Dynamic Multiple Node Failure Recovery in Distributed Storage Systems

May Itani[a], Sanaa Sharafeddine[b], Islam ElKabani[a,c]

[a] *Mathematics and Computer Science Department, Faculty of Science, Beirut Arab University Beirut, Lebanon*
[b] *Department of Computer Science and Mathematics School of Arts and Sciences, Lebanese American University Beirut, Lebanon*
[c] *Mathematics and Computer Science Department Faculty of Science, Alexandria University Alexandria, Egypt*

Abstract

Our daily lives are getting more and more dependent on data centers and distributed storage systems in general, whether at the business or at the personal level. With the advent of fog computing, personal mobile devices in a given geographical area may also comprise a very dynamic distributed storage system. These paradigm changes call for the urgent need of devising efficient and reliable failure recovery mechanisms in dynamic scenarios where failures become more likely and nodes join and leave the network more frequently. Redundancy schemes in distributed storage systems have become essential for providing reliability given the fact of frequent node failures. In this work, we address the problem of multiple failure recovery with dynamic scenarios using the fractional repetition code as a redundancy scheme. The fractional repetition (FR) code is a class of regenerating codes that concatenates a maximum distance separable code (MDS) with an inner fractional repetition code where data is split into several blocks then replicated and multiple replicas of each block are stored on various system nodes. We formulate the problem as an integer linear programming problem and extend it to account for three dynamic scenarios of newly arriving blocks, nodes, and variable priority blocks allocation. The contribution of this paper is four-fold: i. we generate an optimized block distribution scheme that minimizes the total system repair cost of all dependent and independent multiple node failure scenarios; ii. we address the practical scenario of having newly arriving blocks and allocate those blocks to existing nodes without any modification to the original on-node block distribution; iii. we consider new-comer nodes and generate an updated optimized block distribution; iv. we consider optimized storage and recovery of blocks with varying priority using variable fractional repetition codes. The four problems are modeled using incidence matrices and solved heuristically. We present a range of results for our proposed algorithms in several scenarios to assess the effectiveness of the solution approaches that are shown to generate results close to optimal.

Keywords: Distributed storage systems, fractional repetition codes, failure recovery, genetic algorithms, multiple failures, heuristic optimization

1. Introduction

Achieving high reliability in distributed storage systems at a reasonable cost can be a challenge due to the fact that they are built from a large number of commodity devices which undergo frequent failures. Real patterns of failures in data centers show that correlated failures are as common as single-node failures in data centers [1, 2]. In a general failure study [1] on Google's main storage infrastructure revealed that 37% of node failures in distributed systems are due to outbreak of at least two nodes. An availability study performed at two large high-performance computing sites showed that around 23,000 different-cause failures were recorded on more than 20 different systems over a period of nine years[3]. Distributed storage system recovery from multiple failures has become critical especially that large business enterprises highly depend on these systems and any interruptions might cause huge losses. Moreover, fog computing is introduced recently to have computing, communication, and storage closer to the user and thus provide better privacy and improved services. This poses, however, major challenges in failure recovery driven by the dynamic and volatile nature of edge devices and especially personal mobile devices.

Email addresses: may.itani@lau.edu.lb (May Itani), sanaa.sharafeddine@lau.edu.lb (Sanaa Sharafeddine), islam.kabani@bau.edu.lb (Islam ElKabani)

Redundancy schemes that are used to tolerate failures and improve reliability, constitute either replicating the data blocks on the system, or reserving additional information to reconstruct the lost data, as in erasure-coded distributed systems [5, 6].

We apply in this paper a family of regenerating erasure codes that provide exact and uncoded repair where a surviving node reads the exact amount of data it needs to send to a replacement node without any processing. These are the Fractional Repetition (FR) codes that provide a low complexity repair process [15]. We propose a solution for multiple node failures by implementing FR codes using a heuristic algorithm based on natural evolution.

Our work in this paper extends on our preliminary work in [7] by focusing on dynamic operations in addressing multiple node failures to account for practical scenarios including: i. optimization of storage allocation under dynamic environments of newly arriving data blocks to the system; ii. optimization of storage allocation with newly added storage nodes; iii. generating an effective block distribution scheme among the nodes by accounting for varying priorities of data blocks. The problems are modeled and solved using the concept of incidence matrices where all available nodes in the network can be selected as helper nodes to any failed node. The proposed heuristic algorithms generate optimized block distribution schemes for the system that minimize total system multiple failure recovery costs, taking into consideration dependent and independent node failures under various dynamic scenarios. Our algorithms also provide a full system recovery plan together with an optimized repair order pattern. The performance of the suggested algorithms is demonstrated through simulation results by varying node and block system parameters. Our main contributions are: i. generating an optimized block distribution scheme for failure recovery under all possibilities of dependent and independent multiple node failures (hereinafter, referred to as Multiple Failure Recovery (MFR) problem); ii. optimizing storage allocation subject to the arrival of new data blocks (hereinafter, referred to as Multiple Failure Recovery with New-comer Blocks (MFR-NB) problem); iii. optimizing storage allocation subject to the addition of new storage nodes (hereinafter, referred to as Multiple Failure Recovery with New-comer Nodes (MFR-NN) problem); iv. generating an optimized block distribution scheme among the nodes by accounting to varying priorities among data blocks (hereinafter, referred to as Multiple Failure Recovery with Variable Priority Files (MFR-VPF) problem).

This paper is organized as follows: overview on regenerating codes is presented in Section 2; related work is reviewed in Section 3; system model and formulation of the main problem together with its three dynamic extensions are presented in Sections 4 and 5 respectively. In Section 6, we discuss solution methodology of the proposed algorithms. Several simulation scenarios are presented and studied in Section 7. Finally, we conclude in Section 8.

2. Overview on Regenerating Codes

In order to cope with failures, data centers introduce redundancy to the system, which may be either in the form of replication or erasure coding. Replication, or repetition, is the simplest scheme where identical copies of the same data object are kept by system nodes. Whereas in erasure coding, each object is divided into k fragments and re-coded into n fragments which are stored separately, where $n>k$. The key property of erasure codes is that the original object can be reconstructed from any k fragments (where the combined size for the k fragments is equal to the original object size) [4].

With n-way replication, the system can tolerate up to n-1 disk failures with $1/n$ storage efficiency, while erasure codes (those of which are Maximum Distance Separable codes) can tolerate the same number of disk failures with a much better storage efficiency. However, during recovery, classic erasure codes provide this optimal storage on the cost of high network and disk usage [8]. The cause of this high network traffic is that regenerating one of the stored blocks requires downloading an amount of information equal to the original data. Advanced erasure codes that optimize repair bandwidth, disk I/O overhead in the repair, and other metrics have been designed to replace classic erasure codes and be applied in distributed storage systems.

Of these advanced erasure codes, we state regenerating codes that form a basis for fractional repetition codes. Regenerating codes provide trade-off points between communication and storage costs. These codes constitute minimum storage regenerating (MSR) codes that minimize the amount of data stored per node, and minimum bandwidth regenerating (MBR) codes that optimize the repair bandwidth[9]. This class of codes is characterized by uncoded repair and growth property where new nodes are added to increase availability of popular data. This offers efficient and low-complexity distributed repair that requires minimum bandwidth, minimum disk reads and no computations. A class of MBR codes, the fractional repetition (FR) code, is designed and suited for large scale distributed systems that are intended to provide ubiquitous and pervasive presence of data for applications that demand high availability and to serve mobile users. The failure of parity nodes in such systems is a frequent event that should be considered along with arrival of new nodes. It should be stated that this code offers exact repair of a failed node [16].

In this work, we apply FR codes and propose a heuristic solution for the problem of multiple failure recovery.

