4-2018

Coded Caching Schemes with Reduced Subpacketization from Linear Block Codes

Li Tang
Iowa State University, litang@iastate.edu

Aditya Ramamoorthy
Iowa State University, adityar@iastate.edu

Coded Caching Schemes with Reduced Subpacketization from Linear Block Codes

Abstract
Coded caching is a technique that generalizes conventional caching and promises significant reductions in traffic over caching networks. However, the basic coded caching scheme requires that each file hosted in the server be partitioned into a large number (i.e., the subpacketization level) of non-overlapping subfiles. From a practical perspective, this is problematic as it means that prior schemes are only applicable when the size of the files is extremely large. In this work, we propose coded caching schemes based on combinatorial structures called resolvable designs. These structures can be obtained in a natural manner from linear block codes whose generator matrices possess certain rank properties.We obtain several schemes with subpacketization levels substantially lower than the basic scheme at the cost of an increased rate. Depending on the system parameters, our approach allows us to operate at various points on the subpacketization level vs. rate tradeoff.

Keywords
coded caching, resolvable designs, cyclic codes, subpacketization level

Disciplines
Electrical and Computer Engineering | Systems and Communications

Comments
This article is published as Tang, Li, and Aditya Ramamoorthy. "Coded Caching Schemes with Reduced Subpacketization from Linear Block Codes." *IEEE Transactions on Information Theory* 64, no. 4 (2018): 3099-3120. DOI: 10.1109/TIT.2018.2800059. Posted with permission.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

Coded Caching Schemes with Reduced Subpacketization from Linear Block Codes

Li Tang and Aditya Ramamoorthy
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50010
Emails:{litang, adityar}@iastate.edu

***Abstract*—Coded caching is a technique that generalizes conventional caching and promises significant reductions in traffic over caching networks. However, the basic coded caching scheme requires that each file hosted in the server be partitioned into a large number (i.e., the subpacketization level) of non-overlapping subfiles. From a practical perspective, this is problematic as it means that prior schemes are only applicable when the size of the files is extremely large. In this work, we propose coded caching schemes based on combinatorial structures called resolvable designs. These structures can be obtained in a natural manner from linear block codes whose generator matrices possess certain rank properties. We obtain several schemes with subpacketization levels substantially lower than the basic scheme at the cost of an increased rate. Depending on the system parameters, our approach allows us to operate at various points on the subpacketization level vs. rate tradeoff.**

***Index Terms*—coded caching, resolvable designs, cyclic codes, subpacketization level**

I. Introduction

Caching is a popular technique for facilitating large scale content delivery over the Internet. Traditionally, caching operates by storing popular content closer to the end users. Typically, the cache serves an end user's file request partially (or sometimes entirely) with the remainder of the content coming from the main server. Prior work in this area [1] demonstrates that allowing coding in the cache and coded transmission from the server (referred to as *coded caching*) to the end users can allow for significant reductions in the number of bits transmitted from the server to the end users. This is an exciting development given the central role of caching in supporting a significant fraction of Internet traffic. In particular, reference [1] considers a scenario where a single server contains N files. The server connects to K users over a shared link and each user has a cache that allows it to store M/N fraction of all the files in the server. Coded caching consists of two distinct phases: a *placement phase* and a *delivery phase*. In the placement phase, the caches of the users are populated. This phase does not depend on the user demands which are assumed to be arbitrary. In the delivery phase, the server sends a set of *coded* signals that are broadcast to each user such that each user's demand is satisfied.

The original work of [1] considered the case of centralized coded caching, where the server decides the content that needs to be placed in the caches of the different users. Subsequent work considered the decentralized case where the users populate their caches by randomly choosing parts of each file while respecting the cache size constraint. Recently, there have been several papers that have examined various facets of coded caching. These include tightening known bounds on the coded caching rate [2], [3], considering issues with respect to decentralized caching [4], explicitly considering popularities of files [5], [6], network topology issues [7], [8] and synchronization issues [9], [10].

In this work, we examine another important aspect of the coded caching problem that is closely tied to its adoption in practice. It is important to note that the huge gains of coded caching require each file to be partitioned into $F_s \approx \binom{K}{\frac{KM}{N}}$ non-overlapping subfiles of equal size; F_s is referred to as the *subpacketization level*. It can be observed that for a fixed cache size $\frac{M}{N}$, F_s grows exponentially with K. This can be problematic in practical implementations. For instance, suppose that $K = 64$, with $\frac{M}{N} = 0.25$ so that $F_s = \binom{64}{16} \approx 4.8 \times 10^{14}$ with a rate $R \approx 2.82$. In this case, it is evident that at the bare minimum, the size of each file has to be at least 480 terabits for leveraging the gains in [1]. It is even worse in practice. The atomic unit of storage on present day hard drives is a sector of size 512 bytes and the trend in the disk drive industry is to move this to 4096 bytes [11]. As a result, the minimum size of each file needs to be much higher than 480 terabits. Therefore, the scheme in [1] is not practical even for moderate values of K. Furthermore, even for smaller values of K, schemes with low subpacketization levels are desirable. This is because any practical scheme will require each of the subfiles to have some header information that allows for decoding at the end users. When there are a large number of subfiles, the header overhead may be non-negligible. For these same parameters ($K = 64, M/N = 0.25$) our proposed approach in this work allows us obtain, e.g., the following operating points: (i) $F_s \approx 1.07 \times 10^9$ and $R = 3$, (ii) $F_s \approx 1.6 \times 10^4$ and $R = 6$, (iii) $F_s = 64$ and $R = 12$. For the first point, it is evident that the subpacketization level drops by over five orders of magnitude with only a very small increase in the rate. Points (ii) and (iii) show that the proposed scheme

This work was supported in part by the National Science Foundation by grants CCF-1718470, CCF-1320416 and CCF-1149860. This paper was presented in part at the 2016 IEEE Workshop on Network Coding and Applications (NetCod) and at the 2017 IEEE International Symposium on Information Theory (ISIT).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

allows us to operate at various points on the tradeoff between subpacketization level and rate.

The issue of subpacketization was first considered in the work of [12], [13] in the decentralized coded caching setting. In the centralized case it was considered in the work of [14]. They proposed a low subpacketization scheme based on placement delivery arrays. Reference [15] viewed the problem from a hypergraph perspective and presented several classes of coded caching schemes. The work of [16] has recently shown that there exist coded caching schemes where the subpacketization level grows linearly with the number of users K; however, this result only applies when the number of users is very large. We elaborate on related work in Section II-A.

In this work, we propose low subpacketization level schemes for coded caching. Our proposed schemes leverage the properties of combinatorial structures known as resolvable designs and their natural relationship with linear block codes. Our schemes are applicable for a wide variety of parameter ranges and allow the system designer to tune the subpacketization level and the gain of the system with respect to an uncoded system. We note here that designs have also been used to obtain results in distributed data storage [17] and network coding based function computation in recent work [18], [19].

This paper is organized as follows. Section II discusses the background and related work and summarizes the main contributions of our work. Section III outlines our proposed scheme. It includes all the constructions and the essential proofs. A central object of study in our work are matrices that satisfy a property that we call the consecutive column property (CCP). Section IV overviews several constructions of matrices that satisfy this property. Several of the longer and more involved proofs of statements in Sections III and IV appear in the Appendix. In Section V we perform an in-depth comparison of our work with existing constructions in the literature. We conclude the paper with a discussion of opportunities for future work in Section VI.

II. Background, Related Work and Summary of Contributions

We consider a scenario where the server has N files each of which consist of F_s subfiles. There are K users each equipped with a cache of size MF_s subfiles. The coded caching scheme is specified by means of the placement scheme and an appropriate delivery scheme for each possible demand pattern. In this work, we use combinatorial designs [20] to specify the placement scheme in the coded caching system.

Definition 1. A design is a pair $(X, \mathcal{A})$ such that

1) X is a set of elements called points, and
2) $\mathcal{A}$ is a collection of nonempty subsets of X called blocks, where each block contains the same number of points.

A design is in one-to-one correspondence with an incidence matrix $\mathcal{N}$ which is defined as follows.

Definition 2. The incidence matrix $\mathcal{N}$ of a design $(X, \mathcal{A})$ is a binary matrix of dimension $|X| \times |\mathcal{A}|$, where the rows and columns correspond to the points and blocks respectively. Let $i \in X$ and $j \in \mathcal{A}$. Then,

$$\mathcal{N}(i,j) = \begin{cases} 1 & \text{if } i \in j, \\ 0 & \text{otherwise.} \end{cases}$$

It can be observed that the transpose of an incidence matrix also specifies a design. We will refer to this as the transposed design. In this work, we will utilize resolvable designs which are a special class of designs.

Definition 3. A parallel class $\mathcal{P}$ in a design $(X, \mathcal{A})$ is a subset of disjoint blocks from $\mathcal{A}$ whose union is X. A partition of $\mathcal{A}$ into several parallel classes is called a resolution, and $(X, \mathcal{A})$ is said to be a resolvable design if $\mathcal{A}$ has at least one resolution.

For resolvable designs, it follows that each point also appears in the same number of blocks.

Example 1. Consider a block design specified as follows.

$$X = \{1,2,3,4\}, \text{ and}$$
$$\mathcal{A} = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}.$$

Its incidence matrix is given below.

$$\mathcal{N} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

It can be observed that this design is resolvable with the following parallel classes.

$$\mathcal{P}_1 = \{\{1,2\},\{3,4\}\},$$
$$\mathcal{P}_2 = \{\{1,3\},\{2,4\}\}, \text{ and}$$
$$\mathcal{P}_3 = \{\{1,4\},\{2,3\}\}.$$

In the sequel we let $[n]$ denote the set $\{1,\ldots,n\}$. We emphasize here that the original scheme of [1] can be viewed as an instance of the trivial design. For example, consider the setting when $t = KM/N$ is an integer. Let $X = [K]$ and $\mathcal{A} = \{B : B \subset [K], |B| = t\}$. In the scheme of [1], the users are associated with X and the subfiles with $\mathcal{A}$. User $i \in [K]$ caches subfile $W_{n,B}, n \in [N]$ for $B \in \mathcal{A}$ if $i \in B$. The main message of our work is that carefully constructed resolvable designs can be used to obtain coded caching schemes with low subpacketization levels, while retaining much of the rate gains of coded caching. The basic idea is to associate the users with the blocks and the subfiles with the points of the design. The roles of the users and subfiles can also be interchanged by simply working with the transposed design.

Example 2. Consider the resolvable design from Example 1. The blocks in $\mathcal{A}$ correspond to six users U_{12}, U_{34}, U_{13}, U_{24}, U_{14}, U_{23}. Each file is partitioned into $F_s = 4$ subfiles $W_{n,1}, W_{n,2}, W_{n,3}, W_{n,4}$ which correspond to the four points in X. The cache in user U_B, denoted Z_B is specified as $Z_{ij} = (W_{n,i}, W_{n,j})_{n=1}^{N}$. For example, $Z_{12} = (W_{n,1}, W_{n,2})_{n=1}^{N}$.

We note here that the caching scheme is symmetric with respect to the files in the server. Furthermore, each user caches half of each file so that $M/N = 1/2$. Suppose that in the delivery phase user U_B requests file W_{d_B} where $d_B \in [N]$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

These demands can be satisfied as follows. We pick three blocks, one each from parallel classes $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ and generate the signals transmitted in the delivery phase as follows.

$$\begin{aligned} &W_{d_{12},3} \oplus W_{d_{13},2} \oplus W_{d_{23},1}, \\ &W_{d_{12},4} \oplus W_{d_{24},1} \oplus W_{d_{14},2}, \\ &W_{d_{34},1} \oplus W_{d_{13},4} \oplus W_{d_{14},3}, \text{ and} \\ &W_{d_{34},2} \oplus W_{d_{24},3} \oplus W_{d_{23},4}. \end{aligned} \tag{1}$$

The three terms in the in eq. (1) above correspond to blocks from different parallel classes $\{1,2\} \in \mathcal{P}_1, \{1,3\} \in \mathcal{P}_2, \{2,3\} \in \mathcal{P}_3$. This equation has the *all-but-one* structure that was also exploited in [1], i.e., eq. (1) is such that each user caches all but one of the subfiles participating in the equation. Specifically, user U_{12} contains $W_{n,1}$ and $W_{n,2}$ for all $n \in [N]$. Thus, it can decode subfile $W_{d_{12},3}$ that it needs. A similar argument applies to users U_{13} and U_{23}. It can be verified that the other three equations also have this property. Thus, at the end of the delivery phase, each user obtains its missing subfiles.

This scheme corresponds to a subpacketization level of 4 and a rate of 1. In contrast, the scheme of [1] would require a subpacketization level of $\binom{6}{3} = 20$ with a rate of 0.75. Thus, it is evident that we gain significantly in terms of the subpacketization while sacrificing some rate gains.

As shown in Example 2, we can obtain a scheme by associating the users with the blocks and the subfiles with the points. In this work, we demonstrate that this basic idea can be significantly generalized and several schemes with low subpacketization levels that continue to leverage much of the rate benefits of coded caching can be obtained.

A. Discussion of Related Work

Coded caching has been the subject of much investigation in recent work as discussed briefly earlier on. We now overview existing literature on the topic of low subpacketization schemes for coded caching. In the original paper [1], for given problem parameters K (number of users) and M/N (cache fraction), the authors showed that when $N \geq K$, the rate equals

$$R = \frac{K(1-M/N)}{1+KM/N}$$

when M is an integer multiple of N/K. Other points are obtained via memory sharing. Thus, in the regime when KM/N is large, the coded caching rate is approximately $N/M - 1$, which is independent of K. Crucially, though this requires the subpacketization level $F_s \approx \binom{K}{KM/N}$. It can be observed that for a fixed M/N, F_s grows exponentially with K. This is one of main drawbacks of the original scheme and for reasons outlined in Section I, deploying this solution in practice may be difficult.

The subpacketization issue was first discussed in the work of [12], [13] in the context of decentralized caching. Specifically, [13] showed that in the decentralized setting for any subpacketization level F_s such that $F_s \leq \exp(KM/N)$ the rate would scale linearly in K, i.e., $R \geq cK$. Thus, much of the rate benefits of coded caching would be lost if F_s did not scale exponentially in K. Following this work, the authors in [14] introduced a technique for designing low subpacketization schemes in the centralized setting which they called placement delivery arrays. In [14], they considered the setting when $M/N = 1/q$ or $M/N = 1 - 1/q$ and demonstrated a scheme where the subpacketization level was exponentially smaller than the original scheme, while the rate was marginally higher. This scheme can be viewed as a special case of our work. We discuss these aspects in more detail in Section V. In [15], the design of coded caching schemes was achieved through the design of hypergraphs with appropriate properties. In particular, for specific problem parameters, they were able to establish the existence of schemes where the subpacketization scaled as $\exp(c\sqrt{K})$. Reference [21] presented results in this setting by considering strong edge coloring of bipartite graphs.

Very recently, [16] showed the existence of coded caching schemes where the subpacketization grows linearly with the number of users, but the coded caching rate grows as $O(K^{\delta})$ where $0 < \delta < 1$. Thus, while the rate is not a constant, it does not grow linearly with K either. Both [15] and [16] are interesting results that demonstrate the existence of regimes where the subpacketization scales in a manageable manner. Nevertheless, it is to be noted that these results come with several caveats. For example, the result of [16] is only valid in the regime when K is very large and is unlikely to be of use for practical values of K. The result of [15] has significant restrictions on the number of users, e.g., in their paper, K needs to be of the form $\binom{n}{a}$ and $q^t\binom{n}{a}$.

B. Summary of Contributions

In this work, the subpacketization levels we obtain are typically exponentially smaller than the original scheme. However, they still continue to scale exponentially in K, albeit with much smaller exponents. However, our construction has the advantage of being applicable for a large range of problem parameters. Our specific contributions include the following.

- We uncover a simple and natural relationship between a (n,k) linear block code and a coded caching scheme. We first show that any linear block code over $GF(q)$ and in some cases $\mathbb{Z} \mod q$ (where q is not a prime or a prime power) generates a resolvable design. This design in turn specifies a coded caching scheme with $K = nq$ users where the cache fraction $M/N = 1/q$. A complementary cache fraction point where $M/N = 1 - \alpha/nq$ where α is some integer between 1 and $k+1$ can also be obtained. Intermediate points can be obtained by memory sharing between these points.
- We consider a class of (n,k) linear block codes whose generator matrices satisfy a specific rank property. In particular, we require collections of consecutive columns to have certain rank properties. For such codes, we are able to identify an efficient delivery phase and determine the precise coded caching rate. We demonstrate that the subpacketization level is at most $q^k(k+1)$ whereas the coded caching gain scales as $k+1$ with respect to an uncoded caching scheme. Thus, different choices of k

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

allow the system designer significant flexibility to choose the appropriate operating point.
- We discuss several constructions of generator matrices that satisfy the required rank property. We characterize the range of alphabet sizes (q) over which these matrices can be constructed. If one has a given subpacketization budget in a specific setting, we are able to find a set of schemes that fit the budget while leveraging the rate gains of coded caching.

III. Proposed low subpacketization level scheme

All our constructions of low subpacketization schemes will stem from resolvable designs (*cf.* Definition 3). Our overall approach is to first show that any (n, k) linear block code over $GF(q)$ can be used to obtain a resolvable block design. The placement scheme obtained from this resolvable design is such that $M/N = 1/q$. Under certain (mild) conditions on the generator matrix we show that a delivery phase scheme can be designed that allows for a significant rate gain over the uncoded scheme while having a subpacketization level that is significantly lower than [1]. Furthermore, our scheme can be transformed into another scheme that operates at the point $M/N = 1 - \frac{k+1}{nq}$. Thus, intermediate values of M/N can be obtained via memory sharing. We also discuss situations under which we can operate over modular arithmetic $\mathbb{Z}_q = \mathbb{Z} \mod q$ where q is not necessarily a prime or a prime power; this allows us to obtain a larger range of parameters.

A. Resolvable Design Construction

Consider a (n, k) linear block code over $GF(q)$. To avoid trivialities we assume that its generator matrix does not have an all-zeros column. We collect its q^k codewords and construct a matrix $\mathbf{T}$ of size $n \times q^k$ as follows.

$$\mathbf{T} = [\mathbf{c}_0^T, \mathbf{c}_1^T, \cdots, \mathbf{c}_{q^k-1}^T], \tag{2}$$

where the $1 \times n$ vector $\mathbf{c}_\ell$ represents the ℓ-th codeword of the code. Let $X = \{0, 1, \cdots, q^k - 1\}$ be the point set and $\mathcal{A}$ be the collection of all subsets $B_{i,l}$ for $0 \le i \le n-1$ and $0 \le l \le q-1$, where

$$B_{i,l} = \{j : \mathbf{T}_{i,j} = l\}.$$

Using this construction, we can obtain the following result.

Lemma 1. The construction procedure above results in a design $(X, \mathcal{A})$ where $X = \{0, 1, \cdots, q^k - 1\}$ and $|B_{i,l}| = q^{k-1}$ for all $0 \le i \le n-1$ and $0 \le l \le q-1$. Furthermore, the design is resolvable with parallel classes given by $\mathcal{P}_i = \{B_{i,l} : 0 \le l \le q-1\}$, for $0 \le i \le n-1$.

Proof. Let $\mathbf{G} = [g_{ab}]$, for $0 \le a \le k-1$, $0 \le b \le n-1$, $g_{ab} \in GF(q)$. Note that for $\Delta = [\Delta_0\ \Delta_1\ \ldots\ \Delta_{n-1}] = \mathbf{uG}$, we have

$$\Delta_b = \sum_{a=0}^{k-1} \mathbf{u}_a g_{ab},$$

where $\mathbf{u} = [\mathbf{u}_0, \cdots, \mathbf{u}_{k-1}]$. Let a^* be such that $g_{a^*b} \neq 0$. Consider the equation

$$\sum_{a \neq a^*} \mathbf{u}_a g_{ab} = \Delta_b - \mathbf{u}_{a^*} g_{a^*b},$$

where Δ_b is fixed. For arbitrary values of $\mathbf{u}_a$, $a \neq a^*$, this equation has a unique solution for $\mathbf{u}_{a^*}$, which implies that for any Δ_b, $|B_{b,\Delta_b}| = q^{k-1}$ and that $\mathcal{P}_b$ forms a parallel class. ■

Remark 1. A $k \times n$ generator matrix over $GF(q)$ where q is a prime power can also be considered as a matrix over an extension field $GF(q^m)$ where m is an integer. Thus, one can obtain a resolvable design in this case as well; the corresponding parameters can be calculated in an easy manner.

