Array-Code Ensembles -or- Two-Dimensional LDPC Codes

Yuval Cassuto and **M. Amin Shokrollahi**
EPFL
School of Computer and Communication Sciences
ALGO Laboratory
1015 Lausanne, Switzerland
{yuval.cassuto,amin.shokrollahi}@epfl.ch

Abstract— **Probabilistic construction of codes on two-dimensional arrays is proposed and analyzed. Instead of a pure combinatorial erasure model used in traditional array codes, we propose a mixed combinatorial-probabilistic model of limiting the number of column failures, with assuming a binary erasure channel in each failing column. In addition, motivated by practical applications, we maintain an array with a fixed number of columns, while allowing the column size to grow to infinity. As a result, we obtain a framework that allows developing powerful constructions and analysis techniques previously only applicable in the theory of iteratively decoded one-dimensional low-density parity-check codes. The new array-code ensembles are shown to approach the performance of traditional MDS codes, with a simple decoder that offers better scalability in the number of column failures.**

I. Introduction

Linear codes spanned by low density matrices are the keystone of contemporary coding theory, with huge impact on both theoretical research and practical applications. The sparsity of the code matrices offers a compelling complexity advantage in implementation, and achieving this low complexity with optimality in redundancy is the great achievement of a large body of deep research. Interestingly, two separate coding-theory fields aim at the above objective of developing low-density codes with good information efficiency.
One is the field of *array codes*. Array codes, defined over two-dimensional bit arrays, use low-density parity-check matrices to construct codes for erasures or errors of full columns. The low-density property allows updating information bits with "small" subsequent parity updates. Hence the task of array codes is to combat column-level erasures/errors, while being defined by operations over smaller information units (bits or small groups of bits). The ultimate goal of array-code research is code families that are Maximum Distance Separable (MDS) from the column perspective, thus having optimal redundancy for a given column correction requirement. Array codes are widely used in practice, being employed as a central element of RAID (Redundant Arrays of Inexpensive Disks) [8] storage systems. Each array column then represents an individual storage device, whose failure is modeled as a column erasure.
The second, and more famous low-density coding field, is *low-density parity-check (LDPC) codes under iterative decoding*. Without need for detailed introduction, in this area the objective is to construct low-density (one dimensional) codes that will decode well under sub-optimal iterative decoding algorithms.

This work was supported by a grant from the European Research Council.

So far, despite the structural similarities, the two low-density coding fields evolved in essential separation. Array codes have concentrated on algebraic constructions and decoding algorithms that guarantee fixed numbers of column erasures over small array dimensions. In contrast, the theory of iteratively decoded low-density codes has sought probabilistic code constructions that with high probability have good iterative decoding performance in the limit of large block lengths. Previous work exists where iterative decoding of known array codes over one-dimensional channels is experimentally examined [5], [2], but no attempt has been made to construct new array codes for iterative decoding over two-dimensional channels. Such constructions are highly motivated by the current state of matters in array code theory and practice. Algebraic array codes rigidly assume that columns are erased at full, while in practice many storage devices have failure modes that render only part of their data inaccessible. This strong error model introduces significant complexity penalties, with encoding, decoding and update complexities that steeply grow with the number of column erasures. Moreover, algebraic array codes guarantee correction of a certain number of column erasures, with a sharp transition to failure if the specified number of column erasures is exceeded.

In this paper we propose the first application of iterative-decoding construction and analysis techniques to codes over two-dimensional arrays. As the dual title of this paper implies, the authors believe that both coding-theory communities will find interest in it: array codes as the beneficiary application of the new methods, and iterative decoding as the source of expertise upon which the new methods are developed. The paper's main contributions are the set-up of a new two-dimensional coding framework, probabilistic constructions of array codes and their analysis, and insights on good methods to design such codes. In particular, section II introduces the practically-motivated coding structure as a two-dimensional array with one finite dimension and one assumed to grow to infinity. In addition, the erasure model is defined as a hybrid between a combinatorial limit on column failures, with a probabilistic binary erasure channel within each column. Section III presents array-code ensembles, which are probabilistic families of codes that suit the new array structure and erasure model. Array-code ensembles differ from standard LDPC ensembles by adding restrictions to their code graph based on array column locations. In section IV three ensemble families are analyzed with standard tools of iterative decoding analysis. The main novelty in this section is the analysis on a sub-graph induced by the number of columns n and the number of column failures r. In section V the design of array-code ensembles is addressed with some examples and comparisons to traditional MDS array codes. It is shown that irregular array-code ensembles can approach the performance of MDS array codes. Finally, in section VI we discuss directions to continue research toward a rich and useful coding scheme with great theoretical and practical promise.

