Irregular Fractional Repetition Code Optimization for Heterogeneous Cloud Storage

Quan Yu, Chi Wan Sung, *Member, IEEE*, and Terence H. Chan, *Member, IEEE*

***Abstract*—This paper presents a flexible irregular model for heterogeneous cloud storage systems and investigates how the cost of repairing failed nodes can be minimized. The fractional repetition code, originally designed for minimizing repair bandwidth for homogeneous storage systems, is generalized to the irregular fractional repetition code, which is adaptable to heterogeneous environments. The code structure and the associated storage allocation can be obtained by solving an integer linear programming problem. For moderate sized networks, a heuristic algorithm is proposed and shown to be near-optimal by computer simulations.**

***Index Terms*—Cloud Storage, Distributed Storage Systems, Irregular Fractional Repetition Code, Regenerating Code**

I. Introduction

CLOUD storage is a new paradigm of storing data. It allows users to access data anywhere and anytime. Companies such as Google and Apple are providing this service through their data centers, which are network-connected. Such an architecture is called the distributed storage system (DSS). Storage nodes in a DSS are generally unreliable and subject to failure. When a failure occurs, a newcomer needs to *repair* the lost data by retrieving data from surviving storage nodes, called helper nodes, so as to maintain the *reliability* of the DSS. Besides, the DSS should be able to provide data *availability*, which allows users to access their data anywhere and with low delay.

To provide reliability and availability, erasure codes such as replication or Reed-Solomon (RS) code are commonly used. While replication requires less network bandwidth during node repair, RS code is more efficient in terms of storage space. In 2007, Dimakis et al. showed that there is a fundamental tradeoff between storage space and repair bandwidth [1]. Points on the tradeoff curve can be achieved by a class of codes called *regenerating codes*, which is based on the concept of network coding. In their formulation, a newcomer is able to recover the lost data by connecting to any d surviving storage nodes, and a data collector is able to retrieve the data object by downloading data from any k out of the n storage nodes. We call this distributed storage model the *regular* model. Since then, many codes that achieve points on the tradeoff curve have been constructed (e.g. [2], [3], [4], [5]).

The design rationale of regenerating codes is to minimize repair bandwidth. These codes, however, generally incur high disk I/O access during repair, since helper nodes need to read its stored data and linearly combine them to form packets to be sent to a newcomer. The stored data that needs to be read is often much more than the data to be sent to the newcomer. The disk access bandwidth thus becomes the bottleneck. In [6], the repair problem is considered in a different way. It aims to minimize the amount of information to be accessed when the number of node failures is smaller than the erasure correcting capability of an MDS code. Another approach is considered in [7]. It proposes a new code formed by concatenating an outer MDS code with an inner fractional repetition (FR) code. We call it MDS-FR code. This code is a minimum bandwidth regenerating (MBR) code, which means that it minimizes the repair bandwidth of the system. Furthermore, it has the nice *uncoded repair* property: a helper node only needs to read the exact amount of data that it needs to forward to the newcomer without any processing. In other words, it minimizes both repair bandwidth and disk access bandwidth at the same time. While the original construction of the FR code in [7] is based on regular graph and Steiner system, other constructions exist, which are based on bipartite graph [8], randomized algorithm [9], resolvable designs [10], and incidence matrix [11]. Note that the above mentioned works do not strictly follow the regular model, as they have different design considerations in mind. Another notable example is the locally repairable code [12], [13], [14], which aims at reducing the number of nodes that need to be contacted during repair.

In this paper, we focus on heterogeneous distributed storage systems. Examples include heterogenous data centers, peer-to-peer cloud storage systems (e.g. Space Monkey) [15], peer-assisted cloud storage systems, and some wired or wireless caching systems [16], [17]. In these applications, the storage nodes and the network links are *heterogeneous*, meaning that the storage capacities and costs associated with different storage nodes may not be the same, and the communication links between each pair of storage nodes may have different characteristics in terms of bandwidth, communication cost, and transmission rate. Furthermore, it is also possible that some storage nodes are not directly connected. In such an environment, new issues arise. The storage allocation problem, which focuses on how to allocate a given storage budget over the storage nodes such that the probability of successful

Manuscript received May 15, 2013; revised September 30, 2013 and November 20, 2013. This paper was presented in part at the IEEE International Conference on Communications (ICC), Ottawa, Canada, June 2012. This work was supported in part by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08), and in part by the Australian Research Council (DP1094571).

Q. Yu is with Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: Q.Yu@my.cityu.edu.hk).

C. W. Sung is with Department of Electronic Engineering, City University of Hong Kong, Hong Kong (e-mail: albert.sung@cityu.edu.hk).

T. H. Chan is with Institute for Telecommunications Research, University of South Australia, Adelaide, SA 5095, Australia (e-mail: terence.chan@unisa.edu.au).

Digital Object Identifier 10.1109/JSAC.2014.140523.

recovery is maximized, is studied in [18]. A distributed storage system in which the storage nodes have different download costs is considered in [19]. In a distributed storage system with storage cost, how to allocate storage capacities among the storage nodes so as to minimize the total storage cost is investigated in [20]. In [21], the bandwidth heterogeneity is taken into account to demonstrate that the tree-structured regeneration topology is an efficient topology to reduce the regeneration time. Under functional repair model, the link costs and the impact of network topology are jointly considered in [22], and an information-theoretic study is performed in [23].

To address the design issues of heterogenous cloud storage systems, we set up a flexible model, called the *irregular* model, in which the underlying network topology can be arbitrary, the storage capacities and costs of different storage nodes are allowed to be different, and the bandwidth and costs of communication links need not be the same. We relax the constraints of data repair and data retrieval in the regular model by introducing the concepts of repair overlay and retrieval sets. We use the term repair overlay to refer to the structure of an overlay network for data repairing. Note that it is called repair table in [7]. In the work of [7], for single failure case, the repair overlay is restricted to be a regular graph with each vertex having degree d, and the graph is randomly generated. In this paper, we do not restrict the repair overlay to be a regular graph. For the general case of multiple failures, the repair overlay in [7] is a Steiner system. However, the existence of a Steiner system requires the system parameters to satisfy some specific conditions, which makes the system design inflexible. In this paper, hypergraph is used to model the repair overlay, which exists for arbitrary system parameters and can be constructed easily compared with Steiner system.

Recall that the code used in [7] is a concatenation of an outer MDS code and an inner FR code. We call this construction the MDS-FR code. We extend the idea and propose the use of the *Irregular Fractional Repetition* (IFR) code as the inner code. While it preserves the desirable uncoded repair property, it further allows more flexibility in system design. When the distributed storage system and the underlying network is heterogeneous, the IFR code can be constructed and adapted to the given environment by solving an optimization problem, thus further reducing repair bandwidth. In our formulation, we minimize the system repair cost by properly choosing the MDS-IFR code. The problem is shown to be an integer linear programming (ILP) problem. When the number of storage nodes is small, the optimal solution can be found in a reasonable time. For larger networks, we decompose the problem into subproblems and propose a heuristic solution. For small network sizes, our heuristic is shown to be nearly optimal by comparing it with the optimal ILP method.

The rest of the paper is organized as follows. A motivating example is given in Section II. Section III states our irregular model for distributed storage systems. In Section IV, we describe the construction of MDS-IFR code and its relationship with the concept of relay overlay. In Section V, we formulate the repair cost minimization problem as an integer linear problem (ILP). In Section VI, we describe how the storage-repair tradeoff of our code can be found. In Section VII, we design heuristic algorithms to find suboptimal repair overlay and retrieval sets. Section VIII provides our simulation results. We conclude the paper in Section IX.

II. A Motivating Example

The regular model assumes that a newcomer is able to replace a failed storage node by contacting any d surviving storage nodes and a data collector can retrieve the stored data object by downloading data from any k out of the n storage nodes. In some practical scenarios, however, the communication costs between a newcomer and each of the surviving storage nodes are different. Furthermore, the distances and transmission rates between a data collector and each of the n storage nodes vary with the location of the data collector. The d surviving nodes to be contacted by a newcomer and the k storage nodes to be contacted by a data collector need not be arbitrary. It is reasonable to determine some sets of helper nodes, called *helper sets*, for a newcomer and some subsets of the n storage nodes, called *retrieval sets*, for a data collector. The collection of helper sets of all the n storage nodes defines the repair overlay. Thus, we modify data repair and data retrieval mechanisms based on the concepts of repair overlay and retrieval sets. We only require that a newcomer can rebuild the corresponding failed node by contacting the storage nodes in any one of its helper sets and a data collector can retrieve the data object by contacting the storage nodes in any one of the retrieval sets.

Consider a distributed storage system that can tolerate single failures with the following parameters: $n = 6$, and $d = k = 2$. A data object consisting of four packets would be stored in this distributed storage system. For the regular model, the corresponding tradeoff between storage amount and repair bandwidth under functional repair is shown in Fig. 1, where the feasible region is shown as the shaded area. Note that all points on the tradeoff curve are normalized by the number of packets contained in the data object. The points below the tradeoff curve are impossible to achieve by functional repair. Clearly, they cannot be achieved by exact repair either.

