LDPC CODE DESIGN FOR DISTRIBUTED STORAGE SYSTEMS

by

Massoud Pourmandi

B.S., Electrical and Electronics Engineering, Guilan University, 2011

M.S., Telecommunications Engineering, Ferdowsi University of Mashhad, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Electrical and Electronics Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I want to deeply thank my supervisor for his invaluable support and guidance throughout my Ph.D. journey. His profound knowledge, patience, and clear guidance have been instrumental in the successful completion of my dissertation. Thanks, Prof. Ali Emre Pusane.

I appreciate my co-supervisor's mentorship. His wisdom, problem-solving strategies, and willingness to embrace challenges have shaped me personally and professionally. Thanks for your steadfast support, Assoc. Prof. Suayb S. Arslan.

I extend my gratitude to Assist. Prof. Elif Haytaoglu for her key contributions to my Ph.D. research. Thank you, deeply.

My appreciation also goes to my jury committee, Prof. Hakan Deliç, Assist. Prof. Faik Başkaya, Prof. Sinem Coleri, and Prof. Z. Caner Taşkın. Their invaluable input and constructive criticism have significantly shaped my research.

I am grateful for my loving family, who have been my constant support.

My wife, Aynaz, has been an essential support throughout my Ph.D. journey. I am eternally grateful for her steadfast support throughout our marriage and for being my partner through every triumph and setback.

Finally, I express my heartfelt appreciation to Gökhan Kuşçu, Gamze Inan, and Kültigin Demirlioğlu. Their support has been invaluable, and I remain deeply grateful.

This work was carried out through the funding provided by TUBITAK under Project Number 119E235.

ABSTRACT

LDPC CODE DESIGN FOR DISTRIBUTED STORAGE SYSTEMS

An exhaustive study on a cooperative repair scheme for distributed storage systems (DSSs) utilizing base station (BS) assistance was presented, involving the concept of an admissible region and optimization problem formulation. Findings revealed a broader admissible region, optimizing the storage size-repair bandwidth cost tradeoff. Moreover, a dynamic DSS environment model was introduced. The numerical analysis highlighted the superiority of the BS-assisted cooperative scheme over traditional approaches. Furthermore, an in-depth analysis was made concerning the interplay among the decoding threshold, average repair bandwidth, and code rate, crucial for optimizing low-density parity-check (LDPC) code design for a DSS. The influences of two proposed repair protocols - the random access repair protocol and the ideal repair protocol - on these relationships were emphasized. For random access repair protocol, a method for determining the check node degree distribution $\rho(x)$ to achieve the minimum average repair bandwidth was proposed. Using numerical analysis, ideal repair protocol's superiority was demonstrated. Moreover, the theoretical findings were verified, emphasizing the potential for further repair bandwidth improvements. Lastly, the identification and elimination of stopping sets in a DSS with LDPC-encoded data were explored. Additionally, a greedy algorithm for determining the repair bandwidth required for a failed storage unit was suggested, indicating potential reductions in the bandwidth needed for single-node repair.

ÖZET

DAĞITIK DEPOLAMA SİSTEMLERİ İÇİN LDPC KOD TASARIMI

Dağıtılmış depolama sistemleri (DDS'ler) için işbirlikçi onarım planı üzerine kapsamlı bir çalışma sunuldu. Bu çalışma baz istasyon (BI) destegini kullanarak, kabul edilebilir bölge kavramı konsepti ile optimizasyon problemi formulasyonunu kapsamaktadir. Bulgular, depolama boyutu-onarım bant genişliği maliyet dengelemesini optimize eden daha geniş bir kabul edilebilir bölgeyi ortaya cikardi. Ayrıca, dinamik bir DSS ortam modeli tanıtıldı. Sayısal analiz, BS destekli işbirlikçi şemanın geleneksel yöntemlerden üstün olduğunu vurguladı. Ayrıca, düşük yoğunluklu çift parite kontrolu (LDPC) kod tasarımının optimize edilmesi için dekodlama eşiği, ortalama onarım bant genişliği ve kod oranı arasındaki etkileşime iliskin derinlemesine bir analiz yapıldı. İki önerilen onarım protokolünun (rasgele erişim onarım ve ideal onarım protokolleri) bu etkileşim üzerindeki etkilerine vurgu yapıldı. Rasgele erişim onarım protokolü için ve minimum ortalama onarım bant genişliğini elde etmek için kontrol düğümü derece dağılımını belirleme yöntemi önerildi. Sayısal analiz ile ideal onarım protokolünün üstünlüğü gösterildi. Ayrıca, daha fazla onarım bant genişliği iyileştirmesi potansiyeline dikkat çekilerek, teorik bulgular doğrulandı. Son olarak, LDPC kodlu verilere sahip bir DSS'de durdurma setlerinin belirlenmesi ve ortadan kaldırılması araştırıldı. Ayrıca, tek düğüm onarımı için gerekli olan bant genişliğindeki potansiyel azalmalar gösterilerek, başarısız olan bir depolama birimi için gerekli olan onarım bant genişliğini belirleme için açgözlü bir algoritma önerildi.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES xii
LIST OF SYMBOLS xiii
LIST OF ACRONYMS/ABBREVIATIONS xiv
1. INTRODUCTION 1
1.1. Distributed Storage Systems 7
1.1.1. Repair-Bandwidth v.s. Storage Tradeoff 9
1.1.2. Minimum-Repair Bandwidth and Minimum-Storage 16
1.2. Code Families for Distributed Storage Systems 17
1.2.1. Erasure Codes 17
1.2.2. Locally Repairable Codes 20
1.2.3. Regenerating Codes 21
1.2.4. LDPC Codes 23
1.3. LDPC Code Design Techniques 24
1.3.1. Linear Programming Approach 25
1.3.2. Differential Evolution Based Design 26
1.4. LDPC Decoding for BEC 29
1.4.1. Peeling Decoder 29
1.4.2. Stopping Sets 30
1.5. Thesis Structure and Contributions 31
1.5.1. Thesis Structure 31
1.5.2. Thesis Contributions 32
2. BASE STATION ASSISTED COOPERATIVE DSS 33
2.1. System Model Description for BS-Assisted Cooperative Scenario 33
2.2. BS-Assisted Cooperative Repair Bandwidth Cost 38

2.2.1. Exploring BS-MSCR and BS-MBCCR Points 39
2.2.2. Identifying the Minimum Number of Active BSs 41
2.2.3. Admissible Region 42
2.3. Repair Bandwidth Cost v.s. Storage in Cellular Networks 44
2.3.1. Modeling Node Dynamics in Cellular DSS 45
2.3.2. Exploring the File Size Constraint 46
2.3.3. Solving the Repair Optimization Problem 47
2.4. BS-Assisted MSCR Code Construction 50
3. OPTIMIZED LDPC CODE CONSTRUCTIONS FOR DSS 52
3.1. Introduction 52
3.2. Novel Repair Protocol and Primary Findings 54
3.3. Incorporating Repair Protocols in LDPC Code Design 58
3.3.1. LDPC Code Design Optimization Using DE Algorithm 60
3.4. Numerical Results: LDPC Code Design for Different Repair Protocols . 64
4. AN LDPC CODE CONSTRUCTION FRAMEWORK FOR DSS 70
4.1. Introduction 70
4.2. LDPC Regenerating Codes for Repairing Data Fragments in DSSs . . . 72
4.2.1. Symbols Allocation for Storage Nodes 73
4.2.2. Stopping Sets Elimination 76
4.2.3. Stopping Sets Elimination in LDPC-Based DSSs 79
4.3. Average Single Node Repair Bandwidth for a DSS: Greedy Approach . 80
4.4. Numerical Results 84
5. CONCLUSION 88
REFERENCES 92
APPENDIX A: PROOFS IN CHAPTER 2 99
A.1. Proof of Theorem 2.1 99
A.2. Proof of Theorem 2.2 99
APPENDIX B: PROOFS IN CHAPTER 3 101
B.1. Proof of Theorem 3.1 101
B.2. Proof of Theorem 3.2 101
B.3. Proof of Theorem 3.3 102

LIST OF FIGURES

Figure 1.1. Tradeoff curve between normalized repair bandwidth cost (γ_c) and normalized storage size (α) for various coding strategies in DSS with parameters $n = 10, k = 6, d = 9, t = 3, \mathbf{w} = [1.01, 1.1023, 2.2]$ and $\mathbf{b} = [1, 1, 1]$. 5

Figure 1.2. The information flow graph for code with parameters ($n = 5, k = 3, d^{(\mathrm{RC})} = 4, \mathcal{F}$). Failed node is also denoted in red. The red dash line also indicates the hypothetical min-cut. 11

Figure 1.3. The information flow graph for code with parameters ($n = 6, k = 3, d^{(\mathrm{RC})} = 4, \mathcal{F}$). In the initial step, nodes are represented with a pair (x^i_{in}, x^i_{out}), and in the next steps they are shown with tuple $(x^j_{in}, x^j_{mid}, x^j_{out})$. Failed nodes are in red color. 14

Figure 1.4. Hierarchical DSS with backup node. 15

Figure 1.5. Algorithmic representation of the differential evolution algorithm showing its core operations. 28

Figure 1.6. Decoding fails, $\mathcal{S} = \{v_3, v_5\}$. 30

Figure 1.7. Decoding is successful. 31

Figure 2.1. An example two-level ($M = 2$) data storage hierarchy. The repair process utilizes local cooperations and higher-layer assistance if needed. 34

Figure 2.2. Information flow diagram for $M = 2$, $n_1 = n_2 = 1$. For short-hand notation, only subscripts are used. 35

Figure 2.3. Algorithm for the optimal number calculation of active BSs. . . . 42

Figure 2.4. Storage versus repair bandwidth cost tradeoff for $k = 6$, $d = 9$, $t = 3$, $\mathbf{b} = [1, 0.75, 0.5, 0.25]$ and $\mathbf{w} = [1.2, 1.4, 1.8, 1.84]$ [41] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2022 IEEE). 44

Figure 2.5. Bandwidth cost v.s. storage size for $\mathcal{F} = 1$, $n = 10$, $k = 6$, $d = 9$, $\Delta = 1$, $\mathbf{b} = [1.2, 0.8, 1.5]$ and $\mathbf{w} = [1.2, 1.5, 1.7]$ [42] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2021 IEEE). 49

Figure 2.6. Bandwidth cost v.s. d for $\mathcal{F} = 1$, $n = 10$, $k = 6$, $\Delta = 1$, $\mathbf{b} = [1.2, 0.8, 1.5]$ and $\mathbf{w} = [1.2, 1.5, 1.7]$ [42] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2021 IEEE). 50

Figure 3.1. Bipartite graph for (6,3) LDPC code. 53

Figure 3.2. DE-based algorithm for minimum average repair bandwidth calculation. 63

Figure 3.3. Comparing $\bar{\gamma}_{\min}$ to ϵ^* for code rates, $R = \frac{2}{3}$ and $R = \frac{3}{4}$. Dashed black lines represent a regular check node case. [45] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE). 66

Figure 3.4. Comparing code rate-R to decoding threshold ϵ^* for two repair protocols under a repair bandwidth constraint. The bold black line indicates solutions without a specific protocol. [45] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE). 66

Figure 3.5. Comparing block error rate to channel erasure probability for $R = 2/3$. The curve marked with circles characterizes the time shared performance of the codes with $\hat{\gamma}^{(\text{rand})} = 6.98$ and $\hat{\gamma}^{(\text{rand})} = 7.98$ [45] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE). 67

Figure 4.1. A sub-blocked Tanner graph with $M = 2$ and $n = 5$. 'L' and 'J' labels indicate local and joint check nodes, respectively. 71

Figure 4.2. Symbols allocation in storage units, toy example 1. 75

Figure 4.3. Symbols allocation in storage units, toy example 2. 75

Figure 4.4. Symbols allocation in storage units, toy example 3. 75

Figure 4.5. Finding erasure patterns for $n = 4$ and $k = 2$. Circles indicate the storage nodes. 77

Figure 4.6. Parity check row addition scheme. S indicates the set of stopping sets, and S_E is the eliminated stopping sets after adding a new check node. 79

Figure 4.7. Flowchart demonstrating the process of constructing and modifying the matrix $\mathbf{H}$ for a given R_0 and ϵ_0. 80

Figure 4.8. Single node repair bandwidth calculation algorithm. 83

LIST OF TABLES

Table 3.1. The demonstration of Theorem 3.2 for selected code rates. 57

Table 3.2. Some optimized node degree distributions for proposed repair protocols for $R = 2/3$ [45] (This Table was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE). 62

Table 3.3. Ideal repair protocol ($d_c \leq 13$). 68

Table 3.4. Random repair protocol. 69

Table 4.1. LDPC code construction (Ideal protocol): $n = 10$, $k = 6$. 86

Table 4.2. Normalized $\bar{\alpha}$ and $\bar{\gamma}^{(\text{storage})}$(Ideal protocol). 86

Table 4.3. LDPC code construction (Random protocol): $n = 10$, $k = 6$. 87

Table 4.4. Normalized $\bar{\alpha}$ and $\bar{\gamma}^{(\text{storage})}$(Random protocol). 87

LIST OF SYMBOLS

$\mathbf{b}$	Vector of base stations link capacities
$\mathcal{C}$	Given code block
$d^{(\mathrm{RC})}$	The number of repair contributing storage nodes
$\mathcal{F}$	File size
$\mathcal{G}$	Tanner graph
$\mathbf{H}$	Parity check matrix
M	The number of clusters in BS-assisted cooperative DSS
p	Erasure probability
R	Code rate
t	The number of storage nodes that cooperatively are repaired
$\mathbf{w}$	Vector of base station access costs
α	Storage size
β	Repair bandwidth that is downloaded from a helper node
β'	Exchanged repair bandwidth for cooperative repairing nodes
γ	Repair bandwidth
γ_c	Repair bandwidth cost
ϵ	Channel erasure probability
ϵ^*	Erasure decoding threshold
$\lambda(x)$	Variable node degree distribution for Tanner graph
$\rho(x)$	Check node degree distribution for Tanner graph

LIST OF ACRONYMS/ABBREVIATIONS

BEC	Binary Erasure Channel
BS	Base Station
DE	Differential Evolution
DSS	Distributed Storage System
HMBR	Hierarchical Minimum Repair Bandwidth Regenerating
HMSR	Hierarchical Minimum Storage Regenerating
LDPC	Low Density Parity Check
LRC	Locally Repairable Code
MBCCR	Minimum Repair Bandwidth Cost Cooperative Regenerating
MBR	Minimum Repair Bandwidth Regenerating
MBCR	Minimum Repair Bandwidth Cooperative Regenerating
MDS	Maximum Distance Separable
MSCR	Minimum Storage Cooperative Regenerating
MSR	Minimum Storage Regenerating
RAID	Redundant Array of Independent Disks
RC	Regenerating Code

1. INTRODUCTION

The rise of the digital age in the late 20th and early 21st centuries led to an exponential growth in data generation, necessitating robust and reliable systems for storing this voluminous data. As a response to this, distributed storage systems (DSSs) have gained attention due to their inherent fault tolerance and scalability [1]. It is notable that the foundation of DSSs was laid as early as 2000 when Ahlswede *et al.* presented the concept of network information flow, a critical precursor for subsequent DSS advancements [2].

However, a pressing challenge in DSSs has been the repair problem, the need to replace failed storage nodes without significantly impacting the system performance [1]. Early solutions like RAID (redundant array of independent disks) technologies offered protection against disk failures but were inefficient for large-scale DSSs [3]. Therefore, network information theory has focused on characterizing the relationship between storage size and repair bandwidth due to the varying expenses associated with storage and network communication. Researchers have been working to find codes that effectively balance the tradeoff between repair bandwidth and storage size, called regenerating codes (RC) [1, 4–6]. However, some of these codes, like traditional algebraic codes, can be complicated and challenging to implement and scale in practice. These codes might demand considerable computational resources to partially or fully recover data, which could pose issues during periods of high data traffic, particularly for large block lengths.

As distributed storage systems expanded in scale, they began to experience a higher rate of node failures. This increased frequency led to the problem of data repair becoming a core issue for these systems. Dimakis [1], in his seminal paper, addressed this issue, revealing that linear coding offers a solution for partial data loss that doesn't necessitate the transmission of the entire data object. One of the significant aspects of their study was the argument that linear network codes are sufficient for what is

referred to as the multicast problem. This problem pertains to transmitting a message from a single source to multiple destinations over a network. The key to their analysis was a mapping of the node repair problem to a multicasting problem on an information flow graph. An information flow graph is a visual representation of a DSS that aids in resolving a multicast problem. This graph transforms the original storage problem into a network communication problem where the source multicasts the file to the entire set of potential data collectors. When dealing with the repair problem within a DSS, where a newcomer node must generate an erasure fragment from existing nodes following a node failure, the information flow graph is an invaluable tool for this analysis. In [1], an information flow graph is introduced as a tool for modeling a distributed storage system. It graphically depicts how the data object's information is transmitted through the network, stored in memory-limited nodes, and eventually arrives at data reconstruction points (data collectors). They presented an inherent tradeoff between storage and repair bandwidth within a distributed storage system. This relationship was effectively characterized by applying the max flow min cut theorem to the information flow graph. Furthermore, they introduced regenerating codes that successfully embody this tradeoff curve and lead to a significant reduction in the bandwidth required for repair in a distributed storage system [1].

In large-scale storage systems, the frequency of storage node failures makes multiple-failures recovery a common necessity rather than a rare occurrence. The approach termed as cooperative repair addresses the simultaneous and joint repair of multiple node failures, taking advantage of data exchange among new nodes [4,5]. This strategy significantly reduces repair traffic compared to the conventional method of successive and independent repairs [1]. This is particularly crucial in large-scale distributed storage systems, where repair of failed nodes is often resource and time-intensive. Building upon previous work on regenerating codes [1] and cooperative repair [7], the use of linear cooperative regenerating codes was proposed [4]. These codes offered a further reduction of repair traffic in such systems. An optimal tradeoff between the repair bandwidth and the storage size in each node for cooperative repair was also presented in a closed-form expression [4].

However, it should be noted that the system nodes were assumed to be homogeneous and connected by a reliable network in [1, 4, 5]. The potential impact of node heterogeneity and network failures on the performance of (cooperative) regenerating codes is not considered, thereby identifying additional areas for future investigation.

A hierarchical distributed storage system model was introduced by Calis *et al.* [6] as a solution to lessen the repair bandwidth. This model entails the use of three specific node types: a backup node, whose role is to handle repair tasks; storage nodes, which interact with the backup nodes; and mobile nodes, which only connect with the storage nodes. Notable findings from their study include establishing the upper limit for file size and analyzing the statistics of maintenance and data access costs under the Poisson model for node failures and data requests [6].

Within the range of distributed storage system frameworks, two distinct points on the tradeoff curve hold particular significance. The first, the minimum storage regenerating (MSR) point, denotes the scenario where each storage node retains the least possible data volume, yet, the system retains the ability to recover from node failures. The second is the minimum bandwidth regenerating (MBR) point, characterized by the minimal volume of data required to be transmitted during the regenerating of a failed storage node [1, 6]. It's also important to note that within the context of cooperative repair, these points retain their functionality but are referred to differently: the MSR point is known as the minimum cooperative storage regenerating (MSCR) point, and the MBR point is termed the minimum cooperative bandwidth regenerating (MBCR) point [4, 5].

Furthermore, the repair process often resulted in heavy network traffic, causing significant overhead and degrading the overall system performance [1]. This highlighted the need for efficient repair mechanisms in DSSs, thereby increasing the importance of the repair problem. For example, in the context of mobile cloud storage systems, Shivaramaiah *et al.* proposed threshold-based file maintenance strategies [8]. Calis and Koyluoglu focused on maintaining DSS with backup nodes for repair [6]. Pedersen

et al. further addressed repair scheduling in wireless distributed storage, particularly considering device-to-device (D2D) communication [9]. Moreover, the importance of the repair problem in DSSs also lies in the economic aspect. Replacing failed nodes or repairing corrupted data can lead to significant operational costs. Hence, research in this area is not just about improving the technical performance of DSSs but also about reducing the associated maintenance costs [1].

Previous research in this area has experienced limitations due to the homogeneity of the nodes contributing to the repair [1,4,5]. Efforts have been undertaken to extend the scope to heterogeneous distributed storage systems, with research being conducted into non-cooperative extensions [10]. In today's world, data storage incorporates a hierarchical structure, where the lower layer serves as a cache, and the upper layer contains complex data to enable efficient access to the lower level. Hierarchical heterogeneous structures are expected to replace homogeneous networks in 5G and beyond, with the internet of things (IoT) being especially pertinent in this regard [11]. It is, therefore, necessary to find new achievable regions in the bandwidth-storage tradeoff curve, which requires more comprehensive information flow diagrams that can adequately capture general data maintenance scenarios. Fundamental tradeoffs for functional repair [1] must be determined by establishing an upper bound on file size. Optimal code construction is only possible after deriving such fundamental tradeoffs.

Figure 1.1 provides a comparative analysis of the proposed repair configuration, in this thesis, for the suggested hierarchical DSS model that is called base-station (BS) assisted cooperative DSS against previous repair mechanisms [1, 4] using the tradeoff curve. It is evident from Figure 1.1 that the proposed repair mechanism, which is detailed in Chapter 2, demonstrates superiority under certain conditions. Figure 1.1 also illustrates the evolution from a scenario of single-node repair to a BS-assisted cooperative repair scenario. In the proposed model of this thesis, hierarchical layers of backup (base station) nodes exist, each associated with differing costs and ranked by a cost factor w_l. Each backup node's link access has a finite capacity b_l, governing the number of symbols that can be downloaded from it. When multiple costly backup

nodes are involved in the repair process, the vector of cost factors and the link capacity is represented by $\mathbf{w}$ and $\mathbf{b}$, respectively. As depicted in Figure 1.1, the terms BS-MSCR and BS-MBCCR represent the BS-assisted minimum storage cooperative regenerating point and the BS-assisted minimum bandwidth cooperative cost regenerating point, respectively. In this figure, "Coop" indicates the cooperative scenario [4]. In a DSS, n represents the total number of storage nodes holding encoded data. Parameter d indicates the number of available storage nodes that contribute to the repair of a single node. The parameter k indicates the minimum number of storage nodes required for data recovery, while, for cooperative repair, t denotes the quantity of newcomer storage nodes participating in the cooperative repair process.

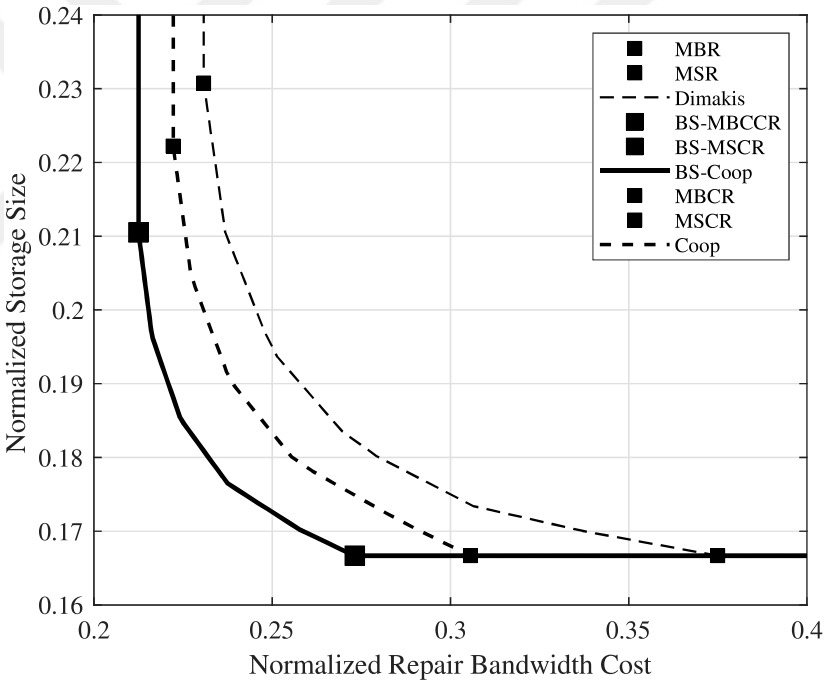

Figure 1.1. Tradeoff curve between normalized repair bandwidth cost (γ_c) and normalized storage size (α) for various coding strategies in DSS with parameters $n = 10, k = 6, d = 9, t = 3, \mathbf{w} = [1.01, 1.1023, 2.2]$ and $\mathbf{b} = [1, 1, 1]$.

The challenges encountered have resulted in the introduction of network coding for DSS, leading to the development of unique repair strategies that significantly improved storage efficiency [1]. The introduction of regenerating codes fundamentally transformed the landscape of repair problems in DSS [1, 4, 5]. These codes significantly

reduced the amount of data that needed to be transmitted during the repair process, thus increasing overall system efficiency. With the evolution of technology, low-density parity-check (LDPC) codes emerged as a promising tool to enhance the functionality of DSS [12–14]. These codes offer a high level of data reliability [15], making them an attractive choice for DSS applications by presenting efficient encoding and decoding strategies [16]. Moreover, LDPC codes are known for their compatibility with various communication and storage scenarios. For instance, they have been applied to distributed storage in mobile wireless networks using device-to-device communication [17]. Several studies have also outlined the advantages of using LDPC codes for big data storage [18, 19].

In summary, the exploration of novel DSS architectures, along with the continuous refinement of regenerating codes, particularly LDPC codes, are key elements in addressing the repair problem in DSS. By reducing the bandwidth required for repair operations and improving system reliability, these coding strategies offer promising avenues for the evolution of DSS, aligning with the emerging paradigms in 5G and future 6G wireless communication networks [20, 21].

This thesis initially introduces the concept of repair bandwidth cost and explores the tradeoff between repair bandwidth cost and storage in a novel DSS called BS-assisted cooperative DSS. Following that, it designs and constructs LDPC codes to decrease the repair bandwidth required in a typical DSS.

This chapter outlines the primary research areas, background, and related works examined in this thesis. The topics covered are as follows: the understanding and characterization of distributed storage systems (Section 1.1), coding methods for distributed storage systems (Section 1.2), the design of LDPC codes for DSS (Section 1.3), an elucidation of the decoding process for LDPC codes (Section 1.4), and thesis outline with summary of the key contributions (Section 1.5).

1.1. Distributed Storage Systems

The growing popularity of instant data access in today's technologically advanced era is undeniable. Continuous connectivity and the internet of things have given rise to a multitude of innovative applications. The constant influx of social media, online news, streaming content, and more can be overwhelming on a daily basis. As a result, it is crucial for storage systems, regardless of size, to adopt a distributed design in order to effectively manage such vast data requirements and ensure dependable access. However, storing massive amounts of data in centralized systems significantly heightens the risk of data loss. By transitioning from centralized to distributed storage systems, data regeneration can be made more efficient.

A DSS consists of numerous storage nodes designed and organized to cooperatively store and manage substantial data files of size $\mathcal{F}$. Given the frequent occurrence of storage node failures, there are two primary methods for data protection: replication and erasure coding [1]. Erasure coding is more prevalent than replication due to its superior storage efficiency. In erasure coding, a file is divided into k equal segments, each of size $\mathcal{F}/k$, and encoded to yield a total size of $n\alpha$ where $\alpha \geq \mathcal{F}/k$. Owing to the system's distributed structure, each node acquires data of size α. To reconstruct the file, it is necessary to contact any k out of n nodes and obtain all the stored data. Nevertheless, the network typically experiences increased traffic when k data fragments are downloaded. This is the primary reason for the use of regenerating codes [1]. They are designed to take care of the huge repair bandwidth problem. In the repair process, a set of remaining nodes enable the recovery of a failed node or nodes as efficiently as possible in the event of node failure(s).

In cooperative repair, a lazy repair approach can be employed, where regeneration does not occur for a certain period, allowing for the accumulation of losses before initiating a joint repair. Furthermore, by tackling all node failures at the same time, network traffic is reduced as the nodes involved in the repair process collaboratively exchange some data [4, 5]. During the regeneration process, it is assumed that there

were t failed nodes. When a new node joins, it first contacts other live and available nodes to download β data units. The t newcomers will exchange data once they have processed the downloaded data in the first phase by downloading β' units of data from the other $t-1$ newcomers. As a final step, the computation is carried out to complete the regeneration process.

The following discussion highlights some of the most relevant works in relation to the thesis. The groundbreaking work by Dimakis *et al.* [1] examined the issue of single node failure repair in distributed storage systems by employing an information flow graph model of network coding. Notably, their approach was to enhance traditional error coding to reduce repair traffic, accomplished through the extension to regenerating codes. Furthermore, their research concentrated on two types of regenerating codes: MSR codes, also known as minimum distance separable (MDS), and MBR codes.

Interestingly, regenerating codes have demonstrated a considerable reduction in repair traffic for recovering the content of failed nodes [1], particularly in the case of MBRs. Additionally, this study [1] identified optimal storage size and repair bandwidth values for single-node recovery in MSR and MBR operating points. In DSSs, multiple failures can occur within a specific timeframe, where failed nodes are repaired periodically; this is sometimes called lazy repair. A cooperative repair model can be employed to repair several nodes simultaneously, necessitating data exchange between repairing nodes based on the data acquired from neighboring nodes [4, 5].

The foundational work on DSSs presented by Dimakis *et al.* [22] proposed a straightforward model, where a failed node is repaired simultaneously. Later, Shivaramaiah *et al.* [8] delved into the realm of threshold-based repair strategies in mobile cloud storage systems, taking into account varying departure-to-repair rate ratios based on repair bandwidth thresholds. Continuing the exploration, Calis *et al.* [6] conducted a study on hierarchical distributed storage systems, introducing backup nodes to the typical DSS in an attempt to reduce overall communication costs. Additionally, the typical model of DSS in cellular networks is composed of mobile storage nodes. These

nodes often move in and out of the network cells, creating a challenge in keeping track of active nodes for repair. This issue can be mitigated by regularly detecting and repairing failed nodes. In light of this, Pedersen *et al.* [9] proposed a time-scheduling-based repair approach.

