Journal of the Operations Research Society of China
https://doi.org/10.1007/s40305-023-00498-w

Fast Algorithm for the Rainbow Disconnection Coloring of 2-Trees

Xu-Qing Bai[1] · Bi Li[1] · Chuan-Dong Xu[1] · Xin Zhang[1]

Received: 7 February 2023 / Revised: 26 May 2023 / Accepted: 2 June 2023

Abstract

Given a connected graph $G = (V, E)$, a *rainbow disconnection k-coloring* of G is a k-edge coloring of G such that for each pair of vertices $u, v \in V$, there is a *rainbow (edge) cut* of u, v, which is a subset $\mathrm{RC}(u, v) \subseteq E$ such that u, v are disconnected in $G\backslash \mathrm{RC}(u, v)$ and that all edges in $\mathrm{RC}(u, v)$ have different colors. The minimum integer k such that G admits a rainbow disconnection k-coloring is the *rainbow disconnection number* of G, denoted by $\mathrm{rd}(G)$. It has been shown that $\mathrm{rd}(G)$ is closely related to the maximum number $\lambda^+(G)$ of edge disjoint paths among all pairs of vertices in G. In this paper, we propose a polynomial-time algorithm for finding a rainbow disconnection $\mathrm{rd}(G)$-coloring of a *2-tree*, a graph G formed by starting with a triangle and then repeatedly adding vertices in such a way that each added vertex is adjacent to two ends of an edge. Moreover, the algorithm shows that $\mathrm{rd}(G) = \lambda^+(G)$ if G is a 2-tree. This justifies a conjecture of Bai et al. [MR4385957] in 2-trees, which states that $\mathrm{rd}(G)$ is either $\lambda^+(G)$ or $\lambda^+(G) + 1$ for each graph G.

This work was supported by the Natural Science Basic Research Plan in Shaanxi Province of China (Nos. 2022JQ-009, 2023-JC-YB-001 and 2023-JC-YB-054), the Fundamental Research Funds for the Central Universities (No. XJS220705) and the National Natural Science Foundation of China (No. 12071370).

✉ Bi Li
libi@xidian.edu.cn

Xu-Qing Bai
baixuqing@xidian.edu.cn

Chuan-Dong Xu
xuchuandong@xidian.edu.cn

Xin Zhang
xzhang@xidian.edu.cn

[1] School of Mathematics and Statistics, Xidian University, Xi'an 710071, Shaanxi, China

Keywords Rainbow disconnection coloring · 2-Tree · Dynamic programming · Polynomial-time algorithm

Mathematics Subject Classification 05C15 · 05C40 · 68W40 · 90C27

1 Introduction

All graphs considered in this paper are finite, undirected and simple. For any notation or terminology not defined here, we follow those used in [1]. Let $G = (V(G), E(G))$ be a graph with vertex set $V(G)$ and edge set $E(G)$. A *u-v path* in G is a path with ends $u, v \in V(G)$. For a positive integer k, we always write $[k]$ for the set $\{1, 2, \cdots, k\}$ of integers. Let G be a graph with an *edge coloring* c: $E(G) \rightarrow [k]$, where adjacent edges may be colored the same. When adjacent edges of G receive different colors under c, the edge coloring c is called *proper*. The *chromatic index* of G, denoted by $\chi'(G)$, is the minimum number of colors needed in a proper edge coloring of G.

The connectivity of graphs plays a very important role in network flow theory and can be applied to aviation networks [2], computer networks [3], transport networks [4] and so on. In the literature, there are two ways to study the connectivity of graphs: One is to use paths, and the other is to use cuts. The rainbow connection using paths has been studied extensively; see, for example, papers [5–7] and books [8] and the references therein. On the other hand, the rainbow connection using cuts was initially investigated by Chartrand et al. [9] who introduced the concept of rainbow disconnection of graphs. Bai and Li [10] called proper edge colorings *local colorings* and called rainbow disconnection colorings *global colorings*.

In fact, the rainbow disconnection of graphs has the following application background. In the movement of people or goods, in order to intercept certain behaviors, such as the spread of COVID-19, drug smuggling and wildlife trade illegal activities, the responsible person needs to block these people or goods and locate them efficiently. Now that we know the people or goods want to move from a node A to another node B in the network, a natural measure to block such a movement is to assign interceptions on the edge cut between A and B, i.e., a collection of roads disconnecting B from A. We model this by installing one annunciator with a certain frequency on each road that can notify the responsible person, and therefore, an efficient interception is the found of an edge cut such that there is no two roads among them that are assigned a same frequency. Such an edge cut helps the responsible person know the location of the blocked objective just according to which frequency is received. To save costs, it is desirable to have as few frequencies as possible that are occupied. Hence, our goal now is equivalent to find an edge coloring of the network with minimum number of colors so that an "rainbow" edge cut of any pair of nodes can be computed.

Formally, an *edge cut* of a connected graph G is a set F of edges such that $G - F$ is disconnected. For two distinct vertices u and v of G, let $\lambda_G(u, v)$ (or simply $\lambda(u, v)$ when the graph G is clear from the context) denote the minimum number of edges in an edge cut F such that u and v lie in different components of $G - F$. The minimum cardinality of an edge cut of G is the *edge connectivity* of G, denoted by $\lambda(G)$ (i.e.,

$\lambda(G)$ is the minimum value of $\lambda_G(u, v)$ taken over all pairs of distinct vertices u, v), whereas the maximum value of $\lambda_G(u, v)$ taken over all pairs of distinct vertices u, v is the *upper edge connectivity* of G, denoted by $\lambda^+(G)$. This graph parameter $\lambda^+(G)$ was introduced and extensively studied by Bollobás in [11, 12]. The following proposition presents another interpretation for $\lambda_G(u, v)$.

