DNAS: Dispersed Network Attached Storage for Reliability and Performance*

Anxiao (Andrew) Jiang Jehoshua Bruck
Parallel and Distributed Systems Lab
California Institute of Technology
Email: {jax,bruck}@paradise.caltech.edu

Abstract

With the advent of merging between communication and storage, there is an increasing need for developing distributed data layout schemes for network attached storage that address reliability and performance challenges. This paper proposes a novel scheme for storing information on networks. In particular, for a fault-free operation, it provides the ability to retrieve data by accessing network nodes within a small proximity. In the event of faults, data is guaranteed to be retrieved by exploring a slightly larger proximity.

The problem of designing layout schemes, namely providing Dispersed Network Attached Storage (DNAS), is formulated as a graph coloring problem that we call *Layered Diversity Coloring*. Consider the following problem: given a graph G(V,E) and N colors, how to color vertices of G so that every vertex can find at least K_1 different colors within m_1 hops (distance m_1), at least K_2 different colors within m_2 hops, ..., at least K_p different colors within m_p hops? The parameters N, K_1, K_2, ..., K_p, m_1, m_2, ..., m_p and p are all positive integers, and $N \geq K_1 > K_2 > \cdots > K_p$, $m_1 > m_2 > \cdots > m_p$.

In this paper we study the layered diversity coloring problem where the graph G(V,E) is a tree. A coloring algorithm of time complexity $O(|V|(K_1 + \sum_{i=1}^{p} m_i))$ is presented, and the sufficient and necessary condition for there to exist a layered diversity coloring on a tree follows the algorithm.

I Introduction

The *file assignment problem* (FAP) studies the placement of multiple copies of files on a network [1, 2, 3]. It has great importance in computer science and has a very strong relationship with the resource allocation problems studied in various other areas [4, 5, 6, 7]. Corresponding to different optimization criteria and different constraints, the file assignment problem can take many different forms. There is one important family of FAP problems which optimize the worst-case performance, the best-known of which is the p-center problem [8, 9, 10]. In the p-center problem, p copies of a file are stored on vertices of a graph, and each vertex reads the file by retrieving the file copy closest to itself. The goal is to minimize the maximum distance between any vertex and its closest file copy.

In recent years there is a trend to combine file assignment with coding for better performance [11, 12]. In [11], we proposed a file assignment scheme called the *K-out-of-N file distribution scheme*. In that scheme, a file is split into K equally long segments, and an MDS code (such as Reed-Solomon code) [13] is used to encode those K segments to get a total of N segments. (Here $N \geq K$.) Each of the N segments is of length $\frac{1}{K}$ of the original file. Because of the maximum-distance-seperable property of an MDS code, a decoder can recover the original file as long as the decoder can get any K of those N segments. The next step is to store one segment copy on each vertex of a graph, in such a way that every vertex can retrieve at least K different segments from vertices within m hops (including itself). (Here m is a parameter. Clearly m is an upper bound on the maximum distance between any vertex and the corresponding closest K segments it needs to retrieve for recovering the file.)

Given the graph and the parameters K and N, the goal of the *K-out-of-N file distribution scheme* is to find the minimum value for m and the corresponding data layout scheme. Like the p-center problem, the *K-out-of-N file distribution scheme* also optimizes the worst-case performance.

If we represent the N file segments with N colors,

*This work was supported in part by the Lee Center for Advanced Networking at Caltech.

and represent 'storing a segment on a vertex' with 'assigning a color to the vertex', then the *K-out-of-N file distribution scheme* becomes the *diversity coloring problem* as defined in [11]. Here we rewrite the definition of the *diversity coloring problem* in the following way:

Definition 1 (Diversity Coloring Problem): Given a graph $G(V, E)$ and parameters K, N and m, (here K, N and m are all positive integers, $K \leq N$,) how to assign one color to each vertex, (different vertices can have the same color,) such that every vertex can find at least K different colors within m hops?

In [11], the diversity coloring problem on several classes of graphs are studied. One of the main results there is for trees, which states that for a tree, there exists a diversity coloring on it if and only if every vertex of the tree can reach at least K vertices (including itself) within m hops.

In this paper, we extend the diversity coloring problem to a much more general form. And we call the new problem the *Layered Diversity Coloring Problem*. We formally define it as follows:

Definition 2 (Layered Diversity Coloring Problem): Given a graph $G(V, E)$ and N colors, how to assign one color to each vertex, (different vertices can have the same color,) so that every vertex can find at least K_1 different colors within m_1 hops, at least K_2 different colors within m_2 hops, ..., at least K_p different colors within m_p hops? (Here parameters N, K_1, K_2, ..., K_p, m_1, m_2, ..., m_p and p are all positive integers, and $N \geq K_1 > K_2 > \cdots > K_p$, $m_1 > m_2 > \cdots > m_p$.)

Fig. 1 shows an example of the layered diversity coloring. In the example, the parameters are $N = 6$, $K_1 = 6$, $K_2 = 4$, $m_1 = 3$ and $m_2 = 2$. It's easy to verify that the graph has totally 6 colors, and every vertex can find at least $K_1 = 6$ different colors within $m_1 = 3$ hops and at least $K_2 = 4$ different colors within $m_2 = 2$ hops. (For example, the shaded vertex can find 6 different colors—color { 4,2,5,6,1,3 } —within 3 hops, and can find 5 different colors—color { 4,2,5,6,1 } —within 2 hops.) So the coloring in Fig.1 is indeed a layered diversity coloring.

The *layered diversity coloring* can be seen as a new resource allocation scheme. And it has direct applications to Dispersed Network Attached Storage (DNAS). The following are two examples of its

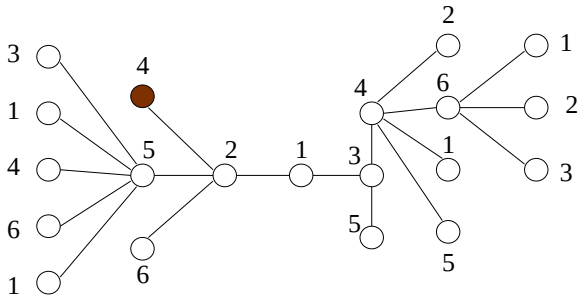

Figure 1: Example of Layered Diversity Coloring, with parameters $N = 6$, $K_1 = 6$, $K_2 = 4$, $m_1 = 3$ and $m_2 = 2$.

applications:

- *File allocation with fault-tolerant performance:* Let a file be split into K equally long segments, and use an MDS code to encode them and get N segments. The file can be recovered by decoding any K of those N segments. We let $N \geq K_1 > K_2 > \cdots > K_p \geq K$. If we represent the N segments with N colors, then a layered diversity coloring corresponds to a data layout scheme on the graph where each vertex can retrieve at least K_p different segments from vertices within m_p hops, at least K_{p-1} different segments from vertices within m_{p-1} hops, ..., at least K_1 different segments from vertices within m_1 hops. So when no data are lost, every vertex can retrieve enough—that is, K—different segments from vertices within m_p hops and recover the file. If data stored on some vertices are lost, then a vertex can explore a slightly larger proximity for K different segments, namely, a vertex can retrieve K different segments from vertices within m_{p-1}, or m_{p-2}, ..., or m_1 hops, depending on how severe the data loss is. Such a file allocation scheme can tolerate data loss, and its performance degrades very gracefully with the increase of data loss.

- *Allocation of files encoded with multi-description codes:* Multi-description codes are a class of codes useful for multimedia files. Given a multimedia file, we can use a multi-description code to encode it and get N segments. (Here $N \geq K_1 > K_2 > \cdots > K_p$.) A user can recover the multimedia file with relatively low resolution (high distortion) by decoding any K_p segments, or recover the file with higher resolution by decoding any K_{p-1} segments, or recover the file with even higher resolution by decoding any K_{p-2} segments,

and so on $\cdots\cdots$. The more segments are used for decoding, the higher the resolution of the recovered file will be. If again we represent the N segments with N colors, then a layered diversity coloring will correspond to a placement of the file encoded with a multi-description code. For each vertex of the graph, the larger a proximity it explores for segments, the more different segments it will be able to retrieve, and the higher-resolutioned multimedia file it will recover. Such a file allocation scheme keeps a very good balance between delay (file-retrieving delay) and quality-of-service (quality of the recovered file). And clearly it also tolerates data loss with a graceful degradation performance.

Just as the p-center problem is useful in various industrial and scientific areas, the layered diversity coloring problem should also have applications to resource allocation (probably information allocation) in other fields besides the two scenarios we introduced above. One such field we can think of is the *threshold secret sharing* in cryptography [14] [15].

In this paper we study the layered diversity coloring problem for trees. The main result can be summarized as follows:

There exists a layered diversity coloring on a tree if and only if every vertex can find no less than K_i vertices (including itself) within m_i hops, for $i = 1, 2, \cdots, p$.

And we'll present a coloring algorithm for trees of time complexity $O(|V|(K_1 + \sum_{i=1}^{p} m_i))$.

The rest of the paper is organized as follows. In Section II, some basic terms and notations used in this paper are introduced. In Section III, the coloring algorithm is presented. In Section IV we prove some important lemmas useful for understanding the correctness of the algorithm. In Section V, we prove the correctness of the algorithm, and analyse its time complexity. In Section VI, we prove two additional properties of the layered diversity coloring which can be used to improve the coloring algorithm. In Section VII, we conclude this paper.

II Terms and Notations

In this section we introduce some basic terms and notations used in this paper. If a notation consistently refers to the same object when it appears in different places, we write it in bold font. For example, from now on G(V,E) always refers to the tree on which we're doing the layered diversity coloring. So we write G(V,E) as $\mathbf{G}(\mathbf{V}, \mathbf{E})$. But for a vertex $v \in \mathbf{V}$, since v can refer to different vertices when it appears in different places of this paper, we write v as 'v', not '$\mathbf{v}$'.

- $\mathbf{G}(\mathbf{V}, \mathbf{E})$ is the tree on which we do the layered diversity coloring.

  For any two vertices $u \in \mathbf{V}$ and $v \in \mathbf{V}$, the distance between them $d(u, v)$ is the number of edges in the path connecting them. (If $u = v$, then $d(u, v) = 0$.)

  The tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ has a vertex called its *root*. $\mathbf{G}(\mathbf{V}, \mathbf{E})$ is always placed in the 'upside down' way, with its root placed at the top. (See Fig. 2 as an example.) For any vertex $v \in \mathbf{V}$, we say v is at level $L(v)$, where $L(v) = d(v,$ *the root of* $\mathbf{G}(\mathbf{V}, \mathbf{E}))$. (So the root of tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ is at level 0, those vertices adjacent to the root are at level 1, and so on.)

  If there is a set of vertices $X \subseteq \mathbf{V}$ such that all vertices in X are at the same level, then we can use $L(X)$ to denote the level number of any vertex in X.

  For any two vertices $u \in \mathbf{V}$ and $v \in \mathbf{V}$, we say 'u is an ancestor of v, and v is a descendant of u' if and only if $d(u, v) = L(v) - L(u) > 0$; we say '$u$ is the parent of v, and v is a child of u' if and only if $d(u, v) = L(v) - L(u) = 1$; we say '$u$ is a quasi-ancestor of v, and v is a quasi-descendant of u' if and only if $d(u, v) = L(v) - L(u)$.

- The parameters $\mathbf{p}$, $\mathbf{m_1}$, $\mathbf{m_2}$, $\cdots$, $\mathbf{m_p}$, $\mathbf{K_1}$, $\mathbf{K_2}$, $\cdots$, $\mathbf{K_p}$ and $\mathbf{N}$ always have the meaning as in Definition 2—that is, we want to color the tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with $\mathbf{N}$ colors in such a way that for any vertex $v \in \mathbf{V}$ and for $1 \leq i \leq \mathbf{p}$, vertex v can find at least K_i different colors within m_i hops.

- For any vertex $v \in \mathbf{V}$ of the tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$, we define $\chi_m(v)$ to be the set of vertices within m hops from v—that is, $\chi_m(v) = \{u | u \in \mathbf{V}, d(u, v) \leq m\}$. We call $\chi_m(v)$ '*the neighborhood of v of radius m*'.

- '*A normal subtree of* $\mathbf{G}(\mathbf{V}, \mathbf{E})$ *with parameter set* (r, a, b, k, m)' is a subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ that satisfies the following 5 properties:

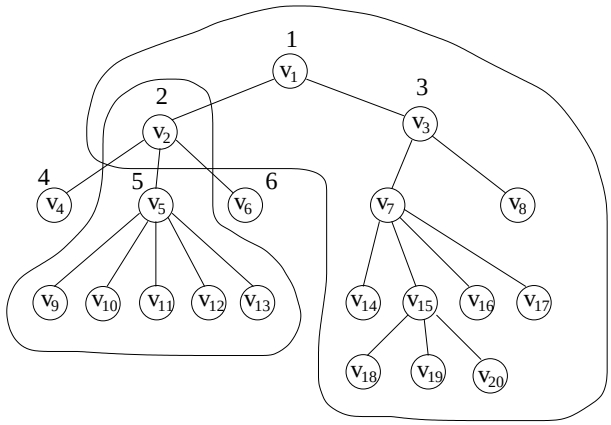

Figure 2: Example of a tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$, with parameters $\mathbf{N} = 6$, $\mathbf{K_1} = 6$, $\mathbf{K_2} = 4$, $\mathbf{m_1} = 3$ and $\mathbf{m_2} = 2$.