In prior studies, researchers assumed that nodes were homogeneous, meaning they had the same storage capacity and equally contributed to the repair process. Lately, clustered architectures are becoming more common in distributed storage system designs. In these architectures, communication between racks is more costly than communication within a rack due to the presence of multiple clusters. As a result, cross-rack bandwidth cost is higher than intra-rack bandwidth cost [23]. A new storage model has been proposed in [23] to examine the relationship between cross-rack and intra-rack repair bandwidths in data centers. According to this model, a new node regenerates a failed node by gathering a certain amount of information, denoted as β_I, from each node within the same cluster and another amount, represented by β_c, from each node in other clusters. Furthermore, code construction frameworks have been developed for multi-rack systems to address node failures using locally repairable codes (LRC) within each rack [24]. Although LRC codes cannot recover from failures caused by extreme circumstances, they can interact with storage nodes in other clusters.

1.1.1. Repair-Bandwidth v.s. Storage Tradeoff

Given fixed code parameters $(n, k, d^{(\mathrm{RC})}, \mathcal{F})$ with $d^{(\mathrm{RC})}$ as the number of nodes to be contacted to download partial data for repair, here, the fundamental goal is to determine the minimum values for the pair (α, γ), with α representing the storage size and γ denoting the necessary repair bandwidth for node regenerating. Dimakis *et al.* demonstrated that erasure codes can be used to repair partial data loss without the need to transmit the entire data object [1]. These codes, known as regenerating codes, can achieve every point on the optimal repair bandwidth-storage points.

On the optimal tradeoff curve, there are two extreme points that represent the highest storage efficiency and the lowest repair bandwidth, respectively. The configuration involves active storage nodes, each holding α symbols, and a newcomer receives some data from a subset of the existing nodes to reconstruct the failed node. It has been proven that the single node recovery repair problem can be associated with a multicast problem, and network coding can attain the cut-set bound throughput for multicasting. Furthermore, an information flow graph [2] can represent this multicast problem, illustrating how data is communicated throughout the network.

The information flow graph is a graphical representation of a distributed storage system that describes how the information of the data object is communicated through the network, stored in nodes with limited memory, and reaches reconstruction points at the data collectors. The min-cut of the graph represents the minimum capacity of any cut that separates the source node S from the data collectors DCs [1]. Specifically, an information flow graph is a directed acyclic graph composed of three types of nodes: a single data source S, storage nodes represented with pairs of (x^i_{in}, x^i_{out}) where i indicates the storage index and data collectors DCs. The edges in the graph represent the communication links between nodes. The link between x^i_{in} and x^i_{out} serves as an indication of the data storage capacity, denoted by α, inherent to a single storage node. Furthermore, the link from x^i_{out} to x^j_{in}, where $j > i$, represents the volume of data transmitted from node i to node j, symbolized by β. Finally, a data collector requests the complete data object by connecting to a subset of active nodes through edges with infinite capacity. Active nodes refer to the surviving nodes currently operational and storing a fragment of the data object, while inactive nodes refer to the failed ones [1]. In figure 1.2, the blue pair of circles indicates active nodes while the red ones represent inactive ones. An example of the information flow graph for code with parameters $(n = 5, k = 3, d^{(\mathrm{RC})} = 4, \mathcal{F})$ can be seen in Figure 1.2.

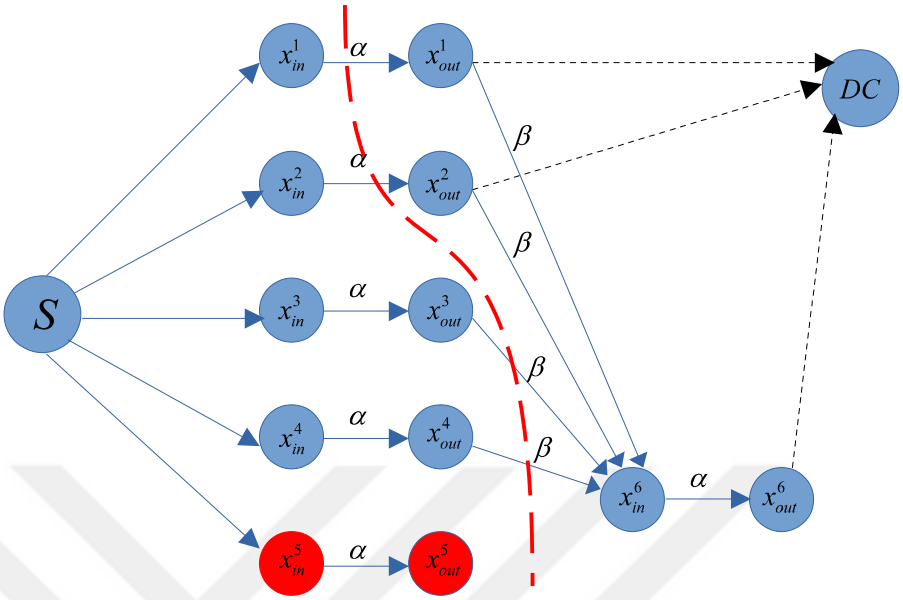

Figure 1.2. The information flow graph for code with parameters $(n = 5, k = 3, d^{(\mathrm{RC})} = 4, \mathcal{F})$. Failed node is also denoted in red. The red dash line also indicates the hypothetical min-cut.

For any given set of parameters, there exists a family of information flow graphs, each corresponding to a specific evolution of node failures/repairs. A tuple is considered feasible if a code with storage and repair bandwidth exists. To optimize the tradeoff between repair bandwidth and storage, the repair bandwidth is fixed, and the minimum value for storage size α is determined. Consequently, the point that minimizes storage or repair bandwidth can be selected [1].

A $(n, k, d^{(\mathrm{RC})}, \alpha, \gamma)$ regenerating code enables the newcomer node to recover the content of the failed node by contacting $d^{(\mathrm{RC})}$ live nodes by downloading β from each of them. Consequently, the repair bandwidth is $\gamma = d^{(\mathrm{RC})}\beta$ where $\alpha \leq \gamma$. This is the main result of [1].

Theorem 1.1. *[1] For any $\alpha \geq \alpha^*(n,k,d^{(RC)},\gamma)$, the code $(n,k,d^{(RC)},\alpha,\gamma)$ is attainable and can be achieved using linear network codes. Consequently, storage-repair bandwidth tradeoff is characterized by,*

$$\alpha^*(n,k,d^{(RC)},\gamma) = \begin{cases} \frac{\mathcal{F}}{k}, & \gamma \in [f(0),+\infty], \\ \frac{\mathcal{F}-g(i)\gamma}{k-i}, & \gamma \in [f(i),f(i-1)), \end{cases} \tag{1.1}$$

where

$$f(i) = \frac{2\mathcal{F}d}{(2k-i-1)i+2k(d-k+1)}, \tag{1.2}$$

$$g(i) = \frac{(2d-2k+i+1)i}{2d}. \tag{1.3}$$

Proof. The proof is given in [1]. □

Theorem 1.1 necessitates that $d \leq n-1$. Furthermore, as the number of repair contributing nodes, $d^{(\mathrm{RC})}$, increases, the repair bandwidth, γ, decreases. When designing regenerating codes specifically for single-node repair, this tradeoff curve serves to identify the boundaries within which optimal codes operate in terms of storage capacity and repair bandwidth.

The potential benefit of allowing data exchange among the nodes being regenerated while repairing multiple node failures simultaneously, was investigated in [4]. The cooperative repair process involves two stages. In the initial stage, each new node chooses a set of $d^{(\mathrm{RC})}$ surviving nodes and downloads a total of $d^{(\mathrm{RC})}\beta$ symbols from them. In the subsequent stage, a new node retrieves β' symbols from each of the other new nodes. When t new nodes are simultaneously reconstructed, the repair bandwidth per new node is $d^{(\mathrm{RC})}\beta+(t-1)\beta'$. As in the cooperative case, there is a tradeoff between the amount of data stored in a node and the repair bandwidth, which is represented in the following theorem.

Theorem 1.2. *[4] The Pareto front of the tradeoff curve for cooperative repair is the convex hull of the following sets:*

$$\{(\tilde{\gamma}_j, \tilde{\alpha}_j) : j = 2, ..., k-1, d \leq (t-1)\mu(j)\}, \tag{1.4}$$

$$\left\{(\tilde{\gamma}'_{\lfloor j/t \rfloor}, \tilde{\alpha}'_{\lfloor j/t \rfloor}) : j = 2, ..., k-1, d > (t-1)\mu(j)\right\}, \tag{1.5}$$

$$\{(\tilde{\gamma}'_0 + c, \tilde{\alpha}'_0) : c \geq 0\}, \tag{1.6}$$

$$\{(\tilde{\gamma}_k, \tilde{\alpha}_k + c) : c \geq 0\}, \tag{1.7}$$

where $\mu(j)$ is given as

$$\mu(j) = \begin{cases} \frac{j(d-k)+(j^2+\psi_{j,t})/2}{jt-\psi_{j,t}} & \psi_{j,t} < jt, \\ \infty, & \psi_{j,t} = jt, \end{cases} \tag{1.8}$$

and $\psi_{j,m}$ is also is introduced as

$$\psi_{j,m} = \lfloor j/m \rfloor m^2 + (j - \lfloor j/m \rfloor m)^2. \tag{1.9}$$

First type points:

$$(\tilde{\alpha}_j, \tilde{\gamma}_j) = \Big(\frac{d-k+j+\frac{t-1}{2}}{D_j}, \frac{d+\frac{t-1}{2}}{D_j}\Big), \tag{1.10}$$

where $D_j = k\left(d-k+j+\frac{t-1}{2}\right) - \frac{j(j-1)}{2}$.

Second type points:

$$\left(\tilde{\alpha}'_j, \tilde{\gamma}'_j\right) = \Big(\frac{d-k+t(j+1)}{D'_j}, \frac{d+t-1}{D'_j}\Big), \tag{1.11}$$

where $D'_j = k\left(d+t(j+1)-k\right) - \frac{t^2 j(j+1)}{2}$, $\tilde{\alpha} = \frac{\alpha}{\mathcal{F}}$ and $\tilde{\gamma} = \frac{\gamma}{\mathcal{F}}$.

Proof. The proof is given in [4]. □

Theorem 1.2 requires that $n \geq d + t$ and each set contains at most $k - 2$ points. In Theorem 1.2, two types of points are denoted by $(\tilde{\gamma}_j, \tilde{\alpha}_j)$ and $(\tilde{\gamma}'_j, \tilde{\alpha}'_j)$ are called first type and second type points respectively. These points serve a crucial role in characterizing the optimal tradeoff between repair bandwidth and storage per node in the context of cooperative regenerating codes. The significance of these operating points lies in their ability to measure the performance of various codes and enable

comparisons based on their effectiveness in minimizing repair bandwidth or storage per node. In other words, Theorem 1.2 facilitates the explicit construction of codes that achieve the desired tradeoff between repair bandwidth and storage per node. Consequently, these operating points contribute to advancing the understanding and development of efficient cooperative regenerating codes. This thesis adopts a similar methodology in the characterization of the corner points within the proposed DSS in the next chapter.

For cooperative regenerating codes, an example of the information flow graph $\mathcal{G}(n, d^{(\mathrm{RC})}, k, r; \alpha, \beta, \beta')$ is illustrated in Figure 1.3.

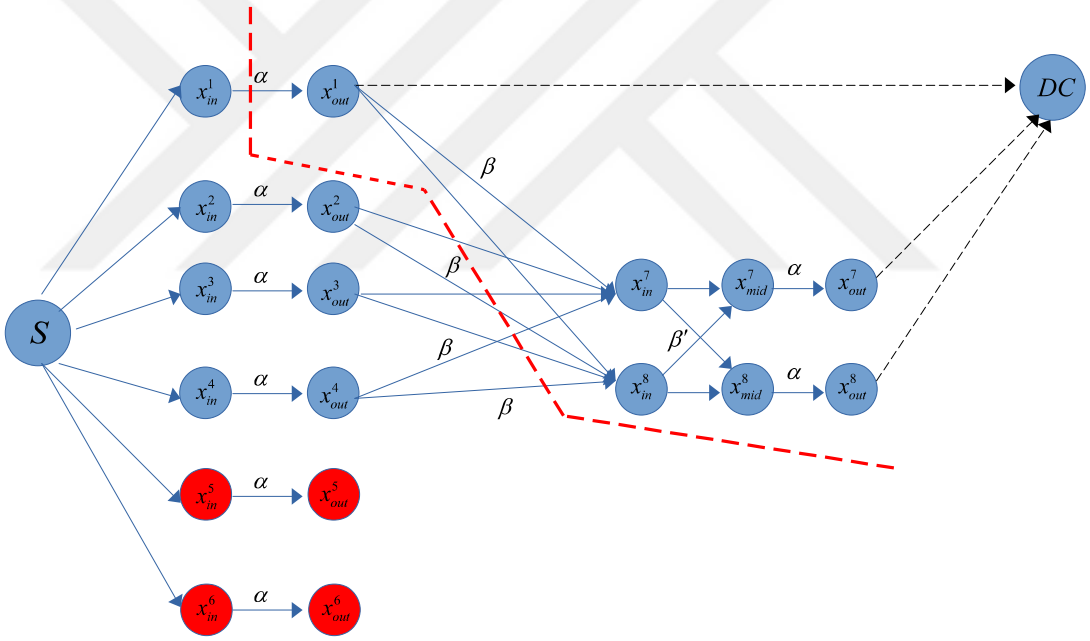

Figure 1.3. The information flow graph for code with parameters $(n = 6, k = 3, d^{(\mathrm{RC})} = 4, \mathcal{F})$. In the initial step, nodes are represented with a pair (x^i_{in}, x^i_{out}), and in the next steps they are shown with tuple $(x^j_{in}, x^j_{mid}, x^j_{out})$. Failed nodes are in red color.

In a separate study addressing the single-node repair scenario [1], the idea of a backup node was introduced [6]. The backup node can be employed in wireless caching to enhance the system's reliability. Consequently, in a non-local cooperative case, Calis *el al.* [6] integrated the concept of a backup node (BN) into distributed

storage in their research project. In this novel arrangement, the content is stored with coding in storage nodes and without coding in a backup node. As depicted in Figure 1.4, the backup node only participates in the repair process, and nodes are categorized into three groups: storage nodes that provide service to mobile nodes, the backup node that exclusively connects to storage nodes, and mobile nodes that request data from storage nodes.

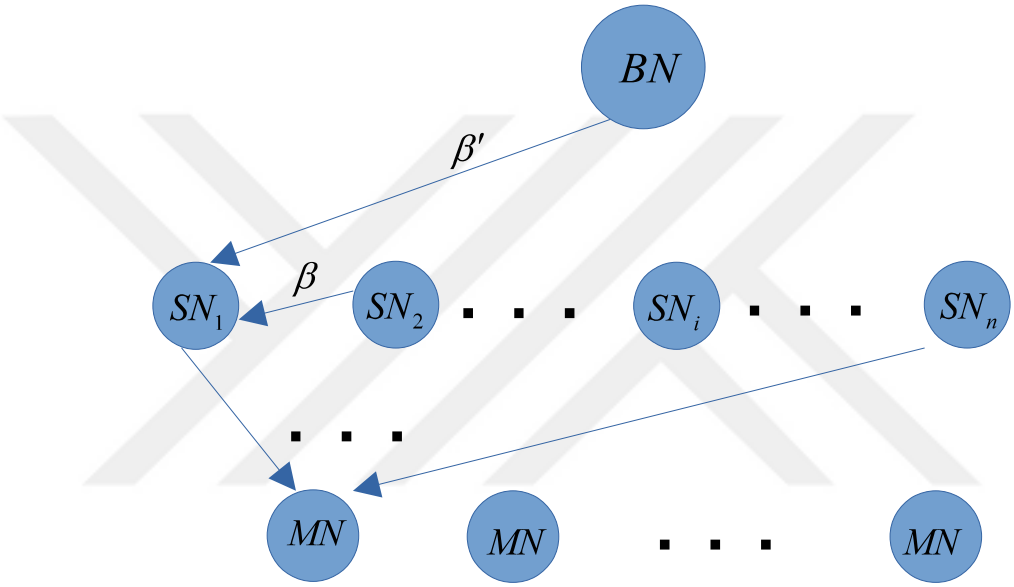

Figure 1.4. Hierarchical DSS with backup node.

The upper bound on file size for hierarchical DSS [6] is also characterized and presented as a Theorem below.

Theorem 1.3. *[6] The file size bound for hierarchical DSS is characterized as*

$$\mathcal{F} \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta + \beta'\}. \tag{1.12}$$

where β' is the amount of data downloaded from the backup node.

Proof. The proof is given in [6]. □

The file size bound, as established by Theorem 1.3, takes into account three crucial factors: α representing the storage capacity per node, β representing the repair bandwidth required from a storage node, and β' representing the repair bandwidth required from backup nodes. By considering these factors, the study aims to explore the impact of backup nodes on the overall performance of the distributed storage system. In [6], the primary focus is on the backup nodes without cost factors within the distributed storage system. However, the notion of backup nodes served as an inspiration to incorporate the costly backup nodes into the proposed DSS model.

1.1.2. Minimum-Repair Bandwidth and Minimum-Storage

The tradeoff curve features two key points: minimum-storage regenerating codes and minimum-bandwidth regenerating codes. MSR codes align with MDS codes, which can be effectively repaired. On the contrary, MBR codes are found at the other extremity of the tradeoff and exhibit the lowest repair bandwidth. One of the primary advantages of MBR structures is that by enabling each storage node to accommodate slightly more symbols than $\mathcal{F}/k$, the repair bandwidth can be substantially decreased [1].

From Theorem 1.1, for single point repair, it can be verified that the critical points are achieved by the pairs:

$$\left(\alpha^{(\mathrm{MSR})}, \gamma^{(\mathrm{MSR})}\right) = \left(\frac{\mathcal{F}}{k}, \frac{\mathcal{F}d}{k(d-k+1)}\right), \tag{1.13}$$

$$\left(\alpha^{(\mathrm{MBR})}, \gamma^{(\mathrm{MBR})}\right) = \frac{2\mathcal{F}d}{2kd-k^2+k}\,(1,1)\,. \tag{1.14}$$

Similar to the mentioned single-point recovery critical points, for the local cooperative repair scenario, there are also two critical points, including MSCR and MBCR codes:

$$\left(\tilde{\alpha}^{(\mathrm{MSCR})}, \tilde{\gamma}^{(\mathrm{MSCR})}\right) = \left(\frac{1}{k}, \frac{d+r-1}{k(d+r-k)}\right), \tag{1.15}$$

$$\left(\tilde{\alpha}^{(\mathrm{MBCR})}, \tilde{\gamma}^{(\mathrm{MBCR})}\right) = \frac{2d+r-1}{k(2d+r-k)}\,(1,1)\,. \tag{1.16}$$

For non-cooperative hierarchical DSS with one backup node, the MSR and MBR point are called hierarchical MSR (HMSR), and hierarchical MBR (HMBR) that are given

$$\left(\alpha^{(\mathrm{HMSR})}, \gamma^{(\mathrm{HMSR})}\right) = \left(\frac{\mathcal{F}}{k}, \frac{\mathcal{F}(d+\tau)}{k(d-k+1+\tau)}\right), \tag{1.17}$$

$$\left(\alpha^{(\mathrm{HMBR})}, \gamma^{(\mathrm{HMBR})}\right) = \frac{2\mathcal{F}(d+\tau)}{k(2d-k+1+2\tau)}\,(1,1)\,, \tag{1.18}$$

where $\tau = \frac{\beta}{\beta'}$ and $(\alpha^{(\mathrm{H.})}, \gamma^{(\mathrm{H.})})$ indicates the storage size and repair bandwidth per node for non-cooperative hierarchical repair.

Moreover, in this thesis, a noteworthy concept of a BS-assisted cooperative repair scenario is introduced, presenting an intriguing extension of the hierarchical repair process that allows for local cooperation. This innovative BS-assisted approach offers a fresh perspective on enhancing the repair process within distributed storage systems, opening doors to potential optimizations.

1.2. Code Families for Distributed Storage Systems

In this section, an exploration of various notable coding methodologies, pertinent to distributed storage systems is presented.

1.2.1. Erasure Codes

Erasure codes are a class of error correction coding techniques typically leveraged in distributed storage systems in which encoded data fragments are scattered across multiple storage units [1]. When a node fails, erasure codes enable efficient repair of the failed node. Within the scope of erasure codes, the focus is on two emerging classes of erasure codes: Locally repairable codes (LRCs) aiming to minimize the involvement of helper nodes in the repair process [25–28] and regenerating codes [1, 4, 5, 27, 29] that minimize the network transfer of repair data as well as LDPC codes [15]. These classes of erasure codes are subsequently explored and discussed in detail.

Erasure codes belong to the category of linear block codes, a class of error-correcting codes used to encode a message by incorporating additional redundant bits into the original message [27]. Therefore, acquiring a comprehensive understanding of the background of linear block codes is crucial. This understanding serves as a foundation for delving into the intricacies of the ongoing research conducted in this thesis.

Information sequence, as its name implies, is a continuous stream of binary numbers over a mathematical field known as GF(2). In block coding, an information sequence is divided into blocks of a fixed length, each containing k bits. There are a total of 2^k unique messages (blocks). A channel encoder is then used to convert each input message into a longer binary sequence, known as a codeword v, containing n bits, where n is greater than k. Since there are 2^k unique messages, there are also 2^k unique codewords. Together, these codewords form a linear block code $\mathcal{C}$ with parameters (n, k) [30]. Alternatively, a linear code can be expressed as $(n, k, d_{\min})$, where $d_{\min}$ is the code's minimum distance [31]. The minimum distance $d_{\min}$ satisfies $d_{\min} = w_{\min}$. That is, the minimum distance of a linear block code is equal to the smallest Hamming distance $w_{\min}$ [31]. In addition, the code rate is defined as $R = \frac{k}{n}$.

The relationship between minimum distance $d_{\min}$ with k and n is expressed using the Singleton bound:

Theorem 1.4. *[31] (Singleton bound) The minimum distance $d_{\min}$ is related to k and n using the following upper bounded*

$$d_{\min} \leq n - k + 1. \tag{1.19}$$

The family of codes that satisfies the inequality in Theorem 1.4 with equality, i.e., a code with $d_{\min} = n - k + 1$ is called Maximum Distance Separable (MDS) codes. Since MDS codes have the unique capability to recover k information symbols from any subset of k symbols out of n symbols, they stand out as a remarkable tool in the coding theory landscape. Their design and efficiency position them as optimal for the redundancy reliability tradeoff, as highlighted in [1].

A linear block code, $\mathcal{C}$, is a k-dimensional vector space made up of k linearly independent vectors (designated as $\mathbf{g_0}, \ldots, \mathbf{g_{k-1}}$) such that any codeword $\mathbf{v}$ in $\mathcal{C}$ can be expressed as a linear combination of these vectors

$$\mathbf{v} = m_0\mathbf{g_0} + \cdots + m_{k-1}\mathbf{g_{k-1}}, \tag{1.20}$$

where $m_i \in \mathrm{GF}(2^l)$, and $l \geq 1$. Alternatively, (1.20) can be rewritten using the $k \times n$ matrix $\mathbf{G}$ as $\mathbf{v} = \mathbf{mG}$, with $\mathbf{m} = [m_0, ..., m_{k-1}]$ and $\mathbf{G} = [\mathbf{g}_0^t, \ldots, \mathbf{g}_{k-1}^t]$. A generator matrix $\mathbf{G}$ is considered systematic if it is arranged in the form such that $\mathbf{I_k}$ is a $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n-k)$ matrix that represents the parity submatrix, i.e., $\mathbf{G} = [\mathbf{I_k}, \mathbf{P}]$. It is also noted that the generator matrix $\mathbf{G}$ can be rewritten as $\mathbf{G} = [\mathbf{P}, \mathbf{I_k}]$ [31]. Therefore, the corresponding parity check matrix can be derived as $\mathbf{H} = [-\mathbf{P}^T, \mathbf{I_{n-k}}]$.

With a $(4, 3)$ MDS code, let's explain how data can be encoded and decoded. With the systematic generator matrix G of a $(4, 3)$ MDS code, $k = 3$ symbols are encoded into $n = 4$ symbols using

$$\underbrace{[m_0, m_1, m_2]}_{\mathbf{m}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{G}} = \underbrace{[v_0, v_1, v_2, v_3]}_{\mathbf{v}}. \tag{1.21}$$

Next, the coded data passes through an erasure channel. If it is assumed that, at most, one symbol is erased during the transmission, the decoder can construct the complete data by at least 3 symbols using

$$\mathbf{v}' = \mathbf{mG}' \rightarrow \mathbf{m} = \mathbf{v}'\mathbf{G}'^{-1}. \tag{1.22}$$

Considering (1.22), decoding the input symbol vector ($\mathbf{m}$) requires using a subset of k symbols from the encoded data ($\mathbf{v}$). The columns in matrix $\mathbf{G}$ that are related to $\mathbf{v}'$ is $\mathbf{G}'$. To achieve the reconstruction of the original data ($\mathbf{m}$), $\mathbf{G}'$ must not be a singular matrix. In general, any $k \times k$ submatrix extracted from $\mathbf{G}$ must be invertible to recover from up to $n-k$ lost symbols. The method for obtaining $\mathbf{G}'$ is by eliminating the columns corresponding to the lost symbols in $\mathbf{G}$. To begin, the inverse of $\mathbf{G}'$ is found, then $\mathbf{m}$ is calculated as $\mathbf{v}'$ multiplied by the inverted $\mathbf{G}'$.

One of the significant types of erasure codes is cyclic codes. Cyclic codes, specifically BCH (Bose–Chaudhuri–Hocquenghem) codes, are important block codes decoded using algebraic algorithms. The next one is Reed Solomon (RS) codes which are a specific type of BCH and MDS codes, which are commonly used in storage devices and communication standards. Although encoding RS codes is relatively straightforward, decoding them can be challenging. Additionally, the block size and decoding time for RS codes are limited, and the Galois field size determines the code's maximum length. However, larger fields result in slower and more complex operations. Therefore, the code length and the number of information bits must be kept small for high transmission rates [30]. Other types of erasure codes include convolutional codes and graph-based codes, with examples being turbo codes and LDPC codes, respectively.

1.2.2. Locally Repairable Codes

Since individual storage nodes can experience failures as time passes, allowing a distributed storage system to repair these nodes locally is beneficial for reducing network traffic. Ideally, it is best to use a code that provides robust protection against an abundance of erasures while also incorporating a fast local recovery mechanism to handle a smaller number of erasures. Locality in distributed storage systems is crucial for reducing data transfer during node repairs, thus expediting the process. Locally repairable codes optimize this by enabling symbol reconstruction with limited access to other symbols, resulting in enhanced storage efficiency and reduced repair bandwidth compared to the traditional erasure codes [25]. For a code with n-length and k information-filled symbols, locality r_i is attributed to symbol i if its reconstruction relies on r_i distinct symbols from the code.

In the context of an (n, k) MDS code, each symbol has a locality k. If a code demonstrates an all-symbol locality of r, then all n symbols share locality r. However, the essential question for LRC codes is: What is the maximum possible code distance d_{min} for a code exhibiting locality r.

Theorem 1.5. *[25] For an $(n, r, d_{min}, \mathcal{F}, \alpha,)$-LRC code, the relationship between the code distance d_{min} and its other parameters is described below:*

$$d_{min} \leq n - \left\lceil \frac{\mathcal{F}}{\alpha} \right\rceil - \left\lceil \frac{\mathcal{F}}{r\alpha} \right\rceil + 2. \tag{1.23}$$

Theorem 1.5 shows an information theoretic bound on code distance $d_{\min}$ [25]. It should be observed that setting $\mathcal{F} = k$ and $\alpha = 1$ corresponds to the following,

$$d_{min} \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{1.24}$$

While locality permits repairs by interacting with a limited node subset, codes with reduced locality are not optimal in terms of repair bandwidth [25].

1.2.3. Regenerating Codes

Regenerating codes are a coding scheme used in distributed storage systems to protect data that is spread across multiple nodes in a network aiming at minimizing the repair bandwidth. The traditional MDS codes are known for their optimal balance between storage and reliability [1]. The question that remains is how these codes are able to encode data in a distributed manner while minimizing the amount of bandwidth required during the repair process [1].

For single node repair, regenerating codes are represented by $(n, k, d^{(\mathrm{RC})})$, where n denotes the total number of storage nodes, k indicates the necessary number of storage nodes to retrieve the entire data, and $d^{(\mathrm{RC})} \geq k$ refers to the subset of storage nodes required to regenerate the content of a failed storage node [1]. Regarding code construction, a significant portion is devoted to MBR and MSR points. Nevertheless, there have been efforts to address the inner points of the tradeoff curve as well [27].

In a regenerating coding scheme, an initial file of size $\mathcal{F}$ is divided into k chunks, each with a size of $\mathcal{F}/k$ symbols. These pieces are encoded and distributed across n storage nodes. As with MDS codes, the original data can be reconstructed by collecting any k of these fragments. In their groundbreaking work, Dimakis *et al.* demonstrated

that this process resembles a multicasting problem within an information flow graph [1]. By employing the max-flow min-cut theorem, their research revealed that nodes could be repaired if the min-cut in the related information flow graph is sufficiently large compared to the file size.

Two extreme points exist on the tradeoff curve where storage/repair bandwidth is minimized. By setting the optimal value $d^{(\mathrm{RC})} = n - 1$ in MSR and MBR points, the corresponding minimum value pairs can be derived as

$$\left(\alpha_{\min}^{(\mathrm{MSR})}, \gamma_{\min}^{(\mathrm{MSR})}\right) = \left(\frac{\mathcal{F}}{k}, \frac{\mathcal{F}}{k}.\frac{n-1}{n-k}\right), \tag{1.25}$$

$$\left(\alpha_{\min}^{(\mathrm{MBR})}, \gamma_{\min}^{(\mathrm{MBR})}\right) = \left(\frac{\mathcal{F}}{k}.\frac{2n-2}{2n-k-1}, \frac{\mathcal{F}}{k}.\frac{2n-2}{2n-k-1}\right). \tag{1.26}$$

For information on the cooperative or hierarchical regenerating models, please refer to Subsection 1.1.2.