Proposition 1 [13, 14] *For every two vertices u and v in a graph G, $\lambda(u, v)$ is equal to the maximum number of pairwise edge disjoint u-v paths in G.*

An edge cut R of an edge-colored connected graph G is a *rainbow cut* if no two edges in R are colored the same. Let u and v be two distinct vertices of G. A rainbow edge cut R of G is called a *u-v-rainbow cut* if u and v belong to different components of $G - R$. An edge-colored graph G is called *rainbow disconnected* if for every two distinct vertices u and v of G, there exists a u-v-rainbow cut in G. In this case, the edge coloring is called a *rainbow disconnection coloring* (or *rd-coloring* for short) of G. For a connected graph G, the *rainbow disconnection number* of G, denoted by $\mathrm{rd}(G)$, is defined as the smallest number of colors required to make G rainbow disconnected. An optimal *rd-coloring* of G is an rd-coloring with $\mathrm{rd}(G)$ colors.

Bai et al. [15] studied the rainbow disconnection numbers of many classes of graphs including connected regular graphs and complete multipartite graphs, and showed that it is NP-complete to decide whether $\mathrm{rd}(G) = 3$ for a connected cubic graph G. Based on the known results concerning the rainbow disconnection numbers of graphs in [9, 15], Bai et al. [16] proposed the following conjecture.

Conjecture 1 *For any connected graph G, $\lambda^+(G) \leqslant \mathrm{rd}(G) \leqslant \lambda^+(G) + 1$.*

Furthermore, Bai et al. [16] verified the conjecture for subgraph-overfull graphs, subcubic graphs and graphs G whose maximum degree is at least $|G| - 3$.

Many algorithmic problems that are NP-complete for arbitrary graphs may be solved efficiently for k-trees by dynamic programming. A graph G is called a *k-tree* if and only if either $G \cong K_{k+1}$ or there is a vertex v of degree k such that $N_G(v)$ induces K_k and $G - v$ is a k-tree. It is easy to observe that the 1-trees are exactly the trees.

Chartrand et al. [9] obtained that for a tree T, $\mathrm{rd}(T) = 1$, namely $\mathrm{rd}(T) = \lambda^+(T)$. Motivated by this and the definition of k-tree, we ask the following natural question.

Question 1 *Given a positive integer k, whether* $\mathrm{rd}(G) = \lambda^+(G)$ *for every k-tree*?

This article is arranged as follows. In Sect. 2, we first present several auxiliary lemmas. In Sect. 3, we design an algorithm to compute $\lambda_G(u, w)$ for any edge uw in a 2-tree G. Finally in Sect. 4, we give a positive answer to the above question for the case that $k = 2$. Precisely, we built up a quadratic-time algorithm for 2-trees of finding rainbow disconnection $\lambda^+(G)$-colorings. This further supports Conjecture 1.

In the rest of this paper, the *intersection* $G \cap H$ of two graphs G and H is the graph on $V(G) \cap V(H)$ with the edge set $E(G) \cap E(H)$, and the *union* $G \cup H$ of two graphs G and H is the graph on $V(G) \cup V(H)$ with the edge set $E(G) \cup E(H)$.

2 Preliminaries

Lemma 1 *Given a graph $G = (V, E)$, let x, y be two vertices of G and let S be a minimum edge cut of x, y in G. If $uw \in S$, then S is also an edge cut of u, w in G.*

Proof If not, u, w are in the same component in $G \setminus S$, which contains at most one of x, y, then $S \setminus \{uw\}$ is an edge cut of x, y in G. This contradicts that S is a minimum edge cut of x, y in G.

Lemma 2 *If $G = (V, E)$ is a 2-tree and $\{u, v, w\} \subseteq V$ induces a triangle T, then there are three connected subgraphs $G(uv|w)$, $G(uw|v)$ and $G(vw|u)$ of G such that*

(1) *$G(uv|w) \cap T = T - w$, $G(uw|v) \cap T = T - v$, and $G(vw|u) \cap T = T - u$;*
(2) *$G(uv|w) \cup G(uw|v) \cup G(vw|u) = G$;*
(3) *$G(uv|w) \cap G(uw|v) = (\{u\}, \varnothing)$, $G(uv|w) \cap G(vw|u) = (\{v\}, \varnothing)$, and $G(uw|v) \cap G(vw|u) = (\{w\}, \varnothing)$;*
(4) *each of $G(uv|w)$, $G(uw|v)$ and $G(vw|u)$ is a 2-tree as long as it has at least three vertices.*

Proof We proceed induction on $|G|$. The base case that $|G| = 3$ is trivial so we assume that $|G| \geqslant 4$. Since G is a 2-tree, it has a vertex y of degree 2 whose two neighbors are adjacent.

If $y \in \{u, v, w\}$, then assume by symmetry that $u = y$. It follows that $G - u$ is still a 2-tree. Let $G(uv|w)$ and $G(uw|v)$ just be the edge uv and uw, respectively, and let $G(vw|u) = G - u$. One can easily check that $G(uv|w)$, $G(uw|v)$ and $G(vw|u)$ satisfy the four conditions.

If $y \notin \{u, v, w\}$, then T is an triangle in the graph $G' = G - y$. By the induction hypothesis, there are three connected subgraphs $G'(uv|w)$, $G'(uw|v)$ and $G'(vw|u)$ of G' such that those conditions hold for G'. Since the intersection of any two graphs among $G'(uv|w)$, $G'(uw|v)$ and $G'(vw|u)$ is a single vertex and $G'(uv|w) \cup G'(uw|v) \cup G'(vw|u) = G'$, the two neighbors of y in G live in exactly one graph among those three, say $G'(uv|w)$, for example. Now $G'(uv|w) + y$ is still a 2-tree. Let $G(uv|w) = G'(uv|w) + y$, $G(uw|v) = G'(uw|v)$, and $G(vw|u) = G'(vw|u)$. One can again quickly check that they are connected subgraphs of G satisfying the four conditions.