Remark 2. We can also consider linear block codes over $\mathbb{Z} \mod q$ where q is not necessarily a prime or a prime power. In this case the conditions under which a resolvable design can be obtained by forming the matrix $\mathbf{T}$ are a little more involved. We discuss this in Lemma 4 in the Appendix.

Example 3. Consider a $(4, 2)$ linear block code over $GF(3)$ with generator matrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}.$$

Collecting the nine codewords, $\mathbf{T}$ is constructed as follows.

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \\ 0 & 1 & 2 & 1 & 2 & 0 & 2 & 0 & 1 \\ 0 & 2 & 1 & 1 & 0 & 2 & 2 & 1 & 0 \end{bmatrix}.$$

Using $\mathbf{T}$, we generate the resolvable block design $(X, \mathcal{A})$ where the point set is $X = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. For instance, block $B_{0,0}$ is obtained by identifying the column indexes of zeros in the first row of $\mathbf{T}$, i.e., $B_{0,0} = \{0, 1, 2\}$. Following this, we obtain

$$\begin{aligned}\mathcal{A} = &\{\{0,1,2\},\{3,4,5\},\{6,7,8\},\{0,3,6\},\{1,4,7\},\{2,5,8\},\\ &\{0,5,7\},\{1,3,8\},\{2,4,6\},\{0,4,8\},\{2,3,7\},\{1,5,6\}\}.\end{aligned}$$

It can be observed that $\mathcal{A}$ has a resolution (*cf.* Definition 3) with the following parallel classes.

$$\begin{aligned}\mathcal{P}_0 &= \{\{0,1,2\},\{3,4,5\},\{6,7,8\}\},\\ \mathcal{P}_1 &= \{\{0,3,6\},\{1,4,7\},\{2,5,8\}\},\\ \mathcal{P}_2 &= \{\{0,5,7\},\{1,3,8\},\{2,4,6\}\},\ \text{and}\\ \mathcal{P}_3 &= \{\{0,4,8\},\{2,3,7\},\{1,5,6\}\}.\end{aligned}$$

B. A special class of linear block codes

We now introduce a special class of linear block codes whose generator matrices satisfy specific rank properties. It turns out that resolvable designs obtained from these codes are especially suited for usage in coded caching.

Consider the generator matrix $\mathbf{G}$ of a (n, k) linear block code over $GF(q)$. The i-th column of $\mathbf{G}$ is denoted by $\mathbf{g}_i$. Let z be the least positive integer such that $k+1$ divides nz (denoted by $k+1 \mid nz$). We let $(t)_n$ denote t mod n.

In our construction we will need to consider various collections of $k+1$ consecutive columns of $\mathbf{G}$ (wraparounds

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

over the boundaries are allowed). For this purpose, let $\mathcal{T}_a = \{a(k+1), \cdots, a(k+1)+k\}$ (a is a non-negative integer) and $\mathcal{S}_a = \{(t)_n \mid t \in \mathcal{T}_a\}$. Let $\mathbf{G}_{\mathcal{S}_a}$ be the $k \times (k+1)$ submatrix of $\mathbf{G}$ specified by the columns in $\mathcal{S}_a$, i.e., $\mathbf{g}_\ell$ is a column in $\mathbf{G}_{\mathcal{S}_a}$ if $\ell \in \mathcal{S}_a$. Next, we define the $(k, k+1)$-consecutive column property that is central to the rest of the discussion.

Definition 4. $(k, k+1)$-*consecutive column property.* Consider the submatrices of $\mathbf{G}$ specified by $\mathbf{G}_{\mathcal{S}_a}$ for $0 \leq a \leq \frac{zn}{k+1} - 1$. We say that $\mathbf{G}$ satisfies the $(k, k+1)$-consecutive column property if all $k \times k$ submatrices of each $\mathbf{G}_{\mathcal{S}_a}$ are full rank.

Henceforth, we abbreviate the $(k, k+1)$-consecutive column property as $(k, k+1)$-CCP.

Example 4. In Example 3 we have $k = 2, n = 4$ and hence $z = 3$. Thus, $\mathcal{S}_0 = \{0, 1, 2\}, \mathcal{S}_1 = \{3, 0, 1\}, \mathcal{S}_2 = \{2, 3, 0\}$ and $\mathcal{S}_3 = \{1, 2, 3\}$. The corresponding generator matrix $\mathbf{G}$ satisfies the $(k, k+1)$ CCP as any two columns of the each of submatrices $\mathbf{G}_{\mathcal{S}_i}, i = 0, \ldots, 3$ are linearly independent over $GF(3)$.

We note here that one can also define different levels of the consecutive column property. Let $\mathcal{T}_a^\alpha = \{a\alpha, \cdots, a\alpha+\alpha-1\}$, $\mathcal{S}_a^\alpha = \{(t)_n \mid t \in \mathcal{T}_a^\alpha\}$ and z is the least positive integer such that $\alpha \mid nz$.

Definition 5. (k, α)-*consecutive column property* Consider the submatrices of $\mathbf{G}$ specified by $\mathbf{G}_{\mathcal{S}_a^\alpha}$ for $0 \leq a \leq \frac{zn}{\alpha} - 1$. We say that $\mathbf{G}$ satisfies the (k, α)-consecutive column property, where $\alpha \leq k$ if each $\mathbf{G}_{\mathcal{S}_a^\alpha}$ has full rank. In other words, the α columns in each $\mathbf{G}_{\mathcal{S}_a^\alpha}$ are linearly independent.

As pointed out in the sequel, codes that satisfy the (k, α)-CCP, where $\alpha \leq k$ will result in caching systems that have a multiplicative rate gain of α over an uncoded system. Likewise, codes that satisfy the $(k, k+1)$-CCP will have a gain of $k+1$ over an uncoded system. In the remainder of the paper, we will use the term CCP to refer to the $(k, k+1)$-CCP if the value of k is clear from the context.

C. Usage in a coded caching scenario

A resolvable design generated from a linear block code that satisfies the CCP can be used in a coded caching scheme as follows. We associate the users with the blocks. Each subfile is associated with a point and an additional index. The placement scheme follows the natural incidence between the blocks and the points; a formal description is given in Algorithm 1 and illustrated further in Example 5.

Example 5. Consider the resolvable design from Example 3, where we recall that $z = 3$. The blocks in $\mathcal{A}$ correspond to twelve users U_{012}, U_{345}, U_{678}, U_{036}, U_{147}, U_{258}, U_{057}, U_{138}, U_{246}, U_{048}, U_{237}, U_{156}. Each file is partitioned into $F_s = 9 \times z = 27$ subfiles, each of which is denoted by $W_{n,t}^s$, $t = 0, \cdots, 8$, $s = 0, 1, 2$. The cache in user U_{abc}, denoted Z_{abc} is specified as $Z_{abc} = \{W_{n,t}^s \mid t \in \{a, b, c\}, s \in \{0, 1, 2\}$ and $n \in [N]\}$. This corresponds to a coded caching system where each user caches $1/3$-rd of each file so that $M/N = 1/3$.

Algorithm 1: Placement Scheme

Input : Resolvable design $(X, \mathcal{A})$ constructed from a (n, k) linear block code. Let z be the least positive integer such that $k+1 \mid nz$.

1 Divide each file W_n, for $n \in [N]$ into $q^k z$ subfiles. Thus, $W_n = \{W_{n,t}^s : t \in \{0, \ldots, q^k - 1\}$ and $s \in \{0, \ldots, z-1\}\}$;

2 User U_B for $B \in \mathcal{A}$ caches $Z_B = \{W_{n,t}^s : n \in [N], t \in B \ \& \ s \in \{0, \ldots, z-1\}\}$;

Output: Cache content of user U_B denoted Z_B for $B \in \mathcal{A}$.

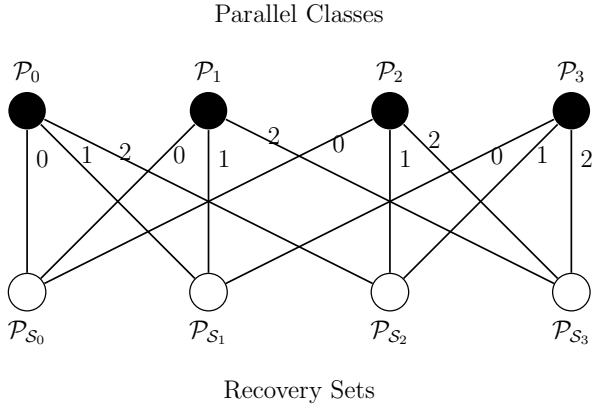

Fig. 1. Recovery set bipartite graph

In general, (see Algorithm 1) we have $K = |\mathcal{A}| = nq$ users. Each file W_n, $n \in [N]$ is divided into $q^k z$ subfiles $W_n = \{W_{n,t}^s \mid 0 \leq t \leq q^k - 1, 0 \leq s \leq z-1\}$. A subfile $W_{n,t}^s$ is cached in user U_B where $B \in \mathcal{A}$ if $t \in B$. Therefore, each user caches a total of $Nq^{k-1}z$ subfiles. As each file consists of $q^k z$ subfiles, we have that $M/N = 1/q$.

It remains to show that we can design a delivery phase scheme that satisfies any possible demand pattern. Suppose that in the delivery phase user U_B requests file W_{d_B} where $d_B \in [N]$. The server responds by transmitting several equations that satisfy each user. Each equation allows $k+1$ users from *different parallel classes* to simultaneously obtain a missing subfile. Our delivery scheme is such that the set of transmitted equations can be classified into various *recovery sets* that correspond to appropriate collections of parallel classes. For example, in Fig. 1, $\mathcal{P}_{\mathcal{S}_0} = \{\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2\}, \mathcal{P}_{\mathcal{S}_1} = \{\mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_3\}$ and so on. It turns out that these recovery sets correspond precisely to the sets $\mathcal{S}_a, 0 \leq a \leq \frac{zn}{k+1} - 1$ defined earlier. We illustrate this by means of the example below.

Example 6. Consider the placement scheme specified in Example 5. Let each user U_B request file W_{d_B}. The recovery sets are specified by means of the recovery set bipartite graph shown in Fig. 1, e.g., $\mathcal{P}_{\mathcal{S}_1}$ corresponds to $\mathcal{S}_1 = \{0, 1, 3\}$. The outgoing edges from each parallel class are labeled arbitrarily with numbers $0, 1$ and 2. Our delivery scheme is such that each user recovers missing subfiles with a specific superscript from each recovery set that its corresponding parallel class participates in. For instance, a user in parallel class $\mathcal{P}_1$ recovers

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

missing subfiles with superscript 0 from $\mathcal{P}_{\mathcal{S}_0}$, superscript 1 from $\mathcal{P}_{\mathcal{S}_1}$ and superscript 2 from $\mathcal{P}_{\mathcal{S}_3}$; these superscripts are the labels of outgoing edges from $\mathcal{P}_1$ in the bipartite graph.

It can be verified, e.g., that user U_{012} which lies in $\mathcal{P}_0$ recovers all missing subfiles with superscript 1 from the equations below.

$$W^1_{d_{012},3} \oplus W^1_{d_{036},2} \oplus W^0_{d_{237},0}, \quad W^1_{d_{012},6} \oplus W^1_{d_{036},1} \oplus W^0_{d_{156},0},$$
$$W^1_{d_{012},4} \oplus W^1_{d_{147},0} \oplus W^0_{d_{048},1}, \quad W^1_{d_{012},7} \oplus W^1_{d_{147},2} \oplus W^0_{d_{237},1},$$
$$W^1_{d_{012},8} \oplus W^1_{d_{258},0} \oplus W^0_{d_{048},2}, \quad W^1_{d_{012},5} \oplus W^1_{d_{258},1} \oplus W^0_{d_{156},2}.$$

Each of the equations above benefits three users. They are generated simply by choosing U_{012} from $\mathcal{P}_0$, *any* block from $\mathcal{P}_1$ and the last block from $\mathcal{P}_3$ so that the *intersection of all these blocks is empty*. The fact that these equations are useful for the problem at hand is a consequence of the CCP. The process of generating these equations can be applied to all possible recovery sets. It can be shown that this allows all users to be satisfied at the end of the procedure.

In what follows, we first show that for the recovery set $\mathcal{P}_{\mathcal{S}_a}$ it is possible to generate equations that benefit $k+1$ users simultaneously.

Claim 1. Consider the resolvable design $(X, \mathcal{A})$ constructed as described in Section III.A by a (n, k) linear block code that satisfies the CCP. Let $\mathcal{P}_{\mathcal{S}_a} = \{\mathcal{P}_i \mid i \in \mathcal{S}_a\}$ for $0 \leq a \leq \frac{zn}{k+1} - 1$, i.e., it is the subset of parallel classes corresponding to $\mathcal{S}_a$. We emphasize that $|\mathcal{P}_{\mathcal{S}_a}| = k+1$. Consider blocks $B_{i_1,l_{i_1}}, \ldots, B_{i_k,l_{i_k}}$ (where $l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from any k distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a}$. Then, $|\cap_{j=1}^{k} B_{i_j,l_{i_j}}| = 1$.

Before proving Claim 1, we discuss its application in the delivery phase. Note that the claim asserts that k blocks chosen from k distinct parallel classes intersect in precisely one point. Now, suppose that one picks $k+1$ users from $k+1$ distinct parallel classes, such that their intersection is empty. These blocks (equivalently, users) can participate in an equation that benefits $k+1$ users. In particular, each user will recover a missing subfile indexed by the intersection of the other k blocks. We emphasize here that Claim 1 is at the core of our delivery phase. Of course, we need to justify that enough equations can be found that allow all users to recover all their missing subfiles. This follows from a natural counting argument that is made more formally in the subsequent discussion. The superscripts $s \in \{0, \ldots, z-1\}$ are needed for the counting argument to go through.

Proof. Following the construction in Section III.A, we note that a block $B_{i,l} \in \mathcal{P}_i$ is specified by

$$B_{i,l} = \{j : \mathbf{T}_{i,j} = l\}.$$

Now consider $B_{i_1,l_{i_1}}, \ldots, B_{i_k,l_{i_k}}$ (where $i_j \in \mathcal{S}_a, l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from k distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a}$. W.l.o.g. we assume that $i_1 < i_2 < \cdots < i_k$. Let $\mathcal{I} = \{i_1, \ldots, i_k\}$ and $\mathbf{T}_{\mathcal{I}}$ denote the submatrix of $\mathbf{T}$ obtained by retaining the rows in $\mathcal{I}$. We will show that the vector $[l_{i_1}\ l_{i_2}\ \ldots\ l_{i_k}]^T$ is a column in $\mathbf{T}_{\mathcal{I}}$ and only appears once.

Algorithm 2: Signal Generation Algorithm for $\mathcal{P}_{\mathcal{S}_a}$

Input : For $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a}$, $E(\mathcal{P}) = \text{label}(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a})$. Signal set $Sig = \emptyset$.

while *any user* $U_B \in \mathcal{P}_j, j \in \mathcal{S}_a$ *does not recover all its missing subfiles with superscript* $E(\mathcal{P}_j)$ **do**

  Pick blocks $B_{j,l_j} \in \mathcal{P}_j$ for all $j \in \mathcal{S}_a$ and $l_j \in \{0, \ldots, q-1\}$ such that $\cap_{j \in \mathcal{S}_a} B_{j,l_j} = \emptyset$;

  /* Pick blocks from distinct parallel classes in $\mathcal{P}_{\mathcal{S}_a}$ such that their intersection is empty */

  Let $\hat{l}_s = \cap_{j \in \mathcal{S}_a \setminus \{s\}} B_{j,l_j}$ for $s \in \mathcal{S}_a$;

  /* Determine the missing subfile index that the user from $\mathcal{P}_s$ will recover */

  Add signal $\oplus_{s \in \mathcal{S}_a} W^{E(\mathcal{P}_s)}_{\kappa_{s,l_s}, \hat{l}_s}$ to Sig /* User $U_{B_{s,l_s}}$ demands file $W_{\kappa_{s,l_s}}$. This equation allows it to recover the corresponding missing subfile index $\hat{l}_s$. The superscript is determined by the recovery set bipartite graph */

end

Output: Signal set Sig.

To see this consider the system of equations in variables $\mathbf{u}_0, \ldots, \mathbf{u}_{k-1}$.

$$\sum_{b=0}^{k-1} \mathbf{u}_b g_{bi_1} = l_{i_1},$$
$$\vdots$$
$$\sum_{b=0}^{k-1} \mathbf{u}_b g_{bi_k} = l_{i_k}.$$

By the CCP, the vectors $\mathbf{g}_{i_1}, \mathbf{g}_{i_2}, \ldots, \mathbf{g}_{i_k}$ are linearly independent. Therefore this system of k equations in k variables has a unique solution over $GF(q)$. The result follows. ■

We now provide an intuitive argument for the delivery phase. Recall that we form a recovery set bipartite graph (see Fig. 1 for an example) with parallel classes and recovery sets as the disjoint vertex subsets. The edges incident on each parallel class are labeled arbitrarily from $0, \ldots, z-1$. For a parallel class $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a}$ we denote this label by $\text{label}(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a})$. For a given recovery set $\mathcal{P}_{\mathcal{S}_a}$, the delivery phase proceeds by choosing blocks from distinct parallel classes in $\mathcal{P}_{\mathcal{S}_a}$ such that their intersection is empty; this provides an equation that benefits $k+1$ users. It turns out that the equation allows a user in parallel class $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a}$ to recover a missing subfile with the superscript $\text{label}(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a})$.

The formal argument is made in Algorithm 2. For ease of notation in Algorithm 2, we denote the demand of user $U_{B_{i,j}}$ for $0 \leq i \leq n-1, 0 \leq j \leq q-1$ by $W_{\kappa_{i,j}}$.

Claim 2. Consider a user U_B belonging to parallel class $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a}$. The signals generated in Algorithm 2 can recover all the missing subfiles needed by U_B with superscript $E(\mathcal{P})$.

Proof. Let $\mathcal{P}_\alpha \in \mathcal{P}_{\mathcal{S}_a}$. In the arguments below, we argue that user $U_{B_{\alpha,l_\alpha}}$ that demands file $W_{\kappa_{\alpha,l_\alpha}}$ can recover all its

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

missing subfiles with superscript $E(\mathcal{P}_\alpha)$. Note that $|B_{\alpha,l_\alpha}| = q^{k-1}$. Thus, user $U_{B_{\alpha,l_\alpha}}$ needs to obtain $q^k - q^{k-1}$ missing subfiles with superscript $E(\mathcal{P}_\alpha)$. Consider an iteration of the while loop where block B_{α,l_α} is picked in step 2. The equation in Algorithm 2 allows it to recover $W^{E(\mathcal{P}_\alpha)}_{\kappa_{\alpha,l_\alpha},\hat{l}_\alpha}$ where $\hat{l}_\alpha = \cap_{j\in\mathcal{S}_a\setminus\{\alpha\}} B_{j,l_j}$. This is because $\cap_{j\in\mathcal{S}_a} B_{j,l_j} = \emptyset$ and because of Claim 1.

Next we count the number of equations that $U_{B_{\alpha,l_\alpha}}$ participates in. We can pick $k-1$ users from some $k-1$ distinct parallel classes in $\mathcal{P}_{\mathcal{S}_a}$. This can be done in q^{k-1} ways. Claim 1 ensures that the blocks so chosen intersect in a single point. Next we pick a block from the only remaining parallel class in $\mathcal{P}_{\mathcal{S}_a}$ such that the intersection of all blocks is empty. This can be done in $q-1$ ways. Thus, there are a total of $q^{k-1}(q-1) = q^k - q^{k-1}$ equations in which user $U_{B_{\alpha,l_\alpha}}$ participates in.