Now we consider the irregular model, which includes the concepts of repair overlay and retrieval sets. We require that the number of retrieval sets are large enough. In this example, we require that there should be at least 9 retrieval sets. Assume that the chosen repair overlay, denoted by τ, is a ring with six nodes, as shown in Fig. 2(a) (solid lines). We show how to construct MDS-FR code based on the repair overlay. The data object consisting of four packets are first encoded into six packets, F_1, F_2, ..., F_6 by a $(6, 4)$-MDS code. Each edge in the 6-node ring is then associated with a coded packet. Each node stores the two packets that are associated with its incident edges, as shown in Fig. 2(a). Thus, the storage amount of each of the six storage nodes is 2. In this example, each storage node has one helper set and a newcomer can recover the lost data by connecting to $d = 2$ nodes in its helper set, i.e., its two neighboring nodes in the ring, rather than *any* $d = 2$ surviving nodes. Since each newcomer downloads one packet from each of its two helper nodes to recover the lost data, the repair bandwidth of a failed node is 2. Suppose node 3 fails. A newcomer can replace it by downloading coded packets F_2

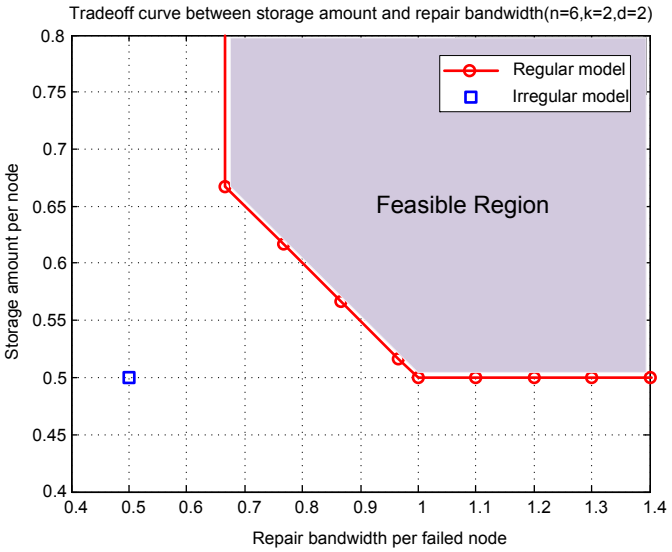


Fig. 1. Tradeoff between storage amount and repair bandwidth ($n = 6, d = 2$, and $k = 2$).

and F_3 from its two helper nodes 2 and 4, respectively, as shown in Fig. 2(b). As for data retrieval, we can have nine retrieval sets of cardinality $k = 2$, which are listed as $R_1 = \{1,3\}$, $R_2 = \{1,4\}$, $R_3 = \{1,5\}$, $R_4 = \{2,4\}$, $R_5 = \{2,5\}$, $R_6 = \{2,6\}$, $R_7 = \{3,5\}$, $R_8 = \{3,6\}$ and $R_9 = \{4,6\}$. A data collector can reconstruct the data object by connecting to the $k = 2$ storage nodes in any one of the nine retrieval sets. After normalizing by the number of packets of the original data object, the storage amount of each storage node is 0.5 and the repair bandwidth per failed node is also 0.5. This point is also plotted in Fig. 1, which is below the tradeoff curve of the regular model. From Fig. 1, we can see that with the same storage amount per node, the repair bandwidth is reduced by 50%, which shows that the potential gain can be enormous *if the constraints of data repair and data retrieval are relaxed*.

In the irregular model, different storage nodes are allowed to have different storage costs and different links are allowed to have different communication costs. An irregular model with storage cost and communication cost is given in Fig. 2(c), where node i has a (per-packet) storage cost s_i and the link connecting node i and node j has a (per-packet) communication cost c_{ij}. For both FR code and IFR code, we assume that the number of packets assigned to edge $\{i,j\} \in \tau$ is β_{ij} and the number of packets stored in node i is α_i. The total storage cost can then be obtained as $\sum_{i=1}^{6} \alpha_i s_i$ and the total repair cost of all possible single node failures can be calculated as $\sum_{i=1}^{6} \sum_{\{i,j\}\in\tau} c_{ij}\beta_{ij}$. If we use the same MDS-FR code as before, the total storage cost of the six storage nodes is $2\sum_{i=1}^{6} s_i = 34$ since each node stores 2 packets, and the total repair cost of all possible single node failures can be calculated as $(c_{12}+c_{16})+(c_{12}+c_{23})+(c_{23}+c_{34})+(c_{34}+c_{45})+(c_{45}+c_{56})+(c_{56}+c_{16}) = 36$, where the six components of the summation correspond to the repair costs of node 1 to node 6, respectively. If we use MDS-IFR code, we first encode the data object into seven packets by using a $(7,4)$-MDS code. Then we assign coded packet F_1 to edge $\{1,2\}$, F_2 to edge $\{2,3\}$, F_3 to edge $\{3,4\}$, F_4, F_5, and F_6 to edge $\{5,6\}$, and F_7 to edge $\{1,6\}$. Each node then stores the packets associated with its incident edges, as shown in Fig. 2(d). In this example, a newcomer can recover the lost data by connecting to only one node or to two nodes, depending on which node is failed. This contrasts with the MDS-FR code, in which a newcomer always connects to *exactly* d nodes. For example, if node 4 fails, a newcomer can replace it by downloading packet F_3 from node 3. As for data retrieval, we can have ten retrieval sets of cardinality $k = 2$, which are listed as $R_1 = \{1,3\}$, $R_2 = \{1,5\}$, $R_3 = \{1,6\}$, $R_4 = \{2,5\}$, $R_5 = \{2,6\}$, $R_6 = \{3,5\}$, $R_7 = \{3,6\}$, $R_8 = \{4,5\}$, $R_9 = \{4,6\}$ and $R_{10} = \{5,6\}$. The total storage cost of the six nodes is $2s_1 + 2s_2 + 2s_3 + s_4 + 3s_5 + 4s_6 = 33$ and the total repair cost of all possible single node failures can be calculated as $(c_{12}+c_{16})+(c_{12}+c_{23})+(c_{23}+c_{34})+c_{34}+3c_{56}+(3c_{56}+c_{16}) = 22$, where the six components of the summation corresponding to the repair costs of node 1 to node 6, respectively. Compared with MDS-FR code, we can see that both storage cost and repair cost can be reduced if MDS-IFR code is adopted in the irregular model. Although the retrieval sets in the two cases are different, in the latter case, more retrieval sets are provided, which is often more desirable. Should exactly the same number of retrieval sets are needed for a fairer comparison, one can simply remove one of the retrieval sets for the latter case, as that would not affect the storage and repair costs.

In the above example, we show that there can be large performance gain in designing distributed storage systems. However, the result should be interpreted with caution. We do not claim that MDS-IFR code outperforms well-known regenerating code and MDS-FR code under their respective problem settings. In fact, they are known to be optimal under their respective problem definitions. Instead, the example serves two purposes. First, it justifies the setup of the irregular model, which is more appropriate for heterogeneous cloud storage systems. Second, it explains the irregular model and the MDS-IFR code in an intuitive way, which facilitates the understanding of the next two sections, which formally define these concepts.

III. System Model

Consider a distributed storage network, in which n storage nodes are distributed across a wide geographical area and connected by a network with a specific topology. A data object is encoded and distributed among the n storage nodes. Let the data object be represented by a collection of B packets, where each packet is an element drawn from a finite field GF(q) of size q. Note that a packet is the minimum unit for all storage and transmission operations in a storage system.

A. Storage and Communication Costs

We model the underlying storage network as a connected weighted undirected graph $\tilde{G} = (\mathcal{V}, \tilde{\mathcal{E}})$, where the storage nodes are vertices in the vertex set $\mathcal{V}$ and the communication links correspond to the edges in the edge set $\tilde{\mathcal{E}}$. Throughout this paper, we assume that $\mathcal{V} \triangleq \{1, 2, \ldots, n\}$. Each vertex $i \in \mathcal{V}$ has an associated storage cost s_i indicating the cost of storing a packet in node i. We define the storage cost

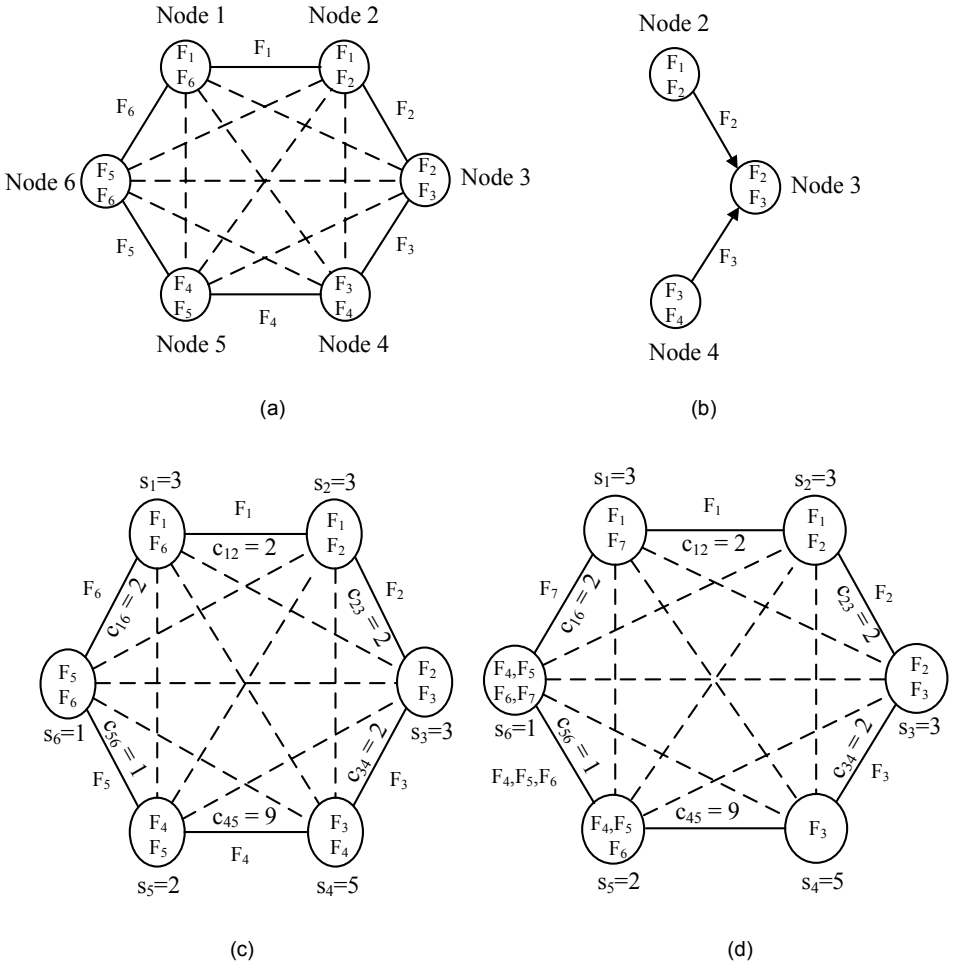


Fig. 2. An example of constructing MDS-FR code and MDS-IFR code for the irregular model.

vector $\mathbf{s} \triangleq [s_1, s_2, ..., s_n]$. Besides, each edge $\tilde{e} = \{i, j\} \in \tilde{\mathcal{E}}$ connecting vertices i and j ($i \neq j$) has an associated weight $\tilde{c}_{ij}$, called single-hop cost, which represents the cost of transmitting a packet along this edge. If there is no direct communication link between two vertices, we let the corresponding single-hop cost be infinite. The cost to transmit a packet from vertex i to vertex j is called the communication cost and is denoted by c_{ij}. The values of c_{ij}'s can be obtained from $\tilde{c}_{ij}$, depending on the underlying communication assumptions. For example, if multi-hop transmissions are allowed, then c_{ij} can be defined as the cost of the minimum-cost path from i to j, where the cost of a path is the sum of the single-hop costs of its constituent edges. If only single-hop transmissions are allowed, then c_{ij} equals $\tilde{c}_{ij}$ for all i and j. The matrix $\tilde{\mathbf{C}} = [\tilde{c}_{ij}]$ is called the single-hop cost matrix, and the matrix $\mathbf{C} = [c_{ij}]$ is called the communication cost matrix. Note that both $\tilde{\mathbf{C}}$ and $\mathbf{C}$ are symmetric.

