A Combinatorial View of the Service Rates of Codes Problem, its Equivalence to Fractional Matching and its Connection with Batch Codes

*Fatemeh Kazemi, *Esmaeil Karimi, †Emina Soljanin, and *Alex Sprintson
*ECE Dept., Texas A&M University, USA {fatemeh.kazemi, esmaeil.karimi, spalex}@tamu.edu
†ECE Dept., Rutgers University, USA {emina.soljanin}@rutgers.edu

***Abstract*— We propose a novel technique for constructing a graph representation of a code through which we establish a significant connection between the service rate problem and the well-known fractional matching problem. Using this connection, we show that the service capacity of a coded storage system equals the fractional matching number in the graph representation of the code, and thus is lower bounded and upper bounded by the matching number and the vertex cover number, respectively. This is of great interest because if the graph representation of a code is bipartite, then the derived upper and lower bounds are equal, and we obtain the capacity. Leveraging this result, we characterize the service capacity of the binary simplex code whose graph representation is bipartite. Moreover, we show that the service rate problem can be viewed as a generalization of the multiset primitive batch codes problem.**

I. Introduction

Serving a large number of users simultaneously is a major concern in cloud/edge storage systems. The service capacity has been recently recognized as an important performance metric of coded storage systems. It has a wide relevance [1], and can be interpreted as a measure of the maximum number of users that can be simultaneously served by a coded storage system [2]–[7]. Thus, maximizing the service capacity is of great significance for the emerging applications such as distributed learning and fog computing. Moreover, maximizing the service capacity reduces the users' experienced latency, particularly in a high traffic regime, which is important for the delay-sensitive applications such as live streaming, where many users wish to get the same content at the same time.

The service rate problem is concerned with a distributed storage system in which k files $f_1, \ldots, f_k$ are stored across n servers using a linear $[n, k]_q$ code such that the requests to download file f_i arrive at rate λ_i, and the server l operates at rate μ_l. A goal of the service rate problem is to determine the service rate region of this storage system which is the set of all request arrival rates $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ that can be served by this system given the finite service rate of servers. The service rate problem is generally a complex optimization problem that has been studied only in some limited cases [3]–[5]. We show that the service rate problem is equivalent to the fractional matching problem which were extensively studied in the context of graph theory. This equivalence result allows one to leverage the techniques in the rich literature of the graph theory for solving the service rate problem.

This material is based upon work supported by the National Science Foundation under Grant No. CIF-1717314.

Existing studies on data access pursue various directions. Many are focused on providing efficient storage maintenance under possible failures of a subset of nodes accessed (see e.g., [8]–[12]). These studies typically assume infinite service rate for each storage node. Hence, they do not address the problem of serving a large number of users simultaneously. Another important line of work is concerned with caching (see e.g., [13]–[16]), in which generally the limited capacity of the backhaul link is considered as the main bottleneck of the system, and the goal is usually to minimize the backhaul traffic by prefetching the popular contents at storage nodes of limited size. Thus, these works do not address the scenarios where many users want to get the same content concurrently given the limited capacity of the access part of the network. The other related body of work is concerned with minimizing the download latency (see e.g., [17]–[28]). These papers assume that the storage nodes can serve the customers at some finite rate, and aim to compute download latency for intractable queueing systems that appear in coded storage. We note now and explain in detail later that because of the constraints on the service rate of servers, maximizing the service capacity provides load balancing (see [7]). In that sense, the most relevant work to this paper includes batch codes, switch codes and PIR codes (see e.g., [29]–[34]). The problems considered in these papers, as we will show, can be often seen as special cases of the service rate problem.

Main Contributions: We first construct a special graph representation of a linear code in Sec. III-A. We then show the following results in Sec. III-D: 1) equivalence between the service rate problem and the well-known fractional matching problem and 2) equivalence between the integral service rate problem and the matching problem. These equivalence results allow us to show that the service capacity of a code equals the fractional matching number in the graph representation of a code, and thus is lower bounded and upper bounded by the matching number and the vertex cover number, respectively. This is beneficial because if the graph representation of a code is bipartite, then the upper and lower bounds are equal, which allows us to establish the service capacity of the system. Leveraging this result, we determine the service capacity of the binary simplex codes whose graph representation, as we will show, is bipartite. Furthermore, we show that the service rate problem can be viewed as a generalization of batch codes problem in Sec. IV.

In particular, we show that the multiset primitive batch codes problem is a special case of the service rate problem when the solution (the portion of requests assigned to the recovery sets) is restricted to be integral. *Due to the space constraints all proofs are omitted and can be found in [35].*

II. Coded System and its Service Rate Region

Throughout this work, we use bold-face lower-case letters for vectors and bold-face capital letters for matrices. Let $\mathbb{N}$ denote the set of positive integers. We denote the finite field with q elements by $\mathbb{F}_q$. For $i \in \mathbb{N}$, $[i] \triangleq \{1, \ldots, i\}$. For $n \in \mathbb{N}$, $\mathbf{1}_n$ denotes the all-one vector of length n.

