CRC Error Correction for Energy-Constrained Transmission

Evgeny Tsimbalo, Xenofon Fafoutis, Robert J Piechocki
Communication Systems & Networks, University of Bristol, UK
Email: {e.tsimbalo, xenofon.fafoutis, r.j.piechocki}@bristol.ac.uk

***Abstract*—In this paper, we investigate the application of iterative decoding algorithms to error correction of cyclic redundancy check (CRC) codes widely used in low-energy communication standards. We consider the case when traditional error correction codes are not available due to energy constraints at the transmitter. Using the CRC-24 code adopted by the Bluetooth Low Energy standard as an example, we show how two iterative techniques traditionally used for decoding of low-density parity check codes - Belief Propagation (BP) and the Alternating Direction Method of Multipliers (ADMM) - can be applied to the high-density parity check matrix of the code. The performance of both techniques is evaluated through simulation, and it is demonstrated that a gain of up to 1.7 dB in terms of the SNR per bit and a total reduction of the packet error rate by more than 70% can be achieved compared with the non-correction scenario, at no extra cost for the transmitter. We also compare the two techniques and use the standard syndrome look-up method as a benchmark. Both schemes enable the correction of multiple errors, with the ADMM-based decoder demonstrating better overall performance than BP.**

I. Introduction

Since their invention by W. Peterson in 1961 [1], CRC codes have been widely used in various communication systems to provide data integrity. More recently, CRC codes were employed by a number of low-energy communication standards such as Bluetooth Low Energy [2] and IEEE 802.15.4. The popularity of CRC codes is due to their simple hardware implementation and excellent error detection properties.

While CRC codes are traditionally used for error detection only, they have an inherent error correction potential due to redundancy they introduce to transmitted data. Even if a small number of bit errors were corrected in the receiver, it would reduce the number of retransmissions and therefore the energy wasted by the transmitter. This is especially important for cases when the energy of the transmitter is constrained, while some additional energy and computational power is available for the receiver.

Some error correction techniques for CRC codes have been proposed over the years. A simple look-up algorithm correcting all single errors was described by the inventors in [1]. More sophisticated look-up techniques correcting some of double-error codewords were also developed [3]. However, all these techniques aim at correcting a particular number of errors. To the best of knowledge of the authors, no unified approach has been proposed to correct an arbitrary number of errors, limited only by the error-correction capabilities of the code itself.

Many state-of-the-art error correction codes employ iterative decoding algorithms. One of those algorithms, known as belief propagation (BP), was originally developed for low density parity check (LDPC) codes [4], a special type of linear codes that has a sparse parity check matrix. In general, BP can be applied to any linear code, but its performance will be determined by the presence of short cycles on the code's Tanner graph. CRC codes, while being linear, usually have a dense parity check matrix with a high number of short cycles. In [5] and [6], the authors proposed a technique to eliminate short cycles from any linear code, demonstrating the results on Hamming and Reed-Solomon codes. The same technique can be applied to CRC codes, making them suitable to BP-based error correction.

As an alternative to BP, the decoding of a linear code can be viewed as a linear program (LP), the idea that was first introduced in [7]. This resulted in an algorithm based on the alternating direction method of multipliers (ADMM) initially proposed in [8] and applied to LDPC codes in [9]. Practical and computationally simple modifications were further developed in [10], [11] and [12]. While the ADMM-based algorithm has only been investigated in the context of LDPC codes, it can also be applied to correct errors for any linear code, such as a CRC one.

In this paper, we apply both decoding algorithms to the problem of error correction of a CRC code in a situation when no additional signal processing is available in the transmitter. To the best of our knowledge, no iterative decoding techniques have been investigated in the context of CRC codes before. Compared with the previous work on CRC error correction, we do not impose any limits on how many errors can be corrected.

The remainder of the paper is organized as follows. In Section II, a CRC-based communication system is presented, and a brief description of the CRC-24 code employed by the Bluetooth Low Energy standard is given. In Section III, the decoding strategies are described. First, it is shown how the parity check matrix of the CRC-24 code can be sparsified. BP and ADMM algorithms are then summarised. Simulation results are presented in Section IV, followed by Section V that concludes the paper.

