Constructing Double-Erasure HoVer codes Using Latin Squares*

Wang Gang, Liu Xiaoguang, Lin Sheng, Xie Guangjun, Liu Jing
Department of Computer, College of Information Technology Science, Nankai University
wgzwp@163.com

Abstract

Storage applications are in urgent need of multi-erasure codes. But there is no consensus on the best coding technique. Hafner has presented a class of multi-erasure codes named HoVer codes [1]. This kind of codes has a unique data/parity layout which provides a range of implementation options that cover a large portion of the performance/efficiency trade-off space. Thus it can be applied to many scenarios by simple tuning. In this paper, we give a combinatorial representation of a family of double-erasure HoVer codes - create a mapping between this family of codes and Latin squares. We also present two families of double-erasure HoVer codes respectively based on the column-Hamiltonian Latin squares (of odd order) and a family of Latin squares of even order. Compared with the double-erasure HoVer codes presented in [1], the new codes enable greater flexibility in performance and efficiency trade-off.

1. Introduction

In recent years, as hard disks have grown greatly in size and storage systems have grown in size and complexity, it is more frequent that a failure of one disk occurs in tandem with unrecovered failures of other disks or latent failures of blocks on other disks. On a system using single-erasure correcting code such as standard RAID-5, this combination of failures leads to a permanent data loss [2]. Hence, applications of multi-erasure correcting codes have become more pervasive. But all of known multi-erasure coding techniques have limitations. HoVer codes provide a wide range of choices in performance/efficiency trade-off space with a unique data/parity layout [1]. In this paper, we present a combinatorial representation of a family of 2-erasure HoVer codes presented in [1]. We also develop two families of 2-erasure HoVer codes that enable greater flexibility in efficiency/performance trade-off than the old codes.

The outline of this paper is as follows. In Section 2 we discuss related works. Section 3 is devoted to representing the combinatorial representation of HoVer codes. The new codes are given in Section 4. Theoretical analysis is given in Section 5. In Section 6, we summarize the paper and discuss some possible future research directions.

2. Current multi-erasure correcting codes

The known multi-erasure codes typically fall into one of three categories: Reed-Solomon codes, binary linear codes and array codes.

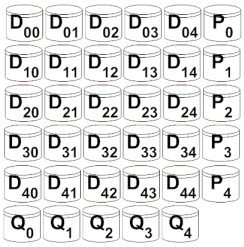
RS codes [3] are the only known MDS codes for arbitrary size and fault tolerance. This means that the optimal storage efficiency and the optimal update penalty are achieved. But as computation over Galois Field is used, the computational complexity is a serious problem though optimization has been presented [4].

Binary linear codes [5] are XOR-based, hence have perfect computational complexity, but bad storage efficiency is their inherent drawback. Fig 1.a shows a 2d-parity code [5], where D_{ij} denotes a data symbol that participates in parity symbols P_i and Q_j. This example illustrates the key idea of linear codes - divide data symbols into several overlapping parity groups.

An array code arranges the data/parity symbols into an array, hence the name. EVENODD [6] is an earlier MDS array code. It is a parity independent 2-erasure horizontal code. "*Dependent*" and "*independent*" are two opposite concepts that tell whether parity symbols play roles as data members of other parity groups. "*Horizontal*" and "*vertical*" classify codes by whether data and parity symbols are stored separately or together. Fig 1.b shows the 7-disk EVENODD code. D_{i*} denotes a data symbol that participates in P_i and all Qs. This code can be constructed by deleting P_4, Q_4

* This paper is supported partly by the National High Technology Research and Development Program of China (2008AA01Z401), NSFC of China (90612001), RFDP of China (20070055054), and Science and Technology Development Plan of Tianjin (08JCYBJC13000)

and D_{40}~D_{44} from the 2d-parity code on the left and then packing the remaining symbols into 7 disks. D_{i*} is D_{i4} actually. In fact, almost all of array codes can be regarded as layouts of binary linear codes. EVENODD and its generalization [7] may be the most important array codes because many array codes, such as X-code [8], RDP [2], STAR-code [9], etc, organize the parity groups in a similar way to them.



a. A 35-disks 2d-parity code

disk0	disk1	disk2	disk3	disk4	disk5	disk6
D_{00}	D_{01}	D_{02}	D_{03}	D_{0*}	P_0	Q_0
D_{11}	D_{12}	D_{13}	D_{1*}	D_{10}	P_1	Q_1
D_{22}	D_{23}	D_{2*}	D_{20}	D_{21}	P_2	Q_2
D_{33}	D_{3*}	D_{30}	D_{31}	D_{32}	P_3	Q_3

b. The 7-disk EVENODD code.

Fig 1. **Current multi-erasure codes.**

These codes have bad parameter flexibility - they all require a prime related size. Although horizontal shortening (deleting some data disks - assuming they contain nothing but zeros) can alleviate this problem, this method is harmful to performance [10, 11]. B-Code [12], BG-HEDP [13], PIHLatin code [10] and PDHLatin [11] have no prime size limitation because they are based on perfect one-factorizations of graphs (P1F) or column-hamiltonian latin squares (CHLS).

The traditional array codes pursue the MDS or near-MDS property. This leads to perfect storage efficiency but bad degraded-/reconstruction- mode performance especially for distributed storage systems because recovering lost data induces more interaction between the surviving hard disks (more network traffic between the surviving storage nodes). WEAVER code [14] is a counter traditional code. Its best storage efficiency is only 50%! But its parity groups are very short, hence good degraded-/reconstruction- mode performance is achieved. WEAVER code also has good parameter flexibility.

