Multi-Cluster Interleaving on Paths and Cycles

Anxiao (Andrew) Jiang, *Member, IEEE*, Jehoshua Bruck, *Fellow, IEEE*

***Abstract*— Interleaving codewords is an important method not only for combatting burst-errors, but also for distributed data retrieval. This paper introduces the concept of Multi-Cluster Interleaving (MCI), a generalization of traditional interleaving problems. MCI problems for paths and cycles are studied. The following problem is solved: how to interleave integers on a path or cycle such that any m ($m \geq 2$) non-overlapping clusters of order 2 in the path or cycle have at least 3 distinct integers. We then present a scheme using a 'hierarchical-chain structure' to solve the following more general problem for paths: how to interleave integers on a path such that any m ($m \geq 2$) non-overlapping clusters of order L ($L \geq 2$) in the path have at least $L+1$ distinct integers. It is shown that the scheme solves the second interleaving problem for paths that are asymptotically as long as the longest path on which an MCI exists, and clearly, for shorter paths as well.**

***Index Terms*— Burst error, cluster, cycle, file placement, interleaving, multi-cluster interleaving, path.**

I. Introduction

Interleaving codewords is an important method for both combatting burst-errors and distributed data retrieval. Every interleaving scheme can be interpreted as labelling a graph's vertices with integers, and traditional interleaving problems all focus on *local* properties of the labelling. Specifically, if we define a *cluster* to be a connected subgraph of certain characteristics (such as size, shape, etc., depending on the specific definition of the interleaving problem), then traditional interleaving problems require that in every *single* cluster, the number of different integers exceeds a threshold, or every integer appears less than a certain number of times, etc.

Applications of interleaving in burst-error correction are well known. The most familiar example is the interleaving of codewords on a path, which has the form '$1, 2, 3, \cdots n, 1, 2, 3, \cdots n, \cdots\cdots$,' for combatting one-dimensional burst-errors of length up to n. This one-dimensional interleaving is generalized to higher dimensions in [3], [4], [5] and [7], where integers are used to label the vertices of a two-dimensional or higher-dimensional array in such a way that in every connected subgraph of order t of the array, each integer appears at most r times. (t and r here are parameters. The *order* of a graph is defined as the number of vertices in that graph.) More work on such a generalized interleaving scheme includes [10], [12], [15], [16] and [17], where the underlying graphs on which integers are interleaved include tori, arrays and circulant graphs. In [1], [2] and [3], codewords are interleaved on arrays to correct burst-errors of rectangular shapes, circular shapes, or arbitrary connected shapes.

Applications of interleaving in distributed data retrieval, 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 [6], [11]. (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 client only needs to receive 5 different components. In this example, the codeword components can be understood as interleaved on a path or a cycle.) Interleaving is also studied in the scenario of file retrieval 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 [9], [13]. (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.)

This paper introduces the concept of *Multi-Cluster Interleaving (MCI)*. In general, an MCI problem is concerned with labelling the vertices of a given graph in such a way that for any m clusters, the integers in them are sufficiently diversified (by certain criteria). Traditional interleaving problems correspond to the case $m = 1$. So MCI is a natural extension of the traditional concept of interleaving.

We focus on Multi-Cluster Interleaving on paths and cycles. In this paper, we study the following problem.

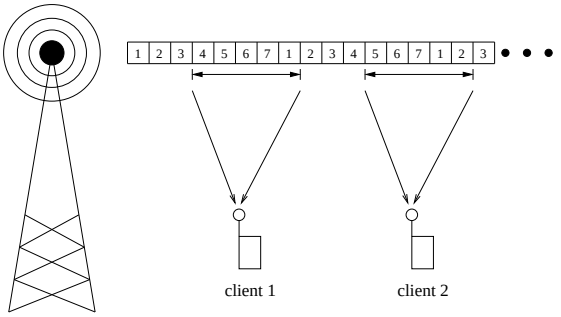
Definition 1: Let $G = (V, E)$ be a path (or cycle) of n vertices. Let N, K, m and L be positive integers such that $N \geq K > L$ and $m \geq 2$. A *cluster* is defined to be a connected subgraph of order L of the path (or cycle). Assign one integer in the set $\{1, 2, \cdots, N\}$ to each vertex. Such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

The above MCI problem is fully characterized by the five parameters — n, N, K, m, L — and the graph $G = (V, E)$. We note that throughout this paper, the parameters n, N, K, m, L and the graph $G = (V, E)$ will always have the meanings

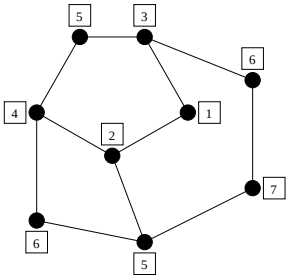
This work was supported in part by the Lee Center for Advanced Networking at the California Institute of Technology, and by NSF grant CCR-TC-0209042. The material in this paper was presented in part at the 7th International Symposium on Communication Theory and Applications, Ambleside, Lake District, UK, July 13 - 18, 2003.

A. Jiang is with the Department of Electrical Engineering, California Institute of Technology, MC 136-93, Pasadena, CA 91125, USA (e-mail: jax@paradise.caltech.edu).

J. Bruck is with the Department of Electrical Engineering, California Institute of Technology, MC 136-93, Pasadena, CA 91125, USA (e-mail: bruck@paradise.caltech.edu).



(a) Broadcast



(b) File storage in a network

Fig. 1. Examples of interleaving for data retrieval.

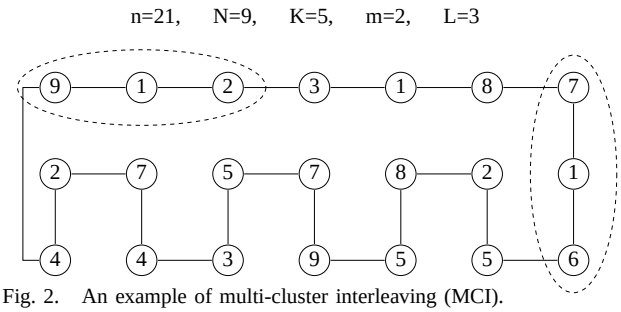


Fig. 2. An example of multi-cluster interleaving (MCI).

as defined in Definition 1.

The following is an example of the MCI problem.

Example 1: A cycle 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 non-overlapping clusters of order 3 have at least 5 distinct integers. For example, the two clusters in dashed circles have integers '9, 1, 2' and '7, 1, 6' respectively, so they together have 5 distinct integers — 1, 2, 6, 7, 9. So the interleaving is a multi-cluster interleaving on the cycle.

If we remove an edge in the cycle, then it will become a path. Clearly if all other parameters remain the same, the interleaving shown in Fig. 2 will be a multi-cluster interleaving on the path. □

Multi-Cluster Interleaving has applications in distributed data storage in networks and data retrieval by clients that are capable of accessing multiple parts of the network. The MCI problem defined in Definition 1 has the following interpretation. The N integers used to label the vertices in the path/cycle represent the N components in a codeword. K is the minimum number of components needed for decoding the codeword. (In other words, the codeword can correct $N - K$ erasures.) An interleaving of the integers represents the placement of the codeword components on the path/cycle. For each client that wants to retrieve data from the path/cycle, we assume it can access m non-overlapping clusters; and we assume different clients can access different sets of clusters. (By imposing the restriction that the m clusters a client can access must be non-overlapping, we ensure that each client can access no less than mL vertices.) Then when the interleaving is an MCI, every client can retrieve enough data for decoding the codeword.

