Locally Repairable Codes over a Network

Quan Yu
Department of Electronic Engineering
City University of Hong Kong
Email: Q.Yu@my.cityu.edu.hk

Chi Wan Sung
Department of Electronic Engineering
City University of Hong Kong
Email: albert.sung@cityu.edu.hk

Terence H. Chan
Institute for Telecommunications Research
University of South Australia
Email: terence.chan@unisa.edu.au

***Abstract*—Locally repairable (LR) codes are used in distributed storage systems to minimize the number of storage nodes involved in node repair. While existing constructions of LR codes do not take the topology of the storage network into account, this work focuses on designing LR codes over a network. A new concept called node locality is introduced. It is shown that the decision problem of determining whether a binary linear LR code exists, subject to the constraints of code rate, symbol locality, node locality, and repair cost, is NP-complete. The corresponding optimization version, which aims to maximize the code rate, is also considered. It is proved that the problem can be reduced to the minimum k-set cover problem, and can be solved in polynomial time for the special case where the symbol locality is one. For the general case where the symbol locality is greater than or equal to two, the problem is NP-hard and can be approximately solved by a greedy algorithm.**

I. Introduction

In distributed storage systems (DSS), data are usually stored in several geographically separated storage nodes, which are network-connected. Storage nodes in a DSS are generally unreliable and subject to failures. To maintain the reliability of a DSS, redundancy must be introduced using different coding techniques. Besides, a data repair mechanism is essential to cope with node failures, which are the norm rather than the exception.

Replication and erasure coding are commonly used to introduce redundancy. In replication schemes, data are duplicated and stored in multiple storage nodes. This method, though simple, has high storage overhead. Compared with replication, erasure coding is more efficient in storage space. However, when a storage node fails, erasure coding requires the retrieval of the whole original file to recover the lost data, which incurs considerable network traffic and lots of input/output (I/O) operations. To minimize the traffic required in repairing a failed storage node, called repair bandwidth, Dimakis et al. proposed a new class of codes called regenerating codes in [1] based on the concept of network coding. Although the use of regenerating codes can reduce the repair bandwidth, a replacement node needs to contact a large number of surviving nodes, which generally causes high I/O bandwidth.

The concept of repair locality, which is defined as the number of storage nodes involved in repairing a failed node, was introduced independently in [2] and [3]. The codes aiming at reducing the repair locality, while still guaranteeing a low repair bandwidth, is called *locally repairable (LR) codes*. For scalar linear codes, a tradeoff between locality and code distance was established in [2]. Given a repair locality and storage amount per node, an upper bound on the minimum distance of any (linear or nonlinear) vector LR code was given in [4]. An LR code is said to be optimal if its distance achieves the bound derived in [4]. Explicit constructions of optimal LR codes can be found in [5], [6]. Using regenerating codes as the local codes, some LR codes were constructed in [7].

In most of the existing works of LR codes, all storage nodes in a DSS are simply assumed to be fully connected with one another and all communication links in the storage network are assumed to be identical. In general, however, distributed storage systems are *heterogeneous* in nature, meaning that the communication links between each pair of storage nodes may have different characteristics in term of bandwidth and communication cost. For example, storage nodes located in the same geographical area can usually communicate with higher bandwidth and lower cost than those belonging to different areas. Furthermore, it is also possible that some storage nodes are not directly connected. While heterogeneous DSS has been considered in [8], [9], [10], [11], to the best of our knowledge, there is no general result on how to design LR codes over a heterogeneous storage network. In [12], the topology of storage network was first taken into account when designing LR codes. Duality results between LR codes and index coding have been established in [12], [13].

In this paper, we focus on designing LR codes over a heterogeneous network. Since the storage nodes may not be fully connected, a new concept called *node locality* is introduced. We consider the decision problem of determining whether a binary linear code exists, subject to the constraints of code rate, symbol locality, node locality, and repair cost. When the symbol locality $r \geq 2$, we prove that the decision problem of a binary linear LR code is NP-Complete. We also consider the corresponding optimization version, which aims to maximize the code rate. It is shown that the code rate maximization of a binary linear LR code with symbol locality r can be reduced to the $\text{MIN-}(r+1)\text{-SETCOVER}$ problem in polynomial time, and can be solved optimally in polynomial time for the special case when $r = 1$. For the general case when $r \geq 2$, the problem is NP-hard and a greedy algorithm is provided to approximate the optimal solution.