Lemma 3 *Given a 2-tree $G = (V, E)$ with order $|V| \geqslant 3$, $\{u, v, w\} \subseteq V$ induces a triangle in G. Let x, y be two adjacent vertices in $G(uw|v)$ and let S be a minimum cut of x, y in G. If $uw \in S$ and $\lambda_{G(vu|w)}(v, u) \leqslant \lambda_{G(vw|u)}(v, w)$, then we have that:*

(1) *For any minimum cut F of v, u in $G(vu|w)$, $(S \cap G(uw|v)) \cup F$ is a minimum cut of x, y in G;*
(2) *If $vu \in S$, then $S \cap G(vu|w)$ is a minimum cut of v, u in $G(vu|w)$ and $S \cap G(vw|u) = \varnothing$;*
(3) *If $\{x, y\} = \{u, w\}$, then $S \cap G(uw|v)$ is a minimum cut of u, w in $G(uw|v)$.*

Proof For easy description, we denote the three pairwise disjoint subsets $S \cap G(uw|v)$, $S \cap G(vu|w)$ and $S \cap G(vw|u)$ of S as S_{uw}, S_{vu} and S_{vw}, respectively. Then $S = S_{uw} \cup S_{vu} \cup S_{vw}$. Note that S_{uw} is a x, y cut in $G(uw|v)$.

Since $uw \in S$, S is also a cut of u, w from Lemma 1. So S_{uw} is a cut of u, w in $G(uw|v)$. Moreover, we claim that S is either a cut of v, u or v, w in G. Since if not, there are a u-v path and a v-w path in $G \setminus S$, and then, connecting these two paths gives a u-w path. It is a contradiction.

(a) If S is a cut of v, u in G, then S_{vu} is a cut of v, u in $G(vu|w)$. Then $S_{uw} \cup S_{vu}$ is a cut of u, w in G. This is because that any u-w path in $G \backslash (S_{uw} \cup S_{vu})$ goes through v and this implies that there is a subpath connecting u and v in $G(vu|w)$. It contradicts with S_{vu} is a cut of v, u in $G(vu|w)$.
We claim that $S_{uw} \cup S_{vu}$ is also a cut of x, y in G. Suppose if not and there is an x-y path P in $G \backslash (S_{uw} \cup S_{vu})$. If $P \subseteq G_{uw|v}$, it contradicts with that S_{uw} is a x, y cut in $G(uw|v)$. If P contains an edge in $G(vu|w)$, then P contains a u-v path in $G(vu|w)$ as a subpath, which contradicts with that S_{vu} is a v, u cut in $G(vu|w)$. If P contains an edge in $G(vw|u)$, then P contains a u-w path in G as a subpath, which contradicts with that $S_{uw} \cup S_{vu}$ is a cut of u, w in G. So $S = S_{uw} \cup S_{vu}$ and $S_{vw} = \emptyset$, which implies $|S| = |S_{uw}| + |S_{vu}| \geqslant |S_{uw}| + \lambda_{G(vu|w)}(v, u)$ from $S_{uw} \cap S_{vu} = \emptyset$.

(b) Otherwise S is a cut of v, w in G and S_{vw} is a cut of v, w in $G(vw|u)$. We similarly prove that $S = S_{uw} \cup S_{vw}$ and $S_{vu} = \emptyset$, which implies $|S| = |S_{uw}| + |S_{vw}| \geqslant |S_{uw}| + \lambda_{G(vw|u)}(v, w) \geqslant |S_{uw}| + \lambda_{G(vu|w)}(v, u)$.

One sees that in both case, $|S| \geqslant |S_{uw}| + \lambda_{G(vu|w)}(v, u)$.

To prove (1), let F be a minimum cut of v, u in $G(vu|w)$. As proved in case (a) for $S_{uw} \cup S_{vu}$, one can prove that $S_{uw} \cup F$ is also a cut of x, y in G. Since $|S_{uw} \cup F| = |S_{uw}| + |F| = |S_{uw}| + \lambda_{G(vu|w)}(v, u) \leqslant |S|$, so $S_{uw} \cup F$ is also a minimum cut of x, y in G. Moreover, $|S| = |S_{uw}| + \lambda_{G(vu|w)}(v, u)$.

To prove (2), if $vu \in S$, then S is a cut of v, u in G from Lemma 1 and we are in case (a). So S_{vu} is a cut of v, u in $G(vu|w)$ and $S = S_{uw} \cup S_{vu}$. Since $|S| = |S_{uw}| + \lambda_{G(vu|w)}(v, u)$ and $|S| = |S_{uw}| + |S_{vu}|$, we have that $|S_{vu}| = \lambda_{G(vu|w)}(v, u)$. So $S \cap G(vu|w)$ is a minimum cut of v, u in $G(vu|w)$.

To prove (3), since $\{x, y\} = \{u, w\}$, if $S \cap G(uw|v)$ is not a minimum cut of u, w in $G(uw|v)$, then replace $S \cap G(uw|v)$ with a minimum cut of u, w in $G(uw|v)$ in S. The obtained set is a smaller cut of u, w in G. It is a contradiction.

A graph is *chordal* if the maximum induced cycle is at most 3. There are several equivalent definitions for chordal graph. We will apply the one related to *clique tree*, or a *tree decomposition* with each bag inducing a maximal clique in the graph.

Definition 1 Given a graph $G = (V, E)$, a tree decomposition of G is a pair $(T, \mathcal{X})$, where T is a tree, and $\mathcal{X} = \{X_t | t \in V(T)\}$ is a family of subsets of V, called bags, such that:

- $\cup_{X \in \mathcal{X}} = V$;
- For any edge $uv \in E$, there is a bag $X_t \in \mathcal{X}$ for some $t \in V(T)$ containing both u and v;
- For any vertex $v \in V$, the set $\{t \in V(T) | v \in X_t\}$ induces a subtree in T.

Lemma 4 [17] *A graph $G = (V, E)$ is chordal graph if and only if G has a clique tree, i.e., a tree decomposition $(T, \mathcal{X})$ such that each bag $X \in \mathcal{X}$ induces a clique in G.*

Lemma 5 [18] *The chromatic number of a chordal graph is equal to its clique number, i.e., the size of the maximum clique. Moreover, an optimal vertex coloring of a chordal graph can be found in linear time.*

3 Computing λ

Let $G = (V, E)$ be a 2-tree and $uw \in E(G)$. We are going to compute the size $\lambda_G(u, w)$ of minimum edge cut of u and w. Every edge $uw \in E(G)$ is in a triangle, induced by, for example, $\{u, w, v\}$, in G. From Lemma 3, in any minimum edge cut of u and w, exactly one of the two edges vu and vw is included, depending on the two sizes $\lambda_{G(vu|w)}(v, u)$ and $\lambda_{G(vw|u)}(v, w)$ of the minimum edge cuts. Note that the subgraphs $G(vu|w)$ and $G(vw|u)$ are both 2-trees from Lemma 2. In the following two lemmas, we prove first the equations of $\lambda_G(u, w)$ with the corresponding values in the related sub-2-trees. Then we describe our algorithm.

