Practical Multiple Node Failure Recovery in Distributed Storage Systems

M. Itani*, S. Sharafeddine[†] and I. ElKabbani*[‡]
*Department of Computer Science and Mathematics
School of Sciences, Beirut Arab University
Beirut, Lebanon
Email:may.itani@lau.edu.lb; Email:islam.kabani@bau.edu.lb
[†]Department of Computer Science and Mathematics
School of Arts and Sciences, Lebanese American University
Beirut, Lebanon
Email: sanaa.sharafeddine@lau.edu.lb

***Abstract*—As multiple node failures are becoming so frequent in distributed storage systems, many erasure coding techniques are emerging to handle such failures. In this paper we use the fractional repetition code to apply as a redundancy scheme for multiple failure recovery with optimized system cost. The fractional repetition (FR) code is a class of regenerating codes that consists of a concatenation of an outer maximum distance separable (MDS) code and an inner fractional repetition code that splits the data into several blocks and stores multiple replicas of each on different nodes in the system. We model the problem as an integer linear programming problem that uses modified versions of the fractional repetition code by allowing different block sizes, and minimizes the recovery cost of all dependent and independent multiple node failure scenarios. First, we generate an optimized block distribution scheme that minimizes the total system repair cost together with a full recovery plan with a node repair order for the system. Moreover, we account for the common scenario of having newcomer blocks. We allocate newcomers to nodes with minimal computations and without changing the original optimized plan. The problem is solved using genetic algorithms that search within the feasible solution space. Fast convergence validates the efficacy of our algorithms for different system parameters. Simulation results are shown to be close to optimal for the case of newly arriving blocks.**

***Keywords*-distributed storage systems; multiple failures; FR codes; failure recovery; genetic algorithms**

I. Introduction

Recent availability studies on distributed storage systems show that hardware failures in data centers occur in groups rather than being single node failures [1]. In [2], two large high-performance computing sites were examined for failures over a period of nine-years and were shown that around 23,000 failures of different causes were recorded on more than 20 different systems. Another extensive one-year availabilty study on Google's main storage infrastructure in [3] showed that 37% of node failures in distributed systems are part of a burst of at least 2 nodes. To tolerate failures, the easiest way to prevent data loss is to store replicas; however, this results in decreased storage efficiency. Thus the other alternative is to store encoded data using erasure coding [4]. Classical erasure codes transform a message of k symbols into a longer message (codeword) with n symbols such that the original message can be recovered from a subset of the n symbols. Although traditional erasure codes can reduce the storage overhead as compared to replication, extensive network resources are needed to repair a lost node. This is due to the fact that a surviving node should read all its data, process them, then send a linear combination of them to the replacement node. To minimize the bandwidth consumed during the repair process, regenerating codes were introduced in literature where a failed node can be recovered by connecting to a subset of d surviving nodes and downloading one block of data from each.

In this work, we address the problem of multiple failure recovery using a family of regenerating erasure codes, fractional repetition (FR) 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. This allows for a low complexity repair process [5]. We present a solution for multiple node failures by implementing FR codes in a genetic algorithm that is based on natural evolution. The algorithm generates an optimized block distribution scheme for the system that minimizes total system multiple failure recovery cost, taking into account different node failure scenarios. The solution also provides a full system recovery plan together with the optimized repair order pattern. We then extend the problem to account for newcomer block allocation on nodes again with optimized multiple failure recovery cost for the system. We demonstrate the performance of the proposed algorithm through simulations with different system parameters.

We first provide a summary of the literature in section II. The system model and our formulated problem are discussed in sections III and IV, respectively. This is followed by algorithm description and implementation in section V, together with running examples and simulation results in section VI. Finally we conclude in section VII.

[‡]On leave from Alexandria University, Egypt

II. Related Work

Several erasure codes [6]–[8] are being designed for the purpose of optimizing performance of failure recovery in distributed storage systems. Performance metrics include storage space, reliability and recovery cost. Of these codes, regenerating codes were first proposed as a new paradigm in coding that achieves minimized system bandwidth requirements during recovery [9]. However, in case of node failure, regenerating codes require a number of helper nodes to read all its data and generate a linear combination of them to send it to the new node designated to substitute the failed node referred to as newcomer node. In this way the recovery approach does not satisfy the uncoded repair property.

