ADMM and Reproducing Sum-Product Decoding Algorithm Applied to QC-MDPC Code-Based McEliece Cryptosystems

Kohtaro Watanabe, Motonari Ohtsuka, and Yuta Tsukie

***Abstract*— QC-MDPC (quasi cyclic moderate density parity check) code-based McEliece cryptosystems are considered to be one of the candidates for post-quantum cryptography. Decreasing DFR (decoding failure rate) is one of important factor for their security, since recent attacks to these cryptosystems effectively use DFR information. In this paper, we pursue the possibility of optimization-base decoding, concretely we examine ADMM (alternating direction method of multipliers) based method, a recent developing method in optimization theory. Further, RSPA (reproducing sum-product algorithm), which efficiently reuse outputs of SPA (sum-product algorithm) is proposed for further reduction of DFR. By numerical simulations, we show that the proposing scheme shows considerable decrement in DFR compared to the conventional decoding methods such as BF (bit-flipping algorithm) or SPA.**

***Index Terms*— QC-MDPC code-based cryptosystem, ADMM method, reproducing sum-product algorithm, McEliece cryptosystem.**

I. Introduction

THE safety of current cryptosystem (for example, RSA, ECDH) are based on the computation difficulty such as integer factoring or discrete logarithmic problem. These crypto schemes are said to be no longer secure under the presence of quantum computers, which are able to solve these problems in polynomial time [25]. Although current number of qubits are relatively small, algorithms resistant for quantum computers equipped with practical qubits are desirable. Following proposals; lattice-based, code-based, hash-based and multivariate, isogeny-base etc. are considered to be the framework of cryptosystems for post quantum generation and these are continuing to be reviewed in NIST post-quantum cryptography standardization [30].

The present paper is concerned with the reduction of DFR (decoding failure rate) in code-based cryptosystem. It is known that there is an attack (GJS attack) [11] that effectively uses decoding failure statistics to retrieve the parity check matrix of the code (which is a secret key). In addition, to achieve λ bits of security under IND-CCA (Indistinguishability under Chosen Ciphertext Attack) circumstance, the condition DFR$\leq 2^{-\lambda}$ should be satisfied; see p. 9 of BIKE specification document [6]. Thus, reducing the DFR is considered to be one of important factors for the code-based cryptosystems. We note the efforts to reduce DFR are seen in recent studies [7], [8], [9], [13], [17], [20], [24].

The security of code-based schemes is based on the hardness of decoding linear random codes (indeed the problem is known to be NP-complete [5], [14]). The first code-based scheme is the McEliece cryptosystem which use Goppa codes. The original McEliece cryptosystem has been withstanding evaluation more than thirty years and still regarded secure. A problem of original McEliece cryptosystem is its large public key size. For the reduction of public key size, various improvements were proposed; see [6], [16], [26], and [29]. It should be noted that the usage of LDPC codes directly for this purpose is known to be dangerous, since parity check matrix of LDPC (low-density parity check) codes might be recovered by decoding low Hamming weight codewords, for example using Stern's method [27], as dual codes of original codes.

On the other hand, QC-MDPC (quasi cyclic moderate density parity check) code is currently regarded as safe and public key size is small, and for such reasons, this scheme is remaining as alternatives of NIST PQC as Public-key Encryption and Key-establishment Algorithms [30]. Standard decoding methods for QC-MDPC codes are variants of BF (bit-flipping) algorithm. Although BF variants algorithms show good performance in execution time and in DFR, these algorithms are not optimization based, so there might be still room for improvement of DFR. In this paper, we show DFR of QC-MDPC codes can be considerably improved compared to BF or SPA (sum-product algorithm) by applying ADMM (alternating direction method of multipliers) based decoding, which is a recently developing subject in optimization theory. The ADMM decoding is a method based on IP (integer programming), which achieve MLD (maximum likelihood decoding); see Feldman, Wainwright and Karger [10]. Although IP decoding presents high precision decoding, it costs a considerable amount of execution time and hence the adoption seems to be limited to relatively short code-length problems (about a few hundred bits or so). While the ADMM decoding is reported to be applicable to more than thousands of bits of code-length;

Manuscript received 19 December 2022; revised 18 May 2023; accepted 16 August 2023. Date of publication 28 August 2023; date of current version 16 February 2024. *(Corresponding author: Kohtaro Watanabe.)*

The authors are with the Department of Computer Science, National Defense Academy, Yokosuka 239-8686, Japan (e-mail: ktrwata@gmail.com).

Communicated by A.-L. Horlemann-Trautmann, Associate Editor for Coding and Decoding.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIT.2023.3309035.

Digital Object Identifier 10.1109/TIT.2023.3309035

see [4] and [15]. In this paper, we apply the algorithm of Barman et al. [4] and Liu and Draper [15] as the ADMM decoding method and show even in the case of middle-density parity checks and middle code-length (in cases code-length are 9602, 10779 etc.), it works well.

Here, we have to note that the original projection algorithm (Algorithm 2 of [4]) seems to be including some error (actually, Algorithm 2 of [4] sometimes shows mismatching values compared to the values computed by MIP solver Gurobi optimizer 9.5 [12]); see Figure 1 and 2 in Section III. We present the corrected projection algorithm to the code-polytope in Algorithm 3 and examine its validity in Section V.

As for the reduction in DFR, by numerical examination in Section IV, we observe in the case of $(n, k) = (9602, 4801)$ code, the correctable number of errors increases from $t = 87$ (BF algorithm) to $t = 103$ under almost the same DFR (less than 10^{-7}). Similarly, in the case of $(n, k) = (10779, 3593)$, correctable number of number of errors increases from $t = 55$ (BF algorithm) to $t = 66$ under almost the same DFR (less than 10^{-7}). In other words, proposing scheme attains far less DFR for fixed error number t compared to BF or SPA and hence prevents the collection of DFR information by adversaries. It is observed that in numerical examinations, ADMM method decreases its DFR rapidly if t is under the threshold value t_0 (for example, for the case $(n, k) = (9602, 4801)$, $t_0 = 103$ and for the case $(n, k) = (10779, 3593)$, $t_0 = 66$). Thus, we can expect further decrement of DFR if with some preprocessing, error number t could be lowered than t_0. For that purpose, we introduce the method RSPA (reproducing sum-product algorithm), which efficiently reuse decoding failure of SPA as a preprocessing for ADMM. We also compare the DFR of RSPA with Backflip$^+$ [8] and BGF decoder [6], [9] for cases $(n, k) = (9602, 4801)$ and $(n, k) = (19714, 9857)$. We note these are 80 bits and 128 bits security parameters achieve IND-CPA security; see Table II of [16].

Finally, we examine the DFR behavior of "weak keys" (actually Type I weak keys of [23]). It may come as a bit surprise that experiments indicate that Type I weak keys are not "weak keys" for ADMM based decoding. Rather, they seem to be "strong keys".

II. QC-MDPC McEliece Cryptosystems

A. McEliece Cryptosystems

We briefly review McEliece cryptosystems. Throughout this paper, a set of codewords is denoted by $\mathcal{C}$. The variables n, k, t denote code-length, information bits length and correctable error number, respectively. We abbreviate $m = n - k$. Also, $k \times n$ binary matrix $\boldsymbol{G}$, $m \times n$ binary matrix $\boldsymbol{H}$, $n \times n$ matrix $\boldsymbol{P}$ and $k \times k$ binary matrix $\boldsymbol{S}$ represent generator matrix, parity-check matrix, random permutation matrix and scramble matrix respectively. Put

$$\boldsymbol{G'} = \boldsymbol{SGP}. \tag{1}$$

Then $(\boldsymbol{G'}, t)$ is a public key and $(\boldsymbol{G}, \boldsymbol{S}, \boldsymbol{P})$ is a private key of Bob. For a message $\boldsymbol{m}$, using public key, Alice encrypts as

$$\boldsymbol{c} = \boldsymbol{mG'} + \boldsymbol{e}, \tag{2}$$

where $\boldsymbol{e}$ is a correctable error vector whose Hamming weight satisfies $w(\boldsymbol{e}) = t$. Decoding process of Bob is as follows:

(i) Multiply $\boldsymbol{P}^{-1}$ to both sides of (2), that is
$$\boldsymbol{cP^{-1}} = \boldsymbol{mG'P^{-1}} + \boldsymbol{eP^{-1}} = \boldsymbol{mSG} + \boldsymbol{eP^{-1}}.$$
(ii) Since $w(\boldsymbol{eP^{-1}}) = w(\boldsymbol{e}) = t$, by applying certain decoding scheme, he obtains $\boldsymbol{c'} = \boldsymbol{mSG}$.
(iii) Multiplying $\boldsymbol{G^{-1}S^{-1}}$ to $\boldsymbol{c'}$, Bob retrieves the sent message $\boldsymbol{m}$.

B. QC-MDPC McEliece Cryptosystems With Code-Rate $R = (n_0 - 1)/n_0$

QC-MDPC codes are linear block codes whose binary parity check matrices have moderate density of "one' and are introduced to reduce public key size of original McEliece cryptosystems; see Misoczki et al. [16]. A parity check matrix of QC-MDPC McEliece cryptosystems with code-rate $R = (n_0 - 1)/n_0$ assumes the following form; see for example Baldi [2, Section 3.6],

$$\boldsymbol{H} = [\boldsymbol{H_0}, \boldsymbol{H_1}, \ldots, \boldsymbol{H_{n_0-1}}], \tag{3}$$

where each $\boldsymbol{H_i}$, $(0 \leq i \leq n_0 - 1)$ is a binary $p \times p$ circulant matrix of column weight d_c. Hence it holds that $n = n_0 p$ and $k = (n_0 - 1)p$ (so $R = (n_0 - 1)/n_0$). Without loss of generality, we can assume $\boldsymbol{H}_{n_0-1}$ is non-singular, then the generator matrix of the code is expressed as (see also [2, Section 3.6]),

$$\boldsymbol{G} = \left[\begin{array}{c|c} \boldsymbol{I_k} & \begin{array}{c} \left(\boldsymbol{H_{n_0-1}^{-1}} \cdot \boldsymbol{H_0}\right)^T \\ \vdots \\ \left(\boldsymbol{H_{n_0-1}^{-1}} \cdot \boldsymbol{H_{n_0-2}}\right)^T \end{array} \end{array}\right], \tag{4}$$

where $\boldsymbol{I_k}$ is the $k \times k$ identity matrix. Since $\left(\boldsymbol{H_{n_0-1}^{-1}} \cdot \boldsymbol{H_i}\right)^T$, $(0 \leq i \leq n_0 - 1)$ are circulant, public key will be only first rows of these matrices, and as a result, reducing the size of the public key.

III. Decoding Methods

A. Bit-Flipping Decoding Algorithm

The fundamental bit-flipping decoding algorithm is described as follows:

Algorithm 1 Bit-Flipping Decoding Algorithm

```
1: $\boldsymbol{s} \leftarrow \boldsymbol{Hc}^T$.
2: while $\boldsymbol{s} \neq 0$ do
3:     $i \leftarrow \arg\min_{0 \leq i \leq n-1} w(\boldsymbol{s} + \boldsymbol{h}_i)$.
4:     if $w(\boldsymbol{s} + \boldsymbol{h}_i) < w(\boldsymbol{s})$ then
5:         $c_i \leftarrow 1 - c_i$.
6:         $\boldsymbol{s} \leftarrow \boldsymbol{s} + \boldsymbol{h}_i$.
7:     else
8:         return decoding failure.
9:     end if
10: end while
11: return $\boldsymbol{c}$.
```

As shown in the numerical experiments of next section, BF decoding algorithm shows lower DFR than SPA, especially in cases $w(\boldsymbol{e})$ are relatively small; see Figure 7 and 8.

