Optim Lett (2018) 12:443–454
https://doi.org/10.1007/s11590-017-1161-6

ORIGINAL PAPER

On the complexity of rainbow spanning forest problem

Francesco Carrabs[1] · **Carmine Cerrone**[3] ·
Raffaele Cerulli[1] · **Selene Silvestri**[2]

Received: 1 July 2016 / Accepted: 12 June 2017 / Published online: 19 June 2017

Abstract Given a graph $G = (V, E, L)$ and a coloring function $\ell : E \rightarrow L$, that assigns a color to each edge of G from a finite color set L, the rainbow spanning forest problem (RSFP) consists of finding a rainbow spanning forest of G such that the number of components is minimum. A spanning forest is rainbow if all its components (trees) are rainbow. A component whose edges have all different colors is called rainbow component. The RSFP on general graphs is known to be NP-complete. In this paper we use the 3-SAT Problem to prove that the RSFP is NP-complete on trees and we prove that the problem is solvable in polynomial time on paths, cycles and if the optimal solution value is equal to 1. Moreover, we provide an approximation algorithm for the RSFP on trees and we show that it approximates the optimal solution within 2.

✉ Carmine Cerrone
carmine.cerrone@unimol.it

Francesco Carrabs
fcarrabs@unisa.it

Raffaele Cerulli
raffaele@unisa.it

Selene Silvestri
selene.silvestri@gmail.com

1 Department of Mathematics, University of Salerno, Via Giovanni Paolo II n. 132, 84084 Fisciano, SA, Italy

2 Department of Computer Science, University of Salerno, Via Giovanni Paolo II n. 132, 84084 Fisciano, SA, Italy

3 Department of Biosciences and Territory, University of Molise, C.da Fonte Lappone, 86090 Pesche, IS, Italy

Keywords Graph theory · Edge-colored graph · Rainbow components · Approximation algorithm

1 Introduction

Given an undirected and edge-colored (labeled) graph G, the *rainbow spanning forest problem* (RSFP) consists of finding a rainbow spanning forest of G having the minimum number of trees. A spanning forest is rainbow if all its trees are rainbow. A tree is rainbow if and only if its edges have different colors. The RSFP belongs to a recently studied class of problems, defined on edge-colored graphs, and is known to be NP-complete [19] on general graphs. A MIP formulation and a metaheuristic approach are proposed in [4]. The edge-colored graphs may be used to model many real-world situations in which we want to distinguish between different types of connections. For example, in telecommunication networks there can be different types of communications media (such as optical fiber, coaxial cable, telephone line), different companies to which the connections belong, or different transmission frequencies. It is then clear that we may be interested in optimizing this factor when we are considering the edges to be included in the solution of the problem that we are going to solve. The RSFP generalizes a well-known problem in the context of edge-colored graphs, that is, the problem of finding the spanning tree of the graph that uses the minimum number of colors (labels) (Minimum Labeling Spanning Tree, or MLST). The MLST was introduced by Chang and Shing-Jiuan [12]. They proved it to be NP-complete and provided an heuristic, the Maximum Vertex Coverage Algorithm (MVCA), as well as an exact algorithm A^*. Brualdi et al. [2] give conditions on color distributions of the complete bipartite graph which guarantee the existence of rainbow subgraph, while Suzuki [21] gives a necessary and sufficient condition for the existence of a rainbow spanning tree in a graph. Other addressed problems in the same field include the Minimum Labeling Steiner Problem [11, 14, 15], the Minimum Labeling Spanning Tree Problem [9, 18], the Minimum Labeling Generalized Forest [3], the Colorful Traveling Salesman Problem [16, 22], the Generalized Minimum Label Spanning Tree Problem [13], the Label-Constrained Minimum Spanning Tree Problem [23], the Labeled Maximum Matching Problem [7], the Maximum Labeled Clique Problem [6], and the Rainbow Cycle Cover Problem [20].

In this paper we prove that the RSFP is NP-complete on trees and we prove that it is easy to solve on paths, cycles and when the optimal solution value is equal to one, namely when there exists a rainbow spanning tree in G.

The sequel of the paper is organized as follows. In Sect. 2 we formally define the problem and introduce definitions and basic notations. Section 3 provides the proof of NP-completeness of the problem on trees. In Sect. 4 we present three polynomial cases for the RSFP. Finally, in Sect. 5 an approximation algorithm, approximating the optimal solution within 2, is introduced and concluding remarks are given in Sect. 6.

2 Definitions and notation

Let $G = (V, E, L)$ be a connected and undirected graph, where V is the set of n vertices, E is the set of m edges and L a set of l colors. In addition, let $\ell : E \rightarrow L$

be a coloring function that assigns a color to each edge of G from the color set L and let $\bar{\ell}(E') = \cup_{e \in E'} \ell(e)$, $E' \subseteq E$. A *spanning forest* of G is an acyclic subgraph of G containing all vertices of G and in which any connected component is a tree. A tree of the forest whose edges have all different colors is called *rainbow tree*. A *rainbow spanning forest (rsf)* of G is a spanning forest of the graph such that all trees are rainbow. The rainbow spanning forest problem (RSFP) consists of finding a *rsf* with the least number of rainbow trees. We denote by $F(G) = (V, E_F, L_F)$ any *rsf* of G and by $T_{F(G)} = \{T_1, \ldots T_{z(F(G))}\}$ the set of rainbow trees of $F(G)$. When no confusion arises, we simply denote them by F and T_F, respectively. Moreover, let $z(F(G)) = |V| - |E_{F(G)}|$ be the number of trees in $F(G)$. According to the definition of *rsf*, if $T_i = (V_{T_i}, E_{T_i}, L_{T_i})$ is a tree of $F(G)$, then $|E_{T_i}| = |L_{T_i}| = |V_{T_i}| - 1$. The RSFP consists of finding the rainbow spanning forest $F^*(G)$ with the minimum number z^* of rainbow components.

