Generalized Rainbow Connectivity of Graphs*

Kei Uchizawa[1], Takanori Aoki[2], Takehiro Ito[2], and Xiao Zhou[2]

[1] Graduate School of Science and Engineering, Yamagata University, Jonan 4-3-16, Yonezawa-shi, Yamagata 992-8510, Japan
uchizawa@yz.yamagata-u.ac.jp
[2] Graduate School of Information Sciences, Tohoku University, Aoba-yama 6-6-05, Sendai, Miyagi 980-8579, Japan
{takanori,takehiro,zhou}@ecei.tohoku.ac.jp

Abstract. Let $C = \{c_1, c_2, \ldots, c_k\}$ be a set of k colors, and let $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_k)$ be a k-tuple of nonnegative integers $\ell_1, \ell_2, \ldots, \ell_k$. For a graph $G = (V, E)$, let $f : E \to C$ be an edge-coloring of G in which two adjacent edges may have the same color. Then, the graph G edge-colored by f is $\boldsymbol{\ell}$-rainbow connected if every two vertices of G have a path P such that the number of edges in P that are colored with c_j is at most ℓ_j for each index $j \in \{1, 2, \ldots, k\}$. Given a k-tuple $\boldsymbol{\ell}$ and an edge-colored graph, we study the problem of determining whether the edge-colored graph is $\boldsymbol{\ell}$-rainbow connected. In this paper, we characterize the computational complexity of the problem with regards to certain graph classes: the problem is NP-complete even for cacti, while is solvable in polynomial time for trees. We then give an FPT algorithm for general graphs when parameterized by both k and $\ell_{\max} = \max\{\ell_j \mid 1 \leq j \leq k\}$.

1 Introduction

Graph connectivity is one of the most fundamental graph-theoretic properties. In the literature, several measures for graph connectivity have been studied, such as requiring hamiltonicity, edge-disjoint spanning trees, or edge- or vertex-cuts of sufficiently large size. Recently, there has been some interest in studying problems on colored graphs, due to their applications in areas such as computational biology, transportation and telecommunications [9]. In this paper, we generalize an interesting concept of graph connectivity on colored graphs, called the *rainbow connectivity*, which was introduced by Chartrand *et al.* [6] and has been extensively studied in the literature [2, 4–8, 11, 12].

Let $G = (V, E)$ be a graph with vertex set V and edge set E; we often denote by $V(G)$ the vertex set of G and by $E(G)$ the edge set of G. Let $C = \{c_1, c_2, \ldots, c_k\}$ be a set of k colors, and let $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_k)$ be a k-tuple of nonnegative integers $\ell_1, \ell_2, \ldots, \ell_k$. Consider a mapping $f : E \to C$, called an *edge-coloring* of G. Note that f is not necessarily a proper edge-coloring, that is, f may assign a same color to two adjacent edges. We denote by $G(f)$ the graph G

* This work is partially supported by JSPS KAKENHI Grant Numbers 22700001, 23500001 and 23700003.

S.K. Ghosh and T. Tokuyama (Eds.): WALCOM 2013, LNCS 7748, pp. 233–244, 2013.

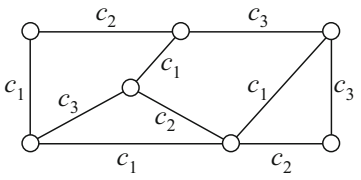

Fig. 1. An $\boldsymbol{\ell}$-rainbow connected graph, where $\boldsymbol{\ell} = (1, 3, 2)$

edge-colored by f. Then, a path P in $G(f)$ connecting two vertices u and v in V is called an *$\boldsymbol{\ell}$-rainbow path* between u and v if the number of edges in P that are colored with c_j is at most ℓ_j for every index $j \in \{1, 2, \ldots, k\}$. The edge-colored graph $G(f)$ is *$\boldsymbol{\ell}$-rainbow connected* if $G(f)$ has an $\boldsymbol{\ell}$-rainbow path between every two vertices in V. Note that these $\boldsymbol{\ell}$-rainbow paths are not necessarily edge-disjoint for pairs of vertices. For example, the edge-colored graph $G(f)$ in Fig. 1 is $\boldsymbol{\ell}$-rainbow connected for $\boldsymbol{\ell} = (1, 3, 2)$.

The concept of $\boldsymbol{\ell}$-rainbow connectivity was originally introduced by Chartrand *et al.* [6] for the special case where $\boldsymbol{\ell} = (1, 1, \ldots, 1)$. Chakraborty *et al.* [4] defined the RAINBOW CONNECTIVITY problem which asks whether a given edge-colored graph is $(1, 1, \ldots, 1)$-rainbow connected, and showed that the problem is NP-complete in general. Then, Uchizawa *et al.* [12] characterized the computational complexity of the problem with regards to certain graph classes, and also settled it with regards to graph diameters. (Remember that the *diameter* of a graph G is the maximum number of edges in a shortest path between any two vertices in G.)

In this paper, we introduce and study the generalized problem, defined as follows: Given a k-tuple $\boldsymbol{\ell}$ and an edge-coloring f of a graph G, the GENERALIZED RAINBOW CONNECTIVITY problem is to determine whether $G(f)$ is $\boldsymbol{\ell}$-rainbow connected. Thus, (ordinary) RAINBOW CONNECTIVITY is a specialization of GENERALIZED RAINBOW CONNECTIVITY. We first give precise complexity analyses for GENERALIZED RAINBOW CONNECTIVITY with regards to certain graph classes. We then give an FPT algorithm for the problem on general graphs when parameterized by both $k = |C|$ and $\ell_{\max} = \max\{\ell_j \mid 1 \leq j \leq k\}$. Below we explain our results more precisely, together with comparisons with known results [12].

