On Decoding Fountain Codes with Erroneous Received Symbols

Xuan He and Kui Cai
Singapore University of Technology and Design (SUTD), Singapore, 487372
Email: helaoxuan@126.com, cai_kui@sutd.edu.sg

Abstract—Motivated by the application of fountain codes in the DNA-based data storage systems, in this paper, we propose the basis-finding algorithm (BFA) for decoding fountain codes for an erasure-error channel where each received symbol has a fixed probability to be correct. The key idea of the BFA is to find a basis of the received symbols, and then use the most reliable basis elements to recover the source symbols with the inactivation decoding. Gaussian elimination can be used to find the basis and to identify the most reliable basis elements. For random fountain codes, we are able to derive some theoretical bounds for the frame error rate (FER) of the BFA, which reveal that the BFA can perform very well for decoding fountain codes for the considered channel.

I. INTRODUCTION

Fountain codes [1], [2] were originally proposed for erasure channels, where an encoded symbol (ES) transmitted over the channels is either lost or received without errors. In this case, inactivation decoding [3]–[5] is the maximum likelihood (ML) decoder. On the other hand, belief propagation (BP) decoding of fountain codes was considered in [6] for binary symmetric channels (BSCs).

Recently, fountain codes have been applied to the DNA-based data storage system [7], which is a promising candidate for archiving massive data in the future [8]–[10]. In a DNA-based data storage system, the information bits are first written into DNA strands. DNA strands can be considered as 4-ary data strings, which are duplicated into many copies before being stored in a DNA pool in an unordered manner. The DNA strands are read out in a random sampling fashion from the DNA pool to recover the original information. Due to the complicated biochemical processes, various types of errors, such as the insertion error, deletion error, and substitution error can occur within each DNA strand. Moreover, a DNA strand may not have any copy at all during read out.

In [7], an efficient DNA data storage architecture named DNA fountain was proposed, which featured a fountain code for a set of DNA strands as the outer code. It can effectively overcome the missing of DNA strands. Note that in this case, each DNA strand corresponds to an ES of the fountain code. Meanwhile, the Reed-Solomon (RS) code [11] is adopted for each DNA strand as the inner code, which can detect or correct the errors within a DNA strand. However, to ensure a high data storage efficiency, only limited amount of redundancy can be introduced to the RS code, which may not be sufficient to detect or correct all the substitution, insertion, and deletion errors that may occur within a DNA strand. As a result, there is a non-negligible chance to have undetectable errors within the DNA strands at the output of the RS decoder [7]. Then, these erroneous DNA strands become a killing factor for the decoding of the outer fountain code with the inactivation decoding [3]–[5].

Motivated by the application of fountain codes in the DNA-based data storage systems and the problems encountered as described above, in this paper, we propose the basis-finding algorithm (BFA) for decoding fountain codes for a simplified erasure-error channel where each received symbol has a fixed probability to be correct. The key idea of the BFA is to find a basis of the received symbols, and then use the most reliable basis elements to recover the source symbols with the inactivation decoding [3]–[5] (same as the decoding for erasure channels). Gaussian elimination can be used to find the basis and to identify the most reliable basis elements such that the BFA has polynomial time complexity. For random fountain codes, we derive some theoretical bounds for the frame error rate (FER) of the BFA, which reveal that the BFA can perform very well for decoding fountain codes for the considered channel.

The remainder of this paper is organized as follows. Section II introduces the preliminaries of fountain codes. Section III first illustrates our system model, and then proposes the BFA. Section IV presents some theoretical bounds for the FER of the BFA for random fountain codes. Finally, Section V concludes this paper. We remark that due to space limitation, we omit all proofs in this paper, which can be found in the extended version of this work [12].

II. PRELIMINARIES

Let $\mathbb{F}_2$ denote the binary field. There are $n$ source symbols denoted by

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{F}_2^{n \times l},$$

where $x_i, i \in [n] = \{1, 2, \ldots, n\}$ denotes the $i$-th source symbol which is a bit string of length $l$. The sender can generate and send potentially limitless ESs. We denote an ES by a two-tuple

$$(a, y) \in (\mathbb{F}_2^n, \mathbb{F}_2^l), \text{ s.t. } y = ax = \oplus_{i \in [n]} a_i x_i,$$

This work is supported by RIE2020 Advanced Manufacturing and Engineering (AME) programmatic grant A18A6b0057 and Singapore Ministry of Education Academic Research Fund Tier 2 MOE2019-T2-2-123.
where ⊕ is the bitwise addition over $\mathbb{F}_2$ and $a_i$ is the $i$-th entry of $a$. In practice, $a$ is chosen from $\mathbb{F}_2^n$ according to a predefined distribution.