3 RSFP complexity on trees

In this section we prove that the RSFP is NP-complete even if the graph is acyclic. To the best of our knowledge, no proof for the NP-completeness of the RSFP on trees has ever been put forward.

Theorem 1 *The RSFP on edge-colored trees is NP-Complete.*

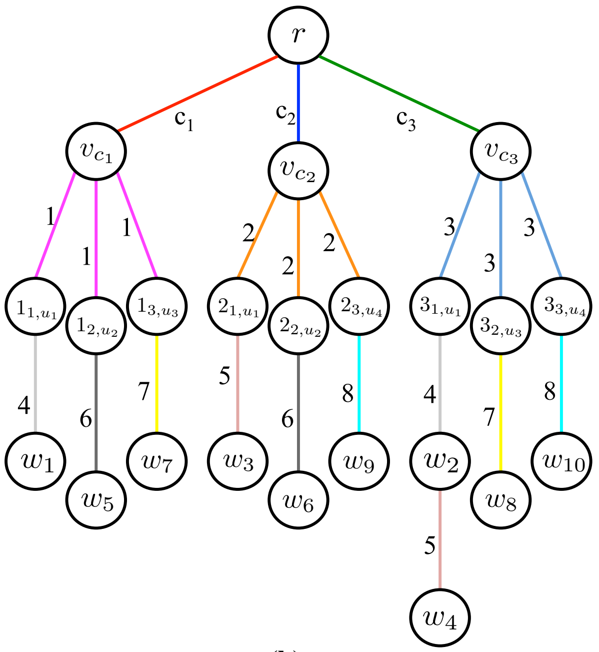
Proof We prove the theorem by reduction from the 3-SAT Problem. Let ϕ be our formula for 3-SAT, written in a conjunctive normal form, containing d literals $U = \{u_1, \ldots, u_d\}$ and b clauses $C = \{c_1, \ldots, c_b\}$. The decisional version of 3-SAT consists in verifying whether there exists an assignment of values to U that makes every clause true. We now define, from the generic instance of 3-SAT, an acyclic graph $T = (V, E, L)$, with a coloring function of the edges (see the example in Fig. 1). At the beginning let the set of the vertices be $V = \{r\}$, where r is the root of the graph T, and let E be the empty set. To each $c_h \in C$ we associate a vertex v_{c_h} and define the edge $(r, v_{c_h}) \in E$ of color c_h. Moreover, for each $c_h \in C$ we define three vertices h_{1,u_i}, h_{2,u_j} and h_{3,u_k}, where u_i, u_j and u_k are the literals of clause c_h, and three edges (v_{c_h}, h_{1,u_i}), (v_{c_h}, h_{2,u_j}), (v_{c_h}, h_{3,u_k}) to which we associate the same color h. Note that the edge (v_{c_h}, h_{i,u_t}) is associated with the ith literal of the clause c_h. Furthermore for all h_{i,u_t}, we build in the graph T a path $P_{h_{i,u_t}}$, whose first vertex is h_{i,u_t}, as follows:

- $P_{h_{i,u_t}}$ has length $|N_t|$ if the clause c_h contains u_t
- $P_{h_{i,u_t}}$ has length $|Y_t|$ if the clause c_h contains $\neg u_t$,

where, for each $u_t \in U$, if u_t is in the position $\bar{p}$ of a clause $\bar{c}$, then the pair $(\bar{p}, \bar{c})$ belongs to Y_t. Otherwise, if $\neg u_t$ is in the position $\bar{p}$ of a clause $\bar{c}$, then the pair $(\bar{p}, \bar{c})$ belongs to N_t. For instance, in Fig. 1, for literal u_1 we have $Y_1 = \{(1, 1), (1, 2)\}$ and $N_1 = \{(1, 3)\}$. More in detail, for each $u_t \in U$ and for each $((y^1, y^2), (n^1, n^2))$, i.e. $(y^1, y^2) \in Y_t$ and $(n^1, n^2) \in N_t$, we add an edge e to the path $P_{y^2_{y^1,u_t}}$ and an edge f to the path $P_{n^2_{n^1,u_t}}$. To e and f we assign a color a, different from all the colors used until now. Since $|Y_1| = 2$ and $|N_1| = 1$, we have two pairs associated to u_1: $((1, 1), (1, 3))$ and $((1, 2), (1, 3))$. For $((1, 1), (1, 3))$, in Fig. 1, we have $(1_{1,u_1}, w_1)$ and $(3_{1,u_1}, w_2)$

$$(u_1 \vee u_2 \vee \neg u_3) \wedge (u_1 \vee \neg u_2 \vee u_4) \wedge (\neg u_1 \vee u_3 \vee \neg u_4)$$