1. 'r' is a vertex in the subtree. (We call 'r' the *knot* of the subtree.)
2. Both 'a' and 'b' are non-negative integers. Both 'k' and 'm' are positive integers. And $a + b \leq m$.
3. {all vertices in the subtree}={all quasi-descendants of r } $\bigcup\{v | v \in \mathbf{V}, d(v, r) \leq m - a\}$.
4. If some of the vertices of the subtree are colored, then no two vertices of the subtree have the same color. There are less than k colors on the subtree, and $k \leq \mathbf{N}$. Any vertex of the subtree that is not in the set $\{v | v \in \mathbf{V}, L(v) - L(r) = d(v, r) \geq b\}$ is colored; there is at least one uncolored vertex in the set $\{v | v \in \mathbf{V}, L(v) - L(r) = d(v, r) = b\}$; and no vertex in the set $\{v | v \in \mathbf{V}, L(v) - L(r) = d(v, r) > b\}$ is colored.
5. If for a vertex v, all vertices in $\chi_m(v)$ are in the subtree, then $L(v) - L(r) = d(v, r) \geq a$.

The following is an example of the above terms and notations.

Example 1: Fig. 2 shows a partially colored tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$. The color of a vertex is the number beside the vertex. Here the parameters are $\mathbf{p} = 2$, $\mathbf{m_1} = 3$, $\mathbf{m_2} = 2$, $\mathbf{K_1} = 6$, $\mathbf{K_2} = 4$ and $\mathbf{N} = 6$.

The tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ is placed in the 'upside down' way. Its root is vertex v_1.

The distance between v_9 and v_{20} is $d(v_9, v_{20}) = 7$. $L(v_1) = 0$, $L(v_2) = L(v_3) = 1$, and $L(v_{20}) = 4$.

All vertices in the set $X = \{v_5, v_6, v_7\}$ are at the same level. Therefore $L(X) = L(v_5) = L(v_6) = L(v_7) = 2$.

v_2 is an ancestor (and also a parent, a quasi-ancestor) of v_5, and v_5 is a descendant (and also a child, a quasi-descendant) of v_2.

The neighborhood of v_1 of radius 2 is $\chi_2(v_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$. And $\chi_3(v_9) = \{v_1, v_2, v_4, v_5, v_6, v_9, v_{10}, v_{11}, v_{12}, v_{13}\}$.

Readers can verify by themselves that the subtree induced by the vertex set $\{v_2, v_5, v_9, v_{10}, v_{11}, v_{12}, v_{13}\}$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set (v_5, 1, 1, 4, 2), and the subtree induced by the vertex set $\{v_1, v_2, v_3, v_7, v_8, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, v_{20}\}$ is a normal subtree with parameter set (v_3, 1, 1, K_1, m_1).

□

III Layered Diversity Coloring Algorithm for Trees

In this section we first present the layered diversity coloring algorithm for trees. Then we use an example to illustrate how the algorithm works.

Algorithm: Layered Diversity Coloring on Tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$

Input: Tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$; parameters $\mathbf{p}$, $\mathbf{N}$, $\mathbf{K_1}$, $\mathbf{K_2}$, $\cdots$, $\mathbf{K_p}$, $\mathbf{m_1}$, $\mathbf{m_2}$, $\cdots$, $\mathbf{m_p}$.

Output: A coloring on vertices of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ such that for $1 \leq i \leq \mathbf{p}$ and for any vertex $v \in \mathbf{V}$, vertices in $\chi_{m_i}(v)$ have no less than K_i different colors.

Prerequisites: $\mathbf{p}$, $\mathbf{N}$, $\mathbf{K_1}$, $\mathbf{K_2}$, $\cdots$, $\mathbf{K_p}$, $\mathbf{m_1}$, $\mathbf{m_2}$, $\cdots$, $\mathbf{m_p}$ are all positive integers; $\mathbf{N} \geq \mathbf{K_1} > \mathbf{K_2} > \cdots > \mathbf{K_p}$; $\mathbf{m_1} > \mathbf{m_2} > \cdots > \mathbf{m_p}$; $\forall\ 1 \leq i \leq \mathbf{p}$ and $v \in \mathbf{V}$, $|\chi_{m_i}(v)| \geq K_i$.

Algorithm:

1. for $1 \leq i \leq \mathbf{p}$ do
2. { $r \leftarrow$ the root of $\mathbf{G}(\mathbf{V}, \mathbf{E})$;
3. $R \leftarrow \emptyset$;
4. $R_{temp} \leftarrow \{r\}$;
5. while $R_{temp} \neq \emptyset$ do
6. { $r_{temp} \leftarrow$ a vertex in R_{temp};
7. $R_{temp} \leftarrow R_{temp} - \{r_{temp}\}$;
8. Check the following vertices in order — r_{temp}, a child of r_{temp}, a descendant of r_{temp} at level $L(r_{temp})+2$, $\cdots$, a descendant of r_{temp} at level $L(r_{temp}) + m_i$. When any one of those vertices is being checked, see if the vertex can only find less than K_i different colors within m_i hops—if yes, let

v_1 denote that vertex, and immediately go to step 9 (the next command). If all those vertices can find at least K_i different colors within m_i hops, go to step 15.

9. Search for such a vertex v_2: v_2 is an uncolored quasi-descendant of r_{temp}, $L(r_{temp}) \leq L(v_2) \leq L(r_{temp}) + m_i$, and $L(v_2)$ is no greater than the level number of any uncolored quasi-descendant of r_{temp}. If such a vertex v_2 is found, go to step 10; otherwise go to step 15.

10. if $L(v_1) - L(r_{temp}) + L(v_2) - L(r_{temp}) \leq m_i$ do

11. { $R \leftarrow R + \{r_{temp}\}$;

12. $p_1(r_{temp}) \leftarrow L(v_1) - L(r_{temp})$;

13. $p_2(r_{temp}) \leftarrow L(v_2) - L(r_{temp})$;

14. }

15. else do

16. { $R_{temp} \leftarrow R_{temp} \bigcup \{$ children of r_{temp} };

17. }

18. }

19. while $R \neq \emptyset$ do

20. { $r \leftarrow$ a vertex in R;

21. $R \leftarrow R - \{r\}$;

22. $v_0 \leftarrow$ a quasi-descendant of r at level $L(r) + p_1(r)$;

23. $C \leftarrow \{$ the $\mathbf{N}$ colors $\} - \{$ colors on vertices within m_i hops from $v_0\}$;

24. First use colors in C to color those uncolored quasi-descendants of r at level $L(r) + p_2(r)$, then to color descendants of r at level $L(r) + p_2(r) + 1$, then to color descendants of r at level $L(r) + p_2(r) + 2$, and so on ..... with each color in C used at most once. Keep coloring until all colors in C are used once or when there is no more uncolored quasi-descendant of r.

25. $R_{temp} \leftarrow \{r\}$;

26. while $R_{temp} \neq \emptyset$ do

27. { $r_{temp} \leftarrow$ a vertex in R_{temp};

28. $R_{temp} \leftarrow R_{temp} - \{r_{temp}\}$;

29. Check the following vertices in order — r_{temp}, a child of r_{temp}, a descendant of r_{temp} at level $L(r_{temp}) + 2$, $\cdots$, a descendant of r_{temp} at level $L(r_{temp}) + m_i$. When any one of those vertices is being checked, see if the vertex can only find less than K_i different colors within m_i hops–if yes, let v_1 denote that vertex, and immediately go to step 30 (the next command). If all those vertices can find at least K_i different colors within m_i hops, go to step 36.

30. Search for such a vertex v_2: v_2 is an uncolored quasi-descendant of r_{temp}, $L(r_{temp}) \leq L(v_2) \leq L(r_{temp}) + m_i$, and $L(v_2)$ is no greater than the level number of any uncolored quasi-descendant of r_{temp}. If such a vertex v_2 is found, go to step 31; otherwise go to step 36.

31. if $L(v_1) - L(r_{temp}) + L(v_2) - L(r_{temp}) \leq m_i$ do

32. { $R \leftarrow R + \{r_{temp}\}$;

33. $p_1(r_{temp}) \leftarrow L(v_1) - L(r_{temp})$;

34. $p_2(r_{temp}) \leftarrow L(v_2) - L(r_{temp})$;

35. }

36. else do

37. { $R_{temp} \leftarrow R_{temp} \bigcup \{$ children of r_{temp} };

38. }

39. }

40. }

41. }

42. Arbitrarily color those uncolored vertices.

□

The algorithm has $\mathbf{p}$ iterations. (See step 1 of the algorithm: 'for $1 \leq i \leq \mathbf{p}$ .....'.) During the i-th iteration, the algorithm only considers the parameters K_i and m_i. ($1 \leq i \leq \mathbf{p}$.) And after the i-th iteration, every vertex of the tree can find at least K_i different colors within m_i hops. (This will be proved later in this paper.)

Below we use an example to show how the algorithm colors a tree.

Example 2: Use the algorithm to get a layered diversity coloring on the tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ shown in Fig. 3 (a). Here the parameters are $\mathbf{p} = 3$, $\mathbf{N} = 18$, $\mathbf{K_1} = 14$, $\mathbf{K_2} = 7$, $\mathbf{K_3} = 3$, $\mathbf{m_1} = 7$, $\mathbf{m_2} = 3$, $\mathbf{m_3} = 2$.

Below we list the main process of the algorithm's coloring the tree.

1. In the algorithm, $i = 1$.

2. R becomes $\{v_1\}$, $p_1(v_1)$ becomes 0, $p_2(v_1)$ becomes 0.

3. Use the $\mathbf{N}$ colors to color the quasi-descendants of v_1, level by level. (See Fig. 3 (b). The number on a vertex is the color of the vertex.)

4. R becomes $\{v_4\}$, $p_1(v_4)$ becomes 5, $p_2(v_4)$ becomes 1.

5. C becomes $\{5, 6, 7, 8, 9, 10, 11, 12\}$. Use colors in C to color quasi-descendants of v_4, level by level. (See Fig. 4 (a).)

6. R becomes $\{v_{25}\}$, $p_1(v_{25})$ becomes 5, $p_2(v_{25})$ becomes 1.

7. C becomes $\{1, 2, 3, 5, 6, 10, 11, 14, 15, 16, 17, 18\}$. Use colors in C to color quasi-descendants of v_{25}, level by level. (See Fig. 4 (b).)

8. R becomes $\{v_{31}\}$, $p_1(v_{31})$ becomes 3, $p_2(v_{31})$ becomes 3.

9. C becomes $\{6, 10, 16, 17, 18\}$. Use colors in C to color quasi-descendants of v_{31}, level by level. (See Fig. 5 (a).)

Readers can verify that now every vertex of the tree can find at least $\mathbf{K_1} = 14$ different colors within $\mathbf{m_1} = 7$ hops.

10. In the algorithm, $i = 2$.

11. R becomes $\{v_7, v_{26}\}$, $p_1(v_7)$ becomes 1, $p_2(v_7)$ becomes 1, $p_1(v_{26})$ becomes 1, $p_2(v_{26})$ becomes 1.

12. C becomes $\{3, 4, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18\}$. Use colors in C to color quasi-descendants of v_7.

13. R becomes $\{v_{26}\}$. C becomes $\{1, 2, 3, 5, 6, 7, 8, 9, 12, 13, 14, 15, 17, 18\}$. Use colors in C to color quasi-descendants of v_{26}. (See Fig. 5 (b).)

Readers can verify that now every vertex of the tree can find at least $\mathbf{K_2} = 7$ different colors within $\mathbf{m_2} = 3$ hops.

14. In the algorithm, $i = 3$. R becomes $\emptyset$. Readers can verify that every vertex of the tree can find at least $\mathbf{K_3} = 3$ different colors within $\mathbf{m_3} = 2$ hops. The tree $\mathbf{G(V, E)}$ has been successfully colored. The algorithm ends.

□

The algorithm is seemingly simple. However, there are two interesting questions we need to ask:

1. For $1 \leq i \leq \mathbf{p}$, during the i-th iteration, the algorithm is trying to color the tree to make every vertex be able to find at least K_i different colors within m_i hops, and it doesn't use any parameter K_j or m_j where $j \neq i$. How does the algorithm guarantee that after the i-th iteration, there is still a feasible way to color the remaining uncolored vertices to make every vertex be able to find at least K_j different colors within m_j hops, for $j = i + 1, i + 2, \cdots, \mathbf{p}$? (That is, why does the greedy coloring method work?)

2. How to prove that after the i-th iteration, $(1 \leq i \leq \mathbf{p})$, every vertex can find at least K_i different colors within m_i hops?

The analysis in the rest of the paper will give answers to those two questions.