Multi-Cluster Interleaving on paths and cycles appears to have natural applications in data-streaming and broadcast [8]. Imagine that the components of a codeword 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 [14], where we assume multiple heads can read different parts of a disk in parallel to accelerate I/O speed.

The MCI problem for paths and cycles can be divided into smaller problems based on the values of the parameters. The key results of this paper are:

- The family of problems with the constraints that $L = 2$ and $K = 3$ are solved for both paths and cycles. We show that when $L = 2$ and $K = 3$, an MCI exists on a path if and only if the number of vertices in the path is no greater than $(N-1)[(m-1)N-1]+2$, and an MCI exists on a cycle if and only if the number of vertices in the cycle is no greater than $(N-1)[(m-1)N-1]$. Structural properties of MCIs in this case are analyzed, and algorithms are presented which can output MCIs on paths and cycles as long as the MCIs exist.
- The family of problems with the constraint that $K = L+1$ are studied for paths. A scheme using a 'hierarchical-chain' structure is presented for constructing MCIs. It is shown that the scheme solves the MCI problem for paths that are asymptotically as long as the longest path on which MCIs exist, and clearly, for shorter paths as well.

The rest of the paper is organized as follows. In Section II, we derive an upper bound for the orders of paths and cycles on which MCIs exist. We then prove a tighter upper bound for paths for the case of $L = 2$ and $K = 3$. In Section III, we present an optimal construction for MCI on paths 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 paths when $K = L + 1$. In Section V we extend our results from paths to cycles. In Section VI, we conclude this paper.

II. Upper Bounds

While traditional one-dimensional interleaving exists on infinitely long paths, that is no longer true for MCI. If $K = mL$, then to get an MCI, every integer can be assigned to

only one vertex of the path/cycle, which means that MCI exists only for paths/cycles of order N or less. When K has smaller values, MCI exists for longer paths/cycles. The following proposition presents an upper bound for the orders of paths/cycles.

Proposition 1: If a Multi-Cluster Interleaving exists on a path (or cycle) of n vertices, then $n \leq (m-1)L\binom{N}{L}+(L-1)$.

Proof of Proposition 1: Let $G=(V,E)$ be a path (or cycle) of n vertices with an MCI on it. G contains at most $\lfloor \frac{n}{L} \rfloor$ non-overlapping clusters. 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 integers in S. S can be one of $\binom{N}{L}$ possible sets. So $\lfloor \frac{n}{L} \rfloor \leq (m-1)\binom{N}{L}$. So $n \leq (m-1)L\binom{N}{L}+(L-1)$. □

Note that for the same set of parameters N, K, m and L, if MCI exists on a path of $n=n_0$ vertices, then it also exists on any path of $n<n_0$ vertices. That is because given an MCI on a path, by removing vertices from the ends of the path, we can get MCIs on shorter paths. However, such an argument does not necessarily hold for cycles.

The upper bound of Proposition 1 is in fact loose. For example, when $N=K=3$ and $m=L=2$, a simple exhaustive search will show that an MCI exists on a path (or a cycle) if and only if the path (respectively, cycle) is of order 6 (respectively, 4) or less. However, Proposition 1 gives an upper bound which is $n \leq (m-1)L\binom{N}{L}+(L-1)=7$.

In the remainder of this section, we shall prove a tighter upper bound for paths for the case of $L=2$ and $K=3$, stated as the following theorem. Later study will show that this bound is exact.

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

Theorem 1 will be established by proving three lemmas below. Before starting the formal analysis, we firstly define some notations that will be used throughout this paper. Let $G=(V,E)$ be a path. We denote the n vertices in the path 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 following lemma reveals a structural property of MCI.

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 path of n vertices. Then in any MCI on a path of n_{max} vertices, no two adjacent vertices are assigned the same integer.

Proof of Lemma 1: Let $G=(V,E)$ be a path of n_{max} vertices with an MCI on it, and assume two adjacent vertices of G are assigned the same integer. We will prove that an MCI exists on a path of more than n_{max} vertices, which is a contradiction.

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 $c(v_i)=1$, $c(v_{i+1})=c(v_{i+2})=2$, $c(v_{i+3})=1$ 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 $c(v_i)=1$, $c(v_{i+1})=c(v_{i+2})=\cdots=c(v_{i+x})=2$, $c(v_{i+x+1})=1$ or 3.

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

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 path of $n_{max}+1$ vertices. Call this new path 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 path 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 they also contain either v_i or v_{i+3} — therefore they 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 path 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. So H has $n_{max}+1$ vertices and has an MCI on it, which is a contradiction.

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 path of $n_{max}+2$ vertices. Call this new path H, assign the integer '4' to v', and assign the integer '3' to v''. Consider any m non-overlapping clusters in H. If the m clusters contain neither v' nor v'', then clearly they are also m non-overlapping clusters in the path G, and therefore are assigned at least $K=3$ distinct integers. If the m clusters contain both v' and v'', then they also contain at least one vertex in the set $\{v_{i+1}, v_{i+2}, \cdots, v_{i+x-1}, v_{i+x}\}$, and therefore are assigned 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 complement set of A. Say there are y clusters in B. Then, if the cluster containing v' also contains v_{i+1} (respectively, v_{i+2}), there exists a set C of y clusters in the path 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 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 cluster containing v'. So the m clusters in $A \cup B$ are assigned at least $K = 3$ distinct integers. So H has $n_{max} + 2$ vertices and has an MCI on it, which is again a contradiction.

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 the four cases. Therefore, this lemma is proved. □

The next two lemmas derive upper bounds for paths, respectively for the case '$N \geq 4$' and the case '$N = 3$'.

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 path of n vertices. Then $n_{max} \leq (N-1)[(m-1)N - 1] + 2$.

Proof of Lemma 2: Let $G = (V, E)$ be a path 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. (See Fig. 3 for an example.)

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.

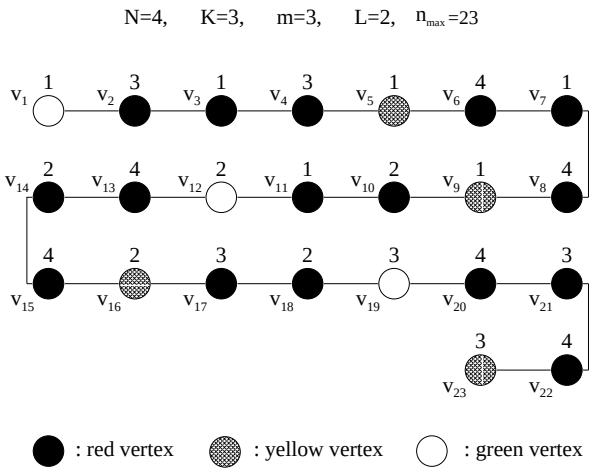


Fig. 3. In this example, $N = 4$, $K = 3$, $m = 3$, $L = 2$. An oracle tells us that $n_{max} = 23$. Let $G = (V, E)$ be the path shown in the figure, which has 23 vertices and an MCI on it. Then the vertices of G will be colored to be *red*, *yellow* and *green* as shown.

