Storage Capacity of Repairable Networks

Arya Mazumdar *Member, IEEE*

***Abstract*—In this paper, we introduce a model of a distributed storage system that is locally recoverable from any single server failure. Unlike the usual local recovery model of codes for distributed storage, this model accounts for the fact that each server or storage node in a network is connectible to only some, and not all other, nodes. This may happen for reasons such as physical separation, inhomogeneity in storage platforms etc. We estimate the storage capacity of both undirected and directed networks under this model and propose some constructive schemes. From a coding theory point of view, we show that this model is *approximately* dual of the well-studied index coding problem.**

Further in this paper, we extend the above model to handle multiple server failures. Among other results, we provide an upper bound on the minimum pairwise distance of a set of words that can be stored in a graph with the local repair guarantee. The well-known impossibility bounds on the distance of locally recoverable codes follow from our result.

I. Introduction

Recently, the local repair property of error-correcting codes is the center of a lot of research activities. In a distributed storage system, a single server failure is the most common error-event, and in the case of a failure the aim is to reconstruct the content of the failed server from as few other servers as possible (or by downloading minimal amount of data from other servers). The study of such *regenerative* storage systems was initiated in [16] and then followed up in several recent works. In [22], a particularly neat characterization of the local repair property is provided. It is assumed that, each symbol of an encoded message is stored at a different node in the storage-network (since the symbol alphabet is unconstrained, a symbol could represent a packet or block of bits of arbitrary size). Accordingly, [22] investigates code-families that allow any single coordinate of a codeword to be recovered from at most a constant number of other coordinates of the codeword, i.e., from a number of coordinates that does not grow with the length of the code.

The work of [22] is then further generalized to several directions and a number of impossibility results and constructions of *locally repairable codes* were presented in [10], [24], [34], [42]–[44] among others. The central result of this body of works is that for any code of length n, dimension k and minimum distance d,

$$d \leq n - k - \left\lceil \frac{k}{r} \right\rceil + 2, \tag{1}$$

where r is such that any single coordinate can be recovered from at most r other coordinates [22].

The author is with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, email: arya@umn.edu. Part of this work was presented in the IEEE International Symposium on Information Theory, 2014 [31] and in the Allerton Conference, 2014 [30]. This work was supported in part by the National Science Foundation CAREER award under grant no. CCF 1453121.

However, the topology of the network of distributed storage system is missing from the above definition of local repairability. Namely, all servers are treated equally irrespective of their physical positions, proximities, and connections. Here, in this paper, we take a step to include the network topology into consideration. We study the case when the architecture of the storage system is fixed and the network of storage is given by a graph. In our model, the servers are represented by the vertices of a graph, and two servers are connected by an edge if it is easier to establish up-or-down link between them, for reasons such as physical locations of the servers, architecture of the distributed system or homogeneity of softwares, etc. It is reasonable to assume that the storage-graph is directed, because there may be varying difficulties in establishing an up or down link between two servers. Under this model, we impose the local recovery or repair condition in the following way: the content of any failed server must be reconstructible from the neighboring servers on *the storage graph*.

Assuming the above model, the main quantity of interest is the amount of information that can be stored in the graph. We call this quantity the *storage capacity* of the graph. Finding this capacity exactly, as well as to construct explicit schemes that achieve this capacity, are both computationally hard problems for an arbitrary graph. However, we show that good approximation schemes are possible – and for some special classes of graphs we can even compute this capacity exactly with constructive schemes. In particular, for any undirected graph, the storage capacity is sandwiched between the *maximum matching* and the *minimum vertex cover*, two quantities within a factor of two of each other. Similar statement, albeit concerning different properties, is possible for directed graphs.

It turns out that, our model is closely related to the popular *index coding* problem on a side information graph. In the index coding problem, a set of users are assigned bijectively to a set of variable that they want to know. However, instead of knowing the assigned variable, each knows a subset of other variables. This scenario can be depicted by a so-called *side-information* graph where each vertex represents a user and there is an edge between users A and B, if A knows the variable assigned to B. Given this graph, how much information should a broadcaster has to transmit, such that each vertex can deduce its assigned variable?

The above problem of index coding was introduced in [5] (it has a predecessor in [6]), and since then is a subject of extensive research. It was shown in [19] that any network coding problem can be reduced to an index coding problem, and the index coding capacity is among the computationally hardest problems of all network coding [7], [27]. A prominent work in the index coding literature is [1], that studies the *broadcast rate* for index coding. It turns out that an auxiliary

quantity (called γ) used in [1] is exactly the storage capacity[1] that we introduce and study in this paper (attachment of γ to any quantity of practical use was absent in [1]). Recently K. Shanmugam [39] pointed out to the author that this quantity has also been studied as *graph entropy*[2] in [37] by Riis.

Using the results of [1] it is possible to show a connection between the broadcast rate of index coding and the storage-capacity when the side-information graph and the storage graph are the same. Indeed, we show that there exists a *duality* between a storage code and an index code on the same graph. This observation, which also connects the *complementary index coding* rate of [11] with the storage capacity, is further explored in this paper.

The local repairability property on a graphical model of storage can be extended to several directions. One may ask for protection against catastrophic failures, and therefore also impose a minimum *distance* condition on codes, which is a common fixture of the local recovery literature. In this scenario, we obtain a general bound that include previous results such as eq. (1) as special cases. Moreover such bounds can also be made dependent on the size of the alphabet (size of storage node).

Furthermore, instead of a single node local repairability, multiple failures can also be considered. Such multiple failures and the corresponding cooperative local recovery model in distributed storage was recently introduced in [35]. In this paper we generalize this model on graphs.

The storage coding problem of our model is a very fundamental network coding problem, and one of our main observation is that reasonable approximation schemes are possible for storage coding. While the index coding rate is very hard to approximate (see, [27]) it is possible to have good approximation constructively for storage capacity with *linear* (explained in Section II) codes. This should be put into contrast with results, such as [7, Thm. 1.2], which show that a rather large gap must exist between vector linear and nonlinear index coding (or general network coding) rates.

Apart from the approximation guarantee, there are other evidences to the fact that index coding and our storage coding are two very different problems by nature. For example, for two disconnected graphs, the total storage capacity is the sum of the capacities of the individual graphs. But the index coding length for the union of two disconnected graphs may be smaller than the sum of individual code lengths of the graphs (see, Thm. 1.1-1.4 and the accompanying discussions in [1]).

In a parallel independent work [40], one of our initial results, namely, the duality between storage and index codes (see Prop. 1) is proved for *vector linear codes*. The authors of [40] further use that observation to give an upper bound on the optimal linear sum rate of the multiple unicast network coding problem. In this paper we have a completely different focus.

A. Results and organization

The paper is organized in the following way.

- *Model of a repairable distributed storage:* In Section II, we introduce formally the model of a recoverable distributed storage system and the notion of an optimal storage code given a graph. This section also introduces the quantities of our interest, namely the capacity of storage.
- *Relation to index coding:* In Section III, we explore the relation of an optimal storage code to the optimal index code. We provide an algorithmic proof of a duality relation between the index code and distributed storage code. Our proof is based on a covering argument of the Hamming space, and rely on the fact that for any given subset of the Hamming space there exist several translations of the set, that have very small overlaps with the original subset.
- *Bounds and algorithms:* In Section IV-A, we provide constructive schemes that achieves a storage-rate within half of what is maximum possible for any undirected graph (the scheme is optimum for bipartite graphs). Some other existential results are also proved in this section. Next, we extend the approximation schemes towards directed graphs in Section IV-B. It turns out to be a harder problem for directed graphs and we provide a scheme with a logarithmically (with graph size) growing approximation factor.
- *Bounds on minimum distance and other multiple failure models:* In the last section, Section V, we generalize the notions of local recovery on graphs to include the minimum distance criterion and cooperative local recovery. In both of these cases, we provide fundamental converse bounds and outline some constructive schemes. In particular, the well-known impossibility results on the minimum distance of a locally repairable codes, such as eq. (1) or the ones presented in [10], simply follow from Thm. 14 and Prop. 15.

II. Recoverable distributed storage systems

In this section, we introduce the basic notion of a single-failure recoverable storage system. Consider a network of distributed storage, for example, one of Fig. 1, where several servers (vertices) are connected via network links (edges). As mentioned in the introduction, the property of two servers connected by an edge is based on the ease of establishing a link between the servers[3]. If the data of any one server is lost, we want to recover it from the *nearby* servers, i.e., the ones with whom it is easy to establish links. This notion is formalized below. It is also possible (and sensible, perhaps) to model this as a directed graph (especially when uplink and downlink constructions have varying difficulties). In the rest of the paper the definitions, claims and arguments hold for both directed and undirected graphs, unless otherwise specified.

Suppose, the graph $G(V, E)$ represents the network of storage. For any $v \in V$, define $N(v) = \{u \in V : (v, u) \in E\}$ to be the neighborhood of v. Each element of V represents a server, and in the case of a server failure (say, $v \in V$ is the

[1] Actually, $\log \gamma$ is the storage capacity that we introduce here.

[2] In literature, the term "graph entropy" usually refers to a different quantity [26].