[Graph classes]
From the viewpoint of graph classes, we clarify a boundary on graph classes between tractability and NP-completeness: GENERALIZED RAINBOW CONNECTIVITY is NP-complete even for cacti, while there is a polynomial-time algorithm for trees. Note that trees and cacti are very close to each other in the following sense: trees form a graph class which is a subclass of cacti, and the treewidth of cacti is two [3]. It is remarkable that the boundary is different from the known one for RAINBOW CONNECTIVITY [12]: it is NP-complete for outerplanar graphs, and is solvable in polynomial time for cacti. Therefore, the NP-complete proof given by [12] does not imply our result. We also remark that our polynomial-time algorithm for trees is always faster than a naive one, as discussed in Section 3.1.

[FPT algorithm]
In Section 3.2, we give an algorithm which solves GENERALIZED RAINBOW CONNECTIVITY for general graphs in time $O(k2^{k\ell_{\max}}mn)$ using $O\big(kn2^{k\ell_{\max}}\log(\ell_{\max}+1)\big)$ space, where n and m are the numbers of vertices and edges in a graph, respectively. Therefore, the problem can be solved in polynomial time for the following two cases: (a) $k = O(\log n)$ and $\ell_{\max}$ is a fixed constant; and (b) k is a fixed constant and $\ell_{\max} = O(\log n)$. We remark that our FPT algorithm generalizes the known one [12]: the same running time and space complexity of the known FPT algorithm for RAINBOW CONNECTIVITY [12] can be obtained from our result as the special case where $\ell_{\max} = 1$.

2 NP-Completeness for Cacti

A graph G is a *cactus* if every edge is part of at most one cycle in G [3]. (See Fig. 2 as an example of cacti.) The main result of this section is the following theorem.

Theorem 1. GENERALIZED RAINBOW CONNECTIVITY *is* NP*-complete even for cacti and* $\boldsymbol{\ell} = (2, 2, \ldots, 2)$.

Let $G(f)$ be a given edge-colored graph. We can clearly cheek in polynomial time whether a given path in $G(f)$ is an $\boldsymbol{\ell}$-rainbow path, and hence GENERALIZED RAINBOW CONNECTIVITY belongs to NP. We below show that the problem is NP-hard even for cacti and $\boldsymbol{\ell} = (2, 2, \ldots, 2)$ by a reduction from the 3-OCCURRENCE 3SAT problem defined as follows: Given a 3CNF formula ϕ such that each variable appears at most three times in ϕ, determine whether ϕ is satisfiable. 3-OCCURRENCE 3SAT is known to be NP-complete [10].

Suppose that the formula ϕ consists of n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$. Without loss of generality, we can assume that any literal of a variable x_i, $1 \le i \le n$, appears at most twice in ϕ; otherwise, ϕ contains only positive (or negative) literals of x_i, and hence we can safely fix its assignment. In what follows, we construct a cactus G_ϕ, an edge-coloring f_ϕ of G_ϕ and a k-tuple $\boldsymbol{\ell} = (2, 2, \ldots, 2)$, as a corresponding instance. We then prove that ϕ is satisfiable if and only if the edge-colored graph $G_\phi(f_\phi)$ is $\boldsymbol{\ell}$-rainbow connected.

[Graph G_ϕ]
We first construct a gadget G_j for each clause C_j, $1 \le j \le m$, as follows: G_j is a cycle consisting of four vertices p_j, u_j, p'_j, u'_j embedded in clockwise order

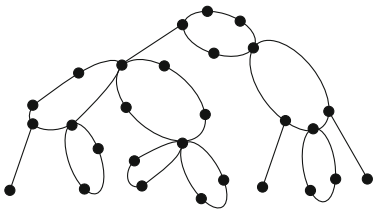

Fig. 2. Cactus G

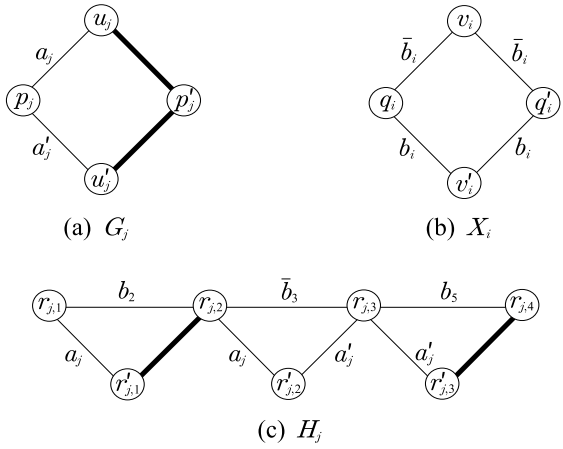

Fig. 3. (a) Gadget G_j for a clause C_j, (b) gadget X_i for a variable x_i, and (c) gadget H_j for the clause $C_j = (x_2 \vee \overline{x_3} \vee x_5)$

on the plane. (See Fig. 3(a).) We then construct a gadget X_i for each variable x_i, $1 \leq i \leq n$, as follows: X_i is a cycle consisting of four vertices q_i, v_i, q'_i, v'_i embedded in clockwise order on the plane. (See Fig. 3(b).) We lastly construct a gadget H_j for each clause C_j, $1 \leq j \leq m$, as follows: (i) make a path of four vertices $r_{j,1}$, $r_{j,2}$, $r_{j,3}$, $r_{j,4}$; (ii) add three vertices $r'_{j,1}$, $r'_{j,2}$, $r'_{j,3}$; and (iii) connect $r'_{j,1}$ to both $r_{j,1}$ and $r_{j,2}$, connect $r'_{j,2}$ to both $r_{j,2}$ and $r_{j,3}$, and connect $r'_{j,3}$ to both $r_{j,3}$ and $r_{j,4}$. (See Fig. 3(c).)

