Diversity Coloring for Distributed Storage in Mobile Networks

Anxiao (Andrew) Jiang and Jehoshua Bruck

California Institute of Technology

Abstract: Storing multiple copies of files is crucial for ensuring quality of service for data storage in mobile networks. This paper proposes a new scheme, called the K-out-of-N file distribution scheme, for the placement of files. In this scheme files are splitted, and Reed-Solomon codes or other *maximum distance seperable* (MDS) codes are used to produce file segments containing parity information. Multiple copies of the file segments are stored on gateways in the network in such a way that every gateway can retrieve enough file segments from itself and its neighbors within a certain amount of hops for reconstructing the orginal files. The goal is to minimize the maximum number of hops it takes for any gateway to get enough file segments for the file reconstruction.

We formulate the K-out-of-N file distribution scheme as a coloring problem we call *diversity coloring*. A diversity coloring is defined to be *optimal* if it uses the smallest number of colors. Upper and lower bounds on the performance of diversity coloring for general graphs are studied. Diversity coloring algorithms for several special classes of graphs—trees, rings and tori—are presented, all of which have linear time complexity. Both the algorithm for trees and the algorithm for rings output optimal diversity colorings. The algorithm for tori guarantees to output optimal diversity coloring when the sizes of tori are sufficiently large.

Index Terms: Data storage, diversity coloring, file assignment problem (FAP), graph coloring, K-out-of-N scheme, maximum distance seperable (MDS) codes, mobile computing, Quality of Service (QoS), Reed-Solomon codes, wireless network.

---

† This work was supported in part by the Lee Center for Advanced Networking at the California Institute of Technology.

I. INTRODUCTION

The Internet is not only an infrastructure for data transmission, but also an important source of information for more and more users. With the increasingly wide sharing of data among users everywhere, there is a trend to allocate files adaptively according to network structures. In mobile networks it's important to store files in such a way that mobile users (hosts) can retrieve files from the network anywhere anytime with good quality of service. Fig. 1 shows a typical model of mobile network. Gateways with wirelss or wired channels among them form the backbone of the network. Here gateways not only serve as routers but also store files. The channels between gateways and mobile hosts are wireless. A mobile host can move anywhere, contact a nearby gateway (called its access point), and ask for some files on the network. An example is shown in Fig. 1, where mobile host U asks its access point, gateway A, for a file stored on gateway C. It takes two hops to transmit the file from C to A before A forwards it to U. The network in Fig. 1 can be seen as a highly-distributed storage system for mobile users.

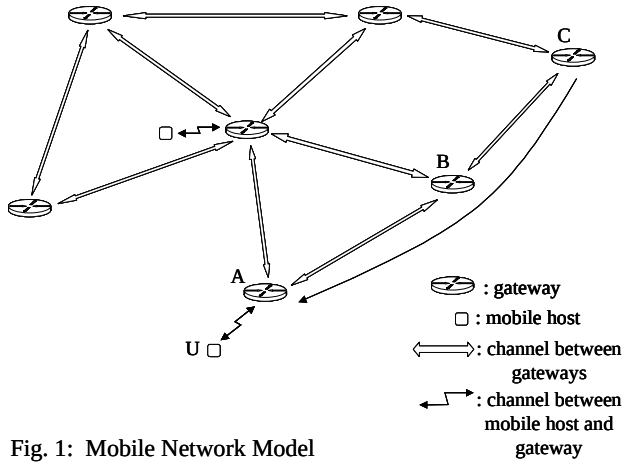


Fig. 1: Mobile Network Model

Storing multiple copies of files in different parts of the network is a classical solution to Quality of Service (QoS) for data in large networks [1] [12] [21] [22] [23] [38] [39] [41]. It reduces response time and flow, and improves the network's fault-tolerant ability, at the cost of increased memory capacity and complexity of file-consistency control.

This paper studies how to distribute files to different gateways so that no matter which gateway mobile hosts choose as access points, they can get the desired file from data stored on gateways within a certain amount of hops from the access point. Limiting the maximum amount of hops an access point takes to get the file ensures low response time (file transmission and queueing delay) effectively, especially if the channels among gateways are wireless, because the limited channel capacity, high error rate and instability of wireless channels make the number of hops in a path a most important factor in determining the delay.

Traditionally there have been two classes of techniques for solving the above problem. The first class is placing multiple copies of each file without segmentation on gateways [1] [12] [21] [22] [23] [25] [38] [39] [41]. That is a classical *multicenter resource allocation problem*, and many techniques including integer programming, approximation and combinatorial analysis have been developed for solving it [9] [11] [12] [14] [15] [18] [19] [20] [21] [25] [26] [27] [36] [44]. It also belongs to the well studied *file assignment*

problem (FAP). For a good survey of FAP, see [12].

The second class of techniques are called *file segmentation* [12] [21] [32] [38]. In this method, each file is split into segments, and multiple copies of each segment are distributed onto gateways. File segmentation relaxes some constraints on multicenter resource allocation and thus can be seen as its generalization.

In this paper we study how to combine file distribution with coding theory to achieve better performance. The scheme is named K-out-of-N file distribution. Reed-Solomn (RS) codes and other maximum-distance-seperable (MDS) codes are used. Combining coding theory with data storage on disks (e.g. RAID) has existed for a long time [2] [7] [10] [16] [17] [29] [30] [33] [34] [35], and in recent years its application on server clusters has also emerged [5] [8] [24] [37] [43], where the purpose is mainly to increase fault-tolerant capability and balance load. To combine coding theory with file distribution in networks according to network structures, however, was proposed only recently by Naor and Roth in [28], which studied how to disperse information of a file among all nodes of a network such that every node could reconstruct the original file by accessing the memory of its own and its adjacent nodes (nodes within 1 hop). Related research in this field has been very limited.

As analysed in Section V, K-out-of-N file distribution scheme is able to more effectively reduce delay and flow, balance load, and improve fault-tolerant capability compared to the *multicenter resource allocation scheme* and *file segmentation scheme*. Below we briefly introduce the concept of K-out-of-N file distribution and its abstration—the diversity coloring problem.

A. K-OUT-OF-N File Distribution Scheme

Let K and N be two positive integers, $N \geq K$. Given a file of length L, use Reed-Solomn (RS) codes or other MDS codes to encode the file and get N file segments—called segment $S_1, S_2, \cdots, S_N$ —each of which has length $\frac{L}{K}$. Because of the maximum-distance-seperable property of RS codes and other MDS codes, the file can be recovered by decoding any K of the N file segments.

Example* 1.1*: Let $K = 2$ and $N = 4$. Split a file of L bits into 4 parts—part A, B, C and D. Each part is a string of $\frac{L}{4}$ bits. See Fig. 2 (a).

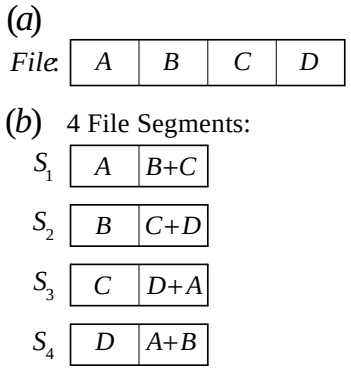


Fig. 2: (a) Split a file into 4 parts— A, B, C, D.
(b) 4 File Segments— S_1, S_2, S_3, S_4.

Now use B-code [42], a class of MDS code, to encode the 4 parts and get 4 file segments—segment S_1, S_2, S_3 and S_4, as shown in Fig. 2 (b). The operation '+' shown in Fig. 2 (b) means 'exclusive-OR'. Clearly each file segment contains $\frac{L}{2}$ bits.

The original file can be reconstructed by using any 2 of the 4 file segments. For example, suppose we have S_1 and S_3. Then since $B = (B + C) + C$, and $D = A + (D + A)$, we can recover A, B, C, D and get the original file. The decoding uses the simple exclusive-OR operation and thus is very efficient.

□

Now given a network and a file, use the above method to get N segments for the file, and distribute multiple copies of each segment onto gateways. Distribute the segments in such a way that each gateway can find at least K different segments on gateways within a certain amount of hops. That is called the K-out-of-N file distribution scheme.

RS codes and many other MDS codes have efficient decoding algorithms, and are widely used in data storage systems. For more details of RS and MDS codes, see [2] [4] [31] [40] [42].

B. Diversity Coloring

Let's abstract the K-out-of-N file distribution scheme as a graph problem. Represent the network with a graph $G(V, E)$, with each vertex representing a gateway. There exists an edge between two vertices if and only if there is a channel between the two corresponding gateways. Define $C = \{1, 2, \cdots, N\}$ to be a set of N colors, with each color in C representing one of the N file segments—S_1, S_2, $\cdots$, S_N—for the file. And we give the definition of 'Neighborhood of Radius m' below:

Definition* 1.1*: Let $G(V, E)$ be a graph, and let v be a vertex in graph G. *'The neighborhood of v of radius m'* are all the vertices that are at most m hops away from v, including v itself. Here m is a positive

integer.

Now we define the diversity coloring problem.

Definition* 1.2*: Diversity Coloring Problem

Instance: Positive integers K and m. Graph $G(V,E)$.

Question: Given N colors, how to assign one color to each vertex in graph G, such that for any vertex v in graph G, the neighborhood of v of radius m has at least K different colors? What's the smallest value for N to make such a coloring possible?

Please note that throughout this paper, K, N and m will **always** have the meaning as in Definition 1.2. The diversity coloring problem corresponds to the K-out-of-N file distribution scheme where on each gateway exactly one file segment is placed.

In diversity coloring problem, we are interested in the minimum number of colors that is necessary for diversity coloring a graph for two reasons. Firstly, in the K-out-of-N scheme, when we are using *RS* codes or other *MDS* codes to encode a file to get the N file segments, the number N cannot be infinitely large—the upper bound for N is determined by the code. So it's important to know if N—which is also the number of colors used—is no less than the minimum number of colors needed, in order to determine whether a feasible diversity coloring exists. Secondly, small N usually means low decoding complexity. For example, when RS codes are used, if N is small, we can use a code over a small field, and then for the mobile host (or its access point), when it reconstructs the original file by decoding K different file segments, the decoding complexity becomes low. Also, when systematic codes—which are so far the most popular codes employed in data storage—are used, among the N file segments, K of them are 'information segments', which are just parts of the original file and need no decoding to get. Then the proportion of file segments a mobile host (or its access point) gets that are information segments is approximately $\frac{K}{N}$, and the average number of segments that need to be got by decoding is proportional to $1 - \frac{K}{N}$. So the smaller N is, the less decoding work is needed. In order to minimize the decoding complexity, we usually want to use as few colors as possible.

Definition* 1.3*: Given graph $G(V,E)$ and parameters K and m, a diversity coloring on G is called *'optimal'* if it uses the smallest number of colors.

Below we give two examples of diversity coloring.

Example* 1.2*: Fig. 3 shows a diversity coloring on a 6×6 torus.

1	5	3	1	5	3
2	6	4	2	6	4
3	1	5	3	1	5
4	2	6	4	2	6
5	3	1	5	3	1
6	4	2	6	4	2

Fig. 3: Diversity Coloring on a 6×6 torus. K = 5, N = 6, m = 1.

It can be checked that in Fig. 3 every neighborhood of radius m contains K different colors. Two example neighborhoods in the torus are highlighted. The neighborhood in the upper-left corner contains 5 different colors—color 5, 2, 6, 4 and 1. And the neighborhood of the most down-right vertex also contains 5 different colors—color 1, 4, 2, 6 and 3. In fact with $K = 5$, $m = 1$ and $G(V,E)$ being a 6×6 torus, any diversity coloring uses at least 6 colors. Therefore the coloring in Fig. 3 is optimal.

□

Example* 1.3*: Fig. 4 shows a diversity coloring on a tree.

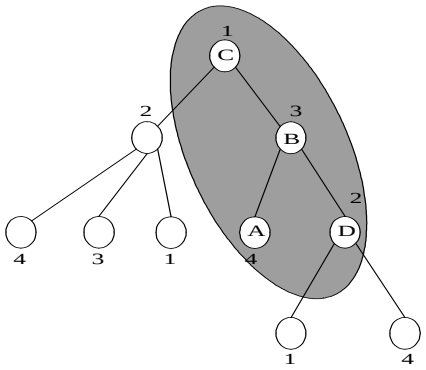


Fig. 4: Diversity Coloring on a tree. K=4, N=4, m=2.

In the tree, the neighborhood of vertex A of radius $m = 2$, which is vertex set $\{A,B,C,D\}$, has 4 colors—color 4, 3, 1 and 2. It can be checked that every neighborhood of radius 2 has 4 different colors. The coloring in Fig. 4 is also optimal.

□

The main contributions of this paper are:

♠ Properties of diversity coloring on general graphs are studied, and a tight upper bound on the minimum number of colors needed for diversity coloring is given. A fast diversity coloring algorithm for general graphs is presented.

♠ An efficient diversity coloring algorithm on trees with complexity linear in the size of the tree is presented. That algorithm can always be used to give optimal diversity coloring on trees. The necessary and sufficient condition for a tree to have a diversity coloring, as well as the exact value of the minimum number of colors needed for a diversity coloring on a tree, follow the algorithm.

♠ Diversity coloring on rings and tori is studied. For rings, the optimal coloring algorithm is given. For tori, the coloring algorithm achieves optimal performance when the tori are sufficiently large compared to the size of a neighborhood, as well as in some other cases.

The rest of the paper is organized as follows. In Section II, diversity coloring on general graphs is studied. In Section III, diversity coloring on trees is studied. Diversity coloring on rings and tori is studied in Section IV. In Section V we discuss the performance of the K-out-of-N file distribution scheme and summarize the paper.

II. Diversity Coloring On General Graphs

In this section we study diversity coloring on general graphs. First we'll define several terms that will be used extensively in this paper.

Definition* 2.1*: Given a graph $G(V,E)$ and any two vertices u, v in G, define $d^G(u,v)$ to be the distance between u and v, namely, the number of hops (or edges) in a shortest path in G connecting u and v. By convention $d^G(u,u) = 0$.

Definition* 2.2*: Given a graph $G(V,E)$ and a positive integer m, let v be a vertex in G. Denote "the neighborhood of v of radius m" by $\chi_m^G(v)$. That is, $\chi_m^G(v) = \{w | w \in V, d^G(w,v) \leq m\}$.

Definition* 2.3*: Given parameters K, N, m and graph $G(V,E)$, G is called *'colorable'* if there exists a diversity coloring on G with those parameters.

Definition* 2.4*: Given parameters K, m and graph $G(V,E)$, the minimum number of colors needed for diversity coloring G is denoted by $N_{\min}$.

The following two propositions are self-evident.

Proposition* 2.1*: Given parameters K, N, m and graph $G(V,E)$, if G is colorable then $K \leq \min_{v \in V} |\chi_m^G(v)|$. Here $|\chi_m^G(v)|$ is the cardinality of neighborhood $\chi_m^G(v)$.

Proof: If G is colorable, then every neighborhood of radius m in G should have at least K vertices in order to have K colors.