[3] One might consider a nonnegative weight on each edge, which would be a natural generalization

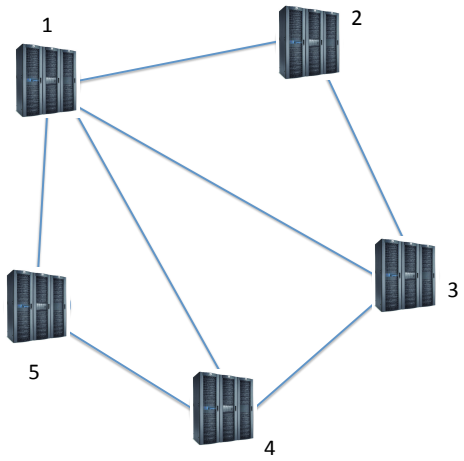

Fig. 1. Example of a distributed storage graph

failed server) one must be able to reconstruct its content from its neighborhood $N(v)$.

Given this constraint what is the maximum amount of information one can store in the system? Without loss of generality, assume $V = \{1, 2, \ldots, n\}$ and the variables $X_1, X_2, \ldots, X_n$ respectively denote the content of the vertices, where, $X_i \in \mathbb{F}_q, i = 1, \ldots, n$. Also for any $I \subseteq V$, let $X_I \in \mathbb{F}_q^{|I|}$ be the projection of $(X_1, X_2, \ldots, X_n)^T$ on to the coordinates of I.

Definition 1: A *recoverable distributed storage system (RDSS) code* $\mathcal{C} \subseteq \mathbb{F}_q^n$ with storage recovery graph $G(V, E), V = \{1, 2, \ldots, n\}$, is a set of vectors in $\mathbb{F}_q^n$ together with a set of deterministic recovery functions, $f_i : \mathbb{F}_q^{|N(i)|} \to \mathbb{F}_q$ for $i = 1, \ldots, n$ such that for any codeword $(X_1, X_2, \ldots, X_n)^T \in \mathbb{F}_q^n$,

$$X_i = f_i(X_{N(i)}), \quad i = 1, \ldots, n. \tag{2}$$

The decoding functions depend on G. The log-size of the code, $\log_q |\mathcal{C}|$, is called the dimension of $\mathcal{C}$, or $\dim(\mathcal{C})$. Given a graph G the maximum possible dimension of an RDSS code is denoted by $\mathbb{CAP}_q(G)$.

Note that, in this paper, $\mathbb{CAP}_q(G)$ is expressed in q-ary units. To convert it to *bits* we need to multiply with $\log_2 q$.

As an example, if G is a complete graph then $\mathbb{CAP}_q(G) = n - 1$. This is possible because in $n - 1$ vertices we can store arbitrary values, and in the last vertex we can store the sum (modulo q) of the stored values.

As another example, consider the graph of Fig. 1 again. Here, $V = \{1, 2, 3, 4, 5\}$. The recovery sets of each vertex (or storage nodes) are given by:

$$N(1) = \{2, 3, 4, 5\},\ N(2) = \{1, 3\}, N(3) = \{1, 2, 4\},$$
$$N(4) = \{1, 3, 5\}, N(5) = \{1, 4\}.$$

Suppose, the contents of the nodes $1, 2, \ldots, 5$ are $X_1, X_2, \ldots, X_5$ respectively, where, $X_i \in \mathbb{F}_q, i = 1, \ldots, 5$. Moreover, $X_1 = f_1(X_2, X_3, X_4, X_5), X_2 = f_2(X_1, X_3), X_3 = f_3(X_1, X_2, X_4), X_4 = f_4(X_1, X_3, X_5), X_5 = f_5(X_1, X_4)$.

Assume, the functions $f_i, i = 1, \ldots, 5$, in this example are linear. That is, for $\alpha_{ij} \in \mathbb{F}_q, 1 \le i, j \le 5$,

$$\begin{aligned} X_1 &= \alpha_{12}X_2 + \alpha_{13}X_3 + \alpha_{14}X_4 + \alpha_{15}X_5 \\ X_2 &= \alpha_{21}X_1 + \alpha_{23}X_3 \\ X_3 &= \alpha_{31}X_1 + \alpha_{32}X_2 + \alpha_{34}X_4 \\ X_4 &= \alpha_{41}X_1 + \alpha_{43}X_3 + \alpha_{45}X_5 \\ X_5 &= \alpha_{51}X_1 + \alpha_{54}X_4. \end{aligned}$$

This implies, $(X_1, X_2, \ldots, X_5)$ must belong to the null-space (over $\mathbb{F}_q$) of

$$D \equiv \begin{pmatrix} 1 & -\alpha_{12} & -\alpha_{13} & -\alpha_{14} & -\alpha_{15} \\ -\alpha_{21} & 1 & -\alpha_{23} & 0 & 0 \\ -\alpha_{31} & -\alpha_{32} & 1 & -\alpha_{34} & 0 \\ -\alpha_{41} & 0 & -\alpha_{43} & 1 & -\alpha_{45} \\ -\alpha_{51} & 0 & 0 & -\alpha_{54} & 1 \end{pmatrix}.$$

The dimension of the null-space of D is n minus the rank of D. At this point the following definition is useful.

Suppose, $A = (a_{ij})$ be an $n \times n$ matrix over $\mathbb{F}_q$. It is said that A *fits* $G(V, E)$ over $\mathbb{F}_q$ if $a_{ii} \neq 0$ for all i and $a_{ij} = 0$ whenever $(i, j) \notin E$ and $i \neq j$.

Definition 2: The minrank [23] of a graph $G(V, E)$ over $\mathbb{F}_q$ is defined to be,

$$\text{minrank}_q(G) = \min\{\text{rank}_{\mathbb{F}_q}(A) : A \text{ fits } G\}. \tag{3}$$

Notice, in the example above, D *fits* the graph G. Hence, it is evident that the dimension of the RDSS code is $n - \text{minrank}_q(G)$ (see, Defn. 2). From the above discussion, we have,

$$\mathbb{CAP}_q(G) \ge n - \text{minrank}_q(G), \tag{4}$$

and, $n - \text{minrank}_q(G)$ is the maximum possible dimension of an RDSS code when the recovery functions are all linear.

Linear RDSS codes are not optimal all the time. This is shown in the following example.

Example 1: This example is present in [1], and the distributed storage graph, a *pentagon*, is shown in Fig. 2. For this graph, a maximum-sized binary RDSS code consists of the codewords $\{00000, 01100, 00011, 11011, 11101\}$. The recovery functions are given by,

$$X_1 = X_2 \wedge X_5, X_2 = X_1 \vee X_3, X_3 = X_2 \wedge \bar{X}_4,$$
$$X_4 = \bar{X}_3 \wedge X_5, X_5 = X_1 \vee X_4.$$

Here $\mathbb{CAP}_2(G) = \log_2 5$ bits. If all the recovery functions are linear, we could not have an RDSS code with so many codewords. Indeed, since minrank of this graph over $\mathbb{F}_2$ is 3, we could have had only $2^{5-3} = 4$ codewords with linear recovery functions.

Literatures of distributed storage often considers *vector codes* and *vector linear codes*. In our case, in a vector code, instead of a symbol, a vector is stored in each of the vertices. In the context of general nonlinear codes, vector codes do not bring any further technical novelty and can just be thought of as codes over a larger alphabet. The capacity of storage can only increase when we consider codes over larger alphabet.

Definition 3: Define the *vector capacity* of a graph $G(V,E)$ to be,

$$\mathbb{CAP}(G) = \lim_{p\to\infty} \mathbb{CAP}_{2^p}(G). \tag{5}$$

Recall that $\mathbb{CAP}_q(G)$ is measured in q-ary units above. The limit of (5) exists and is equal to $\sup_p \mathbb{CAP}_{2^p}(G)$. This follows from the superadditivity,

$$(p_1+p_2)\mathbb{CAP}_{2^{p_1+p_2}}(G) \geq p_1\mathbb{CAP}_{2^{p_1}}(G)+p_2\mathbb{CAP}_{2^{p_2}}(G),$$

and Fekete's lemma.

A vector *linear* RDSS code, on the other hand, is quite different from simple linear codes. Each server stores a vector of p symbols, say. Now in the event of a node failure, each of the p lost symbols must be recoverable by a *linear function* of all the symbols stored in the neighboring vertices. In other words, if q-ary symbols are stored in the vertices, then the recovery functions are over $\mathbb{F}_q$, and not over $\mathbb{F}_{q^p}$ (for nonlinear recovery, this does not make any difference).

The Shannon capacity [41] of a graph is a well-known quantity and it is known to be upper bounded by minrank [23]. Any concrete reasoning relating Shannon capacity to $\mathbb{CAP}$ is of interest, but has not been pursued in this paper. It is to be noted that for the pentagon of Fig. 2, the Shannon capacity is $\sqrt{5}$ while $\mathbb{CAP}_2(G) = \log_2 5$.

III. Relation with Index Coding

We start this section with the definition of an index coding problem. The main objective of this section is to establish and explore the relation of the index coding rate and $\mathbb{CAP}_q(G)$, given a graph G.

In the index coding problem, a possibly *directed* side information graph $G(V,E)$ is given. Each vertex $v \in V$ represents a receiver that is interested in knowing a uniform random variable $Y_v \in \mathbb{F}_q$. The receiver at v knows the values of the variables $Y_u, u \in N(v)$. How much information should a broadcaster transmit, such that every receiver knows the value of its desired random variable? Let us give the formal definition from [5], adapted for q-ary alphabet here.

Definition 4: An *index code* $\mathcal{C}$ for $\mathbb{F}_q^n$ with side information graph $G(V,E), V = \{1,2,\ldots,n\}$, is a set of codewords in $\mathbb{F}_q^\ell$ together with:

1) An encoding function f mapping inputs in $\mathbb{F}_q^n$ to codewords, and
2) A set of deterministic decoding functions $g_1,\ldots,g_n$ such that $g_i\Big(f(Y_1,\ldots,Y_n), Y_{N(i)}\Big) = Y_i$ for every $i = 1,\ldots,n$.

