Improving the Update Complexity of Locally Repairable Linear Block Codes in Distributed Storage Systems

by

Mehrtash Mehrabi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Communications

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Distributed and cloud storage systems are used to reliably store large-scale data. Erasure codes have been recently proposed and used in real-world distributed and cloud storage systems such as Google File System, Microsoft Azure Storage, and Facebook HDFS-RAID, for reliable data storage. Conventional erasure codes, however, are not suitable for distributed storage systems, as they cause significant repair bandwidth and disk I/O. As a solution, a class of erasure codes called locally repairable codes (LRCs) have been proposed. Using LRCs repairing a failed storage nodes requires access to only a small number of available nodes. This property of LRCs results in a relatively low bandwidth and disk I/O operations for repairing a failed node. In this thesis, we study update complexity of LRCs. Update complexity of an erasure code is a measure of the computation, I/O and networking costs associated with updating an information block in a distributed storage system. LRCs with low update complexities are desirable. In this thesis, we derive lower bounds on update complexity of an important class of LRCs. Then, we propose a new set of LRCs, and, using the bound, we show that our designed codes have either optimal or near-optimal update complexities. Interestingly, our proposed codes improve update complexity without sacrificing important code parameters such as minimum distance, rate, or locality.

Preface

The results presented in Chapter 3 were presented in IEEE 86th Vehicular Technology Conference: VTC2017-Fall. Some of the results of Section 4.1 were presented in IEEE 15th Canadian Workshop on Information Theory (CWIT 2017). Finally, the major results presented in Chapter 4 were summited to IEEE Tranactions on Communications in August 2017.

"I would like to dedicate my work to my parents."

Acknowledgements

I want to use this opportunity to thank everyone who has helped me in preparation of this work. First of all, I should thank my supervisors, Dr. Masoud Ardakani and Dr. Majid Khabbazian, who have been my greatest helps during the past two years in every aspect. Second, I want to thank the committee members, Dr. Hao Liang and Dr. Pedram Mousavi, for dedicating their time and energy, reading my thesis and providing their invaluable ideas. Then, I want to thank my parents for their supports and also thank every teacher, professor or friend who has helped me learn something new during all my years of education, fulfilling my thirst for knowledge.

List of Symbols

$(\cdot)^T$	Matrix transpose operation	5
$[a]$	The set of integers from one to a; $[1, a] = \{1, \cdots, a\}$	5
$\lvert\mathcal{A}\rvert$	Cardinality of set $\mathcal{A}$	5
$\mathbf{G}$	The generator matrix of size $k \times n$	9
$\mathbf{H}$	The parity check matrix of size $(n-k) \times n$	10
$\mathbf{x}$	The information vector of size $1 \times k$	9
$\mathbf{y}$	The encoded vector of size $1 \times n$	9
R	Rate of the code	5
$\mathbb{F}_q$	Finite field with cardinality q	5, 6
d	Minimum distance of the code	2
k	Dimension of the code	2
n	Length of the code	2
$\mathbf{0}_{b\times c}$	A zero matrix of size $b \times c$	5
$\mathbf{1}_a$	A column vector of ones with size a	5
$\mathbf{I}_a$	Identity matrix of size a	5
$\mathcal{T}$	Tanner graph	10
$\otimes$	Kronecker product	5
$\mathcal{C}$	A linear block code	9

List of Abbreviations

$GF(q)$	Galois field of order q	7
IO/sec	Input/output operations per second	13
DSS	Distributed storage system	1
HDD	Hard disk drive	12
HDFS	Hadoop distributed file system	2
I/O	Input/output	2
LRC	Locally Repairable Code	2
MDS	Maximum distance separable	2
RAID	Redundant array of independent disks	2
RGC	Regenerating code	17
RS	Reed-Solomon	11
SSD	Solid state drive	12
UC	Update complexity	4

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

A distributed storage system (DSS) uses many storage devices, called data nodes, to store data. Since hardware and software failures can result in data unavailability or even permanent data loss, it is crucial to add redundancy to the stored data. This way, lost data can be restored using the available redundancies. The simplest solution for adding redundancy is to store multiple replicas of data. This simple solution, known as the replication method, is widely used in DSSs [2]. However, with the rapid growth of data and the significant storage overhead (and therefore maintenance cost) of replication method, this solution is becoming less attractive.

Using channel codes (in particular, those suitable for recovering erasures, also known as erasure codes) is another way for introducing redundancy in distributed and cloud storage systems. Erasure codes can provide the redundancy needed in DSSs with significantly lower storage overhead compared to the replication method. Recently, systematic erasure codes[1] have been employed in real-world distributed and cloud storage systems, such as Facebook HDFS-RAID [2], Google File Systems [3], and Microsoft Windows Azure Storage [1]. In order to use an erasure code in a DSS, first,

[1]In systematic codes, information blocks can be directly stored and read with no encoding and decoding processes. This is why, in DSSs, systematic codes are preferred to the non-systematic ones. In this work, we only consider systematic erasure codes [1, 2].

a stripe of data is split into k information blocks. Then, using an (n, k) erasure code, n encoded blocks are generated from k information blocks and stored in n different storage nodes. In systematic erasure codes, the set of n encoded blocks consists of all the k information blocks plus n-k parity blocks. Each parity block is a function of the information blocks.

The minimum distance of a code is the minimum Hamming distance between any two codewords. By using an (n, k) erasure code with minimum distance d, the DSS is able to tolerate up to $d - 1$ node failures. In an (n, k, d) erasure code, the minimum distance d is bounded to

$$d \leq n - k + 1.$$

This bound is known as the Singleton bound. Maximum distance separable codes (MDS codes) are a class of erasure codes that achieve the Singleton bound with equality, i.e., $d = n - k + 1$.

One problem with using conventional erasure codes in DSSs is their high repair cost. While, in the replication method, a failed node is recovered by accessing only one available replica, in erasure codes, recovering one failure may need accessing many nodes, resulting in huge amounts of disk I/O and data traffic. For example, MDS codes provide the largest possible minimum distance for a given storage overhead $\frac{n-k}{k}$, but, recovering even a single node failure needs access to k available nodes. Considering the size of data centers, the traffic load caused by conventional optimal erasure codes is significant [2]. Reducing the number of nodes that must be accessed during the recovery process of a failed node is essential in reducing this traffic.

More recently, locally repairable codes (LRCs) were suggested to reduce the number of required nodes during the recovery process of a failed node [4–9]. An important parameter of an LRC is its locality. The locality of an LRC, denoted by r, is defined as the maximum number of nodes needed to be accessed in order to recover a missing block. It is desirable to find LRCs that have a large minimum distance d as well as a small locality r. Naturally, there is a bound on how much d can be improved for a

given n, k and r. In [4] and [5], the following bound is established,

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{1.1}$$

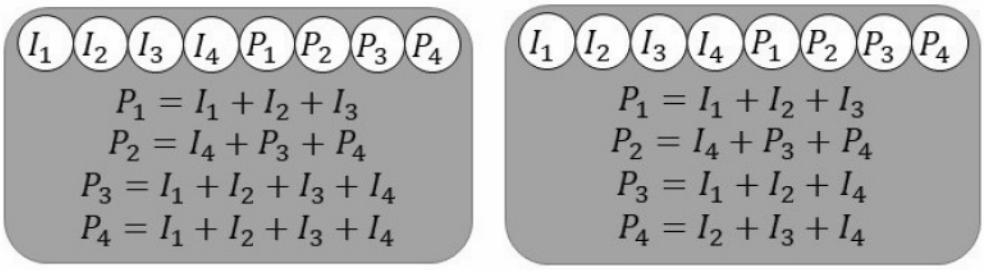
LRCs that achieve this bound with equality are called *optimal*. It is verified that the bound in (1.1) is tight if $(r+1) \mid n$ [5].

Besides network bandwidth and disk I/O, another important measure that should be considered in designing LRCs is their update complexity. For a systematic (n, k, d, r) optimal LRC, there exist $n-k$ parity blocks constructed from k information blocks. While some of these $n-k$ parity blocks are constructed locally from a few blocks to achieve the code locality, some other parity blocks are constructed globally to achieve the required minimum distance. In the existing optimal LRCs, all the information blocks are involved in these globally constructed parity blocks [10–12]. Consequently, if only one information block is updated, all the global parity blocks have to be changed resulting in a costly update process. Is it possible to generate more than one optimal (n, k, d, r) LRC with different update complexity? If so, how can we find optimal LRCs with small update complexity? The latter is the central question we study in this thesis.

Fig. 1.1 shows the simplified structures of two optimal $(n, k, d, r) = (8, 4, 4, 3)$ LRCs. These two LRCs are optimal, that is their minimum distance d achieves the bound (1.1) with equality. In the LRC of Fig. 1.1a, all the information blocks are involved in the two parity blocks P_3 and P_4. However, in the LRC of Fig. 1.1b, only some information blocks are involved in P_3 and P_4. In this example, if one information block is updated, the LRC of Fig. 1.1b, on average, needs 12% less block updates.

1.2 Related Work

In [13], the authors define update complexity as the maximum number of encoded blocks that must be changed when an information block is updated. They construct update-efficient codes that have update complexity scaling sub-linearly with the code length.



(a) (b)

Figure 1.1: Simplified structures of two optimal $(n, k, d, r) = (8, 4, 4, 3)$ LRCs with different update complexity. When one information block is updated, the codes represented in Figs. 1.1a and 1.1b on average need $(3 \times 5 + 1 \times 4)/4 = 4.75$ and $(3 \times 4 + 1 \times 5)/4 = 4.25$ encoded block updates, respectively.

The same definition of update complexity is used in [14, 15], where the authors find proper sufficient conditions to achieve the optimum minimum distance and update-efficiency. To achieve any significant rate with a low probability of error over the binary erasure or binary symmetric channels, they show that the update complexity must scale at least logarithmically in the block-length of the code. Also, they develop tight upper and lower bounds on the number of remaining encoded blocks that are required to recover a single missed encoded block.

1.3 Our Contribution

In this thesis, we study the problem of update complexity (UC) for systematic LRCs. The contributions of this thesis are twofold. First, by taking an existing definition of update complexity and generalizing it, we obtain both upper and lower bounds on UC for an important class of LRCs. Second, we propose an algorithm to design LRCs whose average UC is close or equal to the obtained lower bound.

Considering the size of practical codes, the saving gained by our proposed LRCs can be significant. For example, the $(n, k, d, r) = (16, 10, 5, 5)$ LRC constructed by our proposed algorithm and presented in Chapter 4 leads to more than 16% saving in update complexity compared to the Facebook HDFS-RAID LRC [2], which has identical parameters n, k, d and r. Moreover, due to the size of a data block (e.g., 256

MB in Facebook HDFS-RAID [2]) as well as the massive amount of the stored data in real-world DSSs, even a small reduction in update complexity (UC) can yield significant amount of saving as a single block update requires multiple read, download, and write operations. We remark that our proposed codes improve UC without sacrificing important code parameters such as minimum distance (d), rate ($R= \frac{k}{n}$), or locality (r).

Notations: We denote matrices and vectors by capital bold letters and bold letters, respectively. $\mathbb{F}_q$ and $\otimes$ stand for a finite field of order q and Kronecker product, respectively. $\mathbf{I}_a$ and $\mathbf{0}_{b\times c}$ represent an identity matrix of size a and a zero matrix of size $b \times c$, respectively. $(\cdot)^T$ and $\mathbf{1}_a$ represent matrix transpose operation and a column vector of ones with size a, respectively. For an integer a, $[a]= \{1, \cdots, a\}$. $|\mathcal{A}|$ represents the cardinality of set $\mathcal{A}$.

1.4 Organization of Thesis

The remainder of this thesis is organized as follows. In Chapter 2, we provide the required preliminaries including some backgrounds in linear algebra, coding theory, and storage systems. In Chapter 3, we formally define the update complexity and as an example we study the update complexity of Facebook HDFS-RAID LRC and propose a new LRC with the same code parameters with lower update complexity. In Chapter 4, we obtain lower and upper bounds on update complexity of LRCs and introduce our proposed LRCs with small update complexity and compare our proposed LRCs with other conventional LRCs in terms of update complexity. Finally, in Chapter 5, we conclude the thesis and give some ideas to continue our studies.

Chapter 2

Background

This chapter includes some necessary definitions and assumptions required in the next chapters. It also includes useful preliminaries and backgrounds on the topic of data storage, coding theory and the application of data coding in DSSs.

2.1 Finite Field $\mathbb{F}_q(GF(q))$

Definition 2.1. *Group: Defining a set of elements $\mathcal{A}$ on which a binary operation "$*$" has been defined. The set $\mathcal{A}$ and the operation "$*$" constitute a group, denoted $(\mathcal{A}, *)$, if the operation satisfy the following requirements.*

(i) Closure: $a * b \in \mathcal{A}$ for all $a, b \in \mathcal{A}$.

(ii) Associativity: $(a * b) * c = a * (b * c)$ for all $a, b, c \in \mathcal{A}$.