3. Related Work

There are several research publications that address the problem of failure recovery using erasure coding and these can be broadly classified into: i. designing erasure codes based on mathematical models as in [11]-[18]; ii. proposing

recovery approaches that implement such codes as in [23]-[27]. Erasure codes in [11]-[13] are being designed to optimize performance in distributed storage systems in terms of storage space, reliability and recovery cost. Regenerating erasure codes that were introduced earlier in section 2 were proposed in [14] as a new paradigm in coding that achieves minimum bandwidth requirements. In case of a node failure, they require a helper node to read all its data and generate a linear combination of them to send it to the newcomer node. A special class of regenerating codes is the class of FR codes that allow exact uncoded repair, where a helper node reads only one of its stored packets and sends it to the newcomer node without processing. Different constructions of FR codes are presented in literature and these constitute deterministic constructions based on regular graphs and Steiner systems. [15] present FR constructions based on regular graphs for single node failures for all possible feasible system parameters. For the multiple failure case, where the repetition factor is more than two, constructions based on Steiner systems are proposed. Rouayheb et al. also provide a new definition of capacity for distributed storage systems, called Fractional Repetition capacity. Upper bounds on this capacity are also described in [15] while a precise expression remains an open problem. Other constructions presented in [17] utilize combinatorial designs and present explicit heuristic constructions of FR codes. In 2015, Silberstein et al. introduced a new class of FR codes called FR batch (FRB) codes that allow for uncoded efficient exact repairs and load balancing performed by several users in parallel. An FRB code is a combination of an FR code and a uniform combinatorial batch code so it combines the properties of batch codes and FR codes simultaneously. These are the first codes for DSS that allow uncoded efficient exact repairs and load balancing [20]. A randomized construction of FR codes is proposed in [16] where the balls and bins probabilistic model was used as a ground for construction. Those probabilistic constructions provided more flexibility in terms of possible system parameters compared to the deterministic constructions in [15]. We notice that different design considerations were assumed in different code constructions that are in turn based on mathematical models that mostly provide random access repair without optimizing for recovery cost.

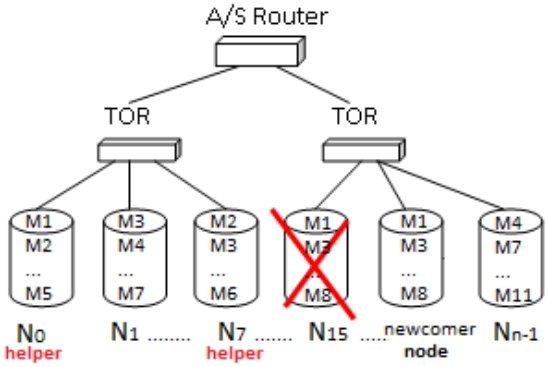
As for the recovery approaches, Xu et al. presented in [23] a recent recovery scheme based on the concept of minimizing the number of disk reads considering only single-disk failure recovery. Authors in [21] formulated the repair problem in a large-scale stream data analysis environment as an optimization problem to select backup sites. The work addressed single and multiple failure recovery where failed applications are recovered automatically in a timely fashion, taking heterogeneity and other factors into account. In [22], the authors formulated a specific repair problem for RAID-6-coded distributed storage systems as an optimization problem using EVENODD codes with balanced disk reads. Of the studies that tackled multiple failure recovery problem, we state that of Kermarrec et al. that addressed simultaneous repair of multiple failures by proposing coordinated regenerating codes that allow devices to control simultaneous repairs. Each of the t failed devices contacts a set of non-failed devices and then coordinates with the other *t-1* failed devices where closed form expressions of optimized quantities of information to be transferred between these devices is derived [26]. Similar to regenerating codes by Dimakis et al. in [14], these codes achieve the optimal tradeoff between storage and repair bandwidth. Li et al. proposed a new design of erasure codes referred to as Beehive, that can tolerate and recover lost blocks from correlated failures in batches while consuming the optimal network transfer with optimized storage overhead [27]. Another study in [24] presented a mutually cooperative recovery mechanism for multi-loss recovery where authors focused on minimizing maintenance bandwidth. Yu in [25] presented a new version of FR codes, irregular FR codes that account for different storage capacities of nodes, different communication costs, and different number of packets per coded block. Yu used hypergraphs in the presented irregular FR code construction where helper nodes were selected from a limited set of given nodes. These codes implemented an advanced version of regenerating erasure codes that achieve minimum bandwidth requirements and good execution time performance.

To the best of our knowledge, the failure recovery approaches proposed in literature did not account for practical scenarios of new block and node arrivals or even the possibility of having data blocks of different priority levels. Our work addresses multiple node failures in these three different dynamic environments. The MFR-NB and MFR-NN problems consider the practical scenarios of having new blocks and/or new nodes. Our work constitutes tolerating new blocks on the current system nodes without varying the optimal allocation of current blocks, and allowing for network extensions via adding nodes and arranging their optimized load configuration again without interrupting the current optimal allocation. The MFR-VPF problem is a new implementation of VFR codes where different repetition degrees are assigned for more popular blocks.

4. System Model

We consider a distributed storage system *(n, k, d)* where n is the total number of storage nodes labeled by N_0,N_1,...,N_{n-1}, the total number of nodes contacted to retrieve a file is denoted by $k<n$, and the number of nodes contacted by a replacement node during node repair is $d \geq k$ [15]. Figure 1a depicts a distributed storage system architecture where multiple storage nodes are connected through top of rack (TOR) switches that are in turn connected to aggregation switches (A/S) via fiber optic cables. TOR switching is a proposed network design in data centers to reduce cabling complexity where network switches

are placed on every rack in the data center and all storage/computing nodes placed on this rack connect to them. Aggregation switches then connect all top of rack switches using few cables. Whenever one or more storage nodes fail, they are being replaced by newcomer node(s) that retrieve the lost blocks from other existing storage nodes in the system that are referred to as helper nodes.



(a) Distributed storage system (an × on a storage node indicates its failure

$$\begin{array}{c} N_0 \\ N_1 \\ \cdot \\ N_7 \\ \cdot \\ N_{15} \\ \cdot \\ \cdot \\ N_{n-1} \end{array} \begin{bmatrix} 0 & c_{01} & \cdots & c_{07} & \cdots & c_{0,15} & \cdots & \cdots & c_{0,n-1} \\ c_{01} & 0 & & & & & & & \\ \cdot & & 0 & & & & & & \\ c_{07} & & & 0 & & & & & \\ \cdot & & & & 0 & & & & \\ c_{0,15} & & & & & 0 & & & \\ \cdot & & & & & & 0 & & \\ \cdot & & & & & & & 0 & \\ c_{0,n-1} & & & & & & & & 0 \end{bmatrix}$$

(b) Communication cost matrix **C**

Figure 1: System model

Different transmission bandwidths and topologies are considered for the underlying networked environment that connects different storage nodes leading to different communication costs among the different nodes. Figure 1b depicts the communication cost matrix **C** that represents the retrieval cost per byte $c_{\alpha\beta}$ from any helper node α in the system to any newcomer node β. $c_{\alpha\beta}$ is evaluated as:

$$c_{\alpha\beta} = \begin{cases} \text{v}, & \text{v} \in R^+ \\ 0, & \text{if } \alpha{=}\beta \end{cases} \quad (1)$$

Let the variable $c_{\alpha\beta,j}$ denote the cost for retrieving block j of size s_j from a helper node α to newcomer node β, where $c_{\alpha\beta,j} = c_{\alpha\beta} \times s_j$. Let the variable f denote the number of failed nodes and γ denote the set of failed nodes. $\gamma_y \in$ Y where Y is the set of all possible sets of combinations of f node failures. Let the redundancy scheme used in the system be an FR code of repetition factor ρ. FR codes consist of a concatenation of an outer maximum distance separable (MDS) code and an inner fractional repetition code.

First, a file of size k packets is encoded using a $(\theta; k)$ MDS code such that any k out of θ packets can recover the whole file(MDS property). Then θ distinct packets are replicated ρ times and distributed on n storage nodes where each coded packet is replicated on distinct nodes. The stored file can be decoded by the user by contacting any k nodes out of the n nodes and this is achieved by the MDS property of the outer code. A new node can be constructed to repair a failed node by contacting a specific set of d nodes for repair depending on the contents of the failed node. d also represents the minimum node storage capacity expressed in packets [15]. Table 1 lists and defines key notations that are further explained in section 5.