It remains to argue that each equation provides a distinct subfile. Towards this end, let $\{i_1,\ldots,i_k\} \subset \mathcal{S}_a$ be an index set such that $\alpha \notin \{i_1,\ldots,i_k\}$. Suppose that there exist sets of blocks $\{B_{i_1,l_{i_1}},\ldots,B_{i_k,l_{i_k}}\}$ and $\{B_{i_1,l'_{i_1}},\ldots,B_{i_k,l'_{i_k}}\}$ such that $\{B_{i_1,l_{i_1}},\ldots,B_{i_k,l_{i_k}}\} \neq \{B_{i_1,l'_{i_1}},\ldots,B_{i_k,l'_{i_k}}\}$, but $\cap_{j=1}^k B_{i_j,l_{i_j}} = \cap_{j=1}^k B_{i_j,l'_{i_j}} = \beta$. This is a contradiction since this in turn implies that $\beta \in \cap_{j=2}^{k+1} B_{i_j,l_{i_j}} \bigcap \cap_{j=2}^{k+1} B_{i_j,l'_{i_j}}$, which is impossible since two blocks from the same parallel class have an empty intersection.

As the algorithm is symmetric with respect to all blocks in parallel classes belonging to $\mathcal{P}_{\mathcal{S}_a}$, we have the required result. ∎

The overall delivery scheme repeatedly applies Algorithm 2 to each of the recovery sets.

Lemma 2. The proposed delivery scheme terminates and allows each user's demand to be satisfied. Furthermore the transmission rate of the server is $\frac{(q-1)n}{k+1}$ and the subpacketization level is $q^k z$.

Proof. See Appendix. ∎

The main requirement for Lemma 2 to hold is that the recovery set bipartite graph be biregular, where multiple edges between the same pair of nodes is disallowed and the degree of each parallel class is z. It is not too hard to see that this follows from the definition of the recovery sets (see the proof in the Appendix for details).

In an analogous manner, if one starts with the generator matrix of a code that satisfies the (k,α)-CCP for $\alpha \leq k$, then we can obtain the following result which is stated below. The details are quite similar to the discussion for the $(k,k+1)$-CCP and can be found in the Appendix (Section B).

Corollary 1. Consider a coded caching scheme obtained by forming the resolvable design obtained from a (n,k) code that satisfies the (k,α)-CCP where $\alpha \leq k$. Let z be the least positive integer such that $\alpha \mid nz$. Then, a delivery scheme can be constructed such that the transmission rate is $\frac{(q-1)n}{\alpha}$ and the subpacketization level is $q^k z$.

D. Obtaining a scheme for $M/N = 1 - \frac{k+1}{nq}$.

The construction above works for a system where $M/N = 1/q$. It turns out that this can be converted into a scheme for $\frac{M}{N} = 1 - \frac{k+1}{nq}$. Thus, any convex combination of these two points can be obtained by memory-sharing.

Towards this end, we note that the class of coded caching schemes considered here can be specified by an *equation-subfile* matrix. This is inspired by the hypergraph formulation and the placement delivery array (PDA) based schemes for coded caching in [15] and [14]. Each equation is assumed to be of the all-but-one type, i.e., it is of the form $W_{d_{t_1},\mathcal{A}_{j_1}} \oplus W_{d_{t_2},\mathcal{A}_{j_2}} \oplus \cdots \oplus W_{d_{t_m},\mathcal{A}_{j_m}}$ where for each $\ell \in [m]$, we have the property that user U_{t_ℓ} does not cache subfile $W_{n,\mathcal{A}_{j_\ell}}$ but caches all subfiles $W_{n,\mathcal{A}_{j_s}}$ where $\{j_s : s \in [m], s \neq \ell\}$.

The coded caching system corresponds to a $\Delta \times F_s$ equation-subfile matrix $\mathbf{S}$ as follows. We associate each row of $\mathbf{S}$ with an equation and each column with a subfile. We denote the i-th row of $\mathbf{S}$ by Eq_i and j-th column of $\mathbf{S}$ by $\mathcal{A}_j$. The value $\mathbf{S}(i,j) = t$ if in the i-th equation, user U_t recovers subfile $W_{d_t,\mathcal{A}_j}$, otherwise, $\mathbf{S}(i,j) = 0$. Suppose that these Δ equations allow each user to satisfy their demands, i.e., $\mathbf{S}$ corresponds to a valid coded caching scheme. It is not too hard to see that the placement scheme can be obtained by examining $\mathbf{S}$. Namely, user U_t caches the subfile corresponding to the j-th column if integer t does not appear in the j-th column.

Example 7. Consider a coded caching system in [1] with $K = 4$, $\Delta = 4$ and $F_s = 6$. We denote the four users as U_1, U_2, U_3, U_4. Suppose that the equation-subfile matrix $\mathbf{S}$ for this scheme is as specified below.

$$\begin{array}{c} \\ Eq_1 \\ Eq_2 \\ Eq_3 \\ Eq_4 \end{array} \begin{array}{c} \begin{array}{cccccc} \mathcal{A}_1 & \mathcal{A}_2 & \mathcal{A}_3 & \mathcal{A}_4 & \mathcal{A}_5 & \mathcal{A}_6 \end{array} \\ \begin{pmatrix} 3 & 2 & 0 & 1 & 0 & 0 \\ 4 & 0 & 2 & 0 & 1 & 0 \\ 0 & 4 & 3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 4 & 3 & 2 \end{pmatrix} \end{array}.$$

Upon examining $\mathbf{S}$ it is evident for instance that user U_1 caches subfiles $\mathcal{A}_1,\ldots,\mathcal{A}_3$ as the number 1 does not appear in the corresponding columns. Similarly, the cache placement of the other users can be obtained. Interpreting this placement scheme in terms of the user-subfile assignment, it can be verified that the design so obtained corresponds to the transpose of the scheme considered in Example 1 (and also to the scheme of [1] for $K = 4$, $M/N = 1/2$).

Lemma 3. Consider a $\Delta \times F_s$ equation-subfile matrix $\mathbf{S}$ whose entries belong to the set $\{0,1,\ldots,K\}$. It corresponds to a valid coded caching system if the following three conditions are satisfied.

- There is no non-zero integer appearing more than once in each column.
- There is no non-zero integer appearing more than once in each row.
- If $\mathbf{S}(i_1,j_1) = \mathbf{S}(i_2,j_2) \neq 0$, then $\mathbf{S}(i_1,j_2) = \mathbf{S}(i_2,j_1) = 0$.

Proof. The placement scheme is obtained as discussed earlier, i.e., user U_t caches subfiles $W_{n,\mathcal{A}_j}$ if integer t does not appear

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

in column $\mathcal{A}_j$. Therefore, matrix $\mathbf{S}$ corresponds to a placement scheme.

Next we discuss the delivery scheme. Note that Eq_i corresponds to an equation as follows.

$$W_{d_{t_1},\mathcal{A}_{j_1}} \oplus W_{d_{t_2},\mathcal{A}_{j_2}} \oplus \cdots \oplus W_{d_{t_m},\mathcal{A}_{j_m}},$$

where $\mathbf{S}(i,j_1) = t_1, \cdots, \mathbf{S}(i,j_m) = t_m$. The above equation can allow m users to recover subfiles simultaneously if (a) U_{t_ℓ} does not cache $W_{n,\mathcal{A}_{j_\ell}}$ and (b) U_{t_ℓ} caches all $W_{n,\mathcal{A}_{j_s}}$ where $\{j_s : s \in [m], s \neq \ell\}$. It is evident that U_{t_ℓ} does not cache $W_{n,\mathcal{A}_{j_\ell}}$ owing to the placement scheme. Next, to guarantee the condition (b), we need to show that integer $t_\ell = \mathbf{S}(i,j_\ell)$ will not appear in column $\mathcal{A}_{j_s}$ in $\mathbf{S}$ where $\{j_s : s \in [m], s \neq \ell\}$. Towards this end, $t_\ell \neq \mathbf{S}(i,j_s)$ because of Condition 2. Next, consider the non-zero entries that lie in the column $\mathcal{A}_{j_s}$ but not in the row Eq_i. Assume there exists an entry $\mathbf{S}(i',j_s)$ such that $\mathbf{S}(i',j_s) = \mathbf{S}(i,j_\ell) = t_\ell$ and $i' \neq i$, then $\mathbf{S}(i,j_s) = t_s \neq 0$, which is a contradiction to Condition 3. Finally, Condition 1 guarantees that each missing subfile is recovered only once. ■

User U_t caches a fraction $\frac{M_t}{N} = \frac{L_t}{F_s}$ where L_t is the number of columns of $\mathbf{S}$ that do not have the entry t. Similarly, the transmission rate is given by $R = \frac{\Delta}{F_s}$.

The crucial point is that the transpose of $\mathbf{S}$, i.e., $\mathbf{S}^T$ also corresponds to a coded caching scheme. This follows directly from the fact that $\mathbf{S}^T$ also satisfies the conditions in Lemma 3. In particular, $\mathbf{S}^T$ corresponds to a coded caching system with K users and Δ subfiles. In the placement phase, the cache size of U_t is $\frac{M'_t}{N} = \frac{\Delta - F_s + L_t}{\Delta}$. In the delivery phase, by transmitting F_s equations corresponding to the rows of $\mathbf{S}^T$, all missing subfiles can be recovered. Then, the transmission rate is $R' = \frac{F_s}{\Delta}$.

Applying the above discussion in our context, consider the equation-subfile matrix $\mathbf{S}$ corresponding to the coded caching system with $K = nq$, $\frac{M_t}{N} = \frac{1}{q}$ for $1 \leq t \leq nq$, $F_s = q^k z$ and $\Delta = q^k(q-1)\frac{nz}{k+1}$. Then $\mathbf{S}^T$ corresponds to a system with $K' = nq$, $\frac{M'}{N} = 1 - \frac{k+1}{nq}$, $F'_s = (q-1)q^k\frac{zn}{k+1}$, and transmission rate $R' = \frac{F_s}{\Delta} = \frac{k+1}{(q-1)n}$. The following theorem is the main result of this paper.

Theorem 1. Consider a (n,k) linear block code over $GF(q)$ that satisfies the $(k,k+1)$ CCP. This corresponds to a coded caching scheme with $K = nq$ users, N files in the server where each user has a cache of size $M \in \left\{\frac{1}{q}N, \left(1 - \frac{k+1}{nq}\right)N\right\}$. Let z be the least positive integer such that $k+1 \mid nz$. When $\frac{M}{N} = \frac{1}{q}$, we have

$$R = \frac{(q-1)n}{k+1}, \text{ and}$$
$$F_s = q^k z.$$

When $\frac{M}{N} = (1 - \frac{k+1}{nq})$, we have

$$R = \frac{k+1}{(q-1)n}, \text{ and}$$
$$F_s = (q-1)q^k\frac{zn}{k+1}.$$

By memory sharing any convex combination of these points is achievable.

In a similar manner for the (n,k) linear block code that satisfies the (k,α)-CCP over $GF(q)$, the caching system where $M/N = 1/q$ can be converted into a system where $K' = nq$, $\frac{M'}{N'} = 1 - \frac{\alpha}{nq}$, $F'_s = (q-1)q^k\frac{zn}{\alpha}$ and $R' = \frac{\alpha}{(q-1)n}$ using the equation-subfile technique. The arguments presented above apply with essentially no change.

IV. Some classes of linear codes that satisfy the CCP

At this point we have established that linear block codes that satisfy the CCP are attractive candidates for usage in coded caching. In this section, we demonstrate that there are a large class of generator matrices that satisfy the CCP. For most of the section we work with matrices over a finite field of order q. In the last subsection, we discuss some constructions for matrices over $\mathbb{Z} \mod q$ when q is not a prime or prime power. We summarize the constructions presented in this section in Table I.

A. Maximum-distance-separable (MDS) codes

(n,k)-MDS codes with minimum distance $n-k+1$ are clearly a class of codes that satisfy the CCP. In fact, for these codes any k columns of the generator matrix can be shown to be full rank. Note however, that MDS codes typically need large field size, e.g., $q+1 \geq n$ (assuming that the MDS conjecture is true) [22]. In our construction, the value of $M/N = 1/q$ and the number of users is $K = nq$. Thus, for large n, we will only obtain systems with small values of M/N, or equivalently large values of M/N (by Theorem 1 above). This may be restrictive in practice.

B. Cyclic Codes

A cyclic code is a linear block code, where the circular shift of each codeword is also a codeword [23]. A (n,k) cyclic code over $GF(q)$ is specified by a monic polynomial $g(X) = \sum_{i=0}^{n-k} g_i X^i$ with coefficients from $GF(q)$ where $g_{n-k} = 1$ and $g_0 \neq 0$; $g(X)$ needs to divide the polynomial $X^n - 1$. The generator matrix of the cyclic code is obtained as below.

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & 0 & \cdot & 0 & g_0 & g_1 & \cdot & \cdot & \cdot & g_{n-k} \end{bmatrix}.$$

The following claim shows that for verifying the CCP for a cyclic code it suffices to pick *any* set of $k+1$ consecutive columns.

Claim 3. Consider a (n,k) cyclic code with generator matrix $\mathbf{G}$. Let $\mathbf{G}_\mathcal{S}$ denote a set of $k+1$ consecutive columns of $\mathbf{G}$. If each $k \times k$ submatrix of $\mathbf{G}_S$ is full rank, then $\mathbf{G}$ satisfies the $(k,k+1)$-CCP.

Proof. Let the generator polynomial of the cyclic code be $\mathbf{g}(X)$, where we note that $\mathbf{g}(X)$ has degree $n-k$. Let

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

Code type	Code construction	Notes
Codes over field $GF(q)$	(n,k) MDS codes	Satisfy $(k,k+1)$-CCP. Need $q+1 \geq n$.
	(n,k) Cyclic codes	Existence depends on certain properties of the generator polynomials. All cyclic codes satisfy the (k,k)-CCP. Additional conditions are needed for the $(k,k+1)$-CCP.
	Kronecker product of $z \times \alpha$ matrices satisfying the (z,z)-CCP with the identity matrix $\mathbf{I}_{t\times t}$	Satisfy the (k,k)-CCP where $k=tz$.
	Kronecker product of Vandermonde and Vandermonde-like matrices with structured base matrices	Satisfy the $(k,k+1)$-CCP for certain parameters.
	CCP matrix extension	Extends a $k \times n$ CCP matrix to a $k \times (n+s(k+1))$ CCP matrix for integer s.
Codes over ring $\mathbb{Z} \mod q$	Single parity-check (SPC) code	Satisfies the $(k,k+1)$-CCP with $n=k+1$.
	Cyclic codes over the ring	Require that $q=q_1 \times q_2 \times \cdots \times q_d$ where q_i's are prime. Satisfy the (k,k)-CCP.
	Kronecker product of $z \times \alpha$ matrices satisfying the (z,z)-CCP with the identity matrix $\mathbf{I}_{t\times t}$	Satisfy the (k,k)-CCP property where $k=tz$.
	CCP matrix extension	Extends a $k \times n$ CCP matrix to a $k \times (n+s(k+1))$ CCP matrix for integer s.

TABLE I
A SUMMARY OF THE DIFFERENT CONSTRUCTIONS OF CCP MATRICES IN SECTION IV

$\mathbf{G}_{\mathcal{S}} = [\mathbf{g}_{(a)_n}, \mathbf{g}_{(a+1)_n}, \cdots, \mathbf{g}_{(a+k)_n}]$ where we assume that $\mathbf{G}_{\mathcal{S}}$ satisfies the $(k,k+1)$-CCP. Let

$$\begin{aligned}\mathbf{G}_{\mathcal{S}\setminus j} =& [\mathbf{g}_{(a)_n}, \ldots, \mathbf{g}_{(a+j-1)_n}, \mathbf{g}_{(a+j+1)_n}, \ldots, \mathbf{g}_{(a+k)_n}], \text{and}\\ \mathbf{G}_{\mathcal{S}'\setminus j} =& [\mathbf{g}_{(a+i)_n}, \ldots, \mathbf{g}_{(a+j-1+i)_n},\\ & \mathbf{g}_{(a+j+1+i)_n}, \ldots, \mathbf{g}_{(a+k+i)_n}].\end{aligned}$$

We need to show that if $\mathbf{G}_{\mathcal{S}\setminus j}$ has full rank, then $\mathbf{G}_{\mathcal{S}'\setminus j}$ has full rank, for any $0 \leq j \leq k$.

As $\mathbf{G}_{\mathcal{S}\setminus j}$ has full rank, there is no codeword $\mathbf{c} \neq \mathbf{0}$ such that $\mathbf{c}((a)_n) = \cdots = \mathbf{c}((a+j-1)_n) = \mathbf{c}((a+j+1)_n) = \cdots = \mathbf{c}((a+k)_n) = 0$. By the definition of a cyclic code, any circular shift of a codeword results in another codeword that belongs to the code. Therefore, there is no codeword $\mathbf{c}'$ such that $\mathbf{c}'((a+i)_n) = \cdots = \mathbf{c}'(a+j-1+i)_n) = \mathbf{c}'((a+j+1+i)_n) = \cdots = \mathbf{c}'((a+k+i)_n) = 0$. Thus, $\mathbf{G}_{\mathcal{S}'\setminus j}$ has full rank. ■

Claim 3 implies a low complexity search algorithm to determine if a cyclic code satisfies the CCP. Instead of checking all $\mathbf{G}_{\mathcal{S}_a}$, $0 \leq a \leq \frac{zn}{k+1} - 1$, in Definition 4, we only need to check an arbitrary $\mathbf{G}_{\mathcal{S}} = [\mathbf{g}_{(i)_n}, \mathbf{g}_{(i+1)_n}, \cdots, \mathbf{g}_{(i+k)_n}]$, for $0 \leq i < n$. To further simplify the search, we choose $i = n - \lfloor \frac{k}{2} \rfloor - 1$.

For this choice of i, Claim 4 shows that $\mathbf{G}_{\mathcal{S}}$ is such that we only need to check the rank of a list of small-dimension matrices to determine if each $k \times k$ submatrix of $\mathbf{G}_{\mathcal{S}}$ is full rank (the proof appears in the Appendix).

Claim 4. A cyclic code with generator matrix $\mathbf{G}$ satisfies the CCP if the following conditions hold.

- For $0 < j \leq \lfloor \frac{k}{2} \rfloor$, the submatrices
$$\mathbf{C}_j = \begin{bmatrix} g_{n-k-1} & g_{n-k} & 0 & \cdot & \cdot & 0 \\ g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & \vdots \\ g_{n-k-j+1} & \cdot & \cdot & \cdot & \cdot & g_{n-k} \\ g_{n-k-j} & \cdot & \cdot & \cdot & \cdot & g_{n-k-1} \end{bmatrix}$$
have full rank. In the above expression, $g_i = 0$ if $i < 0$.
- For $\lfloor \frac{k}{2} \rfloor < j < k$, the submatrices
$$\mathbf{C}_j = \begin{bmatrix} g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & g_{k-j} \\ g_0 & g_1 & \cdot & \cdot & \cdot & \cdot & g_{k-j-1} \\ \vdots & & & & & & \vdots \\ 0 & \cdot & \cdot & 0 & g_0 & g_1 & g_2 \\ 0 & \cdot & & \cdot\cdot & 0 & g_0 & g_1 \end{bmatrix}$$
have full rank.

Example 8. Consider the polynomial $\mathbf{g}(X) = X^4 + X^3 + X + 2$ over $GF(3)$. Since it divides $X^8 - 1$, it is the generator polynomial of a $(8,4)$ cyclic code over $GF(3)$. The generator matrix of this code is given below.

$$\mathbf{G} = \begin{bmatrix} 2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 2 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

It can be verified that the 4×5 submatrix which consists of the two leftmost columns and three rightmost columns of $\mathbf{G}$ is such that all 4×4 submatrices of it are full rank. Thus, by Claim 3 the (4,5)-CCP is satisfied for $\mathbf{G}$.

Remark 3. Cyclic codes form an important class of codes that satisfy the (k,k)-CCP (*cf.* Definition 5). This is because,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

it is well-known [23] that any k consecutive columns of the generator matrix of a cyclic code are linearly independent.

C. Constructions leveraging properties of smaller base matrices

It is well recognized that cyclic codes do not necessarily exist for any choice of parameters. This is because of the divisibility requirement on the generator polynomial. We now discuss a more general construction of generator matrices that satisfy the CCP. As we shall see, this construction provides a more or less satisfactory solution for a large range of system parameters.

Our first simple observation is that the Kronecker product (denoted by $\otimes$ below) of a $z \times \alpha$ generator matrix that satisfies the (z, z)-CCP with a $t \times t$ identity matrix, $\mathbf{I}_{t\times t}$ immediately yields a generator matrix that satisfies the (tz, tz)-CCP.

