BF-Max: an Efficient Bit Flipping Decoder with Predictable Decoding Failure Rate

Alessio Baldelli, Marco Baldi, Franco Chiaraluce and Paolo Santini
Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Ancona, Italy
a.baldelli@pm.univpm.it, {m.baldi, f.chiaraluce, p.santini}@univpm.it

***Abstract*—The Bit-Flipping (BF) decoder, thanks to its very low computational complexity, is widely employed in post-quantum cryptographic schemes based on Moderate Density Parity Check codes in which, ultimately, decryption boils down to syndrome decoding. In such a setting, for security concerns, one must guarantee that the Decoding Failure Rate (DFR) is negligible. Such a condition, however, is very difficult to guarantee, because simulations are of little help and the decoder performance is difficult to model theoretically. In this paper, we introduce a new version of the BF decoder, that we call BF-Max, characterized by the fact that in each iteration only one bit (the least reliable) is flipped. When the number of iterations is equal to the number of errors to be corrected, we are able to develop a theoretical characterization of the DFR that tightly matches with numerical simulations. We also show how BF-Max can be implemented efficiently, achieving low complexity and making it inherently constant time. With our modeling, we are able to accurately predict values of DFR that are remarkably lower than those estimated by applying other approaches.**

***Index Terms*—Bit-flipping decoder, code-based cryptography, decoding failure rate, LDPC codes, MDPC codes.**

I. Introduction

The Bit-Flipping (BF) decoder is probably the simplest decoder for Low-Density Parity-Check (LDPC) codes [1]. Iterative hard-decision decoders of this family have experienced renewed interest in recent years because they are employed in post-quantum, code-based cryptosystems such as LEDAcrypt [2] and BIKE [3]. In such schemes, decryption requires decoding a syndrome through a Moderate-Density Parity-Check (MPDC) code[1]. In these and other applications, BF decoding often is the preferred choice, thanks to its good trade-off between good error correction, low computational complexity and implementation simplicity. However, as typical in iterative decoding algorithms of this kind, BF is characterized by some nonzero decoding failure rate (DFR), that is, some nonzero probability that decoding fails even for genuine ciphertexts, and decryption failures may leak information about the secret key [4]. To avoid such a leakage (formally, to achieve Indistinguishability under Adaptively Chosen Ciphertext Attacks (IND-CCA2)), the DFR must be less than $2^{-\lambda}$, with λ being the security parameter (e.g., $\lambda = 128$) [5]. Clearly, such values of DFR cannot be estimated by simulations.

For this reason, in the last few years research has focused on the development of theoretical models to assess the DFR of BF decoders [6]–[12]. However, all these works are limited by the need for considering decoders that are somewhat non-optimal, e.g., employ a very simple flipping criterion (e.g., majority logic decoding [6]) or a very small number of iterations, say, one as in [6], [8] or two [9]. Indeed, the lowest DFR is achieved when the decoder runs through many iterations, so that erroneous decisions made by the decoder in the earlier iterations can be revised and corrected in subsequent iterations. However, this is also what makes estimating the DFR a rather difficult task, because one cannot assume that the errors to be corrected are uniformly distributed at each iteration, and should instead devise a theoretical model that keeps track of their correlation throughout decoding iterations. To the best of our knowledge, the state of the art in this line of research is the analysis in [9], where the authors derive upper bounds for a two-iteration, *out-of-place*[2] BF decoder.

Our contribution: In this paper we introduce and analyze BF-Max, a BF decoder that flips only one bit per iteration, namely, the one having the lowest reliability. The intuition is that, by doing this, the decoder reduces as much as possible the odds that wrong flips happen. Under standard heuristic assumptions, we are able to derive a closed form formula for the DFR when the number of iterations equals the number of errors. Numerical simulations show that the DFR formula we provide is tight and reliable. Moreover, we describe a strategy to achieve an efficient implementation of the BF-Max decoder. On the negative side, our theoretical model for BF-Max does not take into account the case in which the number of iterations is greater than the number of errors. Hence, we trade some error correction capability for a closed-