(a)



(b)

Fig. 1 **a** A generic instance of the 3-SAT Problem, and, **b** the corresponding instance of the bounded rainbow spanning forest problem in acyclic edge-colored graphs

in $P_{1_{1,u_1}}$ and $P_{3_{1,u_1}}$, respectively, having color 4. Moreover, for $((1, 2), (1, 3))$ we have $(2_{1,u_1}, w_3)$ and (w_2, w_4) in $P_{2_{1,u_1}}$ and $P_{3_{1,u_1}}$, respectively, having color 5. Note that since in a same clause cannot be present a literal u_t and its negated $\neg u_t$, each path $P_{h_{i,u_t}}$ is always rainbow. Therefore the set of vertices, edges and colors of the tree T are the following:

- $V = \{r\} \cup \{v_{c_h}, h_{1,u_i}, h_{2,u_j}, h_{3,u_k} : h = 1, \ldots, b\} \cup \{w_i : i = 1, \ldots, 2\bar{q}\}$,
- $E = \{(r, v_{c_h}), (v_{c_h}, h_{1,u_i}), (v_{c_h}, h_{2,u_j}), (v_{c_h}, h_{3,u_k}) : h = 1, \ldots, b\} \cup \{e_i : i = 1, \ldots, 2\bar{q}\}$,
- $L = \{c_h, h : h = 1, \ldots, b\} \cup \{i : i = 1, \ldots, \bar{q}\}$,

where $\bar{q} = \sum_{t=1}^{d} |Y_t| \times |N_t|$. This construction can be accomplished in polynomial time.

We want to show that there is an assignment of values to U that makes every clause true if and only if exists a spanning forest of T using $2b + 1$ rainbow components. Note that, in order to preserve the rainbow property, at most one of the three edges (v_{c_h}, h_{1,u_i}), (v_{c_h}, h_{2,u_j}), (v_{c_h}, h_{3,u_k}), associated with each clause c_h, for all $h \in \{1, \ldots, b\}$, can appear in a rainbow spanning forest (the three edges are incident to the same vertex v_{c_h} and have the same color h). Consider now an assignment of values to U which makes every clause true. We can define a rainbow spanning forest

with $2b + 1$ components by selecting the edges $\{(r, v_{c_h}) : h \in \{1, \ldots, b\}\}$, whose colors are all different. Furthermore, for each clause c_h, among (v_{c_h}, h_{1,u_i}), (v_{c_h}, h_{2,u_j}), (v_{c_h}, h_{3,u_k}), we select the edge associated with the literal having value true in c_h. If more than one literal is true, we arbitrarily select only one of the corresponding edges. Moreover, we select all the edges of the rainbow path $P_{h_{i,u_t}}$, for all h_{i,u_t}. Note that two edges belonging to the rainbow paths have the same color if and only if they are associated with pairs of literals (L_1, L_2) such that if $L_1 = u_t$, then $L_2 = \neg u_t$, which surely cannot be simultaneously true. Therefore, at least one of the two edges linking these paths to the vertices associated to the clauses containing the literals, does not belong to the rainbow spanning forest. This ensures that the two edges belong to different rainbow components. In total, we do not select $2b$ edges and therefore we obtain a rainbow spanning forest with $2b + 1$ components.
Conversely, suppose that there exists a spanning forest with $2b + 1$ rainbow components. As previously observed, edges (v_{c_h}, h_{1,u_i}), (v_{c_h}, h_{2,u_j}), and (v_{c_h}, h_{3,u_k}), $h \in \{1, \ldots, b\}$, have the same color and are incident to the same vertex v_{c_h}, therefore at most one of them can appear in the rainbow spanning forest. Moreover, since we have supposed that exists a spanning forest with $2b + 1$ rainbow components, we are sure that exactly one of them has to appear in the rainbow spanning forest, otherwise it would be impossible to have the $2b + 1$ components. This ensures that the root node has to be connected to every clause in a single component. Note that to accomplish an objective of less $2b + 1$ components is not possible. Indeed, every time that an edge is removed from T the number of component increases by one and, as previously shown, to preserve the rainbow property, at least $2b$ edges have to be removed. The edges (v_{c_h}, h_{1,u_i}), (v_{c_h}, h_{2,u_j}), and (v_{c_h}, h_{3,u_k}), $h \in \{1, \ldots, b\}$ that appear in the rainbow spanning forest with $2b + 1$ components represent an assignment of values to U, which makes every clause true. □

4 Polynomial cases for the RSFP

In this section we prove that the RSFP is polynomially solvable on paths, cycles and when $z = 1$, namely when there exists into the graph a rainbow spanning tree.

Lemma 1 *Let $P = (V, E, L)$ be a edge-colored path. The RSFP on P can be solved in linear time.*

Proof Let $e_1, \ldots, e_m$ be the sequence of the edges in P. Starting from e_1 the algorithm (summarized in Algorithm 1) visits G according to the previous sequence. As soon as it meets an edge e_k such that $\ell(e_k) \in \bar{\ell}(\{e_1, \ldots, e_{k-1}\})$, the algorithm removes e_k (Algorithm 1 line 5). Let e_h be the last edge removed. Until there are edges to visit (Algorithm 1 line 1), starting from e_{h+1} the algorithm visits the remaining sequence of the path and as soon as it meets an edge e_{h+k} such that $\ell(e_{h+k}) \in \bar{\ell}(\{e_{h+1}, \ldots, e_{h+k-1}\})$, the algorithm removes e_{h+k} and updates e_h. The solution value will be equal to one plus the number of the edges removed. The algorithm runs in $O(n)$.