Later in 2010 constructions of FR codes were introduced by Rouayheb et al [5]. Of these, deterministic constructions based on regular graphs and Steiner systems were presented as a first step in the study of fractional repetition codes, where a helper node reads only one of its stored blocks and sends it to the newcomer with no processing. Pawar et al proposed a randomized construction of FR codes based on the balls and bins probabilistic model [10]. These randomized constructions provide more flexibility in terms of possible system parameters compared to those constructions of [5]. Both constructions take different design considerations into account and are based on theoretical mathematical models but are not applied to provide an optimal block storage scheme to achieve the goal of minimal recovery cost. Studying the feasibility of implementing these new codes in practical storage systems is still under research.

Several availability studies on data centers are being conducted to show that there is a higher need for multiple failure recovery approaches than single failure approaches [1]–[3]. Very few research tackled the multiple failure recovery problem. We state that of Li et al [7], who proposed a new code that minimizes the bandwidth needed to recover from concurrent failures. Another recent and relevant work is that of Yu [11] where the author proposed a new version of FR codes, irregular FR codes, that consider different storage capacities of nodes, different communication costs, and different number of packets per coded block. The design considerations require selection of helper nodes from a limited set of nodes determined using hyper graphs. As for single node failures, Zhu proposed a recovery solution for distributed systems that use row diagonal parity (RDP) and EvenOdd RAID6 erasure codes. Single node failures can be recovered by reconstructing the data of a failed node using decoding techniques [12]. Another work was conducted on Facebook warehouse clusters, where a framework was designed for a recovery approach based on codes that are efficient in the amount of data read and downloaded during node-repair. The basic idea behind this framework was to take multiple stripes of existing data on nodes and carefully add functions of the data of one stripe to other stripes. This technique required extra parities and computation of specific functions, and thus used coded repair [13]. A previous work of the authors addresses the problem of single node recovery [14].

In this work, we present the multiple failure recovery problem with FR codes applied as a redundancy scheme. The problem is approached using the concept of incidence matrices such that all nodes in the network are candidates for being helper nodes. Our goal is to implement a function that generates a feasible optimized $(n \times \theta)$ incidence matrix that satisfies the FR code constraints, given specific system parameters n, d, ρ, and θ. Afterwards, we provide an extension to the problem where a new set of blocks arrives to the system and is to be allocated on the spot without re-allocation of currently residing blocks. The incidence matrix is then augmented with additional columns that account for newcomer blocks and the problem is resolved with minimal computation ; thus leading itself to be easily implemented in reality. A storage allocation matrix is computed based on an integer linear program that selects the optimized recovery cost distribution scheme and the corresponding table-based full-recovery plan for all possible scenarios of dependent and independent node failures.

III. System Model

Consider an *(n, k, d)* distributed storage system composed of a collection of n nodes, labeled by n_0, n_1, ...,n_{n-1}, where the FR code of repetition factor ρ is to be used as a redundancy scheme. FR codes consist of a concatenation of an outer maximum distance separable (MDS) code and an inner fractional repetition code. In an *(n, k, d)* system where n is the total number of storage nodes, $k<n$, is the total number of nodes contacted to retrieve a file, and $d \geq k$, is the number of nodes contacted by a replacement node during node repair [5]. 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 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. A user contacting k nodes can always decode the stored file and this is achieved by the MDS property of the outer code. A node failure can be repaired by constructing a new node that contacts a specific set of d nodes for repair depending on which node has failed. d represents also the minimum node storage capacity expressed in packets [5]. The underlying networked infrastructure connecting the storage nodes may have different transmission bandwidths and topologies. Thus each storage node will have different communication costs. Our system model is depicted in Fig. 1 where two independent nodes n_1 and n_3 are to be recovered in parallel from helper nodes n_2, n_4 and n_5.

To explain our work, we provide a detailed example where we 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 replication factor of 3. Each block will be replicated on three different nodes and each node will store two distinct blocks. The communication cost between two nodes depends on various factors including bandwidth transmission speeds, cabling and I/O disk access. The cost matrix associates with every node a generic retrieval cost for a given block size

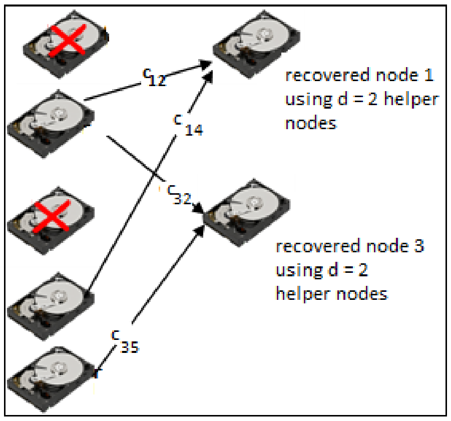
c_{12}
c_{14}
c_{32}
c_{35}
recovered node 1 using d = 2 helper nodes
recovered node 3 using d = 2 helper nodes

Fig. 1. Independent node recovery in a distributed storage system

from another node. In case of different block sizes, block retrieval costs are calculated proportionally from the retrieval cost matrix based on a minimum unit block size of 1500 byte. 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 given below.

