<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Some constructions of storage codes from grassmann graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Frédérique, Oggier</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Oggier, F. (2014). Some Constructions of Storage Codes from Grassmann Graphs. International Zurich Seminar on Communications (IZS), February 26–28, 2014 (pp.51-54).</td>
</tr>
<tr>
<td><strong>Date</strong></td>
<td>2014</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/10220/19354">http://hdl.handle.net/10220/19354</a></td>
</tr>
</tbody>
</table>

© 2014 ETH-Zürich. This paper was published in International Zurich Seminar on Communications (IZS) and is made available as an electronic reprint (preprint) with permission of ETH-Zürich. The paper can be found at the following official DOI: [http://dx.doi.org/10.3929/ethz-a-010094830]. One print or electronic copy may be made for personal use only. Systematic or multiple reproduction, distribution to multiple locations via electronic or other means, duplication of any material in this paper for a fee or for commercial purposes, or modification of the content of the paper is prohibited and is subject to penalties under law.
Some Constructions of Storage Codes from Grassmann Graphs

Frédérique Oggier
Division of Mathematical Sciences
Nanyang Technological University
Singapore
Email: frederique@ntu.edu.sg

Abstract — Codes for distributed storage systems may be seen as families of $m$-dimensional subspaces of the vector space $\mathbb{F}_q^n$, where $\mathbb{F}_q$ is the finite field with $q$ elements, $q$ a prime power. We consider the Grassmann graph $G_{m,n}$ which has for vertex set the collection of $m$-dimensional subspaces of $\mathbb{F}_q^n$, and two vertices are adjacent whenever they intersect in a hyperplane. To obtain subspaces with regular intersection pattern, we look for cliques in the Grassmann graph, and obtain preliminary examples of storage codes, whose parameters we study, in terms of storage overhead, and repairability.

I. INTRODUCTION

When data is stored across a network of nodes, it is usually replicated several times and the copies are stored on distinct nodes, to prevent data loss in case of node failures. From a coding point of view, this means that the data is encoded using a repetition code. It is thus natural to replace this code by a more efficient code, such as a maximal distance separable (MDS) code, which ensures the maximum reliability, given a storage overhead (or amount of data stored, versus amount of actual data). There is however a major difference between classical coding theory, and the design of codes for distributed storage systems, that of repairability. When some coefficients of a codeword are missing, it is desirable to recover these missing coefficients by downloading data from live nodes, without having to (necessarily) decode the codeword.

There has been an intense research activity around the notion of repairability over the past few years, and there is no complete consensus as to what “good” repairability is. In [1], the authors propose adaptations of Reed-Solomon codes, where extra bits of parity are added to allow easy degraded reads, that is, to allow the data to be read, even though some coefficients of the codeword are missing. In [2], [3], repairs are done in a collaborative manner, that is once several coefficients of a codeword are missing, several nodes try to reconstruct the missing coefficients, by possibly exchanging data among each others. The authors focus on minimizing the amount of data downloaded per repair, called repair bandwidth. In [4] instead, repairs are optimized by contacting as little live nodes as possible. A survey of different design criteria for good repairability, and corresponding code constructions, is available in [5].

In this paper, we consider a different view point. We do not try to a priori design codes with respect to one of the known design criteria - repair bandwidth, degraded reads, or local repairs. Instead, we abstract codes for distributed storage systems as families of $m$-dimensional subspaces of the vector space $\mathbb{F}_q^n$ (or subset of the Grassmannian $G(m,n)$), try to design these subspaces with regular intersections, and analyze the preliminary examples obtained in terms of the relevance of their parameters to storage applications.

A similar formalization of storage codes in terms of linear subspaces (not in the context of collaborative repair) has been presented in [6]. The design of subsets of $G(m,n)$ with particular intersection has also been studied in the context of constant dimension codes for network coding [7].

We start by describing codes for distributed storage systems and abstract them in terms of subspaces and their intersection in Section II. To obtain subspaces whose pairwise intersection is of a given dimension, we look for cliques in the Grassmann graph $G_{2}(m,n)$. The graph $G_{2}(n-1,n)$ is considered in Subsection III-A. It is the simplest to understand, gives codes with minimum repair bandwidth, but unreasonable storage overhead. We then compute some other examples from other Grassmann graphs. A clique from the graph $G_{2}(5,3)$ is computed in Subsection III-B, yielding a storage code with a slightly better overhead than the previous examples. A clique from the graph $G_{2}(6,3)$ is reported in Subsection III-C, which offers different (collaborative) repair options.