This work was partially supported by Agenzia per la Cybersicurezza Nazionale (ACN) under the programme for promotion of XL cycle PhD research in cybersecurity (CUP I32B24001750005), by the Italian Ministry of University and Research (MUR) under the PRIN 2022 program with projects "Mathematical Primitives for Post Quantum Digital Signatures" (CUP I53D23006580001) and "Post quantum Identification and eNcryption primiTives: dEsign and Realization (POINTER)" (CUP I53D23003670006), by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan, funded by the European Union - Next Generation EU and by MUR under the Italian Fund for Applied Science (FISA 2022), Call for tender No. 1405 published on 13-09-2022 - project title "Quantum-safe cryptographic tools for the protection of national data and information technology assets" (QSAFEIT) - No. FISA 2022-00618 (CUP I33C24000520001), Grant Assignment Decree no. 15461 adopted on 02.08.2024. The views expressed are those of the authors and do not represent the funding institutions

[1] An MDPC code can be thought of as an LDPC code with a somewhat denser parity-check matrix. This is necessary to prevent attacks (such as key recoveries and distinguishers) that exploit the sparsity in the secret key.

[2] By out-of-place decoder we refer to a decoder that, in each iteration, first flips all the bits it has to flip, and updates the syndrome only after that, before starting the next iteration. This is opposed to an in-place decoder, which instead updates the syndrome after each bit flip.

form, simple and reliable formula for estimating the DFR. Still, even with this suboptimal setting, BF-Max outperforms significantly the two-iteration BF decoder studied in [9]: for all parameters considered in [9], our decoder has comparable time complexity and achieves a significantly lower DFR. We remark that, by flipping only one bit at each iteration, BF-Max is inherently constant-time. This, together with the possibility of DFR prediction and its good error correction capability, makes BF-Max an ideal candidate for post-quantum cryptographic schemes based on LDPC and MDPC codes.

II. Notation and background

We denote by $\mathbb{F}_2$ the binary field. For clarity, we use different operators for the sum: $+$ and $\sum$ when summing over the reals, and $\oplus$ when summing over $\mathbb{F}_2$. Vectors (resp., matrices) over $\mathbb{F}_2$ are denoted with bold lowercase (resp., uppercase) letters. The null vector of length r is indicated as $\mathbf{0}_r$. For a vector $\mathbf{a} = (a_1, \cdots, a_n)$, we indicate its support as $\mathrm{Supp}(\mathbf{a})$. The Hamming weight of $\mathbf{a}$ corresponds to the number of its non-zero entries, and is denoted as $\mathrm{wt}(\mathbf{a})$. The set of vectors with length n and weight u is indicated as $\mathcal{S}_{n,u}$. For a set A, $x \xleftarrow{\$} A$ indicates that x is sampled uniformly at random from A.

A. LDPC codes

A *linear code* with length n and redundancy $r < n$ is a linear subspace of $\mathbb{F}_2^n$ with dimension $n - r$. A code can be represented with a full-rank *parity-check matrix* $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, so that the code is the space of all vectors $\mathbf{c} \in \mathbb{F}_2^n$ for which $\mathbf{c}\mathbf{H}^\top = \mathbf{0}_r$, where $^\top$ denotes transposition. Given a vector $\mathbf{e} \in \mathbb{F}_2^n$, its *syndrome* is $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$. If $\mathbf{s} \neq \mathbf{0}_r$, then $\mathbf{e}$ is not a codeword; (syndrome) decoding consists in reconstructing $\mathbf{e}$ from the pair $\{\mathbf{H}, \mathbf{s}\}$. In this paper, we are concerned with LDPC codes, which are characterized by parity-check matrices having low density, i.e., the number of zero entries in $\mathbf{H}$ is much less than rn. Among these codes, we focus on those characterized by parity-check matrices having a constant amount v of set entries in each column.

B. Bit-Flipping decoding

The BF decoder works by iteratively building an estimate for the error vector; such an estimate is initially null and gets refined through an iterative process. In this process, a key role is played by *counters*: the i-th counter, noted by σ_i, corresponds to the number of unsatisfied parity-check equations in which the i-th coordinate participates, namely,

$$\sigma_i = |\{j \in \{1, \cdots, r\} : (h_{j,i} = 1) \wedge (s_j = 1)\}|,$$

where $h_{j,i}$ denotes the entry of $\mathbf{H}$ in the j-th row and i-th column. Starting from the computation of counters, it is easy to define the *out-of-place* BF decoding algorithm, which is described in the form of pseudocode in Algorithm 1.