Lemma 6 *Given a 2-tree $G = (V, E)$, $u, v, w \in V(G)$ induce a triangle in G. Then*

$$\lambda_G(u, w) = \min\{\lambda_{G(vu|w)}(v, u), \lambda_{G(vw|u)}(v, w)\} + \lambda_{G(uw|v)}(u, w).$$

Proof From the proof of Lemma 3, we see that any a minimum edge cut of u, w in G is the union of a minimum cut of u, w in $G(uw|v)$ and the minimum one between a minimum cut of v, u in $G(vu|w)$ and a minimum cut of v, w in $G(vw|u)$. This proves the lemma directly.

Corollary 1 *Given a 2-tree $G = (V, E)$, let $u, w \in V(G)$ and $uw \in E(G)$. There are d common neighbors $z_1, z_2, \cdots, z_d$ for some $d \geqslant 1$ of u, w in G. Then $\lambda_G(u, w) = 1 + \sum_{i=1}^{d} \min\{\lambda_{G(z_i u|w)}(z_i, u), \lambda_{G(z_i w|u)}(z_i, w)\}$.*

Proof For any $1 \leqslant i \leqslant d$, $\{u, z_i, w\}$ induces a triangle in G. From Lemma 6,

$$\lambda_G(u, w) = \min\{\lambda_{G(z_1 u|w)}(z_1, u), \lambda_{G(z_1 w|u)}(z_1, w)\} + \lambda_{G(uw|z_1)}(u, w).$$

Then we apply Lemma 6 in $G^1 = G(uw|z_1)$ recursively for the $d - 1$ triangles induced by $\{u, z_i, w\}$ and denote by $G^i = G^{i-1}(uw|z_i)$ for $2 \leqslant i \leqslant d$. In the end, the connected component G^d contains exactly one edge uw. So

$$\begin{aligned}\lambda_G(u, w) &= \sum_{i=1}^{d} \min\{\lambda_{G(z_i u|w)}(z_i, u), \lambda_{G(z_i w|u)}(z_i, w)\} + \lambda_{G^d}(u, w)\\ &= \sum_{i=1}^{d} \min\{\lambda_{G(z_i,u|w)}(z_i, u), \lambda_{G(z_i,u|w)}(z_i, w)\} + 1.\end{aligned}$$

From Corollary 1, one sees that for any edge $uw \in E(G)$, $\lambda_G(u, w)$ can be obtained by computing the corresponding value of the edges of the triangles containing the edge uw in corresponding subgraphs. Then we apply this corollary recursively until the subgraphs are only edges.

For easy computing, given a 2-tree $G = (V, E)$, we construct a tree T_G, called the *triangle tree* of G, as follows: The vertex set of T_G is the set of triangles and edges in G, for a triangle vertex $t \in V(T_G)$ and an edge vertex $e \in V(T_G)$, $te \in E(T_G)$ if and only if e is an edge of the triangle t in G.

Lemma 7 *Given a 2-tree $G = (V, E)$, the triangle tree T_G of G is a tree.*

Proof From the construction of a 2-tree, one sees that there are $|V(G)| - 2$ triangles and $2|V(G)| - 3$ edges in G, so there are $3|V(G)| - 5$ vertices and $3(|V(G)| - 2) = 3|V(G)| - 6$ edges in T. It is sufficient to prove that there is no cycle in T_G. We see that T_G is bipartite with two parts of the set of triangle vertices and the set of edge vertices. So every cycle in T_G, if exists, contains two edge disjoint paths of some pair of edge vertices of distance two in T_G. Let $u, v, w \in V(G)$ induce a triangle in G. Then the corresponding vertices of the two edges vu and vw in T_G have a common neighbor the triangle vertex uvw in T_G. So vu and vw have distance two in T_G. We are going to prove that these three vertices in T_G consist of the unique path of the two vertices vu and vw in T_G. This is because that from Lemma 2, in $G \setminus \{u, v, w\}$, each connected component is adjacent to exactly two vertices of the triangle. Correspondingly, each connected component in $T_G \setminus \{vu, vw, uvw\}$ is adjacent to exactly one vertex of $\{vu, vw, uw, uvw\}$. So there is no other path from vu to vw. So T_G is a tree.

Now we describe how to compute $\lambda_G(u, w)$ for any edge uw in a 2-tree G in Algorithm 1.

Algorithm 1: Linear algorithm for computing $\lambda_G(u, w)$ for any edge uw in a 2-tree G

```
Input: A 2-tree $G = (V, E)$ and an edge $uw \in E$.
Output: $\lambda_G(u, w)$.
1  Construct the triangle tree $T_G$ of $G$;
2  Root $T_G$ at $uw$;
3  $\gamma \leftarrow$ the furthest distance of any vertex in $T_G$ to the root $uw$;
4  for $i = 0$ to $\gamma$ do
5      $L_i \leftarrow$ the set of vertices in $T_G$ at distance $i$ to the root $uw$;
6  for each leaf vertex $l \in V(T_G)$ do
7      $\lambda(l) \leftarrow 1$;
8  for $j = \gamma/2$ to $1$ do
9      for $t \in L_{2j-1}$ do
10         $\lambda(t) \leftarrow \min\{\lambda(e_1), \lambda(e_2) | e_1, e_2$ are the two children of $t$ in $T_G\}$;
11     for $e \in L_{2j-2}$ do
12         $\Delta_e \leftarrow$ the set of children of $e$ in $T_G$;
13         if $\Delta_e \neq \emptyset$ then
14             $\lambda(e) \leftarrow 1 + \sum_{t \in \Delta_e} \lambda(t)$;
15 Output $\lambda(uw)$.
```

Theorem 1 *Given a 2-tree $G = (V, E)$ and an edge $uw \in E$, Algorithm 1 outputs $\lambda_G(u, w)$ in linear time.*

Proof From lines 1–3 in Algorithm 1, one sees that the triangle tree T_G are rooted at the vertex uw and the furthest vertex from uw in T_G is at distance γ. Note that every leaf vertex in T_G is an edge vertex since every triangle vertex has degree 3 in T_G. Then γ is an even number since the furthest vertex is a leaf vertex, and then, it is an edge vertex. As in line 5, L_i is the set of vertices at distance i from the root uw in T_G for $i = 0, 1, \cdots, \gamma$. Then $L_0 = \{uw\}$ and for $j = 1, \cdots, \gamma/2$, each L_{2j} is set of edge vertices and each L_{2j-1} is set of triangle vertices.