Vertical shortening (deleting some rows) is another common code transformation. The result codes perform comparably to WEAVER codes. This method isn't quite fit for horizontal codes. A $\text{HoVer}_{v,h}^{t}[r,n]$ code is a t-erasure parity independent mixed code consisting of h dedicated parity disks in which each stores r parity symbols (h-*parity*), and n mixed disks in which each stores r data symbols and v parity symbols (v-*parity*). Fig 2.b shows an array code that conforms to the structure of $\text{HoVer}_{1,1}^{2}[3,3]$. If h=1 and the parity disk stores horizontal parity symbols, vertically shortening a $\text{HoVer}_{v,1}^{t}[r,n]$ code produces a regular $\text{HoVer}_{v,1}^{t}[r',n]$ where $r' < r$. That is to say, HoVer codes cover whole efficiency/performance trade-off space with a unique and neat data/parity layout.

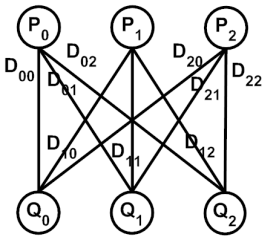
3. Combinatorial representation

3.1. Some related combinatorics knowledge

Some literature refers to simple graph representation of parity independent 2-erasure linear codes which satisfy that each data symbol participates in exactly two parity groups [5][12][15]: each vertex denote a parity symbol (group) and each edge denote a data symbol - the two endpoints of an edge is just the two parity symbols of the data symbol. Then an array code can be described by a graph partition if its underlying linear code can be described by a simple graph. We have proven the following theorem [15]:

Theorem 1. If an array code can be described by a partition of a simple graph, it is a 2-erasure code iff the union of any pair of subgraphs of the partition doesn't contain the following two types of structures:

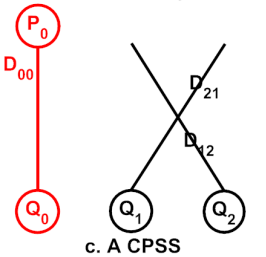
1. A path and its two endpoints. We call this kind of unrecoverable erasure ***C**losed **P**arity **S**ymbols **S**ubset*, CPSS for short.
2. A cycle. We call it CDSS - ***C**losed **D**ata **S**ymbols **S**ubset*.



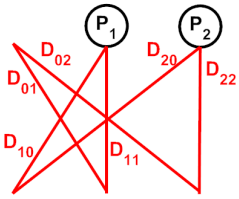
a. The graph of a 15-disk 2d-parity code

disk0	disk1	disk2	disk3
D_{00}	D_{01}	D_{02}	Q_0
D_{12}	D_{10}	D_{11}	Q_1
D_{21}	D_{22}	D_{20}	Q_2
P_0	P_1	P_2	

b. An array code



c. A CPSS



d. A CDSS

Fig 2. **Graph representation.**

Fig 2.a shows the graph of a 15-disk 2d-parity code. The code shown in Fig 2.b is based on this 2d-parity code. Fig 2.c shows a CPSS and Fig 2.d shows a CDSS. They correspond to unrecoverable 2-erasures (disk0, disk3) and (disk1, disk2) of the array code on the left respectively. Namely this code is not a real 2-erasure HoVer code. Theorem 1 can interpret most 2-erasure array codes including $\text{HoVer}_{1,1}^{2}[r,n]$.

Theorem 1 doesn't suggest construction method of 2-erasure array codes. Perfect one-factorization of

graph [16] is a good tool to do this. A *one-factor* of a graph G is a set of edges in which every vertex appears exactly once. A *one-factorization* (1F) of G is a partition of the edge-set of G into one-factors. A *perfect one-factorization* (P1F) is a one-factorization in which every pair of distinct one-factors forms a Hamiltonian cycle. There is a widely believed conjecture in graph theory: every complete graph with an even number of vertices has a P1F [16].

For $k \le n$, a $k*n$ *Latin rectangle* is a $k*n$ matrix of entries chosen from some set of symbols of cardinality n, so that no symbol is duplicated within any row or any column. We use $Z_n=\{0, 1, \ldots, n\text{-}1\}$ as the symbol set. It also can be used as the row/column number set. When $k=n$, the Latin rectangles are called *Latin squares* of order n. The symbol in row r, column c of a Latin rectangle R is denoted by R_{rc}. A Latin square of order n can be described by a set of n^2 triples of the form (*row*, *column*, *symbol*).

Each row r of a Latin rectangle R is the image of some permutation σ_r of Z_n, namely $R_{ri}=\sigma_r(i)$. Each pair of rows (r; s) defines a permutation by $\sigma_{r,s}=\sigma_r\sigma_s^{-1}$. If $\sigma_{r,s}$ consists of a single cycle for each pair of rows (r, s) in a Latin square L, we say L is *row-hamiltonian*. Similar concepts can be defined in terms of the *column* and *symbol*. In this paper, we are concerned with column-hamiltonian Latin squares, CHLS for short.

There is a CHLS L of order n iff $K_{n,n}=(V,W,E)$ has a P1F $F=\{F_0, \ldots, F_{n\text{-}1}\}$ [16]. To show this, we create three one-to-one correspondence: between the row set and V, between the symbol set and W, and between the column set and F. Namely, (i, j, k) $(\in L)$ corresponds to the edge (v_i, w_k) in F_j. Obviously, the cycle pattern in $\sigma_{r,s}$ in L corresponds to that in $F_r \cup F_s$. There is another conclusion [16]: if K_{n+1} has a P1F, then so does $K_{n,n}$. Thus we have a conjecture: $K_{n,n}$ has a P1F (CHLS of order n exists) for n=2 and all odd positive integers n. Graph theorists have proven that all even(odd) numbers less than 54(53) are "K_n P1F numbers" (CHLS/$K_{n,n}$ P1F numbers) and have found many larger K_n P1F numbers (CHLS/$K_{n,n}$ P1F numbers).