II. System description

Fig. 1 depicts a general CRC-based communication system. Prior to transmission, each packet is processed by a systematic CRC encoder which adds redundant bits to the packet. For

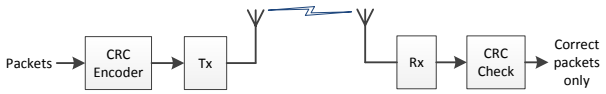

Fig. 1. CRC-based communication system.

the BLE CRC-24 code, the encoder adds 24 redundant bits to 312 data bits such that the length of the encoded packet is 336 bits[1]. The generator polynomial of the CRC-24 has the following form [2]:

$$g(x) = x^{24} + x^{10} + x^9 + x^6 + x^4 + x^3 + x + 1. \quad (1)$$

At the receiver, CRC check is performed over each group of 336 demodulated bits by calculating a syndrome. This can be expressed in terms of the parity check matrix $\mathbf{H}$ of the code that can be obtained from its generator polynomial as $\mathbf{s} = \mathbf{Hr}$, where $\mathbf{s}$ is the syndrome and $\mathbf{r}$ is the received vector of demodulated bits. If $\mathbf{s}$ is a zero vector, the corresponding packet is correct. If at least one element of $\mathbf{s}$ is non-zero, the packet contains errors and is discarded.

The system between the output of the CRC encoder and the input of the CRC check block in Fig. 1 can be viewed as a binary symmetric channel (BSC). The probability of bit error, or the crossover probability, of such a channel can be computed as

$$p = Q\left(\sqrt{2E_b/N_0}\right), \quad (2)$$

where E_b/N_0 is the equivalent signal to noise ratio (SNR) calculated per bit and Q is the Q-function defined as

$$Q(x) = \Pr\left[\mathcal{N}(0,1) > x\right] = \frac{1}{2\pi}\int_x^\infty e^{-t^2/2}dt, . \quad (3)$$

where $\mathcal{N}(0,1)$ is a Gaussian random variable with a zero mean and the variance equal to one.

The parity check matrix $\mathbf{H}$ and the equivalent channel model described by (2) will now be used to introduce and simulate decoding algorithms.

III. Decoding

It is known that the performance of iterative decoding techniques such as BP applied to a general linear code is affected by the presence of short cycles on the Tanner graph of the code [5]. In particular, cycles of length four, or four-cycles, should be avoided. As shown in [6], the total number of four-cycles for an $m \times n$ parity check matrix $\mathbf{H}$ can be calculated as

$$\sum_{i=1}^{m}\sum_{j=i+1}^{m}\binom{\left(\mathbf{H}\mathbf{H}^T\right)_{ij}}{2}. \quad (4)$$

Based on (4), the number of four-cycles of the BLE CRC-24 code is 813816. Therefore, direct application of BP would result in poor decoding performance.

The maximum cycle strategy (MCS) algorithm to sparsify the parity check matrix of a general linear code by removing all four-length cycles was presented in [5]. In this algorithm, auxiliary variable and check nodes are added to the Tanner graph for every four-cycle such that all parity check equations remain intact. Compared with the original approach given in [6], the MCS algorithm significantly reduces the number of auxiliary nodes. When applied to the parity check matrix of the CRC-24 code, the MCS algorithm results in only 276 additional variable and check nodes, making the size of the sparsified matrix 300×612. This matrix does not have any cycles of length four and therefore iterative decoding techniques can now be applied.

A. Belief Propagation Decoding

BP decoding in the form of the sum-product algorithm is a standard approach to decode LDPC codes. We will use the log-likelihood version of the algorithm presented, for example, in [13]. The input of the algorithm are the log-likelihood ratios (LLRs) of the bits at the input of the decoder. For a BSC with the crossover probability p given in (2), the LLR corresponding to the i-th received bit r_i can be calculated as

$$\gamma_i = (2r_i - 1)\ln\left(\frac{1-p}{p}\right). \quad (5)$$

We note that (5) assumes hard inputs at the decoder, which is relevant to a practical scenario where CRC error correction is applied outside the physical layer of the receiver.