Claim 5. Consider a (n, k) linear block code over $GF(q)$ whose generator matrix is specified as $\mathbf{G} = \mathbf{A} \otimes \mathbf{I}_{t\times t}$ where $\mathbf{A}$ is a $z \times \alpha$ matrix that satisfies the (z, z)-CCP. Then, $\mathbf{G}$ satisfies the (k, k)-CCP where $k = tz$ and $n = t\alpha$.

Proof. The recovery set for $\mathbf{A}$ is specified as $\mathcal{S}_a^z = \{(az)_\alpha, \cdots, (az+z-1)_\alpha\}$ and the recovery set for $\mathbf{G}$ is specified as $\mathcal{S}_a^k = \{(ak)_n, \cdots, (ak+k-1)_n\}$. Since $\mathbf{A}$ satisfies the (z, z)-CCP, $\mathbf{A}_{\mathcal{S}_a^z}$ has full rank. Note that $\mathbf{G}_{\mathcal{S}_a^k} = \mathbf{A}_{\mathcal{S}_a^z} \otimes \mathbf{I}_{t\times t}$. Then $\det(\mathbf{G}_{\mathcal{S}_a^k}) = \det(\mathbf{A}_{\mathcal{S}_a^z} \otimes \mathbf{I}_{t\times t}) = \det(\mathbf{A}_{\mathcal{S}_a^z})^t \neq 0$. Therefore, $\mathbf{G}$ satisfies the (k, k)-CCP. ■

Remark 4. Let $\mathbf{A}$ be the generator matrix of a cyclic code over $GF(q)$, then $\mathbf{G} = \mathbf{A} \otimes \mathbf{I}_{t\times t}$ satisfies the (k, k)-CCP by Claim 5.

Our next construction addresses the $(k, k+1)$-CCP. In what follows, we use the following notation.

- $\mathbf{1}_a$: $[\underbrace{1, \cdots, 1}_{a}]^T$;
- $\mathbf{C}(c_1, c_2)_{a\times b}$: $a \times b$ matrix where each row is the cyclic shift (one place to the right) of the row above it and the first row is $[c_1\ c_2\ 0\ \cdots\ 0]$; and
- $\mathbf{0}_{a\times b}$: $a \times b$ matrix with zero entries.

Consider parameters n, k. Let the greatest common divisor of n and $k + 1$, $\gcd(n, k + 1) = t$. It is easy to verify that $z = \frac{k+1}{t}$ is the smallest integer such that $k + 1 \mid nz$. Let $n = t\alpha$ and $k + 1 = tz$. Claim 6 below constructs a (n, k) linear code that satisfies the CCP over $GF(q)$ where $q > \alpha$. Since $\alpha = \frac{n}{t}$, the required field size in Claim 6 is lower than the MDS code considered in Section IV-A.

Claim 6. Consider a (n, k) linear block code over $GF(q)$ whose generator matrix is specified as eq. (3), where

$$\begin{bmatrix} b_{00} & b_{01} & \cdots & b_{0(\alpha-1)} \\ b_{10} & b_{11} & \cdots & b_{1(\alpha-1)} \\ \vdots & & & \vdots \\ b_{(z-1)0} & b_{(z-1)1} & \cdots & b_{(z-1)(\alpha-1)} \end{bmatrix}$$

is a Vandermonde matrix and $q > \alpha$. Then, $\mathbf{G}$ satisfies the $(k, k + 1)$-CCP.

Proof. The proof again leverages the idea that $\mathbf{G}$ can be expressed succinctly by using Kronecker products. The arguments can be found in the Appendix. ■

Consider the case when $\alpha = z + 1$. We construct a (n, k) linear code satisfy the CCP over $GF(q)$ where $q \geq z$. It can be noted that the constraint of field size is looser than the corresponding constraint in Claim 6.

Claim 7. Consider $nz = (z + 1) \cdot (k + 1)$. Consider a (n, k) linear block code whose generator matrix (over $GF(q)$) is specified as follows.

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_1\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times t} & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_2\mathbf{I}_{t\times t} \\ \vdots & & & & & & \vdots \\ \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{(t-1)\times t} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{1}_{t-1} & \mathbf{C}(c_1,c_2)_{(t-1)\times t}, \end{bmatrix} \tag{4}$$

where $t = n - k - 1$. If $q \geq z$, $b_1, b_2, \cdots, b_{z-1}$ are non-zero and distinct, and $c_1 + c_2 = 0$, then $\mathbf{G}$ satisfies the CCP.

Proof. See Appendix. ■

Given a (n, k) code that satisfies the CCP, we can use it obtain higher values of n in a simple manner as discussed in the claim below.

Claim 8. Consider a (n, k) linear block code over $GF(q)$ with generator matrix $\mathbf{G}$ that satisfies the CCP. Let the first $k+1$ columns of $\mathbf{G}$ be denoted by the submatrix $\mathbf{D}$. Then the matrix $\mathbf{G}'$ of dimension $k \times (n + s(k + 1))$ where $s \geq 0$

$$\mathbf{G}' = [\underbrace{\mathbf{D}|\cdots|\mathbf{D}}_{s}|\mathbf{G}]$$

also satisfies the CCP.

Proof. See Appendix. ■

Claim 8 can provide more parameter choices and more possible code constructions. For example, given n, k, q, where $k+1+(n)_{k+1} \leq q+1 < n$, there may not exist a (n, k)-MDS code over $GF(q)$. However, there exists a $(k+1+(n)_{k+1}, k)$-MDS code over $GF(q)$. By Claim 8, we can obtain a (n, k) linear block code over $GF(q)$ that satisfies the CCP. Similarly, combining Claim 4, Claim 6, Claim 7 with Claim 8, we can obtain more linear block codes that satisfy the CCP.

A result very similar to Claim 8 can be obtained for the (k, α)-CCP. Specifically, consider a (n, k) linear block code with generator matrix $\mathbf{G}$ that satisfies the (k, α)-CCP and let $\mathbf{D}$ be the first α columns of $\mathbf{G}$. Then, $\mathbf{G}' = [\underbrace{\mathbf{D}|\cdots|\mathbf{D}}_{s}|\mathbf{G}]$ of dimension $k \times (n + s\alpha)$ also satisfies the (k, α)-CCP.

D. Constructions where q is not a prime or a prime power

We now discuss constructions where q is not a prime or a prime power. We attempt to construct matrices over the ring $\mathbb{Z} \mod q$ in this case. The issue is somewhat complicated by the fact that a square matrix over $\mathbb{Z} \mod q$ has linearly independent rows if and only if its determinant is a unit in the ring [24]. In general, this fact makes it harder to

$$\mathbf{G} = \begin{bmatrix} b_{00}\mathbf{I}_{t\times t} & b_{01}\mathbf{I}_{t\times t} & \cdots & b_{0(\alpha-1)}\mathbf{I}_{t\times t} \\ b_{10}\mathbf{I}_{t\times t} & b_{11}\mathbf{I}_{t\times t} & \cdots & b_{1(\alpha-1)}\mathbf{I}_{t\times t} \\ \vdots & & & \vdots \\ b_{(z-2)0}\mathbf{I}_{t\times t} & b_{(z-2)1}\mathbf{I}_{t\times t} & \cdots & b_{(z-2)(\alpha-1)}\mathbf{I}_{t\times t} \\ \mathbf{C}(b_{(z-1)0}, b_{(z-1)0})_{(t-1)\times t} & \mathbf{C}(b_{(z-1)1}, b_{(z-1)1})_{(t-1)\times t} & \cdots & \mathbf{C}(b_{(z-1)(\alpha-1)}, b_{(z-1)(\alpha-1)})_{(t-1)\times t} \end{bmatrix} \tag{3}$$

obtain constructions such as those in Claim 6 that exploit the Vandermonde structure of the matrices. Specifically, the difference of units in a ring is not guaranteed to be a unit. However, we can still provide some constructions. It can be observed that Claim 5 and Claim 8 hold for linear block codes over $\mathbb{Z} \mod q$. We will use them without proof in this subsection.

Claim 9. Let $\mathbf{G} = [\mathbf{I}_{k\times k}|\mathbf{1}_k]$, i.e., it is the generator matrix of a $(k+1, k)$ single parity check (SPC) code, where the entries are from $\mathbb{Z} \mod q$. The $\mathbf{G}$ satisfies the $(k, k+1)$-CCP and the (k, k)-CCP. It can be used as base matrix for Claim 5.

Proof. It is not too hard to see that when $\mathbf{G} = [\mathbf{I}_{k\times k}|\mathbf{1}_k]$, any $k \times k$ submatrix of $\mathbf{G}$ has a determinant which is ± 1, i.e., it is a unit over $\mathbb{Z} \mod q$. Thus, the result holds in this case. ■

Claim 10. The following matrix with entries from $\mathbb{Z} \mod q$ satisfies the $(k, k+1)$-CCP. Here $k = 2t-1$ and $n = 3t$.

$$\mathbf{G} = \begin{bmatrix} \mathbf{I}_{t\times t} & \mathbf{0} & \mathbf{1}_t & \mathbf{I}_{t\times t} \\ \mathbf{0} & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{1}_t & \mathbf{C}(1,-1)_{(t-1)\times t} \end{bmatrix}.$$

Proof. This can be proved by following the arguments in the proof of Claim 7 while treating elements to be from $\mathbb{Z} \mod q$ and setting $z = 2$. We need to consider three different $k \times (k+1)$ submatrices for which we need to check the property. These correspond to simpler instances of the submatrices considered in Types I - III in the proof of Claim 7. In particular, the corresponding determinants will always be ± 1 which are units over $\mathbb{Z} \mod q$. ■

Remark 5. We note that the general construction in Claim 7 can potentially fail in the case when the matrices are over $\mathbb{Z} \mod q$. This is because in one of the cases under consideration (specifically, Type III, Case 1), the determinant depends on the difference of the b_i values. The difference of units in $\mathbb{Z} \mod q$ is not guaranteed to be a unit, thus there is no guarantee that the determinant is a unit.

Remark 6. We can use Claim 8 to obtain higher values of n based on the above two classes of linear block codes over $\mathbb{Z} \mod q$.

While most constructions of cyclic codes are over $GF(q)$, there has been some work on constructing cyclic codes over $\mathbb{Z} \mod q$. Specifically, [25] provides a construction where $q = q_1 \times q_2 \cdots \times q_d$ and $q_i, i = 1, \ldots, d$ are prime. We begin by outlining this construction. By the Chinese remainder theorem any element $\gamma \in \mathbb{Z} \mod q$ has a unique representation in terms of its residues modulo q_i, for $i = 1, \ldots, d$. Let $\psi : \mathbb{Z} \mod q \to GF(q_1) \times \cdots \times GF(q_d)$ denote this map.

- Suppose that (n, k_i) cyclic codes over $GF(q_i)$ exist for all $i = 1, \ldots, d$. Each individual code is denoted $\mathcal{C}^i$.
- Let $\mathcal{C}$ denote the code over $\mathbb{Z} \mod q$. Let $\mathbf{c}^{(i)} \in \mathcal{C}^i$ for $i = 1, \ldots, d$. The codeword $\mathbf{c} \in \mathcal{C}$ is obtained as follows. The j-th component of $\mathbf{c}$, $\mathbf{c}_j = \psi^{-1}(\mathbf{c}_j^{(1)}, \ldots, \mathbf{c}_j^{(d)})$

Therefore, there are $q_1^{k_1} q_2^{k_2} \cdots q_d^{k_d}$ codewords in $\mathcal{C}$. It is also evident that $\mathcal{C}$ is cyclic. As discussed in Section III-A, we form the matrix $\mathbf{T}$ for the codewords in $\mathcal{C}$. It turns out that using $\mathbf{T}$ and the technique discussed in Section III-A, we can obtain a resolvable design. Furthermore, the gain of the system in the delivery phase can be shown to be $k_{min} = \min\{k_1, k_2, \cdots, k_d\}$. We discuss these points in detail in the Appendix (Section C).

V. Discussion and Comparison with Existing Schemes

A. Discussion

When the number of users is $K = nq$ and the cache fraction is $\frac{M}{N} = \frac{1}{q}$, we have shown in Theorem 1 that the gain $g = k+1$ and $F_s = q^k z$. Therefore, both the gain and the subpacketization level increase with larger k. Thus, for our approach given a subpacketization budget F_s', the highest coded gain that can be obtained is denoted by $g_{max} = k_{max}+1$ where k_{max} is the largest integer such that $q^{k_{max}} z \leq F_s'$ and there exists a (n, k_{max}) linear block code that satisfies the CCP.

For determining k_{max}, we have to characterize the collection of values of k such that there exists a (n, k) linear code satisfies the CCP over $GF(q)$ or $\mathbb{Z} \mod q$. We use our proposed constructions (MDS code, Claim 4, Claim 6, Claim 7, Claim 8, Claim 9, Claim 10) for this purpose. We call this collection $\mathcal{C}(n, q)$ and generate it in Algorithm 3. We note here that it is entirely possible that there are other linear block codes that fit the appropriate parameters and are outside the scope of our constructions. Thus, the list may not be exhaustive. In addition, we note that we only check for the $(k, k+1)$-CCP. Working with the (k, α)-CCP where $\alpha \leq k$ can provide more operating points.

Example 9. Consider a caching system with $K = nq = 12 \times 5 = 60$ users and cache fraction $\frac{M}{N} = \frac{1}{5}$. Suppose that the subpacketization budget is 1.5×10^6. By checking all $k < n$ we can construct $\mathcal{C}(n, q)$ (see Table II). As a result, $\mathcal{C}(n, q) = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 11\}$. Then $k_{max} = 8$, $F_s \approx 1.17 \times 10^6$ and the maximal coded gain we can achieve is $g_{max} = 9$. By contrast, the scheme in [1] can achieve coded gain $g = \frac{KM}{N} + 1 = 13$ but requires subpacketization level $F_s = \binom{K}{\frac{KM}{N}} \approx 1.4 \times 10^{12}$.

We can achieve almost the same rate by performing memory-sharing by using the scheme of [1] in this example. In particular, we divide each file of size Ω into two smaller subfiles W_n^1 and W_n^2, where the size of W_n^1, $|W_n^1| = \frac{9}{10}\Omega$ and the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

k	n'	z	α	Construction	Notes
11	12	1	1	(12, 11) SPC code	$k+1=n'$
10	12	11	12	-	-
9	12	5	6	Claim 7	$\alpha=z+1$ and $q\geq z$
8	12	3	4	Claim 7	$\alpha=z+1$ and $q\geq z$
7	12	2	3	Claim 7	$\alpha=z+1$ and $q\geq z$
6	12	7	12	Claim 4	Generator polynomial is $X^6+X^5+3X^4+3X^3+X^2+4X+3$
5	6	1	1	(6,5) SPC code and Claim 8	Extend (6,5) SPC code to (12,5) code
4	7	5	7	Claim 4	Generator polynomial is $X^8+X^7+4X^6+3X^5+2X^3+X^2+4X+4$
3	4	1	1	(4,3) SPC code and Claim 8	Extend (4,3) SPC code to (12,3) code
2	3	1	1	(3,2) SPC code and Claim 8	Extend (3,2) SPC code to (12,2) code
1	2	1	1	(2,1) SPC code and Claim 8	Extend (2,1) SPC code to (12,1) code

TABLE II
LIST OF k VALUES FOR EXAMPLE 9. THE VALUES OF n', α AND z ARE OBTAINED BY FOLLOWING ALGORITHM 3.

size of W_n^2, $|W_n^2|=\frac{1}{10}\Omega$. The scheme of [1] is then applied separately on W_n^1 and W_n^2 with $\frac{M_1}{N_1}=\frac{2}{15}$ (corresponding to W_n^1) and $\frac{M_2}{N_2}=\frac{13}{15}$ (corresponding to W_n^2). Thus, the overall cache fraction is $0.9\times\frac{2}{15}+0.1\times\frac{13}{15}\approx\frac{1}{5}$. The overall coded gain of this scheme is $g\approx 9$. However, the subpacketization level is $F_s^{MN}=\binom{K}{KM_1/N_1}+\binom{K}{KM_2/N_2}\approx 5\times 10^9$, which is much greater than the subpacketization budget.

In Fig. 2, we present another comparison for system parameters $K=64$ and different values of M/N. The scheme of [1] works for all M/N such that KM/N is an integer. In Fig. 2, our plots have markers corresponding to M/N values that our scheme achieves. For ease of presentation, both the rate (left y-axis) and the logarithm of the subpacketization level (right y-axis) are shown on the same plot. We present results corresponding to two of our construction techniques: (i) the SPC code and (ii) a smaller SPC code coupled with Claim 8. It can be seen that our subpacketization levels are several orders of magnitude smaller with only a small increase in the rate.

An in-depth comparison for general parameters is discussed next. In the discussion below, we shall use the superscript $*$ to refer to the rates and subpacketization levels of our proposed scheme.

B. Comparison with memory-sharing within the scheme of [1]

Suppose that for given K, M and N, a given rate R can be achieved by the memory sharing of the scheme in [1] between the corner points $(M_1,R_1),(M_2,R_2),\cdots,(M_d,R_d)$ where $M_i=\frac{t_iN}{K}$ for some integer t_i. Then $R_i=\frac{K(1-\frac{M_i}{N})}{1+\frac{KM_i}{N}}$, $R=\sum_{i=1}^d\lambda_iR_i$, $M/N=\sum_{i=1}^d\lambda_i\frac{M_i}{N}$ and $\sum_{i=1}^d\lambda_i=1$. The subpacketization level is $F_s^{MS}=\sum_{i=1}^d\binom{K}{\frac{KM_i}{N}}$. In addition, we note that the function $h(x)=K(1-x)/(1+Kx)$ is convex in the parameter $0\leq x\leq 1$. This can be verified by a simple second derivative calculation.

We first argue that F_s^{MS} is lower bounded by $\binom{K}{\frac{KM'}{N}}$, where M' is obtained as follows. For a given M/N, we first determine λ and $\frac{M^*}{N}$ that satisfy the following equations.

$$R=\lambda\frac{K(1-\frac{M^*}{N})}{1+\frac{KM^*}{N}}+(1-\lambda)\frac{K\frac{M^*}{N}}{1+K(1-\frac{M^*}{N})},\text{ and}\quad(5)$$

$$\frac{M}{N}=\lambda\frac{M^*}{N}+(1-\lambda)\left(1-\frac{M^*}{N}\right).\quad(6)$$

Here, $\frac{M^*}{N}\leq\frac{1}{2}$, and $M'=\frac{t'N}{K}$, where t' is the least integer such that $M'\geq M^*$.

To see this, consider the following argument. Suppose that the above statement is not true. Then, there exists a scheme that operates via memory sharing between points $(M_1,R_1),\cdots,(M_d,R_d)$ such that $F_s<\binom{K}{\frac{KM'}{N}}$. Note that $\binom{K}{\frac{KM_1}{N}}<\binom{K}{\frac{KM_2}{N}}$ if $\frac{M_1}{N}<\frac{M_2}{N}\leq\frac{1}{2}$ or $\frac{M_1}{N}>\frac{M_2}{N}\geq\frac{1}{2}$. By the convexity of $h(\cdot)$, we can conclude that (M,R) is not in the convex hull of the corner points $(M_1,R_1),\cdots,(M_d,R_d)$. This is a contradiction.