In this paper, multi-hop transmissions are allowed in the underlying storage network. Since the storage network is assumed to be connected, we can construct a complete weighted graph $G = (\mathcal{V}, \mathcal{E})$ on the vertex set $\mathcal{V}$, where the weight of an edge $e = \{i, j\} \in \mathcal{E}$ is equal to the cost of the minimum-cost path between vertices i and j, say the communication cost c_{ij}. We call G the metric closure of $\tilde{G}$. To compute the metric closure G of $\tilde{G}$, we can use Johnson's algorithm [24, Chapter 25] to find the costs of the minimum-cost paths between all pairs of vertices in $\mathcal{V}$.

B. Repair and Retrieval Requirements

We formally define a distributed storage system with specific repair and retrieval requirements as follows:

Definition 1 (Distributed Storage System)**.** *DSS(n, ρ, d, k, w) is a distributed storage system with n storage nodes which satisfies the following requirements:*

1) *(Data Repair) As long as there are no more than ρ simultaneous node failures, the lost packets of any failed node can be exactly recovered from no more than d surviving nodes.*
2) *(Data Retrieval) A collection of w retrieval sets of cardinality k, denoted by $\Psi \triangleq \{R_1, R_2, \ldots, R_w\}$, is specified such that the data object can be obtained from any retrieval set in Ψ.*

In realistic distributed storage systems, ρ is typically a small value. For example, the 3-replication scheme where $\rho = 2$ serves the Google File System (GFS) well [25]. On the other hand, the data repair requirement is different from the regular model in that we do not require that a failed node can be repaired by contacting *any* d surviving nodes. As for data retrieval, we require that the storage system has

w retrieval sets of cardinality k. This encompasses the data retrieval requirement of the regular model as a special case, which corresponds to the setting of $w = \binom{n}{k}$. Although in our formulation, all the retrieval sets have the same cardinality, it does not mean that all the storage nodes in a retrieval set need to be contacted for data retrieval, since it is allowed that the data object can be retrieved from a subset of $R_j \in \Psi$.

IV. Code Construction

A. MDS-IFR Code

Our code construction is a concatenation of an outer MDS code and an inner Irregular Fractional Repetition (IFR) code. We call it MDS-IFR code. The data object comprised of B packets is first encoded into F packets over GF(q) by using an (F, B)-MDS code. Note that such code exists provided that $q \geq F$ (e.g. [26]). In practice, the decoding complexity may be high for large values of q. In that case, the vector linear code recently proposed in [27] can be used instead. This code is easy to decode as it has the property called zigzag decodability and all computations are performed over GF(2). The price to pay is some extra storage overhead. We refer interested readers to [27] for details.

After encoding by the outer code, the set of the F coded packets, denoted by $\mathcal{F}$, is partitioned into θ *coded blocks*, $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_\theta$, where $|\mathcal{B}_i| \triangleq \beta_i \leq B$ for all i. Note that β_i denotes the number of packets in $\mathcal{B}_i$, and $F = \sum_{i=1}^{\theta} \beta_i$. We call $\mathbf{b} \triangleq [\beta_1, \beta_2, \ldots, \beta_\theta]$ the *block assignment vector*, which will be optimized in the next section. We remark that the block assignment vector $\mathbf{b}$, rather than what packets contained in $\mathcal{B}_i$ for $i = 1, 2, \ldots, \theta$, will affect the solution of minimizing the system repair cost in the next section. That is why we introduce the definition of coded blocks instead of working directly on packets. Each coded block is then replicated $\rho + 1$ times and stored on $\rho + 1$ different storage nodes according to an IFR code, defined as follows:

Definition 2 (Irregular Fractional Repetition Code)**.** *An Irregular Fractional Repetition (IFR) code $\mathcal{C}$ for DSS($n, \rho, d, \cdot, \cdot$) is a collection $\mathcal{C}$ of n subsets of $\Omega \triangleq \{1, 2, \ldots, \theta\}$, satisfying the requirements that each set in $\mathcal{C}$ has cardinality at most d and each element of Ω belongs to exactly $\rho + 1$ sets in $\mathcal{C}$.*

Note that IFR code generalizes FR code in that it only requires the cardinality of each set in $\mathcal{C}$ to be no more than d, rather than exactly d. That is why we call it *irregular*. Besides, it addresses only the repair issue, which will become clear after we introduce the concept of repair overlay, and is independent of the parameters related to data retrieval, that is, k and w.

An IFR code can be represented by a hypergraph $\tau = (\mathcal{V}, \mathcal{H}^\tau)$, where $\mathcal{V}$ is the vertex set and $\mathcal{H}^\tau \triangleq \{E_1, E_2, \ldots, E_\theta\}$ is a family of θ non-empty subsets of $\mathcal{V}$, called hyperedges. A hypergraph is said to be ζ-uniform if all of its hyperedges have the same size ζ. The following fact is evident:

Fact 1. *An Irregular Fractional Repetition (IFR) code $\mathcal{C}$ for DSS(n, ρ, d, k, w) is equivalent to a $(\rho + 1)$-uniform hypergraph τ with θ hyperedges and n vertices, each of which has degree less than or equal to d.*

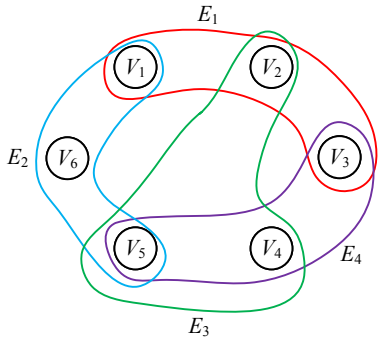


Fig. 3. A 3-uniform hypergraph of six vertices with maximum degree 3.

We call such kind of hypergraph τ a *repair overlay*, or an *overlay hypergraph*. The above fact follows directly from the definitions of IFR code and uniform hypergraph, which can be illustrated by the example below.

Example: Let $\Omega = \{1, 2, 3, 4\}$ and $\mathcal{C} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{3, 4\}, \{2, 3, 4\}, \{2\}\}$. Note that $n = |\mathcal{C}| = 6$. Furthermore, it can be checked that each element of Ω belongs to three sets in $\mathcal{C}$, so $\rho = 2$. Besides, the cardinality of each set in $\mathcal{C}$ is at most three, so $d = 3$. Therefore, $\mathcal{C}$ is an IFR code for DSS$(6, 2, 3, k, w)$.

This IFR code $\mathcal{C}$ can be represented by a 3-uniform hypergraph $\tau = (\mathcal{V}, \mathcal{H}^\tau)$, where $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $\mathcal{H}^\tau = \{E_1 = \{v_1, v_2, v_3\}, E_2 = \{v_1, v_5, v_6\}, E_3 = \{v_2, v_4, v_5\}, E_4 = \{v_3, v_4, v_5\}\}$, as shown in Fig. 3. The hypergraph τ has $n = 6$ vertices, each of which has degree less than or equal to $d = 3$. Each hyperedge in τ contains $\rho + 1 = 3$ vertices.

B. Data Distribution and Data Repair

Let $\tau = (\mathcal{V}, \mathcal{H}^\tau)$ be a given repair overlay. As described before, the data object is first encoded into θ coded blocks by an outer MDS code. For $i = 1, 2, \ldots, \theta$, the coded block $\mathcal{B}_i$ is then assigned to $E_i \in \mathcal{H}^\tau$. All vertices contained in E_i then store $\mathcal{B}_i$ in common. The storage amount α_v of a vertex $v \in \mathcal{V}$ can then be obtained as $\alpha_v = \sum_{i: v \in E_i} \beta_i$. Note that F and α_v are related by $(\rho + 1)F = \sum_{v \in \mathcal{V}} \alpha_v$, since each coded block is replicated $\rho + 1$ times.

Data repair is very simple. When there is a node failure, a newcomer will replace the failed node by retrieving the previously stored data from a set of helper nodes. For example, suppose node v which contains coded blocks $\{\mathcal{B}_i : v \in E_i\}$ fails, the newcomer can directly retrieve $\mathcal{B}_i$ from any surviving node in E_i for all i such that $v \in E_i$. This is what we call *uncoded* and *exact* repair. Since the cardinality of a hyperedge is $\rho + 1$, this kind of repair can be done successfully provided that the number of node failures is no more than ρ.