Table 1: Key Notation

Parameter	Definition
n	number of nodes in the distributed storage system
k	minimum number of blocks needed to recover a file in MDS code
d	number of nodes contacted for repair of a specific node
θ	original number of blocks to be distributed on the system
ρ	repetition degree of FR code
C	communication cost matrix
B or $B_{i,j}$	block distribution matrix
CC	recovery cost of a specific set of blocks distributed on specific nodes in the NN problem variant
θ_1&θ_2	set of blocks with higher and lower priorities
ρ_1&ρ_2	different repetition factors
α	helper node
β	newcomer node
RC_{opt}	optimized recovery cost
x_{ij}	boolean indicating the presence of block j on node i
$x_{\alpha j}$	boolean indicating the presence of block j on a helper node α
s_j	size of block j
SC	total storage capacity of a node
$c_{\alpha\beta}$	cost per byte of retrieving data from helper node α to newcomer node β
$c_{\alpha\beta,j}$	cost of retrieving block j from helper node α to newcomer node β
$c_{\alpha,j}$	cost of retrieving block j from helper node α to any newcomer node
f	number of failed nodes
γ_y	set of failed nodes
$RC_{\gamma y}$	multiple failure recovery cost for one possible combination of f failed nodes
Y	set of all possible sets of combinations of multiple node failures

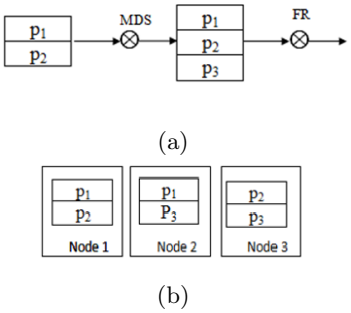


Figure 2: (a) An FR code with repetition degree ρ =2. A file consisting of k=2 packets is encoded using (3,2) MDS code. (b) θ =3 distinct encoded packets are replicated and distributed on n=3 nodes.

Figure 2 shows an example of a two-packet file encoded using (3, 2) MDS code that generates three packets from the two input packets. The three output packets are distributed and replicated on three different nodes. If node 1 fails then it can be recovered by downloading packet 1 from node 3 and packet 2 from node 2 in parallel.

Consider a {0, 1}- incidence matrix B of dimension $n \times \theta$ defined by:

$$B_{i,j} = \begin{cases} 1, & \text{if block j} \in \text{node i} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

that represents a fractional repetition code. Rows represent storage nodes (n) and columns represent blocks (θ). These FR codes will be used in modeling the MFR, MFR-NB, and MFR-NN problems. As for the MFR-VPF problem, a new modification of FR codes inferred from [28] will be used; where higher priority blocks will be associated with higher repetition degrees. The variable fractional repetition (VFR) code is defined using two different repetition factors $\{\rho_1, \rho_2\}$. Note that if $\rho_1 = \rho_2$ then VFR codes are identical to regular FR codes.

To illustrate more the operation of VFR codes, a VFR code for a (6, 2, 3) distributed storage system is depicted in Figure 3. A file of M = 4 packets $(M_1, ..., M_4)$ needs to be stored on the system. After a (7, 4) systematic MDS code, we obtain 3 parity packets P_5, P_6, P_7. Each systematic packet is then repeated 3 times, and each parity packet has a repetition degree of 2; i.e., $\rho_1 = 3, \rho_2 = 2$. When a node fails, regeneration can be preceded by downloading one packet each from a specific set of $d = 3$ nodes.

Assuming different communication link costs, we model and solve the four multiple failure recovery problems under various practical failure scenarios using incidence matrices.

5. Problem Formulation

We start with a numerical example to better illustrate the main multiple failure recovery problem that is depicted in system model section above. Mathematical formulation

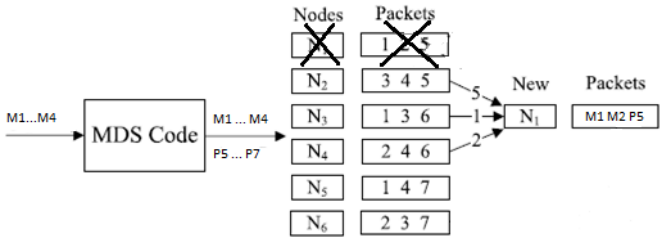


Figure 3: A VFR code with $\rho_1 = 3, \rho_2 = 2$ for a (6, 2, 3) distributed storage system

of the main problem and its three dynamic variants will follow.

5.1. Numerical Example

Consider a distributed system with 6 nodes such that an encoded data object with 4 distinct blocks is to be stored. Assume that the system can tolerate 2 node failures for a repetition factor of 3. Each block will be replicated on three different nodes and each node will store two distinct blocks.

An initial random block assignment matrix B where rows correspond to nodes and columns correspond to blocks together with a given communication cost matrix C, is generated below. The cost matrix associates with every node a generic retrieval cost per byte from another node. For different block sizes, block retrieval costs are calculated accordingly. Link transmission speeds, cabling, I/O disk access and network load are examples of factors that can affect the communication cost between two nodes.

$$B = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$C = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \end{array} \begin{array}{c} \begin{array}{cccccc} n_1 & n_2 & n_3 & n_4 & n_5 & n_6 \end{array} \\ \begin{pmatrix} 0 & 7 & 3 & 5 & 4 & 2 \\ 7 & 0 & 9 & 2 & 3 & 6 \\ 3 & 9 & 0 & 4 & 9 & 8 \\ 5 & 2 & 4 & 0 & 2 & 3 \\ 4 & 3 & 9 & 2 & 0 & 1 \\ 2 & 6 & 8 & 3 & 1 & 0 \end{pmatrix} \end{array}$$

For all possible two-node failure patterns, we first check the cases where nodes have at least one block in common thus we name them dependent. In such cases, the order of recovery affects the failure recovery cost unless the common lost block has its replica available on two helper nodes. In other words, the number of failing nodes f should satisfy the condition $f < \rho/2$ to be possible to recover them in parallel even if they have blocks in common. In Figure 4, we sketch the classification flowchart that determines whether parallel or sequential retrieval can take place.

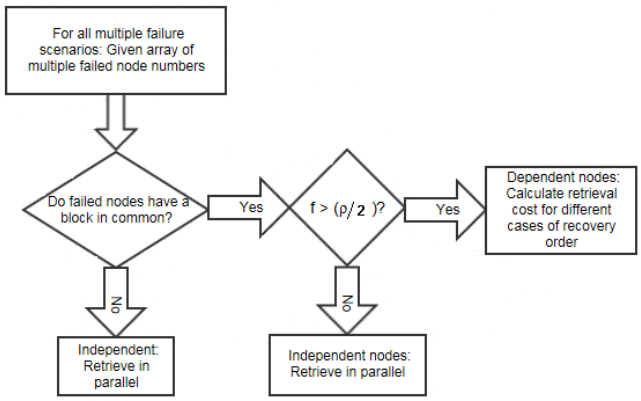


Figure 4: Node failure classification flowchart

Assume in our example that nodes 2 and 4 fail simultaneously. Since they have block 4 in common which cannot be retrieved for both failed nodes in parallel due to the fact that no two helper nodes with replicas of block 4 are available. If both nodes were to use node 6 as a helper node for block 4 retrieval, then the total recovery cost to fully recover both nodes would be $c_{52} + c_{62} + c_{34} + c_{64} = 3+3+4+6 = 16$, assuming blocks of same size. Moreover, if node 2 was to be retrieved before node 4 and later used as a helper node to node 4 then the two-node recovery cost would be 15. Whereas, if node 4 blocks were retrieved before node 2 blocks then the recovery cost for both nodes would be 12. The order of recovery of nodes (ORN) would be then ORN = [4 2].

The total system recovery cost would be the cost of recovering all two-node failures by selecting the optimized recovery pattern order as described above. All independent node failures will be recovered in parallel while dependent failure scenarios would be tested for node recovery order and the order with minimum cost would be selected. For the above block distribution matrix B, the total 2-node failure recovery cost with all failure combinations would be

$$\begin{aligned} RC_{tot} = RC_{12min} + RC_{13min} + RC_{14min} + RC_{15min} \\ + RC_{16min} + RC_{23min} + RC_{24min} + RC_{25min} \\ + RC_{26min} + RC_{34min} + RC_{35min} + RC_{36min} \\ + RC_{45min} + RC_{46min} + RC_{56min} = \\ 10 + 18 + 11 + 9 + 9 + 17 + 12 + 15 + 9 + 18 + 16 + \\ 16 + 10 + 10 + 9 = 189. \end{aligned} \tag{3}$$

However, if we consider the optimized block distribution matrix B_{min} given as:

$$B_{min} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

then the total recovery cost for all 2-node failure scenarios would be 171. Hence, there should be a wise distribution of blocks among nodes so that the system recovery cost would be minimized.