Consider a storage system where k files $f_1, \ldots, f_k$ are stored across n servers labeled $1, \ldots, n$, using an $[n, k]_q$ code with generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$. A set of stored symbols that can be used to recover file f_i is referred to as a recovery set for file f_i. Let $\mathbf{g}_j$ be the jth column of $\mathbf{G}$. The set $R \subseteq [n]$ is a recovery set for file f_i if there exists non-zero α_j's$\in \mathbb{F}_q$ such that $\sum_{j \in R} \alpha_j \mathbf{g}_j = \mathbf{e}_i$, where $\mathbf{e}_i$ denotes the ith unit vector. In other words, a set R is a recovery set for file f_i if the unit vector $\mathbf{e}_i$ can be recovered by a linear combination of the columns of $\mathbf{G}$ indexed by the set R.

Let $t_i \in \mathbb{N}$ denote the number of recovery sets of file f_i, and $\mathcal{R}_i = \{R_{i,1}, \ldots, R_{i,t_i}\}$ denote the set of recovery sets of file f_i. We assume w.l.o.g. that the time to download a file from server $l \in [n]$ is exponential with rate $\mu_l \in \mathbb{R}_{\geq 0}$, i.e., μ_l is the average rate at which server l executes file requests. We denote the service rates of servers $1, \ldots, n$ by the vector $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$. We further assume that the arrival of requests for file f_i is Poisson with rate λ_i, $i \in [k]$. We denote the request rates for files $1, \ldots, k$ by the vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$. We consider the class of scheduling strategies that assign a fraction of requests for a file to each of its recovering sets. Let $\lambda_{i,j}$ be the portion of requests for file f_i that are assigned to the recovery set $R_{i,j}$, $j \in [t_i]$. The service rate problem seeks to determine the set of arrival rates $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ that can be served by a coded storage system with generator matrix $\mathbf{G}$ and service rate $\boldsymbol{\mu}$, referred to as service rate region $\mathcal{S}(\mathbf{G}, \boldsymbol{\mu}) \subseteq \mathbb{R}_{\geq 0}^k$.

Definition 1. *An $(\mathbf{G}, \boldsymbol{\mu})$ system is a coded storage system in which k files are stored redundantly across n servers using a linear $[n, k]_q$ code with generator matrix $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ such that file f_i for $i \in [k]$ has $t_i \in \mathbb{N}$ recovery sets denoted by $\mathcal{R}_i = \{R_{i,1}, \ldots, R_{i,t_i}\}$, and the service rate of servers in the system is $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$.*

Definition 2. *The service rate region of an $(\mathbf{G}, \boldsymbol{\mu})$ system, denoted by $\mathcal{S}(\mathbf{G}, \boldsymbol{\mu})$, is the set of vectors $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ for which there exist $\lambda_{i,j}$ satisfying the following constraints:*

$$\sum_{j=1}^{t_i} \lambda_{i,j} = \lambda_i, \qquad \textit{for all} \quad i \in [k] \tag{1a}$$

$$\sum_{i=1}^{k} \sum_{\substack{j \in [t_i] \\ l \in R_{i,j}}} \lambda_{i,j} \leq \mu_l, \quad \textit{for all} \quad l \in [n] \tag{1b}$$

$$\lambda_{i,j} \in \mathbb{R}_{\geq 0}, \qquad \textit{for all} \quad i \in [k],\ j \in [t_i] \tag{1c}$$

Note that constraints (1a) ensure that the demands for all files are served, and constraints (1b) guarantee that no node is sent requests in excess of its service rate.

Proposition 1. *[5, Lemma 1] The service rate region of an $(\mathbf{G}, \boldsymbol{\mu})$ system $\mathcal{S}(\mathbf{G}, \boldsymbol{\mu})$ is a non-empty, convex, closed, and bounded subset of the $\mathbb{R}_{\geq 0}^k$.*

The service capacity of an $(\mathbf{G}, \boldsymbol{\mu})$ system, $\lambda^\star(\mathbf{G}, \boldsymbol{\mu})$, is defined as the maximum sum of arrival rates that can be served simultaneously by the storage system. We define a maximum demand vector, denoted by $\boldsymbol{\lambda}^\star = (\lambda_1^\star, \ldots, \lambda_k^\star)$, as a vector in the service rate region for which $\sum_{i=1}^k \lambda_i^\star = \lambda^\star(\mathbf{G}, \boldsymbol{\mu})$. An instance of the maximum demand vector is obtained by solving the following linear programming (LP):

$$\max \quad \sum_{i=1}^{k} \lambda_i \qquad \text{s.t.} \qquad \text{(1a), (1b), (1c) hold.} \tag{2}$$

Definition 3. *The integral service rate region of an $(\mathbf{G}, \boldsymbol{\mu})$ system, denoted by $\mathcal{S}_I(\mathbf{G}, \boldsymbol{\mu})$, is the set of all vectors $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ for which there exist $\lambda_{i,j} \in \mathbb{Z}_{\geq 0}$ satisfying the sets of constraints* (1a) *and* (1b)*.*

Note that each demand vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ in the integral service rate region has integral coordinates, i.e., $\mathcal{S}_I(\mathbf{G}, \boldsymbol{\mu}) \subseteq \mathbb{Z}_{\geq 0}^k$. However, because of the fractional relaxation of $\lambda_{i,j}$, it is not guaranteed that the vectors with integral coordinates in the service rate region $\mathcal{S}(\mathbf{G}, \boldsymbol{\mu})$ are also in the integral service rate region $\mathcal{S}_I(\mathbf{G}, \boldsymbol{\mu})$.