II. SYSTEM MODEL

We consider a storage network, composed of $N$ storage nodes. Let $o \in \mathbb{F}_q^n$ be a data object, represented as a row vector of length $B$ with coefficients in the finite field $\mathbb{F}_q$, to be stored over this network. The object is stored using a linear erasure code, that is $o$ is mapped to a codeword whose coefficients are stored over the storage network. Since the erasure code used for storage is linear, we will represent it as a family of vectors $\{v_j \in \mathbb{F}_q^n, j = 1, \ldots, n\}$, $n \geq B$. Every storage node then contains some codeword coefficients of the form $o v_j^T$, for some $j \in I \subset \{1, \ldots, n\}$. Since every node is enabled of computational power, it can compute linear combinations of the stored data, that is $o \sum_{j \in I} a_j v_j^T$, $a_j \in \mathbb{F}_q$. This means that we can model the data stored at each node by a vector subspace $W_i = \langle v_j, j \in I_i \rangle \subset \mathbb{F}_q^n$.

We assume that $\dim_{\mathbb{F}_q}(W_i) = \alpha, i = 1, \ldots, N$. 

51
A. Collaborative Repair

Suppose that a repair process is triggered after \( t \) failures, thus \( t \) live nodes will start downloading coefficients \( v_{ij} \), \( i \) of them, each from \( d \) live nodes. Thus every node participating in the repair process obtains \( d \) subspaces

\[
W_{il} \leq W_i, \quad l \in D_r, \quad |D_r| = d, \quad r = 1, \ldots , t.
\]

The second index \( l \) tells the provenance of the subspace (the live node \( W_i \)), while the first index \( r \) tells which node is being repaired (without loss of generality, and to simplify the notation, we have reordered the nodes so that \( W_1, \ldots , W_t \) are repaired). We assume that \( \dim(V_r(W_t)) = \beta \), for all \( r \). There is no point for a node to download redundant data, thus we may assume that every of the \( t \) nodes each gets a subspace \( V_r, \ r = 1, \ldots , t \), where

\[
V_r = \langle \oplus_{l \in D_r} W_{il} \rangle, \quad |D_r| = d,
\]

thus \( \dim(V_r) = d\beta \). Finally, these \( t \) nodes exchange some data among each other, say each of them will receive some subspaces \( V_{il} \) each of dimension \( \beta' \), where \( l \in T_r \) indicates the provenance of the data, and \( |T_r| = T \) how many subspaces are received. Note that \( T \) may vary from 1 to \( t-1 \), but is the same among the nodes performing the repair. The case \( t = 1 \) corresponds to the repair of one node failure, when there is no collaboration, while \( |T| = t-1 \) is the scenario studied in [2], [3], where every repair node exchanges data with the others. For a repair of \( t \) faults to be successful, it is necessary that

\[
\dim(V_r) + \sum_{l \in T_r} \dim(V_{il}) = \alpha.
\]

We may indeed assume that the \( V_{il} \) at one node are not intersecting, since there is no need to transfer redundant data.

Finally, the information stored across the network must be preserved through the repair process. In the case of exact repair, every of the \( t \) subspaces lost has been reconstructed, while for functional repair, the \( t \) subspaces generated during the collaborative process might be different from those lost, but the overall amount of information about the stored object stays the same.

Consider the case of exact repair. Then we must have

\[
W_1 = \langle v_j, \ j \in I_1 \rangle = \langle V_i, \oplus_{l \in T_i} W_{il} \rangle = \langle \oplus_{l \in D_i} W_{il}, \oplus_{l \in T_i} V_i \rangle,
\]

which forces the subspaces \( W_i \) to intersect in a specific manner. For example, if \( t = 2 \) nodes cooperate, the node repairing node 1 will receive \( V_{12} \) from the node repairing node 2, and send \( V_{21} \). Thus after the cooperation phase, both nodes will intersect on \( \langle V_{12}, V_{21} \rangle \), a subspace of dimension \( 2\beta' \).