The encoding and decoding functions depend on G. The integer ℓ is called the length of $\mathcal{C}$, or $\text{len}(\mathcal{C})$. Given a graph G the minimum possible length of an index code is denoted by $\text{INDEX}_q(G)$.

It is not very difficult to deduce the connection between the length of an index code to the minrank of the graph – and it was shown in [5] that,

$$\text{INDEX}_q(G) \leq \text{minrank}_q(G). \tag{6}$$

The above inequality can be strict in many cases [1], [29]. However, $\text{minrank}_q(G)$ is the minimum length of an index code on G when the encoding function, and the decoding functions are all *linear*. The following proposition is also immediate.

Proposition 1: The null-space of a linear index code for G is a linear RDSS code for the same graph G.

Proof: All the vectors that are mapped to zero by the encoding function of an index code form an RDSS code, as any symbol stored at a vertex can be recovered by the corresponding index coding decoding function for that vertex. On the other hand the cosets of an RDSS code partition the space and the set of cosets is isomorphic to the null-space of the RDSS code. Hence an index code can be formed that encodes a vector to the coset it belongs to. ∎

Note that, it is not true that $\mathbb{CAP}_q(G) = n - \text{INDEX}_q(G)$, although Eq. (6) and Eq. (4) suggest a similar relation. This is shown in the graph of Example 1. There, the minimum length of an index code for this graph is 3, i.e., $\text{INDEX}_2(G) = 3$, and this is achieved by the following linear mappings. The broadcaster transmit $Y_1 = X_2 + X_3, Y_2 = X_4 + X_5$ and $Y_3 = X_1 + X_2 + X_3 + X_4 + X_5$. The decoding functions are, $X_1 = Y_1+Y_2+Y_3; X_2 = Y_1+X_3; X_3 = Y_1+X_2; X_4 = Y_2+X_5; X_5 = Y_2 + X_4$.

Although in general $\mathbb{CAP}_q(G) \neq n - \text{INDEX}_q(G)$, these two quantities are not too far from each other. In particular, for large enough alphabet, the left and right hand sides can be arbitrarily close. This is reflected in Thm. 2 below.

A. Implication of the results of [1]

At this point we cast a result of [1] in our context. In [1], the problem of index coding was considered and to characterize the optimal size of an index code, the notion of a *confusion graph* was introduced. Two input strings, $\mathbf{x} = (x_1,\ldots,x_n), \mathbf{y} = (y_1,\ldots,y_n) \in \mathbb{F}_q^n$ are called *confusable* if there exists some $i \in \{1,\ldots,n\}$, such that $x_i \neq y_i$, but $x_j = y_j$, for all $j \in N(i)$. In the confusion graph of G, the total number of vertices are q^n, and each vertex represents a different q-ary-string of length n. There exists an edge between two vertices if and only if the corresponding two strings are confusable with respect to the graph G. The maximum size of an *independent set* of the confusion graph is denoted by $\gamma(G)$.

The confusion graph and $\gamma(G)$ in [1] were used as auxilaries to characterize the the *rate* of index coding; they were not used to model any practical problem. From our definition of RDSS codes (see Def. 1), it is evident that this notion of *confusable* strings fits perfectly to the situation of *local recovery* of a distributed storage system. Namely, $\gamma(G)$, in our problem becomes the largest possible size of an RDSS code for a system with storage-graph given by G.

We restate one of the main theorems of [1] using the terminology we have introduced so far.

Theorem 2: Given a graph $G(V,E)$, we must have,

$$\begin{aligned} n-\mathbb{CAP}_q(G) \leq \text{INDEX}_q(G) \leq n - \mathbb{CAP}_q(G) \\ + \log_q\Big(\min\{n\ln q, 1+\mathbb{CAP}_q(G)\ln q\}\Big). \end{aligned} \tag{7}$$

This result is purely graph-theoretic, the way it was presented in [1]. In particular, the size of maximum independent set of

the confusion graph, $\gamma(G)$ can be identified as the size of the RDSS code, and its relation to the *chromatic number* of the confusion graph, which represents the size of the index code, was found. Namely the proof was dependent on the following two crucial steps.

1) The *chromatic number* of the graph can only be so much away from the *fractional chromatic number* (see, [1] for detailed definition).
2) The confusion graph is *vertex transitive*. This implies that the maximum size of an independent set is equal to the number of vertices divided by the fractional chromatic number.

A proof of the first fact above can be found in [28]. In what follows, we give a simple *coding theoretic* proof of Thm. 2, where the technique is same as [1]; but it bypasses the graph-theoretic notations. However, our proof will expose some further nuances in the relation of index coding and RDSS codes (see, Sec. III-C and Lemma. 7). Because of the derandomization of Lemma. 7, we can get rid of a look-up table to decode the index code that is 'dual' of a given RDSS code.

B. The proof of the duality

We prove Theorem 2 with the help of following two lemmas. The first of them is immediate and can be proved by a simple averaging argument.

Lemma 3: If there exists an index code $\mathcal{C}$ of length ℓ for a side information graph G on n vertices, then there exists an RDSS code of dimension at least $n-\ell$ for the distributed storage graph G.

Proof: Suppose, the encoding and decoding functions of the index code $\mathcal{C}$ are $f:\mathbb{F}_q^n\to\mathbb{F}_q^\ell$ and $g_i:\mathbb{F}_q^{\ell+N(i)}\to\mathbb{F}_q, i=1,\ldots,n$. There must exists some $\mathbf{x}\in\mathbb{F}_q^\ell$ such that $|\{\mathbf{y}\in\mathbb{F}_q^n : f(\mathbf{y})=\mathbf{x}\}|\geq q^{n-\ell}$. Let, $\mathcal{D}_{\mathbf{x}}\equiv\{\mathbf{y}\in\mathbb{F}_q^n : f(\mathbf{y})=\mathbf{x}\}$ be the RDSS with recovery functions,

$$f_i(\{X_j, j\in N(i)\})\equiv g_i(\mathbf{x},\{X_j, j\in N(i)\}).$$

■

The second lemma might be of more interest as it is a bit less obvious.

Lemma 4: If there exists an RDSS code $\mathcal{C}$ of dimension k for a distributed storage graph G on n vertices, then there exists an index code of length $n-k+\log_q\min\{n\ln q, 1+k\ln q\}$ for the side information graph G.

Combining these two lemmas we get the proof of Theorem 2 immediately.

To prove Lemma 4 , we need the help of two other lemmas. First of all notice that, translation of any RDSS code is an RDSS code.

Lemma 5: Suppose, $\mathcal{C}\subseteq\mathbb{F}_q^n$ is an RDSS code. Then any known translation of $\mathcal{C}$ is also an RDSS code of same dimension. That is, for any $\mathbf{a}\in\mathbb{F}_q^n$, $\mathcal{C}+\mathbf{a}\equiv\{\mathbf{y}+\mathbf{a}:\mathbf{y}\in\mathcal{C}\}$ is an RDSS code of dimension $\log_q|\mathcal{C}|$.

Proof: Let, $(X_1,\ldots,X_n)\in\mathcal{C}$. Also assume, $\mathbf{a}=(a_1,\ldots,a_n)$, and $X_i'=X_i+a_i$. We know that, there exist recovery functions such that, $X_i=f_i(\{X_j : j\in N(i)\})$. Now, $X_i'=X_i+a_i=f_i(\{X_j : j\in N(i)\})+a_i\equiv f_i'(\{X_j' : j\in N(i)\}$. ■

The proof of Lemma 4 crucially use the existence of a *covering* of the entire $\mathbb{F}_q^n$, by translations of an RDSS code. Indeed, we have the following result.

Lemma 6: Suppose, $\mathcal{C}\in\mathbb{F}_q^n$ is an RDSS code for a graph G. There exists m vectors $\mathbf{x}_j\in\mathbb{F}_q^n, j=1,\ldots,m$, such that

$$\cup_{i=1}^m(\mathcal{C}+\mathbf{x}_i)=\mathbb{F}_q^n$$

where

$$m=\frac{q^n}{|\mathcal{C}|}\min\{n\ln q, 1+\ln|\mathcal{C}|\}.$$

Proof: Suppose, $\mathbf{x}_i, i=1,\ldots,m$ are randomly and independently chosen from $\mathbb{F}_q^n$. Now, the expected number of points in the space not covered by any of the translations is at most $q^n(1-|\mathcal{C}|/q^n)^{m'}<q^n e^{-m'|\mathcal{C}|/q^n}\leq 1$, when we set $m'=q^{n-k}n\ln q\leq m$ in the above expression (see [2, Prop. 3.12]).

If, instead we set $m'=\frac{q^n}{|\mathcal{C}|}\ln|\mathcal{C}|$ then the expected number of points, that do not belong to any of the m' translations is at most $\frac{q^n}{|\mathcal{C}|}$. To cover all these remaining points we need at most $\frac{q^n}{|\mathcal{C}|}$ other translations. Hence, there must exists a covering such that $\frac{q^n}{|\mathcal{C}|}\ln|\mathcal{C}|+\frac{q^n}{|\mathcal{C}|}=\frac{q^n}{|\mathcal{C}|}(\ln|\mathcal{C}|+1)\leq m$ translations suffice. ■

Using Lemmas 5 and 6 we now prove Lemma 4.