As for the case of new-comer blocks, once a block arrives, it is replicated ρ times and split among the nodes in an optimal way such that the original optimized block distribution matrix and the recovery plan are not changed. The new block locations are added on top of the optimized MFR problem plan as shown next in the re-optimized block assignment matrix B'_{opt}.

$$B'_{opt} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

given new retrieval costs $\mathrm{c}_{\alpha 6}{}^{\mathrm{T}} = \begin{pmatrix} 3 & 1 & 4 & 8 & 9 & 8 \end{pmatrix}$.

Next, we will propose the general problem formulation for the main multiple failure recovery problem followed by the proposed variations that are to be explained subsections 5.3,5.4, and 5.5 respectively.

5.2. Multiple Failure Recovery (MFR) Problem

We need to minimize the total communication cost resulting from retrieving the lost blocks of the failed node from specific helper nodes taking into consideration different combinations of node failures. We assume cost from helper node α to any new comer node β depends only on parameters specific to node α; therefore, in the sequel, we replace $c_{\alpha\beta,j}$ by $c_{\alpha,j}$ to denote cost of retrieving block j from helper node α to any given newcomer node. The retrieval cost for every possible combination is:

$$\mathbf{RC}_{\gamma_y} = \min_{y=1,2,\ldots,(f!+1)} \sum_{\substack{i=1 \\ i\in\gamma}}^{n} \sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\notin\gamma}} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j} \tag{4}$$

The multiple failure recovery cost for the whole system is then

$$\mathbf{RC_{opt}} = \sum_{\forall\gamma_y\in Y} \mathbf{RC}_{\gamma_y} \tag{5}$$

subject to

$$\sum_{i=1}^{n} x_{ij} = \rho \qquad \forall \text{ block j} \tag{6}$$

$$\sum_{j=1}^{\theta} x_{ij} = d \qquad \forall \text{ node i} \tag{7}$$

$$\sum_{j=1}^{\theta} s_j \leq SC \qquad \forall \text{ node i} \tag{8}$$

The total system retrieval cost **RC** in (5) is to be minimized. Thus we are required to solve for a block assignment matrix B that minimizes this value knowing that it constitutes the sum of single block retrieval costs per node taking into consideration all possible orders of retrieval. That is $f!$ permutations of γ for failed nodes in case of dependent nodes, and an additional cost evaluated for the case of parallel retrieval for independent nodes, then we select the plan with minimum cost for each combination. Moreover, we generalize the total cost value to handle all possible multiple node failure scenarios. The helper node α holds a retrieval block j that satisfies $c_{\alpha,j}$ minimal. Note that the helper node α is one of the nodes that have block j in common with the failed node i where a boolean variable $x_{ij} \in \{0,1\}$ is used to indicate that node i holds block j ($x_{\alpha j}$ & $x_{ij} \neq 0$ where i $\neq$ j). We should minimize the objective function value RC subject to the constraints (6) to (8) that the block assignment matrix should satisfy for feasibility. Constraint (6) sets the limit on the total number of replicas per block on all nodes, constraint (7) sets the limit on the total number of blocks that can be stored on any given node, and constraint (8) sets the limit on the total storage size per node to be below a given storage capacity (SC). s_j is the size associated with every block j and **SC** is the total storage capacity of each node in the system.

5.3. Multiple Failure Recovery under New Block (MFR-NB) Arrivals

As for the practical case of new block arrivals, the MFR problem cost is modified to include the new block retrieval costs for new comer blocks and will be evaluated as:

$$\mathbf{RC'_{opt}} = \sum_{\forall \gamma_y \in Y} \mathbf{RC'}_{\gamma_y} \tag{9}$$

where

$$\mathbf{RC'}_{\gamma_y} = \mathbf{RC}_{\gamma_y} + \min_{y=1,2,\ldots,(f!+1)} \sum_{\substack{i=1 \\ i\in\gamma}}^{n} \sum_{j=\theta+1}^{\theta'} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\notin\gamma}} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j} \tag{10}$$

given θ' is an integer such that $\theta' - \theta$ is the number of new-comer blocks.

When a new comer block arrives, it is replicated ρ times and split among the nodes in an optimized way where the previously optimized block distribution matrix and the original recovery plan are not altered. The new block locations are added on top of the optimized MFR problem plan.

5.4. Multiple Failure Recovery under New Node (MFR-NN) Arrivals

As for the practical setting of new nodes entering the system, new rows will be added to the optimized block assignment matrix and specific entries of optimal matrix columns will be modified. Modifications depend on the location of blocks in the newly arriving nodes (represented by rows) together with the corresponding column recovery cost (CC) of specific blocks (represented by columns) generated as follows:

$$\mathbf{CC_{MFR-NN}} = \sum_{\forall \gamma_y \in Y} \mathbf{CC}_{\gamma_y} \tag{11}$$

$$\mathrm{CC}_{\gamma_y} = \min_{y=1,2,\ldots,(f!+1)} \sum_{\substack{i=1 \\ i\in\gamma}}^{n+n'} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\notin\gamma}} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j} \tag{12}$$

where n' is an integer that represents the number of newcomer nodes.

5.5. Multiple Failure Recovery with Variable Priority Files (MFR-VPF)

As for the case of variable priority files, the problem constitutes assigning blocks on nodes with variable repetition degrees based on priority. The objective function will be that of (4). However, constraint (6) is substituted by constraints (13) and (14).

$$\sum_{\substack{i=1 \\ j\in\theta_1}}^{n} x_{ij} = \rho_1 \tag{13}$$

$$\sum_{\substack{i=1 \\ j\in\theta_2}}^{n} x_{ij} = \rho_2 \tag{14}$$

where θ_1 represents the set of blocks with higher priority that should be replicated ρ_1 times, and θ_2 represents the set of blocks with less priority that should be replicated ρ_2 times, such that $\rho_1 > \rho_2$ and $\theta = \theta_1 + \theta_2$.

The four failure recovery problems are integer linear programming problems (ILPs) and cannot be solved optimally in polynomial time for large network sizes since they are NP-hard problems. As an example, if the storage network consists of no more than 10 nodes and let's say it takes one week to get an optimal solution of our ILP (averaged result over 100 runs), then if there are $10 \sim 20$ storage nodes, it roughly needs more than one month to solve our ILP optimally (brute force) $O(2^n)$. This motivates proposing and implementing a heuristic solution that can get an estimation of the minimum system repair cost within few minutes when there are more than 50 storage nodes.

The next section presents the heuristic approach we used to tackle the problems.

6. Genetic Algorithm Based Solution and Implementation

Our implementation methodology constitutes of designing a self-cross-over genetic algorithm that initially distributes blocks on nodes randomly. The algorithm then searches within the feasible solution space by redistributing blocks and generating an optimized distribution scheme [30, 31]. A description of the major GA steps that we used in the main failure recovery problem is shown next.

1. Represent the problem variable domain as a chromosome of a fixed length, choose the size of a chromosome population N, the crossover probability p_c and the mutation probability p_m.
2. Define a fitness function to measure the performance, or fitness, of an individual chromosome in the problem domain. The fitness function establishes the basis for selecting chromosomes that will be mated during reproduction.
3. Randomly generate an initial population of chromosomes of size N: $x_1, x_2, .., x_N$.
4. Calculate the fitness of each individual chromosome: $f(x_1), f(x_2), .., f(x_N)$.
5. Select a pair of chromosomes for mating from the current population. Parent chromosomes are selected with a probability related to their fitness. Highly fit chromosomes have a higher probability of being selected for mating than less fit chromosomes.
6. Create a pair of offspring chromosomes by applying the genetic operators: crossover and mutation.
7. Place the created offspring chromosomes in the new population.
8. Repeat Step 5 until the size of the new chromosome population becomes equal to the size of the initial population, N.
9. Replace the initial (parent) chromosome population with the new (offspring) population.
10. Go to Step 4, and repeat the process until the termination criterion is satisfied.