Suppose our algorithm does not find an optimal solution, i.e. a spanning forest with the minimum number of rainbow components. Let $\alpha + \beta$ be the best solution value provided by our algorithm, with $\beta > 0$. Moreover, let $S = (V, E_S)$ be an optimal solution, let α be the corresponding optimal solution value and let $\bar{e}_1, \ldots, \bar{e}_{\alpha-1}$ be the edges that have been removed, i.e. the edges belonging to $E \backslash E_S$. For the sake of simplicity, if $i < j$, $\bar{e}_i$ appears in $e_1, \ldots, e_m$ before $\bar{e}_j$.

Note that since as soon as our algorithm meets an edge e_k such that $\ell(e_k) \in \bar{\ell}(\{e_1, \ldots, e_{k-1}\})$ it removes e_k, this implies that $e_1, \ldots, e_{k-1}$ is a rainbow subsequence. Therefore, it is easy to see that $\bar{e}_1$ appears in $e_1, \ldots, e_m$ before e_k or it is e_k. Let t be the number of edges in $\bar{e}_2, \ldots, \bar{e}_{\alpha-1}$ that appear in $e_1, \ldots, e_m$ before e_k. If $t = 0$, $S' = (V, E_{S'})$ with $E_{S'} = E_S \backslash \{e_k\} \cup \{\bar{e}_1\}$ is a feasible solution having α rainbow components, otherwise $S' = (V, E_{S'})$ with $E_{S'} = E_S \backslash \{e_k\} \cup \{\bar{e}_1, \ldots, \bar{e}_{1+t}\}$ is a feasible solution having $\alpha' = \alpha - t$ rainbow components. By iterating this procedure we prove that S can be easily transformed into a solution S', having at most α rainbow components, that our algorithm would provide, but this is absurd because we have assumed $\beta > 0$. □

Algorithm 1: pathAlgorithm(P)

```
Input: edge-colored path $P = (V, E, L)$ with $E = \{e_1, \ldots, e_m\}$, $E_S = E, h = 0, k = 2$
Output: a rsf $S = (V, E_S, L_S)$ of $P$ and the last edge $e_k$ removed
1 while $k \le m$ do
2     if $\ell(e_k) \notin \bar{\ell}(\{e_{h+1}, \ldots, e_{k-1}\})$ then
3         $k = k + 1$
4     else
5         $E_S = E_S \backslash e_k, h = k$ and $k = k + 2$
6 $L_S = \bar{\ell}(E_S), e_k = e_h$
```

Corollary 1 *Let $G = (V, E, L)$ be a edge-colored cycle. The RSFP on G can be solved in polynomial time.*

Proof Note that if E contains exactly two edges having the same color, it is sufficient to remove one of these two edges to obtain an optimal rainbow forest having optimal value equal to one. Otherwise, for $e \in E$, the algorithm removes edge e, obtaining a path P_e, and invokes *pathAlgorithm*(P_e). Let s_e be the optimal solution value on P_e. It is easy to see that the optimal solution value for G is equal to $\min\{s_1, \ldots, s_m\}$. The algorithm runs in $O(n^2)$. □

Now we want to prove that the RSFP can be solved in polynomial time when $z = 1$. Obviously, it is not possible to obtain this goal by enumerating all the spanning trees of G. This is because the algorithms to enumerate all the spanning tree of G are pseudo-polynomial [17].

Given a spanning tree T of G, it is a *maximum tree* of G if and only if $|L_T|$ is maximum. The following theorem holds:

Theorem 2 [1] *The problem of finding a maximum tree T in G is solvable in polynomial time.*

The algorithm of Broersma and Li [1] computes the maximum tree of G in $O(n^2m)$ and the following theorem shows how to use this algorithm to individuate a rainbow spanning tree in G with the same running time.

Theorem 3 *RSFP is solvable in polynomial time if in G there exists a rainbow spanning tree.*

Proof Given a graph G, let T' be the maximum tree of G computed by algorithm of Broersma and Li. It is easy to see that if $|L_{T'}| = n-1$ than T' is a rainbow spanning tree of G. □

