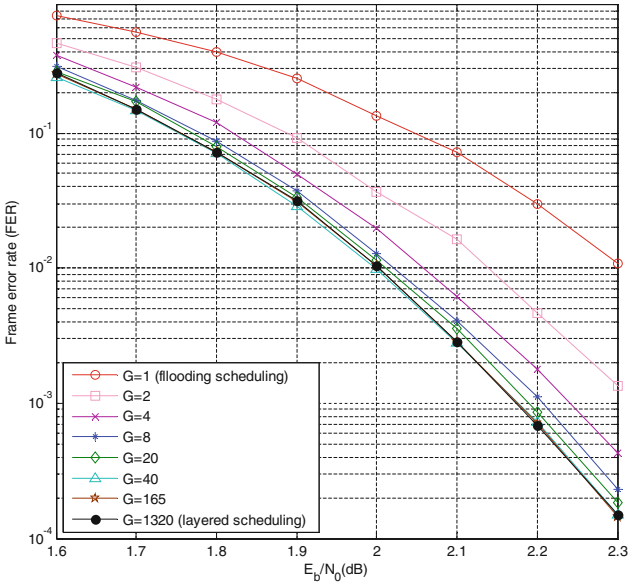

Fig. 1. FER performances for the ADMM penalized decoding with the proposed group-based layered scheduling for C_1 using different group sizes.

Figure 1 shows the FER performances of the ADMM penalized decoding with the proposed group-based layered scheduling for C_1 using $G = 1$ (flooding scheduling), 2, 4, 8, 20, 40, 165, and 1320 (layered scheduling). The maximum number of iterations is set to 20. It can be observed that the layered scheduling performs much better than the flooding scheduling and the larger the value of G, the better the FER performances. We can also see that there is no significant difference of the FER performances for $G = 40$, 165, and 1320. This means that a degree of parallelism with 33 can be obtained without performance degradation when compared to the layered scheduling with $G = 1320$.

Figure 2 depicts the average number of iterations for C_1 using the proposed method with different group sizes. It can be seen that the average number of iterations decreases as the group size increases. Moreover, as for the FER performances, the average number of iterations for $G = 40$, 165, and 1320 are almost the same. Therefore, there is almost no loss for both the FER performances and the average number of iterations when the degree of parallelism increased from 1 to 33.

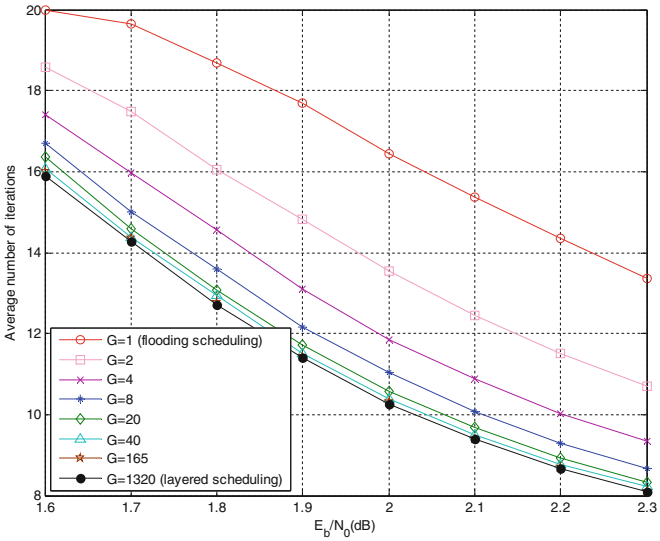

Fig. 2. Average number of iterations for the ADMM penalized decoding with the proposed group-based layered scheduling for C_1 using different group sizes, and at most 20 iterations.

Similarly, Figs. 3 and 4 show the FER performances and the average number of iterations, respectively, for the ADMM penalized decoding with the proposed group-based layered scheduling for C_2 using different group sizes. Similar observations can be obtained as for C_1.

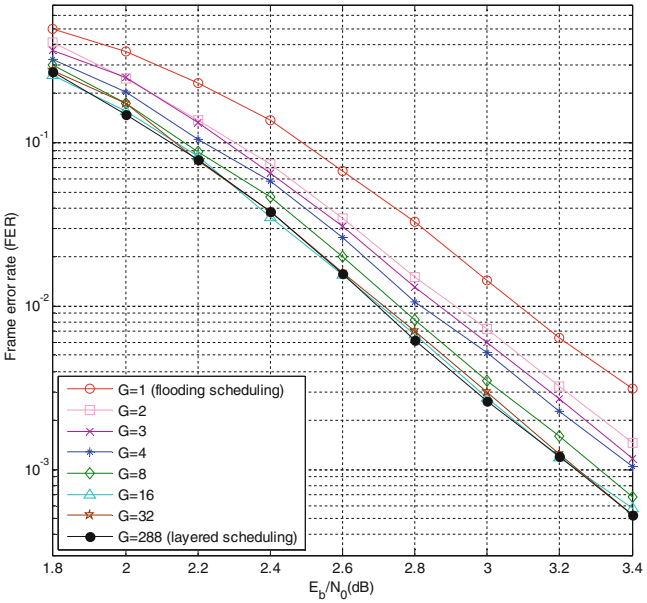

Fig. 3. FER performances for the ADMM penalized decoding with the proposed group-based layered scheduling for C_2 using different group sizes.