Example: Consider a distributed storage network $\tilde{G}$ shown in Fig. 4(a). The number associated with an edge denotes the single-hop cost between its two endpoints. Fig. 4(b) is the metric closure, G, of $\tilde{G}$, where the number associated with an edge is the corresponding communication cost. Suppose this storage network can tolerate up to $\rho = 2$ node failures, and each failed node can be recovered from at most $d = 3$ available storage nodes. One feasible repair overlay is shown

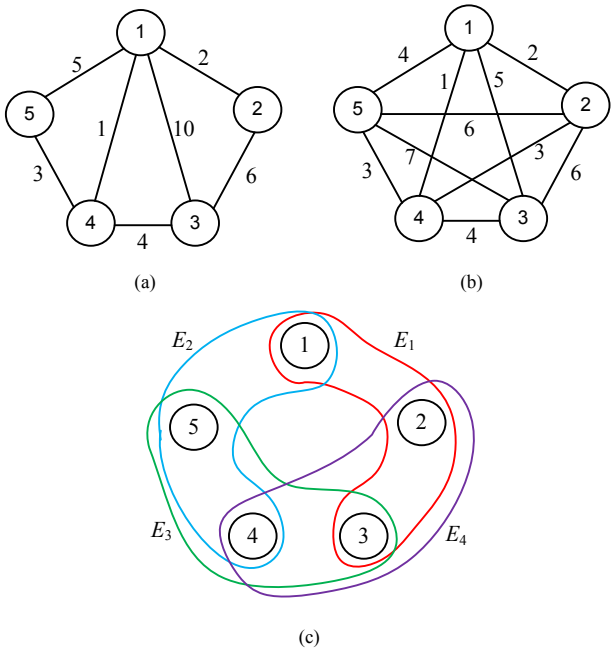


Fig. 4. An example of constructing MDS-IFR code on hypergraph.

in Fig. 4(c), where every hyperedge has $\rho+1=3$ nodes and the degree of each node is less than or equal to $d=3$. Assign coded blocks $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$ to hyperedges E_1, E_2, E_3 and E_4 respectively. Then node 1 would store blocks $\mathcal{B}_1$ and $\mathcal{B}_2$, node 2 would store $\mathcal{B}_1$ and $\mathcal{B}_4$, node 3 would store $\mathcal{B}_1$, $\mathcal{B}_3$ and $\mathcal{B}_4$, node 4 would store $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$, and node 5 would store $\mathcal{B}_2$ and $\mathcal{B}_3$. Suppose nodes 1 and 2 fail. The newcomer for node 1 can download $\mathcal{B}_1$ from node 3 and download $\mathcal{B}_2$ from either node 4 or node 5, while the newcomer for node 2 can download $\mathcal{B}_1$ from node 3 and download $\mathcal{B}_4$ from node 3 or node 4.

V. Repair Cost Minimization

In this section, we consider the problem of minimizing repair cost. We first present an algorithm to find optimal repair order when there is more than one failed node. Based on this result, we further construct an optimization framework to determine the rate of the outer MDS code, the structure of the IFR code by means of repair overlay, the storage amount of each node, and the collection of retrieval sets.

A. Repair Order under Multiple Failures

In this subsection, we assume that a repair overlay, τ, is given. We use Υ to denote a set of failed nodes and call it a failure pattern. Let $\Upsilon_i \triangleq \Upsilon \bigcap E_i$ be the set of failed nodes in hyperedge $E_i \in \tau$ under failure pattern Υ. To repair all the failed nodes in Υ, we need $|\Upsilon|$ newcomers in total. All lost blocks, i.e., $\{\mathcal{B}_i : \Upsilon_i \neq \emptyset\}$, need to be regenerated in the corresponding newcomers. This can always be done provided that $|\Upsilon| \leq \rho$. In that case, we say that Υ is *repairable*.

Let us focus on one particular lost block, $\mathcal{B}_i$. Each newcomer of a failed node in Υ_i needs to get a copy of $\mathcal{B}_i$ from a certain helper node, which can be a surviving node or another newcomer that has already recovered the coded block $\mathcal{B}_i$. The cost for a newcomer to repair $\mathcal{B}_i$ is simply the block size, β_i, multiplied by the communication cost between the newcomer and its helper node. If there is only one node in Υ_i, then it is clear that the only newcomer, say node v, should choose a helper node u that minimizes the communication cost c_{uv}. If there are multiple nodes in Υ_i, the repair order will affect the total repair cost. To minimize the total repair cost for $\mathcal{B}_i$, a greedy algorithm, which is stated in Algorithm 1, can be used.

Algorithm 1: Repair process of coded block $\mathcal{B}_i$ in hyperedge E_i

1) Pick the minimum-weight edge $e=\{u,v\} \in G$, where $u \in E_i \backslash \Upsilon_i$ and $v \in \Upsilon_i$. Then the newcomer of node v chooses node u to be its helper node and downloads a copy of $\mathcal{B}_i$ from node u along the minimum-cost path.
2) Remove node v from Υ_i, i.e., $\Upsilon_i \leftarrow \Upsilon_i \backslash \{v\}$, which means that the newcomer of node v has already recovered $\mathcal{B}_i$ and is able to act as helper node of other newcomers that still need to recover $\mathcal{B}_i$.
3) Repeat Steps 1) and 2) until all the newcomers in E_i recover $\mathcal{B}_i$, i.e., $\Upsilon_i = \emptyset$.

Theorem 1. *For any given repairable failure pattern Υ, Algorithm 1 minimizes the cost of repairing $\mathcal{B}_i$, for all i such that $\Upsilon_i \neq \emptyset$.*

Proof. Replace all nodes $E_i \backslash \Upsilon_i$ by a virtual node s. For $v \in \Upsilon_i$, let $c_{sv} \triangleq \min\{c_{uv} : u \in E_i \backslash \Upsilon_i\}$. A weighted complete graph K can be constructed on the vertex set $\Upsilon_i \cup \{s\}$, where the edge weight is the corresponding communication cost. The repairing of $\mathcal{B}_i$ is equivalent to sending $\mathcal{B}_i$ from node s to all nodes in Υ_i. Therefore, the minimum cost of repairing $\mathcal{B}_i$ is equal to β_i times the weight of the minimum spanning tree of the graph K. Let $e_1, e_2, \ldots, e_{|\Upsilon_i|}$ be the sequence of edges of G chosen by Algorithm 1. Let $f_1, f_2, \ldots, f_{|\Upsilon_i|}$ be the sequence of edges of K chosen by the well-known Prim's algorithm [24, Chapter 23] for finding a minimum spanning tree on graph K with s being the initial vertex. It can be seen that the communication cost of e_i is equal to the weight of f_i for all i. Therefore, Algorithm 1 is optimal. □

Note that the repair processes of different coded blocks are independent, and thus can be executed in parallel. In repairing $\mathcal{B}_i$ under the failure pattern Υ, let the set of edges chosen by Algorithm 1 be denoted by T_i^{Υ}. Given a repair overlay τ and a block assignment vector $\mathbf{b}$, the total repair cost, normalized by the object size B, under failure pattern Υ is then given by

$$\tilde{c}_r(\tau, \mathbf{b}, \Upsilon) = \frac{1}{B} \sum_{i:\Upsilon_i \neq \emptyset} \sum_{\{u,v\} \in T_i^{\Upsilon}} c_{uv}\beta_i. \tag{1}$$

B. MDS-IFR Code Optimization

Our objective is to design the MDS-IFR code so as to minimize the system repair cost. If a non-repairable failure pattern Υ occurs, the whole data object will be decoded by downloading data from one of the retrieval sets and then re-encoded for storage in the newcomers. We assume that the

system is properly designed so that the probability of occurrence of a non-repairable failure pattern is small. Therefore, we focus on minimizing the expected repair cost per unit data, where the expectation is taken over all repairable patterns. We call it the *system repair cost* and denote it by $c_r(\tau, \mathbf{b})$. Given a repair overlay τ and a block assignment vector $\mathbf{b}$, it can be written as

$$c_r(\tau, \mathbf{b}) \triangleq \sum_{\Upsilon:\text{repairable}} p(\Upsilon)\tilde{c}_r(\tau, \mathbf{b}, \Upsilon), \tag{2}$$

where $p(\Upsilon)$ be the probability of occurrence of Υ, on the condition that the failure pattern Υ is repairable.

Let $\tilde{E}_1, \tilde{E}_2, \ldots, \tilde{E}_{\binom{n}{\rho+1}}$ be all the $(\rho+1)$-subsets of $\mathcal{V}$. We use a binary variable to indicate whether $\tilde{E}_i$ belongs to the hyperedge set $\mathcal{H}^\tau$ of a repair overlay τ:

$$x_i \triangleq \begin{cases} 0 & \text{if } \tilde{E}_i \notin \mathcal{H}^\tau, \\ 1 & \text{if } \tilde{E}_i \in \mathcal{H}^\tau. \end{cases}$$

Let $\mathbf{x} \triangleq [x_1, x_2, ..., x_{\binom{n}{\rho+1}}]$. We call it an *overlay selection vector*. To ensure that the degree of each vertex v in $\mathcal{V}$ is not larger than d, we have the following constraints

$$\sum_{i:v\in\tilde{E}_i} x_i \le d, \ \forall v \in \mathcal{V}. \tag{3}$$

Note that the binary vector $\mathbf{x}$ defines a repair overlay, which we denote it by $\tau(\mathbf{x})$.

As a coded block $\mathcal{B}_i \subseteq \mathcal{F}$ is assigned to hyperedge $\tilde{E}_i$ if and only if it would be contained in the overlay hypergraph, we therefore have the constraints

$$0 \le \beta_i \le Bx_i, \ i = 1, 2, \ldots, \binom{n}{\rho+1}. \tag{4}$$

For each storage node, it stores all the coded blocks associated with the hyperedges containing it. Thus the storage amount of node v, denoted by α_v, can be obtained as

$$\alpha_v = \sum_{i:v\in\tilde{E}_i} \beta_i, \ \forall v \in \mathcal{V}. \tag{5}$$

Note that considering data retrieval, α_v need not be greater than B due to the outer MDS code. To facilitate efficient uncoded repair, however, we allow α_v to exceed B. Given a repair overlay τ and a block assignment vector $\mathbf{b}$, we assume that there is a constraint on the *system storage cost*, denoted by $c_s(\tau, \mathbf{b})$, which is defined as the cost of storing one unit data object in DSS(n, ρ, d, k, w). Let C_s be the maximum allowable system storage cost. Then we have

$$c_s(\tau(\mathbf{x}), \mathbf{b}) \triangleq \frac{1}{B}\sum_{v=1}^{n} s_v\alpha_v \le C_s. \tag{6}$$

Note that C_s is a given constant, which constrains the total storage cost in the system. We do not impose any constraint on the storage amount of each storage node. If needed, that kind of constraints can be easily added, and our proposed algorithm, to be described in a later section, can still be applied without any modification.