Remark 1. *In the integral setting of the service rate problem where $\lambda_{i,j}$ are non-negative integers, if each server can serve up to one request at a time, i.e., $\mu_l = 1$ for all servers $l \in [n]$, then one can easily conclude that $\lambda_{i,j}$ are binary and the recovery sets used for each file f_i, $i \in [k]$, are disjoint.*

III. Equivalence to Fractional Matching

We first introduce a graph representation of a code which is useful for characterizing the service capacity of a coded storage system through relating this problem with the well-known problem of finding the maximum fractional matching in a graph. In particular, we show that the service capacity of a code is exactly equal to the fractional matching number in our graph representation of a code. Another way of determining the service capacity of a coded storage system is providing tight bounds on the maximum sum of the arrival rates that can be served by the coded storage system. We show that the matching number and the vertex cover number in the graph representation of a code, respectively are a lower bound and an upper bound on the service capacity of a code. Thus, if the graph representation of a code is a bipartite graph, according to the Duality Theorem [36], the matching number and vertex cover number are identical, and we are able to determine the capacity. As an application of this result, we determine the service capacity of the binary simplex codes whose graph representation, as we will show, is a bipartite graph. We next describe how to construct the graph representation of a code, and then we present the interesting connections.

A. Graph Representation of Codes

We focus on the settings with recovery sets of size 1 and 2 where the recovery sets for each file is either a systematic symbol or a group of two symbols. Extensions to the general case are mostly straightforward and involve hypergraphs in which each edge can be incident to an arbitrary number of vertices. The graph representation of a code with generator matrix $\mathbf{G}$ is denoted by $G(V, E)$ where the vertices in V correspond to the n encoded symbols (the servers of the storage system), and the edges in E correspond to recovery sets of files. In $G(V, E)$, each self-loop represents a recovery set of size 1 for the vertex (file) that it is connected to, and each edge between two vertices represents a recovery set of size 2 for the file that can be recovered from these two vertices. Each edge is assigned a color such that the edges that correspond to the recovery sets of the same file are assigned the same color. In that sense, we have an edge-colored graph. It should be noted that a graph with self-loops can be simply converted to a graph without any self-loops by adding sufficient number of dummy vertices (servers). We assume that the label of all dummy servers is zero and thus we denote a systematic recovery set for file f_i by $\{0, r\}$ where r is the label of the systematic server storing file f_i. Section III-C provides an example that shows the graph representation of $[7, 3]_2$ simplex code.

B. Matching and Vertex Cover Problems [36]

Definition 4. *A matching in a graph is a set of all pairwise non-adjacent edges.*

Alternatively, a matching in a graph $G(V, E)$ is an assignment of the values $\tilde{x}_e \in \{0, 1\}$ to the edges $e \in E$ in such a way that for each vertex $v \in V$, the sum of the values on the incident edges is at most 1. All the edges $e \in E$ with value $\tilde{x}_e = 1$ are in the matching. Thus, a *matching vector* in a graph $G(V, E)$ can be defined as a vector $\tilde{\boldsymbol{x}} = (\tilde{x}_e : e \in E)$ satisfying the following conditions:

$$\sum_{\text{e incident to } v} \tilde{x}_e \leq 1, \qquad \text{for all} \quad v \in V \tag{3a}$$

$$\tilde{x}_e \in \{0, 1\}, \qquad \text{for all} \quad e \in E \tag{3b}$$

Definition 5. *A maximum matching in a graph is a matching that contains the largest number of edges. The maximum matching vector is denoted by $\tilde{\boldsymbol{x}}^\star$.*

The size of a maximum matching in a graph $G(V, E)$ is called matching number, denoted by $m(G)$. There may be several instances of maximum matchings in a graph. The problem of finding an instance of maximum matching can be formulated as the following integer LP:

$$\max \sum_{e \in E} \tilde{x}_e \qquad \text{s.t.} \qquad \text{(3a), (3b) hold.} \tag{4}$$

Definition 6. *A fractional matching in a graph $G(V, E)$ is an assignment of the values $x_e \in [0, 1]$ to the edges $e \in E$ in such a way that for each vertex $v \in V$, the sum of the values on the incident edges is at most* 1.