3.2. Combinatorial representation of *s*-shift $\mathbf{HoVer_{1,1}^2[r,n]}$ codes

An 2-erasure array code can be described by a partition, a P1F is just a partition, and there is a bijection between CHLS and P1F of $K_{n,n}$. Thus a natural idea is constructing 2-erasure array codes by CHLS. In [10] and [11], we have tried this idea. Two algorithms are developed, one constructs EVENODD-like codes by CHLS, and the other constructs RDP-like codes by CHLS. We name the first kind of codes PIHLatin codes (Parity Independent Horizontal Latin codes), and the second kind PDHLatin codes (Parity Dependent Horizontal Latin codes). The key ideas of the two algorithms are similar: construct the j^{th} disk by the j^{th} column - construct the i^{th} data/parity symbol in the j^{th} disk by the symbol (i, j, k); one row is deleted to break the Hamiltonian cycles in the disk pairs - CDSS are avoided; finally, parity symbols are arranged properly to avoid CPSS. Therefore, 2-erasure correcting is guaranteed.

PIHLatin and PDHLatin are superior to EVENODD and RDP in parameter flexibility because the distribution of P1F numbers is far denser than that of prime numbers. The PIHLatin and PDHLatin codes constructed by Cayley tables of cyclic groups of order p are just the (p+2)-disk EVENODD code and the (p+1)-disk RDP code respectively when p is a prime (a Cayley tables of cyclic groups of order n C_n is a LS, and is a CHLS when n is a prime). This means that PIHLatin and PDHLatin codes are the supersets of EVENODD and RDP codes respectively. We have shown that the relationship is proper superset [10][11]. Besides parameter flexility, PIHLatin and PDHLatin codes have advantage in structure flexibility: maybe have more than one heterogeneous instance for a given size.

The above discussion provides a complete method for 2-erasure array codes: describing codes by graph partitions and constructing codes by clipped CHLS. This is also the technique used by B-Code, although B-Code is based on P1F of K_n instead of CHLS or P1F of $K_{n,n}$. The HoVer codes can be subsumed in the class of "Latin codes" too. Hafner presented a family of $\text{HoVer}_{1,1}^2$[r,n] codes named s-shift $\text{HoVer}_{1,1}^2$[r,n] codes in [1]. We can interpret this kind of codes by Cayley tables of cyclic groups. Before discussing the combinatorial representation, we give some important properties of Cayley tables of cyclic groups that we presented in [11].

A Cayley table of the cyclic group of order n - C_n can be described by Formula 1, where $<x>_m$ denotes x mod m. It is not hard to verify that Cayley tables of the cyclic groups are Latin squares. C_n corresponds to a 1F of $K_{p,p}$ $F=\{F_0, F_1,\ldots, F_{n\text{-}1}\}$, $F_j=\{(v_i, w_{<i+j>n}), 0\le i\le n\text{-}1\}$, $0\le j\le n\text{-}1$.

$$(i, j, <i+j>_n) \in C_n, \quad 0 \le i, j \le n-1 \qquad (1)$$

Property 1. $\sigma_{j,k}$ of C_n $(0\le j<k\le n\text{-}1)$ consists of $gcd(n, d)$ cycles of length $2l$ (l=n/$gcd(n,d)$). The i^{th} cycle is $(i, j, <i+j>_n)$ -- $(i, k, <i+k>_n)$ -- $(<i+k\text{-}j>_n, j, <i+k>_n)$ -- $(<i+k\text{-}j>_n, k, <i+2k\text{-}j>_n)$ -- $(<i+2k\text{-}2j>_n, j, <i+2k\text{-}j>_n)$ -- .. .-- $(<i+(l\text{-}1)k\text{-}(l\text{-}1)j>_n, k, <i+lk\text{-}(l\text{-}1)j>_n)$ -- $(<i+lk\text{-}lj>_n, j, <i+lk\text{-}(l\text{-}1)j>_n)$.

Where d=k-j, and $gcd(n, d)$ denotes the greater

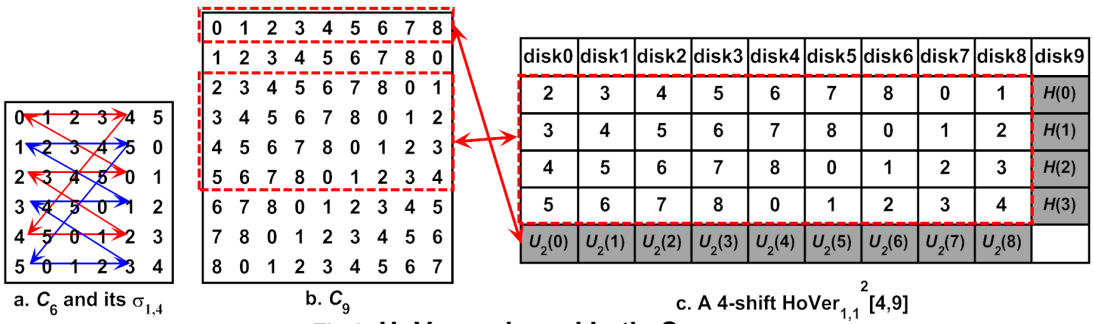


Fig 3. **HoVer code and Latin Squares.**

common divisor of n and d. According to formula 1, the i^{th} chain in $\sigma_{r,s}$ is as property 1 describes. Since l=n/gcd(n, k-j), we know l(k-j) is the lowest common multiple of n and k-j. Thus the only duplicated symbols in the chain are two endpoints (i, j, $<i+j>_n$) and ($<i+lk-lj>_n$, j, $<i+lk-(l-1)j>_n$). Moreover, different chains contain no common symbols. Therefore $\sigma_{j,k}$ consists of gcd(n, d) cycles of length 2l. Fig 3.a shows $\sigma_{0,4}$ of C_6. It has 2 3-long cycles because gcd(4, 6)=2. Let pr(n) be the smallest prime divisor of n. We can see that $\sigma_{j,k}$ induces a Hamiltonian cycle when n and d are relatively prime. The maximum number of cycles is n/pr(n) (of length pr(n) each) when d=n/pr(n) and the maximum cycle length is also n/pr(n) (pr(n) cycles) when d=pr(n). This conclusion deduces directly the well known fact: C_n is a CHLS when n is a prime number.