For auxiliary variable nodes added after sparsifying the parity check matrix, the LLRs are set to zeros [6]. After each iteration, hard decisions are made on the vector of updated LLRs and the syndrome is calculated using the original, non-sparse parity check matrix neglecting the bits corresponding to the auxiliary variables. If the syndrome is a zero vector, the original packet bits are extracted and the algorithm stops. The algorithm also stops if a maximum number of iterations is achieved.

B. ADMM

We begin by showing how a general decoding problem can be converted to a convex optimisation problem. Let $\mathbf{x} \in \{0,1\}^N$ be a codeword transmitted through a memoryless BSC with the crossover probability defined by (2). Let $\mathbf{r} \in \{0,1\}^N$ be the received vector. We use $\mathbf{H}$ to denote the parity check matrix of the code, which in general can be either the original matrix or its sparsified version. Let c_i, $i = 1,..,N$ be the degree of the i-th variable node and d_j, $j = 1,...,M$ be the degree of the j-th check node. Maximum likelihood (ML) decoding finds a codeword that maximises the probability $p(\mathbf{r}|\mathbf{x})$ that $\mathbf{r}$ was received given $\mathbf{x}$ was transmitted. Assuming a memoryless channel, in the log domain this probability can be expressed as

$$\log p(\mathbf{r}|\mathbf{x}) = \log\prod_{i=1}^{N} p(r_i|x_i) = \sum_{i=1}^{N}\log p(r_i|x_i). \quad (6)$$

By noting that

$$\log p(r_i|x_i = 1) = \gamma_i + \log p(r_i|x_i = 0),$$

[1]We assume the maximum packet length defined by the BLE standard.

where γ_i is the LLR for the i-th bit calculated by (5), the ML problem can be restated as

$$\arg\max_{\mathbf{x}:\mathbf{Hx}=\mathbf{0}} \left(\sum_{i=1}^{N} \left[\gamma_i x_i + \log p(r_i | x_i = 0) \right] \right) = \arg\max_{\mathbf{x}:\mathbf{Hx}=\mathbf{0}} \left(\sum_{i=1}^{N} \gamma_i x_i \right). \quad (7)$$

Finally, by introducing a negative LLR $\bar{\gamma}_i \triangleq -\gamma_i$, ML decoding can be converted to a minimisation problem

$$\text{minimise } \bar{\gamma}^T \mathbf{x}, \text{ subject to } \mathbf{Hx} = \mathbf{0}. \quad (8)$$

In [7], it was shown how the minimisation problem (8) can be formulated as a linear program (LP) over the convex hull of all codewords. Using an LDPC code as an example, it was demonstrated that LP decoding performs similarly to BP decoding. In addition, it was shown that LP decoding is guaranteed to produce an ML codeword. However, the computational complexity of the original LP decoder is much higher than that of BP, and as a result in [9] the authors introduced a faster algorithm based on the alternating direction method of multipliers (ADMM) which was originally developed in [8]. With this modification, the ADMM for ML decoding can be formulated as follows:

$$\begin{aligned} &\text{minimise } \bar{\gamma}^T \mathbf{x} \\ &\text{subject to } \forall j,\ \mathbf{P}_j \mathbf{x} = \mathbf{z}_j, \\ &\qquad \mathbf{z}_j \in \mathbb{PP}_{d_j}. \end{aligned} \quad (9)$$

Here, $\mathbf{P}_j$ is the operation of selecting those bits of $\mathbf{x}$ that participate in the j-th check; $\mathbf{z}_j$ is a replica vector for the j-th check; $\mathbb{PP}_{d_j}$ is the parity polytope of dimension d_j [9]. Denoting $y(\mathbf{x})$ the objective function to be minimised, the augmented Lagrangian in the unscaled form [8] for (9) can be written as

$$\mathcal{L}_\mu(\mathbf{x}, \mathbf{z}, \lambda) = y(\mathbf{x}) + \sum_j \lambda_j (\mathbf{P}_j \mathbf{x} - \mathbf{z}_j) + \frac{\mu}{2} \sum_j \|\mathbf{P}_j \mathbf{x} - \mathbf{z}_j\|_2^2, \quad (10)$$

where $\mu > 0$ is the augmented Lagrangian parameter and λ_j is an auxiliary variable. The solution to (9) is an iterative algorithm with the k-th iteration being