First, a chromosome is modeled to represent a binary block assignment matrix generated using different permutations and satisfying the repetition factor, blocks per node, and storage constraints (6), (7) and (8) respectively. Thus every chromosome represents a feasible solution. Note that feasible chromosomes are generated by a separate method called by the main GA. An example of a chromosome representation for the values (n=4, d=3, θ=6, ρ=2) is [0 0 1 0 1 1 1 1 0 0 0 1 0 1 0 1 1 0 1 0 1 1 0 0] where the n$\times\theta$ block assignment matrix is transformed to a single (n$\times\theta$, 1) row matrix. Second, an initial population is generated for reproduction after carrying out the chromosome encoding phase. Population size p is pre-specified and the population will include p chromosomes that are generated at random using a constructive method that generates a feasible allocation scheme. Using the above-specified encoding scheme, we can generate feasible block allocation matrices from one another by exchanging rows within a single matrix to ensure that the number of blocks per node constraint together with the storage are constraint not violated. This enforces a self-cross-over scheme in our genetic algorithm. The conventional cross-over technique will yield an in-feasible new chromosome that does not satisfy the FR code constraints (6) and (7) in our model. To help keep up the best chromosome in every generation, we used elitism in our implementation, where the best individuals of the current generation are transferred to the next generation to improve the GA performance. Moreover, we used mutation to maintain diversity in the population together with making sure that the achieved solution is not a local optima, we used mutation. Mutation is a genetic operator that alters one or more gene values in a chromosome from its initial state. Every chromosome or individual is associated with a fitness function equivalent to the optimization problem objective function defined in (4) and calculated using algorithmic steps of Algorithm 1.

In Algorithm 1, we use a nested loop that checks all combinations of node failures based on a given number of failed nodes. For each combination, the algorithm should first check whether the failing nodes are dependent that is they have any blocks in common. If so is the case; then the failure recovery cost should be generated and treated in a different way than that for independent nodes which can be retrieved in parallel. After checking for dependency, the algorithm will call a function that generates permutations to check the best order of retrieving failed nodes if it happened they were dependent. If not, the recovery cost is calculated by implementing a nested loop that mimics the operation of the double summation to evaluate the fitness. Inside the inner loop of the algorithm we check for the failed node and the blocks it holds, then we check all nodes that can be candidates to help in retrieving each block. Out of the candidate nodes for each lost block, we choose the one that minimizes the recovery cost and generate a list of all helper nodes. This list is then to be used by the main algorithm to prepare the recovery plan matrix.

Algorithm 2 accounts for newcomer blocks and generates a post-optimized recovery plan with the best possible distribution of newcomer blocks on the system nodes. First, the optimal chromosome output from the genetic algorithm of the main MFR problem is passed as a two-dimensional array parameter together with its fitness value to the Matlab function implementing Algorithm 2. Then using a loop we start assigning the newcomer blocks and their replicas on nodes by augmenting the optimal block assignment matrix passed as an argument, and then we calculate the fitness by adding on top of the optimized fitness value. After a specified number of generations we then select the chromosome with the best new fitness and update the fitness value together with the previous block assignment matrix and recovery plan.

Algorithm 3 accounts for newcomer nodes and generates a post-optimized recovery plan with the best possible reallocation of on-system blocks on current and new

Algorithm 1 Fitness Function Calculation

function CALCULATEFITNESS ($n \times \theta$ matrix) ▷ where the matrix is a feasible chromosome representing binary block allocation scheme
  for $w = 1$, while $w < \frac{n!f!}{(n-f)!}$, w++ **do**
    Let γ be the set of failed nodes
    Check whether failed nodes are dependent using steps in Figure 4
    if nodes are independent **then**
      go to 9
    else if go to 18 **then**
    end if
    for every block j belongs to failed node i **do**
      Check for all nodes α such that node α has a replica of block j
      Select the node α with minimum retrieval cost $c_{\alpha\beta,j}$
      Assign α as one of the helper nodes for failed node i and save it in recovery plan matrix
      Update node i retrieval cost value to $\sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\notin\gamma}} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j}$
      Update recovery plan to include helper nodes for recovering all blocks j of node i
    end for
    Update total retrieval cost value for the failed nodes in set γ
    Update system recovery plan for next failure scenario
    $\forall$ permutations P(γ_y) of array γ
    Calculate retrieval cost for failed node i using $\sum_{j=1}^{\theta} \min_{\alpha=1,2,\ldots,n\alpha\notin\gamma} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j}$
    Remove i from set of failed nodes γ Repeat till $y = f!$ and update the retrieval cost for the specific permutation that constitutes a recovery order
    From set of all permutations select the one associated with the minimum recovery cost of dependent nodes
    Update total system recovery cost for all cases of multiple node failures
  end for
end function

Algorithm 2 Newcomer Blocks Allocation

function ALLOCATENB(optimized allocation scheme)
  $j = 1$
  while $j <$number of newcomer blocks **do**
    Replicate block j , ρ times
    Distribute replicas of j by augmenting the optimal block assignment matrix that was generated based on best fitness using steps 3 to 10
    for $w = 1$, $w < \frac{n!f!}{(n-f)!}$, w++ **do**
      Let γ be the set of failed nodes
      Check if any of the failed nodes γ are dependent using steps in Figure 4
      Check for all nodes α that have a replica of block j present in failed node
      Select the node α with minimum retrieval cost $c_{\alpha,j}$
      Assign α as one of the helper nodes and update recovery plan matrix and retrieval cost to $\sum_{j=1}^{\theta} \min_{\alpha=1,2,\ldots,n\alpha\notin\gamma} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j}$
      Update recovery plan to recover all blocks j of node i and the total retrieval cost in set γ accordingly
    end for
    Calculate retrieval cost for failed node i using $\sum_{j=1}^{\theta} \min_{\alpha=1,2,\ldots,n\alpha\notin\gamma} c_{\alpha,j} \cdot x_{ij} \cdot x_{\alpha j}$
    Select the chromosome associated with the minimum total recovery cost
    Calculate the fitness of the newly augmented column for different multiple failure scenarios
    Update the value of the optimized fitness
    Select chromosomes with best fitness
    Apply cross-over and mutation operations only on augmented columns of chromosomes with best fitness
    After 20 or more generations, select the best chromosome that has minimal retrieval cost and update the original optimized block allocation scheme together with the optimized recovery plan to account for the newcomers
    j++
  end while
end function

nodes. It is a greedy algorithm that also uses the optimal distribution generated earlier for the main recovery problem. It modifies and extends it to account for new comer nodes and solve the MFR-NN problem. A loop is used to allocate different blocks on the newcomer nodes by augmenting the matrix with rows instead of columns. Then costs are computed column-wise to determine which blocks to extract from current allocation and assign to newcomer nodes based on the minimum recovery cost criteria.

Finally, the algorithm that handles the MFR-VPF problem constitutes of modifying the feasible solution space used in Algorithm 1. The modification is in the function generating feasible binary block assignment matrices that satisfy constraints (13) and (14) instead of constraint (6) to get feasible candidate solutions, and then generating an optimized one.

7. Simulation Results and Analysis

The results of our implementation for different cases of multiple node failures are presented in this section. The machine employed for simulation is a Lenovo laptop with

Algorithm 3 Newcomer Nodes Allocation

function AllocateNN (optimized allocation scheme)
  for $k = 1$, $k <$number of newcomer nodes, k++ **do**
    Distribute d blocks on node k randomly and augment them to the given optimized allocation matrix
    For all blocks j of the d blocks present on the newcomer node k
    Select the corresponding column j for all current nodes
    Check which set of ρ nodes best suits for allocating block j based on column cost value of (12)
    Repeat 4 to 6 for all d blocks present on node k
    Calculate the new fitness value based on the sum of all minimum column costs
    Select chromosomes with best fitness
    Apply cross over and mutation operations only on augmented rows of chromosomes with best fitness
    After a specific number of generations select the optimal chromosome and update the originally optimized block allocation scheme together with the optimal recovery plan to account for the newcomers
  end for
end function

an Intel (R) Core i7 CPU running at 2 GHz with 8 GB RAM. The operating system is Windows 7, and the computer is a 64-bit machine.

The simulation programs were written in MATLAB. For more than 20 storage nodes, simulation was done on an eight-node Linux Beowulf cluster, all running Linux Cent OS 5.5, 3.4 GHz Intel Core i7 processor, and 8 GB RAM. We used simulations to test our proposed solutions on large number of nodes under different set of parameters to explore the feasibility, practicality and efficiency prior to embarking on real test bed implementations. FR codes are modeled as incidence matrices and thus we chose Matlab for our simulations as it is originally designed for matrix operations.