Using $G_1, G_2, \ldots, G_m$, $X_1, X_2, \ldots, X_n$ and $H_1, H_2, \ldots, H_m$ given above, we construct the corresponding graph G_ϕ as follows. (See Fig. 4.) We connect p'_j to p_{j+1} for every $j \in \{1, 2, \ldots, m-1\}$, and connect p'_m to q_1. We then connect q'_i to q_{i+1} for every $i \in \{1, 2, \ldots, n-1\}$, and connect q'_n to $r_{1,1}$. We complete the construction of G_ϕ by connecting $r_{j,4}$ to $r_{j+1,1}$ for every $j \in \{1, 2, \ldots, m-1\}$. Since each gadget consists of either a single cycle or consecutive three cycles, G_ϕ is clearly a cactus.

Before constructing the edge-coloring f_ϕ of G_ϕ, we define some terms. For each gadget G_j, $1 \leq j \leq m$, we call the path $p_j u_j p'_j$ the *j-th upper path*, and

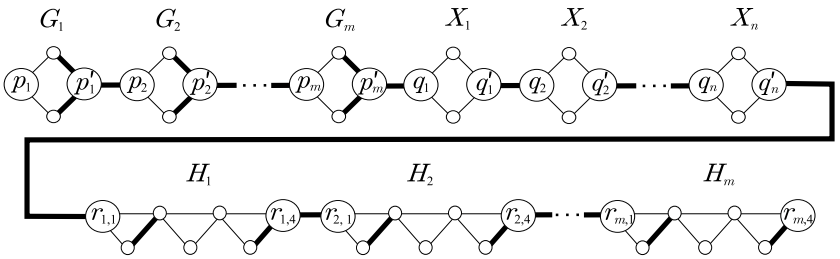

Fig. 4. Graph G_ϕ

call the path $p_j u'_j p'_j$ the *j-th lower path*. For each gadget X_i, $1 \leq i \leq n$, we call the path $q_i v_i q'_i$ the *i-th positive path*, and call the path $q_i v'_i q'_i$ the *i-th negative path*; the i-th positive path corresponds to $x_i = 1$, and the i-th negative path corresponds to $x_i = 0$. Let $\phi = \wedge_{j=1}^{m}(l_{j,1} \vee l_{j,2} \vee l_{j,3})$ be the given formula, where $l_{j,1}$, $l_{j,2}$ and $l_{j,3}$ are literals of $x_1, x_2, \ldots, x_n$ contained in the clause C_j.

[Edge-coloring f_ϕ of G_ϕ]

We construct f_ϕ as in the following four steps (i)–(iv).

(i) Let $a_1, a_2, \ldots, a_m$ and $a'_1, a'_2, \ldots, a'_m$ be $2m$ distinct colors. For each $j \in \{1, 2, \ldots, m\}$, we assign a_j to (p_j, u_j), and a'_j to (p_j, u'_j). (See Fig. 3(a).)

(ii) Let $b_1, b_2, \ldots, b_n$ and $\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n$ be $2n$ distinct (new) colors. For each $i \in \{1, 2, \ldots, n\}$, we assign $\bar{b}_i$ to both (q_i, v_i) and (v_i, q'_i), and b_i to both (q_i, v'_i) and (v'_i, q'_i). (See Fig. 3(b).)

(iii) For each clause $C_j = l_{j,1} \vee l_{j,2} \vee l_{j,3}$, $1 \leq j \leq m$, we assign some of $b_1, b_2, \ldots, b_n$ and $\bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n$ to the edges $(r_{j,1}, r_{j,2}), (r_{j,2}, r_{j,3})$ and $(r_{j,3}, r_{j,4})$ in the gadget H_j, as follows: For each $k \in \{1, 2, 3\}$, we assign b_i to $(r_{j,k}, r_{j,k+1})$ if the k-th literal $l_{j,k}$ is a positive literal of x_i; and assign $\bar{b}_i$ to $(r_{j,k}, r_{j,k+1})$ if the k-th literal $l_{j,k}$ is a negative literal of x_i. Moreover, we assign a_j to both $(r_{j,1}, r'_{j,1})$ and $(r_{j,2}, r'_{j,2})$, and assign a'_j to both $(r'_{j,2}, r_{j,3})$ and $(r_{j,3}, r'_{j,3})$. (See Fig. 3(c).)

(iv) Let U be the set of the edges that are not assigned colors in (i)–(iii) above. We assign a new distinct color for each edge in U. (See Figs. 3 and 4, where the edges in U are depicted by thick lines.)

Remember that any literal of a variable x_i, $1 \leq i \leq n$, appears at most twice in ϕ. Therefore, in the step (iii), each of the colors $b_1, b_2, \ldots, b_n, \bar{b}_1, \bar{b}_2, \ldots, \bar{b}_n$ is assigned to at most two edges in $H_1, H_2, \ldots, H_m$.

We finally set $\boldsymbol{\ell} = (2, 2, \ldots, 2)$, and complete the construction of the corresponding instance.

The following two lemmas for $G_\phi(f_\phi)$ clearly imply Theorem 1.

Lemma 1. *$G_\phi(f_\phi)$ is $\boldsymbol{\ell}$-rainbow connected if and only if $G_\phi(f_\phi)$ has an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$.*

Lemma 2. *$G_\phi(f_\phi)$ has an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$ if and only if ϕ is satisfiable.*

In the rest of the section, we prove Lemmas 1 and 2.

[Proof of Lemma 1]

It is trivially true that, if $G_\phi(f_\phi)$ is $\boldsymbol{\ell}$-rainbow connected, then $G_\phi(f_\phi)$ has an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$. Below we prove that $G_\phi(f_\phi)$ is $\boldsymbol{\ell}$-rainbow connected if $G_\phi(f_\phi)$ has an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$.