However, one of the challenges to tackle in classical regenerating codes based on algebraic constructions is their complexity in decoding [15]. An alternative option that offers low decoding complexity is using LDPC codes. Cellular networks can also benefit from using LDPC codes, particularly when downloading symbols from the base station is not costly compared to the local storage nodes [32]. A comparison of LDPC codes with available coding schemes is also presented in [18] to determine whether they suit DSSs. According to [33], using LDPC codes for commodity computing clusters requires less redundancy than RS codes and replication for the same fault tolerance level. It has also been demonstrated that different configurations of LDPC codes consume different amounts of system resources [34]. In [35], it is shown that LDPC codes can be used as viable storage options because of their well-known asymptotic and finite-length performance. As part of [19], LDPC codes of relatively large block lengths are evaluated in terms of their recovery capability and repair complexity compared to RS codes and LRCs. In contrast to algebraic codes, LDPC codes can also be made more data-recovery capable without affecting their repair bandwidth performance [19].

1.2.4. LDPC Codes

Over the past few years, LDPC codes have experienced a notable resurgence of interest, owing to the requirements for efficient coding schemes that can effectively recover messages that approach the channel capacity in 5G and upcoming technologies. The linear block codes are denoted by (n, k) pair where n is the number of symbols in the codeword, and k is the length of the message symbols. A linear block code is typically described as a matrix or Tanner graph [16]. For example, binary LDPC codes are linear block codes characterized by $(n-k) \times n$ parity check matrix $\mathbf{H}$ with sparse non-zeros elements. Here, sparse means that the number of "1"s compared to "0"s is very small in matrix $\mathbf{H}$. In matrix, $\mathbf{H}$, elements of a row and a column correspond with variable nodes $v_i, i = 1, \ldots, n$ and check nodes $c_i, i = 1, \ldots, n-k$ respectively. Furthermore, the element h_{ij} of the matrix $\mathbf{H}$ is a 1 when there is a connection between c_i and v_j in the corresponding tanner graph. According to the degree distributions, LDPC codes can be classified as regular or irregular [14]. According to Tanner graphs, the maximum degree for variable nodes and the maximum degree for check nodes are also indicated by $d_{v_{\max}}$ and $d_{c_{\max}}$. The node degree distributions are typically used to describe an ensemble of irregular LDPC codes. The following is a convenient way of describing node degree distributions for variable and check nodes in the literature [16]:

$$\lambda(x) = \sum_{d=2}^{d_{v_{\max}}} \lambda_d x^{d-1}, \tag{1.27}$$

$$\rho(x) = \sum_{d=2}^{d_{c_{\max}}} \rho_d x^{d-1}, \tag{1.28}$$

where λ_d is the fraction of all nodes connected to a variable node with degree d, and $d_{v_{\max}}$ is the maximum number of edges connected to a variable node. It is also similar for ρ_d and $d_{c_{\max}}$ with check nodes. Thus, average variable node degree and average check node degree are denoted as

$$\bar{d}_v = \frac{1}{\sum_{i=2}^{d_{v_{\max}}} \frac{\lambda_i}{i}}, \tag{1.29}$$

$$\bar{d}_c = \frac{1}{\sum_{i=2}^{d_{c_{\max}}} \frac{\rho_i}{i}}. \tag{1.30}$$

Additionally, the code rate, R, of an LDPC code with $\lambda(x)$ and $\rho(x)$ is given as

$$R = 1 - \frac{\bar{d}_v}{\bar{d}_c}. \tag{1.31}$$

The binary erasure channel model is ideally suited to describe all data loss in storage systems [15], so density evolution is used to describe how an LDPC code performs under belief propagation (BP) decoding. A density evolution is a method of tracking erasure during decoding iterations, which is given by

$$x_l = \epsilon\lambda(1 - \rho(1 - x_{l-1})), \tag{1.32}$$

where $x_0 = \epsilon$ is the input channel erasure probability and x_l is the erasure probability of any information bit after completing l iterations of the BP decoding algorithm [16]. Furthermore, the decoding threshold of the BP decoder is characterized as follows:

$$\epsilon^* = \sup\left\{\epsilon \in (0,1] : \lim_{l\to\infty} x_l = 0\right\}. \tag{1.33}$$

Expanding the density evolution formula of (1.32) around zero yields

$$x_l = x_0\lambda'(0)\rho'(1)x_{l-1} + O(x_{l-1}^2). \tag{1.34}$$

The convergence of the density evolution formula in (1.32) depends on the *stability condition*, which is determined by whether $x_0\lambda'(0)\rho'(1)$ is smaller or greater than one (as stated in [36]). If $x_0\lambda'(0)\rho'(1) > 1$, the formula is not guaranteed to converge to zero as l approaches infinity. However, if $x_0\lambda'(0)\rho'(1) < 1$, then x_l will converge to zero as l approaches infinity.

1.3. LDPC Code Design Techniques

LDPC codes are a type of error-correcting code that is used to ensure the accuracy of data transmission or storage. They are defined by a sparse matrix, known as a parity-check matrix, which allows for efficient decoding algorithms. Tanner graph is a graphical representation of the parity-check matrix of an LDPC code. Each node in a Tanner graph represents either a bit in the code word (variable node) or a parity-check equation (check node). The degree of a node in a Tanner graph is the number of edges connected to it. Therefore, LDPC code design refers to finding optimal coefficients for the node degree distributions of the Tanner graph. The optimization of node

degree distributions usually refers to maximizing the code rate for a specific decoding threshold [36].

1.3.1. Linear Programming Approach

For a binary erasure channel (BEC), asymptotic capacity approaching LDPC code design is to find the optimal node degree distributions. Initially, this optimization problem is solved using linear programming in that check node degree distribution is fixed [37]. Considering this, for fixed maximum variable node degree $d_{v_{\max}}$ and the decoding threshold ϵ^*, the density evolution formula of (1.32) can be rewritten in a different form as

$$f(x, \lambda_2, ..., \lambda_{d_{v_{\max}}}) = \epsilon\lambda\left(1 - \rho(1-x)\right) - x = \epsilon\sum_{i\geq 2}\lambda_i\left(1-\rho(1-x)\right)^{i-1} - x, \quad (1.35)$$

where $\epsilon = \epsilon^*$ is the targeted channel erasure rate. It can also be noticed that (1.35) is a linear function of λ_is for fixed $\rho(x)$.

In order to optimize the code rate, denoted as $R = 1 - \frac{\int\rho}{\int\lambda}$, where $\rho(x)$ is constant, it necessitates the maximization of $\int\lambda$. This results in a linear optimization problem to maximize the code rate R while satisfying other constraints:

$$\begin{aligned} &\max_{\lambda}\left\{\sum_{i\geq 2}\frac{\lambda_i}{i}\right\}, \\ &\lambda_i \geq 0, \\ &\sum_{i\geq 2}\lambda_i = 1, \\ &f \leq 0. \end{aligned} \quad (1.36)$$

Likewise, fixing $\lambda(x)$ leads to maximizing the code rate R by minimizing $\int\rho$. This requires using a modified form of the density evolution formula that is given by

$$1 - y - \rho\left(1 - \epsilon\lambda(y)\right) \leq 0 \quad , y \in [0, 1]. \quad (1.37)$$

An optimal degree distribution pair (λ, ρ) can be obtained by beginning with a chosen pair and iterating between the fixed check node degree and the fixed variable node degree optimization problems [16]. Employing this iterative methodology makes the density evolution formula linear with respect to the coefficients of node degree distri-

bution. However, a significant limitation of this technique is the inherent difficulty in identifying an optimal distribution. Consequently, numerous potential node degree distributions remain unexplored throughout this process. In practice, check node degree distribution $\rho(x)$ is fixed as

$$\rho(x) = \frac{d(d+1-\bar{d}_c)}{\bar{d}_c} x^{d-1} + \frac{\bar{d}_c - d(d+1-\bar{d}_c)}{\bar{d}_c} x^d, \tag{1.38}$$

where $d = \lfloor \bar{d}_c \rfloor$. The algorithm for the optimization problem runs with various values of d until the optimal node degree distributions are found.

1.3.2. Differential Evolution Based Design

For optimization problems with some conflicting objective functions, there is not an optimal solution but a set of optimal solutions, which is called Pareto front optimal solutions. In general, the multi-objective optimization problem is described as [38]

$$\begin{aligned} \text{Minimize} \quad & f_m(\boldsymbol{x}), m = 1, \ldots, M, \\ \text{subject to} \quad & g_i(\boldsymbol{x}) \geq 0, i = 1, \ldots, q, \\ & h_j(\boldsymbol{x}) = 0, j = 1, \ldots, p, \\ & \boldsymbol{x} \in R^n, \end{aligned} \tag{1.39}$$

where $\boldsymbol{x}$ indicates the feasible solution space.

Conventionally, this multi-objective problem is solved as a single objective optimization problem using a weights vector, which is called the weighted sum method [38]. However, this method is largely sensitive to the weights vector and tuning this weights vector is a tedious task. It is also not applicable to non-convex problems. In order to alleviate the difficulties of this method, the epsilon constraint method is suggested [38]. In this method, except for one objective function, which is minimized, other objectives are constrained. For example, let us consider a two objectives optimization problem with f_1 and f_2 as objective functions. If f_2 is constrained such that $f_2 \leq f_0$, the pair (f_1, f_0) is always dominated by (f_1, f_2) [38]. Moreover, since f_1 and f_2 are conflicting objectives, minimizing f_1 makes that f_2 get close to f_0. This is equivalent to the simultaneous minimizing of both objectives. Then constraints are handled by a con-

straint handling method in that the constraint $g(x) \geq g_0$ is forced to be satisfied by adding the related cost, i.e., $\max(1 - g(x)/g_0, 0)$, to the main objective function using a multiplier [38].

Differential evolution (DE) is an algorithm that optimizes a single objective in a continuous D-dimensional domain. It uses a population of individuals, where each individual represents a potential solution in the form of an D-dimensional vector. DE generates a candidate solution by adding a scaled version of the difference between two individuals to a third individual. The candidate solution set, consisting of N_P solutions represented by D-dimensional real-valued vectors $\boldsymbol{x}_i$, is denoted as $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_P\}$. During the initialization phase, the solutions are randomly generated within their domain. The main part of the technique involves using mutation, crossover, and selection operators with specific parameters. The conventional notation for each solution $\boldsymbol{x}_i$ at generation t is denoted by $\boldsymbol{x}_i^t$ for all $i \in \{1, \ldots, N_P\}$. Note that, several mutation strategies have been suggested for DE; some of them include "DE/$rand$/1", "DE/$best$/1" and "DE/$rand - to - best$/1" listed respectively as:

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + F(\boldsymbol{x}_{r_2}^t - \boldsymbol{x}_{r_3}^t), \tag{1.40}$$

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{best}^t + F(\boldsymbol{x}_{r_1}^t - \boldsymbol{x}_{r_2}^t), \tag{1.41}$$

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + F(\boldsymbol{x}_{best}^t - \boldsymbol{x}_{r_1}^t) + F(\boldsymbol{x}_{r_2}^t - \boldsymbol{x}_{r_3}^t), \tag{1.42}$$

where $\boldsymbol{v}_i^t$ is called mutation vector and r_1,r_2, and r_3 are randomly selected integer numbers in the set $\{1, \ldots, N_P\}$. The scaling factor F is a positive control parameter that scales the size of the difference vector $(\boldsymbol{x}_{r_i} - \boldsymbol{x}_{r_j})$, and $\boldsymbol{x}_{best}^t$ refers to the current individual vector with the lowest cost value in the previous generation. Then cross-over operation between $\boldsymbol{x}_i^t$ and $\boldsymbol{v}_i^t$ is performed to generate a test vector $\boldsymbol{u}_i^t$ as

$$u_{i,j}^t = \begin{cases} v_{i,j}^t, & (rand_j(0,1) < CR)\, or\, (j == j_{rand})\,, \\ x_{i,j}^t, & otherwise, \end{cases} \tag{1.43}$$

where $u_{i,j}^t$ is the j-th element of $\boldsymbol{u}_i^t$. The crossover rate, denoted as CR, is a constant with a defined value in the range $[0, 1]$ and regulates the amount of duplication from the mutation vector. j_{rand} refers to a randomly picked integer from the interval $[1, D]$ where D is the dimension of the input vector [39].

The selection follows the same process as genetic algorithms, where the minimum objective value is chosen for minimization problems. In the next generation t, if the objective function value of the test vector $\boldsymbol{u}_i^t$ is less than or equal to that of the target vector $\boldsymbol{x}_i^t$, the target vector is replaced with the test vector. If not, the target vector remains unchanged, i.e.,

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^t, & f(u_i^t) \leq f(x_i^t), \\ \boldsymbol{x}_i^t, & \text{otherwise}, \end{cases} \tag{1.44}$$

where $f(.)$ is the objective function.

Finally, the pseudo-code of differential evolution is outlined in Algorithm 1. The non-convex nature of the LDPC code design optimization problem makes differential evolution a more effective algorithm when both variable node degree distribution and check node degree distribution are unknown and need to be optimized simultaneously.

Algorithm 1 Differential Evolution Algorithm [40]

Initialize the population $\boldsymbol{x}$ with randomly generated solutions
Set the weight $F \in (0, 1)$ and crossover probability $Cr \in [0, 1]$
while (Stopping criteria) **do**
    for $i = 1$ to n **do**
        For each $\boldsymbol{x}_i$, randomly choose 3 distinct vectors $\boldsymbol{x}_{r_1}$, $\boldsymbol{x}_{r_2}$ and $\boldsymbol{x}_{r_3}$
        Generate a new mutation vector $\boldsymbol{v}_i$ ▷ DE/$rand$/1 is used.
        Generate a random index $J_r \in \{1, \ldots, D\}$ by permutation
        Generate a randomly distributed number $r_i \in [0, 1]$
        for $j = 1$ to D **do**
            Generate $u_{i,j}^t$
        end for
        Update $\boldsymbol{x}_i^t$ using the cost criteria (1.44)
    end for
end while

Figure 1.5. Algorithmic representation of the differential evolution algorithm showing its core operations.

1.4. LDPC Decoding for BEC

A binary erasure channel is a model used to represent a communication channel in which each transmitted bit has a certain probability of being lost or corrupted (erased) during transmission. As symbols are sent over the BEC, they can either be accurately received (probability of $1 - p$) or erased (probability of p), referred to as the erasure probability. LDPC codes use a decoding algorithm known as an LDPC decoder to detect and correct errors in the transmitted data. The decoder works by checking the parity of the received data against a pre-defined matrix and making adjustments based on the results of this check. In order to fully grasp the contributions of this thesis, a review of the peeling decoder and stopping sets, which constitute a crucial constraint on such decoders, is required.

1.4.1. Peeling Decoder

The peeling decoder is a decoding algorithm utilized for LDPC codes that iteratively removes rows of the code's parity-check matrix. This process continues until the decoder either successfully decodes the transmitted message or determines that it is unable to do so. The peeling decoder has low complexity and can be efficiently implemented, making it an optimal choice for communication systems [16].

In the context of a bipartite graph, where one set of nodes represents variable nodes (data) and the other set represents check nodes (parity checks), the peeling decoder operates as follows. Initially, it identifies variable nodes that are connected to only a single check node, as these can be readily resolved. The decoder then peels off or eliminates these variable nodes from the graph, along with their associated edges. This action may result in other variable nodes being left with only one connection to a check node. Consequently, the process iterates, continuously peeling off variable nodes. This iterative procedure persists until all variable nodes have been peeled off, signifying successful data decoding or no more nodes can be peeled off, indicating a decoding failure [36].

1.4.2. Stopping Sets

In the realm of LDPC codes, a stopping set is a group of code bits that, when erroneous, cannot be corrected by the decoding algorithm. This means that the decoder will not be able to fix the errors and fails to recover the transmitted message [16].

Definition 1. *[16] Let $\mathcal{V}$ be the set of variable nodes. A subset $\mathcal{S}$ of $\mathcal{V}$ is called a stopping set if, for every check node c connected to $\mathcal{S}$, the number of edges connecting c and $\mathcal{S}$ is at least 2.*

In the subsequent, a detailed example has been provided that illustrates the workings of the peeling decoder. Furthermore, any stopping sets that might be present are identified, underscoring their role in the decoding process.

Example: For a given code block $C = (v_1, v_2, v_3, v_4, v_5)$, consider $\mathbf{H}$ matrix as

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{bmatrix}, \tag{1.45}$$

where Figure 1.6 shows that the peeling decoder cannot recover the data, and in Figure 1.7, decoding fails.

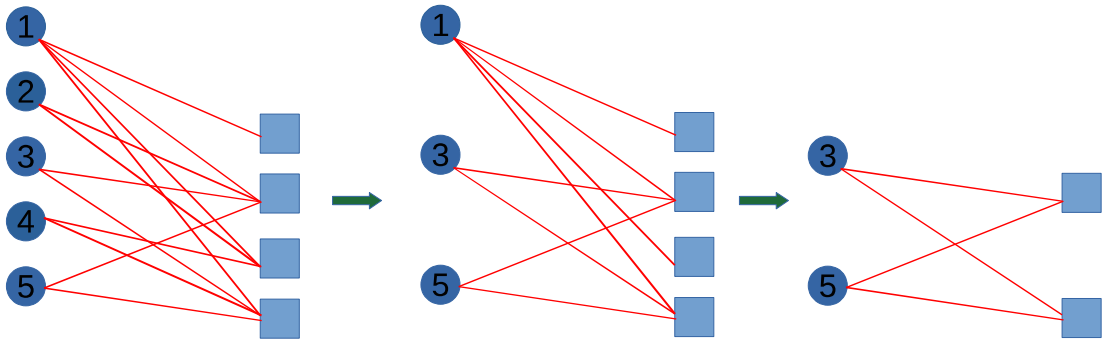

Figure 1.6. Decoding fails, $\mathcal{S} = \{v_3, v_5\}$.

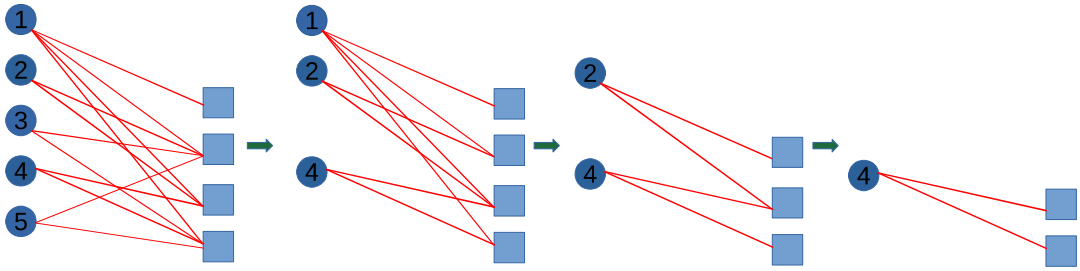

Figure 1.7. Decoding is successful.

1.5. Thesis Structure and Contributions

Here, the structure of the thesis and its significant contributions are presented.

1.5.1. Thesis Structure

The thesis is structured as follows: Chapter 1 provides an introduction to the research areas, background, and related works examined in the thesis. Chapter 2 introduces the concept of repair bandwidth cost and explores the tradeoff between repair bandwidth cost and storage in a novel DSS called BS-assisted cooperative DSS. Chapter 3 presents an in-depth analysis of the balances between the decoding threshold, average repair bandwidth, and code rate, elucidated through the numerical resolution of a specifically designated optimization problem. This chapter also proposes a more efficient procedure called the ideal repair protocol to overcome the limitations of the random access repair protocol. This chapter also examines the joint optimization of the decoding threshold and repair bandwidth in LDPC code design, emphasizing the use of the DE algorithm. Chapter 4 demonstrates how stopping sets are identified and eliminated in a DSS with LDPC-encoded data. This process ensures successful data retrieval from k out of n storage units. Finally, Chapter 5 concludes the thesis.

1.5.2. Thesis Contributions

The contributions of the thesis are: Introducing the concept of repair bandwidth cost and exploring the tradeoff between repair bandwidth cost and storage in a novel DSS called BS-assisted cooperative DSS. Providing an in-depth analysis of the balances between the decoding threshold, average repair bandwidth, and code rate, and proposing a more efficient procedure called the ideal repair protocol to overcome the limitations of the random access repair protocol. Examining the joint optimization of the decoding threshold and repair bandwidth in LDPC code design, emphasizing the use of the DE algorithm. Demonstrating how stopping sets are identified and eliminated in a DSS with LDPC-encoded data.

2. BASE STATION ASSISTED COOPERATIVE DSS

As discussed in Section 1.1, previous studies in the literature have not considered the integration of BS access costs within code design problems [1, 4, 6]. However, a comprehensive approach to designing code involves assessing access costs through stochastic analysis [17]. As a novel approach in the field, this thesis incorporated cost considerations into such code design. In associated DSS frameworks, the stage is set for the development of future regenerating codes by beginning with the establishment of bounds for the tradeoff between repair bandwidth cost and storage size.

2.1. System Model Description for BS-Assisted Cooperative Scenario

This section discusses a hierarchical distributed storage system, where storage nodes across various hierarchical levels have distinct access costs. Figure 2.1 provides an illustration of this concept. The model includes a local cluster with n storage nodes, as well as multiple backup clusters at higher levels, each with n_l storage nodes, where $l = 1, \ldots, M$. Each cluster is assigned a cost factor w_l, and clusters are organized based on their cost factors, with cluster 0 representing the local cluster and cluster M representing the one with the highest cost factor. The model also assumes that each link has a limited capacity b_l, and $\beta_l{}''$ number of symbols can be downloaded from l-th layer, subject to the constraint $\beta_l{}'' \leq b_l\beta$. The information flow graph is characterized by the tuple $\mathcal{G}(n, k, d, t; \alpha, \beta, \beta', \{\beta_l{}''\}_{l=1}^{M})$ and M layers, as depicted in Figure 2.2 for the case of $M = 2$. The data file $\mathcal{F}$ is denoted by source node S at stage $s = -1$. Initial nodes at stage $s = 0$ are represented by pairs of vertices $(x^i_{In_j}, x^i_{Out_j})$, where i refers to the time index and j to the device label. For stage $s \geq 1$, secondary nodes are denoted by the tuple $(x^i_{In_j}, x^i_{Coop_{j,1}}, x^i_{Coop_{j,2}}, x^i_{Out_j})$. Lastly, data collectors (DCs) are represented by a single node at stages $s \geq 1$. In the suggested configuration, the first stage of the repair involves reaching out to d local live nodes within the same cell and downloading β symbols from each one via edges $x^k_{Out_j} \rightarrow x^i_{In_{j'}}$ where $k < i$.

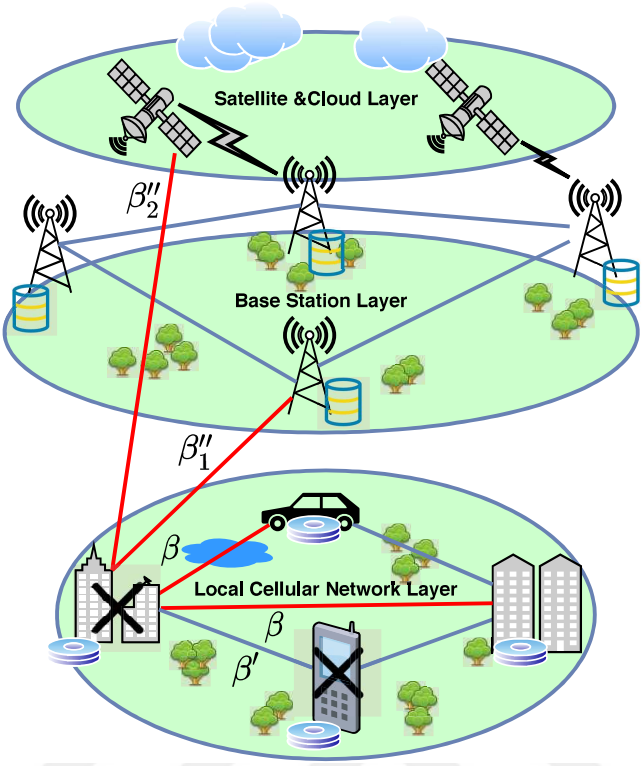

Figure 2.1. An example two-level ($M = 2$) data storage hierarchy. The repair process utilizes local cooperations and higher-layer assistance if needed.

Next, the newcomer storage node connects with potential M higher layers (as $n_l = 1$, contact is made with a server in each layer.) and downloads β_l'' symbols from the l-th layer. Each layer only stores $\mathcal{F}_l \geq b_l\beta$ symbols, where $b_l \in \mathbb{R}$, and the j-th node in the l-th layer has an associated cost of $w_l \geq 1$ per symbol downloaded. In the final phase, all newcomers engage in joint local cooperation by downloading β' symbols from $t-1$ other newcomer nodes within their local cell via edges $x^i_{Coop_{j,1}} \rightarrow x^i_{Coop_{j',2}}$ with $j \neq j'$. The newcomer holds α symbols, denoted by a directed edge from $x^i_{Coop_{j,2}}$ to $x^i_{Out_j}$ with a capacity of α. To successfully rebuild the file, a data collector connects to k living *Out* nodes with unique indices, though not necessarily from the same stage, through k edges with infinite capacity. It is finally realized that the majority of previous descriptions of information flow diagrams found in the literature [1,4] would be a special case of this general description. A cut in an information flow graph is defined as a partition of the set of vertices, $(\mathcal{U}, \overline{\mathcal{U}})$, where S $\in \mathcal{U}$ and DC $\in \overline{\mathcal{U}}$. Each cut is usually associated with the capacity equivalent to the weight of directed edges between vertices in $\mathcal{U}$ and vertices in $\overline{\mathcal{U}}$.

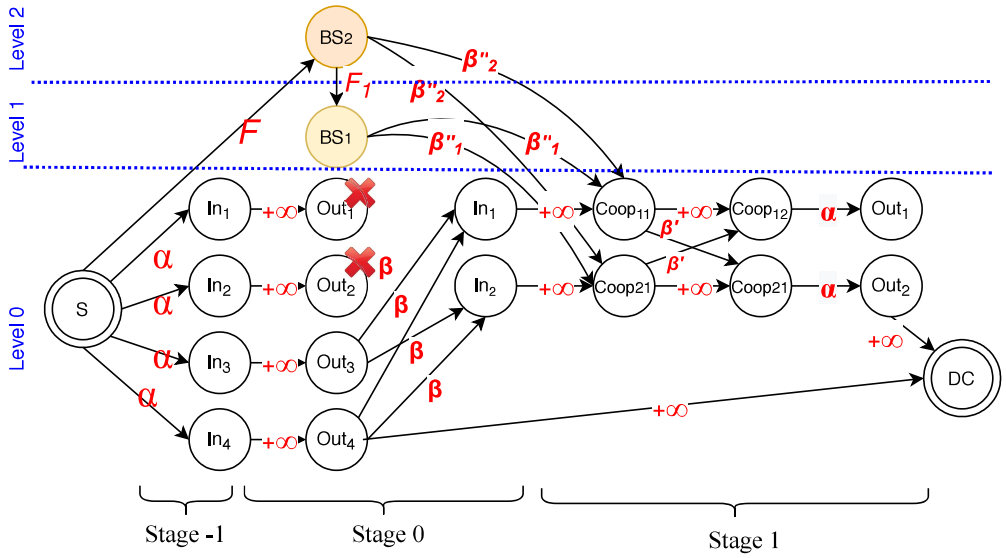

Figure 2.2. Information flow diagram for $M = 2$, $n_1 = n_2 = 1$. For short-hand notation, only subscripts are used.

For a single-source multicast problem, the max-flow min-cut bound states that $\mathcal{F}$ cannot be greater than the capacity of the smallest, or minimum, cut within that network. Dimakis *et al.* [1] identified that the repair problem in distributed storage systems can be associated with a multicast problem. It is observed that the work in [1,4] on distributed storage systems is a specific instance of the suggested repair model in this thesis. The proposed model considers hierarchical clustered structures with various constraints, such as bandwidth costs and link capacities. In this thesis, the cooperative repair framework to BS-assisted cooperative repair is extended, as previously presented by Shum *et al.* [4], by incorporating costly BSs into a DSS under some considerations. In addition, the tradeoff curve on this enhanced DSS is comprehensively characterized by formulating and solving an optimization problem, thereby contributing a novel model to the field. Additionally, an algorithm is introduced specifically designed to identify the minimal number of BSs ($\rho_{\min}$) that participate in the repair process for BS-MSCR and BS-MBCCR points. This algorithm plays a key role in achieving the goal of minimizing repair bandwidth cost within the BS-assisted cooperative repair model.

The key assumptions are listed as follows:

(i) Base stations are assumed to be ordered by their cost factor w.
(ii) Each layer is assumed to contain one cumulative base station with a cost factor w_l, with the $l-$th layer corresponding to the l-th BS.
(iii) The newcomer node downloads $\beta_l{''}$ symbols from the l-th BS.
(iv) The size of $\beta_l{''}$ is a real multiple of β, i.e., $\beta_l{''} = r_l\beta$.
(v) It is assumed that $0 \leq r_l \leq b_l$, where b_l indicates the link capacity.