Group A: [1,3]

Group B: [1,2] , [2,3]

Group C: [1,4] , [2,4] , [3,4]

Group D: empty.

Fig. 4. Let's continue the example in Fig. 3. Then groups A, B, C, D are as shown here.

(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'.

(See Fig. 4 for an example.)

For any $1 \leq i \neq j \leq N$, let $E(i, j) \subseteq E$ denote the following 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 $E(i, j)$. (See Fig. 5 for an example.) Below we derive upper bounds for $z(i, j)$.

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 each of which is assigned only integers 'i' and 'j', which would contradict the assumption that the interleaving on G is an MCI. (See Fig. 6 for an example.)

Now consider a *pair* $[i, j]$ in group B. $z(i, j) \leq 2m - 2$ for the same reason as in the previous case. In the following, we will prove that $z(i, j) \leq 2m - 3$ by using contradiction. Assume $z(i, j) = 2m - 2$. Then in order to avoid the existence

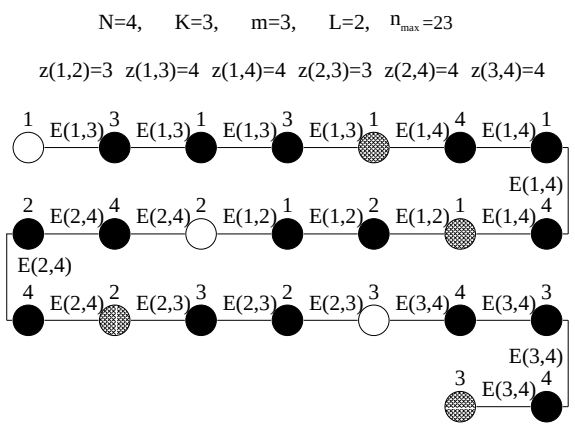


Fig. 5. Let's continue the example in Fig. 3. Then the set $E(i,j)$ that an edge belongs to is labelled beside that edge. The value of each $z(i,j)$ is shown in the figure.

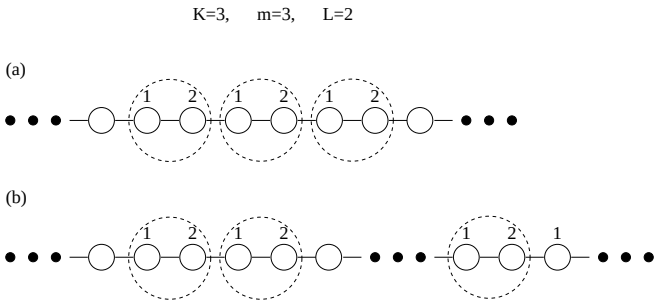


Fig. 6. In this example, $K=3$, $m=3$, $L=2$. Two paths are shown respectively in (a) and (b), each of which has more than $2m-2=4$ edges in the set $E(1,2)$. Then both of them contain $m=3$ non-overlapping clusters (as shown in dashed circles) that are assigned only two distinct integers, which proves that the interleaving on them cannot be MCI.

of m non-overlapping clusters in G that are assigned only integers 'i' and 'j', the $z(i,j)=2m-2$ edges in $E(i,j)$ must be consecutive in the path 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 $[c(v_{y+1}), c(v_{y+2}), \cdots, c(v_{y+2m-1})] = [i,j,i,j,\cdots,i,j,i]$. (See Fig. 7 for an example.)

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*. There are two possible cases:

Case 1: $k_1 < k_2$. Then the path G is interleaved as in Fig. 8 (a). We use v_{p_1}, v_{p_2}, $\cdots$, v_{p_t} to denote all the *yellow* vertices between v_{k_1} and v_{k_2}. (The other vertices between v_{k_1} and v_{k_2} are all *red*.)

By the definition of '*yellow* vertices', we can see that $c(v_{p_t}) = c(v_{p_{t-1}}) = \cdots = c(v_{p_1}) = c(v_{k_1}) = i$. Since the

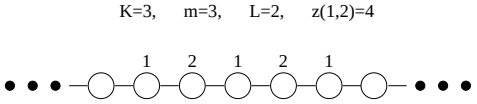


Fig. 7. In this example, $K=3$, $m=3$, $L=2$, $z(1,2)=2m-2=4$. Then for a path with an MCI on it, the 4 edges whose endpoints are labelled by '1' and '2' have to be consecutive, as shown in the figure.

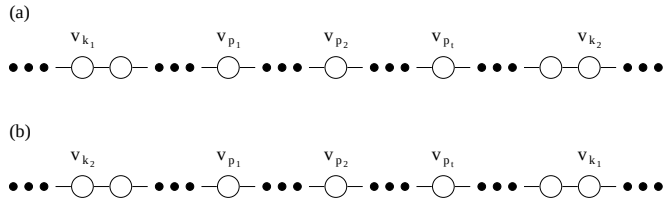


Fig. 8. (a) Case 1: $k_1 < k_2$. (b) Case 2: $k_2 < k_1$.

vertices between v_{p_t} and v_{k_2} are all *red*, and the two vertices adjacent to any *red* vertex must be assigned the same integer, we can see that $c(v_{k_2-1}) = c(v_{p_t}) = i$. 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} must be in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$. However, it is 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 be *red* — so v_{k_2} should be *red* instead of *green* — therefore a contradiction exists.

Case 2: $k_2 < k_1$. Then the path G is interleaved as in Fig. 8 (b). We use v_{p_1}, v_{p_2}, $\cdots$, v_{p_t} to denote all the *yellow* vertices between v_{k_2} and v_{k_1}. (The other vertices between v_{k_2} and v_{k_1} are all *red*.)

We can see that $c(v_{k_1-1}) = j$. 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 $\{v_{y+2}, v_{y+3}, \cdots, v_{y+2m-1}\}$. Since every vertex in the set $\{v_{y+1}, v_{y+2}, \cdots, v_{y+2m-1}\}$ that is assigned the integer 'j' must be *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 are assigned the integer 'i' must be *red* (because otherwise v_{k_1} would have to be *yellow*). Then since the color of v_{y+1} is *red*, the vertex v_y exists and it must have been assigned the integer '$c(v_{y+2}) = j$' — 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 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 is 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 any *red* vertex must be assigned the same integer, there exists an infinitely long subgraph of the path 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 is 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 is 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 have 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 path of n vertices. Then $n_{max} \leq (N - 1)[(m - 1)N - 1] + 2$.

Proof of Lemma 3: Let $G = (V, E)$ be a path of n vertices that has an MCI on it. We need to show that $n \leq (N-1)[(m-1)N-1]+2$.

If no two adjacent vertices of G 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 two adjacent vertices of G 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 on G 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. □

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

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

In this section, we present a construction for MCI on paths whose orders 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 path 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 path $G = (V, E)$ of $n = (N-1)[(m-1)N-1]+2$ vertices.

Output: An MCI on G.