A *fractional matching vector* in a graph $G(V, E)$ can be defined as a vector $\boldsymbol{x} = (x_e : e \in E)$ satisfying the below:

$$\sum_{\text{e incident to } v} x_e \leq 1, \qquad \text{for all} \quad v \in V \tag{5a}$$

$$x_e \geq 0, \qquad \text{for all} \quad e \in E \tag{5b}$$

Definition 7. *A maximum fractional matching, denoted by $\boldsymbol{x}^\star$, is a fractional matching vector in the graph that has the maximum value $\sum_{e \in E} x_e$ over all fractional matching vectors in the graph.*

The value of a maximum fractional matching in a graph $G(V, E)$ is called the fractional matching number, denoted by $m_f(G)$. Finding an instance of maximum fractional matching in a graph can be formulated as the following LP:

$$\max \sum_{e \in E} x_e \qquad \text{s.t.} \qquad \text{(5a), (5b) hold.} \tag{6}$$

Definition 8. *A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex in the set.*

Alternatively, a vertex cover of a graph $G(V, E)$ is an assignment of the values $y_v \in \{0, 1\}$ to the vertices $v \in V$ in such a way that for each edge $e \in E$, the sum of the values on the endpoint vertices is at least 1. All the vertices $v \in V$ with value $\tilde{y}_v = 1$ are in the vertex cover. Thus, a *vertex cover vector* of a graph $G(V, E)$ can be defined as a vector $\boldsymbol{y} = (y_v : v \in V)$ satisfying the following conditions:

$$\sum_{v \text{ incident to } e} y_v \geq 1, \qquad \text{for all} \quad e \in E \tag{7a}$$

$$y_v \in \{0, 1\}, \qquad \text{for all} \quad v \in V \tag{7b}$$

Definition 9. *A minimum vertex cover in a graph is a vertex cover that contains the minimum number of vertices.*

The cardinality of a minimum vertex cover in a graph $G(V, E)$ is called vertex cover number, denoted by $v(G)$. There may be several instances of a minimum vertex cover in a graph. Finding an instance of minimum vertex cover in a graph can be formulated as the following integer LP:

$$\min \sum_{v \in V} y_v \qquad \text{s.t.} \qquad \text{(7a), (7b) hold.} \tag{8}$$

Proposition 2. *For an arbitrary graph G, it is known that $m(G) \leq m_f(G) \leq v(G)$. For a bipartite graph G, it holds that $m(G) = m_f(G) = v(G)$.*

In what follows, we assume that each server in the coded distributed storage system can serve up to one request at each moment, i.e., $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n) = (1, \ldots, 1)$. Thus, $\mathcal{S}(\mathbf{G}, \boldsymbol{\mu})$ and $\lambda^\star(\mathbf{G}, \boldsymbol{\mu})$ only depend on the generator matrix $\mathbf{G}$ and are respectively denoted by $\mathcal{S}(\mathbf{G})$ and $\lambda^\star(\mathbf{G})$. Next, we present an example to show how the service rate of a code is connected to the matching and the vertex cover problems.

Fig. 1. A distributed storage system consists of 7 servers storing files f_1, f_2, and f_3 using a binary $[7,3]_2$ simplex code.

C. Example of Equivalence

Here, we present an example to give more intuition about the subsequent results and to provide a sketch of the proofs. Consider a distributed storage system in which files f_1, f_2, and f_3 are stored across 7 servers, labeled $1,\ldots,7$, using a binary $[7,3]_2$ simplex code with the service rate $\mu_l = 1$, $l \in [7]$. The generator matrix of this code is given as follows:

$$\mathbf{G} = \begin{array}{c} \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array},$$

where the number above each column shows the label of the corresponding column (server). Fig. 1 depicts this distributed storage system. The recovery sets for each file are given by

$$\mathcal{R}_1 = \{R_{1,1},\ldots,R_{1,4}\} = \{\{0,1\},\{2,3\},\{4,5\},\{6,7\}\}$$
$$\mathcal{R}_2 = \{R_{2,1},\ldots,R_{2,4}\} = \{\{0,2\},\{1,3\},\{4,6\},\{5,7\}\}$$
$$\mathcal{R}_3 = \{R_{3,1},\ldots,R_{3,4}\} = \{\{0,4\},\{1,5\},\{2,6\},\{3,7\}\}$$

The graph representation of $[7,3]_2$ simplex code is drawn in Fig. 2. The vertices $\emptyset_{f_1}$, $\emptyset_{f_2}$ and $\emptyset_{f_3}$ are the dummy vertices added to the graph for the purpose of removing the self-loops of systematic vertices f_1, f_2, and f_3, respectively. The edges with color magenta, green, and blue represent recovery sets for files f_1, f_2, and f_3, respectively. Moreover, the label $\lambda_{i,j}$ above an edge indicates the portion of requests for file f_i that is assigned to the recovery set $R_{i,j}$.