Recall that the DSS(n, ρ, d, k, w) needs to satisfy the data retrieval requirement. There is a collection of retrieval sets, $\Psi = \{R_1, R_2, \ldots, R_w\}$. Part or all of these sets may be pre-determined based on considerations other than storage and repair costs. For example, if a data object is mainly needed by users in a specific geographical region, it would be more convenient if one or more retrieval sets are formed by storage nodes in that region, so that the response time for a user to download that object can be shortened. To provide more flexibility in our optimization framework, we allow $w_1 \le w$ retrieval sets be given while the remaining $w_2 = w - w_1$ retrieval sets are obtained by our optimization procedure. For the pre-determined retrieval sets, $R_1, R_2, \ldots, R_{w_1}$, we need to ensure that each of them stores at least B coded packets in $\mathcal{F}$. Therefore, we have the following constraints:

$$\sum_{i:\tilde{E}_i\cap R_j\neq\emptyset} \beta_i \ge B, \ \ j = 1, 2, \ldots, w_1. \tag{7}$$

It remains to determine the other w_2 retrieval sets. Denote the k-subsets of $\mathcal{V}$, excluding the pre-determined retrieval sets, by $Q_1, Q_2, \ldots, Q_W$, where $W \triangleq \binom{n}{k} - w_1$. To indicate whether Q_j is a retrieval set or not, we introduce a binary variable

$$y_j \triangleq \begin{cases} 0 & \text{if } Q_j \notin \Psi, \\ 1 & \text{if } Q_j \in \Psi. \end{cases}$$

Let $\mathbf{y} \triangleq [y_1, y_2, ..., y_W]$. We call it a *retrieval set selection vector*. To guarantee that there are w_2 more retrieval sets, we have

$$\sum_{j=1}^{W} y_j = w_2. \tag{8}$$

Note that $\mathbf{y}$ defines a collection of retrieval sets, which we denote it by $\Psi(\mathbf{y})$. Similar as before, we have

$$\sum_{i:\tilde{E}_i\cap Q_j\neq\emptyset} \beta_i \ge By_j, \ j = 1, 2, \ldots, W. \tag{9}$$

As mentioned before, our objective function is the system repair cost of DSS(n, ρ, d, k, w). Formally, the repair cost minimization problem can be stated as follows.

$$\text{Minimize } \ c_r(\tau(\mathbf{x}), \mathbf{b}) \triangleq \sum_{\Upsilon:\text{repairable}} p(\Upsilon)\tilde{c}_r(\tau(\mathbf{x}), \mathbf{b}, \Upsilon), \tag{10}$$

subject to

$$(3) - (9),$$

$$x_i \in \{0, 1\}, \ i = 1, 2, \ldots, \binom{n}{\rho+1}, \tag{11}$$

$$y_j \in \{0, 1\}, \ j = 1, 2, \ldots, W, \tag{12}$$

$$\beta_i \in \mathbb{N}, \ i = 1, 2, \ldots, \binom{n}{\rho+1}. \tag{13}$$

The optimization is an integer linear programming (ILP) problem, where $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{b}$ are the optimization variables.

VI. Repair-Storage Tradeoff

In our formulation, it is clear that there is a tradeoff between the system storage cost, c_s, and the system repair cost, c_r. To make this relationship more explicit, we introduce the following notions. For the ease of presentation, we assume that $w_1 = 0$ for the rest of this paper.

Definition 3 (Achievability)**.** *A cost pair (c_r^*, c_s^*) is B-achievable by the MDS-IFR code if given any data object of size B, there exists a repair overlay τ, a collection of retrieval sets Ψ, and a block assignment vector $\mathbf{b}$ such that $c_r(\tau, \mathbf{b}) \leq c_r^*$, $c_s(\tau, \mathbf{b}) \leq c_s^*$, and the data repair and retrieval requirements are satisfied.*

Note that β_i's must be integers no more than B. Therefore, the achievable region enlarges when B increases. It is therefore natural to define the asymptotic achievable region for arbitrarily large value of B:

Definition 4 (Asymptotic achievability)**.** *A cost pair (c_r^*, c_s^*) is asymptotically achievable by the MDS-IFR code if for any $\epsilon > 0$, there exists for sufficiently large B, a repair overlay τ, a collection of retrieval sets Ψ, and a block assignment vector $\mathbf{b}$ such that $c_r(\tau, \mathbf{b}) < c_r^* + \epsilon$, $c_s(\tau, \mathbf{b}) < c_s^* + \epsilon$, and the data repair and retrieval requirements are satisfied.*

The following result shows that the asymptotically achievable cost can be obtained by relaxing the integer constraint on $\mathbf{b}$:

Theorem 2. *Given any B and C_s, let $\mathbf{b}^* = [\beta_i^*]$, $\mathbf{x}^*$, and $\mathbf{y}^*$ be the solution to the repair cost minimization after relaxing the integer constraint on $\mathbf{b}$, and $c_r^* \triangleq c_r(\tau(\mathbf{x}^*), \mathbf{b}^*)$ be the corresponding system repair cost. The cost pair (c_r^*, C_s) is asymptotically achievable by the MDS-IFR code.*

Proof. First of all, note that c_r and c_s are invariant to scaling B and all β_i's by the same amount, no matter whether β_i's are integers or not. Suppose we scale up B and β_i's all by $\gamma > 1$. Then we round *up* all β_i's to the nearest integers. By (1) and (10), the new system repair cost is given by

$$\tilde{c}_r = \frac{1}{\gamma B} \sum_{\Upsilon:\text{repairable}} p(\Upsilon) \sum_{i:\Upsilon_i \neq \emptyset} \sum_{\{u,v\} \in T_i^\Upsilon} c_{uv}(\gamma\beta_i + z_i) \quad (14)$$

$$= \frac{1}{B} \sum_{\Upsilon:\text{repairable}} p(\Upsilon) \sum_{i:\Upsilon_i \neq \emptyset} \sum_{\{u,v\} \in T_i^\Upsilon} c_{uv}\beta_i + \frac{1}{\gamma B} \sum_{\Upsilon:\text{repairable}} p(\Upsilon) \sum_{i:\Upsilon_i \neq \emptyset} \sum_{\{u,v\} \in T_i^\Upsilon} c_{uv} z_i, \quad (15)$$

where $0 \leq z_i < 1$. Since γ can be arbitrarily large, the second term can always be made smaller than ϵ. Similarly, the new storage cost can be proven to be smaller than $C_s + \epsilon$ for sufficiently large γ. Therefore, (c_r^*, C_s) is asymptotically achievable. □

To identify the optimal tradeoff between system repair cost and system storage cost, we need to introduce the following two concepts:

Definition 5 (Pareto-optimality)**.** *A B-achievable cost pair (c_r^*, c_s^*) is called Pareto-optimal if and only if there does not exist other B-achievable cost pair (c_r, c_s) such that the following two conditions are satisfied:*

1) *$c_r \leq c_r^*$ and $c_s \leq c_s^*$, and*
2) *$c_r < c_r^*$ or $c_s < c_s^*$.*

Algorithm 2: Find the Pareto frontier

1) Generate a solution which minimizes c_r subject to the constraints (3)-(4), (7)-(9) and (11)-(13). Exit if no solution is found. Otherwise, let c_r^* be the optimal value for c_r.
2) Constrain c_r to be equal to c_r^*, and generate a solution which minimize c_s subject to constraints (3)-(4), (7)-(9) and (11)-(13). Let c_s^* be the optimal value for c_s. Output (c_r^*, c_s^*).
3) Replace the constraint $c_r = c_r^*$ (which was added in Step 2) by the constraint $c_s < c_s^*$.
4) Go to Step 1.

Roughly speaking, Pareto-optimal B-achievable cost pairs are "on the boundary" of the set of all B-achievable cost pairs. In fact, to characterize the set of B-achievable cost pairs, it is necessary and sufficient to characterize only the so-called Pareto frontier:

Definition 6 (Pareto frontier)**.** *The Pareto frontier is the set of all Pareto-optimal B-achievable cost pairs.*

Theorem 3. *The Pareto frontier is a finite set.*

Proof. Note that β_i's are non-negative integers. According to (4), they are all less than or equal to B. Since the other variables, x_i's and y_i's, are all binary, the solution space is finite. Hence, the Pareto frontier is finite too. □

Since the Pareto frontier is finite, it is possible to list all of them in finite time, which can be done by Algorithm 2.

Theorem 4. *All the cost pairs in the Pareto frontier can be listed by Algorithm 2 in finite time.*

Proof. By Theorem 3, there is a finite number of Pareto-optimal B-achievable cost pairs. Denote them by $(a_1, b_1), (a_2, b_2), \ldots, (a_M, b_M)$, where M is the cardinality of the Pareto frontier. Furthermore, let them be ordered so that $a_i < a_j$ and $b_i > b_j$ for $i < j$.

We claim that the first point output by Algorithm 2 is (a_1, b_1). To see this, note that it first minimizes c_r, without any constraint on c_s. The result so obtained must be $c_r^* = a_1$, for otherwise (a_1, b_1) would not be the first point in the Pareto frontier. Then in Step 2, it minimizes c_s, with the constraint $c_r = c_r^* = a_1$. The result c_s^* must be less than or equal to b_1, since (a_1, b_1) is B-acheivable. Moreover, c_s^* cannot be strictly less than b_1, for otherwise (a_1, b_1) is not Pareto-optimal. As a consequence, the first point (a_1, b_1) is output.