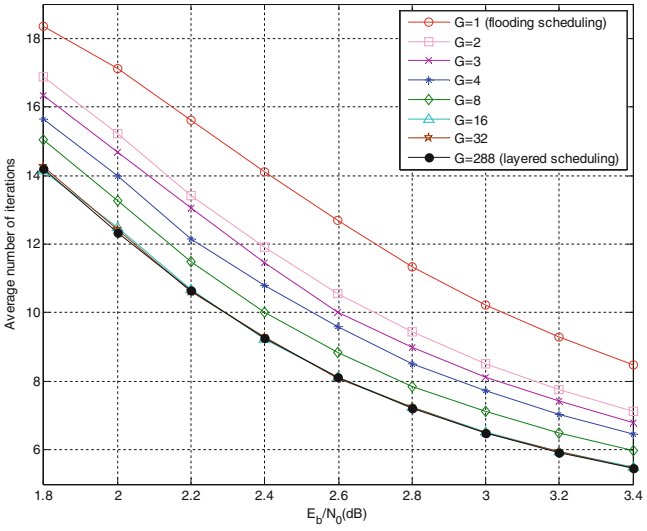

Fig. 4. Average number of iterations for the ADMM penalized decoding with the proposed group-based layered scheduling for C_2 using different group sizes, and at most 20 iterations.

5 Conclusion

In conclusion, a group-based layered scheduling for the ADMM decoding of LDPC codes is proposed. It shares the advantage of the flooding scheduling with fully parallelism and the advantage of the layered scheduling with fast convergence rates. Simulation results show that the proposed method can lift the degree of parallelism to some extent without performance degradation and convergence rate loss when compared to the layered scheduling which is totally serial.

References

1. Feldman, J., Wainwright, M.J., Karger, D.R.: Using linear programming to decode binary linear codes. IEEE Trans. Inf. Theory **51**, 954–972 (2005)
2. Barman, S., Liu, X., Draper, S.C., Recht, B.: Decomposition methods for large scale LP decoding. IEEE Trans. Inf. Theory **59**, 7870–7886 (2013)
3. Zhang, X., Siegel, P.H.: Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers. In: 2013 IEEE International Symposium on Information Theory, pp. 1501–1505. IEEE Press, New York (2013)
4. Liu, X., Draper, S.C.: The ADMM penalized decoder for LDPC codes. IEEE Trans. Inf. Theory **62**, 2966–2984 (2016)
5. Debbabi, I., Khouja, N., Tlili, F., Gal, B.L., Jego, C.: Multicore implementation of LDPC decoders based on ADMM algorithm. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 971–975. IEEE Press, New York (2016)
6. Zhang, G., Heusdens, R., Kleijn, W.B.: Large scale LP decoding with low complexity. IEEE Commun. Lett. **17**, 2152–2155 (2013)
7. Jiao, X., Mu, J., He, Y., Chen, C.: Efficient ADMM decoding of LDPC codes using look-up tables. IEEE Trans. Commun. **65**, 1425–1437 (2017)
8. Wei, H., Jiao, X., Mu, J.: Reduced-complexity linear programming decoding based on ADMM for LDPC codes. IEEE Commun. Lett. **19**, 909–912 (2015)
9. Debbabi, I., Gal, B.L., Khouja, N., Tlili, F., Jego, C.: Fast converging ADMM penalized algorithm for LDPC decoding. IEEE Commun. Lett. **20**, 644–647 (2016)
10. Jiao, X., Mu, J., Wei, H.: Reduced complexity node-wise scheduling of ADMM decoding for LDPC codes. IEEE Commun. Lett. **21**, 472–475 (2017)
11. Jiao, X., Wei, H., Mu, J., Chen, C.: Improved ADMM penalized decoder for irregular low-density parity-check codes. IEEE Commun. Lett. **19**, 913–916 (2015)
12. Wang, B., Mu, J., Jiao, X., Wang, Z.: Improved penalty functions of ADMM penalized decoder for LDPC codes. IEEE Commun. Lett. **21**, 234–237 (2017)
13. MacKay, D.J.C.: Encyclopedia of spare graph codes. http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
14. LDPC coding for OFDMA PHY: IEEE standard C802.16e-05/0066r3 (2005)