Multi-Cluster Interleaving on Linear Arrays and Rings

Anxiao (Andrew) Jiang and Jehoshua Bruck
California Institute of Technology
Electrical Engineering Department
MC 136-93
Pasadena, CA 91125, U.S.A.
E-mail: {jax,bruck}@paradise.caltech.edu

Abstract

Interleaving codewords is an important method not only for combatting burst-errors, but also for flexible data-retrieving. This paper defines the Multi-Cluster Interleaving (MCI) problem, an interleaving problem for parallel data-retrieving. The MCI problems on linear arrays and rings are studied. The following problem is solved: how to interleave integers on a linear array or ring such that any m ($m \geq 2$) non-overlapping segments of length 2 in the array or ring have at least 3 distinct integers. We then present a scheme using a 'hierarchical-chain structure' to solve the following more general problem for linear arrays: how to interleave integers on a linear array such that any m ($m \geq 2$) non-overlapping segments of length L ($L \geq 2$) in the array have at least $L+1$ distinct integers. It is shown that the scheme using the 'hierarchical-chain structure' solves the second interleaving problem for arrays that are asymptotically as long as the longest array on which an MCI exists, and clearly, for shorter arrays as well.

Index Terms

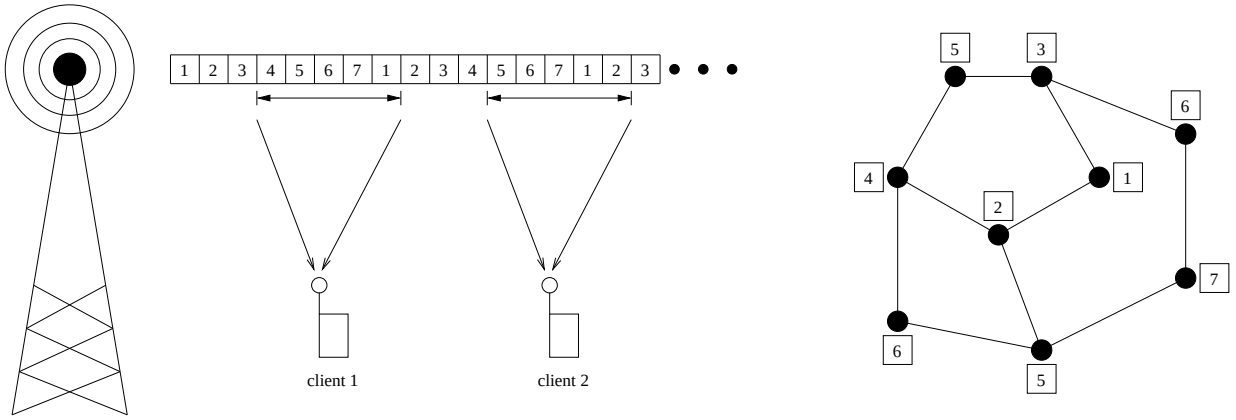
Array, cluster, error-correcting code, interleaving, multi-cluster interleaving, ring.

I. Introduction

Interleaving codewords is an important method for both data-retrieving and error-correction. Its application in error-correction is well-known. The most familiar example is the interleaving of codewords on a linear array, which has the form '$-1-2-3-\cdots-n-1-2-3-\cdots-n-$', for combatting one-dimensional burst-errors of length up to n. Other interesting examples include [1] [2] [3] [6] [7] [12], which are mainly for correcting burst-errors of different shapes on two- or three-dimensional arrays.

The applications of codeword interleaving in data-retrieving, although maybe less well-known, are just as broad. Data streaming and broadcast schemes using erasure-correcting codes have received extensive interest in both academia and industry, where interleaved components of a codeword are transmitted in sequence, and every client can listen to this data flow for a while until enough codeword components are received for recovering the information [4] [9]. (An example is shown in Fig. 1 (a), where a codeword of 7 components is broadcast repeatedly. We assume that the codeword can tolerate 2 erasures. Therefore every

[1]This work was supported in part by the Lee Center for Advanced Networking at the California Institute of Technology, and by NSF grant CCR-TC-0209042.



(a) Broadcast

(b) File storage in a network

Fig. 1. Examples of interleaving for data retrieving

client only needs to receive 5 different components.) Interleaving is also studied in the scenario of file storage in networks, where a file is encoded into a codeword, and components of the codeword are interleavingly placed on a network, such that every node in the network can retrieve enough distinct codeword components from its proximity for recovering the file [8] [10]. (An example is shown in Fig. 1 (b), where the codeword again has length 7 and can tolerate 2 erasures. We assume that all edges have length 1. Then every network node can retrieve 5 distinct codeword components from its proximity of radius 2 for recovering the file.) In all the above cases, the codeword components are interleaved on some graph structure. For example, in the data streaming and broadcast case, the codeword components can be seen as interleaved on a linear array, because they are sequentially transmitted along the time axis. (If the sequence of data are transmitted repeatedly — e.g., using a broadcast disk — then they can be seen as interleaved on a ring.) For the file storage case, the codeword components are interleaved on more general graphs. We see that a client (or a network node) usually retrieves data from a connected subgraph of the graph. We call every such connected subgraph a *cluster*.

By using interleaving, the above schemes all enable 'flexible' data-retrieving, in the sense that the information contained in the interleaved data can be recovered by accessing *any* reasonably large cluster. The data-retrieving performance can be further improved if multiple clusters can be accessed in parallel. Accessing data placed in different parts of a graph in parallel has the benefits of balancing load and reducing access time, and has been studied to some extent [5] [11]. Then, it is natural to ask the following question: what is the appropriate form of interleaving for parallel data-retrieving?

If it is required that for any m ($m \geq 2$) non-overlapping clusters, the interleaved codeword components on them are all distinct, then each codeword component can be placed only once on the graph, even if m is as small as 2. Such an interleaving scheme, although minimizing the total size of the clusters that each client needs to access, is not scalable because it requires the number of components in the codeword to equal the size of the graph, which would imply very high encoding/decoding complexity or even non-existence of the code if the graph is huge. So in an interleaving scheme for parallel data-retrieving, a tradeoff is needed between the scheme's scalability and the amount of overlapping among codeword components on different clusters.

In this paper, we only study interleaving for parallel data-retrieving on linear arrays and rings, which seems to have natural applications in data-streaming and broadcasting. Imagine that codeword components

interleaved the same way are transmitted asynchronously in several channels. Then a client can simultaneously listen to multiple channels in order to get data faster, which is equivalent to retrieving data from multiple clusters. Another possible application is data storage on disks, where we assume multiple heads can read different parts of a disk in parallel to accelerate I/O speed.

Now assume the codeword in consideration contains N components, and assume this codeword can be decoded by using any K ($K \leq N$) distinct codeword components. Assume every client can access m ($m \geq 2$) clusters in the linear array or ring in parallel, where a cluster is defined to be a connected subgraph of the linear array or ring containing L vertices. The only constraint on the m clusters a client can access is that those m clusters don't overlap each other. That constraint is for guaranteeing that each client can access no less than mL vertices. Then to enable clients to decode the codeword, there need to be at least K distinct codeword components placed on any m non-overlapping clusters. Therefore we define the following general interleaving problem for parallel data-retrieving:

Definition 1: Let $G = (V, E)$ be a linear array (or ring) of n vertices. Let N, K, m and L be positive integers such that $N \geq K > L$ and $m \geq 2$. A *cluster* is defined to be a connected subgraph of the array (or ring) containing L vertices. Assign one integer in the set $\{1, 2, \cdots, N\}$ to each vertex. Such an assignment is called a *Multi-Cluster Interleaving* (MCI) if and only if any m clusters that are non-overlapping are assigned no less than K distinct integers.

□

From the above definition, it can be seen that an MCI problem is fully characterized by the five parameters n, N, K, m, L and the graph $G = (V, E)$. For the ease of explanation later on, we note that throughout this paper, the parameters n, N, K, m, L and the graph $G = (V, E)$ will always have the meanings as defined in Definition 1.

Clearly if we let $m = 1$ in Definition 1 (and then let $K = L$), then it becomes the traditional interleaving. And if an interleaving on a linear array (or ring) is an MCI for some given value of m, then it is an MCI for larger values of m as well.

The following is an example of MCI.

Example 1: A ring $G = (V, E)$ of $n = 21$ vertices is shown in Fig. 2. The parameters are $N = 9$, $K = 5$, $m = 2$ and $L = 3$. An interleaving is shown in the figure, where the integer on every vertex is the integer assigned to it. It can be verified that any 2 clusters of size 3 that don't overlap have at least 5 distinct integers. For example, the two clusters in circle in Fig. 2 have integers '9, 1, 2' and '7, 1, 6' respectively, so they together have no less than 5 distinct integers. So the interleaving is a multi-cluster interleaving on the ring G.

If we remove an edge in the ring, then G will become a linear array. Clearly if all other parameters remain the same, the interleaving shown in Fig. 2 will be a multi-cluster interleaving on the array.

□

The general MCI problem can be divided into smaller problems according to the values of the parameters. The key results of this paper are:

- The family of problems with constraints that $L = 2$ and $K = 3$ are solved for both linear arrays and rings. We show that when $L = 2$ and $K = 3$, an MCI exists on a linear array if and only if the number

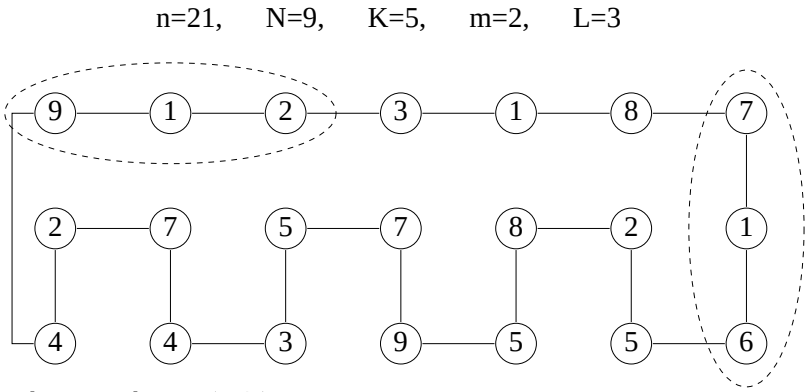


Fig. 2. An example of multi-cluster interleaving (MCI)

of vertices in the array is no greater than $(N-1)[(m-1)N-1]+2$, and an MCI exists on a ring if and only if the number of vertices in the ring is no greater than $(N-1)[(m-1)N-1]$. Structural properties of MCIs in this case are analyzed, and algorithms are presented which output MCIs on arrays or rings as long as the MCIs exist.

- The family of problems with the constraint that $K = L+1$ are studied for linear arrays. A scheme using a 'hierarchical-chain' structure is presented for constructing MCI on arrays. It is shown that the scheme solves the MCI problem for arrays that are asymptotically as long as the longest array on which MCIs exist, and clearly, for shorter arrays as well.

The rest of the paper is organized as follows. In Section II, we firstly derive an upper bound for the lengths of linear arrays and rings on which MCIs exist. We then prove a tighter upper bound for linear arrays for the case of $L = 2$ and $K = 3$. In Section III, we present an optimal construction for MCI on linear arrays for the case of $L = 2$ and $K = 3$, which meets the upper bound presented in Section II. In Section IV, we study the MCI problem for linear arrays when $K = L+1$. In Section V we generalize our results on MCI from linear arrays to rings. In Section VI, we conclude this paper.

II. Upper Bounds

While the traditional interleaving exists on infinitely long linear arrays, that is no longer true for MCI. The following proposition derives a very simple upper bound for the lengths of linear arrays and rings on which MCIs exist.

Proposition 1: If an MCI exists on a linear array (or ring) of n vertices, then $n \leq (m-1)L\binom{N}{L} + (L-1)$.

Proof: Let $G = (V, E)$ be a linear array (or ring) of n vertices with an MCI on it. We can find at most $\lfloor \frac{n}{L} \rfloor$ non-overlapping clusters in G. Let $S \subseteq \{1, 2, \cdots, N\}$ be an arbitrary set of L distinct integers. Then since the interleaving on G is an MCI, among those $\lfloor \frac{n}{L} \rfloor$ non-overlapping clusters, at most $m-1$ of them are assigned only (all or part of the) integers in S and no integers in $\{1, 2, \cdots, N\} - S$. S can be one of $\binom{N}{L}$ possible sets. So $\lfloor \frac{n}{L} \rfloor \leq (m-1)\binom{N}{L}$. So we get $n \leq (m-1)L\binom{N}{L} + (L-1)$.