By the same argument, we can see that all points output by Algorithm 2 must be Pareto-optimal, and thus belong to the Pareto frontier. Assume the pair (a_i, b_i) has just been output. Algorithm 2 first minimizes c_r, with the constraint $c_s < b_i$. The result so obtained must be $c_r^* > a_i$, for otherwise, (a_i, b_i) is not a Pareto-optimal point. It must be equal to a_{i+1},

Algorithm 3: Find a repair overlay τ

Input: $\tilde{G} = (\mathcal{V}, \tilde{\mathcal{E}}), d, \rho$
Output: $\tau = (\mathcal{V}, \mathcal{H}^\tau)$

1) Compute the metric closure G of $\tilde{G}$.
2) Initialize τ with $\mathcal{H}^\tau \leftarrow \emptyset$, and $N_v^\tau \leftarrow 0$ for all $v \in \mathcal{V}$. (Note that N_v^τ represents the degree of vertex v in τ.)
3) Sort all the $(\rho+1)$-subsets of $\mathcal{V}$ in ascending order of MST weight and get the sequence $\tilde{E}_1, \tilde{E}_2, \ldots, \tilde{E}_{\binom{n}{\rho+1}}$.
4) **for** $i = 1$ to $\binom{n}{\rho+1}$ **do**
   if $N_v^\tau < d$ for all $v \in \tilde{E}_i$ **then**
   $\mathcal{H}^\tau \leftarrow \mathcal{H}^\tau \cup \{\tilde{E}_i\}$
   $N_v^\tau \leftarrow N_v^\tau + 1$ for all $v \in \tilde{E}_i$
   end
   end
5) Return $\tau = (\mathcal{V}, \mathcal{H}^\tau)$.

Algorithm 4: Find a collection of retrieval sets Ψ=RS$(\mathcal{V}, \mathcal{H}^\tau, k, w)$

Input: $\tau = (\mathcal{V}, \mathcal{H}^\tau), k, w$
Output: Ψ

1) **if** $(\mathcal{V} = \emptyset) \bigvee (|\Psi| = w)$ **do**
   return Ψ
   end
2) Find $u \in \mathcal{V}$ that hits the maximum number of hyperedges in $\mathcal{H}^\tau$.
3) Let $\tau' = (\mathcal{V}', \mathcal{H}^{\tau'})$ be the hypergraph obtained from $\tau = (\mathcal{V}, \mathcal{H}^\tau)$ by removing u from $\mathcal{V}$ and all hyperedges containing u from $\mathcal{H}^\tau$.
4) $\Psi \leftarrow u \bigoplus$ RS$(\mathcal{V}', \mathcal{H}^{\tau'}, k-1, w)$
5) **if** $|\Psi| < w$ **do**
   $\Psi \leftarrow \Psi \bigcup$ RS$(\mathcal{V}', \mathcal{H}^{\tau'}, k, w - |\Psi|)$
   end
6) Return Ψ

for otherwise (a_{i+1}, b_{i+1}) is not in the Pareto frontier. Next, Algorithm 2 minimizes c_s, with the constraint $c_r = a_{i+1}$. The result will then be $c_s^* = b_{i+1}$. Therefore, the next Pareto-optimal pair (a_{i+1}, b_{i+1}) is output.

As a result, all points in the Pareto frontier will be output. The algorithm terminates when no more Pareto-optimal points can be found. □

We remark that Algorithm 2 cannot be replaced by solving a family of weighted sum minimization problems (with different weights), since not all Pareto optimal cost pairs lie on the boundary of the convex hull of all achievable cost pairs.

VII. A Heuristic Solution

Our repair cost minimization problem is a joint repair overlay, retrieval sets, and block assignment optimization problem. In theory, it can be solved by ILP. For large network size, however, ILP is too time consuming due to the fast-growing problem dimension. In this section, we present an efficient heuristic to solve the problem.

Our heuristic algorithm is divided into the following three steps:

1) Determine the repair overlay, τ, or equivalently, the overlay selection vector $\mathbf{x}$.
2) Determine the collection of retrieval sets, Ψ, or equivalently, the retrieval set selection vector, $\mathbf{y}$.
3) Determine the block assignment vector, $\mathbf{b}$.

First, we determine the repair overlay $\tau = (\mathcal{V}, \mathcal{H}^\tau)$ by a greedy approach. We examine all $\binom{n}{\rho+1}$ possible hyperedges that could be put into $\mathcal{H}^\tau$. Let them be $\tilde{E}_1, \tilde{E}_2, \ldots, \tilde{E}_{\binom{n}{\rho+1}}$. Let G_i be the subgraph of the metric closure G induced by the vertices in $\tilde{E}_i$. The cost of the minimum spanning tree of G_i is called the *MST weight* of $\tilde{E}_i$. At each step, we choose an hyperedge, not previously chosen, with the smallest MST weight while obeying the degree constraint. Such an hyperedge is then added to $\mathcal{H}^\tau$. The procedure then repeats. We formally state our method as Algorithm 3.

Let m be the number of edges in the underlying storage network $\tilde{G} = (\mathcal{V}, \tilde{\mathcal{E}})$. The complexity of Algorithm 3 is $O(n^2 \log n + mn + \rho n^{\rho+1}(\rho + \log n))$, since computing the metric closure G of $\tilde{G}$ by Johnson's algorithm has the complexity of $O(n^2 \log n + mn)$, finding the MST weight of the $\binom{n}{\rho+1}$ hyperedges has the complexity of $O(\rho^2 n^{\rho+1})$, the sorting in the second step has complexity of $O(\rho n^{\rho+1} \log n)$ and the third step has complexity of $O(n^{\rho+1})$. Since $\rho \geq 1$, the complexity of Algorithm 3 can be simplified as $O(mn + \rho n^{\rho+1}(\rho + \log n))$.

Next, we need to find Ψ, given a fixed repair overlay τ obtained in the previous step. According to the structure of the MDS-IFR code, we know that the coded blocks associated with two different hyperedges are distinct. For this reason, we require the k storage nodes in a retrieval set jointly hit as many hyperedges of τ as possible. In other words, we require Ψ be the collection of the first w k-subsets of $\mathcal{V}$ that hit the maximum number of hyperedges of τ. To find it, we use a recursive approach, which is formally stated as Algorithm 4. Note that in Step 4 of Algorithm 4, we use $u \bigoplus \Psi$, where u is a vertex and Ψ is a collection of vertex sets, to denote the operation of adding u to each set in Ψ. For example, $v_1 \oplus \{\{v_2, v_4\}, \{v_3, v_6\}\} = \{\{v_1, v_2, v_4\}, \{v_1, v_3, v_6\}\}$.

We remark that Algorithm 4 is for the case where $w_1 = 0$. If there are $w_1 > 0$ pre-determined retrieval sets, Algorithm 4 can be applied with a very minor modification. Before a k-subset of $\mathcal{V}$ is added to the collection of retrieval sets Ψ, the algorithm first check whether that k-subset happens to be one of the pre-determined retrieval sets. It would be added if and only if it is not one of them. The procedure repeats until $w - w_1$ retrieval sets have been added to Ψ.

The complexity of Step 2 in Algorithm 4 is $O(n|\mathcal{H}^\tau|)$, which is the same as $O(\frac{n^2 d}{\rho+1})$, since $|\mathcal{H}^\tau| \leq \frac{nd}{\rho+1}$. Step 2 would be implemented k times to find a retrieval set containing k vertices, and Algorithm 4 needs to find w retrieval sets. Thus, the complexity of Algorithm 4 is $O(\frac{wkn^2 d}{\rho+1})$.

Last, we need to find $\mathbf{b}$, given a fixed repair overlay τ and a fixed collection of retrieval sets Ψ. This can be done by solving the ILP problem while fixing $\mathbf{x}$ and $\mathbf{y}$ to the values corresponding to τ and Ψ, respectively. Alternatively, we can

solve the LP problem by relaxing the integer constraint on $\mathbf{b}$ if we want to minimize the asymptotically achievable cost.

For our ILP formulation of the repair cost minimization problem, the number of variables is $2\binom{n}{\rho+1}+\binom{n}{k}$ and the number of constraints is $\binom{n}{\rho+1}+n+w+2$. If the repair overlay and the collection of retrieval sets are fixed, the number of variables can be reduced to $|\mathcal{H}^\tau|$ while the number of constraints can be reduced to $|\mathcal{H}^\tau|+w+1$. Since $|\mathcal{H}^\tau| \leq nd/(\rho+1)$ and $d \leq n-\rho+1$, the number of variables and constraints of the ILP problem can be reduced from $O(n^{\rho+1}+n^k)$ and $O(n^{\rho+1})$, both to $O(n^2)$.

To conclude, the heuristic method consists of three steps. The first two steps have complexities $O(mn+\rho n^{\rho+1}(\rho+\log n))$ and $O(\frac{wkn^2d}{\rho+1})$, respectively. For practical scenarios, ρ is a small constant, typically equal to 1 or 2. In theory, linear programming can be solved in polynomial time. Therefore, regarding ρ as a constant, the overall computational complexity of the heuristic method is polynomial in n.

Example: Consider a 5-node ring, $\tilde{G}$, shown in Fig. 5(a). The number associated with an edge denotes the single-hop cost between its two endpoints. Fig. 5(b) shows G, the metric closure of $\tilde{G}$, where the number associated with an edge is the corresponding communication cost. Suppose the storage network is able to tolerate double failures, i.e., $\rho=2$, and the degree constraint is $d=3$. We need to consider all hyperedges whose cardinality is equal to $\rho+1=3$, i.e., $\tilde{E}_1=\{1,2,3\}, \tilde{E}_2=\{3,4,5\}, \tilde{E}_3=\{1,2,5\}, \tilde{E}_4=\{2,3,4\}, \tilde{E}_5=\{1,2,4\}, \tilde{E}_6=\{1,3,4\}, \tilde{E}_7=\{1,4,5\}, \tilde{E}_8=\{2,3,5\}, \tilde{E}_9=\{2,4,5\}, \tilde{E}_{10}=\{1,3,5\}$. Their MST weights are 5, 5, 6, 6, 7, 7, 8, 9, 9, 10, respectively. According to Algorithm 3, the hyperedges $\tilde{E}_1, \tilde{E}_2, \tilde{E}_3, \tilde{E}_4$ and $\tilde{E}_7$ are successively added into $\mathcal{H}^\tau$. The resulting repair overlay τ is shown in Fig. 5(c). Furthermore, suppose that $k=3$ and $w=6$. According to Algorithm 4, we can obtain a collection of retrieval sets $\Psi=\{R_1=\{1,3,2\}, R_2=\{1,3,4\}, R_3=\{1,3,5\}, R_4=\{1,4,2\}, R_5=\{1,4,5\}, R_6=\{1,2,5\}\}$.

