A pre-processing heuristic for the k-labeled spanning forest problem

Yasser dos Santos Djalo
Departamento de Computação - Universidade Federal do Ceará
Av. Humberto Monte, s/n, Campus do Pici, Bloco 910. CEP 60440-900. Fortaleza, Ceará, Brasil
yasserdjalo@alu.ufc.br

Manoel Campêlo
Departamento de Estatística e Matemática Aplicada - Universidade Federal do Ceará
Av. Humberto Monte, s/n, Campus do Pici, Bloco 910. CEP 60440-900. Fortaleza, Ceará, Brasil
mcampelo@ufc.br

ABSTRACT

Given a positive integer k and a graph in which each edge has an associated label, the k-labeled spanning forest problem (kLSFP) is an NP-Hard combinatorial optimization problem that consists in finding a spanning forest of the graph with minimum number of connected components by using at most k labels. The problem was originally defined to model an application related to telecommunications networks and generalizes another better known problem called the minimum labeling spanning tree problem (MLSTP). In this paper, we propose a new pre-processing heuristic that breaks cycles of the input graph. We performed computational experiments and a statistical analysis to evaluate the effectiveness of the pre-processing heuristic used within a branch-and-cut method.

KEYWORDS. Pre-processing. Edge-labeled graph. Spanning forest.

Paper topics: Combinatorial Optimization. Graph Theory and Algorithms.

DOI: 10.59254/sbpo-2024-193802
https://proceedings.science/p/193802?lang=pt-br

1. Introduction

The k-labeled spanning forest problem (kLSFP) is an NP-Hard problem that consists in, given a graph with labels associated to the edges and a positive integer k, finding a spanning forest of the graph with minimum number of connected components by using at most k labels. It was first formulated by Cerulli et al. [2014] with the intent of modeling an application related to telecommunications networks. It is a generalization of another problem called minimum labeling spanning tree problem (MLSTP), which is also NP-Hard [Cerulli et al., 2014]. Still in [Cerulli et al., 2014], the authors proposed heuristics and a Branch & Bound algorithm to solve the problem. Later, Figueredo [2020] proposed three mathematical programming formulations. It is also worth mentioning that Pinheiro [2022] proposed a matheuristic for the problem and proved that the problem remains NP-Hard even when the components of the input graph are only triangles and edges.

1.1. Problem characterization

In this section we will introduce key definitions and propositions that are necessary for understanding this paper, based on [Figueredo, 2020]. For further details and proofs of the propositions, we recommend consulting Figueredo's work.

First, we will define what is an edge-labeled graph:

Definition 1.1. *An edge-labeled graph (ELG) is a tuple $G = (V, E, L, l)$ where V is the set of vertices, E is the set of edges, L is the set of labels and $l : E \mapsto L$ is a function that maps each edge to a label. For $E' \subseteq E$, let $l(E')$ denote the set $\{l(e) : e \in E'\}$.*

Now we can give a precise definition of the problem:

Definition 1.2. *Let $\mathcal{W}(G)$ denote the number of connected components of G. Furthermore, let $\mathcal{F}_G(k)$ be the set of all spanning forests of G where $|\,l(E_F)\,| \leq k$. Given an ELG $G = (V, E, L, l)$ and $k \in \mathbb{Z}^+$, the k-labeled spanning forest problem (kLSFP) consists in finding a spanning forest $F^* \in \mathcal{F}_G(k)$ such that $\mathcal{W}(F^*)$ is minimized.*

Figure 1 illustrates an instance of kLSFP for $k = 2$, where Figure 1(a) shows the input graph and Figure 1(b) shows the optimal solution.

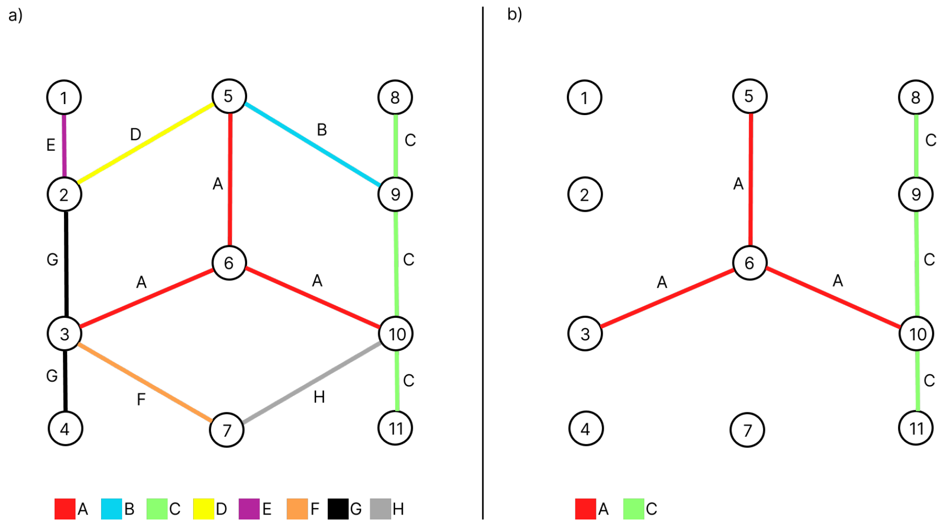

Figure 1: Example of kLSFP. **(a)** Input graph. **(b)** Optimal solution.