It is noteworthy to state that the heuristic simulation results shown in Figures 8 and 9 are calculated as the average cost of 20 runs for each value of n ; i.e. a total of 200 runs for each figure and then normalized by the maximum average cost in each generation for fair comparison. In the figure we denote "normalized optimized fitness" to represent the optimized fitness cost divided by the value of the maximum average fitness for all generations. "Normalized average fitness" refers to the average cost of each generation divided by the maximum average fitness for all generations.

The first result shown in Figure 5 is for the case of a system with 25 storage nodes and 50 distinct blocks being replicated three times for a total of 150 blocks ($\theta = 50$, $\rho = 3$) and to be distributed on the 25-node system in an optimized scheme for multiple failure recovery. The convergence curve of the optimized repair cost is shown together with the average cost for each generation. This example handles 2-node failure scenarios. The average cost is evaluated over 50 individuals. As observed, mutation occurs in the fifth and again in the fourteenth generation. This explains why the average fitness increased after decreasing. In Figure 6, we consider system parameters of $n = 50$ nodes, $\theta = 100$ blocks and repetition factor $\rho = 3$.

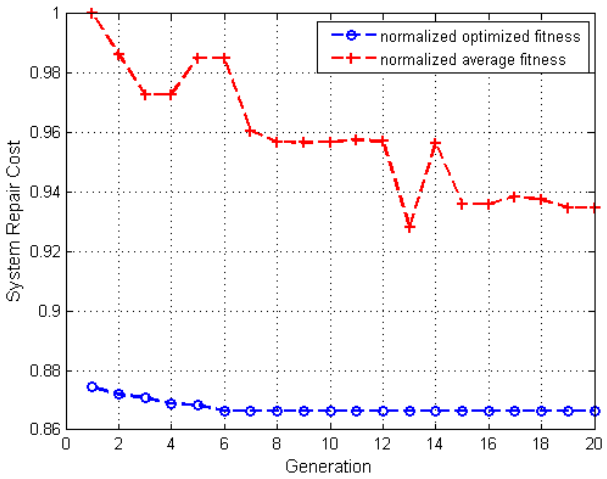


Figure 5: Average and minimum system repair cost for n=25, θ=50, ρ=3

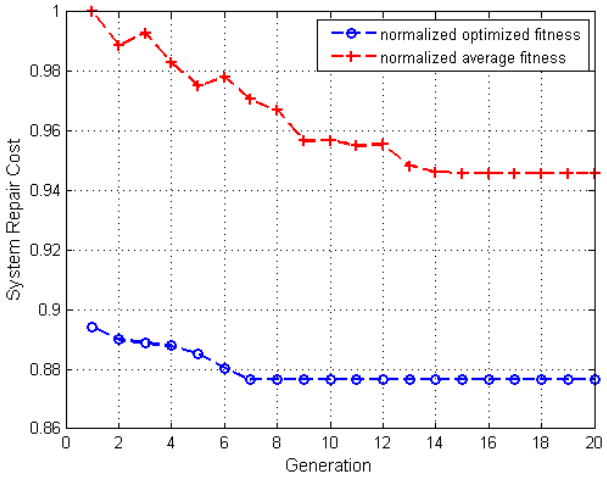


Figure 6: Average and minimum system repair cost for n=50, θ=100, ρ=3

We notice from Figure 5 and Figure 6 that the average fitness evaluated over 50 individuals for each population set decreases in every generation except in the case of a mutation. This illustrates the fact that better solutions are being generated in newer populations. Moreover, we ob-

serve that the optimized system repair cost; i.e. optimized fitness, decreases till it converges to a minimal value.

To avoid being stuck in a local optima, we ty to increase the mutation rate and re-run the algorithm.

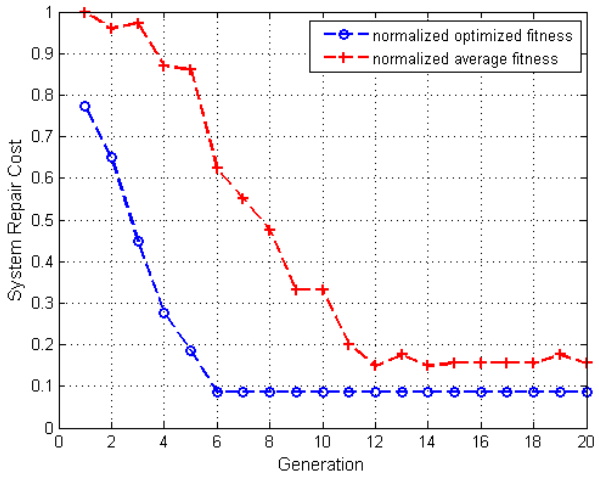


Figure 7: Average and minimum system repair cost for n=25, θ=50, ρ=7

It is noteworthy to state that the optimized solution is achieved as early as the sixth generation for $n = 25$ nodes (Figure 5) and at the seventh generation when n is increased to 50 nodes (Figure 6). That shows the quick convergence of the algorithm even when we increase the number of nodes and blocks. Also in Figure 7 when we consider the case of increasing the number of tolerated failures, early convergence at the sixth generation is still achieved.

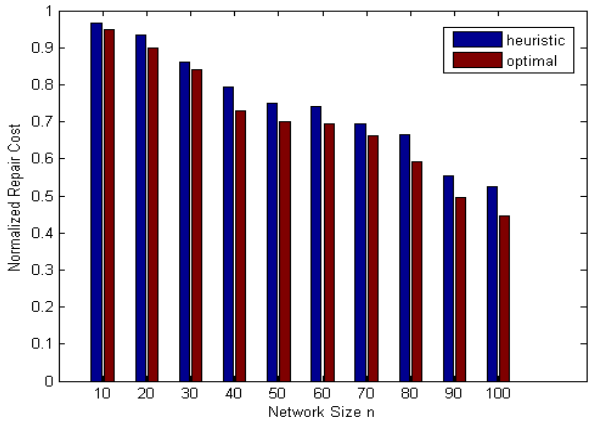


Figure 8: Minimal heuristic versus brute-force post-optimal system repair cost for different network sizes for the MFR-NB problem

We next compare the minimum system repair cost of our heuristic implementation for the MFR-NB and MFR-NN problems respectively to that of the optimal brute force implementation for different network sizes and this is shown in Figure 8 and Figure 9 respectively. The difference between the heuristic and optimal solutions for the tested scenarios is calculated at most as 1% for the MFR-NB problem and 4% for the MFR-NN problem. This indicates that our results are near optimal for the selected parameters.

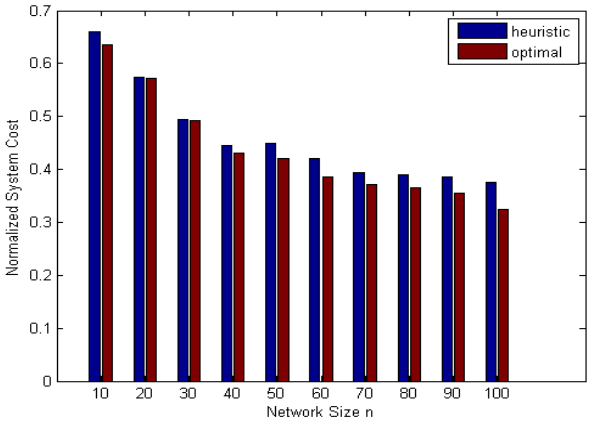


Figure 9: Minimal post-optimal brute-force vs. heuristic system repair cost for different network sizes for the MFR-NN problem

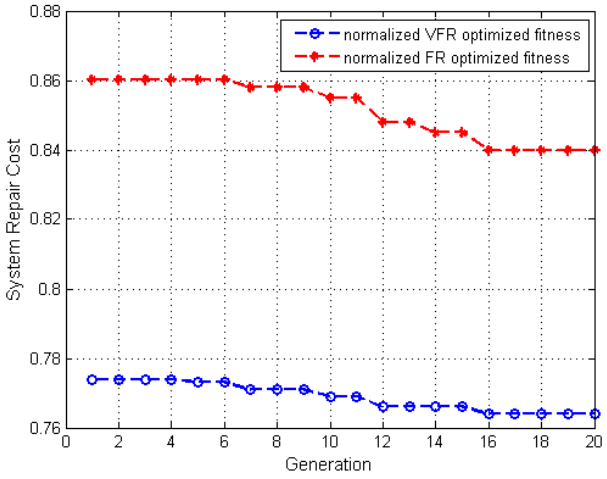


Figure 10: Normalized optimal recovery cost for n=20, θ = 50, FR (ρ = 4), VFR(ρ_1= 4, ρ_2 = 3)

Figure 10 shows simulation results for the recovery problems implementing FR and VFR codes respectively for the same cost matrix. The chosen parameters for simulation were 20 storage nodes and 50 different blocks. For the case of FR code, all blocks were replicated 4 times. Whereas, for the case of VFR code, 20 of the blocks were replicated 4 times and 30 were replicated 3 times. This shows a significant difference in cost where VFR costs can achieve same availability for specific blocks obviously with less costs. Upon recovery, the search space for the candidate helper blocks in the system where VFR code is implemented as a redundancy scheme, proved to include more options for recovery with lower cost than the search space

in a system where FR code is implemented.