From lines 6–7, we see that $\lambda(l) = 1$ for each leaf vertex $l \in V(T_G)$. Suppose that a leaf vertex $l \in V(T_G)$ corresponds to the edge $pq \in E(G)$. Let t be the parent of l in T_G. Then for some vertex $o \in V(G)$, pq is contained in the triangle opq of G corresponding to $t \in V(T_G)$. Note that every triangle vertex has exactly two children and one parent in T_G. Without loss of generality, we assume that po corresponds to the other child of t in $V(T_G)$ and oq corresponds to the parent of t in $V(T_G)$. So $p \in V(G)$ has degree two and is adjacent to $o, q \in V(G)$. Then one sees that $G(pq|o)$ has exactly two vertices p, q and only one edge pq. So $\lambda_{G(pq|o)}(p, q) = 1$, which is equal to $\lambda(l) = 1$. From line 10 for $j = \gamma/2$ and $t \in L_{2j-1}$, $\lambda(t) = \min\{\lambda(pq), \lambda(po)\} = 1$. We assume that:

- For $j = \gamma/2, \cdots, 2$ and any edge vertex $xy \in L_{2j-2}$, $\lambda(xy)$ is equal to $\lambda_{G(xy|z)}(x, y)$, where z satisfies that xyz is a triangle in G and xz is the parent of the triangle vertex in T_G corresponding to xyz;
- For $j = \gamma/2, \cdots, 1$ and any triangle vertex $abc = t \in L_{2j-1}$, $\lambda(t)$ is equal to $\min\{\lambda_{G(ab|c)}(a, b), \lambda_{G(bc|a)}(b, c)\}$, where ac is (ab, bc are) the parent (two children) of the triangle vertex abc in T_G.

We see that the above assumptions are satisfied for $L_\gamma, L_{\gamma-1}$, where $j = \gamma/2$. Suppose that the assumptions are true for $L_\gamma, \cdots, L_1$. Then we prove it for L_0. When $j = 1$, in $L_{2j-2} = L_0 = \{uw\}$, $\lambda(uw) = 1 + \sum_{t\in\Delta_{uw}} \lambda(t)$ as in line 14, where Δ_{uw} is the set of children of uw in T_G defined in line 12. From the assumption and Corollary 1, we see that $\lambda(uw) = \lambda_G(u, w)$.

Since $\gamma \leqslant |V(G)|$, the algorithm outputs $\lambda(uw)$ in linear time. The theorem is proved.

4 Quadratic Algorithm of Finding an Optimal Rd-coloring

Given a 2-tree $G = (V, E)$, one can find an order of the vertices in $V = \{v_1, v_2, \cdots, v_n\}$ such that, for any $3 \leqslant i \leqslant n$, v_i is adjacent to exactly the two ends of an edge in the subgraph G_i induced by $V_i = \{v_1, v_2, \cdots, v_i\}$. So for each $3 \leqslant i \leqslant n$, v_i has degree 2 in G_i and v_1v_2 is the unique edge in G_2. For easy description, we call the above order as a *construction ordering* of G.

The main idea of our dynamic programming algorithm is as follows: Start from G_2 with the cut $\{v_1v_2\}$ of v_1, v_2; in each step, add some vertex v_i for $3 \leqslant i \leqslant n$, updating the cuts of pairs of vertices in G_{i-1} to get the cuts in G_i. In the end, we get a cut for each pair of vertices in $G_n = G$. To color the edges in G such that each obtained cut is rainbow, a graph H is constructed: The vertex set of H is the edge set of G and two vertices in H are adjacent if and only if their corresponding edges in G are contained in a common cut obtained in G. So a proper vertex coloring of H gives a rainbow disconnection coloring of G. We will see that every pair of non-adjacent vertices can be disconnected by some cut of two adjacent vertices in some G_i for $2 \leqslant i \leqslant n$, and so, it is sufficient to find the rainbow cut of adjacent vertices in all G_is. Moreover, for $2 \leqslant i \leqslant n$ and any adjacent vertices $u, v \in G_i$, we will prove that in each step i the size of the obtained cut of u, v is at most $\lambda_G(u, v)$, and so, the clique number $\omega(H)$, i.e., the size of the maximum clique of H, is $\lambda^+(G)$. Furthermore, we will prove that H is chordal and its chromatic number is equal to $\omega(H) = \lambda^+(G)$. Thus, an optimal rd-coloring of G is achieved.

Notations. Before describe the algorithm, for any two vertices $u, w \in V(G)$ we explain some notations used in the algorithm:

- $\mathrm{RC}(u, w)$ denotes a rainbow cut of u and w;
- $\mathcal{S}_{\mathrm{RC}}(uw)$ denotes the set of rainbow cuts containing the edge uw.