Property 2. Given C_n, for any $0\le i\le n$-1, and any $0\le r<s\le n$-1, rows i, $<i+1>_n$, ..., $<i+n/\mathrm{pr}(n)-1>_n$ intersect all cycle of $\sigma_{r,s}$. And there exist $0\le r<s\le n$-1 and $0\le k\le gcd(n,s-r)$-1, rows i, $<i+1>_n$, ..., $<i+n/\mathrm{pr}(n)-2>_n$ don't intersect the k^{th} cycle in $\sigma_{r,s}$.

Examining cycle patterns of any $\sigma_{r,s}$ we can conclude this property. This property can deduce the following property.

Property 3. Deleting any k (n/pr(n)$\le k\le n$-1) consecutive rows (with wrap-around) from C_n, we get a Latin rectangle in which $\sigma_{r,s}$ contains no cycles for any $0\le r<s\le n$-1.

These properties can be converted into P1F version according to CHLS-P1F transformation described in Section 3.1.

The s-shift HoVer$_{1,1}^{2}$[r,n] codes can be described by:

$$U_s(j)=\bigoplus_{k=0}^{r-1}X(r-1-k,<j+k+s>_n) \qquad (2)$$

where s is the shift distance, $U_s(j)$ and $X(i,j)$ are the v-parity and the i^{th} data symbol on the j^{th} mixed disk respectively. Hafner presented an existence theorem for this kind of codes:

Theorem 2. The s-shift HoVer$_{1,1}^{2}$[r,n] codes defined by formula 2 has fault tolerance 2 iff

$$r\le\begin{cases}n-s & \text{if } s>n/\mathrm{pr}(n)\\ n-n/\mathrm{pr}(n)-s & \text{otherwise}\end{cases} \qquad (3)$$

Fig 3.b shows C_9, Fig 3.c shows the 4-shift HoVer$_{1,1}^{2}$[4,9] code. Every data symbol is described by the number of its v-parity. As the figure shows, C_n is divided vertically into four areas of height 1, (n-r-s), r and (s-1). The first row (in natural order) designates the distribution of the v-parity symbols. The next (n-r-s) rows are discarded. The next r rows are used to construct the r data rows of the code. The symbol (i, j, k) (n-r-s+1$\le i\le n$-s, $0\le j\le n$-1) points out that X(i-(n-r-s+1), j) participates in H(i-(n-r-s+1)) and $U_s(k)$. The last s-1 rows are deleted, too.

According to property 3, deleting n/pr(n) consecutive rows in C_n breaks all Hamiltonian cycles in the disk pairs. If we construct data symbols by the remaining symbols in the way discussed in the last paragraph, we will avoid CDSS. If we delete one more row and arrange the v-parity symbols by it, CPSS will be avoided too (this operation breaks the path induced by any pair of disks into two segments, and attaches one vertex to each). Then we produce a 2-erasure HoVer code. Deleting further rows increases performance and decreases storage efficiency.

But why does Hafner's construction method delete two inconsecutive parts? The reason is that this method pursues s-shift structure: the v-parity symbols are put into natural order, the lowest data row is constructed by shifting the v-parity row s places right circularly, and each of the other rows is just the 1-shift of its lower neighbor. Therefore C_n is split into four areas as mentioned above. The second and fourth areas are deleted. According to property 3, in order to avoid CDSS, at least one of the two areas is not shorter than n/pr(n) rows. Namely, (s-1)$\ge n$/pr(n) or (n-r-s)$\ge n$/pr(n) must be met. They are just the two cases of theorem 2. Hafner believes that a wide range of choices in shift distance benefits reconstruction performance. But there is no any further discussion in [1]. So we think it is not necessary to construct HoVer$_{1,1}^{2}$[r,n] codes following formula 2. It's better to construct a simple structure by deleting n/pr(n) consecutive rows from C_n and

designating another row as the v-parity row.

By the way, there is negligence in the proof of theorem 2 in [1]. When unrecoverable 2-erasures consisted of two mixed disks are examined, only "recovery dependency chains that the two lost v-parity symbols" are concerned. That is to say, only CPSS are taken into consideration, never CDSS. Moreover, the proof about CPSS doesn't take wrapping into account when examines the reconstruction dependency chains (chains in $\sigma_{r,s}$). Now we give *s*-shift HoVer$_{1,1}^2$[r,n] codes a neat description using the Cayley tables of the cyclic groups. We call the codes Cayley HoVer codes.

Algorithm 1. Cayley HoVer$_{1,1}^2$[r,n] codes Construction Algorithm

Input: The Cayley table of the cyclic group of order *n* - C_n. F_n=$\{F_0, F_1,\ldots, F_{n-1}\}$ is the 1F of $K_{n,n}$ = (V, W, E) corresponding to C_n. An integer $r \le n-n/\mathrm{pr}(n)-1$.

Output: A Cayley HoVer$_{1,1}^2$[r,n] code.