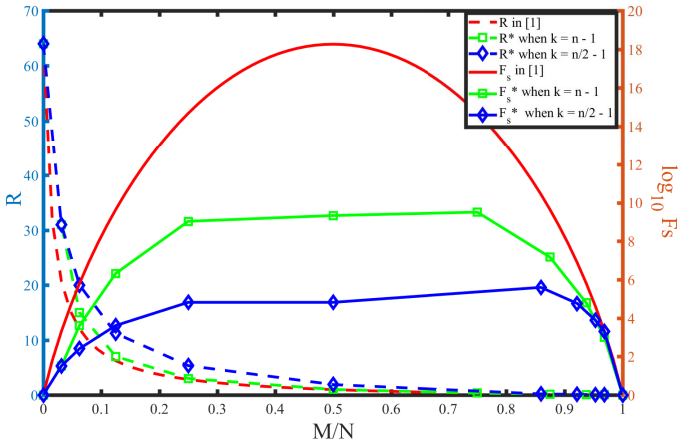

Fig. 2. A comparison of rate and subpacketization level vs. M/N for a system with $K=64$ users. The left y-axis shows the rate and the right y-axis shows the logarithm of the subpacketization level. The green and the blue curves correspond to two of our proposed constructions. Note that our schemes allow for multiple orders of magnitude reduction in subpacketization level and the expense of a small increase in coded caching rate.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

Algorithm 3: $\mathcal{C}(n,q)$ Construction Algorithm

Input : n, q, $\mathcal{C}(n,q)=\emptyset$

if q *is a prime power* **then**
  for $k=1:(n-1)$ **do**
    $n' \leftarrow (n)_{k+1}+k+1$;
    $z \leftarrow \frac{k+1}{\gcd(n',k+1)}$;
    $\alpha \leftarrow \frac{n'}{\gcd(n',k+1)}$;
    if *there exists a* $(n'+i(k+1),k)$ *cyclic code which satisfies the condition in Claim 4 for some* i *such that* $n'+i(k+1)\leq n$ **then**
      $\mathcal{C}(n,q) \leftarrow k$. Corresponding codes are constructed by using Claim 8.
    else
      if $z \leq 2$ **then**
        $\mathcal{C}(n,q) \leftarrow k$. Corresponding codes are constructed by SPC code and Claim 8 when $z=1$ or Claim 7 and Claim 8 when $z=2$.
      else
        if $q+1 \geq n'$ **then**
          $\mathcal{C}(n,q) \leftarrow k$. Corresponding codes are constructed by MDS code and Claim 8.
        else
          if $\alpha = z+1$ *and* $q \geq z$ **then**
            $\mathcal{C}(n,q) \leftarrow k$. Corresponding codes are constructed by Claim 7 and Claim 8.
          end
          if $\alpha > z+1$ *and* $q > \alpha$ **then**
            $\mathcal{C}(n,q) \leftarrow k$. Corresponding codes are constructed by Claim 6 and Claim 8.
          end
        end
      end
    end
  end
end
if q *is not a prime power* **then**
  for $k=1:(n-1)$ **do**
    if $z \leq 2$ **then**
      $\mathcal{C}(n,q) \leftarrow k$. Corresponding codes are constructed by Claim 9 when $z=1$ and Claim 8 or Claim 10 and Claim 8 when $z=2$.
    end
  end
end

Output: $\mathcal{C}(n,q)$

Next, we compare this lower bound on F_s^{MS} to the subpacketization level of our proposed scheme. In principle, we can solve the system of equations (5) and (6) for $R=\frac{n(q-1)}{k+1}$ and $\frac{M}{N}=\frac{1}{q}$ and obtain the appropriate λ and M^* values[1]. Unfortunately, doing this analytically becomes quite messy and does not yield much intuition. Instead, we illustrate the reduction in subpacketization level by numerical comparisons.

[1]Similar results can be obtained for $\frac{M}{N}=1-\frac{k+1}{nq}$

Example 10. Consider a $(9,5)$ linear block code over $GF(2)$ with generator matrix specified below.

$$\mathbf{G}=\begin{bmatrix}1&0&0&0&0&1&1&0&0\\0&1&0&0&0&1&0&1&0\\0&0&1&0&0&1&0&0&1\\0&0&0&1&0&1&1&1&0\\0&0&0&0&1&1&0&1&1\end{bmatrix}.$$

It can be checked that $\mathbf{G}$ satisfies the $(5,6)$-CCP. Thus, it corresponds to a coded caching system with $K=9\times 2=18$ users. Our scheme achieves the point $\frac{M_1}{N}=\frac{1}{2}$, $R_1=\frac{3}{2}$, $F_{s,1}^*=64$ and $\frac{M_2}{N}=\frac{2}{3}$, $R_2=\frac{2}{3}$, $F_{s,2}^*=96$.

On the other hand for $\frac{M_1}{N}=\frac{1}{2}$, $R_1=\frac{3}{2}$, by numerically solving (5) and (6) we obtain $\frac{M_1^*}{N}\approx 0.227$ and therefore $\frac{M_1'}{N}=\frac{5}{18}$. Then $F_{s,1}^{MS}\geq\binom{18}{5}=8568$, which is much higher than $F_{s,1}^*=64$. A similar calculation shows that $\frac{M_2^*}{N}\approx\frac{1}{4}$ and therefore $\frac{M_2'}{N}=\frac{5}{18}$. Thus $F_{s,2}^{MS}$ is also at least as large as 8568, which is still much higher than $F_{s,2}^*=96$.

The next set of comparisons are with other proposed schemes in the literature. We note here that several of these are restrictive in the parameters that they allow.

C. Comparison with [1], [14], [21], [15] and [16]

For comparison with [1], denote R^{MN} and F_s^{MN} be the rate and the subpacketization level of the scheme of [1], respectively. For the rate comparison, we note that

$$\frac{R^*}{R^{MN}}=\frac{1+n}{1+k},\ \text{for}\ \frac{M}{N}=\frac{1}{q}$$
$$\frac{R^*}{R^{MN}}=\frac{nq-k}{nq-n},\ \text{for}\ \frac{M}{N}=1-\frac{1+k}{nq},$$

For the comparison of subpacketization level we have the following results.

Claim 11. When $K=nq$, the following results hold.

- If $\frac{M}{N}=\frac{1}{q}$, we have
$$\lim_{n\to\infty}\frac{1}{K}\log_2\frac{F_s^{MN}}{F_s^*}=H_2\left(\frac{1}{q}\right)-\frac{\eta}{q}\log_2 q. \quad (7)$$
- If $\frac{M}{N}=1-\frac{k+1}{nq}$, we have
$$\lim_{n\to\infty}\frac{1}{K}\log_2\frac{F_s^{MN}}{F_s^*}=H_2\left(\frac{\eta}{q}\right)-\frac{\eta}{q}\log_2 q. \quad (8)$$

In the above expressions, $0<\eta=k/n\leq 1$ and $H_2(\cdot)$ represents the binary entropy function.

Proof. Both results are simple consequences of approximating $\binom{K}{Kp}\approx 2^{KH_2(p)}$ [26]. The derivations can be found in the Appendix. ■

It is not too hard to see that F_s^* is exponentially lower than F_s^{MN}. Thus, our rate is higher, but the subpacketization level is exponentially lower. Thus, the gain in the scaling exponent of with respect to the scheme of [1] depends on the choice of R

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

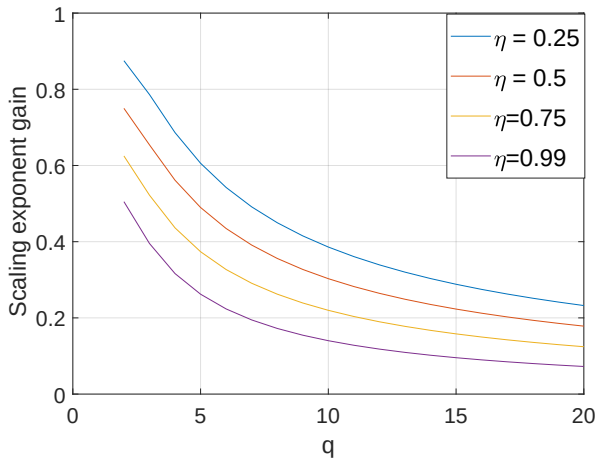

Fig. 3. The plot shows the gain in the scaling exponent obtained using our techniques for different value of $M/N = 1/q$. Each curve corresponds to a choice of $\eta = k/n$.

and the value of M/N. In Fig. 3 we plot this value of different values of R and q. The plot assumes that codes satisfying the CCP can be found for these rates and corresponds to the gain in eq. (7).

In [14] a scheme for the case when $M/N = 1/q$ or $M/N = 1-1/q$ with subpacketization level exponentially smaller with respect to [1] was presented. This result can be recovered a special case of our work (Theorem 1) when the linear block code is chosen as a single parity check code over $\mathbb{Z} \mod q$. In this specific case, q does not need to be a prime power. Thus, our results subsume the results of [14].

In a more recent preprint, reference [21], proposed a caching system with $K = \binom{m}{a}$, $F_s = \binom{m}{b}$ and $M/N = 1 - \binom{a}{\lambda}\binom{m-a}{b-\beta}/\binom{m}{b}$. The corresponding rate is

$$R = \frac{\binom{m}{a+b-2\beta}}{\binom{m}{b}} \min\left\{\binom{m-(a+b-2\beta)}{\lambda}, \binom{a+b-2\beta}{a-\beta}\right\},$$

where m, a, b, β are positive integers and $0 < a < m$, $0 < b < m$, $0 \leq \beta \leq \min\{a, b\}$. While a precise comparison is somewhat hard, we can compare the schemes for certain parameter choices, that were also considered in [21].

Let $a = 2$, $\beta = 1$, $m = 2b$. This corresponds to a coded caching system with $K = b(2b-1) \approx 2b^2$, $\frac{M}{N} = \frac{b-1}{2b-1} \approx \frac{1}{2}$, $F_s = \binom{2b}{b} \approx 2^{2b}$, $R = b$. For comparison with our scheme we keep the transmission rates of both schemes roughly the same and let $n = b^2$, $q = 2$, $k = b-1$. We assume that the corresponding linear block code exists. Then $F_s^* \approx 2^b$, which is better than F_s.

On the other hand if we let $\beta = 0$, $a = 2$, $m = 2qb$, we obtain a coded caching system with $K = \frac{m(m-1)}{2}$, $\frac{M}{N} \approx \frac{1}{q}$, $F_s = \binom{m}{\frac{m}{2q}} \approx (2q)^{\frac{m}{2q}}$, $R^{YAN} = (2q-1)^2$. For keeping the rates the same, we let $n = \frac{m(m-1)}{2q}$, $k = \frac{m(m-1)}{4q(2q-1)} - 1$ so that $F_s^* \approx q^{\frac{m(m-1)}{4q(2q-1)}} \approx q^{\frac{m^2}{8q^2}}$. In this regime, the subpacketization level of [21] will typically be lower.

The work of [15] proposed caching schemes with parameters (i) $K = \binom{m}{a}$, $\frac{M}{N} = 1 - \frac{\binom{m-a}{b}}{\binom{m}{b}}$, $F_s = \binom{m}{b}$ and $R = \frac{\binom{m}{a+b}}{\binom{m}{b}}$, where a, b, m are positive integers and $a + b \leq m$ and (ii) $K = \binom{m}{t}q^t$, $\frac{M}{N} = 1 - \frac{1}{q^t}$, $F_s = q^m(q-1)^t$ and $R = \frac{1}{(q-1)^t}$, where q, t, m are positive integers.

Their scheme (i), is the special case of scheme in [21] when $\beta = 0$. For the second scheme, if we let $t = 2$, [15] shows that $R \approx R^*$, $F_s \approx q^{\sqrt{\frac{K}{2q}}}(\sqrt{q}-1)^2$ and $F_s^* \approx (q-1)q^{\frac{K}{q}-1}$, which means F_s is again better than F_s^*. We emphasize here that these results require somewhat restrictive parameter settings.

Finally, we consider the work of [16]. In their work, they leveraged the results of [27] to arrive at coded caching schemes where the subpacketization is linear in K. Specifically, they show that for any constant M/N, there exists a scheme with rate K^δ, where $\delta > 0$ can be chosen arbitrarily small by choosing K large enough. From a theoretical perspective, this is a positive result that indicates that regimes where linear subpacketization scaling is possible. However, these results are only valid when the value of K is very large. In particular, $K = C^n$ and the result is asymptotic in the parameter n. For these parameter ranges, the result of [16] will clearly be better as compared to our work.

VI. Conclusions and Future Work

In this work we have demonstrated a link between specific classes of linear block codes and the subpacketization problem in coded caching. Crucial to our approach is the consecutive column property which enforces that certain consecutive column sets of the corresponding generator matrices are full-rank. We present several constructions of such matrices that cover a large range of problem parameters. Leveraging this approach allows us to construct families of coded caching schemes where the subpacketization level is exponentially smaller compared to the approach of [1].

There are several opportunities for future work. Even though our subpacketization level is significantly lower than [1], it still scales exponentially with the number of users. Of course, the rate of growth with the number of users is much smaller. There have been some recent results on coded caching schemes that demonstrate the existence of schemes where the subpacketization scales sub-exponentially in the number of users. It would be interesting to investigate whether some of these ideas can be leveraged to obtain schemes that work for practical systems with tens or hundreds of users.

References

[1] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. on Info. Th.*, vol. 60, no. 5, pp. 2856–2867, May 2014.
[2] H. Ghasemi and A. Ramamoorthy, "Improved lower bounds for coded caching," *IEEE Trans. on Info. Th.*, vol. 63, no. 7, pp. 4388–4413, 2017.
[3] A. Sengupta, R. Tandon, and T. C. Clancy, "Improved approximation of storage-rate tradeoff for caching via new outer bounds," in *IEEE Intl. Symposium on Info. Th.*, 2015, pp. 1691–1695.
[4] M. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1029–1040, 2015.
[5] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. on Info. Th.*, vol. 63, no. 2, pp. 1146–1158, Feb 2017.
[6] J. Hachem, N. Karamchandani, and S. Diggavi, "Multi-level coded caching," in *IEEE Intl. Symposium on Info. Th.*, 2014, pp. 56–60.
[7] L. Tang and A. Ramamoorthy, "Coded caching for networks with the resolvability property," in *IEEE Intl. Symposium on Info. Th.*, 2016.

[8] M. Ji, M. F. Wong, A. M. Tulino, J. Llorca, G. Caire, M. Effros, and M. Langberg, "On the fundamental limits of caching in combination networks," in *IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, June 2015, pp. 695–699.
[9] H. Ghasemi and A. Ramamoorthy, "Asynchronous coded caching," in *IEEE Intl. Symposium on Info. Th.*, 2017, pp. 2438–2442.
[10] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *IEEE Intl. Conf. Comm.*, 2015, pp. 5559–5564.
[11] M. E. Fitzpatrick, "4K Sector Disk Drives: Transitioning to the Future with Advanced Format Technologies," Toshiba 4K White Paper, 2011, [Online] Available:http://cdaweb01.storage.toshiba.com/docs/services-support-documents/toshiba_4kwhitepaper.pdf.
[12] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis, "Finite length analysis of caching-aided coded multicasting," in *52nd Annual Allerton Conference on Communication, Control, and Computing*, Sept 2014, pp. 914–920.
[13] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Trans. on Info. Th.*, vol. 62, no. 10, pp. 5524–5537, 2016.
[14] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Trans. on Info. Th.*, vol. 63, no. 9, pp. 5821–5833, 2017.
[15] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," preprint, 2016, [Online] Available: https://arxiv.org/abs/1608.03989.
[16] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using Ruzsa-Szeméredi graphs," in *IEEE Intl. Symposium on Info. Th.*, June 2017, pp. 1237–1241.
[17] O. Olmez and A. Ramamoorthy, "Fractional repetition codes with flexible repair from combinatorial designs," *IEEE Trans. on Info. Th.*, vol. 62, no. 4, pp. 1565 –1591, 2016.
[18] A. S. Tripathy and A. Ramamoorthy, "Capacity of sum-networks for different message alphabets," in *IEEE Intl. Symposium on Info. Th.*, 2015, pp. 606–610.
[19] ——, ""Sum-networks from incidence structures: construction and capacity analysis," *IEEE Trans. on Info. Th.*, 2017 (to appear).
[20] D. R. Stinson, *Combinatorial Designs: Construction and Analysis*. Springer, 2003.
[21] Q. Yan, X. Tang, Q. Chen, and M. Cheng, "Placement delivery array design through strong edge coloring of bipartite graphs," *IEEE Communications Letters*, 2017 (to appear).
[22] R. M. Roth, *Introduction to Coding Theory*. Cambridge University Press, 2006.
[23] S. Lin and D. J. Costello, *Error Control Coding, 2nd Ed.* Prentice Hall, 2004.
[24] D. S. Dummit and R. M. Foote, *Abstract algebra*. Wiley, 3rd Ed., 2003.
[25] I. F. Blake, "Codes over certain rings," *Information and Control*, vol. 20, no. 4, pp. 396–404, 1972.
[26] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete mathematics: a foundation for computer science (2nd ed.)*. Addison-Wesley Professional, 1994.
[27] N. Alon, A. Moitra, and B. Sudakov, "Nearly complete graphs decomposable into large induced matchings and their applications," in *Proc. of the 44-th Annual ACM symposium on Theory of computing (STOC)*, 2012, pp. 1079–1090.
[28] R. A. Horn and C. R. Johnson, *Topics in matrix analysis*. Cambridge University Press, 1991.

APPENDIX

A. Resolvable design over Z mod q

Lemma 4. A (n, k) linear block code over $\mathbb{Z} \bmod q$ with generator matrix $\mathbf{G} = [g_{ab}]$ can construct a resolvable block design by the procedure in Section III-A if $\gcd(q, g_{0b}, g_{1b}, \cdots, g_{(k-1)b}) = 1$ for $0 \leq b < n$.

Proof. Assume $q = q_1 \times q_2 \times \cdots \times q_d$ where q_i, $1 \leq i \leq d$ is a prime or a prime power. If the $\gcd(q, g_{0b}, g_{1b}, \cdots, g_{(k-1)b}) = 1$, then it is evident that $\gcd(q_i, g_{0b}, g_{1b}, \cdots, g_{(k-1)b}) = 1$ for $1 \leq i \leq d$. As q_i is either a prime or a prime power, it follows that there exists a g_{a^*b} which is relatively prime to q_i, i.e., g_{a^*b} is a unit in the ring $\mathbb{Z} \mod q_i$.

Note that for $\Delta = [\Delta_0 \ \Delta_1 \ \ldots \ \Delta_{n-1}] = \mathbf{uG}$, we have

$$\Delta_b = \sum_{a=0}^{k-1} \mathbf{u}_a g_{ab}, \tag{9}$$

where $\mathbf{u} = [\mathbf{u}_0, \cdots, \mathbf{u}_{k-1}]$. We consider eq. (9) over the ring $\mathbb{Z} \mod q_i$ and rewrite eq. (9) as

$$\Delta_b - \mathbf{u}_{a^*} g_{a^*b} = \sum_{a \neq a^*} \mathbf{u}_a g_{ab},$$

For arbitrary $\mathbf{u}_a$, $a \neq a^*$, this equation has a unique solution for $\mathbf{u}_{a^*}$ since g_{a^*b} is a unit in $\mathbb{Z} \mod q_i$. This implies that there are q_i^{k-1} distinct solutions for (9) over $\mathbb{Z} \mod q_i$. Using the Chinese remainder theorem, eq. (9) has $q_1^{k-1} \times q_2^{k-1} \times \cdots q_d^{k-1} = q^{k-1}$ solutions over $\mathbb{Z} \mod q$ and the result follows. ■

Remark 7. From Lemma 4, it can be easily verified that a linear block code over $\mathbb{Z}$ mod q can construct a resolvable block design if one of the following conditions for each column $\mathbf{g}_i$ of the generator matrix is satisfied.

- At least one non-zero entry of $\mathbf{g}_i$ is a unit in $\mathbb{Z} \mod q$, or
- all non-zero entries in $\mathbf{g}_i$ are zero divisors but their greatest common divisor is 1.

For the SPC code over $\mathbb{Z}$ mod q, all the non-zero entries in the generator matrix are 1, which is an unit. Therefore, the construction always results in a resolvable design.

Proof of Lemma 2

First, we show that the proposed delivery scheme allows each user's demand to be satisfied. Note that Claim 2 shows that each user in a parallel class that belongs to the recovery set $\mathcal{S}_a$ recovers all missing subfiles with a specified superscript from it. Thus, we only need to show that if signals are generated (according to Claim 2) for each recovery set, we are done. This is equivalent to showing that the bipartite recovery set graph is such that each parallel class has degree z and multiple edges between nodes are disallowed.