Algorithm 2: Quadratic algorithm for rainbow disconnection coloring of 2-trees

```
Input: A 2-tree $G = (V, E)$.
Output: A rainbow disconnection coloring $\varphi$ of $G$.
Find a construction ordering $V = \{v_1, v_2, \cdots, v_n\}$ of $G$;
$\mathrm{RC}(v_1, v_2) \leftarrow \{v_1v_2\}$;    $\mathcal{S}_{\mathrm{RC}}(v_1v_2) \leftarrow \{\mathrm{RC}(v_1, v_2)\}$;
$H \leftarrow$ a graph with only one vertex $v_1v_2$;
for $i = 3$ to $n$ do
    $u_iw_i \leftarrow$ the unique edge in $G_{i-1}$ with two ends adjacent to $v_i$;
    if $\lambda_{G(v_iu_i|w_i)}(v_i, u_i) \leqslant \lambda_{G(v_iw_i|u_i)}(v_i, w_i)$ then
        $x \leftarrow u_i$;    $y \leftarrow w_i$;
    else
        $x \leftarrow w_i$;    $y \leftarrow u_i$;
    Add two new vertices $v_ix$ and $v_iy$ in $H$;
    for each $F \in \mathcal{S}_{\mathrm{RC}}(xy)$ do
        Make $v_ix$ adjacent to each element of $F$ in $H$;
        $F \leftarrow \{v_ix\} \cup F$;
    $\mathcal{S}_{\mathrm{RC}}(v_ix) \leftarrow \mathcal{S}_{\mathrm{RC}}(xy)$;
    if $\lambda_{G(v_iy|x)}(v_i, y) \leqslant \lambda_{G(xy|v_i)}(x, y)$ then
        $\mathrm{RC}(v_i, x) \leftarrow \{v_ix, v_iy\}$;    $\mathcal{S}_{\mathrm{RC}}(v_ix) \leftarrow \mathcal{S}_{\mathrm{RC}}(v_ix) \cup \{\mathrm{RC}(v_i, x)\}$;
        $\mathrm{RC}(v_i, y) \leftarrow \{v_ix, v_iy\}$;    $\mathcal{S}_{\mathrm{RC}}(v_iy) \leftarrow \{\mathrm{RC}(v_i, y)\}$;
        Make $v_iy$ adjacent to $v_ix$ in $H$;
    else
        $\mathrm{RC}(v_i, x) \leftarrow \mathrm{RC}(x, y)$;
        if $\lambda_{G(v_ix|y)}(v_i, x) \leqslant \lambda_{G(xy|v_i)}(x, y)$ then
            $\mathrm{RC}(v_i, y) \leftarrow \{v_ix, v_iy\}$;    $\mathcal{S}_{\mathrm{RC}}(v_iy) \leftarrow \{\mathrm{RC}(v_i, y)\}$;
            Make $v_iy$ adjacent to $v_ix$ in $H$;
        else
            $\mathrm{RC}(v_i, y) \leftarrow \{v_iy\} \cup \mathrm{RC}(x, y) \setminus \{v_ix\}$;    $\mathcal{S}_{\mathrm{RC}}(v_iy) \leftarrow \{\mathrm{RC}(v_i, y)\}$;
            Make $v_iy$ adjacent to each element of $\mathrm{RC}(v_i, y) \setminus \{v_iy\}$ in $H$;
$\varphi \leftarrow$ an optimal proper (vertex) coloring of $H$;
Output $\varphi$.
```

Theorem 2 *Given a 2-tree $G = (V, E)$, Algorithm 2 outputs an optimal* rd*-coloring of G in quadratic time.*

Proof Given a 2-tree $G(V, E)$, in line 1 the algorithm finds a construction order $V = \{v_1, v_2, \cdots, v_n\}$ of G. The proof will proceed by induction on i. For $i = 2$, line 2 describes a cut $\mathrm{RC}(v_1, v_2) = \{v_1 v_2\}$ of v_1, v_2 in G_2 and line 3 constructs a graph H with only one vertex $v_1 v_2$. Note that $v_1 v_2$ denotes a vertex in H, corresponding to the edge in G. The following assumptions are satisfied for $i = 2$:

(I) For each adjacent pair of vertices $u, w \in V(G_i)$, $\mathrm{RC}(u, w)$ is a cut of u, w in G_i and there is a minimum cut $M(u, w)$ of u, w in G such that $\mathrm{RC}(u, w) \subseteq M(u, w)$, which replies that $|\mathrm{RC}(u, w)| \leqslant \lambda_G(u, w)$;

(II) For any pair of non-adjacent vertices $v, z \in V(G_i)$, there is a pair of adjacent vertices $u, w \in V(G_i)$, such that the cut $\mathrm{RC}(u, w)$ is also a cut of v, z in G_i;

(III) The constructed graph H is chordal; and an obtained cut of G corresponds to a clique in H and vice verse.

Suppose that for $i = j - 1$ for some $3 \leqslant j \leqslant n$ in G_{j-1} all the above assumptions are satisfied according to lines 4–26 in Algorithm 2. We are going to prove that they are also satisfied for $i = j$ in G_j. Then in $G_n = G$, from properties (I) and (II), we get a cut of each pair of vertices in G.

Note that $V(G_i)\backslash V(G_{i-1}) = \{v_i\}$ and u_i, w_i are the two adjacent neighbors of v_i in G_i. To be distinguishable, we denote the cut of u_i, w_i in G_{i-1} by $\mathrm{RC}_{i-1}(u_i, w_i)$, which is contained in a minimum cut of u_i, w_i in G; and the graph H obtained until step $i-1$ is denoted as H_{i-1}. From lines 6–9, one sees that $\lambda_{G(v_i x|y)}(v_i, x) \leqslant \lambda_{G(v_i y|x)}(v_i, y)$ for $x, y \in \{u_i, w_i\}$.

To prove (I), first for any two adjacent vertices $u, w \in V(G_{i-1})$, if $xy \notin \mathrm{RC}_{i-1}(u, w)$, then $\mathrm{RC}_i(u, w) = \mathrm{RC}_{i-1}(u, w)$ is still a cut of u, w in G_i; otherwise, $xy \in \mathrm{RC}_{i-1}(u, w)$ and so $\mathrm{RC}_{i-1}(u, w) \in \mathcal{S}_{\mathrm{RC}_{i-1}}(xy)$. Then it is sufficient to add $v_i x$ or $v_i y$ in $\mathrm{RC}_{i-1}(u, w)$ to get a cut of u, w in G_i. As in line 13, $\mathrm{RC}_i(u, w) = \mathrm{RC}_{i-1}(u, w) \cup \{v_i x\}$ is a cut of u, w in G_i. Let F be a minimum cut of v_i, x in $G(v_i x|y)$ and so $v_i x \in F$. Since $xy \in \mathrm{RC}_{i-1}(u, w) \subseteq (M(u, w) \cap G(xy|v_i))$, then from Lemma 3, $M(u, w) \cap G(xy|v_i) \cup F$ is a minimum cut of u, w in G. So we have that $\mathrm{RC}_i(u, w) \subseteq M(u, w) \cap G(xy|v_i) \cup F$ and $|\mathrm{RC}_i(u, w)| \leqslant \lambda_G(u, w)$.