5 Approximability for the RSPF on trees

In this section we provide an approximation algorithm for the RSFP on trees and we prove that it approximates the optimal solution within 2, i.e. it finds a *rsf* with at most 2 times the minimum number of rainbow components. Before describing the algorithm, we need to introduce some notations. Given an edge-colored tree $T = (V, E, L)$, let $B = \{b \in V : |\delta(b)| \geq 3\}$, where $\delta(b) = \{(v, u) \in E : b = v \text{ or } b = u\}$. Moreover, let $P_{v,w} = (V_{P_{v,w}}, E_{P_{v,w}}, L_{P_{v,w}})$ be a path in T from v to w. For any vertex $b \in B$, we call *leaf path from b to w* a path $P_{b,w}$ such that $(V_{P_{b,w}} \backslash \{b\}) \subset (V \backslash B)$ and $|\delta(w)| = 1$ and we call *internal path from b to w* a path $P_{b,w}$ such that $b, w \in B$, $(V_{P_{b,w}} \backslash \{b, w\}) \subset (V \backslash B)$. Let $\alpha(b) = \{P_{b,w} : P_{b,w} \text{ is a leaf path from } b \text{ to } w\}$ and let $\beta(b) = \{P_{b,w} : P_{b,w} \text{ is an internal path from } b \text{ to } w\}$. Note that for any $b \in B$, $|\delta(b)| = |\alpha(b)| + |\beta(b)|$. We can write the set B as

$$B = B^L \cup B^I \tag{1}$$

where $B^L = \{b \in B : |\beta(b)| \leq 1\}$ and $B^I = \{b \in B : |\beta(b)| > 1\}$. For instance, in Fig. 2, $B = \{v_1, v_5, v_6\}$, more in detail $B^L = \{v_5, v_6\}$ and $B^I = \{v_1\}$. Moreover, $\alpha(v_1) = \{P_{v_1,v_{13}}, P_{v_1,v_{16}}\}$, $\alpha(v_5) = \{P_{v_5,v_{14}}, P_{v_5,v_{15}}, P_{v_5,v_{19}}, P_{v_5,v_{20}}\}$, $\alpha(v_6) = \{P_{v_6,v_{17}}, P_{v_6,v_7}\}$, $\beta(v_1) = \{P_{v_1,v_5}, P_{v_1,v_6}\}$, $\beta(v_5) = \{P_{v_5,v_1}\}$, $\beta(v_6) = \{P_{v_6,v_1}\}$.

The approximation algorithm, that is summarized in Algorithm 5, takes in input an edge-colored tree $T = (V, E, L)$ and an initial feasible solution $S = (V_S, E_S, L_S)$ with no edges, i.e. a spanning forest in which each vertex is a component, and therefore $E_S = L_S = \emptyset$. The algorithm has two main functions: *clear* and *reduce*. The function *clear*, thanks to *pathAlgorithm* modifies the tree T and updates the feasible solution S. In particular, while there exists a leaf path $\bar{P}_{b,w}$, with $b \in B$, such that $\bar{P}_{b,w}$ is not rainbow (Algorithm 2 line 1), the function *clear* invokes *pathAlgorithm*$(\bar{P}_{w,b})$, i.e. on the path from w to b (Algorithm 2 line 2), and obtains on it an optimal solution (V_W, E_W, L_W). The function *pathAlgorithm* returns also the last edge f that it removes from $\bar{P}_{w,b}$ (Algorithm 1 line 6). According to (V_W, E_W, L_W) and f, it updates the sets E and E_S (Algorithm 2 lines 3 and 4). The function *clear* returns a new tree T such that, for any $b \in B$ and for any $P_{b,w} \in \alpha(b)$, $P_{b,w}$ is rainbow. After applying the function *clear*(T, S) on the graph in Fig. 2, we obtain the new tree in Fig. 3 and a current feasible solution S having $E_S = \{(v_3, v_8), (v_8, v_{12}), (v_{12}, v_{16}), (v_9, v_{13})\}$.

The function *reduce* is invoked only after the function *clear*, i.e. only on trees such that, for any $b \in B$ and for any $P_{b,w} \in \alpha(b)$, $P_{b,w}$ is rainbow. This function,

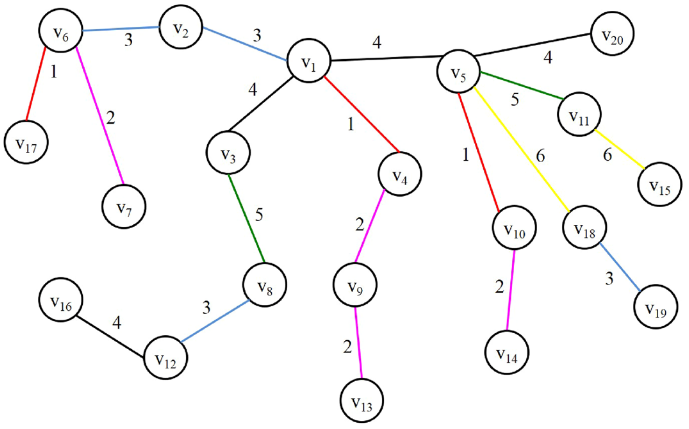


Fig. 2 Edge-colored tree: example

Algorithm 2: clear(T, S)