We summarize the BF operating principle assuming it receives, as input, a syndrome $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$ with $\mathbf{e} \in \mathcal{S}_{n,t}$ and t being properly low. The decoder either outputs an estimate $\widehat{\mathbf{e}} \in \mathbb{F}_2^n$ for the error vector, or $\perp$ whenever decoding fails.

Algorithm 1: BF decoder

Data: parity-check matrix $\mathbf{H} \in \mathbb{F}_2^{r \times n}$, maximum number of iterations $\mathtt{IterMax} \in \mathbb{N}$, thresholds $(b_1, \cdots, b_{\mathtt{IterMax}})$
Input: syndrome $\mathbf{s} \in \mathbb{F}_2^r$
Output: decoding failure $\perp$, or vector $\widehat{\mathbf{e}} \in \mathbb{F}_2^n$ such that $\mathrm{wt}(\widehat{\mathbf{e}}) \leqslant \mathtt{IterMax}$ and $\mathbf{s} = \widehat{\mathbf{e}}\mathbf{H}^\top$

```
/* Initialize error estimate and number of performed iterations */
Set ê = 0_n,  Iter = 1;
while (s ≠ 0_r) ∨ (Iter ⩽ IterMax) do
    Compute counters (σ_1, ··· , σ_n);
    /* Update syndrome and error estimate */
    for i such that σ_i ⩾ b_Iter do
        s ← s ⊕ h_i;// h_i: i-th column of H
        ê_i ← ê_i ⊕ 1;
    Iter ← Iter + 1;
/* Output error estimate or report failure */
if s = 0_r, return ê; else, return ⊥;
```

In each iteration, the decoder receives as input the current estimate for the error (which is initially set as the null vector $\mathbf{0}_n$) and the syndrome; inside the iteration, the estimate is refined and the syndrome is updated accordingly. Decisions are taken on the basis of the counters: whenever a counter σ_i is high enough (as large as the threshold value for the iteration), it is very likely that the corresponding entry $\widehat{e}_i$ of the error estimate $\widehat{\mathbf{e}}$ is wrong and thus gets flipped. The syndrome is updated accordingly by summing the i-th column of $\mathbf{H}$. After all updates have been applied, the resulting syndrome is $(\mathbf{e} \oplus \widehat{\mathbf{e}})\mathbf{H}^\top$: if $\mathbf{e} \oplus \widehat{\mathbf{e}}$ is a codeword[3], then the syndrome is null and decoding stops, otherwise, another BF iteration starts. If, after the maximum number of iterations has been reached, the syndrome is still not null, then $\widehat{\mathbf{e}}$ is guaranteed to be wrong and failure is reported. If, instead, the syndrome is null, then the decoder may have found the true error vector: since the decoder is expected to perform a small number of bit flips, the weight of $\widehat{\mathbf{e}}$ is expected to be low. Hence, it is very likely that indeed $\widehat{\mathbf{e}} = \mathbf{e}$.

The term *out-of-place* stands for the fact that the syndrome is updated and the counters are recomputed only after all the bit flips of each iteration have been executed; actually, other strategies may be pursued. Moreover, there exist variants of BF in which the flipping thresholds are not constant (see e.g. [10]). In this paper, we stick to the choice of constant thresholds, since this is the variant analyzed in [9].

Computational complexity: In each iteration, the time complexity is dominated by the cost of computing counters and comparing them with the flipping threshold; exploiting sparsity, which holds, although at different levels, for both LDPC and MDPC codes, this can be done in $O\big(n \cdot (1+v) \cdot \log_2(v)\big)$ operations (here, $\log_2(v)$ accounts for the fact that counters take values in $[0; v]$). Each bit flip requires $O(1+v)$ operations (one for the error update, v for the syndrome

[3] We remind that any linear code includes the all-zero vector among its codewords.

update). Assuming the number of flips is more or less equal to the number of errors to be corrected, we get that, on average, the time complexity of BF is in

$$O\big(\texttt{IterMax} \cdot n \cdot (v+1) \cdot \log_2(v) + t \cdot (1+v)\big),$$

where, according to Algorithm 1, IterMax represents the maximum number of iterations. For typical MDPC code parameters, IterMax is a small constant while both t and v grow as $\sqrt{n}$, resulting in an average complexity $O\big(n^{1.5} \cdot \log_2(n)\big)$.