II. Two-Dimensional Erasure Model

In the main target application for this work – arrays of storage devices – erasures are not uniformly distributed across the two-dimensional array. Rather, a few of the columns, corresponding to failing devices, will have a large number of erasures, and the rest of the columns, corresponding to non-failing devices, will have no erasures at all. We now seek to define a general two-dimensional erasure model that captures this bi-modality of erasure probability. But before considering the characterization of erasures over the array, we need to define the structure of the array itself.

bn bits are organized in a two dimensional array $A = (a_{i,j})$, $1 \leqslant i \leqslant b$, $1 \leqslant j \leqslant n$. Note that in practice each $a_{i,j}$ may not be a single bit, but a larger information unit. Nevertheless, we assume the basic array unit to be a bit throughout the paper, since XOR operations over bits can be easily extended to larger sets of bits. Since the total capacity of the storage device is much larger than the desirable unit of XOR operations for the code, the column size (number of XOR units per device) b will be assumed large, and growing to infinity for the purpose of analysis. The number of columns n will however be assumed a fixed integer. This array structure is natural for real storage systems, in which the number of devices stay fixed at the order of roughly 10s of devices, while the capacities of constituent devices grow rapidly with the progression of product generations. This chosen array structure is depicted in Figure 1.

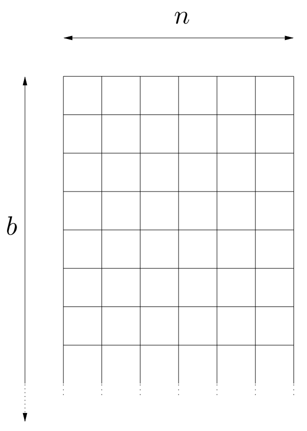

Figure 1. Two-dimensional array with dimensions $b \times n$, where n is fixed and $b \to \infty$.

A. Mixed probabilistic-combinatorial erasure model

Let F_i be an indicator function for the event that column i is in failure state.

$$F_i = \begin{cases} 1 & \text{if column } i \text{ fails} \\ 0 & \text{otherwise} \end{cases}$$

In a failing column, every bit is erased independently and with equal probability. In a non-failing column, none of the bits are erased. Formally, the erasure probability of bits in column i is specified as a function of F_i:

$$\epsilon_i = \begin{cases} \epsilon & \text{if } F_i{=}1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where ϵ is a global erasure probability applying to all failed columns.

To the probabilistic intra-column erasure model above we now add a combinatorial column-failure model. Within an array of n columns, at most r columns are failing. Hence

$$|\{i : F_i = 1\}| \leqslant r.$$

III. Array Code Ensembles

The usual approach to combat erasures in two-dimensional arrays is to algebraically construct array codes for a prescribed number of column failures r (see for example [3], [1]; and [4] for a more general column erasure model). Algebraic array codes are defined explicitly, by specifying sets of array locations with restricted parity values. In addition, algebraic array codes aim at the extreme case of $\epsilon = 1$, i.e. *all* bits are assumed erased in a failing column. In this paper, in contrast, array codes are constructed *probabilistically*, by specifying ensembles of parity restrictions over the array. Moreover, the case of interest here is $\epsilon < 1$, which better describes realistic failure modes in storage systems.

Every parity constraint in the proposed array code ensembles will be of the form:

$$a_{i_1,1} + a_{i_2,2} + \cdots + a_{i_n,n} = 0$$

where $+$ represents the binary XOR operation. The row indices $i_1, i_2, \ldots, i_n$ will be uniformly and independently chosen at random from $\{1, \ldots, b\}$. A key property of these array-code ensembles is that each parity constraint has exactly one bit from each column. In Figure 2, two sample parity constraints, $a_{2,1} + a_{3,2} + a_{4,3} + a_{2,4} = 0$ and $a_{3,1} + a_{1,2} + a_{1,3} + a_{5,4} = 0$, are illustrated. The initial definition of array-code ensembles

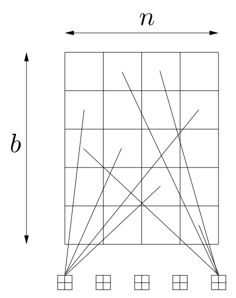

Figure 2. Array parity constraints. Each column contributes exactly one bit to each parity constraint.

above is a special case of more general array-code ensembles, which we now discuss.