Now we consider the two pairs of adjacent vertices v_i, x and v_i, y in G_i. There are two cases in lines 15–26:

- If $\lambda_{G(v_i y|x)}(v_i, y) \leqslant \lambda_{G(xy|v_i)}(x, y)$, then $\lambda_{G(v_i x|y)}(v_i, x) \leqslant \lambda_{G(v_i y|x)}(v_i, y) \leqslant \lambda_{G(xy|v_i)}(x, y)$. One sees that $\mathrm{RC}_i(v_i, x) = \{v_i x, v_i y\}$, as in line 16, is a cut of v_i, x in G_i and $|\mathrm{RC}_i(v_i, x)| \leqslant \lambda_G(v_i, x)$. The case for v_i, y can be proved similarly.
- Otherwise, $\lambda_{G(v_i y|x)}(v_i, y) > \lambda_{G(xy|v_i)}(x, y)$. Then $\mathrm{RC}_i(v_i, x) = \mathrm{RC}_{i-1}(x, y) \cup \{v_i x\} = \mathrm{RC}_i(x, y)$, as in line 20, is a cut of v_i, x in G_i. Let $M(x, y)$ be a minimum cut of x, y in G such that $\mathrm{RC}_{i-1}(x, y) \subseteq M(x, y)$. Then $(M(x, y) \cap G(xy|v_i)) \supseteq \mathrm{RC}_{i-1}(x, y)$ is a minimum cut of x, y in $G(xy|v_i)$ from (3) of Lemma 3. So $\mathrm{RC}_i(v_i, x) = \mathrm{RC}_{i-1}(x, y) \cup \{v_i x\}$ is contained in a minimum cut of v_i, x in G which consists of a minimum cut of v_i, x in $G(v_i x|y)$ and $M(x, y) \cap G(xy|v_i)$ from Lemma 3. So $|\mathrm{RC}_i(v_i, x)| \leqslant \lambda_G(v_i, x)$. There are two subcases for the pair v_i, y:

- If $\lambda_{G(v_ix|y)}(v_i, x) \leqslant \lambda_{G(xy|v_i)}(x, y)$, then $\mathrm{RC}_i(v_i, y) = \{v_ix, v_iy\}$, as in line 22, is a cut of v_i, y in G_i and $|\mathrm{RC}_i(v_i, y)| \leqslant \lambda_G(v_i, y)$.
- Otherwise, $\lambda_{G(v_ix|y)}(v_i, x) > \lambda_{G(xy|v_i)}(x, y)$. Then $\mathrm{RC}_i(v_i, y) = \mathrm{RC}_{i-1}(x, y) \cup \{v_iy\} = \mathrm{RC}_i(x, y) \cup \{v_iy\} \backslash \{v_ix\}$, as in line 25, is a cut of v_i, y in G_i. Similarly as the case for v_i, x, we can prove from Lemma 3 that $\mathrm{RC}_i(v_i, y)$ is contained in a minimum cut of v_i, y in G. So $|\mathrm{RC}_i(v_i, y)| \leqslant \lambda_G(v_i, y)$.

To prove (II), for any non-adjacent two vertices $v, z \in G_i$, if $v_i \notin \{v, z\}$, then from the assumption, $\mathrm{RC}_{i-1}(v, z)$ is equal to some $\mathrm{RC}_{i-1}(u, w)$ for some adjacent pair u, w in G_{i-1}. So $\mathrm{RC}_i(v, z) = \mathrm{RC}_i(u, w)$ is a cut of v, z in G_i as proved in the proof of (I).

Otherwise, $v_i \in \{v, z\}$ and without loss of generality, assume that $v = v_i$. If $\mathrm{RC}_i(v_i, x) = \{v_ix, v_iy\}$ or $\mathrm{RC}_i(v_i, y) = \{v_ix, v_iy\}$, then $\{v_ix, v_iy\}$ is a cut of v_i, z in G_i. Otherwise we have that $\mathrm{RC}_i(v_i, x) = \mathrm{RC}_{i-1}(x, y) \cup \{v_ix\}$ and $\mathrm{RC}_i(v_i, y) = \mathrm{RC}_{i-1}(x, y) \cup \{v_iy\}$. Since $z \neq v_i$, $z \in V(G_{i-1})$. If z is in the same component with x in $G_{i-1} \backslash \mathrm{RC}_{i-1}(x, y)$, then $\mathrm{RC}_i(v_i, x)$ is a cut of v_i, z in G_i; otherwise, $\mathrm{RC}_i(v_i, y)$ is a cut of v_i, z in G_i.

To prove (III), from the assumption we have that H_{i-1} is chordal; moreover, any edges of G_{i-1} contained in some recorded cut together induce a clique in H_{i-1} and vice verse. One sees that in line 3, the graph H_2 has only one vertex v_1v_2, which satisfies the assumption. It is sufficient to prove that H_i is chordal; and any edges of G_i contained in some recorded cut together induce a clique in H_i and vice verse.

As in line 10, each step two new vertices are added and $V(H_i) \setminus V(H_{i-1}) = \{v_ix, v_iy\}$. Let us see the new added edges related to the two new vertices v_ix and v_iy in H_i: as in lines 11–12, v_ix is adjacent to any one, which is contained in a cut F_{i-1} together with xy, i.e., $F_{i-1} \in \mathcal{S}_{\mathrm{RC}_{i-1}}(xy)$. Note that at the same time, F_{i-1} is also updated to be F_i including v_ix. So in H_i, v_ix is adjacent to any edge e of G_{i-1} if and only if e is a neighbor of xy in H_{i-1}. Moreover, a clique in H_{i-1} containing xy becomes a larger clique in H_i by adding v_ix. In lines 18 and 23, v_iy is adjacent to v_ix if $\mathrm{RC}_i(v_i, y) = \{v_ix, v_iy\}$. Otherwise, in line 26, v_iy is adjacent to each element of $\mathrm{RC}_i(v_i, y) \backslash \{v_iy\} = \mathrm{RC}_{i-1}(x, y)$. Since $\mathcal{S}_{\mathrm{RC}_i}(v_iy) = \mathrm{RC}_i(v_i, y)$, in any case we have that the neighborhood of v_iy induces a clique in H_i, denoted as Y. So any edges of G_i contained in some recorded cut together induce a clique in H_i and vice verse.