Let s and t be an arbitrary pair of vertices in G_ϕ. We consider a partition of the vertex set $V(G_\phi)$ into the following three groups: $V^1 = \bigcup_{j=1}^{m} V(G_j)$, $V^2 = \bigcup_{i=1}^{n} V(X_i)$ and $V^3 = \bigcup_{j=1}^{m} V(H_j)$. In any subgraph induced by only one of V^1, V^2 and V^3, every color is assigned to at most two edges in the subgraph. Similarly, in the subgraph induced by V^1 and V^2, every color is assigned to at

most two edges; in the subgraph induced by V^2 and V^3, when we remove the edges $(r_{j,1}, r_{j,2}), (r_{j,2}, r_{j,3}), (r_{j,3}, r_{j,4})$ for every $j \in \{1, 2, \ldots, m\}$, every color is assigned to at most two edges. Thus, it suffices to verify the case where $s \in V^1$ and $t \in V^3$. Let P be the $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$ in $G_\phi(f_\phi)$, and let j_1 and j_2 be the indices satisfying $s \in V(G_{j_1})$ and $t \in V(H_{j_2})$. Then, we can easily construct an $\boldsymbol{\ell}$-rainbow path P' between s and t, as follows: P' consists of a subpath of P between p_{j_1} and $r_{j_2,4}$ together with some of the five edges $(u_{j_1}, p'_{j_1}), (u'_{j_1}, p'_{j_1}), (r'_{j_2,1}, r_{j_2,2}), (r'_{j_2,2}, r_{j_2,3}), (r'_{j_2,3}, r_{j_2,4})$. □

[**Proof of Lemma 2**]
Necessity: We prove that, if $G_\phi(f_\phi)$ has an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$, then ϕ is satisfiable. Let P be an $\boldsymbol{\ell}$-rainbow path in $G_\phi(f_\phi)$ between p_1 and $r_{m,4}$. For each gadget G_j, $1 \le j \le m$, we denote by $P \cap G_j$ the graph (path) induced by $E(P) \cap E(G_j)$. Then, each subpath $P \cap G_j$, $1 \le j \le m$, is either j-th upper path or j-th lower path. Similarly, for each gadget X_i, $1 \le i \le n$, we denote by $P \cap X_i$ the graph (path) induced by $E(P) \cap E(X_i)$. Then, each subpath $P \cap X_i$, $1 \le i \le n$, is either i-th positive path or i-th negative path. Consider the following truth assignment $\boldsymbol{z} = (z_1, z_2, \ldots, z_n) \in \{0,1\}^n$: for each index $i \in \{1, 2, \ldots, n\}$,

$$z_i = \begin{cases} 1 & \text{if } P \cap X_i \text{ is the } i\text{-th positive path;} \\ 0 & \text{if } P \cap X_i \text{ is the } i\text{-th negative path.} \end{cases} \tag{1}$$

We now show that $\boldsymbol{z}$ is a satisfying truth assignment for ϕ. Clearly, any $\boldsymbol{\ell}$-rainbow path must go through either (p_j, u_j) with the color a_j or (p_j, u'_j) with the color a'_j for each $j \in \{1, 2, \ldots, m\}$. Then, since $\boldsymbol{\ell} = (2, 2, \ldots, 2)$, P must pass through at least one of the edges $(r_{j,1}, r_{j,2})$, $(r_{j,2}, r_{j,3})$ and $(r_{j,3}, r_{j,4})$ in each clause gadget H_j, $1 \le j \le m$. Let $(r_{j,k}, r_{j,k+1})$ be such an edge. We show that the literal $l_{j,k}$ corresponding to the edge $(r_{j,k}, r_{j,k+1})$ is true by $\boldsymbol{z}$, and hence the clause C_j is satisfied; consequently, $\boldsymbol{z}$ is satisfying, as required. Consider the case where the edge $(r_{j,k}, r_{j,k+1})$ receives the color $\bar{b}_i$ for some i, $1 \le i \le n$. (The proof is similar for the other case where $(r_{j,k}, r_{j,k+1})$ receives the color b_i.) Then, by the construction of f_ϕ, the literal $l_{j,k}$ corresponding to $(r_{j,k}, r_{j,k+1})$ is a negative literal of the variable x_i. Since the color $\bar{b}_i$ is assigned to each of the two edges in the i-th positive path in X_i, P must go through the i-th negative path in X_i. Consequently, by Eq. (1), we have $z_i = 0$ in $\boldsymbol{z}$, and hence the literal $l_{j,k}$ is true by $\boldsymbol{z}$.

Sufficiency: We prove that $G_\phi(f_\phi)$ has an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$ if ϕ is satisfiable. Let $\boldsymbol{z} = (z_1, z_2, \ldots, z_n)$ be a satisfying truth assignment for ϕ. For each $j \in \{1, 2, \ldots, m\}$, we denote by l_{j,k_j}, $1 \le k_j \le 3$, a literal satisfied by $\boldsymbol{z}$ in C_j.