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

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

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 $f < \rho/2$ as generalized in Fig. 2 to be possible to recover them in parallel even if they have blocks in common. 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 retrieval plan $\mathbf{R_{24}}$ for these 2 nodes would be in this case evaluated as:

$$\mathbf{R_{24min}} = \begin{array}{c} \\ n_2 \\ n_4 \end{array} \begin{array}{c} \begin{array}{cccc} \theta_1 & \theta_2 & \theta_3 & \theta_4 \end{array} \\ \begin{pmatrix} 5 & 0 & 0 & 4 \\ 0 & 3 & 0 & 6 \end{pmatrix} \end{array}$$

The interpretation of $\mathbf{R_{24min}}$ would be to recover node 4 blocks from node 3 for block 2 (θ_2), and from node 6 for block 4 (θ_4) , then to recover node 2 from node 5 and recovered node 4 according to the recovery plan given. 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 specified above and taking into consideration all multiple node failure scenarios. 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

$$\begin{aligned} \mathbf{RC_{tot}} &= RC_{12min} + RC_{13min} + RC_{14min} + RC_{15min} + \\ & RC_{16min} + RC_{23min} + RC_{24min} + RC_{25min} + RC_{26min} + \\ & RC_{34min} + RC_{35min} + RC_{36min} + RC_{45min} + RC_{46min} + \\ & RC_{56min} = 10 + 18 + 11 + 9 + 9 + 17 + 12 + 15 + 9 + 18 + \\ & 16 + 16 + 10 + 10 + 9 = 189. \end{aligned} \tag{1}$$

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

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

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.

IV. Problem Formulation

Given a distributed system with n nodes. θ distinct blocks are to be stored on n nodes with a given replication factor ρ. Every node can store a minimum of d blocks. Given different communication costs and link bandwidths between nodes, the problem of failure recovery with different failure scenarios is modeled and solved using incidence matrices.

A fractional repetition code can be represented by a $\{0, 1\}$-incidence matrix **B** of dimensions n $\times$ θ defined as the matrix (b_{ij}) where:

$$b_{ij} = \begin{cases} 1, & \text{if block j} \in \text{node i} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where rows represent storage nodes n and columns represent blocks θ. These codes will be used in modeling our failure recovery problem.

Let the communication cost matrix $\mathbf{C}$ represent the cost of newcomer node β to retrieve a typical size block from a helper node α defined as the matrix $(c_{\beta\alpha})$ where:

$$c_{\beta\alpha} = c_{\alpha\beta} = \begin{cases} \text{v}, & \text{v} \in R^{+} \\ 0, & \text{if } \alpha{=}\beta \end{cases} \tag{3}$$

Let the variable $c_{\alpha j}$ denote the cost for retrieving a specific block j from a helper node α, where $c_{\alpha j}$ = $c_{\alpha\beta}\times$normalized size of block *j* (with respect to typical size block). 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. The mathematical formulation is as follows: For every possible combination we have:

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

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

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

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

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

We need to solve for a block assignment matrix B that minimizes the value of the system retrieval cost **RC** in Equation (5). The retrieval cost constitutes the sum of single block retrieval costs per node and evaluated to include all possible orders of retrieval; i.e., $f!$ 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 selecting the plan with minimum cost for each combination. Moreover, the total cost value is then generalized to handle all possible multiple node failure scenarios whose count is $\binom{n}{f}$. 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* ($x_{\alpha j}\&x_{ij} \neq 0$).

$x_{ij} \in \{0,1\}$ is a Boolean variable indicating that node *i* holds block *j*. The value RC is to be minimized under the condition that the block assignment matrix satisfies a number of constraints.

Equation (6) is a replication factor constraint where the total number of replicas of a specific block *j* in all nodes is ρ.