(iii) Identity element: There exists $e \in \mathcal{A}$, such that $e * a = a * e = a$ for all $a \in \mathcal{A}$,

(iv) Inverse element: For every element $a \in \mathcal{A}$, there exists a unique element $a^{-1} \in \mathcal{A}$, such that $a * a^{-1} = a^{-1} * a = e$ (e is the identity element).

A group is called commutative or abelian if it further satisfies

(v) Commutativity: $a * b = b * a$ for all $a, b \in \mathcal{A}$.

Example 2.1. *The set of integers $\mathbb{Z}$ and the integer addition operation form an abelian group, but, it cannot constitute an abelian group under integer multiplication operation since there is no inverse element in $\mathcal{G}$ under multiplication operation.*

Definition 2.2. *Field: A set $\mathbb{F}$ together with two binary operations "$+$" and "$\times$" is called a Field and is denoted by $(\mathbb{F}, +, \times)$ if*

(i) *$\mathbb{F}$ and the addition operation "$+$" form an abelian group and the additive identity element be "0",*

(ii) *$\mathbb{F} - \{0\}$ (set $\mathbb{F}$ with no additive identity element) forms an abelian group under multiplication operation "$\times$" with multiplicative identity element "1",*

(iii) *For all $a, b, c \in \mathbb{F}$, $(a + b) \times c = a \times c + b \times c$.*

Whenever set $\mathbb{F}$ contains a finite q number of elements, the field is called finite field, denoted $\mathbb{F}_q$, where q is called the order of field $\mathbb{F}$.

A finite field is sometimes referred to as Galois field, as it was first discovered by Evariste Galois, as is denoted by $GF(q)$, where q denotes the order of the field. Each element of a Galois field is called a symbol. A block is simply a vector of symbols.

Definition 2.3. *Addition and multiplication modulo m (or mod m): Addition modulo m is expressed as the following*

$$a + b \equiv c \mod m,$$

and reads as "$a + b$ is equivalent to $c \mod m$. With same expression for multiplication modulo m, we have

$$a \times b \equiv c \mod m,$$

and reads as "$a \times b$ is equivalent to $c \mod m$.

The set $\{0, 1, 2, \ldots p - 1\}$, where p is a prime power, constitute the field $GF(p)$ under modulo p addition and multiplication.

Example 2.2. *$GF(3)$ can be constructed by the set $\{0,1,2\}$ and modulo 3 addition and multiplication is shown in the following tables:*

$+$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

For every prime p and integer m, there is a finite field of order p^m. The order of any finite field is a power of a prime.

Definition 2.4. *Primitive element: A primitive element of a finite field $GF(q)$ is a generator of the multiplicative group of the field. Let $\alpha \in GF(q)$, be the primitive element then we have $\alpha^{(q-1)} = 1$. Consequently, all the non-zero elements of $GF(q)$ can be written as powers of α. For example, 2 is a primitive element of the field $GF(3)$ and $GF(5)$, but not of $GF(7)$.*

2.2 Linear Block codes

Erasure codes introduce controlled amount of redundancy to the original data, providing the ability to recover the lost or unavailable data. A linear block code of size n and dimension k, denoted (n,k) code, creates n encode blocks with applying linear operations on k information blocks.

The simplest example of linear block codes, is the replication method, which is an $(n,1)$ erasure code. In an n-replication method, the original data is replicated $n-1$ times and finally there are n replicas of the original data and thus, the storage overhead is $n-1$. Therefore, using this method, we are able to recover the original data in the case of up to $n-1$ block erasures. As mentioned earlier, 3-replication (which is a (3,1) linear code) is widely used in practice [1,2].

2.2.1 Systematic Linear Block Codes

Let $\mathbf{x} = [x_1, x_2, ..., x_k] \in \mathbb{F}_q^{1\times k}$ represent the k information symbols. An (n, k) systematic linear block code converts $\mathbf{x}$ into n encoded symbols $\mathbf{y} = [y_1, y_2, ..., y_n] \in \mathbb{F}_q^{1\times n}$ by multiplying x by a generator matrix $\mathbf{G}$, that is $\mathbf{y} = \mathbf{xG}$. The vector $\mathbf{y}$ is called codeword. The matrix $\mathbf{G}$ can be presented as $\mathbf{G} = [\mathbf{I}_k, \mathbf{P}] \in \mathbb{F}_q^{k\times n}$, where $\mathbf{P} \in \mathbb{F}_q^{k\times(n-k)}$. The parity check matrix of the code is $\mathbf{H} = [-\mathbf{P}^T, \mathbf{I}_{n-k}] \in \mathbb{F}_q^{(n-k)\times n}$ satisfying $\mathbf{GH}^T = \mathbf{0}_{1\times(n-k)}$ and consequently, $\mathbf{yH}^T = \mathbf{0}_{1\times(n-k)}$.

Definition 2.5. *Hamming weight and Hamming distance: Let* $\mathbf{a}$ *be a vector. Then, Hamming weight of* $\mathbf{a}$*, denoted* $wt(\mathbf{a})$*, is defined as the number of non-zero elements in* $\mathbf{a}$*. Hamming distance between two vectors* $\mathbf{a}$ *and* $\mathbf{b}$*, is defined as* $wt(\mathbf{a} - \mathbf{b})$ *and denoted* $d(\mathbf{a}, \mathbf{b})$*.*

Definition 2.6. *Minimum distance of code (d): The minimum distance d of an erasure code is defined as the minimum Hamming distance between any two distinct codewords. Any* (n, k) *erasure code with minimum distance* d *tolerates any* $d - 1$ *symbol erasures. In an* (n, k, d) *erasure code, the minimum distance* d *is bounded to*

$$d \leq n - k + 1.$$

This bound is known as the Singleton bound [16].

Definition 2.7. *Tanner/Factor graph: Consider an* (n, k) *linear block code. Tanner graph (Factor graph)* $\mathcal{T}$ *of this code is a bipartite graph with* n *variable nodes on one side (usually shown by circles) and* $n - k$ *check nodes on the other side (usually shown by squares) [17, 18]. The variable nodes represent the encoded symbols, and the check nodes capture the dependencies between the encoded symbols. There exists an edge between* j*-th (*$j \in [n]$*) variable node and* i*-th (*$i \in [n - k]$*) check node in the Tanner graph if and only if element* $h_{i,j}$ *(element of* i*-th row and* j*-th column of the parity check matrix* $\mathbf{H}$*) is non zero. Hence, the variable nodes connected to the same check node are linearly dependent and in binary code, XOR of them is zero. Note that, there can be many Tanner graph representation for a single code.*

Example 2.3. *In an* $(n, k, d) = (5, 3, 3)$ *systematic code in* $\mathbb{F}_{2^2}$*, let the generator matrix* $\mathbf{G}$ *be*

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 3 & 2 \\ 0 & 1 & 0 & 2 & 2 \\ 0 & 0 & 1 & 3 & 1 \end{bmatrix} \in \mathbb{F}_{2^2}^{3\times 5}.$$

Let $\mathbf{x} = [x_1, x_2, x_3] \in \mathbb{F}_{2^2}^{1\times 3}$ *be the information symbol vector. Then, the encoded symbol vector (i.e., the codeword)* $\mathbf{y}$ *is generated as*

$$\mathbf{y} = \mathbf{xG} = [x_1, x_2, x_3, 3x_1 + 2x_2 + 3x_3, 2x_1 + 2x_2 + x_3].$$

In other words, $\mathbf{y} = [y_1, y_2, y_3, y_4, y_5] = [x_1, x_2, x_3, 3x_1+2x_2+3x_3, 2x_1+2x_2+x_3] \in \mathbb{F}_{2^2}^{1\times 5}$*. It can be easily shown that, information symbols* x_1, x_2 *and* x_3 *can be recovered by accessing any* $n - d + 1 = 3$ *encoded symbols. For example, you can construct* y_4 *by* y_1, y_2 *and* y_3*, that is* $y_4 = 3y_1 + 2y_2 + 3y_3$*.*

2.2.2 Maximum Distance Separable Codes (MDS codes)

Linear block codes that achieve the Singleton bound with equality are called MDS (maximum distance separable) codes. MDS codes provide the largest possible minimum distance for a given storage overhead $\frac{n-k}{k}$. In an MDS code, recovering a block requires accessing $n - d + 1 = k$ other blocks.

Definition 2.8. *Reed-Solomon (RS) codes: An important class of MDS codes proposed in [19] is RS codes. An* (n, k) *RS code has the parity check matrix* $\mathbf{H}_{RS} = [x^{(i-1)(j-1)}] \in \mathbb{F}_q^{(n-k)\times n}$*, where* $i \in [1, n-k]$*,* $j \in [1, n]$*, and* x *is a primitive element in* $\mathbb{F}_q$*. In matrix* $\mathbf{H}_{RS}$*, any sub-matrix of size* $(n-k)\times(n-k)$ *is full-rank and thus, every* $n-k = d-1$ *column of* $\mathbf{H}_{RS}$ *are independent. In order to find the systematic form of the parity check matrix* $\mathbf{H}_{RS}$*, we can easily multiply the inverse form of an* $(n-k)\times(n-k)$ *sub-matrix of* $\mathbf{H}_{RS}$ *by* $\mathbf{H}_{RS}$*.*

The significant cost of recovery a single data block is a major drawback of RS codes. For an (n, k) RS code, in order to recover one single data block, k available data blocks must be accessed and downloaded. This results in a high traffic load between data

nodes which is not desirable in real-world DSSs. Consequently, recently some DSSs have switched from RS codes to new classes of erasure codes to reduce their overall costs [1–3]. In the following examples we review some RS codes that were used in some real-world DSSs.

Example 2.4. *Facebook HDFS-RAID used an* $(n, k) = (14, 10)$ *RS code with minimum distance* $d = 14 - 10 + 1 = 5$ *in 2012 [2]. The storage overhead of this code is* $\frac{14-10}{10} = 40\%$*. Using this code, 10 data blocks must be accessed to recover a single failed data block.*

Example 2.5. *An* $(n, k, d) = (9, 6, 4)$ *RS code has been used in Google File System, since 2011 [3]. The storage overhead of this RS code is* $\frac{9-6}{6} = 50\%$*, and its minimum distance is equal to* $d = n - k + 1 = 4$*. Furthermore, six data blocks must be accessed and downloaded to recover a single failed/missed data block.*

2.3 Distributed Storage Systems

This section is including the description of some important data storage systems such as hard disk drive (HDD), solid state drive (SSD), and redundant array of independent disks (RAID).

In the following, we formally define HDD and SSD which are the two widely used storage devices. Then, we introduce data stripping and data mirroring as two popular storage techniques which are useful in description of different types of RAID known as the important data storage system.

Definition 2.9. *HDD: A storage device made by a set of accumulated disks is known as a HDD. Each disk is containing some tracks, which are concentric circles, where data is stored. In a HDD, the read/write operation is performed by mechanical arms with two heads, each on one side of a disk.*

Definition 2.10. *SSD: A storage component, constructed by microchips is called a SSD. A SSD has no moving arm and an embedded controller processor is responsible for read/write operations. Compared to HDDs, SSDs are faster in read/write operations*

and they are more energy efficient and are providing safety against magnetism effects with a low failure rate.

Definition 2.11. *Data stripping and data mirroring (replication): In order to store data in a storage device, a stripe of data is partitioned into some blocks and each data block is stored on a HDD. This process is called data stripping and based on the number of HDDs, it can boost throughput (I/O operation per sec). The main drawback of data stripping is the unavailability of data in the case of even a single disk failure.*

Data mirroring is the process of data replication in more than one disk providing high tolerance against failure.

Example 2.6. *Considering two HDDs, each is running at* 100 IO/sec *(I/O operation per sec). Then, using a data stripping process on these two HDDs, we are able to boost throughput to* 200 IO/sec.