B. Notations

Next, we introduce ADMM algorithm according to [4]. Although, an algorithm developed in [4] assumes LDPC codes, we show that the algorithm is also effective for decoding MDPC codes. We define some necessary notations: $\mathcal{X}$ and $\mathcal{Y}$ denote: $\mathcal{X} = \{0,1\}$ and $\mathcal{Y} = \{0,1\}$, since we assume BSC (binary symmetric channel). Thus, sending messages belong to k-dimensional subspace $\mathcal{C}$ of $\mathcal{X}^n$ and corresponding received messages belong to $\mathcal{Y}^n$. The neighborhood of a each "check" i is denoted by $N_c(i)$ and the neighborhood of a each "vertex" j is denoted by $N_v(j)$, namely:

$$N_c(i) = \{j \mid H_{i,j} = 1\} \text{ and } N_v(j) = \{i \mid H_{i,j} = 1\},$$

where $H_{i,j}$ is a (i,j) element of parity check matrix $\boldsymbol{H}$. In the case, H is assumed to be as (3), since each row and column of $\boldsymbol{H}_k$, $(0 \le k \le n_0 - 1)$ has d_c non-zero element, it holds that $|N_c(i)| = n_0 d_c$, $(0 \le i \le p)$ and $|N_v(j)| = d_c$, $(0 \le j \le n_0 p)$.

Let $N_c(i) = \{j_1, j_2, \cdots, j_d\}$ (note that in the case of (3), $d = n_0 d_c$), then we define a linear map $\boldsymbol{P}_i : \mathbb{R}^n \to \mathbb{R}^d$ as

$$(\boldsymbol{P}_i)_{k,l} = \begin{cases} 1, & l = j_k \\ 0, & \text{else.} \end{cases}$$

For example, in the case

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{5}$$

we have

$$\boldsymbol{P}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \ \boldsymbol{P}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\boldsymbol{P}_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Define

$$\mathbb{P}_d := \left\{ \boldsymbol{x} \in \{0,1\}^d \mid \|\boldsymbol{x}\|_1 \text{ is even} \right\},$$

where $\|\cdot\|_p$ represents a l^p norm. Then, we can write

$$\boldsymbol{x} \in \mathcal{C} \Leftrightarrow \boldsymbol{P}_i \boldsymbol{x} \in \mathbb{P}_d, \ (1 \le i \le m).$$

For $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, $W(y|x)$ represents the conditional probability of BSC, i.e. $W(1|0) = W(0|1) = p$ and $W(0|0) = W(1|1) = 1 - p$ where $p \in [0, 1/2)$. Assume a sending message is $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathcal{X}^n$ and a received message be $\boldsymbol{y} = (y_1, \dots, y_n) \in \mathcal{Y}^n$. Since we assume memoryless channel, the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ is represented as: $p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{j=1}^n W(y_j|x_j)$. The ML decoding choose $\boldsymbol{x} \in \mathcal{X}^n$ that maximizes $p(\boldsymbol{y}|\boldsymbol{x})$, thus equivalently maximizes $\sum_{j=1}^n \log W(y_j|x_j)$. We put

$$\gamma_j := \log\left(W(y_j|0)/W(y_j|1)\right), \quad (1 \le j \le n). \tag{6}$$

Since $\log W(y_j|x_j) = -\gamma_j x_j + \log W(y_j|0)$, it reduces to determine $\boldsymbol{x} \in \mathcal{C}$ that minimizes $\boldsymbol{\gamma}^T \boldsymbol{x} = \sum_{j=1}^n \gamma_j x_j$. Thus, ML decoding is described as:

$$\text{Minimize:} \quad \sum_{j=1}^n \gamma_j x_j \ \text{ subject to } \boldsymbol{P}_i \boldsymbol{x} \in \mathbb{P}_d, \ (1 \le i \le m). \tag{7}$$

By the relaxation of each binary variable x_i $(1 \le i \le n)$ to the interval $[0,1]$, we obtain LP (linear programming) relaxation of IP (7) (The LP relaxation by Feldman etc. [10] is a kind of this relaxation):

$$\text{Minimize:} \quad \sum_{j=1}^n \gamma_j x_j \ \text{ subject to } \boldsymbol{P}_i \boldsymbol{x} \in \mathbb{PP}_d, \ (1 \le i \le m), \tag{8}$$

where $\mathbb{PP}_d$ is a convex hull of $\mathbb{P}_d$, i.e. $\mathbb{PP}_d = \text{conv}(\mathbb{P}_d)$.

C. ADMM Formulation

For expressing LP decoding formulation (8) to corresponding ADMM formulation, the "replica" vectors $\boldsymbol{z}_i$, $(1 \le i \le m)$ are introduced. Moreover, to accelerate the computation, we also introduce a penalty term. Hence, we consider

$$\begin{aligned} &\text{Minimize:} \\ &\sum_{j=1}^n \gamma_j x_j - \alpha \sum_{j=1}^n (x_j - 0.5)^2 \\ &\text{subject to } \boldsymbol{P}_i \boldsymbol{x} = \boldsymbol{z}_i, \ \boldsymbol{z}_i \in \mathbb{PP}_d, \ (1 \le i \le m), \end{aligned} \tag{9}$$

where $\alpha > 0$ is an acceleration parameter and the second term of objective function is a penalty term. Put $\boldsymbol{z} = (\boldsymbol{z}_1, \dots, \boldsymbol{z}_m)$ and $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_m)$. The ADMM method optimize the augmented Lagrangian:

$$\begin{aligned} L_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) := &\sum_{j=1}^n \gamma_j x_j - \alpha \sum_{j=1}^n (x_j - 0.5)^2 \\ &+ \sum_{i=1}^m \boldsymbol{\lambda}_i^T (\boldsymbol{P}_i \boldsymbol{x} - \boldsymbol{z}_i) + \frac{\mu}{2} \sum_{i=1}^m \|\boldsymbol{P}_i \boldsymbol{x} - \boldsymbol{z}_i\|_2^2, \end{aligned} \tag{10}$$

The update process of ADMM is as follows:

$$\begin{aligned} \boldsymbol{x}^{q+1} &:= \arg \min_{\boldsymbol{x} \in [0,1]^n} L_\mu(\boldsymbol{x}, \boldsymbol{z}^q, \boldsymbol{\lambda}^q) \\ \boldsymbol{z}_i^{q+1} &:= \arg \min_{\boldsymbol{z} \in (\mathbb{PP}_d)^m} L_\mu\left(\boldsymbol{x}^{q+1}, \boldsymbol{z}, \boldsymbol{\lambda}^q\right), \ (1 \le i \le m) \\ \boldsymbol{\lambda}_i^{q+1} &:= \boldsymbol{\lambda}_i^q + \mu\left(\boldsymbol{P}_i \boldsymbol{x}^{q+1} - \boldsymbol{z}_i^{q+1}\right), \ (1 \le i \le m). \end{aligned} \tag{11}$$

Let $\boldsymbol{\delta x} \in \mathbb{R}^n$ be a variation vector of $\boldsymbol{x}$, $\boldsymbol{\gamma} = (\gamma_1, \dots, \gamma_n)$ and $\boldsymbol{1} = (1, \dots, 1)^T$. Then, we obtain

$$\begin{aligned} &L_\mu(\boldsymbol{x} + \boldsymbol{\delta x}, \boldsymbol{z}, \boldsymbol{\lambda}) - L_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) \\ &= \langle \boldsymbol{\gamma}, \boldsymbol{\delta x} \rangle - 2\alpha \langle \boldsymbol{x} - 0.5 \cdot \boldsymbol{1}, \boldsymbol{\delta x} \rangle \\ &\quad + \sum_{i=1}^m \langle \boldsymbol{\lambda}_i, \boldsymbol{P}_i \boldsymbol{\delta x} \rangle + \mu \sum_{i=1}^m \langle \boldsymbol{P}_i \boldsymbol{x} - \boldsymbol{z}_i, \boldsymbol{P}_i \boldsymbol{\delta x} \rangle \\ &= \langle \boldsymbol{\gamma}, \boldsymbol{\delta x} \rangle - 2\alpha \langle \boldsymbol{x} - 0.5 \cdot \boldsymbol{1}, \boldsymbol{\delta x} \rangle \end{aligned}$$

$$
\begin{aligned}
&+\sum_{i=1}^{m}\langle \boldsymbol{P}_i^T\boldsymbol{\lambda}_i,\boldsymbol{\delta x}\rangle+\mu\sum_{i=1}^{m}\langle \boldsymbol{P}_i^T\boldsymbol{P}_i\boldsymbol{x}-\boldsymbol{P}_i^T\boldsymbol{z}_i,\boldsymbol{\delta x}\rangle\\
&=\langle\boldsymbol{\gamma},\boldsymbol{\delta x}\rangle-2\alpha\langle\boldsymbol{x}-0.5\cdot\boldsymbol{1},\boldsymbol{\delta x}\rangle\\
&+\sum_{i=1}^{m}\langle \boldsymbol{P}_i^T\boldsymbol{\lambda}_i,\boldsymbol{\delta x}\rangle+\langle\mu\boldsymbol{P}\boldsymbol{x}-\mu\sum_{i=1}^{m}\boldsymbol{P}_i^T\boldsymbol{z}_i,\boldsymbol{\delta x}\rangle,
\end{aligned}
$$

where $\boldsymbol{P}=\sum_{i=1}^{m}\boldsymbol{P}_i^T\boldsymbol{P}_i$. Thus, we obtain,

$$
\left(\boldsymbol{P}-\frac{2\alpha}{\mu}\boldsymbol{I}\right)\boldsymbol{x}=\sum_{i=1}^{m}\left(\boldsymbol{P}_i^T\boldsymbol{z}_i-\frac{1}{\mu}\boldsymbol{P}_i^T\boldsymbol{\lambda}_i\right)-\frac{\alpha}{\mu}\boldsymbol{1}-\frac{\boldsymbol{\gamma}}{\mu}. \tag{12}
$$