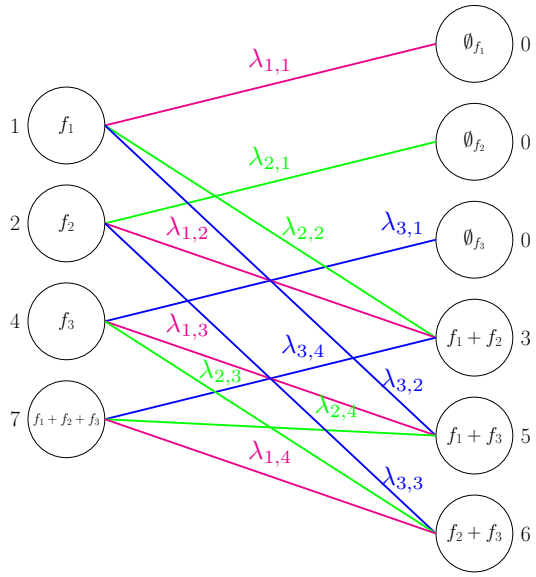

Fig. 2. Graph representation of $[7,3]_2$ simplex code.

The service rate region $\mathcal{S}(\mathbf{G})$ of this system is the set of vectors $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ for which there exist $\lambda_{i,j}$'s, $i \in [3]$ and $j \in [4]$, satisfying the constraints (1a)-(1c) as follows:

$$\text{(1a)} \Rightarrow \begin{cases} \lambda_1 = \lambda_{1,1} + \lambda_{1,2} + \lambda_{1,3} + \lambda_{1,4} \\ \lambda_2 = \lambda_{2,1} + \lambda_{2,2} + \lambda_{2,3} + \lambda_{2,4} \\ \lambda_3 = \lambda_{3,1} + \lambda_{3,2} + \lambda_{3,3} + \lambda_{3,4} \end{cases} \tag{9}$$

$$\text{(1b)} \Rightarrow \begin{cases} \lambda_{1,1} + \lambda_{2,2} + \lambda_{3,2} \le 1 \\ \lambda_{2,1} + \lambda_{1,2} + \lambda_{3,3} \le 1 \\ \lambda_{3,1} + \lambda_{1,3} + \lambda_{2,3} \le 1 \\ \lambda_{3,4} + \lambda_{2,4} + \lambda_{1,4} \le 1 \\ \lambda_{2,2} + \lambda_{1,2} + \lambda_{3,4} \le 1 \\ \lambda_{3,2} + \lambda_{1,3} + \lambda_{2,4} \le 1 \\ \lambda_{3,3} + \lambda_{2,3} + \lambda_{1,4} \le 1 \end{cases} \tag{10}$$

$$\text{(1c)} \Rightarrow \left\{ \lambda_{i,j} \in \mathbb{R}_{\ge 0}, \quad \text{for all} \quad i \in [3],\ j \in [4] \right. \tag{11}$$

Fig. 3 shows the service rate region $\mathcal{S}(\mathbf{G})$ of this system.

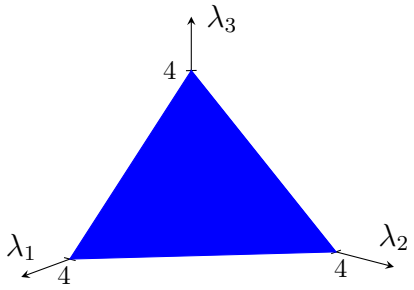

Fig. 3. Service rate region of $[7,3]_2$ simplex code.

Based on (5), a fractional matching of the graph depicted in Fig. 2, $\boldsymbol{x} = (\lambda_{1,1},\ldots,\lambda_{1,4},\lambda_{2,1},\ldots,\lambda_{2,4},\lambda_{3,1},\ldots,\lambda_{3,4})$, satisfies the constraints (10) and (11). Thus, from Definition 2, a vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ obtained from $\boldsymbol{x}$ using (9) is in the service rate region of $[7,3]_2$ simplex code. Conversely, for a vector $\boldsymbol{\lambda}$ in the service rate region of $[7,3]_2$ simplex code, there exist $\lambda_{i,j}$'s, $i \in [3]$ and $j \in [4]$, satisfying the constraints (10) and (11), that define a fractional matching vector $\boldsymbol{x} = (\lambda_{i,j} : i \in [3] \text{ and } j \in [4])$ in the graph of Fig. 2.

Based on (6), a maximum fractional matching vector $\boldsymbol{x}^\star$ is obtained by solving the following LP:

$$\max \quad \sum_{i=1}^{3}\sum_{j=1}^{4} \lambda_{i,j} \qquad \text{s.t.} \qquad (10), (11) \text{ hold.} \tag{12}$$

We want to indicate that the vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ obtained from $\boldsymbol{x}^\star$ using (9) is in fact a maximum demand vector $\boldsymbol{\lambda}^\star$ in the service rate region of $[7,3]_2$ simplex code. From (9), $\sum_{i=1}^{3}\sum_{j=1}^{4} \lambda_{i,j} = \lambda_1 + \lambda_2 + \lambda_3$. Thus, it can be easily verified that $\boldsymbol{x}^\star$ provides a solution for the following LP:

$$\max \quad \lambda_1 + \lambda_2 + \lambda_3 \qquad \text{s.t.} \qquad (9), (10), (11) \text{ hold.} \tag{13}$$