Definition 2.12. *RAID: An array of HDDs used to improve throughput and enhance the tolerance against failure is known as RAID. In the following, we review some types of RAID.*

- *RAID 0: Storing data blocks uniformly in more than one disk with no redundant data. A RAID using n disk drives can improve I/O by a factor of n. RAID 0 is beneficial in the case of need for high I/O performance.*

- *RAID 1: Improving the read performance by running mirroring process on data. Hence, data can be read by accessing any single disk. Whenever high reliability and simple read process is crucial, RAID 1 is useful.*

- *RAID 2: Using Hamming code in bit-level instead of block-level to detect errors. Today, HDDs are also profit from applying Hamming code in bit-level to find erasures and thus RAID 2 is no longer in use.*

- *RAID 3: Providing the data stripping process in byte-level and assign a single disk to store parity checksum and consequently, it can recover data in the case of at most one disk failure.*

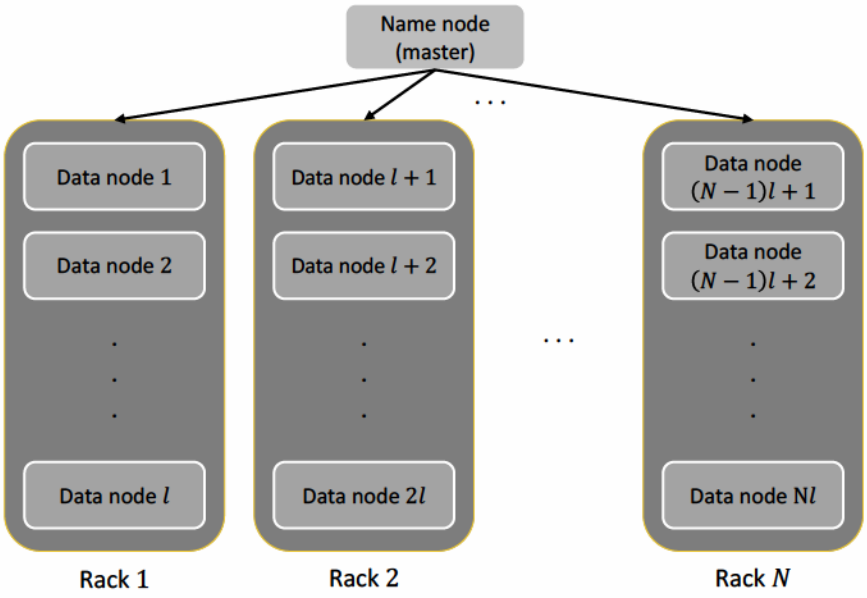


Figure 2.1: A simple structure of HDFS with N racks, each including l data nodes.

- *RAID 4: Same as RAID 3 but in block-level and today, is totally obsolete.*
- *RAID 5: Storing the parity checksum in more than one disk by performing data stripping process in block-level. RAID 5 is able to tolerate one single disk failure in the array and compare to RAID 4, it enhances the data read performance by involving all drives to read data.*
- *RAID 6: Performing a block-level data stripping process and stores two parity checksum in many dicks, causing tolerating up to two disk failures in the array.*
- *RAID 10: Mixing RAID 0 and RAID 1, providing both data stripping and data mirroring methods increasing read and I/O performance, respectively.*

2.3.1 Hadoop Distributed File System (HDFS)

HDFS is a master/slave system which is used to store large-scale data in a reliable way. HDFS uses one master node and N racks, where each rack has a separate network and power cable and consists l data nodes. Fig. 2.1 shows a simple structure of HDFS.

Example 2.7. *Facebook data center uses a HDFS with 100 racks, each rack with 30*

data nodes. Each data node has 20 TB storage capacity and thus, the overall storage capacity of the HDFS is $100 \times 30 \times 20\ TB = 60\ PB$ *[2].*

In HDFS, master node is called name node. Name node is an administrator computer providing some principal data such as file directory of the stored data. Providing the backups of these name nodes is useful to recover the missed data and consequently, improving the reliability.

Data is stored in the data nodes. Data nodes send report to name node frequently to update their status. They also communicate with each other to, for example, store redundant data. In such storage systems, data node failures occur frequently due to several reasons such as hardware/software problems associated with the underlying network or data nodes. In order to recover the lost/erased data and make HDFS reliable, redundancy is required. For example, the approach of keeping several replicas of data in distinct data nodes, known as replication, is widely used [2].

Consider HDFS illustrated in Fig. 2.1. Assume the data nodes 2, $l+1$ and Nl each have a replica of a data block $\mathcal{A}$. Then, if one of these data nodes, for example, $l+1$-th data node fails, the data block $\mathcal{A}$ can be recovered by accessing data node 2, or data node Nl. Although, this 3-replication method offers a simple implementation, it results in a high storage overhead of 200%. As mentioned earlier, recently, systematic erasure codes have been proposed and used in DSSs to decrease storage overhead. In the following section, we provide some more details about the benefits of using systematic erasure codes in such storage systems.

2.4 Erasure Coding in Distributed Storage Systems

Systematic erasure codes have been used in real-world cloud storage systems such as Facebook HDFS-RAID [2], Microsoft Windows Azure Storage [1], and Google File System [3]. In systematic codes, information blocks are stored and read with no encoding and decoding processes. This is why, in DSSs, systematic codes are preferred to the non-systematic ones. In this work, we only consider systematic erasure codes [1, 2].

In a DSS, in order to store a stripe of data of size L symbols by an (n, k) systematic linear block code, first, the stripe is partitioned into k data blocks each of size $l = \frac{L}{k}$ symbols. Assume that $x_{i,j}$ is i-th symbol of j-th data block, where $i \in [l]$ and $j \in [k]$. Then, $\mathbf{x}_i = [x_{i,1}, \cdots, x_{i,k}] \in \mathbb{F}_q^{1\times k}$. The coded vector $\mathbf{y}_i = [y_{i,1}, \cdots, y_{i,n}] \in \mathbb{F}_q^{1\times n}$ is generated as $\mathbf{y}_i = \mathbf{x}_i\mathbf{G} = \mathbf{x}_i[\mathbf{I}_k, \mathbf{P}]$, where $\mathbf{G}$ is the generator matrix of the (n, k) systematic linear block code. Then, matrix $\mathbf{Y} \in \mathbb{F}_q^{l\times n}$ is constructed by stacking l encoded vectors $\mathbf{y}_i$. Each column of $\mathbf{Y}$ is an encoded block which is stored in a data node. Without loss of generality and for simplicity, from now on, we assume that $l = 1$. Therefore, the terms block and symbol can be used interchangeably.

Distributed storage systems that use systematic erasure codes, are able to reduce the costly storage overhead and adjust the reliability level. The n-replication method has a considerable storage overhead of $n - 1$ which can be moderated to $\frac{n-k}{k}$ by using an (n, k, d) erasure code. The storage overheads of real-world DSSs that use erasure codes such as Facebook HDFS-RAID [2], Microsoft Windows Azure Storage [1], and Google File System [3] are 60%, 33%, and 50%, respectively.

Furthermore, the DSS using an (n, k, d) erasure code is capable of adjusting the level of reliability. Since, an (n, k, d) can tolerate up to $d-1$ symbol erasure, by changing the minimum distance d of code, we can improve the fault tolerance level of the DSS. The minimum distance of the erasure codes used in Facebook HDFS-RAID [2], Microsoft Windows Azure Storage [1], and Google File System [3] are 5, 4, and 4, respectively.

Although, erasure codes decrease the costly storage overhead, they introduce new shortcomings. In the following, we review some of these shortcomings, as well as the solutions proposed in the literature.

One of the shortcomings of erasure codes is their high disk I/O operations required in a block recovery process. Recently, a class of erasure codes have been proposed to reduce this disk I/O overhead [20–22]. There is still no general bound on the minimum number of disk I/O operations required to recover a failed data block [23].

2.4.1 Regenerating Codes (RGCs)

One of the shortcomings of traditional erasure codes is their high repair bandwidth. Repair bandwidth is referred to the required amount of data download from the active data nodes during the recovery process of a single failed data block. For example, in an (n, k, d) MDS code, recovering one single data black requires accessing and downloading k data blocks. To reduce the high repair bandwidth of MDS codes, yet preserve their optimum minimum distance, a class of erasure codes, named regenerating codes (RGCs) have been proposed. A general lower bound on the repair bandwidth of erasure codes was proven in [23]. Erasure codes that achieve this bound and thus, have the minimum repair bandwidth are called RGCs.

2.4.2 Locally Repairable Codes (LRCs)

Another drawback of using conventional erasure codes, is their associated repair locality. Repair locality is defined as the number of active data nodes that must be accessed to recover a single missed data block. Recently, locally repairable codes (LRCs) have been proposed to decrease the repair locality. LRCs is a class of codes offering low repair locality and bandwidth as well as moderate storage overhead. In the following we define the code locality as an important measure of performance in erasure codes.

Definition 2.13. *Code locality: In an (n, k) linear block code, r_i (denoting the locality of the i-th encoded block) is defined as the minimum number of other blocks needed for recovering y_i (i-th encoded block). In other words, in the case that y_i is missing, at least r_i other blocks are required to reconstruct it. Locality of a code, denoted r, is defined as the maximum of r_i for $i \in [n]$, i.e. $r = \max_{i \in [n]} r_i$. LRCs are a class of codes that are designed to have small r.*

LRCs achieve a smaller code locality at the cost of smaller code minimum distance. The following bound on the minimum distance of LRCs was proven in [4, 5].

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2. \tag{2.1}$$

LRCs that achieve this bound with equality are called *optimal*. Using optimal LRCs

in distributed and cloud storage systems leads to high tolerance against node failures, improved storage efficiency, repair bandwidth, and disk I/O. In the following table, we compare the storage overhead and repair bandwidth of three erasure codes.

	Replication method	LRCs	RS codes
Storage overhead	High	Reasonable	Low
Repair bandwidth	Low	Reasonable	High

Table 2.2: Storage overhead and repair bandwidth of three erasure codes.

There are many recent works about LRCs in the literature. An upper bound on the minimum distance of LRCs is established in [24]. Unlike (2.1), this upper bound takes the field order into account. In [7–9], in order to decrease the computational complexity associated with coding, LRCs over small fields are proposed. In [6], a class of LRCs called t-LRCs are introduced. in t-LRCs, for any missing block, there exists t disjoint group of blocks; each group can be used to fully recover the missing block. By generalizing (2.1), an upper bound on the minimum distance of t-LRCs is proposed in [6].

2.4.3 Tanner Graph Representation for LRCs

Tanner graphs can capture the locality of an erasure code. For example, suppose that the Tanner graph of a code is such that every variable node is connected to a check node of degree at most $r + 1$. Then the locality of the code must be at most r. To construct and analyze LRCs, we work on their Tanner graphs, and use several terms that we define next.

Definition 2.14. *Information and parity nodes: Consider the Tanner graph associated with a systematic (n, k) linear block code. Among all the n variable nodes, the k variable nodes that correspond to the k information blocks are called information nodes, and are represented by white (unshaded) circles in the Tanner graph. The remaining $n - k$ variable nodes correspond with the $n - k$ parity blocks. We call these variable nodes*

parity nodes, and represent them by shaded circles in the Tanner graph (see Fig. 2.2 as an example).

Definition 2.15. *Local and global check nodes: In the Tanner graph of an (n, k, d, r) LRC, among $n - k$ check nodes, a minimal set of check nodes, where each having at most $r + 1$ edges to achieve the code locality, that cover all variable nodes are called local check nodes. Check nodes which are not local are called global. For example, in Fig. 2.2, the check nodes that are below the variable nodes are local check nodes and the rest are global.*

Definition 2.16. *Local and global parity nodes: Parity nodes associated with local check nodes are called local parity nodes; other parity nodes (i.e., those associate with global check nodes) are called global parity nodes.*

Definition 2.17. *Local and mixed group: In the Tanner graph of an (n, k, d, r) LRC, variable nodes connected to a local check node constitute a local group. Therefore, a failed variable node can be reconstructed within its local group. Note that the locality r_i of each variable node is the size of its local group minus one. Local groups containing global parity nodes as well as local parity nodes are called mixed groups.*

To generate an LRC, one can first construct a Tanner graph. The Tanner graph will determine the zero elements of the parity check matrix of the code. The non-zero elements can then be chosen randomly from non-zero elements of a finite field. If the order of the finite field used is large enough, the correspoinding constructed code will have the optimal minimum distance, with high probability. After constructing the Tanner graph of an (n, k, d, r) LRC, the zero and non-zero elements of the parity check matrix are determined $\mathbf{H} \in \mathbb{F}_q^{(n-k)\times n}$. Consequently, in this work, we mainly focus on the Tanner graphs of LRCs. The following example illustrate how an LRC can be constructed given a Tanner graph.