III. BF-MAX: EFFICIENT IMPLEMENTATION

In this section we introduce the BF-Max decoder and describe how it can be implemented efficiently.

A. Main intuition

The BF variant we call BF-Max works by flipping a unique bit in each decoding iteration; this bit is selected as the one having the highest counter value among all bits (the case of more counters simultaneously having the highest value is discussed afterwards). Remember that, as a rule of thumb, the larger a counter, the higher the probability that the associated bit is wrongly estimated. By flipping a unique bit, one of those with the largest counter, we "guarantee" that the probability of flipping a wrong bit is reduced to its minimum[4]. After the bit is selected, both the error estimate and the syndrome get updated: counters are recomputed and a new iteration starts.

As a little technical caveat, one has to deal with the case in which there are more bits having the largest counter value. Many strategies are possible; in this paper, we deal with this case by selecting one of such positions at random.

A straightforward implementation of the BF-Max decoder would recompute all counters in each iteration. This would lead to a cost of $\texttt{IterMax} \cdot \big(n \cdot v \cdot \log_2(v) + v + 1\big)$ operations. When there are t errors, we must set $\texttt{IterMax} \geqslant t$, which would result in a rather large cost: for typical MDPC code parameters, this would lead to a cost $O\big(n^2 \cdot \log_2(n)\big)$. In the next section, we describe how one can instead implement BF-Max with a much lower complexity by exploiting sparsity: for typical MDPC code parameters, we obtain a complexity $O\big(n^{1.5} \cdot \log_2(n)\big)$, which is in the same order of the complexity of the out-of-place BF.

B. Implementation of BF-Max

We now show how the computational cost of BF-Max can be significantly decreased through an efficient implementation. The main intuition, here, lies in the observation that, since BF-Max flips only one bit in each iteration, the number of counters that change, with respect to the ones from the previous iteration, is much less than n. Indeed, every bit participates in exactly v parity-check equations and, in each equation, we have on average $\overline{w} = v \cdot n/r$ set entries. Hence, on average, the number of operations required for updating counters (after one bit flip) is $v \cdot \overline{w} \cdot \log_2(v)$. Again, considering typical MDPC code parameters, this results in a cost of $O\big(n \cdot \log_2(n)\big)$ operations. Repeating this for approximately $t = O(\sqrt{n})$ iterations, we get an overall cost of $O\big(n^{1.5} \cdot \log_2(n)\big)$ operations.

Full details about how BF-Max can be implemented with this strategy are given in Algorithm 2. Counters are computed at the beginning of the process (lines 2–3) and then get updated at the end of each iteration (lines 12–14). This is done by considering, for each parity-check equation that changes its value due to the syndrome update (line 11), only the counters corresponding to indices that are in the support of the parity-check equation. The counter update is either $d = -1$, if the parity-check equation becomes satisfied, or $d = 1$, if the parity-check equation is unsatisfied.

The average time complexity of the algorithm is reported in the next proposition.

Proposition 3.1. *Let $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ have constant column weight v and average row weight $\overline{w}$. Then, Algorithm 2 runs in average time which is well approximated by*

$$n \cdot v \cdot \log_2(v) + \texttt{IterMax} \cdot \big(n \cdot \log_2(v) + 1 + v + v\overline{w}\big).$$

Proof: The term $n \cdot v \cdot \log_2(v)$ accounts for the initial counters computation. For each iteration, we neglect some costs (e.g., sampling i^* from C) and consider only the following costs: $n \cdot \log_2(v)$ for finding the maximum counter (lines 5–7), 1 for updating the error estimate and the syndrome (line 9), $v \cdot (1 + \overline{w})$ for counters and syndrome updates. Indeed, lines 11–14 are repeated for v times, line 11 takes 1 operation while lines 12–14 take on average $\overline{w}$ operations (since the average size of Z_j is $\overline{w}$). ∎

Remark 3.1. *A non-constant time implementation of a BF decoder would leak side channel information about the secret key [13], [14]. In particular, in the out-of-place BF, the number of bits that are flipped in each iteration is inherently not constant and needs to be properly masked (e.g., by performing a certain number of fake flips): this normally comes with some non-trivial complexity overhead. Instead, our decoder is inherently constant time. Indeed, when working with (v, w)-regular LDPC codes, every iteration flips a unique bit and, moreover, takes the same number $v \cdot w \cdot \log_2(v)$ of operations for updating the counters.*

IV. BF-MAX: MODELING THE DFR

We now describe how the DFR of the BF-Max decoder can be predicted, at least in the case in which the decoder performs a number of iterations exactly equal to the number of errors.