Consider the following path P^X from q_1 to q'_n: For each gadget X_i, $1 \le i \le n$, take the i-th positive path if $z_i = 1$, and otherwise take the i-th negative path. Clearly, for each $i \in \{1, 2, \ldots, n\}$, both of the edges in $P^X \cap E(X_i)$ receive $\bar{b}_i$ if $z_i = 1$, and receive b_i if $z_i = 0$. Consider then the following path P_j^H from $r_{j,1}$ to $r_{j,4}$ for each $j \in \{1, 2, \ldots, m\}$: make a path consisting of (r_{j,k_j}, r_{j,k_j+1}) together

with the four edges $(r_{j,\alpha}, r'_{j,\alpha})$, $(r'_{j,\alpha}, r_{j,\alpha+1})$, $(r_{j,\beta}, r'_{j,\beta})$ and $(r'_{j,\beta}, r_{j,\beta+1})$, where $\alpha, \beta \in \{1,2,3\}\backslash\{k_j\}$ and $\alpha < \beta$. We obtain the path P from q_1 to $r_{m,4}$ by connecting P^X and $P_1^H, P_2^H, \ldots, P_m^H$ in this order. Clearly, every color is assigned to at most two edges in P. Moreover, for each $j \in \{1,2,\ldots,m\}$, one of a_j and a'_j is assigned to only one edge in P; such a color is said to be *available*. Then, we can obtain a path P^G from p_1 to p'_m such that, for each $j \in \{1,2,\ldots,m\}$, it takes the j-th upper path if a_j is available, and otherwise takes the j-th lower path. Finally, we can obtain an $\boldsymbol{\ell}$-rainbow path between p_1 and $r_{m,4}$ by connecting the paths P^G and P. □

3 Algorithms

As we have shown in Theorem 1, GENERALIZED RAINBOW CONNECTIVITY is NP-complete even for cacti and hence it cannot be solved in polynomial time unless P = NP. However, we give two algorithms in this section: in Section 3.1, we give an efficient polynomial-time algorithm for trees; in Section 3.2, we give an FPT algorithm for general graphs when parameterized by both k and $\ell_{\max}$.

3.1 Polynomial-Time Algorithm for Trees

The main result of this subsection is the following theorem.

Theorem 2. GENERALIZED RAINBOW CONNECTIVITY *can be solved for a tree T in time $O(kn)$, where $k = |C|$ and n is the number of vertices in T.*

In the remainder of this subsection, we give an $O(kn)$-time algorithm as a proof of Theorem 2. It is obvious that the problem is in P for trees, because a naive $O(n^3)$-time algorithm exists: for each pair of vertices in a tree, it determines whether the unique path between the pair is an $\boldsymbol{\ell}$-rainbow path. We remark that our $O(kn)$-time algorithm is always faster than the naive one; our algorithm runs in linear time if k is a constant, and in time $O(n^2)$ even if $k = O(n)$; notice that $k \leq n-1$.

[Terms and ideas]

Let $T = (V, E)$ be a given tree. One may assume without loss of generality that T is a rooted tree with an arbitrarily chosen root r. Let u be a vertex of T, and we denote by $d(u)$ the number of children of u. For each $i \in \{1,2,\ldots,d(u)\}$, let u_i be a child of u ordered arbitrarily, and let e_i be the edge joining u and u_i, as illustrated in Fig. 5. Let T_{u_i} be the subtree of T which is rooted at u_i and is induced by all descendants of u_i in T. We denote by T_u^i the subtree of T which consists of the vertex u, the edges $e_1, e_2, \ldots, e_i$ and the subtrees $T_{u_1}, T_{u_2}, \ldots, T_{u_i}$. In Fig. 5, T_u^i is indicated by a dotted closed curve. Clearly $T_u = T_u^{d(u)}$. For the sake of notational convenience, we denote by T_u^0 the tree consisting of a single vertex u.

Let $C = \{c_1, c_2, \ldots, c_k\}$ be the color set, and let $f : E \to C$ be a given edge-coloring of T. Note that any path P in T must be an $\boldsymbol{\ell}$-rainbow path; otherwise

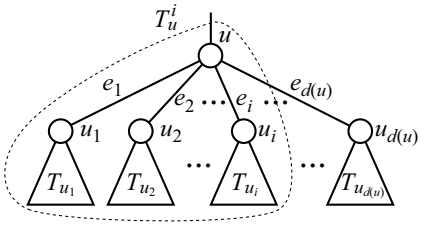

Fig. 5. Tree T_u

there is no $\boldsymbol{\ell}$-rainbow path between the end-vertices of P. For a pair of vertices $v, w \in V(T)$ and an index $j \in \{1, 2, \ldots, k\}$, we denote by $t_j(v, w)$ the number of edges in the (unique) path between v and w that are colored with c_j by f.

Consider the subtree T_u^i for a vertex u of T and an integer $i \in \{0, 1, \ldots, d(u)\}$. We classify the paths in T_u^i into two subclasses, and check whether every path in T_u^i is an $\boldsymbol{\ell}$-rainbow path. For an index $j \in \{1, 2, \ldots, k\}$, we define $a_j(T_u^i)$ as follows:

$$a_j(T_u^i) = \max\{t_j(u, w) \mid w \in V(T_u^i)\}.$$

Therefore, by the values $a_j(T_u^i)$, $1 \leq j \leq k$, we can check whether all paths between the root u of T_u^i and vertices of T_u^i are $\boldsymbol{\ell}$-rainbow paths; more specifically, such paths are all $\boldsymbol{\ell}$-rainbow paths if $a_j(T_u^i) \leq \ell_j$ for all indices $j \in \{1, 2, \ldots, k\}$. Similarly, for an index $j \in \{1, 2, \ldots, k\}$, we define $b_j(T_u^i)$ as follows:

$$b_j(T_u^i) = \max\{t_j(v, w) \mid v, w \in V(T_u^i)\}.$$

Then, by the values $b_j(T_u^i)$, $1 \leq j \leq k$, we can check whether all paths that do not necessarily contain u are $\boldsymbol{\ell}$-rainbow paths; indeed, T_u^i is $\boldsymbol{\ell}$-rainbow connected if and only if $b_j(T_u^i) \leq \ell_j$ for all indices $j \in \{1, 2, \ldots, k\}$.