Example 2.8. *Fig. 2.2 illustrates the Tanner graph of an $(n, k, d, r) = (12, 7, 5, 4)$ LRC. In this Tanner graph there are 5 squares and 12 circles representing check nodes and variable nodes, respectively. Among all the* 12 *variable nodes in the Tanner*

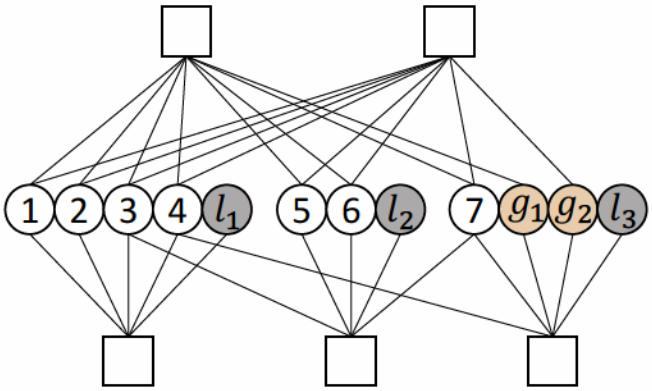


Figure 2.2: Tanner graph of an optimal $(n, k, d, r) = (12, 7, 5, 4)$ LRC

graph, the 7 *unshaded circles represent the information nodes and the* 5 *colored circles represent parity nodes. Also, the* 3 *parity nodes that are distinguished by gray color are local parity nodes and others are global parity nodes. The three squares drawn bellow the variable nodes (circles) are local check nodes as their degree is at most* $r + 1 = 5$*, and they cover all the variable nodes (that is, every variable node is connected to at least one of them). The other* 2 *check nodes (the top two squares) are global check nodes linked to all information nodes. The parity check matrix* $\mathbf{H} \in \mathbb{F}_q^{5\times 12}$*, corresponding to Tanner graph in Fig. 2.2, is*

$$\mathbf{H}_0 = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} & 0 & 0 & 0 & h_{1,8} & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{2,3} & 0 & h_{2,5} & h_{2,6} & h_{2,7} & 0 & h_{2,9} & 0 & 0 & 0 \\ 0 & 0 & 0 & h_{3,4} & 0 & 0 & h_{3,7} & 0 & 0 & h_{3,10} & h_{3,11} & h_{3,12} \\ h_{4,1} & h_{4,2} & h_{4,3} & h_{4,4} & h_{4,5} & h_{4,6} & h_{4,7} & 0 & 0 & 0 & h_{4,11} & h_{4,12} \\ h_{5,1} & h_{5,2} & h_{5,3} & h_{5,4} & h_{5,5} & h_{5,6} & h_{5,7} & 0 & 0 & 0 & h_{5,11} & h_{5,12} \end{bmatrix}.$$

To make sure that the minimum distance of the LRC corresponding to the parity matrix $\mathbf{H}_0$ *is maximized, the non-zero elements of* $\mathbf{H}_0$*, i.e.,* $h_{i,j}$ *, for* $i \in [1, 5]$ *and* $j \in [1, 12]$*, should be selected randomly from a sufficiently large finite field.*

2.4.4 Using LRCs in Real-World Distributed Storage Systems

Microsoft Windows Azure Storage and Facebook HDFS-RAID are two examples of real-world distributed storage systems that are using LRCs. We will cover the LRC

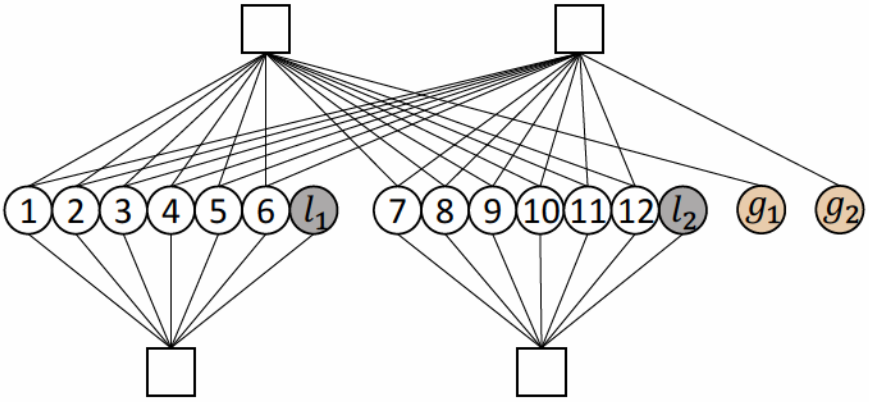


Figure 2.3: Tanner graph of the $(n, k, d, r_i) = (16, 12, 4, 6)$ LRC employed in Microsoft Windows Azure Storage [1]

used in Facebook HDFS-RAID in details in Chapter 3, and show how we can improve its update complexity. The LRC used in Windows Azure Storage is explained in the next example.

Example 2.9. *Microsoft Windows Azure Storage is configured by an* $(n, k, d, r_i) = (16, 12, 4, 6)$ *LRC, where* r_i *denotes the locality of information nodes. This LRC has the storage overhead* $\frac{16-12}{12} = 33\%$ *and has been used since 2012. Fig. 2.3 illustrates the Tanner graph of this LRC, where all the 12 information nodes have locality 6. The local parity nodes are* l_1 *and* l_2*. The global parity nodes* g_1 *and* g_2 *are linear function of all information nodes. In this code, the locality of 14 variable nodes including 12 information nodes and two local parity nodes is 6 and locality of two global parity nodes is 12.*

Chapter 3

Minimizing the Update Complexity of Facebook HDFS-RAID Locally Repairable Code

In this chapter we discuss the impact of improving update complexity in energy saving. In particular, we propose some results about the update complexity of Facebook HDFS-RAID LRC. Before employing the $(n, k, d, r) = (16, 10, 5, 5)$ LRC in Facebook HDFS-RAID, in 2011 an $(n, k, d) = (14, 10, 5)$ RS code was used. Although, the mentioned LRC increased the storage overhead from 40% to 60%, it reduced the code locality considerably.

Data, hence parity blocks, are frequently updated in many applications. Hence, codes with efficient update complexity are desirable as they require fewer I/O operations. This also results in energy saving at data centers [25]. Moreover, a code with minimum update complexity requires less time to update all parity blocks and thus can complete the operation faster. Also, since the power usage of hard-drives in idle mode is less than read/write mode [26], improving the update efficiency of LRCs result in lower power consumption by data nodes.

3.1 Update Complexity Of Facebook HDFS-RAID LRC

In this section, we study the update complexity of Facebook HDFS-RAID LRC. Then, we propose a new LRC with the same parameters (n, k, d, r) as the Facebook HDFS-RAID LRC, but with minimum possible update complexity. We start with defining the update complexity.

In [13], [14] and [15], update complexity is defined as the maximum number of blocks needed to be updated when any single information block is changed. Thus, update complexity for a code C with generator matrix G is equal to the maximum weight of the rows of G, where the weight of each row is the number of nonzero elements of that row. This definition can be extended to the case where multiple information blocks are changed. Let w_i denote the set of blocks that need to be updated when i-th, $i \in [k]$, information node is changed. For a set $S \subseteq [k]$, we define

$$W_S = \bigcup_{i \in S} w_i \tag{3.1}$$

and

$$u_x = \mathbf{E}[|W_S|],$$

where $\mathbf{E}[\cdot]$ denotes expected value, expectation is taken over all subsets of $[k]$ of size x, and information block changes are assumed i.i.d. The parameter u_x can be thought of as the average number of nodes that need to be updated when x information blocks are changed. In the following, we improve u_x for $x = 1$. The reason that we focus on this special case is that i) considering (4.1), reducing u_1 can lead to smaller u_x for $x > 1$; ii) u_1 is the dominant term in update complexity when updates are not frequent (i.e., when multiple block updates in a single stripe is unlikely). By the above definition, u_1 in (n, k) code with generator matrix G, where $\mathbf{G} = [\mathbf{I}_k, \mathbf{P}_{k\times(n-k)}]$, is equivalent to the average weight of rows of matrix $\mathbf{P}$ plus one.

By the definition of code's minimum distance, changing an information block leads to changing at least d encoded blocks including $d-1$ parity blocks and one information block, thus we have

$$u_1 \geq d.$$

The minimum distance of the $(n, k, r, d) = (16, 10, 5, 5)$ LRC used in Facebook HDFS-RAID is 5. By the above inequality, the minimum average update complexity that can be achieved in a code with minimum distance $d = 5$ is $u_1 = 5$. The update complexity of the Facebook HDFS-RAID LRC is $u_1 = 6$, as will be shown later. Our main contribution is designing an $(n, k, r, d) = (16, 10, 5, 5)$ LRC with the minimum possible update complexity of $u_1 = 5$. We remark that, as proven in [2,27], the largest possible minimum distance for the code parameters $(n, k, r) = (16, 10, 5)$ is $d = 5$, which is achieved by both our proposed LRC, and the LRC used in Facebook HDFS-RAID.

In order to find an $(n, k, r, d) = (16, 10, 5, 5)$ LRC with minimum update complexity $u_1 = 5$, we need to construct a generator matrix in which the wight of every row is equal to $d = 5$. Currently, the generator matrix $\mathbf{G}$ associated with the $(16, 10, 5, 5)$ LRC used in Facebook HDFS-RAID [2] is $\mathbf{G} = [\mathbf{I}_{10}, \mathbf{P}_{10\times 16}]$, where

$$\mathbf{P}_{10\times 16} = \begin{bmatrix} p_{11} & 0 & p_{13} & p_{14} & p_{15} & p_{16} \\ p_{21} & 0 & p_{23} & p_{24} & p_{25} & p_{26} \\ p_{31} & 0 & p_{33} & p_{34} & p_{35} & p_{36} \\ p_{41} & 0 & p_{43} & p_{44} & p_{45} & p_{46} \\ p_{51} & 0 & p_{53} & p_{54} & p_{55} & p_{56} \\ 0 & p_{62} & p_{63} & p_{64} & p_{65} & p_{66} \\ 0 & p_{72} & p_{73} & p_{74} & p_{75} & p_{76} \\ 0 & p_{82} & p_{83} & p_{84} & p_{85} & p_{86} \\ 0 & p_{92} & p_{93} & p_{94} & p_{95} & p_{96} \\ 0 & p_{102} & p_{103} & p_{104} & p_{105} & p_{106} \end{bmatrix},$$

and $p_{ij} \in \mathbb{F}_{256}\backslash\{0\}$. The weight of each row of $\mathbf{G}$ is 6, hence the code's update complexity is 6, which is one more than the minimum possible update complexity for a code with minimum distance $d = 5$. The Tanner graph associated with matrix G is also shown in Fig. 3.1.

Instead of working directly with a generator matrix and minimizing the weight of its rows, in our approach, we first construct a Tanner graph $\mathcal{T}$ for the given code

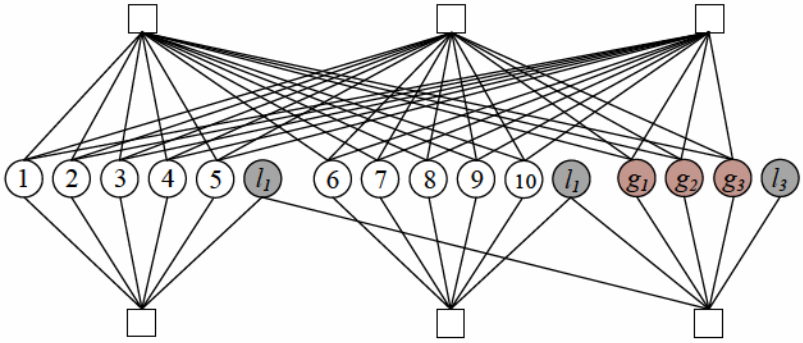


Figure 3.1: Tanner graph of the $(n, k, d, r) = (16, 10, 5, 5)$ LRC used in Facebook HDFS-RAID.

parameters $(n, k, r, d) = (16, 10, 5, 5)$, and then derive a generator matrix from it. While constructing a Tanner graph, we have to ensure the minimum distance constraint is satisfied; for that we use the following theorem from [28].

Theorem 3.1. *[28] There is an erasure code with minimum distance d associated with Tanner graph $\mathcal{T}$ iff every γ check node of $\mathcal{T}$ cover $\gamma + k$ variable nodes, where $\gamma \in [n - k - d + 2, n - k]$.*

Proof. Please see the proof in [28]. □

By Theorem 3.1, in the Tanner graph $\mathcal{T}$ for an $(n, k, d, r) = (16, 10, 5, 5)$ LRC, a necessary condition to satisfy minimum distance $d = 5$ is that any collection of $\gamma = 3$ check nodes including two local check nodes and one global check node cover at least $\gamma + k = 3 + 10 = 13$ variable nodes. To satisfy this condition, the single global check node in any such collections must be connected to all the variable nodes of the local group outside the collection with at most $d - 2 = 3$ exceptions. In other words, at least 3 variable nodes in each local group have to be connected to each of the 3 global check nodes. To have a Tanner graph achieving $u_1 = 5$, after constructing 3 local groups by 3 local check nodes we connect each global check to exactly $r - d + 3$ information nodes with smallest degree in each local groups. Then, if there is any information nodes not connected to at least 2 global check nodes, we connect it to another global check node

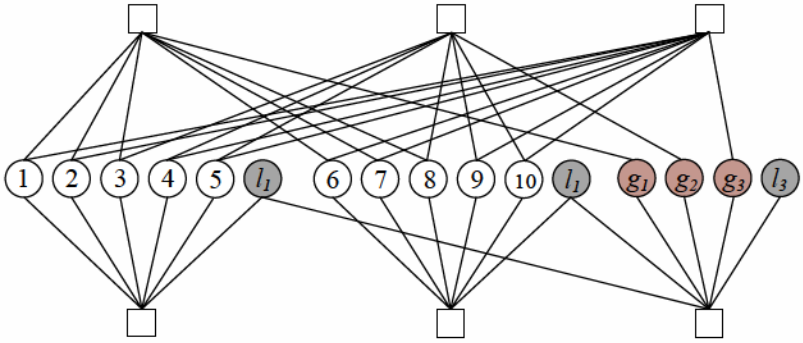


Figure 3.2: Tanner graph of our $(n, k, d, r) = (16, 10, 5, 5)$ LRC with minimum possible u_1 equal to $d = 5$.

to have connection with at least 2 global check nodes. Consequently, each variable node will have connection with 2 global and 2 local check nodes and thus, weight of each row in parity-check matrix $\mathbf{P}$ will be 4. By following these steps we derive a Tanner graph with u_1 equal to $d = 5$. The constructed Tanner graph is shown in Fig. 3.2. It is easy to verify that when this necessary condition is satisfied, then every γ check node, $\gamma \in [3, 6]$, covers at least $\gamma + 10$ variable nodes. Hence, using Theorem 1, minimum distance $d = 5$ is guaranteed.