In the case of (5), we have

$$
\boldsymbol{P}=\sum_{i=1}^{3}\boldsymbol{P}_i^T\boldsymbol{P}_i=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&2&0&0&0\\0&0&0&2&0&0\\0&0&0&0&1&0\\0&0&0&0&0&2\end{bmatrix},
$$

and in general, we have $(\boldsymbol{P})_{i,j}=|N_v(j)|\delta_{i,j}$. For simplicity, we denote by $z_i^{(j)}$, the j-th component of $\boldsymbol{P}_i^T\boldsymbol{z}_i$. Similarly, $\lambda_i^{(j)}$ denotes the j-th component of $\boldsymbol{P}_i^T\boldsymbol{\lambda}_i$. Hence from (12) and the restriction $x_j\in[0,1]$ for $(1\le j\le n)$, it holds that

$$
\begin{aligned}
&x_j\\
&=\Pi_{[0,1]}\left(\frac{1}{|N_v(j)|-\frac{2\alpha}{\mu}}\right)\left\{\sum_{i=1}^{m}\left(z_i^{(j)}-\frac{\lambda_i^{(j)}}{\mu}\right)-\frac{\alpha+\gamma_j}{\mu}\right\},\\
&\quad(1\le j\le n),
\end{aligned}\tag{13}
$$

where $\Pi_{[0,1]}$ is a projection to the interval $[0,1]$. To obtain the minimizer of the second expression of (11), we rewrite the third and fourth terms of (10) as follows:

$$
\begin{aligned}
&\boldsymbol{\lambda}_i^T\left(\boldsymbol{P}_i\boldsymbol{x}-\boldsymbol{z}_i\right)+\frac{\mu}{2}\|\boldsymbol{P}_i\boldsymbol{x}-\boldsymbol{z}_i\|_2^2\\
&=\frac{\mu}{2}\left\{\|\boldsymbol{P}_i\boldsymbol{x}+\frac{\boldsymbol{\lambda}_i}{\mu}-\boldsymbol{z}_i\|_2^2-\frac{\|\boldsymbol{\lambda}_i\|_2^2}{\mu^2}\right\}.
\end{aligned}
$$

Hence it holds

$$
\boldsymbol{z}_i^{q+1}=\arg\min_{\boldsymbol{z}_i\in\mathbb{PP}_d}\|\boldsymbol{P}_i\boldsymbol{x}^{q+1}+\frac{\boldsymbol{\lambda}_i^q}{\mu}-\boldsymbol{z}_i\|_2,
$$

which is a metric projection of $\boldsymbol{P}_i\boldsymbol{x}^{q+1}+\frac{\boldsymbol{\lambda}_i^q}{\mu}$ onto the closed convex set $\mathbb{PP}_d$ with $d=N_0d_c$, i.e.

$$
\boldsymbol{z}_i^{q+1}=\Pi_{\mathbb{PP}_d}\left(\boldsymbol{P}_i\boldsymbol{x}^{q+1}+\frac{\boldsymbol{\lambda}_i^q}{\mu}\right). \tag{14}
$$

Summarizing the above argument, the ADMM decoding procedure is described as follows. Recall that $\boldsymbol{y}=\boldsymbol{x}+\boldsymbol{e}$ is a received message where $\boldsymbol{e}$ satisfies $w(\boldsymbol{e})=t$.

$(1\le$

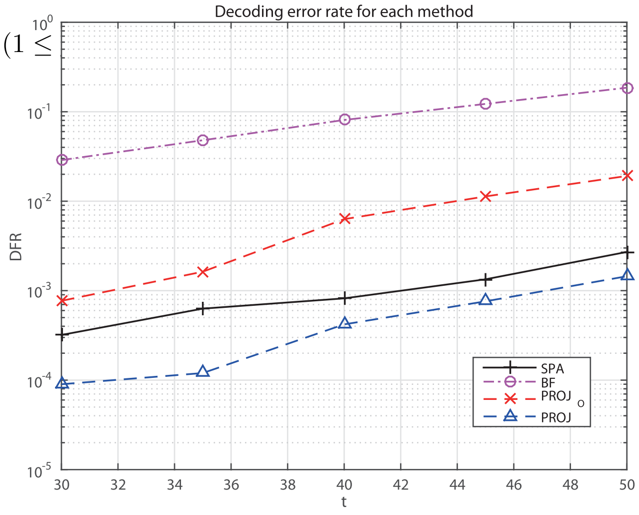

Fig. 1. DFR of each decoding method for $(n,k)=(2000,1000)$ and $d_c=5$ LDPC code.

Algorithm 2 The ADMM Decoding Algorithm

```
1: Initialize $\boldsymbol{\gamma}$ as (6), $q=0$, set a positive integer $M$ and $\epsilon>0$, a small number.
2: Initialize $\boldsymbol{z}_i=\boldsymbol{P}_i\boldsymbol{y}$, $\boldsymbol{\lambda}_i=0$,  $(1\le i\le m)$.
3: repeat
4:     for all $j$ $(1\le j\le n)$ do
5:     Update $x_j$ accrding to (13).
6:     end for
7:     for all  $i$ $(1\le i\le m)$ do
8:         set $\boldsymbol{v}_i=\boldsymbol{P}_i\boldsymbol{x}+\boldsymbol{\lambda}_i/\mu$.
9:         $\boldsymbol{z}_i\leftarrow\Pi_{\mathbb{PP}_{n_0d_c}}(\boldsymbol{v}_i)$.
10:        $\boldsymbol{\lambda}_i\leftarrow\boldsymbol{\lambda}_i+\mu\left(\boldsymbol{P}_i\boldsymbol{x}-\boldsymbol{z}_i\right)$.
11:    end for
12:    $q\leftarrow q+1$
13: until  $H\boldsymbol{x}^T=\boldsymbol{0}$ (Parity satisfied)  or  $q>M$
14: return $\boldsymbol{x}$.
```

The most time consuming part of the Algorithm 2 is Step 9: $\boldsymbol{z}_i\leftarrow\Pi_{\mathbb{PP}_{n_0d_c}}(\boldsymbol{v}_i)$, a projection of $\boldsymbol{v}_i$ to code-polytope $\mathbb{PP}_{n_0d_c}$.

Here, we have to note the projection algorithm to the code-polytope $\mathbb{PP}_{n_0d_c}$ (Algorithm 2 of [4]) seems to need some modification, and without such modification, non-negligible degradation in decoding performance arises. Indeed, sometimes mismatching values were observed for the convex-quadratic programming (21) compared to the one obtained by MIP solver Gurobi optimizer 9.5 [12] (we note replacing the projection algorithm for (21) with MIP solver is unrealistic from the time-consuming view point).

We will show these degradations with numerical examples. Figure 1 shows the DFR of each decoding method for $(n,k)=(2000,1000)$ and $d_c=5$ LDPC code. In the figure labels "SPA" and "BF" mean sum-product algorithm and bit-flipping algorithm, respectively. While labels "PROJ$_\text{O}$" and "PROJ" mean ADMM decoding with original projection algorithm of [4] (Algorithm 2) and corrected projection algorithm respectively. We can observe that "PROJ" decreases DFR about $1/10$ times to "PROJ$_\text{O}$" for each error number t

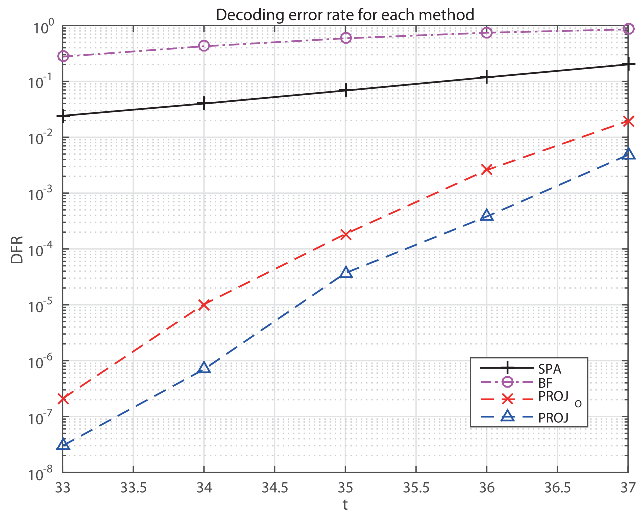

Fig. 2. DFR of each decoding method for $(n, k) = (2000, 1000)$ and $d_c = 25$ MDPC code.

and even superior to SPA. We note both ADMM decoding use parameters $\alpha = 10$, $\mu = 6$ and $M = 10000$. Figure 2 shows the DFR of each decoding method for $(n, k) = (2000, 1000)$ and $d_c = 25$ MDPC code. In this case SPA deteriorates its performance compared to the of Figure 1 since the low-density property of parity check matrix was lost. Comparing "PROJ$_\mathrm{O}$" with "PROJ", we observe that in this case "PROJ" also decreases DFR about $1/10$ times to "PROJ$_\mathrm{O}$" for each error number t. We note both ADMM decoding use parameters $\alpha = 10$, $\mu = 6$ and $M = 10000$.

From these observations, we conclude that the correction of projection algorithm is an important factor for the improvement of DFR.

D. Modified Projection Algorithm

Following is a correction of projection algorithm to $\mathbb{PP}_{n_0 d_c}$. For simplicity, we abbreviate $d = n_0 d_c$ and denote by $\lfloor x \rfloor_{\text{even}}$ the largest even number integer smaller than or equals to x. The correctness of the following algorithm is shown in Appendix A.

IV. Numerical Experiments

We examine the proposing ADMM based decoding with some parameters of Table II of [16]. Numerical examinations were executed on Intel Core i9-7980XE CPU @ 2.60GHz processor, with no parallelization (except Step 8 in Algorithm 5) and ANSI C implementation (icc 2021.6.0 with -O2 option). Although we are concerned with DFR properties of each method, we show their decoding time in our implementation. Table I shows the average execution time (ms) for $(n, k, d_c, n_0) = (9802, 4601, 45, 2)$ code. The number of experiments for each method is 100000. In SP, maximum iteration number is set as 50 and in ADMM, parameters are set as $(\mu, \alpha, M) = (6, 15, 10000)$. The label "ME" is the "matrix-extension" method proposed in [7]. The maximum extension size of parity matrix is 14403×19204.

TABLE I
Average Decoding Time (Ms) of Each Decoding Methods for $(n, k, d_c, n_0) = (9802, 4601, 45, 2)$ Code

method	$t = 90$	$t = 100$
BF	275	322
SPA	597	680
ADMM	561	876
ME	276	352
RSPA-ADMM	602	718