□

Proposition* 2.2*: Given parameters K, m and graph $G(V,E)$, if $K \leq \min_{v\in V} |\chi_m^G(v)|$, then there exists an integer $N_{\min}$ $(K \leq N_{\min} \leq |V|)$ s.t. G is colorable if and only if $N \geq N_{\min}$.

Proof: Clearly $N_{\min} \geq K$. If $K \leq \min_{v\in V} |\chi_m^G(v)|$, then by assigning $|V|$ different colors to the $|V|$ vertices in G, we make each neighborhood contain at least K different colors. So $N_{\min} \leq |V|$. If graph G can be diversity colored with $N = i$ colors, then G can also be diversity colored with $N = i + 1$ colors.

□

As discussed in Section I, for a diversity coloring problem, we are always interested in determining the value of $N_{\min}$, and want to color the graph with N, the number of colors used, as close to $N_{\min}$ as possible. By proposition 2.2, $N_{\min} \geq K$. Then can $N_{\min}$ always be small if K is small? The answer is negative. The following theorem shows that $N_{\min}$ is not bounded by functions of K for general graphs if $K \geq m + 2$.

Theorem* 2.1*: Let K and m be positive integers, $K \geq m + 2$. For general graphs, $N_{\min}$ can be arbitrarily greater than K.

Proof: We prove this theorem by constructing a graph $G(V,E)$ for which $N_{\min}$ can be arbitrarily greater than K.

First we consider the case $K = 3$, $m = 1$. Let K_X be a complete graph with X vertices. Call the X vertices in K_X $v_1, v_2, \cdots, v_X$. Now for any two of those X vertices, say vertex v_i and v_j $(i > j)$, add a new vertex called $u_{i,j}$ to the graph, and add two new edges: edge $(v_i, u_{i,j})$ and edge $(v_j, u_{i,j})$. There are $\binom{X}{2}$ ways to select two vertices out of those X vertices. As a result we get a graph, denoted by $G(V,E)$, with vertex set $V = \{v_i \mid 1 \leq i \leq X\} \cup \{u_{i,j} \mid 1 \leq i \leq X,\ 1 \leq j \leq X,\ i > j\}$ and edge set $E = \{(v_i, v_j) \mid 1 \leq i \leq X,\ 1 \leq j \leq X,\ i \neq j\} \cup \{(v_i, u_{i,j}),\ (v_j, u_{i,j}) \mid 1 \leq i \leq X,\ 1 \leq j \leq X,\ i > j\}$. $\forall u_{i,j} \in V$, $|\chi_m^G(u_{i,j})| = 3 = K$. $\forall v_i \in V$, $|\chi_m^G(v_i)| = 2X - 1$. Let's make $X \geq 2$ so that $\min_{w\in V} |\chi_m^G(w)| \geq K = 3$ and thus $N_{\min}$ exists. Now for any diversity coloring on G, $\forall v_i \in V$, $v_j \in V$ $(i > j)$, v_i and v_j are in neighborhood $\chi_m^G(u_{i,j})$ which contains only 3 vertices, so v_i and v_j must have different colors. Therefore all the X vertices— $v_1, v_2, \cdots, v_X$ —must have different colors, so $N_{\min} \geq X$. If we make X arbitrarily large, $N_{\min}$ also becomes arbitrarily large. So for the case $K = 3$ and $m = 1$, $N_{\min}$ can be arbitrarily greater than K for general graphs.

Now we extend the above technique to general cases. Let K and m be positive integers, $K \geq m + 2$. Let K_X be a complete graph with X vertices, X being sufficiently large. Call the X vertices in K_X $v_1, v_2, \cdots, v_X$. For every $K - m$ vertices out of those X vertices, add a linear array with m vertices, and connect those $K - m$ vertices to the same end vertex of that array. As a result we get a graph with $X + m\binom{X}{K-m}$ vertices, which we call graph $G(V,E)$. For example, if $X = 5$, $K = 4$, $m = 2$, then graph K_X and G will be as in Fig. 5:

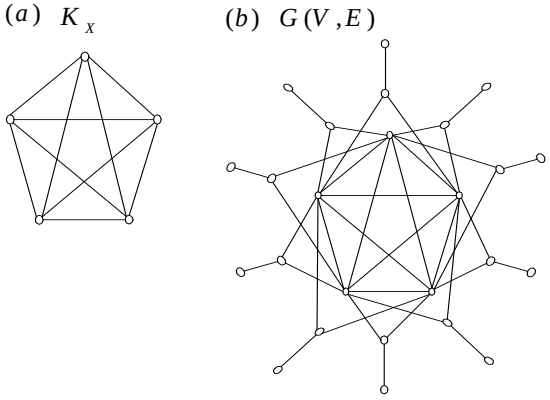


Fig. 5: Graph K_X and graph $G(V,E)$ when X=5, K=4 and m=2

The smallest neighborhoods in G are the neighborhoods of those end vertices of arrays far from vertices originally in K_X, and their cardinality is $\min_{w\in V} |\chi_m^G(w)| = K$. For any diversity coloring on G, $\forall$ $v_i \in V, v_j \in V$, $(1 \le i \le X, 1 \le j \le X, i \ne j)$, v_i and v_j are contained in at least one same neighborhood of cardinality K, so they must have different colors. Therefore all the X vertices— $v_1, v_2, \cdots, v_X$ —must have different colors. Again we have $N_{\min} \ge X$. By making X arbitrarily large, we see that $N_{\min}$ can be arbitrarily greater than K for general graphs when $K \ge m + 2$.

□

Very interestingly, when $K \le m + 1$, the conclusion becomes totally different from that in Theorem 2.1, as will be seen in Theorem 2.2. The proof of Theorem 2.2 requires results obtained in Section III, Diversity Coloring On Trees, so we leave that proof in Section III.

***Theorem* 2.2:** Let K and m be positive integers, $K \le m + 1$. For any graph $G(V,E)$, either $N_{\min} = K$, or G is not colorable—which means $\min_{v\in V} |\chi_m^G(v)| < K$.

Below we present two diversity coloring algorithms for general graphs. They reduce the diversity coloring problem to a vertex coloring problem.

Algorithm* 2.1*: Diversity coloring on general graph $G(V,E)$

Input: Graph $G(V,E)$, positive integers K, N, m.

Output: A diversity coloring on G.

Prerequisite: $\min_{v\in V} |\chi_m^G(v)| \ge K$. $N \ge K$. $N \ge \max_{v\in V} |\chi_{2m}^G(v)|$.

Algorithm 2.1.*A:* Arbitrarily select an uncolored vertex $v \in V$, assign to v a color that isn't contained in $\chi_{2m}^G(v)$. Repeat that step until all vertices in graph G are colored.

Algorithm 2.1.*B:* Let $H(V,E^H)$ be a graph with $E^H = \{(u,v) \mid u \in V,\ v \in V,\ u \neq v,\ d^G(u,v) \leq 2m\}$. Do *vertex coloring* on *H*, and let $c(v)$ denote the color of vertex *v* in H. $\forall\ v \in V$ in *G*, assign color $c(v)$ to *v*.

□

Proposition* 2.3*: Both Algorithm 2.1.A and Algorithm 2.1.B are correct.

Proof: Let's first prove the correctness of Algorithm 2.1.B. For any two vertices *u* and *v* in *V*, they are adjacent in *H* if and only if $d^G(u,v) \leq 2m$ —which means if and only if *u* and *v* are in the same neighborhood of radius *m* in *G*. For a *vertex coloring* on *H*, any two adjacent vertices in *H* have different colors. So when we do the same coloring on *G*, for any neighborhood $\chi_m^G(v)$ in *G*, all vertices in it have different colors, therefore it contains $|\chi_m^G(v)| \geq K$ different colors. So it's a diversity coloring on *G*, and Algorithm 2.1.B is correct.

Now let's look at Algorithm 2.1.A. If we let $H(V,E^H)$ have the same definition as in Algorithm 2.1.B, then Algorithm 2.1.A is simply utilizing the well known greedy algorithm of *vertex coloring*, which uses maximum degree plus 1 colors [pp. 147, [6]], to *vertex colo*r *H*—notice that the maximum degree of *H* is $\max_{v\in V} |\chi_{2m}^G(v)| - 1$. It's easy to verify that when Algorithm 2.1.A is finished, for every neighborhood of radius *m* in *G*, colors in it are all different. Now it's easy to see the coloring on *G* is a diversity coloring. Also note that there are enough colors to use in the algorithm.

□

Algorithm 2.1.A has computational complexity linear in the size of the graph, but it could use much more than $N_{\min}$ colors. It is in fact a special case of Algorithm 2.1.B. Algorithm 2.1.B could use fewer colors, but its complexity may increase.

The following theorem gives an upper bound for $N_{\min}$ for general graphs, which is better than the upper bound $|V|$ as shown in Proposition 2.2. It's especially useful for graphs whose vertex degrees are relatively 'evenly' distributed.

Theorem* 2.3*: *K* and *m* are fixed positive integers. $G(V,E)$ is a graph with $\min_{v\in V} |\chi_m^G(v)| \geq K$. Then for graph *G*, $N_{\min} \leq \max_{v\in V} |\chi_{2m}^G(v)|$. Further more, define $H(V,E^H)$ to be a graph with edge set $E^H = \{(u,v) \mid u \in V,\ v \in V,\ u \neq v,\ d^G(u,v) \leq 2m\}$. If *H* is neither a complete graph nor an odd cycle, then for graph *G*, $N_{\min} \leq \max_{v\in V} |\chi_{2m}^G(v)| - 1$.

Proof: The idea is similar to that in the proof of Proposition 2.3. Define $\Delta(H)$ to be the maximum degree of graph *H*. Then $\Delta(H) = \max_{v\in V} |\chi_{2m}^G(v)| - 1$. By the well-known greedy vertex-coloring method [6], there exists a vertex coloring on *H* using no more than $\Delta(H) + 1 = \max_{v\in V} |\chi_{2m}^G(v)|$ colors. And further more, if *H* is neither a complete graph nor an odd cycle, there exists a vertex coloring on *H* using no more than $\Delta(H)$ colors [See Theorem 3, pp. 148–149, [6]]. As shown in the proof of Proposition 2.3, any vertex coloring on *H* is a diversity coloring on *G*.

□

The following two simple examples show that the upper bound for $N_{\min}$ in Theorem 2.3 is tight.

***Example* 2.1:** Let $K = 3$, $m = 1$, $G(V,E)$ be a ring with 5 vertices. Clearly $\max_{v \in V} |\chi_{2m}^{G}(v)| = 5$, and $N_{\min} = 5$. Therefore the bound in Theorem 2.3 is met here.

□

***Example* 2.2:** Let $K = 3$, $m = 1$, $G(V,E)$ be a ring with 7 vertices. Let $E^H = \{(u,v) \mid u \in V,\ v \in V,\ u \neq v,\ d^G(u,v) \leq 2m\}$, then graph $H(V,E^H)$ is neither a complete graph nor an odd cycle. Clearly $\max_{v \in V} |\chi_{2m}^{G}(v)| - 1 = 4$. It's easy to find out that for G, $N_{\min} = 4$. Again the bound in Theorem 2.3 is met.

□

Let's end this section by discussing the relationship between diversity coloring and vertex coloring. Generally speaking, the diversity coloring problem is totally different from the traditional vertex coloring problem. The only goal of diversity coloring is to color a graph in such a way that every neighborhood of radius m in G contains no less than K different colors, and it doesn't care if two adjacent vertices have the same color or not. But in vertex coloring, the only goal is to make every pair of adjacent vertices have two different colors. However, for a special class of graphs—graphs all of whose neighborhoods have the same cardinality, if K equals the cardinality of a neighborhood, then the diversity coloring problem is equivalent to a vertex coloring problem, as Proposition 2.4 shows. The proof of Proposition 2.4 is omitted because it's similar to that of Proposition 2.3.

***Proposition* 2.4:** Let K, N and m be positive integers, $N \geq K$. $G(V,E)$ is such a graph that $\forall\, v \in V$, $|\chi_{m}^{G}(v)| = K$. $H(V,E^H)$ is a graph with $E^H = \{(u,v) \mid u \in V,\ v \in V,\ u \neq v,\ d^G(u,v) \leq 2m\}$. Define a function $c(v)$, where $v \in V$, $c(v) \in$ *color set* $\{1, 2, \cdots, N\}$. Then the function $c(v)$ is a diversity coloring on G if and only if it's a vertex coloring on H.

III. Diversity Coloring On Trees

In addition to being one of the popular forms of network structures, trees are also used by many network algorithms as spanning graphs. In this section we'll show that trees have a very special property of diversity coloring—given K, m and a tree $G(V,E)$, if $K \leq \min_{v \in V} |\chi_{m}^{G}(v)|$, then $N_{\min} = K$; otherwise G is uncolorable.

Given a tree $G(V,E)$ and parameters K, N and m, how to color the tree G? Naively, since the goal is that after coloring, every neighborhood of radius m in the tree has at least K different colors, we might want to use the following greedy algorithm:

***Naive Algorithm* 1:** Given tree $G(V,E)$ and parameters K, N and m, color vertices of the tree 'from top

to bottom', beginning with the root. Whenever a vertex $v \in V$ is to be colored, assign a color that is not in $\chi_m^G(v)$ to vertex *v*. If all colors are already in $\chi_m^G(v)$ before *v* is colored, then either color vertex *v* arbitrarily, or leave *v* uncolored and try to figure out how to color *v* later.

□

However, the Naive Algorithm 1 doesn't work. An example is shown below.

Example* 3.1:* Fig. 6 (a) shows a tree to be diversity colored with $K = N = 7$ and $m = 2$. The numbers on vertices shows the order in which vertices are colored when using the Naive Algorithm 1. Fig. 6 (b) shows the colors of vertices when the Naive Algorithm 1 is used. Clearly for the vertex labelled '2' in Fig. 6 (a), its neighborhood of radius *m* contains only 5 different colors—color 1, 2, 3, 4 and 5—in Fig. 6 (b). So the Naive Algorithm 1 failed to give a diversity coloring. Fig. 6 (c) shows the coloring on vertices by using Algorithm 3.1, the tree diversity coloring algorithm which will be presented later. Every neighborhood of radius *m* in Fig. 6 (c) contains 7 different colors, so the coloring is a diversity coloring.

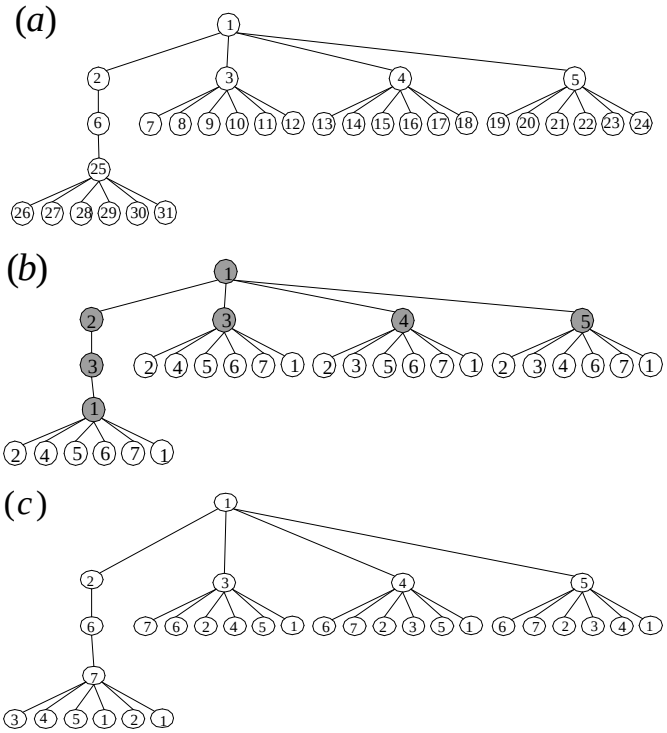


Fig. 6: Diversity coloring a tree with N=K=7, m=2. (a) The tree to be diversity colored. The numbers on vertices indicate the order in which vertices are colored when using the 'Naive Algorithm 1'. (b) The incorrect coloring by using the 'Naive Algorithm 1'. (c) The diversity coloring by using our Algorithm 3.1.