After constructing the Tanner graph, in order to have a code, we need an explicit generator matrix, or equivalently a parity-check matrix. As mentioned earlier, a Tanner graph determines the zero elements of the parity check matrix H. To find the nonzero elements of H, one can pick random numbers from a sufficiently large finite field. Finding the smallest finite field order needed for explicit code construction can be a very difficult problem [29]. Here, we were able to find a proper generator matrix over $GF(2^4)$. The generator matrix of an $(n, k, d, r) = (16, 10, 5, 5)$ LRC found by the

program is: $\mathbf{G} = [\mathbf{I}_{10}, \mathbf{P}]$ where

$$\mathbf{P} = \begin{bmatrix} 13 & 0 & 10 & 0 & 7 & 10 \\ 10 & 0 & 12 & 0 & 14 & 3 \\ 15 & 0 & 7 & 6 & 0 & 5 \\ 5 & 0 & 0 & 10 & 5 & 12 \\ 4 & 0 & 0 & 12 & 5 & 10 \\ 0 & 15 & 3 & 0 & 7 & 11 \\ 0 & 12 & 11 & 0 & 7 & 3 \\ 0 & 5 & 13 & 9 & 0 & 2 \\ 0 & 10 & 0 & 15 & 8 & 14 \\ 0 & 15 & 0 & 12 & 5 & 11 \end{bmatrix}.$$

We remark that, the elements of the generator matrix of the LRC used in Facebook HDFS-RAID come from $GF(2^8)$. Since the code is based on a $(n, k, d) = (14, 10, 5)$ Reed-Solomon code [2], the smallest finite field order that can be used in the code is $GF(2^4)$, which is the finite field used in our proposed LRC.

3.2 Numerical Results

In this section we provide some comparisons between our proposed $(n, k, d, r) = (16, 10, 5, 5)$ LRC and the one currently used in Facebook HDFS-RAID.

In the following table we compare the u_1 and u_2 for our proposed $(n, k, d, r) = (16, 10, 5, 5)$ LRC and the LRC used in Facebook HDFS-RAID.

Type of code	(n,k,d,r)	u_1	u_2
Facebook HDFS-RAID LRC	(16,10,5,5)	6	7.556
Our proposed LRC	(16,10,5,5)	5	7.267

Table 3.1: Comparison between two different LRCs with identical parameters as the one used in Facebook HDFS-RAID in terms of update complexity.

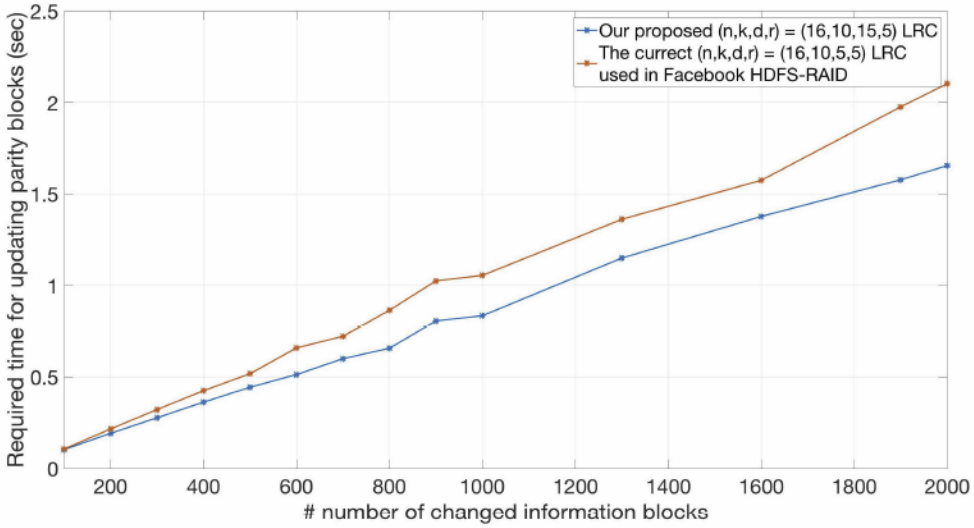


Figure 3.3: The comparison between the required time to update the parity blocks that must be updated when a subset of information blocks are changed, in our $(n, k, d, r) = (16, 10, 5, 5)$ LRC and the one that used in Facebook HDFS-RAID.

Observing the above table, our proposed $(n, k, d, r) = (16, 10, 5, 5)$ LRC improves the update complexity by 16.67% and 3.82% for u_1 and u_2, respectively without sacrificing other code parameters such as code size n, code dimension k, code rate $\frac{k}{n}$, minimum distance d, and locality r.

Fig. 3.3 shows the time required to complete a sequence of single information block change requests for our proposed LRC as well as the LRC used in Facebook HDFS-RAID. The x-axis shows the number of information block change requests in the sequence. Recall that each single information block change requires, on average, updating u_1 blocks. We run this simulation on a system with Intel core i7 6700HQ CPU and 16GB RAM, and set the size of each blocks to 94 bytes. This comparison shows that our proposed code results in more than 30% saving in update time required for a sequence of 1000 information block change requests as well as 94 KB saving in disk I/O and network bandwidth. Please also note that in the clusters used in Facebook HDFS-RAID, the size of each block is 256 MB, which results in a higher gap between the running time performances of the two LRCs.

Chapter 4

The problem of Update Complexity in LRCs

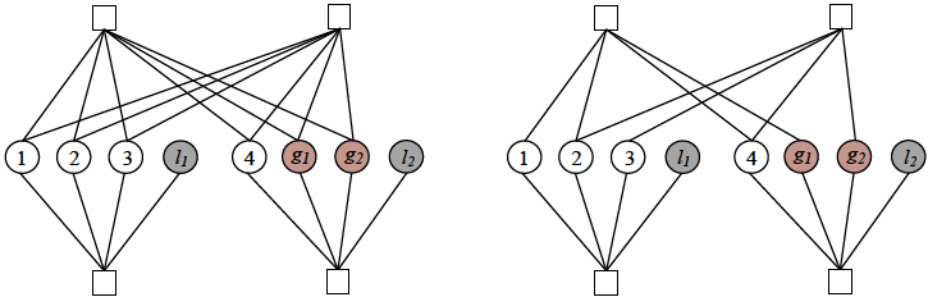
In this chapter, we study the problem of update complexity (UC) for any systematic LRCs. By using the definition of update complexity that we proposed in Chapter 3, in first step we obtain both upper and lower bounds on UC for an important class of LRCs. Then, we propose an algorithm to design LRCs whose average UC is close to the obtained lower bound.

4.1 Bound On the Update Complexity (UC) Of An Important Class Of LRCs

Here, first we repeat the definition of update complexity from Chapter 3. Let w_i denote the set of variable nodes that need to be changed when the ith, $i \in [k]$, information node is updated. Recall that the encoded vector $\mathbf{y}$ is equal to $\mathbf{xG}$. When the ith element of the information vector $\mathbf{x}$ is updated, the jth element of $\mathbf{y}$ is changed iff $\mathbf{G}[i,j]$ (i.e., the jth element of the ith row of $\mathbf{G}$) is non-zero. Therefore, $|w_i|$ is equal to the weight of the i-th row in the generator matrix.

For a set $S \subseteq [k]$, we define

$$W_S = \bigcup_{i \in S} w_i \tag{4.1}$$



(a) (b)

Figure 4.1: Tanner graphs of two optimal $(n, k, d, r) = (8, 4, 4, 3)$ LRCs with different update complexity. When one information block is updated, the codes represented in Figs. 4.1a and 4.1b on average need $(3\times 5+1\times 4)/4 = 4.75$ and $(3\times 4+1\times 5)/4 = 4.25$ encoded block updates, respectively.

and

$$u_x = \mathbf{E}[|W_S|],$$

where $\mathbf{E}[\cdot]$ denotes expected value. Here, the expectation is taken over all subsets of $[k]$ of size x, chosen uniformly at random. The parameter u_x can be thought of as the average number of variable nodes that need to be changed when x information nodes are updated. In the following example, we compute the update complexity of two different LRCs with an identical code parameters and different Tanner graphs.

Example 4.1. *In Fig.4.1, Tanner graphs for two different* $(n, k, d, r) = (8, 4, 4, 3)$ *LRCs are shown. We have*

$$u_1 = \mathbf{E}[|W_S|] = \frac{|w_1| + |w_2| + |w_3| + |w_4|}{4},$$

Therefore, for the LRC shown in Fig.4.1a, we get $u_1 = (3 \times 5 + 1 \times 4)/4 = 4.75$*, while for the LRC shown in Fig.4.1b we have* $u_1 = (3 \times 4 + 1 \times 5)/4 = 4.25$*. Notice that the two codes have identical parameters* (n, k, r, d)*, yet the second code enjoys a lower UC.*

In the following, we define an important class of LRC. First, we study the problem of UC in this class, and later we extend our results to other LRCs.

Non-overlapped LRCs (NO-LRCs): This is an important class of LRCs which has been the focus of many influential studies (e.g., [10–12,30]). In a NO-LRC, all the local

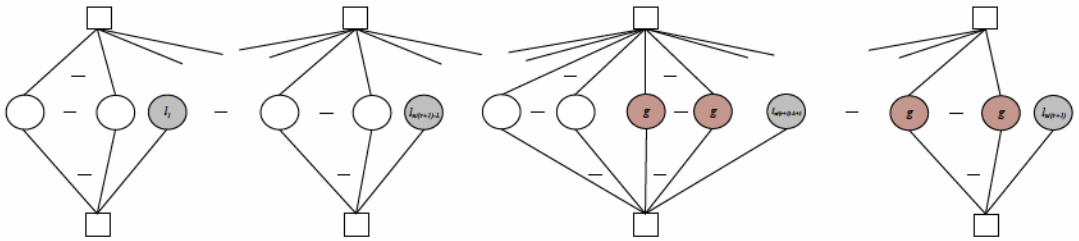


Figure 4.2: Construction of an (n, k, d, r) NO-LRC. There are $m = \lceil \frac{n}{r+1} \rceil$ local and $n-k-m$ global parity nodes, where $r+1$ is cardinality of each local group except the $m-$th local group where its cardinality is $s = n \mod (r+1)$.

groups or non-overlapping. All local groups are of size $r+1$, except at most one local group whose size s, $1 < s < r+1$.

By the above definition, we have

$$s = n \mod (r+1),$$

and among the total $(n-k)$ check nodes, there are $\lceil \frac{n}{r+1} \rceil$ local and $n-k-\lceil \frac{n}{r+1} \rceil$ global check nodes (see Fig. 4.2 as an example).

To study the UC of NO-LRCs and seek NO-LRCs that have low UC, we first start with studying a special variable node arrangement for NO-LRCs. Later, we will show how this arrangement helps us to establish our bounds.

The structure shown in Fig. 4.2 represents an (n, k, d, r) NO-LRCs with a special variable node arrangement. In this structure, based on the construction of NO-LRCs, n encoded nodes are partitioned into $\lceil \frac{n}{r+1} \rceil - 1$ local groups of size $r+1$ and one local group of size s ($s = n \mod (r+1) \neq 1$) called the *defective* group. Each local group has a local parity node, denoted by l_i, where $i \in [\lceil \frac{n}{r+1} \rceil]$. Also, each local group except mixed groups and the defective local group has r information nodes. Among all structures representing an (n, k, d, r) NO-LRCs, the above structure minimizes the number of mixed groups, which is the first step towards reducing UC. Note that in this structure, there is at most one mixed group containing both information and global parity nodes. We call this group *infomixed group*[1] .

[1]There is no infomixed group, if $r \mid k$.

Let

$$z_1 = n \mod (r+1),$$

$$z_2 = d-2 \mod (r+1)$$

and

$$z_3 = k + \left\lceil \frac{k}{r} \right\rceil.$$

These defined variable are useful in following lemma which is from [27].

Lemma 4.1. *In an (n,k,d,r) NO-LRC where $z_1 < z_3$ or even $z_3 = 0$, the largest minimum distance d is one less than the upper bound (1.1), which is*

$$d = n - k - \left\lceil \frac{k}{r} \right\rceil + 1.$$

On the other hand, in an (n,k,d,r) NO-LRC where $z_1 \geq z_3 \neq 0$, the upper bound (1.1) is achieved and the largest minimum distance d is

$$d = n - k - \left\lceil \frac{k}{r} \right\rceil + 2.$$

Proof. The proof is provided in [27]. □

To establish the bounds on the UC, we need to know the exact number of global and local parity nodes. The number of local parity nodes is $\lceil \frac{n}{r+1} \rceil$. The following lemmas determine the number of global parity nodes in terms of code parameters.