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).

II. LR Codes over a Network

Consider the case where we want to store some data to a distributed storage system (DSS), which consists of n storage nodes connected by a network with arbitrary connectivity. The data to be stored in the DSS are assumed to be symbols from an alphabet Σ of size $q \geq 2$. Let there be k source symbols to be stored, which are denoted by a k-dimensional vector, $\boldsymbol{x} \triangleq (x_1, x_2, \ldots, x_k)$. We represent a DSS over a network by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} \triangleq \mathcal{N} \triangleq \{1, 2, \ldots, n\}$ is the set of storage nodes and $\mathcal{E}$ is the edge set. Each edge $(i, j) \in \mathcal{E}$ has an associated weight w_{ij}, which represents the cost of transmitting one symbol from vertex i to vertex j, or from vertex j to vertex i.

A storage code $\mathcal{C}$ is defined by an *encoding* function $f : \Sigma^k \to \Sigma^n$, which maps a k-dimensional vector to an n-dimensional vector:

$$f(\boldsymbol{x}) = \boldsymbol{y} \triangleq (y_1, y_2, \ldots, y_n), \tag{1}$$

where $k \leq n$. Define $\mathcal{C}$ to be the image of f. The *rate* of the code $\mathcal{C}$ is defined as k/n. After encoding the source data by a storage code $\mathcal{C}$, the n coded symbols are stored in a DSS such that each storage node stores one symbol. Without loss of generality, we assume that node $i \in \mathcal{V}$ stores the symbol $y_i \in \boldsymbol{y}$.

Given any vector $\boldsymbol{v}$ and any subset $\mathcal{I}$ of the set of natural numbers, we let $\boldsymbol{v}(\mathcal{I})$ be the sub-vector of $\boldsymbol{v}$ obtained by removing all elements of $\boldsymbol{v}$ except those whose indices are in $\mathcal{I}$. When a storage node, say node i, fails, it is important that the failed node can be easily regenerated in the sense that it is not necessary to first decode $\boldsymbol{x}$ and then re-encode it to get y_i. A sub-vector of symbols $\boldsymbol{y}(\mathcal{I})$, $\mathcal{I} \subset \mathcal{N}$ is said to be *regenerable* from $\mathcal{J} \subset \mathcal{N} \setminus \mathcal{I}$ if there is a *regenerating* function $h_{\mathcal{I},\mathcal{J}} : \Sigma^{|\mathcal{J}|} \to \Sigma^{|\mathcal{I}|}$ such that $h(\boldsymbol{y}(\mathcal{J})) = \boldsymbol{y}(\mathcal{I})$ for all possible codewords $\boldsymbol{y}$'s.

We consider an important class of storage codes called *locally repairable (LR) code*. A key concept of LR code is that any symbol of a codeword is associated with a measure called *symbol locality*. Specifically, a coded symbol y_i is said to have *symbol locality* r_i if there exists a set $\mathcal{H} \subseteq \mathcal{N} \setminus \{i\}$ of cardinality $|\mathcal{H}| = r_i$ such that y_i is regenerable from $\boldsymbol{y}(\mathcal{H})$. The set $\mathcal{H}$ is called a *helper set* of node i.

To repair the failed node i, we need to regenerate y_i by computing the regenerating function $h_{\{i\},\mathcal{H}}$, where $\mathcal{H}$ is a helper set of node i. To do so, node i needs to receive information from the nodes in $\mathcal{H}$ through the network. As mentioned before, transmitting one symbol along an edge (j, k) involves a cost of w_{jk}. The total communication cost needed for node i to regenerate y_i is called the *repair cost* of y_i. In the communication process, we assume not only routing, but network coding can also be used. In other words, a node in the network is able to compute new symbols based on its stored symbol and incoming symbols, and send these new symbols to its neighboring nodes. The total number of nodes, ϕ_i, involved in this communication process is called the *node locality* of y_i. It is clear that $\phi_i \geq r_i$ for all i since some storage nodes may just forward its received symbols during the regeneration of symbol y_i.

A storage code $\mathcal{C}$ over a graph $\mathcal{G}$ is said to be a *locally repairable (LR) code* of parameters (r, ϕ, c) if each of its coded symbol has symbol locality no more than r, node locality no more than ϕ, and repair cost no more than c. If Σ is a finite field, and the encoding function, decoding functions and regenerating functions are all linear functions, we say that the code is *linear*.