□

We comment that for the same set of parameters N, K, m and L, if an MCI exists on a linear array of $n = n_0$ vertices, then an MCI also exists on any linear array of $n < n_0$ vertices. That is because given an MCI on a linear array, the interleaving on the array's connected subgraph (which is a shorter linear array) is

also an MCI for the same set of parameters N, K, m and L. However, such an argument does not necessarily hold for rings.

The upper bound of Proposition 1 is in fact loose. In the remainder of this section, we shall prove a tighter upper bound for MCI on linear arrays for the case of $L = 2$ and $K = 3$. We present it as the following theorem. Later study will show that this upper bound is exact.

Theorem 1: When $L = 2$ and $K = 3$, if there exists an MCI on a linear array of n vertices, then $n \leq (N-1)[(m-1)N-1]+2$.

Before proving the above theorem, we firstly define some notations that will be used throughout this paper as follows. Let $G = (V, E)$ be a linear array. We denote the n vertices in the linear array G by v_1, v_2, $\cdots$, v_n. For $2 \leq i \leq n-1$, the two vertices adjacent to v_i are v_{i-1} and v_{i+1}. A connected subgraph of G induced by vertices $v_i, v_{i+1}, \cdots, v_j$ ($j \geq i$) is denoted by $(v_i, v_{i+1}, \cdots, v_j)$. If a set of integers are interleaved on G, then $c(v_i)$ denotes the integer assigned to vertex v_i. The integers assigned to a connected subgraph of G, $(v_i, v_{i+1}, \cdots, v_j)$, are denoted by $[c(v_i) - c(v_{i+1}) - \cdots - c(v_j)]$.

The following lemma reveals a certain relationship between the structure of MCIs and the lengths of linear arrays.

Lemma 1: Let the values of N, K, m and L be fixed, where $N \geq 4$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a linear array of n vertices. Then in any MCI on a linear array of n_{max} vertices, no two adjacent vertices are assigned the same integer.

Proof: Let $G = (V, E)$ be a linear array of n_{max} vertices. Assume that there is an MCI on G. We'll prove this lemma by showing that in this MCI, if two adjacent vertices of G are assigned the same integer, then there exists a longer array that also has an MCI, which would clearly contradict the definition of n_{max}.

Assume that in the MCI on G, there are two adjacent vertices that are assigned the same integer. Then without loss of generality (WLOG), one of the following four cases must be true (because we can always get one of the four cases by permuting the names of the integers and by reversing the indices of the vertices):

Case 1: There exist 4 consecutive vertices in G — v_i, v_{i+1}, v_{i+2}, v_{i+3} — such that $c(v_i) = 1$, $c(v_{i+1}) = c(v_{i+2}) = 2$, and $c(v_{i+3}) = 1$ or 3.

Case 2: There exist $x+2 \geq 5$ consecutive vertices in G — v_i, v_{i+1}, $\cdots$, v_{i+x}, v_{i+x+1} — such that $c(v_i) = 1$, $c(v_{i+1}) = c(v_{i+2}) = \cdots = c(v_{i+x}) = 2$, and $c(v_{i+x+1}) = 1$ or 3.

Case 3: $c(v_1) = c(v_2) = 1$, and $c(v_3) = 2$.

Case 4: $c(v_1) = c(v_2) = \cdots = c(v_x) = 1$ and $c(v_{x+1}) = 2$, where $x \geq 3$.

We analyze the four cases one by one.

Case 1: In this case, we insert a vertex v' between v_{i+1} and v_{i+2}, and get a new array of $n_{max}+1$ vertices. Call this new array H, and assign the integer '4' to v'. Consider any m non-overlapping clusters in H. If none of those m clusters contains v', then clearly they are also m non-overlapping clusters in the array G, and therefore have been assigned at least $K = 3$ distinct integers. If the m clusters contain all the three vertices v_{i+1}, v' and v_{i+2}, then either v_i or v_{i+3} is also contained in the m clusters because each cluster contains $L = 2$ vertices, and therefore the m clusters have been assigned at least $K = 3$ distinct integers: '1,2,4' or '2,3,4'. WLOG, the only remaining possibility is that one of the m clusters contains v_{i+1} and v'

while none of them contains v_{i+2}. Note that among the m clusters, the $m-1$ of them which don't contain v' are also $m-1$ clusters in the array G, and they together with (v_{i+1}, v_{i+2}) are m non-overlapping clusters in G and therefore are assigned at least $K=3$ distinct integers. Since $c(v_{i+1}) = c(v_{i+2})$, the original m clusters including (v_{i+1}, v') must also have been assigned at least $K=3$ distinct integers. Now we can conclude that the interleaving on H is also an MCI. But H's length is greater than n_{max}, which contradicts the definition of n_{max}.

Case 2: In this case, we insert a vertex v' between v_{i+1} and v_{i+2}, and insert a vertex v'' between v_{i+x-1} and v_{i+x}, and get a new array of $n_{max}+2$ vertices. Call this new array H, assign the integer '4' to v', and assign the integer '3' to v''. Consider any m non-overlapping clusters in H. If neither v' nor v'' is contained in the m clusters, then clearly those m clusters are also m non-overlapping clusters in the array G, and therefore are assigned at least $K=3$ distinct integers. If both v' and v'' are contained in the m clusters, then at least one vertex in the set $\{v_{i+1}, v_{i+2}, \cdots, v_{i+x-1}, v_{i+x}\}$ is also in the m clusters, and therefore the m clusters have at least these 3 integers: '2', '3' and '4'. WLOG, the only remaining possibility is that the m clusters contain v' but not v''. (Note that the cluster containing v' is assigned integers '2' and '4'.) When that possibility is true, if the m clusters contain v_{i+x+1}, then they are assigned at least 3 distinct integers — '1,2,4' or '2,3,4'; if the m clusters don't contain v_{i+x+1}, then they don't contain v_{i+x} either — then we divide the m clusters into two groups A and B, where A is the set of clusters none of which contains any vertex in $\{v', v_{i+2}, v_{i+3}, \cdots, v_{i+x-1}\}$, and B is the set of clusters none of which is in A. Say there are y clusters in B. Then, if the cluster containing v' also contains v_{i+1} (respectively, v_{i+2}), there exist a set C of y clusters in the array G that only contain vertices in $\{v_{i+1}, v_{i+2}, \cdots, v_{i+x-1}, v_{i+x}\}$ (respectively, $\{v_{i+2}, v_{i+3}, \cdots, v_{i+x-1}, v_{i+x}\}$), such that (in both cases) the m clusters in $A \cup C$ are non-overlapping in G. Those m clusters in $A \cup C$ are assigned at least $K=3$ distinct integers since the interleaving on G is an MCI; and they are assigned no more distinct integers than the original m clusters in $A \cup B$ are, because $c(v_{i+1}) = c(v_{i+2}) = \cdots = c(v_{i+x})$ and either v_{i+1} or v_{i+2} is in the same clusters as v'. So the m clusters in $A \cup B$ are assigned at least $K=3$ distinct integers. Now we can conclude that the interleaving on H is also an MCI. And that again contradicts the definition of n_{max}.

Case 3: In this case, we insert a vertex v' between v_1 and v_2, and assign the integer '3' to v'. The rest of the analysis is very similar to that for Case 1.

Case 4: In this case, we insert a vertex v' between v_1 and v_2, and insert a vertex v'' between v_{x-1} and v_x, assign the integer '3' to v', and assign the integer '2' to v''. The rest of the analysis is very similar to that for Case 2.

So a contradiction exists in all four cases. Therefore, this lemma is proved.

□

The next two lemmas derive upper bounds on the lengths of linear arrays, respectively for the case of $N \geq 4$ and $N=3$.

Lemma 2: Let the values of N, K, m and L be fixed, where $N \geq 4$, $K=3$, $m \geq 2$ and $L=2$. Let n_{max} denote the maximum value of n such that an MCI exists on a linear array of n vertices. Then $n_{max} \leq (N-1)[(m-1)N-1]+2$.

Proof: Let $G=(V,E)$ be a linear array of n_{max} vertices. Assume there is an MCI on G. By Lemma 1, no two adjacent vertices in G are assigned the same integer. We color the vertices in G with three colors —

'red', 'yellow' and 'green' — through the following three steps:

Step 1, for $2 \leq i \leq n_{max} - 1$, if $c(v_{i-1}) = c(v_{i+1})$, then color v_i with the 'red' color;

Step 2, for $2 \leq i \leq n_{max}$, color v_i with the 'yellow' color if v_i is not colored 'red' and there exists j such that these four conditions are satisfied: (1) $1 \leq j < i$, (2) v_j is not colored 'red', (3) $c(v_j) = c(v_i)$, (4) the vertices between v_j and v_i — that is, v_{j+1}, v_{j+2}, $\cdots$, v_{i-1} — are all colored 'red';

Step 3, for $1 \leq i \leq n_{max}$, if v_i is neither colored 'red' nor colored 'yellow', then color v_i with the 'green' color.

Clearly, each vertex of G is assigned exactly one of the three colors.

If we arbitrarily pick two different integers — say 'i' and 'j' — from the set $\{1, 2, \cdots, N\}$, then we get a *pair* $[i, j]$ (or $[j, i]$, equivalently). There are totally $\binom{N}{2}$ such un-ordered *pairs*. We partition those $\binom{N}{2}$ *pairs* into four groups 'A', 'B', 'C' and 'D' in the following way:

(1) A *pair* $[i, j]$ belongs to group A if and only if the following two conditions are satisfied: (i) at least one 'green' vertex is assigned the integer 'i' and at least one 'green' vertex is assigned the integer 'j', (ii) for any two 'green' vertices that are assigned integers 'i' and 'j' respectively, there is at least one 'green' vertex between them.

(2) A *pair* $[i, j]$ belongs to group B if and only if the following two conditions are satisfied: (i) at least one 'green' vertex is assigned the integer 'i' and at least one 'green' vertex is assigned the integer 'j', (ii) there exist two 'green' vertices that are assigned integers 'i' and 'j' respectively such that there is no 'green' vertex between them.

(3) A *pair* $[i, j]$ belongs to group C if and only if one of the following two conditions is satisfied: (i) at least one 'green' vertex is assigned the integer 'i' and no 'green' vertex is assigned the integer 'j', (ii) at least one 'green' vertex is assigned the integer 'j' and no 'green' vertex is assigned the integer 'i'.

(4) A *pair* $[i, j]$ belongs to group D if and only if no 'green' vertex is assigned the integer 'i' or 'j'.

E is the set of edges in $G = (V, E)$. For any $1 \leq i \neq j \leq N$, let $E(i, j) \subseteq E$ denote such a subset of edges of G: an edge of G is in $E(i, j)$ if and only if one endpoint of the edge is assigned the integer 'i' and the other endpoint of the edge is assigned the integer 'j'. Let $z(i, j)$ denote the number of edges in $E(i, j)$. Upper bounds for $z(i, j)$ are derived below.

For any *pair* $[i, j]$ in group A or group C, $z(i, j) \leq 2m - 2$. That's because otherwise there would exist m non-overlapping clusters in G — note that a cluster contains exactly one edge — each of which is assigned only integers 'i' and 'j', which would contradict the assumption that the interleaving on G is an MCI.