Output: (T, S)

```
1 while exists a leaf path $\bar{P}_{b,w}$ in $T$, with $b \in B$, such that $\bar{P}_{b,w}$ is not rainbow do
2     $((V_W, E_W, L_W), f) = \text{pathAlgorithm}(\bar{P}_{w,b})$
3     $E = E \backslash (\{e \in E_{\bar{P}_{w,b}} : e \text{ appears in } \bar{P}_{w,b} \text{ before } f\} \cup f)$
4     $E_S = E_S \cup \{e \in E_W : e \text{ appears in } \bar{P}_{w,b} \text{ before } f\}$
```

given $b \in B$, verifies for each pair $P_{b,i}, P_{b,j} \in \alpha(b)$ if the two paths have at least a color in common (Algorithm 3 line 1). If $\bar{\ell}(P_{b,i}) \cap \bar{\ell}(P_{b,j}) \neq \emptyset$, the function *reduce* adds the edges $e \in (E_{P_{b,i}} \cup E_{P_{b,j}}) \backslash \{(b, \bar{i}), (b, \bar{j})\}$ to the set E_S, where $(b, \bar{i})$ and $(b, \bar{j})$ (Algorithm 3 line 2) are the edges incident to b in $P_{b,i}$ and $P_{b,j}$, respectively, and deletes both the paths from the graph T (Algorithm 3 lines 3 and 4). Note that, by adding these edges to E_S, we are creating two new rainbow components. In the example in Fig. 3, for $b = v_5$, $\bar{\ell}(E_{P_{v_5,v_{19}}}) \cap \bar{\ell}(E_{P_{v_5,v_{15}}}) \neq \emptyset$, therefore, the function

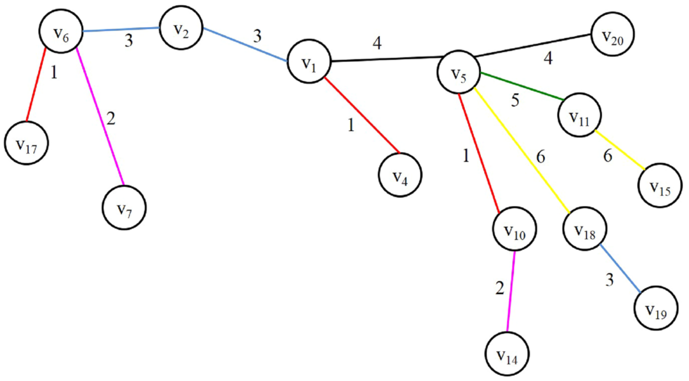


Fig. 3 Edge-colored tree: clear(T, S)

Algorithm 3: reduce(T, S, b)

```
Output: $(T, S)$
1 $\forall P_{b,i}, P_{b,j} \in \alpha(b)$ if $\bar{\ell}(P_{b,i}) \cap \bar{\ell}(P_{b,j}) \neq \emptyset$ then
2     $(b, \bar{i}) = E_{P_{b,i}} \cap \delta(b)$ and $(b, \bar{j}) = E_{P_{b,j}} \cap \delta(b)$
3     $E_S = E_S \cup ((E_{P_{b,i}} \cup E_{P_{b,j}}) \backslash \{(b, \bar{i}), (b, \bar{j})\})$
4     $E = E \backslash (E_{P_{b,i}} \cup E_{P_{b,j}})$
5 merge$(\alpha(b))$
```

reduce deletes both the paths and updates E_S by adding $\{(v_{18}, v_{19}), (v_{11}, v_{15})\}$. In this example there are no other paths such that $\bar{\ell}(P_{b,i}) \cap \bar{\ell}(P_{b,j}) \neq \emptyset$. When there are no more paths $P_{b,i}, P_{b,j} \in \alpha(b)$ such that $\bar{\ell}(P_{b,i}) \cap \bar{\ell}(P_{b,j}) \neq \emptyset$, the function *merge*$(\alpha(b))$, summarized in Algorithm 4, merges all the rainbow paths $P_{b,i} \in \alpha(b)$ and creates a new path $\bar{P}_{b,u}$. More in detail, suppose $P_{b,i_1}, \ldots, P_{b,i_s}$ are the s leaf rainbow paths from b that have to be merged. The function *merge* creates the path $\bar{P}_{b,u}$ by disconnecting P_{b,i_j} from b and connecting it to $P_{b,i_{j-1}}$, for $j = 2, \ldots, s$ (Algorithm 4 line 3). Note that the order in which we consider the s paths is irrelevant. In the example in Fig. 3, the algorithm merges $P_{v_5,v_{14}}$ and $P_{v_5,v_{20}}$. The tree obtained is shown in Fig. 4. The function merge, although modifying the structure of the tree T, does not affect the solution. In Algorithm 5 line 1, the algorithm invokes the function *clear* that returns a new tree T , for any $b \in B$ and for any $P_{b,w} \in \alpha(b)$, $P_{b,w}$ is rainbow and updates the current feasible solution S. While there exists $b \in B$ such that $|\beta(b)| \leq 1$ (Algorithm 5 line 2), the algorithm invokes the function *reduce*(T, S, b) (Algorithm 5 line 3). In the tree T, obtained by applying the function *reduce*, vertex $b \notin B$. On this new tree T the algorithm applies again the function *clear* (Algorithm 5

Algorithm 4: merge$(\alpha(b))$

```
Input: $\alpha(b) = \{P_{b,i_1}, \ldots, P_{b,i_s}\}$, $\bar{P}_{b,u} = P_{b,i_1}$, $u = i_1$ and $j = 2$
Output: $\bar{P}_{b,u}$
1 while $j \leq s$ do
2     $(b, \bar{j}) = E_{P_{b,i_j}} \cap \delta(b)$
3     $E_{\bar{P}_{b,u}} = E_{\bar{P}_{b,u}} \cup \{E_{P_{b,i_j}} \backslash (b, \bar{j})\} \cup (u, \bar{j})$
4     $u = i_j$, $j = j + 1$
```