$$\mathbf{x}^{[k+1]} = \arg\min_{\mathbf{x}} \mathcal{L}_\mu(\mathbf{x}, \mathbf{z}^{[k]}, \lambda^{[k]}), \quad (11)$$

$$\mathbf{z}^{[k+1]} = \arg\min_{\mathbf{z}} \mathcal{L}_\mu(\mathbf{x}^{[k+1]}, \mathbf{z}, \lambda^{[k]}), \quad (12)$$

$$\lambda^{[k+1]} = \lambda_j^{[k+1]} + \mu(\mathbf{P}_j \mathbf{x}^{[k+1]} - \mathbf{z}^{[k+1]}). \quad (13)$$

In (11) and (12), the two primal variables - $\mathbf{x}$ and $\mathbf{z}$ - are updated in an alternating fashion. Therefore, the ADMM can be viewed as a message passing algorithm on a graph, with x_i, $i = 1, ..., N$ and z_j, $j = 1, ..., M$ being variable and check nodes respectively.

To improve the performance of the algorithm at low SNRs and to avoid error floors at high SNRs, a penalty function was introduced in [10] such that the modified objective function $y(\mathbf{x})$ can be rewritten as

$$\bar{\gamma}^T \mathbf{x} + \sum_i f(x_i), \quad (14)$$

where $f : [0, 1] \mapsto \mathbb{R} \cup \{\pm\infty\}$ is a penalty function. Two penalty functions are proposed in [10]:

$$f_1(x) = -\alpha_1 \|x - 0.5\|_1, \quad (15)$$

$$f_2(x) = -\alpha_2 \|x - 0.5\|_2^2. \quad (16)$$

They are called l_1 and l_2 penalty functions respectively, with $\alpha_1, \alpha_2 > 0$ being the penalty coefficients. As shown in [10], the l_2 penalty function provides better PER performance than the l_1 penalty function.

To finalise the algorithm, we also adopt the over-relaxation technique advocated in [8] to improve decoding convergence. Denoting $\rho > 1$ the over-relaxation parameter, the ADMM-PD algorithm with with the l_2 penalty function is summarised in Algorithm 1.

In the update for $\mathbf{z}_j$ in Algorithm 1, $\Pi_{\mathbb{PP}_{d_j}}(\cdot)$ is the projection onto the parity polytope $\mathbb{PP}_{d_j}$ [9]. In our implementation, we use the original projection algorithm proposed by [9]. More computationally effective techniques were derived in [11] and [12].

Algorithm 1 ADMM-PD with over-relaxation.

Input: Vector of negative LLRs $\bar{\gamma}$ and parity check matrix $\mathbf{H}$.
Output: Decoded vector $\mathbf{x}$.

1: **Initialisation**: Construct the selection matrix $\mathbf{P}_j$ for each check node j based on $\mathbf{H}$. Initialise λ_j as the all zeros vector and $\mathbf{z}_j$ as the all 0.5 vector.

2: **Variable node update**: For each variable node i, do:
Calculate $\bar{\mathbf{z}}_j = \mathbf{P}_j^T \mathbf{z}_j^{[k]}$, $\bar{\lambda}_j = \mathbf{P}_j^T \lambda_j^{[k]}$, $\forall j$.
Calculate $t_i = \sum_j \left(\bar{\mathbf{z}}_j - \frac{\bar{\lambda}_j}{\mu} \right) - \frac{\bar{\gamma}_i}{\mu}$.
Update

$$x_i^{[k+1]} \leftarrow \frac{1}{c_i - 2\alpha_2} (t_i - \frac{\alpha_2}{\mu}).$$

Project $x_i^{[k+1]}$ onto $[0, 1]$: $x_i^{[k+1]} \leftarrow \Pi_{[0,1]} x_i^{[k+1]}$.