Lemma 4.2. *In an (n,k,d,r) NO-LRC with the largest minimum distance d, where $z_1 < z_3$, we have*

$$n - k - \left\lceil \frac{n}{r+1} \right\rceil = d - 1 - \left\lceil \frac{d-1}{r+1} \right\rceil.$$

Proof.

$$d = n - k - \left\lceil \frac{k}{r} \right\rceil + 1 \leq n - k(1 + \frac{1}{r}) + 1.$$

Hence,

$$d \leq n - k(1 + \frac{1}{r}) + 1 < d + 1$$

and

$$k \leq n(1 - \frac{1}{r+1}) - (d-1)(1 - \frac{1}{r+1}) < \frac{r}{r+1} + k.$$

Therefore,

$$d-1-\frac{d-1}{r+1} \leq n-k-\frac{n}{r+1} < \frac{r}{r+1}+d-1-\frac{d-1}{r+1}.$$

By taking the floor of both sides of the inequality, we get

$$\left\lfloor d-1-\frac{d-1}{r+1} \right\rfloor \leq \left\lfloor n-k-\frac{n}{r+1} \right\rfloor < \left\lfloor \frac{r}{r+1}+d-1-\frac{d-1}{r+1} \right\rfloor.$$

We have

$$\left\lfloor d-1-\frac{d-1}{r+1} \right\rfloor = d-1-\left\lceil \frac{d-1}{r+1} \right\rceil$$

and

$$\left\lfloor \frac{r}{r+1}+d-1-\frac{d-1}{r+1} \right\rfloor < 1+d-1-\left\lceil \frac{d-1}{r+1} \right\rceil.$$

Thus,

$$d-1-\left\lceil \frac{d-1}{r+1} \right\rceil \leq n-k-\left\lceil \frac{n}{r+1} \right\rceil < 1+d-1-\left\lceil \frac{d-1}{r+1} \right\rceil. \tag{4.2}$$

By (4.2) and the fact that $n-k-\lceil \frac{n}{r+1} \rceil$ is an integer, we get

$$n-k-\left\lceil \frac{n}{r+1} \right\rceil = d-1-\left\lceil \frac{d-1}{r+1} \right\rceil.$$

□

Lemma 4.3. *It was verified in [27] that in an (n,k,d,r) LRC with the largest minimum distance, where $z_1 \geq z_3$ or $z_1 = 0$, we have*

$$n-k-\left\lceil \frac{n}{r+1} \right\rceil = d-2-\left\lfloor \frac{d-2}{r+1} \right\rfloor,$$

Proof. The proof is provided in [27]. □

Consequently, based on the obtained results, in an (n,k,d,r) NO-LRC the number of global check nodes, denoted ϕ, is

$$\phi = n-k-\left\lceil \frac{n}{r+1} \right\rceil$$

We can determine the exact value of ϕ based on previous lemmas. The total number of mixed groups, denoted λ, is

$$\lambda = \left\lceil \frac{\phi}{r} \right\rceil.$$

Consequently, the infomixed group, if exists, has $\phi-(\lambda-1)r$ global parity nodes and one local parity node. Thus, there are $r+1-\big(\phi-(\lambda-1)r\big)-1$ information nodes in the infomixed group. The total number of information nodes is

$$k=(\Big\lceil\frac{n}{r+1}\Big\rceil-\lambda-\Big\lceil\frac{z_1}{r+1}\Big\rceil)r+\Big(r-\phi+(\lambda-1)r\Big)+\Big\lceil\frac{z_1}{r+1}\Big\rceil(z_1-1),$$

where the first term is the number of information nodes in non-mixed groups, the second term is that in infomixed group and third term is that in the only defective local group. Now by using the given properties of NO-LRCs, in the following theorem, we establish both lower and upper bounds on the number of parity nodes that need to be changed when a subset S of information nodes are updated.

Theorem 4.1. *For an (n,k,d,r) NO-LRC, let S be an arbitrary subset of $[k]$ with $|S|=x$. Then,*

$$d+\Big\lceil\frac{x-r\lambda+\phi}{r}\Big\rceil\leq|W_S|\leq d+\theta,$$

where $\lambda=\Big\lceil\frac{\phi}{r}\Big\rceil$ and $\theta=\min\Big(x,\lceil\frac{n}{r+1}\rceil-\lambda\Big)$.

Proof. Please see Appendix A.1. □

The following corollary directly follows from Theorem 4.1.

Corollary 4.1. *For an (n,k,d,r) NO-LRC, we have*

$$d+\Big\lceil\frac{x-(r\lambda-\phi)}{r}\Big\rceil\leq u_x\leq d+\theta,$$

where $\lambda=\Big\lceil\frac{\phi}{r}\Big\rceil$, and $\theta=\min\Big(x,\lceil\frac{n}{r+1}\rceil-\lambda\Big)$.

Next, we improve the bound on u_x for $x=1$. The reason that we focus on this special case is that i) considering (4.1), reducing u_1 can lead to smaller u_x for $x>1$; ii) u_1 is the dominant term in UC when updates are not frequent (i.e., when multiple block updates in a single stripe is unlikely); iii) the new bound leads to a design of NO-LRCs with near-optimal/optimal u_1.

Theorem 4.2. *For an (n,k,d,r) NO-LRC, u_1 is bounded as*

$$d+\frac{\eta\Big\lfloor\frac{\lceil\frac{n}{r+1}\rceil-\lambda}{d-1-\phi}\Big\rfloor}{k}\leq u_1\leq d+1,$$

where $\lambda = \left\lceil \frac{\phi}{r} \right\rceil$ *and*

$$\eta = \max\Big(0, r(d-1) - \phi(\phi + r - 1)\Big).$$

Proof. Please see Appendix A.2. □

By Theorem 4.1, we have $d \leq u_1$. This lower bound is is improved by by an extra term in Theorem 4.2. In the following section, we propose a class of NO-LRCs whose u_1 matches or is very close to the new lower bound obtained in Theorem 4.2.

4.2 Construction Of LRCs With Small UC

Here, we present our proposed LRCs using their Tanner graphs. Our proposed LRCs follow the structure of NO-LRCs. They benefit from a small u_1, close or even equal to the lower bound obtained in Theorem 4.2. In other words, in comparison with the existing LRCs, our proposed LRCs require accessing and changing a smaller number of encoded blocks in the case of information block updates.

4.2.1 Construction of Our Proposed Optimal LRCs

In order to construct our proposed LRCs, first, n variable nodes are partitioned into $\lceil \frac{n}{r+1} \rceil$ local groups each containing $r+1$ variable nodes except the defective local group which has z_1 ($z_1 \neq 1$) variable nodes. Hence, there are $\lceil \frac{n}{r+1} \rceil$ local check nodes associated with the $\lceil \frac{n}{r+1} \rceil$ local groups, and each local group constructs one local parity block. The remaining $n-k-(\lceil \frac{n}{r+1} \rceil) = \phi$ check nodes construct ϕ global parity blocks which are placed in the mixed groups (Fig. 4.2).

By Theorem 4.1, we have $d \leq |W_S| \leq d+1$, when $|S| = 1$. This implies that updating a single information block requires updating either d or $d+1$ encoded blocks. Therefore, to minimize u_1, we have to find Tanner graphs with minimum number of information nodes whose update requires changing $d+1$ variable nodes. While constructing such Tanner graphs, we have to ensure the minimum distance constraint is satisfied, and for that we Theorem 3.1.

By Theorem 3.1, a necessary condition to achieve minimum distance d for our proposed LRCs is that any collection of $n-k-d+2$ check nodes consisting of $n-k-d+1$ local check nodes and a single global check node cover at least $n-(d-2)$ variable nodes. The number of local groups outside the selected collection is $\lceil \frac{n}{r+1} \rceil - (n-k-d+1) = d-1-\phi$. To satisfy this condition, the single global check node in any such collections must be connected to all the variable nodes of the local groups outside the collection with at most $d-2$ exceptions. In other words, at least $(r+1)(d-1-\phi)-(d-2)$ variable nodes of any set of $d-1-\phi$ local groups have to be connected to each of the ϕ global check nodes. Algorithm 1, presented next, is based on this idea.

4.2.2 Properties of Our Proposed Optimal LRCs

In the following, we verify some important properties of our proposed LRCs generated by Algorithm 1.

Proposition 4.1. *The (n,k,d,r) NO-LRCs constructed by Algorithm 1 based on the structure of NO-LRCs have minimum distance d equal to $n-k-\lceil \frac{k}{r} \rceil+2$ or $n-k-\lceil \frac{k}{r} \rceil+1$.*

Proof. For an (n,k) erasure code with Tanner graph $\mathcal{T}$, if any φ variable nodes are connected to φ distinct check nodes, where $\varphi \in [d-1]$, then any φ variable nodes can be recovered using equations associated with the distinct check nodes. This implies that the code can recover up to any $d-1$ failures and therefore, it has minimum distance d.

In our proposed LRCs, every information node is connected to at least $((d-2-\lambda)+\lambda)+1 = d-1$ distinct check nodes, where $\lambda = \left\lceil \frac{\phi}{r} \right\rceil$. Also, each of the $n-k$ parity nodes is connected to exactly one distinct check node. Hence, any $d-1$ variable nodes are connected to at least $d-1$ distinct check nodes and our proposed code has minimum distance d. □

Remark 4.1. *Our proposed optimal LRCs improve the UC compared to the existing solutions. The construction of our proposed LRCs ensures that not all information nodes are involved in global check nodes. In fact, it tries to keep the number of information nodes involved in any given global check node small. This means a small*

Algorithm 1 Construction of NO-LRCs with small UC

First, construct local groups based on the structure of a NO-LRC depicted in Fig. 4.2.
• Connect each of the ϕ global check node to a distinct global parity node located in the mixed groups.
• Connect all the information nodes of the infomixed group, if exists, to all the ϕ global check nodes.
for $i \in [\lceil \frac{n}{r+1} \rceil]$, **do**
Choose $\lceil \frac{\eta}{d-1-\phi} \rceil$ information nodes in the i-th local group and connect them to all the ϕ global check nodes. Connect each of the other information nodes in i-th local group to $\phi - 1$ global check nodes that have the smallest degree. ($\eta = \max\Big(0, r(d-1) - \phi(\phi + r - 1)\Big)$)
end

number of global parity blocks need update when information blocks are updated. In the case of one information block update, our construction achieves the minimum average number of parity block updates. Based on the bounds in Theorem 4.2, if $\eta = 0$, then the required update when any single information node is changed is d. Also if $\eta \neq 0$, then $u_1 = d + \alpha$, where since there are more zeros in the generator matrix of our proposed LRC than the LRCs constructed in [11], α in our LRCs is lower.

The following remark identifies a condition under which our proposed codes achieve $u_1 = d$.

Remark 4.2. *Based on Theorem 4.2, in the LRCs that $\eta = 0$ we can achieve $u_1 = d$ which is the smallest possible u_1. This happens when in an NO-LRC we have*

$$r(d-1) \leq \phi(\phi + r - 1),$$

where ϕ is the number of global check nodes or $n - k - \lceil \frac{n}{r+1} \rceil$ in an NO-LRC. For instance, in the NO-LRCs that $d \geq r + 3$ we can easily show that the above inequality can be achieved and consequently, we have $\eta = 0$ resulting $u_1 = d$. Also, in the NO-LRCs that $d < r + 3$ we have $\phi = d - 2$ and consequently in the case that

$$r \leq (d-2)(d-3)$$

is achieved, we have $\eta = 0$ resulting $u_1 = d$.

Remark 4.3. *As discussed in Theorem 4.2, u_1 in NO-LRCs is at most $d+1$. However there is a class of NO-LRCs where u_1 is at most d and consequently, there is no way to remove any edge without sacrificing minimum distance and improve u_1. In these NO-LRCs there are at most $d-1$ parity blocks that must be updated in the case that one information block is changed. Hence, the total number of all ϕ global parity blocks and the λ local parity blocks of the mixed groups and the single local parity block located in the same group as the updated information block have to be $d-1$ and we have*

$$\phi + \lambda + 1 = d - 1,$$

resulting

$$\phi + \left\lceil \frac{\phi}{r} \right\rceil = d - 2.$$

Based on Remark 4.3, we know that in a NO-LRC whenever we have

$$\phi + \lambda + 1 \neq d - 1,$$

we are able to remove some redundant edges between information nodes and global check nodes and consequently reducing u_1.

Note that the only part of Algorithm 1 with super linear complexity is the loop. The number of iterations of the loop is $\lceil \frac{n}{r+1} \rceil$, and the number of operations performed in each iteration is $\mathcal{O}(\phi \cdot r)$. Since $\phi \leq n$, the complexity of the loop is $\mathcal{O}(n^2)$.