III. Linear LR Code and Repair Group

From now on, we consider only linear LR code $\mathcal{C}$ over a finite field of size q. Its orthogonal complement, also called its dual code, is denoted by $\mathcal{C}^\perp$. We say that $\mathcal{H}$ is a *minimal* helper set of storage node i if y_i cannot be regenerated from any proper subset of $\mathcal{H}$. If $\mathcal{H}$ is minimal, y_i can be written as

$$y_i = \sum_{j \in \mathcal{H}} a_j y_j, \tag{2}$$

where $a_j \neq 0$. Then y_k, where $k \in \mathcal{H}$ and $k \neq i$, can be written as

$$y_k = a_k^{-1} y_i - a_k^{-1} \sum_{j \in \mathcal{H} \setminus \{k\}} a_j y_j. \tag{3}$$

Since the coefficients are all non-zero, $\mathcal{H} \cup \{i\} \setminus \{k\}$ is a helper set of node k. Furthermore, this helper set is minimal, for otherwise y_k can be regenerated from a proper subset of $\mathcal{H} \cup \{i\} \setminus \{k\}$ and $\mathcal{H}$ cannot be a minimal helper set of node i. For this reason, we say that $\mathcal{H} \cup \{i\}$ is a repair group, which is defined formally below:

Definition 1 (Repair group). *Given an n-dimensional linear code $\mathcal{C}$, a subset $\mathcal{R}$ of $\mathcal{N}$ is said to be a repair group of $\mathcal{C}$ if every element $i \in \mathcal{R}$ is regenerable from $\mathcal{R} \setminus \{i\}$.*

The following lemma shows the relationship between the repair group of a liner code $\mathcal{C}$ and the dual code $\mathcal{C}^\perp$ of $\mathcal{C}$.

Definition 2 (Support). *The support of a codeword $\mathbf{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}$, denoted by supp$(\boldsymbol{c})$, is the set containing the indices of the non-zero entries of* **c**, *i.e., supp$(\boldsymbol{c}) = \{i \in \mathcal{N} : c_i \neq 0\}$.*

Lemma 1. *$\mathcal{R}$ is a repair group of a linear code $\mathcal{C}$ if and only if there is a codeword $\boldsymbol{y} \in \mathcal{C}^\perp$ such that supp$(\boldsymbol{y}) = \mathcal{R}$.*

Proof: Suppose $\mathcal{R}$ is a repair group of $\mathcal{C}$. Let $\boldsymbol{c} \triangleq (c_1, \ldots, c_n) \in \mathcal{C}$. By the definition of repair group, we can find $y_i \neq 0$ for $i \in \mathcal{R}$ such that $\sum_{i \in \mathcal{R}} y_i c_i = 0$. Let $y_i = 0$ for $i \notin \mathcal{R}$. Clearly, $\boldsymbol{y} \triangleq (y_1, \ldots, y_n) \in \mathcal{C}^\perp$ with supp$(\boldsymbol{y}) = \mathcal{R}$.

Conversely, suppose there is a codeword $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathcal{C}^\perp$ with supp$(\boldsymbol{y}) = \mathcal{R}$. Given any $\boldsymbol{c} \triangleq (c_1, \ldots, c_n) \in \mathcal{C}$, we have $\sum_{i \in \mathcal{R}} y_i c_i = 0$, and $c_i = -y_i^{-1} \sum_{j \in \mathcal{R} \setminus \{i\}} y_j c_j$ for any $i \in \mathcal{R}$. Hence, $\mathcal{R}$ is a repair group of $\mathcal{C}$. ■

Recall that during the repair process, communications and cost constraints have to be satisfied. Consider the repair of node i, where $i \in \mathcal{R}$, in the storage network $\mathcal{G}$. Since the code is linear, the regenerating function of y_i can be expressed as in (2). To regenerate y_i from $\mathcal{R} \setminus \{i\}$, the following repair procedure can be used:

1) Find a minimum-weight Steiner tree in $\mathcal{G}$ such that the following three conditions are satisfied:
   a) it spans all vertices in $\mathcal{R}$;
   b) it contains no more than $\phi + 1$ vertices;
   c) the sum of its edge weights is no more than c.
2) Let node i be the root node of the Steiner tree. Note that all the leaf nodes of the Steiner tree belongs to $\mathcal{R}$.
3) Each of the leaf nodes, say node j, sends $a_j y_j$ to its parent node.
4) Each of non-leaf nodes, say node k, after receiving all symbols from its child nodes, add all the incoming symbols together with $a_k y_k$, and send the sum to its parent node.
5) Node i, after receiving all symbols from its child nodes, regenerates y_i by adding all incoming symbols together.