Our algorithm computes $a_j(T_u^i)$ and $b_j(T_u^i)$ for each vertex u of T and all indices $i \in \{0, 1, \ldots, d(u)\}$ and $j \in \{1, 2, \ldots, k\}$ from the leaves to the root r of T by means of dynamic programming. Then, the edge-colored tree $T(f) = T_r(f)$ is $\boldsymbol{\ell}$-rainbow connected if and only if $b_j(T_r) \leq \ell_j$ for all indices $j \in \{1, 2, \ldots, k\}$.

[Algorithm]

We first compute $a_j(T_u^0)$ and $b_j(T_u^0)$ for each vertex u of T and all indices $j \in \{1, 2, \ldots, k\}$. Since T_u^0 consists of a single vertex u, there is no edge in T_u^0. Therefore, we clearly have $a_j(T_u^0) = 0$ and $b_j(T_u^0) = 0$.

We then compute $a_j(T_u^i)$ and $b_j(T_u^i)$, $1 \leq i \leq d(u)$, for each internal vertex u of T from the counterparts of T_u^{i-1} and T_{u_i}. (See Fig. 5.) Remember that $T_u = T_u^{d(u)}$, and that T_u^i is obtained from T_u^{i-1} and T_{u_i} by joining u and u_i. For an edge e in T, let

$$h_j(e) = \begin{cases} 1 & \text{if } f(e) = c_j; \\ 0 & \text{otherwise.} \end{cases}$$

We first compute $a_j(T_u^i)$, that is, check whether all paths between the root u of T_u^i and vertices of T_u^i are $\boldsymbol{\ell}$-rainbow paths. Consider an arbitrary path P between u and some vertex v of T_u^i. Then, there are the following two cases:

(i) v is in T_u^{i-1}, and hence P is a path in T_u^{i-1}; and
(ii) v is in T_{u_i}, and hence P consists of e_i and the path between u_i and v.

Therefore, the value $a_j(T_u^i)$ can be computed as follows:

$$a_j(T_u^i) = \max\{a_j(T_u^{i-1}),\ a_j(T_{u_i}) + h_j(e_i)\}. \tag{2}$$

We then compute $b_j(T_u^i)$, that is, check whether all paths that do not necessarily contain u are $\boldsymbol{\ell}$-rainbow paths. Consider an arbitrary path P between two vertices v and w in T_u^i. Then, there are the following three cases:

(i) both v and w are in T_u^{i-1}, and hence P is a path in T_u^{i-1};
(ii) both v and w are in T_{u_i}, and hence P is a path in T_{u_i}; and
(iii) one of v and w is in T_u^{i-1} and the other is in T_{u_i}, and hence P is a path starting from v, passing through u and e_i, and ending with w.

Therefore, the value $b_j(T_u^i)$ can be computed as follows:

$$b_j(T_u^i) = \max\{b_j(T_u^{i-1}),\ b_j(T_{u_i}),\ a_j(T_u^{i-1}) + h_j(e_i) + a_j(T_{u_i})\}. \tag{3}$$

[Proof of Theorem 2]

We now show that our algorithm above runs in time $O(kn)$.

For each vertex u of T and all indices $j \in \{1, 2, \ldots, k\}$, we can compute $a_j(T_u^0)$ and $b_j(T_u^0)$ in time $O(k)$. Therefore, the initialization can be done in time $O(kn)$ for all vertices in T and all indices $j \in \{1, 2, \ldots, k\}$.

For an internal vertex u of T and all indices $j \in \{1, 2, \ldots, k\}$, as described in Eqs. (2) and (3), we can compute $a_j(T_u^i)$ and $b_j(T_u^i)$, $i \geq 1$, from the counterparts of T_u^{i-1} and T_{u_i} in time $O(k)$. Since there are $n-1$ edges in T, the computation occurs $n-1$ times for each of $a_j(T_u^i)$ and $b_j(T_u^i)$. Therefore, for the root r of T, we can compute the values $a_j(T_r)$ and $b_j(T_r)$ for all indices $j \in \{1, 2, \ldots, k\}$ in time $O(kn)$.

Then, from the values $b_j(T_r)$, $1 \leq j \leq k$, we can determine in time $O(k)$ whether the edge-colored tree $T(f)$ is $\boldsymbol{\ell}$-rainbow connected.

In this way, our algorithm solves GENERALIZED RAINBOW CONNECTIVITY for a tree in time $O(kn)$ in total. This completes the proof of Theorem 2.

3.2 FPT Algorithm for General Graphs

In this subsection, we give an FPT algorithm for GENERALIZED RAINBOW CONNECTIVITY when parameterized by both k and $\ell_{\max}$.

Theorem 3. *For a k-tuple $\boldsymbol{\ell}$ and an edge-coloring f of a graph G, one can determine whether the edge-colored graph $G(f)$ is $\boldsymbol{\ell}$-rainbow connected in time $O(k2^{k\ell_{\max}}mn)$ using $O\big(kn2^{k\ell_{\max}}\log(\ell_{\max}+1)\big)$ space, where n and m are the numbers of vertices and edges in G, respectively.*

As a proof of Theorem 3, we give an algorithm to determine whether $G(f)$ has $\boldsymbol{\ell}$-rainbow paths from a vertex s to all the other vertices. It suffices to design such an algorithm which runs in time $O(k2^{k\ell_{\max}}m)$ using $O\big(kn2^{k\ell_{\max}}\log(\ell_{\max}+1)\big)$ space. Then, Theorem 3 clearly holds.

[Terms and ideas]

We first introduce some terms. For a vertex v of a graph $G = (V, E)$, we denote by $N(v)$ the set of all neighbors of v (which does not include v itself), that is, $N(v) = \{w \in V \mid (v, w) \in E\}$. We remind the reader that a *walk* in a graph is a sequence of adjacent vertices and edges, each of which may appear more than once; while a *path* is a walk in which each vertex appears exactly once. The *length* of a walk is defined as the number of edges in the walk. A walk W in $G(f)$ is called an *$\boldsymbol{\ell}$-rainbow walk* if the number of edges in W that are colored with c_j is at most ℓ_j for every index $j \in \{1, 2, \ldots, k\}$.