Without compromising the overall generality of the discussion, the model has been streamlined for clarity. In this simplification, it has been assumed that a single BS is contained in each layer. This decision was made to facilitate a more focused analysis. Additionally, in line with previous work [6], some auxiliary variables r_ls are introduced to express β_l'' in terms of β, i.e., $\beta_l'' = r_l\beta$. The link constraints are characterized by bounding r_l as $0 \leq r_l \leq b_l$ for a fixed $b_l \in \mathbb{R}$. Adhering to the concept of node regeneration, the total repair bandwidth cost per failed node (γ_c) can be calculated as

$$\gamma_c(\mathbf{s}) = d\beta + (t-1)\beta^{'} + \sum_{l=1}^{M} s_l w_l r_l \beta. \tag{2.1}$$

Here, for some $\rho \in 0, 1, \ldots, M$ and $\mathbf{s}$, which is the vector of binary entries s_l, the number of utilized base stations is defined as:

$$\mathbf{s}_\rho = \Big\{(\underbrace{1, ..., 1}_{\rho}, \underbrace{0, ..., 0}_{M-\rho}) : \sum_{l=1}^{M} s_l = \rho\Big\}. \tag{2.2}$$

The minimum cut of the flow diagram $\mathcal{G}$ imposes a constraint on the file size that can be obtained as

$$\min_{r_l, \mathbf{u}\in P} \Big\{u_0\alpha + \sum_{j=1}^{g} u_j(d' - \sum_{i=0}^{j-1} u_i)\beta + u_j\,(t-u_j)\,\beta'\Big\} \geq \mathcal{F}, \tag{2.3}$$

where $P = \{\mathbf{u} = (u_i)_{0\leq i<g} : 1 \leq u_i \leq t \text{ and } \sum_i^g u_i = k\}$ and $d' = d + \sum_{l=1}^{M} s_l r_l$. In this context, u_i represents the number of connecting nodes in each repair group of size t for the data recovery, and g denotes the number of repairing stages. It is important to note that d' is not necessarily an integer, unlike d.

By slightly modifying the steps provided in [4], the file size constraint in (2.3) can be obtained, and it can be further expressed for $g = 0, \ldots, k$ as

$$0 \leq \beta, \beta' \leq \frac{\mathcal{F}}{d}, \tag{2.4}$$

$$b_l \in \mathbb{R}, r_l \in [0, b_l], \quad l = 1, ..., M, \tag{2.5}$$

$$\mathbf{s}_\rho = [s_1, ..., s_M], s_l \in \{0, 1\}. \tag{2.6}$$

Hence, the solution to the repair bandwidth cost-storage tradeoff can be found through the constrained optimization problem described as

$$\min_{\mathbf{s}_\rho, \beta, \beta', (r_i)_{1 \leq i \leq M}} \Big\{ d\beta + (t-1)\beta' + \sum_{l=1}^{M} s_l w_l r_l \beta \Big\}, \tag{2.7}$$

subject to the constraints mentioned above.

Given the similar BS nodes within a layer, it is reasonable to assume that a single BS node is present in each layer without compromising generality. As a result, cost factors of BS nodes can be described using $\mathbf{w} = (w_l)_{1 \leq l \leq M}$. Here, w_l denotes the access cost factor assigned to each BS in l-th layer. To fully characterize the bound on file size, the approach proposed in [4] is built upon, which introduces the notion of different stages in the repair process. This process can be divided into a range of stages, spanning from 0 to s. The zero stage, denoted as $s = 0$, encompasses unrepaired nodes and BS nodes. On the other hand, the stages represented by $s \geq 1$ involve nodes that are actively engaged in the repair process, as illustrated in Figure 2.1. Building on this framework, a bound for file size is derived, and the derived file size bound is given as

$$l_0 \alpha + \sum_{j=1}^{s} \Big(l_j \Big(d' - \sum_{i=0}^{j-1} l_i \Big) \beta + l_j \Big(t - l_j \Big) \beta' \Big) \geq \mathcal{F}, \tag{2.8}$$

where $d' = d + \sum_{j=1}^{M} \beta''_j$ with $\beta''_j = r_j \beta$ and $\sum_{j=0}^{s} l_j = k$ with $1 \leq l_j \leq t$ for $j \geq 1$. Proceeding further, the inclusion of stage s into the file size bound inequality of (2.8) is extended. This is achieved by evaluating two extreme cut sets, $(\rho, l_0, ..., l_s) = (\rho, k - s, 1, ..., 1)$ with s number of 1s and $(\rho, l_0, ..., l_{Q+1}) = (\rho, k - s, t, ..., t, R)$, where $s = Qt + R$ and Q number of ts. By substituting the former tuple into (2.8), the following derivation is obtained:

$$(k - s)\, \alpha + s \Big(d + \sum_{j=1}^{M} r_j - k + \frac{s+1}{2} \Big) \beta + s\, (t-1)\, \beta' \geq \mathcal{F}. \tag{2.9}$$

In a similar manner, the latter case can be expressed as

$$(k-s)\,\alpha + \Big(s\Big(d + \sum_{j=1}^{M} r_j - k\Big) + \frac{s^2 + \Psi_{s,t}}{2}\Big)\beta + (st - \Psi_{s,t})\,\beta' \geq \mathcal{F}. \tag{2.10}$$

2.2. BS-Assisted Cooperative Repair Bandwidth Cost

At times, peer-to-peer connections may encounter significant bandwidth restrictions owing to security constraints or reliability standards related to the power usage of storage nodes. This situation is particularly relevant for local mobile storage devices, where limiting data sharing with new nodes is desirable to manage network traffic and maintain system efficiency. As such, the involvement of additional BSs, which are traditionally considered to be relatively expensive, may not necessarily entail an increase in the cost of repair bandwidth. This strategy thus presents a viable option to enhance system performance and efficiency despite the higher costs associated with the use of BSs. The findings in this thesis underscore the critical role of BSs in enhancing the tradeoff between repair bandwidth and storage size, reinforcing the importance of proper BSs deployment in distributed storage system design.

The presented finding also reveals that using this novel BS-assisted cooperative DSS can potentially reduce the repair bandwidth while simultaneously decreasing the required storage size compared to the solo cooperative repair scenario [4]. This dual functionality, in turn, optimizes repair performance and enhances overall efficiency, thereby demonstrating the profound benefits of this strategy. This section aims to elaborate on a semi-closed form expression for two key operating points on the tradeoff curve that holds particular significance, namely BS-MSCR and BS-MBCCR points, respectively. Additionally, an algorithm is provided that determines the minimum number of repair-contributing BSs in each operating point. And finally, a numerical illustration of the Pareto-front of repair bandwidth cost versus storage size is presented.

2.2.1. Exploring BS-MSCR and BS-MBCCR Points

Theorem 2.1 characterizes the tradeoff between repair bandwidth cost and storage size under BS-assisted cooperative repair. Specifically, it describes the coordinates for two key points on this tradeoff curve, namely BS-MSCR and BS-MBCCR points.

Theorem 2.1. *For a given set values of $r_1, ..., r_M$ in a BS-assisted cooperative repair scenario, the minimum storage and minimum repair bandwidth cost regeneration, namely BS-MSCR and BS-MBCCR points on the tradeoff curve can be characterized as*

$$\left(\gamma^{(\mathrm{BS-MSCR})}, \alpha^{(\mathrm{BS-MSCR})}\right) = \left(\frac{\mathcal{F}(d + \sum_{l=1}^{M} s_i w_i r_i + t - 1)}{k(d + \sum_{l=1}^{M} s_i r_i + t - k)}, \frac{\mathcal{F}}{k}\right), \tag{2.11}$$

$$\left(\gamma^{(\mathrm{BS-MBCCR})}, \alpha^{(\mathrm{BS-MBCCR})}\right) = \left(\frac{\mathcal{F}(2(d + \sum_{l=1}^{M} s_i w_i r_i) + t - 1)}{k(2(d + \sum_{l=1}^{M} s_i r_i) + t - k)}, \frac{\mathcal{F}(2(d + \sum_{l=1}^{M} s_i r_i) + t - 1)}{k(2(d + \sum_{l=1}^{M} s_i r_i) + t - k)}\right), \tag{2.12}$$

where the points $\left(\gamma^{\mathrm{BS-MSCR}}, \alpha^{(\mathrm{BS-MSCR})}\right)$ and $\left(\gamma^{(\mathrm{BS-MBCCR})}, \alpha^{(\mathrm{BS-MBCCR})}\right)$ represent the coordinates for the BS-MSCR and BS-MBCCR points, respectively.

Proof. The proof is given in Appendix A.1. □

It is essential to highlight that the process of determining the operating points for BS-MSCR and BS-MBCCR is intricately tied to the identification of the set of values denoted as r_ls. While this might appear straightforward, the complexity underlying these calculations is profound. Therefore, in order to calculate BS-MSCR and BS-MBCCR operating points, the set of values r_ls need to be identified. Next theorem demonstrates that minimum bandwidth cost is indeed attained at the upper bounds.

Theorem 2.2. *For the given values of* ρ, $\mathbf{s}_\rho = [1, \ldots, 1, 0, \ldots, 0]$ *with* ρ *number of* $1s$ *and* $M - \rho$ *number of* $0s$ *and* $0 \leq r_l \leq b_l$ *for* $l = 1, ..., M$, *satisfying*

$$\sum_{l=1}^{\rho} w_l r_l \leq \bar{w}_t \sum_{l=1}^{\rho} r_l, \tag{2.13}$$

with $\bar{w}_t \in \left\{\frac{d+t-1}{d+t-k}, \frac{2d+t-1}{2d+t-k}\right\}$, *repair bandwidth cost* $\gamma_c \in \left\{\gamma^{(\mathrm{BS-MSCR})}, \gamma^{(\mathrm{BS-MBCCR})}\right\}$ *is minimized at the upper bounds* $[b_1, ..., b_\rho]$.

Proof. The proof is given in Appendix A.2. □

The result of Theorem 2.2 implies that BS-MSCR and BS-MBCCR points can be obtained simply by replacing r_i with b_i as

$$\Big(\frac{\mathcal{F}(d + \sum_{l=1}^{\rho^{(\mathrm{BS-MSCR})}} w_i b_i + t - 1)}{k(d + \sum_{l=1}^{\rho^{(\mathrm{BS-MSCR})}} b_i + t - k)}, \frac{\mathcal{F}}{k}\Big), \tag{2.14}$$

and

$$\Big(\frac{\mathcal{F}(2(d + \sum_{l=1}^{\rho^{(\mathrm{BS-MBCCR})}} w_i b_i) + t - 1)}{k(2(d + \sum_{l=1}^{\rho^{(\mathrm{BS-MBCCR})}} b_i) + t - k)}, \frac{\mathcal{F}(2(d + \sum_{l=1}^{\rho^{(\mathrm{BS-MBCCR})}} b_i) + t - 1)}{k(2(d + \sum_{l=1}^{\rho^{(\mathrm{BS-MBCCR})}} b_i) + t - k)}\Big). \tag{2.15}$$

When the cost factors w_l, where $l = 1, \ldots, M$, assumed to have relatively large values, the points of both BS-MSCR and BS-MBCCR essentially converge to MSCR and MBCR points, respectively. This is primarily due to the fact that b_i, $i = 1, \ldots, M$, equals zero under these circumstances. In addition to the previous scenario, it should be noted that the repair process can be executed without any cooperative actions ($t = 1$ and $\rho = 0$). This specific operational mode leads to the subsequent outcome for minimum storage case:

$$(\gamma^{(\mathrm{MSR})}, \alpha^{(\mathrm{MSR})}) = \Big(\frac{\mathcal{F}d}{k(d+1-k)}, \frac{\mathcal{F}}{k}\Big), \tag{2.16}$$

and, for minimum repair bandwidth case:

$$(\gamma^{(\mathrm{MBR})}, \alpha^{(\mathrm{MBR})}) = \frac{2\mathcal{F}d}{k(2d+1-k)}(1, 1). \tag{2.17}$$

These corner points on the tradeoff curve represent a comprehensive solution, reflecting key outcomes from the field, notably from [1, 4].

2.2.2. Identifying the Minimum Number of Active BSs

In the intricate domain of DSS analysis, the dynamics of BSs and their role in the tradeoff curve have been regarded as pivotal. By focusing on the corner points of this curve, which encapsulate key findings from significant works like [1,4], in this thesis, an exploration has been initiated to identify the optimal number of active BSs necessary for efficient repair process. Though many BSs are characterized by their cost factors $(w_l)_{1\leq l\leq M}$, only a subset directly impacts the repair process, leading to optimal repair bandwidth cost.

Considering a set of BSs characterized by their cost factors $(w_l)_{1\leq l\leq M}$, it's crucial to recognize that only a subset of them, precisely $\rho = \rho_{\min}$, actively participate in the repair process at any specific point on the tradeoff curve. Here, $\rho_{\min}$ is the minimal number of BSs contributing to the repair that results in the minimum repair bandwidth cost. The value of $\rho_{\min}$ is not fixed and can vary depending on the position along the tradeoff curve. For example, as the shift is made from the minimum storage point towards the minimum repair bandwidth cost point, the number of contributing BSs might decrease [41]. However, it is noteworthy that this reduction may not be observed for certain cost factor values.

Even with the comprehensive analysis so far, a gap still exists in identifying the minimum number of BSs necessary for the points on the tradeoff curve. To partially address this gap, efforts are made to determine the minimum number, denoted as $\rho_{\min}$, at corner operating points. The aim is to discover a condition where a BS can positively influence the repair bandwidth cost. Taking this into consideration, a linear time search algorithm is proposed (Algorithm 2) to identify the optimal number of contributing BSs, denoted as $\rho_{\min}^{(\mathrm{BS-MSCR})}$ and $\rho_{\min}^{(\mathrm{BS-MBCCR})}$, for both BS-MSCR and BS-MBCCR operating points. Here, p_t serves as an indicator variable, signifying whether the results correspond to BS-MSCR ($p_t = 1$) or BS-MBCCR ($p_t = 0$). This approach not only enhances the efficiency of the analytical analysis but also paves the way for more sophisticated and targeted repair strategies in future research.

Algorithm 2 Optimal Number of BSs (ρ_{min})

function OptBSNumCal$(k, d, t, M, \mathbf{b}, \mathbf{w}, p_t)$

$\rho_{min} \leftarrow 0$

for $i = 1 : M$ **do**

  $\bar{d} \leftarrow d + \sum_{l=1}^{i-1} w_l b_l \quad , \quad \bar{b} \leftarrow d + \sum_{l=1}^{i-1} b_l$ ▷ $\mathbf{b} = \{b_l\}$

  $\bar{w}_t \leftarrow p_t \left(\frac{\bar{d}+t-1}{\bar{b}+t-k}\right) + (1-p_t)\left(\frac{2\bar{d}+t-1}{2\bar{b}+t-k}\right)$ ▷ $p_t \in \{0, 1\}$

  if $w_i > \bar{w}_t$ **then** ▷ $\mathbf{w} = \{w_l\}$

    break;

  end if

  $\rho_{min} \leftarrow \rho_{min} + 1$

end for

return ρ_{min}

Figure 2.3. Algorithm for the optimal number calculation of active BSs.

2.2.3. Admissible Region

In order to ensure a robust and efficient repair process in the BS-assisted cooperative DSS, a minimum of one BS (referred to as a 'backup node' in [6]) is necessitated. Such BSs are generally selected from the ordered levels of the access cost hierarchy, thereby possessing a cost factor that exceeds one. This specific aspect acts as a constraint on the optimal number of BSs that can be involved in the repair process.

Moreover, it is of paramount importance that such a setting is supported by a feasible BS-assisted cooperative regeneration scheme. Such a scheme should be characterized by specific parameters: the file size $\mathcal{F}$, the repair bandwidth cost γ_c, and a storage size α. This is under the condition of fixed values for the total number of nodes, n, the number of repair helpers in the initial phase, d, the required number of nodes to reconstruct the file, k, and the number of nodes that repaired simultaneously t. Also, deployed BSs have associated cost factors vector $\mathbf{w} = (w_l)_{1 \leq l \leq M}$. In the subsequent, the notion of the *admissible region* is introduced. Essentially, a point (γ_c, α) is termed as admissible if it supports the existence of a BS-assisted cooperative

regeneration scheme, characterized by the stated parameters. The aggregation of all these admissible points, represented as $\mathcal{C}_{AD}(n, d, k, t, \mathbf{w})$, comprises a closure, known as the admissible region. Considering the previous discussion, the identification of this region is achieved by tackling the optimization problem given as

$$\min_{\beta,\beta',(r_i)_{1\leq i\leq M}} \Big\{ d\beta + (t-1)\beta' + \sum_{l=1}^{M} w_l r_l \beta \Big\}, \tag{2.18}$$

subject to file size constraint 1:

$$(k-s)\,\alpha + s\Big(d + \sum_{j=1}^{M} r_j - k + \frac{s+1}{2}\Big)\beta + s\,(t-1)\,\beta' \geq \mathcal{F}, \tag{2.19}$$

and file size constraint 2:

$$(k-s)\,\alpha + \Big(s\Big(d + \sum_{j=1}^{M} r_j - k\Big) + \frac{s^2 + \Psi_{s,t}}{2}\Big)\beta + (st - \Psi_{s,t})\,\beta' \geq \mathcal{F}, \tag{2.20}$$

where $s = 0, \ldots, k$.

Following this, the above optimization problem is solved through numerical methods. The admissible region correlating with this problem is then illustrated in Figure 2.4. The parameters utilized in Figure 2.4 include $k = 6$, $d = 9$, and $t = 3$. Additionally, the BS link capacities and cost factors are defined by vectors $\mathbf{b} = [1, 0.75, 0.5, 0.25]$ and $\mathbf{w} = [1.2, 1.4, 1.8, 1.84]$ respectively. And note that in this specific scenario, each layer accommodates only one BS.

Figure 2.4 offers a comparative illustration between the previously established works [1, 4] and the derived new tradeoff curve. The advantage of the used approach becomes evident as it demonstrates a wider achievable region. This result underscores the effectiveness of the proposed model in finding better points for the tradeoff curve between repair bandwidth cost and storage size. Interestingly, the participation of BSs in the repair process varies depending on the operating point. Specifically, in Figure 2.4, at the BS-MBCCR point, only BSs from the first layer contribute to the repair. In contrast, at the BS-MSCR point, BSs from both the first and second layers are actively involved in the repair process.

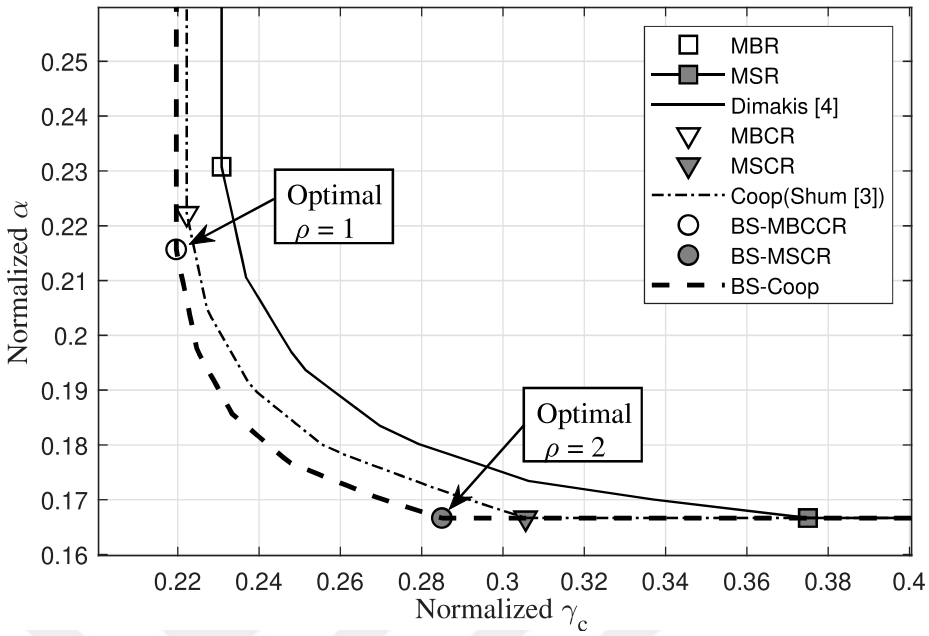

Figure 2.4. Storage versus repair bandwidth cost tradeoff for $k = 6$, $d = 9$, $t = 3$, $\mathbf{b} = [1, 0.75, 0.5, 0.25]$ and $\mathbf{w} = [1.2, 1.4, 1.8, 1.84]$ [41] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2022 IEEE).

2.3. Repair Bandwidth Cost v.s. Storage in Cellular Networks

This section is dedicated to exploring the tradeoff between average repair bandwidth cost and storage size within a cellular network context. Here, a scenario is considered where nodes may randomly enter and leave the network according to a Poisson process. This implies that number of nodes that are cooperatively repaired might vary over time, adding a layer of complexity to the network dynamics. Consequently, the parameter t, which represents the number of repairing nodes in the cooperative case [4], will not remain fixed and could fluctuate over time. These fluctuations could lead to corresponding changes in download costs and network link constraints, presenting a more dynamic scenario for the analysis. To examine this dynamic environment, an optimization problem is set up, which is then solved using numerical methods. The goal is to demonstrate how the inclusion of BSs can potentially enhance the admissible region of the tradeoff curve [42] when the number of repairing nodes varies at different time steps.

In this section, a step further is taken in the repair problem analysis compared to Section 2.2. The main assumption is that the value of t is variable, thereby making the scheme more adaptable and relevant for dynamic DSS applications. This element of adaptability can make the scheme allow for effective operation even under conditions of considerable network variation.

2.3.1. Modeling Node Dynamics in Cellular DSS

In this model, the cellular DSS conforms to the M/M/∞ queuing model, as elaborated in [17]. This system comprises U nodes, where $U > n$. The arrival and departure of nodes within the cell follow a Poisson process with a rate denoted by μ [17]. The number of failed or departed nodes at each repair interval, denoted by Δ, is represented by a truncated Poisson random variable and is bound by the condition that $t \leq \min\{k, n-k\}$. Given the dynamic nature of node presence, it is anticipated that the number of failed or departed nodes at each repair duration Δ is a variable that can change over time [17].

Here, the randomness of node arrivals and departures is captured by assuming that the number of nodes requiring regeneration at each stage equals the number of failed/departed nodes. Hence, a system with the following parameters is considered: $(n, k, d, \alpha, \beta, \{r_l\}_{l=1}^{M}, \{w_l\}_{l=1}^{M}, \Delta, \mu)$, and apply a coding scheme to repair the lost content optimally on average. For successful data reconstruction, it is crucial that the number of failed nodes does not exceed $n-k$. Furthermore, it is a common assumption that $d \geq k$, thus allowing us to posit, without loss of generality, that $t \leq k$ [4, 5]. This statement logically infers that $t \leq \min\{k, n-k\}$.The assumption that node failures occur according to a truncated Poisson process, with a failure rate of μ, is incorporated into the DSS model [17]. Consequently, the probability of node failures during the repair period, represented by Δ, is derived as

$$p(t=i) = \frac{(\mu\Delta)^i}{i! \sum_{j=0}^{\min\{k,n-k\}} \frac{(\mu\Delta)^j}{j!}}, \tag{2.21}$$

where $i \in \{1, ..., \min\{n-k, k\}\}$.

Finally, the average repair bandwidth cost for a given t is expressed as

$$\begin{aligned}\bar{\gamma}_c &= \sum_{i=1}^{\min\{n-k,k\}} p(t=i)\Big\{d\beta + (i-1)\beta' + \sum_{l=1}^{M} w_l r_l \beta\Big\}, \\ &= d\beta + (\bar{t}-1)\beta' + \sum_{l=1}^{M} r_l w_l \beta,\end{aligned} \tag{2.22}$$

where $\bar{t} = \sum_{i=1}^{\min\{n-k,k\}} ip_i$ is the expected lost nodes during Δ [42].

2.3.2. Exploring the File Size Constraint

This subsection uses an information flow graph to describe the file size constraint. The utilized approach is similar to that presented in [4]. The information flow graph is composed of various BS nodes, each of which stores the entire data object. These BSs are capable of offering assistance if their repair participation could result in a decrease in the total repair bandwidth cost for the regeneration node. It is important to note that the composition of the information flow graph evolves over time, to perform repair operations. In the event of node failures, the repair process is carried out in stages, involving a group of t local nodes, with potential assistance from the BSs.

Consider a scenario where DC retrieves a file of size $\mathcal{F}$ by reaching out to k out of n local nodes (as shown in Figure 2.2). In order for the file to be successfully reconstructed, the condition of the max-flow-min-cut theorem [2] must be met between the source S and DC. According to this theorem, the maximum flow of information from the source to DC must match the minimum cut, which is the minimum amount of information that needs to be removed from the graph to disconnect the source S from DC. This means the maximum amount of information flowing from the source S to DC should be equal to, if not greater than, the file size for successful file reconstruction. Consequently, the max-flow-min-cut theorem [2] becomes a critical requirement to satisfy the connection between the source S and DC. Mathematically, this condition can be represented as

$$\min_{u\in P}\Big\{\sum_{i=0}^{g-1} u_i \min\Big\{\alpha, \Big(d - \sum_{j=0}^{i-1} u_j\Big)\beta + (t-u_i)\beta' + \sum_{l=1}^{M} r_l\beta\Big\}\Big\} \geq \mathcal{F}, \tag{2.23}$$

where $u = (u_0, u_1, ..., u_{g-1})$ and $P = \left\{u : 0 \leq u_i \leq t \wedge \sum_{j=0}^{g-1} u_j = k\right\}$, such that $t|k$. In equation (2.23), the max flow-min cut bound is only determined by failed nodes, which results in the requirement $t|k$. However, following the approach in [4], the possibility of initial nodes contribution to the max flow-min cut bound is also accommodated in this analysis. As explained in the previous section, the file size constraint can be stated as a set of inequalities,

$$\alpha(k-g) + g\beta(d + \sum_{l=1}^{M} r_l - k + \frac{g+1}{2}) + g\beta'(t-1) \geq \mathcal{F}, \tag{2.24}$$

$$\alpha(k-g) + \beta\Big(g(d + \sum_{l=1}^{M} r_l - k) + \frac{g^2 + \psi_{g,t}}{2}\Big) + \beta'(gt - \psi_{g,t}) \geq \mathcal{F}, \tag{2.25}$$

where $g = 0, \ldots, k$ and $\psi_{g,t} = \lfloor g/t \rfloor\, t^2 + (g - \lfloor g/t \rfloor\, t)^2$.

The ensuing subsection presents the results of the conducted analysis, elucidating the potential of BS-assisted cooperative repair in optimizing network performance under dynamic conditions.

2.3.3. Solving the Repair Optimization Problem

In this subsection, the objective of this thesis is to tackle an optimization problem that navigates the tradeoff between the average repair bandwidth cost and storage size while considering a variable number of repairing nodes, represented as t. This problem is constrained by both the file size and the link capacities.

Mathematically, the problem is set up to minimize $\bar{\gamma}_c$, as shown in the expression (2.26), which represents the average cooperative repair bandwidth cost. The minimization is over parameters β, β', and the sequence of r_l values, where l ranges from 1 to M. This problem is subject to certain constraints. Specifically, inequalities (2.24) and (2.25) must be satisfied for all values of t equal to $\bar{t}$ and g ranging from 0 to k. Moreover, β and β' must fall within the range of 0 to $\mathcal{F}/d$, where $\mathcal{F}$ denotes the file size and k denotes the number of nodes contacted by the data collector for file reconstruction. Lastly, each r_l value must lie within the interval from 0 to b_l, where b_l is a real number.

In this context, Δ represents the repair interval, while t symbolizes the count of nodes that are lost or fail. Subsequently, the problem is represented as

$$\min_{\beta,\beta',(r_l)_{1\leq l\leq M}} \bar{\gamma}_c, \tag{2.26}$$

subject to inequalities (2.24) and (2.25) for $t = \bar{t}$, $g = 0, \ldots, k$ and

$$0 \leq \beta, \beta' \leq F/d, \tag{2.27}$$

$$r_l \in [0, b_l], \quad b_l \in \mathbb{R}, \quad l = 1, ..., M. \tag{2.28}$$

A plot illustrating the relationship between the average repair bandwidth cost and storage size under various repair scenarios has been created, employing the subsequent parameters: $n = 10$, $d = 9$, $k = 6$, $F = 1$, $\mathbf{b} = [1.2, 0.8, 1.5]$, $\mathbf{w} = [1.2, 1.5, 1.7]$ and $\Delta = 1$. In this plot, the repair bandwidth cost of the BS-assisted cooperative repair scenario is compared with the local cooperative repair scenario and the single node repair scenario, assuming the cost of a symbol download within the cell to be unity. The points on the tradeoff curve corresponding to minimum storage and minimum repair bandwidth cost are of particular interest. Therefore, these points are studied by varying the number of local helping nodes, d, over a range of values. Figure 2.5 demonstrates that the BS-assisted cooperative repair expands the capacity region of both the local cooperative scheme (as described in [4]) and the single node repair scheme (as described in [1]). However, at the point of minimum repair bandwidth cost, the difference between the curves is significantly smaller compared to the corresponding point for the minimum storage repair scheme.

Several plots have been generated to demonstrate the relationship between the average repair bandwidth cost and storage size under different repair scenarios [1, 4, 41]. The parameters used to generate this plot are as follows: the total number of nodes, n, is 10; the number of helping nodes, d, is 9; the number of nodes required to successfully retrieve the data, k, is 6; the file size, $\mathcal{F}$, is 1; the vector link capacities, $\mathbf{b}$, is $[1.2, 0.8, 1.5]$; the cost factors vector, $\mathbf{w}$, is $[1.2, 1.5, 1.7]$; and the repair interval, Δ, is 1. In this plot, the repair bandwidth cost is assessed for three distinct scenarios: BS-assisted cooperative repair, local cooperative repair, and single node repair, where the cost of downloading a symbol within the cell is assumed to be unity.

Points of particular interest on the tradeoff curve are those corresponding to minimum storage and minimum repair bandwidth cost. Further exploration of these points was conducted by varying the number of local helping nodes, d, across a spectrum of values. As illustrated in Figure 2.5, the BS-assisted cooperative repair extends the admissible region of both the local cooperative scheme and the single-node repair scheme. However, it is important to note that the gap between the curves is considerably smaller at the point of minimum repair bandwidth cost compared to the point of minimum storage repair scheme.