Moreover, according to (2), an instance of maximum demand vector is obtained by solving the LP in (13). Thus, the vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ obtained from $\boldsymbol{x}^\star$ using (9) is a maximum demand vector $\boldsymbol{\lambda}^\star$. On the other hand, for an instance of $\boldsymbol{\lambda}^\star$ in the service rate region of $[7,3]_2$ simplex code obtained from (13), there exists a fractional matching vector $\boldsymbol{x}$ which

according to the same reasoning, provides a solution for (12). Thus, the vector $\boldsymbol{x}$ is a maximum fractional matching vector $\boldsymbol{x}^\star$ in the graph representation of $[7,3]_2$ simplex code in Fig. 2. Since a maximum demand vector $\boldsymbol{\lambda}^\star = (\lambda_1^\star, \lambda_2^\star, \lambda_3^\star)$ is obtained from a maximum fractional matching vector $\boldsymbol{x}^\star$ by (9), it follows that $\lambda_1^\star + \lambda_2^\star + \lambda_3^\star = \sum \lambda_{i,j}^\star$, where $\lambda_{i,j}^\star$'s are the elements of $\boldsymbol{x}^\star$. Hence, we have $\lambda^\star(\mathbf{G}) = m_f(G)$, and based on Proposition 2, $m(G) \leq \lambda^\star(\mathbf{G}) \leq v(G)$ holds.

We show that the service capacity of $[7,3]_2$ simplex code is 4. The proof consists of two parts. First, we need to prove the converse by showing that the service capacity cannot be bigger than 4. It is easy to see that the set of vertices $\{f_1, f_2, f_3, f_1+f_2+f_3\}$ is a minimum vertex cover for the graph in Fig. 2. Thus, the vertex cover number of this graph is $v(G) = 4$ which indicates that $\lambda^\star(\mathbf{G}) \leq 4$. Next, we show the achievability proof by showing that there exists a demand vector $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3)$ in the service rate region such that $\lambda_1 + \lambda_2 + \lambda_3 = 4$. For this purpose, one can consider the set of edges labeled by $\lambda_{1,1}$, $\lambda_{1,2}$, $\lambda_{1,3}$, and $\lambda_{1,4}$ as a matching in the graph. Corresponding to this matching, a demand vector $\boldsymbol{\lambda} = (4,0,0)$ is obtained by applying (9).

D. Equivalence Results

Theorem 1. *Consider an $(\mathbf{G}, \boldsymbol{\mu})$ system with the service rate $\boldsymbol{\mu} = \mathbf{1}_n$. There exists a demand vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ in the service rate region of this system if and only if there exists a fractional matching vector $\boldsymbol{x} = (\lambda_{i,j} : i \in [k] \text{ and } j \in [t_i])$ in the graph representation of $[n,k]_q$ code such that $\boldsymbol{\lambda}$ and $\boldsymbol{x}$ are related based on* (1a)*.*

Corollary 1. *Consider an $(\mathbf{G}, \boldsymbol{\mu})$ system with $\boldsymbol{\mu} = \mathbf{1}_n$. There exists a maximum demand vector $\boldsymbol{\lambda}^\star = (\lambda_1^\star, \ldots, \lambda_k^\star)$ in the service rate region $S(\mathbf{G})$ of this storage system if and only if there exists a maximum fractional matching vector $\boldsymbol{x}^\star = (\lambda_{i,j}^\star : i \in [k] \text{ and } j \in [t_i])$ in the graph representation of $[n,k]_q$ code such that $\boldsymbol{\lambda}^\star$ and $\boldsymbol{x}^\star$ are related based on* (1a)*.*

Theorem 2. *Consider an $(\mathbf{G}, \boldsymbol{\mu})$ system with the service rate $\boldsymbol{\mu} = \mathbf{1}_n$. The service capacity $\lambda^\star(\mathbf{G})$ of this system is lower bounded by the matching number and upper bounded by the vertex cover number of the graph representation of $[n,k]_q$ code. i.e., $m(G) \leq \lambda^\star(\mathbf{G}) \leq v(G)$.*

Note that if the graph representation of a code is bipartite, based on Proposition 2, we have $m(G) = \lambda^\star(\mathbf{G}) = v(G)$.

Theorem 3. *The graph representation of $[2^k - 1, k, 2^{k-1}]_2$ simplex code is a bipartite graph.*

Corollary 2. *For an $(\mathbf{G}, \boldsymbol{\mu})$ system with $[2^k - 1, k, 2^{k-1}]_2$ simplex code and service rate $\boldsymbol{\mu} = \mathbf{1}_n$, the service capacity is given by $m(G) = \lambda^\star(\mathbf{G}) = v(G) = 2^{k-1}$.*

Corollary 3. *Consider an $(\mathbf{G}, \boldsymbol{\mu})$ system with $\boldsymbol{\mu} = \mathbf{1}_n$. There exists a demand vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k)$ in the integral service rate region $S_I(\mathbf{G})$ of this system if and only if there exists a matching vector $\tilde{\boldsymbol{x}} = (\lambda_{i,j} : i \in [k] \text{ and } j \in [t_i])$ in the graph representation of $[n,k]_q$ code such that $\boldsymbol{\lambda}$ and $\tilde{\boldsymbol{x}}$ are related based on* (1a)*.*