Towards this end, consider the parallel class and we claim that there exist exactly z solutions (a_α, b_α) for integer values of $\alpha = 1, \ldots, z$ to the equation

$$a_\alpha(k+1) + b_\alpha = j + n(\alpha - 1) \tag{10}$$

such that $a_{\alpha_1} \neq a_{\alpha_2}$ for $\alpha_1 \neq \alpha_2$ and $j < n$. The existence of the solution for each equation above follows from the division algorithm. Note that $a_\alpha < nz/(k+1)$ as the $RHS < nz$. Furthermore, note that for $1 \leq \alpha_1 \leq z$ and $1 \leq \alpha_2 \leq z$, we cannot have solutions to eq. (10) such that $a_{\alpha_1} = a_{\alpha_2}$ as this would imply that $|b_{\alpha_1} - b_{\alpha_2}| \geq n$ which is a contradiction. This shows that each parallel class P_j participates in at least z different recovery sets.

The following facts follow easily from the construction of the recovery sets. The degree of each recovery set in the bipartite graph is $k+1$ and there are $\frac{nz}{k+1}$ of them; multiple edges between a recovery set and a parallel class

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

are disallowed. Therefore, the total number of edges in the bipartite graph is nz. As each parallel class participates in at least z recovery sets, by this argument, it participates in exactly z recovery sets. .

Finally, we calculate the rate of the delivery phase. In total, the server transmits $q^k(q-1)\frac{zn}{k+1}$ equations, where the symbol transmitted has the size of a subfile. Thus, the rate is

$$\begin{aligned} R &= q^k(q-1)\frac{zn}{k+1}\frac{1}{q^k z} \\ &= \frac{(q-1)n}{k+1}. \end{aligned}$$

Proof of Claim 4

The matrix $\mathbf{G}_{\mathcal{S}}$ is shown below.

$$\begin{bmatrix} g_0 & g_1 & \cdot & \cdot & g_{\lceil \frac{k}{2} \rceil -1} & 0 & \cdot & \cdot & 0 & 0 \\ 0 & g_0 & g_1 & \cdot & g_{\lceil \frac{k}{2} \rceil -2} & 0 & \cdot & \cdot & 0 & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & \cdot & 0 & g_0 & g_1 & 0 & \cdot & \cdot & 0 & 0 \\ 0 & \cdot & \cdot & 0 & g_0 & g_{n-k} & 0 & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & 0 & 0 & g_{n-k-1} & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & \cdot & \cdot & 0 & 0 & g_{n-k-\lfloor \frac{k}{2} \rfloor} & \cdot & \cdot & \cdot & g_{n-k} \end{bmatrix}$$

In the above expression and the subsequent discussion if i is such that $i < 0$, we set $g_i = 0$.

By Claim 3, a cyclic code with generator matrix $\mathbf{G}$ satisfies the CCP if all submatrices

$$\begin{aligned} \mathbf{G}_{\mathcal{S}\setminus(a+j)_n} = &[\mathbf{g}_{(a)_n}, \mathbf{g}_{(a+1)_n}, \cdots, \mathbf{g}_{(a+j-1)_n}, \\ &\mathbf{g}_{(a+j+1)_n}, \cdots, \mathbf{g}_{(a+k)_n}], \end{aligned}$$

where $a = n - \lfloor \frac{k}{2} \rfloor - 1$, $0 \le j \le k$, have full rank. In what follows, we argue that this is true. Note that in the generator matrix of cyclic code, any k consecutive columns are linearly independent [23]. Therefore for $j = 0$ and k, $\mathbf{G}_{\mathcal{S}\setminus(a+j)_n}$ has full rank, without needing the conditions of Claim 4. For $0 < j \le \lfloor \frac{k}{2} \rfloor$, $\mathbf{G}_{\mathcal{S}\setminus(a+j)_n}$ is as eq. (11).

Rewriting $\mathbf{G}_{\mathcal{S}\setminus(a+j)_n}$ in block form, we get

$$\mathbf{G}_{\mathcal{S}\setminus(a+j)_n} = \left[\begin{array}{c|cc} \mathbf{A}_j & \mathbf{B}_j & \\ \hline \mathbf{0} & \begin{matrix} \mathbf{C}_j & \mathbf{0} \\ \mathbf{D}_j & \mathbf{E}_j \end{matrix} \end{array}\right],$$

where

$$\mathbf{A}_j = \begin{bmatrix} g_0 & g_1 & \cdot & \cdot & g_{\lceil \frac{k}{2} \rceil -1} \\ 0 & g_0 & g_1 & \cdot & g_{\lceil \frac{k}{2} \rceil -2} \\ \vdots & & & & \vdots \\ 0 & \cdot & & 0 & g_0 \end{bmatrix},$$

$$\mathbf{C}_j = \begin{bmatrix} g_{n-k-1} & g_{n-k} & 0 & \cdot & \cdot & 0 \\ g_{n-k-2} & g_{n-k-1} & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & & \vdots \\ g_{n-k-j+1} & \cdot & \cdot & \cdot & \cdot & g_{n-k} \\ g_{n-k-j} & \cdot & \cdot & \cdot & \cdot & g_{n-k-1} \end{bmatrix},$$

and

$$\mathbf{E}_j = \begin{bmatrix} g_{n-k} & 0 & \cdot & \cdot & 0 \\ g_{n-k-1} & g_{n-k} & 0 & \cdot & 0 \\ \vdots & & & & \vdots \\ g_{n-k-\lfloor \frac{k}{2} \rfloor +j+1} & \cdot & \cdot & \cdot & g_{n-k} \end{bmatrix}.$$

Matrices $\mathbf{A}_j$ and $\mathbf{E}_j$ have full rank as they are respectively upper triangular and lower triangular, with non-zero entries on the diagonal (as g_0 and g_{n-k} are non-zero in a cyclic code). Therefore, $\mathbf{G}_{\mathcal{S}\setminus(a+j)_n}$ has full rank if $\mathbf{C}_j$ has full rank. For $\lfloor \frac{k}{2} \rfloor < j < k$, $\mathbf{G}_{\mathcal{S}\setminus(a+j)_n}$ can be partitioned into a similar form and the result in Claim 4 follows.

Proof of Claim 6

We need to argue that all $k \times k$ submatrices of $\mathbf{G}_{\mathcal{S}_a}$ where $0 \le a < \alpha$ are full rank. In what follows we argue that all $k \times k$ submatrices of $\mathbf{G}_{\mathcal{S}_0}$ are full rank. The proof for any $\mathbf{G}_{\mathcal{S}_a}$ is similar. Note that $\mathbf{G}_{\mathcal{S}_0}$ can be written compactly as follows by using Kronecker products.

$$\mathbf{G}_{\mathcal{S}_0} = \begin{bmatrix} \mathbf{A} \otimes \mathbf{I}_t \\ \mathbf{B} \otimes \mathbf{C}_{(t-1)\times t}(1,1) \end{bmatrix},$$

where

$$\mathbf{A} = \begin{bmatrix} b_{00} & \cdots & b_{0(z-1)} \\ \vdots & & \vdots \\ b_{(z-2)0} & \cdots & b_{(z-2)(z-1)} \end{bmatrix}$$

and $\mathbf{B} = [b_{(z-1)0}, \cdots, b_{(z-1)(z-1)}]$.

Next, we check the determinant of submatrices $\mathbf{G}_{\mathcal{S}_0\setminus i}$ obtained by deleting i-th column of $\mathbf{G}_{\mathcal{S}_0}$. W.l.o.g, we let $i = (z-1)t + j$ where $0 \le j < t$. The block form of the resultant matrix $\mathbf{G}_{\mathcal{S}_0\setminus i}$ can be expressed as

$$\mathbf{G}_{\mathcal{S}_0\setminus i} = \begin{bmatrix} \mathbf{A}' \otimes \mathbf{I}_t & \mathbf{A}'' \otimes \Delta_1 \\ \mathbf{B}' \otimes \mathbf{C}_{(t-1)\times t}(1,1) & \mathbf{B}'' \otimes \Delta_2 \end{bmatrix},$$

where $\mathbf{A}'$ and $\mathbf{A}''$ are the first $z-1$ columns and last column of $\mathbf{A}$ respectively. Likewise, $\mathbf{B}'$ and $\mathbf{B}''$ are the first $z-1$ components and last component of $\mathbf{B}$. The matrices Δ_1 and Δ_2 are obtained by deleting the j-th column of $\mathbf{I}_t$ and $\mathbf{C}_{(t-1)\times t}(1,1)$ respectively. Then, using the Schur determinant identity [28], we have

$$\begin{aligned} \det(\mathbf{G}_{\mathcal{S}_0\setminus i}) &= \det(\mathbf{A}' \otimes \mathbf{I}_t)\det(\mathbf{B}'' \otimes \Delta_2 - \\ &\quad \mathbf{B}' \otimes \mathbf{C}_{(t-1)\times t}(1,1) \cdot (\mathbf{A}' \otimes \mathbf{I}_t)^{-1} \cdot \mathbf{A}'' \otimes \Delta_1) \\ &\overset{(1)}{=} \det(\mathbf{A}' \otimes \mathbf{I}_t)\det(\mathbf{B}'' \otimes \Delta_2 - \\ &\quad \mathbf{B}'\mathbf{A}'^{-1}\mathbf{A}'' \otimes \mathbf{C}_{(t-1)\times t}(1,1)\Delta_1) \\ &\overset{(2)}{=} \det(\mathbf{A}' \otimes \mathbf{I}_t)\det((\mathbf{B}'' - \mathbf{B}'\mathbf{A}'^{-1}\mathbf{A}'') \otimes \Delta_2), \end{aligned}$$

where (1) holds by the properties of the Kronecker product [28] and (2) holds since $\mathbf{C}_{(t-1)\times t}(1,1)\Delta_1 = \Delta_2$. Next note that $\det(\Delta_2) \neq 0$. This is because Δ_2 can be denoted as

$$\Delta_2 = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B} \end{array}\right],$$

$$\begin{bmatrix} g_0 & g_1 & \cdot & g_{\lceil \frac{k}{2} \rceil -1} & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & g_0 & \cdot & g_{\lceil \frac{k}{2} \rceil -2} & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & \cdot & g_0 & g_1 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & 0 & g_0 & g_{n-k} & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \cdot & \cdot & 0 & g_{n-k-1} & g_{n-k} & 0 & \cdot & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & \cdot & \cdot & 0 & g_{n-k-j+1} & \cdot & g_{n-k} & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & 0 & g_{n-k-j} & \cdot & g_{n-k-1} & 0 & \cdot & 0 \\ 0 & \cdot & \cdot & 0 & g_{n-k-j-1} & \cdot & g_{n-k-2} & g_{n-k} & \cdot & 0 \\ \vdots & & & & & & & & & \vdots \\ 0 & \cdot & \cdot & 0 & g_{n-k-\lfloor \frac{k}{2} \rfloor} & \cdot & g_{n-k-\lfloor \frac{k}{2} \rfloor +j-1} & g_{n-k-\lfloor \frac{k}{2} \rfloor +j+1} & \cdot & g_{n-k} \end{bmatrix}. \tag{11}$$

where $\mathbf{A}$ is a $j \times j$ upper-triangular matrix;

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 1 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

and $\mathbf{B}$ is a $(t-1-j) \times (t-1-j)$ lower-triangular matrix;

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & \cdots & 1 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 \end{bmatrix}.$$

Next, we define the matrix

$$\mathbf{F} = \begin{bmatrix} b_{00} & b_{01} & \cdots & b_{0(z-1)} \\ b_{10} & b_{11} & \cdots & b_{1(z-1)} \\ \vdots & & & \vdots \\ b_{(z-1)0} & b_{(z-1)1} & \cdots & b_{(z-1)(z-1)} \end{bmatrix}.$$

Another application of the Schur determinant identity yields

$$\begin{aligned} \det(\mathbf{B}'' - \mathbf{B}'\mathbf{A}'^{-1}\mathbf{A}'') &= \frac{\det(\mathbf{F})}{\det(\mathbf{A}')} \\ &\neq 0, \end{aligned}$$

since $\det(\mathbf{F})$ and $\det(\mathbf{A}')$ are both non-zero as their columns have the Vandermonde form. In $\mathbf{F}$, the columns correspond to distinct and non-zero elements from $GF(q)$; therefore, $q > z$. Note however, that the above discussion focused only $\mathbf{G}_{\mathcal{S}_0}$. As the argument needs to apply for all $\mathbf{G}_{\mathcal{S}_a}$ where $0 \leq a < \alpha$, we need $q > \alpha$.

Proof of Claim 7

Note that the matrix in eq. (4) is the generator matrix of (n, k) linear block code over $GF(q)$ where $nz = (z+1)(k+1)$. Since z and $z+1$ are coprime, z is the least positive integer such that $k+1 \mid nz$. To show $\mathbf{G}$ satisfies the CCP, we need to argue that all $k \times k$ submatrices of $\mathbf{G}_{\mathcal{S}_a}$ where $0 \leq a \leq z$ are full rank. It is easy to check that $\mathcal{S}_a = \{0, \cdots, n-1\} \setminus \{t(z-a), t(z-a)+1, \cdots, t(z-a)+t-1\}$. We verify three types of matrix $\mathbf{G}_{\mathcal{S}_a}$ as follows: I. $a = 0$ II. $a = 1$ III. $a > 1$.

- Type I
  When $a = 0$, it is easy to verify that any $k \times k$ submatrix of $\mathbf{G}_{\mathcal{S}_0}$ has full rank since $\mathbf{G}_{\mathcal{S}_0}$ has the form $[\mathbf{I}_{k\times k} | \mathbf{1}_k]$, which is the generator matrix of the SPC code.
- Type II
  When $a = 1$, $\mathbf{G}_{\mathcal{S}_1}$ has the form in eq. (12),
  Case 1: Suppose that we delete any of first $(z-1)t$ columns in $\mathbf{G}_{\mathcal{S}_1}$ (this set of columns is depicted by the underbrace in eq. (12)), say the i-th column of $\mathbf{G}_{\mathcal{S}_1}$, where $z_1 t \leq i < (z_1+1)t$ and $0 \leq z_1 \leq z-2$. Let $i_1 = i - z_1 t$, $i_2 = (z_1+1)t - i - 1$. The resultant matrix $\mathbf{G}_{\mathcal{S}_1 \setminus i}$ can be expressed as follows.

$$\mathbf{G}_{\mathcal{S}_1 \setminus i} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{C} \\ \hline \mathbf{B} & \mathbf{D} \end{array}\right],$$

where

$$\mathbf{A} = \mathbf{I}_{i\times i},$$
$$\mathbf{B} = \mathbf{0}_{(k-i)\times i},$$
$$\mathbf{C} = \begin{bmatrix} \mathbf{0}_{t\times(k-t-i)} & b_1\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times(k-t-i)} & b_2\mathbf{I}_{t\times t} \\ \vdots & \vdots \\ \mathbf{0}_{t\times(k-t-i)} & b_{z_1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{i_1\times(k-t-i)} & b_{z_1+1}\mathbf{I}_{i_1\times i_1} \quad \mathbf{0}_{i_1\times(i_2+1)} \end{bmatrix},$$

and $\mathbf{D}$ has the form in eq. (13).
Note that if $z_1 = 0$, $\mathbf{C} = [\mathbf{0}_{i_1\times(k-t-i)} \; b_1\mathbf{I}_{i_1\times i_1} \; \mathbf{0}_{i_1\times(i_2+1)}]$ and if $z_1 = z-2$,

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_{1\times i_2} & \mathbf{0}_{1\times i_1} & b_{z-1} & \mathbf{0}_{1\times i_2} \\ \mathbf{I}_{i_2\times i_2} & \mathbf{0}_{i_2\times(i_1+1)} & b_{z-1}\mathbf{I}_{i_2\times i_2} \\ \mathbf{0}_{(t-1)\times i_2} & \mathbf{C}(c_1,c_2)_{(t-1)\times t} \end{bmatrix}.$$

To verify $\mathbf{G}_{\mathcal{S}_1 \setminus i}$ has full rank, we just need to check that $\mathbf{D}$ has full rank (as $\mathbf{A}$ is full rank). Checking that $\mathbf{D}$ has full rank can be further simplified as follows. As $b_{z_1+1} \neq 0$, we can move the corresponding column that has b_{z_1+1} as its first entry so that it is the first column of $\mathbf{D}$.
Following this, consider $\mathbf{C}(c_1,c_2)_{(t-1)\times t} \setminus \mathbf{c}_{i_1}$ which is obtained by deleting the i_1-th column of $\mathbf{C}(c_1,c_2)_{(t-1)\times t}$.

$$\mathbf{C}(c_1,c_2)_{(t-1)\times t} \setminus \mathbf{c}_{i_1} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 \\ \mathbf{D}_3 & \mathbf{D}_4 \end{bmatrix},$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

$$\mathbf{G}_{\mathcal{S}_1} = \begin{bmatrix} \mathbf{I}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & b_1\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times t} & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & b_2\mathbf{I}_{t\times t} \\ \vdots & & & & \vdots \\ \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & b_{z-1}\mathbf{I}_{t\times t} \\ \underbrace{\mathbf{0}_{(t-1)\times t} \quad \mathbf{0}_{(t-1)\times t} \quad \cdots \quad \mathbf{0}_{(t-1)\times t}}_{\text{Case 1}} & & & & \underbrace{\mathbf{C}(c_1,c_2)_{(t-1)\times t}}_{\text{Case 2}} \end{bmatrix} \tag{12}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_{1\times i_2} & \mathbf{0}_{1\times t} & \cdots & \mathbf{0}_{1\times t} & \mathbf{0}_{1\times i_1} \; b_{z_1+1} & \mathbf{0}_{1\times i_2} \\ \mathbf{I}_{i_2\times i_2} & \mathbf{0}_{i_2\times t} & \cdots & \mathbf{0}_{i_2\times t} & \mathbf{0}_{i_2\times(i_1+1)} & b_{z_1+1}\mathbf{I}_{i_2\times i_2} \\ \vdots & & & & \vdots & \\ \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & b_{z-1}\mathbf{I}_{t\times t} & \\ \mathbf{0}_{(t-1)\times i_2} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \mathbf{C}(c_1,c_2)_{(t-1)\times t} & \end{bmatrix} \tag{13}$$

where $\mathbf{D}_1$ is a $i_1 \times i_1$ matrix as follows

$$\mathbf{D}_1 = \begin{bmatrix} c_1 & c_2 & 0 & 0 & \cdots & 0 & 0 \\ 0 & c_1 & c_2 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & c_1 & c_2 \\ 0 & 0 & 0 & \cdots & 0 & 0 & c_1 \end{bmatrix},$$

$\mathbf{D}_4$ is a $i_2 \times i_2$ matrix as follows

$$\mathbf{D}_4 = \begin{bmatrix} c_2 & 0 & 0 & \cdots & 0 & 0 & 0 \\ c_1 & c_2 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & \cdots & 0 & c_1 & c_2 & 0 \\ 0 & 0 & \cdots & 0 & 0 & c_1 & c_2 \end{bmatrix},$$

and $\mathbf{D}_2$ and $\mathbf{D}_3$ are $i_1 \times i_2$ and $i_2 \times i_1$ all zero matrices respectively. Then $\det(\mathbf{C}(c_1,c_2)_{(t-1)\times t} \setminus \mathbf{c}_{i_1}) = c_1^{i_1}c_2^{i_2}$ and $\det(\mathbf{G}_{\mathcal{S}_1\setminus i}) = \pm b_{z_1+1}c_1^{i_1}c_2^{i_2} \neq 0$.

Case 2: By deleting any of last t columns in $\mathbf{G}_{\mathcal{S}_1}$, say the i-th column of $\mathbf{G}_{\mathcal{S}_1}$, where $(z-1)t \leq i < zt$, the block form of resultant matrix $\mathbf{G}_{\mathcal{S}_1\setminus i}$ can be expressed as follows.

$$\mathbf{G}_{\mathcal{S}_1\setminus i} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{C} \\ \hline \mathbf{B} & \mathbf{D} \end{array}\right],$$

where $\mathbf{A} = \mathbf{I}_{(z-1)t\times(z-1)t}$, $\mathbf{B} = \mathbf{0}_{(t-1)\times(z-1)t}$, $\mathbf{C}$ is obtained by deleting the $(i-(z-1)t)$-th column of matrix $[b_1\mathbf{I}_{t\times t} \;\; b_2\mathbf{I}_{t\times t} \;\; \cdots \;\; b_{z-1}\mathbf{I}_{t\times t}]^T$ and $\mathbf{D}$ is $\mathbf{C}(c_1,c_2)_{(t-1)\times t} \setminus \mathbf{c}_{i-(z-1)t}$. Since $\det(\mathbf{D}) = c_1^{i-(z-1)t}c_2^{zt-i-1} \neq 0$, $\det(\mathbf{G}_{\mathcal{S}_1\setminus i}) = \pm c_1^{i-(z-1)t}c_2^{zt-i-1} \neq 0$ and therefore $\mathbf{G}_{\mathcal{S}_1\setminus i}$ has full rank.