The receiver collects $m$ ESs which are denoted by

$$ (A, y) = \begin{bmatrix} a_1 & y_1 \\ a_2 & y_2 \\ \vdots & \vdots \\ a_m & y_m \end{bmatrix} \in (\mathbb{F}_2^{m \times n}, \mathbb{F}_2^{m \times l}), $$

where the $i$-th row $(a_i, y_i) \in \mathbb{F}_2^{n+l}$, $i \in [m]$ denotes the $i$-th received symbol. For erasure channels, all received symbols are correct, i.e., $Ax = y$. In this case, the inactivation decoding [3]–[5] can efficiently recover $x$ from $(A, y)$ if $\text{rank}(A) = n$, where $\text{rank}(\cdot)$ is the rank of a matrix. We have the following result for $\text{rank}(A)$.

**Lemma 1:** Assume $A$ is independently and uniformly chosen from $\mathbb{F}_2^{n \times n}$ with $m \geq n \geq 0$. The probability for $\text{rank}(A) = n$ is given by

$$ P_{rk}(m, n) = \prod_{i=0}^{i=n-1} (1 - 2^{-i}) \geq 1 - 2^{-m}, $$

where for $n = 0$, we have $P_{rk}(m, n) = 1$.

Moreover, we present another similar lemma below. Both Lemmas 1 and 2 are important for the error performance analysis in Section IV.

**Lemma 2:** Assume $A$ is independently and uniformly chosen from $\mathbb{F}_2^{n \times n}$ with $m \geq n \geq 0$. In addition, each column of $A$ is not a zero vector. The probability for $\text{rank}(A) = n$ is given by

$$ P_{rk}'(m, n) = \prod_{i=0}^{i=n-1} \frac{2^m - 2^i}{2^m - 1} \geq 1 - 2^{-m}, $$

where for $n = 0$, we have $P_{rk}'(m, n) = 1$.

### III. System Model and Basis-Finding Algorithm

Consider the system model shown by Fig. 1. The source symbols $x$ are encoded to form $m$ symbols $(A, z)$ which are transmitted over the channel, and $m$ symbols $(A, y)$ are received. We define the error patterns of $(A, y)$ by

$$ s = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix} = \begin{bmatrix} z_1 \oplus y_1 \\ z_2 \oplus y_2 \\ \vdots \\ z_m \oplus y_m \end{bmatrix} \in \mathbb{F}_2^{m \times l}, $$

which are introduced by the channel. For each $i \in [m]$, the $i$-th received symbol $(a_i, y_i)$ is correct with probability $p_0$, i.e.,

$$ P(y_i = z_i) = P(s_i = \{0\}^l) = P(a_i x = y_i) = p_0, $$

where $p_0$ is a predefined and fixed positive constant throughout this paper. It is reasonable to assume that

$$ p_0 > P(y_i = z_i \oplus s) = P(s_i = s), \forall s \in \mathbb{F}_2^l \setminus \{0\}^l. $$

Otherwise, the channel condition is too bad to recover the source symbols. We remark that our proposed BFA does not rely on the specific values of $P(s_i = s), \forall s \in \mathbb{F}_2^l$ to perform decoding (instead, the BP decoder [6] does), but the performance analysis of the BFA in Section IV does.

Here for simplicity we use the same $A$ at both the sender and receiver. That is, we ignore the channel erasure and assume that errors only occur in $y$ but not in $A$. This is reasonable due to two reasons. First, for fountain codes, which particular ES is received does not matter. Only the number of ESs received matters [5]. Second, assume that $(a, y)$ is sent and $(a', y')$ is received. In practice, not $a$, but a corresponding seed with which as input the predefined pseudo-random number generator (PRNG) can generate $a$, is transmitted so as to save resources. When errors occur in the seed, we may have $a \neq a'$. However, $a$ and $a'$ must follow the same distribution. Considering the first reason illustrated above, the situation is equivalent to the case that $(a', a'x)$ is sent and $(a', y')$ is received, indicating that errors can be regarded to occur only in $y$ instead of in $A$.