Note that each edge of the Steiner tree has been used to transmit one and only one symbol. As a result, the repair cost is equal to the weight of the minimum-weight Steiner tree. The above repair procedure motivates the following definition:

Definition 3 (Feasible repair group)**.** *Given a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a subset $\mathcal{R}$ of $\mathcal{V}$ is said to be a feasible repair group if $|\mathcal{R}| \leq r + 1$, and there is a minimum-weight Steiner tree which spans all vertices in $\mathcal{R}$, contains no more than $\phi+1$ vertices, and have a total edge weights of no more than c.*

IV. The Linear LR Code Problem

In this section, we consider the problem of determining whether a linear LR code with a given rate exists, subject to the constraints of symbol locality, node locality, and repair cost. We assume that r and ϕ are small constants, which are independent of n. All the feasible repair groups can be found by Algorithm 1. To analyze its computational complexity in terms of n, we only need to consider the two outermost for-loops. We need to examine $\binom{n}{i}$ subsets for each given i, where $i = 2, 3, \ldots, \phi + 1$. Since the term for $i = \phi + 1$ dominates, the complexity of Algorithm 1 is $O(n^{\phi+1})$, which grows polynomially in n.

After finding all the feasible repair groups, it remains to answer the following question:

Problem: LR-Code$_{q,r}$
Instance: Integers n, k, and a collection of feasible repair groups $\mathcal{R}_0 = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_l\}$, where $|\mathcal{R}_i| \leq r + 1$ for $i = 1, 2, \ldots, l$, and $\cup_{i=1}^{l} \mathcal{R}_i = \mathcal{N}$
Question: Is there a q-ary linear LR code, with rate at least k/n?

Now we prove that LR-Code$_{2,r}$ is NP-complete. The key idea is a reduction from the k-SetCover problem, described below:

Problem: k-SetCover
Instance: A universe $\mathcal{U}$ of n elements, a collection of subsets of $\mathcal{U}$, $\mathcal{S}_0 = \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_l\}$, where $|\mathcal{S}_i| \leq k$ for $i = 1, 2, \ldots, l$ and $\cup_{i=1}^{l} \mathcal{S}_i = \mathcal{U}$.

Algorithm 1: Finding all feasible repair groups

```
Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}), r, \phi, c$
Output: The collection of all feasible repair groups, $\mathcal{R}_0$
  1) Initialization: $\mathcal{R}_0 \leftarrow \emptyset$.
  2) for $i = 2$ to $\phi + 1$ do
     Choose all $i$-subsets of $\mathcal{V}$ and get $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{\binom{n}{i}}$.
     $z \leftarrow \min(i, r + 1)$.
       for $j = 1$ to $\binom{n}{i}$ do
         if $\mathcal{G}_{\mathcal{S}_j}$ is connected then
         Compute the weight of a minimum-weight
         spanning tree of $\mathcal{G}_{\mathcal{S}_j}$, which is $T_{\mathcal{S}_j}$.
         (Note that $\mathcal{G}_{\mathcal{S}_j}$ denotes the subgraph of $\mathcal{G}$
         induced by the vertices in $\mathcal{S}_j$.)
           if $T_{\mathcal{S}_j} \leq c$ then
             for $k = 2$ to $z$ do
             Choose all $k$-subsets of $\mathcal{S}_j$ and get
             $\mathcal{W}_1, \mathcal{W}_2, \ldots, \mathcal{W}_{\binom{i}{z}}$. Add each of them to
             $\mathcal{R}_0$ if it is not already in $\mathcal{R}_0$.
             end for
           end if
         end if
       end for
     end for
  3) Return $\mathcal{R}_0$.
```

Question: Is there a collection of $m < n$ subsets, chosen from $\mathcal{S}_0$, whose union covers all elements in $\mathcal{U}$?