Figure 2.6 illustrates the role that local nodes play in aiding the regeneration of other nodes in the network. Notably, the variability of repair bandwidth cost in scenarios involving BS assistance is relatively less compared to other data regeneration methods, especially at the minimum storage point. To analyze this, MATLAB's *fmincon* function is utilized, which allows for the optimization of constrained nonlinear multi-variable functions.

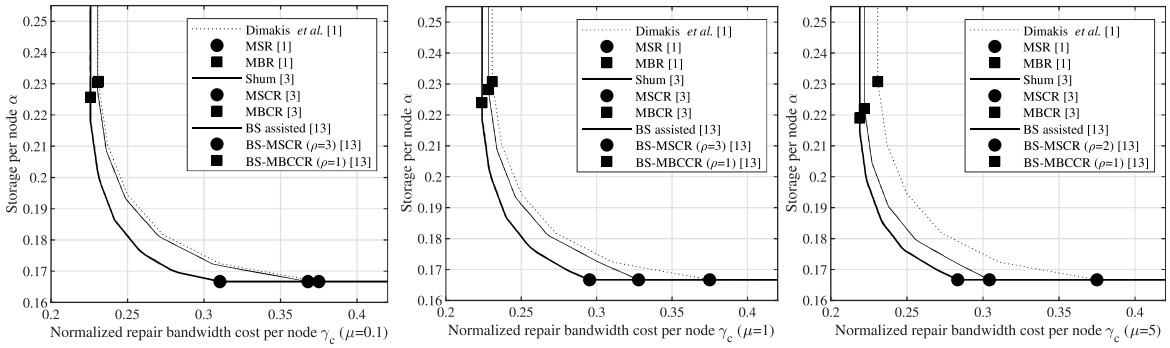

Figure 2.5. Bandwidth cost v.s. storage size for $\mathcal{F} = 1$, $n = 10$, $k = 6$, $d = 9$, $\Delta = 1$, $\mathbf{b} = [1.2, 0.8, 1.5]$ and $\mathbf{w} = [1.2, 1.5, 1.7]$ [42] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2021 IEEE).

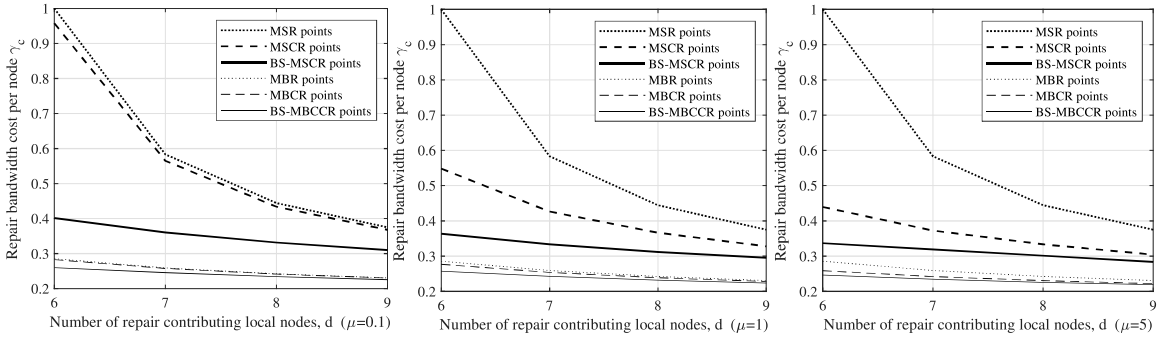

Figure 2.6. Bandwidth cost v.s. d for $\mathcal{F} = 1$, $n = 10$, $k = 6$, $\Delta = 1$, $\mathbf{b} = [1.2, 0.8, 1.5]$ and $\mathbf{w} = [1.2, 1.5, 1.7]$ [42] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2021 IEEE).

2.4. BS-Assisted MSCR Code Construction

As mentioned in some of the previous studies [4], the benefits of exact repair, including its superior maintenance efficiency relative to functional repair, have been substantially recognized. Exact MSR and MSCR code constructions to a specified set of code parameters have already been demonstrated in prior works [43]. This section details the proposal of a new family of regeneration codes, specifically tailored for BS-assisted cooperative repair, characterized by parameters ρ and $d = k \leq n - t$. For simplicity, the assumption $b_l = 1$ is made. Building on existing work, multiple MDS codes of length n with dimension k are employed. These consist of symbols from $GF(p^q)$, with p being prime and $n \leq p^q$ for a given positive integer q. A consistent set of $k \times n$ generator matrices $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_n]$ is utilized, where $\mathbf{G}$ is a $k \times k$ invertible matrix and $\mathbf{g}_i$ denotes the i-th column of $\mathbf{G}$. The file of size F is partitioned into $k(t + \rho)$ chunks with symbols drawn from $GF(p^q)$. These chunks are restructured into $(t+\rho) \times k$ message matrices $\mathbf{M}$. The matrices $\mathbf{MG}$ are multiplied and, for $j = 1, \ldots, n$, the j-th column of the output is distributed to the j-th node for storage. The notation $\mathbf{m}_1^T, ..., \mathbf{m}_{t+\rho}^T$ is employed to denote each row of $\mathbf{M}$. Here, k message symbols in each row are configured into a vector form $\mathbf{m}_i^T = [m_1, ..., m_k]$, where each $m_i \in GF(p^q)$ is encoded into the codeword $\mathbf{m}_i^T\mathbf{G}$. Consequently, the j-th node would store $\mathbf{m}_i^T\mathbf{g}_j$, for $i = 1, \ldots, t + \rho$.

In the event of permanent unavailability of nodes with indices $j_1, j_2, ..., j_t$, the l-th newcomer (j_l) connects to any $d = k$ remaining nodes, symbolized as $\pi(1), \ldots, \pi(k)$. It downloads $\mathbf{g}_{\pi_l(1)}, \ldots, \mathbf{g}_{\pi_l}(k)$ and successfully computes/reconstructs $\mathbf{m}_l^T$. During the cooperative phase, the newcomer indexed by j_l computes $\mathbf{m}_l^T \mathbf{g}_{j_h}$ to send to the newcomer indexed by j_h (where $h \neq l$). Each newcomer exchanges $t - 1$ chunks in total. Following this phase, the l-th newcomer indexed by j_l possesses t chunks, specifically $\mathbf{m}_h^T \mathbf{g}_{j_l}$ for $h = 1, \ldots, t$. In the final regeneration phase, the remaining ρ chunks are downloaded from the available ρ base stations, each providing one chunk of information. Note that the last two phases of regeneration can be exchanged. A chunk of information is of size $\frac{F}{k(t+\rho)}$ symbols, and each newcomer downloads $d + \rho + t - 1$ chunks, achieving the minimum bandwidth as reported in [41].

3. OPTIMIZED LDPC CODE CONSTRUCTIONS FOR DSS

3.1. Introduction

In the intricate world of digital communication, the complexities and interdependencies of repair mechanisms within the framework of LDPC codes have been deemed crucial to understand. Therefore, this thesis has been dedicated to a thorough exploration of its associated relationships. Considering this, in this chapter, a thorough analysis of the relationship between decoding threshold and average repair bandwidth (or code rate) is carried out through the numerical resolution of a specified optimization problem.

The study illuminates the varying impacts of distinct repair protocols, namely the random access repair protocol and the ideal repair protocol, on this relationship. Special attention is devoted to investigating the effects of different code rates on the tradeoff curve within LDPC codes family for DSSs. This examination offers insightful revelations regarding the tradeoff between the average repair bandwidth and the decoding threshold for specific code rates, reinforcing the pivotal role that these parameters play in optimizing repair strategies within the context of LDPC codes.

LDPC codes, pivotal in digital communications, are characterized by a sparse parity-check matrix, often denoted as $\mathbf{H}$, and represented by a Tanner graph which is a bipartite graph. A crucial aspect of LDPC codes lies in the structure of this graph and matrix. This structure, represented as a bipartite graph, consists of variable nodes and check nodes, each playing a significant role in the decoding process. In the context of a $(6, 3)$ LDPC code, for instance, the bipartite graph consists of 6 variable nodes and 3 check nodes, each represented by circles and squares, respectively, is illustrated in Figure 3.1. Understanding this bipartite representation is key to unlocking the full potential and efficiency of LDPC codes in various communication scenarios.

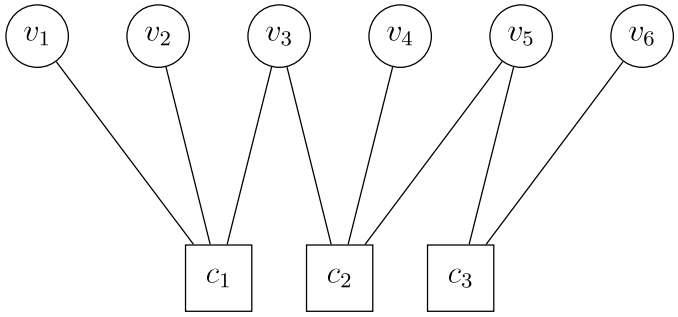

Figure 3.1. Bipartite graph for (6,3) LDPC code.

The matrix **H** is a parity-check matrix characterized by a bipartite graph shown in Figure 3.1, and it given as

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \tag{3.1}$$

Variable nodes, denoted as $v_i, i = 1, \ldots, n$, represent the elements of the rows in the parity check matrix. Similarly, check nodes, denoted as $c_i, i = 1, \ldots, n-k$, represent the elements of the columns in this matrix. Connections between variable nodes and check nodes are symbolized by the value 1 in the matrix. One of the key aspects of these LDPC codes is their degree distributions, which are used to categorize the codes as either regular or irregular. Furthermore, the maximum degree of variable nodes and check nodes, as indicated by $d_{v_{\max}}$ and $d_{c_{\max}}$, respectively. The degree of a node here refers to the count of edges or connections the node has. The code rate, R, is another vital characteristic of an LDPC code. This represents the efficiency of the code, calculated using $\lambda(x)$ and $\rho(x)$, which are degree distributions of variable and check nodes respectively. It is given by the equation $R = 1 - \frac{\bar{d}_v}{\bar{d}_c}$, where $\bar{d}_v$ is the average degree of variable nodes and $\bar{d}_c$ is the average degree of check nodes.

Belief Propagation (BP) is a decoding algorithm used for LDPC codes. Its performance is analyzed using a method called density evolution, which tracks the erasure probability across decoding iterations. The erasure probability after l iterations of BP decoding is given by the equation $x_l = \epsilon\lambda(1 - \rho(1 - x_{l-1}))$, where $x_0 = \epsilon$ is the initial channel erasure probability. The decoding threshold is a significant characteristic of the BP decoder. It represents the maximum channel erasure prob-

ability that the decoder can effectively handle. The decoding threshold is given by $\epsilon^* = \sup\Big\{\epsilon \in (0,1] : \lim_{l\to\infty} x_l = 0\Big\}$. Finally, the convergence of the density evolution formula depends on a factor known as the stability condition. This condition essentially checks whether $x_0\lambda'(0)\rho'(1)$ is smaller or greater than one. If the value is less than one, the erasure probability x_l will converge to zero as l approaches infinity, implying successful decoding. However, if the value is greater than one, the formula may not converge, signaling potential issues in decoding.

In this thesis, for LDPC codes, the reduction of the average repair bandwidth in the case of the random access repair protocol is demonstrated, underscoring the effectiveness of check node degree relaxation in code design optimizations. Moreover, a comparative study is presented, demonstrating the superior performance of the ideal repair protocol over the random repair protocol. The findings also confirmed that while higher code rates contribute to greater efficiency in raw data storage, they simultaneously necessitate larger repair bandwidths. In summary, this chapter presents a comprehensive study into the intricate dynamics of repair bandwidth, decoding threshold, and repair protocols within the family of LDPC code ensembles.

3.2. Novel Repair Protocol and Primary Findings

In this section, the main findings, which encompass theoretical advancements, are showcased, focusing specifically on the suggested repair protocols and their effects on the average repair bandwidth and the decoding threshold. Subsequent sections will further furnish numerical outcomes.

This section details the exploration of two distinct repair protocols. Under the assumption that the code length extends infinitely and that the constructed code's Tanner graph possesses a tree-like structure, it is thus feasible to use the density evolution formula for the performance assessment of the constructed code. Additionally, when a node failure occurs, the repair mechanism is activated to search for all parity equations that include the lost symbol. Subsequently, a selection from the available

parity equations is made in accordance with the repair protocol. Due to the irregularity inherent in the parity check matrix $\mathbf{H}$, an exact determination of the repair bandwidth proves nearly unattainable. However, it is feasible to formulate the repair bandwidth in an average sense. The two protocols under consideration vary significantly; the first is based on random selection, while the second one, due to its optimal nature, is referred to as the ideal repair protocol. Furthermore, the analysis proposed herein corresponds with that in [15] when it pertains to the random access repair protocol.

Definition 2. *Random access repair protocol: Upon the failure of a variable node (symbol erasure), the replacement node regenerates its data by randomly selecting from the list of available variable nodes defined by the check relations.*

The utilization of the random access repair protocol can be implicitly observed in [15]. This protocol involves a check node $r \in \{1, \ldots, m\}$ with degree $d_r^{(c)}$, connected to $d_r^{(c)}$ variable nodes when m is the number of total check nodes. In this scenario, the average repair bandwidth per variable node is presented as

$$\bar{\gamma} = \frac{\sum_{i=1}^{m} d_r^{(c)}(d_r^{(c)} - 1)}{E}. \tag{3.2}$$

Here, E denotes the total number of edges in the corresponding Tanner graph. It has been demonstrated in [15] that the minimum of $\bar{\gamma}$ can be reached when the regularity of check nodes is enforced.

Next, this thesis proposes a process to determine $\rho(x)$ by minimizing (3.2). As advised by Richardson *et al.* [36], for the creation of near-optimal degree distributions, they proposed the usage of continuous degrees, a sort of relaxation on the integer constraints. This suggestion can be easily understood through the fractional phantom distribution, detailed in [36]. The aforementioned approach can also be leveraged to compute the real minimum repair bandwidth as expressed in (3.2). Specifically, for a fixed value of E, the minimum value of (3.2) can be expressed as $\bar{\gamma}_{\min}^{(\text{rand})} = d - 1$, where $d_i^{(c)} = d$, and $i = 1, ..., m$. If $\bar{\gamma}_{\min}^{(\text{rand})}$ is non-integer, then d must be real number as well. Considering this, d should be distributed in accordance with a fractional phantom distribution [36].

Theorem 3.1. *For an LDPC code ensemble, achieving the minimum average repair bandwidth, with* $\bar{\gamma}_{\min}^{(\text{rand})} = d - 1$*, the check node degree distribution is given by*

$$\rho(x) = \begin{cases} x^{d-1}, & d \in \mathrm{Z}, \\ (\lceil d \rceil - d)\, x^{\lfloor d \rfloor - 1} + (d - \lfloor d \rfloor)\, x^{\lfloor d \rfloor}, & d \notin \mathrm{Z}. \end{cases} \tag{3.3}$$

Proof. The proof is given in Appendix B.1. □

As a result of both check and variable nodes being regular, the value of $\bar{\gamma}_{\min}^{(\text{rand})}$ can be obtained with $d_{c_{\max}} = \frac{d_{v_{\min}}}{1-R}$ and $d_{v_{\max}} = d_{v_{\min}}$, where $d_{v_{\min}}$ is defined as the minimum value of the variable node degrees [15]. It is also clear that the minimum value of the decoding threshold $\epsilon^*_{\min}$ is obtained when the code rate is $R = 1 - \frac{2}{d_c}$. As a consequence, for any LDPC code with constant code rate-R and $\epsilon^* = \epsilon^*_{\min}$, a closed form expression for $\bar{\gamma}_{(\min)}^{(\text{rand})}$ is attainable and is presented in Theorem 3.2.

Theorem 3.2. *Using random access repair protocol, for a code rate-R, the minimum average repair bandwidth* $\bar{\gamma}_{(\min)}^{(\text{rand})}$*, when* $\epsilon^* = \epsilon^*_{\min}$ *,is determined by*

$$\bar{\gamma}_{\min}^{(\text{rand})} = f(R) - 1, \tag{3.4}$$

where $f(R) = \lfloor \phi_R \rfloor + \lceil \phi_R \rceil - \frac{1}{\phi_R} \lfloor \phi_R \rfloor \lceil \phi_R \rceil$ *and* $\phi_R = \frac{2}{1-R}$.

Proof. The proof is given in Appendix B.2. □

In order to offer a more detailed insight into Theorem 3.2, Table 3.1 has been presented as a tangible representation. In this table, the implications of Theorem 3.2 are demonstrated across a range of selected code rates. Through the entries of the table, insights into the intricate relationship between the code rate-R and its impacts on the minimum average repair bandwidth, the decoding threshold, and the node degree distributions can be gleaned. For instance, with increasing code rates, larger values are required for check node degrees. In this manner, a clearer comprehension of the theorem's underlying dynamics in relation to varying code rates is facilitated.

Table 3.1. The demonstration of Theorem 3.2 for selected code rates.

R	$\bar{\gamma}_{\min}^{(\text{rand})}$	$\epsilon_{\min}^{*}$	$\lambda(x)$	$\rho(x)$
0.55	3.5	0.2860	x	$0.5x^3 + 0.5x^4$
0.65	4.75	0.2109	x	$0.25x^4 + 0.75x^5$
0.7	5.7	0.1758	x	$0.3x^5 + 0.7x^6$
0.75	7	0.1432	x	x^7
0.85	12.35	0.0814	x	$0.65x^{12} + 0.35x^{13}$

Random access repair protocol is not anticipated to yield optimal performance for LDPC codes, given its random selection of neighboring check nodes for data repair. As a response to this issue, the thesis proposes a more efficient procedure termed ideal repair protocol. The concept of ideal repair protocol is that a check node with the smallest degree from the available check nodes is selected to repair the failed node. Subsequently, a theorem is formulated that elucidates the method for calculating the average repair bandwidth for this protocol based on the parameters $\lambda(x)$ and $\rho(x)$.

Definition 3. *Ideal repair protocol: In this protocol, the selection of the check relation to repair the failed symbol is performed based on the degree number, i.e., the check relation with the minimum elements (smallest degree) is chosen to complete the data (bit) repair.*

As a consequence of the previous definition of ideal repair protocol, an expression for the average repair bandwidth can be derived.

Theorem 3.3. *For ideal repair protocol, the average repair bandwidth of a failed node(symbol) for an LDPC code ensemble defined by $(\lambda(x), \rho(x))$ is calculated as*

$$\bar{\gamma}^{(\text{ideal})}(\lambda, \rho) = \bar{d}_v \sum_{j=2}^{d_{v\max}} \frac{\lambda_j}{j} \Big(\sum_{\underline{n} \in C(j)} P(\underline{n}, j) \gamma_{\underline{n}} \Big), \tag{3.5}$$

where $C(j)$ is the set of all $\underline{n}$s where $\underline{n} = \{n_i : \rho_i > 0, i = 2, \ldots, d_{c\max}\}$ satisfying $\sum_{i=2}^{d_{c\max}} n_i = j$ and n_i is the number of edges connected to check nodes with degree i, i.e., $C(j) = \{\underline{n} : \sum_{i=2}^{d_{c\max}} n_i = j, \rho_i > 0\}$. Moreover, when edge selections are made

independently, $P(\underline{n}, j)$ *can be given by*

$$P(\underline{n}, j) = \frac{j!}{n_{i_1}!...n_{i_{|\underline{n}|}}!} \prod_{k \in \{i_1,...,i_{|\underline{n}|}\}} (\rho_k)^{n_k}, \tag{3.6}$$

and $\gamma_{\underline{n}} = \min\{i_1, ..., i_{|\underline{n}|}\} - 1$*, where* $i_k \in \{2, \ldots, d_{c_{\max}}\}$ *is the distinct node degree.*

Proof. The proof is given in Appendix B.3. □

In the following example, $\bar{\gamma}^{(\text{ideal})}(\lambda, \rho)$ is computed for specific node degree distributions.

Example: Let $\lambda(x) = x$ and $\rho(x) = 0.5x + 0.5x^2$, then $C(2) = \{\{n_2 = 2\}, \{n_3 = 2\}, \{n_2 = 1, n_3 = 1\}\}$. Consequently, $\bar{\gamma}^{(\text{ideal})}(\lambda, \rho)$ can be calculated as

$$\bar{\gamma}^{(\text{ideal})}(\lambda, \rho) = (0.5)^2(1) + (0.5)^2(2) + 2!(0.5)^2(1) = 1.25.$$

In the following section, an in-depth examination of the joint optimization of the decoding threshold and repair bandwidth in LDPC code design is conducted. Emphasis is placed on the usage of the differential evolution algorithm in this process. The differential evolution algorithm, with its distinctive capability, is preferred since it simplifies complex multi-objective problems into manageable single-objective tasks. This transformation is achieved through the implementation of the epsilon constraint method. To clarify this intricate process, a comprehensive algorithm is provided, illustrating the practical implementation of the LDPC code design using the DE methodology.

3.3. Incorporating Repair Protocols in LDPC Code Design

This section integrates the repair strategies previously developed for a general DSS with LDPC code design. For a binary erasure channel, optimal node degree distributions that satisfy the reliability constraint are identified by the capacity-approaching design of LDPC codes. However, since one of the primary objectives of this thesis is minimizing the repair bandwidth, the joint optimization of the decoding threshold and repair bandwidth for a specific code rate is formulated and then numerically solved.

The initial approach to LDPC code design optimization problem used linear programming with a fixed check node degree distribution [37]. However, this made finding optimal node degree distributions complex. By removing this assumption, the LDPC code design becomes a nonlinear optimization problem. It is noted that, depending on the repair protocol, the optimization variables could be the coefficients of both the variable and check node degree distributions or solely the variable node degree distribution. Under the ideal repair protocol, it is required to ascertain the coefficients of both the variable and check node degree distributions. Additionally, the density evolution formula can converge to different fixed points [16]. This property leads to a susceptibility of classical optimization techniques, which often focus on a specific region. Hence, following the path similar to the study conducted in [36], a variant of the DE algorithm is utilized to solve this nonlinear optimization problem, aiding in avoiding such fixed points to achieve a global minimum. As presented here, the asymptotic repair-efficient code design optimization problem can be regarded as a bi-objective optimization problem, with the average repair bandwidth $\bar{\gamma}^{(\text{ideal})}(\lambda, \rho)$ and the targeted decoding threshold $1 - \epsilon^*(\lambda, \rho)$ as objectives when ideal repair protocol is consider.

For minimizing this bi-objective optimization problem, using the epsilon constraint method, it is only needed to minimize one objective function while constraining other objective function. For example, when targeted decoding threshold $\epsilon^*(\lambda, \rho)$ is constrained as $0 < \epsilon^*(\lambda, \rho) < 1 - R_0$, such optimization problem can be described as

$$\begin{aligned} (\hat{\lambda}, \hat{\rho}) &= \arg \min_{(\lambda,\rho)\in\mathcal{S}} \left\{ \bar{\gamma}^{(\text{ideal})}(\lambda, \rho) \right\}, \\ \text{subject to} \quad & \epsilon^*(\lambda, \rho) \geq \epsilon_0, \\ & R(\lambda, \rho) \geq R_0, \end{aligned} \tag{3.7}$$

where R_0 is the minimum code rate requirement as well as ϵ_0 indicates the channel erasure rate and the region $\mathcal{S}$ is denoted as

$$\mathcal{S} = \left\{ (\lambda, \rho) : \sum_{i=2}^{d_v} \lambda_i = 1, \sum_{i=2}^{d_c} \rho_i = 1, \lambda_i \geq 0, \rho_i \geq 0 \right\}. \tag{3.8}$$

Similarly, when the objective function is $1 - \epsilon^*(\lambda, \rho)$, the optimization problem that

maximizes the decoding threshold and constraints repair bandwidth is expressed as

$$\begin{aligned} (\hat{\lambda}, \hat{\rho}) &= \arg \min_{(\lambda,\rho)\in\mathcal{S}} \left\{1 - \epsilon^*(\lambda, \rho)\right\}, \\ \text{subject to} \quad & \bar{\gamma}^{(\text{ideal})} \leq \gamma_0, \\ & R(\lambda, \rho) \geq R_0, \end{aligned} \tag{3.9}$$

where γ_0 indicates the maximum tolerable repair bandwidth. By replacing $\bar{\gamma}^{(\text{ideal})}$ with $\bar{\gamma}^{(\text{rand})}$, the aforementioned problems are optimized specifically for random access repair protocol.

It is noted that optimization variables can either be coefficients of both the variable and check node degree distributions or just the variable node degree distribution, depending on the repair protocol. This way, if $\bar{\gamma}_{\min}^{(\text{rand})}$ is constant, $\rho(x)$ is explicitly derived using Theorem 3.1. Using this repair protocol, all that remains unknown are the variable node degree distribution coefficients. Moreover, for the ideal repair protocol, the coefficients of both the variable and check node degree distributions must be determined.

3.3.1. LDPC Code Design Optimization Using DE Algorithm

Here, the utility of DE, an evolutionary algorithm, is explored for the design of asymptotic repair-efficient codes. Herein, an algorithmic perspective is also provided on the implementation of DE to both random and ideal repair protocols.

Compared to the gradient-based techniques that are greedy and only feasible points play a role in finding the optimal solution, evolutionary techniques are often probabilistic techniques in that combination of infeasible points also may lead to the optimal solution or feasible points. Therefore, instead of completely rejecting infeasible points, they can be retained in the population by giving them higher costs with the hope that their combinations move forward to the feasible points. Therefore, considering only a feasible population in the initial phase of any evolutionary algorithm may lead to a local minimum. However, due to the problem nature, this proposition may be violated [44]. Evolutionary algorithms are designed such that infeasible points are pushed to be

feasible first, then among feasible points, objectives are minimized. In compliance with the previous works [36,37], the thesis also uses a variant of DE optimizer that is simple to implement and efficient for finding global numerical solutions. The main advantages of differential evolution to other evolutionary techniques are that firstly, updating costs is performed locally, which is computationally more efficient. Secondly, the mutation is accomplished in a smarter fashion. Since the DE technique works based on a single cost function, transforming the multi-objective problem to a single objective seems to be critical. In addition, constraints can be handled by a constraint handling method that is detained in [3]. Despite this, since DE has been demonstrated to be useful for designing storage-efficient LDPC codes [36, 37], a variant of it is used in this work to find repair-efficient codes. Because DE is based on a single objective (or cost) function, it is essential to transform a multi-objective problem into a single objective problem using the epsilon constraint method [38]. This method preserves one objective as the main objective function while constraining the rest. Then constraints are handled by a constraint handling method in that the constraint $g(x) \geq g_0$ is forced to be satisfied by adding the related cost, i.e., $\max(1 - g(x)/g_0, 0)$, to the main objective function using a multiplier [38]. In the case of random access repair protocol, the main cost function is formulated as $1-\epsilon^*(\lambda, \rho)$ where $\epsilon^*(\lambda, \rho)$ is the decoding threshold. Whereas in ideal repair protocol, the optimization problem is converted into a single objective one by utilizing the epsilon constraint method [38]. For instance, when $\bar{\gamma}^{(\text{ideal})}(\lambda, \rho)$ is considered as the main cost function, the optimization problem can be formulated as

$$\arg \min_{(\lambda,\rho)\in\mathcal{S}} \left\{ \bar{\gamma}^{(\text{ideal})}(\lambda, \rho) + \xi \max(1 - \epsilon_t(\lambda, \rho)/\epsilon_0, 0) + \xi \max(1 - R(\lambda, \rho)/R_0, 0) \right\}, \quad (3.10)$$

where $\mathcal{S} = \{(\lambda, \rho) : \sum_{i=2}^{d_{v\max}} \lambda_i = 1, \sum_{i=2}^{d_{c\max}} \rho_i = 1, \lambda_i \geq 0, \rho_i \geq 0\}$. Additionally, $\epsilon^*(\lambda, \rho) \geq \epsilon_0$ and $R(\lambda, \rho) \geq R_0$ with ϵ_0 as a fixed decoding threshold that optimizer can tolerate. In addition, R_0 is the minimum code rate requirement, and ξ is a large number that forces the optimization problem to satisfy the constraints.

Despite satisfying the single objective in DE, there are three major control parameters: crossover coefficient $C_r \in (0, 1)$, mutation coefficient $\beta \in [0, 1]$, and population size N_P [40]. There are also four steps involved in the DE: Initialization of

parameter vectors $\boldsymbol{x}_i = (\lambda_i(x), \rho_i(x)), i = 1, \ldots, N_P$, mutation with difference vectors, crossover, and selection. There are different mutation strategies such as DE/rand/1 and DE/rand-to-best/1,including [39]:

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + F(\boldsymbol{x}_{r_2}^t - \boldsymbol{x}_{r_3}^t), \tag{3.11}$$

$$\boldsymbol{v}_i^t = \boldsymbol{x}_{r_1}^t + F(\boldsymbol{x}_{best}^t - \boldsymbol{x}_{r_1}^t) + F(\boldsymbol{x}_{r_2}^t - \boldsymbol{x}_{r_3}^t), \tag{3.12}$$

where $\boldsymbol{x}_{r_i}^t$, $i = 1, 2, 3$ are some selected individuals such that r_1, r_2, r_3 are randomly generated integers in the interval $[1, N_P]$, and $\boldsymbol{v}_i^t$ and $\boldsymbol{x}_{best}^t$ are mutation vector and the best individual in the current iteration, respectively. In addition, a crossover is also performed such that the current individual $\boldsymbol{x}_i^t$ and its corresponding mutation vector $\boldsymbol{v}_i^t$, $i \in [1, N_P]$, are controlled by C_r. In addition, the following changes are applied for a better performance. First, instead of having a fixed mutation strategy, the algorithm iterates randomly between two mutation strategies, increasing the diversity of the synthetic individuals. A mutation strategy is randomly selected for each evaluation of the current individual. Secondly, degrees of variable nodes and check nodes that are integers are transformed to real degrees without violating node degree distribution using the fractional phantom distribution [36]. For example, the pseudo-code of the proposed DE-based algorithm for ideal repair protocol is summarized in Algorithm 3. A similar algorithm can be used when random access repair protocol is considered.