We remark that unlike BSC where errors occur to each bit of the transmitted symbol independently, the channel presented in Fig. 1 independently introduces errors to each transmitted symbol. Given the channel transition probability, the BP algorithm proposed in [6] is applicable for decoding. However, due to the existence of erroneous received symbols, the inactivation decoding [3]–[5] cannot work in this situation except for $p_0 = 1$. In the rest of this section, we propose the BFA for decoding.

Let $D$ be an arbitrary matrix over $\mathbb{F}_2$ with $\text{rank}(D) > 0$. With a little abuse of notations, we may also use $D$ as a multiset (a set allows for multiple instances for each of its elements) which takes all its rows as the set elements. We define a basis of $D$, denoted by $B(D)$, as a set that consists of $\text{rank}(D)$ linearly independent rows of $D$. Given $B(D)$, for any row $d \in D$, there exists a unique subset $B_d \subseteq B(D)$ such that $d = \oplus_{b \in B_d} b$. We name $\oplus_{b \in B_d} b$ the linear combination (LC) of the basis elements in $B(D)$ that represents $d$, or equivalently, name it the linear representation (LR) of $d$. A basis element $b' \in B(D)$ is said to attend $\oplus_{b \in B_d} b$ if $b' \in B_d$. The LRs of $D$ refer to all the LRs of the rows of $D$.

For example, consider the following matrix formed by five rows of $D$.

$$ (A, y) = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}. $$

We have $\text{rank}(A, y) = 3$. Let $B(A, y)$ consist of the first three rows of $(A, y)$. We can easily verify that $B(A, y)$ is a basis of $(A, y)$. Then, each row of $(A, y)$ can be uniquely represented as an LC of $B(A, y)$. For example, we have

$$ (a_4, y_4) = (a_2, y_2) \oplus (a_3, y_3), $$

$$ (a_5, y_5) = (a_1, y_1) \oplus (a_2, y_2) \oplus (a_3, y_3). $$
Algorithm 1 Basis-finding algorithm (BFA)

1. Step 1 (Find a basis of (A, y)): Assume rank(A, y) ≥ n, otherwise we can never recover x. Find a basis of (A, y), denoted by B(A, y), which consists of rank(A, y) linearly independent rows of (A, y).

2. Step 2 (Find the n most reliable basis elements of B(A, y)): Find the n basis elements of B(A, y), which form a set denoted by B_n(A, y), that attend the largest numbers of linear representations of (A, y) \ B(A, y).

3. Step 3 (Recover x): Recover x from B_n(A, y) by using the inactivation decoding [3–5].

Accordingly, we say that (a_1, y_1) attends the LR of (a_5, y_5), and both (a_2, y_2) and (a_3, y_3) attend the LRs of \{(a_1, y_4), (a_5, y_5)\}.

Our proposed BFA is to find a basis of (A, y) and then use the n most reliable basis elements to recover x, as shown in Algorithm 1. In Step 1 of Algorithm 1, we can easily generate B(A, y) in the way below:

(W1) Let B(A, y) = ∅.

(W2) For i = 1, 2, ..., m, if (a_i, y_i) is not an LC of B(A, y), add (a_i, y_i) into B(A, y).

The rationale behind Step 2 of Algorithm 1 is that the correct basis elements of B(A, y) generally attend more LRs of (A, y) \ B(A, y) than the incorrect basis elements, which provides a way to measure the reliability of each basis element. We will discuss this in details later in Section IV when analyzing the error performance of Algorithm 1. Accordingly, B_n(A, y) has a high chance to consist of n correct basis elements. If B_n(A, y) consists of n correct basis elements, Step 3 of Algorithm 1 can successfully recover x with the inactivation decoding [3–5]. We remark that the Gaussian elimination can be used to generate B(A, y) and identify the LRs each basis element of B(A, y) attends.