Equation (7) specifies the total number of blocks per node. Assuming different size blocks and a specific storage capacity per node, we should add the storage constraint (8), where s_j is the size associated with every block *j* and **SC** is the total storage capacity of each node in the system.

V. Genetic Algorithm Based Solution and Implementation

The implemented code for the solution methodology was designed as a self-cross-over genetic algorithm that starts with a random distribution of blocks on nodes and then searches within the feasible space by redistributing the blocks and generating an optimized solution [15], [16]. A description of the algorithm phases is stated next.

A. Chromosome Representation

The modeling of a chromosome was that each one constitutes a binary block assignment matrix generated using different permutations and satisfying the constraints (6), (7) and (8). Every chromosome represents a feasible solution. An example of a chromosome representation for the values (n=3, d=2, θ=3, ρ=2) is [1 0 1 1 1 0 0 1 1] where the n$\times\theta$ block assignment matrix is transformed to a single (n$\times\theta$,1) row matrix.

B. Generation of Initial Population

The phase that follows is generating an initial population for reproduction after carrying out the chromosome encoding phase. Given a pre-specified population size *p*, the initial population will include *p* chromosomes that are generated at random using a self-designed constructive method implemented as a function that generates a feasible allocation scheme.

C. Self Cross-over and Mutation Operations

The conventional cross-over technique cannot be applied in our model since it will result in an in-feasible new chromosome. This paper adopts the self-cross-over method inspired by the fact that the feasible allocation matrices can be generated from one another by exchanging rows within a single matrix. Note that elitism was used to help keep up the optimal chromosome in every generation. Mutation is also used to maintain diversity in the population and to make sure that the achieved solution is not a local optima.

D. Fitness Function

Every chromosome or individual is associated with a fitness function equivalent to the optimization problem objective function defined in Equation 5 and calculated using Algorithm 1 after following the steps in Fig. 2 introduced earlier in section III.

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.

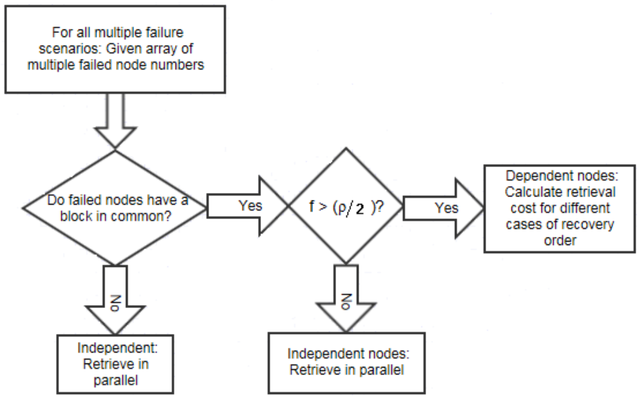
For all multiple failure scenarios: Given array of multiple failed node numbers
Do failed nodes have a block in common?
Yes
No
Independent: Retrieve in parallel
f > (ρ/2)?
Yes
No
Independent nodes: Retrieve in parallel
Dependent nodes: Calculate retrieval cost for different cases of recovery order

Fig. 2. Node failure classification flowchart

VI. Simulation Results and Analysis

In this section, we present the results of our implementation for different cases of multiple node failures. The machine employed for simulation is a Lenovo laptop with 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. We consider a system with 10 storage nodes and 30 distinct blocks being replicated three times for a total of 90 blocks (θ = 30, ρ = 3) to be distributed on the 10-node (n = 10) system in an optimized scheme for multiple failure recovery. The resulting simulation for these system parameters is presented in Fig. 3. 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. We notice that mutation occurs in the seventh generation and by that we can explain why the average fitness peaked to a maximum value. As for Fig. 4, the system parameters are n = 25 nodes, θ = 50 blocks and replication factor ρ = 3.