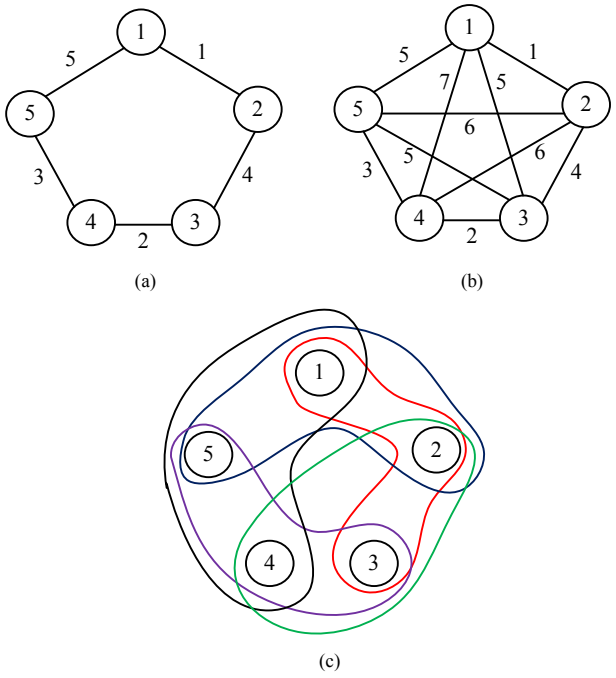


Fig. 5. An example of finding a repair overlay τ and a collection of retrieval sets Ψ in a given graph $\tilde{G}$.

VIII. Simulation Results

In this section, we consider heterogeneous storage systems. We compare the optimal tradeoff between system storage cost and system repair cost that can be achieved by the MDS-IFR code with that achieved by the regenerating code. Moreover, we compare the minimum system repair cost that can be achieved by the MDS-IFR code with that achieved by the regenerating code for different network size. Here, we use the term "regenerating code" to refer to any code that achieve points on the tradeoff curve under the regular model.

In our simulations, both the storage cost vector $\mathbf{s}=[s_i]$ and the single-hop cost matrix $\tilde{\mathbf{C}}=[\tilde{c}_{ij}]$ are randomly generated. For the storage system whose size is less than or equal to 20, each entry in $\mathbf{s}$ and $\tilde{\mathbf{C}}$ is an integer selected from the uniform distribution on the interval $[0,50]$. For the storage system whose size is larger than 20, each entry in $\mathbf{s}$ and $\tilde{\mathbf{C}}$ is an integer selected from the uniform distribution on the interval $[0,100]$. We assume that the probabilities of occurrence of all repairable failure patterns are the same.

Consider a distributed storage system with parameters: $n=10$, $\rho=2$, $d=3$, $k=3$. For the MDS-IFR code, all Pareto-optimal B-achievable cost pairs (c_s^*, c_r^*) can be obtained by running Algorithm 2 and solving the corresponding ILP problems. The curve connecting all Pareto-optimal B-achievable cost pairs is the optimal tradeoff between system storage cost and system repair cost that can be achieved by the MDS-IFR code, as shown in Fig. 6. For the regenerating code, there exists a fundamental tradeoff between the storage amount per node, α, and the amount of data downloaded from each surviving node when repairing a failed node, β. Based on the tradeoff between α and β, if each newcomer downloads data along the d paths with the least communication costs, the optimal tradeoff between system storage cost and system repair cost can be obtained. From Fig. 6, it can be seen that, compared with the regenerating code, under the same data retrieval requirement, i.e., $w=\binom{n}{k}=\binom{10}{3}=120$, the system repair cost that achieved by the MDS-IFR code can be reduced if the system storage cost are increased. However, if the data retrieval requirement are properly relaxed, i.e., $w=10$, both the system repair cost and system storage cost achieved by the MDS-IFR code can be reduced.

For the heuristic of minimizing repair cost in the irregular model by using the MDS-IFR code, to illustrate the integrality gap, we consider a distributed storage system with parameters: $n=10, d=6, k=4$, $w=\binom{10}{4}$, and $\rho=1$. We increase the data object size B from 10 to 30, with step size 10. For each value of B, we increase the maximum system storage cost per unit data object, C_s, from 80 to 100 and solve the corresponding ILP. The tradeoff curves between system storage cost, c_s, and system repair cost, c_r, are plotted in Fig. 7. We can observe that the gap between the solution of the ILP and of its relaxation is tiny and decreases with the growing of the data object size B. Thus, to improve the efficiency of simulation, we solve the LP problem by relaxing

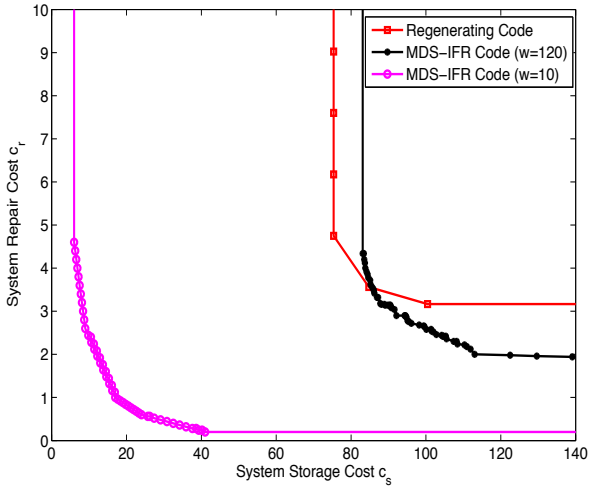


Fig. 6. Optimal tradeoff between system storage cost and system repair cost ($n = 10, d = 4, k = 3, \rho = 2$, and $B = 20$).

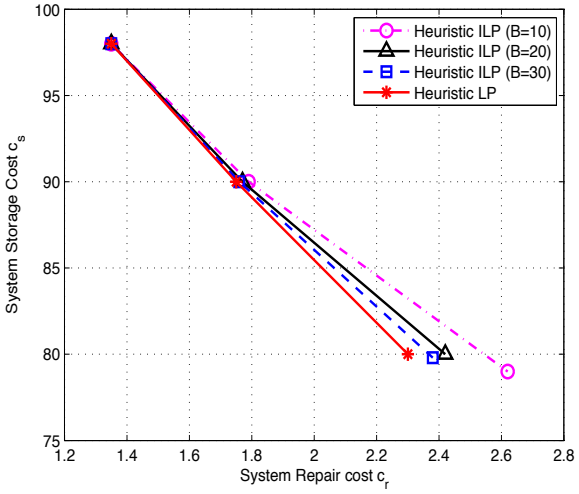


Fig. 7. Tradeoff curves between system storage cost and system repair cost ($n = 10, d = 6, k = 4, w = \binom{10}{4} = 210$, and $\rho = 1$).

the integer constraints on $\mathbf{b}$ to minimize the asymptotically achievable repair cost in our simulation.

We next compare the minimum system repair cost of the MDS-IFR code with that of the regenerating code for different network size. The maximum allowable system storage cost per unit data object C_s is set to a sufficiently large value, 1000000. The simulation for each value of n is averaged over 100 runs. For small storage networks, from Fig. 8, it can be seen that if the number of retrieval sets is equal to $\binom{n}{k}$, the minimum system repair cost that can be achieved by the MDS-IFR code is roughly reduced at least by 20%. Moreover, the asymptotically achievable minimum system repair cost found by our heuristic is near-optimal. The gap between heuristic solution and optimal solution is at most 6%. If the data retrieval requirement is relaxed, for example the number of retrieval sets is reduced to $w = 50$, the asymptotically achievable minimum system repair cost achieved by the MDS-IFR code can be reduced at least by 70%. Since the constraints of data repair and data retrieval are relaxed in the irregular model, it is not surprising that there exists a performance gain. Nevertheless, it demonstrates that there is a large room for improvement if the regular model is refined. This is particularly relevant when the networking environment is heterogeneous.

To gain more understanding about the computational efficiency of our heuristic, we increase the network size and measure its running time. The machine employed for simulation is a Dell computer with an Intel(R) Core(TM)2 Quad CPU running at 3 GHz with 4 GB RAM. The operating system is Windows 7, and the computer is a 32-bit machine. The simulation programs were written in MATLAB. Our method requires solving LP and ILP problems. These tasks were done by a free linear integer programming solver called "lp_solve", which was called from our MATLAB program. In our simulation, the system parameters are set as follows: $d = 5, k = 4, \rho = 2, w = 100, Cs = 1000000$ and $B = 50$. The simulation for each value of n is averaged over 100 runs. The minimum system repair cost obtained by using our heuristic for different network size is shown in Fig. 9. The

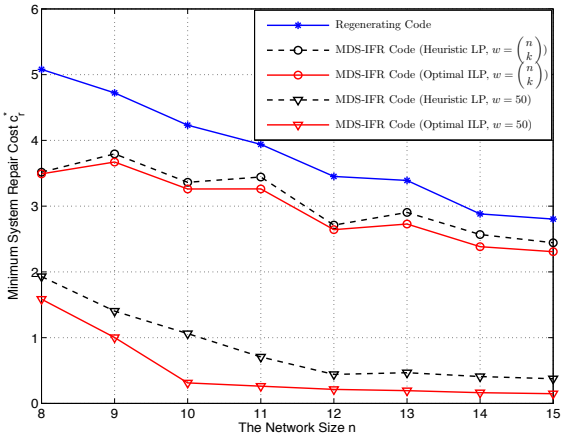


Fig. 8. The minimum system repair cost for different network size ($d = 4, k = 3, \rho = 2$, and $B = 30$).

average running time of the three steps of our heuristic for a given problem instance is also recorded in Fig. 10. From Fig. 10, it can be seen that the most time consuming step of our heuristic is solving the LP problem after determining the repair overlay and retrieval sets. Moreover, the total time consumed by our heuristic is less than 4 minutes when the network size is less than 150.