We give a toy example to show how Algorithm 1 works. Assume that (n, l, m) = (2, 2, 5), (A, y) is given by (3), and only (a_1, y_1) and (a_5, y_5) are incorrect. In Step 1, according to (W1) and (W2), we have B(A, y) = \{(a_1, y_1), (a_2, y_2), (a_3, y_3)\}. In Step 2, we have B_n(A, y) = \{(a_2, y_2), (a_3, y_3)\}, since both (a_2, y_2) and (a_3, y_3) attend two LRs of (A, y) \ B(A, y) = \{(a_1, y_4), (a_5, y_5)\}, while (a_1, y_1) only attends one LR (the LR of (a_5, y_5)). By assumption, B_n(A, y) consist of two correct received symbols, indicating that Step 3 can successfully recover x from B_n(A, y).

IV. ERROR PERFORMANCE ANALYSIS

In this section, we analyze the error performance of the proposed BFA given by Algorithm 1. For the ease of analysis, we use the following three assumptions throughout this section. However, these assumptions are not necessary for the BFA to apply.

i) Random fountain codes are considered, i.e., each entry of A is independently and uniformly chosen from F_2.

ii) For each incorrect received symbol (a, y), its error pattern s = ax ⊕ y is independently and uniformly picked from F_2 \ {0}^l, i.e.,

\[ \mathbb{P}(s = s') = \frac{1-p_0}{2^l-1} \quad \forall s' \in F_2 \setminus \{0\}^l \]

iii) B(A, y) in Step 1 of Algorithm 1 is generated in the way of (W1) and (W2), where (a_1, y_1), (a_2, y_2), ..., (a_m, y_m) are handled one-by-one in order.

For i ∈ [m], let (A[i], y[i]) denote the first i received symbols. For convenience, we use B(i) to refer to B(A[i], y[i]), the basis that is formed after handling (A[i], y[i]). In particular, we let B(0) = ∅. Denote B_c(i) and B_e(i) as the sets of correct and incorrect basis elements of B(i), respectively. We have B(i) = B_c(i) ∪ B_e(i). Moreover, we define the following events.

- E: the event that |B_e(m)| = n, i.e., B(A, y) has n correct basis elements.
- E_1: the event that (A, y) contains n linearly independent correct received symbols.
- E_2: the event that the error patterns of B_e(m) are linearly independent.
- E_3: the event that the error patterns of the incorrect received symbols of (A, y) are linearly independent.
- F: the event that each basis element of B_c(m) attends more LRs of (A, y) \ B(A, y) than each basis element of B_e(m).
- G: the joint event of E_1, E_3, and F, i.e., (E_1, E_3, F).

It is clear that if the joint event (E, F) happens, Algorithm 1 can correctly recover x. However, it is hard to find a practical method to explicitly compute \mathbb{P}(E, F). Instead, in the following, we first analyze \mathbb{P}(E), which is an upper bound for \mathbb{P}(E, F) such that

\[ \mathbb{P}(E) \geq \mathbb{P}(E, F). \]

Then, we analyze \mathbb{P}(G), which is expected to be a tight lower bound for \mathbb{P}(E, F) for relatively small to moderate values of m (explained later). Finally, we analyze \mathbb{P}(E, F) for relatively large m. To start, we present Lemma 3 below which will be extensively used in the subsequent analysis.

Lemma 3: Consider a coin that shows heads with probability p and tails with probability 1 − p. We toss the coin t times and denote H(t) as the number of times the coin comes up heads. We have

\[ \mathbb{P}(H(t) = i) = \binom{t}{i} p^i (1-p)^{t-i}, \quad i = 0, 1, \ldots, t. \]

Moreover, for any given ε > 0, we have [13]

\[ \mathbb{P}(H(t) ≤ (p - ε) t) ≤ e^{-2ε^2 t} \] and

\[ \mathbb{P}(H(t) ≥ (p + ε) t) ≤ e^{-2ε^2 t}. \]

A. Analysis of \mathbb{P}(E)

Lemma 4: (E_1, E_2) is equivalent to E.

Lemma 4 leads to two necessary conditions for (p_0, n, l, m) in order to ensure a sufficiently high \mathbb{P}(E). On the one hand, it is necessary to have

\[ m > n/p_0, \quad \text{(4)} \]

otherwise \mathbb{P}(E_1) cannot be sufficiently high according to Lemmas 3 and 1. On the other hand, it is also necessary to have

\[ l > \frac{1 - p_0}{p_0} n. \quad \text{(5)} \]

Otherwise, we have n + l ≤ n/p_0 < m. Then, (A[n + l], y[n + l]) will have a high chance to include less than n correct and
more than $l$ incorrect received symbols according to Lemma 3. Moreover, according to Lemma 1, these incorrect received symbols are very likely to contain more than $l$ linearly independent received symbols. As a result, $|B_c(n + l)| > l$ happens with a high chance, indicating that $\mathbb{P}(E_2)$ as well as $\mathbb{P}(E)$ cannot be sufficiently high.