Corollary 4. *Consider an $(\mathbf{G}, \boldsymbol{\mu})$ system with $\boldsymbol{\mu} = \mathbf{1}_n$. There exists a maximum demand vector $\boldsymbol{\lambda}^\star = (\lambda_1^\star, \ldots, \lambda_k^\star)$ in the integral service rate region $S_I(\mathbf{G})$ of this storage system if and only if there exists a maximum matching vector $\tilde{\boldsymbol{x}}^\star = (\lambda_{i,j}^\star : i \in [k] \text{ and } j \in [t_i])$ in the graph representation of $[n,k]_q$ code such that $\boldsymbol{\lambda}^\star$ and $\tilde{\boldsymbol{x}}^\star$ are related based on* (1a)*.*

IV. Generalization of Batch codes

In this section, we show how the service rate problem can be viewed as a generalization of the batch codes problem. That further illustrates connections with PIR codes, switch codes and locally repairable codes (see [30]).

Definition 10. *[29] An (n,k,t,m,τ) batch code $\mathcal{C}$ over a finite alphabet $\sum$ encodes any string $\boldsymbol{x} = (x_1, \ldots, x_k)$ into m strings (buckets) $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ of total length n by an encoding mapping $\mathcal{C} : \sum^k \to \sum^n$, such that for each t-tuple (batch) of indices $i_1, \ldots, i_t \in [k]$, the entries $x_{i_1}, \ldots, x_{i_t}$ can be decoded by reading at most τ symbols from each bucket.*

Definition 11. *[30] An (n,k,t) primitive batch code is an (n,k,t,m,τ) batch code, where each bucket contains exactly one symbol, i.e., $n = m$. Note that in this setting $\tau = 1$, i.e., at most one symbol can be recovered from each bucket.*

Definition 12. *An (n,k,t) multiset primitive batch code is an (n,k,t) primitive batch code where the information symbols $x_{i_1}, \ldots, x_{i_t}$ are requested by t distinct users such that the indices $i_1, \ldots, i_t$ are not necessarily distinct and in general they form a multiset. Moreover, the requested symbols can be reconstructed from the data read by t different users independently (i.e., x_{i_j} can be recovered by the user j) so that the sets of the symbols read by these users are disjoint.*

For simplicity, we refer to a linear (n,k,t) multiset primitive batch code over $\mathbb{F}_q$ as $[n,k,t]_q$ batch code.

Proposition 3. *[31, Theorem 1] A linear $[n,k]_q$ code $\mathcal{C}$ with generator matrix $\mathbf{G}$ is an $[n,k,t]_q$ batch code if and only if there exist t non-intersecting sets $T_1, \ldots, T_t$ of indices of columns in the generator matrix $\mathbf{G}$ such that for each $j \in [t]$, there exists a linear combination of columns of $\mathbf{G}$ indexed by T_j which equals to the vector $\mathbf{e}_{i_j}$, for all $j \in [t]$ and $i_j \in [k]$.*

Theorem 4. *Given the integral service rate region $\mathcal{S}_I(G)$ of code $\mathbf{G} \in \mathbb{F}_q^{k \times n}$ with service rate $\boldsymbol{\mu} = \mathbf{1}_n$, if all vectors in the set $S_t = \{\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_k) | \sum_{i=1}^k \lambda_i = t, \lambda_i \in \mathbb{Z}_{\geq 0}\}$ are in the $\mathcal{S}_I(G)$, the code $\mathbf{G}$ is a linear $[n,k,t]_q$ batch code.*

Theorem 4 shows that the integral setting of the service rate problem where the solution (the portion of requests that are assigned to the recovery sets) is restricted to be integral, is the same as the setting of the multiset primitive batch code problem. Thus, the general setting of the service rate problem where a fractional solution is allowed, can be viewed as a generalization of the multiset primitive batch code problem.

An example regarding the application of Theorem 4 that shows a binary $[7,3]_2$ simplex code is a $[7,3,4]_2$ batch code, is provided in [35]. Note that a binary $[2^k - 1, k]_2$ simplex code is a $[2^k - 1, k, 2^{k-1}]_2$ batch code (see [33]).

REFERENCES

[1] M. Aktaş, G. Joshi, S. Kadhe, and E. Soljanin, "Service rate region: A new aspect of coded distributed system design." [Online]. Available: https://emina.flywheelsites.com

[2] M. Noori, E. Soljanin, and M. Ardakani, "On storage allocation for maximum service rate in distributed storage systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 240–244.

[3] M. Aktaş, S. E. Anderson, A. Johnston, G. Joshi, S. Kadhe, G. L. Matthews, C. Mayer, and E. Soljanin, "On the service capacity region of accessing erasure coded content," in *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2017, pp. 17–24.

[4] S. E. Anderson, A. Johnston, G. Joshi, G. L. Matthews, C. Mayer, and E. Soljanin, "Service rate region of content access from erasure coded storage," in *2018 IEEE Information Theory Workshop (ITW)*. IEEE, 2018, pp. 1–5.