A. Regular array-code ensembles

An array-code ensemble is called (l, d)-regular if every array bit participates in l parity constraints, and every parity constraint consists of d array bits, each from a distinct column. Array codes will be constructed from array-code ensembles by randomly sampling $|E| = nbl$ edges in a bipartite graph with $N = nb$ variable nodes and $M = nbl/d$ check nodes (assuming $d|nbl$). The sampling process differs from the standard graph sampling process of LDPC codes in the restriction on check nodes to have at most one neighbor at any column. The (l, d)-regular array-code ensemble is now formally defined via its sampling rule.

Definition 1. [(l, d)-regular array-code ensemble] *Let V_1 be a set of N variable nodes, each with l sockets. Let V_2 be a set of M check nodes, each with d sockets. Initially we pick a variable node and sequentially connect its sockets to random check-node sockets. Then from each connected check node we connect the remaining sockets to random vacant sockets of variable nodes in a random set of distinct columns. Then we connect sockets of these variable nodes to random sockets at check nodes which are connected to no other variable nodes at the same column. We proceed by alternating between variable and check nodes until all the sockets are connected.*

B. Irregular array-code ensembles

As in one-dimensional iterative decoding [7], better performance is achieved when the degree regularity constraint is lifted. We defer the discussion of irregular array-code ensembles to the next section.

IV. Analysis of Iterative Erasure Decoding

A natural method to decode array codes from the proposed ensembles is by iterative message-passing decoding. Since this is a standard way to decode one-dimensional low-density parity-check codes over erasure channels, we will borrow and adapt the analysis tools of these well studied codes [9]. In particular, we will use tree-analysis [6] on the decoding trees induced by the array-code ensembles. The primary outcome of this analysis is a decoding threshold ϵ_0 such that decoding succeeds with high probability if $\epsilon < \epsilon_0$. Messages in erasure-channel tree analysis simply represent erasure probabilities. Throughout the analysis, we will assume that exactly r columns are in failed state. Figure 3 shows the initial messages from variable nodes (corresponding to array bits) to check nodes (corresponding to parity constraints). By the error model defined in (1), if an array bit is in a failed column, then its initial message is ϵ. If it is in a non-failed column, its message is 0. It is clear that messages out of variable nodes in non-failing columns are 0 in all subsequent tree levels as well. Hence these variable nodes can be completely removed from the decoding graph. Considering only variable nodes in failing columns is central to the analysis of array-code ensembles, hence the following formal definition is warranted.

Definition 2. *A decoding graph $\mathcal{G}^{\mathcal{R}}$ is* **induced** *from a code graph $\mathcal{G}$ by taking the subset of its variable nodes that reside in the columns of the set $\mathcal{R} = \{i_1, \ldots, i_r\}$.*

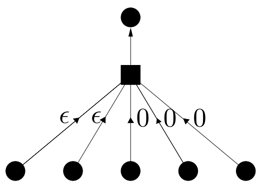

Figure 3. Initial variable-to-check messages. Variables in failed columns send an ϵ message. Variables in non-failed columns send a 0 message.

In the following sub-sections, different array-code ensembles are analyzed and designed through their induced decoding graphs.

A. Regular codes with degree-n check nodes

When every check node has one variable-node neighbor from each column, the induced code graph is regular as well, but with a different check-degree parameter. As a result, the following theorem can be proved.

Theorem 1. *An (l, n)-regular array-code ensemble has design rate $1 - l/n$ and an erasure decoding threshold of an (l, r)-regular one-dimensional LDPC code ensemble.*

Proof: The argument regarding the design rate is identical to (l, n)-regular one-dimensional LDPC codes. We now prove the statement on the decoding threshold. After removing all $(n-r)b$ variable nodes in non-failing columns, the code graph becomes an (l, r)-regular graph, with the added restriction that each of the r neighbors of a check node comes from a different column. It thus remains to prove that given the sampling process of Definition 1, the computation graph for a (finite) number ℓ of decoding iterations is a tree with high probability. The reason this is the case even in the presence of column-based restrictions is that the column size b tends to infinity, leaving sufficient choices of distinct variable nodes even within a small number of allowable columns. By the properties of the sampling process of Definition 1, the computation graph starting at a random variable node is *not* a tree only if one or both of the following cases occur.