Given $(p_0, n, l, m)$, we can explicitly compute $\mathbb{P}(E)$ by using dynamic programming [14, Section 15.3]. To this end, for $0 \leq i \leq m$, $0 \leq n_e \leq n$, and $0 \leq n_e \leq l$, define $P_E(i, n_e, n_e)$ as the probability that $|B_c(i)| = n_e$, $|B(i)| = n_e$, and the error patterns of $B_c(i)$ are linearly independent. We require the error patterns of $B_c(i)$ to be linearly independent so as to ensure $E_2$.

We have the following theorem.

**Theorem 1:** We have

$$\mathbb{P}(E) = \sum_{0 \leq n_e \leq l} P_E(m, n, n_e).$$

Moreover, we have $P_E(0, 0, 0) = 1$, and for $0 < i \leq m$, $0 \leq n_e \leq n$, and $0 \leq n_e \leq l$, we have

$$P_E(i, n_e, n_e) = P_E(i - 1, n_e, n_e)\frac{2^{n_e}}{2^{2n}} + P_E(i - 1, n_e, n_e)(1 - p_0)\frac{2^{n_e + n_e - 2^{n_e}}}{2^{2n + l - 2^{n_e}}} + P_E(i - 1, n_e - 1, n_e)p_0\frac{2^{n_e}}{2^{2n - 2^{n_e - 1}}} + P_E(i - 1, n_e, n_e - 1)(1 - p_0)\frac{2^{n_e + l + 2^{n_e + l - 2^{n_e - 1}}}{2^{2n + l - 2^{n_e}}}},$$

where for simplicity, we let $P_E(\cdot, n_e', n_e') = 0$ for any $n_e' < 0$ or $n_e' < 0$.

According to (7), we can compute $P_E(i, n_e, n_e)$ for all $0 < i \leq m$, $0 \leq n_e \leq n$, and $0 \leq n_e \leq l$ with time complexity $O(mn)$. Then, we can compute $\mathbb{P}(E)$ with time complexity $O(l)$ based on (6). As an example, we show the numerical result of $1 - \mathbb{P}(E)$ in Fig. 2 for $(n, l) = (100, 100)$ and different $(m, p_0)$. From Fig. 2, we can see that $1 - \mathbb{P}(E)$ decreases rapidly for moderate values of $m$ (waterfall region), but almost stops decreasing for large $m$ (error floor region). We note that in the waterfall (resp. error floor) region, the error patterns of incorrect received symbols have a relatively high (resp. low) probability to be linearly independent. Accordingly, we can approximate $\mathbb{P}(E)$ in the waterfall region based on the following theorem.

**Theorem 2:** $(E_1, E_3)$ is a sufficient condition for $E$. We have

$$\mathbb{P}(E) \geq \mathbb{P}(E_1, E_3)$$

$$= \sum_{i = \max(n, m - l)}^{m} \binom{m}{i} p_0(1 - p_0)^{m - i} \times P_e^*(i, n)P_r^*(l, m - i).$$

Since $P_e^*(\cdot, \cdot)$ and $P_r^*(\cdot, \cdot)$ can be precomputed with time complexity $O(m)$ [12], the time complexity for computing (8) is $O(\max(m, l))$, which is much lower than that for computing (6). Note that as $m$ increases, $1 - \mathbb{P}(E_1)$ keeps decreasing from 1 and $1 - \mathbb{P}(E_3)$ keeps increasing from 0. This explains why $1 - \mathbb{P}(E_1, E_3)$ first decreases and then increases, as can be seen from Fig. 2. Moreover, we can see from Fig. 2 that $\mathbb{P}(E_1, E_3)$ coincides very well with $\mathbb{P}(E)$ in almost the whole waterfall region. This verifies that in the waterfall region, $(E_1, E_3)$ (resp. $E_3$) is the major event for $E$ (resp. $E_2$).