We extend ideas in [1, 12]. For a graph $G = (V, E)$ and a color set C with $|C| = k$, let $f : E \to C$ be a given edge-coloring of G. We choose an arbitrary vertex $s \in V$. We indeed give an algorithm which determines whether the edge-colored graph $G(f)$ has an $\boldsymbol{\ell}$-rainbow *walk* W from s to each vertex $v \in V \setminus \{s\}$; one can obtain an $\boldsymbol{\ell}$-rainbow path between s and v, as the sub-walk of W. Since $|C| = k$ and $\ell_{\max} = \max\{\ell_j \mid 1 \leq j \leq k\}$, every $\boldsymbol{\ell}$-rainbow walk is of length at most $k\ell_{\max}$. Therefore, our algorithm is based on a dynamic programming approach with respect to the lengths of walks from s: $G(f)$ has an $\boldsymbol{\ell}$-rainbow walk from s to a vertex v with length exactly i if and only if there exists at least one vertex w in $N(v)$ such that $G(f)$ has an $\boldsymbol{\ell}$-rainbow walk from s to w with length exactly $i - 1$ in which the color $c_{j'} = f((w, v))$ is assigned to at most $(\ell_{j'} - 1)$ edges.

Based on the idea above, for an integer $i \in \{1, 2, \ldots, k\ell_{\max}\}$ and a vertex $v \in V$, we define a set $\Gamma_s(i, v)$ of k-tuples $\boldsymbol{\ell}' = (\ell'_1, \ell'_2, \ldots, \ell'_k)$ of nonnegative integers $\ell'_1, \ell'_2, \ldots, \ell'_k$, as follows:

$$\Gamma_s(i, v) = \{(\ell'_1, \ell'_2, \ldots, \ell'_k) \mid G(f) \text{ has an } \boldsymbol{\ell}\text{-rainbow walk } W \text{ between } s \text{ and } v \text{ such that } \ell'_1 + \ell'_2 + \cdots + \ell'_k = i \text{ and } c_j \text{ is assigned to exactly } \ell'_j \text{ edges in } W \text{ for all } j \in \{1, 2, \ldots, k\}\}.$$

Note that $\Gamma_s(i, v) = \emptyset$ if $G(f)$ has no $\boldsymbol{\ell}$-rainbow walk between s and v of length exactly i. Clearly, $G(f)$ has an $\boldsymbol{\ell}$-rainbow path from s to a vertex $v \in V \setminus \{s\}$ if and only if $\Gamma_s(i, v) \neq \emptyset$ for some integer $i \in \{1, 2, \ldots, k\ell_{\max}\}$. By a dynamic programming approach, we compute the sets $\Gamma_s(i, v)$ from $i = 1$ to $k\ell_{\max}$ for all vertices $v \in V$. Then, using the sets $\Gamma_s(i, v)$, it can be determined in time $O(k\ell_{\max} n)$ whether $G(f)$ has $\boldsymbol{\ell}$-rainbow paths from s to all vertices $v \in V \setminus \{s\}$.

[Algorithm]

We first compute the set $\Gamma_s(1, v)$ for each vertex $v \in V$. Clearly, the walks with length exactly 1 from s are only the edges (s, v) for the vertices v in $N(s)$. Therefore, if $v \in N(s)$ and $\ell_{j'} \geq 1$ where $c_{j'} = f((s, v))$, then we have

$$\Gamma_s(1, v) = \{(\ell'_1, \ell'_2, \ldots, \ell'_k)\}, \tag{4}$$

where $\ell'_{j'} = 1$ and $\ell'_j = 0$ for the other indices $j \in \{1, 2, \ldots, k\} \setminus \{j'\}$; otherwise

$$\Gamma_s(1, v) = \emptyset. \tag{5}$$

We then compute the set $\Gamma_s(i,v)$ for an integer $i \geq 2$ and each vertex $v \in V$. Suppose that we have already computed $\Gamma_s(i-1,w)$ for all vertices $w \in V$. Obviously, $G(f)$ has an $\boldsymbol{\ell}$-rainbow walk from s to a vertex v with length exactly i if and only if, for some vertex $w \in N(v)$, there exists a k-tuple $(\ell'_1, \ell'_2, \ldots, \ell'_k) \in \Gamma_s(i-1,w)$ such that $\ell'_{j'} \leq \ell_{j'} - 1$ for the color $c_{j'} = f((w,v))$. Therefore, we can compute $\Gamma_s(i,v)$ for a vertex $v \in V$, as follows:

$$\Gamma_s(i,v) = \Big\{(\ell'_1, \ldots, \ell'_{j'-1}, \ell'_{j'}+1, \ell'_{j'+1}, \ldots, \ell'_k) \mid w \in N(v),\ (\ell'_1, \ell'_2, \ldots, \ell'_k) \in \Gamma_s(i-1,w),\ \text{and } \ell'_{j'} \leq \ell_{j'} - 1 \text{ for the color } c_{j'} = f((w,v))\Big\}. \quad (6)$$

[Proof of Theorem 3]

Using Eqs. (4)–(6) one can correctly compute $\Gamma_s(i,v)$, $1 \leq i \leq k\ell_{\max}$, for all vertices $v \in V$. Thus, we now show that our algorithm runs in time $O(k2^{k\ell_{\max}}m)$ and uses $O\big(kn2^{k\ell_{\max}} \log(\ell_{\max}+1)\big)$ space.