Proof of Lemma 4: Lemmas 5 and 6 show that there exist, $\mathcal{C}_1,\ldots,\mathcal{C}_m$, $\mathcal{C}_i\subseteq\mathbb{F}_q^n, i=1,\ldots,m$, all of which are RDSS codes of dimension k such that

$$\cup_{i=1}^m\mathcal{C}_i=\mathbb{F}_q^n, \tag{8}$$

where $m=q^{n-k}\min\{n\ln q, 1+k\ln q\}$. Indeed, $\mathcal{C}_i$ can set to be equal to $\mathcal{C}+\mathbf{x}_i$, which is an RDSS code by Lemma 5.

Now, any $\mathbf{y}\in\mathbb{F}_q^n$ must belong to at least one of the C_is. Suppose, $\mathbf{y}\equiv(Y_1,\ldots,Y_n)\in\mathbb{F}_q^n$ and $\mathbf{y}\in C_i$. Then, the encoding function of the desired index code $\mathcal{D}$ is simply given by, $f(\mathbf{y})=i$. If the recovery functions of $\mathcal{C}_i$ are $f_j^i, j=1,\ldots,n$, then, the decoding functions of $\mathcal{D}$ are given by:

$$g_j(i,\{Y_l : l\in N(j)\})=f_j^i(\{Y_l : l\in N(j)\}).$$

Clearly the length of the index code is $\log_q m=n-k+\log_q(\min\{n\ln q, 1+k\ln q\})$. ■

The most crucial step in the proof of Thm. 2 is Lemma 6, that show existence of a desired set of points in $\mathbb{F}_q^n$: we need to show the existence of a *covering* of the entire $\mathbb{F}_q^n$, by translations of an RDSS code. Next we show that stronger statements in lieu of Lemma 6 is possible: the translations themselves form a linear subspace. This leads to a derandomization and ease of decoding of the index code in each of the receivers.

C. Refinements of Lemma 6 and decoding of index code

In this section, we show that the m points whose existence is guaranteed by Lemma 6 can be made to satisfy some extra properties. In particular, when $q=2$, any randomly chosen *linear* subspace of dimension $\log_2 m$ suffices for our purpose with high probability.

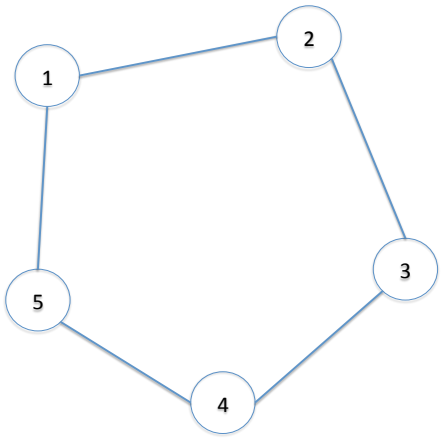

Fig. 2. A distributed storage graph (the pentagon) that shows $\mathbb{CAP}(G) \neq n - \text{INDEX}(G)$.

Definition 5: Given a set of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_\ell$ from $\mathbb{F}_q^n$, define the *binary span* of the set to be $\{\sum_{i=1}^{\ell} a_i \mathbf{x}_i : (a_1, \ldots, a_\ell) \in \{0,1\}^\ell\}$.

Lemma 7: Suppose, $\mathcal{C} \subseteq \mathbb{F}_q^n$ is an RDSS code for a graph G. There exists a set of $\ell = \log_2 \frac{q^n}{|\mathcal{C}|} + \log_2(\min\{n \ln q, \ln(e|\mathcal{C}|)\}) = \log_2 \frac{q^n}{|\mathcal{C}|} + O(\log n)$ vectors, whose binary span $\mathcal{D} \subseteq \mathbb{F}_q^n$ is such that

$$\cup_{\mathbf{x} \in \mathcal{D}} (\mathcal{C} + \mathbf{x}) = \mathbb{F}_q^n. \tag{9}$$

To prove this lemma we construct a greedy algorithm that chooses about $\log_2 m$ vectors recursively instead of m random vectors of Lemma 6. The proof is deferred to the appendix. The greedy covering argument that we employ in the proof was used to show the existence of good linear covering codes in [15] (see, also, [13], [21], [32]). We can use Lemma 7 instead of Lemma 6 to complete the proof of Lemma 4. Lemma 7 gives some algorithmic advantage in decoding an index code that we explain next.

Suppose $\mathcal{C}$ is an RDSS code with known recovery functions. Let $\mathcal{D}$ be the set of vectors promised in Lemma 6 such that $\cup_{\mathbf{x} \in \mathcal{D}} (\mathcal{C} + \mathbf{x}) = \mathbb{F}_q^n$. Consider next the corresponding index code constructed in the proof of Lemma 4. Given any $\mathbf{y} \in \mathbb{F}_q^n$ as input, the encoding of this index code finds a $\mathbf{z} \in \mathcal{D}$ such that $\mathbf{y} \in \mathcal{C} + \mathbf{z}$. A bijection $\psi : \mathcal{D} \to \mathbb{F}_q^l$ that maps $\mathbf{z}$ to a q-ary vector of length $l = \log_q |\mathcal{D}|$ completes the encoding of the index coding (here l is the length of the index code[4]). In short, the encoding of the index code maps $\mathbf{y}$ to $\psi(\mathbf{z})$ for an $\mathbf{z} : \mathbf{y} \in \mathcal{D} + \mathbf{z}$. Now for decoding of this index code, one first needs to map back any given encoded vector $\mathbf{u} \in \mathbb{F}_q^l$ to $\mathbf{x}' \equiv \psi^{-1}(\mathbf{u}) \in \mathcal{D}$, and then use the recovery functions of the RDSS code $\mathcal{C} + \mathbf{x}'$. The recovery functions of RDSS code $\mathcal{C} + \mathbf{x}'$ is known, as they are known for the RDSS code $\mathcal{C}$.

[4]We assume l to be an integer, which not necessarily is the case. The argument remains the same when l is not an integer, except for the fact that we have to deal with ceiling and floor functions.

In the above decoding of index code, we must maintain a look-up table of size exponential in n, that stores the bijective map ψ^{-1} between $\mathbb{F}_q^l$ to $\mathcal{D}$. This map tells us recovery function of which RDSS code to use (among all the translations). However, using Lemma 7 this constraint can be removed.

Assume $g : \mathbb{F}_q^{\log_q |\mathcal{D}|} \to \mathbb{F}_2^\ell$ is an arbitrary polynomial time bijective mapping that produces a binary sequence from a q-ary sequence. There are many such mappings that can be trivially constructed. Using Lemma 7, $\mathcal{D}$ is the binary span of $\ell = \log_2 |\mathcal{D}|$ vectors $\{\mathbf{d}_1, \ldots, \mathbf{d}_\ell\}$ such that $\cup_{\mathbf{x} \in \mathcal{D}} (\mathcal{C} + \mathbf{x}) = \mathbb{F}_q^n$. Then the decoding of the obtained index code can be performed from $\mathbf{u} \in \mathbb{F}_q^{\log_q |\mathcal{D}|}$ in two steps. First, suppose $g(\mathbf{u}) = (a_1, \ldots, a_\ell)$. Next, we compute $\mathbf{x}' = \sum_{i=1}^{\ell} a_i \mathbf{d}_i$. For the decoding of the index code, we now use the recovery functions of $\mathcal{C} + \mathbf{x}'$. The map from $\mathbf{u}$ to $\mathbf{x}'$ defines ψ^{-1} in this case. Hence, we no longer need to maintain a look-up table, and the required RDSS code, that we need to decode, can be found in polynomial time.

Remark 1: Note that, a random subset of $\mathbb{F}_q^n$, generated as the binary span of $\log_2 m$ random and uniformly chosen vectors from $\mathbb{F}_q^m$, satisfies Eq. (9) with high probability. This can be proved along the line of [8], [14] where it was shown almost all linear codes are good covering codes.

Given an RDSS code, our derandomization benefits only the decoding of the obtained index code, and not the encoding. But also notice that, encoding is performed by the broadcaster in one place, while decoding is performed in every receiver (that is likely to have less computational power compared to the broadcaster).

IV. Algorithmic results and constructions of RDSS codes

In this section we provide some constructions of RDSS codes, both for directed and undirected graphs. First, note that, existential results similar to Gilbert-Varshamov bound for codes can be provided for RDSS codes.

Theorem 8: For the graph $G(V = \{1, \ldots, n\}, E)$, define,

$$Q_q(G) = \{\mathbf{x} \in \mathbb{F}_q^n : \exists i \text{ with } x_i \neq 0, x_j = 0 \forall j \in N(i)\}.$$

Then,

$$\mathbb{CAP}_q(G) \geq n - \log_q(|Q_q(G)| + 1).$$

Proof: Recall that, any RDSS code can be found as an independent set of the confusion graph. The confusion graph is regular with degree exactly equal to $|Q_q(G)|$. Indeed, if for $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, $\mathbf{y} = \mathbf{x} + \boldsymbol{\nu}$ for some $\boldsymbol{\nu} \in Q_q(G)$, then $\mathbf{x}$ and $\mathbf{y}$ both cannot be part of an RDSS code without violating the repair condition. Now, using Turán's Theorem, there must exist an RDSS code of size

$$\frac{q^n}{|Q_q(G)| + 1}.$$

■

$|Q_q(G)|$ can be bounded from above in a number of ways if some properties of the graph is known. We give an example next.

Example 2 (Degree distribution): Using a simple union bound for counting, we get the following:

$$|Q_q(G)| \leq q^n(q-1)\sum_{i=1}^{n}\delta_i q^{-(i+1)},$$

where δ_i is the number of vertices with degree i. This shows that, the capacity of G is at least,

$$\mathbb{CAP}_q(G) \geq -\log_q[(q-1)\sum_{i=1}^{n}\delta_i q^{-(i+1)}].$$

For a large class of networks such as the internet, world-wide-web and social networks, the empirical degree distributions δ_i have been estimated (most of the times it follows a power-law decay). Using these, the achievable storage-capacity of the networks can be approximated.