Now consider a *pair* $[i, j]$ in group B. $z(i, j) \leq 2m - 2$ for the same reason as in the previous case. Assume $z(i, j) = 2m - 2$. Then in order to avoid the existence of m non-overlapping clusters in G each of which is assigned only integers 'i' and 'j', the $z(i, j) = 2m - 2$ edges in $E(i, j)$ must be consecutive in the array G, which means, WLOG, that there are $2m - 1$ consecutive vertices v_{y+1}, v_{y+2}, $\cdots$, v_{y+2m-1} ($y \geq 0$) whose assigned integers are in the form of $[c(v_{y+1}) - c(v_{y+2}) - \cdots - c(v_{y+2m-1})] = [i - j - i - j - \cdots - i - j - i]$. According to the definition of 'group B', there exist a 'green' vertex v_{k_1} and a 'green' vertex v_{k_2}, such that v_{k_1} is assigned the integer 'i', v_{k_2} is assigned the integer 'j', and there is no 'green' vertex between them. Therefore every vertex between v_{k_1} and v_{k_2} is either 'red' or 'yellow'. By the way the vertices are colored, it's not difficult to see that either '$k_1 < k_2$ and v_{k_2-1} is assigned the integer $c(v_{k_1}) = i$' (let's call this 'case 1'), or '$k_2 < k_1$ and v_{k_1-1} is assigned the integer $c(v_{k_2}) = j$' (let's call this 'case 2'). If 'case 1' is true, then

since there is an edge between v_{k_2-1} (which is assigned the integer 'i') and v_{k_2} (which is assigned the integer 'j'), v_{k_2} is in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$. However, it's simple to see that every vertex in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$ that is assigned the integer 'j' must have the color 'red' — so v_{k_2} must be 'red' instead of 'green' — therefore a contradiction exists. Now if 'case 2' is true, since there is an edge between v_{k_1-1} (which is assigned the integer 'j') and v_{k_1} (which is assigned the integer 'i'), both v_{k_1-1} and v_{k_1} are in the set $\{v_{y+2}, v_{y+3}, \cdots, v_{y+2m-1}\}$. Then again since every vertex in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$ that is assigned the integer 'j' must have the color 'red', and since the color of v_{k_1} is 'green', it is simple to see that all the vertices in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{k_1-1}\}$ that is assigned the integer 'i' must have the color 'red'. Then since the color of v_{y+1} is 'red', the vertex v_y exists and it must have been assigned the integer 'j' — and that contradicts the statement that all the edges in $E(i, j)$ are in the subgraph $(v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1})$. Therefore a contradiction always exists when $z(i, j) = 2m - 2$. So $z(i, j) \neq 2m - 2$. So for any *pair* $[i, j]$ in group B, $z(i, j) \leq 2m - 3$.

Now consider a *pair* $[i, j]$ in group D. By the definition of 'group D', no 'green' vertex is assigned the integer 'i' or 'j'. Let $\{v_{k_1}, v_{k_2}, \cdots, v_{k_t}\}$ denote the set of vertices that are assigned the integer 'i', where $k_1 < k_2 < \cdots < k_t$. If $\{v_{k_1}, v_{k_2}, \cdots, v_{k_t}\} \neq \emptyset$, by the way vertices are colored, it's simple to see that v_{k_1} cannot be 'yellow' — so v_{k_1} must be 'red'. Then similarly, v_{k_2}, v_{k_3}, $\cdots$, v_{k_t} must be 'red', too. Therefore all the vertices that are assigned the integer 'i' are of the color 'red'. Similarly, all the vertices that are assigned the integer 'j' are of the color 'red'. Assume there is an edge whose two endpoints are assigned the integer 'i' and the integer 'j' respectively. Then since the two vertices adjacent to a 'red' vertex must be assigned the same integer, there exists an infinitely long segment (subgraph) of the array G to which the assigned integers are in the form of '$\cdots - i - j - i - j - i - j - \cdots$', which is certainly impossible. Therefore a contradiction exists. So for any *pair* $[i, j]$ in group D, $z(i, j) = 0$.

Let 'x' denote the number of *distinct* integers assigned to 'green' vertices, and let 'X' denote the set of those x distinct integers. It's simple to see that exactly $\binom{x}{2}$ *pairs* $[i, j]$ are in group A or group B, where $i \in X$ and $j \in X$ — and among them at least $x - 1$ pairs are in group B. It's also simple to see that exactly $x(N - x)$ pairs are in group C and exactly $\binom{N-x}{2}$ pairs are in group D. By using the upper bounds we've derived for $z(i, j)$, we see that the number of edges in G is at most $[\binom{x}{2} - (x - 1)] \cdot (2m - 2) + (x - 1) \cdot (2m-3) + x(N-x) \cdot (2m-2) + \binom{N-x}{2} \cdot 0 = (1-m)x^2 + (2mN - 2N - m)x + 1$, whose maximum value (at integer solutions) is achieved when $x = N - 1$ — and that maximum value is $(N-1)[(m-1)N-1]+1$. So n_{max}, the number of vertices in G, is at most $(N - 1)[(m - 1)N - 1] + 2$.

□

Lemma 3: Let the values of N, K, m and L be fixed, where $N = 3$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a linear array of n vertices. Then $n_{max} \leq (N - 1)[(m - 1)N - 1] + 2$.

Proof: Let $G = (V, E)$ be a linear array of n vertices that has an MCI on it. We need to show that $n \leq (N - 1)[(m - 1)N - 1] + 2$.

If in the MCI on G, no two adjacent vertices are assigned the same integer, then with the same argument as in the proof of Lemma 2, it can be shown that $n \leq (N - 1)[(m - 1)N - 1] + 2$.

Now assume there are two adjacent vertices in G that are assigned the same integer. Clearly we can find t non-overlapping clusters in G, such that $n \leq 2t + 2$ and at least one of the t clusters contains two vertices

that are assigned the same integer. Among those t non-overlapping clusters, let x, y, z, a, b and c respectively denote the number of clusters that are assigned only the integer '1', only the integer '2', only the integer '3', both the integers '1' and '2', both the integers '2' and '3', and both the integers '1' and '3'. Since the interleaving is an MCI, any m non-overlapping clusters are assigned at least $K = 3$ distinct integers. Therefore $x+y+a \leq m-1$, $y+z+b \leq m-1$, $z+x+c \leq m-1$. So $2x+2y+2z+a+b+c \leq 3m-3$. So $x+y+z+a+b+c \leq 3m-3-(x+y+z)$. Since $x+y+z \geq 1$, $t = x+y+z+a+b+c$, and $n \leq 2t+2$, we get $n \leq 2(x+y+z+a+b+c+1) \leq 2[3m-3-(x+y+z)+1] \leq 6m-6 = (N-1)[(m-1)N-1]+2$.

Therefore this lemma is proved.

□

So with Lemma 2 and Lemma 3 proved, we see that Theorem 1 becomes a natural conclusion.

III. Optimal Construction for MCI on Linear Arrays with Constraints $L = 2$ and $K = 3$

In this section, we present a construction for MCIs on linear arrays whose lengths attain the upper bound of Theorem 1, therefore proving the exactness of that bound. The construction is shown as the following algorithm.

Algorithm 1: MCI on the longest linear array with constraints $L = 2$ and $K = 3$

Input: Parameters N, K, m and L, where $N \geq 3$, $K = 3$, $m \geq 2$ and $L = 2$. A linear array $G = (V, E)$ of $n = (N-1)[(m-1)N-1]+2$ vertices.

Output: An MCI on G.

Algorithm:

Let $H = (V_H, E_H)$ be a multi-graph. The vertex set of H, V_H, is $\{u_1, u_2, \cdots, u_N\}$. For any two vertices u_i and u_j ($i \neq j$), there are $2m-3$ edges between them if $2 \leq i = j+1 \leq N-1$ or $2 \leq j = i+1 \leq N-1$, and there are $2m-2$ edges between them otherwise. There is no loop in H. (Therefore H has exactly $n-1$ edges.)

Find a walk in H, $u_{k_1} \rightarrow u_{k_2} \rightarrow \cdots \rightarrow u_{k_n}$, that satisfies the following two requirements: (1) the walk starts with u_1 and ends with u_{N-1} — namely, $u_{k_1} = u_1$ and $u_{k_n} = u_{N-1}$ — and passes every edge in H exactly once; (2) for any two vertices of H, the walk passes all the edges between them *consecutively*.

For $i = 1, 2, \cdots, n$, assign the integer 'k_i' to the vertex v_i in G, and we get an MCI on G.

□

Here is an example of the above algorithm.

Example 2: Assume $G = (V, E)$ is a linear array of $n = 11$ vertices, and the parameters are $N = 4$, $K = 3$, $m = 2$ and $L = 2$. Therefore $n = (N-1)[(m-1)N-1]+2$. Algorithm 1 constructs a graph $H = (V_H, E_H)$, which is shown in Fig. 3(a). The walk in H, $u_{k_1} \rightarrow u_{k_2} \rightarrow \cdots \rightarrow u_{k_n}$, can be easily found. For example, we can let the walk be $u_1 \rightarrow u_3 \rightarrow u_1 \rightarrow u_4 \rightarrow u_1 \rightarrow u_2 \rightarrow u_4 \rightarrow u_2 \rightarrow u_3 \rightarrow u_4 \rightarrow u_3$. Corresponding to that walk, we get the interleaving on G as shown in Fig. 3(b). It can be easily verified that the interleaving is indeed an MCI.

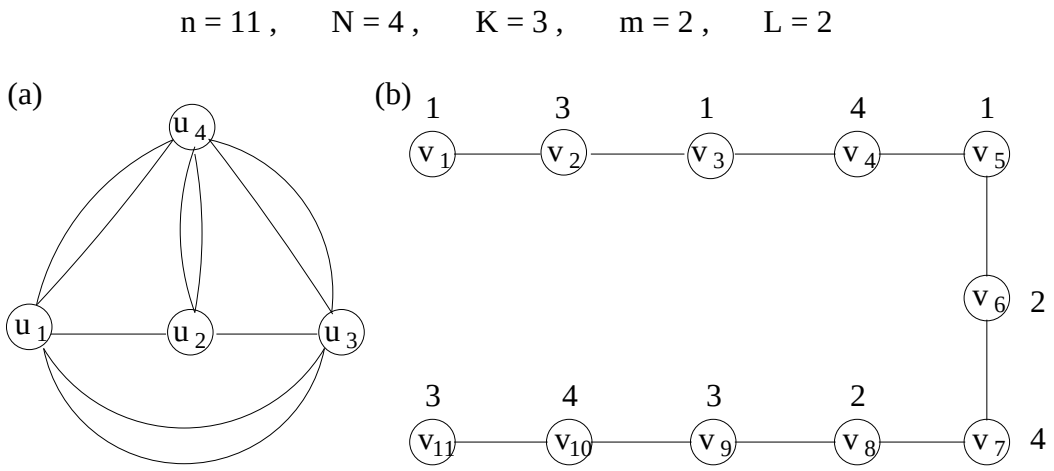


Fig. 3. (a) The graph $H = (V_H, E_H)$ (b) MCI on the array $G = (V, E)$

□

Theorem 2: Algorithm 1 correctly outputs an MCI on the linear array G.

Proof: The interleaving on G that Algorithm 1 outputs corresponds to a walk in the graph $H = (V_H, E_H)$. The N vertices of H correspond to the N integers interleaved on G. First let us assume such a walk exists and show that the interleaving on G is indeed an MCI. Every cluster in G are assigned two different integers since there is no loop in H. For any two vertices in H, there are at most $2m - 2$ edges between them, and those edges are passed consecutively in the walk. So there do not exist m or more than m non-overlapping clusters in G that are assigned the same two integers. So every m non-overlapping clusters in G are assigned at least $K = 3$ different integers. So the interleaving on G is an MCI.

Now we show that the walk in H exists. The following is a simple way to find a valid walk. Note that the graph H has the following property: only the number of edges between u_1 and u_2, or u_2 and u_3, $\cdots$, or u_{N-2} and u_{N-1}, is odd; the number of edges between any two other vertices is even. So we first get the following walk: $u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow \cdots \rightarrow u_{N-2} \rightarrow u_{N-1}$. Then, for $i = 1, 2, \cdots, N-2$, replace the segment '$u_i \rightarrow u_{i+1}$' in the walk with the segment '$u_i \rightarrow u_{i+1} \rightarrow u_i \rightarrow u_{i+1} \rightarrow \cdots \rightarrow u_i \rightarrow u_{i+1}$', where in the latter segment, each of the $2m - 3$ edges between u_i and u_{i+1} is passed exactly once. Next, for any vertex u_i and vertex u_j such that the edges between them have not been passed by the walk, we make the walk pass the edges between them by replacing a node 'u_i' in the walk with a segment '$u_i \rightarrow u_j \rightarrow u_i \rightarrow u_j \rightarrow \cdots \rightarrow u_i \rightarrow u_j \rightarrow u_i$', where in the segment, each of the $2m - 2$ edges between u_i and u_j is passed exactly once. When doing the replacement, ensure that the edges between any other two vertices are still passed consecutively. It is simple to verify that the final walk we get satisfies all the requirements in Algorithm 1. And that completes our proof.