A. DFR prediction

We first model the probability distribution of the counter values, relying on the following assumption.

Assumption 1. *Each counter behaves as the sum of independent Bernoulli variables, all with the same parameter, which depends only on the value of the corresponding error bit (either 1 or 0).*

This assumption has been employed in many other papers about BF decoders, at least for what concerns the first decoding iteration (e.g., [6], [9], [13]), and is largely accepted

[4]The word guarantee is in quotes since, as we have stressed out, this holds only on the basis of a heuristic reasoning.

Algorithm 2: Efficient implementation of the BF-Max decoder exploiting sparsity

Data: columns supports $\{J_1, \cdots, J_n\}$, rows supports $\{Z_1, \cdots, Z_r\}$, maximum number of iterations $\texttt{IterMax} \in \mathbb{N}$
Input: syndrome $\mathbf{s} \in \mathbb{F}_2^r$
Output: decoding failure $\perp$, or vector $\widehat{\mathbf{e}} \in \mathbb{F}_2^n$ such that $\mathrm{wt}(\widehat{\mathbf{e}}) \leqslant \texttt{IterMax}$ and $\mathbf{s} = \widehat{\mathbf{e}}\mathbf{H}^\top$

Set $\widehat{\mathbf{e}} = \mathbf{0}_n$, $\texttt{Iter} = 1$;// Initialize error estimate and number of performed iterations
/* Initial computation of counters */
for $i = 1, \cdots, n$ **do**
  Compute $\sigma_i = \sum_{j \in J_i} s_j$// Integer sum of the syndrome bits indexed by J_i
while $\left(\mathbf{s} \neq \mathbf{0}_r\right) \vee \left(\texttt{Iter} \leqslant \texttt{IterMax}\right)$ **do**
  /* Find indices associated to maximum counter value $\tilde{\sigma}$; store all indices in list C */
  Set $C = \varnothing$, $\tilde{\sigma} = 0$;
  for $i = 1, \cdots, n$ **do**
    if $\sigma_i = \tilde{\sigma}$, update $C \leftarrow C \cup \{i\}$; **else**, overwrite $\tilde{\sigma} \leftarrow \sigma_i$ and $C \leftarrow \{i\}$;
  Set $i^* \xleftarrow{\$} C$;// Sample at random one of the position with maximum counter
  Update $\widehat{e}_{i^*} \leftarrow \widehat{e}_{i^*} \oplus 1$;
  /* Update syndrome and counters: consider only parity-check equations indexed by J_{i^*} (support of column i^*), for each equation $j \in J_{i^*}$ update only the counters indexed by Z_j (support of row j) */
  for $j \in J_{i^*}$ **do**
    Update $s_j \leftarrow s_j \oplus 1$;// Update of bit j of the syndrome
    if $s_j = 0$, set $d = -1$; **else** set $d = 1$;// d is the counter variation
    for $\ell \in Z_j$ **do**
      Update $\sigma_\ell \leftarrow \sigma_\ell + d$;// Counter update due to row j changing parity
  $\texttt{Iter} \leftarrow \texttt{Iter} + 1$;// Update number of performed iterations
/* Output error estimate or report failure */
if $\mathbf{s} = \mathbf{0}_r$, **return** $\widehat{\mathbf{e}}$; **else**, **return** $\perp$;

when the error positions are uncorrelated. It is instrumental in deriving the probability distribution for counters [7], [9], [10].

Proposition 4.1. *Let $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ with constant column weight v and row weight w. Let $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,u}$. Under Assumption 1, we have*

$$\Pr\left[\sigma_i = x \mid e_i = 1\right] = g_1^{(u)}(x) = \binom{v}{x} \rho_1^x (1-\rho_1)^{v-x},$$

$$\Pr\left[\sigma_i = x \mid e_i = 0\right] = g_0^{(u)}(x) = \binom{v}{x} \rho_0^x (1-\rho_0)^{v-x},$$

where

$$\rho_1 = \frac{1}{\binom{n-1}{w-1}} \cdot \sum_{\substack{\ell=0 \\ \ell \text{ even}}}^{\min\{w-1,u-1\}} \binom{u-1}{\ell} \binom{n-u}{w-1-\ell},$$

$$\rho_0 = \frac{1}{\binom{n-1}{w-1}} \cdot \sum_{\substack{\ell=1 \\ \ell \text{ odd}}}^{\min\{w-1,u\}} \binom{u}{\ell} \binom{n-1-u}{w-1-\ell}.$$

We model the DFR of BF-Max using the next assumption.