Algorithm 3 The Projection Algorithm $\Pi_{\mathbb{PP}_d}(\boldsymbol{u})$

```
1: Permutate $\boldsymbol{u}$ to $\boldsymbol{v}$ in descending order: $v_1 \geq v_2 \geq \cdots \geq v_d$. Let $\boldsymbol{Q}$ be the corresponding permutation matrix i.e. $\boldsymbol{v} = \boldsymbol{Q}\boldsymbol{u}$.
2: $\hat{\boldsymbol{z}} \leftarrow \Pi_{[0,1]^d}(\boldsymbol{v})$. Put $r = \lfloor \|\hat{\boldsymbol{z}}\|_1 \rfloor_{\text{even}}$.
3: Put $\boldsymbol{f} = (f_1, \ldots, f_d)$ and set $f_i = 1$, $(1 \leq i \leq r+1)$, $f_i = -1$, $(r+2 \leq i \leq d)$.
4: if $r = d$ or $r = d-1$ then
5:     return $\boldsymbol{Q}^T \hat{\boldsymbol{z}}$.
6: end if
7: if $\sum_{i=1}^{d} f_i \hat{z}_i \leq r$ then
8:     return $\boldsymbol{Q}^T \hat{\boldsymbol{z}}$.
9: end if
10: set $\beta_{\max} \leftarrow \frac{1}{2}(v_{r+1} - v_{r+2})$.
11: set $\hat{\boldsymbol{\beta}} \leftarrow \left\{\hat{\beta}_1, \ldots, \hat{\beta}_{2d}\right\}$ as
    $$\hat{\beta}_i = \begin{cases} v_i - 1, & (1 \leq i \leq r+1) \\ -v_i, & (r+2 \leq i \leq d) \\ v_{i-d}, & (d+1 \leq i \leq d+r+1) \\ 1 - v_{i-d}, & (d+r+2 \leq i \leq 2d). \end{cases}$$
12: set $\tilde{\boldsymbol{\beta}} \leftarrow \left\{\tilde{\beta} \in \hat{\boldsymbol{\beta}} \cup \{0\} \cup \{\beta_{\max}\} \mid 0 \leq \tilde{\beta} \leq \beta_{\max}\right\}$.
13: Sort $\tilde{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$ in ascending order. Hence $\boldsymbol{\beta} = \{\beta_1, \beta_2, \ldots \beta_p\}$ satisfies $0 = \beta_0 < \beta_1 < \cdots < \beta_p = \beta_{\max}$.
14: set $L \leftarrow 0, R \leftarrow p, M \leftarrow \lfloor (L+R)/2 \rfloor$.
15: while $R - L > 1$ do
16:     set
        $$\hat{z}_i = \begin{cases} v_i - \beta_M, & (1 \leq i \leq r+1) \\ v_i + \beta_M, & (r+2 \leq i \leq d). \end{cases}$$
17:     if $\sum_{i=1}^{d} f_i \cdot \left(\Pi_{[0,1]}(\hat{z}_i)\right) < r$ then
18:         $R \leftarrow M$.
19:     else
20:         $L \leftarrow M$.
21:     end if
22:     $M \leftarrow \lfloor (L+R)/2 \rfloor$.
23: end while
24: set
    $$\hat{z}_i = \begin{cases} v_i - \beta_R, & (1 \leq i \leq r+1) \\ v_i + \beta_R, & (r+2 \leq i \leq d). \end{cases}$$
25: set $f_z \leftarrow \sum_{i=1}^{d} f_i \cdot \left(\Pi_{[0,1]}(\hat{z}_i)\right)$.
```

```
26: set
      $$\hat{z}_i = \begin{cases} v_i - \beta_L, & (1 \le i \le r+1) \\ v_i + \beta_L, & (r+2 \le i \le d). \end{cases}$$
27: set $f_{z_0} \leftarrow \sum_{i=1}^{d} f_i \cdot \left(\Pi_{[0,1]}\left(\hat{z}_i\right)\right)$.
28: set $\beta_{\text{opt}} \leftarrow \frac{(r-f_z)(\beta_R-\beta_L)}{f_z - f_{z_0}} + \beta_R$
29: set $z_i$, $(1 \le i \le d)$ as
      $$z_i = \begin{cases} \Pi_{[0,1]}\left(v_i - \beta_{\text{opt}}\right), & (1 \le i \le r+1) \\ \Pi_{[0,1]}\left(v_i + \beta_{\text{opt}}\right), & (r+1 \le i \le d). \end{cases}$$
30: return $\boldsymbol{Q}^T \boldsymbol{z}$.
```

As we can recognize from Figures 7 and 8, error correction ability of ADMM is superior to BF SP and ME. It is observed that DFR falls rapidly if t is less than certain threshold value t_0 (for the case $(n, k) = (9802, 4601)$, $t_0 = 103$ and for the case $(n, k) = (10779, 3593)$, $t_0 = 66$). Thus, we can expect further decrement of DFR if with some preprocessing, error number t could be lowered than t_0. For that purpose, we introduce the method RSPA (reproducing sum-product algorithm), which efficiently reuse decoding failure results of SPA as a preprocessing for ADMM. The algorithm RSPA is as follows:

Algorithm 4 The reproducing sum-product algorithm

```
 1: set $N_{\text{SPA}}$ as maximum iteration number, $i = 0$.
 2: while $i < N_{\text{SPA}}$ do
 3:     set $\boldsymbol{c}'$ as decoding result of SPA for $\boldsymbol{c}$ (after hard
        decision is applied to each bits).
 4:     if $H\boldsymbol{c}'^T = \boldsymbol{0}$ then
 5:         return $\boldsymbol{c}'$.
 6:     end if
 7:     $\boldsymbol{c} \leftarrow \boldsymbol{c}'$.
 8:     $i \leftarrow i + 1$.
 9: end while
10: return decoding failure.
```

We consider the case $H\boldsymbol{c}^T \neq 0$. Put the difference between noiseless received information $\boldsymbol{mG'}$ and decoding result of SPA (after hard decision to each bits) as $d_h(\boldsymbol{c}')$: i.e. $d_h(\boldsymbol{c}') = w\left(\boldsymbol{mG'} - \boldsymbol{c}'\right)$. Thus $d_h(\boldsymbol{c}') - t$ represents decrease (if $d_h(\boldsymbol{c}') - t < 0$) and increase (if $d_h(\boldsymbol{c}') - t > 0$) of error bits number $\|\boldsymbol{e}\|_1$ after hard decision. Figure 3 show the number of observed values of $d_h(\boldsymbol{c}') - t$ when $t = 110$. For example, the number of trials which take $-5 \le d_h(\boldsymbol{c}') - t \le -1$ occupies 51, while the number of trials which take $1 \le d_h(\boldsymbol{c}') - t \le 5$ is 5, under total 100 decoding errors. Figure 4 shows the case $t = 95$. In this case, there is no case which takes $d_h(\boldsymbol{c}') - t \ge 0$. From Figure 3 and 4, we can expect that as the iteration number i increases, the value $d_h(\boldsymbol{c}') - t$ tends to decrease. As a result, all errors will be fixed as the iteration number increases. We note that the same effect cannot be expected for BF algorithm. Figure 5 and 6 are results obtained by replacing the decoding method SPA in Algorithm 4 to BF under the circumstance of Figure 3 and 4 respectively. We observe that Figure 5 and 6 show $d_h(\boldsymbol{c}') - t > 0$ almost all

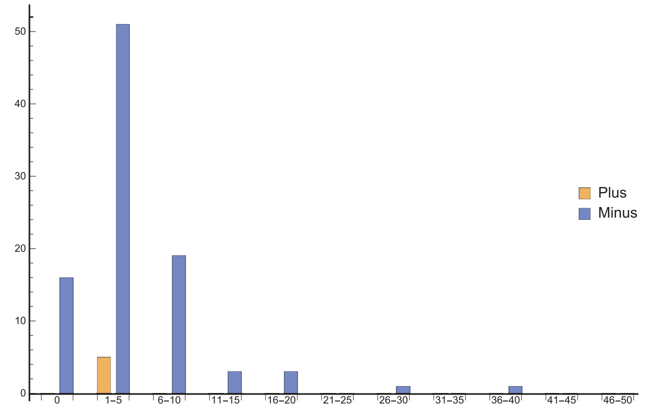

Fig. 3. The barchart of the number of observed values of $d_h(\boldsymbol{c}') - t$ in the case of SPA decoding and $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code $(t = 110)$.

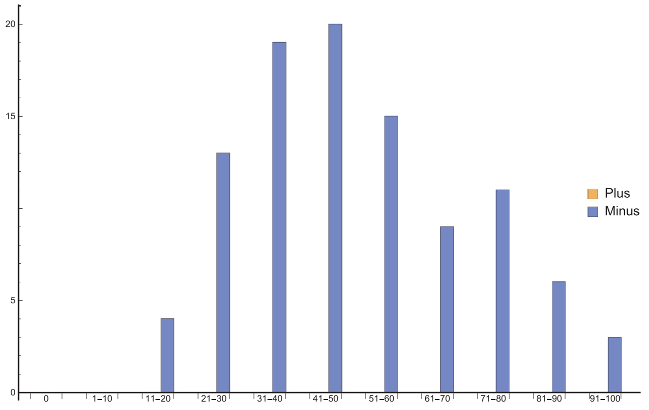

Fig. 4. The bar chart of the number of observed values of $d_h(\boldsymbol{c}') - t$ in the case of SPA decoding and $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code $(t = 95)$.

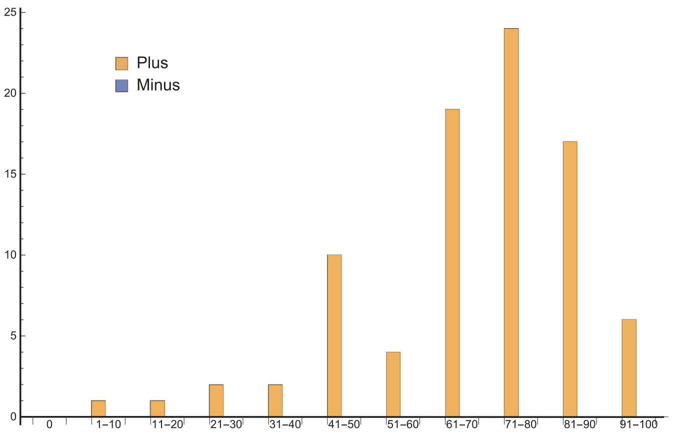

Fig. 5. The bar chart of the number of observed values of $d_h(\boldsymbol{c}') - t$ in the case of BF decoding and $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code $(t = 110)$.

cases, and hence we cannot expect the reduction of erroneous bits by successive hard decision of decoding results.

After each iteration of RSPA, in most case, the number of erroneous bits is expected to reduce, hence by applying ADMM method to $\boldsymbol{c}$ obtained by hard decision in Step 7, further improvement in DFR can be expected. We call this hybrid algorithm RSPA-ADMM method.

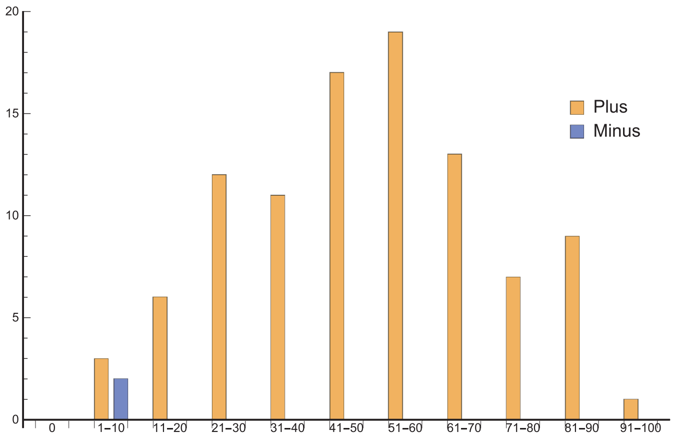

Fig. 6. The bar chart of the number of observed values of $d_h(\boldsymbol{c}') - t$ in the case of BF decoding and $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code ($t = 95$).

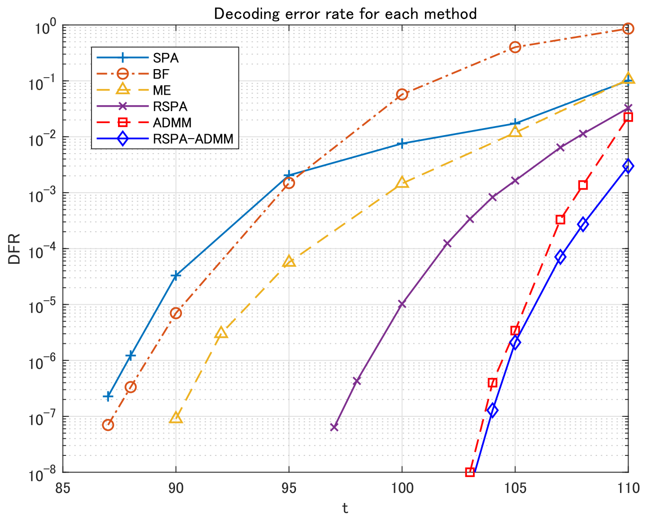

Fig. 7. DFR of each decoding method for $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code.