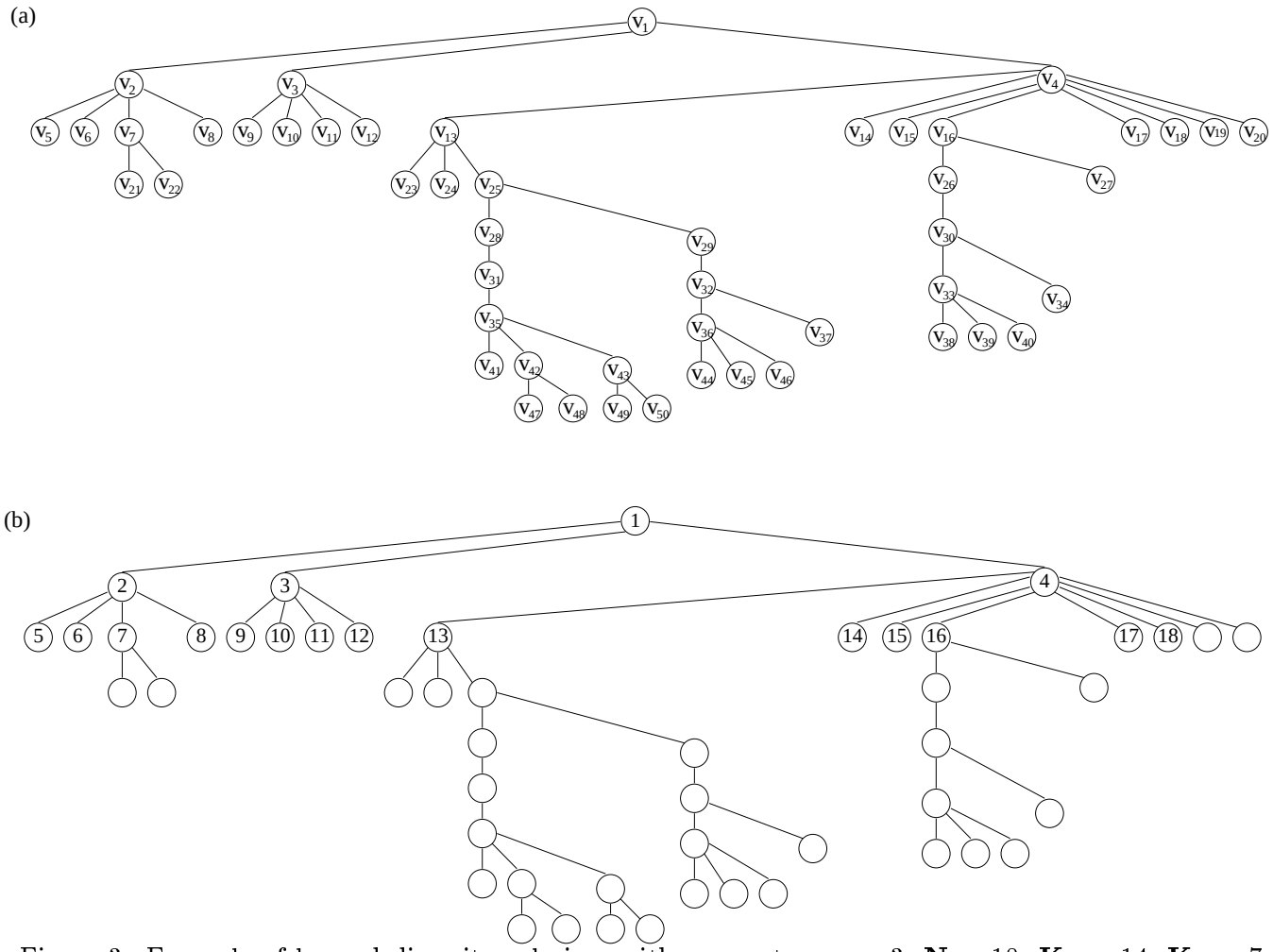

Figure 3: Example of layered diversity coloring, with parameters $\mathbf{p} = 3$, $\mathbf{N} = 18$, $\mathbf{K_1} = 14$, $\mathbf{K_2} = 7$, $\mathbf{K_3} = 3$, $\mathbf{m_1} = 7$, $\mathbf{m_2} = 3$, $\mathbf{m_3} = 2$.

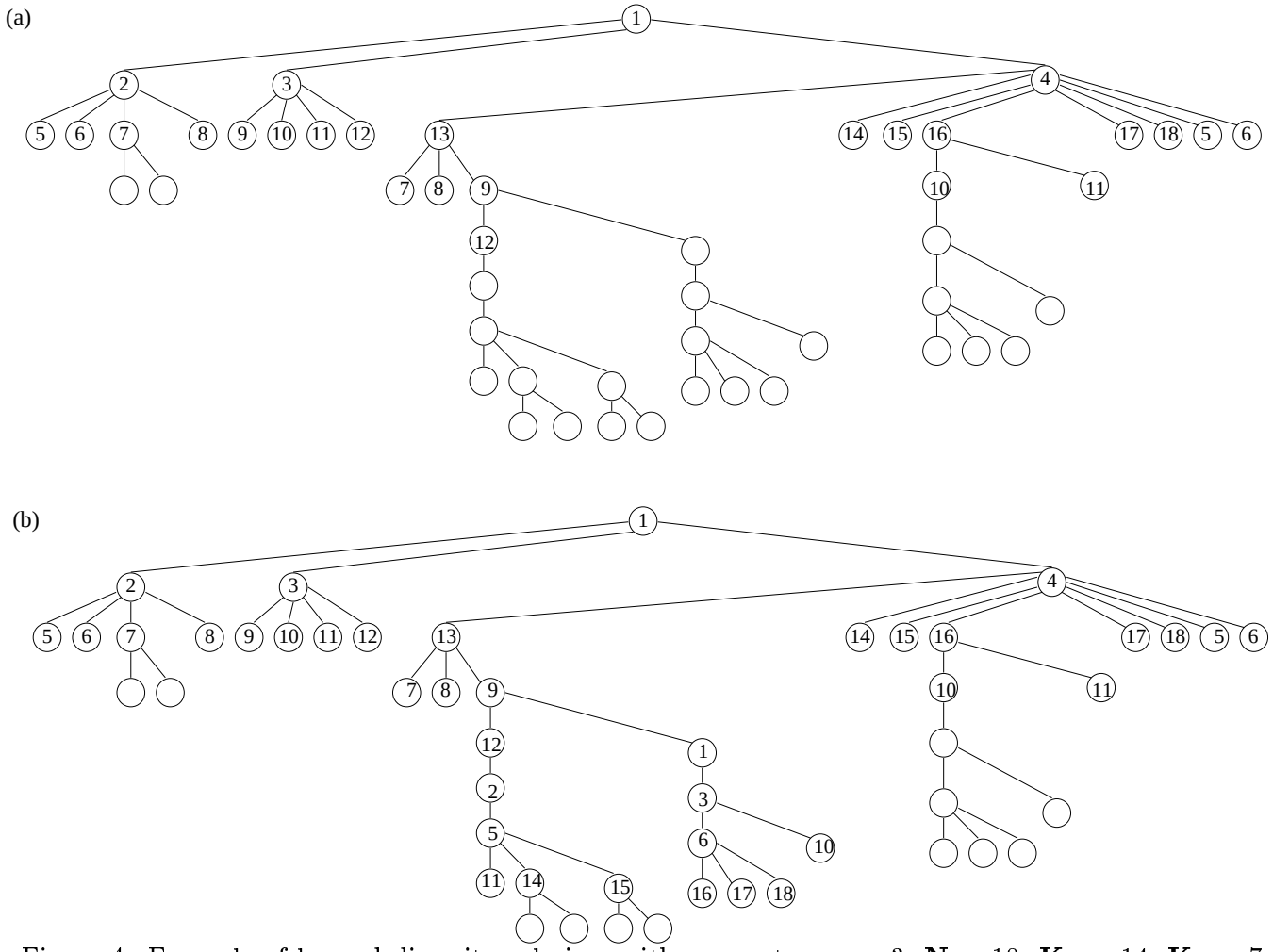

Figure 4: Example of layered diversity coloring, with parameters $\mathbf{p} = 3$, $\mathbf{N} = 18$, $\mathbf{K_1} = 14$, $\mathbf{K_2} = 7$, $\mathbf{K_3} = 3$, $\mathbf{m_1} = 7$, $\mathbf{m_2} = 3$, $\mathbf{m_3} = 2$.

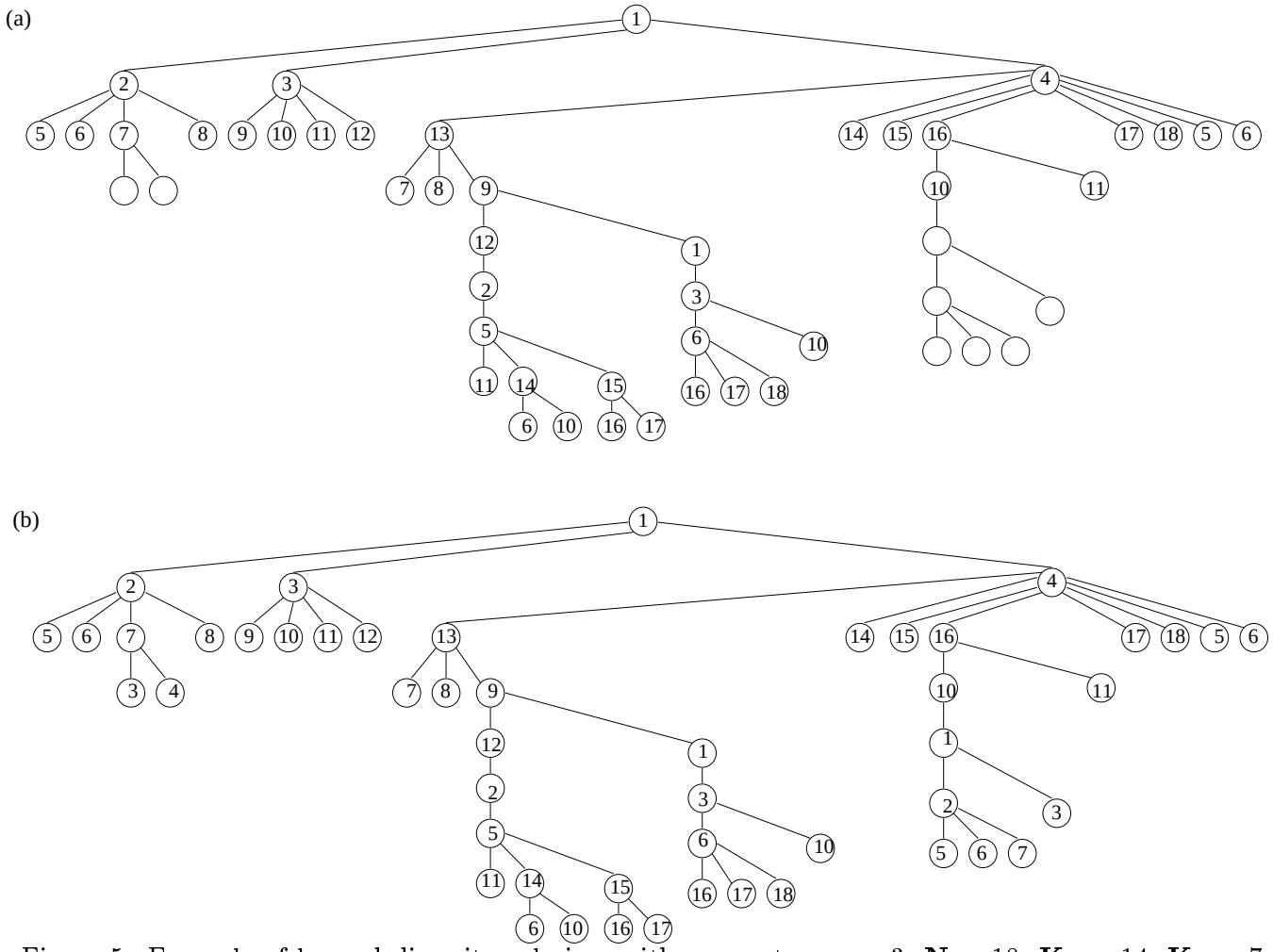

Figure 5: Example of layered diversity coloring, with parameters $\mathbf{p} = 3$, $\mathbf{N} = 18$, $\mathbf{K_1} = 14$, $\mathbf{K_2} = 7$, $\mathbf{K_3} = 3$, $\mathbf{m_1} = 7$, $\mathbf{m_2} = 3$, $\mathbf{m_3} = 2$.

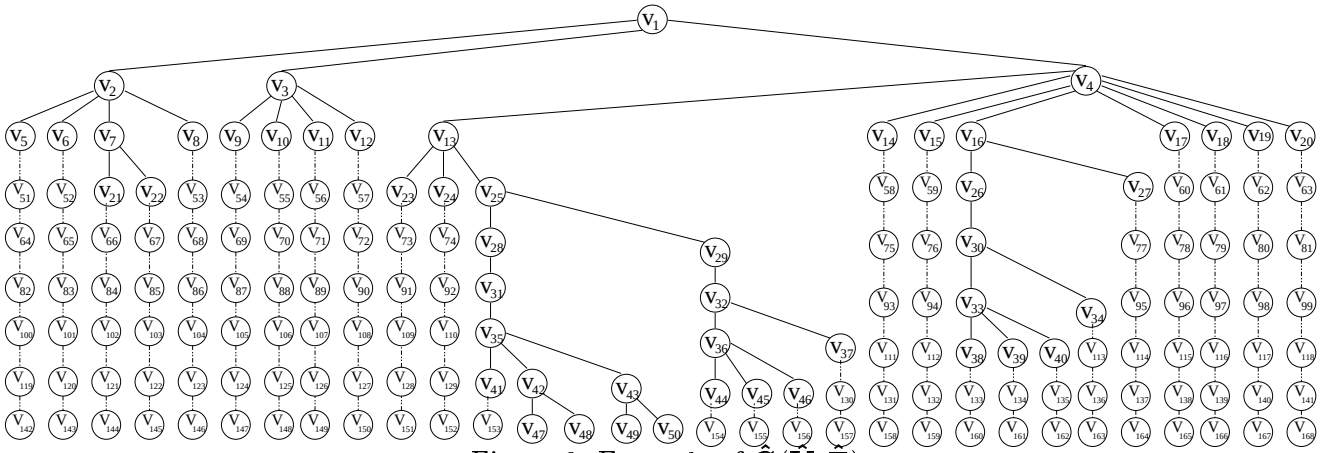

Figure 6: Example of $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$.