B. Object Recovery

If needed, the data object \( o \) should be retrievable, despite the presence of potential node failures. We may want the constraint that \( o \) can be computed by contacting any choice of \( k \) out of the \( N \) storage nodes that store \( o \) (as in [2], [3]). This is not a necessary condition, one may alternatively prefer that \( o \) can be recover out of many sets of \( k \) storage nodes (as in [4]).

III. Some Examples of Constructions

Let \( V \) be an \( n \)-dimensional vector space over \( \mathbb{F}_q \), for \( q \) a prime power.

**Definition 1:** [8, 9.3] The Grassmann graph \( G_q(n, m) \) of the \( m \)-subspaces of \( V \) has for vertex set the collection of linear subspaces of \( V \) of dimension \( m \). Two vertices \( W, W' \) are adjacent whenever \( \dim(W \cap W') = m-1 \), that is, \( W \) and \( W' \) intersect in a hyperplane.

Let \( \left[ \begin{array}{c} n \\ m \end{array} \right] \) be the \( q \)-ary Gaussian binomial coefficient

\[
\binom{n}{m} = \frac{(q^n-1) \cdots (q^{n-m+1}-1)}{(q^m-1) \cdots (q-1)}.
\]  

The number of vertices of \( G_q(n, m) \) is \( \left[ \begin{array}{c} n \\ m \end{array} \right] \), and every vertex has degree

\[
q \binom{n-m}{m} = q \frac{(q^{n-m}-1)(q^m-1)}{(q-1)^2}.
\]

We recall some well-known formulas about the dimension of sums of vector subspaces.

**Lemma 1:** Let \( W_1, W_2, W_3, W_4 \) be any \( m \)-dimensional subspaces of \( \mathbb{F}_q^n \). Then

\[
dim(W_1 + W_2) = \dim(W_1) + \dim(W_2) - \dim(W_1 \cap W_2).
\]

Similarly for 3 subspaces

\[
dim(W_1 + W_2 + W_3) = \sum_{i=1}^{3} \dim(W_i) - \dim(W_2 \cap W_3) - \dim(W_1 \cap (W_2 + W_3))
\]

and for 4 subspaces:

\[
dim(W_1 + W_2 + W_3 + W_4) = \sum_{i=1}^{4} \dim(W_i) - \dim(W_2 \cap W_3) - \dim((W_1 + W_2)^{\perp} \cap (W_3 + W_4))
\]

**Proof:** The first formula (3) is well-known. The second is obtained by applying it recursively:

\[
dim(W_1 + W_2 + W_3) = \dim(W_1) + \dim(W_2 + W_3) - \dim(W_1 \cap (W_2 + W_3))
\]

\[
= \sum_{i=1}^{3} \dim(W_i) - \dim(W_2 \cap W_3) - \dim(W_1 \cap (W_2 + W_3))
\]

and so is (5):

\[
dim(W_1 + W_2 + W_3 + W_4) = \dim(W_1 + W_2) + \dim(W_3 + W_4)
\]

\[
- \dim((W_1 + W_2) \cap (W_3 + W_4))
\]

\[
= \sum_{i=1}^{4} \dim(W_i) - \dim(W_1 \cap W_2)
\]

\[
- \dim(W_3 \cap W_4) - \dim((W_1 + W_2) \cap (W_3 + W_4)).
\]

\[\blacksquare\]
A. The Graph $G_2(n,n-1)$

If $m = n - 1$, then from (1) the number of vertices of $G_2(n,n-1)$ is

$$\frac{(2^n - 1)}{(2 - 1)} = 2^n - 1$$

and from (2) the degree of each vertex is

$$\frac{2(2^n - 1)}{(2 - 1)} = 2^n - 2$$

showing that the graph is complete, and any two subspaces $W,W'$ intersecting are intersecting in a subspace of dimension $m - 1 = n - 2$. Now from (3)

