Achievable Schemes and Limits for Local Recovery on a Graph

Arya Mazumdar
Department of ECE
University of Minnesota– Twin Cities
Minneapolis, MN 55455
email: arya@umn.edu

***Abstract*—Recently, a graph-theoretic model for a single-failure-recoverable distributed storage system was proposed. 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. Here we provide bounds and constructive schemes for data storage in such networks. We also impose an additional requirement on the codes for such model - a minimum distance guarantee. The model is further generalized for multiple node failures and cooperative repairs.**

I. Introduction

In a locally repairable code, introduced in [11], any single symbol of a codeword can be recovered from a fixed number of other symbols. Local repairability is a desirable property for application in distributed storage. If each symbol of an encoded message is stored at a different node of a storage-network, then a single node failure can be quickly repaired by accessing only few other nodes.

The central result of [11] and subsequent 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.

One particular generalization of the local repair property, that considers the topology of the network of distributed storage system appears in [13] (and also in [18], but we will adhere to the notations of [13]). In the model of [13], the architecture of the storage system is fixed and the network of storage is given by a graph. 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, the local recovery or repair condition is imposed 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 or 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 arbitrary graphs. However, here 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 (see, Section II-A). Similar statement, albeit concerning different properties, is possible for directed graphs (see, Section II-B).

The local repairability property on this 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 (see, Section III). 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 [15]. In Section III we generalize this model as well on graphs.

A. Recoverable distributed storage systems

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 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, $X_i \in \mathbb{F}_q, i = 1, \ldots, n$.

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 $(X_1, X_2, \ldots, X_n) \in \mathbb{F}_q^n$,

$$X_i = f_i(\{X_j : j \in N(i)\}), \quad i = 1, \ldots, n. \tag{2}$$

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.

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.

Using the results of [1], it can be deduced that the storage capacity of any graph is related closely to the length of *index coding* [4] for that graph. In particular, the storage capacity is only an additive logarithmic term away from the complementary index coding rate of [7].

II. Algorithmic results and constructions of RDSS codes

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

$$\mathbb{CAP}_q(G) \leq VC(G), \tag{3}$$

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 [9].

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 [20].

Lemma 2: 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 1, 2, and the discussion above, we have,

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

Hence, we can store via a constructive procedure $M(G) \geq \frac{1}{2}\mathbb{CAP}_q(G)$ amount of information for any arbitrary graph G.

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 [12] 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 [8].

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 [2], [3], 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 3: For any graph $G(V, E)$, and any $q \geq 2$,

$$\mathbb{CAP}_q(G) \leq FVS(G), \tag{4}$$

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 [4] or [7] along with [13] imply 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. 3: 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. 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 $(u_i, u_{(i+1) \mod \ell}) \in E$ 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* 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 $\mathbb{CAP}_q(G) \geq \mathrm{VD}(G)$ 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 [10]. 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 [16] 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 [16] 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 [17]. 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 [17] 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 4: 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

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

Proof: 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 [17] and Prop. 3. 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 $\mathbf{x} \in \mathbb{F}_q^{pK}$. 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 $\mathbf{x}$, 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 need 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 [20, 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. ∎

Remark 1: The above construction is comparable to that of [7], where the *complementary index coding problem* is studied. The analysis of [7] is more complicated – it converts the vertex disjoint packing in to a edge-disjoint packing problem and then converts it back – and uses crucially a result of [14] that does not hold for all graphs. Moreover, if we blindly use the result of [7], in terms of approximation guarantee there will be an extra additive error term of $O(\log n)$, due to the gap between complementary index coding rate and $\mathbb{CAP}_q(G)$. By a direct analysis, we have avoided this term above.

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

III. 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 [15]. 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 $(X_1, \ldots, X_n) \in \mathcal{C}$, $X_i = f_i^U(\{X_j : j \in \cup_{l \in U} N(l) \setminus U\})$ 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. II-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* [5].

Proposition 5: 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* [22]. 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 [21]. 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 next 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 [11]. 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 6: For any graph $G(V, E)$, suppose there exists an RDSS code with distance d and dimension k. Then,

$$d \leq |V| - k + 1 - \max_{U \in \mathcal{I}(G) : |N(U)| \leq k-1} |U|, \tag{5}$$

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 though of a complete graph. In that case the above bound reduced to the well-known Singleton bound of coding theory. When no distance property is required (i.e., $d = 1$), the bound reduces to Equations (3) or (4). Finally, when the graph is regular with degree r, the bound becomes (1), as an independent set (or acyclic induced subgraph) of size $\left\lceil \frac{k}{r} \right\rceil + 1$ is guaranteed to exist via Turán's theorem.

Proof of Thm. 6: The proof follows a generalization of the proof of Eq. 1 from [6], [19]. Below we provide the proof for undirected graphs which extends straight-forwardly 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 ant $I \subseteq V$, 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. ■

Acknowledgement: This work is supported in part by a grant from University of Minnesota.

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

[3] 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.

[4] 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.

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

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

[7] 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.

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

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

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

[11] 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.

[12] 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.

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

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

[15] 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.

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

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

[18] 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.

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

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

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

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