3: **Check node update**: For each check node j, do:
Calculate

$$\mathbf{v}_j^{[k+1]} \leftarrow \rho \mathbf{P}_j \mathbf{x}^{[k+1]} + (1 - \rho) \mathbf{z}_j^{[k]} + \frac{\lambda_j^{[k]}}{\mu}.$$

Update $\mathbf{z}_j^{[k+1]} \leftarrow \Pi_{\mathbb{PP}_{d_j}} (\mathbf{v}_j^{[k+1]})$.
Update

$$\lambda_j^{[k+1]} \leftarrow \lambda_j^{[k]} + \mu \left[\rho \mathbf{P}_j \mathbf{x}^{[k+1]} + (1 - \rho) \mathbf{z}_j^{[k]} - \mathbf{z}_j^{[k+1]} \right].$$

4: Make a tentative hard decision on $\mathbf{x}^{[k+1]}$: if $x_i^{[k+1]} \geq 0.5$, $\hat{x}_i = 1$; otherwise $\hat{x}_i = 0$.

5: If $\mathbf{H}\hat{\mathbf{x}} = \mathbf{0}$, then **return** $\mathbf{x} = \hat{\mathbf{x}}$. Otherwise, if $k + 1$ is smaller then the maximum number of iterations T_{max}, do $k \leftarrow k + 1$ and loop to **Variable node update**. Otherwise, declare decoding failure and **Stop**.

It can be observed that the ADMM-PD algorithm has four parameters: the augmented Lagrangian parameter μ, the penalty coefficient α (for a given penalty function), the over-relaxation parameter ρ and the maximum number of iterations T_{max}. Investigation into the selection of these parameters for some LDPC codes and the AWGN channel was carried out in [10], resulting in the following recommendations:

- $\mu \in [3, 5]$ provide optimal results for both the l_1 and l_2 penalty functions;
- $\alpha_1 = 0.6$ and $\alpha_2 = 0.8$ are good choices for the l_1 and l_2 penalty functions respectively;
- $\rho = 1.9$ is optimal for decoding performance and convergence;
- Increasing T_{max} improves the probability of error correction, at the expense of decoding speed; up to 1000 iterations are usually considered.

As noted in [10], however, the ADMM-PD algorithm is rather sensitive to parameters settings, and extra care should be taken when selecting the parameters for a particular code and channel.

IV. Simulation Results

In this section, we demonstrate the performance of BP and ADMM decoding applied to the CRC-24 code operating in the BSC based on Monte Carlo simulation. The total number of encoded bits is 336, corresponding to the maximum packet size defined by BLE and the worst case scenario from the error correction point of view. BP decoding is implemented as the sum-product algorithm in the log-likelihood domain [13]. The ADMM-based decoder is implemented as the ADMM-PD algorithm with the l_2 penalty function according to Algorithm 1. The ADMM-PD parameters were selected as described in the previous section, with the exception of the penalty coefficient and the over-relaxation parameter: we noticed that $\alpha_2 = 1$ and $\rho = 1.8$ provide better performance for the CRC-24 code than the values recommended in [10].

Fig. 2 illustrates the PER performance of the two decoders as a function of the equivalent SNR per bit E_b/N_0 of the BSC (2), for two values of the maximum number of decoding iterations T_{max}, 200 and 1000. The PER without error correction is plotted as a reference. Both ADMM-based and BP decoders iterate until either a correct codeword is found or the maximum number of iterations is achieved. It can be observed that when $T_{max} = 200$, the BP decoder outperforms the ADMM in the high SNR region by up to 0.5 dB. At the same time, when T_{max} is increased to 1000, the ADMM-based decoder provides better performance. To illustrate this further, Table I lists the probability of successful correction, or success rate, for different numbers of errors per packet. For the ADMM decoder, the success rate rises significantly when T_{max} is increased to 1000, while for the BP decoder the benefit is almost negligible. Based on Table I, it is clear that the ADMM-based decoder is superior in terms of single- and double-error packets. Both decoders fail to correct more than four errors per packet due to the high code rate used in the simulation. All in all, it can be seen that error correction enables up to 1.7 dB benefit in terms of E_b/N_0 and 72% total reduction in the PER.