- Type III when $a > 1$, $\mathbf{G}_{\mathcal{S}_a}$ has the form in eq. (14). As before we perform a case analysis. Each of the cases is specified by the corresponding underbrace in eq. (14).

Case 1: By deleting the i-th column of $\mathbf{G}_{\mathcal{S}_a}$, where $z_1 t \leq i < (z_1+1)t$, $z_1 \leq z-a-1$, $i_1 = i - z_1 t$, and $i_2 = (z_1+1)t - i - 1$, the block form of the resultant matrix $\mathbf{G}_{\mathcal{S}_a\setminus i}$ can be expressed as follows,

$$\mathbf{G}_{\mathcal{S}_a\setminus i} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{C} \\ \hline \mathbf{B} & \mathbf{D} \end{array}\right],$$

where $\mathbf{A} = \mathbf{I}_{i\times i}, \mathbf{B} = \mathbf{0}_{(k-i)\times i}$, $\mathbf{C}$ and $\mathbf{D}$ has the form in eq. (15) and eq. (16), respectively. Note that if $z_1 = 0$, $\mathbf{C} = [\mathbf{0}_{i_1\times i_2} \; \mathbf{0}_{i_1\times t} \; \cdots \; \mathbf{0}_{i_1\times(t-1)} \; \mathbf{1}_{i_1} \; b_1\mathbf{I}_{i_1\times i_1} \; \mathbf{0}_{i_1\times(i_2+1)}]$ and if $z_1 = z-a-1$, $\mathbf{D}$ has the form in (17).

To verify that $\mathbf{G}_{\mathcal{S}_a\setminus i}$ has full rank, we just need to check $\mathbf{D}$ has full rank. Owing to the construction of $\mathbf{D}$, we have to check the determinant of the following $(t+1)\times(t+1)$ matrix.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & 0 & \cdots & b_{z_1+1} & \cdots & 0 \\ 1 & b_{z-a+1} & 0 & \cdots & 0 & \cdots & 0 \\ 1 & 0 & b_{z-a+1} & \cdots & 0 & \cdots & 0 \\ \vdots & & & & \vdots & & \\ 1 & 0 & 0 & \cdots & 0 & \cdots & b_{z-a+1} \end{bmatrix};$$

$\det(\mathbf{F}) = (b_{z-a+1} - b_{z_1+1})b_{z-a+1}^{t-1}$. Since $z_1 \neq z-a$ and then $b_{z_1+1} \neq b_{z-a+1}$, the above matrix has full rank and $\det(\mathbf{G}_{\mathcal{S}_a\setminus i}) = \pm(b_{z-a+1} - b_{z_1+1})b_{z-a+1}^{t-1} \neq 0$, so that $\mathbf{G}_{\mathcal{S}_a\setminus i}$ has full rank.

Case 2: By deleting the i-th column of $\mathbf{G}_{\mathcal{S}_a}$, where $z_1 t \leq i < (z_1+1)t$, $z-a \leq z_1 \leq z-3$, the proof that the resultant matrix has full rank is similar to the case that $z_1 \leq z-a-1$ and we omit it here.

Case 3: By deleting the i-th column of $\mathbf{G}_{\mathcal{S}_a}$, where $(z-2)t \leq i \leq (z-1)t-2$, $i_1 = i-(z-2)t$ and $i_2 = (z-1)t-2-i$, the resultant matrix is as follows,

$$\mathbf{G}_{\mathcal{S}_a\setminus i} = \left[\begin{array}{c|c} \mathbf{A} & \mathbf{C} \\ \hline \mathbf{B} & \mathbf{D} \end{array}\right],$$

where

$$\mathbf{A} = \mathbf{I}_{(z-a)t\times(z-a)t}$$

$$\mathbf{B} = \mathbf{0}_{(k-(z-a)t)\times(z-a)t}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-2)} & \mathbf{1}_t & b_1\mathbf{I}_{t\times t} \\ \vdots & & & & \vdots \\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-2)} & \mathbf{1}_t & b_{z-a}\mathbf{I}_{t\times t} \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-2)} & \mathbf{1}_t & b_{z-a+1}\mathbf{I}_{t\times t} \\ \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-2)} & \mathbf{1}_t & b_{z-a+2}\mathbf{I}_{t\times t} \\ \vdots & & & \vdots & \\ & & \mathbf{I}_{i_1\times i_1} \; \mathbf{0}_{i_1\times i_2} & & \\ \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{1\times i_1} \; \mathbf{0}_{1\times i_2} & \mathbf{1}_{t-1} & \mathbf{C}(c_1,c_2)_{(t-1)\times t} \\ & & \mathbf{0}_{i_2\times i_1} \; \mathbf{I}_{i_2\times i_2} & & \end{bmatrix}$$

$$\mathbf{G}_{\mathcal{S}_a} = \left[\begin{array}{ccc|ccc|c|c|c} \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_1\mathbf{I}_{t\times t} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a}\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+2}\mathbf{I}_{t\times t} \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times(t-1)} & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{1}_{t-1} & \mathbf{C}(c_1,c_2)_{(t-1)\times t} \end{array}\right]$$

Case 1 | Case 2 | Case 3 | Case 4 | Case 5 (14)

$$\mathbf{C} = \begin{bmatrix} \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_1\mathbf{I}_{t\times t} \\ \vdots & & & & & \vdots \\ \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z_1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{i_1\times i_2} & \mathbf{0}_{i_1\times t} & \cdots & \mathbf{0}_{i_1\times(t-1)} & \mathbf{1}_{i_1} & b_{z_1+1}\mathbf{I}_{i_1\times i_1} \quad \mathbf{0}_{i_1\times(i_2+1)} \end{bmatrix} \tag{15}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_{1\times i_2} & \mathbf{0}_{1\times t} & \cdots & \cdots & \cdots & \mathbf{0}_{1\times(t-1)} & 1 & \mathbf{0}_{1\times i_1} \quad b_{z_1+1} \quad \mathbf{0}_{1\times i_2} \\ \mathbf{I}_{i_2\times i_2} & \mathbf{0}_{i_2\times t} & \cdots & \cdots & \cdots & \mathbf{0}_{i_2\times(t-1)} & \mathbf{1}_{i_2} & \mathbf{0}_{i_2\times(i_1+1)} \quad b_{z_1+1}\mathbf{I}_{i_2\times i_2} \\ \mathbf{0}_{t\times i_2} & \mathbf{I}_{t\times t} & \cdots & \cdots & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z_1+2}\mathbf{I}_{t\times t} \\ \vdots & & & & & \vdots & & \\ \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+2}\mathbf{I}_{t\times t} \\ \vdots & & & & & & & \vdots \\ \mathbf{0}_{(t-1)\times i_2} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{1}_{t-1} & \mathbf{C}(c_1,c_2)_{(t-1)\times t} \end{bmatrix} \tag{16}$$

$$\mathbf{D} = \begin{bmatrix} \mathbf{0}_{1\times i_2} & \mathbf{0}_{1\times t} & \cdots & \cdots & \cdots & \mathbf{0}_{1\times(t-1)} & 1 & \mathbf{0}_{1\times i_1} \quad b_{z-a} \quad \mathbf{0}_{1\times i_2} \\ \mathbf{I}_{i_2\times i_2} & \mathbf{0}_{i_2\times t} & \cdots & \cdots & \cdots & \mathbf{0}_{i_2\times(t-1)} & \mathbf{1}_{i_2} & \mathbf{0}_{i_2\times(i_1+1)} \quad b_{z-a}\mathbf{I}_{i_2\times i_2} \\ \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+1}\mathbf{I}_{t\times t} \\ \mathbf{0}_{t\times i_2} & \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+2}\mathbf{I}_{t\times t} \\ \vdots & & & & & & & \vdots \\ \mathbf{0}_{(t-1)\times i_2} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{1}_{t-1} & \mathbf{C}(c_1,c_2)_{(t-1)\times t} \end{bmatrix}. \tag{17}$$

To verify $\mathbf{G}_{\mathcal{S}_a\setminus i}$ has full rank, we need to check the determinant of $\mathbf{D}$. Owing to the construction of $\mathbf{D}$, the following matrix is required to be full rank,

$$\begin{aligned}\mathbf{D}' &= \begin{bmatrix} \mathbf{1}_t & b_{z-a+1}\mathbf{I}_{t\times t} \\ 1 & \mathbf{C}(c_1,c_2)_{(t-1)\times t}(i_1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & b_{z-a+1} & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & b_{z-a+1} & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & & \vdots \\ 1 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & b_{z-a+1} \\ 1 & 0 & \cdots & 0 & c_1 & c_2 & 0 & \cdots & 0 \end{bmatrix},\end{aligned}$$

where $\mathbf{C}(c_1,c_2)_{(t-1)\times t}(i_1)$ denotes the i_1-th row of $\mathbf{C}(c_1,c_2)_{(t-1)\times t}$, $0\le i\le t-2$.

$$\begin{aligned}\det\mathbf{D}' =& \det(b_{z-a+1}\mathbf{I}_{t\times t})\cdot\det(1- \\ & \mathbf{C}(c_1,c_2)_{(t-1)\times t}(i_1)\cdot(b_{z-a+1}^{-1}\mathbf{I}_{t\times t})\cdot\mathbf{1}_t) \\ =& b_{z-a+1}^t(1-b_{z-a+1}^{-1}(c_1+c_2))\end{aligned}$$

Since $b_{z-a+1}\neq 0$ and $c_1+c_2=0$, $\det\mathbf{D}'\neq 0$ and $\mathbf{D}'$ has full rank. Then $\det(\mathbf{D}) = b_{z-a+1}^t(1-b_{z-a+1}^{-1}(c_1+c_2))\neq 0$ and thus $\mathbf{G}_{\mathcal{S}_a\setminus i}$ is full rank.

Case 4: By deleting the i-th column of $\mathbf{G}_{\mathcal{S}_a}$, where $i=(z-1)t-1$, the block form of the resultant matrix $\mathbf{G}_{\mathcal{S}_a\setminus i}$ can be expressed as eq. (18). Evidently, $\det(\mathbf{G}_{\mathcal{S}_a\setminus i}) = \pm b_{z-a+1}^t$, so that $\mathbf{G}_{\mathcal{S}_a\setminus i}$ has full rank.

Case 5: By deleting i-th column of $\mathbf{G}_{\mathcal{S}_a}$, where $(z-1)t\le i<zt$ and $i_1=i-(z-1)t$, the block form of the resultant matrix $\mathbf{G}_{\mathcal{S}_a\setminus i}$ can be expressed as eq. (19).
where $b_s\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}$ denotes the submatrix obtained by deleting i_1-th column of $b_s\mathbf{I}_{t\times t}$ and $\mathbf{C}(c_1,c_2)_{(t-1)\times t}\setminus\mathbf{c}_{i_1}$ denotes the submatrix obtained by deleting i_1-th column of $\mathbf{C}(c_1,c_2)_{(t-1)\times t}\setminus\mathbf{c}_{i_1}$. To verify $\mathbf{G}_{\mathcal{S}_a\setminus i}$ has full rank, we just need to check $[\mathbf{1}_t|b_{z-a+1}\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}]$ has full rank. Since $[\mathbf{1}_t|b_{z-a+1}\mathbf{I}_{t\times t}]$ has the following form,

$$\begin{bmatrix} 1 & b_{z-a+1} & 0 & \cdots & 0 \\ 1 & 0 & b_{z-a+1} & \cdots & 0 \\ \vdots & & & & \vdots \\ 1 & 0 & 0 & \cdots & b_{z-a+1} \end{bmatrix},$$

by deleting any column of above matrix, it is obvious that $\det([\mathbf{1}_t|b_{z-a+1}\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}]) = \pm b_{z-a+1}^{t-1}$ and $\det(\mathbf{G}_{\mathcal{S}_a\setminus i})\neq 0$.

Proof of Claim 8

Let z be the least integer such that $k+1\mid nz$. First, we argue that z is the least integer such that $k+1\mid(n+s(k+1))z$. Assume that this is not true, then there exists $z'<z$ such that

$$\mathbf{G}_{\mathcal{S}_a\setminus i}=\begin{bmatrix}\mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & b_1\mathbf{I}_{t\times t}\\ \vdots & & & & & & \vdots\\ \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & b_{z-a}\mathbf{I}_{t\times t}\\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & b_{z-a+1}\mathbf{I}_{t\times t}\\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & b_{z-a+2}\mathbf{I}_{t\times t}\\ \vdots & & & & & & \vdots\\ \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{C}(c_1,c_2)_{(t-1)\times t}\end{bmatrix} \tag{18}$$

$$\mathbf{G}_{\mathcal{S}_a\setminus i}=\begin{bmatrix}\mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_1\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}\\ \vdots & & & & & & & \vdots\\ \mathbf{0}_{t\times t} & \cdots & \mathbf{I}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a}\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}\\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+1}\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}\\ \mathbf{0}_{t\times t} & \cdots & \mathbf{0}_{t\times t} & \mathbf{I}_{t\times t} & \cdots & \mathbf{0}_{t\times(t-1)} & \mathbf{1}_t & b_{z-a+2}\mathbf{I}_{t\times t}\setminus\mathbf{c}_{i_1}\\ \vdots & & & & & & & \vdots\\ \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{0}_{(t-1)\times t} & \mathbf{0}_{(t-1)\times t} & \cdots & \mathbf{I}_{(t-1)\times(t-1)} & \mathbf{1}_{t-1} & \mathbf{C}(c_1,c_2)_{(t-1)\times t}\setminus\mathbf{c}_{i_1}\end{bmatrix} \tag{19}$$

$k+1\mid(n+s(k+1))z'$. As $n\geq k+1$ and $k+1\mid s(k+1)z'$ this implies that $k+1\mid nz'$ which is a contradiction.

Next we argue that $\mathbf{G}'$ satisfies the CCP, i.e., all $k\times k$ submatrices of each $\mathbf{G}'_{\mathcal{S}'_a}$, where $\mathcal{T}'_a=\{a(k+1),\cdots,a(k+1)+k\}$ and $\mathcal{S}'_a=\{(t)_{n+s(k+1)}|t\in\mathcal{T}'_a\}$ and $0\leq a\leq\frac{nz}{k+1}+sz$, are full rank. Let $n'=n+s(k+1)$. We argue it in three cases.

- *Case 1. The first column of $\mathbf{G}'_{\mathcal{S}'_a}$ lies in the first $s(k+1)$ columns of $\mathbf{G}'$.*
  Suppose $ln'\leq a(k+1)<ln'+s(k+1)$ where $0\leq l<z-1$. By the construction of $\mathbf{G}'$, $\mathbf{G}'_{\mathcal{S}_a}=\mathbf{D}$. Since $\mathbf{D}=\mathbf{G}_{\mathcal{S}_0}$, all $k\times k$ submatrices of $\mathbf{D}$ have full rank and so does $\mathbf{G}'_{\mathcal{S}_a}$.
- *Case 2. The first and last column of $\mathbf{G}'_{\mathcal{S}'_a}$ lie in the last n columns of $\mathbf{G}'$.*
  Suppose $ln'+s(k+1)\leq a(k+1)$ and $a(k+1)+k<(l+1)n'$ where $0\leq l<z-1$. As $n'>s(k+1)$, $a(k+1)-(l+1)s(k+1)>0$ and $k+1\mid a(k+1)-(l+1)s(k+1)$. Let $a'=a-(l+1)s$, then $\mathbf{G}'_{\mathcal{S}_a}=\mathbf{G}_{\mathcal{S}_{a'}}$ and hence all $k\times k$ submatrices of $\mathbf{G}'_{\mathcal{S}_a}$ have full rank.
- *Case 3. The first column of $\mathbf{G}'_{\mathcal{S}'_a}$ lies in the last n columns of $\mathbf{G}'$ but the last column lies in the first $(k+1)$ columns of $\mathbf{G}'$.*
  Suppose $ln'+s(k+1)\leq a(k+1)$ and $a(k+1)+k>(l+1)n'$ where $0\leq l<z-2$. Again, we can get $k+1\mid a(k+1)-(l+1)s(k+1)$ and let $a'=a-(l+1)s$. Let $\mathcal{S}'^1=\{(a(k+1))_{n'},\cdots,(ln'+n'-1)_{n'}\}$ and $\mathcal{S}^1=\{(a'(k+1))_n,\cdots,(ln+n-1)_n\}$. As $(ln'+n'-1)-a(k+1)=(ln+n-1)-a'(k+1)$, $\mathbf{G}'_{\mathcal{S}'^1}=\mathbf{G}_{\mathcal{S}^1}$. Let $\mathcal{S}'^2=\{(ln'+n')_{n'},\cdots,(a(k+1)+k)_{n'}\}$ and $\mathcal{S}^2=\{(ln+n)_n,\cdots,(a'(k+1)+k)_n\}$. By the construction of $\mathbf{G}'$, $\mathbf{G}_{\mathcal{S}^2}=\mathbf{G}'_{\mathcal{S}'^2}$. Then $\mathbf{G}'_{\mathcal{S}_a}=[\mathbf{G}'_{\mathcal{S}'^1}\mathbf{G}'_{\mathcal{S}'^2}]=[\mathbf{G}_{\mathcal{S}^1}\mathbf{G}_{\mathcal{S}^2}]=\mathbf{G}_{\mathcal{S}_{a'}}$ and hence all $k\times k$ submatrices of $\mathbf{G}'_{\mathcal{S}_a}$ have full rank.

Proof of Claim 11

- $\frac{M}{N}=\frac{1}{q}$. We have
$$\frac{1}{K}\log_2\frac{F_s^{MN}}{F_s^*}=\frac{1}{K}\log_2\binom{K}{K/q}-\frac{1}{K}\log_2 z-\frac{k}{K}\log_2 q.$$
Using the fact that $z\leq k+1$ and taking limits as $n\to\infty$, we get that
$$\lim_{n\to\infty}\frac{1}{K}\log_2\frac{F_s^{MN}}{F_s^*}=H_2\left(\frac{1}{q}\right)-\frac{\eta}{q}\log_2 q.$$
- $\frac{M}{N}=1-\frac{k+1}{nq}$. We have
$$\frac{1}{K}\log_2\frac{F_s^{MN}}{F_s^*}=\frac{1}{K}\log_2\binom{K}{k+1}-\frac{k+1}{K}\log_2 q-\frac{1}{K}\log_2\frac{zn}{k+1}.$$
Using the fact that $z\leq k+1$ and taking limits as $n\to\infty$, we get that
$$\lim_{n\to\infty}\frac{1}{K}\log_2\frac{F_s^{MN}}{F_s^*}=H_2\left(\frac{\eta}{q}\right)-\frac{\eta}{q}\log_2 q.$$

B. Discussion on coded caching systems constructed by generator matrices satisfying the (k,α)-CCP where $\alpha\leq k$

Consider the (k,α)-CCP (*cf.* Definition 5) where $\alpha\leq k$. Let z be the least integer such that $\alpha\mid nz$, and let $\mathcal{T}_a^\alpha=\{a\alpha,\cdots,a\alpha+\alpha-1)\}$ and $\mathcal{S}_a^\alpha=\{(t)_n\mid t\in\mathcal{T}_a^\alpha\}$. Let $\mathbf{G}_{\mathcal{S}_a^\alpha}=[\mathbf{g}_{i_0},\cdots,\mathbf{g}_{i_{\alpha-1}}]$ be the submatrix of $\mathbf{G}$ specified by the columns in $\mathcal{S}_a^\alpha$, i.e, $\mathbf{g}_{i_j}\in\mathbf{G}_{\mathcal{S}_a^\alpha}$ if $i_j\in\mathcal{S}_a^\alpha$. We demonstrate that the resolvable design generated from a linear block code that satisfies the (k,α)-CCP can also be used in a coded caching scheme. First, we construct a $(X,\mathcal{A})$ resolvable design as described in Section III.A., which can be partitioned into n parallel classes $\mathcal{P}_i=\{B_{i,j}:0\leq j<q\}$, $0\leq i<n$. By the constructed resolvable design, we partition each subfile W_n into q^kz subfiles $W_n=\{W_{n,t}^s\mid 0\leq t<q^k,0\leq s<z\}$ and operate the placement scheme in Algorithm 1. In the delivery phase, for each recovery set, several equations are generated, each of which benefit α users simultaneously. Furthermore, the equations generated by all the recovery sets can recover all the missing subfiles. In this section, we only show that for the recovery set $\mathcal{P}_{\mathcal{S}_a^\alpha}$, it is possible to generate equations which benefit α users and allow the recovery of all of missing subfiles with given superscript. The subsequent discussion

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

exactly mirrors the discussion in the $(k, k+1)$-CCP case and is skipped.