We first claim that $|\Gamma_s(i,v)| \leq \binom{k\ell_{\max}}{i}$ for a vertex v in T and each integer $i \in \{1,2,\ldots,k\ell_{\max}\}$. Consider an arbitrary k-tuple $(\ell'_1, \ell'_2, \ldots, \ell'_k) \in \Gamma_s(i,v)$. Then, $0 \leq \ell'_j \leq \ell_j \leq \ell_{\max}$ holds for each index $j \in \{1,2,\ldots,k\}$, and $i = \ell'_1 + \ell'_2 + \cdots + \ell'_k \leq k\ell_{\max}$. Thus, the number of k-tuples in $\Gamma_s(i,v)$ can be bounded by the number of combinations which choose i elements from $k\ell_{\max}$ elements, and hence $|\Gamma_s(i,v)| \leq \binom{k\ell_{\max}}{i}$.

We now show that our algorithm uses $O\big(kn2^{k\ell_{\max}} \log(\ell_{\max}+1)\big)$ space. For a vertex v and an integer i, each k-tuple $\boldsymbol{\ell}' \in \Gamma_s(i,v)$ can be represented by $O(k \cdot \log(\ell_{\max}+1))$ bits, and hence the set $\Gamma_s(i,v)$ can be represented by using $O\big(\binom{k\ell_{\max}}{i} \cdot k \log(\ell_{\max}+1)\big)$ space. Therefore, we can represent the sets $\Gamma_s(i,v)$ for all vertices $v \in V$ and all integers $i \in \{1,2,\ldots,k\ell_{\max}\}$ using the space of

$$\sum_{i=1}^{k\ell_{\max}} \sum_{v \in V} O\left(k \cdot \binom{k\ell_{\max}}{i} \cdot \log(\ell_{\max}+1)\right) = O\big(kn2^{k\ell_{\max}} \log(\ell_{\max}+1)\big).$$

We finally estimate the running time of our algorithm. By Eqs. (4) and (5) the sets $\Gamma_s(1,v)$ can be computed in time $O(kn)$ for all vertices $v \in V$. By Eq. (6) the set $\Gamma_s(i,v)$ for a vertex v and an integer i can be computed in time $O\left(|N(v)| \cdot \binom{k\ell_{\max}}{i-1} \cdot k\right)$, because $|\Gamma_s(i-1,w)| \leq \binom{k\ell_{\max}}{i-1}$, the condition $\ell'_{j'} \leq \ell_{j'} - 1$ for the color $c_{j'} = f((w,v))$ can be checked in time $O(1)$, and $O(k)$ time is required to store the obtained k-tuple $(\ell'_1, \ldots, \ell'_{j'-1}, \ell'_{j'}+1, \ell'_{j'+1}, \ldots, \ell'_k)$ into $\Gamma_s(i,v)$. Therefore, the sets $\Gamma_s(i,v)$ can be computed for all vertices $v \in V$ and all integers $i \in \{2,3,\ldots,k\ell_{\max}\}$, in time

$$\sum_{i=2}^{k\ell_{\max}} \sum_{v \in V} O\left(k \cdot \binom{k\ell_{\max}}{i-1} \cdot |N(v)|\right) = O(k2^{k\ell_{\max}}m).$$

Using the sets $\Gamma_s(i,v)$, $1 \leq i \leq k\ell_{\max}$, it can be determined in time $O(k\ell_{\max}n)$ whether $G(f)$ has $\boldsymbol{\ell}$-rainbow paths from s to all vertices $v \in V \setminus \{s\}$. Since G

can be assumed to be a connected graph, $n - 1 \leq m$ and hence our algorithm takes time $O(k2^{k\ell_{\max}}m)$ in total.

This completes the proof of Theorem 3. □

4 Conclusion

In this paper, we introduced GENERALIZED RAINBOW CONNECTIVITY. We proved that the problem is NP-complete even for cacti, while is solvable in polynomial time for trees. We also gave an FPT algorithm for general graphs when parameterized by both k and $\ell_{\max} = \max\{\ell_j \mid 1 \leq j \leq k\}$.

References

1. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM 42, 844–856 (1996)
2. Ananth, P., Mande, M., Sarpatwar, K.: Rainbow connectivity: hardness and tractability. In: Proc. of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, pp. 241–251 (2011)
3. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: A Survey. Society for Industrial and Applied Mathematics, Philadelphia (1999)
4. Chakraborty, S., Fischer, E., Matsliah, A., Yuster, R.: Hardness and algorithms for rainbow connection. J. Combinatorial Optimization 21, 330–347 (2011)
5. Chandran, L.S., Das, A., Rajendraprasad, D., Varma, N.M.: Rainbow connection number and connected dominating sets. J. Graph Theory 71, 206–218 (2012)
6. Chartrand, G., Johns, G.L., McKeon, K.A., Zhang, P.: Rainbow connection in graphs. Mathematica Bohemica 133, 85–98 (2008)
7. Chartrand, C., Johns, G.L., McKeon, K.A., Zhang, P.: The rainbow connectivity of a graph. Networks 54, 75–81 (2009)
8. Caro, Y., Lev, A., Roditty, Y., Tuza, Z., Yuster, R.: On rainbow connectivity. The Electronic J. Combinatorics 15, R57 (2008)
9. Fellows, M.R., Guo, J., Kanj, I.: The parameterized complexity of some minimum label problems. J. Computer and System Sciences 76, 727–740 (2010)
10. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
11. Krivelevich, M., Yuster, R.: The rainbow connection of a graph is (at most) reciprocal to its minimum degree. J. Graph Theory 63, 185–191 (2010)
12. Uchizawa, K., Aoki, T., Ito, T., Suzuki, A., Zhou, X.: On the rainbow connectivity of graphs: complexity and FPT algorithms. To appear in Algorithmica, doi:10.1007/s00453-012-9689-4