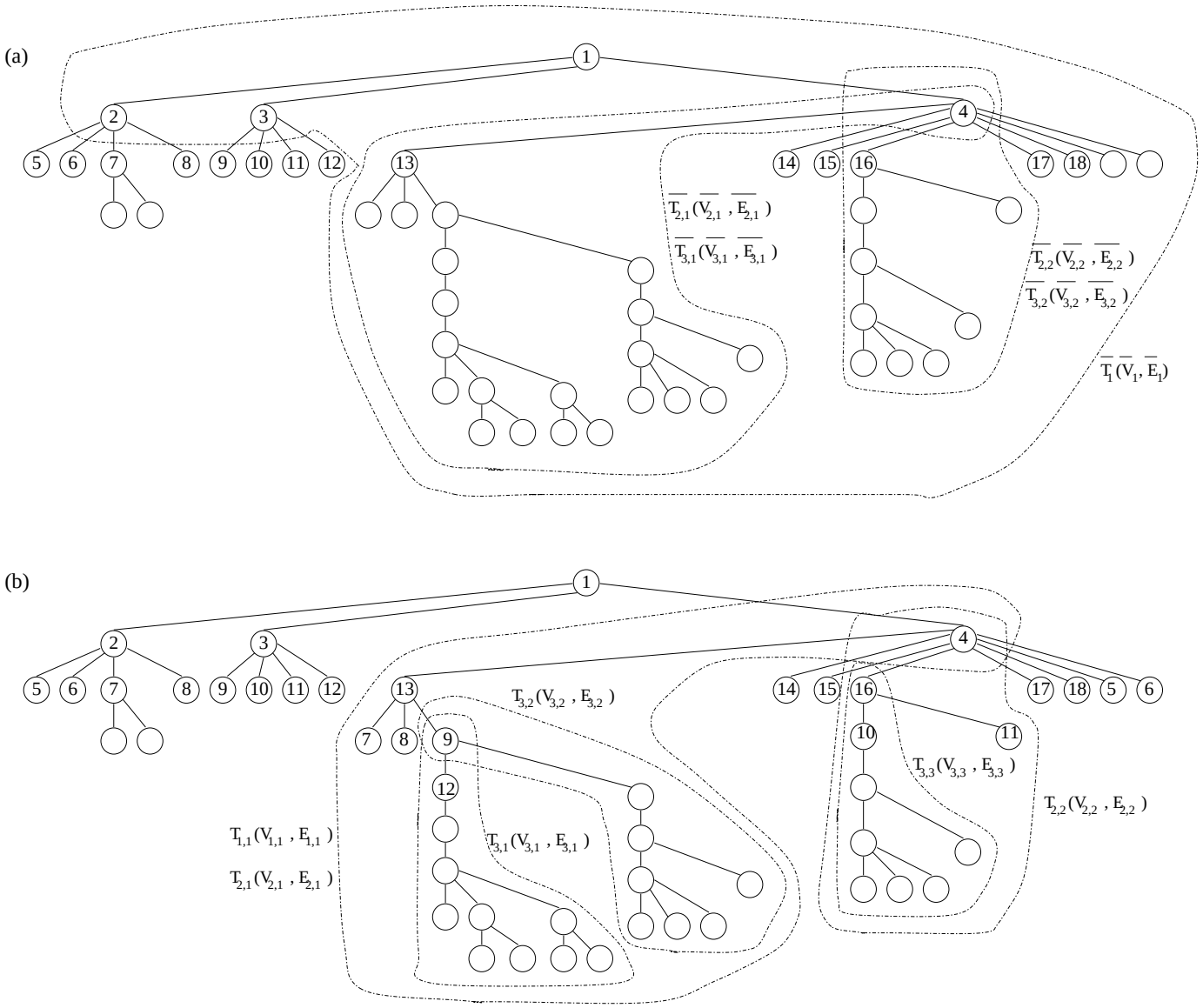

Figure 7: Example of layered diversity coloring, with parameters $\mathbf{p} = 3$, $\mathbf{N} = 18$, $\mathbf{K_1} = 14$, $\mathbf{K_2} = 7$, $\mathbf{K_3} = 3$, $\mathbf{m_1} = 7$, $\mathbf{m_2} = 3$, $\mathbf{m_3} = 2$.

IV Some Lemmas

In this section, we'll prove some important lemmas useful for understanding the coloring algorithm. Again, as said in Section II, if a notation refers to the same object when it appears in different places of the paper, we write it in bold font.

As mentioned before, $\mathbf{G(V,E)}$ is the tree for which we want to find a layered diversity coloring. Let the following conditions be true: $\forall\ v \in \mathbf{V}$ and $1 \le i \le \mathbf{p}$, $|\chi_{m_i}(v)| \ge K_i$. (Clearly that is a necessary condition for there to exist a layered diversity coloring on the tree.) Let's say $\mathbf{G(V,E)}$ has been partially colored. And define $\mathbf{L_{max}}$ to be the maximum level number of all vertices of $\mathbf{G(V,E)}$, that is, $\mathbf{L_{max}} = max_{v\in\mathbf{V}} L(v)$.

We create a new tree $\hat{\mathbf{G}}(\hat{\mathbf{V}},\hat{\mathbf{E}})$ by adding some new vertices and edges to $\mathbf{G(V,E)}$ in the following way: for every vertex $v \in \mathbf{V}$, if v is a leaf and $0 < L(v) < \mathbf{L_{max}}$, add descendants to v and add the corresponding edges, so that v has exactly one descendant at each level from level $L(v)+1$ to level $\mathbf{L_{max}}$. We denote the set of new vertices by $\mathbf{V_{IR}}$. (So $\hat{\mathbf{V}} = \mathbf{V} \bigcup \mathbf{V_{IR}}$.) And for any vertex $v \in \hat{\mathbf{V}}$ and for $i = 1, 2, \cdots, \mathbf{p}$, we define $\hat{\chi}_{m_i}(v) = \{u | u \in \hat{\mathbf{V}}, d(u,v) \le m_i\}$.

Example 3: If we let $\mathbf{G(V,E)}$ be the tree shown in Fig. 3 (a), then $\hat{\mathbf{G}}(\hat{\mathbf{V}},\hat{\mathbf{E}})$ will be the tree shown in Fig. 6. Those dashed edges are the new edges, and the vertices below the dashed edges are the new vertices. Here $\mathbf{L_{max}} = 8$, $\mathbf{V} = \{v_1, v_2, \cdots, v_{50}\}$, $\mathbf{V_{IR}} = \{v_{51}, v_{52}, \cdots, v_{168}\}$, and $\hat{\mathbf{V}} = \{v_1, v_2, \cdots, v_{168}\}$.
□

Let $\mathbf{J}$ be an integer, $1 \le \mathbf{J} \le \mathbf{p}$. Let $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ be a *normal subtree* of $\mathbf{G(V,E)}$ with parameter set $(\overline{\mathbf{r_J}}, \overline{\mathbf{a_J}}, \overline{\mathbf{b_J}}, \mathbf{K_J}, \mathbf{m_J})$. Let $\overline{\mathbf{C_J}}$ denote the set of colors that appear on the subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$. Let this condition be true: if $\overline{\mathbf{r_J}}$ is not the root of $\mathbf{G(V,E)}$, then $\overline{\mathbf{a_J}} + \overline{\mathbf{b_J}} + 2 > \mathbf{m_J}$.

Let $\overline{\mathbf{T_{J+1,2}}}(\overline{\mathbf{V_{J+1,2}}}, \overline{\mathbf{E_{J+1,2}}})$, $\overline{\mathbf{T_{J+1,1}}}(\overline{\mathbf{V_{J+1,1}}}, \overline{\mathbf{E_{J+1,1}}})$, $\overline{\mathbf{T_{J+1,3}}}(\overline{\mathbf{V_{J+1,3}}}, \overline{\mathbf{E_{J+1,3}}})$, $\cdots$, $\overline{\mathbf{T_{J+2,1}}}(\overline{\mathbf{V_{J+2,1}}}, \overline{\mathbf{E_{J+2,1}}})$, $\overline{\mathbf{T_{J+2,2}}}(\overline{\mathbf{V_{J+2,2}}}, \overline{\mathbf{E_{J+2,2}}})$, $\overline{\mathbf{T_{J+2,3}}}(\overline{\mathbf{V_{J+2,3}}}, \overline{\mathbf{E_{J+2,3}}})$, $\cdots\cdots$, $\overline{\mathbf{T_{p,1}}}(\overline{\mathbf{V_{p,1}}}, \overline{\mathbf{E_{p,1}}})$, $\overline{\mathbf{T_{p,2}}}(\overline{\mathbf{V_{p,2}}}, \overline{\mathbf{E_{p,2}}})$, $\overline{\mathbf{T_{p,3}}}(\overline{\mathbf{V_{p,3}}}, \overline{\mathbf{E_{p,3}}})$, $\cdots$, be subtrees of $\mathbf{G(V,E)}$ that satisfy the following 4 conditions:

1. For $i = \mathbf{J}+1, \mathbf{J}+2, \cdots, \mathbf{p}$ and $j = 1, 2, 3, \cdots$, $\overline{T_{i,j}}(\overline{V_{i,j}}, \overline{E_{i,j}})$ is a *normal subtree* of $\mathbf{G(V,E)}$ with parameter set $(\overline{r_{i,j}}, \overline{a_{i,j}}, \overline{b_{i,j}}, K_i, m_i)$, where $\overline{V_{i,j}} \subseteq \overline{\mathbf{V_J}}$, and $L(\overline{r_{i,j}}) - L(\overline{\mathbf{r_J}}) = d(\overline{r_{i,j}}, \overline{\mathbf{r_J}})$. If $\overline{r_{i,j}}$ is not the root of $\mathbf{G(V,E)}$, then $\overline{a_{i,j}} + \overline{b_{i,j}} + 2 > m_{i,j}$.

2. For $i = \mathbf{J}+1, \mathbf{J}+2, \cdots, \mathbf{p}$, $j_1 = 1, 2, 3, \cdots$ and $j_2 = 1, 2, 3, \cdots$, if $j_1 \ne j_2$, then all vertices in the set $\overline{V_{i,j_1}} \bigcap \overline{V_{i,j_2}}$ are colored vertices.

3. For $i_1 = \mathbf{J}+1, \mathbf{J}+2, \cdots, \mathbf{p}$, $j_1 = 1, 2, 3, \cdots$, $i_2 = \mathbf{J}+1, \mathbf{J}+2, \cdots, \mathbf{p}$ and $j_2 = 1, 2, 3, \cdots$, if $i_1 < i_2$, then exactly one of the following two cases is true:

   (a) All vertices in the set $\overline{V_{i_1,j_1}} \bigcap \overline{V_{i_2,j_2}}$ are colored vertices.

   (b) $\overline{V_{i_2,j_2}} \subseteq \overline{V_{i_1,j_1}}$, and $L(\overline{r_{i_2,j_2}}) - L(\overline{r_{i_1,j_1}}) = d(\overline{r_{i_2,j_2}}, \overline{r_{i_1,j_1}})$.

4. For any vertex $v \in \overline{\mathbf{V_J}}$ and for $i = \mathbf{J}+1, \mathbf{J}+2, \cdots, \mathbf{p}$, if $\chi_{m_i}(v) \subseteq \overline{\mathbf{V_J}}$ and vertices in $\chi_{m_i}(v)$ have less than K_i different colors, then there exists a subtree $\overline{T_{i,*}}(\overline{V_{i,*}}, \overline{E_{i,*}})$ (here '$*$' means some unspecified parameter) such that $\chi_{m_i}(v) \subseteq \overline{V_{i,*}}$.

Example 4: Let $\mathbf{G(V,E)}$ be the same tree as in Fig. 3, and let the parameters be the same (that is, $\mathbf{p} = 3$, $\mathbf{N} = 18$, $\mathbf{K_1} = 14$, $\mathbf{K_2} = 7$, $\mathbf{K_3} = 3$, $\mathbf{m_1} = 7$, $\mathbf{m_2} = 3$, $\mathbf{m_3} = 2$.). Let $\mathbf{G(V,E)}$ be partially colored as in Fig. 3 (b). Let $\mathbf{J} = 1$, and let $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}}) = \overline{T_1}(\overline{V_1}, \overline{E_1})$ be the subtree induced by the vertex set $\mathbf{V_J} = V_1 = \{$ quasi-descendants of $v_4\} \bigcup \{v_1, v_2, v_3\}$; let $\overline{\mathbf{T_{J+1,1}}}(\overline{\mathbf{V_{J+1,1}}}, \overline{\mathbf{E_{J+1,1}}}) = \overline{T_{2,1}}(\overline{V_{2,1}}, \overline{E_{2,1}})$ be the subtree induced by the vertex set $\overline{\mathbf{V_{J+1,1}}} = \overline{V_{2,1}} = \{$ quasi-descendants of $v_{13}\} \bigcup \{v_4\}$; let $\overline{\mathbf{T_{J+1,2}}}(\overline{\mathbf{V_{J+1,2}}}, \overline{\mathbf{E_{J+1,2}}}) = \overline{T_{2,2}}(\overline{V_{2,2}}, \overline{E_{2,2}})$ be the subtree induced by the vertex set $\overline{\mathbf{V_{J+1,2}}} = \overline{V_{2,2}} = \{$ quasi-descendants of $v_{16}\} \bigcup \{v_4\}$; let $\overline{\mathbf{T_{J+2,1}}}(\overline{\mathbf{V_{J+2,1}}}, \overline{\mathbf{E_{J+2,1}}}) = \overline{T_{3,1}}(\overline{V_{3,1}}, \overline{E_{3,1}})$ be the subtree induced by the vertex set $\overline{\mathbf{V_{J+2,1}}} = \overline{V_{3,1}} = \{$ quasi-descendants of $v_{13}\} \bigcup \{v_4\}$; let $\overline{\mathbf{T_{J+2,2}}}(\overline{\mathbf{V_{J+2,2}}}, \overline{\mathbf{E_{J+2,2}}}) = \overline{T_{3,2}}(\overline{V_{3,2}}, \overline{E_{3,2}})$ be the subtree induced by the vertex set $\overline{\mathbf{V_{J+2,2}}} = \overline{V_{3,2}} = \{$ quasi-descendants of $v_{16}\} \bigcup \{v_4\}$. Those five subtrees—$\overline{T_1}(\overline{V_1}, \overline{E_1})$, $\overline{T_{2,1}}(\overline{V_{2,1}}, \overline{E_{2,1}})$, $\overline{T_{2,2}}(\overline{V_{2,2}}, \overline{E_{2,2}})$, $\overline{T_{3,1}}(\overline{V_{3,1}}, \overline{E_{3,1}})$ and $\overline{T_{3,2}}(\overline{V_{3,2}}, \overline{E_{3,2}})$ — are circled and shown in Fig. 7 (a). (Note that $\overline{T_{2,1}}(\overline{V_{2,1}}, \overline{E_{2,1}})$ and $\overline{T_{3,1}}(\overline{V_{3,1}}, \overline{E_{3,1}})$ happen to be the same subtree; and $\overline{T_{2,2}}(\overline{V_{2,2}}, \overline{E_{2,2}})$ and $\overline{T_{3,2}}(\overline{V_{3,2}}, \overline{E_{3,2}})$ happen to be the same subtree.)