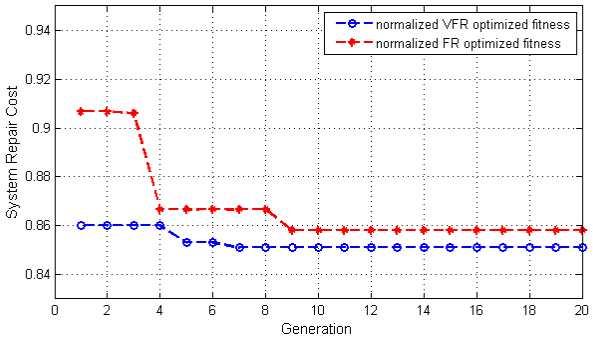


Figure 11: Normalized optimal recovery cost for n=20, $\theta = 125$, FR($\rho = 3$), VFR($\rho_1 = 5$, $\rho_2 = 2$)

As for Figure 11, we vary the parameters of the system, and consider distributing more blocks with different repetition factors. Simulation results show that VFR codes can be efficient in failure recovery cost-wise even when we doubled the number of blocks and increased the repetition factor for blocks with higher priority only.

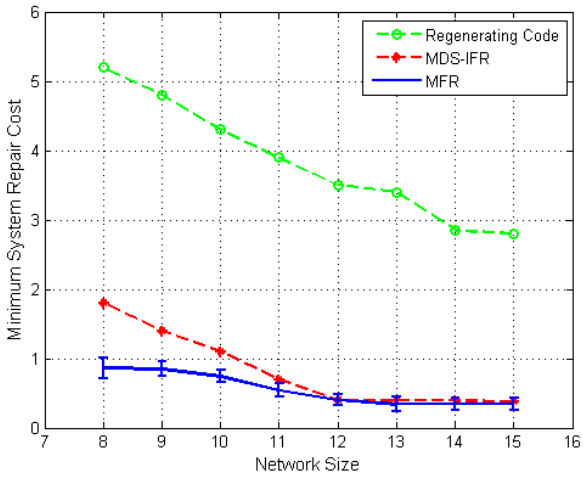


Figure 12: Minimum system recovery cost using different approaches, $\theta = 30$, $\rho = 2$

We next compare the minimum system repair cost using our approach with that of the MDS-IFR code in [25] and that of the regenerating code that can achieve an optimized tradeoff between system storage cost and system repair cost for different network sizes, also presented in [25].

We used a confidence interval of 95% to find the margin of error that was calculated using the formula $Z_{a/2} \times (\frac{\sigma}{n})$, where $Z_{a/2}$ is the confidence coefficient and a is the confidence level, σ is the standard deviation, and n is the sample size. The corresponding value in the z statistic table that matches a confidence interval of 95% and is used happens to be 1.96.

For small storage networks, from Figure 12, it can be seen that minimum system repair cost that can be achieved using our proposed approach is reduced by at least by 20% compared to the simple regenerating code used in [25]. As for MDS-IFR, we are at least 5% better in reduction of system cost but our work goes beyond the basic problem of failure recovery and includes multiple dynamic scenarios that yields this work more practical. We considered incremental changes rather than full re-optimization of the whole system and showed that results stay close to optimal with much less processing.

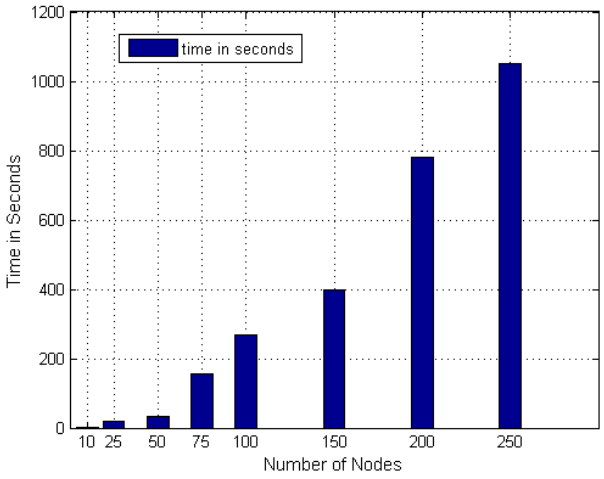


Figure 13: Average time to generate optimized minimum system cost for different number of storage nodes for two node failures

To gain more understanding about the performance regarding the complexity issue, we vary the parameters n, θ and ρ. We then increase the network size and measure its running time for different cases of multiple failure recovery problems. In Figure 13, we show the average time to generate the optimized solutions for the case of two node failures. The elapsed time for generating an optimized solution is around 1.8 seconds for a system with 10 nodes, 10 seconds for 25 nodes, and 34 seconds for 50 nodes. This value goes up to around 13 minutes for distributed systems as large as 200 nodes. In Figure 14, we present the average times needed to generate optimized solutions to account for multiple node failures. It is worth noting that this time reflects the running time to generate an initial configuration of the whole system and this is done only once.

In fact, a general formula that relates complexity to the genetic algorithm parameters evaluates the complexity as

$$\begin{aligned} &\mathbf{P} \times \mathbf{G} \times \mathbf{O}(\mathbf{Fitness}) \\ &\times \{\mathbf{P_c} \times \mathbf{O}(\mathbf{Crossover}) + \mathbf{P_m} \times \mathbf{O}(\mathbf{Mutation})\} \end{aligned} \tag{15}$$

where $\mathbf{P}$ is the population size; $\mathbf{G}$ is the number of generations; $\mathbf{P_c}$ is the crossover probability; and $\mathbf{P_m}$ is the mutation probability.

For two node failures, the fitness function is roughly of

order $O(\frac{n^2 d}{\rho})$ and for a larger number of failures in the recovery problem and its variants, the complexity will be of higher order due to the fact that we are considering different combinations of node failures. However, the main algorithm will be run only once for the initial configuration and then all dynamic scenarios are updated on the first run basis avoiding high complexity.

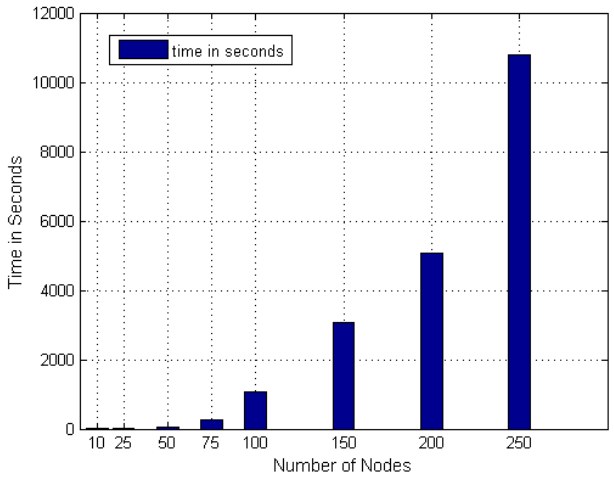


Figure 14: Average time to generate optimized minimum system cost for different number of storage nodes in multiple failure scenario with $\rho > 3$

Eventhough the complexity maybe considered theoretically high, if the configuration was to be done only in the preprocessing phase that would take place in a reasonable time. As the aim is to recover from failures, we will consider further enhancement to achieve better complexity.