For general graphs, the union bound can be quite loose and it might be difficult to compute $|Q_q(G)|$. However, it is possible to construct codes and compute $\mathbb{CAP}_q(G)$ via deterministic algorithms using more sophisticated ways than above. We consider the cases of undirected and directed graphs separately as different algorithms are needed in these scenarios. For impossibility results, however, the technique is same: we show that there exists a large enough subset of vertices that cannot store any information on top of what the rest of the vertices already store.

A. Undirected graph

In this section, we show that for an undirected graph G, an RDSS code can be constructed in polynomial time that achieves a rate within half of what is optimal for G. In particular, if G is bipartite, then the optimal code achieving a rate equal to $\mathbb{CAP}_q(G)$ can be constructed. Hence, for undirected graph it is relatively easy to compute or approximate $\mathbb{CAP}_q(G)$.

To achieve the above goal, start with the following lemma first. Recall that, a *vertex cover* of a graph $G(V,E)$ is a subset $U \subseteq V$ such that $\forall(u,v) \in E$ either $u \in U$ or $v \in U$ or both.

Lemma 9: For any undirected graph $G(V,E)$, and any $q \geq 2$,

$$\mathbb{CAP}_q(G) \leq VC(G), \tag{10}$$

where $VC(G)$ is the size of the minimum vertex cover of G.

Proof: Suppose, $A \subset V$ is an independent set in G. Any vertex $v \in A$ has $N(v) \subseteq V \setminus A$. Hence, $\mathbb{CAP}_q(G) \leq n - |A|$. Notice, $V \setminus A$ is a vertex cover of G. When A is the largest independent set, we have, $\mathbb{CAP}_q(G) \leq VC(G)$. ∎

1) Construction of code: A *matching* in a graph $G(V,E)$ is a set of edges such that no two edges share a common vertex. The size of the largest possible matching of the graph G is denoted by $M(G)$ below. Polynomial time algorithms to find the maximum matching is well-known [18].

To store information in the graph, first we find a maximum matching $F \subset E$. Then for any $(u,v) \in F, u,v \in V$, we store the same variable in both u and v. In this way we will be able to store $M(G)$ amount of information. Whenever one vertex fails we can go to only one other vertex to retrieve the information. Hence, $M(G) \leq \mathbb{CAP}_q(G)$.

Surprisingly, this simple constructive scheme is optimum for bipartite graphs, within a factor 2 of optimum storage for arbitrary graphs and is very unlikely to get improved upon via any other constructive scheme.

First of all, we need the following well-known lemma [45].

Lemma 10: For any graph G,

$$M(G) \leq VC(G) \leq 2M(G).$$

The proof is straight-forward. To cover all the edges one must include at least one vertex from the edges of any matching. On the other hand, if both the endpoints of the edges of a maximal matching is deleted, no two other vertices can be connected (from the maximality of the matching).

Now using Lemmas 9, 10, and the discussion above, we have,

$$M(G) \leq \mathbb{CAP}_q(G) \leq VC(G) \leq 2M(G). \tag{11}$$

Hence, for any graph G, we can store via a constructive procedure $M(G) \geq \frac{1}{2}\mathbb{CAP}_q(G)$ amount of information. Indeed, for a 2-approximation, we do not even need to find the maximum matching; a maximal matching, that can be found by a simple greedy algorithm, is sufficient.

It is unlikely that anything strictly better than the matching-code above can be found for an arbitrary graph G in polynomial-time, because that would imply a better-than-2-approximation for the minimum vertex cover. Khot and Regev [25] have shown that if the unique game conjecture is true then such algorithm is not possible. Inapproximability of minimum vertex cover under milder assumptions appear in the famous paper of Dinur and Safra [17].

However for some particular classes of graphs we can do much better. Specifically if the graph G is bipartite then König's theorem asserts $M(G) = VC(G)$. Hence for a bipartite graph G, $\mathbb{CAP}_q(G) = M(G)$ and an RDSS code can be designed in polynomial time.

Other special graphs, such as planar graphs [3], [4], that have better approximation algorithms for minimum vertex cover, might also allow us to approximate $\mathbb{CAP}_q(G)$ better. We left that exercise as future work.

B. Directed graphs

Next we attempt to extend the above techniques to construct RDSS codes for directed graphs. The following proposition is a simple result that proves to be an useful converse bound.

Proposition 11: For any graph $G(V,E)$, and any $q \geq 2$,

$$\mathbb{CAP}_q(G) \leq FVS(G), \tag{12}$$

where $FVS(G)$ is the minimum number of vertices to be removed to make G acyclic (also called the minimum *feedback vertex set*.

Note that, results of [5] or [11] imply that for any directed graph G, $\text{INDEX}_q(G)$ is at least the size of the maximum acyclic induced subgraph of G. From this, and from Thm. 2, we can deduce that $\mathbb{CAP}_q(G) \leq FVS(G) + O(\log n)$. The above proposition is stronger in the sense that we get rid of the log term.

Proof of Prop. 11: Suppose, $U \subset V$ is such that the subgraph induced by U is acyclic. We first claim that, the dimension of any RDSS code in G must be at most $|V \setminus U|$. Let us prove this claim with a simple reasoning that appear in [43]. Suppose $u \in U$ is such that all edges in E that are outgoing from u has the other end in $V \setminus U$. As the induced subgraph from U is acyclic, there will always exist such vertex. Hence, whatever we store in u, must be a function of what are stored in vertices of $V \setminus U$. Now, consider the subgraph induced by $U \setminus \{u\}$. As this subgraph is also acyclic, there must exist a vertex whose content is a function of the the contents of vertices of $V \setminus U$. Proceeding as this, we deduce that, no more than $|V \setminus U|$ amount of information can be stored in the graph G.

Now consider the maximum induced acyclic subgraph of G. If the vertex set of such subgraph is U, then $|V \setminus U| = \mathrm{FVS}(G)$. Hence, $\mathbb{CAP}_q(G) \leq \mathrm{FVS}(G)$. ∎

It is not possible to construct a code by a matching, as in the case of undirected graph. In the undirected graph we could do that because, if $(u, v) \in E$, then just by replicating the symbol of u in v we can guarantee recovery for both u and v. In the case of directed graph, such recovery is possible, if we have a *directed cycle*: $u_0 \to u_1 \to \cdots \to u_{\ell-1} \to u_0$, where $u_i \in V$ and $(u_i, u_{(i+1) \bmod \ell}) \in E$ for all $0 \leq i < \ell$. We can just store one symbol in u_1, and then replicate this symbol over all vertices of the cycle. Whenever one node fails we can go to the next node in the cycle to recover what we lost.

Two cycles in the graph $G(V, E)$ will be called *vertex-disjoint* if they do not have a common vertex.

Suppose, P is a set of vertex-disjoint cycles of the graph G. Then it is possible to store $|P|$ symbols in the graph. Hence

$$\mathbb{CAP}_q(G) \geq \mathrm{VD}(G), \tag{13}$$

where $\mathrm{VD}(G)$ is the maximum number of vertex-disjoint cycles in the graph G.

At this point it would be helpful to establish a relation between $\mathrm{VD}(G)$ and $\mathrm{FVS}(G)$. Such relation appear in the work of Erdös and Pósa [20]. Namely, for any undirected graph, it was shown that $\mathrm{FVS}(G) \leq \mathrm{VD}(G) \log \mathrm{VD}(G)$. There are two bottlenecks of using this result for our purpose. First, this only holds for undirected graphs. Second, computing the optimal vertex-disjoint cycle packing is a computationally hard problem even for undirected graphs.

There are a number of efforts towards generalizing the Erdös and Pósa theorem for directed graphs culminating in [36] that shows that for directed graph there exists an increasing function $h : \mathbb{Z} \to \mathbb{Z}$ such that,

$$\mathrm{FVS}(G) \leq h(\mathrm{VD}(G)).$$

However, the function h implied in [36] can be super-exponential. Hence, for our purpose it is not of much interest.

In what follows, we show that a *fractional vertex-disjoint cover* also lead to an RDSS code. Albeit the code is *vector-linear* as opposed to the scalar codes we have been considering so far. We need the following fractional vertex-disjoint packing result of Seymour [38]. Suppose, $\mathcal{P}$ is the set of all directed cycles of $G(V, E)$. Suppose, $\phi : \mathcal{P} \to \mathbb{Q}$ assigns a rational number to every directed cycle. Let $V(C), C \in \mathcal{P}$ denote the vertices of the cycle C. We impose a condition that ϕ must satisfy,

$$\sum_{C : v \in V(C)} \phi(C) \leq 1,$$

for all $v \in G$. Under this condition we maximize the value of $\sum_{C \in \mathcal{P}} \phi(C)$ over all functions ϕ. Suppose this value is K. Then [38] asserts,

$$\mathrm{FVS}(G) \leq 4K \ln 4K \ln \log_2 4K.$$

We will now show a construction of RDSS codes using Seymour's result.