As observed from Fig. 3 and Fig. 4, the average fitness

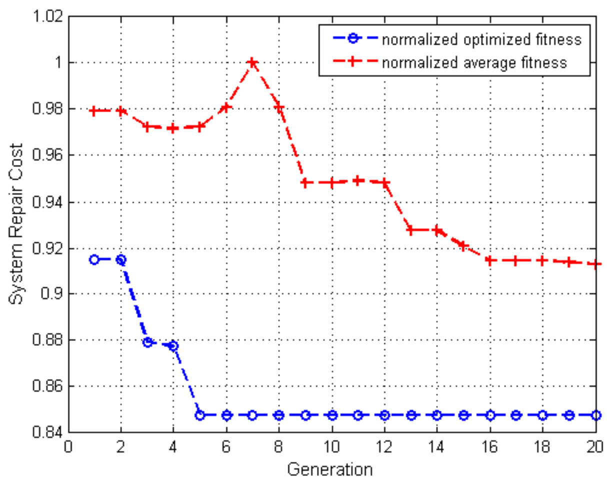

Fig. 3. Average and minimum system repair cost for n=10, θ=30, ρ=3

```
Algorithm 1.  Fitness Function Calculation
Given a chromosome that represents a
binary block assignment matrix, perform the
following:
 1.   for $w = 1$, while $w < \frac{n!f!}{(n-f)!}$, $w$++
 2.      Let $\gamma$ be the set of failed nodes
 3.      check whether failed nodes are
         dependent using steps in Fig. 2
 4.      If nodes are independent then Repeat
         steps 5 through 9, else jump to step
          14
 5.         ∀ block j belongs to failed node
            i
 6.         Check for all nodes $\alpha$ such that
            node $\alpha$ has a replica of block j
 7.         Select the node $\alpha$ with minimum
            retrieval cost $c_{\alpha j}$
 8.         Assign $\alpha$ as one of the helper
            nodes for failed node i and save
            it in recovery plan matrix
 9.         Update node i retrieval cost value
            to be $\sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\notin\gamma}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j}$
10.         Update recovery plan to include
            helper nodes for recovering all
            blocks j of node i
11.      End of inner loop
12.      Update total retrieval cost value
         for the failed nodes in set $\gamma$
13.      Update system recovery plan for next
         failure scenario
14.         ∀ permutations P($\gamma_y$) of array $\gamma$
15.         Calculate retrieval cost for
            failed node i using
            $\sum_{j=1}^{\theta} \min_{\substack{\alpha=1,2,\ldots,n \\ \alpha\notin\gamma}} c_{\alpha j} \cdot x_{ij} \cdot x_{\alpha j}$
16.         Remove i from set of failed nodes
            $\gamma$
17.         Repeat till y=f! and update the
            retrieval cost for the specific
            permutation that constitutes a
            recovery order
18.         From set of all permutations
            select the one associated with
            the minimum recovery cost of
            dependent nodes
19.      Update total system recovery cost
         for all cases of multiple node
         failures
20.   End of outer loop
```

of individuals decreases in each generation. This illustrates the fact that better solutions are being generated in newer populations. It is also clear that the optimized fitness, i.e., optimized system repair cost decreases till it converges to a minimal value. We can try to avoid being stuck in a local optima by increasing the mutation rate and re-running the algorithm.

Moreover, it is worth to note that the optimized solution is achieved as early as the fifth generation for n = 10 nodes (Fig. 3) and as the sixth generation when the number of nodes is increased to 25 nodes (Fig. 4). That shows the

```
Algorithm 2.  Newcomer Blocks Allocation
Given a chromosome that represents an
optimized binary block assignment matrix,
perform the following:
for $j = 1$, while $j$ <number of newcomer blocks,
$j$++
1.  Replicate block j , $\rho$ times
2.  Distribute replicas of j by augmenting
    the optimal block assignment matrix that
    was generated based on best fitness using
    Algorithm 1
3.  Calculate the fitness of the newly
    augmented column for different multiple
    failure scenarios
4.  Update the value of the optimized fitness;
    i.e. cost generated by Algorithm 1
5.  Select chromosomes with best fitness
6.  Apply cross-over and mutation operations
    only on augmented columns of chromosomes
    with best fitness
7.  After 20 or more generations, select the
    best chromosome that has minimal retrieval
    cost and update the original optimized
    block allocation schema together with the
    optimized recovery plan to account for the
    newcomers
8.  Repeat steps 1-7 for all newcomer blocks
```

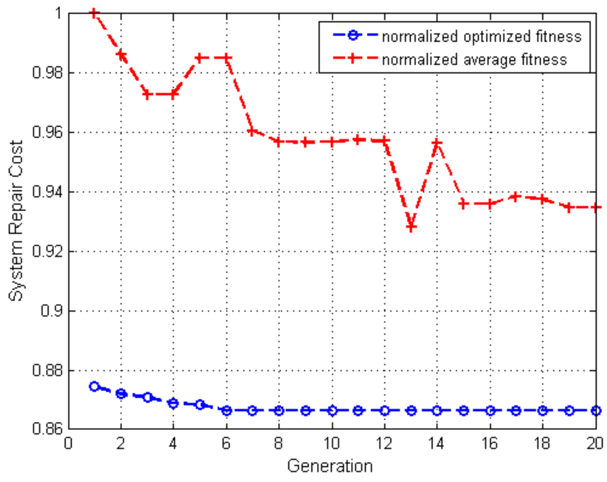

Fig. 4. Average and minimum system repair cost for n=25, θ=50, ρ=3

quick convergence of the algorithm even when we increase the number of nodes and blocks.