When $k = 2$, the above problem can be solved in polynomial time by matching techniques. When $k \geq 3$, it is NP-complete [14]. Before proceeding, we introduce the following definition:

Definition 4 (Characteristic vector)**.** *The characteristic vector of a set $\mathcal{S} \subseteq \mathcal{N}$ is an n-dimensional binary vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ where $a_i = 1$ if and only if $i \in \mathcal{S}$.*

Theorem 2. LR-Code$_{2,r}$ *is NP-complete for $r \geq 2$.*

Proof: Given an instance of $(r+1)$-SetCover, we let the number of storage nodes be n. Each $\mathcal{S}_i$ is regarded as a feasible repair group $\mathcal{R}_i$. Let the parameter k in the LR-Code$_{2,r}$ problem be equal to $n - m$. Now we have an instance of LR-Code$_{2,r}$, and the transformation can be done in polynomial time.

Suppose the answer to the given instance of $(r + 1)$-SetCover is Yes. Then there exists $\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_m \in \mathcal{S}_0$ whose union covers all the n elements. For $i = 1, 2, \ldots, m$, let $\boldsymbol{h}_i$ be the characteristic vector of $\mathcal{S}'_i$. Let the vector space spanned by $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_m$ be $\mathcal{C}^{\perp}$. Since the union of the m subsets covers all the elements, every storage node belongs to at least one of the m repair groups specified by $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_m$. Since these m repair groups are all feasible, $\mathcal{C}$ is a binary LR code. Since the dimension of $\mathcal{C}^{\perp}$ must be less than or equal to m, the dimension of $\mathcal{C}$ must be at least

$n - m = k$, and its code rate must be at least k/n. Therefore, the derived LR-CODE$_{2,r}$ instance is also a YES instance.

Conversely, suppose the answer to the derived instance of LR-CODE$_{2,r}$ is YES. Let the LR code be $\mathcal{C}$. Since each storage node must belong to at least one repair group, there must be a collection of repair groups which jointly cover all n storage nodes. Denote the cardinality of this collection by p. By Lemma 1, there are p codewords $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p \in \mathcal{C}^{\perp}$ such that $supp(\boldsymbol{y}_i) \in \mathcal{R}_0$ for $i = 1, 2, \ldots, p$, and $\cup_{i=1}^{p} supp(\boldsymbol{y}_i) = \mathcal{N}$. Since the dimension of $\mathcal{C}$ is at least k, the dimension of $\mathcal{C}^{\perp}$ is at most m. Let $\mathcal{Y}$ be the linear span of $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p$, and let ω be its dimension. Clearly, $\omega \leq m$. Among these p vectors, we can find ω of them which form a basis of $\mathcal{Y}$. Let them be $\boldsymbol{y}_{j_1}, \boldsymbol{y}_{j_2}, \ldots, \boldsymbol{y}_{j_\omega}$, where $1 \leq j_k \leq p$ for $k = 1, 2, \ldots, \omega$. Since $\cup_{i=1}^{p} supp(\boldsymbol{y}_i) = \mathcal{N}$, we must have $\cup_{k=1}^{\omega} supp(\boldsymbol{y}_{j_k}) = \mathcal{N}$, for otherwise there exists $\boldsymbol{y}_i$, where $1 \leq i \leq p$, which is not in the linear span of $\boldsymbol{y}_{j_1}, \boldsymbol{y}_{j_2}, \ldots, \boldsymbol{y}_{j_\omega}$. The supports of these ω codewords correspond to the collection of ω subsets required by the $(r+1)$-SETCOVER instance. Hence, the given $(r+1)$-SETCOVER instance is a YES instance.

Since LR-CODE$_{2,r}$ is clearly in NP, it is NP-complete. ■

V. CODE RATE MAXIMIZATION PROBLEM

In this section, we focus on the code rate maximization problem of a linear LR code $\mathcal{C}$, which is formally stated as follows:

Problem: LR-CODEOPT$_{q,r}$
Instance: Integer n, and a collection of feasible repair groups $\mathcal{R}_0 = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_l\}$, where $|\mathcal{R}_i| \leq r+1$ for $i = 1, 2, \ldots, l$, and $\cup_{i=1}^{l} \mathcal{R}_i = \mathcal{N}$
Objective: To maximize the code rate of a q-ary linear LR code.