In Appendix A.3, we present a generalization of Algorithm 1, which constructs any (n, k, d, r) LRCs for any given structure of local groups.

4.3 Numerical Results

In this section, the update complexity of optimal LRCs designed using our proposed algorithms in previous sections are numerically compared with those of conventional optimal LRCs with the same (n, k, r, d) parameters.

First we study one code in detail. Fig. 4.3 depicts the Tanner graph of an optimal $(n, k, d, r) = (15, 9, 5, 4)$ LRC, which is obtained using Algorithm 1. In this example, we have

$$\alpha = 6, \ \beta(d - 2 - \lambda) = 8.$$

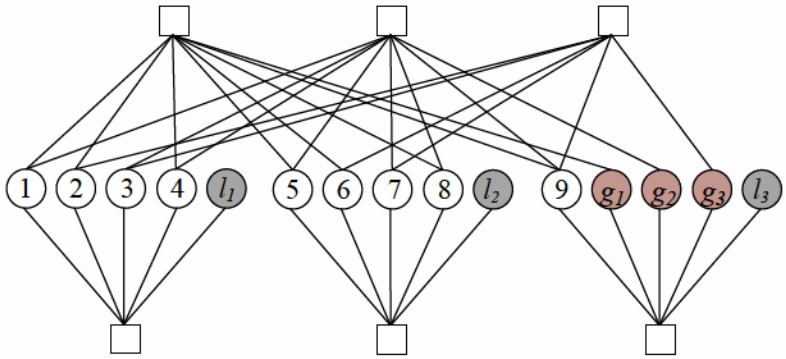


Figure 4.3: Tanner graph of an $(n, k, d, r) = (15, 9, 5, 4)$ optimal LRC. In this figure, the global check nodes are connected to variable nodes based on our proposed method.

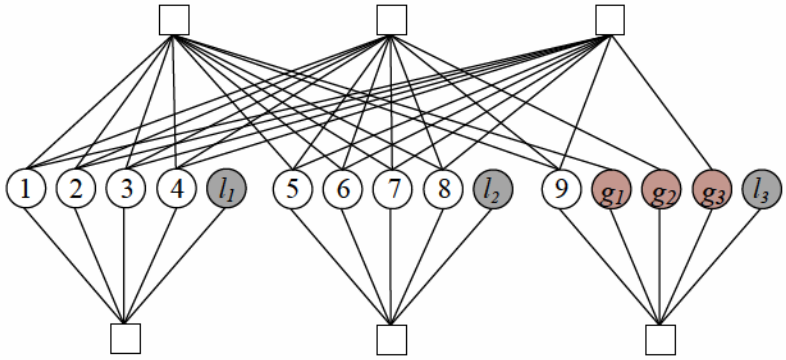


Figure 4.4: Tanner graph of an $(n, k, d, r) = (15, 9, 5, 4)$ optimal LRC. In this figure, each of the global check node are connected to $k + 1 = 10$ variable nodes.

Hence, the condition in Remark 4.2 is satisfied, and all the information nodes are involved in $d = 5$ variable nodes.

Now let us compare u_1 and u_2 of this code (denoted as LRC_1) with u_1 and u_2 of another optimal $(n, k, d, r) = (15, 9, 5, 4)$ LRC (LRC_2) designed using the conventional approach of connecting global check nodes to $k+1$ variable nodes illustrated in Fig. 4.4.

Let $u_x^{LRC_1}$ and $u_x^{LRC_2}$ denote the average update complexity of LRC_1, and LRC_2, respectively. We have

$$u_1^{LRC_1} = \frac{9d}{9} = d = 5$$

and

$$u_1^{LRC_2} = \frac{d + 8(d+1)}{9} = d + 0.89 = 5.89.$$

Furthermore,

$$u_2^{LRC_1} = (26 \times 7 + 10 \times 8) / \binom{9}{2} = 7.27$$

and

$$u_2^{LRC_2} = (20 \times 7 + 16 \times 8) / \binom{9}{2} = 7.44.$$

By Corollary 4.1, we have $u_1 \geq 5$, and $u_2 \geq 7$. Therefore, our constructed optimal LRC (i.e., LRC_1) achieves the bound on u_1 by 15.11% reduction on u_1 of LRC_2. Moreover, our code reduces the gap to bound on u_2 from 0.44 to 0.27 (more than 38%). We would like to emphasize that these improvements are obtained without sacrificing other parameters of LRC_2 such as its rate, dimension, locality and minimum distance and also both of them can be constructed over $GF(2^4)$ which is the suggested field size for them in [11].

In Table 4.1, we compare the update complexity u_1 of some codes designed through Algorithm 1 and the conventional LRCs which connect global check nodes to all information nodes. In this comparison, we consider typical minimum distance of $d = 4$ or $d = 5$, and design codes of length between $n = 12$ and $n = 18$. All these LRCs are NO-LRCs (i.e., have non-overlapped local groups and the largest possible minimum distance). It can be seen here that our proposed codes always improve update complexity u_1.

Note that the output of Algorithm 1 is a Tanner graph, which determines the non-zero elements of the parity-check matrix our LRC. To achieve the promised minimum distance, the non-zero element of the parity-check matrix can be selected randomly from a large enough finite field. In theory, the size of finite field required in our LRCs can be larger than what is used in existing explicit LRC constructions. However, our simulation results show that our LRCs do not require large finite field sizes for a wide range of practical code lengths ($8 \leq n \leq 20$), and minimum distance $d \in \{3, 4, 5\}$. In fact, in our extensive simulation, we could not spot a single case where our code requires a larger finite field size than its counterpart LRC in Tamo and Barg

Parameters of (n, k, d, r) LRCs	u_1 of our proposed LRCs	u_1 of the conventional LRCs	improvement
$(12, 7, 4, 3)$	4.28	4.86	11.93%
$(15, 10, 4, 4)$	4.4	4.8	8.33%
$(16, 10, 4, 3)$	4.3	4.9	12.24%
$(16, 10, 5, 5)$	5	6	16.66%
$(18, 12, 5, 5)$	5	5.83	14.24%
$(18, 13, 4, 5)$	4.46	4.77	6.5%

Table 4.1: Comparison between u_1 of our proposed LRCs and other LRCs for different practical code parameters.

construction. This suggests that, at least for current practical code parameters, our algorithm does not sacrifice finite field size for update complexity. As an example, a generator matrix in $GF(13)$ found in our simulations for $(n, k, d, r) = (12, 7, 4, 3)$ is $\mathbf{G} = [\mathbf{I}_7, \mathbf{P}]$ where

$$\mathbf{P} = \begin{bmatrix} 11 & 0 & 10 & 0 & 9 \\ 11 & 0 & 5 & 3 & 7 \\ 5 & 0 & 0 & 7 & 8 \\ 0 & 5 & 7 & 0 & 3 \\ 0 & 3 & 11 & 12 & 8 \\ 0 & 11 & 0 & 2 & 7 \\ 0 & 0 & 11 & 6 & 7 \end{bmatrix}.$$

For the same parameters (i.e., $(n, k, d, r) = (12, 7, 4, 3)$), the finite field order needed for code construction by Tamo and Barg [11] is at least 13.

Chapter 5

Conclusion

The class of locally repairable codes (LRCs) is an important class of erasure codes to store data efficiently in DSSs. In this thesis, first we studied the update complexity of the $(n, k, d, r) = (16, 10, 5, 5)$ LRC used in Facebook HDFS-RAID. Using the same code parameters, we constructed a new LRC with more than 16% improved update complexity. Then, we established bounds on the update complexity of an important class of LRCs. Furthermore, we proposed a class of LRCs with small update complexity. In fact, our codes could achieve the lower bound on the update complexity associated with one information block update. We identified some of the cases where our proposed codes achieves the lower bound $u_1 = d$ on the update complexity. Considering the recent usage of LRCs in practice, e.g. in Facebook HDFS-RAID and Windows Azure Storage, our results is also of interest from a practical point of view, as it can lead to lower power I/O operations, hence lower power consumption in data centers.

Bibliography

[1] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin *et al.*, "Erasure coding in Windows Azure storage." *Proc. USENIX Annual Technical Conference*, pp. 15–26, 2012.

[2] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," *Proc. VLDB*, vol. 6, no. 5, pp. 325–336, 2013.

[3] D. Ford, F. Labelle, F. Popovici, M. Stokely, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in globally distributed storage systems," *Proc. USENIX Symposium on Operating Systems Design and Implementation*, pp. 61–74, 2010.

[4] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *Information Theory, IEEE Transactions on*, vol. 58, no. 11, pp. 6925–6934, Nov 2012.

[5] D. Papailiopoulos and A. Dimakis, "Locally repairable codes," *Information Theory, IEEE Transactions on*, vol. 60, no. 10, pp. 5843–5855, Oct 2014.

[6] I. Tamo, A. Barg, and A. Frolov, "Bounds on the parameters of locally recoverable codes," *IEEE Transactions on Information Theory*, vol. 62, no. 6, pp. 3070–3083, June 2016.

[7] S. Goparaju and R. Calderbank, "Binary cyclic codes that are locally repairable," in *Information Theory (ISIT), 2014 IEEE International Symposium on*, June 2014, pp. 676–680.

[8] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "An efficient binary locally repairable code for hadoop distributed file system," *Communications Letters, IEEE*, vol. 18, no. 8, pp. 1287–1290, Aug 2014.

[9] T. Ernvall, T. WesterbÃđck, R. Freij-Hollanti, and C. Hollanti, "Constructions and properties of linear locally repairable codes," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1129–1143, March 2016.

[10] N. Silberstein, A. Rawat, O. Koyluoglu, and S. Vishwanath, "Optimal locally repairable codes via rank-metric codes," pp. 1819–1823, July 2013.

[11] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *Information Theory, IEEE Transactions on*, vol. 60, no. 8, pp. 4661–4676, Aug 2014.

[12] I. Tamo, D. Papailiopoulos, and A. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, July 2013, pp. 1814–1818.

[13] N. P. Anthapadmanabhan, E. Soljanin, and S. Vishwanath, "Update-efficient codes for erasure correction," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, Sept 2010, pp. 376–382.

[14] A. Mazumdar, G. W. Wornell, and V. Chandar, "Update efficient codes for error correction," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, July 2012, pp. 1558–1562.

[15] A. Mazumdar, V. Chandar, and G. W. Wornell, "Update-efficiency and local repairability limits for capacity approaching codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 976–988, May 2014.

[16] R. Singleton, "Maximum distanceq-nary codes," *IEEE Transactions on Information Theory*, vol. 10, no. 2, pp. 116–118, April 1964.

[17] R. Tanner, "A recursive approach to low complexity codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, Sep 1981.

[18] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[19] I. S. Reed and G. Solomon, "Polynomial codes over certain finite field," *Journal of The Society for Industrial and Applied Mathematics*, vol. 8, 1960.

[20] K. V. Rashmi, N. B. Shah, and K. Ramchandran, "A piggybacking design framework for read-and download-efficient distributed storage codes," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 331–335.

[21] K. Rashmi, P. Nakkiran, J. Wang, N. B. Shah, and K. Ramchandran, "Having your cake and eating it too: Jointly optimal erasure codes for i/o, storage, and network-bandwidth," in *13th USENIX Conference on File and Storage Technologies (FAST 15)*. Santa Clara, CA: USENIX Association, 2015, pp. 81–94. [Online]. Available: https://www.usenix.org/conference/fast15/technical-sessions/presentation/rashmi

[22] K. V. Rashmi, N. B. Shah, D. Gu, H. Kuang, D. Borthakur, and K. Ramchandran, "A solution to the network challenges of data recovery in erasure-coded distributed storage systems: A study on the facebook warehouse cluster," in *Presented as part of the 5th USENIX Workshop on Hot Topics in Storage and File Systems*. San Jose, CA: USENIX, 2013. [Online]. Available: https://www.usenix.org/conference/hotstorage13/workshop-program/presentation/Rashmi

[23] A. G. Dimakis, 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, Sept 2010.

[24] V. Cadambe and A. Mazumdar, "An upper bound on the size of locally recoverable codes," in *2013 International Symposium on Network Coding (NetCod)*, June 2013, pp. 1–5.

[25] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 732–794, Firstquarter 2016.

[26] J. Zedlewski, S. Sobti, N. Garg, F. Zheng, A. Krishnamurthy, and R. Wang, "Modeling hard-disk power consumption," in *Proceedings of the 2Nd USENIX Conference on File and Storage Technologies*, ser. FAST '03. Berkeley, CA, USA: USENIX Association, 2003, pp. 217–230.

[27] M. Mehrabi and M. Ardakani, "On minimum distance of locally repairable codes," in *IEEE 15th Canadian Workshop on Information Theory (CWIT 2017)*, Quebec city, Canada, Jun. 2017, pp. 36–40.

[28] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "On the average locality of locally repairable codes," *IEEE Transactions on Communications*, vol. PP, no. 99, pp. 1–1, 2017.

[29] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *2013 IEEE International Symposium on Information Theory*, July 2013, pp. 1814–1818.