[5] F. Kazemi, S. Kurz, and E. Soljanin, "A geometric view of the service rates of codes problem and its application to the service rate of the first order Reed-Muller codes," Jan 2020. [Online]. Available: arXiv:2001.09121

[6] P. Peng and E. Soljanin, "On distributed storage allocations of large files for maximum service rate," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2018, pp. 784–791.

[7] M. F. Aktas, A. Behrouzi-Far, E. Soljanin, and P. Whiting, "Load balancing performance in distributed storage with regular balanced redundancy," Dec 2019. [Online]. Available: arXiv:1910.05791

[8] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, 2010.

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

[10] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," *ACM Transactions on Storage (TOS)*, vol. 9, no. 1, p. 3, 2013.

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

[12] M. Sardari, R. Restrepo, F. Fekri, and E. Soljanin, "Memory allocation in distributed storage networks," in *2010 IEEE International Symposium on Information Theory*. IEEE, June 2010, pp. 1958–1962.

[13] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[14] M. A. Maddah-Ali and U. Niesen, "Coding for caching: fundamental limits and practical challenges," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 23–29, 2016.

[15] K. Hamidouche, W. Saad, and M. Debbah, "Many-to-many matching games for proactive social-caching in wireless small cell networks," in *2014 12th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. IEEE, 2014, pp. 569–574.

[16] T. X. Tran, F. Kazemi, E. Karimi, and D. Pompili, "Mobee: Mobility-aware energy-efficient coded caching in cloud radio access networks," in *2017 IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*. IEEE, 2017, pp. 461–465.

[17] G. Joshi, Y. Liu, and E. Soljanin, "Coding for fast content download," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2012, pp. 326–333.

[18] ——, "On the delay-storage trade-off in content download from coded distributed storage systems," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 989–997, 2014.

[19] N. B. Shah, K. Lee, and K. Ramchandran, "The mds queue: Analysing the latency performance of erasure codes," in *2014 IEEE International Symposium on Information Theory*. IEEE, 2014, pp. 861–865.

[20] G. Liang and U. C. Kozat, "Fast cloud: Pushing the envelope on delay performance of cloud storage with coding," *IEEE/ACM Transactions on Networking (TON)*, vol. 22, no. 6, pp. 2012–2025, 2014.

[21] S. Kadhe, E. Soljanin, and A. Sprintson, "Analyzing the download time of availability codes," in *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 1467–1471.

[22] ——, "When do the availability codes make the stored data more available?" in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 2015, pp. 956–963.

[23] K. Gardner, S. Zbarsky, S. Doroudi, M. Harchol-Balter, and E. Hyytia, "Reducing latency via redundant requests: Exact analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 347–360, 2015.

[24] G. Joshi, E. Soljanin, and G. W. Wornell, "Efficient replication of queued tasks for latency reduction in cloud systems," in *53rd Annual Allerton Conference on Communication, Control, and Computing*, 2015, pp. 107–114.

[25] ——, "Efficient redundancy techniques for latency reduction in cloud systems," *TOMPECS*, vol. 2, no. 2, pp. 12:1–12:30, 2017.

[26] M. F. Aktaş, E. Najm, and E. Soljanin, "Simplex queues for hot-data download," in *Proceedings of the SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*. ACM, 2017, pp. 35–36.

[27] M. F. Aktaş and E. Soljanin, "Heuristics for analyzing download time in MDS coded storage systems," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018.

[28] M. F. Aktaş, S. Kadhe, E. Soljanin, and A. Sprintson, "Analyzing the download time of availability codes," Dec 2019. [Online]. Available: arXiv:1912.09765

[29] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch codes and their applications," in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*. ACM, 2004, pp. 262–271.

[30] V. Skachek, "Batch and pir codes and their connections to locally repairable codes," in *Network Coding and Subspace Designs*. Springer, 2018, pp. 427–442.

[31] H. Lipmaa and V. Skachek, "Linear batch codes," in *Coding Theory and Applications*. Springer, 2015, pp. 245–253.

[32] A. Fazeli, A. Vardy, and E. Yaakobi, "Pir with low storage overhead: coding instead of replication," May 2015. [Online]. Available: arXiv:1505.06241

[33] Z. Wang, H. M. Kiah, and Y. Cassuto, "Optimal binary switch codes with small query size," in *2015 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 636–640.

[34] Z. Wang, H. M. Kiah, Y. Cassuto, and J. Bruck, "Switch codes: Codes for fully parallel reconstruction," *IEEE Transactions on Information Theory*, vol. 63, no. 4, pp. 2061–2075, 2017.

[35] F. Kazemi, E. Karimi, E. Soljanin, and A. Sprintson, "A combinatorial view of the service rates of codes problem, its equivalence to fractional matching and its connection with batch codes," Jan 2020. [Online]. Available: arXiv:2001.09146

[36] E. R. Scheinerman and D. H. Ullman, *Fractional graph theory: a rational approach to the theory of graphs*. Courier Corporation, 2011.