IX. Conclusion

Due to the emergence of heterogeneous cloud storage systems, we generalize the concept of the FR code and propose the IFR code. A key property of the FR code is its uncoded repair process. This simple repair mechanism minimizes the repair bandwidth and the disk access bandwidth simultaneously, without any computational cost. The IFR code preserves this nice property. Moreover, its irregular structure allows the repair pattern and the storage amount of each node to be different, thus enabling the cloud system to be optimized according to network heterogeneity including different storage costs of the storage nodes and different communication costs

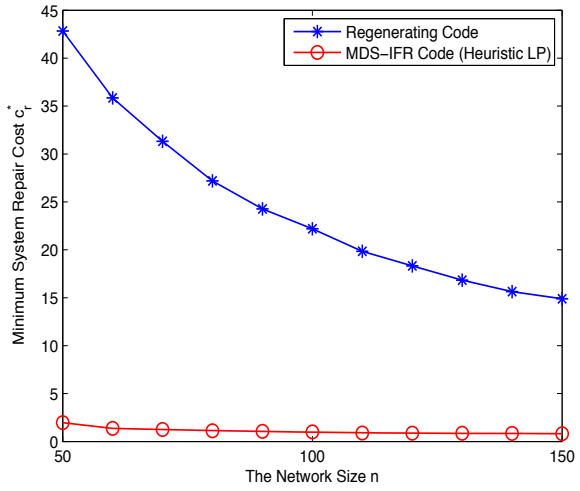


Fig. 9. The minimum system repair cost for different network size ($d = 5, k = 4, \rho = 2, w = 100$, and $B = 50$).

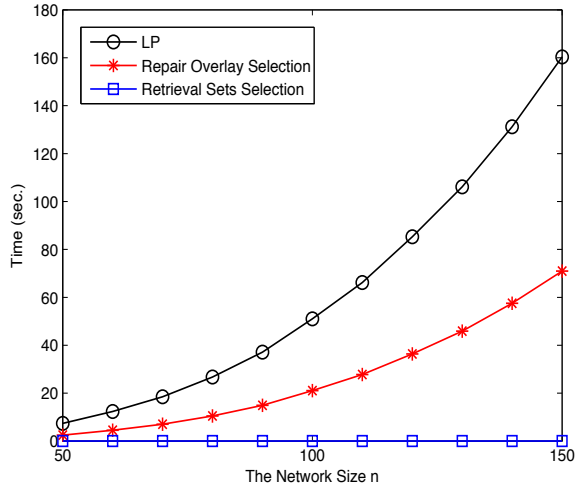


Fig. 10. The average running time of the three steps of heuristic for different network size ($d = 5, k = 4, \rho = 2, w = 100$, and $B = 50$).

of the links. To determine the repair pattern, which we call the repair overlay, and the storage allocation, we formulate the whole problem based on a new irregular model, with the aim of minimizing the system repair cost by properly designing the MDS-IFR code and the retrieval sets. For large networks, we decompose the repair cost minimization problem into three subproblems: repair overlay selection, retrieval sets selection, and block assignment, and propose a heuristic solution. For small network sizes, it is shown to be nearly optimal by comparing it with the optimal ILP method.

While the optimization framework established in this paper concerns mainly on system repair cost, it can be modified to include other system objectives and extended by incorporating more resource constraints. On the other hand, as it is based on the MDS-IFR code, it provides very low repair cost at the expense of higher storage overhead. If higher storage efficiency is needed in some applications, other codes will be needed (using at the expense of higher repair cost or computing cost). This problem is beyond the scope of this paper. Nevertheless, we have demonstrated how optimization techniques can be used to construct good codes, providing insights and new methodology on how to design future heterogeneous cloud storage systems.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions to improve the quality of the paper.

REFERENCES

[1] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. IEEE Int. Conf. on Computer Commun. (INFOCOM)*, Anchorage, Alaska, May 2007, pp. 2000–2008.

[2] K. V. Rashmi, N. B. Shah, and P. V. Kumar, "Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5227–5239, Aug. 2011.

[3] C. Suh and K. Ramchandran, "Exact-repair MDS codes for distributed storage using interference alignment," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Austin, Jun. 2010, pp. 161–165.

[4] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Explicit codes minimizing repair bandwidth for distributed storage," in *IEEE Information Theory Workshop (ITW)*, Cairo, Jan. 2010, pp. 1–5.

[5] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A survey on network codes for distributed storage," *Proc. IEEE*, vol. 99, no. 3, pp. 476–489, Mar. 2011.

[6] I. Tamo, Z. Wang, and J. Bruck, "Zigzag codes: MDS array codes with optimal rebuilding," *IEEE Trans. Inf. Theory*, vol. 59, no. 3, pp. 1597–1616, Mar. 2013.

[7] S. El Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in *Proc. 48th Annual Allerton conference on commun. control and computing*, Monticello, IL, Sep. 2010, pp. 1510–1517.

[8] J. C. Koo and J. T. Gill, "Scalable constructions of fractional repetition codes in distributed storage systems," in *Proc. 49th Annual Allerton conference on commun. control and computing*, Monticello, IL, Sep. 2011, pp. 1366–1373.

[9] S. Pawar, N. Noorshams, S. El Rouayheb, and K. Ramchandran, "DRESS codes for the storage cloud: Simple randomized constructions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Saint Petersburg, 2011, pp. 2338–2342.

[10] O. Olmez and A. Ramamoorthy, "Repairable replication-based storage systems using resolvable designs," in *Proc. 50th Annual Allerton conference on commun. control and computing*, Monticello, IL, Oct. 2012, pp. 1174–1181.

[11] S. Anil, M. K. Gupta, and T. A. Gulliver, "Enumerating some fractional repetition codes," arXiv:1303.6801 [cs.IT], Mar. 2013.

[12] D. S. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2771–2775.

[13] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin, "On the locality of codeword symbols," *IEEE Trans. Inf. Theory*, vol. 58, no. 11, pp. 6952–6934, Nov. 2012.

[14] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proc. IEEE Int. Conf. on Computer Commun. (INFOCOM)*, Shanghai, China, Apr. 2011, pp. 1215–1223.

[15] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: an architecture for global-scale persistent storage," in *Proc. 9th Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, Cambridge, MA, Nov. 2000, pp. 190–201.

[16] S. Pawar, S. E. Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a distributed caching based video-on-demand system," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, Nov. 2011, pp. 1783–1787.

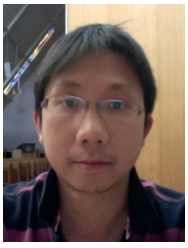
[17] N. Golrezaei, A. G. Dimakis, and A. F. Molisch, "Wireless device-to-device communications with distributed caching," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Cambridge, MA, Jul. 2012, pp. 2781–2785.

[18] D. Leong, A. G. Dimakis, and T. Ho, "Distributed storage allocations," *IEEE Trans. Inf. Theory*, vol. 58, no. 7, pp. 4733–4752, Jul. 2012.

[19] S. Akhlaghi, A. Kiani, and M. R. Ghanavati, "Cost-bandwidth tradeoff in distributed storage systems," *Computer Communications*, vol. 33, no. 17, pp. 2105–2115, Nov. 2010.
[20] Q. Yu, K. W. Shum, and C. W. Sung, "Minimization of storage cost in distributed storage systems with repair consideration," in *Proc. IEEE Telecommunications Conference (GLOBECOM)*, Houston, Texas, Dec. 2011, pp. 1–5.
[21] J. Li, S. Yang, X. Wang, and B. Li, "Tree-structured data regeneration in distributed storage systems with regenerating codes," in *Proc. IEEE Int. Conf. on Computer Commun. (INFOCOM)*, San Diego, Mar. 2010, pp. 1–9.
[22] M. Gerami, M. Xiao, and M. Skoglund, "Optimal-cost repair in multi-hop distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Saint Pertersburg, Jul. 2011, pp. 1437–1441.
[23] T. Ernvall, S. E. Rouayheb, C. Hollanti, and H. V. Poor, "Capacity and security of heterogeneous distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Jul. 2013, pp. 1247–1251.
[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA: The MIT Press, 2009.
[25] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. ACM Symp. on Operating Systems Principles (SOSP)*, New York, Oct. 2003, pp. 29–43.
[26] F. J. Macwilliams and N. J. A. Sloane, *The theory of error-correcting codes*. New York: North-Holland, 1977.
[27] C. W. Sung and X. Gong, "A zigzag-decodable code with the MDS property for distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Jul. 2013, pp. 341–345.

Quan Yu received her B.S. degree in Electronic Information Engineering from Huazhong Normal University, Wuhan, China, in 2009, and is currently pursuing the Ph.D degree at the Department of Electronic Engineering, City University of Hong Kong. Her research interests include cloud storage systems and network coding.

Chi Wan Sung (M'98) received his B.Eng, M.Phil, and Ph.D. degrees in Information Engineering from the Chinese University of Hong Kong in 1993, 1995, and 1998, respectively. He joined the faculty at City University of Hong Kong in 2000, and is now an Associate Professor with the Department of Electronic Engineering. He is an Adjunct Associate Research Professor at University of South Australia, and is on the editorial boards of the Transactions on Emerging Telecommunications Technologies (ETT) and of the ETRI journal. His research interests include wireless communications, network coding, cloud storage systems, and algorithms and complexity.



Terence H. Chan received his B.Sc (Math), Master's and Ph.D. degrees in Information Engineering in 1996, 1998 and 2000 respectively, all from The Chinese University of Hong Kong. In 2001, he was a visiting assistant professor in the Department of Information Engineering at the same university. From February 2002 to June 2004, he was a Postdoctoral Fellow at the Department of Electrical and Computer Engineering at the University of Toronto. He was an assistant professor in University of Regina from 2004-2006. He is currently an Associate Professor in Institute for Telecommunications Research at University of South Australia. He received the Croucher Foundation Fellowship and Sir Edward Youde Fellowship in 2002 and 2000 respectively.

He is the IEEE Information Theory Society, Joint South Australia/ACT/VIC/NSW/QLD Sections Chapter Chair. He serves as the Technical Co-Chair for the 2011 and 2015 IEEE International Symposium On Network Coding.