$$\dim(W + W') = 2m - \dim(W \cap W') = 2n - 2 - (n - 2) = n.$$

This corresponds to the case $k = 2$, where an object may be recovered from any two nodes. The repair of one failure can (of course) be done by contacting 2 live nodes (and 2 live nodes are needed). Indeed, if $W_i$ needs to be repaired, contacting any node $W_j$ allows to get $W_i \cap W_j$ of dimension $n - 2$, and only one subspace of dimension 1 is missing, which can obtained from another node $W_i$, $i \neq l$; every $W_i$ intersects $W_j$ in a subspace of dimension $n - 2$, thus either $W_i \cap W_j = W_{il}$, in which case adding $W_i$ does not allow to recover $W_{il}$, or $W_i \subset (W_j, W_i)$. For the latter to fail, it is needed that all subspaces intersect in the same subspace $W_{il}$, which is not possible.

Example 1: The smallest such graph is $G_2(3,2)$. It is a complete graph with 7 vertices, given by

- $W_1 = (100,010)$,
- $W_2 = (100,001)$,
- $W_3 = (100,111)$,
- $W_4 = (010,001)$,
- $W_5 = (101,110)$,
- $W_6 = (010,101)$,
- $W_7 = (011,101)$.

If $W_1 = \{100,010,110\}$ fails, 100 may be repaired by contacting $W_2$ or $W_3$, 010 by contacting $W_4$ or $W_5$, and 110 by contacting $W_6$ or $W_7$. There are then $\frac{1}{2} \binom{7}{3} \binom{4}{2} = 12$ ways of doing this repair, while the maximum would be $\frac{1}{2} \binom{7}{2} = 21$ (for $d = 2$). This is true for each of the 7 nodes. The repair bandwidth reaches the minimum: two symbols downloaded to repair two, however a huge amount of storage is used: 14 symbols are stored, for a length 3 data object. This gives a storage overhead of $14/3 > 9/3$ which is the cost of 3-way replication.

It is possible to get a lesser storage overhead by reducing the length of the code, and take only 4 nodes, giving $8/3 < 9/3$. However then, 2 failures only can be tolerated.

Example 2: The graph $G_2(4,3)$ is a complete graph with 15 vertices.

- $W_1 = (1000,0100,0010)$,
- $W_2 = (1110,0001,1000)$,
- $W_3 = (1111,1000,1100)$,
- $W_4 = (0011,1010,1101)$,
- $W_5 = (1001,0101,1100)$,
- $W_6 = (0110,1011,0001)$,
- $W_7 = (0100,0110,0001)$,
- $W_8 = (1010,1101,0010)$,
- $W_9 = (1011,1110,0001)$,
- $W_{10} = (0110,0101,1101)$,
- $W_{11} = (1001,0110,1100)$,
- $W_{12} = (0100,0110,0001)$,
- $W_{13} = (0010,0011,1101)$,
- $W_{14} = (0001,1001,0010)$,
- $W_{15} = (1000,0100,0010)$,
- $W_{16} = (1101,1110,0010)$,
- $W_{17} = (1101,1111,0001)$.

The storage overhead of 3-way replication is $12/4 = 3$, thus we should keep at most 4 nodes to equate the amount of storage overhead, and 3 nodes to get less. This makes the length of the code too short, only two, respectively one failure(s) can then be tolerated.

This family of graphs clearly suffer from a terrible storage overhead of

$$\frac{(2^n - 1)m}{n}$$

if all the nodes are used. To number of nodes used should be (strictly) less than $3n/m$ to get a reasonable overhead, which in turn reduces significantly the number of failures tolerated. This overall behavior is likely to be caused by the fact that these subspaces share too big an intersection, though this in turn results in a minimum repair bandwidth.

B. The Graph $G_2(2m-1,m)$

Consider a clique of the graph $G_2(2m-1,m)$, such that every pair of subspaces intersect in a subspace of dimension 1. Then

$$2m - \dim(W \cap W') = n = 2m - 1$$

which shows that the object may be recovered from any choice of $k = 2$ nodes. When $m = 2$, we get the graph $G_2(3,2)$ already considered above. When $m = 3$, this is the graph $G_2(5,3)$.