Table 3.2. Some optimized node degree distributions for proposed repair protocols for $R = 2/3$ [45] (This Table was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE).

Protocol	ϵ^*	$\bar{\gamma}$	Node degree distributions
rand	0.306	7.5	$\lambda(x) = 0.4303x + 0.1918x^2 + 0.0707x^3 + 0.1913x^4 + 0.1159x^5$ $\rho(x) = 0.5x^7 + 0.5x^8$
ideal	0.312	7.39	$\lambda(x) = 0.3506x + 0.1673x^2 + 0.1704x^3 + 0.1568x^6 + 0.0540x^7$ $+0.1008x^8$ $\rho(x) = 0.0095x^5 + 0.1477x^6 + 0.1138x^7 + 0.1459x^8 + 0.1847x^9$ $+0.1575x^{10} + 0.1132x^{11} + 0.0355x^{12} + 0.0922x^{13}$

Algorithm 3 Find $(\lambda_s(x), \rho_s(x))$ pair approaching $\bar{\gamma}_{\min}^{(\text{ideal})}$

Input: code rate: R_0, decoding threshold: ϵ_0, maximum variable node degree: $d_{v_{\max}}$, maximum check node degree: $d_{c_{\max}}$, number of non-zero variable node degree: $Numd_v$, number of non-zero check node degree: $Numd_c$, maximum number of iterations: $MaxIt$, population size: N_P, crossover coefficient: C_r

Output: $\boldsymbol{x}_s = (\lambda_s(x), \rho_s(x))$

Generate initial random population $(\lambda_i(x), \rho_i(x))$, $i = 1, \ldots, N_P$ based on the fractional phantom distribution for given $d_{v_{\max}}$, $d_{c_{\max}}$, $Numd_v$, $Numd_c$ [36].
Calculate the input argument of (3.10) for each individual as its cost function for given R_0 and ϵ_0. ▷ $\epsilon^*(\lambda_i(x), \rho_i(x))$ is calculated by the bisection method.
Save the best solution $(\lambda_s(x), \rho_s(x))$ possessing the minimum cost function.
for $i = 1$ to $MaxIt$ **do**
    for $j = 1$ to N_P **do**
        Generate random mutation coefficients $\beta_j \in [0, 1]$.
        sFlag ← rand ▷ A binary flag for strategy selection
        Create $\boldsymbol{v}_j = \boldsymbol{x}_{r_1} + \beta_j((\boldsymbol{x}_{r_2} sFlag + (1 - sFlag)\boldsymbol{x}_{\text{best}}) - \boldsymbol{x}_{r_3})$,
            where the different indices $r_1 \neq r_2 \neq r_3$ are selected uniformly
            at random in $\{1, ..., N_P\}\backslash\{j\}$. ▷ $\mathbf{x_i}$ is the vector
            representation of $(\lambda_i(x), \rho_i(x))$.
        $(\lambda_t(x), \rho_t(x)))$←Crossover$((\lambda_j(x), \rho_j(x)), \boldsymbol{v}_j, C_r)$
        if $cost_t < cost_j$ **then**
            Update j_{th} individual, $(\lambda_j(x), \rho_j(x)) \leftarrow (\lambda_t(x), \rho_t(x))$.
            if $cost_j < cost_s$ **then**
                Update s_{th} individual, $(\lambda_s(x), \rho_s(x)) \leftarrow (\lambda_j(x), \rho_j(x))$.
            end if
        end if ▷ Cost is computed based on (3.10).
    end for
end for

Figure 3.2. DE-based algorithm for minimum average repair bandwidth calculation.

Table 3.2 provides optimal degree distributions for code rate $R = 2/3$. In Table 3.2, the first row corresponds to the random repair protocol, derived by a variant of Algorithm 3, while the second row is associated with the ideal repair bandwidth, obtained using Algorithm 3 itself.

3.4. Numerical Results: LDPC Code Design for Different Repair Protocols

In this section, a detailed numerical analysis is undertaken to comprehend the interplay between repair protocols in LDPC codes. Through a series of figures and tables, the comparative performance of two different repair protocols is demonstrated. Here, the tradeoff curve between the minimum average repair bandwidth $\bar{\gamma}_{\min}$ (or the code rate R) and the decoding threshold ϵ^* is numerically represented by resolving an associated optimization problem.

Figure 3.3 provides a graphical representation of this relationship, where different code rates, specifically $R = \frac{2}{3}$ and $R = \frac{3}{4}$, are considered within the framework of a family of LDPC codes employing two distinct protocols. Notably, the graph demonstrates a substantial reduction in the minimum average repair bandwidth $\bar{\gamma}_{\min}$ with the application of the random access repair protocol, compared to the results found in [15]. This evidence of the effectiveness of the suggested check node degree relaxation in the optimization of code designs is clearly illustrated. The graph also shows the superior performance of ideal repair protocol over random repair protocol. It is further observed that higher code rates, which are more efficient in terms of raw data storage, necessitate larger repair bandwidths. The lower bounds for $\bar{\gamma}_{\min}^{(\text{rand})}$ and $\bar{\gamma}_{\min}^{(\text{ideal})}$ are additionally included in Figure 3.3.

Figure 3.4 illustrates how the decoding threshold changes as a function of the code rate for two fixed bandwidth values 8 and 9, where the maximum check node degree is assumed to satisfy $d_{c_{\max}} \leq 15$ as larger values did not show any significant improvements. As can be seen, as the code rate increases, the gap between the two protocols increases. For instance, with a code rate $R = 3/4$, using ideal repair protocol

improves ϵ^* more than 0.02. Ideal repair protocol's performance shows significant improvements at high code rates, which are more suitable for data storage systems. In the same figure, the rate of the capacity-achieving code is also plotted for the BEC. As can be seen, constraining the code construction to have low repair bandwidth leads them to lose their optimal data recoverability features, more so for higher code rates. Such an observation is quite interesting for data storage in terms of what is achievable and what is not.

The finite length performance of the codes is also evaluated using Monte Carlo simulations and presented in Figure 3.5. Simulations leveraged the node degree distributions given in Table 3.2 and in [15] (Section 4, Table I) for regular check node degrees 8 and 9. In the simulation, the block length of the constructed codes is chosen to be 2000. The block error rate is evaluated under a belief propagation decoder [16]. $\hat{\gamma}^{(\text{rand})}$ and $\hat{\gamma}^{(\text{ideal})}$ are the average repair bandwidths calculated by the achieved node degree distributions of constructed parity check matrix $\mathbf{H}$ via modified PEG algorithm [46]. Figure 3.5 illustrates how a decline in repair bandwidth negatively impacts the code's error recovery properties for random repair protocol. Furthermore, thanks to check node irregularity, for nearly the same (or even less) bandwidth, the code designed for ideal protocol demonstrates better recovery performance than that of the code designed for random access repair protocol.

The data presented in Table 3.3 also offers an insightful view into the optimized coefficients of node degree distributions for both variable and check nodes. It becomes evident that a nearly fixed code rate-R necessitates an increase in repair bandwidth when the decoding threshold expands. This implies a fundamental tradeoff between the parameters α and $\bar{\gamma}$. Analogously, Table 3.4 exhibits a similar trend. It elaborates on the outcomes of an optimization problem when a random repair protocol is taken into account.

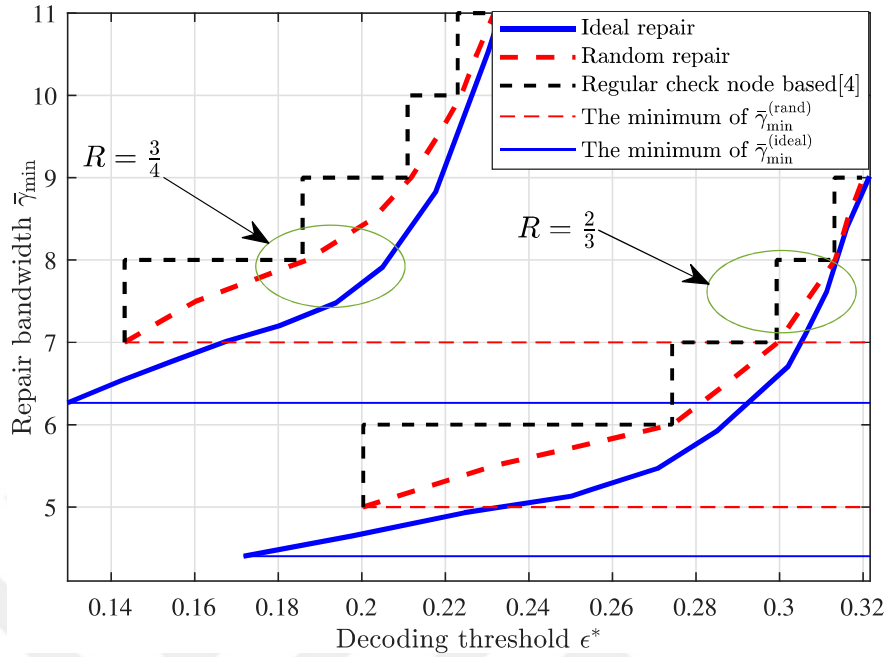

Figure 3.3. Comparing $\bar{\gamma}_{\min}$ to ϵ^* for code rates, $R = \frac{2}{3}$ and $R = \frac{3}{4}$. Dashed black lines represent a regular check node case. [45] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE).

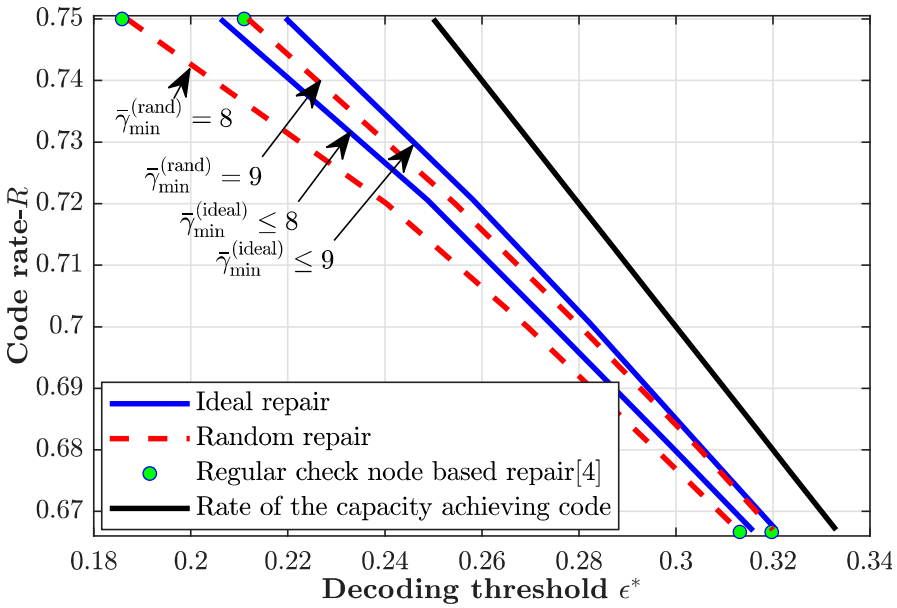

Figure 3.4. Comparing code rate-R to decoding threshold ϵ^* for two repair protocols under a repair bandwidth constraint. The bold black line indicates solutions without a specific protocol. [45] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE).

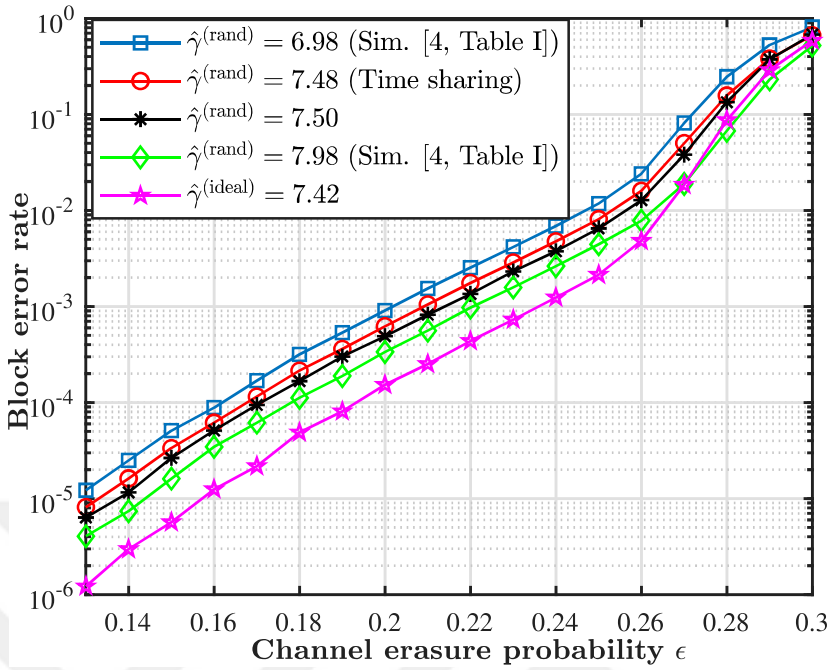

Figure 3.5. Comparing block error rate to channel erasure probability for $R = 2/3$. The curve marked with circles characterizes the time shared performance of the codes with $\hat{\gamma}^{(\text{rand})} = 6.98$ and $\hat{\gamma}^{(\text{rand})} = 7.98$ [45] (This graphic was reused as IEEE's policy on authors reusing their own work. © 2023 IEEE).

.

Table 3.3. Ideal repair protocol ($d_c \leq 13$).

R	ε^*	$\overline{\gamma}$	Node degree distributions
0.6667	0.1693	4.3857	$\lambda(x) = x$ $\rho(x) = 0.1522x^3 + 0.3004x^4 + 0.1574x^5 + 0.0858x^6 + 0.0850x^7 + 0.0941x^8 + 0.0164x^9 + 0.1088x^{14}$
0.6673	0.1898	4.4705	$\lambda(x) = 0.9613x + 0.0387x^2$ $\rho(x) = 0.1269x^3 + 0.3139x^4 + 0.0965x^5 + 0.1563x^6 + 0.0800x^7 + 0.0802x^8 + 0.0485x^9 + 0.0026x^{11} + 0.0951x^{12}$
0.6670	0.2117	4.5926	$\lambda(x) = 0.9027x + 0.0973x^2$ $\rho(x) = 0.0677x^3 + 0.3305x^4 + 0.1722x^5 + 0.0561x^6 + 0.1482x^7 + 0.1282x^8 + 0.0455x^9 + 0.0091x^{11} + 0.0426x^{12}$
0.6673	0.2307	4.7644	$\lambda(x) = 0.8404x \quad 0.1596x^2$ $\rho(x) = 0.2825x^4 + 0.3076x^5 + 0.1675x^6 + 0.0681x^7 + 0.1571x^9 + 0.0172x^{10}$
0.6667	0.2515	4.9963	$\lambda(x) = 0.7058x + 0.2942x^2$ $\rho(x) = 0.2304x^4 + 0.2552x^5 + 0.1089x^6 + 0.1651x^7 + 0.1738x^8 + 0.0178x^9 + 0.0467x^{10} + 0.0022x^{11}$
0.6674	0.2711	5.4387	$\lambda(x) = 0.4879x + 0.5121x^2$ $\rho(x) = 0.0740x^4 + 0.2997x^5 + 0.1643x^6 + 0.1465x^7 + 0.1319x^8 + 0.0325x^9 + 0.1185x^{10} + 0.0325x^{11}$
0.6668	0.2816	5.7465	$\lambda(x) = 0.4629x + 0.3246x^2 + 0.2125x^3$ $\rho(x) = 0.0621x^4 + 0.1570x^5 + 0.2385x^6 + 0.2204x^7 + 0.1356x^8 + 0.0423x^9 + 0.0457x^{11} + 0.0985x^{12}$
0.6670	0.3032	6.7924	$\rho(x) = 0.2822x^6 + 0.1604x^7 + 0.1912x^8 + 0.1455x^9 + 0.1375x^{11} + 0.0833x^{12}$ $\lambda(x) = 0.4099x + 0.0982x^2 + 0.2730x^3 + 0.0892x^4 + 0.0012x^5 + 0.1286x^6$
0.6671	0.3223	9.1272	$\rho(x) = 0.1485x^8 + 0.1054x^9 + 0.4178x^{10} + 0.1931x^{11} + 0.0149x^{13} + 0.1203x^{14}$ $\lambda(x) = 0.3039x + 0.1201x^2 + 0.1353x^3 + 0.0372x^4 + 0.0464x^5 + 0.0036x^6 + 0.3535x^{11}$

Table 3.4. Random repair protocol.

R	ε^*	$\overline{\gamma}$	Node degree distributions
0.7500	0.1432	7	$\lambda(x) = x$ $\rho(x) = x^7$
0.7500	0.1695	7.7000	$\lambda(x) = 0.7667x + 0.2333x^2$ $\rho(x) = 0.3000x^7 + 0.7000x^8$
0.7500	0.2027	8.5000	$\lambda(x) = 0.5334x + 0.4666x^2$ $\rho(x) = 0.5000x^8 + 0.5000x^9$
0.7500	0.2154	9.3000	$\lambda(x) = 0.4685x + 0.2639x^2 + 0.2677x^3.$ $\rho(x) = 0.7000x^9 + 0.3000x^{10}.$
0.7501	0.2214	9.8000	$\lambda(x) = 0.4598x + 0.1356x^2 + 0.3114x^3 +$ $0.0757x^4 + 0.0175x^5.$ $\rho(x) = 0.2000x^9 + 0.8000x^{10}.$
0.7501	0.2260	10.3000	$\lambda(x) = 0.4302x + 0.1163x^2 + 0.2118x^3 + 0.2321x^4 +$ $0.0003x^5 + 0.0092x^6.$ $\rho(x) = 0.7000x^{10} + 0.3000x^{11}.$
0.7500	0.2316	11	$\lambda(x) = 0.3911x + 0.1747x^2 + 0.0696x^3 + 0.1084x^4 +$ $0.1639x^5 + 0.0923x^6.$ $\rho(x) = x^{11}.$
0.7502	0.2364	12	$\lambda(x) = 0.3446x + 0.2139x^2 + 0.0113x^4 + 0.1933x^5 +$ $0.1261x^6 + 0.0712x^8 + 0.0396x^9.$ $\rho(x) = x^{12}.$
0.7503	0.2383	12.5000	$\lambda(x) = 0.3220x + 0.2242x^2 + 0.0317x^3 + 0.0597x^5 +$ $0.0671x^6 + 0.0960x^7 + 0.1771x^8 + 0.0222x^9.$ $\rho(x) = 0.5000x^{12} + 0.5000x^{13}.$

4. AN LDPC CODE CONSTRUCTION FRAMEWORK FOR DSS

4.1. Introduction

Distributed storage systems embody a distinctive paradigm for data transmission that operates on the binary erasure channel. Leveraging the use of LDPC codes for data file encoding, these systems facilitate the potential for both partial and complete recovery of encoded data via a peeling decoder. The peeling decoder, while characterized by its simplicity, provides considerable efficiency and features a range of impactful functionalities. Its computational structure is minimalistic, promoting ease of implementation and operation [16]. Furthermore, it exhibits the capability to regenerate any data segment independent of its size. Such flexibility in data recovery constitutes a vital benefit within DSSs. It empowers the storage system to restore a faulty subset of symbols or, when required, retrieve the total data volume. Such a feature strengthens the resilience and robustness of the data storage system.

Recently, a range of studies have focused on enhancing the functionality of LDPC codes, while preserving high levels of data reliability. There has been an exploration of extensions to basic LDPC codes, such as repeat-accumulate codes [12], protograph-based codes [13], spatially-coupled codes [14], and multi-edge-type (MET) codes [14]. As a variant of MET codes, bi-layer (multi-layer) LDPC codes are first introduced in [47] for relay channels. In bilayer LDPC codes, there are two types of check nodes, and thus, each type has a different connectivity to the variable nodes [48]. By making use of multi-layer codes, it becomes possible to decode the same variable nodes with successively more powerful decoding tools. Considering this, applications of distributed storage are motivated to design *multi-sub-block coding* using multi-layer LDPC codes [48, 49]. Multi-sub-block coding schemes have been developed to provide robust read access to small blocks with minimal data protection [49]. In a multi-sub-block coding scheme, a code block of length N is divided into M sub-blocks of length n [49]. These

codes are typically based on Reed-Solomon and similar algebraic codes [3, 50, 51], but their use can be computationally expensive for large data sizes. As an alternative, multi-layer LDPC codes such as LDPCL (suffix L represents locality), which allows for local access, have been suggested [49] . This LDPCL coding paradigm works in such a way that if any adjacent symbols of the same encoded fragment are erased, the system initially triggers local decoding in order to recover the missing symbols. In LDPCL coding paradigm, a data unit of Mn bits across M blocks is scattered, with each block required to access its local sub-block of n bits. The associated graph is structured so that each local check node is only connected to variable nodes within the same sub-block. The edges in the graph can be divided into two types: *local edges*, which connect variable nodes to local check nodes, and *joint edges*, which connect joint check nodes to variable nodes, see Figure 4.1.

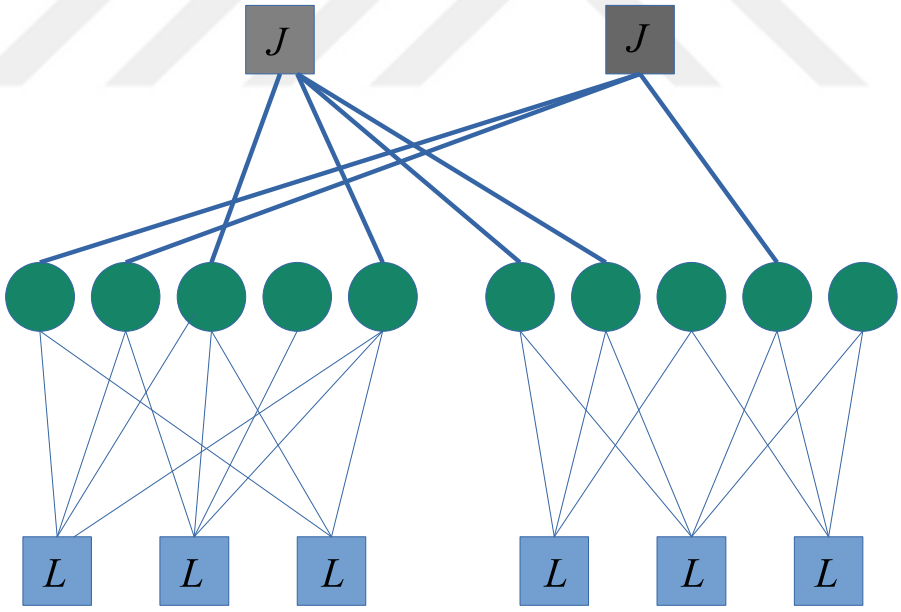

Figure 4.1. A sub-blocked Tanner graph with $M = 2$ and $n = 5$. 'L' and 'J' labels indicate local and joint check nodes, respectively.

Multi-edge type LDPC codes can ensure both complete data reliability and localized recovery of specific subsets of encoded data. [48] introduces a theoretical framework for these types of codes, specifically targeting approaching-capacity code rates for large code lengths. However, the design and construction of these codes can be complex due

to their hierarchical structure. Additionally, compared to standard LDPC codes, they require more redundant data and were initially developed to increase data reliability by handling both local and global error correction simultaneously.

It is noteworthy that within the realm of DSSs, storage nodes are susceptible to failure over time, necessitating their eventual replacement with new nodes. Each of these storage nodes carries a fragment of encoded data. Therefore, what's crucial is not gaining local symbol access in a storage node but also the ability to locally recover the content of the failed node(s) from a subset of remaining storage nodes. This requirement can be met using standard LDPC codes with less storage overhead. With careful construction of the parity check matrix $\mathbf{H}$, it is possible to achieve the desired level of reliability and the ability to repair local storage node(s). This ensures a resilient distributed storage system that can maintain data integrity, even when some individual nodes experience failures.

Finally, in this chapter, a comprehensive illustration of the process involved in the construction of the parity check matrix $\mathbf{H}$, specifically designed for a DSS, is provided. Each phase inherent in this construction process is also meticulously elaborated upon, with the inclusion of illustrative examples.

4.2. LDPC Regenerating Codes for Repairing Data Fragments in DSSs

In a DSS, a regenerating code is typically represented by a tuple $(n, k, \alpha, \gamma, \mathcal{F})$ [1]. Here, n denotes the total number of storage nodes, while k specifies the minimum number of nodes that must be connected to recover the complete data file. The term α corresponds to the size of the storage, whereas γ stands for the repair bandwidth required for the regenerating of a failed node. When LDPC codes serve as $(n, k, \alpha, \gamma, \mathcal{F})$ regenerating codes, henceforth referred to as LDPC regenerating codes, they must adhere to specific constraints during both their asymptotic design and finite length construction. In order to guarantee the (n, k) recovery characteristic, the decoding threshold must meet the condition $\epsilon^* \geq 1 - \frac{k}{n}$. Furthermore, to facilitate the construc-

tion of these codes, the storage size α is necessitated to be greater than the ratio $\frac{\mathcal{F}}{k}$. Note, however, that if $\alpha = \frac{\mathcal{F}}{k}$, then the rate R, defined as the ratio of $\mathcal{F}$ to $n\alpha$, is equal to k/n. This implies that ϵ^* is greater or equal to $1 - R$, a condition that does not hold true for the design of asymptotic LDPC codes.

For LDPC regenerating codes denoted by the tuple $(n, k, \alpha, \gamma, \mathcal{F})$, it is important to note that the parameter γ, which represents the repair bandwidth, is estimated in an average manner. This estimation is conducted using a greedy algorithm that will be explained later. Therefore, initially, a family of LDPC codes is designed with a fixed code rate, denoted by $R = \frac{\mathcal{F}}{n\alpha}$, and the constraint $\epsilon^* \geq 1 - \frac{k}{n}$. The code design optimization problem is specifically tailored for each repair protocol discussed in the previous chapter, taking into account the associated constraints. The main goal of this optimization process is to reduce the repair bandwidth. The design principles for such codes have been elaborated in detail in the previous chapter, hence this chapter predominantly focuses on the construction of the code.

4.2.1. Symbols Allocation for Storage Nodes

This thesis initially constructs a parity check matrix $\mathbf{H}$, characterized by the dimensions $n\alpha \times m$, for a specified $(\lambda(x), \rho(x))$ employing a modified PEG algorithm [46]. Following this, within the DSS framework, this matrix $\mathbf{H}$ undergoes modification to become a new matrix, denoted as $\mathbf{H}^{(\text{dev})}$. This new matrix can be conceptualized as a concatenation of distinct sub-matrices, with each sub-matrix corresponding to a separate storage unit s_i. To express this more explicitly, the matrix $\mathbf{H}^{(\text{dev})}$ is comprised of n sub-matrices denoted by $s_i.\mathbf{H}$, such that the overall matrix structure is represented as $\mathbf{H}^{(\text{dev})} = [s_1.\mathbf{H}, \ldots, s_n.\mathbf{H}]$. In this configuration, each sub-matrix $s_i.\mathbf{H}$ is representative of the portion of the parity check matrix $\mathbf{H}^{(\text{dev})}$ that is relevant to the corresponding storage unit s_i. To determine the $\mathbf{H}^{(\text{dev})}$ matrix, several steps are required. Initially, without any change in dimensions, the $\mathbf{H}$ matrix is mapped to the form $[s_1.\mathbf{H}, \ldots, s_n.\mathbf{H}]$, targeting a semi-balanced distribution of variable nodes' degrees on each storage unit.

The rationale behind this semi-balanced distribution is as follows: The pair $(\lambda(x), \rho(x))$ has been specifically optimized to minimize symbol repair, indicating an improved performance for each storage unit. In addition, the emergence of stopping sets is mostly due to low-degree variable nodes. By organizing the data fragment distribution in such a way that each storage unit corresponds to a sub-matrix with a nearly identical variable node distribution, the probability of the co-occurrence of low-degree variable nodes is reduced. In light of this, the subsequent example provides a demonstration of how this arrangement of the columns is carried out within the parity check matrix $\mathbf{H}$ to derive $\mathbf{H}^{(\text{dev})}$.

Example: Consider a scenario where each variable node (symbol) is represented by $v_i^{(q)}$. In this notation, i points to the respective column index in the $\mathbf{H}$ matrix, while q denotes the degree of that variable node.

Case 1: Figure 4.2 illustrates the division of variable nodes (symbols) into two groups based on their degree. The nodes are then allocated as evenly as possible across the storage units. Given that there are three storage units ($n = 3$) and two groups of nodes, each comprising three nodes, each storage unit will receive a similar portion of nodes, resulting in a balanced distribution of variable node degrees across all storage units.

Case 2: Referring to Figure 4.3, the nodes are divided into three groups based on their degree: three nodes of degree 2 and three nodes of degree 3, and two nodes of degree 4. While the aim is to evenly distribute the variable nodes across the storage units, in this instance, it is not completely balanced. This arrangement can be regarded as a semi-balanced distribution. Figure 4.4 also depicts a situation akin to the one illustrated in Figure 4.3

It should be emphasized that, in these scenarios, the parity check matrix is formed as $\mathbf{H}^{(\text{dev})} = [s_1.\mathbf{H}, s_2.\mathbf{H}, s_3.\mathbf{H}]$. The columns of each sub-matrix, $s_i.\mathbf{H}$, is determined by the variable nodes that are assigned to their corresponding storage unit s_i.