□

The reason why Naive Algorithm 1 doesn't work is that two vertices are in the same neighborhood of radius *m* if the distance between them is within 2*m*, so by only checking colors within *m* hops while doing

coloring in Navie Algorithm 1 cannot guarantee every neighborhood to have K different colors.

The above analysis about Naive Algorithm 1 naturally leads to another greedy coloring method—what if whenever a vertex v is to be colored, we assign to v a color that is not used in $\chi_{2m}^{G}(v)$, namely, the neighborhood of v of radius $2m$? However, that greedy method essentially is the same as Algorithm 2.1, and for trees such a greedy method is 'overkilling'. Readers can verify by themselves that when such a method is used to diversity color a tree, at most times either much more colors than necessary are used, or the method fails to give a diversity coloring for lack of colors when a diversity coloring actually exists.

As will be shown in this section, there is indeed a greedy diversity coloring algorithm on trees, which we will present as Algorithm 3.1. The algorithm colors vertices greedily, and when doing coloring it keeps dividing the original problem into some smaller sub-problems. So it's also a 'divide and conquer' algorithm.

First let's define some terms of the diversity coloring problem on trees. Those terms will be used frequently in this section.

Definition* 3.1*: Let $G(V,E)$ be the tree to be diversity colored. G is placed in the 'upside down' way, with its root at the top—we say the root is at the 0-th layer. The children of the root are at the 1st layer. The grandchildren of the root are at the 2nd layer. And so on. For simplicity we denote the layer of a vertex v by $L(v)$. If all the vertices in a set A are at the same layer, we denote the layer of any vertex in A by $L(A)$.

Definition* 3.2*: Let $G(V,E)$ be the tree to be diversity colored. $\forall\ u \in V$ and $v \in V$, we say "u is an ancestor of v, and v is a descendent of u" if and only if $L(u) < L(v)$ and $d^G(u,v) = L(v) - L(u)$. And we say "u is the parent of v, and v is a child of u" if and only if $L(u) = L(v) - 1$ and $d^G(u,v) = 1$.

Definition* 3.3*: Let $G(V,E)$ be the tree to be diversity colored. A 'relation structure' is a tuple of the form $(A;r;d)$, where $A = \{a_1, a_2, \cdots\}$ is a set of vertices in the tree G, r is a vertex in G and is called 'the root of the relation structure $(A;r;d)$', and d is an integer.

Definition* 3.4*: Let $G(V,E)$ be the tree to be diversity colored, and let $(A;r;d)$ be a relation structure. $\forall\ v \in V$, if v is a descendent of r, then we say 'v is at depth $d^G(v,r)$ for the relation structure $(A;r;d)$'; otherwise we say 'v is at depth $-d^G(v,r)$ for the relation structure $(A;r;d)$'.

Definiton* 3.5*: Let $G(V,E)$ be the tree to be diversity colored, and let $x = (A;r;d)$ be a relation structure. Then we define $W(x)$ and $U(x)$ as follows: $W(x) = A \cup \{v \mid v \in V;\ \exists u \in A\ \ s.t.\ \ v$ is a descendent of $u\}$, $U(x) = W(x) \cup \{v \mid v \in V;\ \exists u \in A\ \ s.t.\ \ d^G(u,v) \le m\}$.

The following is an example of the above terms.

Example* 3.2*: A tree $G(V,E)$ is shown in Fig. 7.

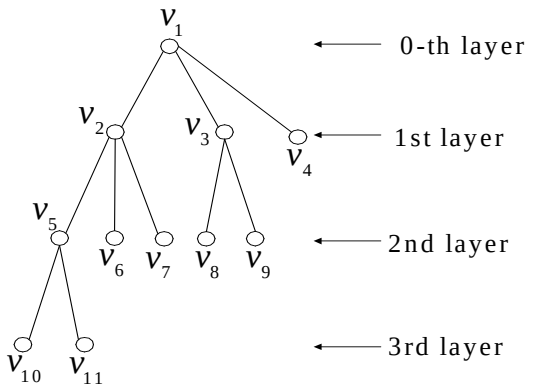


Fig. 7: A tree G(V,E)

In Fig. 7, v_1 is the root of tree G, and v_1 is at the 0-th layer. If we let $A = \{v_2, v_3\}$, then $L(v_2) = L(v_3) = L(A) = 1$. The set of descendents of v_2 are $\{v_5, v_6, v_7, v_{10}, v_{11}\}$. The set of children of v_2 are $\{v_5, v_6, v_7\}$. $x = (\{v_{10}, v_{11}\}; v_5, 0)$ is a relation structure, and v_5 is called 'the root of x'. v_{10} is at depth 1 for x, v_5 is at depth 0 for x, and v_6 is at depth -2 for x. Let $m = 3$. Then $W(x) = \{v_{10}, v_{11}\}$, $U(x) = \{v_1, v_2, v_5, v_6, v_7, v_{10}, v_{11}\}$.

□

If $K = 1$ or $K = 2$, the diversity coloring on tree $G(V,E)$ is simple. For $K = 1$, we just assign the same color to all vertices. For $K = 2$, assign color 1 to all vertices at odd layers and color 2 to all vertices at even layers. So in the algorithm on diversity coloring on trees below, we only consider the non-trivial case $K \geq 3$.

Algorithm* 3.1*: Diversity Coloring on Tree $G(V,E)$

Input: Tree $G(V,E)$, positive integers K, N, m.

Output: A diversity coloring on G.

Prerequisite: $\min_{v \in V} |\chi_m^G(v)| \geq K$. $N \geq K \geq 3$.

Algorithm:

1. Let x_0 be a relation structure ({the root of G}; the root of G; 0), $X = \{x_0\}$, $Y = \emptyset$.

2. If $X = Y = \emptyset$, arbitrarily color the uncolored vertices in G, and exit the algorithm; otherwise go to step 3.1.

3.1 If $X = \emptyset$, go to step 4.1; otherwise go to step 3.2.

3.2 Arbitrarily choose a relation structure, say $x = (A; r; d)$, in X.

Define $C_{un} = \{1, 2, \cdots N\} - \{c \mid c \text{ is a color used in } U(x)\}$.

Use colors in C_{un} to color uncolored vertices in $U(x)$ in this way: first color vertices at depth d for

x; after all vertices at depth d for x are colored, color vertices at depth $d+1$ for x; after all vertices at depth $d+1$ for x are colored, color vertices at depth $d+2$ for x; and so on• • • • • •until all the colors in C_{un} are used or all vertices in $U(x)$ are colored. Each color in C_{un} can be used at most once.

Let $A_{new} = A$, and do:

While ($A_{new} \neq \emptyset$ AND $\forall\, a \in A_{new}$, $\chi_m^G(a)$ contains no less than K different colors)

{

$A_{temp} \leftarrow \{v \mid \exists a \in A_{new}\ s.t.\ v \text{ is a child of } a\}$.

$A_{new} \leftarrow A_{temp}$.

}

If $A_{new} \neq \emptyset$, do:

{

$d_{new} \leftarrow$the depth for x at which the colors in C_{un} are used last.

If all the vertices at depth d_{new} for x are colored, let $d_{new} \leftarrow d_{new} + 1$.

If $L(A_{new}) - L(r) + d_{new} \leq m$, insert $x_{new} = (A_{new}; r; d_{new})$ into X; otherwise insert $x_{new} = (A_{new}; r; d_{new})$ into Y.

}

Delete $x = (A; r; d)$ from X, and go to step 3.1.

4.1 If $Y = \emptyset$, go to step 2; otherwise go to step 4.2.

4.2 Arbitrarily choose a relation structure, say $x = (A; r; d)$, in Y.

Let $S = \{s \mid s$ is a vertex at the $\min(\lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor, L(A))$-th layer; s is a descendent of r; s has at least 1 descendent in A or $s \in A\}$.

For every $s \in S$, do:

{

If $s \in A$, $P \leftarrow \{s\}$; otherwise $P \leftarrow \{p \mid p \in A, p$ is a descendent of $s\}$.

While ($P \neq \emptyset$ AND $\forall\, p \in P$, $\chi_m^G(p)$ contains no less than K different colors)

{

$P_{temp} \leftarrow \{v \mid \exists p \in P\ s.t.\ v$ is a child of $p\}$.

$P \leftarrow P_{temp}$.

}

If ($P \neq \emptyset$) do:

{

$d_{new} \leftarrow d - L(s) + L(r)$.

If $\forall\, v \in \{e \mid e$ is at depth d_{new} for the relation structure $(P; s; d_{new})\}$, v is colored, then $d_{new} \leftarrow d_{new} + 1$.

If $L(P) - L(s) + d_{new} \leq m$, insert $y = (P; s; d_{new})$ into X; otherwise insert $y = (P; s; d_{new})$ into Y.

}

}

Delete $x = (A; r; d)$ from Y, and go to step 4.1.

□

Below is an example of diversity coloring on tree using Algorithm 3.1.

Example* 3.3*: Diversity color the tree $G(V,E)$ in Fig. 8 (a). $K = N = 9$, $m = 3$.

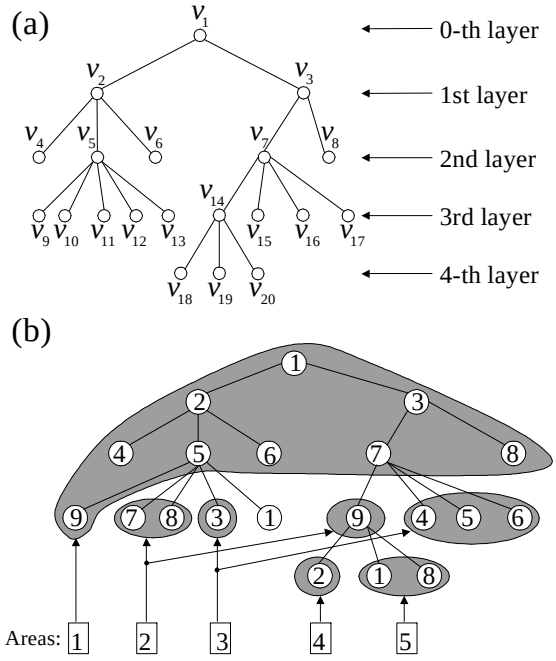


Fig. 8: (a) A tree G(V,E).
(b) Diversity coloring on tree G(V,E).
Here N=K=9, m=3.

When we color the tree G using Algorithm 3.1, first we use the 9 colors to color vertices in the tree from top to bottom. Therefore vertices $v_1, v_2, \cdots, v_9$ are assigned color 1, 2, $\cdots$, 9 respectively. (See shaded area 1 in Fig. 8 (b).) At that moment, the set of vertices whose neighborhoods of radius $m = 3$ contains $K = 9$ different colors are $\{ v_1, v_2 \}$.

To continue coloring, we derive two subtrees from the tree G. The first subtree, denoted by T_1, is the subtree with vertex set $\{ v_3, v_1, v_2, v_4, v_5, v_6, v_9, v_{10}, v_{11}, v_{12}, v_{13} \}$. The second subtree, denoted by T_2, is the subtree with vertex set $\{ v_4, v_5, v_6, v_2, v_1, v_3, v_7, v_8, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20} \}$. Notice that every vertex of G which is in both T_1 and T_2 is already colored, so T_1 and T_2 can be colored seperately in the future. What's more, for any vertex of G whose neighborhood of radius $m = 3$ contains less than $K = 9$ different colors, its neighborhood is either totally contained in subtree T_1 or totally contained in subtree T_2, and therefore the subtree containing its neighborhood has enough information to decide how to do coloring so that that neighborhood will have K different colors.

T_1 doesn't contain color 7 or 8, so we assign color 7 and 8 to uncolored vertices in T_1 "from top to bottom". So vertices v_{10} and v_{11} are assigned color 7 and 8 respectively. T_2 doesn't contain color 9, so we assign color 9 to vertex v_{14}. (See shaded area 2 in Fig. 8 (b).) Now the set of vertices whose neighborhoods of radius $m = 3$ contains $K = 9$ different colors are $\{ v_1, v_2, v_3, v_4, v_5, v_6 \}$.

Now we derive two subtrees from T_1 and T_2. The first subtree, denoted by T_3, is the subtree with vertex set $\{ v_1, v_2, v_4, v_5, v_6, v_9, v_{10}, v_{11}, v_{12}, v_{13} \}$. The second subtree, denoted by T_4, is the subtree with vertex set $\{ v_2, v_1, v_3, v_7, v_8, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20} \}$. T_3 doesn't contain color 3, so we assign color

3 to vertex v_{12}. T_4 doesn't contain color 4, 5, 6, so we assign color 4, 5, 6 to vertices v_{15}, v_{16}, v_{17}. (See shaded area 3 in Fig. 8 (b).) Then the set of vertices whose neighborhoods of radius $m = 3$ contains $K = 9$ different colors are $\{ v_1, v_2, \cdots, v_{13} \}$.

We derive a subtree, dentoed by T_5, from T_4. T_5 has vertex set $\{ v_1, v_3, v_7, v_8, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20} \}$. T_5 doesn't contain color 2, so we assign color 2 to vertex v_{18}. (See shaded area 4 in Fig. 8 (b).) Then the set of vertices whose neighborhoods of radius $m = 3$ contains $K = 9$ different colors are $\{ v_1, v_2, \cdots, v_{17} \}$.

Finally we derive a subtree, denoted by T_6, from T_5. T_6 has vertex set $\{ v_3, v_{15}, v_{16}, v_{17}, v_7, v_{14}, v_{18}, v_{19}, v_{20} \}$. T_6 doesn't color 1, 8, so we assign color 1 and 8 to vertex v_{19} and v_{20}. (See shaded area 5 in Fig. 8 (b).) At this moment, every neighborhood of radius m in G contains K different colors, even though vertex v_{13} hasn't been colored yet. We arbitrarily assign a color (e.g. color 1) to v_{13}, and get the diversity coloring shown in Fig. 8 (b).