Algorithm 5: approximation algorithm for the RSFP on trees

```
Input: edge-colored tree $T = (V, E, L)$ and initial feasible solution $S = (V, \emptyset, \emptyset)$
Output: a rainbow spanning forest $S = (V, E_S, L_S)$ of $T$
1 $(T, S) = $ clear$(T, S)$
2 while $\exists b \in B$ such that $|\beta(b)| \leq 1$ do
3     $(T, S) = $ reduce$(T, S, b)$
4     $(T, S) = $ clear$(T, S)$
```

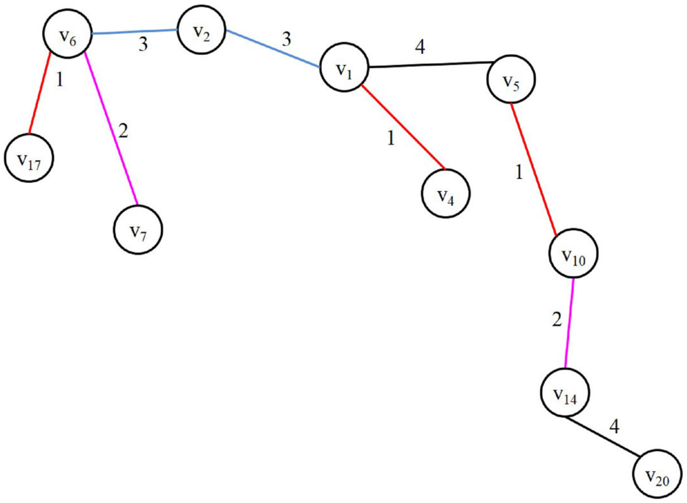


Fig. 4 Edge-colored tree: reduce(T, S)

line 4). The algorithm stops when no $b \in B$ exists such that $|\beta(b)| \leq 1$. The solution S provided is a *rsf*.

Before showing that the algorithm approximates the optimal solution within 2, we need some notations. Given an edge-colored tree $T = (E, V, L)$, let $e, f \in E$ such that $\ell(e) = \ell(f)$. We denote $P^{e,f} = (V_{P^{e,f}}, E_{P^{e,f}}, L_{P^{e,f}})$ the path connecting e and f in T. Such path exists and it is unique since T is a tree. It is easy to see that at least one edge of the path $P^{e,f}$ including e and f, i.e. at least one edge $g \in E_{P^{e,f}}$, cannot belong to any *rsf*.

Theorem 4 *The approximation algorithm for the RSFP on trees finds a rsf with at most 2 times the number of rainbow components on any rsf.*

Proof Before proceeding with the proof, we need the following remark.

Remark 1 Given an edge-colored tree $T = (E, V, L)$, let P^{e_i, f_i}, $i = 1, \ldots, k$, with $\ell(e_i) = \ell(f_i)$, be k paths linking e_i, f_i. If $E_{P^{e_h, f_h}} \cap E_{P^{e_j, f_j}} = \emptyset$, $h \neq j$, then $k + 1$ is a lower bound on the number of components in the optimal *rsf* of T.

Proof P^{e_i, f_i}, $i = 1, \ldots, k$, are edge disjoint, therefore to preserve the rainbow property of any feasible solution we have to remove at least k edges from T, i.e. one edge from each path. By removing k edges from a tree, we obtain a forest having $k + 1$ components.

Our approximation algorithm identifies only disjoint paths. We need to distinguish two cases:

- paths identified in the function *clear*;
- paths identified in the function *reduce*.

In the function *clear*, as soon as it meets an edge e_k such that $\ell(e_k) \in \bar{\ell}\{e_{h+1}, \ldots, e_{k-1}\}$, with $h \leq k - 2$, *pathAlgorithm* removes e_k. It is easy to see that the sequence

$e_{h+1}, \dots, e_k$ contains a path $P^{e,f}$ with $f = e_k$, such that $\ell(e) = \ell(f)$. Let q be the number of paths $P^{e,f}$, with $\ell(e) = \ell(f)$ identified in the function *clear*. Note that these paths are edge disjoint and for each $P^{e,f}$ the function *pathAlgorithm* removes one edge. Moreover, note that in the function *reduce*, given $b \in B$, if $P_{b,i}, P_{b,j} \in \alpha(b)$ have at least a color in common, there exist two edges $e \in E_{P_{b,i}}$ and $f \in E_{P_{b,j}}$ such that $\ell(e) = \ell(f)$. The union of the two paths $P_{b,i}, P_{b,j}$ contains the path $P^{e,f}$, therefore by deleting the two edges belonging to $E_{P_{b,i}} \cup E_{P_{b,j}}$ and incident to b, the algorithm is removing two edges from the path $P^{e,f}$. Note that, if $P_{b,i}, P_{b,j} \in \alpha(b)$ have more than one color in common, we consider only one $P^{e,f}$, with $\ell(e) = \ell(f)$, per pair. Let p be the number of paths $P^{e,f}$, with $\ell(e) = \ell(f)$, identified. It is easy to see that, thanks to the previous assumption, these paths are edge disjoints. In total the algorithm identifies $k = p + q$ edge disjoint paths.