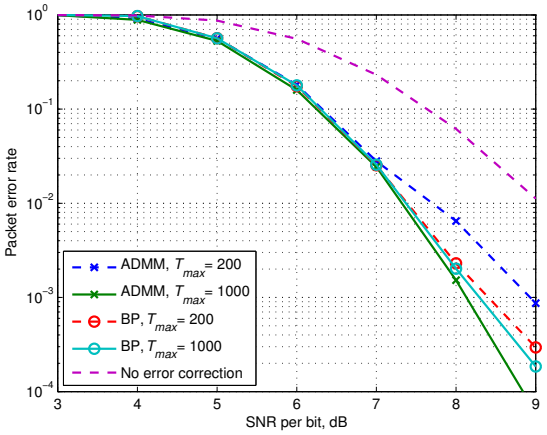

Fig. 2. PER performance of ADMM and BP decoding of the CRC-24 code in the BSC.

Table I
Correction success rate for different numbers of errors.

# of bit errors	ADMM 200	ADMM 1000	BP 200	BP 1000
1	98.2%	100%	99.3%	99.5%
2	11.2%	24.2%	13.1%	13.2%
3	0%	0.2%	0.6%	0.6%
4	0%	0%	0.1%	0.1%
Total:	68.2%	72.3%	70.7%	70.7%

Fig. 3 shows the success rate of each decoder for each SNR point. It can be observed that in the low and mid SNR regions, the ADMM-based decoder can correct more packets, even when $T_{max} = 200$. The BP decoder performs especially poorly when the SNR is low. It can also be seen that at high SNRs, the ADMM-based decoder benefits significantly from the increase in T_{max}, while the gain is only marginal for the BP decoder at all SNR points. Both Fig. 2 and Fig. 3 suggest that although the ADMM-based decoder converges slower, in the end its probability of successful correction is higher than that of the BP decoder. This fully agrees with the theoretical results presented in [7] which claim that the ADMM is guaranteed to produce the optimum (i.e. ML) codeword.

Finally, Fig. 4 shows how the iterative decoders perform compared with the look-up method that can correct all single-error packets. The number of corrected packets relative to the number of single-error packets for each SNR point is shown. In line with the previous plot, the BP decoder performs very poorly in the low SNR region, with the look-up method being able to correct up to 10 times more packets. Quite the opposite, the ADMM-based decoder can correct up to two times as many packets as the look-up method when the SNR is small. At high SNRs, the performance of all decoders is the same, since single-error packets prevail in that region. This suggest a hybrid error correction scheme, where the ADMM decoder is used at low and medium SNRs and the simple look-up method to correct all single-error packets at high SNRs.

It should be noted that the simulation results presented in this paper for the ADMM-based decoder were obtained for one

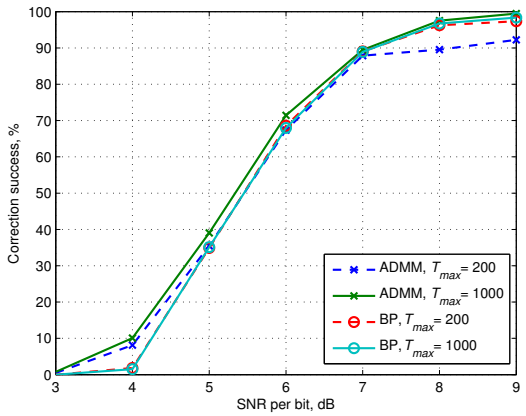

Fig. 3. Packet correction success rate for ADMM and BP decoding of the CRC-24 code in the BSC.

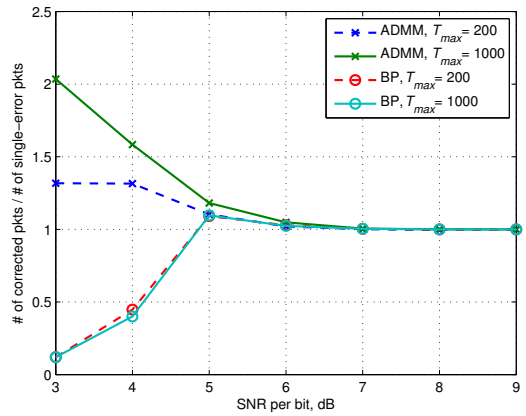

Fig. 4. Number of corrected packets for ADMM and BP decoders relative to the number of single-error packets.

set of the ADMM-PD parameters and one particular penalty function. This suggests that there is some potential for further improvement of the ADMM-based error correction, especially in terms of convergence and the probability of correction of multiple-error packets.