It's easy to verify that $\overline{T_1}(\overline{V_1}, \overline{E_1})$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set $(v_4, 5, 1, \mathbf{K_1}, \mathbf{m_1})$, $\overline{T_{2,1}}(\overline{V_{2,1}}, \overline{E_{2,1}})$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set $(v_{13}, 2, 1, \mathbf{K_2}, \mathbf{m_2})$, $\overline{T_{2,2}}(\overline{V_{2,2}}, \overline{E_{2,2}})$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set $(v_{16}, 2, 1, \mathbf{K_2}, \mathbf{m_2})$, $\overline{T_{3,1}}(\overline{V_{3,1}}, \overline{E_{3,1}})$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set $(v_{13}, 1, 1, \mathbf{K_3}, \mathbf{m_3})$, $\overline{T_{3,2}}(\overline{V_{3,2}}, \overline{E_{3,2}})$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set $(v_{16}, 1, 1, \mathbf{K_3}, \mathbf{m_3})$. It's also easy to verify that those five subtrees satisfy all the conditions described before (at the beginning of this section.)

□

Now we color the normal subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ in the following way: we use colors in the set $\{1, 2, \cdots, \mathbf{N}\} - \overline{\mathbf{C_J}}$ —that is, colors that have not been used on the normal subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ — to color the uncolored vertices of $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$, with each color used at most once; we color the vertices of $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ level by level, from small level to large level (that is, first color vertices at level $L(\overline{\mathbf{r_J}}) + \overline{\mathbf{b_J}}$, then color vertices at level $L(\overline{\mathbf{r_J}}) + \overline{\mathbf{b_J}} + 1$, then color vertices at level $L(\overline{\mathbf{r_J}}) + \overline{\mathbf{b_J}} + 2$, and so on ...); keep coloring until all colors are used or all vertices in the subtree are colored.

Example 5: Let's follow Example 4. We color the subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}}) = \overline{T_1}(\overline{V_1}, \overline{E_1})$ in the way described above. Then the colored tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ will be as shown in Fig. 7 (b). (Note that the coloring is the same as in Fig. 4 (a).)

□

After the coloring, the subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ is not necessarily a normal subtree with parameter set $(\overline{\mathbf{r_J}}, \overline{\mathbf{a_J}}, \overline{\mathbf{b_J}}, \mathbf{K_J}, \mathbf{m_J})$ anymore (and actually it's not). Let's study some of its properties.

Lemma 1 *Let u and v be two vertices, $u \in \mathbf{V}$, $v \in \mathbf{V}$. i is an integer, $\mathbf{J} \le i \le \mathbf{p}$. u is an ancestor of v. $\chi_{m_i}(u) \subseteq \overline{\mathbf{V_J}}$. Then, if vertices in $\chi_{m_i}(u)$ have less than K_i different colors, vertices in $\chi_{m_i}(v)$ also have less than K_i different colors.*

Proof: We'll prove Lemma 1 with contradiction. Suppose that vertices in $\chi_{m_i}(u)$ have less than K_i different colors, but vertices in $\chi_{m_i}(v)$ have no less than K_i different colors. Then there must exist a colored vertex w such that $w \in \chi_{m_i}(v)$ and $w \notin \chi_{m_i}(u)$. Since u is an ancestor of v, w must be a descendant of u, and $L(w) > L(u) + m_i$.

It's not difficult to see that u is a quasi-descendant of $\overline{\mathbf{r_J}}$. As the normal subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ was colored 'level by level', and there is a colored vertex w at level greater than $L(u) + m_i$, we know that any uncolored vertex in $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ is now at level greater than $L(u) + m_i$. So there is no uncolored vertex in $\chi_{m_i}(u)$. No two vertices in the subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ have the same color. So since $\chi_{m_i}(u) \subseteq \overline{\mathbf{V_J}}$ and $|\chi_{m_i}(u)| \ge K_i$, vertices in $\chi_{m_i}(u)$ have no less than K_i different colors, and that's a contradiction. So we've proved Lemma 1.

□

Now for $\mathbf{J} \le i \le \mathbf{p}$, we define A_i to be such a set: $A_i = \{v | v \in \mathbf{V}; \chi_{m_i}(v) \subseteq \overline{\mathbf{V_J}}$; vertices in $\chi_{m_i}(v)$ have less than K_i different colors; if v has a parent, then either $\chi_{m_i}($ the parent of $v) \not\subseteq \overline{\mathbf{V_J}}$, or vertices in $\chi_{m_i}($ the parent of $v)$ have no less than K_i different colors. $\}$.

Example 6: Let's follow Example 5. Then $\mathbf{A_1} = \{v_{47}, v_{48}, v_{49}, v_{50}\}$, $\mathbf{A_2} = \{v_{28}, v_{29}, v_{30}\}$, $\mathbf{A_3} = \{v_{31}, v_{32}, v_{30}\}$. (Readers can see the current coloring in Fig. 7 (b) and the names of vertices in Fig. 3 (a).)

□

From Lemma 1, it's easy to see that the following two statements are true:

- For any $u \in A_i$, $v \in A_i$, if $u \ne v$, then u is neither a quasi-ancestor nor a quasi-descendant of v. ($\mathbf{J} \le i \le \mathbf{p}$.)
- For any $\mathbf{J} \le i \le \mathbf{p}$ and for any vertex $v \in \mathbf{V}$, if $\chi_{m_i}(v) \subseteq \overline{\mathbf{V_J}}$ and vertices in $\chi_{m_i}(v)$ have less than K_i different colors, then v must be a quasi-descendant of some vertex in A_i, and vice versa.

Lemma 2 $\mathbf{J} \le i \le \mathbf{p}$, $u_1 \in A_i$, $u_2 \in A_i$. *If there exists an uncolored vertex in the set $\chi_{m_i}(u_1) \bigcap \chi_{m_i}(u_2)$, then $L(u_1) = L(u_2)$.*

Proof: We'll prove Lemma 2 with contradiction. Suppose that there exists an uncolored vertex $w_0 \in \chi_{m_i}(u_1) \bigcap \chi_{m_i}(u_2)$, but $L(u_1) > L(u_2)$.

Let w_1 be the ancestor of u_1 at level $L(u_2)$. And let w_r be the common ancestor of both u_1 and u_2 that satisfies this condition: for any common ancestor of both u_1 and u_2, its level number is no greater than $L(w_r)$. It's easy to see that w_r is a quasi-descendant of $\overline{\mathbf{r_J}}$, and w_1 is a descendant of w_r.

Clearly w_r is on the path connecting u_1 and u_2. So w_r is either on the path connecting u_1 and w_0, or on the path connecting u_2 and w_0. As $w_0 \in \chi_{m_i}(u_1) \bigcap \chi_{m_i}(u_2)$, we get $d(u_2, w_r) + d(w_r, w_0) \leq m_i$. Since $L(w_0) - L(w_r) \leq d(w_r, w_0)$, we get $L(w_0) \leq L(w_r) + m_i - d(u_2, w_r)$. Since w_0 is uncolored, and the subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ has been colored 'level by level', we know there is no colored vertex that is a quasi-descendant of $\overline{\mathbf{r_J}}$ at level greater than $L(w_0)$.

Now it's easy to verify that for any vertex in $\chi_{m_i}(w_1)$, if the vertex is not a descendant of w_r or if it is colored, the vertex is also in $\chi_{m_i}(u_2)$. So clearly $\chi_{m_i}(w_1) \subseteq \overline{\mathbf{V_J}}$, and vertices in $\chi_{m_i}(w_1)$ have less than K_i different colors because vertices in $\chi_{m_i}(u_2)$ do. However, w_1 is an ancestor of u_1. That is a contradiction to the fact that $u_1 \in A_i$. So we've proved Lemma 2.

□

For $\mathbf{J} \leq i \leq \mathbf{p}$, we define B_i to be such a set: $B_i = \{v | v \in \mathbf{V_{IR}}$; v is a descendant of $\overline{\mathbf{r_J}}$ in the tree $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$; $\exists\ u \in A_i$ such that $L(v) = L(u)$ and there is an uncolored vertex in $\chi_{m_i}(u) \bigcap \hat{\chi}_{m_i}(v)$.$\}$.

Example 7: Let's follow Example 6. Then $\mathbf{B_1} = \{v_{153}, v_{154}, v_{155}, v_{156}, v_{157}\}$, $\mathbf{B_2} = \emptyset$, $\mathbf{B_3} = \emptyset$. (Readers can see the current coloring in Fig. 7 (b) and the names of vertices in Fig. 6.)

□

Lemma 3 *Let u and v be two vertices in B_i. ($u \neq v$, $\mathbf{J} \leq i \leq \mathbf{p}$.) Then u is neither an ancestor nor a descendant of v in the tree $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$. If there is an uncolored vertex in $\hat{\chi}_{m_i}(u) \bigcap \hat{\chi}_{m_i}(v) \bigcap \mathbf{V}$, then $L(u) = L(v)$.*

Proof: We'll prove the first part of Lemma 3 with contradiction. Suppose that u is an ancestor of v in the tree $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$. Then there exists $a_u \in A_i$ such that $L(a_u) = L(u)$ and there is an uncolored vertex in $\chi_{m_i}(a_u) \bigcap \hat{\chi}_{m_i}(u)$. (Then similar to Lemma 2, we can prove that a colored vertex is in $\chi_{m_i}(u)$ if and only if it's in $\chi_{m_i}(a_u)$.) And there exists $a_v \in A_i$ such that $L(a_v) = L(v)$ and there is an uncolored vertex w_0 in $\chi_{m_i}(a_v) \bigcap \hat{\chi}_{m_i}(v)$. Let w_v be the ancestor of a_v at level $L(u)$.

Both u and w_v are on the path connecting a_v and v. So w_0 is within m_i hops from both u and w_v. So similar to Lemma 2, we can prove that a colored vertex is in $\chi_{m_i}(w_v)$ if and only if it's in $\chi_{m_i}(u)$, and $\chi_{m_i}(w_v) \subseteq \overline{\mathbf{V_J}}$. So vertices in $\chi_{m_i}(w_v)$ have less than K_i different colors because vertices in $\chi_{m_i}(a_u)$ do. However w_v is an ancestor of a_v, and that contradicts the fact that $a_v \in A_i$. So we've proved the first part of Lemma 3.

We'll prove the second part of Lemma 3 also with contradiction. The proof uses similar techniques as used before, so for simplicity we give only its sketch. Suppose there is an uncolored vertex w_0 in $\hat{\chi}_{m_i}(u) \bigcap \hat{\chi}_{m_i}(v) \bigcap \mathbf{V}$, but $L(u) > L(v)$. Let a_u and a_v have the same meaning as before. Let w_u be the ancestor of u at level $L(v)$, and let w_{au} be the ancestor of a_u at level $L(v)$. Then v, a_v, w_u and w_{au} share the same set of colored vertices—so vertices in $\chi_{m_i}(w_{au})$ have less than K_i different colors. However w_{au} is an ancestor of a_u and $a_u \in A_i$. That's a contradiction. So we've proved the second part of Lemma 3.

□

Lemma 4 *Let u and v be two vertices in $A_i \bigcup B_i$. ($u \neq v$, $\mathbf{J} \leq i \leq \mathbf{p}$.) Then u is neither an ancestor nor a descendant of v in the tree $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$. If there is an uncolored vertex in $\hat{\chi}_{m_i}(u) \bigcap \hat{\chi}_{m_i}(v) \bigcap \mathbf{V}$, then $L(u) = L(v)$.*

Proof: The proof for Lemma 4 is similar to the proof of Lemma 3. So for simplicity we omit it. Lemma 4 generalizes both Lemma 2 and Lemma 3.

□

Now let's define a relation called '*i-dependent*' on vertices in $A_i \bigcup B_i$, for $\mathbf{J} \leq i \leq \mathbf{p}$, as follows—for any two vertices $u \in A_i \bigcup B_i$ and $v \in A_i \bigcup B_i$, we say *u and v are i-dependent* if and only if there is an uncolored vertex in $\hat{\chi}_{m_i}(u) \bigcap \hat{\chi}_{m_i}(v) \bigcap \mathbf{V}$.

Lemma 5 *The relation* 'i-dependent' *defined on vertices in $A_i \bigcup B_i$ is an equivalence relation. ($\mathbf{J} \leq i \leq \mathbf{p}$.)*

Proof: Clearly the relation '*i-dependent*' is reflexive and symmetric. To prove that this relation is also transitive, use Lemma 4 and the fact that 'for any three vertices u, v and w in a tree, either the common ancestor of u and v or the common ancestor of v and w is the common ancestor of u, v and w'.

□

Let's use the equivalence relation '*i-dependent*' to divide the set $A_i \bigcup B_i$ into subsets $G_{i,1}$, $G_{i,2}$, $G_{i,3}$, $\cdots\cdots$ in this way: any two vertices of $A_i \bigcup B_i$ are in the same subset if and only if those two vertices are '*i-dependent*'. ($\mathbf{J} \leq i \leq \mathbf{p}$).