□

The graph $H = (V_H, E_H)$ in Algorithm 1 has either $2m - 2$ or $2m - 3$ edges between any two of its vertices. H has the same number of edges as the linear array for which the MCI is computed. If we reduce the number of edges between the vertices of H, then the walk in H will correspond to an MCI on a shorter linear array (since the walk will pass fewer edges). Based on this idea, we get an algorithm that can find

MCIs for any linear array on which MCIs exist, instead of just for the longest linear array. We present it as Algorithm 4, and leave it in Appendix I for simplicity. Obviously, an MCI on a short linear array can be got by simply taking a segment of the interleaving on the longest linear array computed by Algorithm 1. However such a method has some unnecessary computation and becomes inefficient when the linear array is substantially shorter than the longest linear array. Algorithm 4 can be easily seen to have complexity $O(n)$.

The following theorem presents the necessary and sufficient condition for there to exist an MCI on a linear array, when $L = 2$ and $K = 3$. Equivalently, it shows that the upper bound given in Theorem 1 is exact.

Theorem 3: When $L = 2$ and $K = 3$, there exists an MCI on a linear array of n vertices if and only if $n \leq (N-1)[(m-1)N - 1] + 2$.

Proof: By Theorem 1 and Theorem 2.
□

IV. MCI ON LINEAR ARRAYS WITH CONSTRAINT $K = L + 1$

In this section we study the MCI problem on linear arrays with the constraint that $K = L + 1$. It covers the MCI problem with constraints that $L = 2$ and $K = 3$, which is studied in the previous sections, as a special case.

We define three operations on arrays — 'remove a vertex', 'insert a vertex' and 'combine two arrays'. Let G be an array of n vertices: $(v_1, v_2, \cdots, v_n)$. By 'removing the vertex v_i' from G ($1 \leq i \leq n$), we get a new array '$(v_1, v_2, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n)$'. By 'inserting a vertex $\hat{v}$' in front of the vertex v_i in G ($1 \leq i \leq n$), we get a new array '$(v_1, v_2, \cdots v_{i-1}, \hat{v}, v_i, \cdots, v_n)$'. (Similarly we can define 'inserting a vertex $\hat{v}$ behind the vertex v_i in G' and 'inserting a vertex $\hat{v}$ between the vertices v_i and v_{i+1} in G'.) Let H be an array of n' vertices: $(u_1, u_2, \cdots, u_{n'})$. Assume for $1 \leq i \leq n$, v_i is assigned the integer $c(v_i)$; and assume for $1 \leq i \leq n'$, u_i is assigned the integer $c(u_i)$. Also, let l be a positive integer between 1 and $min(n, n')$, and assume for $1 \leq i \leq l$, $c(v_i) = c(u_{n'-l+i})$. Then by saying 'combining H with G such that the last l vertices of H overlap the first l vertices of G', we mean to construct an array of $n' + n - l$ vertices whose assigned integers are $[c(u_1) - c(u_2) - \cdots - c(u_{n'}) - c(v_{l+1}) - c(v_{l+2}) - \cdots - c(v_n)]$ (which is the same as $[c(u_1) - c(u_2) - \cdots - c(u_{n'-l}) - c(v_1) - c(v_2) - \cdots - c(v_n)]$).

Examples of those operations are shown below.

Example 3: Let G be a linear array as shown in Fig. 4(a), where the integer above each vertex is the integer assigned to it. (So there is an interleaving on G.) By removing the vertex v_1 from G, we get the linear array shown in Fig. 4(b). By inserting a vertex $\hat{v}$ in front of the vertex v_3 in G (or equivalently, behind the vertex v_2 in G, or between the vertex v_2 and v_3 in G), we get the array shown in Fig. 4(c).

Let H be a linear array as shown in Fig. 4(d), where the integer above each vertex is the integer assigned to it. (So there is an interleaving on H, too.) By combining H with G such that the last 2 vertices of H overlap the first 2 vertices of G, we get the linear array shown in Fig. 4(e).
□

Now we present an algorithm which computes an MCI on a linear array. Being different from Algorithm 1, in this algorithm the length of the array is unknown. The algorithm tries to find the longest array that has

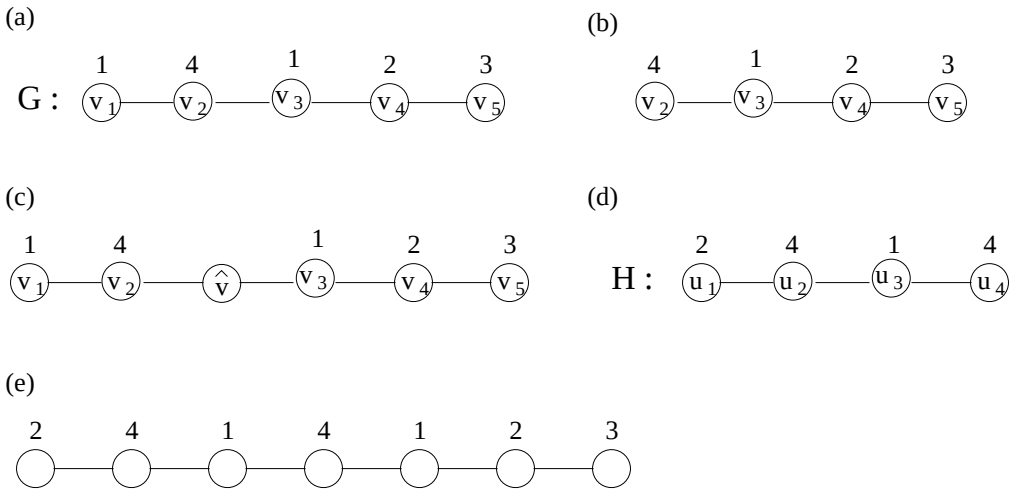


Fig. 4. Illustrations of three operations on linear arrays.

an MCI, and computes an MCI for it. Thus the output of this algorithm not only provides an MCI solution, but also gives a lower bound on the maximum length of the array on which MCIs exist.

Algorithm 2: MCI on linear array with the constraint $K = L + 1$

Input: Parameters N, K, m and L, where $N \geq K = L + 1 \geq 3$ and $m \geq 2$.

Output: An MCI on a linear array $G = (V, E)$ of n vertices, with the value of n as large as possible.

Algorithm:

1. If $L = 2$, then let $G = (V, E)$ be a linear array of $n = (N - 1)[(m - 1)N - 1] + 2$ vertices, and use Algorithm 1 to find an MCI on G. Output G and the MCI on it, then exit. (So step 2 to step 4 will be executed only if $L \geq 3$.)

2. Find a linear array B_{L+1} as long as possible that satisfies the following two conditions:

(1) Each vertex of B_{L+1} is assigned an integer in $\{1, 2, \cdots, L\}$, namely, there is an interleaving of the integers in $\{1, 2, \cdots, L\}$ on B_{L+1};

(2) Define a *segment* of a linear array to be a connected subgraph of the linear array. Then any m non-overlapping *segments* of B_{L+1} each of which contains $L - 1$ vertices are assigned at least L distinct integers.

To find the array B_{L+1}, (recursively) call Algorithm 2 in the following way: when calling Algorithm 2, replace the inputs of the algorithm — N, K, m and L — respectively with L, L, m and $L - 1$; then let the output of Algorithm 2 be the array B_{L+1} with an interleaving on it.

Scan the vertices in B_{L+1} backward (from the last vertex to the first vertex), and insert a new vertex after every $L - 1$ vertices in B_{L+1}. (In other words, if the vertices in B_{L+1} are v_1, v_2, $\cdots$, $v_{\hat{n}}$, then after inserting vertices into B_{L+1} in the way described above, we get a new array of $\hat{n} + \lfloor \frac{\hat{n}}{L-1} \rfloor$ vertices; and if we look at the new array in the reverse order — from the last vertex to the first vertex — then the array is of the form $(v_{\hat{n}}, v_{\hat{n}-1}, \cdots, v_{\hat{n}+1-(L-1)}$, a new vertex, $v_{\hat{n}-(L-1)}, v_{\hat{n}-(L-1)-1}, \cdots, v_{\hat{n}+1-2(L-1)}$, a new vertex, $v_{\hat{n}-2(L-1)}, v_{\hat{n}-2(L-1)-1}, \cdots, v_{\hat{n}+1-3(L-1)}$, a new vertex, $\cdots\cdots)$. In this new array, every connected subgraph of L vertices contains exactly one newly inserted vertex.) Assign the integer '$L + 1$' to every newly inserted

vertex in the new array, and denote this new array by 'A_{L+1}'.

3. for $i = L + 2$ to N do

{ Find a linear array B_i as long as possible that satisfies the following three conditions:

(1) Each vertex of B_i is assigned an integer in $\{1, 2, \cdots, i-1\}$, namely, there is an interleaving of the integers in $\{1, 2, \cdots, i-1\}$ on B_i;

(2) Define a *segment* of a linear array to be a connected subgraph of the linear array. Then any m non-overlapping *segments* of B_i each of which contains $L-1$ vertices are assigned at least L distinct integers;

(3) for $j = 1$ to $L-1$, the j-th last vertex of B_i is assigned the same integer as the $(L-j)$-th vertex of A_{i-1}.

To find the array B_i, (recursively) call Algorithm 2 in the following way: when calling Algorithm 2, replace the inputs of the algorithm — N, K, m and L — respectively with $i-1$, L, m and $L-1$; then let the output of Algorithm 2 be the array B_i with an interleaving on it.

Scan the vertices in B_i backward (from the last vertex to the first vertex), and insert a new vertex after every $L-1$ vertices in B_i. Assign the integer 'i' to every newly inserted vertex in the new array, and denote this new array by 'A_i'.

}

4. Get a new array by combining the arrays A_N, A_{N-1}, $\cdots$, A_{L+1} in the following way: combine A_N with A_{N-1}, combine A_{N-1} with A_{N-2}, $\cdots$, and combine A_{L+2} with A_{L+1} such that the last $L-1$ vertices of A_N overlap the first $L-1$ vertices of A_{N-1}, the last $L-1$ vertices of A_{N-1} overlap the first $L-1$ vertices of A_{N-2}, $\cdots$, and the last $L-1$ vertices of A_{L+2} overlap the first $L-1$ vertices of A_{L+1}. (In other words, if we denote the number of vertices in A_i by l_i, for $L+1 \leq i \leq N$, then the new array we get has $\sum_{i=L+1}^{N} l_i - (L-1)(N-L-1)$ vertices.) Let this new array be $G = (V, E)$. Output G and the interleaving (which is an MCI) on it, then exit.

□

The following is an example of Algorithm 2.

Example 4: In this example, the input parameters for Algorithm 2 are $N = 6$, $K = 4$, $m = 2$ and $L = 3$. That is, we use Algorithm 2 to compute an array that is as long as possible and interleave 6 integers on it, such that in the array, any 2 non-overlapping clusters of length 3 are assigned at least 4 distinct integers.

Algorithm 2 firstly computes a linear array B_4 as long as possible that satisfies the following two conditions: (1) each vertex of B_4 is assigned an integer in $\{1, 2, 3\}$; (2) any $m = 2$ non-overlapping *segments* of B_4 each of which contains $L - 1 = 2$ vertices are assigned at least $L = 3$ distinct integers. To compute B_4, Algorithm 2 calls itself in a recursive way, by setting the inputs of the algorithm — N, K, m and L — to be 3, 3, 2 and 2; during that call, it uses Algorithm 1 to compute B_4. There are more than 1 possible outcomes of Algorithm 1; without loss of generality (WLOG), let us assume the output here is that B_4 is assigned integers in the form of $[1-3-1-2-3-2]$. The array B_4 is shown in Figure 5 (a).