We next compare the minimum system repair cost of our heuristic implementation for the case of newcomer blocks to that of the optimal brute force implementation for different network sizes and this is shown in Table I. Our results are shown to be near optimal since the difference between the heuristic solution and optimal solution for the specific tested scenarios is calculated at most as 1%. As for Fig. 5, we present the results of heuristic minimal normalized system cost for the

TABLE I
MINIMAL POST-OPTIMAL SYSTEM REPAIR COST FOR DIFFERENT NETWORK SIZES FOR THE CASE OF NEWCOMER BLOCKS

Network Size n	Optimal Normalized Repair Cost	Heuristic Normalized Repair Cost	True Error (%)
6	0.966	0.966	0%
7	0.965	0.965	0%
8	0.9645	0.9645	0%
9	0.9641	0.9647	0.6%
10	0.963	0.964	1%

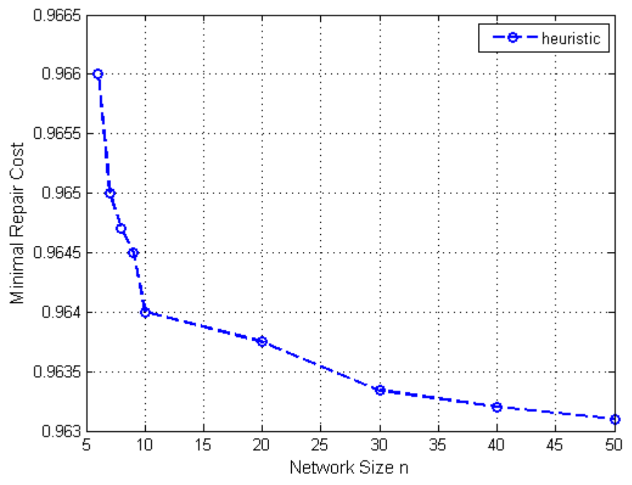

Fig. 5. Minimal post-optimal heuristic system repair cost for different network sizes for the case of newcomer blocks

case of newcomer blocks for different number of nodes. We can interpret the decreasing cost result due to the fact that as the network increases, the repair cost becomes higher, thus the normalized cost decreases.

The heuristic simulation results in Table 1 and those of Fig. 5 are calculated as the average cost of 20 runs for each value of n ; i.e. a total of 80 runs and then normalized by the maximum average cost in each generation for fair comparison.

VII. CONCLUSION

We presented the design of a failure recovery approach that minimizes the system repair cost for the solution of multiple node failures in a distributed storage system. We then extended the work to consider a practical dynamic scenario where new blocks arrive to the system. Our solution used FR codes to provide a simple repair mechanism that minimizes the repair and communication costs. We formulated the problem using incidence matrices and solved it heuristically using a genetic algorithm for all combinations of multiple node failures. For the problem of newcomer blocks, a post-optimal storage allocation matrix was computed in a way that can be easily implemented in reality without the need to redistribute actual residing blocks. Our simulation results achieved fast convergence for different system parameters in the first scenario. As for the second one, the solution was shown to be close to optimal for the tested system parameters.

References

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

[2] 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.

[3] D. Ford et al, "Availability in Globally Distributed Storage Systems," *In OSDI,* pp. 61-74, 2010.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Y. Zhu, P.P. Lee, L. Xiang, Y. Xu, and L. Gao,"A Cost Based Heterogeneous Recovery Scheme for Distributed Storage Systems with RAID-6 Codes," *42nd IEEE/IFIP Intl. Conf. on Dependable Systems and Networks,* pp. 1-12, June 2012.

[13] 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.

[14] M. Itani, S. Shaafeddine, I. ElKabbani. "Practical Single Node Failure Recovery Using Fractional Repetition Codes in Data Centers," *AINA Intl. Conf.,* May 2016.

[15] 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.

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