Example 8: Let's follow Example 7. Then $\mathbf{G_{1,1}} = \{v_{47}, v_{48}, v_{49}, v_{50}, v_{153}, v_{154}, v_{155}, v_{156}, v_{157}\}$, $\mathbf{G_{2,1}} = \{v_{28}, v_{29}\}$, $\mathbf{G_{2,2}} = \{v_{30}\}$, $\mathbf{G_{3,1}} = \{v_{31}\}$, $\mathbf{G_{3,2}} = \{v_{32}\}$, $\mathbf{G_{3,3}} = \{v_{30}\}$.

□

For $\mathbf{J} \leq i \leq \mathbf{p}$ and $j = 1, 2, 3, \cdots\cdots$, we define $r^0_{i,j}$ to be such a vertex: $r^0_{i,j}$ is a common quasi-ancestor of all the vertices in $G_{i,j}$ (in the tree $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$); and for any common quasi-ancestor of the vertices in $G_{i,j}$, its level number is no greater than $L(r^0_{i,j})$. And we define $V_{i,j}$ to be such a set:

$V_{i,j} = \{v | v \in \mathbf{V}; v$ is a quasi-descendant of $r^0_{i,j}\} \bigcup \{v | v \in \mathbf{V}; d(v, r^0_{i,j}) \leq m_i - (L(G_{i,j}) - L(r^0_{i,j}))\}$.

We define $T_{i,j}(V_{i,j}, E_{i,j})$ to be the subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ induced by the vertex set $V_{i,j}$.

Lemma 6 *For any uncolored vertex $v \in V_{i,j}$, v is either a quasi-descendant or an ancestor of $r^0_{i,j}$.* ($\mathbf{J} \leq i \leq \mathbf{p}$, $j = 1, 2, 3, \cdots$.)

Proof: We'll prove Lemma 6 with contradiction. Suppose that there is an uncolored vertex $v \in V_{i,j}$, but v is neither a quasi-descendant nor an ancestor of $r^0_{i,j}$. Then by the definition of $V_{i,j}$, we know $d(v, r^0_{i,j}) \leq m_i - (L(G_{i,j}) - L(r^0_{i,j}))$. So v is within m_i hops from any vertex in $G_{i,j}$.

Let $\hat{v} \in \hat{\mathbf{V}}$ be such a vertex: $\hat{v}$ is either an ancestor or a quasi-descendant of v, and $L(\hat{v}) = L(G_{i,j})$. Clearly $d(\hat{v}, v)$ is less than the distance between v and any vertex in $G_{i,j}$. So $d(\hat{v}, v) < m_i$. So if we use u to denote any vertex in $G_{i,j}$, then v is an uncolored vertex in $\hat{\chi}_{m_i}(u) \bigcap \hat{\chi}_{m_i}(\hat{v}) \bigcap \mathbf{V}$. So $\hat{v}$ is *i-dependent* to vertices in $G_{i,j}$. Therefore $\hat{v} \in G_{i,j}$, and $r^0_{i,j}$ is an ancestor of $\hat{v}$. As $\hat{v}$ is either an ancestor or a quasi-descendant of v, v is either a quasi-descendant or an ancestor of $r^0_{i,j}$. And that's a contradiction. So we've proved Lemma 6.

□

For $\mathbf{J} \leq i \leq \mathbf{p}$ and $j = 1, 2, 3, \cdots\cdots$, we define $r^1_{i,j}$ to be such a vertex: if there are x ($x \geq 0$) uncolored vertices in $V_{i,j}$ that are not quasi-descendants of $r^0_{i,j}$, then $r^1_{i,j}$ is the quasi-ancestor of $r^0_{i,j}$ at level $L(r^0_{i,j}) - x$.

Let's use $u^b_{i,j}$ to denote such a vertex: $u^b_{i,j}$ is an uncolored quasi-descendant of $r^1_{i,j}$ in $\mathbf{V}$; for any uncolored quasi-descendant of $r^1_{i,j}$ in $\mathbf{V}$, its level number is no less than $L(u^b_{i,j})$.

Lemma 7 *$u^b_{i,j}$ exists. And $u^b_{i,j} \in V_{i,j}$.* ($\mathbf{J} \leq i \leq \mathbf{p}$, $j = 1, 2, 3, \cdots$.)

Proof: At least one vertex in $G_{i,j}$ is also in $\mathbf{V}$. Let's denote that vertex by v. Vertices in $\chi_{m_i}(v)$ have less than K_i different colors, and no two vertices in $\chi_{m_i}(v)$ have the same color. $|\chi_{m_i}(v)| \geq K_i$, so there is at least one uncolored vertex in $\chi_{m_i}(v)$. From the definition of $V_{i,j}$, it's easy to see that $\chi_{m_i}(v) \subseteq V_{i,j}$. So there is at least one uncolored vertex in $V_{i,j}$. By Lemma 6 and the fact that subtree $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ was colored 'level by level', clearly any uncolored vertex in $V_{i,j}$ is a quasi-descendant of $r^1_{i,j}$. So $r^1_{i,j}$ has at least one uncolored quasi-descendant in $\mathbf{V}$. So $u^b_{i,j}$ exists.

Let's say there are x ($x \geq 0$) uncolored vertices in $V_{i,j}$ that are not quasi-descendants of $r^0_{i,j}$. If $x = 0$, then clearly $r^1_{i,j} = r^0_{i,j}$, and $u^b_{i,j} \in V_{i,j}$. If $x > 0$, then $u^b_{i,j} = r^1_{i,j}$, so again $u^b_{i,j} \in V_{i,j}$.

□

Now define $r_{i,j}$ to be such a vertex: $r_{i,j}$ is a quasi-ancestor of $r^1_{i,j}$, $(L(G_{i,j}) - L(r_{i,j})) + (L(u^b_{i,j}) - L(r_{i,j})) \leq m_i$; if $r_{i,j}$ is not the root of $\mathbf{G}(\mathbf{V}, \mathbf{E})$, then $(L(G_{i,j}) - L(r_{i,j})) + (L(u^b_{i,j}) - L(r_{i,j})) + 2 > m_i$. ($\mathbf{J} \leq i \leq \mathbf{p}$, $j = 1, 2, 3, \cdots$.)

Lemma 8 *$V_{i,j} = \{v | v \in \mathbf{V}; v$ is a quasi-descendant of $r_{i,j}\} \bigcup \{v | v \in \mathbf{V}; d(v, r_{i,j}) \leq m_i - (L(G_{i,j}) - L(r_{i,j}))\}$. And $r_{i,j}$ is a quasi-descendant of $\overline{\mathbf{r_J}}$.* ($\mathbf{J} \leq i \leq \mathbf{p}$, $j = 1, 2, 3, \cdots$.)

Proof: First we claim that $r^0_{i,j}$ is the only quasi-descendant of $r_{i,j}$ at level $L(r^0_{i,j})$ in the tree $\hat{\mathbf{G}}(\hat{\mathbf{V}}, \hat{\mathbf{E}})$ —because if not, then with arguemnts similar to the proof of Lemma 6, we can prove that there exists a vertex in $\hat{\mathbf{V}}$ which is not a quasi-descendant of $r^0_{i,j}$ but is *i-dependent* to vertices in $G_{i,j}$, which cannot be true. Then by checking the definition of $V_{i,j}$, we see that the first part of Lemma 8 is naturally true.

Now let's prove that $r_{i,j}$ is a quasi-descendant of $\overline{\mathbf{r_J}}$ with contradiction. Suppose $r_{i,j}$ is not a quasi-descendant of $\overline{\mathbf{r_J}}$. Since both $\overline{\mathbf{r_J}}$ and $r_{i,j}$ are common quasi-ancestors of vertices in $G_{i,j}$, $r_{i,j}$ is an ancestor of $\overline{\mathbf{r_J}}$. Now let's first consider the case where $i = \mathbf{J}$. By the definition of $G_{\mathbf{J},j}$, clearly $L(G_{\mathbf{J},j}) - L(\overline{\mathbf{r_J}}) \geq \overline{\mathbf{a_J}}$. Since $u^b_{\mathbf{J},j}$ is an uncolored vertex in $\overline{\mathbf{V_J}}$, $L(u^b_{\mathbf{J},j}) - L(\overline{\mathbf{r_J}}) \geq \overline{\mathbf{b_J}}$. From the definition of $r_{\mathbf{J},j}$, we see that $\overline{\mathbf{a_J}} + \overline{\mathbf{b_J}} + 2 \leq L(G_{\mathbf{J},j}) - L(\overline{\mathbf{r_J}}) + L(u^b_{\mathbf{J},j}) - L(\overline{\mathbf{r_J}}) + 2 \leq L(G_{\mathbf{J},j}) - L(r_{\mathbf{J},j}) + L(u^b_{\mathbf{J},j}) - L(r_{\mathbf{J},j}) \leq \mathbf{m_J}$. However, that's a contradiction to the fact that if $\overline{\mathbf{r_J}}$ is not the root of $\mathbf{G}(\mathbf{V}, \mathbf{E})$, then $\overline{\mathbf{a_J}} + \overline{\mathbf{b_J}} + 2 > \mathbf{m_J}$ (which is stated at the beginning part of this section). For the case

where $\mathbf{J} < i \leq \mathbf{p}$, we can similarly show there is a contradiction. So we've proved the last part of Lemma 8.

□

We let $a_{i,j} = L(G_{i,j}) - L(r_{i,j})$, and let $b_{i,j} = L(u^b_{i,j}) - L(r_{i,j})$. ($\mathbf{J} \leq i \leq \mathbf{p}$, $j = 1, 2, 3, \cdots$.) Then we have the following lemma:

Lemma 9 *$T_{i,j}(V_{i,j}, E_{i,j})$ is a normal subtree of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ with parameter set $\{r_{i,j}, a_{i,j}, b_{i,j}, K_i, m_i\}$. ($\mathbf{J} \leq i \leq \mathbf{p}$, $j = 1, 2, 3, \cdots$.)*

Proof: To prove Lemma 9, we only need to prove that $T_{i,j}(V_{i,j}, E_{i,j})$ satisfies all the 5 properties listed in the definition of '*normal subtree*' (in Section II). Property 1 to property 3 are trivially true. So we only prove property 4 and property 5 here.

First we give the sketch for proving property 4. $r_{i,j}$ is a quasi-descendant of $\overline{\mathbf{r_J}}$ by Lemma 8. Then it's easy to see $V_{i,j} \subseteq \overline{\mathbf{V_J}}$ by the definition of $V_{i,j}$. All colored vertices in $T_{i,j}(V_{i,j}, E_{i,j})$ are within m_i hops from any vertex in $G_{i,j}$. So $T_{i,j}(V_{i,j}, E_{i,j})$ have less than K_i colors, all of which are different. The rest of property 4 follows the fact that $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ was colored 'level by level'.

For property 5, we'll prove it with contradiction. Suppose there is a vertex v such that $\chi_{m_i}(v) \subseteq V_{i,j}$, but the condition '$L(v) - L(r_{i,j}) = d(v, r_{i,j}) \geq a_{i,j}$' is false. There are less than K_i colored vertices in $V_{i,j}$, so vertices in $\chi_{m_i}(v)$ have less than K_i different colors. From Lemma 4 and the definition of $G_{i,j}$, it's easy to see that v cannot be the ancestor of any vertex in $G_{i,j}$. So v is neither a quasi-descendant nor an ancestor of $r_{i,j}$. There is at least one uncolored vertex within m_i hops from v, and any such uncolored vertex must be a quasi-descendant of $r_{i,j}$. From the definition of $u^b_{i,j}$, it's clear that $u^b_{i,j}$ is within m_i hops from v. $u^b_{i,j}$ is also within m_i hops from any vertex in $G_{i,j}$. So from Lemma 2 it's easy to see that not only $L(v) \geq L(G_{i,j})$, but the quasi-ancestor of v at level $L(G_{i,j})$ is *i-dependent* to vertices in $G_{i,j}$. But then $r_{i,j}$ must be a quasi-ancestor of v, and that's a contradiction. So we've proved property 5.

So Lemma 9 is proved.

□

Lemma 10 *When $j_1 \neq j_2$, all vertices in $V_{i,j_1} \bigcap V_{i,j_2}$ are colored. ($\mathbf{J} \leq i \leq \mathbf{p}$.)*

Proof: We'll prove Lemma 10 with contradiction. Suppose $j_1 \neq j_2$, and there is an uncolored vertex w_0 in $V_{i,j_1} \bigcap V_{i,j_2}$.

w_0 is uncolored, so w_0 is a quasi-descendant of both r_{i,j_1} and r_{i,j_2}. So without loss of generality, we can say that r_{i,j_1} is a quasi-ancestor of r_{i,j_2}. Let w_2 be a vertex in V_{i,j_2} at level $L(G_{i,j_2})$. Clearly w_2 is a quasi-descendant of r_{i,j_1}. Let w_1 be such a vertex: w_1 is either a quasi-ancestor or a descendant of w_2, $w_1 \in G_{i,j_1}$. We know $w_1 \in A_i \bigcup B_i$ and $w_2 \in A_i$. So by Lemma 4, w_1 and w_2 are the same vertex. $w_1 \in G_{i,j_1}$ and $w_2 \in G_{i,j_2}$, so G_{i,j_1} and G_{i,j_2} have a vertex in common. But '*i-dependent*' is an equivalent relation, so G_{i,j_1} and G_{i,j_2} cannot have any vertex in common. Therefore there is a contradiction. So we've proved Lemma 10.

□

Lemma 11 *When $\mathbf{J} \leq i_1 < i_2 \leq \mathbf{p}$, exactly one of the following two cases is true:*

(a) All vertices in $V_{i_1,j_1} \bigcap V_{i_2,j_2}$ are colored.

(b) $V_{i_2,j_2} \subseteq V_{i_1,j_1}$, and $L(r_{i_2,j_2}) - L(r_{i_1,j_1}) = d(r_{i_2,j_2}, r_{i_1,j_1})$.