Algorithm 5 The RSPA-ADMM Method

```
1:  set $N_{\text{SPA}}$ as maximum iteration number, $i = 0$.
2:  while $i < N_{\text{SPA}}$ do
3:      set $\boldsymbol{c}'$ as decoding result of SPA for $\boldsymbol{c}$ (after hard decision is applied to each bits).
4:      if $H\boldsymbol{c}'^T = \boldsymbol{0}$ then
5:          return $\boldsymbol{c}'$.
6:      end if
7:      $\boldsymbol{c} \leftarrow \boldsymbol{c}'$.
8:      Apply ADMM decoding to $\boldsymbol{c}$. set $\boldsymbol{c}'$ as decoding result of ADMM for $\boldsymbol{c}$.
9:      if $H\boldsymbol{c}'^T = \boldsymbol{0}$ then
10:         return $\boldsymbol{c}'$.
11:     end if
12:     $i \leftarrow i + 1$.
13: end while
14: return decoding failure.
```

We note Step 8 can be done in parallel.

Figure 7 shows the DFR of 6 methods (SPA, BF, ME, RSPA, ADMM, RSPA-ADMM) for $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code. The maximum iteration number of SP was set to 50. The parameters (μ, α, M) in ADMM and RSPA-ADMM methods were fixed to $(6, 15, 10000)$. We note parameters μ and α were chosen by heuristic trials, hence these may not be an optimal choice. In spite of this, it holds that ADMM and RSPA-ADMM attains DFR lower than 10^{-7} at $t = 103$ compared to the result that BF decoding requires $t = 87$ for attaining almost the same DFR. We note DFR of RSPA-ADMM when $t = 103$ might be less than 10^{-8}, since no decoding error was observed under 10^8 trials. We can observe that RSPA-ADMM improves the DFR compared to ADMM. Figure 8 shows the DFR of 6 methods (SPA, BF, ME, RSPA, ADMM, RSPA-ADMM) for $(n, k, d_c, n_0) = (10779, 3593, 51, 3)$ code. The maximum iteration number of SP was set to 50 and the maximum extension size of parity matrix in ME is 14372×21588. The parameters (μ, α, M) in ADMM and RSPA-ADMM methods were fixed to $(8, 15, 10000)$. It seems to be remarkable that ADMM and RSPA-ADMM attains DFR lower than 10^{-7} at $t = 66$ compared to the result that BF decoding requires $t = 55$ for attaining almost the same DFR. We note DFR of RSPA-ADMM when $t = 66$ might be less than 10^{-8}, since no decoding error was observed under 10^8 trials. In this case, we also observe that RSPA-ADMM improves the DFR compared to ADMM.

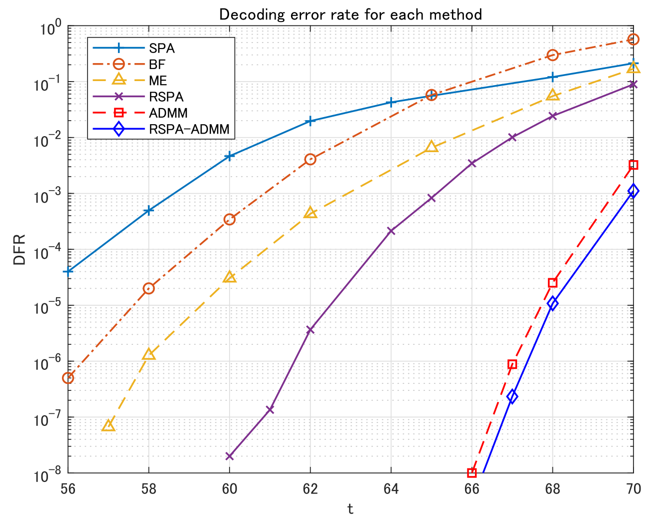

Fig. 8. DFR of each decoding method for $(n, k, d_c, n_0) = (10779, 3593, 51, 3)$ code.

A. Comparison With the Decoding Algorithm of BIKE

BIKE (Bit Flipping Key Encapsulation) is submitted for round 4 selection of NIST PQC [30]. For its λ bits security under IND-CPA circumstance, work-factor of QCSDP (QC-Syndrome Decoding Problem) and QCCFP (QC-Codeword Finding Problem) are required to be over 2^λ; see p.8-9 of [6] and p.7 of [23]. These assumptions are satisfied by the appropriate choice of (m, t) and (m, d_c) respectively. For example, Table II of [16] represents parameters (n, m, d_c, t) for λ bit IND-CPA security. However, for the λ bits IND-CCA security, further requires that DFR$\leq 2^{-\lambda}$; see p. 9 of [6] and p. 7 of [23]. Direct confirmation of this assumption is impossible, so DFR are computed for some

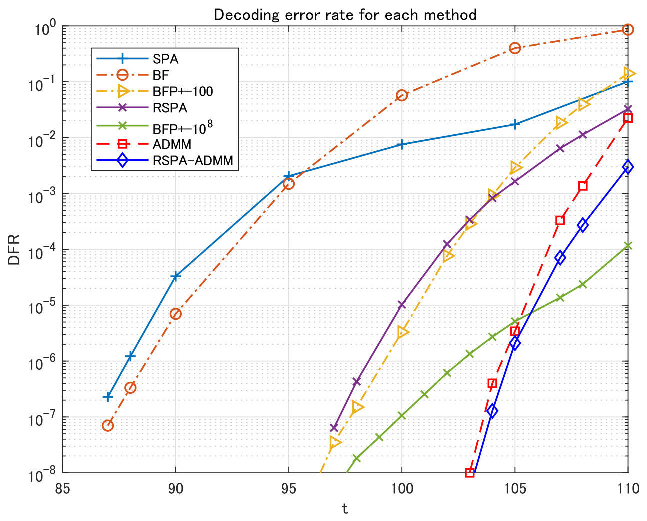

Fig. 9. DFR of each decoding method for $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code.

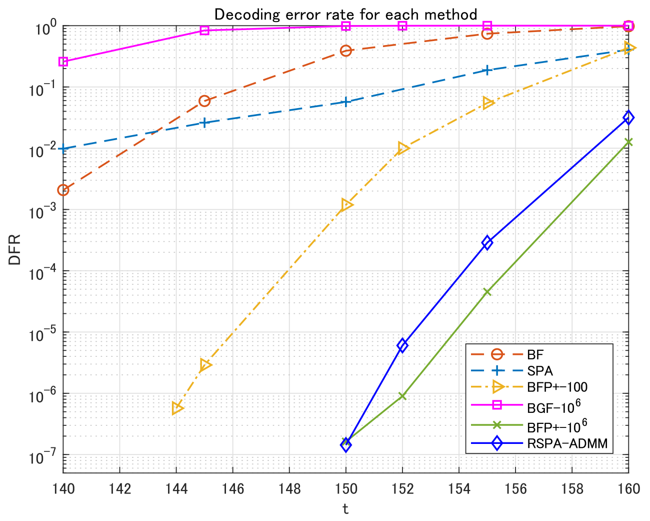

Fig. 10. DFR of each decoding method for $(n, k, d_c, n_0) = (19714, 9857, 71, 2)$ code.

feasible (m, t, d_c) cases, and using these information, fixing (t, d_c), the matrix size m that reaches DFR$\leq 2^{-\lambda}$ is computed by the method of extrapolation. The current version of BIKE uses BGF (Black-Gray-Flip) decoder [9] that fix over flipped erroneous bits belonging to black and gray lists once. The backflip$^+$ algorithm [8] achieves the same effect as the BGF decoder by introducing the concept of TTL (Time-To-Live) period for each bit. As shown on p.3 of [8], the backflip$^+$ algorithm performs better than the BG decoder (so possibly BGF decoder) when a large number of iterations for flipping is allowed (it is also noted for practical iteration number and large matrix size m, BG decoder performs better).

Figure 9 is the comparison with back-flip$^+$ decoder (we have used the decoder from [24]) with RSPA-ADMM algorithm for $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ (80-bit security parameter from Table II of [16]). The label "BFP$^+$-100" and "BFP$^+$-10^8" are the results of backflip$^+$ algorithm with an iteration number of 100 and 10^8 respectively. It can be seen that the DFR curve of "BFP$^+$-100" is almost the same as the "RSPA". It is observed that when $t > 105$, the DFR of BFP$^+$-10^8 is lower than that of the RSPA-ADMM method. While for the case $t \leq 105$, the DFR of RSPA is lower than that of BFP$^+$-10^8 and at $t = 105$ it shows rapid decrease of DFR compared to the "BFP$^+$-10^8" DFR curve. We note the DFR of BFP$^+$-10^8 seems to be gradually approaching the result of BFP$^+$-100.

Figure 10 is the comparison BGF and backflip$^+$ decoders with RSPA-ADMM algorithm for $(n, k, d_c, n_0) = (19714, 9857, 71, 2)$ (128-bit security parameter from Table II of [16]). The reason we examined the BGF algorithm is that the parameter of the threshold function is given by on page 8 of BIKE [6] for $(d_c, t) = (71, 134)$. The label "BGF-10^6" is th result of BGF algorithm with an iteration number of 10^6. The labels "BFP$^+$-10^6" and "BFP$^+$-100" are the results of the Backflip+ algorithm with an iteration number of 10^6 and 100 respectively. Since we make the number of iterations very large, as mentioned above, the backflip$^+$ algorithm outperforms the BGF algorithm. We can see that the DFR of RSPA-ADMM is almost the same as that of "BFP$^+$-10^6" at $t = 150$, but shows a rapid decrease in DFR compared to the "BFP$^+$-10^6" DFR curve. In the RSPA-ADMM algorithm, the parameters are set as $(\mu, \alpha, M) = (13, 26, 10000)$ in the ADMM part and the iteration number is 100 in the sum-product part. We note that the average decoding time of RSPA-ADMM is large (about 3000ms even for $t = 134$), and therefore improvement of the algorithm seems necessary for larger values of m.

B. Weak Key Examination

Weak keys are a set of keys that have relatively high DFR compared to ordinary random keys [1], [3], [8], [18], [23]. Thus, weak keys are a potential thread for IND-CCA security if the probability of their occurrence is not negligible. In this subsection, we examine Type I weak keys of [23] (originally defined in [8]) and show that Type I weak keys are not weak keys for the ADMM decoder, but rather, these keys are "strong keys". First, we introduce two lemmas about ring isomorphism.