Theorem 12: Suppose in each vertex of the directed graph $G(V, E)$ it is possible to store a vector of length p, i.e., from $\mathbb{F}_q^p$, for a large enough integer p. Then, for any $q \geq 2$, it is possible to store constructively pK q-ary symbols in the graph, such that content of any vertex can be recovered from its neighbors, and

$$4K \ln 4K \ln \log_2 4K \geq \mathrm{FVS}(G) \geq \mathbb{CAP}_q(G).$$

Remark 2: We can use the method of [11], where the *complementary index coding problem*, i.e., maximization of $n - \mathrm{INDEX}_q(G)$ is studied, to prove this theorem. Perhaps their result cannot be used as a blackbox as that would lead to an extra additive error term of $O(\log \mathbb{CAP}_q(G))$, due to the gap between $n - \mathrm{INDEX}_q(G)$ and $\mathbb{CAP}_q(G)$. By a direct analysis, we can avoid this error term. However, the analysis of [11] is more complicated than the proof below. To find a vertex-disjoint packing in polynomial time the authors of [11] first constructs a so-called vertex-split graph and converts the vertex disjoint packing in to a edge-disjoint packing problem and then converts it back. It also uses crucially a result of [33] to find a fractional edge-disjoint packing. Below we follow a much simpler path.

Proof of Thm. 12: Suppose, $\mathcal{P}$ is the set of all directed cycles of $G(V, E)$, and $\phi : \mathcal{P} \to \mathbb{Q}$ is a function such that

1) $\sum_{C : v \in V(C)} \phi(C) \leq 1$, for all $v \in G$.
2) $\mathbb{CAP}_q(G) \leq 4K \ln 4K \ln \log_2 4K$, where $K = \sum_{C \in \mathcal{P}} \phi(C)$.

We know such function ϕ exists from [38] and Prop. 11. Without loss of generality, we can assume $\phi(C) = \frac{n(C)}{p}$ for all $C \in \mathcal{P}$, $n : \mathcal{P} \to \mathbb{Z}_+ \cup \{0\}$, and p is a positive integer.

Suppose we want to store a vector $x \in \mathbb{F}_q^{pK}$. In each vertex we store a vector of length at most p, i.e., content of each vertex belong to $\mathbb{F}_q^p$. These vectors are decided in the following way. We partition the coordinates of x, that is $[1, 2, \ldots, pK]$, in to $|\mathcal{P}|$ parts. Each cycle $C \in \mathcal{P}$ is assigned $n(C)$ coordinates to it. We can do such partition, because $\sum_{C \in \mathcal{P}} n(C) = pK$. For any $C \in \mathcal{P}$, the $n(C)$ coordinates assigned to C are stored in v for all $v \in V(C)$. Hence the length of the vector need to be stored in $v \in V$ is $\sum_{C : v \in V(C)} n(C) \leq p$ which is consistent with our assumption.

Now if the content of any vertex v is needed to be restored, we can use the contents of the neighboring vertices. If $v \in V(C)$, then the $n(C)$ symbols stored in v can be restored from the copy stored in the vertex u where (v, u) is an edge in C. This holds true for all $C \in \mathcal{P}$ such that $v \in V(C)$.

The function ϕ can be found by solving a linear program: maximize $\sum_{C\in\mathcal{P}}\phi(C)$, subject to $\sum_{C:v\in V(C)}\phi(C)\leq 1$, for all $v\in G$. The number of variables in the linear program is equal to the number of cycles in the graph G. The dual problem is given by means of finding a function $\psi: V\to\mathbb{Q}$ that minimizes $\sum_{v\in V}\psi(v)$ such that $\sum_{v\in V(C)}\psi(v)\geq 1$ for every directed cycle C. Although the number of constraints in this dual linear program can be exponentially large, there exists a separation oracle that can differentiate between a feasible solution and an infeasible one. For example, given any $\psi: V\to\mathbb{Q}$, one can just calculate the shortest weight cycle, $\min_{C\in\mathcal{P}}\sum_{v\in V(C)}v$, in polynomial time and check whether that is greater than 1 or not. If such separation oracle exists, then the dual linear program can be solved in polynomial time [45, p. 102]– and at the same time a primal optimal solution can also be found (by using say, ellipsoid method).

Hence, it is possible to explicitly construct the above-mentioned vector RDSS code. ∎

Subsequently, we consider multiple node failures in our storage model.

V. Multiple failures

In this section, we describe two possible generalizations of the quantity $\mathbb{CAP}_q(G)$ that are consistent with the distributed storage literature and take care of the situation when more than one server-nodes simultaneously fail.

A. Collaborative Local Repair on Graphs

The notion of *cooperative local repair* was introduced as a generalization of the definition of local recovery in [35]. In this definition, instead of one server failure, provisions for multiple server failures are kept. Next we extend this notion to distributed storage on graphs.

Given a graph $G(V=\{1,\ldots,n\},E)$, we use each vertex to store a q-ary symbol. A code $\mathcal{C}\subseteq\mathbb{F}_q^n$ is called cooperative t-RDSS code if for any set of connected vertices $U\subset V, |U|\leq t$, there exist deterministic functions $f_i^U, i\in U$ such that for any codeword $(X_1,\ldots,X_n)\in\mathcal{C}$, $X_i=f_i^U(X_{\cup_{l\in U}N(l)\setminus U})$ for all $i\in U$. This means that if any set of t or less connected vertices fail, then one should be able to recover them from the neighbors of that set.

Note that, it is necessary in the definition to consider all sets of size less than t as well, because the local recovery of any set $U, |U|=t$ does not imply that all proper subsets of U are locally recoverable (i.e., not all neighbors of U are neighbors of a given vertex in U).

The reason it is sufficient to consider connected sets for the definition is that two disconnected sets of vertices of total size t are locally recoverable as any set less than size t is.

We below consider as example only the special case of $t=2$ for undirected graphs. In this case, apart from being a usual RDSS code, the code must also be able to deal with the case when both vertices of an edge fail. Hence the construction based on *matching* of Sec. IV-A will not work. Instead, for our first result, we will need the following definition.

A k-*path* in a graph is a set of vertices $v_1,v_2,\ldots,v_k$ such that (v_i,v_{i+1}) is an edge in the graph for all $1\leq i\leq k-1$. A subset S of vertices, such that for any k-path $\{v_1,v_2,\ldots,v_k\}$ of the graph at least one of v_is must belong to S, is called a k-*path vertex cover* [9].

Proposition 13: Suppose, given an undirected graph $G(V,E), |V|=n$, $S\subset V$ is the smallest 3-path vertex cover. Then the dimension of any cooperative 2-RDSS code is at most $|S|$.

Proof: Assume, $W\subset V$ is such that every vertex in the the induced subgraph of W has degree 1 or 0. Such sets are called *dissociation set* and the size of smallest dissociation set is called the *dissociation number* [47]. From the definition of cooperative 2-RDSS codes, content of any vertex of W can be reconstructed from vertices outside of W. Then the dimension of any cooperative 2-RDSS code is at most $n-|W|$. On the other hand, $V\setminus W$ is such that for any $u,v,w\in V$: $(u,v),(v,w)\in E$, at least one of u,v or w is in $V\setminus W$. ∎

In other words, the dimension of any cooperative 2-RDSS code is at most n minus the dissociation number. It is possible to find all vertex-disjoint 3-paths in a graph G in polynomial time [46]. Note that the smallest 3-path vertex cover must contain at least one vertex from any 3-path. This allows us to construct a cooperative 2-RDSS code that has dimension at least one-third of what is optimal possible. Indeed, we just repeat the same variable in all three vertices of a 3-path.

To generalize the above procedure beyond 2 erasures becomes cumbersome and also leads to substantial loss in the dimension of RDSS codes. Instead, in the following, we consider the usual scenario where a provision of recovery from catastrophic failures is included via minimum distance of the code.

B. Considerations for Minimum distance

Inclusion of the *minimum distance* as a necessary parameter in a locally repairable code is the norm in distributed storage [22]. In this subsection, on the RDSS codes, we further impose the constraint of minimum distance between the codewords. Given a graph $G(V,E)$ an *RDSS code with distance* d is an RDSS code $\mathcal{C}\subseteq\mathbb{F}_q^{|V|}$ such that for any $\mathbf{x},\mathbf{y}\in\mathcal{C}$, the Hamming distance between them, $d_H(\mathbf{x},\mathbf{y})\geq d$.

By abusing notations slightly, for any graph $G(V,E)$ and any $U\subset V$, define $N(U)$ to be the set of all vertices in $V\setminus U$ that has at least one (incoming) edge from U. We have the following proposition.

Theorem 14: For any graph $G(V,E)$, suppose there exists an RDSS code with distance d and dimension k. Then,

$$d\leq |V|-k+1-\max_{U\in\mathcal{I}(G):|N(U)|\leq k-1}|U|, \tag{14}$$

where for an undirected graph $\mathcal{I}(G)$ is the set of all independent sets of G and for directed graphs $\mathcal{I}(G)$ is the set of vertex-sets of all induced acyclic subgraphs of G.

When no local recovery property is required, the graph G can be thought as a complete graph. In that case, the above bound reduces to the well-known Singleton bound of coding theory. When no distance property is required (i.e., $d=1$), the bound reduces to

$$k\leq |V|-\max_{U\in\mathcal{I}(G):|N(U)|\leq k-1}|U|. \tag{15}$$

We claim that this implies Equations (10) or (12) (for the cases of undirected and directed graphs respectively). Let us show this for the case of undirected graphs as the case of directed graph is analogous. Assume that (15) is satisfied but $k > VC(G)$. However this means that for the largest independent set $U^* \subset V$, $|N(U^*)| \leq |V \setminus U^*| = VC(G) < k$. Hence, from (15), we have $k \leq |V| - |U^*| = VC(G)$, which is a contradiction. Hence, $k \leq VC(G)$.