Algorithm:

Let $H = (V_H, E_H)$ be a graph with parallel edges. 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, $u_{k_1} \to u_{k_2} \to \cdots \to u_{k_n}$, 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 path 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. 9 (a). The walk in H, $u_{k_1} \to u_{k_2} \to \cdots \to u_{k_n}$, can be easily found. For example, we can let the walk be $u_1 \to u_3 \to u_1 \to u_4 \to u_1 \to u_2 \to u_4 \to u_2 \to u_3 \to u_4 \to u_3$. Corresponding to that walk, we get the interleaving on G as shown in Fig. 9 (b). It can be easily verified that the interleaving is indeed an MCI. □

Theorem 2: Algorithm 1 correctly outputs a Multi-Cluster Interleaving on the path G.

Proof of Theorem 2: 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. It is not difficult to realize that the walk in H satisfying its two requirements indeed exists. For any two vertices u_i and u_j in H, there are at most $2m - 2$ edges between them, which are passed consecutively by the walk. So G has at most $2m-2$ edges whose endpoints are assigned the integers i and j, and those edges are consecutive in G. So G has at most $m-1$ non-overlapping clusters that are assigned only integers i and j. Now it is simple to see that the interleaving on G is an MCI. □

Algorithm 1 is optimal in the sense that it produces Multi-Cluster Interleaving for the longest path on which MCI exists. It is clear that the algorithm can be modified easily to produce

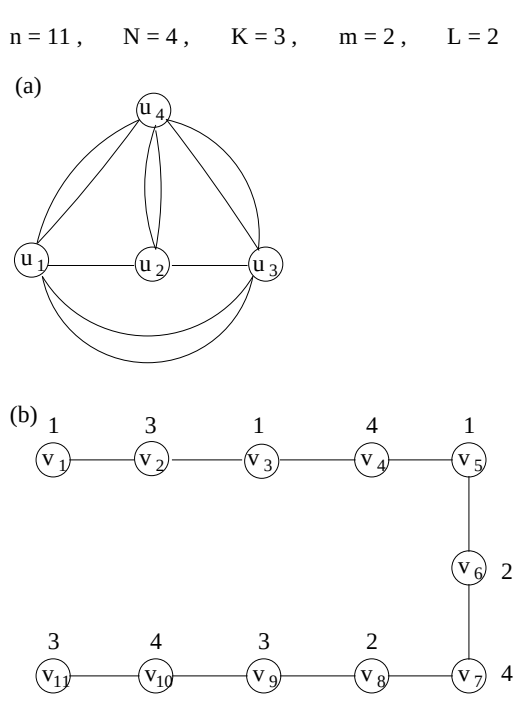


Fig. 9. (a) The graph $H = (V_H, E_H)$ (b) MCI on the path $G = (V, E)$

MCI for shorter paths as well — the method is to find shorter walks in the auxiliary graph $H = (V_H, E_H)$. We skip the details for simplicity. By Theorem 1 and Theorem 2, we find the exact condition for MCI's existence when $L = 2$ and $K = 3$, as the following theorem says.

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

IV. MCI on Paths with Constraint $K = L + 1$

In this section, we study the MCI problem for paths with a more general constraint: $K = L + 1$.

We define three operations on paths — 'remove a vertex', 'insert a vertex' and 'combine two paths'. Let G be a path 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 path $(v_1, v_2, \cdots, v_{i-1}, v_{i+1}, \cdots, v_n)$. By 'inserting a vertex $\hat{v}$' in front of the vertex v_i in G ($1 \leq i \leq n$), we get a new path $(v_1, v_2, \cdots v_{i-1}, \hat{v}, v_i, \cdots, v_n)$. Let H be a path 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 a path of $n' + n - l$ vertices whose assigned integers are in the form of $[c(u_1), c(u_2), \cdots, c(u_{n'}), c(v_{l+1}), c(v_{l+2}), \cdots, c(v_n)]$, which is the same as $[c(u_1), c(u_2), \cdots, c(u_{n'-l}), c(v_1), c(v_2), \cdots, c(v_n)]$. The following are examples of the three operations.

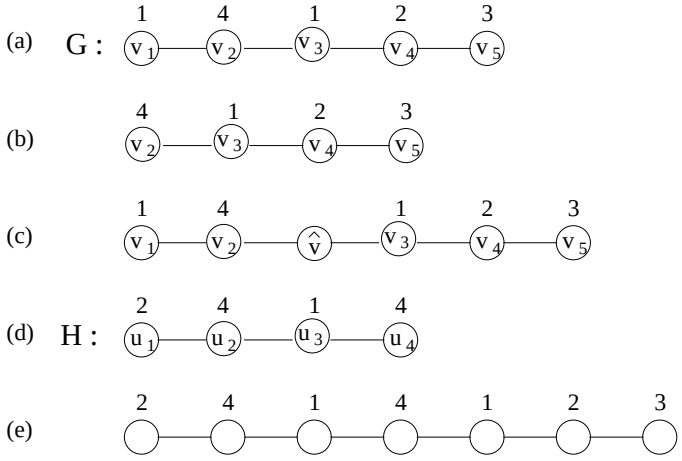


Fig. 10. Illustrations of three operations on paths.

Example 3: Let G be the path shown in Fig. 10 (a). By removing the vertex v_1 from G, we get the path shown in Fig. 10 (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 path shown in Fig. 10 (c).

Let H be the path shown in Fig. 10 (d). By combining H with G such that the last 2 vertices of H overlap the first 2 vertices of G, we get the path shown in Fig. 10 (e). □

Now we present an algorithm which computes an MCI on a path while $K = L + 1$. Being different from Algorithm 1, in this algorithm the order of the path is not preset. Instead, the algorithm tries to find a long path on which MCI exists (the longer, the better), and computes an MCI for it. Thus the output of this algorithm not only provides an MCI solution, but also gives a lower bound for the maximum order of the path on which MCI exists.

Algorithm 2: MCI on a path 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 path $G = (V, E)$.
Algorithm:

1. If $L = 2$, then let $G = (V, E)$ be a path of $(N-1)[(m-1)N-1]+2$ vertices, and use Algorithm 1 to find an MCI on G. Output G and the MCI on it, then exit. (So Step 2 and Step 3 will be executed only if $L \geq 3$.)
2. for $i = L + 1$ to N do
   { Find a path B_i (the longer, the better) 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) Any m non-overlapping connected subgraphs of B_i, each of which is of order $L-1$, are assigned at least L distinct integers;
   (3) If $i > L + 1$, then 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 path 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 (which is a path with an interleaving on it) be the path B_i.

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. (In other words, if the vertices in B_i are u_1, u_2, $\cdots$, $u_{\hat{n}}$, then after inserting vertices into B_i in the way described above, we get a new path of $\hat{n}+\lfloor\frac{\hat{n}}{L-1}\rfloor$ vertices; and if we look at the new path in the reverse order — from the last vertex to the first vertex — then the path is of the form ($u_{\hat{n}}$, $u_{\hat{n}-1}$, $\cdots$, $u_{\hat{n}+1-(L-1)}$, a new vertex, $u_{\hat{n}-(L-1)}$, $u_{\hat{n}-(L-1)-1}$, $\cdots$, $u_{\hat{n}+1-2(L-1)}$, a new vertex, $u_{\hat{n}-2(L-1)}$, $u_{\hat{n}-2(L-1)-1}$, $\cdots$, $u_{\hat{n}+1-3(L-1)}$, a new vertex, $\cdots\cdots$). In this new path, every cluster of order L contains exactly one newly inserted vertex.)