Lemma 1: Assume $h_i \in \mathrm{GF}_2$ $(0 \leq i \leq m)$. Then

$$\varphi : \boldsymbol{H} \mapsto h_0 + h_1 x + \cdots + h_{m-2}x^{m-2} + h_{m-1}x^{m-1} \quad (15)$$

is ring isomorphism from $r \times r$ circurant matrices to quotient ring $\mathrm{GF}_2[x]/(x^m - 1)$.
For example, in the case

$$\boldsymbol{H}_a = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \ \boldsymbol{H}_b = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix},$$

$$\varphi\left(\boldsymbol{H}_a + \boldsymbol{H}_b\right) = \varphi\left(\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}\right)$$

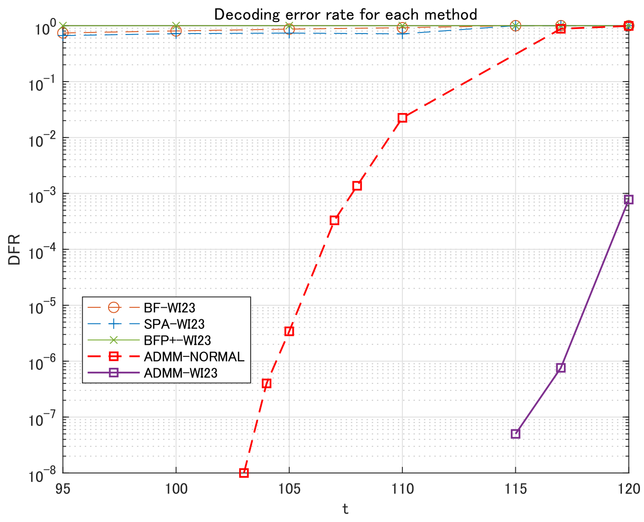

Fig. 11. DFR of each decoding method for $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ and $(f, \delta) = (23, 0)$ Type I weak key code.

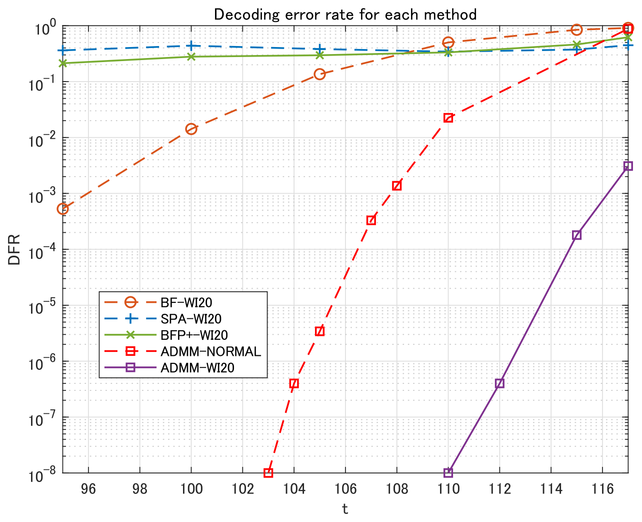

Fig. 12. DFR of each decoding method for $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ and $(f, \delta) = (20, 7)$ Type I weak key codecode.

$$
\begin{aligned}
&= x + x^2 = \left(1 + x + x^3\right) + \left(1 + x^2 + x^3\right)\\
&= \varphi(\boldsymbol{H}_a) + \varphi(\boldsymbol{H}_b),
\end{aligned}
$$

and

$$
\begin{aligned}
\varphi\left(\boldsymbol{H}_a \cdot \boldsymbol{H}_b\right) =& \varphi\left(\begin{bmatrix} 0 & 0 & 1 & 1 & 1\\ 1 & 0 & 0 & 1 & 1\\ 1 & 1 & 0 & 0 & 1\\ 1 & 1 & 1 & 0 & 0\\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}\right)\\
=& x^2 + x^3 + x^4 = \left(1 + x + x^3\right) \cdot \left(1 + x^2 + x^3\right)\\
=& \varphi(\boldsymbol{H}_a) \cdot \varphi(\boldsymbol{H}_b).
\end{aligned}
$$

Also, following holds.

Lemma 2: For any $\delta \in \boldsymbol{Z}_m \setminus \{0\}$, $\phi_\delta : x \mapsto x^\delta$ is the ring isomorphism from $\mathrm{GF}_2[x]/(x^m - 1)$ to $\mathrm{GF}_2[x]/(x^m - 1)$.

For example, in the case $\delta = 3$ and $h(x) = 1 + x + x^3 \in \mathrm{GF}_2[x]/(x^5 - 1)$,

$$\phi_3(h(x)) = 1 + x^3 + x^9 = 1 + x^3 + x^4.$$

Using above lemmas, we define Type I weak keys [23].

Definition 3: Let

$$\boldsymbol{H} = [\boldsymbol{H_0}, \boldsymbol{H_1}],$$

where each $\boldsymbol{H_0}, \boldsymbol{H_1}$ is $m \times m$ binary circulant matrix and each row weight is d_c. Further, put $\varphi(\boldsymbol{H_0}) = h_0$ and $\varphi(\boldsymbol{H_1}) = h_1$ respectively, where φ is a ring isomorphism defined in Lemma 1. Type I weak keys are specified as the pair (h_0, h_1) with

$$h_i = \phi_\delta\left((1 + x + \cdots + x^{f-1}) + h'_i\right), \quad i \in \{0, 1\},$$

such that $|h'_i| = d_c - f$, $(i \in \{0, 1\})$ and degree of h'_i, $(i \in \{0, 1\})$ is higher than $f - 1$.

Figure 11 is the DFR of the individual methods BF, SPA, BFP$^+$-10^6 and ADMM for a Type I weak key of $(f, \delta) = (23, 0)$ and $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code. We can see that only ADMM decreases its DFR rapidly at $t = 115$ and the other three methods degrade their DFR compared to the normal random key case; compare figures 11 and 9. It may be interesting to note that the DFR of the ADMM method shows an improvement compared to the normal random key case, so "weak keys" seem to be rather "strong keys" for the ADMM algorithm (at present, we can not provide a satisfactory explanation for this phenomena). Figure 12 is the DFR of the individual methods BF, SPA, BFP$^+$-10^6 and ADMM for a Type I weak key of $(f, \delta) = (20, 7)$ and $(n, k, d_c, n_0) = (9602, 4801, 45, 2)$ code. In this case, we can see the almost the same DFR behavior as in Figure 11.

V. Discussion

This paper presented the effectiveness of ADMM based methods in decoding QC-MDPC codes. Since proposing two methods ADMM and RSPA-ADMM have abilities to correct more errors than standard error number assumed for BF algorithm, these ADMM based decoding can detect irregular usage of error numbers that aimed for the collection of DFR information by adversaries. Hence, proposing schemes may serve as an audit device which guard the conventional BF algorithm based decoding systems.

Although ADMM based decoding showed certain improvements in DFR, following problems remains. First, the systematic methods for the determination of the optimal parameters (μ, α) in ADMM decoding algorithm.

Second, an acceleration problem of the ADMM decoding methods. In Algorithm 2, Step 5 and 7 can be done in parallel. Thus, the acceleration of the projection to $\Pi_{\mathbb{PP}_{n_0 d_c}}$ remains as an essential problem. Zhang-Siegel [28] proposes an alternative projection algorithm that is reported to be effective in improving the decoding speed for LDPC codes. The essential improvement of [28] is replacing sorting operation in Step 1 of Algorithm 3 with "cut search algorithm". Since the numbers of $n_0 d_c$ that should be sorted in Step 1 of Algorithm 3 are rather large number for MDPC codes compared to LDPC codes cases, the "cut search algorithm" might work more effectively than LDPC codes cases in [28].

Third, the error correction capability analysis done for the bitflipping algorithm as [19] and [22] seems to be necessary for the proposed ADMM-based algorithm.

Source Code Usage: The source code for validating the results in Figures 11 and 12 is available from

https://www.nda.ac.jp/ wata/ADMM_prog/ADMM4.zip

They are implemented in ANSI C.

Appendix

A. Preparation

In this section, we show the validity of projection algorithm 3 for the sake of completeness. The main difference with Algorithm 2 of [4] is the set $\hat{\boldsymbol{\beta}}$ in Step 11 of Algorithm 3. In Algorithm 2 of [4], the set $\hat{\boldsymbol{\beta}}$ is defined as:

$$\hat{\beta}_i = \begin{cases} v_i - 1, & (1 \le i \le r+1) \\ -v_i, & (r+2 \le i \le d). \end{cases}$$

This seems to be the main factor which cause non-negligible degradation of DFR in Figure 1 and 2.

First, we prepare lemmas necessary for our purpose.

Lemma 4 ([4, Propostion 3]): Assume the components of $\boldsymbol{v} \in \mathbb{R}^d$ is sorted in descending (ascending) order, then the components of $\Pi_{\mathbb{PP}_d}$ maintain this order.

Definition 5 Assume the components of $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$ are sorted in descending order, i.e.

$$v_1 \ge v_2 \ge \cdots \ge v_d, \quad w_1 \ge w_2 \ge \cdots \ge w_d.$$

Then the vector $\boldsymbol{w}$ is said to "majorizes" $\boldsymbol{u}$ if

$$\begin{cases} \sum_{k=1}^{q} u_k \le \sum_{k=1}^{q} w_k, & (1 \le q \le d-1) \\ \sum_{k=1}^{d} u_k = \sum_{k=1}^{d} w_k. \end{cases}$$

Lemma 6 ([4, Theorem 1]): Suppose vectors $\boldsymbol{u}$ and $\boldsymbol{w}$ are sorted in descending order. Then $\boldsymbol{u}$ is in the convex hull of all permutations of $\boldsymbol{w}$ if and only if $\boldsymbol{w}$ majorizes $\boldsymbol{u}$.

We define $\mathbb{P}_d^s = \{\boldsymbol{x} \in \{0,1\}^d \mid \|\boldsymbol{x}\|_1 = s\}$ and $\mathbb{PP}_d^s = \mathrm{conv}(\mathbb{P}_d^s)$. Recall that $\mathbb{PP}_d = \mathrm{conv}(\mathbb{P}_d)$, so it holds

$$\mathbb{PP}_d = \mathrm{conv}\left(\cup_{0 \le s \le d, s:\mathrm{even}} \mathbb{PP}_d^s\right). \tag{16}$$

From Lemma 6, it holds that $\boldsymbol{u} \in \mathbb{PP}_d^s \iff$

$$\begin{cases} \sum_{k=1}^{q} u_k \le \min(q,s), & (1 \le q \le d-1) \\ \sum_{k=1}^{d} u_k = s. \end{cases} \tag{17}$$

Further, following lemma (two-slice lemma) holds:

Lemma 7 ([4, Lemma 2]): Suppose the components of $\boldsymbol{z} \in \mathbb{PP}_d$ is sorted in descending order. Define r be the minimum even integer greater than or equal to $\lfloor \|\boldsymbol{z}\|_1 \rfloor_{\mathrm{even}}$. Then $\boldsymbol{z}$ is expressed by convex combination of $\mathbb{PP}_d^r$ and $\mathbb{PP}_d^{r+2}$.