Assumption 2. *Assumption 1 holds for any iteration of BF-Max, if in the previous iterations no wrong bit flip has been performed.*

Let i^* be the bit which is flipped and initially assume $i^* \notin E$, where E is the support of the error vector. Then, after the first iteration, the vector to be corrected has support $E' = E \cup \{i^*\}$. While E is chosen uniformly at random among all subsets of $\{1, \cdots, n\}$ of size t, this is not true anymore for E': i^* is correlated with the positions in E (since the positions indexed by E caused its flip). If instead $i^* \in E$, then the error vector which must be corrected in the second iteration has support $E' = E \backslash \{i^*\}$. E' contains $t-1$ indices which are not correlated, since they have been chosen at the beginning, uniformly at random. The same reasoning can be repeated for all the subsequent iterations: after iteration $\texttt{Iter}$, if all the bit flips performed were correct, the number of residual errors is $t - \texttt{Iter}$ and their positions are uncorrelated. Thus, for all iterations, Proposition 4.1 is expected to yield valid approximations for the counter distributions. In fact, the rationale of considering a number of iterations exactly equal to the number of errors lies in the ability to separate the case in which all flips performed by the decoder are correct from the case in which at least one flipped bit was not affected by error. In the former case, in fact, the decoder succeeds, while in the latter case it fails, as it cannot perform more iterations than the number of errors to be corrected.

Proposition 4.2. *Let $\mathbf{H} \in \mathbb{F}_2^{r \times n}$ be a parity-check matrix with constant column weight v and constant row weight w. Let $\mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,t}$ and consider BF-Max on input $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$, with $\texttt{IterMax} = t$. Then, under Assumptions 1 and 2, the DFR of BF-Max is well approximated by $1 - \prod_{u=1}^{t} \sum_{x=0}^{v-1} f_1^{(u)}(x) \cdot f_0^{(u)}(x)$, where $f_1^{(u)}(x) = 1 - \left(\sum_{z=0}^{x} g_1^{(u)}(z)\right)^u$ and*

$$f_0^{(u)}(x) = \begin{cases} \left(g_0^{(u)}(0)\right)^{n-u} & \text{if } x = 0, \\ \left(\sum_{z=0}^{x} g_0^{(u)}(z)\right)^{n-u} - \left(\sum_{z=0}^{x-1} g_0^{(u)}(z)\right)^{n-u} & \text{if } x > 0. \end{cases}$$

Proof: Since $\texttt{IterMax} = t$, the decoder will not fail if and only if the only bit that flips at each iteration is actually affected by one error. Thus, after iteration $u \in \{1, \cdots, t\}$, the number of residual errors is $t - u$. Thanks to Assumption 2, we model the error to be corrected in iteration u as a uniformly random sample from $\mathcal{S}_{n,t+1-u}$. For iteration u, let $J_0^{(u)}$ and $J_1^{(u)}$ denote the sets of indices of bits in which $\widehat{\mathbf{e}}$ and $\mathbf{e}$ are, respectively, equal and different. Then, in iteration u, the decoder will surely take a good choice if

$$\underbrace{\max\left\{\sigma_j \;\middle|\; j \in J_0^{(u)}\right\}}_{\tilde{\sigma}_0^{(u)}} < \underbrace{\max\left\{\sigma_j \;\middle|\; j \in J_1^{(u)}\right\}}_{\tilde{\sigma}_1^{(u)}}.$$

Notice that the decoder can still take a good decision, with some probability, even when the above inequality is actually an equality. For the sake of simplicity, we neglect this possibility (in any case, we expect it happens with low probability).