8. Conclusions and Future Work

In this work, we addressed the problem of multiple failure recovery by designing and implementing an algorithmic approach that allocates blocks on nodes in a way to minimize the total system repair cost in case of multiple node failures. Moreover, we considered very common and dynamic scenarios that constitute the arrival of new blocks, new nodes, and variable priority files in a distributed storage system. Our solution used FR codes to provide a simple repair mechanism that minimizes the repair and communication costs. The four problems were formulated using incidence matrices and solved heuristically using genetic algorithms for all cases of multiple node failures. For the main problem with different system parameters considered, simulation results achieved fast convergence. As for the three dynamic settings, we demonstrated the effectiveness of the proposed mechanism by performing several runs considering different parameters and the obtained experimental results were shown to be close to optimal.

The novelty of the proposed solution approaches is in its ease of practical implementation where post-optimal storage allocation is achieved without the need to redistribute actual residing blocks. The work in this research holds several branches for further study. A first possible option is to extend the theoretical framework and perform testbed implementation. In fact demands cannot be predicted theoretically, while in practice they are accessible to some certain extent. Other benefits of experimenting large-scale tests of the prototype system is that it can capture many practical aspects including real network bandwidth fluctuation and transmission delay, real cache server performance characteristics such as disk read/write speed and other issues including control plane overhead, error handling and etc. A full scale test of the system helps understand both the potential and limitations of the theoretical model and design schemes under various scenarios. A second possible extension is to try to enhance the model to include privacy and security issues on top of repair bandwidth, reliability, availability, scalability and computational complexity. Finally, a major and significant extension of this work that we are currently addressing is to customize the proposed models and mechanisms to apply to fog scenarios including personal mobile devices collectively acting as a small distributed storage system.

References

[1] D. Ford, F. Labelle, F.I. Popovici, M. Stokely, V.A. Truong, L. Barroso, C. Grimes, and S. Quinlan "Availability in Globally Distributed Storage Systems", *In OSDI*, pp. 61-74, Oct. 2010.

[2] S. Nath, H. Yu, P. Gibbons, and S. Seshan, "Subtleties in Tolerating Correlated Failures in Wide-area Storage Systems" *NSDI,* vol. 6, pp. 225-238, 2006.

[3] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *IEEE Transactions on Dependable and Secure Computing,* vol. 7, no. 4 ,pp. 337-350, 2010.

[4] R. Rodrigues, and B. Liskov, "High availability in DHTs: Erasure coding vs. replication," *In Springer Berlin Heidelberg Peer-to-Peer Systems,* vol. VI, pp. 226-239, 2005.

[5] A. Carroll, "Why Data Centers are Necessary for Enterprise Businesses," 2013, Retrieved from: http://www.lifelinedatacenters.com/data-center/why-data-centers-are-necessary-for-enterprise-businesses/

[6] B. Lavalle, "Proliferation of Remote Data Centers Creates New Networking Challenges," 2015, Retrieved from: http://www.datacenterknowledge.com/archives/2015/05/06/proliferation-remote-data-centers-creates-new-networking-challenges/

[7] M. Itani, S. Sharafeddine, I. ElKabani. "Practical Multiple Node Failure Recovery in Distributed Storage Systems", *IEEE Symposium on Computers and Communication (ISCC),* pp. 901-907, June 2016.

[8] K. Rashmi, N. Shah, P. Kumar, and K. Ramchandran "Explicit construction of optimal exact regenerating codes for distributed storage" *47th Annual Allerton Conference on Communication, Control, and Computing* , pp. 1243-1249, Sep. 2009.

[9] 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 Storage Systems: A Study on the Facebook Warehouse Cluster," *UNISEX Hotstorage,* 2013.

[10] L. Xiang, Y. Xu, J. Lui, and Q. Chang, " Optimal recovery of

single disk failure in RDP code storage systems," *ACM SIGMETRICS,* June 2010.

[11] O. Khan, R.C. Burns, J.S. Plank, W. Pierce and C. Huang, "Rethinking Erasure Codes for Cloud File Systems: Minimizing I/O for Recovery and Degraded Reads," *FAST Proc.,* pp. 20, Feb. 2012.

[12] M. Li and P.P. Lee, "STAIR Codes: A General Family of Erasure Codes for Tolerating Device and Sector Failures in Practical Storage Systems," *ACM Trans. on Storage (TOS),* vol. 10, no. 4, pp.14, Oct. 2014.

[13] D.S. Papailiopoulos, J. Lou, A.G. Dimakis, C. Huang and J. Li, "Simple Regenerating Codes: Network Coding for Cloud Storage," *IEEE INFOCOM Proc.,* pp. 2801-2805, March 2012.

[14] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Systems," *IEEE Trans. on Information Theory,* vol. 56, no.9, pp. 4539-4551, 2010.

[15] S. ElRouayheb and K. Ramchandran, "Fractional Repetition Codes for Repair in Distributed Storage Systems," *48th IEEE Annual Allerton Conf. on Communication, Control and Computing,* 2010.

[16] S. Pawar, N. Noorshams, N. ElRouyaheb and K. Ramchandran,"DRESS Codes for the Storage Cloud: Simple Randomized Constructions," *Proc. of IEEE on Information Theory (ISIT),* pp. 2338-2342, July 2011.

[17] B. Zhu, K. Shum, and H. Li, "Heterogeneity-aware codes with uncoded repair for distributed storage systems," *IEEE Communications Letters,* vol. 19, no. 6, pp. 901-904, June 2015.

[18] B. Zhu, K. Shum, H. Li, and H. Hou, "General fractional repetition codes for distributed storage systems, " *IEEE Communications Letters,* vol.18, no.4, pp. 660-663, April 2014.

[19] N. Silberstein and T. Etzion, "Optimal fractional repetition codes based on graphs and designs, " *IEEE Transactions on Information Theory,* vol. 61, no. 8, pp. 4164-4180, Aug. 2015.

[20] N. Silberstein, "Fractional repetition and erasure batch codes," *Coding Theory and Applications: 4th International Castle Meeting (CIM Series in Math. Sciences),* USA Springer-Verlag, Sep. 2014.

[21] B. Rong, F. Douglis, Z. Liu, and C. Xia, "Failure recovery in cooperative data stream analysis", *IEEE Second International Conference on Availability, Reliability and Security,* pp. 77-84, 2007.

[22] Y. Zhu, P. Lee, L. Xiang, Y. Xu, and L. Gao, "A cost-based heterogeneous recovery scheme for distributed storage systems with RAID-6 codes", *IEEE/IFIP International Conference on Dependable Systems and Networks,* pp. 1-12, 2012.

[23] S. Xu, R. Li, P.P. Lee, Y. Zhu, L. Xiang, Y. Xu and J. Lui, "Single disk failure recovery for x-code-based parallel storage systems," *IEEE Transactions on Computers,* vol. 63, no.4, pp.995-1007, April 2014.

[24] Y. Hu, Y. Xu, X. Wang, C. Zhan, P. Li, "Cooperative recovery of distributed storage systems from multiple losses with network coding," *IEEE Journal on Selected Areas in Communications,* vol. 28, no. 2, pp.268-76, Feb. 2010.

[25] Q. Yu, C.W. Sung, and T.H. Chan, "Irregular Fractional Repetition Code Optimization for Heterogeneous Cloud Storage," *IEEE Journal on Selected Areas in Communications,* vol. 32, no. 5, pp. 1048-1060, 2014.

[26] A. Kermarrec, N. Scouarnec and G. Straub, "Repairing multiple failures with coordinated and adaptive regenerating codes," *IEEE International Symposium on Networking Coding*, pp. 1-6, Jul. 2011.

[27] J. Li and B. Li, "Beehive: erasure codes for fixing multiple failures in distributed storage systems," *7th USENIX Workshop on Hot Topics in Storage and File Systems*, 2015.

[28] B. Zhu, H. Li, H. Hou, and K.W. Shum,"Replication-based distributed storage systems with variable repetition degrees," *IEEE Twentieth National Conference on Communications (NCC),* pp. 1-5, Feb 2014.

[29] J.S. Plank and M. Blaum, "Sector-Disk (SD) Erasure Codes for Mixed Failure Modes in RAID Systems," *ACM Trans. on Storage (TOS),* vol. 10, no. 1 , pp. 4, Jan. 2014.

[30] S. Hou, Y. Liu, H. Wen and Y. Chen, "A Self-crossover Genetic Algorithm for Job Shop Scheduling Problem," *IEEE Intl. Conf. on Industrial Engineering and Engineering Management,* pp. 549-554, Dec. 2011.

[31] M. Negnevitsky, "Artificial Intelligence: A Guide to Intelligent Systems," Pearson Education, 2005.