($j_1 = 1, 2, 3, \cdots$, $j_2 = 1, 2, 3, \cdots$.)

Proof: We've proved before that there exists at least one uncolored vertex in V_{i_2,j_2}, so case (a) and case (b) cannot both be true. So we only need to prove that when there is an uncolored vertex in $V_{i_1,j_1} \bigcap V_{i_2,j_2}$, case (b) is true.

Let's suppose that there is an uncolored vertex w_0 in $V_{i_1,j_1} \bigcap V_{i_2,j_2}$. Then w_0 is a quasi-descendant of both r_{i_1,j_1} and r_{i_2,j_2}. So r_{i_2,j_2} is either an ancestor or a quasi-descendant of r_{i_1,j_1}. So we consider the following two cases:

1. In this case, r_{i_2,j_2} is an ancestor of r_{i_1,j_1}.

   First let's prove that $L(G_{i_1,j_1}) \leq L(G_{i_2,j_2}) + (m_{i_1} - m_{i_2})$ with contradiction. Suppose $L(G_{i_1,j_1}) > L(G_{i_2,j_2}) + (m_{i_1} - m_{i_2})$. Let w_1 be a vertex in V_{i_1,j_1} at level $L(G_{i_1,j_1})$, let w_2 be the ancestor of w_1 at level $L(G_{i_2,j_2})$, and let w_3 be the ancestor of w_1 at level $L(G_{i_2,j_2}) + (m_{i_1} - m_{i_2})$. It's easy to see that $w_2 \in G_{i_2,j_2}$, and a colored vertex is within m_{i_2} hops from w_2 if and only if that colored vertex is within m_{i_1} hops from w_3; what's more, $\chi_{m_{i_1}}(w_3) \subseteq \overline{\mathbf{V_J}}$. So vertices in $\chi_{m_{i_1}}(w_3)$ have less than K_{i_2} different colors. (And note that $K_{i_2} < K_{i_1}$.) However $w_1 \in A_{i_1}$, and from the definition of A_{i_1}, we know that w_3 cannot be the ancestor of w_1. So there is a contradiction. So we've proved that $L(G_{i_1,j_1}) \leq L(G_{i_2,j_2}) + (m_{i_1} - m_{i_2})$.

Let w_1 be a vertex in V_{i_1,j_1} at level $L(G_{i_1,j_1})$. Then $d(w_1, u^b_{i_2,j_2}) \leq d(w_1, r_{i_2,j_2}) + d(r_{i_2,j_2}, u^b_{i_2,j_2}) \leq (a_{i_2,j_2} + m_{i_1} - m_{i_2}) + b_{i_2,j_2} \leq m_{i_1}$. So $u^b_{i_2,j_2} \in V_{i_1,j_1}$. So $L(u^b_{i_1,j_1}) = L(u^b_{i_2,j_2})$. So $a_{i_1,j_1} + b_{i_1,j_1} + 2 \leq d(w_1, r_{i_2,j_2}) + d(r_{i_2,j_2}, u^b_{i_1,j_1}) = d(w_1, r_{i_2,j_2}) + d(r_{i_2,j_2}, u^b_{i_2,j_2}) \leq m_{i_1}$. But that contradicts the definition of r_{i_1,j_1}. So this case cannot be true.

2. In this case, r_{i_2,j_2} is a quasi-descendant of r_{i_1,j_1}. Then $L(r_{i_2,j_2}) - L(r_{i_1,j_1}) = d(r_{i_2,j_2}, r_{i_1,j_1})$. With arguments similar to those in the previous case, we can prove that in this case, we also have $L(G_{i_1,j_1}) \leq L(G_{i_2,j_2}) + (m_{i_1} - m_{i_2})$.

   Let $w_1 \in V_{i_2,j_2}$ be a vertex that is not a quasi-descendant of r_{i_1,j_1}. Then w_1 is not a quasi-descendant of r_{i_2,j_2}. So $d(r_{i_2,j_2}, r_{i_1,j_1}) + d(r_{i_1,j_1}, w_1) = d(w_1, r_{i_2,j_2}) \leq m_{i_2} - L(G_{i_2,j_2}) + L(r_{i_2,j_2})$. So $d(r_{i_1,j_1}, w_1) \leq m_{i_2} - L(G_{i_2,j_2}) + L(r_{i_2,j_2}) - d(r_{i_2,j_2}, r_{i_1,j_1}) = m_{i_2} - L(G_{i_2,j_2}) + L(r_{i_1,j_1}) \leq m_{i_1} - L(G_{i_1,j_1}) + L(r_{i_1,j_1})$. So $w_1 \in V_{i_1,j_1}$. Now it's easy to see that $V_{i_2,j_2} \subseteq V_{i_1,j_1}$.

So we've proved Lemma 11.
□

Let's now take a close look at those normal subtrees $T_{\mathbf{J},1}(V_{\mathbf{J},1}, E_{\mathbf{J},1})$, $T_{\mathbf{J},2}(V_{\mathbf{J},2}, E_{\mathbf{J},2})$, $\cdots$, $T_{\mathbf{J}+1,1}(V_{\mathbf{J}+1,1}, E_{\mathbf{J}+1,1})$, $T_{\mathbf{J}+1,2}(V_{\mathbf{J}+1,2}, E_{\mathbf{J}+1,2})$, $\cdots$, $\cdots\cdots$, $T_{\mathbf{p},1}(V_{\mathbf{p},1}, E_{\mathbf{p},1})$, $T_{\mathbf{p},2}(V_{\mathbf{p},2}, E_{\mathbf{p},2})$, $\cdots$. They have the following properties:

- For any vertex v and for $\mathbf{J} \leq i \leq \mathbf{p}$, if $\chi_{m_i}(v) \subseteq \overline{\mathbf{V_J}}$ and vertices in $\chi_{m_i}(v)$ have less than K_i different colors, then there is exactly one normal subtree $T_{i,*}(V_{i,*}, E_{i,*})$ that contains all the vertices in $\chi_{m_i}(v)$. (Here '$*$' means an unspecified parameter.)

- For any two normal subtrees $T_{i_1,j_1}(V_{i_1,j_1}, E_{i_1,j_1})$ and $T_{i_2,j_2}(V_{i_2,j_2}, E_{i_2,j_2})$, either they don't have any uncolored vertex in common, or one is totally contained in the other. (If $V_{i_1,j_1} \subset V_{i_2,j_2}$, then $i_1 > i_2$.)

- Consider the properties of any normal subtree $T_{i,j}(V_{i,j}, E_{i,j})$ and the properties of those normal subtrees contained in it, and consider the way they are contained in one another. Then compare them with the properties of the subtrees $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$, $\overline{\mathbf{T_{J+1,1}}}(\overline{\mathbf{V_{J+1,1}}}, \overline{\mathbf{E_{J+1,1}}})$, $\overline{\mathbf{T_{J+1,2}}}(\overline{\mathbf{V_{J+1,2}}}, \overline{\mathbf{E_{J+1,2}}}), \cdots$, $\overline{\mathbf{T_{J+2,1}}}(\overline{\mathbf{V_{J+2,1}}}, \overline{\mathbf{E_{J+2,1}}})$, $\overline{\mathbf{T_{J+2,2}}}(\overline{\mathbf{V_{J+2,2}}}, \overline{\mathbf{E_{J+2,2}}}), \cdots, \cdots\cdots$, $\overline{\mathbf{T_{p,1}}}(\overline{\mathbf{V_{p,1}}}, \overline{\mathbf{E_{p,1}}})$, $\overline{\mathbf{T_{p,2}}}(\overline{\mathbf{V_{p,2}}}, \overline{\mathbf{E_{p,2}}})$, $\cdots$ and the way they were contained in one another, as described at the beginning of this section. We find that they satisfy the same properties, and the ways they are contained in one another are also the same.

Let's introduce the concept of an '*extremal subtree*' in the following way: for a subtree in the set $\{T_{i,j}(V_{i,j}, E_{i,j}) | \mathbf{J} \leq i \leq \mathbf{p}; j = 1, 2, 3, \cdots\}$, if it is not contained in another subtree in the set $\{T_{i,j}(V_{i,j}, E_{i,j}) | \mathbf{J} \leq i \leq \mathbf{p}; j = 1, 2, 3, \cdots\}$, then we call that subtree an *extremal subtree* (if two subtrees $T_{i_1,j_1}(V_{i_1,j_1}, E_{i_1,j_1})$ and $T_{i_2,j_2}(V_{i_2,j_2}, E_{i_2,j_2})$ are induced by the same set of vertices—which means they contain each other,—then we say $T_{i_1,j_1}(V_{i_1,j_1}, E_{i_1,j_1})$ is an *extremal subtree* but $T_{i_2,j_2}(V_{i_2,j_2}, E_{i_2,j_2})$ is not an *extremal subtree* if $i_1 < i_2$).

Example 9: Let's follow Example 8. Then $\mathbf{G_{1,1}}$, $\mathbf{G_{2,1}}$, $\mathbf{G_{2,2}}$, $\mathbf{G_{3,1}}$, $\mathbf{G_{3,2}}$ and $\mathbf{G_{3,3}}$ correspond to 6 normal subtrees— $T_{1,1}(V_{1,1}, E_{1,1})$, $T_{2,1}(V_{2,1}, E_{2,1})$, $T_{2,2}(V_{2,2}, E_{2,2})$, $T_{3,1}(V_{3,1}, E_{3,1})$, $T_{3,2}(V_{3,2}, E_{3,2})$ and $T_{3,3}(V_{3,3}, E_{3,3})$, —whose parameter sets are $(v_{25}, 5, 1, 14, 7)$, $(v_{25}, 1, 1, 7, 3)$, $(v_{26}, 1, 1, 7, 3)$, $(v_{28}, 1, 1, 3, 2)$, $(v_{29}, 1, 0, 3, 2)$, $(v_{26}, 1, 1, 3, 2)$ respectively. (Those 6 normal subtrees are circled and shown in Fig. 7 (b).) Readers can easily verify that they satisfy all the properties proved so far. From Fig. 7 (b), it's easy to see that $T_{1,1}(V_{1,1}, E_{1,1})$ and $T_{2,2}(V_{2,2}, E_{2,2})$ are the 2 extremal subtrees among those 6 subtrees.
□

Based on the above analysis, we find a method to do the layered diversity coloring on a tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$:

- At the beginning, let $T_{1,1}(V_{1,1}, E_{1,1}) = T_{2,1}(V_{2,1}, E_{2,1}) = \cdots = T_{\mathbf{p},1}(V_{\mathbf{p},1}, E_{\mathbf{p},1}) = \mathbf{G}(\mathbf{V}, \mathbf{E})$. (At this moment, all vertices in $\mathbf{G}(\mathbf{V}, \mathbf{E})$ are uncolored.) Here $T_{1,1}(V_{1,1}, E_{1,1})$ is the extremal subtree. Color the extremal subtree $T_{1,1}(V_{1,1}, E_{1,1})$ in the same way we colored $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ (described at the beginning of this section). After the coloring, $T_{1,1}(V_{1,1}, E_{1,1})$, $T_{2,1}(V_{2,1}, E_{2,1})$, $\cdots$, $T_{\mathbf{p},1}(V_{\mathbf{p},1}, E_{\mathbf{p},1})$ are replaced by some new normal subtrees, some of which are new extremal subtrees. Since no two extremal subtrees have any uncolored vertex

in common, we can color each extremal subtree independently. For each extremal subtree, we color it in the same way we colored $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$—and after the coloring, that extremal subtree will be replaced by some new extremal subtrees, each of which we will again color independently in the same way we colored $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ $\cdots\cdots$ This process keeps going on until there is no extremal subtree left, which means for any vertex v and for $\mathbf{J} \leq i \leq \mathbf{p}$, vertex v can find at least K_i different colors within m_i hops. Then we just need to color those remaining uncolored vertices arbitrarily, and we get a layered diversity coloring.

Actually that is the way the algorithm presented in Section III colors trees. The only thing to note is that in the algorithm, when there exist two extremal subtrees $T_{i_1,j_1}(V_{i_1,j_1}, E_{i_1,j_1})$ and $T_{i_2,j_2}(V_{i_2,j_2}, E_{i_2,j_2})$, if $i_1 < i_2$, then the algorithm always colors the subtree $T_{i_1,j_1}(V_{i_1,j_1}, E_{i_1,j_1})$ first. We will prove the correctness of the coloring algorithm in Section V.

V Proof of the Algorithm and Complexity Analysis

In this section, we'll prove the correctness of the coloring algorithm presented in Section III. Then we'll analyse its time complexity.

Proof of the Algorithm

Theorem 1 *The layered diversity coloring algorithm for trees presented in Section III is correct.*

Proof: The algorithm's many details have been shown and proved in the previous sections, so here we give just the sketch of the proof.

The algorithm has $\mathbf{p}$ iterations. In the 1-st iteration, the algorithm searches for normal subtrees of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ of parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$ satisfying the properties those corresponding normal subtrees described in Section IV satisfy. (The first such normal subtree the algorithm finds is $\mathbf{G}(\mathbf{V}, \mathbf{E})$ itself.) Every vertex v added to the set R corresponds to a normal subtree of parameter set $(v, p_1(v), p_2(v), \mathbf{K_1}, \mathbf{m_1})$, ($S$, $p_1(v)$ and $p_2(v)$ are all notations used in the algorithm), and vice versa. (To see why, note that the algorithm checks vertices to see if they are eligible to be added to R in a distributed breadth-first way, beginning with the root of $\mathbf{G}(\mathbf{V}, \mathbf{E})$. Then Lemma 8 and the definition of '$r_{i,j}$' in Section IV lead us to the conclusion.) No two normal subtrees of parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$ have any uncolored vertex in common, so they can be colored independently. Each normal subtree of parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$ is an 'extremal subtree' because no other normal subtree contains it, and the algorithm will color it in the same way we colored $\overline{\mathbf{T_J}}(\overline{\mathbf{V_J}}, \overline{\mathbf{E_J}})$ (see command line 20–24 in the algorithm)—and after the coloring, that subtree will disappear and some new normal subtrees of parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$ contained in it will appear, which will be found by the algorithm. The algorithm keeps coloring in this way until there is no normal subtree of parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$ left, which means every vertex of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ can find at least $\mathbf{K_1}$ different colors within $\mathbf{m_1}$ hops, and the 1-st iteration ends.