Finally, when the graph is regular with degree r, the bound of (14) becomes (1), as an independent set (or acyclic induced subgraph) U' of size $\left\lfloor \frac{k-1}{r} \right\rfloor = \left\lceil \frac{k}{r} \right\rceil - 1$ is guaranteed to exist via Turán's theorem. Indeed, Turán's theorem guarantees existence of an independent set of size $\frac{|V|}{r+1}$. Hence $k \leq |V| - \frac{|V|}{r+1} = \frac{|V|r}{r+1}$. We therefore have, $\frac{|V|}{r+1} \geq \frac{k}{r} > \left\lfloor \frac{k-1}{r} \right\rfloor$. This guarantees existence of the independent set U'. Note that, $N(U') \leq k-1$ as the graph has degree r.

Proof of Thm. 14: The proof follows a generalization of the proof of Eq. 1 from [10], [43]. Below we provide the proof for undirected graphs which extends straightforwardly to directed graphs.

Let $\mathcal{C} \in \mathbb{F}_q^n, n = |V|$ be an RDSS code with distance d and dimension k for the graph G. For any $I \subseteq V$, let $\mathcal{C}_I$ denote the restriction of codewords of $\mathcal{C}$ to the vertices of I.

Suppose, $U \subset V$ is the largest independent set such that $|N(U)| \leq k-1$. Let R be the $k-1$ sized subset that is formed by the union of $N(U)$ and any arbitrary $k-1-|N(U)|$ vertices. Hence,

$$|\mathcal{C}_{U \cup R}| \leq q^{k-1},$$

which imply d must be at most $n - |U \cup R|$. On the other hand $|U \cup R| = |U| + k - 1$. This proves the theorem. ∎

The bound of (14) can be made to be dependent on per node storage, or the alphabet size q. Indeed, we can have the following proposition.

Proposition 15: For any q-ary RDSS code on $G(V, E)$ with distance d and dimension k,

$$k \leq \min_{U \in \mathcal{I}(G)} |N(U)| + \log_q \mathcal{A}_q(|V| - |U \cup N(U)|, d), \quad (16)$$

where, $\mathcal{A}_q(n, d)$ is the maximum size of a q-ary error-correcting code of length n and distance d, and $\mathcal{I}(G)$ is defined in Thm. 14.

Proof: As before, let $\mathcal{C} \in \mathbb{F}_q^n, n = |V|$ be an RDSS code with distance d and dimension k for the graph G. We have, for any $U \in \mathcal{I}(G)$,

$$\mathcal{C}_{U \cup N(U)} \leq q^{|N(U)|}.$$

Hence, there must exist an $x \in \mathbb{F}_q^{|U \cup N(U)|}$, such that $\mathcal{D} \triangleq |\{y \in \mathcal{C} : y_{U \cup N(U)} = x\}| \geq q^{k - |N(U)|}$. Since, $\mathcal{D}$ is a code with length $|V| - |U \cup N(U)|$ and minimum distance d, we must have,

$$k - |N(U)| \leq \log_q \mathcal{A}_q(|V| - |U \cup N(U)|, d).$$

∎

A very elegant construction of locally repairable codes appeared in [43] that achieves the bound of (1). That construction can also be used for RDSS codes with distance. We outline it below.

For construction of index codes, the *clique partition* method is a well-known heuristic [12]. This method can be easily adopted for construction of RDSS codes. Given a graph $G(V, E)$ the set of vertices are partitioned into minimum number of subsets, such that the subgraph induces by any subset is a complete subgraph (or in to minimum number of cliques). If the size of any one clique is t, then it is possible to store $t-1$ symbols in the vertices of the clique, with recovery guarantee from neighbors in case of single failure. In this way it is possible to construct an RDSS code with dimension $n - CL(G)$, where $CL(G)$ is the minimum number of cliques that partition V. The construction of Sec. IV-A via *matching* is a special case of clique-partition. However, as a downside, the problem of minimum clique-partition is NP-complete (while there exists polynomial-time algorithms for matching).

On the other hand, if a clique-partition of a graph is given, then it is possible to construct an RDSS code with distance d for that graph. Suppose, $V = V_1 \sqcup \cdots \sqcup V_\ell$ such that the induced subgraphs on $V_1, V_2, \ldots, V_\ell$ are all cliques. Suppose $r_{\min} \geq 2$ is the size of the smallest clique in the partition. By [43], it is possible to construct a locally repairable code with locality $r = r_{\min} - 1$ and length n. Such code is a RDSS code with distance d for G.

This construction will be good if the sizes of the cliques in the partition $V_1, V_2, \ldots, V_\ell$ do not vary. If there is large discrepancy among the sizes then it is still be possible to use the methods of [43]. In particular, a method of constructing locally repairable code with disjoint repair groups of different sizes have been proposed in [43, Thm. 5.3], that can be adapted straight-forwardly for this scenario.

Acknowledgements: We thank A. Agarwal, A. G. Dimakis and K. Shanmugam for useful references. We also thank B. Saha for pointing out the constructive nature of Thm. 12.

REFERENCES

[1] N. Alon, E. Lubetzky, U. Stav, A. Weinstein, and A. Hassidim. Broadcasting with side information. In *Foundations of Computer Science, 2008 (FOCS'08), 49th Annual Symposium on*, pages 823–832. IEEE, 2008.

[2] L. Babai. Automorphism groups, isomorphism, and reconstruction, chapter 27 of handbook of combinatorics. *North-Holland–Elsevier*, pages 1447–1540, 1995.

[3] B. S. Baker. Approximation algorithms for np-complete problems on planar graphs. *Journal of the ACM (JACM)*, 41(1):153–180, 1994.

[4] R. Bar-Yehuda and S. Even. On approximating a vertex cover for planar graphs. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 303–309. ACM, 1982.

[5] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol. Index coding with side information. In *Foundations of Computer Science, 2006 (FOCS'06), 47th Annual Symposium on*, pages 197–206. IEEE, 2006.

[6] Y. Birk and T. Kol. Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients. *IEEE transactions on information theory*, 52(6):2825–2830, 2006.

[7] A. Blasiak, R. Kleinberg, and E. Lubetzky. Lexicographic products and the power of non-linear network coding. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 609–618. IEEE, 2011.

[8] V. M. Blinovskii. Covering the Hamming space with sets translated by linear code vectors. *Probl. Inform. Transm*, 26:196–201, 1990.

[9] B. Brešar, F. Kardoš, J. Katrenič, and G. Semanišin. Minimum k-path vertex cover. *Discrete Applied Mathematics*, 159(12):1189–1195, 2011.

[10] V. Cadambe and A. Mazumdar. An upper bound on the size of locally recoverable codes. In *Proc. IEEE Int. Symp. Network Coding*, June 2013.

[11] M. A. R. Chaudhry, Z. Asad, A. Sprintson, and M. Langberg. On the complementary index coding problem. In *Information Theory Proceedings (ISIT), 2011 IEEE International Symposium on*, pages 244–248. IEEE, 2011.

[12] M. A. R. Chaudhry and A. Sprintson. Efficient algorithms for index coding. In *INFOCOM Workshops 2008, IEEE*, pages 1–4. IEEE, 2008.

[13] G. Cohen. A nonconstructive upper bound on covering radius. *Information Theory, IEEE Transactions on*, 29(3):352–353, 1983.

[14] G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering codes*, volume 54. Access Online via Elsevier, 1997.

[15] P. Delsarte and P. Piret. Do most binary linear codes achieve the goblick bound on the covering radius?(corresp.). *Information Theory, IEEE Transactions on*, 32(6):826–828, 1986.

[16] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Trans. Inform. Theory*, 56(9):4539–4551, Sep. 2010.

[17] I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, pages 439–485, 2005.

[18] J. Edmonds. Paths, trees, and flowers. In *Classic Papers in Combinatorics*, pages 361–379. Springer, 1987.

[19] S. El Rouayheb, A. Sprintson, and C. Georghiades. On the index coding problem and its relation to network coding and matroid theory. *Information Theory, IEEE Transactions on*, 56(7):3187–3195, 2010.

[20] P. Erdős and L. Pósa. On independent circuits contained in a graph. *Canad. J. Math*, 17:347–352, 1965.

[21] T. J. Goblick. *Coding for a discrete information source with a distortion measure*. PhD thesis, Massachusetts Institute of Technology, 1963.

[22] P. Gopalan, C. Huang, H. Simitci, and S. Yekhanin. On the locality of codeword symbols. *IEEE Trans. Inform. Theory*, 58(11):6925–6934, Nov. 2012.

[23] W. Haemers. An upper bound on the shannon capacity of a graph. *Algebraic methods in Graph Theory*, 25:267–272, 1978.

[24] G. M. Kamath, N. Prakash, V. Lalitha, and P. V. Kumar. Codes with local regeneration. *arXiv preprint arXiv:1211.1932*, 2012.

[25] S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[26] J. Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *6th Prague conference on information theory*, pages 411–425, 1973.

[27] M. Langberg and A. Sprintson. On the hardness of approximating the network coding capacity. In *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*, pages 315–319. IEEE, 2008.

[28] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975.

[29] E. Lubetzky and U. Stav. Nonlinear index coding outperforming the linear optimum. *Information Theory, IEEE Transactions on*, 55(8):3544–3551, 2009.

[30] A. Mazumdar. Achievable schemes and limits for local recovery on a graph. In *Communication, Control, and Computing (Allerton), 2014 52nd Annual Allerton Conference on*, pages 909–913. IEEE, 2014.

[31] A. Mazumdar. On a duality between recoverable distributed storage and index coding. In *Information Theory, International Symposium on*, pages 1977–1981. IEEE, 2014.