Algorithm 2 then scans B_4 backward, inserts a new vertex into B_4 after every $L - 1 = 2$ vertices, and assigns the integer '4' to every newly inserted vertex. As a result, we get a linear array whose assigned integers are in the form of $[4-1-3-4-1-2-4-3-2]$. We call this new array A_4. A_4 is shown in

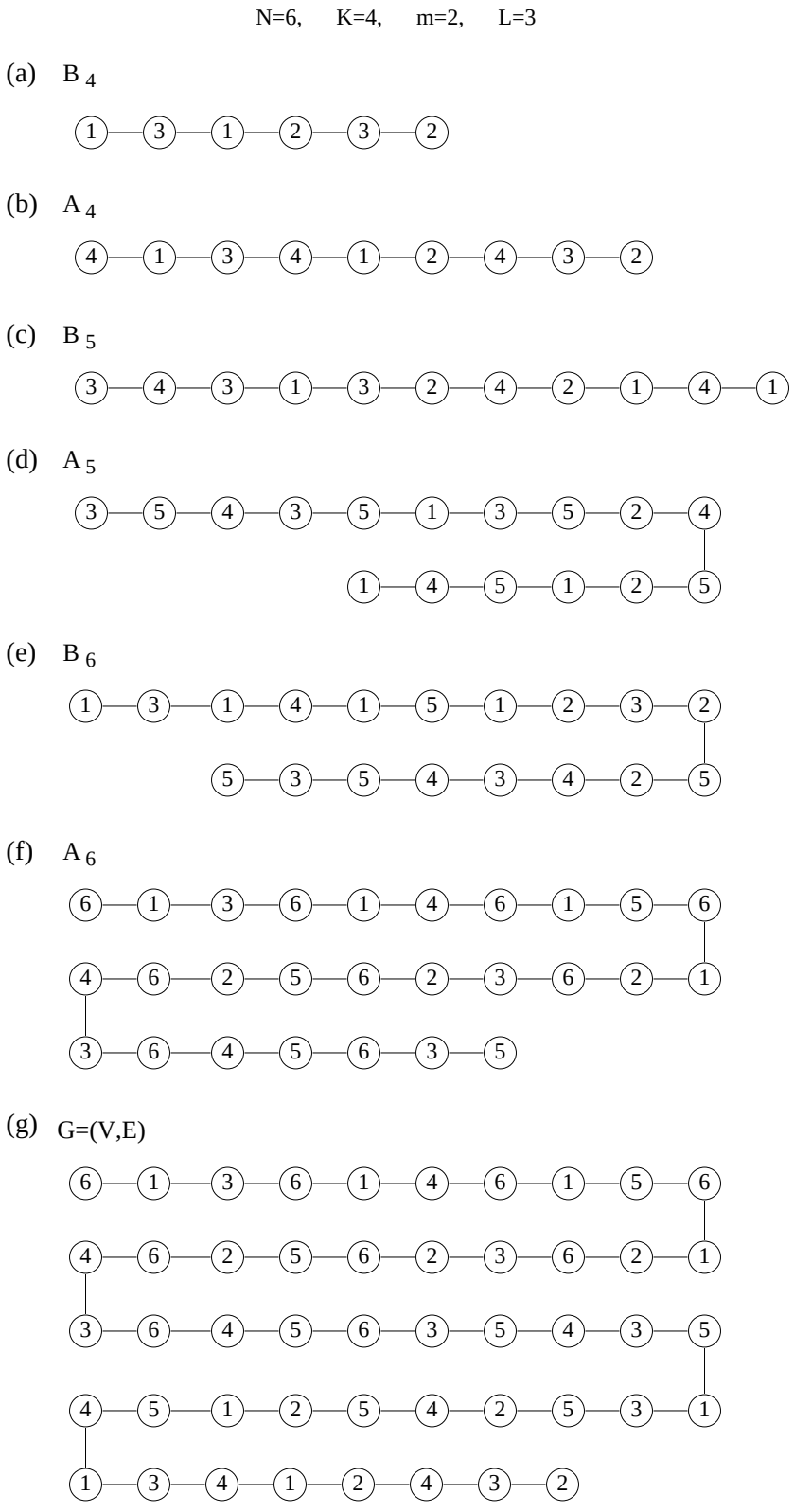


Fig. 5. An example of Algorithm 2.

Figure 5 (b).

Algorithm 2 then computes a linear array B_5 as long as possible that satisfies the following three conditions: (1) each vertex of B_5 is assigned an integer in $\{1, 2, 3, 4\}$; (2) any $m = 2$ non-overlapping *segments* of B_5 each of which contains $L - 1 = 2$ vertices are assigned at least $L = 3$ distinct integers; (3) the last vertex of B_5 is assigned the same integer as the 2nd vertex of A_4 (which is the integer '1'), and the 2nd last vertex of B_5 is assigned the same integer as the 1st vertex of A_4 (which is the integer '4').

Algorithm 2 computes B_5 by once again calling itself. Algorithm 2 can use the following method to find an array that satisfies all the above 3 conditions. Firstly, use Algorithm 1 to find a linear array that satisfies the first 2 conditions, which is easy, and call this linear array C_5. All the integers assigned to C_5 are in the set $\{1, 2, 3, 4\}$; and from Algorithm 1, it is simple to see that the last two vertices in C_5 are assigned two different integers. (Note that the first two vertices in A_4 are also assigned two different integers.) So by permuting the names of the integers assigned to C_5, we can get a linear array that satisfies not only the first 2 conditions but also the 3rd condition. Call this array B_5. There are more than 1 possible result of B_5. WLOG, we assume the integers assigned to B_5 are in the form of $[3-4-3-1-3-2-4-2-1-4-1]$. B_5 is shown in Figure 5 (c). Then Algorithm 2 inserts vertices into B_5 and gets a new array A_5, whose assigned integers are in the form of $[3-5-4-3-5-1-3-5-2-4-5-2-1-5-4-1]$. A_5 is shown in Figure 5 (d).

Next, Algorithm 2 computes a linear array B_6, by calling itself again. WLOG, we assume the integers assigned to B_6 are in the form of $[1-3-1-4-1-5-1-2-3-2-5-2-4-3-4-5-3-5]$. B_6 is shown in Figure 5 (e). Then Algorithm 2 inserts vertices into B_6 and gets a new array A_6, whose assigned integers are in the form of $[6-1-3-6-1-4-6-1-5-6-1-2-6-3-2-6-5-2-6-4-3-6-4-5-6-3-5]$. A_6 is shown in Figure 5 (f).

Finally, Algorithm 2 combines A_6, A_5 and A_4 such that the last $L - 1 = 2$ vertices of A_6 overlap the first 2 vertices of A_5, and the last $L-1 = 2$ vertices of A_5 overlap the first 2 vertices of A_4. As a result, we get an array $G = (V, E)$ of 48 vertices which is assigned the integers $[6-1-3-6-1-4-6-1-5-6-1-2-6-3-2-6-5-2-6-4-3-6-4-5-6-3-5-4-3-5-1-3-5-2-4-5-2-1-5-4-1-3-4-1-2-4-3-2]$. G is shown in Figure 5 (g). This is the output of Algorithm 2. It can be verified that the interleaving on G is indeed an MCI.

□

Algorithm 2 outputs an array G, which is as long as the algorithm can find, and an MCI on G. The MCI on G has a 'hierarchical-chain' structure, because G is a chain of the sub-arrays A_{L+1}, A_{L+2}, $\cdots$, A_N, and these sub-arrays form the horizontal hierarchy because on them more and more integers are interleaved and they have increasing lengths. Each A_i ($L + 1 \leq i \leq N$) is derived from an array B_i. Since B_i is got by recursively calling Algorithm 2, it also is a chain of its own sub-arrays — and the same analysis can be performed for the sub-arrays in B_i $\cdots\cdots$ So such sub-arrays form the vertical hierarchy in the MCI. G's length, n, is unknown before Algorithm 2 ends. But if we can use n to evaluate the complexity of Algorithm 2, then Algorithm 2 can be easily seen to have complexity $O(n)$. Algorithm 2 constructs the array G piece by piece. So it is simple to see that the algorithm can be easily modified to efficiently compute MCIs on any array of less than n vertices.

Below we prove the correctness of Algorithm 2.

Theorem 4: Algorithm 2 is correct.

Proof: We prove this theorem by induction. If $L = 2$, then Algorithm 2 uses Algorithm 1 to compute the MCI — so the result is clearly correct. Also, we notice that for any MCI output by Algorithm 1, any two adjacent vertices are assigned different integers. We use those as the base case.

Let I be an integer such that $2 < I \leq L$. Let's assume the following statement is true: if we replace the inputs of Algorithm 2 — parameters N, K, m and L — with any other set of valid inputs $\hat{N}, \hat{K} = i + 1, \hat{m}$ and i such that $2 \leq i < I$, Algorithm 2 will correctly output an MCI on an array; and in that MCI, any i consecutive vertices are assigned i different integers. (This is our induction assumption.)

Now let's replace the inputs of Algorithm 2 — parameters N, K, m and L — with a set of valid inputs $N', K' = I + 1, m'$ and I. Then Algorithm 2 needs to compute (in its step 2 and step 3) $N' - I$ arrays: B_{I+1}, B_{I+2}, $\cdots$, $B_{N'}$. For $I + 1 \leq j \leq N'$, B_j is (recursively) computed by calling Algorithm 2. The interleaving on B_j is in fact an MCI where the size of each cluster is $I - 1$ — so by the induction assumption, Algorithm 2 will correctly output the interleaving on B_j. B_j is assigned the integers in $\{1, 2, \cdots, j - 1\}$; and by the induction assumption, any $I - 1$ consecutive vertices in B_j are assigned $I - 1$ different integers. The array A_{I+1} is constructed by inserting vertices into B_{I+1} such that every I consecutive vertices in A_{I+1} contains exactly one newly inserted vertex, and all the newly inserted vertices are assigned the integer '$I + 1$'. So any I consecutive vertices in A_{I+1} are assigned I different integers. Therefore it is always feasible to adjust the interleaving on B_{I+2} to make the last $I - 1$ vertices of B_{I+2} be assigned the same integers as the first $I - 1$ vertices of A_{I+1}. Noticing that the last $I - 1$ vertices of B_{I+2} are assigned the same integers as the last $I - 1$ vertices of A_{I+2}, we see that A_{I+2} and A_{I+1} can be successfully combined with $I - 1$ overlapping vertices by Algorithm 2. Similarly, for $I + 3 \leq t \leq N'$, A_t and A_{t-1} can be successfully combined by Algorithm 2; and for $I + 2 \leq t \leq N'$, any I consecutive vertices in A_t are assigned I different integers. Algorithm 2 uses G to denote the array got by combining A_{L+1}, A_{L+2}, $\cdots$, A_N. Clearly any I consecutive vertices in G are also I consecutive vertices in A_j for some j ($I + 1 \leq j \leq N'$), therefore are assigned I different integers. And for any m' non-overlapping clusters in G — each cluster here contains I vertices — either they are all contained in A_j for some j ($I + 1 \leq j \leq N'$), or at least one cluster is contained in $A_{j'}$ for some j' and one other cluster is contained in $A_{j''}$ for some $j'' \neq j'$ ($I + 1 \leq j', j'' \leq N'$). In the former case, by removing those vertices that are assigned the integer 'j' in those m' clusters, we get m' non-overlapping connected subgraphs in B_j each of which contains $I - 1$ vertices, which are assigned at least I different integers not including 'j' — so the m' clusters in G (which are also in A_j) are assigned at least $I + 1$ different integers. In the latter case, without loss of generality, let's say $j' < j''$. Then the cluster in $A_{j'}$ are assigned I different integers not including 'j''', and the cluster in $A_{j''}$ is assigned an integer 'j''' — so the m' clusters in G are assigned at least $I + 1$ different integers. Therefore the interleaving on G is an MCI (with parameters N', K', m' and I). So the induction assumption also holds when $i = I$.

Algorithm 2 computes the result for the original problem by recursively calling itself. By the above induction, every intermediate time Algorithm 2 is called, the output is correct. So the final output of Algorithm 2 is also correct.

□

The length of the longest array on which an MCI exists increases when N, the number of integers that are interleaved, increases. The performance of Algorithm 2 can be evaluated by the difference between the

length of the array constructed by Algorithm 2 and the length of the longest array on which an MCI exists. We're interested in studying how the difference goes when N increases. The following theorem shows the result.

Theorem 5: Fix the values of the parameters K, m and L, where $K = L + 1 \geq 3$ and $m \geq 2$, and let N be a variable ($N \geq K$). Then the longest linear array on which an MCI exists has $\frac{m-1}{(L-1)!}N^L + O(N^{L-1})$ vertices. And the array output by Algorithm 2 also has $\frac{m-1}{(L-1)!}N^L + O(N^{L-1})$ vertices.

Proof: Let $G = (V, E)$ be a linear array of n vertices with an MCI on it. Then by Proposition 1, $n \leq (m-1)L\binom{N}{L} + (L-1)$. So $n \leq \frac{m-1}{(L-1)!}N^L + O(N^{L-1})$.