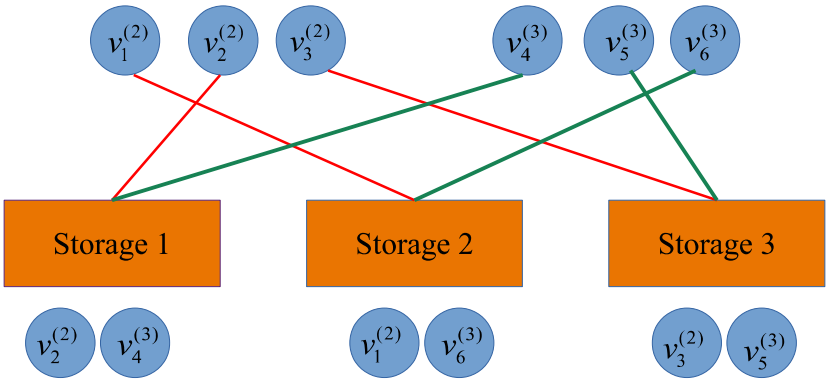

Figure 4.2. Symbols allocation in storage units, toy example 1.

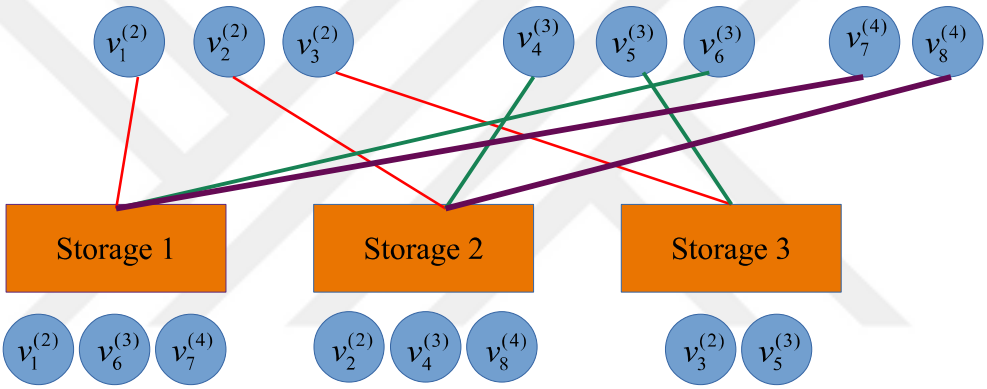

Figure 4.3. Symbols allocation in storage units, toy example 2.

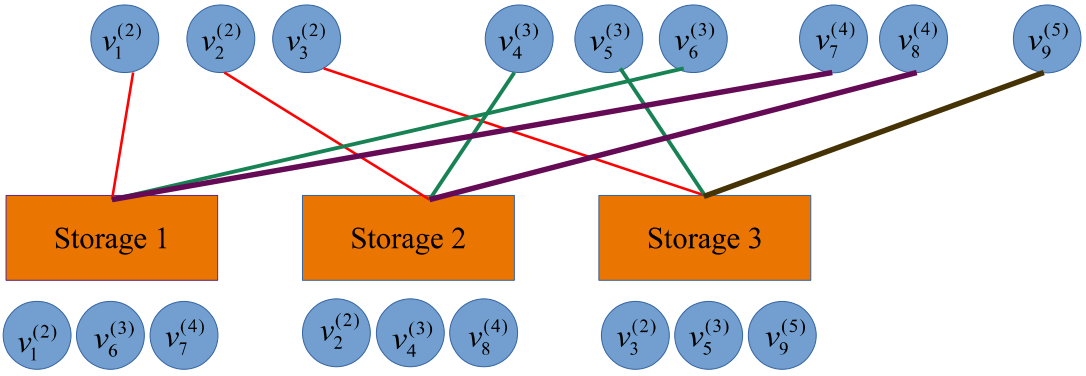

Figure 4.4. Symbols allocation in storage units, toy example 3.

This thesis also follows similar technique to transforms the matrix $\mathbf{H}$ into $\mathbf{H}^{(\mathrm{dev})}$. This transformation aims to lessen the detrimental effects of stopping sets within the distributed storage system environment. However, it is important to note that this technique does not guarantee the complete elimination of stopping sets. Thus, there is

a requirement to identify and remove stopping sets through a distinct process. However, this is not an easy task. In some cases, when the stopping sets are relatively small, a method suggested in [52] can be used to eliminate stopping sets efficiently with minimal impact on the code rate. This thesis adopts the same technique for the purpose of removing stopping sets.

4.2.2. Stopping Sets Elimination

In a DSS with LDPC coding, a peeling decoder is employed to recover data by connecting to k out of n storage units. When employing a peeling decoder that operates on a parity check matrix $\mathbf{H}$, the decoder will fail to interpret the remaining symbols if no columns are left in the residual matrix $\mathbf{H}^{(\mathrm{RM})}$ that have at least one element corresponding to a row with a unit row weight. In such instances, the columns with non-zero weights serve to identify the stopping set. The residual matrix $\mathbf{H}^{(\mathrm{RM})}$ is primarily formed by concatenating the parity check matrices of the erased storage units, i.e., $[s_{i_{k+1}}.\mathbf{H}, \ldots, s_{i_n}.\mathbf{H}]$, where $\{i_{k+1}, \ldots, i_n\}$ are a set of indices of storage units that are regarded as erasure when recovering the complete data file (k out of n storage units). The peeling decoder aims to transform this residual matrix into a zero matrix, effectively decoding the remaining symbols to recover the entire data.

Considering a binary erasure transmission channel, finding stopping sets for a $\mathbf{H}$ matrix generally imposes a significant challenge [53, 54]. However, the complexity of the task is somewhat mitigated within the context of DSS due to the storage units erasure instead of symbols erasure, where the objective is to ensure the recovery of the complete data from k out of n storage units. Therefore, it is only needed to independently examine all $n - k$ out of n combinations of storage units to locate stopping sets. If the number of stopping sets is relatively small, the method proposed in [52] is efficient. In what follows, a simplified example is provided to illustrate how stopping sets are detected and eliminated in this thesis.

Example: In a setting involving $n = 4$ storage units and conditioned to the data file recovery, $k = 2$ out n , it is assumed that storage unit 1 holds symbols $v_1, \ldots, v_5$, storage unit 2 contains symbols $v_6, \ldots, v_{10}$, storage unit 3 stores symbols $v_{11}, \ldots, v_{15}$, and storage unit 4 maintains symbols $v_{16}, \ldots, v_{20}$. Under this specific arrangement, assume that the stopping sets are as follows:

$$\begin{aligned} S_1 &= \{\}, \\ S_2 &= \{v_1, v_3, v_{11}, v_{14}\}, \\ S_3 &= \{v_3, v_5, v_{16}, v_{19}\}, \\ S_4 &= \{v_6, v_7, v_{11}, v_{12}\}, \\ S_5 &= \{v_6, v_8\}, \\ S_6 &= \{v_{11}, v_{12}, v_{17}, v_{20}\}, \end{aligned} \tag{4.1}$$

where each S_i , $i \in \{1, \ldots, 6\}$, stems from a particular storage units pair.

Figure 4.5 shows the outline of identifying stopping sets. As can be seen, the initial step (Step 1) involves symbols allocation to the storage units. In the subsequent step (Step 2), storage units' erasure patterns for each pair of storage nodes are determined.

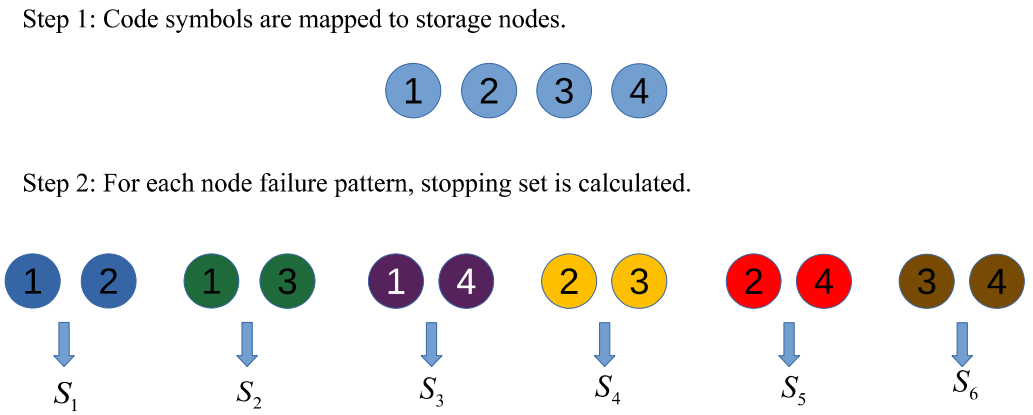

Figure 4.5. Finding erasure patterns for $n = 4$ and $k = 2$. Circles indicate the storage nodes.

Therefore, it is necessary to systematically identify which combinations of storage units from all storage units could cause decoding failure (recognized as an erasure pattern) which is exemplified in the following.

Example: Suppose five storage units, labeled from 1 to 5, hold different pieces of encoded data. The goal is to determine which possible pairings of storage units, if lost, would make it impossible to recover the original data.

Initially, all possible pair combinations that could be lost from the five available storage units are generated. In this scenario, possible outcomes include $(1, 2), (1, 3)$, and so on up to $(4, 5)$. Subsequently, for every pair, a residual matrix is created using the data from the lost units. As an example, if the pair being considered is (1,2), the data from units 1 and 2 is employed to generate the residual matrix.

After identifying all erasure patterns, it becomes crucial to eliminate all stopping sets. For this purpose, the methodology suggested in [52] is employed in this thesis. A comprehensive explanation of this process is further elaborated on in the following.

Given a set of n small stopping sets $S_1, S_2, ..., S_n$, and the introduction of a new check node c_a to the Tanner graph $\mathcal{G}$ representation of the code, it can be established that the elimination of a stopping set S can only be achieved through the inclusion of an edge connecting a variable node within S to c_a. It is imperative to consider the potential intersection of the stopping sets $S_1, S_2, ..., S_n$, and thus, the following two cases shall be analyzed [52]:

1. There is no overlap between stopping sets. Therefore, a variable node should be chosen from each stopping set and linked to check node c_a.
2. There exists an intersection or overlap between two or more stopping sets. A variable node that appears most frequently in the intersected stopping sets is connected to c_a.

S_E	S	$V \setminus V_E \rightarrow v_t$
$\{\}$	$\{S_2, S_3, S_4, S_5, S_6\}$	
$\{S_2, S_4, S_6\}$	$\{S_3, S_5\}$	v_{11}
$\{S_2, S_3, S_4, S_6\}$	$\{S_5\}$	v_5
$\{S_2, S_3, S_4, S_5, S_6\}$	$\{\}$	v_8

Figure 4.6. Parity check row addition scheme. S indicates the set of stopping sets, and S_E is the eliminated stopping sets after adding a new check node.

4.2.3. Stopping Sets Elimination in LDPC-Based DSSs

The flowchart, shown in Figure 4.7, outlines the series of steps involved in enhancing the reliability of data storage systems using LDPC regenerating codes. Starting from a given LDPC code represented by a pair of degree distribution polynomials, $(\lambda(x), \rho(x))$, for $(n, k, \mathcal{F})$, an initial parity check matrix, $\mathbf{H}$, is constructed. Then, a mapping operation is carried out to transform the initial $\mathbf{H}$ matrix into $\mathbf{H}^{(\text{dev})}$. Following this, a procedure for identifying patterns of erasures, specifically for $n-k$ out of n nodes, is executed. These patterns are further analyzed to detect problematic stopping sets, which are then strategically eliminated by introducing new rows to $\mathbf{H}^{(\text{dev})}$. The flowchart concludes with the completion of the stopping set elimination process, which leaves the system robust against the identified erasure patterns.

Until now, the primary focus has been on constructing LDPC regenerating codes that are tailored to ensure data recovery. Intriguingly, when the data recovery condition is fulfilled, it is possible to single-node repair without introducing additional problems provided the inequality, (n choose $(n-k)$) $\geq n$, is satisfied. Here, (n choose $(n-k)$) symbolizes the number of ways to select $(n-k)$ elements from a set of n elements.

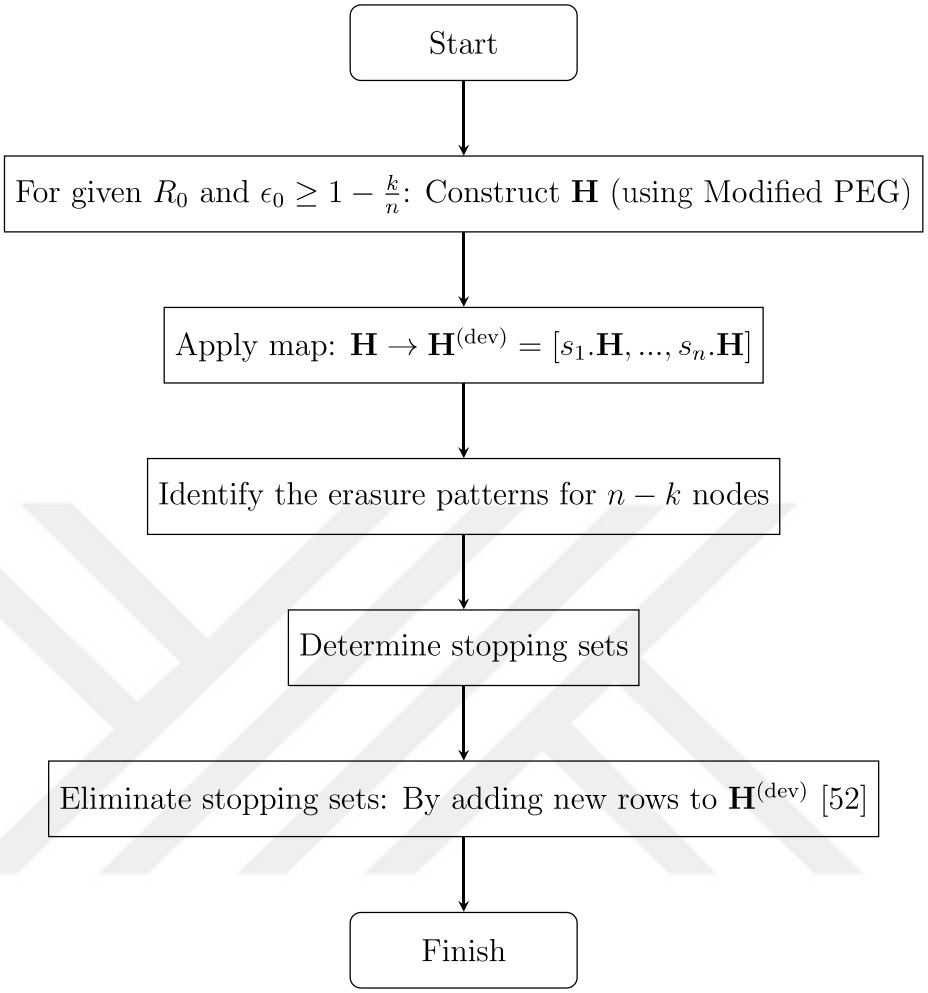

Figure 4.7. Flowchart demonstrating the process of constructing and modifying the matrix $\mathbf{H}$ for a given R_0 and ϵ_0.

The underlying rationale is as follows: when all stopping sets related to any combination of $n - k$ out of n storage units have been successfully eliminated, no stopping sets remain within each individual storage unit. Consequently, it can be inferred that the contents of each failed node can also be retrieved using the peeling decoder.

4.3. Average Single Node Repair Bandwidth for a DSS: Greedy Approach

This section determines the bandwidth required to repair a failed storage unit, employing a peeling decoder. Besides, a modified version of the peeling decoder is

utilized to regenerate the failed storage unit, with the primary objective of reducing repair bandwidth. The conventional implementation of the peeling decoder follows a specific protocol. This protocol involves selecting a check node from the residual matrix. The selected check node needs to have a unitary row weight and be associated with the erasure node(s) to enable the regeneration of symbols. In scenarios where multiple such check nodes exist for a single erasure symbol, the decoder randomly selects one. This procedure is iterative and continues until either all recoverable symbols are retrieved or it reaches a point where no further progress can be made.

This thesis suggests an enhancement to the traditional peeling decoder strategy. Instead of randomly selecting from multiple check nodes with unit row weight in the residual matrix, a *greedy* approach is proposed, prioritizing the check node that results in the minimum repair bandwidth at the current step. This modification could potentially improve the overall efficiency of the recovery process by reducing the bandwidth required for node(s) repair operations.

Algorithm 4 provides a greedy approach to compute the single node average repair bandwidth for a DSS. The algorithm receives n storage units. Each storage unit, s_i, has a code length c_i, computed as the number of columns in the unit's $\mathbf{H}$ matrix, $s_i.\mathbf{H}$. For each storage unit, the code length is computed, the repair cache is initialized, and a remaining matrix $\mathbf{H}_i^{(\text{remaining})}$, where i indicates the current repairing storage unit, is constructed by replacing the storage unit's $\mathbf{H}$ matrix with a zero matrix in the overall $\mathbf{H}$ matrix. An iteration begins over each storage unit, during which a repair cache R_i is initialized as an empty set. The algorithm then creates a remaining matrix, $\mathbf{H}_i^{(\text{remaining})}$, by replacing the current storage unit's $\mathbf{H}$ matrix in the complete $\mathbf{H}$ matrix with zeros. This matrix keeps track of the remaining symbols that need to be recovered. An inner loop then begins and continues until the number of recovered symbols is equal to the code length of the storage unit. In each iteration of this loop, the algorithm selects a column vector from the current storage unit's $\mathbf{H}$ matrix. It then identifies rows with non-zero values in this vector and unit row weights in the storage unit's $\mathbf{H}$ matrix, designating this set as $I_j^{(\text{Deg1})}$. If $I_j^{(\text{Deg1})}$ is not empty, the algorithm proceeds to

calculate the row indices with the minimum row weight in $\mathbf{H}_i^{(\text{remaining})}$ matrix. For each of these indices, the algorithm computes the union of the current repair cache with the set of associated symbols with non-zero values of the current row in $\mathbf{H}_i^{(\text{remaining})}$ matrix. Among these, the algorithm selects the one that yields the smallest union and updates the repair cache with this minimal set.

The algorithm then updates the $\mathbf{H}$ matrix of the storage unit to indicate that the current symbol has been successfully recovered and increments the count of recovered symbols. The symbol index is updated in each loop iteration, and if it exceeds the code length of the storage unit, it is reset to 1. After all symbols in a storage unit have been recovered, the elements of the repair cache set are stored in the repair set of the storage unit. Once all storage units have been processed, the algorithm computes the average repair bandwidth.

The greedy characteristic of Algorithm 4 lies in its decision-making process. This algorithm operates by making a sub-optimal immediate decision at every step of the repair process. Specifically, it chooses the check nodes that currently minimize the repair bandwidth rather than randomly selecting them. As such, the algorithm can lead to a reduction in repair bandwidth.

This section has presented a comprehensive exploration of bandwidth computation for repairing a failed storage unit using a peeling decoder. Traditional implementations of this decoder, involving the methodical selection of check nodes from a residual matrix, often result in random selections when multiple check nodes fulfill the required criteria. The primary contribution of this thesis is the introduction of a modified peeling decoder approach that abandons this randomness in favor of a greedy strategy. Rather than leaving the decision to chance, this revised algorithm prioritizes check nodes based on their potential to minimize the repair bandwidth at each step. The meticulous description of Algorithm 4 unveils its structured and greedy decision-making, aiming to reduce overall repair bandwidth, potentially leading to more efficient recovery processes in DSSs.

Algorithm 4 Greedy Algorithm for Single Node Repair Bandwidth Computation

Require: $(s_i)_{1 \le i \le n}$

Ensure: $averageRepairBW$

for each storage unit s_i where $i \in \{1, 2, ..., n\}$ **do**
    Compute code length c_i for each storage unit as the number of columns in $s_i.\mathbf{H}$
    Initialize repair cache R_i as empty set.
    Construct the matrix $\mathbf{H}_i^{(\text{remaining})}$ by replacing $s_i.\mathbf{H}$ in $\mathbf{H}$ with zero matrix.
    Set initial values: symbol index $j = 1$ and recovered symbol count $r = 0$.
    while $r < c_i$ **do**
        From $s_i.\mathbf{H}$, select the jth column vector $s_i.h_j$, i.e., $s_i.h_j = s_i.\mathbf{H}[:, j]$.
        Find rows with non-zero values in $s_i.h_j$ and unit row weights in $s_i.\mathbf{H}$, denoted as set $I_j^{(\text{Deg1})}$.
        if $I_j^{(\text{Deg1})}$ is not empty **then**
            For each index $i \in I_j^{(\text{Deg1})}$, find row indices with minimum row weight in $\mathbf{H}_i^{(\text{remaining})}$, denoted as $J^{(\text{remaining})}$.
            For each $j \in J^{(\text{remaining})}$, determine $R_i^j \leftarrow R_i \cup R_j$. ▷ R_j is the associated symbols set to non-zero values of jth row in $\mathbf{H}_i^{(\text{remaining})}$.
            Select a row that yields the smallest $|R_i^j|$, denoted as $R_{\min}$.
            Update repair cache with smallest cache found, i.e., $R_i = R_{\min}$.
            Update $s_i.\mathbf{H}$ to indicate successful recovery of jth symbol, i.e., $s_i.h_j = \mathbf{0}$.
            Increase recovered symbol count by 1, i.e., $r = r + 1$.
        end if
        Update symbol index $j = j + 1$.
        if j exceeds code length c_i **then**
            $j = 1$.
        end if
    end while
    Store repair cache in s_i.Rep: $s_i.\text{Rep} = R_i$
end for
Calculate average repairs as $averageRepairBW = \sum_{i=1}^{n} |s_i.\text{Rep}|/n$

Figure 4.8. Single node repair bandwidth calculation algorithm.

4.4. Numerical Results

As discussed in the preceding sections, constructing an LDPC code is initiated with a particular optimization problem. The expectation is that this LDPC code will effectively meet the requirements for data recovery. Hence, for a file of size $\mathcal{F}$, and with a fixed $R_0 = \frac{\mathcal{F}}{n\alpha}$, the minimizing repair bandwidth optimization problem is given

$$\min_{(\lambda,\rho)\in\mathcal{S}} \bar{\gamma}^{(p)}(\lambda,\rho), \tag{4.2}$$

subject to the reliability constraint,

$$\varepsilon^* \geq 1 - \frac{k}{n}, \tag{4.3}$$

where $p \in \{\text{ideal}, \text{rand}\}$ indicates the repair protocol.

Table 4.1 and Table 4.3, which are both divided into two parts: Design and Construction, incorporate the parameters $(n = 10, k = 6)$. Table 4.1 corresponds to ideal repair protocol, while Table 4.3 reports the results of random repair protocol. In the Design columns, it becomes evident that as the data rates increase, the repair bandwidth for symbol repair also increases, a pattern that holds true across the introduced repair protocols. This observation was substantiated in the preceding chapter. For a given initial file size $\mathcal{F} = 1000$ bits, in the column labeled with Dim. $\mathbf{H}$, the code length is calculated using the formula $N = \frac{\mathcal{F}}{R_0}$. Nevertheless, there is a slight deviation in the number of rows in $\mathbf{H}$ from its theoretical value.

There are two steps that can lead to this slight change: firstly, the construction of $\mathbf{H}$ using a modifiedPEG method, and secondly, the elimination of the stopping sets algorithm in the construction pipeline, which both can alter the number of rows. Given that the number of rows in the constructed $\mathbf{H}$ exhibits slight variations, causing a small change in the accepting file size associated with this code, parameters related to the file size are presented in a normalized form for consistency and ease of comparison. When considering a scenario where each storage unit contains one symbol, it becomes apparent from Figure 3.5 that achieving full data recovery becomes nearly impossible as the code rate approaches $1-\epsilon^*$. However, in the proposed code construction pipeline

designed for practical DSSs, stopping sets that cause decoder failure, are identified and eliminated with only a minor impact on the code rate. Given that each storage unit possesses a segment of the encoded data, the simultaneous recovery of the content of a storage unit can reduce the symbol repair bandwidth. This is due to the intra-node symbols dependency between the co-existing symbols within the same storage unit. As a consequence it can be seen that $\bar{\gamma}_{\mathbf{H}}^{(\text{symbol})} \leq \bar{\gamma}_{\min}^{(\text{symbol})}$ on both Tables. Tables 4.1 and 4.3 both demonstrate, as anticipated, that increased data rates necessitate the addition of more parity check rows in order to eliminate all the stopping sets.

In the constructed $\mathbf{H}$ matrix, $\alpha = \frac{N}{n}$ and its associated file size $\mathcal{F}$ is $N - \text{Rank}(\mathbf{H})$ where $\text{Rank}(\mathbf{H})$ is approximated by number of rows. Considering this, Tables 4.2 and 4.4 are populated. To facilitate comparison, these tables present the storage unit's normalized storage size and normalized repair bandwidth. As expected, there is a tradeoff between storage size and repair bandwidth. In order to substantiate the outcomes presented in Tables 4.2 and 4.4, it is crucial to begin by clarifying the concept of a stopping set. Specifically, a stopping set pertains to the unprocessed variable nodes that are linked to the residual parity check matrix $\mathbf{H}^{(\text{RM})}$, which corresponds to multiple storage units. The level of sparsity exhibited by this matrix directly influences the probability of achieving successful decoding. Notably, as the residual matrix becomes sparser, there is a reduction in the interdependence among the variable nodes it encompasses. Interestingly, this sparsity is correlated with an increase in repair bandwidth, particularly in instances where the $\mathbf{H}$ matrix incorporates higher row weights.

The objective of the suggested code construction framework is to limit the addition of check nodes to the $\mathbf{H}$ matrix, thus minimally affecting the code rate. Consequently, it is expected that the potential repair bandwidth of a storage unit, when it has higher row weights, would be greater than in a situation where the row weights are concentrated on lower values. This explains the slight enhancement in the repair bandwidth computation observed with the random protocol in comparison to the ideal protocol.

Table 4.1. LDPC code construction (Ideal protocol): $n = 10$, $k = 6$.

Design					Construction			
$(\lambda(x), \rho(x))$	$\epsilon^* \simeq$	R_0	$\bar{\gamma}_{\min}^{(\text{symbol})}$	Protocol	Dim. $\mathbf{H}$	R'	$\bar{\gamma}_{\mathbf{H}}^{(\text{symbol})}$	Num. of new rows
$\lambda(x) = 0.5625x + 0.4338x^2$ $+0.0037x^3$, $\rho(x) = 0.4654x^3+0.1294x^4$ $+0.2613x^5+0.0590x^6$ $+0.0849x^7$.	0.4	0.52	3.391	Ideal	$(927, 1924)$	0.5182	2.867	4
$\lambda(x) = 0.4679x + 0.4330x^2$ $+0.0991x^3$, $\rho(x) = 0.1752x^3+0.3822x^4$ $+0.0851x^5+0.3513x^6+0.0062x^7$.	0.4	0.54	3.863	Ideal	$(859, 1852)$	0.5362	3.189	8
$\lambda(x) = 0.4573x + 0.2961x^2$ $+0.2399x^3+0.0068x^4$, $\rho(x) = 0.0257x^3+0.3834x^4$ $+0.3776x^5+0.0980x^6$ $+0.1153x^7$.	0.4	0.55	4.2502	Ideal	$(834, 1819)$	0.5415	3.365	15
$\lambda(x) = 0.3706x + 0.1727x^2$ $+0.1628x^3+0.1842x^4$ $+0.1097x^5$, $\rho(x) = 0.1886x^4+0.4365x^5$ $+0.0932x^6+0.0426x^7$ $+0.0368x^8+0.2023x^{12}$.	0.4	0.56	4.8242	Ideal	$(812, 1786)$	0.5454	3.689	27
$\lambda(x) = 0.4083x + 0.1200x^2$ $+0.1265x^3+0.2811x^4$ $+0.0038x^{10}+0.0603x^{11}$, $\rho(x) = 0.0541x^4+0.3099x^5+$ $0.3226x^6+0.1302x^7+$ $0.1729x^8+0.0018x^9+$ $0.0084x^{10}$.	0.4	0.57	5.2451	Ideal	$(786, 1755)$	0.5521	3.812	31

Table 4.2. Normalized $\bar{\alpha}$ and $\bar{\gamma}^{(\text{storage})}$(Ideal protocol).

$\frac{\bar{\alpha}}{\mathcal{F}}$	$\frac{\bar{\gamma}^{(\text{storage})}}{\mathcal{F}}$
0.193	0.553
0.186	0.595
0.185	0.621
0.183	0.676
0.181	0.690

Table 4.3. LDPC code construction (Random protocol): $n = 10$, $k = 6$.

Design					Construction			
$(\lambda(x), \rho(x))$	$\epsilon^* \simeq$	R_0	$\bar{\gamma}_{\min}^{(\text{symbol})}$	Protocol	Dim. $\mathbf{H}$	R'	$\bar{\gamma}_{\mathbf{H}}^{(\text{symbol})}$	Num. of new rows
$\lambda(x) = 0.6500x + 0.3500x^2$, $\rho(x) = 0.2400x^3 + 0.7600x^4$.	0.4	0.52	3.76	Rand	$(926, 1924)$	0.5187	2.862	3
$\lambda(x) = 0.5303x + 0.4697x^2$, $\rho(x) = 0.8197x^4 + 0.1803x^5$.	0.4	0.54	4.1803	Rand	$(858, 1852)$	0.5367	3.126	7
$\lambda(x) = 0.5057x + 0.3020x^2 + 0.1923x^3$, $\rho(x) = 0.4213x^4 + 0.5787x^5$.	0.4	0.55	4.5787	Rand	$(832, 1819)$	0.5426	3.320	14
$\lambda(x) = 0.4919x + 0.1770x^2 + 0.2579x^3 + 0.0732x^4$, $\rho(x) = 0.0665x^4 + 0.9335x^5$.	0.4	0.56	4.9335	Rand	$(805, 1785)$	0.5490	3.524	21
$\lambda(x) = 0.4376x + 0.2048x^2 + 0.1174x^3 + 0.0232x^4 + 0.0484x^5 + 0.1686x^6$, $\rho(x) = 0.3758x^5 + 0.6242x^6$.	0.4	0.57	5.6242	Rand	$(783, 1755)$	0.5538	3.755	29

Table 4.4. Normalized $\bar{\alpha}$ and $\bar{\gamma}^{(\text{storage})}$(Random protocol).