Method:

1. Delete any $n/\mathrm{pr}(n)$ consecutive rows from C_n with wrap-round. By rotational symmetry, the last $n/\mathrm{pr}(n)$ rows are deleted (all edges incident to $v_{n-n/\mathrm{pr}(n)}, \ldots, v_{n-1}$ are deleted from F_j for all j), then we get a $(n-n/\mathrm{pr}(n))*n$ Latin rectangle R (a partition $F'=\{F_0', F_1',\ldots, F_{n-1}'\}$ of $K_{n-n/\mathrm{pr}(n),n}$).
2. Delete arbitrary $n-n/\mathrm{pr}(n)-r$ rows from R, suppose rows $a_0, a_1, \ldots, a_{n-n/\mathrm{pr}(n)-r-1}$ are deleted (delete all edges incident to $v_{a_0},\ldots,v_{a_{n-n/pr(n)-r-1}}$ from F_j' for all j), then we get a $r*n$ Latin rectangle R' (a partition $F''=\{F_0'', F_1'',\ldots, F_{n-1}''\}$ of $K_{r,n}$). Select one row from these rows arbitrarily, suppose row a_0 is selected without loss of generality. Let the k^{th} v-parity symbol be stored in the j^{th} disk for all $(a_0, j, k) \in R'$ (add w_k into F_j'' for all $(v_{a_0}, w_k) \in F_j''$).
3. Let the i^{th} data symbol on the j^{th} data disk participate in the i^{th} h-parity group and the k^{th} v-parity group for all $(i, j, k) \in R'', 0 \le i \le r-1$, $0 \le j \le n-1$ ($(v_i, w_k) \in F_j''$).

Theorem 3. The Cayley HoVer$_{1,1}^2$[r,n] codes are 2-erasure correcting codes.

Proof: According to property 3, after step 1, $F_i' \cup F_j'$ consists of only acyclic paths for all $0 \le i < j \le n-1$ (so does $F_i'' \cup F_j''$ because it is a subgraph of $F_i' \cup F_j'$).

Step 2 deletes the edge incident to v_{a_0} from F_j' and add its other endpoint into for all $0 \le j \le n-1$. So the two edges incident to v_{a_0} are deleted from a path in $F_i' \cup F_j'$, and the other two endpoints of them are added into. Therefore every $F_i'' \cup F_j''$ contains several acyclic paths, and two of them have an endpoint attached. In other words, they contain neither CPSS nor CDSS.

Thus codes produced by algorithm 1 are 2-erasure correcting codes. Deleting further rows in step 2 apparently doesn't change fault tolerance. □

The more rows deleted in step 2, the worse storage efficiency and the better performance. So we can choose satisfactory efficiency/performance trade-off by adjusting how many rows are deleted. How to choose rows to be deleted in step 2? According to property 2, deleting consecutive rows with wrap-round touches more paths, thus breaks more paths into small paths which will improve the degree of concurrency of reconstruction. So deleting consecutive rows is a good strategy. Moreover, which row should be selected as "v-parity row"? We think it is really arbitrary, namely has no impact on performance.

4. New families of 2-erasure HoVer codes

4.1. A family of 2-erasure HoVer codes based on CHLS

When *n* is not a prime number, *s*-shift HoVer$_{1,1}^2$[r,n] codes provide a relatively narrow range of choices in performance/efficiency trade-off space. We can improve this deficiency using CHLS (P1F). We call the new codes CHLS HoVer$_{1,1}^2$[r,n] codes.

Algorithm 2. CHLS HoVer$_{1,1}^2$[r,n] codes Construction Algorithm

Input: L - A CHLS of order n. L is reduced (the first row is put into natural order). F_n=$\{F_0, F_1,\ldots, F_{n-1}\}$ is the 1F of $K_{n,n}$=(V, W, E) corresponding to L.

Output: A CHLS HoVer$_{1,1}^2$[r,n] code.

Method:

1. Delete the last row of L (delete (v_{n-1}, w_k) from F_j for all j), then we get a $(n-1)*n$ Latin rectangle R (a partition $F'=\{F_0', F_1',\ldots, F_{n-1}'\}$ of $K_{n-1,n}$).
2. Delete arbitrary $n-1-r$ rows from R, suppose rows $a_0, a_1, \ldots, a_{n-r-2}$ are deleted (all edges incident to $v_{a_0},\ldots,v_{a_{n-r-2}}$ are deleted from F_j' for all j), then we get a $r*n$ Latin rectangle R' (a partition $F''=\{F_0'', F_1'',\ldots, F_{n-1}''\}$ of $K_{r,n}$). Select one row from deleted rows arbitrarily, suppose row a_0 is selected without loss of generality. Let the k^{th} v-parity symbol be stored in the j^{th} disk for all $(a_0, j, k) \in R'$ (add w_k into F_j'' for all $(v_{a_0}, w_k) \in F_j''$).
3. Let the i^{th} data symbol on the j^{th} data disk participates in the i^{th} h-parity group and the k^{th}

0	1	2	3	4	5	6	7	8
1	3	0	2	7	8	5	6	4
2	0	6	4	1	3	7	8	5
3	8	5	6	0	7	4	1	2
4	2	7	0	8	6	1	5	3
5	7	8	1	6	4	2	3	0
6	5	4	7	3	2	8	0	1
7	4	3	8	5	1	0	2	6
8	6	1	5	2	0	3	4	7

a. A CHLS of order 9

disk0	disk1	disk2	disk3	disk4	disk5	disk6	disk7	disk8	disk9
1	3	0	2	7	8	5	6	4	*H*(0)
2	0	6	4	1	3	7	8	5	*H*(1)
3	8	5	6	0	7	4	1	2	*H*(2)
4	2	7	0	8	6	1	5	3	*H*(3)
5	7	8	1	6	4	2	3	0	*H*(4)
6	5	4	7	3	2	8	0	1	*H*(5)
7	4	3	8	5	1	0	2	6	*H*(6)
U(0)	*U*(1)	*U*(2)	*U*(3)	*U*(4)	*U*(5)	*U*(6)	*U*(7)	*U*(8)	

b. A CHLS HoVer$_{1,1}^{2}$[7,9]

Fig 4. **CHLS HoVer code.**

v-parity group for all $(i,j,k)\in R'', 0\le i\le r-1,$ $0\le j\le n-1$ ($(v_i,w_k)\in F_j''$).