We've narrated the process Algorithm 3.1 colored the tree $G(V, E)$. In Algorithm 3.1, each subtree (as the ones mentioned above) is characterized by a *relation structure*. There are two sets, X and Y, in Algorithm 3.1, each of which contains relation structures as its elements. Every derived subtree corresponds to a relation structure—say x—in set X, and the vertices of that subtree are the vertices in $U(x)$. Each relation structure in X satisfies some special conditions so that coloring can be done correctly on them. If a relation structure doesn't satisfies those conditions, it'll be placed in set Y, and will be 'normalized' (either 'splitted' or modified) at step 4.2 in Algorithm 3.1 before inserted into X. The real process of coloring using Algorithm 3.1 is briefly listed in *Appendix I* for better understanding of the algorithm's data structure.

□

The essence of Algorithm 3.1 is that it utilizes the fundamental property of a tree, which is there is no cycle in the tree, so neighborhoods in the tree have a good 'ordering'. Algorithm 3.1 colors the vertices in a sequence according to the 'ordering' of neighborhoods, so it can color vertices in a greedy fasion. To visualize Algorithm 3.1, let's say $W(x)$ is "the area covered by relation structure x". When the algorithm begins, after step 1 is executed, there is only one relation structure in $X \cup Y$, which is *({the root of the tree}; the root of the tree;* 0*)*, and that relation structure covers the 'whole area of the tree'. When the algorithm proceeds, for any relation structure in X or Y, the area it covers either shrinks or splits into some smaller areas, each of which is covered by a new relation structure. At no moment do any two relation structures in X or Y have any 'overlapping' in the areas they cover (which will be proved as Property 10 in Lemma 3.1), and the total area covered by relation structures in $X \cup Y$ keeps shrinking throughout the algorithm, which finally becomes nothing. For any vertex not in the area covered by some relation structure in X or Y, its neighborhood contains no less than K different colors. (That will be proved as Property 12 in Lemma 3.1.) So at the end of the algorithm all neighborhoods contain no less than K different colors, and the tree is correctly diversity colored. Algorithm 3.1 is a greedy algorithm which uses the 'divide and conquer' method. Analysis shows that the computational complexity of Algorithm 3.1 is linear in the size of the tree. So it's very efficient.

Lemma 3.1 below lists 12 properties of elements (relation structures) in set X and Y in Algorithm 3.1. They are helpful for understanding the underlying ideas of the algorithm. And they will also be used to prove the correctness of the algorithm.

Lemma* 3.1*: Suppose Algorithm 3.1 is used to diversity color a tree $G(V,E)$. Let N, K and m have their meanings as before. Algorithm 3.1 has totally 6 steps. After the complete execution of any step, elements (relation structures) in set X and Y have the following properties:

Property 1: $\forall\, x \in X$ or $x \in Y$, say $x = (A; r; d)$, then all vertices in A are at the same layer.

Property 2: (I) $\forall\, x \in X$, say $x = (A; r; d)$, then exactly one of the following two cases is true:
(1) r is a common ancestor of all the vertices in A.
(2) $A = \{r\}$.
(II) $\forall\, x \in Y$, say $x = (A; r; d)$, then only case (1) is true.

Property 3: $\forall\, x \in X$ or $x \in Y$, say $x = (A; r; d)$, then $\exists$ vertex $v \in A$ $s.t.$ $\chi_m^G(v)$ contains less than K different colors.

Property 4: $\forall\, x \in X$ or $x \in Y$, say $x = (A; r; d)$, then $d \geq 0$.

Property 5: $\forall\, x \in X$ or $x \in Y$, say $x = (A; r; d)$, then $\forall$ vertex v at depth γ for x, (1) if $\gamma < 0$ and $\exists\, u \in A$ $s.t.$ $d^G(v,u) \leq m$, then v is colored; (2) if $0 \leq \gamma < d$, then v is colored.
(3) if $\gamma > d$, then v is uncolored.

Property 6: $\forall\, x \in X$, say $x = (A; r; d)$, then $L(A) - m \leq L(r) \leq L(A)$.

Property 7: $\forall\, x \in X$, say $x = (A; r; d)$, then $L(A) - L(r) + d \leq m$.

Property 8: $\forall\, x \in Y$, say $x = (A; r; d)$, then $L(A) - L(r) + d > m$.

Property 9: $\forall\, x \in X$, say $x = (A; r; d)$, then $\forall$ vertex $a \in A$, (1) $\chi_m^G(a) \subseteq U(x)$;
(2) $\{v | v \in \chi_m^G(a)$, v is colored$\}$ $= \{v | v \in U(x)$, v is colored.$\}$.

Property 10: $\forall$ two elements x and y in set $X \cup Y$, (1) $W(x) \cap W(y) = \emptyset$; (2) all vertices in $U(x) \cap U(y)$ are colored.

Property 11: $\forall\, x \in X$ or $x \in Y$, all the colored vertices in $U(x)$ have different colors.

Property 12: $\forall$ vertex $v \in V$, if $v \notin \bigcup_{x \in X \cup Y} W(x)$, then $\chi_m^G(v)$ contains no less than K different colors.

The proof of Lemma 3.1 is presented in *Appendix II*.

Theorem* 3.1*: Algorithm 3.1 diversity colors the tree $G(V,E)$ correctly.

Proof: First let's show that Algorithm 3.1 will terminate in a finite number of steps, which is equivalent to showing that both step 3.2 and step 4.2 will be executed only finite number of times.

Say step 3.2 is to be executed, and let $x = (A; r; d)$, C_{un} and d have the same meaning as in step 3.2. By Property 3, there exists $v \in A$ $s.t.$ $\chi_m^G(v)$ contains less than K colors. By Property 9, $\chi_m^G(v) \subseteq U(x)$, and $\{u | u \in \chi_m^G(v)$, u is colored$\}$ $= \{u | u \in U(x)$, u is colored$\}$. So $U(x)$ contains less than K colors, and $|U(x)| \geq |\chi_m^G(v)| \geq K$. So $C_{un} \neq \emptyset$. By Property 11, all colored vertices in $U(x)$ have different colors, so

there are less than K colored vertices in $U(x)$. So in $U(x)$ there is no less than 1 uncolored vertex, and by Property 5 all those uncolored vertices in $U(x)$ are at depth no less than d for x. In step 3.2, colors in C_{un} are used to color vertices in $U(x)$ 'depth by depth' beginning at depth d for x, so at least one vertex is colored every time step 3.2 is executed. There are only finite vertices in tree $G(V,E)$. So step 3.2 will be executed only finite number of times in Algorithm 3.1.

By Property 10, for any two relation structures x and y in $X \cup Y$, $W(X) \cap W(Y) = \emptyset$. So $\forall v \in \bigcup_{z \in X \cup Y} W(z)$, there is a unique relation structure $z \in X$ or $z \in Y$ $s.t.$ $v \in W(z)$. Define $L_{\max} = \max_{v \in V} L(v)$. Now associate each vertex $v \in V$ with a value $R(v)$ this way: if $v \in \bigcup_{z \in X \cup Y} W(z)$, then say $z_v = (A_v; r_v; d_v)$ is the unique relation structure in X or Y $s.t.$ $v \in W(z_v)$, and we let $R(v) = L(r_v)$; if $v \notin \bigcup_{z \in X \cup Y} W(z)$, then we let $R(v) = L_{\max}$.

Say step 3.2 is to be executed, and let $x = (A; r; d)$ and $x_{new} = (A_{new}; r; d_{new})$ have the same meaning as in step 3.2. Note that either $A_{new} = A$, or every vertex in A_{new} is a descendent of some vertex in A. So $W(x_{new}) \subseteq W(x)$. For any vertex v, if $v \notin W(x)$, then $R(v)$ remains the same before and after step 3.2 is executed. If $v \in W(x) - W(x_{new})$, then $R(v) = L(r) \leq L_{\max}$ before step 3.2 is executed, and $R(v) = L_{\max}$ after step 3.2 because after step 3.2, $v \notin \bigcup_{z \in X \cup Y} W(z)$. If $v \in W(x_{new})$, $R(v) = L(r)$ both before and after step 3.2. So for any vertex v, $R(v)$ doesn't decrease after step 3.2 compared to its value before step 3.2.

Say step 4.2 is to be executed. Let $x = (A; r; d) \in Y$ have the same meaning as in step 4.2, and let $y_1 = (P_1; s_1; d_{new(1)})$, $y_2 = (P_2; s_2; d_{new(2)})$, $\cdots$, $y_n = (P_n; s_n; d_{new(n)})$ be all those relation structures inserted into X or Y at that step 4.2. Since $x \in Y$, by Property 2 $L(r) < L(A)$. We've shown in the proof of Lemma 3.1 that $L(s_1) = L(s_2) = \cdots = L(s_n) > L(r)$. And we know $W(y_i) \subseteq W(x)$ for $i = 1, 2, \cdots, n$. For any vertex v, if $v \notin W(x)$, then $R(v)$ remains the same before and after step 4.2 is executed. If $v \in W(x) - \bigcup_{1 \leq i \leq n} W(y_i)$, then $R(v) = L(r) < L_{\max}$ before step 4.2 is executed, and $R(v) = L_{\max}$ after step 4.2. If $v \in W(y_i)$, then $R(v) = L(r)$ before step 4.2, and $R(v) = L(s_i) > L(r)$ after step 4.2. So $\forall v \notin W(x)$, $R(v)$ remains the same before and after step 4.2; $\forall v \in W(x)$, $R(v)$ strictly increases after step 4.2 compared to its value before step 4.2—and note that $W(x) \neq \emptyset$. Since step 4.2 increases the value of $R(v)$ for some vertices v, and no step in Algorithm 3.1 decreases the value of $R(v)$ for any vertex v, and for any vertex v the value $R(v)$ has an upper bound $L_{\max}$, so step 4.2 will be executed only finite number of times in Algorithm 3.1.

So Algorithm 3.1 will terminate in finite number of steps, and the last step executed in the algorithm is step 2, when $X = Y = \emptyset$. $X = Y = \emptyset$ means that for any vertex v in the tree $G(V,E)$, $v \notin \bigcup_{x \in X \cup Y} W(x)$, and by Property 12 $\chi_m^G(v)$ contains no less than K different colors. So every neighborhood of radius m in the tree $G(V,E)$ contains no less than K different colors when Algorithm 3.1 terminates. So the tree $G(V,E)$ is correctly diversity colored by Algorithm 3.1.

□

Notice that in Algorithm 3.1, we can always let N be equal to K and diversity color the tree with only K

colors. Since $N_{\min} \geq K$ for any diversity coloring, we have the following two nice corollaries:

Corollary* 3.1*: Given positive integers K, m and tree $G(V,E)$, if $K \leq \min_{v \in V} |\chi_m^G(v)|$, then $N_{\min} = K$. If $K > \min_{v \in V} |\chi_m^G(v)|$, then the tree is not colorable.

Corollary* 3.2*: Algorithm 3.1 outputs optimal diversity coloring on tree when we let $N = K$.

At the end of this section, we present the proof of Theorem 2.2, which is firstly mentioned in Section II.

Theorem* 2.2*: Let K and m be positive integers, $K \leq m + 1$. For any graph $G(V,E)$, either $N_{\min} = K$, or G is not colorable—which means $\min_{v \in V} |\chi_m^G(v)| < K$.

Proof: First consider the case where G is a connected graph. If $\min_{v \in V} |\chi_m^G(v)| < K$, then by Proposition 2.1 G is not colorable. Now suppose $\min_{v \in V} |\chi_m^G(v)| \geq K$. Let $T(V,E^T)$ be a spanning tree of G. It's not hard to see that for any $v \in V$, $|\chi_m^T(v)| \geq \min(m+1, |V|) \geq K$. By Corollary 3.1, we can color T using K colors such that $\chi_m^T(v)$ contains at least K different colors for any $v \in V$. Let's apply the same coloring to G, then since $\chi_m^T(v) \subseteq \chi_m^G(v)$ for any $v \in V$, each neighborhood in G also contains at least K different colors, so the coloring is a diversity coloring on G. So $N_{\min} = K$.

If G is not connected, apply the above proof to each component of G, and the theorem still holds. □

IV. Diversity Coloring On Rings And Tori

In this section we study diversity coloring problems on rings and tori. Rings are common network structures, and they are also often embedded into other networks to realize some functions. Tori are very popular network structures in parallel and distributed systems.

A. Diversity Coloring on Rings

Definition* 4.1*: A ring is a graph $G(V,E)$ with vertex set $V = \{v_0, v_1, \cdots, v_{|V|-1}\}$ and edge set $E = \{(v_i, v_{(i+1) \bmod |V|}) | \ i = 0, 1, \cdots, |V| - 1\}$.

In a ring with no less than $2m + 1$ vertices, each neighborhood of radius m has exactly $2m + 1$ vertices. In this sub-Section A, we generalize the concept of a neighborhood in a ring such that the number of vertices in a neighborhood can be not only odd, but also even. And we'll see that the results we'll get in this sub-section applies equally well to both cases.

Definition* 4.2*: A neighborhood in a ring $G(V,E)$ is a set of L consecutive vertices in ring G: $\{v_{i \bmod |V|}, v_{(i+1) \bmod |V|}, \cdots, v_{(i+L-1) \bmod |V|}\}$. Here L is a given positive integer. Notice that when L is odd, then

the neighborhood corresponds to the 'neighborhood of radius $\frac{L-1}{2}$' as defined before.

Since in a ring all neighborhoods have the same size, we usually want to color the ring such that for any neighborhood, all vertices in it have different colors—that is, let $K = L$—because that kind of coloring corresponds to the K-out-of-N file distribution scheme where each gateway stores a file segment of length $\frac{1}{L}$ of the original file, which is the minimum possible memory requirement. In this sub-section we'll always let $K = L$. And the parameter L will always have the same meaning as in Definition 4.2.

Below we present an algorithm for diversity coloring a ring. It'll be shown later that this algorithm always outputs optimal diversity coloring on rings.

Algorithm* 4.1*: Diversity Coloring on Ring $G(V,E)$

Input: Ring $G(V,E)$, positive integers K, N, L.

Output: A diversity coloring on G.

Prerequisite: $|V| \geq L$. $K = L$. $N = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$.

Algorithm:

1. Let $x = |V| + \lfloor \frac{|V|}{K} \rfloor - N \cdot \lfloor \frac{|V|}{K} \rfloor$, $y = N \cdot \lfloor \frac{|V|}{K} \rfloor - |V|$.
2. For $0 \leq i \leq x-1$, $0 \leq j \leq N-1$, assign color $j+1$ to vertex v_{iN+j}.
3. For $0 \leq i \leq y-1$, $0 \leq j \leq N-2$, assign color $j+1$ to vertex $v_{xN+i(N-1)+j}$.

□

Algorithm 4.1 has complexity $\Theta(|V|)$. The following theorem proves the correctness of Algorithm 4.1.

Theorem* 4.1*: Algorithm 4.1 diversity colors ring $G(V,E)$ correctly.

Proof: Algorithm 4.1 colors each of the $xN + y(N-1) = (K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil)[|V| + \lfloor \frac{|V|}{K} \rfloor - (K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil)\lfloor \frac{|V|}{K} \rfloor] + (K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil - 1)[(K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil)\lfloor \frac{|V|}{K} \rfloor - |V|] = |V|$ vertices in ring G once and only once. It's simple to verify that $y \geq 0$. If $y = 0$, then the distance between any two vertices of the same color is at least $N = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil \geq K = L$. If $y > 0$, then $N \cdot \lfloor \frac{|V|}{K} \rfloor > |V|$ and therefore $\lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil \geq 1$, so the distance between any two vertices of the same color is at least $N - 1 \geq K = L$. So the distance of any two vertices of the same color is no less than L anyway. Therefore any neighborhood in the ring contains $L = K$ different colors. So the coloring on the ring is a diversity coloring.