**B. Analysis of $\mathbb{P}(G)$**

We use $\mathbb{P}(G)$ to approximate $\mathbb{P}(E, F)$ due to two considerations. First, $G = (E_1, E_3, F)$ is a sufficient condition for $(E, F)$ such that

$$\mathbb{P}(E, F) \geq \mathbb{P}(G).$$

Second, inspired by Section IV-A, it is shown that in the waterfall region (for moderate values of $m$), $\mathbb{P}(E_1, E_3)$ coincides very well with $\mathbb{P}(E)$, implying $\mathbb{P}(G)$ should also coincide very well with $\mathbb{P}(E, F)$ in the waterfall region.

Given $(p_0, n, l, m)$, we can explicitly compute $\mathbb{P}(G)$ by using dynamic programming again [14, Section 15.3]. To this end, for $0 \leq i \leq m$ and $0 \leq r_0 \leq r \leq n$, assume that $D$ consists of $i$ arbitrary correct received symbols whose basis is $B(D)$. We define $P_G(i, r, r_0)$ as the probability that $|B(D)| = r$ and there are $r_0$ basis elements of $B(D)$ attending zero LRs of $D \setminus B(D)$. We have the following theorem.

**Theorem 3:** We have

$$\mathbb{P}(G) = \sum_{i = \max(n, m - l)}^{m} \binom{m}{i} p_0(1 - p_0)^{m - i} \times P^*_G(l, m - i)P^*_G(i, n, 0).$$

Moreover, we have $P_G(0, 0, 0) = 1$, and for $0 < i \leq m$ and $0 \leq r_0 \leq r \leq n$, we have

$$P_G(i, r, r_0) = P_G(i - 1, r - 1, r_0 - 1)(1 - 2^{r - 1 - n}) + \sum_{r_0 \leq r' \leq r} P_G(i - 1, r, r'_0)\frac{r'_0}{r_0} 2^{r' - r_0 - n},$$

where for simplicity, we let $P_G(\cdot, r', r'_0) = 0$ for any $r' < 0$ or $r'_0 < 0$.

According to Theorem 3, we can compute $P_G(i, r, r_0)$ for all $0 \leq i \leq m$ and $0 \leq r_0 \leq r \leq n$ with time complexity $O(mn^3)$, and then compute $\mathbb{P}(G)$ with time complexity $O(\max(m, l))$. 
As an example, we show the numerical result of $1 - P(G)$ in Fig. 3 for $(n, l) = (100, 100)$ and different $(m, p_0)$. From Fig. 3, we can see that $1 - P(G)$ first decreases and then increases as $m$ increases. This trend of $1 - P(G)$ coincides well with that of $1 - P(E_1, E_3)$ shown by Fig. 2.

We are now to approximate $P_G(i, n, 0)$ so as to reduce the complexity for computing $P(G)$.

**Theorem 4:** We have

$$P(G) \geq \sum_{i = \max(m, n-l)}^{m} \binom{m}{i} p_0^i (1 - p_0)^{m-i} P_{rk}(l, m-i) \times \max_{l \leq h \leq |l|} P_{rk}(i-h, n)(1 - 2^{-h})^n, \quad (11)$$

where $\hat{h} = \log_2 \left( (n-1)^2 + n2^{n+2} - n + 1 \right) - 1$.

The time complexity for computing (11) is $O(\max(m, l))$, which is much lower than that for computing (9). From Fig. 3, we can see that (11) offers an acceptable lower bound for $P(G)$.

**C. Analysis of $P(E, F)$ for Relatively Large $m$**

As $m$ increases, $E_3$ becomes less likely to happen than $(E, F)$ but $E_3$ is actually not necessary for $(E, F)$. Thus, it is not suitable to use $P(G)$ to approximate $P(E, F)$ for relatively large $m$. On the other hand, for relatively large $m$, $1 - P(E)$ almost stops decreasing (encountering error floor), as can be seen from Fig. 2. Since $P(E)$ is an upper bound of $P(E, F)$, $1 - P(E, F)$ must also encounter the error floor. To see whether $1 - P(E, F)$ can approach $1 - P(E)$ or not, we need to check how $P(F|E)$ behaves as $m$ continuously increases.