Theorem 4. The CHLS HoVer$_{1,1}^{2}$[r,n] codes are 2-erasure correcting codes.

Proof: Because $F_i\cup F_j$ induces a Hamiltonian cycle for all $0\le i<j\le n-1$, after step 1, every $F_i'\cup F_j'$ consists of a path. The rest is similar to the proof of theorem 3. □

Fig 4.a shows a CHLS of order 9. Fig 4.b shows a CHLS HoVer$_{1,1}^{2}$[7,9] code based on this CHLS. The first row and the 9th row of the CHLS are deleted, and the first row is used to construct the v-parity row. This code contains 7data rows, while a Cayley HoVer$_{1,1}^{2}$[r,9] code contains at most 5 data rows. The advantage of CHLS HoVer$_{1,1}^{2}$[r,n] codes in wider performance space over Cayley HoVer$_{1,1}^{2}$[r,n] codes is obvious.

How to select rows to be deleted in step 2? The answer is dependent on the specific CHLS used. Different selections maybe lead to different performance. Concrete method needs further study.

4.2. A family of 2-erasure HoVer codes based on a family of LS of even order

CHLS HoVer$_{1,1}^{2}$[r,n] code widen the range of choices in performance/efficiency trade-off space when n is not a prime number but an odd number. Wanless has presented a family of LS [17] that can achieve the widest range of performance/efficiency trade-off space for some even sizes. A LS L of this kind is of order $p=2q$ where q is an odd prime. L is consisted of four blocks. Let $h=\{0, 2, ..., q-1\}$ and $H=\{q, q+1, ..., 2q-1\}$. The entry L_{ij} satisfies

$$L_{ij}=\begin{cases}(-2-i-j)_h & \text{if } i,j\in h,\\ (i+j)_H & \text{if } i\in h \text{ and } j\in H,\\ (i-j)_H & \text{if } i\in H \text{ and } j\in h,\\ (-i+j)_h & \text{if } i,j\in H.\end{cases} \quad (4)$$

$(x)_m$ denotes that symbol in a set S which is congruent to $x\%m$. For example, when $p=10$, $(5-1)_H=9$.

We can see that each of the four blocks of L is isotopic to the cyclic group of order q, thus any pair of columns chosen both from h or both from H will decompose into two cycles of length q.

Now, let's examine the column cycles between columns $c\in h$ and $d\in H$. Suppose we start in row $a\in h$ of column c at $(-2-a-c)_h$ (upper left block) and trace out the cycle from there. In column d of row a we find $(a+d)_H$ (upper right block). Next we should find in which row x of column c the symbol $(a+d)_H$ lies (lower left block). We have $(x-c)_H=(a+d)_H$, thus $x=(a+d+c)_H$. In this row of column d the symbol is $(-a-c)_h$ (lower right block). Now we return to row y of column c. We have $(-2-y-c)_h=(-a-c)_h$, thus $y=(a-2)_h$, two places to the up of where we started. Iterating this process and noting that p is odd, we see that in following the column cycle we will visit every other row before returning to row a. Thus column c and d induce a Hamiltonian cycle. Fig 5.a shows a LS of this kind of order 10 and its $\sigma_{1,7}$. Based on this kind of LS, we can construct HoVer$_{1,1}^{2}$[r,n] codes that cover wider range of performance/efficiency trade-off space than Cayley HoVer$_{1,1}^{2}$[r,n] codes. We call this kind of codes EVENLS HoVer$_{1,1}^{2}$[r,n] codes.

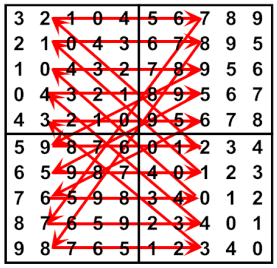
Algorithm 3. EVENLS HoVer$_{1,1}^{2}$[r,2p] codes Construction Algorithm

Input: L - A LS of order $2p$ constructed by formula 4. $F_{2p}=\{F_0, F_1,\ldots, F_{2p-1}\}$ is the 1F of $K_{2p,\,2p}=(V, W, E)$ corresponding to L.

Output: A EVENLS HoVer$_{1,1}^{2}$[r,2p] code.