Suppose $\boldsymbol{u} \in \mathbb{PP}_d^r \cap [0,1]^d$ and $\boldsymbol{v} \in \mathbb{PP}_d^{r+2} \cap [0,1]^d$. Then, from Lemma 7, for α satisfying $0 \le \alpha \le 1$, it holds

$$\begin{aligned} &\sum_{k=1}^{q} z_k \qquad\qquad\qquad\qquad (18) \\ &= \sum_{k=1}^{q} (\alpha u_k + (1-\alpha) v_k) \\ &= \alpha \sum_{k=1}^{q} u_k + (1-\alpha) \sum_{k=1}^{q} v_k \\ &\le \alpha \min(q,r) + (1-\alpha) \min(q, r+2), \quad (1 \le q \le d-1). \end{aligned}$$

and

$$\begin{aligned} \sum_{k=1}^{d} z_k &= \sum_{k=1}^{d} (\alpha u_k + (1-\alpha) v_k) \qquad (19) \\ &= \alpha \sum_{k=1}^{d} u_k + (1-\alpha) \sum_{k=1}^{d} v_k = \alpha r + (1-\alpha)(r+2). \end{aligned}$$

We note in the case $q \le r$, (18) clearly holds. While in the case $q \ge r+2$, (since $d-1 \ge q \ge r+2$, this case occurs when $r+3 \le d$), we can observe (18) holds if(19) is valid. Thus, (18) should be considered only for the case $q = r+1$. For the case $q = r+1$, (since $d-1 \ge q = r+1$, this case occurs when $r+2 \le d$) (18) becomes as follows

$$\sum_{k=1}^{r+1} z_k \le \alpha r + (1-\alpha)(r+1) = r+1-\alpha, \quad (r+2 \le d). \tag{20}$$

From (19), we have

$$\alpha = 1 + \frac{1}{2}\left(r - \sum_{k=1}^{d} z_k\right),$$

and hence the projection $\Pi_{\mathbb{PP}_d}(\boldsymbol{v})$ for $\boldsymbol{v} \in \mathbb{R}^d$ is described as convex quadratic programming problem:

$$\text{(P) Minimize} \quad \|\boldsymbol{v} - \boldsymbol{z}\|_2^2 \tag{21}$$

Subject to

$$0 \le z_i \le 1, \quad (1 \le i \le d) \tag{22}$$

$$z_i \ge z_{i+1}, \quad (1 \le i \le d-1) \tag{23}$$

$$r \le \sum_{k=1}^{d} z_k \le r+2 \leftrightarrow 0 \le \alpha \le 1 \tag{24}$$

$$\sum_{k=1}^{r+1} z_k - \sum_{k=r+2}^{d} z_k \le r, \quad (r+2 \le d) \leftrightarrow (20). \tag{25}$$

We note (25) is valid only in the case $r+2 \le d$ and in this case we can express (25) by

$$\boldsymbol{f}_r^T \boldsymbol{z} \le r \tag{26}$$

where $\boldsymbol{f}_r^T = (1, \ldots, 1, -1, \ldots, -1)$.

Following lemma is well-known.

Lemma 8: Assume f, g_i, $(1 \le i \le m)$ are all convex functions on $\mathbb{R}^d$. Define the minimization problem:

$$\text{(CP) Minimize} \quad f(x) \tag{27}$$

$$\text{Subject to} \quad g_i(x) \le 0, \ (1 \le i \le m), \quad x \in \mathbb{R}^d.$$

Suppose $\bar{x}$ satisfies Karush-Kuhn-Tucker (KKT) conditions. Then $\bar{x}$ is a (global) minimizer of problem (CP).

In the case of problem (P), f is a strictly convex function on $\mathbb{R}^d$, thus we obtain:

Lemma 9: Assume $\bar{x}$ satisfies KKT condition for problem (P). Then $\bar{x}$ is a unique (global) minimizer of problem (P).

B. Varidity of Algorithm 3

Since $r = \lfloor \|\hat{\boldsymbol{z}}\|_1 \rfloor_{\text{even}}$ (step 2), we have

$$r \leq d. \tag{28}$$

As noted (26) is valid only in the case $r+2 \leq d$, we consider the cases $r = d$ or $r+1 = d$ for problem (P). For these cases, from the definition of r it holds

$$r \leq \sum_{k=1}^{d} \hat{z}_k \leq r+2.$$

Since $v_1 \geq v_2 \geq \cdots \geq v_d$, $\hat{z}_k$ $(1 \leq k \leq d)$ keeps this order. Thus

$$\hat{z}_{k+1} - \hat{z}_k \geq 0, \quad (1 \leq k \leq d-1).$$

From the definition $0 \leq \hat{z}_k \leq 1$, $(1 \leq k \leq d)$. Hence $\hat{\boldsymbol{z}}$ is a minimizer of problem (P) under constraint conditions (22), (23) and (24); (see step 4 and 5).

Here after, we fix $r+2 \leq d$. First we consider the case $\boldsymbol{f}_r^T \hat{\boldsymbol{z}} \leq r$. As in the case $r = d$ or $r+1 = d$, $\hat{\boldsymbol{z}}$ satisfies conditions (22), (23) and (24). Moreover, form the assumption, (25) is satisfied. Hence $\hat{\boldsymbol{z}}$ is a minimizer of problem (P); (see step 7 and 8).

Hence we can assume

$$\boldsymbol{f}_r^T \hat{\boldsymbol{z}} > r. \tag{29}$$

The KKT condition for the problem (P) is described as: there exists $\beta, \boldsymbol{\nu}, \boldsymbol{\eta}, \xi, \boldsymbol{\theta}$ and ζ such that following conditions are satisfied.

$$\begin{aligned}
\boldsymbol{z} &= \boldsymbol{v} - \beta \boldsymbol{f}_r - \boldsymbol{\nu} + \boldsymbol{\eta} - (\xi - \zeta)\mathbf{1} + S^T \boldsymbol{\theta}, && (30)\\
&0 \leq \beta \perp \boldsymbol{f}_r^T \boldsymbol{z} - r \leq 0, && (31)\\
&\mathbf{0} \leq \boldsymbol{\nu} \perp \boldsymbol{z} \leq \mathbf{1}, && (32)\\
&\mathbf{0} \leq \boldsymbol{\eta} \perp \boldsymbol{z} \geq \mathbf{0}, && (33)\\
&\boldsymbol{\theta} \perp S\boldsymbol{z} \geq 0, && (34)\\
&0 \leq \xi \perp \mathbf{1}^T \boldsymbol{z} - r - 2 \leq 0, && (35)\\
&0 \leq \zeta \perp \mathbf{1}^T \boldsymbol{z} - r \geq 0, && (36)
\end{aligned}$$

then $\boldsymbol{z}$ is a unique minimizer of (P). Here $\perp$ means for example for (32) $\nu_i(1 - z_i) = 0$, $(1 \leq i \leq d)$ and

$$S = \begin{bmatrix} 1 & -1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & & & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & -1 \end{bmatrix}.$$

We note the uniqueness of the minimizer follows from Lemma 8.

In the following we see that such $\beta, \boldsymbol{\nu}, \boldsymbol{\eta}, \xi, \boldsymbol{\theta}$ and ζ exists. For fixed $\beta > 0$ we decide ν_i, $(1 \leq i \leq d)$ as follows:

$$\nu_i = \begin{cases} 0, & v_i - \beta(\boldsymbol{f}_r)_i \leq 1 \\ v_i - \beta((\boldsymbol{f}_r)_i) - 1, & v_i - \beta(\boldsymbol{f}_r)_i > 1. \end{cases} \tag{37}$$

Similarly, η_i is decided as follows:

$$\eta_i = \begin{cases} 0, & v_i - \beta(\boldsymbol{f}_r)_i \geq 0 \\ -v_i + \beta(\boldsymbol{f}_r)_i, & v_i - \beta(\boldsymbol{f}_r)_i < 0. \end{cases} \tag{38}$$

Moreover, we fix $\xi = \zeta = 0$ and $\boldsymbol{\theta} = \mathbf{0}$. From these, $\boldsymbol{z}$, which is the LHS of (30) is regarded as a function of β and expressed as

$$\boldsymbol{z}(\beta) = \Pi_{[0,1]^d} \left(\boldsymbol{v} - \beta \boldsymbol{f}_r \right). \tag{39}$$

Hence, from (37), condition (32) is satisfied, also form (38), condition (33) is satisfied. As Step 10, range of β is defined as

$$0 \leq \beta \leq \beta_{\max} = \frac{1}{2}\left(v_{r+1} - v_{r+2} \right).$$

We note following holds:

$$\beta \leq \beta_{\max} \longleftrightarrow v_{r+2} + \beta \leq v_{r+1} - \beta.$$

From

$$\begin{aligned}
(\boldsymbol{z}(\beta_{\max}))_{r+1} &= \Pi_{[0,1]} \left(v_{r+1} - \beta_{\max} \right) \\
&= \Pi_{[0,1]} \left(\frac{1}{2}(v_{r+1} + v_{r+2}) \right)
\end{aligned}$$

and

$$\begin{aligned}
(\boldsymbol{z}(\beta_{\max}))_{r+2} &= \Pi_{[0,1]} \left(v_{r+2} + \beta_{\max} \right) \\
&= \Pi_{[0,1]} \left(\frac{1}{2}(v_{r+1} + v_{r+2}) \right) = (\boldsymbol{z}(\beta_{\max}))_{r+1},
\end{aligned}$$

we have

$$\begin{aligned}
\boldsymbol{f}_r \boldsymbol{z}(\beta_{\max}) &= \sum_{i=1}^{r} (\boldsymbol{z}(\beta_{\max}))_i + (\boldsymbol{z}(\beta_{\max}))_{r+1} \qquad (40)\\
&\quad - (\boldsymbol{z}(\beta_{\max}))_{r+2} - \sum_{i=r+3}^{d} (\boldsymbol{z}(\beta_{\max}))_i \\
&= \sum_{i=1}^{r} (\boldsymbol{z}(\beta_{\max}))_i - \sum_{i=r+3}^{d} (\boldsymbol{z}(\beta_{\max}))_i \\
&\leq r - \sum_{i=r+3}^{d} (\boldsymbol{z}(\beta_{\max}))_i \leq r.
\end{aligned}$$

Since from (29)

$$\boldsymbol{f}_r^T \boldsymbol{z}(0) = \boldsymbol{f}_r^T \hat{\boldsymbol{z}} > r,$$

there exists β_{opt} $(0 < \beta_{\text{opt}} \leq \beta_{\max})$ such that

$$\boldsymbol{f}_r^T \boldsymbol{z}(\beta_{\text{opt}}) = r. \tag{41}$$

We will show that $\boldsymbol{z}(\beta_{\text{opt}})$ satisfies KKT condition (recall that conditions (32) and (33) are already satisfied). From (41), the condition (31) is satisfied.

To see (34), $S\boldsymbol{z}(\beta_{\text{opt}}) \geq 0$ is enough (since $\boldsymbol{\theta} = 0$, $\perp$ relation is satisfied). From the assumption (Step 1), for non-negative β it holds

$$v_1 - \beta \geq v_2 - \beta \geq \cdots \geq v_{r+1} - \beta$$

and

$$v_{r+2} + \beta \geq v_{r+3} + \beta \geq \cdots \geq v_d + \beta.$$

Since

$$v_{r+1} - \beta \geq v_{r+2} + \beta \longleftrightarrow \beta \leq \frac{1}{2}\left(v_{r+1} - v_{r+2} \right) = \beta_{\max},$$

we have

$$(\boldsymbol{z}(\beta_{\text{opt}}))_1 \geq (\boldsymbol{z}(\beta_{\text{opt}}))_2 \geq \cdots \geq (\boldsymbol{z}(\beta_{\text{opt}}))_{r+1} \geq (\boldsymbol{z}(\beta_{\text{opt}}))_{r+2} \geq \cdots \geq (\boldsymbol{z}(\beta_{\text{opt}}))_d.$$

Thus, we have shown (34).