Towards this end, we first show that picking α users from α distinct parallel classes can always form $q^{k-\alpha+1} - q^{k-\alpha}$ signals. More specifically, consider blocks $B_{i_1,l_{i_1}}, \ldots, B_{i_\alpha,l_{i_\alpha}}$ (where $l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from α distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^\alpha}$. Then, $|\cap_{j=1}^{\alpha-1} B_{i_j,l_{i_j}}| = q^{k-\alpha+1}$ and $|\cap_{j=1}^{\alpha} B_{i_j,l_{i_j}}| = q^{k-\alpha}$.

Claim 12. Consider the resolvable design $(X, \mathcal{A})$ constructed by a (n, k) linear block code that satisfies the (k, α) CCP. Let $\mathcal{P}_{\mathcal{S}_a^\alpha} = \{\mathcal{P}_i \mid i \in \mathcal{S}_a^\alpha\}$ for $0 \leq a < \frac{zn}{\alpha}$, i.e., it is the set of parallel classes corresponding to $\mathcal{S}_a^\alpha$. We emphasize that $|\mathcal{P}_{\mathcal{S}_a^\alpha}| = \alpha \leq k$. Consider blocks $B_{i_1,l_{i_1}}, \ldots, B_{i_{\alpha'},l_{i_{\alpha'}}}$ (where $l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from any α' distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^\alpha}$ where $\alpha' \leq \alpha$. Then, $|\cap_{j=1}^{\alpha'} B_{i_j,l_{i_j}}| = q^{k-\alpha'}$.

The above argument implies that any $\alpha-1$ blocks from any $\alpha-1$ distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^\alpha}$ have $q^{k-\alpha+1}$ points in common and any α blocks $B_{i_1,l_{i_1}}, B_{i_\alpha,l_{i_\alpha}}$ from any α distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^\alpha}$ have $q^{k-\alpha}$ points in common. These blocks (or users) can participate in $q^{k-\alpha+1} - q^{k-\alpha}$ equations, each of which benefits α users. In particular, each user will recover a missing subfile indexed by an element belonging to the intersection of the other $\alpha-1$ blocks in each equation. A very similar argument to Lemma 2 can be made to justify that enough equations can be found that allow all users to recover all missing subfiles.

Proof. Recall that by the construction in Section III.A, block $B_{i,l} \in \mathcal{P}_i$ is specified as follows,

$$B_{i,l} = \{j : \mathbf{T}_{i,j} = l\}.$$

Let $\mathbf{G} = [g_{ab}]$, for $0 \leq a < k$, $0 \leq b < n$.

Now consider $B_{i_1,l_{i_1}}, \ldots, B_{i_{\alpha'},l_{i_{\alpha'}}}$ (where $i_j \in \mathcal{S}_a^\alpha, l_{i_j} \in \{0, \ldots, q-1\}$) that are picked from α' distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^\alpha}$. W.l.o.g. we assume that $i_1 < i_2 < \cdots < i_{\alpha'}$. Let $\mathcal{I} = \{i_1, \ldots, i_{\alpha'}\}$ and $\mathbf{T}_\mathcal{I}$ denote the submatrix of $\mathbf{T}$ obtained by retaining the rows in $\mathcal{I}$. We will show that the vector $[l_{i_1}\ l_{i_2}\ \ldots\ l_{i_{\alpha'}}]^T$ is a column in $\mathbf{T}_\mathcal{I}$ and appears $q^{k-\alpha'}$ times in it.

We note here that by the (k, α)-CCP, the vectors $\mathbf{g}_{i_1}, \mathbf{g}_{i_2}, \ldots, \mathbf{g}_{i_\alpha}$ are linearly independent and thus the subset of these vectors, $\mathbf{g}_{i_1}, \cdots, \mathbf{g}_{i_{\alpha'}}$ are linearly independent. W. l. o. g., we assume that the top $\alpha' \times \alpha'$ submatrix of the matrix $[\mathbf{g}_{i_1}\ \mathbf{g}_{i_2}\ \ldots\ \mathbf{g}_{i_{\alpha'}}]$ is full-rank. Next, consider the system of equations in variables $\mathbf{u}_0, \ldots, \mathbf{u}_{\alpha'-1}$.

$$\begin{aligned}
\sum_{b=0}^{\alpha'-1} \mathbf{u}_b g_{bi_1} &= l_{i_1} - \sum_{b=\alpha'}^{k-1} \mathbf{u}_b g_{bi_1}, \\
\sum_{b=0}^{\alpha'-1} \mathbf{u}_b g_{bi_2} &= l_{i_2} - \sum_{b=\alpha'}^{k-1} \mathbf{u}_b g_{bi_2}, \\
&\vdots \\
\sum_{b=0}^{\alpha'-1} \mathbf{u}_b g_{bi_{\alpha'}} &= l_{i_{\alpha'}} - \sum_{b=\alpha'}^{k-1} \mathbf{u}_b g_{bi_{\alpha'}}.
\end{aligned}$$

By the assumed condition, it is evident that this system of α' equations in α' variables has a unique solution for a given vector $\mathbf{v} = [\mathbf{u}_{\alpha'}, \cdots, \mathbf{u}_{k-1}]$ over $GF(q)$. Since there are $q^{k-\alpha'}$ possible $\mathbf{v}$ vectors, the result follows. ■

As in the case of the $(k, k+1)$-CCP, we form a recovery set bipartite graph with parallel classes and recovery sets as the disjoint vertex subsets, and the edges incident on each parallel class are labeled arbitrarily from 0 to $z-1$. For a parallel class $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a^\alpha}$ we denote this label by label$(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a^\alpha})$. For a given recovery set $\mathcal{P}_{\mathcal{S}_a^\alpha}$, the delivery phase proceeds by choosing blocks from α distinct parallel classes in $\mathcal{P}_{\mathcal{S}_a^\alpha}$ and it provides $q^{k-\alpha+1} - q^{k-\alpha}$ equations that benefit α users. Note that in the (k, α)-CCP case, randomly picking α blocks from α parallel classes in $\mathcal{P}_{\mathcal{S}_a^\alpha}$ will always result in $q^{k-\alpha}$ intersections, which is different from $(k, k+1)$-CCP. It turns out that each equation allows a user in $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a^\alpha}$ to recover a missing subfile with superscript label$(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a^\alpha})$.

Let the demand of user $U_{B_{i,j}}$ for $i \in n-1, 0 \leq j \leq q-1$ by $W_{\kappa_{i,j}}$. We formalize the argument in Algorithm 4 and prove that equations generated in each recovery set $\mathcal{P}_{\mathcal{S}_a^\alpha}$ can recover all missing subfile with superscript label$(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a^\alpha})$.

Algorithm 4: Signal Generation Algorithm for $\mathcal{P}_{\mathcal{S}_a^\alpha}$

Input : For $\mathcal{P} \in \mathcal{P}_{\mathcal{S}_a^\alpha}$, $E(\mathcal{P}) =$ label$(\mathcal{P} - \mathcal{P}_{\mathcal{S}_a^\alpha})$. Signal set $Sig = \emptyset$.

while *any user* $U_B \in \mathcal{P}_j, j \in \mathcal{S}_a^\alpha$ *does not recover all its missing subfiles with superscript* $E(\mathcal{P})$ **do**

Pick blocks $B_{j,l_j} \in \mathcal{P}_j$ for all $j \in \mathcal{S}_a^\alpha$ and $l_j \in \{0, \ldots, q-1\}$;

`/* Pick blocks from distinct parallel classes in` $\mathcal{P}_{\mathcal{S}_a^\alpha}$`. The cardinality of their intersection is always` $q^{k-\alpha}$ `*/`

Find set $\hat{L}_s = \cap_{j \in \mathcal{S}_a^\alpha \setminus \{s\}} B_{j,l_j} \setminus \cap_{j \in \mathcal{S}_a^\alpha} B_{j,l_j}$ for $s \in \mathcal{S}_a^\alpha$;

`/* Determine the missing subfile indices that the user from` $\mathcal{P}_s^\alpha$ `will recover. Note that` $|\hat{L}_s| = q^{k-\alpha+1} - q^{k-\alpha}$ `*/`

Add signals $\oplus_{s \in S_a^\alpha} W^{E(\mathcal{P}_s)}_{\kappa_{s,l_s}, \hat{L}_s[t]}$, $0 \leq t < q^{k-\alpha+1} - q^{k-\alpha}$, to Sig;

`/* User` $U_{B_{s,l_s}}$ `demands file` $W_{\kappa_{s,l_s}}$`. This equation allows it to recover the corresponding missing subfile index` $\hat{L}_s[t]$`, which is the` t`-th element of` $\hat{L}_s[t]$`. The superscript is determined by the recovery set bipartite graph */`

end

Output: Signal set Sig.

For the sake of convenience we argue that user $U_{B_{\beta,l_\beta}}$ that demands $W_{\kappa_{\beta,l_\beta}}$ can recover all its missing subfiles with superscript $E(\mathcal{P}_\beta)$. Note that $B_{\beta,l_\beta} = q^{k-1}$. Thus user $U_{B_{\beta,l_\beta}}$ needs to obtain $q^k - q^{k-1}$ missing subfiles with superscript $E(\mathcal{P}_\beta)$. The delivery phase scheme repeatedly picks α users from different parallel classes of $\mathcal{P}_{\mathcal{S}_a^\alpha}$. The equations in Algorithm 4 allow $U_{B_{\beta,l_\beta}}$ to recover all $W^{E(\mathcal{P}_\beta)}_{\kappa_{\beta,l_\beta}, \hat{L}_\beta[t]}$ where $\hat{L}_\beta =$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

$\cap_{j\in\mathcal{S}_a^\alpha\setminus\{\beta\}}B_{j,l_j}\setminus\cap_{j\in\mathcal{S}_a^\alpha}B_{j,l_j}$ and $t=1,\cdots,q^{k-\alpha+1}-q^{k-\alpha}$. This is because of Claim 12.

Next, we count the number of equations that $U_{B_{\beta,l_\beta}}$ participates in. We can pick $\alpha-1$ users from $\alpha-1$ parallel classes in $\mathcal{P}_{\mathcal{S}_a^\alpha}$. There are totally $q^{\alpha-1}$ ways to pick them, each of which generate $q^{k-\alpha+1}-q^{k-\alpha}$ equations. Thus there are a total of q^k-q^{k-1} equations in which user $U_{B_{\beta,l_\beta}}$ participates in.

It remains to argue that each equation provides a distinct file part of user $U_{B_{\beta,l_\beta}}$. Towards this end, let $\{i_1,\cdots,i_{\alpha-1}\}\subset\mathcal{S}_a^\alpha$ be an index set such that $\beta\notin\{i_1,\cdots,i_{\alpha-1}\}$ but $\beta\in\mathcal{S}_a^\alpha$. Note that when we pick the same set of blocks $\{B_{i_1,l_{i_1}},\cdots,B_{i_{\alpha-1},l_{i_{\alpha-1}}}\}$, it is impossible that the recovered subfiles $W^{E(\mathcal{P}_\beta)}_{\kappa_{\beta,l_\beta},\hat{L}_\beta[t_1]}$ and $W^{E(\mathcal{P}_\beta)}_{\kappa_{\beta,l_\beta},\hat{L}_\beta[t_2]}$ are the same since the points in $\hat{L}_\beta$ are distinct. Next, suppose that there exist sets of blocks $\{B_{i_1,l_{i_1}},\cdots,B_{i_{\alpha-1},l_{i_{\alpha-1}}}\}$ and $\{B_{i_1,l'_{i_1}},\cdots,B_{i_{\alpha-1},l'_{i_{\alpha-1}}}\}$ such that $\{B_{i_1,l_{i_1}},\cdots,B_{i_{\alpha-1},l_{i_{\alpha-1}}}\}\neq\{B_{i_1,l'_{i_1}},\cdots,B_{i_{\alpha-1},l'_{i_{\alpha-1}}}\}$, but $\gamma\in\cap_{j=1}^{\alpha-1}B_{i_j,l_{i_j}}\setminus B_{\beta,l_\beta}$ and $\gamma\in\cap_{j=1}^{\alpha-1}B_{i_j,l'_{i_j}}\setminus B_{\beta,l'_\beta}$. This is a contradiction since this in turn implies that $\gamma\in\cap_{j=2}^{\alpha}B_{i_j,l_{i_j}}\bigcap\cap_{j=2}^{\alpha}B_{i_j,l'_{i_j}}$, which is impossible since two blocks from the same parallel class have an empty intersection.

Finally we calculate the transmission rate. In Algorithm 4, for each recovery set, we transmit $q^{k+1}-q^k$ equations and there are totally $\frac{zn}{\alpha}$ recovery sets. Since each equation has size equal to a subfile, the rate is given by

$$\begin{aligned}R&=(q^{k+1}-q^k)\times\frac{zn}{\alpha}\times\frac{1}{zq^k}\\&=\frac{n(q-1)}{\alpha}.\end{aligned}$$

The (n,k) linear block codes that satisfy the (k,α)-CCP over $GF(q)$ correspond to a coded caching system with $K=nq$, $\frac{M}{N}=\frac{1}{q}$, $F_s=zq^k$ and have a rate $R=\frac{n(q-1)}{\alpha}$. Thus, the rate of this system is a little higher compared to the $(k,k+1)$-CCP system with almost the same subpacketization level.

However, by comparing Definitions 4 and 5 it is evident that the rank constraints of the (k,α)-CCP are weaker as compared to the $(k,k+1)$-CCP. Therefore, in general we can find more instances of generator matrices that satisfy the (k,α)-CCP. For example, a large class of codes that satisfy the (k,k)-CCP are (n,k) cyclic codes since any k consecutive columns in their generator matrices are linearly independent [23]. Thus, (n,k) cyclic codes always satisfy the (k,k)-CCP but satisfy $(k,k+1)$-CCP if they satisfy the additional constraints discussed in Claim 3.

C. Cyclic codes over $\mathbb{Z}\mod q$ [25]

First, we show that matrix $\mathbf{T}$ constructed by constructed by the approach outlined in Section IV-D still results in a resolvable design. Let $\Delta=[\Delta_0\Delta_1\cdots\Delta_{n-1}]$ be a codeword of the cyclic code over $\mathbb{Z}\mod q$, denoted $\mathcal{C}$ where $q=q_1q_2\cdots q_d$, and $q_i,i=1,\ldots,d$ are prime. By using the Chinese remaindering map ψ (discussed in Section IV-D), Δ can be uniquely mapped into d codewords $\mathbf{c}^{(i)},i=1,\ldots,d$ where each $\mathbf{c}^{(i)}$ is a codeword of $\mathcal{C}^i$ (the cyclic code over $GF(q_i)$). Thus, the b-th component Δ_b can be mapped to $(\mathbf{c}_b^{(1)},\mathbf{c}_b^{(2)},\ldots,\mathbf{c}_b^{(d)})$

Let $\mathbf{G}^i=[g_{ab}^{(i)}]$ represent the generator matrix of the code $\mathcal{C}^i$. Based on prior arguments, it is evident that there are $q_i^{k_i-1}$ distinct solutions over $GF(q_i)$ to the equation $\sum_{a=0}^{k_i-1}\mathbf{u}_ag_{ab}^{(i)}=\mathbf{c}_b^{(i)}$. In turn, this implies that Δ_b appears $q_1^{k_1-1}q_2^{k_2-1}\cdots q_d^{k_d-1}$ times in the b-th row of $\mathbf{T}$ and the result follows.

Next, we show that any α blocks from distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^{k_{min}}}$ have $q_1^{k_1-\alpha}q_2^{k_2-\alpha}\cdots q_d^{k_d-\alpha}$ intersections, where $\alpha\leq k_{min}$ and $\mathcal{S}_a^{k_{min}}=\{(ak_{min})_n,(ak_{min}+1)_n,\cdots,(ak_{min}+k_{min}-1)_n\}$

Towards this end consider $B_{i_1,l_{i_1}},\ldots,B_{i_\alpha,l_{i_\alpha}}$ (where $i_j\in\mathcal{S}_a^{k_{min}},l_{i_j}\in\{0,\ldots,q-1\}$) that are picked from α distinct parallel classes of $\mathcal{P}_{\mathcal{S}_a^{k_{min}}}$. W.l.o.g. we assume that $i_1<i_2<\cdots<i_\alpha$. Let $\mathcal{I}=\{i_1,\ldots,i_\alpha\}$ and $\mathbf{T}_\mathcal{I}$ denote the submatrix of $\mathbf{T}$ obtained by retaining the rows in $\mathcal{I}$. We will show that the vector $[l_{i_1}\ l_{i_2}\ \ldots\ l_{i_\alpha}]^T$ is a column in $\mathbf{T}_\mathcal{I}$ and appears $q_1^{k_1-\alpha}q_2^{k_2-\alpha}\cdots q_d^{k_d-\alpha}$ times.

Let $\psi_m(l_{i_j})$ for $m=1,\ldots,d$ represent the m-th component of the map ψ. Consider the (n,k_1) cyclic code over $GF(q_1)$ and the system of equations in variables $\mathbf{u}_0,\ldots,\mathbf{u}_{\alpha-1}$ that lie in $GF(q_1)$.

$$\begin{aligned}\sum_{b=0}^{\alpha-1}\mathbf{u}_bg_{bi_1}^{(1)}&=\psi_1(l_{i_1})-\sum_{b=\alpha}^{k_1-1}\mathbf{u}_bg_{bi_1}^{(1)},\\\sum_{b=0}^{\alpha-1}\mathbf{u}_bg_{bi_2}^{(1)}&=\psi_1(l_{i_2})-\sum_{b=\alpha}^{k_1-1}\mathbf{u}_bg_{bi_2}^{(1)},\\&\vdots\\\sum_{b=0}^{\alpha-1}\mathbf{u}_bg_{bi_\alpha}^{(1)}&=\psi_1(l_{i_\alpha})-\sum_{b=\alpha}^{k_1-1}\mathbf{u}_bg_{bi_\alpha}^{(1)}.\end{aligned}$$

By arguments identical to those made in Claim 12 it can be seen that this system of equations has $q_1^{k_1-\alpha}$ solutions. Applying the same argument to the other cyclic codes we conclude that the vector $[l_{i_1},l_{i_2},\cdots,l_{i_\alpha}]$ appears $q_1^{k_1-\alpha}q_2^{k_2-\alpha}\cdots q_d^{k_d-\alpha}$ times in $\mathbf{T}_\mathcal{I}$ and the result follows.

Li Tang received his B.E. degree in mechanical engineering from Beihang University, Beijing, China in 2011, and M.S. degree in electrical and information engineering from Beihang University, Beijing, China in 2014. He is currently working towards the Ph.D degree in the Department of Electrical and Computer Engineering at Iowa State University, Ames, IA, USA. His research interests include network coding and channel coding.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIT.2018.2800059, IEEE Transactions on Information Theory

Aditya Ramamoorthy (M'05) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, in 1999, and the M.S. and Ph.D. degrees from the University of California, Los Angeles (UCLA), in 2002 and 2005, respectively. He was a systems engineer with Biomorphic VLSI Inc. until 2001. From 2005 to 2006, he was with the Data Storage Signal Processing Group of Marvell Semiconductor Inc. Since fall 2006, he has been with the Electrical and Computer Engineering Department at Iowa State University, Ames, IA 50011, USA. His research interests are in the areas of network information theory, channel coding and signal processing for bioinformatics and nanotechnology. Dr. Ramamoorthy served as an editor for the IEEE Transactions on Communications from 2011 – 2015. He is currently serving as an associate editor for the IEEE Transactions on Information Theory. He is the recipient of the 2012 Early Career Engineering Faculty Research Award from Iowa State University, the 2012 NSF CAREER award, and the Harpole-Pentair professorship in 2009 and 2010.