Then, the DFR can be approximated as

$$\begin{aligned}
&1 - \prod_{u=1}^{t} \Pr\left[\tilde{\sigma}_0^{(u)} < \tilde{\sigma}_1^{(u)} \;\middle|\; \mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,t+1-u}\right] \\
&= 1 - \prod_{u=1}^{t} \Pr\left[\tilde{\sigma}_0^{(u)} < \tilde{\sigma}_1^{(u)} \;\middle|\; \mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,u}\right] \\
&= 1 - \prod_{u=1}^{t} \sum_{x=0}^{v-1} \Pr\left[(\tilde{\sigma}_0^{(u)} = x) \wedge (\tilde{\sigma}_1^{(u)} > x) \;\middle|\; \mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,u}\right].
\end{aligned}$$

Under Assumption 1, the counters are independent random variables, hence

$$\begin{aligned}
&\Pr\left[(\tilde{\sigma}_0^{(u)} = x) \wedge (\tilde{\sigma}_1^{(u)} > x) \;\middle|\; \mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,u}\right] \\
&\quad = \underbrace{\Pr\left[\tilde{\sigma}_0^{(u)} = x \;\middle|\; \mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,u}\right]}_{f_0^{(u)}(x)} \cdot \underbrace{\Pr\left[\tilde{\sigma}_1^{(u)} > x \;\middle|\; \mathbf{e} \xleftarrow{\$} \mathcal{S}_{n,u}\right]}_{f_1^{(u)}(x)}.
\end{aligned}$$

With further probability theory arguments and recalling Proposition 4.1, we obtain the expressions for $f_0^{(u)}(x)$ and $f_1^{(u)}(x)$. ∎

B. Numerical results

To validate Proposition 4.2, we can compare it with estimates obtained by Monte Carlo simulations. In Fig. 1, we compare the DFR of some quasi-cyclic (QC) codes predicted by Proposition 4.2, with the one estimated through numerical simulations [5]. As in BIKE and LEDAcrypt, we consider codes whose parity-check matrix is in the form $(\mathbf{H}_1, \mathbf{H}_2)$, with both $\mathbf{H}_1$ and $\mathbf{H}_2$ being circulant matrices with size $r \times r$ and column weight v. The resulting parity-check matrix has $n = 2r$ and constant row weight $w = 2v$. As can be seen from Fig. 1, the results of the simulations are aligned with the theoretical predictions. In Fig. 2 we compare the DFR resulting from Proposition 4.2 with the theoretical models for the BF decoder, considering one and two iterations [7], [9]. As we can see, our decoder has a much lower DFR. Moreover, all decoders have comparable computational complexity: for instance, for $v = 9$, the complexity of BF-Max is approximately 30% greater than that of the 2-iterations BF, while for $v = 17$ they have essentially the same complexity.

[5]The code employed for the simulations, as well as an implementation for the formula in Proposition 4.2, can be found at https://github.com/secomms/bf-max.

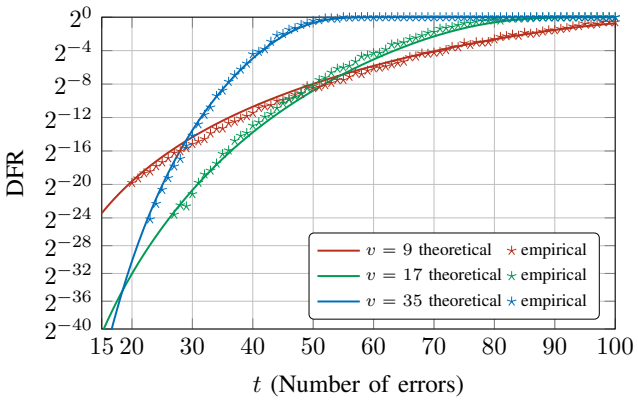

Fig. 1: Comparison between theoretical and empirical DFR of the BF-Max decoder, for QC-LDPC codes with fixed $r = 2003$, $n = 2r$, $w = 2v$ and several values of v.