In particular, we focus on the special case LR-CODEOPT$_{2,r}$. It follows from Theorem 2 that LR-CODEOPT$_{2,r}$ is NP-hard for $r \geq 2$.

To tackle the problem, we apply the concept of Cook reduction [15]. The problem LR-CODEOPT$_{2,r}$ is first polynomial-time reduced to the minimization version of $(r+1)$-SETCOVER, which is denoted by MIN-$(r+1)$-SETCOVER. This reduction is done by simply regarding the feasible repair groups as the subsets $\mathcal{S}_i$ in MIN-$(r+1)$-SETCOVER. A single oracle call to solve MIN-$(r+1)$-SETCOVER is performed. Let the returned value be m^*. The maximum code rate for LR-CODEOPT$_{2,r}$ is given in the following theorem:

Theorem 3. *Let m^* be the optimal solution to a* MIN-$(r+1)$-SETCOVER *instance, reduced from a given* LR-CODEOPT$_{2,r}$ *instance as described above. The maximum code rate for* LR-CODEOPT$_{2,r}$ *is given by $(n - m^*)/n$.*

Proof: Since m^* is an optimal solution to a MIN-$(r+1)$-SETCOVER instance, there exists m^* feasible repair groups, say $\mathcal{R}'_1, \mathcal{R}'_2, \ldots, \mathcal{R}'_{m^*} \in \mathcal{R}_0$, whose union covers all the n elements. For $i = 1, 2, \ldots, m^*$, let $\boldsymbol{h}_i$ be the characteristic vector of $\mathcal{R}'_i$. Let the linear span of $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_{m^*}$ be $\mathcal{C}^{\perp}$. If the dimension of $\mathcal{C}^{\perp}$ is $m < m^*$, we can find $\boldsymbol{h}_{j_1}, \boldsymbol{h}_{j_2}, \ldots, \boldsymbol{h}_{j_m}$, where $1 \leq j_k \leq m^*$ for $k = 1, 2, \ldots, m$, which form a basis of $\mathcal{C}^{\perp}$. Then we have $\cup_{i=1}^{m} supp(\boldsymbol{h}_{j_k}) = \mathcal{N}$, for otherwise there exists $\boldsymbol{h}_i$, where $1 \leq i \leq m^*$, which is not in the linear span of $\boldsymbol{h}_{j_1}, \boldsymbol{h}_{j_2}, \ldots, \boldsymbol{h}_{j_m}$. In other words, the union of the supports of $\boldsymbol{h}_{j_1}, \boldsymbol{h}_{j_2}, \ldots, \boldsymbol{h}_{j_m}$ covers all the n elements, which contradicts with the assumption that m^* is optimal for the MIN-$(r+1)$-SETCOVER instance. Therefore, the dimension of $\mathcal{C}^{\perp}$ must be equal to m^*, implying that the dimension of $\mathcal{C}$ is equal to $n - m^*$. Hence, the maximum code rate for LR-CODEOPT$_{2,r}$ is $(n - m^*)/n$. ■

A. Optimal Polynomial-Time Algorithm for $r = 1$

When the symbol locality $r = 1$, the collection of feasible repair groups for LR-CODEOPT$_{2,r}$ corresponds to a set of edges in the storage network. The MIN-$(r+1)$-SETCOVER problem becomes a graph problem known as MIN-EDGECOVER, which is required to find an edge set of minimum cardinality such that every vertex of the graph is incident to at least one edge in the set. The MIN-EDGECOVER instance can be solved optimally in polynomial time by finding a maximum matching and extending it greedily so that all vertices are covered. As a result, LR-CODEOPT$_{2,1}$ can be solved optimally in polynomial time by a maximum matching algorithm.

B. Approximation Algorithm for $r \geq 2$

When the symbol locality $r \geq 2$, we use a greedy algorithm for the MIN-$(r+1)$-SETCOVER problem to find an approximate solution to LR-CODEOPT$_{2,r}$. The greedy algorithm for the MIN-$(r+1)$-SETCOVER problem starts with an empty collection $\mathcal{T}$ of subsets of the universe. At each step, a subset, not previously chosen, with the largest number of uncovered elements is chosen and added to $\mathcal{T}$. The procedure repeats until all elements are covered. We formally state it as Algorithm 2.