□

Theorem* 4.2*: Given ring $G(V,E)$ and parameters K and L, $(K = L \leq |V|)$, $N_{\min} = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$.

Proof: Algorithm 3.1 diversity colors ring G with $N = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$ colors, so $N_{\min} \leq K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$.

Now suppose we have a diversity coloring on the ring. Since any neighborhood contains $K = L$ different colors, the distance between any two vertices of the same color is at least K. So for any color in the ring, there are at most $\lfloor \frac{|V|}{K} \rfloor$ vertices of that color. Therefore there are at least

$\lceil \frac{|V|}{\lfloor \frac{|V|}{K} \rfloor} \rceil = \lceil \frac{K \cdot \lfloor \frac{|V|}{K} \rfloor + (|V| \bmod K)}{\lfloor \frac{|V|}{K} \rfloor} \rceil = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$ different colors in the ring. So $N_{\min} \geq K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$. Therefore we get $N_{\min} = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil$.

□

The following three corollaries describe the relationship between $N_{\min}$ and $|V|$—the number of vertices in the ring $G(V,E)$.

***Corollary* 4.1:** $K \leq N_{\min} \leq 2K-1$ for any ring $G(V,E)$ and parameters K and L. (Here $K = L \leq |V|$).

Proof: $0 \leq \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil \leq \lceil \frac{K-1}{\lfloor \frac{|V|}{K} \rfloor} \rceil \leq K-1$, so by theorem 4.2 $K \leq N_{\min} \leq 2K-1$.

□

***Corollary* 4.2:** Given ring $G(V,E)$ and parameters K, N and L, ($|V| \geq K$), if $N = K = L$, then ring G is colorable if and only if $|V|$ is multiples of K.

Proof: Apply Theorem 4.2 directly.

□

***Corollary* 4.3:** Let $G(V,E)$ be a ring, $K = L$ and $N > K$. If $|V| \geq K\lceil \frac{K-1}{N-K} \rceil$, then ring G is colorable. If $|V| = K\lceil \frac{K-1}{N-K} \rceil - 1$, then ring G is not colorable.

Proof: If $|V| \geq K\lceil \frac{K-1}{N-K} \rceil$, then $N_{\min} = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil \leq K + \lceil \frac{K-1}{\lceil \frac{K-1}{N-K} \rceil} \rceil \leq N$, so ring G is colorable. If $|V| = K\lceil \frac{K-1}{N-K} \rceil - 1$, then $N_{\min} = K + \lceil \frac{|V| \bmod K}{\lfloor \frac{|V|}{K} \rfloor} \rceil = K + \lceil \frac{K-1}{\lceil \frac{K-1}{N-K} \rceil - 1} \rceil > K + (N-K) = N$, so ring G is not colorable.

□

The following example shows the value of $N_{\min}$ as a function of $|V|$. It 'visualizes' the three corollaries above.

***Example* 4.1:** Let $G(V,E)$ be a ring. Fig. 9 plots $N_{\min}$ as a function of $|V|$ for the problem of diversity coloring ring G, when $K = L = 6$.

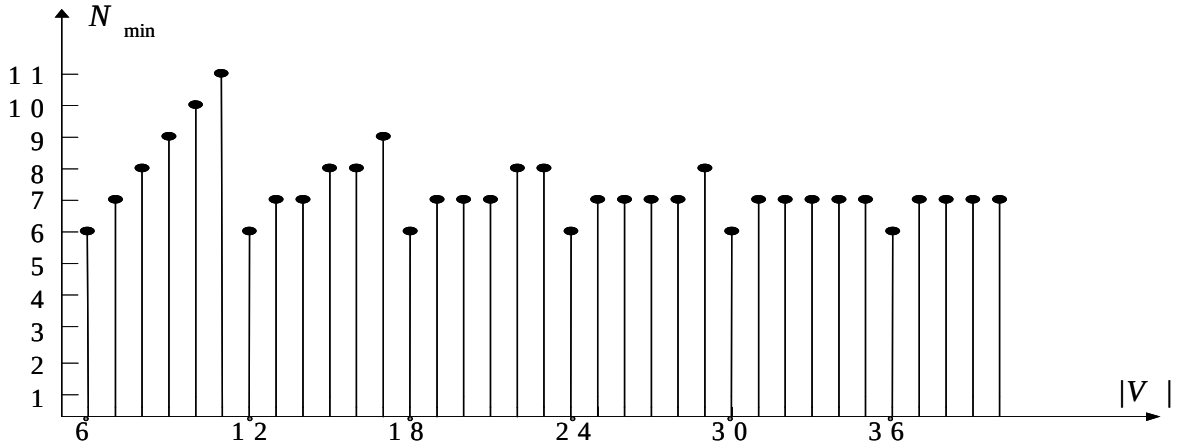


Fig. 9: $N_{\min}$ v.s. $|V|$, for the problem of diversity coloring ring G(V,E) with K=L=6.

□

We see that when L, the length of the ring, increases, $N_{\min}$ gets into a smaller and smaller range around K. For reasons discussed in Section I, the average decoding complexity usually gets low when N is close to K. What's more, from Corollary 4.3, we see that when ring $G(V,E)$ has no less than $K(K-1)$ vertices, at most $K+1$ colors are needed to diversity color the ring. That property is useful because when $N = K+1$, for the K-out-of-N file distribution scheme, K file segments can be got by simply splitting the file into K parts, and the extra file segment can just be the exclusive-OR of the other K segments. Then for the decoding, only the exclusive-OR operation is needed, and that makes the decoding very simple. Similarly when ring $G(V,E)$ has no less than $K\lceil\frac{K-1}{2}\rceil$ vertices, at most $K+2$ colors are needed to diversity color the ring. When $N \leq K+2$ we can use the EVENODD code [2] for the K-out-of-N file distribution scheme, and the decoding of EVENODD code also uses only the exclusive-OR operation. So when the size of the ring is sufficiently large, the decoding work of the K-out-of-N file distribution scheme has very low complexity.

B. Diversity Coloring on Tori

Definition* 4.3*: An $X \times X$ torus is a graph $G(V,E)$ with vertex set $V = \{v_{i,j} \mid i = 0,1,\cdots,X-1; j = 0,1,\cdots,X-1\}$ and edge set $E = \{(v_{i,j}, v_{(i+1) \bmod X,j}),\ (v_{i,j}, v_{i,(j+1) \bmod X}) \mid i = 0,1,\cdots,X-1; j = 0,1,\cdots,X-1\}$.

Example* 4.2*: Fig. 10 shows a 3 × 3 torus. Note that in this paper, vertices in a torus will **always** be labelled the same way as in Fig. 10.

$v_{0,0}$	$v_{0,1}$	$v_{0,2}$
$v_{1,0}$	$v_{1,1}$	$v_{1,2}$
$v_{2,0}$	$v_{2,1}$	$v_{2,2}$

Fig. 10: A 3 × 3 Torus

□

In an $X \times X$ torus ($X \geq 2m+1$), each neighborhood of radius m contains $1 + \sum_{i=1}^{m} 4i = 2m^2 + 2m + 1$ vertices. For the same reason as discussed in sub-Section A, in this section we always let $K = 2m^2 + 2m + 1$. Actually when $K = 2m^2 + 2m + 1$, a diversity coloring on tori ensures that any two vertices of the same color have distance at least $2m+1$ between them, and therefore the diversity coloring problem for tori falls into the category of 2-*dimensional interleaving scheme* as studied in [3].

In [3], Blaum *et al.* defines a graph to be *t-interleaved* if the distance between any two vertices of the same color is no less than *t*. (So here $2m+1$ corresponds to *t*.) Their results, if applied to $X \times X$ tori, solves the case where X is multiples of K. The algorithm we'll present in this section gives the solution to general cases.

Before we present the algorithm for diversity coloring on tori, note that the main feature of the algorithm is its 'shift property'—for any vertex $v_{i,j}$ in the torus, vertex $v_{(i+m+1) \bmod X, (j+m) \bmod X}$ has the same color as $v_{i,j}$. As will be shown in Corollary 4.6, the algorithm guarantees to output optimal diversity coloring when the size of the torus is sufficiently large compared to K, and also in some other cases.

Algorithm 4.2: Diversity Coloring on Torus $G(V,E)$

Input: An $X \times X$ torus $G(V,E)$, positive integers K, N, m.
Output: A diversity coloring on G.
Prerequisite: $X \geq 2m^2 + 2m + 1$. $K = 2m^2 + 2m + 1$. $N = K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$.

Algorithm:

1. Let $\alpha = \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$, $\beta = \lfloor \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rfloor$, $x = X \bmod (\beta + K)$, $y = \lfloor \frac{X}{\beta + K} \rfloor - x$, $x_0 = x(\alpha - m)$, $y_0 = x(\alpha + m + 1)$.
   Let $f : \{(\triangle_x, \triangle_y) | \triangle_x$ *and* $\triangle_y$ *are integers,* $|\triangle_x| + |\triangle_y| \leq m\} \rightarrow \{1, 2, \cdots, 2m^2 + 2m + 1\}$ be any bijective mapping.
2. *for* $0 \leq i \leq x - 1$
   {
   for $-m \leq \triangle_x \leq m$
   for $-m + |\triangle_x| \leq \triangle_y \leq m - |\triangle_x|$
   for $0 \leq j \leq X - 1$
   {
   $p = [i(\alpha - m) + \triangle_x + j(m+1)] \bmod X$;
   $q = [i(\alpha + m + 1) + \triangle_y + jm] \bmod X$;
   assign color $f(\triangle_x, \triangle_y)$ to vertex $v_{p,q}$.
   }
   for $1 \leq j \leq \alpha$
   for $0 \leq l \leq X - 1$
   {
   $p = [i(\alpha - m) + (j-1) \bmod (m+1) + 1 + l(m+1)] \bmod X$;
   $q = [i(\alpha + m + 1) + (j-1) \bmod (m+1) + \lceil \frac{j}{m+1} \rceil + m + lm] \bmod X$;

assign color $(2m^2 + 2m + 1) + j$ to vertex $v_{p,q}$.

}

}

for $0 \le i \le y - 1$

{

for $-m \le \triangle_x \le m$

for $-m + |\triangle_x| \le \triangle_y \le m - |\triangle_x|$

for $0 \le j \le X - 1$

{

$p = [x_0 + i(\beta - m) + \triangle_x + j(m + 1)] \bmod X$;

$q = [y_0 + i(\beta + m + 1) + \triangle_y + jm] \bmod X$;

assign color $f(\triangle_x, \triangle_y)$ to vertex $v_{p,q}$.

}

for $1 \le j \le \beta$

for $0 \le l \le X - 1$

{

$p = [x_0 + i(\beta - m) + (j - 1) \bmod (m + 1) + 1 + l(m + 1)] \bmod X$;

$q = [y_0 + i(\beta + m + 1) + (j - 1) \bmod (m + 1) + \lceil \frac{j}{m+1} \rceil + m + lm] \bmod X$;

assign color $(2m^2 + 2m + 1) + j$ to vertex $v_{p,q}$.

}

}

□

Algorithm 4.2 directly computes the color assigned to each vertex. It's computational complexity is $\Theta(|V|)$.

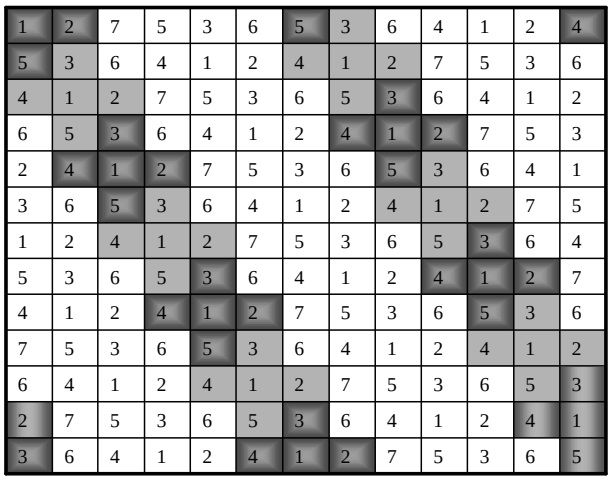
Below we give an example of diversity coloring a torus using Algorithm 4.2. The example reveals the ideas underlying Algorithm 4.2.

***Example* 4.2:** Let $G(V, E)$ be an $X \times X$ torus with $X = 13$. Let $m = 1$, $K = 5$, $N = 7$. We use Algorithm 4.2 to diversity color G, and let the function f in the algorithm be defined this way:

$(\triangle_x, \triangle_y)$	$(0,0)$	$(0,1)$	$(-1,0)$	$(0,-1)$	$(1,0)$
$f(\triangle_x, \triangle_y)$	1	2	3	4	5

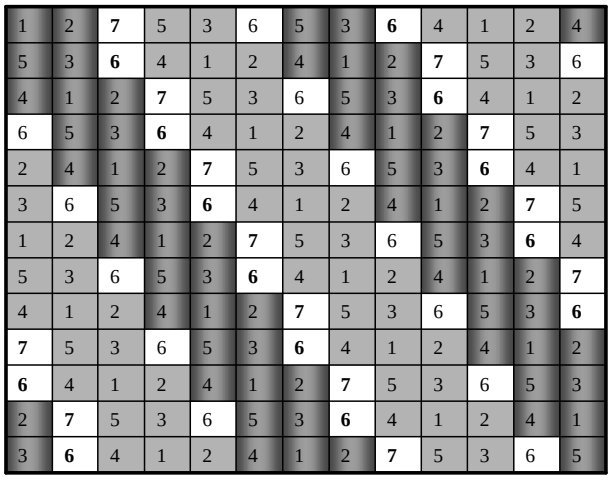
The final coloring is shown in Fig. 11 (both (a) and (b)).

(a) A “ring” consisting of connected neighborhoods



1	2	7	5	3	6	5	3	6	4	1	2	4
5	3	6	4	1	2	4	1	2	7	5	3	6
4	1	2	7	5	3	6	5	3	6	4	1	2
6	5	3	6	4	1	2	4	1	2	7	5	3
2	4	1	2	7	5	3	6	5	3	6	4	1
3	6	5	3	6	4	1	2	4	1	2	7	5
1	2	4	1	2	7	5	3	6	5	3	6	4
5	3	6	5	3	6	4	1	2	4	1	2	7
4	1	2	4	1	2	7	5	3	6	5	3	6
7	5	3	6	5	3	6	4	1	2	4	1	2
6	4	1	2	4	1	2	7	5	3	6	5	3
2	7	5	3	6	5	3	6	4	1	2	4	1
3	6	4	1	2	4	1	2	7	5	3	6	5

(b) Two “rings” and two “gaps”



1	2	**7**	5	3	6	5	3	**6**	4	1	2	4
5	3	**6**	4	1	2	4	1	2	**7**	5	3	6
4	1	2	**7**	5	3	6	5	3	**6**	4	1	2
6	5	3	**6**	4	1	2	4	1	2	**7**	5	3
2	4	1	2	**7**	5	3	6	5	3	**6**	4	1
3	6	5	3	**6**	4	1	2	4	1	2	**7**	5
1	2	4	1	2	**7**	5	3	6	5	3	**6**	4
5	3	6	5	3	**6**	4	1	2	4	1	2	**7**
4	1	2	4	1	2	**7**	5	3	6	5	3	**6**
7	5	3	6	5	3	**6**	4	1	2	4	1	2
6	4	1	2	4	1	2	**7**	5	3	6	5	3
2	**7**	5	3	6	5	3	**6**	4	1	2	4	1
3	**6**	4	1	2	4	1	2	**7**	5	3	6	5

Fig. 11: Diversity coloring on a 13×13 torus.
m=1, K=5, N=7.