1. Collision at check node

Two edges from one or more variable nodes in the computation graph hit sockets of the same check node.

2. Collision at variable node

Two edges from one or more check nodes in the computation graph hit sockets of the same variable node.

Both cases are shown to occur with negligible probability as $b \to \infty$. Since the computation graph is finite, there is a constant number of sockets that would cause collision on check or variable nodes. On the other hand, an edge from a variable node can go to any of at least $(M - \sigma)r$ sockets (σ is the number of check nodes in the computation graph), a number that grows to infinity with b. Similarly, an edge from a check node can go to any of at least $b - \mu$ sockets (μ is the number of variable nodes in the computation graph), again a number that tends to infinity. Since sockets are chosen uniformly at random, the probability of a collision is vanishing with the column dimension tending to infinity. ■

The nice consequence of Theorem 1 is that array codes with high rate (thanks to the high check-node degrees) have high thresholds that gracefully decrease with a growing number of failed columns.

B. Regular codes with general check-node degrees

In the previous sub-section, regular check-degree n induced regular check-degree r in the decoding graph. Now we want to consider the more general case where check nodes have a general, but still constant, degree $d \leqslant n$. As we will see, the induced graph in this case is no longer regular.

Given a set of r failed columns, a check node has between 0 and r neighbors in the induced graph (assuming $r \leqslant d$). Since the columns to which the check node is connected are drawn uniformly at random, the induced degrees are distributed with the following probabilities

$$\Pr(\text{degree } i) = \frac{\binom{r}{i}\binom{n-r}{d-i}}{\binom{n}{d}} \triangleq T_i^{(n,r,d)}. \tag{2}$$

Note that if $r + d \leqslant n$, there is a non-zero probability that a check node has degree 0 in the induced graph, in which case it is removed from the graph. The induced graph remains variable regular, and altogether we obtain the following result.

Theorem 2. *An (l, d)-regular array-code ensemble has design rate $1 - l/d$ and an erasure decoding threshold of an irregular one-dimensional LDPC code ensemble with variable- and check-node normalized degree distributions (from node perspective), respectively, given in*

$$\tilde{L}(x) = x^l \quad , \quad \tilde{R}(x) = \sum_{i=0}^{r} T_i^{(n,r,d)} x^i$$

Proof: From (2), the check-degree distribution of the induced graph is given by $\tilde{R}(x)$. The degree distribution of variable nodes is unchanged in the induced graph, since removed (induced degree 0) check nodes affect only variable nodes outside the induced graph. To prove that finite-depth computation graphs are trees with high probability we essentially repeat the proof of Theorem 1. ■

The $\tilde{}$ sign over the weight-distribution polynomials represent *induced* degree distributions. The variable and check degree distributions from edge perspective are as usual, $\tilde{\lambda}(x) = \tilde{L}'(x)/\tilde{L}'(1)$ and $\tilde{\rho}(x) = \tilde{R}'(x)/\tilde{R}'(1)$, respectively.

C. Irregular codes

The most general array-code ensembles have irregular degrees in the original code graph as well, and not only in the induced graph as in the previous sub-section. We now examine the induced decoding graphs of such ensembles. Let $L(x) = \sum_{l=1}^{\mathtt{l}_{\max}} L_l x^l$ and $R(x) = \sum_{d=1}^{\mathtt{d}_{\max}} R_d x^d$ be the node-perspective variable and check normalized degree distributions, respectively. A given check node with degree d will have induced degree i according to the distribution in (2). Now considering all possible degrees d in the degree distribution $R(x)$, we obtain the induced check-degree distribution

$$\tilde{R}(x) = \sum_{d=1}^{\mathtt{d}_{\max}} R_d \sum_{i=0}^{d} \frac{\binom{r}{i}\binom{n-r}{d-i}}{\binom{n}{d}} x^i = \sum_{i=0}^{\mathtt{d}_{\max}} x^i \sum_{d=i}^{\mathtt{d}_{\max}} R_d \cdot T_i^{(n,r,d)}$$

where the last equality is obtained by reversing the order of summation. Therefore, the i^{th} coefficient of the induced check-degree distribution equals in the irregular case

$$\tilde{R}_i = \sum_{d=1}^{\mathtt{d}_{\max}} R_d \cdot T_i^{(n,r,d)}$$

(starting the summation at $d = 1$ instead of $d = i$ simplifies the expression without changing the sum). This expression leads to the following theorem.