Algorithm 2: Greedy Algorithm for Repair Group Selection

```
Input: A set $\mathcal{N} = \{1, 2, \ldots, n\}$, and a collection of
       feasible repair groups $\mathcal{R}_0 = \{\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_l\}$,
       where $\mathcal{R}_i \subseteq \mathcal{N}$ and $|\mathcal{R}_i| \leq r+1$ for
       $i = 1, 2, \ldots, l$, and $\cup_{i=1}^{l} \mathcal{R}_i = \mathcal{N}$.
Output: A collection of repair groups, $\mathcal{T} \subseteq \mathcal{R}_0$, whose
        union covers all elements in $\mathcal{N}$.

Let UNCOV and SET($i$), where $i = 1, 2, \ldots, l$, represent
the set of elements that are not yet covered in $\mathcal{N}$ and $\mathcal{R}_i$,
respectively.
  1) Initialization: $\mathcal{T} \leftarrow \emptyset$, UNCOV $\leftarrow \mathcal{N}$, SET($i$) $\leftarrow \mathcal{R}_i$
     for $i = 1, 2, \ldots, l$.
  2) If UNCOV = $\emptyset$, exit and return $|\mathcal{T}|$.
  3) Choose $j \leq l$ such that $|$SET($j$)$|$ is maximized.
  4) $\mathcal{T} \leftarrow \mathcal{T} \bigcup \{\mathcal{R}_j\}$, UNCOV $\leftarrow$ UNCOV $\setminus$ SET($j$),
     SET($i$) $\leftarrow$ SET($i$) $\setminus$ SET($j$) for $i = 1, 2, \ldots l$.
  5) Go to step 2.
```

Algorithm 2 yields an approximate solution to LR-CodeOpt$_{2,r}$ with $r \geq 2$. The code rate obtained is equal to $\frac{n-|\mathcal{T}|}{n}$. It is well known that the greedy algorithm stated in Algorithm 2 is an H_{r+1} factor approximation algorithm for Min-$(r+1)$-SetCover [16], where $H_{r+1} \triangleq \sum_{i=1}^{r+1} \frac{1}{i}$ is the $(r+1)$-th harmonic number. The approximation factor of Algorithm 2 for LR-CodeOpt$_{2,r}$ is given in the following theorem:

Theorem 4. *Algorithm 2 is an* $\frac{n-H_{r+1}}{n-1}$ *factor approximation algorithm for* LR-CodeOpt$_{2,r}$.

Proof: Given an LR-CodeOpt$_{2,r}$ instance, let m^* be the optimal solution to the corresponding Min-$(r+1)$-SetCover instance. Then $t^* = \frac{n-m^*}{n}$ is the maximum code rate. Let m be the solution to the Min-$(r+1)$-SetCover instance using Algorithm 2, and $t = \frac{n-m}{n}$ is the code rate obtained. Since the greedy algorithm is an H_{r+1} factor approximation algorithm for Min-$(r+1)$-SetCover, we have $m \leq H_{r+1}m^*$. The approximation ratio for the maximum code rate is given by

$$\frac{t}{t^*} = \frac{n-m}{n-m^*} \geq \frac{n-m^*H_{r+1}}{n-m^*} = 1 - \frac{H_{r+1}-1}{\frac{n}{m^*}-1}. \quad (4)$$

Since $1-\frac{H_{r+1}-1}{\frac{n}{m^*}-1}$ decreases monotonically with m^* and $m^* \geq 1$, we can guarantee that