In Fig. 11 (a), 13 neighborhoods of radius $m = 1$ are shaded. Each shaded neighborhood contains $K = 5$ colors—color 1, 2, 3, 4 and 5. And the relative positions of the 5 colors in those neighborhoods are all the same. Those 13 neighborhoods connect to each other and form a “ring” in the torus. In Fig. 11 (b) two such “rings” are shown. (One “ring” is of dark shade and the other “ring” is of light shade.) If we see the torus as a two-dimensional linear space, then the two “rings” partition the space and leave two “gaps” between them—we filled one “gap” with color 7 and 6, and filled the other “gap” with color 6. Now it becomes clear why the coloring in Fig. 11 is a diversity coloring—for any two vertices of the same color in the torus, it’s easy to verify that the two neighborhoods of them of radius 1 must be disjoint.

The above ideas actually apply to general cases. When Algorithm 4.2 is used to color an $X \times X$ torus,

there are $X \cdot \lfloor \frac{X}{K} \rfloor$ disjoint neighborhoods each of which contains K colors—color 1, 2, $\cdots$, K—and in all of which colors have the same relative positions. X of those neighborhoods connect to each other and form a "ring", and therefore there are $\lfloor \frac{X}{K} \rfloor$ such "rings". Each ring contains XK vertices. Those $\lfloor \frac{X}{K} \rfloor$ "rings" partition the two-dimensional space of the torus and leave $\lfloor \frac{X}{K} \rfloor$ "gaps" among them. Those $\lfloor \frac{X}{K} \rfloor$ "gaps" contain totally $X^2 - \lfloor \frac{X}{K} \rfloor \cdot XK = X(X \bmod K)$ vertices, and Algorithm 4.2 makes those "gaps" as even as possible so that the biggest "gap" can be filled with $\lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$ colors—color $K+1$, $K+2$, $\cdots$, $K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$—and the smallest "gap" can be filled with $\lfloor \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rfloor$ colors—color $K+1$, $K+2$, $\cdots$, $K + \lfloor \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rfloor$. That's why Algorithm 4.2 can use as few as $K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$ colors to diversity color the $X \times X$ torus.

□

Theorem* 4.3*: Algorithm 4.2 diversity colors the $X \times X$ torus correctly.

Proof: First let's prove that Algorithm 4.2 colors each vertex in the torus exactly once. Denote the 'total number of times a color is assigned to a vertex in the process of coloring' by T, and let all variables have the same meanings as in Algorithm 4.2, then clearly:

$$T = \sum_{i=0}^{x-1} \Big(\sum_{\Delta x=-m}^{m} \sum_{\Delta y=-m+|\Delta x|}^{m-|\Delta x|} \sum_{j=0}^{X-1} 1 + \sum_{j=1}^{\alpha} \sum_{l=0}^{X-1} 1 \Big) + \sum_{i=0}^{y-1} \Big(\sum_{\Delta x=-m}^{m} \sum_{\Delta y=-m+\Delta x}^{m-|\Delta x|} \sum_{j=0}^{X-1} 1 + \sum_{j=1}^{\beta} \sum_{l=0}^{X-1} 1 \Big)$$
$$= X[X - x(1 + \beta - \alpha)].$$

By the definitions of α and β we know either $\beta = \alpha$ or $\beta = \alpha - 1$. If $\beta = \alpha$, then $\beta = \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor}$ and β is an integer, so $x = X \bmod (\beta + K) = 0$, and therefore $T = X[X - 0] = X^2$. If $\beta = \alpha - 1$, then $T = X[X - x(1 + \alpha - 1 - \alpha)] = X^2$. So $T = X^2$ anyway. Therefore Algorithm 4.2 assigned colors to vertices in the torus totally X^2 times. It can be verified that at any two moments of coloring in the algorithm, the two vertices colored have different indices and thus are different vertices, (the verification method is straightforward calculation, and for simplicity we omit the details), so each of the X^2 vertices in the torus is colored at most once. Now we know that each vertex in the torus is colored once and only once by Algorithm 4.2.

Now we need to show that for any two vertices of the same color, the distance between them is at least $2m + 1$. Again the method is straightforward calculation, and for simplicity we omit the details. Since any two vertices of the same color are at least $2m + 1$ hops away from each other, they cannot be contained in the same neighborhood of radius m. Therefore in every neighborhood of radius m, all the colors are different. So every neighborhood of radius m in the torus contains $2m^2 + 2m + 1$ different colors. So the coloring Algorithm 4.2 assigned to the torus is a diversity coloring.

□

Corollary* 4.4*: Given an $X \times X$ torus $G(V, E)$ and parameters K and m, ($X \geq 2m^2 + 2m + 1$, $K = 2m^2 + 2m + 1$), we have $N_{\min} \leq K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$.

Proof: Algorithm 4.2 diversity colors the torus $G(V, E)$ with $N = K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$ colors, so

$N_{\min} \le K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$.

□

Corollary* 4.5*: Let $G(V,E)$ be an $X \times X$ torus, and let $K = 2m^2 + 2m + 1$. (1) $N_{\min} = K$ if and only if $K \mid X$; (2) When $X \ge K(K-1)$, if $K \nmid X$, then $N_{\min} = K + 1$.

Proof: (1) First let's show that $N_{\min} = K$ if and only if $K \mid X$. When $K \mid X$, let $N = K$, and we can use Algorithm 4.2 to color the $X \times X$ torus. So $N_{\min} = K$ if $K \mid X$.

Now Suppose there is a diversity coloring on an $X \times X$ torus that uses $2m^2 + 2m + 1 = K$ colors. For color i ($1 \le i \le K$), since the distance between any two vertices of color i is at least $2m + 1$, all neighborhoods of those vertices of color i of radius m are disjoint. Now by the sphere packing principle, since each neighborhood contains $2m^2 + 2m + 1$ vertices, there are at most $\lfloor \frac{X^2}{2m^2+2m+1} \rfloor = \lfloor \frac{X^2}{K} \rfloor$ vertices of color i. Since all vertices in the $X \times X$ torus are colored and there are totally K colors, there are exactly $\frac{X^2}{K}$ vertices of color i in the torus ($1 \le i \le K$). So for any color i ($1 \le i \le K$), the neighborhoods of those vertices of color i of radius m have a 'close packing' in the torus [3] [13]. It's easy to verify that there is only one way of 'close packing' neighborhoods in torus, and it's a well-known result that such a 'close packing' exists in an $X \times X$ torus only if $(2m^2 + 2m + 1) | X$. Therefore if $N_{\min} = 2m^2 + 2m + 1 = K$, then $K \mid X$.

(2) When $X \ge K(K-1)$ and $K \nmid X$, by part (1) we know $N_{\min} \ge K + 1$. Since we can let $N = K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$ and use Algorithm 4.2 to color the $X \times X$ torus, and $K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil \le K + 1$ when $X \ge K(K-1)$, we have $N_{\min} \le K + 1$ when $X \ge K(K-1)$ and $K \nmid X$. Therefore when $X \ge K(K-1)$, if $K \nmid X$, then $N_{\min} = K + 1$.

□

Corollary* 4.6*: When $X \ge K(K-1)$ or $X \bmod K \le 1$, Algorithm 4.2 guarantees to output optimal diversity coloring on the $X \times X$ torus. (Here $K = 2m^2 + 2m + 1$.)

Proof: Suppose $X \ge K(K-1)$ or $X \bmod K \le 1$. Then $N_{\min} = K$ if $K \mid X$, and $N_{\min} = K + 1$ otherwise. In Algorithm 4.2 $N = K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$, which equals K if $K \mid X$ and equals $K + 1$ otherwise. Therefore Algorithm 4.2 colors the torus using exactly $N_{\min}$ colors.

□

We see that Algorithm 4.2 uses as few as $K + \lceil \frac{X \bmod K}{\lfloor \frac{X}{K} \rfloor} \rceil$ colors to color an $X \times X$ torus. When X, the size of the torus, is sufficiently large compared to K, the number of colors used can get very colse to K. So for the same reason as discussed in sub-Section A, when the size of the torus is sufficiently large, the decoding work of the K-out-of-N file distribution scheme has very low complexity.

V. Concluding Remarks

We have proposed the K-out-of-N file distribution scheme for distributively storing data in mobile networks, and formulated the scheme as a diversity coloring problem when each gateway stores one file segment. We've studied properties of diversity coloring on general graphs, mainly on $N_{\min}$, the minimum number of colors necessary for diversity coloring. A diversity coloring algorithm for general graphs is presented. That algorithm essentially reduces the diveristy coloring problem to a vertex coloring problem, and it has linear complexity if sufficient colors are used. We've presented a linear complexity diversity coloring algorithm for trees, which is an algorithm using greedy coloring and the 'divide and conquer' technique. The algorithm for trees always outputs optimal coloring when we let N be equal to K. Also we've given diversity coloring algorithms for rings and tori. The algoirthm for rings always gives optimal coloring. And the algorithm for tori guarantees to give optimal coloring when the sizes of tori are sufficiently large compared to K, as well as in some other cases. For both rings and tori, the number of colors the algorithms need gets very close to K when the rings and tori are large, therefore enabling the decoding complexity to become very low.

The K-out-of-N file distribution scheme can be seen as a generalization of both the *multicenter resource allocation scheme* and the *file segmentation scheme*.It not only allows the segmentation of files, but also employes parity information. The existence of parity information increases the variety of file segments and makes better file placements possible. For example, consider a ring with 7 vertices. Suppose we want to assign segments of a file to vertices such that each vertex in the ring can retrieve enough file segments from its two neighbors and itself for recovering the file, and it's required that each vertex can store at most one third of the file. By using the K-out-of-N scheme with $K = 3$ and $N = 4$, we can find the placement solution which corresponds to such a coloring on the ring: '1–2–3–4–1–2–3'. However no feasible solution exists if parity information is not used.

Generally speaking, the K-out-of-N file distribution scheme adapts to network structures better because of the freedom induced by the increased variety of file segments. So it can find placement of file segments of better performance (delay, data flow, load balance, etc.). The K-out-of-N file distribution scheme also enhances the network's capability to tolerate faults (failure of gateways, loss of file segments, etc.), because the greater varity of file segments increases the possibility that K different segments exist when faults appear, which are enough information for the file recovery.

Naturally we can generalize the diversity coloring problem into a *multi-diversity-coloring problem*—suppose each vertex can be assigned any number of colors, how to color the vertices in a graph so that each neighborhood of radius m contains at least K different colors? The coloring should be optimized according to some criteria such as memory requirement, load balancing, etc. The multi-diversity-coloring problem remains as our future research work.

APPENDIX I

In this appendix, the process of diversity coloring the tree $G(V, E)$ in example 3.3 is briefly listed in the table below.

Execution Sequence #	**Execution Content**	**Vertex Colored during this execution: Color of the vertex**	**Set X and Y when this execution is done.**
1	Execute step 1.		$X = \{(\{v_1\}; v_1; 0)\}$. $Y = \emptyset$.
2	Execute step 2 and 3.1.		As in execution #1.
3	Execute step 3.2. $x = (A; r; d) = (\{v_1\}; v_1; 0)$ in this execution.	$(v_1 : 1)$; $(v_2 : 2)$; $(v_3 : 3)$; $(v_4 : 4)$; $(v_5 : 5)$; $(v_6 : 6)$; $(v_7 : 7)$; $(v_8 : 8)$; $(v_9 : 9)$.	$X = \emptyset$. $Y = \{(\{v_2, v_3\}; v_1; 3)\}$.
4	Execute step 3.1 and 4.1.		As in execution #3.
5	Execute step 4.2. $x = (A; r; d) = (\{v_2, v_3\}; v_1; 3)\}$ in this execution.		$X = \{(\{v_4, v_5, v_6\}; v_2; 2), (\{v_3\}; v_3; 2)\}$. $Y = \emptyset$.
6	Execute step 4.1, 2, 3.1.		As in execution #5.
7	Execute step 3.2. $x = (A; r; d) = (\{v_4, v_5, v_6\}; v_2; 2)$ in this execution.	$(v_{10} : 7)$; $(v_{11} : 8)$.	$X = \{(\{v_3\}; v_3; 2)\}$. $Y = \{(\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_2; 2)\}$.
8	Execute step 3.1.		As in execution #7.
9	Execute step 3.2. $x = (A; r; d) = (\{v_3\}; v_3; 2)$ in this execution.	$(v_{14} : 9)$.	$X = \{(\{v_7, v_8\}; v_3; 2)\}$. $Y = \{(\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_2; 2)\}$.
10	Execute step 3.1.		As in execution #9.
11	Execute step 3.2. $x = (A; r; d) = (\{v_7, v_8\}; v_3; 2)$ in this execution.	$(v_{15} : 4)$; $(v_{16} : 5)$; $(v_{17} : 6)$.	$X = \emptyset$. $Y = \{(\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_2; 2), (\{v_{14}, v_{15}, v_{16}, v_{17}\}; v_3; 3)\}$.
12	Execute step 3.1, 4.1.		As in execution #11.
13	Execute step 4.2. $x = (A; r; d) = (\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_2; 2)$ in this execution.		$X = \{(\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_5; 1)\}$. $Y = \{(\{v_{14}, v_{15}, v_{16}, v_{17}\}; v_3; 3)\}$.
14	Execute step 4.1.		As in execution #13.