[30] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "A class of binary locally repairable codes," *Communications, IEEE Transactions on*, accepted for publication, 2016.

Appendices

Appendix A

Proofs for Chapter 4

A.1 Proof of Theorem 4.1

Here, we prove Theorem 4.1. First, we state a lemma to find the minimum number of parity nodes required to be connected to each information node.

Lemma A.1. *In the Tanner graph associated with an (n, k, d, r) NO-LRC shown in Fig. 4.2, any information node of non-infomixed and infomixed groups is connected to at least d and exactly d variable nodes, respectively.*

Proof. The generator matrix associated with the NO-LRC related to Fig. 4.2 is

$$\mathbf{G} = [\mathbf{I}_k, \mathbf{P}] \in \mathbb{F}_q^{k \times n},$$

where $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2] \in \mathbb{F}_q^{k \times (n-k)}$ is the parity matrix generator. Matrix $\mathbf{P}_1 \in \mathbb{F}_q^{k \times (\lceil \frac{n}{r+1} \rceil - \lambda)}$, which generates the local parity nodes of the $\lceil \frac{n}{r+1} \rceil - \lambda$ non-mixed groups, can be presented as

$$\mathbf{P}_1 = \begin{bmatrix} \mathbf{I}_{\lceil \frac{n}{r+1} \rceil - \lambda - 1} \otimes \mathbf{1}_r \\ \mathbf{0}_{(z_1-1) \times (\lceil \frac{n}{r+1} \rceil - \lambda - 1)} \mathbf{1}_{z_1 - 1} \\ \mathbf{0}_{(r\lambda - \phi) \times (\lceil \frac{n}{r+1} \rceil - \lambda)} \end{bmatrix} \in \mathbb{F}_q^{k \times (\lceil \frac{n}{r+1} \rceil - \lambda)}.$$

As well, matrix $\mathbf{P}_2 \in \mathbb{F}_q^{k \times (\phi + \lambda)}$ generates the local parity nodes of λ mixed groups and also the global parity nodes. Observe that the i-th row of $\mathbf{P}_1$ has one non-zero element

for $i \in [k - r\lambda + \phi]$ and the last $r\lambda - \phi$ rows of $\mathbf{P}_1$ are all zero. In order to satisfy the minimum distance constraint, each row of $\mathbf{G}$ must have at least d non-zero elements. Considering the first $k - r\lambda + \phi$ rows of $\mathbf{G}$, each row has two nonzero elements from $\mathbf{I}_k$ and $\mathbf{P}_1$. Thus, the first $k - r\lambda + \phi$ rows of $\mathbf{P}_2$ must have at least $d - 2$ non-zero elements. From the definition of λ and ϕ we also know that $\lambda + \phi = d - 1$. Hence, the first $k - r\lambda + \phi$ rows of $\mathbf{P}_2$ must have at most one zero element. Note that the last $r\lambda - \phi$ rows of $\mathbf{P}_1$ have no non-zero elements. Hence, the last $r\lambda - \phi$ rows of $\mathbf{P}_2$ are non-zero. Note that if $r\lambda - \phi > 0$, i.e., there exists an infomixed group, we have $\phi + \lambda = d - 1$. Thus, $\mathbf{P}_2$ has $d - 1$ columns; and all $d - 1$ elements of the last $r\lambda - \phi$ rows of $\mathbf{P}_2$ are non-zero. □

Proof of Lower bound:

According to Lemma A.1, each row of $\mathbf{G}$ has at least d non-zero elements. Thus, any information node update leads to at least d variable node updates. Regarding Fig. 4.2, $r\lambda - \phi$ information nodes located in the infomixed group are involved in the λ local parity nodes and ϕ global parity nodes. Hence, information nodes associated with the infomixed group require exactly d variable node updates which is the minimum required updates. Now, assume that the number of information nodes to be updated is less than or equal to the number of information nodes associated with the infomixed group, i.e., $x \leq r\lambda - \phi$. Then, assuming all the x information nodes are in the infomixed group, the lower bound on UC associated with x information node updates is d. On the other hand, if the number of information nodes to be updated exceeds the number of information nodes associated with the infomixed group, i.e if $x > r\lambda - \phi$, we assume that $r\lambda - \phi$ information nodes are in the infomixed group and the remaining $x - r\lambda + \phi$ information nodes are in non-mixed groups. Note that if an information node associated with non-mixed groups is updated, then its local parity node has to be updated too. Thus, in this case, the total number of local parity nodes which have to be updated is $\left\lceil \frac{x - r\lambda - \phi}{r} \right\rceil$ and the lower bound on $|W_S|$ is obtained as $|W_S| \geq d + \left\lceil \frac{x - r\lambda - \phi}{r} \right\rceil$.

Proof of Upper bound:

In order to obtain the upper bound on $|W_S|$, we assume that the x information

nodes in S are located in x distinct local groups. By Lemma A.1, any information node update requires at least d variable node updates. Assuming that $x \leq \lceil \frac{n}{r+1} \rceil - \lambda$, among all x information node updates, there exists only one information node update in each local group. Thus, in this case, the number of local parity node updates is exactly x. On the other hand, if $x > \lceil \frac{n}{r+1} \rceil - \lambda$, each of the non-mixed groups has exactly one local parity node update. Hence, the upper bound on $|W_S|$ is obtained as $|W_S| \leq d + \theta$, where $\theta = x$ if $x \leq \lceil \frac{n}{r+1} \rceil - \lambda$ and $\theta = \lceil \frac{n}{r+1} \rceil - \lambda$ otherwise. Observe that for $x > \lceil \frac{n}{r+1} \rceil - \lambda$, $d + \theta = n - k + 1$ which is the amount of all parity nodes plus the changed information node.

A.2 Proof of Theorem 4.2

By Theorem 4.1, we know that updating an information node requires updating either d or $d+1$ parity nodes. In order to follow Theorem 3.1, we need the minimum required connections in any set of $d - 1 - \phi$ non-mixed groups. In the following, first, we state a lemma to find the minimum number of information nodes involved in $d + 1$ variable nodes in any set of $d - 1 - \phi$ non-mixed groups. Then, by using this lemma, we prove the established lower and upper bounds.

Lemma A.2. *Within any collection of $d - 1 - \phi$ non-mixed groups in an (n, k, d, r) NO-LRC (Fig. 4.2), the minimum number of information nodes, denoted η, constructed exactly $d + 1$ variable nodes is*

$$\eta = \max\Big(0, r(d-1) - \phi(\phi + r - 1)\Big).$$

Proof. In an (n, k, d, r) NO-LRC, there are ϕ global check nodes. By Theorem 3.1, a necessary condition for a NO-LRC to achieve the minimum distance d is that any collection of check nodes consisting of $n-k-d+1$ local check nodes and a single global check node cover at least $n - (d-2)$ variable nodes. The local groups that not covered by the selected local check nodes is

$$n - k - \phi - (n - k - d + 1) = d - 1 - \phi.$$

Hence, the selected global check node can have no connection with at most $d-2$ variable nodes in any set of $d-1-\phi$ local groups containing

$$d-2-(d-1-\phi)=\phi-1$$

information nodes and $d-1-\phi$ local parity nodes. Consequently, in order to have the minimum possible connections between global check nodes and information nodes, there must be at least $\phi-1$ information nodes which have no connection with a specific global check node. Hence, in order to satisfy this condition we need at least

$$\phi(\phi-1)$$

information nodes in any set of $d-1-\phi$ local groups. We also know that there are $r(d-1-\phi)$ information nodes in any set of $d-1-\phi$ local groups. Thus, whenever

$$r(d-1-\phi)\leq\phi(\phi-1)$$

is satisfied, every information node is constructing exactly $\phi-1+\lambda+1+1=d$ variable nodes which is minimum possible number. On the other hand, if

$$r(d-1-\phi)>\phi(\phi-1),$$

then we need extra $r(d-1-\phi)-\phi(\phi-1)$ connections to satisfy minimum distance constraint which forces $r(d-1)-\phi(\phi+r-1)$ information nodes to get involved in construction of exactly $d+1$ variable nodes.

Thus, the minimum number of information nodes in any collection of $d-1-\phi$ local groups constructing $d+1$ variable nodes, denoted by η, is

$$\eta=\max\Big(0,r(d-1)-\phi(\phi+r-1)\Big).$$

□

By Lemma A.2, the minimum total number of information nodes constructing exactly $d+1$ variable nodes is equal to $\eta\Big\lfloor\frac{\lceil\frac{n}{r+1}\rceil-\lambda}{d-1-\phi}\Big\rfloor$. Hence, we have

$$\frac{\eta\Big\lfloor\frac{\lceil\frac{n}{r+1}\rceil-\lambda}{d-1-\phi}\Big\rfloor(d+1)+(k-\eta\Big\lfloor\frac{\lceil\frac{n}{r+1}\rceil-\lambda}{d-1-\phi}\Big\rfloor)d}{k}\leq u_1\leq\frac{k(d+1)}{k}.$$

Algorithm 2 Construction of LRCs with small UC for a given local groups

• Choose the minimum set of local groups containing at least ϕ variable nodes except their local parity nodes together to be considered as the mixed groups. Connect each of the ϕ global check node to a distinct global parity node in the mixed groups.

for $i \in [\lceil \frac{n}{r+1} \rceil]$, **do**

  if *There is any information node in i-th local group* **then**

    Connect $\max\left(0, r - \lfloor \frac{\phi(\phi-1+O_i)}{d-1-\phi} \rfloor\right)$ information nodes in i-th local group to all the global check nodes. (O_i is the number of variable nodes in i-th local group shared with other local groups)

    Connect the remaining information nodes to at most $\phi - 1$ global check nodes with the smallest degree.

  end

end

for $j \in [\phi]$, **do**

  if *Any set of* $n-k-d+1$ *local check nodes and j-th global check node do not cover at least* $n-d+2$ *variable nodes* **then**

    Connect j-th global check node to sufficient information nodes with the smallest degree.

  end

end

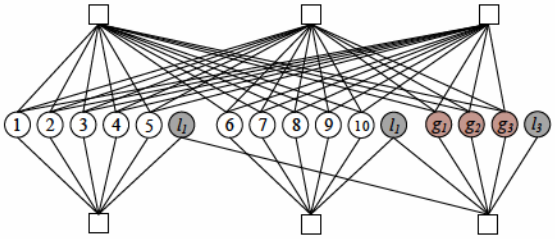
Therefore,

$$d + \frac{\eta\left\lfloor \frac{\lceil \frac{n}{r+1} \rceil - \lambda}{d-1-\phi} \right\rfloor}{k} \leq u_1 \leq d+1.$$

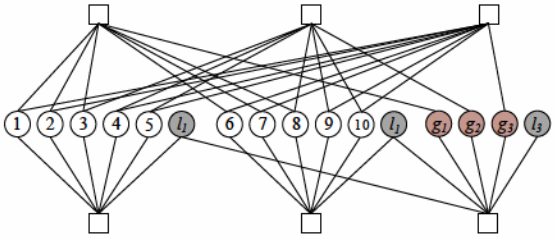
A.3 General form of Algorithm 1

In this section, we propose an algorithm to improve the UC of any LRC with given local groups. This algorithm is the extended version of Algorithm 1 considering the overlaps between local groups. Here, we show the number of shared variable nodes between i-th local group with others by O_i. In the following we propose an example of the LRC with overlaps used in Facebook HDFS-RAID [2] which is an $(n, k, d, r) = (16, 10, 5, 5)$ LRC where $(r+1) \nmid n$.

Example A.1. *Using Algorithm 2, we can improve the UC of the* $(n, k, d, r) = (16, 10, 5, 5)$ *LRC used in Facebook HDFS-RAID [2] from* $u_1 = 6$ *to* $u_1 = 5$. *Fig. A.1 shows the Tanner graphs of this code in two structure. The generator matrix of*



(a)



(b)

Figure A.1: Tanner graphs of two $(n, k, d, r) = (16, 10, 5, 5)$ LRCs used in Facebook HDFS-RAID [2] with different update complexity. u_1 in the codes represented in Figs. A.1a and A.1b is 5 and 6, respectively.

this code found by the program is: $\boldsymbol{G} = [\boldsymbol{I}_{10}, \boldsymbol{P}]$ *where*

$$\boldsymbol{P} = \begin{bmatrix} 13 & 0 & 10 & 0 & 7 & 10 \\ 10 & 0 & 12 & 0 & 14 & 3 \\ 15 & 0 & 7 & 6 & 0 & 5 \\ 5 & 0 & 0 & 10 & 5 & 12 \\ 4 & 0 & 0 & 12 & 5 & 10 \\ 0 & 15 & 3 & 0 & 7 & 11 \\ 0 & 12 & 11 & 0 & 7 & 3 \\ 0 & 5 & 13 & 9 & 0 & 2 \\ 0 & 10 & 0 & 15 & 8 & 14 \\ 0 & 15 & 0 & 12 & 5 & 11 \end{bmatrix}.$$