Comparison between the ADMM and BP decoders in terms of their computational complexity is not presented in this paper. The interested reader is referred to [11], where the authors showed that the ADMM algorithm is faster than BP for the same PER performance.

V. Conclusions

In this paper, we apply iterative decoding methods to CRC codes in a situation when no additional encoding can be done in the transmitter, exploiting the redundancy provided by the codes themselves. The CRC-24 code employed by the Bluetooth Low Energy standard is used as an example. Two different iterative strategies are investigated - the classical belief propagation, and a relatively new algorithm based on convex optimisation in the form of the alternating direction method of multipliers. Both strategies are traditionally applied to low-density parity check codes, so the problem of sparsifying the parity check matrix of the CRC-24 was first investigated, with the emphasis being put on the elimination of cycles of length four on the code's Tanner graph. It was shown how all such cycles can be eliminated and an equivalent representation of the parity check matrix of the CRC-24 code can be obtained, without a significant increase in its size. Simulations showed that the ADMM-based decoder has an advantage over the BP decoder both in terms of the SNR per bit and the number of packets corrected. This advantage is particularly clear in the low SNR region, where the probability of successful correction of the ADMM-based method is several times higher than that of the BP decoder. Both methods were also compared with the syndrome look-up algorithm that is able to correct all single-error packets. Due to its ability to correct multiple errors, the ADMM-based decoder showed a better correction rate than the look-up method in the low SNR region. Overall, CRC error correction enabled up to 1.7 dB benefit in terms of the SNR per bit and 72% total reduction in the PER, with no extra cost for the transmitter. This can significantly reduce the number of retransmissions, therefore increasing the energy efficiency of the transmitter. One of the potential applications of CRC error correction is energy-efficient communication in wireless sensor networks, where reliability can be improved by introducing additional processing at the receiver.

VI. Acknowledgments

This work was performed under the SPHERE IRC funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

[1] W. Peterson and D. Brown, "Cyclic Codes for Error Detection," *Proc. IRE*, vol. 49, no. 1, pp. 228–235, Jan. 1961.

[2] *Specification of the Bluetooth system. Core Version 4.1*, Bluetooth SIG, 2013. [Online]. Available: http://www.bluetooth.com

[3] S. Babaie, A. K. Zadeh, S. H. Es-hagi, and N. J. Navimipour, "Double Bits Error Correction Using CRC Method," in *2009 Fifth Int. Conf. Semant. Knowl. Grid*. IEEE, 2009, pp. 254–257.

[4] R. G. Gallager, "Low-Density Parity-Check Codes," p. 112, 1963.

[5] V. Kumar and O. Milenkovic, "On graphical representations of algebraic codes suitable for iterative decoding," *IEEE Commun. Lett.*, vol. 9, no. 8, pp. 729–731, 2005.

[6] S. Sankaranarayanan and B. Vasic, "Iterative Decoding of Linear Block Codes: A Parity-Check Orthogonalization Approach," *IEEE Trans. Inf. Theory*, vol. 51, no. 9, pp. 3347–3353, Sep. 2005.

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

[8] S. Boyd, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[9] S. Barman, X. Liu, S. Draper, and B. Recht, "Decomposition methods for large scale LP decoding," *2011 49th Annu. Allert. Conf. Commun. Control. Comput.*, vol. 59, no. 12, pp. 253–260, 2011.

[10] X. Liu, S. C. Draper, and B. Recht, "Suppressing pseudocodewords by penalizing the objective of LP decoding," in *2012 IEEE Inf. Theory Work.* IEEE, Sep. 2012, pp. 367–371.

[11] X. Zhang and P. H. Siegel, "Efficient iterative LP decoding of LDPC codes with alternating direction method of multipliers," in *2013 IEEE Int. Symp. Inf. Theory*. IEEE, Jul. 2013, pp. 1501–1505.

[12] G. Zhang, R. Heusdens, and W. B. Kleijn, "Large Scale LP Decoding with Low Complexity," *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2152–2155, Nov. 2013.

[13] T. K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley-Interscience, 2005.