When $L = 2$, Algorithm 2 outputs an array of $(N-1)[(m-1)N-1]+2$ vertices. When $L \geq 3$, to get the output, Algorithm 2 needs to construct the arrays A_{L+1}, A_{L+2}, $\cdots$, A_N; and for $L+1 \leq i \leq N$, A_i is got by inserting vertices into the array B_i. B_i is again an output of Algorithm 2, which is assigned $i-1$ distinct integers, and in which an corresponding 'cluster' has $L-1$ vertices. Let's use $F(N, m, L)$ to denote the number of vertices in the array output by Algorithm 2, and use $A(i, m, L)$ to denote the number of vertices in the array A_i. Then based on the above observed relations, we get the following 3 equations:

(1) $F(N, m, 2) = (N-1)[(m-1)N-1]+2$;

(2) when $L \geq 3$, $F(N, m, L) = \sum_{i=L+1}^{N} A(i, m, L) - (N-L-1)(L-1)$;

(3) when $i \geq L+1 \geq 4$, $A(i, m, L) = \lfloor \frac{L}{L-1} \cdot F(i-1, m, L-1) \rfloor$. (Note that $F(i-1, m, L-1)$ is the number of vertices in the array B_i.)

By solving the above equations, we get $F(N, m, L) = \frac{m-1}{(L-1)!}N^L + O(N^{L-1})$, which meets the upper bound on the array's length we've derived. So the longest linear array on which an MCI exists has $\frac{m-1}{(L-1)!}N^L + O(N^{L-1})$ vertices; and the array output by Algorithm 2 also has $\frac{m-1}{(L-1)!}N^L + O(N^{L-1})$ vertices.

□

Theorem 5 shows that the array output by Algorithm 2 is asymptotically as long as the longest array on which an MCI exists. What's more, the lengths of those two arrays have the same highest-degree term (in N).

We end this part of discussion with some numerical results which are representative. In Table 1, the length of the array output by Algorithm 2 (denoted by 'Output of Algorithm 2' in the table) is compared with the upper bound on array lengths of Theorem 1 (denoted by 'Upper bound' in the table), for four different sets of parameters — $m = 2$ and $L = 3$, $m = 2$ and $L = 5$, $m = 5$ and $L = 3$, and $m = 5$ and $L = 5$. (Note that here $K = L + 1$, so the value of parameter K is determined by L. The length of an array is defined to be the number of vertices in it.) If we use n to denote the length of the array output by Algorithm 2, and use U_{bound} to denote the upper bound of Theorem 1, then the 'relative difference' in the table is defined to be $\frac{U_{bound}-n}{U_{bound}} = 1 - \frac{n}{U_{bound}}$. Table 1 shows how the 'relative difference' decreases when N increases. With Theorem 5, we can prove that as N approaches $+\infty$, the 'relative difference' becomes arbitrarily close to 0 — the optimal value. We comment that U_{bound} is a quite loose upper bound, so the relative difference between the length of the array output by Algorithm 2 and the true length of the longest array on which an MCI exists is even smaller than that shown in the table.

	$m = 2$ and $L = 3$			$m = 2$ and $L = 5$		
N	Output of Algorithm 2	Upper bound	Relative difference	Output of Algorithm 2	Upper bound	Relative difference
10	312	362	0.1381	930	1264	0.2642
20	3177	3422	0.0716	68265	77524	0.1194
50	57072	58802	0.0294	10081020	10593804	0.0484
100	477897	485102	0.0149	367196445	376437604	0.0245
150	1637472	1653902	0.0099	2.9093×10^9	2.9580×10^9	0.0165
200	3910797	3940202	0.0075	1.2521×10^{10}	1.2678×10^{10}	0.0124
	$m = 5$ and $L = 3$			$m = 5$ and $L = 5$		
N	Output of Algorithm 2	Upper bound	Relative difference	Output of Algorithm 2	Upper bound	Relative difference
10	1383	1442	0.0409	4395	5044	0.1287
20	13428	13682	0.0186	298785	310084	0.0364
50	233463	235202	0.0074	41846205	42375204	0.0125
100	1933188	1940402	0.0037	1.4964×10^9	1.5058×10^9	0.0062
150	6599163	6615602	0.0025	1.1783×10^{10}	1.1832×10^{10}	0.0041
200	15731388	15760802	0.0019	5.0556×10^{10}	5.0713×10^{10}	0.0031

Table 1: Comparison between the length of the array output by Algorithm 2 and an upper bound, and their relative difference.

V. MCI ON RINGS

In this section, we generalize our results on MCI from linear arrays to rings, for the case of $L = 2$ and $K = 3$. The analysis for the two kinds of graphs bears plenty of similarity, even though the 'circular' structure of the ring leads to certain difference sometimes.

Let $G = (V, E)$ be a ring. The following notations will be used in this section and the appendices of this paper. We denote the n vertices in the ring $G = (V, E)$ by v_1, v_2, $\cdots$, v_n. For $2 \leq i \leq n-1$, the two vertices adjacent to v_i are v_{i-1} and v_{i+1}. Vertex v_1 and v_n are adjacent to each other. A connected subgraph of G induced by vertices $v_i, v_{i+1}, \cdots, v_j$ is denoted by $(v_i, v_{i+1}, \cdots, v_j)$. If a set of integers are interleaved on G, then $c(v_i)$ denotes the integer assigned to vertex v_i. The integers assigned to a connected subgraph of G, $(v_i, v_{i+1}, \cdots, v_j)$, are denoted by $[c(v_i) - c(v_{i+1}) - \cdots - c(v_j)]$.

The following three lemmas reveal some structural properties of MCI on rings and present an upper bound on the rings' lengths.

Lemma 4: Let the values of N, K, m and L be fixed, where $N \geq 4$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a ring of n vertices. Then in any MCI on a ring $G = (V, E)$ of n_{max} vertices, no two adjacent vertices are assigned the same integer.

Lemma 5: Let the values of N, K, m and L be fixed, where $N \geq 4$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a ring of n vertices. Then $n_{max} \leq (N-1)[(m-1)N - 1]$.

Lemma 6: Let the values of N, K, m and L be fixed, where $N = 3$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a ring of n vertices. Then $n_{max} \leq (N-1)[(m-1)N-1]$.

The techniques used to prove the above three lemmas are similar to the proofs of Lemma 1, 2 and 3 (but also have difference, especially for Lemma 5 and Lemma 6). For simplicity, we present the proofs of the above three lemmas in Appendix II.

Below we present an algorithm for finding MCIs on rings. Note that an Eulerian walk in a graph is a closed walk that passes every edge of the graph exactly once.

Algorithm 3: MCI on ring with constraints $L = 2$ and $K = 3$

Input: A ring $G = (V, E)$ of n vertices. Parameters N, K, m and L, where $N \geq 3$, $K = 3$, $m \geq 2$ and $L = 2$.

Output: An MCI on G.

Algorithm:

1. If $n > (N-1)[(m-1)N-1]$, there does not exist an MCI on G, so exit the algorithm.

2. If $n \leq N$, arbitrarily select n integers in the set $\{1, 2, \cdots, N\}$, and assign one distinct integer to each vertex, then exit the algorithm.

3. If $N < n \leq (N-1)[(m-1)N-1]$ and $n-\{(N-1)[(m-1)N-1]\}$ is even, then let $H = (V_H, E_H)$ be such a multi-graph: its vertex set $V_H = \{u_1, u_2, \cdots, u_N\}$; for any $1 \leq i < j \leq N$, there are $x_{i,j}$ undirected edges between u_i and u_j; there is no loop in H. The integers $x_{i,j}$ ($1 \leq i < j \leq N$) satisfy the following four requirements:

(1) for $1 \leq i \leq N-1$, $x_{i,N}$ is even and $0 \leq x_{i,N} \leq 2m-2$;

(2) for $1 \leq i \leq N-2$ and $j = i+1$, $x_{i,j}$ is odd and $1 \leq x_{i,j} \leq 2m-3$; also, $x_{1,N-1}$ is odd and $1 \leq x_{1,N-1} \leq 2m-3$;

(3) for $1 \leq i \leq N-4$ and $i+2 \leq j \leq N-2$, $x_{i,j}$ is even and $0 \leq x_{i,j} \leq 2m-2$; also, for $2 \leq i \leq N-3$ and $j = N-1$, $x_{i,j}$ is even and $0 \leq x_{i,j} \leq 2m-2$;

(4) if we define S as $S = \{x_{i,j} | i = 1, 2, \cdots, N; j = 1, 2, \cdots, N; i < j\}$, then $\sum_{x \in S} x = n$.

Find an Eulerian walk in H, $u_{k_1} \rightarrow u_{k_2} \rightarrow \cdots \rightarrow u_{k_n}$ (and finally back to u_{k_1}), that satisfies the following condition: for any $1 \leq i < j \leq N$, the walk passes all the $x_{i,j}$ edges between u_i and u_j *consecutively*.

For $i = 1, 2, \cdots, n$, assign the integer 'k_i' to the vertex v_i in G, then exit the algorithm.

4. If $N < n \leq (N-1)[(m-1)N-1]$ and $n-\{(N-1)[(m-1)N-1]\}$ is odd, then let $H = (V_H, E_H)$ be such a multi-graph: its vertex set $V_H = \{u_1, u_2, \cdots, u_N\}$; for any $1 \leq i < j \leq N$, there are $x_{i,j}$ undirected edges between u_i and u_j; there is no loop in H. The integers $x_{i,j}$ ($1 \leq i < j \leq N$) satisfy the following three requirements:

(1) for $1 \leq i \leq N-1$ and $j = i+1$, $x_{i,j}$ is odd and $1 \leq x_{i,j} \leq 2m-3$; also, $x_{1,N}$ is odd and $1 \leq x_{1,N} \leq 2m-3$;

(2) for $1 \leq i \leq N-3$ and $i+2 \leq j \leq N-1$, $x_{i,j}$ is even and $0 \leq x_{i,j} \leq 2m-2$; also, for $2 \leq i \leq N-2$ and $j = N$, $x_{i,j}$ is even and $0 \leq x_{i,j} \leq 2m-2$;

(3) if we define S as $S = \{x_{i,j} | i = 1, 2, \cdots, N; j = 1, 2, \cdots, N; i < j\}$, then $\sum_{x \in S} x = n$.

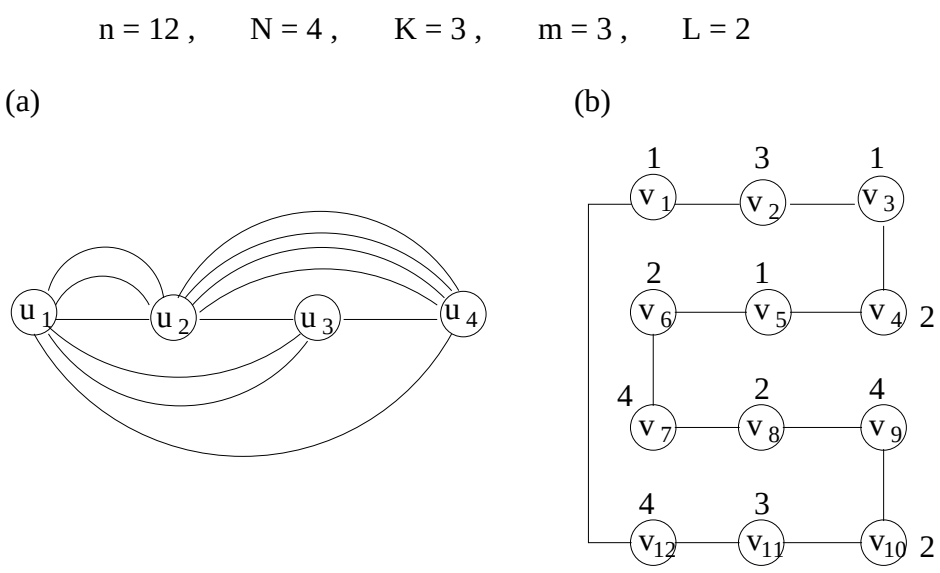


Fig. 6. (a) The graph $H = (V_H, E_H)$ (b) MCI on the ring $G = (V, E)$

Find an Eulerian walk in H, $u_{k_1} \rightarrow u_{k_2} \rightarrow \cdots \rightarrow u_{k_n}$ (and finally back to u_{k_1}), that satisfies the following condition: for any $1 \leq i < j \leq N$, the walk passes all the $x_{i,j}$ edges between u_i and u_j *consecutively*.

For $i = 1, 2, \cdots, n$, assign the integer 'k_i' to the vertex v_i in G, then exit the algorithm.

□

The following is an example of Algorithm 3.