Method:

1. Delete the first row and the p^{th} row of L (delete (v_0, w_k) and (v_p, w_k) from F_j for all j), then we get a $(2p-2)*2p$ Latin rectangle R (a partition $F'=\{F_0', F_1',\ldots, F_{2p-1}'\}$ of $K_{2p-1,2p}$).
2. Delete arbitrary $2p-1-r$ rows from R, suppose rows $a_0, a_1, ..., a_{2p-r-3}$ are deleted (all edges incident to $v_{a_0},\ldots,v_{a_{2p-r-3}}$ are deleted from F_j' for all j), then we get a $r*2p$ Latin rectangle R' (a partition $F''=\{F_0'', F_1'',\ldots, F_{2p-1}''\}$ of $K_{r,2p}$). Select one row from deleted rows arbitrarily, suppose row a_0 is selected without loss of generality. Let the k^{th} v-parity symbol be stored in the j^{th} disk for all $(a_0,j,k)\in R'$ (add w_k into F_j'' for all $(v_{a_0},w_k)\in F_j''$).



a. A LS of order 10

disk0	disk1	disk2	disk3	disk4	disk5	disk6	disk7	disk8	disk9	disk10
1	0	4	3	2	7	8	9	5	6	*H*(0)
0	4	3	2	1	8	9	5	6	7	*H*(1)
4	3	2	1	0	9	5	6	7	8	*H*(2)
6	5	9	8	7	4	0	1	2	3	*H*(3)
7	6	5	9	8	3	4	0	1	2	*H*(4)
8	7	6	5	9	2	3	4	0	1	*H*(5)
9	8	7	6	5	1	2	3	4	0	*H*(6)
U(2)	*U*(1)	*U*(0)	*U*(4)	*U*(3)	*U*(6)	*U*(7)	*U*(8)	*U*(9)	*U*(5)	

b. A EVENLS HoVer$_{1,1}^{2}$[7,10]

Fig 5. **EVENLS HoVer code.**

3. Let the i^{th} data symbol on the j^{th} data disk participates in the i^{th} h-parity group and the k^{th} v-parity group for all $(i,j,k)\in R'', 0\le i\le r-2$, $0\le j\le 2p-1$ ($(v_i,w_k)\in F_j''$).

Theorem 5. The EVENLS HoVer$_{1,1}^{2}$[r,n] codes are 2-erasure correcting codes.

Proof: $F_i\cup F_j$ induces a Hamiltonian cycle for all $i\in h$ and $j\in H$, and two cycles for all $i,j\in h$ or $i,j\in H$. Thus after step 1, every $F_i'\cup F_j'$ consists of one or two paths. The rest is similar to the proofs of theorem 3 and theorem 4. □

Fig 5.b shows an EVENLS HoVer$_{1,1}^{2}$[7,10] code based on the LS shown in Fig 5.a. The first row, the second row and the 6th row are deleted, and the second row is used to construct the v-parity row. This code has 7 data rows, while the Cayley HoVer$_{1,1}^{2}$[r,10] code has at most 4 data rows. EVENLS HoVer$_{1,1}^{2}$[r,n] codes fill the blank that CHLS HoVer$_{1,1}^{2}$[r,n] codes can't deal with. These two kinds of codes cooperatively cover most areas of performance/efficiency trade-off space.

5. Performance analysis

CHLS/EVENLS HoVer$_{1,1}^{2}$[r,n] codes have all advantages of other families of HoVer codes, such as optimal small write IO costs, near-optimal encoding/decoding/updating performance, good reconstruction performance, extended fault tolerance, and so on. Moreover, compared with *s*-shift HoVer$_{1,1}^{2}$[r,n] codes, the new codes cover wider performance space and have better structure variety.

When n is an odd number, CHLS HoVer$_{1,1}^{2}$[r,n] codes cover the widest range of r - from 1 to n-2. While *s*-shift HoVer$_{1,1}^{2}$[r,n] codes only cover the range from 1 to n-n/pr(n)-1. When p is a prime number, EVENLS HoVer$_{1,1}^{2}$[r,2p] codes cover the widest range of r - from 1 to 2p-3. While *s*-shift HoVer$_{1,1}^{2}$[r,2p] codes only cover the range from 1 to p-1. Obviously, CHLS/EVENLS HoVer codes provide more options in performance/efficiency trade-off.

We use EFF_L for the best packed efficiency [1] of CHLS/EVENLS HoVer$_{1,1}^{2}$[r,n] codes and EFF_S for that of *s*-shift HoVer$_{1,1}^{2}$[r,n] codes. We have the following formulas.

$$EFF_S=\frac{(n-n/\mathrm{pr}(n)-1)\times n}{(n-n/\mathrm{pr}(n))\times(n+1)-1}$$

$$EFF_L=\begin{cases}\dfrac{(n-2)\times n}{(n-1)\times(n+1)-1}, & n \text{ is odd}\\ \dfrac{(n-3)\times n}{(n-2)\times(n+1)-1}, & \dfrac{n}{2} \text{ is prime}\end{cases} \quad (5)$$

Fig 6 shows the ratio of the best efficiency of CHLS/EVENLS HoVer$_{1,1}^{2}$[r,n] codes and *s*-shift HoVer$_{1,1}^{2}$[r,n] codes to the optimal efficiency, where EFF_L, EFF_S and EFF_O are efficiencies of CHLS/EVENLS HoVer$_{1,1}^{2}$[r,n] codes, *s*-shift HoVer$_{1,1}^{2}$[r,n] codes and MDS codes respectively. It is easy to see, Latin HoVer$_{1,1}^{2}$[r,n] codes outperform *s*-shift HoVer$_{1,1}^{2}$[r,n] codes and are very close to MDS codes.