Theorem 3. *An $(L(x), R(x))$-irregular array-code ensemble has design rate $1 - L'(1)/R'(1)$ and an erasure decoding threshold of an irregular one-dimensional LDPC code ensemble with variable and check normalized degree distributions*

$$\tilde{L}(x) = L(x) \quad , \quad \tilde{R}(x) = \sum_{i=0}^{\mathtt{d}_{\max}} \left(\sum_{d=1}^{\mathtt{d}_{\max}} R_d \cdot T_i^{(n,r,d)} \right) x^i$$

Proof: The induced check-degree distribution $\tilde{R}(x)$ is proved in the preceding analysis. As in the regular case, the variable degree distribution is unchanged in the induced graph, since removed check nodes affect only variable nodes outside the induced graph. Convergence to tree ensembles is proved identically to Theorem 1. ■

V. Design of Array-Code Ensembles

With the constructions and analysis tools developed above, we now turn our attention to the design of array-code ensembles. The purpose of this section is to consider different design parameters, and examine code design alternatives to meet those parameters. As part of the presentation, we will compare different code ensembles, both among each other and to traditional array codes. Unlike traditional array codes which are designed for a specific r number of failures, array-code ensembles may be designed to achieve a good *spectrum* of thresholds for different r values.

A. Regular codes with degree-n check nodes

Suppose we want to design an array code for $r = 3$ column failures. One alternative is to use a traditional MDS array code with rate $1-3/n$. Another alternative is to use a $(3, n)$-regular array-code ensemble, which has the same rate. In Figure 4 the thresholds of both options are plotted for $1 \leqslant r \leqslant 7$. Since the MDS array code is decoded algebraically, its threshold is either 1 for $r \leqslant 3$, or undefined for $r > 3$. On the other hand, the array-code ensemble can tolerate many erasures even for large r values. But clearly the regular ensemble is inferior to the MDS code at $r = 3$, with a threshold of 0.844, strictly smaller than 1. The problem with the regular ensemble is that at $r = 3$ we get a threshold of a $(3, 3)$-regular induced graph, which is not an effective rate 0 code. Improvements over this restricted family of ensembles are proposed in the remainder of this section.

B. Regular codes with general check-node degrees

The main drawback of check-degree n regular ensembles is that specifying their rate fixes the variable degree to $l = n(1 - \text{rate})$. This rigidity costs in performance when large l values are manifested. In these cases check-degrees $d < n$ can be used to improve decoding thresholds. An example of

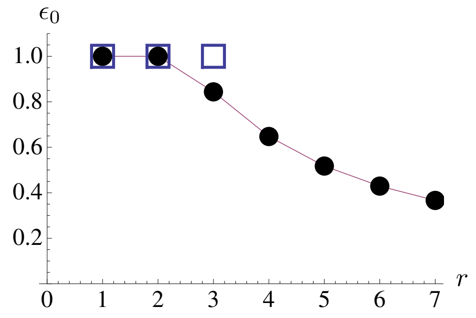

Figure 4. Threshold values as a function of the number of failing columns. dots: $(3, n)$-regular ensembles, squares: traditional MDS array codes.

this improvement is given in Figure 5 for the parameters $n = 16$, $\text{rate} = 3/4$ and $3 \leqslant r \leqslant 6$. A $(3, 12)$-regular code is proposed as an alternative to the $(4, 16 = n)$-regular one. The former (square markers) gives better thresholds for every r compared to the latter (dot markers). Note that unlike check-degree n regular codes whose thresholds are only a function of l, codes with check-degree $d < n$ have thresholds that depend on the length n as well, due to the $T_i^{(n,r,d)}$ coefficients in their induced degree distributions, which depend on n.

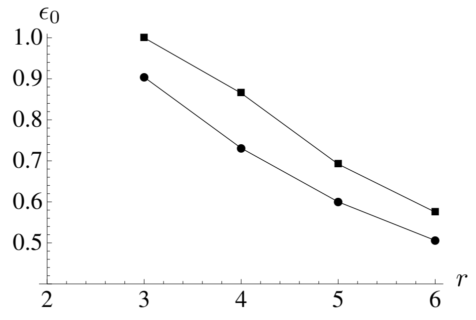

Figure 5. Threshold value comparison between degree n (dots) and degree $d < n$ (squares) ensembles.