$$\frac{t}{t^*} \geq \frac{n-H_{r+1}}{n-1}. \quad (5)$$

■

Example: Consider a binary linear LR code $\mathcal{C}$ with symbol locality $r=4$ over a distributed storage network. Let the set of storage nodes be $\mathcal{N} = \{1,2,3,4,5,6,7,8\}$ and the collection of feasible repair groups be $\mathcal{R}_0 = \{\mathcal{R}_1 = \{1,2,3,6,7\}, \mathcal{R}_2 = \{1,2,3,4\}, \mathcal{R}_3 = \{5,6,7,8\}, \mathcal{R}_4 = \{4,5\}, \mathcal{R}_5 = \{4,8\}\}$. To maximize the code rate of $\mathcal{C}$, we need to choose a subset $\mathcal{T} \subseteq \mathcal{R}_0$ with the smallest number of repair groups such that every storage node in $\mathcal{N}$ is contained in at least one repair group in $\mathcal{T}$. We can see that the optimal solution is $|\mathcal{T}| = 2$ because the union of $\mathcal{R}_2$ and $\mathcal{R}_3$ covers all elements in $\mathcal{N}$. Thus the maximum code rate of $\mathcal{C}$ is $\frac{8-2}{8} = \frac{3}{4}$. If we use Algorithm 2, the largest feasible repair group $\mathcal{R}_1 = \{1,2,3,6,7\}$ is first chosen. Excluding the elements in $\mathcal{R}_1$ from the feasible repair groups, we are left with the sets $\{4\}$, $\{5,8\}$, $\{4,5\}$, and $\{4,8\}$. The greedy algorithm will successively add $\mathcal{R}_3$ and $\mathcal{R}_2$ into $\mathcal{T}$. The resultant code rate is $\frac{8-3}{8} = \frac{5}{8}$ and the approximation ratio is $\frac{5}{6}$, which is greater than the approximation guarantee, $\frac{n-H_{r+1}}{n-1} = \frac{343}{420}$, for this instance.

VI. Conclusion

This paper focuses on locally repairable codes over a network. We consider the network topology in the repair process, and accordingly introduce the new concept of node locality. We give an algorithm to find all the feasible repair groups. After finding all the feasible repair groups, we consider the problem of determining whether a linear LR code over a network with a given code rate exists, subject to the constraints of symbol locality, node locality, and repair cost. We prove that the decision instance of a binary linear LR code is NP-Complete. Besides, given a storage network with arbitrary topology, we investigate the problem of maximizing the code rate of a linear LR code by properly choosing repair groups from feasible repair groups such that the local repair requirement of every storage node is satisfied. We show that the code rate maximization of a binary linear LR code with symbol locality r can be reduced to the Min-$(r+1)$-SetCover problem in polynomial time, and can be solved optimally when $r=1$. When $r \geq 2$, the code rate maximization is NP-hard and a greedy algorithm is provided to approximate the optimal solution.

While this paper considers only single node failure, the results can be extended to multiple node failure by considering the set multicover problem. Future works also include the consideration of non-binary codes.

References

[1] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. IEEE Int. Conf. on Computer Commun. (INFOCOM '07)*, Anchorage, Alaska, May 2007, pp. 2000–2008.

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

[3] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proc. IEEE Int. Conf. on Computer Commun. (INFOCOM '11)*, Shanghai, China, Apr. 2011, pp. 1215–1223.

[4] 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.

[5] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis, "Optimal locally repairable codes and connections to matroid theory," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Jul. 2013, pp. 1814–1818.

[6] N. Silberstein, A. S. Rawat, O. O. Koyluolu, and S. Vishwannath, "Optimal locally repairable codes via rank-metric codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Jul. 2013, pp. 1819–1823.

[7] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar, "Codes with local regeneration," in *Proc. IEEE Information Theory and Applications Workshop (ITA)*, San Diego, CA, Feb. 2013, pp. 1–5.

[8] 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'10)*, San Diego, Mar. 2010, pp. 1–9.

[9] M. Gerami, M. Xiao, and M. Skoglund, "Optimal-cost repair in multi-hop distributed storage systems," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, St. Pertersburg, Jul. 2011, pp. 1437–1441.

[10] 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.

[11] Q. Yu, C. W. Sung, and T. H. Chan, "Irregular fractional repetition code optimization for heterogeneous cloud storage," *IEEE J. on Selected Areas in Commun.*, vol. 32, no. 5, pp. 1048–1060, 2014.

[12] A. Mazumdar, "On a duality between recoverable distributed storage and index coding," Mar. 2014, arXiv:1401.2672v2 [cs.IT].

[13] K. Shanmugam and A. G. Dimakis, "Bounding multiple unicasts through index coding and locally repairable codes," Feb. 2014, arXiv:1402.3895v1 [cs.IT].

[14] R. M. Karp, *Reducibility among combinatorial problems*. New York: Plenum Press, 1972.

[15] O. Goldreich, *P, NP, and NP-Completeness: The Basics of Computational Complexity*. New York: Cambridge University Press, 2010.

[16] D. S. Johnson, "Approximation algorithms for combinatorial problems," *Journal of Computer and System Sciences*, vol. 9, no. 3, pp. 256–278, 1974.