In addition to the definition we have shown, there is another equivalent way to see the problem.

DOI: 10.59254/sbpo-2024-193802
https://proceedings.science/p/193802?lang=pt-br

Proposition 1.1. *[Cerulli et al., 2014] For $L' \subseteq L$, let $G[L']$ denote the the spanning subgraph of G induced by the edges $E[L'] = \{e \in E : l(e) \in L'\}$. Given an ELG $G = (V, E, L, l)$ and $k \in \mathbb{Z}^+$, the kLSFP is equivalent to finding a subset of labels $L^* \subseteq L$ such that $| L^* | \leq k$ and $\mathcal{W}(G[L^*])$ is minimized.*

Intuitively, this proposition means that the problem is equivalent to selecting at most k labels such that the spanning subgraph induced by these labels is as connected as possible.

Now we will show another characterization of the problem that is particularly important due to the fact that it is used in the definition of the integer linear programming model that we will present later. To state it, we need to introduce the following definition:

Definition 1.3. *Given an ELG $G = (V, E, L, l)$, the extended graph of G is another ELG $G^+ = (V^+, E^+, L^+, l^+)$ such that $V^+ = V \cup \{s\}$, where s is a new vertex, $E^+ = E \cup \{(s, v) : v \in V\}$, $L^+ = L \cup C$, where $C = \{c_v : v \in V\}$ is a new set of distinct labels ($C \cap L = \emptyset$), and l^+ is defined as:*

$$l^+(e) = \begin{cases} l(e) & \text{if } e \in E \\ c_v & \text{if } e = (s, v), v \in V \end{cases}$$

Now we can state the equivalent characterization of the problem:

Proposition 1.2. *[Figueredo and Campêlo, 2020] Given an ELG $G = (V, E, L, l)$ and $k \in \mathbb{Z}^+$, the kLSFP is equivalent to finding $L^* \cup C^*$, such that $L^* \subseteq L$, $C^* \subseteq C$, $| L^* | \leq k$, the subgraph $G^+[L^* \cup C^*]$ is connected and $|C^*|$ is minimized.*

1.2. Integer linear programming formulation

Using Proposition 1.2, it is possible to formulate the flow model (M) for the kLSFP [Figueredo and Campêlo, 2020], which uses the following decision variables:

- $z_l = \begin{cases} 1 & \text{if the label } l \in L \cup C \text{ is in the solution} \\ 0 & \text{otherwise} \end{cases}$
- $f_{ij} \geq 0$: the amount of flow from i to j, where $i, j \in V(G^+)$

$$(M) \min \sum_{l \in C} z_l \tag{1}$$

$$\text{s.t: } f_{sj} + \sum_{i:(i,j) \in E} f_{ij} - \sum_{i:(i,j) \in E} f_{ji} = 1 \qquad \forall j \in V \tag{2}$$

$$0 \leq f_{ij} \leq M \cdot z_{l(i,j)} \qquad \forall (i,j) \in E \tag{3}$$

$$0 \leq f_{ji} \leq M \cdot z_{l(i,j)} \qquad \forall (i,j) \in E \tag{4}$$

$$0 \leq f_{sj} \leq M \cdot z_{l(s,j)} \qquad \forall j \in V \tag{5}$$

$$\sum_{l \in L} z_l \leq k \tag{6}$$

$$z_l \in \{0, 1\} \qquad \forall l \in L \cup C \tag{7}$$

The objective function (1) minimizes the number labels from C used in the solution. Equations (2) establish an s-branching flow from the source s that is distributed to all vertices.

Inequalities (3)-(5) ensure the following: if the label of a particular edge is chosen, the amount of flow f^* passing through that edge is $0 \leq f^* \leq M$, where in general $M = |V|$. However, if the label of a particular edge is not chosen, the flow passing through it is 0. Since equations (2) ensure that positive flow reaches every vertex, there exists a path between s and every vertex, guaranteeing connectivity. Inequality (6) ensures the upper limit on the number of colors in the kLSF.

2. Transformations on edge-labeled graphs

In this section, we explore the transformations that can be applied to the input graph without altering the problem. First, we need to introduce some concepts:

Definition 2.1. *An ELG $G = (V, E, L, l)$ is called monochromatic if all its edges have the same label, i.e. $|l(E)| = 1$. Therefore, for each $l \in L$, $G[\{l\}]$ is a monochromatic subgraph.*

Definition 2.2. *Given an ELG $G = (V, E, L, l)$, the set of vertices of each connected component of a monochromatic subgraph $G[\{l\}]$, for any label l, is called a monochromatic component of G.*

Proposition 2.1. *[Krumke and Wirth, 1998] Any monochromatic cycle of the input graph can be broken by arbitrarily choosing an edge and removing it from the graph.*