Accordingly, the number of components of the rainbow spanning forest that the approximation algorithm identifies is $z = 1 + q + 2p \leq 1 + 2q + 2p = 1 + 2k < 2(k+1)$. Due to Remark 1, $k+1$ is a lower bound on the minimum number of rainbow components, therefore $k + 1 \leq z^* \leq z < 2(k+1)$, and hence $z < 2z^*$, where z^* represents the optimal solution value.

6 Conclusion

In this paper we proved that RSFP is NP-complete on trees too. Moreover, we have provided some polynomial cases for the problem and we introduced a 2-approximate algorithm. A possible direction for future works is to develop new approaches based on Carousel Schema [8] or metaheuristics like a Tabu Search [5,10].

References

1. Broersma, H., Li, X.: Spanning trees with many or few colors in edge-colored graphs. Graph Theory **17**, 259–269 (1997)
2. Brualdi, R.A., Hollingsworth, S.: Multicolored forests in complete bipartite graphs. Discret. Math. **240**, 239–245 (2001)
3. Carr, R.D., Doddi, S., Konjedov, G., Marathe, M.: On the red-blue set cover problem. In: 11th ACN-SIAM Symposium on Discrete Algorithms, pp. 345–353 (2000)
4. Carrabs, F., Cerrone, C., Cerulli, R., Silvestri, S.: The rainbow spanning forest problem. Soft. Comput. pp. 1–12 (2017)
5. Carrabs, F., Cerrone, C., Cerulli, R.: A tabu search approach for the circle packing problem. In: IEEE 2014 17th International Conference on Network-Based Information Systems, pp. 165–171 (2014)
6. Carrabs, F., Cerulli, R., Dell'Olmo, P.: A mathematical programming approach for the maximum labeled clique problem. Proc. Soc. Behav. Sci. **108**, 69–78 (2014)
7. Carrabs, F., Cerulli, R., Gentili, M.: The labeled maximum matching problem. Comput. Oper. Res. **36**, 1859–1871 (2009)
8. Cerrone, C., Cerulli, R., Golden, B.: Carousel greedy: a generalized greedy algorithm with applications in optimization. Comput. Oper. Res. **85**, 97–112 (2017)
9. Cerrone, C., Cerulli, R., Gaudioso, M.: Omega one multi ethnic genetic approach. Optim. Lett. **10**(2), 309–324 (2016)
10. Cerrone, C., Cerulli, R., Gentili, M.: Vehicle-id sensor location for route flow recognition: Models and algorithms. Eur. J. Oper. Res. **247**(2), 618–629 (2015)
11. Cerulli, R., Fink, A., Gentili, M., Voß, S.: Extensions of the minimum labelling spanning tree problem. J. Telecommun. Inf. Technol. **4**, 39–45 (2006)

12. Chang, R.S., Leu, S.J.: The minimum labeling spanning trees. Inf. Process. Lett. **63**, 277–282 (1997)
13. Chen, Y., Cornick, N., Hall, A.O., Shajpal, R., Silberholz, J., Yahav, I., Golden, B.: Comparison of heuristics for solving the gmlst problem. In: Raghavan, S., Golden, B., Wasil, E. (eds.) Telecommunications Modeling, Policy, and Technology, pp. 191–217. Springer, Berlin (2008)
14. Consoli, S., Darby-Dowman, K., Mladenović, N., Moreno-Pérez, J.A.: Variable neighbourhood search for the minimum labelling steiner tree problem. Ann. Oper. Res. **172**, 71–96 (2009)
15. Consoli, S., Moreno-Pérez, J.A., Darby-Dowman, K., Mladenović, N.: Discrete particle swarm optimization for the minimum labelling steiner tree problem. Nat. Comput. **9**, 29–46 (2010)
16. Jozefowiez, N., Laporte, G., Semet, F.: A branch-and-cut algorithm for the minimum labeling hamiltonian cycle problem and two variants. Comput. Oper. Res. **38**, 1534–1542 (2011)
17. Kapoor, S., Ramesh, H.: Algorithms for enumerating all spanning trees of undirected and weighted graphs. SIAM J. Comput. **24**(2), 247–265 (1995)
18. Krumke, S., Wirth, H.: On the minimum label spanning tree problem. Inf. Process. Lett. **66**(2), 81–85 (1998)
19. Li, X., Zhang, X.Y.: On the minimum monochromatic or multicolored subgraph partition problems. Theor. Comput. Sci. **385**, 1–10 (2007)
20. Silvestri, S., Laporte, G., Cerulli, R.: The rainbow cycle cover problem. Networks **68**, 260–270 (2016)
21. Suzuki, K.: A necessary and sufficient condition for the existence of a heterochromatic spanning tree in a graph. Graph Comb. **22**, 261–269 (2006)
22. Xiong, Y., Golden, B., Wasil, E.: The colorful traveling salesman problem. In: Baker, E.K., Joseph, A., Mehrotra, A., Trick, M.A. (eds.) Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, pp. 115–123. Springer, Berlin (2007)
23. Xiongm, Y., Golden, B., Wasil, E., Chen, S.: The label-constrained minimum spanning tree problem. In: Raghavan, S., Golden, B., Wasil, E. (eds.) Telecommunications Modeling, Policy, and Technology, pp. 39–58. Springer, Berlin (2008)