Assign the integer 'i' to every newly inserted vertex in the new path, and denote this new path by 'A_i'.

}

3. Obtain a new path by combining the paths 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 $L+1 \le i \le N$, then the new path we get has $\sum_{i=L+1}^{N} l_i - (L-1)(N-L-1)$ vertices.) Let this new path 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 a path that is the longer the better and interleave 6 integers on it, such that in the path, any 2 non-overlapping clusters are assigned at least 4 distinct integers.

Algorithm 2 firstly computes a path B_4 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 connected subgraphs of B_4 of order $L-1=2$ 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 is more than one possible outcome of Algorithm 1; 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 path B_4 is shown in Figure 11 (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 path whose assigned integers are in the form of $[4,1,3,4,1,2,4,3,2]$. We call this new path A_4. A_4 is shown in Figure 11 (b).

Algorithm 2 then computes a path B_5 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 connected subgraphs of B_5 of order $L-1=2$ 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').

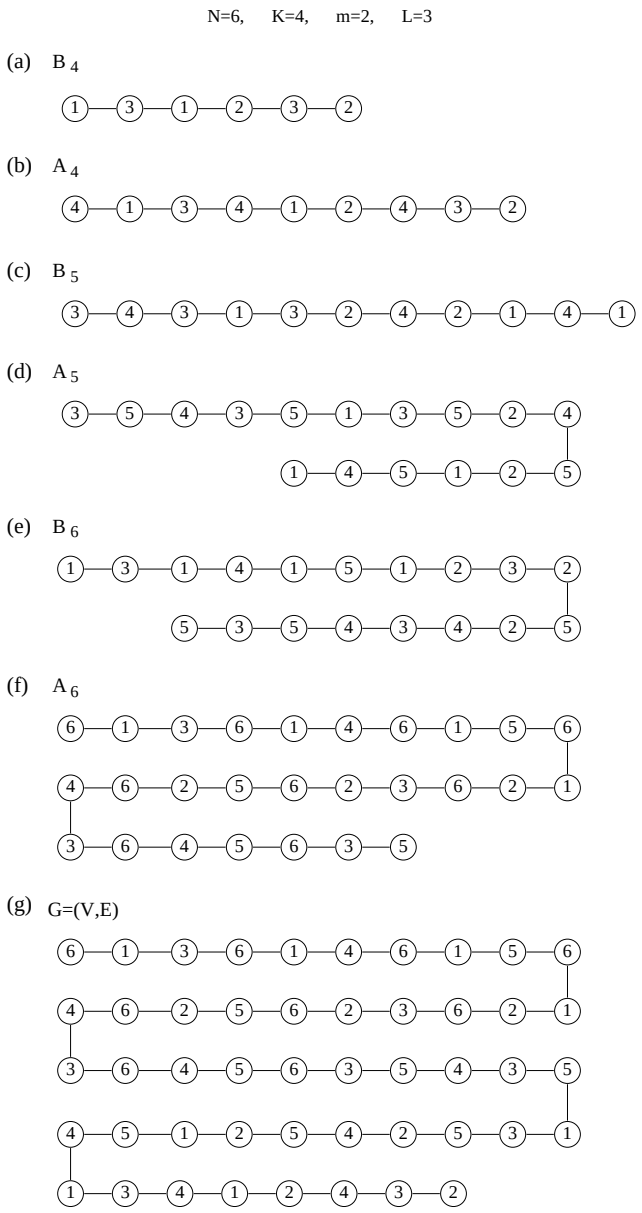


Fig. 11. An example of Algorithm 2.

Algorithm 2 computes B_5 by once again calling itself. Algorithm 2 can use the following method to find a path that satisfies all the above 3 conditions. Firstly, use Algorithm 1 to find a path that satisfies the first 2 conditions, which is easy, and call this path 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

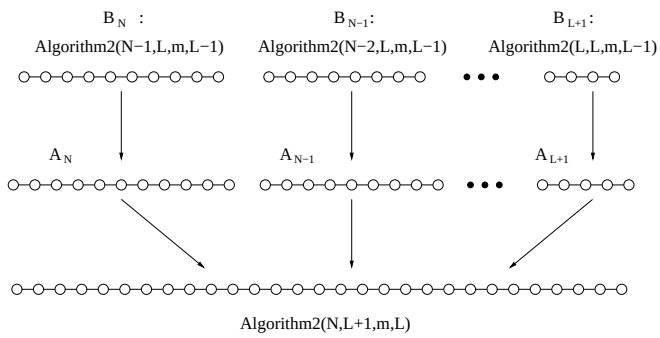


Fig. 12. Algorithm 2 has four input parameters: N, K, m and L. Let's use 'Algorithm2(a,b,c,d)' to denote the path output by Algorithm 2 when $N = a$, $K = b$, $m = c$ and $L = d$. The final output of Algorithm 2 — Algorithm2(N,L+1,m,L) — is obtained by combining the paths A_N, A_{N-1}, $\cdots$, A_{L+1}, while A_i (for $i = N, N-1, \cdots, L+1$) is obtained by inserting vertices into the path B_i. B_i is an output of Algorithm 2 as well, which is a path with an interleaving of $i-1$ different integers; specifically, B_i is Algorithm2(i-1,L,m,L-1). So from this figure, we can see the recursive structure of Algorithm 2, and the 'hierarchical-chain structure' of its output.

the names of the integers assigned to C_5, we can get a path that satisfies not only the first 2 conditions but also the 3rd condition. Call this path B_5. There is more than one 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 11 (c). Then Algorithm 2 inserts vertices into B_5 and gets a new path 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 11 (d).

Next, Algorithm 2 computes a path 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 11 (e). Then Algorithm 2 inserts vertices into B_6 and gets a new path 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 11 (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 a path $G = (V, E)$ 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 11 (g). This is the output of Algorithm 2. It can be verified that the interleaving on G is indeed an MCI. □

The path output by Algorithm 2 is a chain of the sub-paths A_{L+1}, A_{L+2}, $\cdots$, A_N. The interleavings on those paths use more and more integers, and those sub-paths are of increasing orders. In that sense, they form a 'hierarchy'. Each sub-path A_i is derived from a path B_i, and B_i is a chain of some shorter sub-paths; then, each of the sub-paths that constitute B_i is derived through the chaining of some even shorter sub-paths, and so on $\cdots\cdots$ That is another 'hierarchy'. Therefore we say that the path output by Algorithm 2 has a 'hierarchical-chain structure'. (See Fig. 12 for an illustration.)

The complexity of Algorithm 2 is dominated by the total number of vertices generated during the process of Algorithm 2's running. That number is greater than the order of the final output path $G = (V, E)$ (except when $L = 2$), because when Algorithm 2 is combining paths, there are overlapping vertices. However we can show that the total number of vertices generated is less than twice the order of $G = (V, E)$. A proof of this claim is presented in Appendix I.

Below we prove the correctness of Algorithm 2.

Theorem 4: Algorithm 2 is correct.