From Lemma 4, we assume that $(T_{i-1}, \mathcal{X}_{i-1})$ is a tree decomposition of H_{i-1} such that each bag of $\mathcal{X}_{i-1}$ induces a clique in H_{i-1}. We will construct such a tree decomposition of H_i in the following to show that H_i is chordal. As shown in Fig. 1, to construct a tree decomposition of H_i, add a vertex v_ix to each bag containing xy in $(T_{i-1}, \mathcal{X}_{i-1})$. Moreover, add a new bag Y adjacent to the bag containing $\mathrm{RC}_{i-1}(x, y)$. (Note that $\mathrm{RC}_{i-1}(x, y)$ induces a clique in H_{i-1} and so there is a bag in $(T_{i-1}, \mathcal{X}_{i-1})$ containing $\mathrm{RC}_{i-1}(x, y)$.) The obtained tree decomposition $(T_i, \mathcal{X}_i)$ is a tree decomposition H_i such that each bag induces a clique in H_i. So H_i is chordal.

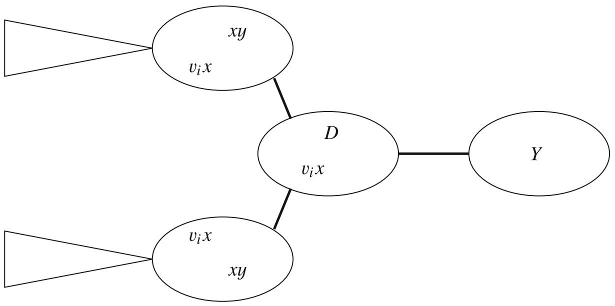

Fig. 1 Construct a tree decomposition $(T_i, \mathcal{X}_i)$ of H_i based on a tree decomposition $(T_{i-1}, \mathcal{X}_{i-1})$: add a vertex $v_i x$ to each bag containing xy in $(T_{i-1}, \mathcal{X}_{i-1})$; and add a new bag Y adjacent to the bag containing $D = \mathrm{RC}_{i-1}(x, y)$ (Note that $xy \in D$ and Y is the neighborhood of $v_i y$ in H_i)

From Lemma 5, we have that a proper coloring φ of H can be found in linear time and the needed number of colors is the clique number of H, which is exactly $\lambda^+(G)$. So φ is an optimal rd-coloring of G. From the for loop and Theorem 1, we see that the algorithm runs in $O(n^2)$ time.

Acknowledgements We are very grateful to the reviewers for their helpful comments and suggestions.

Author Contributions X.-Q. Bai, B. Li, C.-D. Xu and X. Zhang conceived or designed the work. X.-Q. Bai and B. Li drafted the article. C.-D. Xu and X. Zhang critically revised the article.

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

[1] Bondy, J.A., Murty, U.S.R.: Graph theory. In: Graduate Texts in Mathematics, vol. 244. Springer, Berlin (2008)
[2] Goedeking, P.: Networks in Aviation: Strategies and Structures. Springer Science & Business Media, London (2010)
[3] Peterson, L.L., Davie, B.S.: Computer Networks: a Systems Approach. Elsevier, Amsterdam (2007)
[4] Bell, M.G., Iida, Y.: Transportation Network Analysis. Wiley, New York (1997)
[5] Chartrand, G., Johns, G.L., McKeon, K.A., Zhang, P.: Rainbow connection in graphs. Math. Bohem. **133**, 85–98 (2008)
[6] Li, X., Shi, Y., Sun, Y.: Rainbow connections of graphs: a survey. Graphs Combin. **29**, 1–38 (2013)
[7] Li, X., Sun, Y.: An updated survey on rainbow connections of graphs—a dynamic survey. Theory Appl. Graphs **0**, 1–67 (2017)
[8] Li, X., Sun, Y.: Rainbow Connections of Graphs. Springer, New York (2012)
[9] Chartrand, G., Devereaux, S., Haynes, T.W., Hedetniemi, S.T., Zhang, P.: Rainbow disconnection in graphs. Discuss. Math. Graph Theory **38**, 1007–1021 (2018)
[10] Bai, X., Li, X.: Graph colorings under global structural conditions. arXiv:2008.07163 (2020)
[11] Bollobás, B.: On graphs with at most three independent paths connecting any two vertices. Studia Sci. Math. Hungar. **1**, 137–140 (1966)
[12] Bollobás, B.: Extremal Graph Theory. Academic Press Inc, London (1978)
[13] Elias, P., Feinstein, A., Shannon, C.E.: A note on the maximum flow through a network. IRE Trans. Inform. Theory IT **2**, 117–119 (1956)
[14] Ford, L.R., Jr., Fulkerson, D.R.: Maximal flow through a network. Canad. J. Math. **8**, 399–404 (1956)
[15] Bai, X., Chang, R., Huang, Z., Li, X.: More on rainbow disconnection in graphs. Discuss. Math. Graph Theory. **42**, 1185–1204 (2022)

[16] Bai, X., Huang, Z., Li, X.: Bounds for the rainbow disconnection number of graphs. Acta Math. Sin. Engl. Ser. **38**, 384–396 (2022)
[17] Gavril, F.: The intersection graphs of subtrees in trees are exactly the chordal graphs. J. Combin. Theory Ser. B **16**, 47–56 (1974)
[18] Heggernes, P.: Treewidth, partial *k*-trees, and chordal graphs. In: Partial Curriculum in INF334 Advanced Algorithmical Techniques. Department of Informatics, University of Bergen, Norway (2005)
[19] Wald, J.A., Colbourn, C.J.: Steiner trees, partial 2-trees, and minimum IFI networks. Networks **13**(2), 159–167 (1983)