Example 5: Assume $G = (V, E)$ is a ring of $n = 12$ vertices, and the parameters are $N = 4$, $K = 3$, $m = 3$ and $L = 2$. Therefore $N < n \leq (N-1)[(m-1)N - 1]$ and $n - \{(N-1)[(m-1)N - 1]\} = -9$ is odd. So Algorithm 3's step 4 is used to compute the interleaving. We can (easily) choose the following values for $x_{i,j}$: $x_{1,2} = 3$, $x_{1,3} = 2$, $x_{1,4} = x_{2,3} = x_{3,4} = 1$, $x_{2,4} = 4$. Then the graph $H = (V_H, E_H)$ is as shown in Fig. 6(a). We can then (easily) find the following Eulerian walk in H: $u_1 \rightarrow u_3 \rightarrow u_1 \rightarrow u_2 \rightarrow u_1 \rightarrow u_2 \rightarrow u_4 \rightarrow u_2 \rightarrow u_4 \rightarrow u_2 \rightarrow u_3 \rightarrow u_4$ (then back to u_1). Corresponding to that walk, we get the MCI as shown in Fig. 6(b).

□

Theorem 6: Algorithm 3 correctly outputs an MCI on the ring G.

Theorem 7: When $L = 2$ and $K = 3$, there exists an MCI on a ring of n vertices if and only if $n \leq (N-1)[(m-1)N - 1]$.

For simplicity, we skip the proofs of the above two theorems. (The correctness of these two theorems should be very clear once the proofs for Theorem 2 and Theorem 3 are understood.)

It is simple to see that Algorithm 3 has complexity $O(n)$.

VI. Conclusion

In this paper, the multi-cluster interleaving (MCI) problem is defined for flexible parallel data-retrieving. Two families of MCI problems are solved for arrays/rings. MCI seems to have natural applications in data-streaming, broadcasting, disk storage, etc.

There exist many open problems in MCI. How to optimally construct MCI without the constraint that $K = L + 1$ is still unknown. On the other hand, in the MCI problem, the linear array / ring can also be extended to be more general graphs. We hope that the techniques for solving the MCI problems presented in this paper will provide insights of value for solving those more general MCI problems.

Appendix I

In this appendix, we present Algorithm 4, which computes MCIs for linear arrays when $L = 2$ and $K = 3$.

Algorithm 4: MCI on linear array with constraints $L = 2$ and $K = 3$

Input: A linear array $G = (V, E)$ of n vertices. Parameters N, K, m and L, where $N \geq 3$, $K = 3$, $m \geq 2$ and $L = 2$.

Output: An MCI on G.

Algorithm:

1. If $n > (N-1)[(m-1)N-1] + 2$, there does not exist an MCI on G, so exit the algorithm.

2. If $n \leq N$, arbitrarily select n integers in the set $\{1, 2, \cdots, N\}$, and assign one distinct integer to each vertex, then exit the algorithm.

3. If $N < n \leq (N-1)[(m-1)N-1] + 2$ and $n - \{(N-1)[(m-1)N-1] + 2\}$ is even, then let $H = (V_H, E_H)$ be such a multi-graph: its vertex set $V_H = \{u_1, u_2, \cdots, u_N\}$; for any $1 \leq i < j \leq N$, there are $x_{i,j}$ undirected edges between u_i and u_j; there is no loop in H. The integers $x_{i,j}$ ($1 \leq i < j \leq N$) satisfy the following four requirements:

(1) for $1 \leq i \leq N-1$, $x_{i,N}$ is even and $0 \leq x_{i,N} \leq 2m-2$;

(2) for $1 \leq i \leq N-2$ and $j = i+1$, $x_{i,j}$ is odd and $1 \leq x_{i,j} \leq 2m-3$;

(3) for $1 \leq i \leq N-3$ and $i+2 \leq j \leq N-1$, $x_{i,j}$ is even and $0 \leq x_{i,j} \leq 2m-2$;

(4) if we define S as $S = \{x_{i,j} | i = 1, 2, \cdots, N; j = 1, 2, \cdots, N; i < j\}$, then $\sum_{x \in S} x = n - 1$.

Find a walk in H, $u_{k_1} \rightarrow u_{k_2} \rightarrow \cdots \rightarrow u_{k_n}$, that satisfies the following two requirements:

(1) the walk starts with u_1 and ends with u_{N-1} — namely, $u_{k_1} = u_1$ and $u_{k_n} = u_{N-1}$ — and passes every edge in H exactly once;

(2) for any $1 \leq i < j \leq N$, the walk passes all the $x_{i,j}$ edges between u_i and u_j *consecutively*.

For $i = 1, 2, \cdots, n$, assign the integer 'k_i' to the vertex v_i in G, then exit the algorithm.

4. If $N < n \leq (N-1)[(m-1)N-1] + 2$ and $n - \{(N-1)[(m-1)N-1] + 2\}$ is odd, then use step 3 to find an interleaving on a linear array of $n+1$ vertices. Remove one vertex from an end of that linear array, and we get an interleaving on G. Then exit the algorithm.

□

We comment that in step 3 of the above algorithm, the numbers of edges between the vertices of H, $x_{i,j}$ ($1 \leq i < j \leq N$), can be easily determined using just elementary methods, since the constraints there are very simple.

APPENDIX II

In this appendix, we present the proofs of Lemma 4, 5 and 6.

Lemma 4: Let the values of N, K, m and L be fixed, where $N \geq 4$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a ring of n vertices. Then in any MCI on a ring $G = (V, E)$ of n_{max} vertices, no two adjacent vertices are assigned the same integer.

Proof: The proof is by contradiction. Let $G = (V, E)$ be a ring of n_{max} vertices. Assume there is an MCI on G; and assume two adjacent vertices of G are assigned the same integer. Then WLOG, one of the following two cases must be true (because we can always get one of the two cases by permuting the names of the integers and by shifting the indices of the vertices):

Case 1: There exist 4 consecutive vertices in G — v_i, v_{i+1}, v_{i+2}, v_{i+3} — such that $c(v_i) = 1$, $c(v_{i+1}) = c(v_{i+2}) = 2$, and $c(v_{i+3}) = 1$ or 3.

Case 2: There exist $x+2 \geq 5$ consecutive vertices in G — v_i, v_{i+1}, $\cdots$, v_{i+x}, v_{i+x+1} — such that $c(v_i) = 1$, $c(v_{i+1}) = c(v_{i+2}) = \cdots = c(v_{i+x}) = 2$, and $c(v_{i+x+1}) = 1$ or 3.

With the same argument as for the case 1 and case 2 in Lemma 1's proof, we can show that for both cases here, there is a ring of more than n_{max} vertices on which an MCI exists. And that contradicts the definition of n_{max}. So this lemma is proved.

□

Lemma 5: Let the values of N, K, m and L be fixed, where $N \geq 4$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a ring of n vertices. Then $n_{max} \leq (N-1)[(m-1)N - 1]$.

Proof: Let $G = (V, E)$ be a ring of n_{max} vertices. Assume there is an MCI on G. By Lemma 4, no two adjacent vertices in G are assigned the same integer. And clearly, each one of the $N \geq 4$ integers is assigned to at least one vertex in G. We color the vertices in G with three colors — 'red', 'yellow' and 'green' — through the following three steps:

Step 1, for $1 \leq i \leq n_{max}$, if the two vertices adjacent to v_i are assigned the same integer, then we color v_i with the 'red' color;

Step 2, for $1 \leq i \leq n_{max}$, we color v_i with the 'yellow' color if v_i is not colored 'red' and there exists j such that these four conditions are satisfied: (1) $j \neq i$, (2) v_j is not colored 'red', (3) $c(v_j) = c(v_i)$, (3) the following vertices between v_j and v_i — v_{j+1}, v_{j+2}, $\cdots$, v_{i-1} (note that if a lower index exceeds n_{max}, it is subtracted by n_{max}, so that the lower index is always between 1 and n_{max}) — are all colored 'red';

Step 3, for $1 \leq i \leq n_{max}$, if v_i is neither colored 'red' nor colored 'yellow', then we color v_i with the 'green' color.

Clearly, each vertex in G is assigned exactly one of the three colors.

If we arbitrarily pick two different integers — say 'i' and 'j' — from the set $\{1, 2, \cdots, N\}$, then we get a *pair* $[i, j]$ (or $[j, i]$, equivalently). There are totally $\binom{N}{2}$ such un-ordered *pairs*. We partition those $\binom{N}{2}$ *pairs* into four groups 'A', 'B', 'C' and 'D' in the following way:

(1) A *pair* $[i, j]$ is placed in group A if and only if the following two conditions are satisfied: (i) at least one 'green' vertex is assigned the integer 'i' and at least one 'green' vertex is assigned the integer 'j', (ii) there doesn't exist a connected subgraph of G such that the subgraph contains exactly two green vertices (and possibly also vertices of other colors), where one of the green vertices is assigned the integer 'i' and the other green vertex is assigned the integer 'j'.

(2) A *pair* $[i, j]$ is placed in group B if and only if the following two conditions are satisfied: (i) at least one 'green' vertex is assigned the integer 'i' and at least one 'green' vertex is assigned the integer 'j', (ii) there exists a connected subgraph of G such that the subgraph contains exactly two green vertices (and possible also vertices of other colors), where one of the green vertices is assigned the integer 'i' and the other green vertex is assigned the integer 'j'.

(3) A *pair* $[i, j]$ is placed in group C if and only if one of the following two conditions is satisfied: (i) at least one 'green' vertex is assigned the integer 'i' and no 'green' vertex is assigned the integer 'j', (ii) at least one 'green' vertex is assigned the integer 'j' and no 'green' vertex is assigned the integer 'i'.

(4) A *pair* $[i, j]$ is placed in group D if and only if no 'green' vertex is assigned the integer 'i' or 'j'.

E is the set of edges in $G = (V, E)$. For any $1 \leq i \neq j \leq N$, let $E(i, j) \subseteq E$ denote such a subset of edges of G: an edge of G is in $E(i, j)$ if and only if the two endpoints of the edge are assigned the integer 'i' and the integer 'j' respectively. Let $z(i, j)$ denote the number of edges in $E(i, j)$. Upper bounds for $z(i, j)$ are derived below.

For any *pair* $[i, j]$ in group A or group C, $z(i, j) \leq 2m-2$. That is because otherwise there would exist m non-overlapping clusters in G — note that a cluster contains exactly one edge — each of which is assigned only integers 'i' and 'j', which would contradict the assumption that the interleaving on G is an MCI.

Now consider a *pair* $[i, j]$ in group B. $z(i, j) \leq 2m-2$ for the same reason as in the previous case. Assume $z(i, j) = 2m - 2$. Then in order to avoid the existence of m non-overlapping clusters in G each of which is assigned only integers 'i' and 'j', the $z(i, j) = 2m - 2$ edges in $E(i, j)$ must be consecutive in the ring G, which means, WLOG, that there are $2m - 1$ consecutive vertices v_{y+1}, v_{y+2}, $\cdots$, v_{y+2m-1} ($y \geq 0$) whose assigned integers are in the form of $[c(v_{y+1}) - c(v_{y+2}) - \cdots - c(v_{y+2m-1})] = [i - j - i - j - \cdots - i - j - i]$. According to the definition of 'group B', there exist a 'green' vertex v_{k_1} and a 'green' vertex v_{k_2}, such that v_{k_1} is assigned the integer 'i', v_{k_2} is assigned the integer 'j', and either all the vertices in the subgraph $(v_{k_1+1}, v_{k_1+2}, \cdots, v_{k_2-1})$ (note that if an index is greater than n_{max}, then it is subtracted by n_{max} so that the index for a vertex is always between 1 and n_{max}; the same applies to the following contexts) are 'yellow' or 'red', or all the vertices in the subgraph $(v_{k_2+1}, v_{k_2+2}, \cdots, v_{k_1-1})$ are 'yellow' or 'red'. By the way the vertices are colored, it's not difficult to see that either 'v_{k_2-1} is assigned the integer $c(v_{k_1}) = i$' (let's call this 'case 1'), or 'v_{k_1-1} is assigned the integer $c(v_{k_2}) = j$' (let's call this 'case 2'). If 'case 1' is true, then since there is an edge between v_{k_2-1} (which is assigned the integer 'i') and v_{k_2} (which is assigned the integer 'j'), v_{k_2} is in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$. However, it's simple to see that every vertex in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$ that is assigned the integer 'j' must have the color 'red' — so v_{k_2} must be 'red' instead of 'green' — therefore a contradiction exists. Now if 'case 2' is true, since there is an edge between v_{k_1-1} (which is assigned the integer 'j') and v_{k_1} (which is assigned the integer 'i'), both v_{k_1-1} and v_{k_1} are

in the set $\{v_{y+2}, v_{y+3}, \cdots, v_{y+2m-1}\}$. Then again since every vertex in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$ that is assigned the integer 'j' must have the color 'red', and since the color of v_{k_1} is 'green', it is simple to see that all the vertices in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{k_1-1}\}$ that is assigned the integer 'i' must have the color 'red'. Then since the color of v_{y+1} is 'red', the vertex v_y must have been assigned the integer 'j' — and that contradicts the statement that all the edges in $E(i,j)$ are in the subgraph $(v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1})$. Therefore a contradiction always exists when $z(i,j) = 2m-2$. So $z(i,j) \neq 2m-2$. So for any *pair* $[i,j]$ in group B, $z(i,j) \leq 2m-3$.