It's simple to use induction to analyse every iteration as above and prove that after the i-th iteration ($1 \leq i \leq \mathbf{p}$), every vertex of $\mathbf{G}(\mathbf{V}, \mathbf{E})$ can find at least K_i different colors within m_i hops. So when the algorithm ends, we get a layered diversity coloring.

Now it's only left to show that the algorithm always terminates in finite time. To see that, note that every normal subtree contains at least one uncolored vertex, and the algorithms uses at least $\mathbf{N} - (\mathbf{K_1} - 1) > 0$ colors to color each normal subtree. So every time the algorithm colors a normal subtree, at least one vertex is colored. The tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$ has only finite vertices, so the algorithm can do only finite times of coloring and then terminates.

□

Now we can prove the following important theorem.

Theorem 2 *Given the tree* $\mathbf{G}(\mathbf{V}, \mathbf{E})$ *and parameters* $\mathbf{p}$, $\mathbf{N}$, $\mathbf{K_1}$, $\mathbf{K_2}$, $\cdots$, $\mathbf{K_p}$ *and* $\mathbf{m_1}$, $\mathbf{m_2}$, $\cdots$, $\mathbf{m_p}$, *(here all those parameters are integers, and* $\mathbf{N} \geq \mathbf{K_1} > \mathbf{K_2} > \cdots > \mathbf{K_p}$, $\mathbf{m_1} > \mathbf{m_2} > \cdots > \mathbf{m_p}$,*) there exists a layered diversity coloring on* $\mathbf{G}(\mathbf{V}, \mathbf{E})$ *if and only if for any vertex* $v \in \mathbf{V}$ *and* $1 \leq i \leq \mathbf{p}$, *the* neighborhood of v of radius m_i *has no less than* K_i *vertices (that is,* $|\chi_{m_i}(v)| \geq K_i$*).*

Proof: '$|\chi_{m_i}(v)| \geq K_i$ for any vertex $v \in \mathbf{V}$ and for any $1 \leq i \leq \mathbf{p}$' is a neccessary condition for there to exist a layered diversity coloring, because in a layered diversity coloring, each vertex needs to

find no less than K_i different colors within m_i hops, while each vertex is assigned exactly one color. On the other hand, when '$|\chi_{m_i}(v)| \geq K_i$ for any vertex $v \in \mathbf{V}$ and for any $1 \leq i \leq \mathbf{p}$', all the prerequiste conditions are satisfied in the coloring algorithm presented in Section III, and the algorithm can output a layered diversity coloring on the tree. So it is also sufficient. Therefore we've proved Theorem 2.

□

Example 10: Example 2 shows how the coloring algorithm colors a tree. Here we use the same example, and show how the coloring process corresponds to coloring extremal subtrees.

At step 2 in Example 2, $R = \{v_1\}$. At that moment, there are 3 normal subtrees (let's call them T_1, T_2 and T_3) with parameter set $(v_1, 0, 0, 14, 7)$, $(v_1, 0, 0, 7, 3)$ and $(v_1, 0, 0, 3, 2)$ respectively. The vertex $v_1 \in R$ corresponds to T_1, the only normal subtree (extremal subtree) with parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$. (And clearly $T_1 = T_2 = T_3 = \mathbf{G}(\mathbf{V}, \mathbf{E})$.)

The algorithm then colors T_1 (see step 3 of Example 2). At step 4 of Example 2, $R = \{v_4\}$. There are 7 normal subtrees at that moment (let's call them T_4, T_5, $\cdots$, T_{10}) with parameter set $(v_4, 5, 1, 14, 7)$, $(v_7, 1, 1, 7, 3)$, $(v_{13}, 2, 1, 7, 3)$, $(v_{16}, 2, 1, 7, 3)$, $(v_7, 1, 1, 3, 2)$, $(v_{13}, 1, 1, 3, 2)$ and $(v_{16}, 1, 1, 3, 2)$ respectively. The vertex $v_4 \in R$ is the only normal subtree (extremal subtree) with parameter set $(*, *, *, \mathbf{K_1}, \mathbf{m_1})$. (In fact, 5 of those 7 subtrees are circled and shown in Fig. 7 (a), where T_4, T_6, T_7, T_9 and T_{10} are the same as subtrees $\overline{T_1}(\overline{V_1}, \overline{E_1})$, $\overline{T_{2,1}}(\overline{V_{2,1}}, \overline{E_{2,1}})$, $\overline{T_{2,2}}(\overline{V_{2,2}}, \overline{E_{2,2}})$, $\overline{T_{3,1}}(\overline{V_{3,1}}, \overline{E_{3,1}})$ and $\overline{T_{3,2}}(\overline{V_{3,2}}, \overline{E_{3,2}})$ respectively. That's not a surprise.)

The algorithm then colors T_4 (see step 5 of Example 2). We can analyse the following steps similarly as above.

□

Analysis of Time Complexity

The coloring algorithm has $\mathbf{p}$ iterations, and each iteration has two main parts of work—coloring extremal subtrees (command line 20—24), and checking vertices to see if they are knots of extremal subtrees (command line 5—18 and 26—39).

- Time complexity of *Coloring Extremal Subtrees*:

  In the *i-th* iteration, when an extremal subtree is being colored, the algorithm needs to find out the colors within m_i hops from a certain vertex v_0. We assume that each vertex has a list of the vertices within m_i hops from itself, (that data structure is stronger than the adjacency matrix, but is still just another way to describe the structure of the tree, so we think this assumption is reasonable,) and those vertices are listed in this order: from small level to large level, from left to right. In this analysis we always assume vertices at the same level in an extremal subtree are colored from left to right. Then it's not hard to see that the (less than K_i) colored vertices within m_i hops from v_0 are all in the front of v_0's list, all with distinct colors—so finding those colors has complexity $O(\mathbf{K_1})$. In the $\mathbf{p}$ iterations, at most $|\mathbf{V}|$ extremal subtrees are colored. So the total time complexity for finding colors within m_i hops ($1 \leq i \leq \mathbf{p}$) from certain vertices is $O(\mathbf{K_1}|\mathbf{V}|)$. Coloring vertices has time complexity $\Theta(|\mathbf{V}|)$. So the time complexity of 'Coloring Extremal Subtrees' is $O(\mathbf{K_1}|\mathbf{V}| + |\mathbf{V}|) = O(\mathbf{K_1}|\mathbf{V}|)$.

- Time complexity of *Checking Vertices to See if They Are Knots of Extremal Subtrees*:

  A vertex v becomes able to find K_i different colors within m_i hops at the moment the K_i-th vertex in its list (the list listing the vertices within m_i hops from v) is colored—and the algorithm can be notified then. Such notification work has time complexity $\Theta(\mathbf{p}|\mathbf{V}|)$ during the whole $\mathbf{p}$ iterations. To inspect a single vertex once to see if it's the knot of some extremal subtree in the *i-th* iteration has time complexity $O(m_i)$, because the work there is to search for two special vertices denoted by v_1 and v_2, either of which has at most $m_i + 1$ candidates. (To search for v_2, we only need to check the right-most vertex in the extremal subtree at each level.)

  Let $x_{i,v}$ denote the number of times vertex v is the knot of some extremal subtree in the *i-th* iteration. Then v will be inspected $x_{i,v} + 1$ times in the *i-th* iteration to see if it's the knot ($x_{i,v}$ times the answer will be yes and once the answer will be no). So in the whole $\mathbf{p}$ iterations there will be $\sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} (x_{i,v} + 1)$ times of inspecting vertices. $\sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} x_{i,v}$ is the number of times the algorithm colors

extremal subtrees, so $\sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} x_{i,v} \leq |\mathbf{V}|$. So 'checking vertices to see if they are knots of extremal subtrees' has a total time complexity of $O(\mathbf{p}|\mathbf{V}| + \sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} (x_{i,v} + 1) m_i) = O(\mathbf{p}|\mathbf{V}| + \sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} x_{i,v} m_i + \sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} m_i) = O(\mathbf{p}|\mathbf{V}| + m_1 \sum_{i=1}^{\mathbf{p}} \sum_{v \in \mathbf{V}} x_{i,v} + |\mathbf{V}| \sum_{i=1}^{\mathbf{p}} m_i) = O(\mathbf{p}|\mathbf{V}| + m_1|\mathbf{V}| + |\mathbf{V}| \sum_{i=1}^{\mathbf{p}} m_i) = O(|\mathbf{V}| \sum_{i=1}^{\mathbf{p}} m_i)$.

The complexity of all the other miscellaneous operations in the algorithm is negligible. So by combining the results we get above, we find that the coloring algorithm presented in Section III has time complexity $O(|\mathbf{V}|(\mathbf{K_1} + \sum_{i=1}^{\mathbf{p}} m_i))$.

VI Improvements for the Coloring Algorithm

In this section, we'll present two propositions which can be used to improve the efficiency of the coloring algorithm. The proof of Proposition 1 is omitted due to its simplicity.

Proposition 1 *In the coloring algorithm presented in Section III, if at some moment a vertex is found to be the knot of an extremal subtree, then from that moment on, no ancestor of that vertex will be the knot of any extremal subtree.*

Proposition 2 *For any vertex v in the tree $\mathbf{G}(\mathbf{V}, \mathbf{E})$, define $F_v = \{u | u \in \mathbf{V}, u$ is a leaf, and u is a quasi-descendant of $v.\}$. Then in the coloring algorithm presented in Section III, for any vertex v, if $L(v) \leq max_{u \in F_v} L(u) - m_i$, then v won't be the knot of any extremal subtree at any moment after the* i-th *iteration ($1 \leq i \leq \mathbf{p} - 1$).*

Proof: We'll prove this proposition with contradiction. Suppose that $L(v) \leq max_{u \in F_v} L(u) - m_i$, and v is the knot of an extremal subtree after the *i-th* iteration. Since after the *i-th* iteration no normal subtree with parameter set $(*, *, *, K_t, m_t)$ $(1 \leq t \leq i)$ is left, without loss of generality we can say v is the knot of a normal subtree with parameter set (v, a, b, K_c, m_c) where $i < c \leq \mathbf{p}$.

Let w_1 be a quasi-descendant of v at level $max_{u \in F_v} L(u)$, and let w_2 be the ancestor of w_1 at level $L(w_1) - (m_i - m_c)$. Since $L(v) \leq max_{u \in F_v} L(u) - m_i$, we get $L(w_1) \geq L(v) + m_i$, and $L(w_2) \geq L(v) + m_c$. So both w_1 and w_2 are descendants of v. Since $L(w_2) - L(v) \geq m_c \geq a$, w_2 can find less than K_c different colors within m_c hops. Since $L(w_1) - m_i = L(w_2) - m_c$, it's easy to see that w_1 can also only find less than K_c different colors within m_i hops. However, after the *i-th* iteration, every vertex can find at least K_i different colors within m_i hops, and $K_i > K_c$. So there is a contradiction. So we've proved Proposition 2.

□

Clearly the coloring algorithm can use the two propositions to help check if a vertex has the possibility to be the knot of an extremal subtree. And the checking using those propositions has very low time complexity. From the previous section, we've learned that 'checking vertices to see if they are knots of extremal subtrees' contributes time complexity $O(|\mathbf{V}| \sum_{i=1}^{\mathbf{p}} m_i)$, without which the algorithm's complexity would go from $O(|\mathbf{V}|(\mathbf{K_1} + \sum_{i=1}^{\mathbf{p}} m_i))$ to $O(\mathbf{K_1}|\mathbf{V}|)$. So reducing complexity in that part of work can help reduce the complexity of the whole algorithm substantially. Our experience shows in lots of cases it does.

VII Conclusions

In this paper we proposed a novel data layout scheme for dispersed network attached storage (DNAS), which was formulated as a new graph coloring problem called the *Layered Diversity Coloring Problem.* Examples show that the scheme tolerates data loss, keeps a good balance between delay and QoS and has a very graceful degradation performance.

We then studied the layered diversity coloring problem for trees. A coloring algorithm is presented, and the necessary and sufficient condition for there to exist a layered diversity coloring on a tree is proven. The algorithm has time complexity $O(|V|(K_1 + \sum_{i=1}^{p} m_i))$.

We're very interested in the layered diversity coloring problem for more general graphs, such as planar graphs. That remains as our future research topic.

References

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

[2] L. W. Dowdy and D. V. Foster. Comparative models of the file assignment problem. *Computing Surveys*, 14(2):287–313, 1982.

[3] J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.

[4] 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.

[5] S. Guha and S. Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the 9th Annual ACM SIAM Symposium on Discrete Algorithms*, pages 649–657, 1998.

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

[7] P. Mirchandani and R. Francis, editors. *Discrete location theory.* Wiley, New York, NY, 1990.

[8] M. E. Dyer and A. M. Frieze. A simple heuristic for the p-centre problem. *Operations Research Letters*, 3(6):285–288, 1985.

[9] D. S. Hochbaum and D. B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[10] E. Minieka. The m-center problem. *SIAM Review*, 12:138–139, 1970.

[11] A. Jiang and J. Bruck. Diversity coloring for distributed storage in mobile networks. Technical Report ETR 038, Parallel and Distributed Systems Lab, California Institute of Technology, http://www.paradise.caltech.edu/papers/etr038.pdf, 2001.

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

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

[14] E. D. Karnin, J. W. Greene, and M. E. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, IT-29(1):35–41, 1983.

[15] A. Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, 1979.