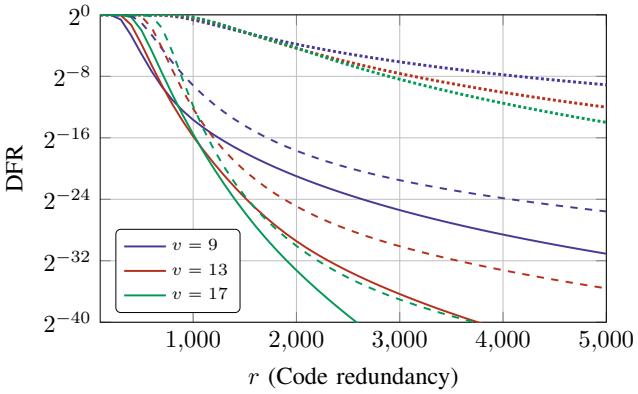

Fig. 2: Comparison between the DFR theoretical models for BF-Max (continuous lines), 2-iteration BF (dashed lines) and 1-iteration BF (dotted lines). For all curves, $t = 18$, $w = 2v$ and $n = 2r$; the flipping thresholds for BF have been optimized in order to achieve the lowest DFR. The same color has been used for curves referred to the same tuple (t, v, w).

V. Conclusion

We have introduced and analyzed the BF-Max decoder, a BF decoder that flips a unique bit in each iteration choosing that (or one of those) with the highest counter. We have shown that this decoder has low computational complexity and is inherently constant time. We have been able to characterize its DFR theoretically, for the case in which the number of iterations equals the number of errors to be corrected. The DFR predicted through our model closely matches the results of numerical simulations and significantly outperforms the theoretical DFR obtained by other models for BF decoders.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on information theory*, vol. 8, no. 1, pp. 21–28, 1962.

[2] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "LEDAcrypt," *Second round submission to the NIST post-quantum cryptography call*, 2019.

[3] N. Aragon, P. Barreto, S. Bettaieb, *et al.*, "BIKE," *First round submission to the NIST post-quantum cryptography call*, 2017.

[4] Q. Guo, T. Johansson, and P. Stankovski, "A key recovery attack on MDPC with CCA security using decoding errors," in *Proc. Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security*, Hanoi, Vietnam, Dec. 2016, pp. 789–815.

[5] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the Fujisaki-Okamoto transformation," in *Proc. 15th Theory of Cryptography Conference (TCC '17)*, Baltimore, MD, Nov. 2017, pp. 341–371.

[6] J.-P. Tillich, "The decoding failure probability of MDPC codes," in *Proc. 2018 IEEE International Symposium on Information Theory (ISIT)*, Vail, CO, Jun. 2018, pp. 941–945.

[7] P. Santini, M. Battaglioni, M. Baldi, and F. Chiaraluce, "Hard-decision iterative decoding of LDPC codes with bounded error rate," in *Proc. 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019.

[8] 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 Transactions on Communications*, vol. 68, no. 8, pp. 4648–4660, 2020.

[9] A. Annechini, A. Barenghi, and G. Pelosi, "Bit-flipping decoder failure rate estimation for (v,w)-regular codes," in *Proc. 2024 IEEE International Symposium on Information Theory (ISIT)*, Athens, Greece, Jul. 2024, pp. 3375–3379.

[10] N. Sendrier and V. Vasseur, "On the decoding failure rate of QC-MDPC bit-flipping decoders," in *Post-Quantum Cryptography - 10th International Conference (PQCrypto 2019) Revised Selected Papers*, Chongqing, China, May 2019, pp. 404–416.

[11] N. Sendrier and V. Vasseur, "About low DFR for QC-MDPC decoding," in *Proc. International Conference on Post-Quantum Cryptography*, Paris, France, Sep. 2020, pp. 20–34.

[12] M. Baldi, A. Barenghi, F. Chiaraluce, G. Pelosi, and P. Santini, "Analysis of in-place randomized bit-flipping decoders for the design of LDPC and MDPC code-based cryptosystems," in *Proc. 17th International Conference on E-Business and Telecommunications (ICETE 2020)*, Virtual, Online, Jul. 2020, pp. 151–174.

[13] P. Santini, M. Battaglioni, F. Chiaraluce, and M. Baldi, "Analysis of reaction and timing attacks against cryptosystems based on sparse parity-check codes," in *Proc. Code-Based Cryptography - 7th International Workshop (CBC 2019), Revised Selected Papers*, Springer, Darmstadt, Germany, May 2019, pp. 115–136.

[14] E. Eaton, M. Lequesne, A. Parent, and N. Sendrier, "QC-MDPC: A timing attack and a CCA2 KEM," in *Proc. International Conference on Post-Quantum Cryptography*, Fort Lauderdale, Florida, Apr. 2018, pp. 47–76.