Now consider a *pair* $[i,j]$ in group D. By the definition of 'group D', no 'green' vertex is assigned the integer 'i' or 'j'. Let $\{v_{k_1}, v_{k_2}, \cdots, v_{k_t}\}$ denote the set of 'yellow' vertices that are assigned the integer 'i', where $k_1 < k_2 < \cdots < k_t$. If $\{v_{k_1}, v_{k_2}, \cdots, v_{k_t}\} \neq \emptyset$, by the way vertices are colored, it's simple to see that every vertex of G that is not in the set $\{v_{k_1}, v_{k_2}, \cdots, v_{k_t}\}$ is 'red'. And if $\{v_{k_1}, v_{k_2}, \cdots, v_{k_t}\} = \emptyset$, then all the vertices that are assigned the integer 'i' are 'red'. Similarly, we can show that either all the vertices that are assigned the integer 'j' are 'red', or there is no 'green' vertex in G and every 'yellow' vertex is assigned the integer 'j'. Together, there are three possible cases in total — case 1, every vertex that is assigned the integer 'i' or 'j' is 'red'; case 2, there is no 'green' vertex in G and every 'yellow' vertex is assigned the integer 'i'; case 3, there is no 'green' vertex in G and every 'yellow' vertex is assigned the integer 'j'. If case 1 is true, assume there is an edge whose two endpoints are assigned the integer 'i' and the integer 'j' respectively. Then since the two vertices adjacent to a 'red' vertex must be assigned the same integer, every vertex in G is assigned either the integer 'i' or the integer 'j', which contradicts the fact that all the $N \geq 4$ integers must have been assigned to the vertices in G. So for any *pair* $[i,j]$ in group D, if case 1 is true, then $z(i,j) = 0$. If case 2 is true, then for any $k \neq i$, there are at most $2m-2$ edges in $E(i,k)$; and for any $k_1 \neq i$ and $k_2 \neq i$, since every vertex that is assigned the integer 'k_1' or 'k_2' is 'red', there is no edge in $E(k_1, k_2)$ — therefore, there are at most $(N-1)\cdot(2m-2) < (N-1)[(m-1)N-1]$ edges in G. So when case 2 is true, there are less than $(N-1)[(m-1)N-1]$ vertices in G, and this lemma is proved. Similarly, when case 3 is true, there are also less than $(N-1)[(m-1)N-1]$ vertices in G, and this lemma is also proved. For this reason, in the following paragraph, we'll simply assume that case 1 is always true — and therefore $z(i,j) = 0$ is true for any *pair* $[i,j]$ in group D.

Let the number of *distinct* integers assigned to 'green' vertices be denoted by 'x', and let 'X' denote the set of those x distinct integers. It's simple to see that exactly $\binom{x}{2}$ *pairs* $[i,j]$ are in group A and group B, where $i \in X$ and $j \in X$ — and among them at least x *pairs* are in group B. It's also simple to see that exactly $x(N-x)$ pairs are in group C and exactly $\binom{N-x}{2}$ *pairs* are in group D. By using the upper bounds we've derived on $z(i,j)$, we see that the number of edges in G is at most $[\binom{x}{2} - x]\cdot(2m-2) + x\cdot(2m-3) + x(N-x)\cdot(2m-2) + \binom{N-x}{2}\cdot 0 = (1-m)x^2 + (2mN-2N-m)x$, whose maximum value (at integer solutions) is achieved when $x = N-1$ — and that maximum value is $(N-1)[(m-1)N-1]$. So n_{max}, the number of vertices in G, is at most $(N-1)[(m-1)N-1]$.

□

Lemma 6: Let the values of N, K, m and L be fixed, where $N = 3$, $K = 3$, $m \geq 2$ and $L = 2$. Let n_{max} denote the maximum value of n such that an MCI exists on a ring of n vertices. Then $n_{max} \leq (N-1)[(m-1)N-1]$.

Proof: Let $G = (V, E)$ be a ring of n_{max} vertices that has an MCI on it. We need to show that $n_{max} \leq$

$(N-1)[(m-1)N-1]$. It's simple to see that G is assigned $N=3$ distinct integers. If in the MCI on G, no two adjacent vertices are assigned the same integer, then with the same argument as in the proof of Lemma 5, it can be shown that $n_{max} \leq (N-1)[(m-1)N-1]$. Now assume there are two adjacent vertices in G that are assigned the same integer. Then there are three possible cases.

Case 1: n_{max} is even.

Case 2: n_{max} is odd, and there are at least 2 non-overlapping clusters in G each of which is assigned only 1 distinct integer.

Case 3: n_{max} is odd, and there don't exist 2 non-overlapping clusters in G each of which is assigned only 1 distinct integer.

We consider the three cases one by one.

Case 1: n_{max} is even. In this case, clearly we can find $\frac{n_{max}}{2}$ non-overlapping clusters (of size $L=2$) such that at least one of them contains two vertices that are assigned the same integer. Among those $\frac{n_{max}}{2}$ non-overlapping clusters, let x, y, z, a, b and c respectively denote the number of clusters that are assigned only integer '1', only integer '2', only integer '3', both integers '1' and '2', both integers '2' and '3', and both integers '1' and '3'. Since the interleaving is an MCI, clearly $x+y+a \leq m-1$, $y+z+b \leq m-1$, $z+x+c \leq m-1$. So $2x+2y+2z+a+b+c \leq 3m-3$. So $x+y+z+a+b+c \leq 3m-3-(x+y+z)$. Since $x+y+z \geq 1$ and $n_{max}=2(x+y+z+a+b+c)$, we get $n_{max} \leq 2[3m-3-(x+y+z)] \leq 6m-8=(N-1)[(m-1)N-1]$.

Case 2: n_{max} is odd, and there are at least 2 non-overlapping clusters in G each of which is assigned only 1 integer. In this case, clearly we can find $\frac{n_{max}-1}{2}$ non-overlapping clusters (of size $L=2$) among which there are at least two clusters each of which is assigned only one integer. Among those $\frac{n_{max}-1}{2}$ non-overlapping clusters, let x, y, z, a, b and c respectively denote the number of clusters that are assigned only integer '1', only integer '2', only integer '3', both integers '1' and '2', both integers '2' and '3', and both integers '1' and '3'. Since the interleaving is an MCI, clearly $x+y+a \leq m-1$, $y+z+b \leq m-1$, $z+x+c \leq m-1$. So $2x+2y+2z+a+b+c \leq 3m-3$. So $x+y+z+a+b+c \leq 3m-3-(x+y+z)$. Since $x+y+z \geq 2$ and $n_{max}=2(x+y+z+a+b+c)+1$, we get $n_{max} \leq 2[3m-3-(x+y+z)]+1 \leq 6m-9 < (N-1)[(m-1)N-1]$.

Case 3: n_{max} is odd, and there don't exist 2 non-overlapping clusters in G each of which is assigned only 1 integer. Let x', y', z', a', b' and c' respectively denote the number of edges in G whose two endpoints are both assigned integer '1', are both assigned integer '2', are both assigned integer '3', are assigned integers '1' and '2', are assigned integers '2' and '3', are assigned integers '1' and '3'. (Then $x'+y'+z'+a'+b'+c'=n_{max}$.) It's simple to see that among x', y' and z', two of them equal 0, and the other one is either 1 or 2. So without loss of generality, we consider the following two sub-cases.

Sub-case 1: $x'=1$, and $y'=z'=0$. In this case, $a' \leq 2m-3$, because otherwise there will be m non-overlapping clusters in G that are assigned only integers '1' and '2'. Similarly, $c' \leq 2m-3$. Also clearly, $b' \leq 2m-2$. If $a'=2m-3$ and $c'=2m-3$, then since there don't exist m non-overlapping clusters in G that are assigned only 1 or 2 distinct integers, the MCI on G can only take the following form: in G, there are $a'=2m-3$ *consecutive* edges each of which has integers '1' and '2' assigned to its endpoints (the segment of the ring G consisting of these edges begins with a vertex that is assigned the integer '2' and ends with a vertex that is assigned the integer '1'), followed by an edge whose two endpoints both are assigned the integer '1', then followed by $c'=2m-3$ *consecutive* edges each of which has the integers

'1' and '3' assigned to its endpoints (the segment of the ring G consisting of these edges begins with a vertex that is assigned the integer '1' and ends with a vertex that is assigned the integer '3'), then followed by b' *consecutive* edges each of which has the integers '2' and '3' assigned to its endpoints (the segment of the ring G consisting of these edges begins with a vertex that is assigned the integer '3' and ends with a vertex that is assigned the integer '2') — then it's simple to see that b' can't be even, which implies that $b' < 2m-2$ here. So in any case, we have $a'+b'+c' < (2m-3)+(2m-2)+(2m-3) = 6m-8$. So $n_{max} = x'+y'+z'+a'+b'+c' < 6m-7$. So $n_{max} \leq 6m-8 = (N-1)[(m-1)N-1]$.

Sub-case 2: $x' = 2$, and $y' = z' = 0$. In this case, with arguments similar to those in sub-case 1, we get $a' \leq 2m-4$, $c' \leq 2m-4$, and $b' \leq 2m-2$. So $n_{max} = x'+y'+z'+a'+b'+c' \leq 2+(2m-4)+(2m-2)+(2m-4) = 6m-8 = (N-1)[(m-1)N-1]$.

So it's proved that in any case, $n_{max} \leq (N-1)[(m-1)N-1]$. And this lemma is proved.

□

REFERENCES

[1] K. A. S. Abdel-Ghaffar, R. J. McEliece, and H. C. A. van Tilborg. Two-dimensional burst identification codes and their use in burst correction. *IEEE Transactions on Information Theory*, 34:494–504, 1988.

[2] C. Almeida and R. Palazzo. Two-dimensional interleaving using the set partition technique. In *Proc. IEEE International Symposium on Information Theory*, page 505, Trondheim, Norway, 1994.

[3] M. Blaum, J. Bruck, and A. Vardy. Interleaving schemes for multidimensional cluster errors. *IEEE Transactions on Information Theory*, 44(2):730–743, 1998.

[4] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. ACM SIGCOMM'98*, Vancouver,Canada, 1998.

[5] J. W. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads. In *Proc. IEEE Infocom'99*, pages 275–283, Boston Univ.,MA,USA, 1999.

[6] P. Delsarte. Bilinear forms over a finite field, with applications to coding theory. *J. Combin. Theory*, 25-A:226–241, 1978.

[7] T. Etzion and A. Vardy. Two-dimensional interleaving schemes with repetitions: constructions and bounds. *IEEE Transactions on Information Theory*, 48(2):428–457, 2002.

[8] A. Jiang and J. Bruck. Diversity coloring for information storage in networks. In *Proceedings of the 2002 IEEE International Symposium on Information Theory*, page 381, Lausanne, Switzerland, 2002.

[9] A. Mahanti, D. L. Eager, M. K. Vernon, and D. Sundaram-Stukel. Scalable on-demand media streaming with packet loss recovery. In *Proc. ACM SIGCOMM 2001*, pages 97–108, San Diego, 2001.

[10] M. Naor and R. M. Roth. Optimal file sharing in distributed networks. *SIAM J. Comput.*, 24(1):158–183, 1995.

[11] P. Rodriguez and E. W. Biersack. Dynamic parallel access to replicated content in the Internet. *IEEE/ACM Transactions on Networking*, 10(4):455–465, 2002.

[12] R. M. Roth. Maximum-rank array codes and their application to crisscross error correction. *IEEE Transactions on Information Theory*, 37:328–336, 1991.