Proof of Theorem 4: We will 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 two facts 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 a path; and in that MCI, any i consecutive vertices are assigned i different integers. (That 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) $N'-I$ paths: 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 order 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 path A_{I+1} is constructed by inserting vertices into B_{I+1} such that any I consecutive vertices in A_{I+1} contain 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 path got by combining A_{L+1}, A_{L+2}, $\cdots$, A_N. For our discussion here, L and N should, respectively, be replaced by I and 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 connected subgraphs of order I in G, either all of them are contained in A_j for some j ($I+1 \leq j \leq N'$), or one of them is contained in $A_{j'}$ and another of them is contained in $A_{j''}$ for some $j'' \neq j'$

($I+1 \le j' \ne j'' \le N'$). In the former case, by removing those vertices that are assigned the integer 'j' in those m' subgraphs, we get m' non-overlapping connected subgraphs in B_j each of which contains $I-1$ vertices, which in total are assigned at least I different integers not including 'j' — so the m' subgraphs in G (which are also in A_j) are assigned at least $I+1$ different integers. In the latter case, WLOG, let's say $j' < j''$. Then the subgraph in $A_{j'}$ is assigned I different integers not including 'j''', and the subgraph in $A_{j''}$ is assigned an integer 'j''' — so the m' subgraphs in G are assigned at least $I+1$ different integers in total. 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 maximum order of a path for which MCI exists increases when N — the number of interleaved integers — increases. The performance of Algorithm 2 can be evaluated by the difference between the order of the path output by Algorithm 2 and the maximum order of a path for which MCI exists. We are interested in how the difference behaves when N increases.

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

Proof of Theorem 5: Let $G = (V, E)$ be a path of n vertices with an MCI on it. Then by Proposition 1, $n \le (m-1)L\binom{N}{L} + (L-1)$. So $n \le \frac{m-1}{(L-1)!}N^L + O(N^{L-1})$.

When $L = 2$, Algorithm 2 outputs a path of $(N-1)[(m-1)N-1]+2$ vertices. When $L \ge 3$, to get the output, Algorithm 2 needs to construct the paths A_{L+1}, A_{L+2}, $\cdots$, A_N; and for $L+1 \le i \le N$, A_i is got by inserting vertices into the path B_i. B_i is again an output of Algorithm 2, which is assigned $i-1$ distinct integers, and in which a considered 'cluster' is of order $L-1$. Let's use $F(N, m, L)$ to denote the number of vertices in the path output by Algorithm 2, and use $A(i, m, L)$ to denote the number of vertices in the path 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 $L \ge 3$, $F(N, m, L) = \sum_{i=L+1}^{N} A(i, m, L) - (N-L-1)(L-1)$;

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

By solving the above equations, we get $F(N, m, L) = \frac{m-1}{(L-1)!}N^L + O(N^{L-1})$, as claimed. □

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

We conclude with some numerical results. In Table 1, the order of the path output by Algorithm 2 — n — is compared with the upper bound of Proposition 1 — U_{bound} — for four different sets of parameters m and L, with $K = L+1$ throughout. The 'relative difference' in Table 1 is defined as $\frac{U_{bound}-n}{U_{bound}} = 1 - \frac{n}{U_{bound}}$. Theorem 5 shows that this relative difference approaches 0 as $N \to \infty$.

	$m=2$ and $L=3$		
N	Output of Algorithm 2 (n)	Upper bound (U_{bound})	Relative difference
10	312	362	0.1381
20	3177	3422	0.0716
50	57072	58802	0.0294
100	477897	485102	0.0149
150	1637472	1653902	0.0099
200	3910797	3940202	0.0075

	$m=2$ and $L=5$		
N	Output of Algorithm 2 (n)	Upper bound (U_{bound})	Relative difference
10	930	1264	0.2642
20	68265	77524	0.1194
50	10081020	10593804	0.0484
100	367196445	376437604	0.0245
150	2.9093×10^9	2.9580×10^9	0.0165
200	1.2521×10^{10}	1.2678×10^{10}	0.0124

	$m=5$ and $L=3$		
N	Output of Algorithm 2 (n)	Upper bound (U_{bound})	Relative difference
10	1383	1442	0.0409
20	13428	13682	0.0186
50	233463	235202	0.0074
100	1933188	1940402	0.0037
150	6599163	6615602	0.0025
200	15731388	15760802	0.0019

	$m=5$ and $L=5$		
N	Output of Algorithm 2 (n)	Upper bound (U_{bound})	Relative difference
10	4395	5044	0.1287
20	298785	310084	0.0364
50	41846205	42375204	0.0125
100	1.4964×10^9	1.5058×10^9	0.0062
150	1.1783×10^{10}	1.1832×10^{10}	0.0041
200	5.0556×10^{10}	5.0713×10^{10}	0.0031

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

V. MCI ON CYCLES

In this section, we extend our results on MCI from paths to cycles, for the case of "$L = 2$ and $K = 3$". The analysis for the two kinds of graphs bears similarity; but the 'circular' structure of the cycle leads to certain differences sometimes.

Let $G = (V, E)$ be a cycle. The following notations will be used throughout this section. We denote the n vertices in $G = (V, E)$ by v_1, v_2, $\cdots$, v_n. For $2 \le i \le n-1$, 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.

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 cycle of n vertices. Then in any MCI on a cycle of n_{max} vertices, no two adjacent vertices are assigned the same integer.

The proof of Lemma 4 is skipped because it is very similar to that of Lemma 1.

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 cycle of n vertices. Then $n_{max} \leq (N-1)[(m-1)N-1]$.

Proof of Lemma 5: This lemma can be proved in the same way as the proof for Lemma 2, except for a few small differences. For simplicity, we just point out those differences here, and skip the rest of the proof.

The first difference is that due to the 'circular' topology of the cycle G, the specific way to color the vertices of G with the *red*, *yellow* and *green* colors should be modified to be the following: "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."

The second difference is that compared to paths, for cycles there are two extra cases to consider in the proof:

Case 1: all the vertices in the cycle G are *red*. If that is true, then G must have been assigned only two distinct integers, which implies that G contains less than $mL = 2m < (N-1)[(m-1)N-1]$ vertices (since we assume the interleaving on G is an MCI).

Case 2: there is no *green* vertex in G, and all the *yellow* vertices are assigned the same integer — say it is integer 'i'. If that is true, then the integers on G must look like the following: $\{i, a, i, a, \cdots, i, a, i, b, i, b, \cdots, i, b, \cdots\cdots, i, c, i, c, \cdots, i, c\}$. For any $j \neq i$ ($1 \leq j \leq N$), there are at most $2m-2$ edges in G whose endpoints are assigned i and j respectively (because the interleaving on G is an MCI). So the order of G (which equals the number of edges in G) is at most $(N-1)(2m-2) < (N-1)[(m-1)N-1]$. □

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 cycle of n vertices. Then $n_{max} \leq (N-1)[(m-1)N-1]$.

Proof of Lemma 6: Let $G = (V, E)$ be a cycle 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 is 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 one 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 one 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 such that at least one of them is assigned only one 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 one distinct integer. In this case, clearly we can find $\frac{n_{max}-1}{2}$ non-overlapping clusters among which there are at least two clusters each of which is assigned only one distinct 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 one distinct 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 WLOG, we consider the following two sub-cases.