[32] A. Mazumdar, R. M. Roth, and P. O. Vontobel. On linear balancing sets. *Advances in Mathematics of Communications (AMC)*, 4(3):345–361, 2010.

[33] Z. Nutov and R. Yuster. Packing directed cycles efficiently. In *Mathematical Foundations of Computer Science 2004*, pages 310–321. Springer, 2004.

[34] D. S. Papailiopoulos and A. G. Dimakis. Locally repairable codes. In *Proc. Int. Symp. Inform. Theory*, pages 2771–2775, Cambridge, MA, July 2012.

[35] A. S. Rawat, A. Mazumdar, and S. Vishwanath. On cooperative local repair in distributed storage. In *Information Sciences and Systems (CISS), 2014 48th Annual Conference on*, pages 1–5. IEEE, 2014.

[36] B. Reed, N. Robertson, P. Seymour, and R. Thomas. Packing directed circuits. *Combinatorica*, 16(4):535–554, 1996.

[37] S. Riis. Graph entropy, network coding and guessing games. *arXiv preprint arXiv:0711.4175*, 2007.

[38] P. D. Seymour. Packing directed circuits fractionally. *Combinatorica*, 15(2):281–288, 1995.

[39] K. Shanmugam. personal communication, June 16 2015.

[40] K. Shanmugam and A. G. Dimakis. Bounding multiple unicasts through index coding and locally repairable codes. In *Information Theory Proceedings (ISIT), 2014 IEEE International Symposium on*, pages 296–300. IEEE, 2014.

[41] C. E. Shannon. The zero error capacity of a noisy channel. *Information Theory, IRE Transactions on*, 2(3):8–19, 1956.

[42] N. Silberstein, A. S. Rawat, O. O. Koyluoglu, and S. Vishwanath. Optimal locally repairable codes via rank-metric codes. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 1819–1823. IEEE, 2013.

[43] I. Tamo and A. Barg. A family of optimal locally recoverable codes. *arXiv preprint arXiv:1311.3284*, 2013.

[44] I. Tamo, D. S. Papailiopoulos, and A. G. Dimakis. Optimal locally repairable codes and connections to matroid theory. In *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, pages 1814–1818. IEEE, 2013.

[45] V. V. Vazirani. *Approximation algorithms*. springer, 2001.

[46] R. Williams. Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, 2009.

[47] M. Yannakakis. Node-deletion problems on bipartite graphs. *SIAM Journal on Computing*, 10(2):310–327, 1981.

APPENDIX

A. Proof of Lemma 7

We prove following general statement below which will imply Lemma 7.

Proposition 16: For every subset $\mathcal{F} \subseteq \mathbb{F}_q^n$, there exists a set $\mathcal{D} \in \mathbb{F}_q^n$ that is binary span of $\log_2(q^n|\mathcal{F}|^{-1} \min\{n \ln q, 1 + \ln|\mathcal{F}|\})$ vectors and

$$\cup_{x\in\mathcal{D}}(\mathcal{F}+x) = \mathbb{F}_q^n.$$

The proof is contingent on the following result from classical coding theory.

Lemma 17 (Bassalygo-Elias): Suppose, $\mathcal{C}, \mathcal{B} \subseteq \mathbb{F}_q^n$. Then,

$$\sum_{x\in\mathbb{F}_q^n} |\ (\mathcal{C}+x)\cap\mathcal{B}\ | = |\mathcal{C}||\mathcal{B}|. \tag{17}$$

Proof:

$$\begin{aligned}\sum_{x\in\mathbb{F}_q^n} |\ (\mathcal{C}+x)\cap\mathcal{B}\ | &= |\{(x,y) : x\in\mathbb{F}_q^n, y\in\mathcal{B}, y\in\mathcal{C}+x\}|\\ &= |\{(x,y) : x\in\mathbb{F}_q^n, y\in\mathcal{B}, x\in y-\mathcal{C}\}|\\ &= |\{(x,y) : y\in\mathcal{B}, x\in y-\mathcal{C}\}|\\ &= |\mathcal{B}||y-\mathcal{C}| = |\mathcal{C}||\mathcal{B}|,\end{aligned}$$

where $y-\mathcal{C} \equiv \{y-a : a\in\mathcal{C}\}$. ∎

Now, for any set $\mathcal{F} \subseteq \mathbb{F}_q^n$, define

$$Q(\mathcal{F}) \equiv 1 - \frac{|\mathcal{F}|}{q^n}. \tag{18}$$

In words, $Q(\mathcal{F})$ denote the proportion of $\mathbb{F}_q^n$ that is not covered by $\mathcal{F}$. The following property is a result of Lemma 17.

Lemma 18: For every subset $\mathcal{F} \subseteq \mathbb{F}_q^n$,

$$q^{-n}\sum_{x\in\mathbb{F}_q^n} Q(\mathcal{F}\cup(\mathcal{F}+x)) = Q(\mathcal{F})^2. \tag{19}$$

Proof: We have,

$$|\mathcal{F}\cup(\mathcal{F}+x)| = 2|\mathcal{F}| - |\mathcal{F}\cap(\mathcal{F}+x)|.$$

Therefore,

$$Q(\mathcal{F}\cup(\mathcal{F}+x)) = 1 - 2|\mathcal{F}|q^{-n} + |\mathcal{F}\cap(\mathcal{F}+x)|q^{-n},$$

and hence,

$$q^{-n}\sum_{x\in\mathbb{F}_q^n} Q(\mathcal{F}\cup(\mathcal{F}+x)) = 1 - 2|\mathcal{F}|q^{-n}$$

$$+ q^{-2n} \sum_{x \in \mathbb{F}_q^n} |\mathcal{F} \cap (\mathcal{F} + x)|$$
$$= 1 - 2|\mathcal{F}|q^{-n} + q^{-2n}|\mathcal{F}|^2$$
$$= (1 - |\mathcal{F}|q^{-n})^2,$$

where in the second line we have used Lemma 17. ∎

Now we are ready to proof Prop. 16.

Proof of Prop. 16: From Lemma 18, for every subset $\mathcal{F} \subseteq \mathbb{F}_q^n$, there exists $x \in \mathbb{F}_q^n$ such that

$$Q(\mathcal{F} \cup (\mathcal{F} + x)) \leq Q(\mathcal{F})^2.$$

For the set $\mathcal{F} \equiv \mathcal{F}_0$, recursively define, for i =1, 2,...

$$\mathcal{F}_i = \mathcal{F}_{i-1} \cup (\mathcal{F}_{i-1} + z_{i-1}),$$

where $z_i \in \mathbb{F}_q^n$ is such that,

$$Q(\mathcal{F}_i \cup (\mathcal{F}_i + z_i)) \leq Q(\mathcal{F}_i)^2, \quad i = 0, 1, \ldots$$

Clearly,

$$Q(\mathcal{F}_t) \leq Q(\mathcal{F}_0)^{2^t} = \left(1 - q^{-n}|\mathcal{F}|\right)^{2^t} \leq e^{-q^{-n}|\mathcal{F}|2^t}.$$

At this point we can just use the argument of the proof of Lemma 6, with 2^t playing the role of m' therein.

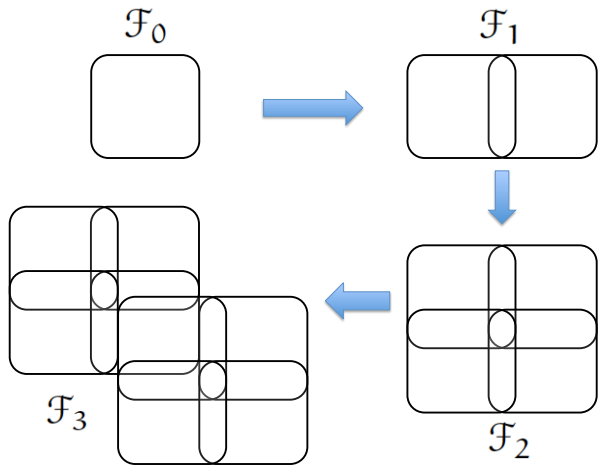

Fig. 3. The recursive construction of the sets $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ of Prop. 16.

On the other hand $\mathcal{F}_t$ contains $\mathcal{F}_0$ and its $2^t - 1$ translations (see, Figure 3 for an illustration). Hence, there exists $m = \min\left\{\frac{q^n n \ln q}{|\mathcal{F}|}, \frac{q^n(1+\ln|\mathcal{F}|)}{|\mathcal{F}|}\right\}$ vectors $x_0 = 0, x_1, x_2, \ldots, x_{m-1} \in \mathbb{F}_q^n$, such that

$$\cup_{i=0}^{m-1}(\mathcal{F} + x_i) = \mathbb{F}_q^n.$$

These m vectors form the binary span of the t vectors we have chosen via the greedy procedure. ∎

Arya Mazumdar (S'05-M'13) is an assistant professor in University of Minnesota-Twin Cities (UMN). Before coming to UMN, he was a postdoctoral scholar at the Massachusetts Institute of Technology. He received the Ph.D. degree from University of Maryland, College Park, in 2011.

Arya is a recipient of the NSF CAREER award, 2015 and the 2010 IEEE ISIT Student Paper Award. He is also the recipient of the Distinguished Dissertation Fellowship Award, 2011, at the University of Maryland. He spent the summers of 2008 and 2010 at the Hewlett-Packard Laboratories, Palo Alto, CA, and IBM Almaden Research Center, San Jose, CA, respectively. Arya's research interests include error-correcting codes, information theory and their applications.