To see (35) and (36)

$$r \leq \sum_{i=1}^{d} (\boldsymbol{z}(\beta_{\text{opt}}))_i \leq r+2 \tag{42}$$

is enough. From (41),

$$r \leq r+2 \sum_{i=r+2}^{d} (\boldsymbol{z}(\beta_{\text{opt}}))_i = \sum_{i=1}^{d} (\boldsymbol{z}(\beta_{\text{opt}}))_i.$$

Also from (41),

$$\sum_{i=r+2}^{d} (\boldsymbol{z}(\beta_{\text{opt}}))_i = \sum_{i=1}^{r+1} (\boldsymbol{z}(\beta_{\text{opt}}))_i - r \leq r+1-r = 1,$$

we obtain

$$\sum_{i=1}^{d} (\boldsymbol{z}(\beta_{\text{opt}}))_i = \sum_{i=1}^{r+1} (\boldsymbol{z}(\beta_{\text{opt}}))_i + \sum_{i=r+2}^{d} (\boldsymbol{z}(\beta_{\text{opt}}))_i \leq r+1+1 = r+2.$$

Hence, we have shown conditions (35) and (36). Moreover, from Lemma 8, β_{opt} is unique. Summarizing the argument, we have shown that $\boldsymbol{z}(\beta_{\text{opt}})$ is the unique minimizer of the problem (P).

C. Searching Method for β_{opt}

Since β_{opt} is unique, bi-section method is effective. Define

$$\varphi(\beta) = \boldsymbol{f}_r \boldsymbol{z}(\beta).$$

We can observe that φ is continuous and piecewise linear function. The possible non-differentiable points are such β, satisfying $v_i - \beta = 1$ or $v_i - \beta = 0$ or $v_i + \beta = 1$ or $v_i + \beta = 0$. We divide set of these possible non-differentiable points in to four sets:

$$\mathcal{E}_1 = \{v_i - 1 \,|\, 1 \leq i \leq r+1\},$$
$$\mathcal{L}_1 = \{v_i \,|\, 1 \leq i \leq r+1\},$$
$$\mathcal{E}_2 = \{1 - v_i \,|\, r+2 \leq i \leq d\},$$
$$\mathcal{L}_2 = \{-v_i \,|\, 1 \leq r+2 \leq d\}$$

and put

$$\hat{\boldsymbol{\beta}} = \mathcal{E}_1 \cup \mathcal{L}_1 \cup \mathcal{E}_2 \cup \mathcal{L}_2.$$

We set

$$\tilde{\boldsymbol{\beta}} \leftarrow \left\{ \tilde{\beta} \in \hat{\boldsymbol{\beta}} \cup \{0\} \cup \{\beta_{\max}\} \,|\, 0 \leq \tilde{\beta} \leq \beta_{\max} \right\}$$

and sort $\hat{\boldsymbol{\beta}}$ to $\boldsymbol{\beta}$ in ascending order. Hence $\boldsymbol{\beta} = \{\beta_1, \beta_2, \dots \beta_p\}$ satisfies $0 = \beta_0 < \beta_1 < \cdots < \beta_p = \beta_{\max}$ (Step 12 and 13). We can find β_{opt} effectively with bi-section method by finding points β_j and β_{j+1} satisfying $\sum_{i=1}^{s} (\boldsymbol{f}_r)_i (\boldsymbol{z}(\beta_j))_i > r$ and $\sum_{i=1}^{s} (\boldsymbol{f}_r)_i (\boldsymbol{z}(\beta_{j+1}))_i < r$ $(0 \leq j \leq p-1)$.

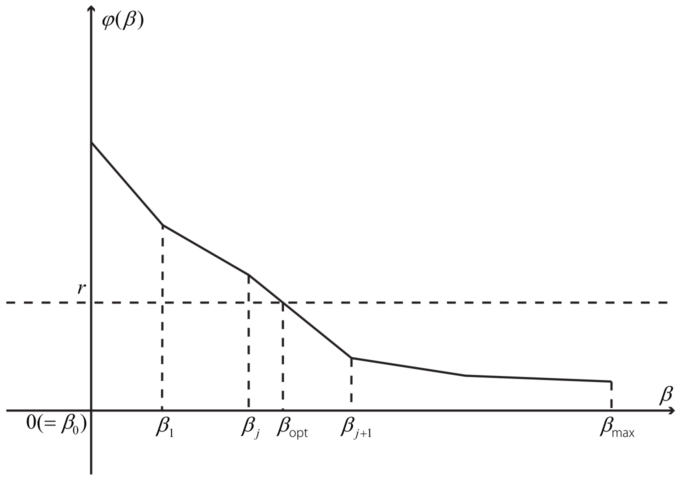

Fig. 13. The graph of φ.

Acknowledgment

The authors are grateful to the referees for their careful reading and invaluable comments. They would also like to thank Prof. Koji Nuida (Kyushu University) for the suggestion that they investigate weak keys.

References

[1] N. Aydin, B. Yildiz, and S. Uludag, "A class of weak keys for the QC-MDPC cryptosystem," in *Proc. Algebr. Combinat. Coding Theory (ACCT)*, Oct. 2020, pp. 1–4.

[2] M. Baldi, *QC-LDPC Code-Based Cryptography*. Cham, Switzerland: Springer 2014.

[3] M. Bardet, V. Dragoi, J. G. Luque, and A. Otmani, "Weak keys for the quasi-cyclic MDPC public key encryption scheme," in *Progress in Cryptology—AFRICACRYPT 2016* (Lecture Notes in Computer Science), vol. 9646, D. Pointcheval, A. Nitaj, and T. Rachidi, Eds. Heidelberg, Germany: Springer, 2016.

[4] S. Barman, X. Liu, S. C. Draper, and B. Recht, "Decomposition methods for large scale LP decoding," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7870–7886, Dec. 2013.

[5] E. Berlekamp, R. McEliece, and H. van Tilborg, "On the inherent intractability of certain coding problems (Corresp.)," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 3, pp. 384–386, May 1978.

[6] *BIKE (Bit Flipping Key Encapsulation)*. Accessed: 2022. [Online]. Available: https://bikesuite.org

[7] I. E. Bocharova, T. Johansson, and B. D. Kudryashov, "Improved iterative decoding of QC-MDPC codes in the McEliece public key cryptosystem," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1882–1886.

[8] N. Drucker, S. Gueron, and D. Kostic, "On constant-time QC-MDPC decoders with negligible failure rate," in *Code-Based Cryptography* (Lecture Notes in Computer Science), vol. 12087, M. Baldi, E. Persichetti, and P. Santini, Eds. Heidelberg, Germany: Springer, 2020.

[9] N. Drucker, S. Gueron, and D. Kostic, "QC-MDPC decoders with several shades of gray," in *Post-Quantum Cryptography* (Lecture Notes in Computer Science), vol. 12100, J. Ding and J. P. Tillich, Eds. Heidelberg, Germany: Springer, 2020.

[10] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 954–972, Mar. 2005.

[11] Q. Guo, T. Johansson, and P. Stankovski Wagner, "A key recovery reaction attack on QC-MDPC," *IEEE Trans. Inf. Theory*, vol. 65, no. 3, pp. 1845–1861, Mar. 2019.

[12] *Gurobi Optimizer*. Accessed: 2023. [Online]. Available: https://www.gurobi.com/

[13] H. Kaneko, "Look-ahead bit-flipping decoding of MDPC code," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 2922–2927.

[14] K. Kurosawa and K. Inaba, "Consideration on the NP completeness of linear codes," *IEICE Trans.*, vol. 10, pp. 953–956, Sep. 1985.

[15] X. Liu and S. C. Draper, “The ADMM penalized decoder for LDPC codes,” *IEEE Trans. Inf. Theory*, vol. 62, no. 6, pp. 2966–2984, Jun. 2016.

[16] R. Misoczki, J.-P. Tillich, N. Sendrier, and P. S. L. M. Barreto, “MDPC-McEliece: New McEliece variants from moderate density parity-check codes,” in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 2069–2073.

[17] A. Nilsson, I. E. Bocharova, B. D. Kudryashov, and T. Johansson, “A weighted bit flipping decoder for QC-MDPC-based cryptosystems,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 1266–1271.

[18] M. R. Nosouhi et al., “Weak-key analysis for BIKE post-quantum key encapsulation mechanism,” *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2160–2174, 2023.

[19] P. Santini, M. Battaglioni, M. Baldi, and F. Chiaraluce, “Analysis of the error correction capability of LDPC and MDPC codes under parallel bit-flipping decoding and application to cryptography,” *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4648–4660, Aug. 2020.

[20] N. Sendrier and V. Vasseur, “About low DFR for QC-MDPC decoding,” in *Proc. Int. Conf. Post-Quantum Cryptogr.*, in Lecture Notes in Computer Science, vol. 12100, 2020, pp. 20–34.

[21] N. Sendrier and V. Vasseur, “On the existence of weak keys for QC-MDPC decoding,” Cryptol. ePrint Arch., Rep. 2020/1232, 2020.

[22] J.-P. Tillich, “The decoding failure probability of MDPC codes,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 941–945.

[23] V. Vasseur, “QC-MDPC codes DFR and the IND-CCA security of BIKE,” Cryptol. ePrint Arch., Rep. 2021/1458, 2021.

[24] *QC-MDPC Decoder*. Accessed: 2022. [Online]. Available: https://github.com/vvasseur/qcmdpc_decoder

[25] P. W. Shor, “Algorithms for quantum computation: Discrete logarithms and factoring,” in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, 1994, pp. 124–134.

[26] P. Santini, E. Persichetti, and M. Baldi, “Reproducible families of codes and cryptographic applications,” *J. Math. Cryptol.*, vol. 16, no. 1, pp. 20–48, Sep. 2021.

[27] J. Stern, “A method for finding codewords of small weight,” in *Coding Theory and Applications* (Lecture Notes in Computer Science), vol. 388. Heidelberg, Germany: Springer, 1989.

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

[29] *HQC (Hamming Quasi-Cyclic)*. Accessed: 2023. [Online]. Available: https://pqc-hqc.org

[30] *Post-Quantum Cryptography*. Accessed: 2023. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/

Kohtaro Watanabe was born in Kanagawa, Japan, in 1965. He received the B.S., M.S., and Ph.D. degrees in mathematics from the Tokyo Institute of Technology in 1989, 1991, and 2004, respectively. Since 2014, he has been a Professor with the Department of Computer Science, National Defense Academy, Japan. His research interests include non-linear ordinary and partial differential equations and information theory.

Motonari Ohtsuka was born in Osaka, Japan, in 1996. He received the B.S. degree in computer science from the National Defense Academy, Japan, in 2019, where he is currently pursuing the M.S. degree in mathematics and computer science.

Yuta Tsukie was born in Saitama, Japan, in 1988. He received the B.S. degree in electrical engineering from the Tokyo University of Science in 2017 and the M.S. degree in mathematics and computer science from the National Defense Academy, Japan, in 2023.