Example 3: Consider the following (non-maximal) clique of $G_2(5,3)$, computed using cliquer [9]:

- $W_1 = (10001,01101,00010)$,
- $W_2 = (10000,01001,00001)$,
- $W_3 = (11000,00110,00001)$,
- $W_4 = (10101,01101,00001)$,
- $W_5 = (10001,01101,00010)$,
- $W_6 = (01010,00110,00001)$,
- $W_7 = (01001,01101,00001)$,
- $W_8 = (01010,00110,00001)$.

It has the property that every pair of subspaces intersects in a subspace of dimension 1, and that every triple of subspaces has trivial intersection. Since $k = 2$ (the object is retrievable from any choice of 2 live nodes), we consider the repair of one failure. Suppose for example that $W_1 = \{10001,01101,00010,11001,00111,11110\}$ fails. These vectors are available across the network as shown in Table I. To repair $W_1$, any two nodes may be contacted. Since the intersection of any 3 nodes is trivial, this will give necessarily two distinct vectors, which generate a subspace
of dimension 2. Now the third vector can be anything, as long as it does not belong to the span of the vectors already obtained. Thus the number of ways of repairing $W_1$ is $\frac{1}{6} \begin{pmatrix} 5 \\ 3 \\ 2 \\ 1 \\ 4 \\ 6 \\ 4 \\ 1, 4, 6 \\ 2, 3, 6 \end{pmatrix} = 28 < 35 = \begin{pmatrix} 5 \\ 3 \end{pmatrix}$ (for $d = 3$).

The storage overhead of 3-way replication is $15/5 = 3$ so we may keep up to 5 nodes, which is slightly better than the code construction of Example 2.

C. The Graph $G_2(6,3)$

Consider the graph $G_2(6,3)$, and the following (non-maximal) clique, computed using cliquer [9]:

\[
\begin{align*}
W_1 &= \{100010, 010100, 001110\}, \\
W_2 &= \{010001, 001000, 000101\}, \\
W_3 &= \{010101, 011101, 000011\}, \\
W_4 &= \{000000, 010011, 001101\}, \\
W_5 &= \{001010, 000100, 000001\}, \\
W_6 &= \{110110, 001100, 000001\},
\end{align*}
\]

Every pair of subspaces intersects in a subspace of dimension 1. By (4), for any $W, W', W''$

\[
\dim(W + W' + W'') = 9 - 1 - \dim(W \cap (W' + W''))
\]

and for this particular clique

\[
\dim(W + W' + W'') = 6
\]

which shows that any choice of 3 subspaces allows a data collector to retrieve the object. Some triples have an intersection of dimension 1, as summarized in Table II. Suppose the node $W_1$ fails. There are 4 repair options: (2,3,4), (2,3,6), (2,4,5) and (2,5,6), since 3,5 cannot be in a triple together, and 4,6 cannot either.

If two nodes fail, say $W_1, W_2$, then the node that repairs $W_1$ may get 001110 from $W_5$, 010100 from $W_2$ and 110110 from either $W_4$ or $W_6$. Then the node that repairs $W_2$ may get 001101 from $W_4$, 100110 from $W_4$ and either 001110 from $W_5$ or 001101 from $W_6$. So each has two repair options. A collaborative repair could also be done: once one node gets 001110, it may give it directly to the other repair node. The storage overhead is 18/6 which is the same as 3-way replication.

IV. CONCLUSION

In this paper, we abstracted codes for distributed storage systems in terms of subspaces and their intersection. This suggested the design of subspaces with regular intersection, and we started with pairwise intersection. To find such subspaces, we computed cliques from Grassmannian graphs, to obtain families of subspaces whose pairwise distance has a given dimension, and studied the obtained parameters in terms of storage codes.

The choice of pairwise intersection is also natural, since it is related to the design of constant dimension codes for network coding [7]. However, though the examples that we found have some potential for storage applications, the requirement of pairwise intersection seems less critical than for network coding. There are obvious continuations of this preliminary study:

1) Find a theoretical characterization of (collaborative) repair in terms of subspace intersection.
2) Find more systematic constructions of such codes, to get instances with interesting parameters for storage applications.
3) Move from pairwise intersection to other types of intersection patterns.

ACKNOWLEDGMENT

This work is supported in part by a grant from the company Xvid, and by the MoE Tier-2 grant "eCODE: Erasure Codes for Datacenter Environments".

REFERENCES