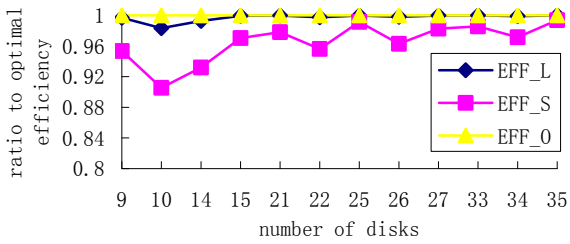


Fig 6. **Storage efficiency.**

6. Conclusion

HoVer codes have a unique data/parity layout which provides a range of implementation options that cover a large portion of the performance/efficiency trade-off space. In this paper, we give a combinatorial representation of *s*-shift HoVer$_{1,1}^{2}$[r,n] codes based on Latin squares. We gave a clean and correct description of this kind of codes. Then we presented a family of HoVer$_{1,1}^{2}$[r,n] codes based on column-hamiltonian Latin squares. We call them CHLS HoVer$_{1,1}^{2}$[r,n] codes. Another family of HoVer$_{1,1}^{2}$[r,n] codes based on a family of Latin Squares of even order is also

presented. We named them EVENLS $HoVer_{1,1}^{2}$[r,n] codes. The first new kind of codes cover the widest range of choices in performance/ efficiency trade-off space when n is an odd number, and the second new kind of codes cover the widest range when $n/2$ is a prime number. The two new kinds of codes cover most of the performance space. The new codes also are superior in structure variety.

CHLS/EVENLS HoVer codes solve odd numbers and the numbers that are twice of prime numbers. Next we plan to design HoVer codes for other even numbers. Hafner presented some families of 3-erasure and 4-erasure HoVer codes [1]. One future work is to study constructing HoVer codes of fault tolerance > 2 using Latin squares. High efficiency HoVer codes (small r) provide an extended fault tolerance. Studying the fault tolerance of HoVer codes in detail by both theoretical analysis and simulation is another significant research direction. Moreover, will deleting different rows in algorithm 3/4/5 lead to different performance? Will different structures of LS/CHLS induce different performance? These are also interesting problems. Implementing these codes in a real system and studying the real performance are also planned.

Acknowledgement

Many thanks to Dr. Ian M. Wanless for his kind help regarding the knowledge of Latin squares!

References

[1] J. L. Hafner, "HoVer Erasure Codes For Disk Arrays," International Conference on Dependable Systems and Networks, Philadelphia, PA, USA, Jun, 2006, pp. 217-226.

[2] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong and S. Sankar, "Row-Diagonal Parity for Double Disk Failure Correction," In Proceedings of the 3th USENIX Conference on File and Storage Technologies, San Francisco, CA, USA, Mar, 2004, pp.1-14.

[3] J. S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems," Software - Practice & Experience 27(9), pp. 995-1012, Sep, 1997.

[4] J. S. Plank and Lihao Xu, "Optimizing Cauchy Reed-Solomon Codes for Fault-Tolerant Network Storage Applications", In Proceedings of the 5th IEEE International Symposium on Network Computing and Applications, Cambridge, MA, Jul, 2006, pp.173-180.

[5] Lisa Hellerstein, Garth A. Gibson, Richard M. Karp, Randy H. Katz and David A. Patterson, "Coding techniques for handling failures in large disk arrays," Algorithmica 12(2/3), pp.182-208, Aug, 1994.

[6] M. Blaum, J. Brady, J. Bruck, J. Menon, "EVENODD: an efficient scheme for tolerating double disk failures in RAID architectures," IEEE Trans. on Computers 44(2), pp. 192-202, Feb, 1995.

[7] M. Blaum, J. Bruck, and A. Vardy, "MDS array codes with independent parity symbols," IEEE Trans. on Information Theory 42(2), pp. 529-542, Mar, 1996.

[8] L. Xu and J. Bruck, "X-Code: MDS Array Codes with Optimal Encoding," IEEE Trans. on Information Theory 45(1), pp.272-276, Jan, 1999.

[9] C. Huang, L. Xu, "STAR: An Efficient Coding Scheme for Correcting Triple Storage Node Failures," In Proceedings of the 4th USENIX Conference on File and Storage Technologies, San Francisco, Dec, 2005, pp.197-210.

[10] Gang Wang, Sheng Lin, Xiaoguang Liu, Guangjun Xie, Jing Liu, "Combinatorial Constructions of Multi-Erasure-Correcting Codes with Independent Parity Symbols for Storage Systems," In Proceedings of the 13th IEEE Pacific Rim Dependable Computing conference, Melbourne, Victoria, Austrilia, Dec, 2007, pp. 61-68.

[11] Wang Gang, Liu Xiaoguang, Lin Sheng, Xie Guangjun, Liu Jing, "Generalizing RDP Codes Using the Combinatorial Method," In NCA-08: 7th IEEE International Symposium on Network Computing Applications, Cambridge, MA, USA, July, 2008, pp.93-100.

[12] L. Xu, V. Bohossian, J. Bruck, and D.G. Wagner, "Low-Density MDS Codes and Factors of Complete Graphs," IEEE Trans. on Information Theory 45(6), pp.1817-1826, Sep, 1999.

[13] Wang Gang, Dong Sha-sha, Liu Xiao-guang, Lin Sheng, Liu Jing, "Construct double-erasure-correcting Data Layout Using P1F," ACTA ELECTRONICA SINICA, 34(12A), pp.2447-2450, Dec, 2006.

[14] J. L. Hafner, "WEAVER Codes: Highly Fault Tolerant Erasure Codes for Storage Systems," In Proceedings of the 4th USENIX Conference on File and Storage Technologies, San Francisco, Dec, 2005, pp.211-224.

[15] Zhou Jie,Wang Gang, Liu Xiaoguang, Liu Jing, "The Study of Graph Decompositions and Placement of Parity and Data to Tolerate Two Failures in Disk Arrays: Conditions and Existance," Chinese Journal of Computer 26(10), pp.1379-1386, Oct, 2003.

[16] I. M. Wanless, "Perfect factorisations of complete bipartite graphs and Latin squares without proper subrectangles", Electron. J. Combin, Vol. 6, 1999, R9.

[17] I. M. Wanless, "Cycle switches in Latin squares," Graphs and Combinatorics, 20(4), pp.545-570, Nov, 2004.