Sub-case 3.1: $x' = 1$, 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 one or two distinct integers, the MCI on G can only take the form described as follows. In G, there are $a' = 2m-3$ *consecutive* edges each of which has integers '2' and '1' assigned to its endpoints, which form a segment in the cycle G that begins with a vertex assigned the integer '2' and ends with a vertex assigned the integer '1'. That segment is 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, and finally followed by b' *consecutive* edges each of which has the integers '3' and '2' assigned to its endpoints, finishing the loop of edges in the cycle G. Then it is 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 3.2: $x' = 2$, and $y' = z' = 0$. In this case, with arguments similar to those in sub-case 3.1, 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 has been proved that in any case, $n_{max} \leq (N-1)[(m-1)N-1]$. □

Below we present an algorithm for generating MCIs on cycles. A distinct feature of this algorithm is that it needs to treat the cases 'n is even' and 'n is odd' somehow differently. Note that a Eulerian walk in a graph is a closed walk that passes every edge of the graph exactly once.

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

Input: A cycle $G = (V, E)$ 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 $n > (N-1)[(m-1)N-1]$, then 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 define a set S as $S = \{(1,2),(2,3),(3,4),\cdots,(N-2,N-1),(N-1,1)\}$; if $N < n \leq (N-1)[(m-1)N-1]$ and $n-(N-1)[(m-1)N-1]$ is odd, then define a set S as $S = \{(1,2),(2,3),(3,4),\cdots,(N-1,N),(N,1)\}$.
   Let $H = (V_H, E_H)$ be a graph with parallel edges that satisfies these four requirements: (1) its vertex set is $V_H = \{u_1, u_2, \cdots, u_N\}$; (2) there is no loop in H, and all the edges in H are undirected; (3) there are n edges in H; (4) for any two vertices u_i and u_j, if the un-ordered pair (i,j) belongs to the set S, then the number of edges between them is odd and is no greater than $2m-3$; otherwise, it is even and is no greater than $2m-2$.
   Find a Eulerian walk in H, $u_{k_1} \to u_{k_2} \to \cdots \to u_{k_n}$ (and finally back to u_{k_1}), that satisfies the following condition: for any two vertices, 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, then exit the algorithm. □

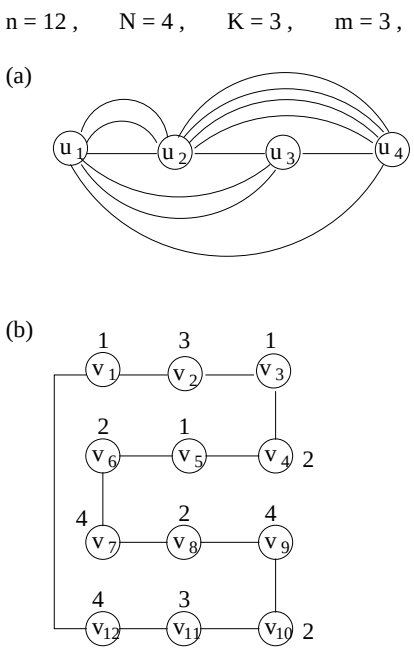


Fig. 13. (a) The graph $H = (V_H, E_H)$ (b) MCI on the cycle $G = (V, E)$

The following is an example of Algorithm 3.

Example 5: Assume $G = (V, E)$ is a cycle 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 3 is used to compute the interleaving, where the set S is defined to be $S = \{(1,2),(2,3),(3,4),(4,1)\}$. Then we can choose the graph $H = (V_H, E_H)$ to be the one shown in Fig. 13(a). We can then (easily) find the following Eulerian walk in H: $u_1 \to u_3 \to u_1 \to u_2 \to u_1 \to u_2 \to u_4 \to u_2 \to u_4 \to u_2 \to u_3 \to u_4$ (then back to u_1). Corresponding to that walk, we get the MCI as shown in Fig. 13(b). □

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

The correctness of the above theorem should be clear once the proof of Theorem 2 is understood. Now we can present the necessary and sufficient condition for MCI to exist on cycles when $L = 2$ and $K = 3$.

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

VI. Conclusion

In this paper, the Multi-Cluster Interleaving (MCI) problem for paths and cycles is studied. Compared to traditional in-

terleaving schemes, Multi-Cluster Interleaving has the distinct feature that the diversity of integers is required in multiple — instead of single — clusters. It has potential applications in data-streaming, broadcast and disk storage.

There exist many open problems in MCI. How to optimally construct MCI without the constraint that $K = L + 1$ is still unknown. Also, in the MCI problem, the path/cycle can be replaced by more general graphs. Such extensions will help bring Multi-Cluster Interleaving into practice. We hope the techniques presented here will provide insights for further study.

APPENDIX I
ON THE COMPLEXITY OF ALGORITHM 2

When Algorithm 2 runs, it generates vertices in paths. We define this more rigorously below. Algorithm 2 has three basic operations: (1) inserting vertices into an existing path to get a longer path; (2) combining two paths with overlapping vertices; (3) using Algorithm 1 to generate a path. For the first operation, we say that those vertices inserted into the existing path are newly generated vertices. For the second operation, we say that no vertex is newly generated. For the third operation, we say that all the vertices in the path output by Algorithm 1 are newly generated vertices. The complexity of Algorithm 2 is dominated by the total number of vertices generated while Algorithm 2 runs.

In this appendix, we shall prove that while Algorithm 2 runs, the total number of vertices generated is less than twice the order of the final output path $G = (V, E)$. The method is to prove the following sufficient condition: "while Algorithm 2 is running, if a vertex overlaps another vertex while the two paths they respectively belong to are combined, then those two vertices will not overlap any more vertex later on." (Namely, for any vertex in the final output path $G = (V, E)$, it is the overlapping of at most two previously generated vertices.)

The recursive structure of Algorithm 2 is illustrated in Fig. 12. Let's consider an arbitrary one of the recursions, whose corresponding input parameters are x, $y + 1$, m, y — namely, the output of this recursion is Algorithm2$(x, y + 1, m, y)$. (For the definition of $Algorithm2(a, b, c, d)$, please see Fig. 12.) The output of this recursion is denoted by 'G' in the algorithm; and if $y \geq 3$, during this recursion, a set of paths denoted by 'B_i' and 'A_i' (for different values of i) will be created. Let's first prove the following lemma.

Lemma 7: If $y \geq 3$, then B_i contains at least $2y$ vertices, A_i contains at least $2y + 2$ vertices, and G contains at least $2y + 2$ vertices.

Proof of Lemma 7: We use induction. When $y = 3$, B_i is the output of another recursion — Algorithm2$(i - 1, y = 3, m, y - 1 = 2)$. (Note that $m \geq 2$ and $i - 1 \geq y = 3$.) The path Algorithm2$(i - 1, y = 3, m, y - 1 = 2)$ is computed by calling Algorithm 1, so its order is $(i - 1 - 1)[(m - 1)(i - 1) - 1] + 2 \geq (i-2)^2 + 2 \geq 6 = 2y$. A_i is created by inserting at least $\lfloor \frac{2y}{y-1} \rfloor = 3$ vertices into B_i, so A_i contains at least $2y + 3 > 2y + 2$ vertices. G contains at least as many vertices as A_i. This serves as our base case.