Execution Sequence #	**Execution Content**	**Vertex Colored during this execution: Color of the vertex**	**Set X and Y when this execution is done.**
15	Execute step 4.2. $x = (A; r; d) = (\{v_{14}, v_{15}, v_{16}, v_{17}\}; v_3; 3)$ in this execution.		$X = \{(\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_5; 1), (\{v_{14}, v_{15}, v_{16}, v_{17}\}; v_7; 2)\}$. $Y = \emptyset$.
16	Execute step 4.1, 2, 3.1.		As in execution #15.
17	Execute step 3.2. $x = (A; r; d) = (\{v_9, v_{10}, v_{11}, v_{12}, v_{13}\}; v_5; 1)$ in this execution.	$(v_{12} : 3)$.	$X = \{(\{v_{14}, v_{15}, v_{16}, v_{17}\}; v_7; 2)\}$. $Y = \emptyset$.
18	Execute step 3.1.		As in execution #17.
19	Execute step 3.2. $x = (A; r; d) = (\{v_{14}, v_{15}, v_{16}, v_{17}\}; v_7; 2)$ in this execution.	$(v_{18} : 2)$.	$X = \emptyset$. $Y = \{(\{v_{18}, v_{19}, v_{20}\}; v_7; 2)\}$.
20	Execute step 3.1, 4.1.		As in execution #19.
21	Execute step 4.2. $x = (A; r; d) = (\{v_{18}, v_{19}, v_{20}\}; v_7; 2)$ in this execution.		$X = \{(\{v_{18}, v_{19}, v_{20}\}; v_{14}; 1)\}$. $Y = \emptyset$.
22	Execute step 4.1, 2, 3.1.		As in execution #21.
23	Execute step 3.2. $x = (A; r; d) = (\{v_{18}, v_{19}, v_{20}\}; v_{14}; 1)$ in this execution.	$(v_{19} : 1)$; $(v_{20} : 8)$.	$X = \emptyset$. $Y = \emptyset$.
24	Execute step 3.1, 4.1.		As in execution #23.
25	Execute step 2.	$(v_{13} : 1)$.	$X = \emptyset$. $Y = \emptyset$.
	Algorithm Ends.		

APPENDIX II

This appendix presents the proof of Lemma 3.1.

Proof: We prove Lemma 3.1 by induction. Of all the 6 steps in Algorithm 3.1, step 2, step 3.1 and step 4.1 don't change any element in set X or Y. So we only need to check what happens at step 1, step 3.2 and step 4.2.

Step 1 is the first step executed in the whole algorithm, and it is executed only once. After step 1 is executed, X contains exactly one element—relation structure $(\{$the root of $G\}$; the root of G; 0), and $Y = \emptyset$. It's easy to verify that all the 12 properties are satisfied.

Now suppose step 3.2 is to be executed, and before the execution of step 3.2 all the 12 properties hold.

Let $x = (A; r; d)$ and $x_{new} = (A_{new}; r; d_{new})$ have the same meaning as in step 3.2 of the algorithm. Note that when step 3.2 is executed, $x = (A; r; d)$ is deleted from X, and $x_{new} = (A_{new}; r; d_{new})$ is the only relation structure newly inserted into X or Y—when $A_{new} \neq \emptyset$. After step 3.2 is executed,

1) By the induction assumption, all vertices in A are at the same layer. The way A_{new} is produced at step 3.2 ensures that all vertices in A_{new} are all at the same layer. Therefore x_{new} satistifes Property 1 after step 3.2 is executed, and so does any other element in X or Y. So Property 1 holds after step 3.2 is executed.

2) By the induction assumption, r is either the common ancestor of all vertices in A, or the only vertex in A. Check the way A_{new} is produced at step 3.2, we see that either $A_{new} = A$, or every vertex in A_{new} has an ancestor in A, if $A_{new} \neq \emptyset$. So x_{new} satisfies either case (1) or case (2) in Property 2.

Next we show that if x_{new} is inserted into Y, then only case (1) in Property 2 is true—namely, r is a common ancestor of all vertices in A_{new}. To prove that we use contradiction. Suppose case (2) of Property 2—$A_{new} = \{r\}$ is true, and x_{new} is inserted into Y. x_{new} is inserted into Y only if $L(A_{new}) - L(r) + d_{new} > m$. Since $A_{new} = \{r\}$, $L(A_{new}) = L(r)$, so $d_{new} > m$. Clearly $A = \{r\}$, and by induction x satisfies Property 4, Property 9 and Property 11. Since step 3.2 colors vertices in $U(x)$ 'depth by depth' using only colors previously unused in $U(x)$, and we already have $d_{new} > m$, it's not hard to see that all vertices in $\chi_m^G(r)$ are not only colored but all have different colors after step 3.2 is executed. So $\chi_m^G(r)$ contains $|\chi_m^G(r)| \geq K$ different colors after step 3.2 is executed, and therefore $A_{new} \neq \{r\}$ according to the way A_{new} is produced at step 3.2. So we have a contradiction here. So if x_{new} is inserted into Y, then r is a common ancestor of all vertices in A_{new}.

So x_{new} satisfies Property 2 after step 3.2 is executed. And so does any other element in X or Y. So Property 2 holds after step 3.2 is executed.

3) The way A_{new} is produced at step 3.2 ensures that $\exists\, v \in A_{new}\ s.t.\ \chi_m^G(v)$ contains less than K different colors, if x_{new} is inserted into X or Y. So x_{new} satisfies Property 3 after step 3.2 is executed. Now let y be any other element (relation structure) in X or Y after step 3.2 is executed. By induction on Property 10, all vertices in $U(x) \cap U(y)$ are colored before step 3.2 is executed. So no vertex in $U(y)$ is colored at step 3.2. So by the induction assumption, we see y satisfies Property 3 after step 3.2 is executed. So Property 3 holds after step 3.2 is executed.

4) By induction x has Property 4, so $d \geq 0$. At step 3.2, $d_{new} \geq d$, so $d_{new} \geq 0$. So x_{new} satisfies Property 4 after step 3.2 is executed. And so does any other element in X or Y. So Property 4 holds after step 3.2 is executed.

5) At step 3.2, vertices in $U(x)$ are colored 'depth by depth' for x. Since x and x_{new} have the same 'root' r, those vertices are also colored 'depth by depth' for x_{new}. And clearly $0 \leq d \leq d_{new}$. So for any vertex v at depth γ for x_{new}, (a) if $0 \leq \gamma < d_{new}$, then the way d_{new} is defined ensures that v is colored; (b) if $\gamma < 0$ and $\exists\, u \in A_{new}\ s.t.\ d^G(v,u) \leq m$, then define u_0 this way—if $A_{new} = A$ then $u_0 = u$, otherwise let u_0 be the unique ancestor of u in A. Since x and x_{new} have the same root r, v is also at depth $\gamma < 0 \leq d$ for x. Since $d^G(v,u_0) = d^G(v,r) + d^G(r,u_0) \leq d^G(v,r) + d^G(r,u) = d^G(v,u) \leq m$, and by induction x satisfies Property 5, so v is colored. (c) if $\gamma > d_{new}$, then the fact that vertices are colored 'depth by depth' for x_{new} at step 3.2 and the way d_{new} is defined ensure that v is uncolored. So x_{new} satisfies Property 5 after step 3.2 is

executed.

Now let y be any relation structure in X or Y other than x_{new} after step 3.2 is executed. We've proved in part (3) that no vertex in $U(y)$ is colored at step 3.2. So by induction we see y also satisfies Property 5.

So Property 5 holds after step 3.2 is executed.

6) We know $L(A_{new}) \geq L(A)$, and by induction $L(A) \geq L(r)$, so $L(A_{new}) \geq L(r)$. x_{new} is inserted into X only if $L(A_{new}) - L(r) + d_{new} \leq m$, and we've proved $d_{new} \geq 0$, so $L(r) \geq L(A_{new}) - m + d_{new} \geq L(A_{new}) - m$. Therefore x_{new} satisfies Property 6 after step 3.2 is executed, and so does any other element in X. So Property 6 holds after step 3.2 is executed.

7) x_{new} is inserted into X at step 3.2 only if $L(A_{new}) - L(r) + d_{new} \leq m$. So x_{new} satisfies Property 7 after step 3.2 is executed, and so does any other element in X. So Property 7 holds after step 3.2 is executed.

8) x_{new} is inserted into Y at step 3.2 only if $L(A_{new}) - L(r) + d_{new} > m$. So x_{new} satisfies Property 8 after step 3.2 is executed, and so does any other element in Y. So property 8 holds after step 3.2 is executed.

9) Suppose x_{new} is inserted into X at step 3.2. By Definition 3.5, clearly $\forall\ a \in A_{new}$, $\chi_m^G(a) \subseteq U(x_{new})$, and therefore $\{v | v \in \chi_m^G(a), v \text{ is colored}\} \subseteq \{v | v \in U(x_{new}), v \text{ is colored}\}$. For any colored vertex $v \in U(x_{new})$, say v is at depth d_v for x_{new}. (a) If $d_v < 0$, by Definition 3.5 we know $\exists\ u \in A_{new}\ s.t.$ $d^G(u,v) \leq m$, and therefore $d^G(a,v) = d^G(a,r) + d^G(r,v) = d^G(u,r) + d^G(r,v) = d^G(u,v) \leq m$, so $v \in \chi_m^G(a)$. (b) If $d_v \geq 0$, $d^G(a,v) \leq d^G(a,r) + d^G(r,v) = L(a) - L(r) + d_v$. By Property 5 we get $d_v \leq d_{new}$. By Property 7 we get $L(A_{new}) - L(r) + d_{new} \leq m$. And $L(a) = L(A_{new})$ since $a \in A_{new}$. So $d^G(a,v) \leq L(A_{new}) - L(r) + d_{new} \leq m$. Therefore $v \in \chi_m^G(a)$. Now it's clear that $\{v | v \in \chi_m^G(a), v \text{ is colored}\} = \{v | v \in U(x_{new}), v \text{ is colored}\}$. So x_{new} satisfies Property 9 after step 3.2 is executed. For any other ralation structure y in X or Y, no vertex in $U(y)$ is colored at step 3.2. So by induction y satisfies Property 9. So Property 9 holds after step 3.2 is executed.

10) Suppose x_{new} is inserted into X or Y at step 3.2, and let y be another relation structure in X or Y after step 3.2 is executed. Since either $A_{new} = A$ or every vertex in A_{new} is a descendent of some vertex in A, we have $W(x_{new}) \subseteq W(x)$ and $U(x_{new}) \subseteq U(x)$. By induction $W(x) \cap W(y) = \emptyset$, and all vertices in $U(x) \cap U(y)$ are colored. So $W(x_{new}) \cap W(y) \subseteq W(x) \cap W(y) = \emptyset$, and all vertices in $U(x_{new}) \cap U(y)$ are also in $U(x) \cap U(y)$ and therefore are colored. So Property 10 holds after step 3.2 is executed.

11) At step 3.2, only colors previously unused in $U(x)$ are used to color vertices in $U(x)$, and each color is used at most once, so after the coloring all colored vertices in $U(x)$ have different colors. Since $U(x_{new}) \subseteq U(x)$, all colored vertices in $U(x_{new})$ have different colors. So x_{new} satisfies Property 11. For any other relation structure y in X or Y, since no vertex in $U(y)$ is colored at step 3.2, by induction y satisfies Property 11. So Property 11 holds after step 3.2 is executed.

12) Let v be a vertex such that $v \notin \bigcup_{y \in X \cup Y} W(y)$ after step 3.2 is executed. Then by Property 10, it's not hard to see that either $v \in W(x) - W(x_{new})$, or $v \notin \bigcup_{y \in X \cup Y} W(y)$ even before step 3.2 is executed. If $v \in W(x) - W(x_{new})$, then by checking the way how A_{new} is produced at step 3.2 we see that clearly $\chi_m^G(v)$ contains no less than K different colors. If $v \notin \bigcup_{y \in X \cup Y} W(y)$ before step 3.2 is executed, then by induction

$\chi_m^G(v)$ contains no less than K different colors. So Property 12 holds after step 3.2 is executed.

Now suppose step 4.2 is to be executed, and before the execution of step 4.2 all the 12 properties hold. Let $x = (A; r; d)$ have the same meaning as in step 4.2. Note that step 4.2 doesn't do any coloring. Also note that at the end of step 4.2, $x = (A; r; d)$ is deleted from Y. And during step 4.2 some new relation structures might be inserted into X or Y. After step 4.2 is executed,

1) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X or Y at step 4.2. By the way P is produced at step 4.2, it's easy to see that all vertices in P are at the same layer. So y satisfies Property 1 after step 4.2 is executed. And so does any other element in X or Y. So Property 1 holds after step 4.2 is executed.

2.I) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X at step 4.2. By the way P and s are produced at step 4.2, it's easy to see that y satisfies either case (1) or case (2) of Property 2. So Property 2.(I) holds after step 4.2 is executed.

The proof of Property 2 (II) is placed after part (5) because it'll use some conclusions to be proved below.

3) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X or Y at step 4.2. The way P is produced ensures that $\exists\, v \in P\ s.t.\ \chi_m^G(v)$ contains less than K different colors. So y satisfies Property 3. So Property 3 holds after step 4.2 is executed.

4) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X or Y at step 4.2. Then $d_{new} \geq d - L(s) + L(r) \geq d - \lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor + L(r)$. By induction $d \geq 0$. So (a) if $L(A) \leq L(r) + m$ and $d = 0$, then $d_{new} \geq -\lfloor \frac{L(A)+L(r)+1-m}{2} \rfloor + L(r) \geq -\lfloor \frac{L(r)+m+L(r)+1-m}{2} \rfloor + L(r) = 0$. (b) If $L(A) \leq L(r) + m$ and $d > 0$, then $d_{new} \geq d - \frac{L(A)+L(r)+d+1-m}{2} + L(r) = \frac{[L(r)+m-L(A)]+(d-1)}{2} \geq 0$. (c) If $L(A) > L(r) + m$, then $L(r) + d \geq L(A) - m$—otherwise let vertex v be the parent of any vertex in A, and since by induction x satisfies Property 5, we can easily verify that $\chi_m^G(v)$ contains at most 1 color; however by induction, Property 10 and Property 12 hold even before step 4.2 is executed, so it's also easy to see that $\chi_m^G(v)$ contains no less than $K \geq 3$ different colors; and that's a contradiction. Therefore $d_{new} \geq d - \frac{L(A)+L(r)+d+1-m}{2} + L(r) = \frac{[L(r)+d-L(A)+m]-1}{2} \geq -0.5$. Since d_{new} is an integer, again we have $d_{new} \geq 0$. So y satisfies Property 4. So Property 4 holds after step 4.2 is executed.

5) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X or Y at step 4.2. By Property 4, $d_{new} \geq 0$. Let v be a vertex at depth γ for y. Consider the following cases:

(a) consider the case $\gamma > d_{new}$. Since s is a descendent of r, the depth of v for x is $\gamma + [L(s) - L(r)] > d_{new} + [L(s) - L(r)]$. The way d_{new} is produced ensures that $d_{new} \geq d - L(s) + L(r)$, so the depth of v for x is $\gamma + L(s) - L(r) > d$. By induction, x satisfies Property 5, so v is uncolored.

(b) Now consider the case $\gamma < d_{new}$ and $\exists\, u \in P\ s.t.\ d^G(u, v) \leq m$. Note that d_{new} can only take two possible values—$d - L(s) + L(r)$ or $d - L(s) + L(r) + 1$. Also note that either $P \subseteq A$ or every vertex in P is a descendent of some vertex in A, so it's easy to see that $U(y) \subseteq U(x)$. Since $\exists\, u \in P\ s.t.\ d^G(u, v) \leq m$, we have $v \in U(y)$ and therefore $v \in U(x)$. There are three sub-cases to consider:

(b1) Considere the case $\gamma = d - L(s) + L(r)$ and $d_{new} = d - L(s) + L(r) + 1$. By checking the way d_{new} is produced we see that clearly v is colored.