$\frac{\bar{\alpha}}{\mathcal{F}}$	$\frac{\bar{\gamma}^{(\text{storage})}}{\mathcal{F}}$
0.193	0.552
0.186	0.583
0.185	0.612
0.182	0.642
0.180	0.678

5. CONCLUSION

This thesis has offered an in-depth exploration of a cooperative repair scheme for distributed storage systems that incorporates the assistance of base stations. It begins by introducing the notion of an admissible region, which encapsulates all possible points that support a cooperative regenerating scheme characterized by certain parameters. The identification of the admissible region was realized through the formulation and solution of a rigorous optimization problem. The results from the numerical study, illustrated by the admissible region plot, confirm the efficacy of the suggested approach over previous works, presenting an expanded admissible region that optimizes the tradeoff between storage size and repair bandwidth cost. Furthermore, it reveals the varied involvement of BSs in the repair process depending on the operating point.

The thesis then delved into the complexities of a dynamic DSS environment where nodes can randomly enter or leave the network, following a Poisson process. A new layer of randomness was added to the previous optimization problem to account for the variable number of nodes that require regeneration at each stage. A model for a cellular DSS following an M/M/∞ queuing model has been presented, accommodating the dynamics of the node population within the network. The Poisson process was utilized to model node arrivals and departures, and the subsequent optimization problem incorporated these dynamics. The numerical analysis revealed that the inclusion of BSs extends the admissible region of the trade-off curve when the number of repairing nodes varies at different time steps. It highlighted the distinctive potential of BS-assisted cooperative repair strategies in comparison with local cooperative and single-node repair schemes, especially at the point of minimum storage repair.

Then this thesis presents an in-depth analysis of the balances between the decoding threshold, average repair bandwidth, and code rate, elucidated through the numerical resolution of a specifically designated optimization problem. This issue is central to LDPC code design optimization within specific repair strategies. This analysis un-

derscores the influences of two repair protocols - the random access repair protocol and the ideal repair protocol - on these relationships. For random access repair protocol, the thesis proposes a method to determine $\rho(x)$ by minimizing equation (3.2). Following Richardson *et al.*'s suggested method, which is using continuous degrees to create near-optimal degree distributions, explained via fractional phantom distribution. For a fixed E, the minimum value of (3.2) can be found where $d_i^{(c)} = d$ for $i = 1, ..., m$. If $\bar{\gamma}_{\min}^{(\text{rand})}$ is a non-integer, then d must also be a real number and should be distributed according to a fractional phantom distribution. The thesis introduces a theorem for an LDPC code ensemble, defining the check node degree distribution $\rho(x)$ to achieve the minimum average repair bandwidth.

The thesis also finds that the minimum value of the decoding threshold $\epsilon_{\min}^*$ is obtained when the code rate is $R = 1 - \frac{2}{d_c}$. As a result, for any LDPC code with a constant code rate-R and $\epsilon^* = \epsilon_{\min}^*$, a closed form expression for $\bar{\gamma}_{(\min)}^{(\text{rand})}$ can be calculated. This relationship is presented in Theorem 3.2 and demonstrated for selected code rates in Table 3.1. The thesis then proposes a more efficient procedure called the ideal repair protocol to overcome the limitations of the random access repair protocol. The ideal repair protocol selects the check node with the smallest degree to repair the failed node. Considering this, the thesis provides a method for calculating the average repair bandwidth for this protocol based on the parameters $\lambda(x)$ and $\rho(x)$.

In addition, the thesis describes an in-depth examination of the joint optimization of the decoding threshold and repair bandwidth in LDPC code design, emphasizing the use of DE algorithm. This algorithm simplifies complex multi-objective problems into manageable single-objective tasks through the epsilon constraint method. The findings of the thesis demonstrate the superiority of the ideal repair protocol in achieving lower decoding thresholds compared to the random repair protocol, especially at higher code rates. This characteristic is notably desirable for data storage systems that demand high efficiency. The implications of these results extend to the field of error correction codes, suggesting that the ideal repair protocol could be a better choice in practical applications when low decoding thresholds are a priority.

From a practical perspective, in this thesis, the derived LDPC codes were subjected to Monte Carlo simulations to assess their finite length performance. The empirical results reaffirmed the theoretical findings, suggesting that a decline in repair bandwidth might negatively impact the code's error recovery properties for the repair protocols. However, the code designed for the ideal protocol displayed superior recovery performance for the same or even reduced bandwidth, demonstrating the potential for further optimization through carefully designed check node irregularity.

In a DSS with LDPC encoded data, a peeling decoder is used to recover data from k out of n storage units. However, issues arise when no columns are left in the residual matrix $\mathbf{H}^{(\mathrm{RM})}$ that have at least one element corresponding to a row with unit row weight, a situation in which the decoder fails to interpret the remaining symbols. These remaining symbols are called stopping sets, a significant problem in binary erasure transmission channels. However, this problem is mitigated in DSSs as the system deals with storage unit erasure rather than symbol erasure, which simplifies the problem of examining all $n - k$ out of n combinations of storage units to find stopping sets.

This thesis demonstrates how stopping sets are identified and eliminated. The stopping sets are found after allocating symbols to storage units by determining the erasure patterns of possible storage units. Once these stopping sets are identified, the goal is to eliminate them using a method proposed in [52]. This thesis also suggests a greedy algorithm for determining the bandwidth required to repair a failed storage unit using a modified version of the peeling decoder. This algorithm prioritizes the check node that results in the minimum repair bandwidth at the current step rather than randomly selecting one, potentially reducing the bandwidth required for single-node repair.

The paper titled “Base Station-Assisted Cooperative Network Coding for Cellular Systems with Link Constraints” was authored by S. S. Arslan, M. Pourmandi, and E. Haytaoglu. It was published in the 2022 proceedings of the IEEE International Symposium on Information Theory (ISIT) [41]. This reference pertains to the content found in Chapter 2.

The paper titled “Average Bandwidth-Cost vs. Storage Trade-off for BS-Assisted Distributed Storage Networks” was authored by M. Pourmandi, S. S. Arslan, A. E. Pusane, E. Haytaoglu, and A. C. Tengiz. This paper was published in the 2021 proceedings of the 29th Signal Processing and Communications Applications Conference (SIU) [42]. This reference is relevant to the content found in Chapter 2.

The paper titled “Minimum Repair Bandwidth LDPC Codes for Distributed Storage Systems” was authored by M. Pourmandi, A. E. Pusane, S. S. Arslan, and E. Haytaoglu. It was published in the IEEE Communications Letters [45]. This reference corresponds to the content related to Chapter 3.

REFERENCES

1. Dimakis, A. G., P. B. Godfrey, Y. Wu, M. J. Wainwright and K. Ramchandran, "Network Coding for Distributed Storage Systems", *IEEE Transactions on Information Theory*, Vol. 56, No. 9, pp. 4539–4551, 2010.

2. Ahlswede, R., N. Cai, S.-Y. Li and R. Yeung, "Network Information Flow", *IEEE Transactions on Information Theory*, Vol. 46, No. 4, pp. 1204–1216, 2000.

3. Han, J. and L. A. Lastras-Montano, "Reliable Memories with Subline Accesses", *IEEE International Symposium on Information Theory*, pp. 2531–2535, Nice, France, 2007.

4. Shum, K. W., "Cooperative Regenerating Codes for Distributed Storage Systems", *IEEE International Conference on Communications (ICC)*, pp. 1–5, Kyoto, Japan, 2011.

5. Kermarrec, A.-M., N. Le Scouarnec and G. Straub, "Repairing Multiple Failures with Coordinated and Adaptive Regenerating Codes", *International Symposium on Networking Coding*, pp. 1–6, Beijing, China, 2011.

6. Calis, G. and O. O. Koyluoglu, "On the Maintenance of Distributed Storage Systems with Backup Node for Repair", *International Symposium on Information Theory and Its Applications (ISITA)*, pp. 46–50, Monterey, CA, USA, 2016.

7. Hu, Y., Y. Xu, X. Wang, C. Zhan and P. Li, "Cooperative Recovery of Distributed Storage Systems from Multiple Losses with Network Coding", *IEEE Journal on Selected Areas in Communications*, Vol. 28, No. 2, pp. 268–276, 2010.

8. Shivaramaiah, S., G. Calis, O. O. Koyluoglu and L. Lazos, "Threshold-Based File Maintenance Strategies for Mobile Cloud Storage Systems", *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Washington, DC, USA, 2016.

9. Pedersen, J., A. Graell i Amat, I. Andriyanova and F. Brännström, "Repair Scheduling in Wireless Distributed Storage with D2D Communication", *IEEE Information Theory Workshop - Fall (ITW)*, pp. 69–73, Jeju, Korea (South), 2015.

10. Gopal Benerjee, K. and M. K. Gupta, "Tradeoff for Heterogeneous Distributed Storage Systems between Storage and Repair Cost", *ArXiv E-prints*, p. arXiv:1503.02276, Mar. 2015.

11. Ali, M., S. Mumtaz, S. Qaisar and M. Naeem, "Smart Heterogeneous Networks: A 5G Paradigm", *Telecommunication Systems*, Vol. 66, No. 2, pp. 311–330, 2017.

12. Jin, H., A. Khandekar, R. McEliece *et al.*, "Irregular Repeat-Accumulate Codes", *Proc. 2nd Int. Symp. Turbo Codes and Related Topics*, pp. 1–8, Brest, France, 2000.

13. Thorpe, J., "Low-Density Parity-Check (LDPC) Codes Constructed from Protographs", *IPN Progress Report*, Vol. 42, No. 154, pp. 42–154, 2003.

14. Kudekar, S., T. J. Richardson and R. L. Urbanke, "Threshold Saturation via Spatial Coupling: Why Convolutional LDPC Ensembles Perform So Well Over the BEC", *IEEE Transactions on Information Theory*, Vol. 57, No. 2, pp. 803–834, 2011.

15. Park, H., D. Lee and J. Moon, "LDPC Code Design for Distributed Storage: Balancing Repair Bandwidth, Reliability, and Storage Overhead", *IEEE Transactions on Communications*, Vol. 66, No. 2, pp. 507–520, 2018.

16. Richardson, T. and R. Urbanke, *Modern Coding Theory*, Cambridge University Press, New York, USA, 2008.

17. Pedersen, J., A. Graell i Amat, I. Andriyanova and F. Brännström, "Distributed Storage in Mobile Wireless Networks With Device-to-Device Communication", *IEEE Transactions on Communications*, Vol. 64, No. 11, pp. 4862–4878, 2016.

18. Bhuvaneshwari, P. and C. Tharini, "Review on LDPC Codes for Big Data Storage", *Wireless Personal Communications*, Vol. 117, No. 2, pp. 1601–1625, 2021.

19. Yongmei, W., C. Fengmin and L. K. Cher, "Large LDPC Codes for Big Data Storage", *Proceedings of the ASE BigData and Social Informatics*, pp. 1–6, New York, NY, USA, 2015.

20. Ma, Z., Z. Zhang, Z. Ding, P. Fan and H. Li, "Key Techniques for 5G Wireless Communications: Network Architecture, Physical Layer, and MAC Layer Perspectives", *Science China Information Sciences*, Vol. 58, No. 4, pp. 1–20, 2015.

21. Nawaz, S. J., S. K. Sharma, S. Wyne, M. N. Patwary and M. Asaduzzaman, "Quantum Machine Learning for 6G Communication Networks: State-of-the-Art and Vision for the Future", *IEEE Access*, Vol. 7, pp. 46317–46350, 2019.

22. Dimakis, A. G., K. Ramchandran, Y. Wu and C. Suh, "A Survey on Network Codes for Distributed Storage", *Proceedings of the IEEE*, Vol. 99, No. 3, pp. 476–489, 2011.

23. Sohn, J.-Y., B. Choi, S. W. Yoon and J. Moon, "Capacity of Clustered Distributed Storage", *IEEE Transactions on Information Theory*, Vol. 65, No. 1, pp. 81–107, 2019.

24. Tebbi, A., T. H. Chan and C. W. Sung, "Multi-Rack Distributed Data Storage Networks", *IEEE Transactions on Information Theory*, Vol. 65, No. 10, pp. 6072–6088, 2019.

25. Papailiopoulos, D. S. and A. G. Dimakis, "Locally Repairable Codes", *IEEE Transactions on Information Theory*, Vol. 60, No. 10, pp. 5843–5855, 2014.

26. Tamo, I. and A. Barg, "A Family of Optimal Locally Recoverable Codes", *IEEE Transactions on Information Theory*, Vol. 60, No. 8, pp. 4661–4676, 2014.

27. Balaji, S., M. N. Krishnan, M. Vajha, V. Ramkumar, B. Sasidharan and P. V. Kumar, "Erasure Coding for Distributed Storage: An Overview", *Science China Information Sciences*, Vol. 61, pp. 1–45, 2018.

28. Jin, L., L. Ma and C. Xing, "Construction of Optimal Locally Repairable Codes via Automorphism Groups of Rational Function Fields", *IEEE Transactions on Information Theory*, Vol. 66, No. 1, pp. 210–221, 2019.

29. Li, J. and B. Li, "Cooperative Repair with Minimum-Storage Regenerating Codes for Distributed Storage", *IEEE INFOCOM Conference on Computer Communications*, pp. 316–324, Toronto, ON, Canada, 2014.

30. Ryan, W. and S. Lin, *Channel Codes: Classical and Modern*, Cambridge University Press, New York, USA, 2009.

31. Moon, T. K., *Error Correction Coding: Mathematical Methods and Algorithms*, John Wiley & Sons, Hoboken, NJ, USA, 2020.

32. Haytaoglu, E., E. Kaya and S. S. Arslan, "Data Repair-Efficient Fault Tolerance for Cellular Networks Using LDPC Codes", *IEEE Transactions on Communications*, Vol. 70, No. 1, pp. 19–31, 2022.

33. Gaidioz, B., B. Koblitz and N. Santos, "Exploring High Performance Distributed File Storage Using LDPC Codes", *Parallel Computing*, Vol. 33, No. 4, pp. 264–274, 2007.

34. Wei, Y., Y. W. Foo, K. C. Lim and F. Chen, "The Auto-Configurable LDPC Codes for Distributed Storage", *IEEE 17th International Conference on Computational Science and Engineering*, pp. 1332–1338, Chengdu, China, 2014.

35. Plank, J. and M. Thomason, "A Practical Analysis of Low-Density Parity-Check Erasure Codes for Wide-Area Storage Applications", *International Conference on Dependable Systems and Networks*, pp. 115–124, Florence, Italy, 2004.

36. Richardson, T., M. Shokrollahi and R. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes", *IEEE Transactions on Information Theory*, Vol. 47, No. 2, pp. 619–637, 2001.

37. Shokrollahi, A. and R. Storn, "Design of Efficient Erasure Codes with Differential Evolution", *IEEE International Symposium on Information Theory*, p. 5, Sorrento, Italy, 2000.

38. Deb, K., "Multi-Objective Optimisation Using Evolutionary Algorithms: An Introduction", *Multi-Objective Evolutionary Optimisation for Product Design and Manufacturing*, pp. 3–34, Springer, London, U.K., 2011.

39. Deng, W., S. Shang, X. Cai, H. Zhao, Y. Song and J. Xu, "An Improved Differential Evolution Algorithm and Its Application in Optimization Problem", *Soft Computing*, Vol. 25, No. 7, pp. 5277–5298, 2021.

40. Yang, X.-S., *Nature-Inspired Optimization Algorithms*, Academic Press, London, U.K., 2020.

41. Arslan, S. S., M. Pourmandi and E. Haytaoglu, "Base Station-Assisted Cooperative Network Coding for Cellular Systems with Link Constraints", *IEEE International Symposium on Information Theory (ISIT)*, pp. 2815–2820, Espoo, Finland, 2022.

42. Pourmandi, M., S. S. Arslan, A. E. Pusane, E. Haytaoglu and A. C. Tengiz, "Bİ-Yardimli Dağitik Depolama Ağlannda Ortalama Bant Genişliği Maliyeti ve Depolama Ödunleşimi Average Bandwidth-Cost vs. Storage Trade-off for BS-assisted Distributed Storage Networks", *29th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Istanbul, Turkey, 2021.

43. Ye, M. and A. Barg, "Explicit Constructions of High-Rate MDS Array Codes With Optimal Repair Bandwidth", *IEEE Transactions on Information Theory*, Vol. 63, No. 4, pp. 2001–2014, 2017.

44. Simon, D., *Evolutionary Optimization Algorithms*, John Wiley & Sons, Hoboken, NJ, USA, 2013.

45. Pourmandi, M., A. E. Pusane, S. S. Arslan and E. Haytaoglu, "Minimum Repair Bandwidth LDPC Codes for Distributed Storage Systems", *IEEE Communications Letters*, Vol. 27, No. 2, pp. 428–432, 2023.

46. Martinez-Mateo, J., D. Elkouss and V. Martin, "Improved Construction of Irregular Progressive Edge-Growth Tanner Graphs", *IEEE Communications Letters*, Vol. 14, No. 12, pp. 1155–1157, 2010.

47. Razaghi, P. and W. Yu, "Bilayer Low-Density Parity-Check Codes for Decode-and-Forward in Relay Channels", *IEEE Transactions on Information Theory*, Vol. 53, No. 10, pp. 3723–3739, 2007.

48. Ram, E. and Y. Cassuto, "Design of Bilayer and Multi-Layer LDPC Ensembles From Individual Degree Distributions", *IEEE Transactions on Information Theory*, Vol. 67, No. 11, pp. 7096–7109, 2021.

49. Ram, E. and Y. Cassuto, "LDPC Codes with Local and Global Decoding", *IEEE International Symposium on Information Theory (ISIT)*, pp. 1151–1155, Vail, CO, USA, 2018.

50. Cassuto, Y., E. Hemo, S. Puchinger and M. Bossert, "Multi-Block Interleaved Codes for Local and Global Read Access", *IEEE International Symposium on Information Theory (ISIT)*, pp. 1758–1762, Aachen, Germany, 2017.

51. Blaum, M. and S. R. Hetzler, "Integrated Interleaved Codes as Locally Recoverable Codes: Properties and Performance", *ArXiv Preprint ArXiv:1602.02704*, 2016.

52. Jiao, X., J. Mu, J. Song and L. Zhou, "Eliminating Small Stopping Sets in Irregular Low-Density Parity-Check Codes", *IEEE Communications Letters*, Vol. 13, No. 6, pp. 435–437, 2009.

53. Schwartz, M. and A. Vardy, "On the Stopping Distance and the Stopping Redundancy of Codes", *IEEE Transactions on Information Theory*, Vol. 52, No. 3, pp. 922–932, 2006.

54. Han, J. and P. H. Siegel, "Improved Upper Bounds on Stopping Redundancy", *IEEE Transactions on Information Theory*, Vol. 53, No. 1, pp. 90–104, 2006.

55. Pääkkönen, J., C. Hollanti and O. Tirkkonen, "Device-to-Device Data Storage with Regenerating Codes", *International Workshop on Multiple Access Communications*, pp. 57–69, Springer, Helsinki, Finland, 2015.

56. Barbi, R., P. Felber, L. Hayez and H. Mercier, "Convolutional LPDC Codes for Distributed Storage Systems", *IEEE International Symposium on Information Theory (ISIT)*, pp. 1572–1576, Paris, France, 2019.

57. Richardson, T., "Error Floors of LDPC Codes", *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, Vol. 41, pp. 1426–1435, The University; 1998, Urbana-Champaign, IL, 2003.

APPENDIX A: PROOFS IN CHAPTER 2

A.1. Proof of Theorem 2.1

At the BS-MBCCR operating point, similar to the reasoning given in [4], for a given set of values of $r_1, ..., r_M$, the minimum repair bandwidth cost is the intersection between $\beta = 2\beta'$ and $\beta - \beta'$ planes for $g = k$ and $\alpha \geq d\beta + (t-1)\beta' + \sum_{l=1}^{M} s_l r_l$. To find the intersection, $\beta = 2\beta'$ is substituted in the $\beta - \beta'$ plane to get

$$\alpha(k-k) + k\beta\Big(d + \sum_{l=1}^{M} s_l r_l - k + \frac{k+1}{2}\Big) + \frac{k\beta}{2}(t-1) = \mathcal{F}, \tag{A.1}$$

which subsequently leads to

$$\beta = \frac{2F}{k\Big(2(d + \sum_{l=1}^{M} s_l r_l) + t - k\Big)}. \tag{A.2}$$

Thus, by substituting β and $\beta' = \beta/2$ in the expression of $\gamma_c(\mathbf{s})$, the result follows. As for the BS-MSCR operating point, let $g = lt + q$, where l is an integer and $0 \leq q \leq t$. Note that $l = 0$ corresponds to the minimum storage point i.e., $\alpha = \frac{F}{k}$. Taking into this consideration, for a given set of values of $r_1, ..., r_M$, the intersection between $\beta = \beta'$ with $\beta - \beta'$ plane subject to the following constraint gives us the BS-MSCR point:

$$\alpha(k-g) + \beta\Big(g(d + \sum_{l=1}^{M} r_l - k) + \frac{g^2 + \psi_{g,t}}{2}\Big) + \beta(gt - \psi_{g,t}) = \mathcal{F}. \tag{A.3}$$

A.2. Proof of Theorem 2.2

In BS-MBCCR operating point, it is assumed that ρ number of base stations that have the cost factors $\{w_1, ..., w_\rho\}$ are utilized. In other words,$\{r_1, ..., r_\rho\}$ are non-zero values satisfying the condition

$$\sum_{l=1}^{\rho} w_l r_l \leq \frac{2d + t - 1}{2d - k + t} \sum_{l=1}^{\rho} r_l. \tag{A.4}$$

Note that to enable the BSs to minimize the overall bandwidth cost, the condition $\gamma_c(\mathbf{s}_\rho) \leq \gamma_c(\mathbf{0})$ must be met, leading to the establishment of equation (A.4). Here, $\mathbf{0}$

denotes the all-zero vector. Accordingly, inequality (A.4) can be rewritten as

$$\sum_{l=1}^{\rho} w_l r_l = \tau \bar{w}_t \sum_{l=1}^{\rho} r_l. \tag{A.5}$$

The repair bandwidth cost can be reformulated if τ satisfies $0 \leq \tau \leq 1$. With the reformulation of $\gamma_c(\mathbf{s}_\rho)$ in terms of ρ, the expression for repair bandwidth cost can be rewritten as

$$\gamma_c(\mathbf{s}_\rho) = \frac{\bar{w}_t}{k} \frac{2d - k + t + 2\tau \sum_{l=1}^{\rho} r_l}{2d - k + t + 2 \sum_{i=1}^{\rho} r_i}. \tag{A.6}$$

Note that equation (A.6) is inversely related to $\sum_{l=1}^{\rho} r_l$ (it can be easily shown by taking the derivative). In other words, in order to minimize $\gamma_c(\mathbf{s}_\rho)$, it is sufficient to maximize the term $\sum_{l=1}^{\rho} r_l$. A similar proof following the same line of reasoning can be given for BS-MSCR point.

APPENDIX B: PROOFS IN CHAPTER 3

B.1. Proof of Theorem 3.1

Let $S_j^{(c)} = \{d_i^{(c)} : d_i^{(c)} = j, \forall i \in \{1, ..., m\}\}$ denote the set of check nodes with degree-j for $\forall j \in \{2, ..., d_{c_{\max}}\}$. Then, ρ_j is given by $\rho_j = j|S_j^{(c)}|\Big/E$, and the numerator of $\bar{\gamma}$ can be decomposed into $\sum_{j=2}^{d_{c\max}} j|S_j^{(c)}|(j-1)$. Asymptotically, note that keeping the number of edges E constant, $\bar{\gamma}$ can be expressed in an alternative form as

$$\bar{\gamma}^{(\text{rand})}(\rho) = \sum\nolimits_{i=2}^{d_{c\max}} \rho_i(i-1). \tag{B.1}$$

For $d \in \mathbb{Z}$, it is proved that the minimum repair bandwidth is achieved when check nodes are regular, i.e, $\rho(x) = x^{d-1}$ [15]. In [16], it is conjectured that two non-zero successive check node degrees are enough to maximize the decoding threshold (for constant code rate). Then, based on (B.1), for $d \notin \mathbb{Z}$, using the idea of the fractional phantom distribution, the real check node degree d that minimizes (B.1) can be mapped to the non-zero check node degrees that satisfy

$$\rho_{\lfloor d \rfloor} + \rho_{\lceil d \rceil} = 1, \tag{B.2}$$

$$\lfloor d \rfloor \, \rho_{\lfloor d \rfloor} + \lceil d \rceil \, \rho_{\lceil d \rceil} = d. \tag{B.3}$$

Solving this system of equations for $\rho(x)$ completes the proof.

B.2. Proof of Theorem 3.2

Based on Theorem 3.2, for a given $\rho(x)$ and a decoding threshold ϵ^*, code design can be tossed as a linear maximization problem in terms of variable node degree distribution $\lambda(x)$, i.e., $\arg\max_{\lambda} \sum_{i\geq 2} \lambda_i/i$, subject to $\sum_{i=2}^{d_{v\max}} \lambda_i = 1$ [16]. Since $R < 1 - \epsilon^*$, for $\epsilon^* = \epsilon^*_{\min}$, the solution is $\lambda(x) = x$. And, remembering that $\bar{\gamma}^{(\text{rand})}_{\min} = d - 1$, $\rho(x)$ is given by Theorem 1, accordingly, $\bar{d}_c = \lfloor d \rfloor \lceil d \rceil / (\lfloor d \rfloor + \lceil d \rceil - d)$. Since $R = 1 - 2/\bar{d}_c$, then d can be obtained as $d = \Psi(d)$ where $\Psi(d) = \lceil d \rceil + \lfloor d \rfloor - (\lceil d \rceil \, \lfloor d \rfloor)\,(1-R)/2$.

Case 1: If $d \in \mathbb{Z}$, then it is obtained that $\bar{\gamma}^{(\text{rand})}_{\min} = 2/(1-R) - 1$.

Case 2: If $d \notin \mathbb{Z}$, $\Psi(d)$ does not change for $\forall d'$ such that $\lfloor d \rfloor < d' < \lceil d \rceil$, consequently,

let $d = \Psi(d')$. One of the candidates for d' is $\lfloor d \rfloor < \bar{d}_c = 2/(1-R) < \lceil d \rceil$. Since d' is arbitrary, replacing d' with $\bar{d}_c$ into $\Psi(d')$ results $\bar{\gamma}_{\min}^{(\text{rand})}$.

B.3. Proof of Theorem 3.3

Let L_j be the probability that any variable node v has degree j, which constitutes the node-perspective variable node degree distribution [16]. Given a variable node v with degree j and the repair bandwidth γ_j, then the average repair bandwidth per node $\bar{\gamma}(\lambda, \rho)$ for an ensemble of LDPC codes can be expressed as $\bar{\gamma}(\lambda, \rho) = \sum_{j=2}^{d_{v\max}} L_j \gamma_j$, where $L_j = \bar{d}_v \frac{\lambda_j}{j}$.

Recall that *multiset* is a regular set that allows repetition of elements. For example $[a, a, b, b]$ is a multiset with 4 elements. If v has degree j, let its adjacent check node degrees'multiset be $[d_1, \ldots, d_j]$. For each realization of this multiset, there are $|\pi(d_1, \ldots, d_j)|$ unique permutations, where $\pi(.)$ is the permutation set. Accordingly, realization of degrees for adjacent check nodes can be specified as a set of multisets denoted by $A^{(c)}(j) = \{[d_1, \ldots, d_j] : d_i \leq d_{c_{\max}}, i = 1, \ldots, j\}$. To find γ_j, the probability of each multiset of check node degrees has to be determined. Once edge assignments are made independently, ρ_{d_i} indicates the probabilities of connecting an edge to a check node of degree d_i. Also, since there are $|\pi(d_1, \ldots, d_j)|$ unique permutations for adjacent check nodes of a variable node v with degree j, the probability of each realization of check node degrees can be found by

$$P\left([d_1, ..., d_j]\right) = |\pi(d_1, \ldots, d_j)| \prod_{k=1}^{j} \rho_{d_k}. \tag{B.4}$$

Then, γ_j is obtained by

$$\gamma_j = \sum_{[d_1,\ldots,d_j] \in A_j^{(c)}} P\left([d_1, \ldots, d_j]\right) \min_{l \leq j}(d_l - 1). \tag{B.5}$$

Consequently, based on the multiset of check node degrees, when the ideal repair protocol is utilized, $\bar{\gamma}^{(\text{ideal})}(\lambda, \rho) = \bar{\gamma}(\lambda, \rho)$.

In the following, let the multiset of check node degrees $[d_1, \ldots, d_j]$ be mapped to

$|\underline{n}|$ node degree types labeled by the set $\{i_1, \ldots, i_{|\underline{n}|}\}$. Given that there are $|\underline{n}|$ node degree types based on their values, $|\pi(d_1, ..., d_j)|$ is calculated as

$$|\pi(d_1, ..., d_j)| = \frac{j!}{n_{i_1}!...n_{i_{|\underline{n}|}}!}. \tag{B.6}$$

Therefore, (B.4) can be rewritten in terms of $|\underline{n}|$ and j as

$$P(\underline{n}, j) = \frac{j!}{n_{i_1}!...n_{i_{|\underline{n}|}}!} \prod_{k \in \{i_1, ..., i_{|\underline{n}|}\}} (\rho_k)^{n_k}. \tag{B.7}$$

Finally, if (B.7) is plugged into (B.5), the result follows.