Now assume the assertions of this lemma are true when $y \leq t - 1$, and let's prove them for the case $y = t$. When $y = t$, B_i is the output of another recursion — Algorithm2$(i - 1, y = t, m, y - 1 = t - 1)$; and by the induction assumption, Algorithm2$(i - 1, y = t, m, y - 1 = t - 1)$ contains at least $2(y-1)+2 = 2y$ vertices. So B_i contains at least $2y$ vertices. A_i is created by inserting at least $\lfloor \frac{2y}{y-1} \rfloor \geq 2$ vertices into B_i, so A_i contains at least $2y + 2$ vertices. G contains at least as many vertices as A_i. That concludes this proof. □

Now we can prove the "sufficient condition" mentioned in the second paragraph of this appendix. Assume in one of the recursions — whose corresponding input parameters are x, $y + 1$, m, y — two paths A_i and A_{i+1} are combined; and let u_j and $u_{j'}$ be two vertices — respectively in A_i and A_{i+1} — that overlap each other in that 'combining' operation. In that recursion, for any integer t, A_t contains at least $2y + 2$ vertices by Lemma 7; and when two paths are combined, only $y - 1$ vertices are overlapped. So u_j and $u_{j'}$ do not overlap any other vertex in that recursion, and they are neither among the first $y + 3$ vertices nor among the last $y + 3$ vertices of the path output by this recursion (which is denoted by 'G' in the algorithm). Now assume this recursion is called as a procedure by a second recursion. The second recursion (whose input parameters are *, $y + 2$, m, $y + 1$) will insert vertices into the path output by the first recursion (with one new vertex inserted after every y vertices of the path, scanning backwards) to obtain a longer path — which we shall denote by A'. So in the path A', there is at least one newly inserted vertex before u_j and $u_{j'}$ (which are now the same vertex) and at least one newly inserted vertex behind them. So u_j and $u_{j'}$ are neither among the first $y+4$ vertices nor among the last $y+4$ vertices of A'. In the second recursion, the combining of two paths will overlap only y vertices. So u_j and $u_{j'}$ will not overlap any other vertex in the second recursion. Similarly, u_j and $u_{j'}$ will not overlap any other vertex in future recursions. That concludes our proof.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor for his great diligence and the anonymous reviewers for their very helpful comments.

REFERENCES

[1] K. A. S. Abdel-Ghaffar, "Achieving the Reiger bound for burst errors using two-dimensional interleaving schemes," in *Proc. IEEE Int. Symp. Information Theory*, Ulm, Germany, 1997, pp. 425.

[2] C. Almeida and R. Palazzo, "Two-dimensional interleaving using the set partition technique," in *Proc. IEEE Int. Symp. Information Theory*, Trondheim, Norway, 1994, pp. 505.

[3] M. Blaum and J. Bruck, "Correcting two-dimensional clusters by interleaving of symbols," in *Proc. IEEE Int. Symp. Information Theory*, Trondheim, Norway, 1994, pp. 504.

[4] M. Blaum, J. Bruck and P. G. Farrell, "Two-dimensional interleaving schemes with repetitions," in *Proc. IEEE Int. Symp. Information Theory*, Ulm, Germany, 1997, pp. 342.

[5] M. Blaum, J. Bruck and A. Vardy, "Interleaving schemes for multidimensional cluster errors," *IEEE Trans. Inform. Theory*, vol. 44, no. 2, pp. 730-743, Mar. 1998.

[6] 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, Sep. 1998, pp. 56-67.

[7] T. Etzion and A. Vardy, "Two-dimensional interleaving schemes with repetitions: constructions and bounds," *IEEE Trans. Inform. Theory*, vol. 48, no. 2, pp. 428-457, Feb. 2002.
[8] K. Foltz, L. Xu and J. Bruck, "Scheduling for efficient data broadcast over two channels," in *Proc. IEEE Int. Symp. Inform. Theory*, Chicago, USA, 2004, pp. 113.
[9] A. Jiang and J. Bruck, "Diversity coloring for information storage in networks," in *Proc. IEEE Int. Symp. Inform. Theory*, Lausanne, Switzerland, 2002, pp. 381.
[10] A. Jiang, M. Cook and J. Bruck, "Optimal *t*-interleaving on tori," in *Proc. IEEE Int. Symp. Inform. Theory*, Chicago, USA, 2004, pp. 22.
[11] 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'01*, San Diego, CA, USA, Aug., 2001, pp. 97-108.
[12] Y. Merksamer and T. Etzion, "On the optimality of coloring with a lattice," in *Proc. IEEE Int. Symp. Inform. Theory*, Chicago, USA, 2004, pp. 21.
[13] M. Naor and R. M. Roth, "Optimal file sharing in distributed networks," *SIAM J. Comput.*, vol. 24, no. 1, pp. 158-183, 1995.
[14] D. A. Patterson, G. A. Gibson and R. Katz, "A case for redundant arrays of inexpensive disks," in *Proc. SIGMOD Int. Conf. Data Management*, Chicago, USA, 1988, pp. 109–116.
[15] M. Schwartz and T. Etzion, "Optimal 2-dimensional 3-dispersion lattices," *Lecture Notes in Computer Science 2643*, pp. 216–225, 2003.
[16] A. Slivkins and J. Bruck, "Interleaving schemes on circulant graphs with two offsets," accepted by *IEEE Trans. Inform. Theory*.
[17] W. Xu and S. W. Golomb, "Optimal interleaving schemes for correcting 2-d cluster errors," in *Proc. IEEE Int. Symp. Inform. Theory*, Chicago, USA, 2004, pp. 23.

Anxiao (Andrew) Jiang (S'00-M'05) received the B.S. degree with honors in 1999 from the Department of Electronic Engineering, Tsinghua University, Beijing, China, and the M.S. and Ph.D. degrees in 2000 and 2004, respectively, from the Department of Electrical Engineering, California Institute of Technology.

He was a recipient of the four-year Engineering Division Fellowship from the California Institute of Technology in 1999. His research interests include optimization, combinatorics, data storage and transmission in networks, evolution and design of complex systems, and wireless and sensor networks.

Jehoshua Bruck (S'86-M'89-SM'93-F'01) is the Gordon and Betty Moore Professor of Computation and Neural Systems and Electrical Engineering at the California Institute of Technology. He also serves as the Director of the Caltech Information Science and Technology (IST) program. His research interests include information theory, distributed systems, computation theory and biological systems. Dr. Bruck has an extensive industrial experience, including working with IBM Research for ten years. Dr. Bruck is a co-founder and Chairman of Rainfinity, a spin-off company from Caltech that is focusing on providing software for management of enterprise storage systems.

Dr. Bruck received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion, Israel Institute of Technology, in 1982 and 1985, respectively and the Ph.D. degree in Electrical Engineering from Stanford University in 1989. Dr. Bruck is the recipient of a 1997 IBM Partnership Award, a 1995 Sloan Research Fellowship, a 1994 National Science Foundation Young Investigator Award, six IBM Plateau Invention Achievement Awards, a 1992 IBM Outstanding Innovation Award, and a 1994 IBM Outstanding Technical Achievement Award for his contributions to the design and implementation of the SP-1, the first IBM scalable parallel computer. He published more than 200 journal and conference papers in his areas of interests and he holds 24 US patents.