(b2) Consider the case $0 \le \gamma < d - L(s) + L(r)$. The depth of v for x is $\gamma + [L(s) - L(r)] < d$. Since s is a descendent of r, $L(s) > L(r)$, and so the depth of v for x is $\gamma + [L(s) - L(r)] > \gamma \ge 0$. By induction x satisfies Property 4, so v is colored.

(b3) Consider the case $\gamma < 0$. Then the depth of v for x is at most $(|\gamma| - 1) + [L(s) - 1] - L(r) = |\gamma| + L(s) - L(r) - 2$. Since $\exists\, u \in P\ s.t.\ d^G(u,v) \le m$, and we know $d^G(u,v) = L(P) - L(s) + |\gamma|$, so we get $|\gamma| \le m - L(P) + L(s)$. Therefore the depth of v for x is at most
$|\gamma| + L(s) - L(r) - 2 \le m - L(P) + 2L(s) - L(r) - 2 \le m - L(A) + 2L(s) - L(r) - 2$
$\le m - L(A) + 2\lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor - L(r) - 2 \le m - L(A) + 2 \cdot \frac{L(A)+L(r)+d+1-m}{2} - L(r) - 2 = d - 1$. Since $v \in U(x)$ and by induction x satisfies Property 5, it's not hard to see that v is colored.

So y satisfies Property 5. So Property 5 holds after step 4.2 is executed.

2.II) In this part we'll prove Property 2.(II) holds after step 4.2 is executed. Let $y = (P; s; d_{new})$ be any of the relation structures inserted into Y at step 4.2. It's easy to see that y satisfies either case (1) or case (2) of Property 2. Now let's show that in fact, y satisfies only case (1) of Property 2—namely, s is a common ancestor of all vertices in P. To prove that we use contradiction. Suppose case (2) of Property 2—$P = \{s\}$—is true. y is inserted into Y only when $L(P) - L(s) + d_{new} > m$, so $d_{new} > m$. Since $P = \{s\}$, we know $s \in A$. So $\chi_m^G(s) \subseteq U(y) \subseteq U(x)$. Since y has Property 5 and $d_{new} > m$, all vertices in $\chi_m^G(s)$ are colored. Since by induction all colored vertices in $U(x)$ have different colors, $\chi_m^G(s)$ contains $|\chi_m^G(s)| \ge K$ different colors. Since y has Property 3, $P \ne \{s\}$. So we have a contradiction here. So y satisfies only case (1) of Property 2. So Property 2.(II) holds after step 4.2 is executed.

By part (2.I) and part (2.II), we see Property 2 holds after step 4.2 is executed.

6) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X at step 4.2. The way P is produced at step 4.2 ensures that $L(s) \le L(P)$. And y is inserted into X only if $L(P) - L(s) + d_{new} \le m$, and by Property 4 $d_{new} \ge 0$, so $L(s) \ge L(P) - m + d_{new} \ge L(P) - m$. Therefore y satisfies Property 6. So Property 6 holds after step 4.2 is executed.

7) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X at step 4.2. y is inserted into X only if $L(P) - L(s) + d_{new} \le m$, so y satisfies Property 7. So Property 7 holds after step 4.2 is executed.

8) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into Y at step 4.2. y is inserted into Y only if $L(P) - L(s) + d_{new} > m$, so y satisfies Property 8. So Property 8 holds after step 4.2 is executed.

9) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X at step 4.2. The method to prove y satisfies Property 9 is the same as the method used to prove that Property 9 holds after step 3.2 is executed, and we omite the details for simplicity. So Property 9 holds after step 4.2 is executed.

10) Let $y_1 = (P_1; s_1; d_{new(1)})$ and $y_2 = (P_2; s_2; d_{new(2)})$ be any two relation structures inserted into X or Y at step 4.2. Since $L(s_1) = L(s_2) = \min(\lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor, L(A))$ and $s_1 \ne s_2$, by Definition 3.5 and Property 2 it's easy to see $W(y_1) \cap W(y_2) = \emptyset$. Let v be any vertex in $U(y_1) \cap U(y_2)$. Since $L(s_1) = L(s_2)$ and $s_1 \ne s_2$, we get $\{s_1$ and all descendents of $s_1\} \cap \{s_2$ and all descendents of $s_2\} = \emptyset$, so without loss of generality we can say $v \notin \{s_1$ and all descendents of $s_1\}$. Then the depth of v for y_1 is less than 0. Since $v \in U(y_1)$, there exists $u \in P_1\ s.t.\ d^G(u,v) \le m$. We've proved that y_1 satisfies Property 5, so v is colored. So all vertices in $U(y_1) \cap U(y_2)$ are colored.

Now let $y = (P; s; d_{new})$ be any of the relation structures inserted into X or Y at step 4.2, and let $z = (B; s; e)$ be any relation structure other than x that is in X or Y even before step 4.2 is executed. By induction $W(x) \cap W(z) = \emptyset$. It's easy to see that $W(y) \subseteq W(x)$, so $W(y) \cap W(z) = \emptyset$. Also by induction, all vertices in $U(x) \cap U(z)$ are colored. It's easy to see $U(y) \subseteq U(x)$, so all vertices in $U(x) \cap U(z)$ are colored, too.

So Property 10 holds after step 4.2 is executed.

11) Let $y = (P; s; d_{new})$ be any of the relation structures inserted into X or Y at step 4.2. Since $U(y) \subseteq U(x)$, and by induction all colored vertices in $U(x)$ have different colors, so all colored vertices in $U(y)$ have different colors, too. Therefore y satisfies Property 11. So Property 11 holds after step 4.2 is executed.

12) Let $y_1, y_2, \cdots, y_n$ be all the relation structures inserted into X or Y at step 4.2. Let v be a vertex such that $v \notin \bigcup_{z \in X \cup Y} W(z)$ after step 4.2 is executed. Then by Property 10, it's not hard to see that either $v \in W(x) - W(y_1) - W(y_2) - \cdots - W(y_n)$, or $v \notin \bigcup_{z \in X \cup Y} W(z)$ even before step 4.2 is executed. Now let's consider two sub-cases.

(a) Consider the case $v \in W(x) - W(y_1) - W(y_2) - \cdots - W(y_n)$. Since $L(s) = \min(\lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor, L(A))$, either $L(s) = \lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor$ or $L(s) = L(A)$. If $L(s) = \lfloor \frac{L(A)+L(r)+d+1-m}{2} \rfloor$, then $L(s) = L(r) + \lfloor \frac{L(A)-L(r)+d+1-m}{2} \rfloor$; and since x satisfies Property 8, we have $L(A) - L(r) + d > m$; therefore $L(s) \geq L(r) + 1$. If $L(s) = L(A)$, since x satisfies Property 2, $L(s) = L(A) > L(r)$. So anyway, $L(r) < L(s) \leq L(A)$ is always true. Then by checking the way how $y_1, y_2, \cdots, y_n$ are produced at step 4.2 we see that clearly $\chi_m^G(v)$ contains no less than K different colors.

(b) If $v \notin \bigcup_{z \in X \cup Y} W(z)$ before step 4.2 is executed, then by induction $\chi_m^G(v)$ contains no less than K different colors.

So Property 12 holds after step 4.2 is executed.

□

ACKNOWLEDGEMENT

The authors would like to acknowledge discussions with Matthew Cook and Kevin Foltz.

REFERENCES

[1] B. Awerbuch, Y. Bartal and A. Fiat, "Competitive distributed file allocation", Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, p 164-173, 1993.

[2] M. Blaum, J. Brady, J. Bruck, J. Menon, "Evenodd: an efficient scheme for tolerating double disk failures in RAID architectures", IEEE Transactions on Computers, vol. 44, no. 2, pp. 192-202, 1995.

[3] M. Blaum, J. Bruck and A. Vardy, “Interleaving schemes for multidimensional cluster errors”, IEEE Transactions on Information Theory, vol. 44, no. 2, pp. 730-743, 1998.

[4] M. Blaum, J. Bruck and A. Vardy, “MDS array codes with independent parity symbols”, IEEE Transactions on Information Theory, vol. 42, no. 2, pp. 529-542, 1996.

[5] V. Bohossian, C. C. Fan, P. S. LeMahieu, M. D. Riedel, L. Xu and J. Bruck, “Computing in the RAIN: a reliable array of independent nodes”, IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 2, pp. 99-114, 2001.

[6] B. Bollobás, *Modern graph theory*, Springer-Verlag New York, 1998.

[7] W. A. Burkhard, J. Menon, “Disk array storage system reliability”, the 23rd International Symposium on Fault-Tolerant Computing, pp. 432-441, 1993.

[8] J. W. Byers, M. Luby and M. Mitzenmacher, “Accessing multiple mirror sites in parallel: Using Tornado codes to speed up downloads”, Proceedings of IEEE INFOCOM, vol. 1, p 275-283, 1999.

[9] M. Charikar, S. Guha, E. Tardos, D. B. Shmoys, “Constant-factor approximation algorithm for the k-median problem”, Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 1-10, 1999.

[10] C. Li, M. Chen, P. S. Yu and H. Hsiao, “Combining replication and parity approaches for fault -tolerant disk arrays”, Proceeedings of the 6th IEEE Symposium on Parallel and Distributed Processing, pp. 360-367, 1994.

[11] W. Domschke and A. Drexl, *Location and Layout Planning: An International Bibliography*, vol. 238 of *Lecture Notes in Economics and Mathematical Systems*, Springer-Verlag, Berlin, 1985.

[12] L. W. Dowdy and D. V. Foster, “Comparative models of the file assignment problem”, Computing Surveys, vol. 14, no. 2, pp. 287-313, 1982.

[13] S. W. Golomb and L. R. Welch, “Perfect codes in the Lee metric and the packing of polyominoes”, SIAM J. Appl. Math., vol. 18, no. 2, pp. 302-317, 1970.

[14] S. Guha, S. Khuller, “Greedy strikes back: Improved facility location algorithms”, Proceedings of the 9th Annual ACM SIAM Symposium on Discrete Algorithms, pp. 649-657, 1998.

[15] G. Y. Handler and P. B. Mirchandani, *Location on networks*, Cambridge, Massachusetts and London, England: The MIT Press, 1979.

[16] M. Holland, G. A. Gibson, “Parity declustering for continuous operation in redundant disk arrays”, the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 23-35, 1992.

[17] M. Holland, G. A. Gibson and D. P. Siewiorek, “Architectures and algorithms for on-line failure recovery in redundant disk arrays”, Distributed and Parallel Databases, vol. 2, pp. 295-335, 1994.

[18] K. Jain, V. Vazirani, “ Primal-dual approximation algorithms for metric facility location and k-median problems”, Proceedings of Symposium on Foundations of Computer Science, pp. 2-13, 1999.

[19] S. Khuller and Y. J. Sussmann, “The capacitated k-center problem”, SIAM J. Discrete Math., vol. 13, no. 3, pp. 403-418, 2000.

[20] M. R. Korupolu, C. G. Plaxton, R. Rajaraman, “Analysis of a local search heuristic for facility location problems”, Proceedings of the 9th Annual ACM SIAM Symposium on Discrete Algorithms, pp. 1-10, 1998.

[21] J. F. Kurose and R. Simha, “A microeconomic approach to optimal resource allocation in distributed computer systems”, IEEE Transactions on Computers, vol. 38, no. 5, pp. 705-717, 1989.

[22] L. J. Laning, M. S. Leonard, “File allocation in a distributed computer communication network”, IEEE Transactions on Computers, vol. C-32, no. 3, pp. 232-244, 1983.

[23] S. Mahmoud and J. S. Riordan, “Optimal allocation of resources in distributed information networks”, ACM Trans. Database Systems, vol. 1, pp. 66-78, 1976.

[24] Q. M. Malluhi and W. E. Johnston, “Coding for high availability of a distributed-parallel storage system”, IEEE Transactions on Parallel and Distributed Systems, vol. 9, no. 12, pp. 1237-1252, 1998.

[25] R. R. Mettu and C. G. Plaxton, “The online median problem”, Proceedings of the 41st Annual Symposium on Foundations of Computer Science, pp. 339-348, 2000.

[26] E. Minieka, “The m-center problem”, SIAM Review, vol. 12, pp. 138-139, 1970.

[27] P. Mirchandani and R. Francis, editors, *Discrete Location Theory*, Wiley, New York, NY, 1990.

[28] M. Naor and R. M. Roth, “Optimal file sharing in distributed networks”, SIAM J. Comput., vol. 24, no. 1, pp. 158-183, 1995.

[29] S. W. Ng and R. L. Mattson, “Uniform parity group distribution in disk arrays with multiple failures”, IEEE Transactions on Computers, vol. 43, no. 4, pp. 501-506, 1994.

[30] D. A. Patterson, G. A. Gibson and R. Katz, “A case for redundant arrays of inexpensive disks”, in Proc. SIGMOD Int. Conf. Data Management, pp. 109-116, 1988.

[31] J. S. Plank, “A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems”, Software—Practice and Experience, vol. 27, no. 9, pp. 995-1012, 1997.

[32] M. O. Rabin, “Efficient dispersal of information for security, load balancing, and fault tolerance”, Journal of the Association for Computing Machinery, vol. 36, no. 2, pp. 335-348, 1989.

[33] A. L. N. Reddy, P. Banerjee, “Gracefully degradable disk arrays”, 21st International Symposium on Fault-Tolerant Computing, pp. 401-408, 1991.

[34] E. J. Schwabe and I. M. Sutherland, “Improved parity-declustered layouts for disk arrays”, the 6th Annual ACM Symposium on Parallel Algorithms and Architectures, pp. 76-84, 1994.

[35] E. J. Schwabe and I. M. Sutherland, “Flexible usage of redundancy in disk arrays”, Theory of Computing Systems, vol. 32, pp. 561-587, 1999.

[36] D. B. Shmoys, E. Tardos and K. Aardal, “Approximation algorithms for facility location problems”, Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp.

265-274, 1997.

[37] H. Sun and S. Shieh, “Optimal information-dispersal for increasing the reliability of a distributed service”, IEEE Transactions on Reliability, vol. 46, no. 4, pp. 462-472, 1997.

[38] B. W. Wah, “File placement on distributed computer systems”, Computer, vol. 17, no. 1, pp. 23-32, 1984.

[39] B. W. Wah, “Efficient heuristic for file placement on distributed databases”, Proceedings -4th IEEE Computer Society’s International Computer Software & Applications Conference, pp. 462-468, 1980.

[40] S. B. Wicker, *Error control systems for digital communication and storage*, Prentice Hall, 1995.

[41] O. Wolfson, S. Jajodia and Y. Huang, “An adaptive data replication algorithm”, ACM Transactions on Database Systems, vol. 22, no. 2, pp. 255-314, 1997.

[42] L. Xu, V. Bohossian, J. Bruck, D. Wagner, “Low-density MDS codes and factors of complete graphs”, IEEE Transactions on Information Theory, vol. 45, no. 6, pp. 1817-1826, 1999.

[43] L. Xu and J. Bruck, “Improving the performance of data servers using array codes”, Electronic Technical Report 027, Paradise Lab of Caltech, http://www.paradise.caltech.edu/ETR.html, 1998.

[44] N. E. Young, “ K-medians, facility location, and the Chernoff-Wald bound”, Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 86-95, 2000.