C. Irregular codes

General regular codes in the previous sub-section have given better thresholds than ones with check-degree n. However, to approach the optimality of MDS array codes, array-code ensembles need to have irregular degrees. This fact is in line with the well known necessity of irregular degrees for achieving channel capacity over the standard erasure channel. But in our case the design problem is to find *induced* degree distributions, with *induced* rates close to 0 at $r = n(1 - \text{rate})$, that have thresholds close to 1. Such codes perform equivalently to MDS array codes (which similarly have $r = n(1 - \text{rate})$), with the added advantage of predictable thresholds for larger r values. The main tool for practically designing irregular array-code ensembles is by linear-programming optimization of bounded-degree distributions [7],[9, Sec.3-18]. In lieu of a detailed discussion, which exceeds the scope of this paper, we mention the following notes and conclude with an example.

1) When the check-degree distribution is fixed, we adjust the distribution to the induced one and run the standard linear program to find the optimal variable-degree distribution.
2) When the variable-degree distribution is fixed, the linear program has to be adjusted to find the optimal check-degree distribution given the $T_i^{(n,r,d)}$ coefficients. Note that the constraints of the program are still linear for fixed n and r.
3) We can specify individual thresholds for different r values by adding additional linear constraints to the program. However, a joint optimization for a global objective over multiple r values results in non-linear constraints.

As an example, we take the parameters $n = 16$, $\text{rate} = 3/4$ from the previous sub-section and seek an optimal variable degree distribution for $r = 4$, given a fixed check-degree distribution $\rho(x) = x^{11}$ (corresponding to $d = 12$ as in the code of the previous sub-section). The resulting degree distribution $\lambda(x)$ below yields rate 0.7553, with a threshold of 0.974 for $r = 4$.

$$\lambda(x) = 0.4641x + 0.1771x^2 + 0.1318x^4 + 0.0725x^5 + 0.146x^{13} + 0.0084x^{14}$$

Having a threshold very close to 1 at a rate higher than an MDS array code offers equivalent performance with much simpler decoding algorithms. The induced check-degree distribution of $\rho(x)$ for $n = 16$ and $r = 4$ is

$$\tilde{\rho}(x) = 0.0088 + 0.1451x + 0.4835x^2 + 0.3626x^3$$

VI. Conclusions and Future Work

While this paper serves its purpose to bridge two important but disjoint research areas, significant work is still needed to obtain a workable coding scheme with deep theoretical understanding. On the theory side, optimal design and its limits given multi-r threshold spectra are still open. In particular, analytical treatment of induced degree distributions needs to be carried out. On the practice side, systematic codes are required to allow low-complexity updates. In addition, combinations of ϵ failing columns with columns that are completely erased is a useful case to consider.

References

[1] M. Blaum, J. Bruck, and A. Vardy, “MDS array codes with independent parity symbols,” *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 529–542, 1996.
[2] M. Blaum, J. L. Fan, and L. Xu, “Soft decoding of several classes of array codes,” in *Proc. of the IEEE International Symposium on Info. Theory*, Lausanne, Switzerland, June 2002, p. 368.
[3] M. Blaum and R. Roth, “New array codes for multiple phased burst correction,” *IEEE Transactions on Information Theory*, vol. 39, no. 1, pp. 66–77, 1993.
[4] Y. Cassuto and J. Bruck, “Low-complexity array codes for random and clustered 4-erasures,” *IEEE Transactions on Information Theory*, vol. in review, 2010.
[5] J. L. Fan, “Array codes as low-density parity check codes,” in *Proc. of the Intl. Symp. on Turbo Codes*, 2000, pp. 543–546.
[6] M. Luby, M. Mitzenmacher, and M. A. Shokrollahi, “Analysis of random processes via and-or trees,” in *Proc. of the 9th Annual ACM SIAM Symposium on Discrete Algorithms*, 1998, pp. 364–373.
[7] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. Spielman, “Efficient erasure correcting codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.
[8] D. A. Patterson, G. A. Gibson, and R. Katz, “A case for redundant arrays of inexpensive disks,” in *Proc. SIGMOD Int. Conf. Data Management*, 1988, pp. 109–116.
[9] T. Richardson and R. Urbanke, *Modern coding theory*. New York USA: Cambridge University Press, 2008.