**Theorem 5:** Let $B_r$ be an arbitrary set of $r$ linearly independent received symbols, which includes $n$ correct and $r - n$ incorrect received symbols ($n \leq r \leq n + l$). Let $(a, y)$ be an arbitrary received symbol which is an LC of $B_r$. Define $p_c$ (resp. $p_e$) as the probability that a correct (resp. incorrect) received symbol of $B_r$ attains the LR of $(a, y)$. We have

$$p_c = 1/2$$

$$p_e = \frac{(1 - p_0)2^{r-n-1}/(2^l - 1)}{p_0 + (1 - p_0)(2^{r-n} - 1)/(2^l - 1)} < p_c.$$

Theorem 5 indicates that if $E$ happens, the correct basis elements have a better chance to attend more LRs of $(A, y) \setminus B(A, y)$ than the incorrect basis elements. Combining with Lemma 3, we are likely to have $P(F|E) \to 1$ and $P(E, F) \to P(E)$ as $m \to \infty$. However, it seems not easy to get a satisfied bound of $P(E, F)$ for relatively large $m$, mainly because the incorrect basis elements do not independently attend the LRs of $(A, y) \setminus B(A, y)$.

**V. CONCLUSION**

In this paper, we have proposed the basis-finding algorithm (BFA) for decoding fountain codes for a channel where each received symbol has a fixed probability to be correct. The BFA can be realized based on Gaussian elimination and hence only has polynomial time complexity. For random fountain codes, we have derived the theoretical bounds for the frame error rate (FER) of the BFA, which indicated that the BFA can perform very well for decoding fountain codes for the considered channel.

**REFERENCES**

[1] M. Luby, “LT codes,” in Proc. IEEE Symposium on Foundations of Computer Science, 2002, pp. 271–280.

[2] A. Shokrollahi, “Raptor codes,” IEEE Trans. Inf. Theory, vol. 14, no. 6, pp. 2551–2567, Jun. 2006.

[3] X. He and K. Cai, “Disjoint-set data structure-aided structured Gaussian elimination for solving sparse linear systems,” IEEE Commun. Letts., vol. 24, no. 11, pp. 2445–2449, Nov. 2020.

[4] F. Lázaro, “Fountain codes under maximum likelihood decoding,” Ph.D. dissertation, Institute for Telecommunications, Hamburg University of Technology, Hamburg, Germany, 2017.

[5] A. Shokrollahi and M. Luby, “Raptor codes,” Foundations and Trends® in Communications and Information Theory, vol. 6, no. 3–4, pp. 213–322, 2011.

[6] O. Etesami and A. Shokrollahi, “Raptor codes on binary memoryless symmetric channels,” IEEE Trans. Inf. Theory, vol. 52, no. 5, pp. 2033–2051, May 2006.

[7] Y. Erlich and D. Zielinski, “DNA fountain enables a robust and efficient storage architecture,” Science, vol. 355, no. 6328, pp. 950–954, Mar. 2017.

[8] L. Organick, S. D. Ang, Y.-J. Chen, R. Lopez, S. Yekhanin, K. Makarychev, M. Z. Racz, G. Kamath, P. Gopalan, B. Nguyen et al., “Random access in large-scale DNA data storage,” Nature biotechnology, vol. 36, no. 3, p. 242, Mar. 2018.

[9] K. Cai, Y. M. Chee, R. Gabrys, H. M. Kiah, and T. T. Nguyen, “Optimal codes correcting a single indeledit for DNA-based data storage,” arXiv preprint arXiv:1910.06501, 2019.

[10] W. Song, K. Cai, and K. A. S. Immink, “Sequence-subset distance and coding for error control for DNA-based data storage,” in Proc. IEEE Int. Symp. Inf. Theory, Jul. 2019, pp. 86–90.

[11] S. Lin and D. J. Costello, Error Control Coding: 2nd Edition. NJ, Englewood Cliffs: Prentice-Hall, 2004.

[12] X. He and K. Cai, “Basis-finding algorithm for decoding fountain codes with erroneous received symbols,” arXiv, 2020. [Online]. Available: https://arxiv.org/abs/2006.06492

[13] W. Hoefding, “Probability inequalities for sums of bounded random variables,” Journal of the American Statistical Association, vol. 58, no. 301, pp. 13–30, 1963.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms: 2nd Edition. Cambridge, MA, USA: MIT Press, 2001.