Based on this proposition, Krumke and Wirth [1998] proposed a pre-processing algorithm that breaks all monochromatic cycles in the input graph. However, besides breaking monochromatic cycles, other transformations can be applied to monochromatic components:

Proposition 2.2. *[Figueredo, 2020] Given two ELGs with the same monochromatic components, solving the kLSFP in either graph is equivalent.*

Proof. Let $G = (V, E, L, l)$ and $G' = (V, E', L, l')$ be two ELGs with same monochromatic components. Thus, for any subset of labels $L' \subseteq L$, $G[L']$ and $G'[L']$ have the same number of connected components, even if $E[L'] \neq E'[L']$. Therefore, the equivalence follows from Proposition 1.1. □

According to Proposition 2.2, as long as each monochromatic component is preserved, we can add or remove edges in G without altering the problem. This is illustrated in Figure 2, where some possible transformations in a monochromatic component are: a cycle, a star, a complete graph and a path.

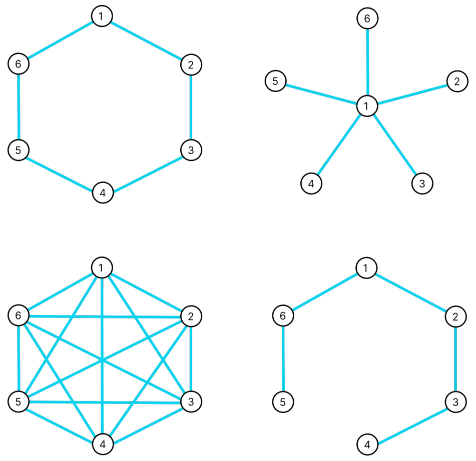

Figure 2: Possible transformations in a monochromatic component

DOI: 10.59254/sbpo-2024-193802
https://proceedings.science/p/193802?lang=pt-br

When such transformations related to different labels are applied, parallel edges with different labels may appear between a pair of vertices, converting the graph into a multigraph. This is illustrated in Figure 3. Rather than considering multiple edges each with a different label, we can treat these multiple edges as a single edge with multiple labels assigned to it.

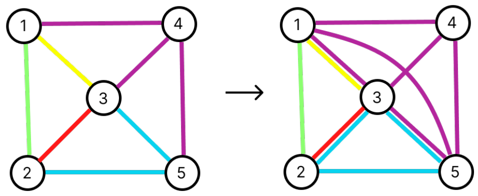

Figure 3: Transformation into a multigraph

With all this in mind, our main question was: is it possible to break more cycles than just the monochromatic ones using transformations in the monochromatic components? The answer is yes. Ideally, if we could break all cycles, the problem would become trivial [Figueredo, 2020]. We will show some examples of non-monochromatic cycles that can be broken and how to break them.

Suppose C is a cycle that is formed by two monochromatic paths, i.e. $l(E_C) = \{A, B\}$ and there exist paths P_1 and P_2 such that $l(E_{P_1}) = \{A\}$ and $l(E_{P_2}) = \{B\}$ and $P_1 \cup P_2 = C$. Let $\{u, v\} = V_{P_1} \cap V_{P_2}$. Adding the multi-labeled edge $e = (u, v)$ with labels $\{A, B\}$ is a valid transformation, since it does not change the monochromatic components. Removing one edge in P_1 and removing one edge in P_2 are also valid transformations. With this, we were able to break the cycle C, without adding any other cycle. This process is illustrated in Figure 4.

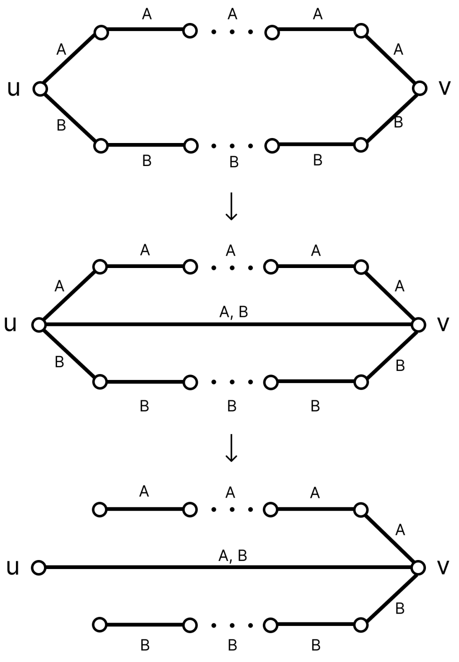

Figure 4: Breaking a non-monochromatic cycle

Essentially, it is possible to break more cycles than just the monochromatic ones by transforming the graph into a multigraph. We illustrate a more complex example in Figure 5. Suppose that we want to break all the cycles in the graph shown in Figure 5(a). We can start by breaking the cycle with blue, green, red, purple and yellow labels by removing the edge $(13, 14)$, obtaining the graph in Figure 5(b). Next, we can break the outermost monochromatic cycle with red label by removing the edge $(14, 19)$, obtaining the graph in Figure 5(c). Finally, we can break the remaining cycles in a similar way to what we showed earlier in Figure 4, obtaining the graphs in Figure 5(d) and Figure 5(e).

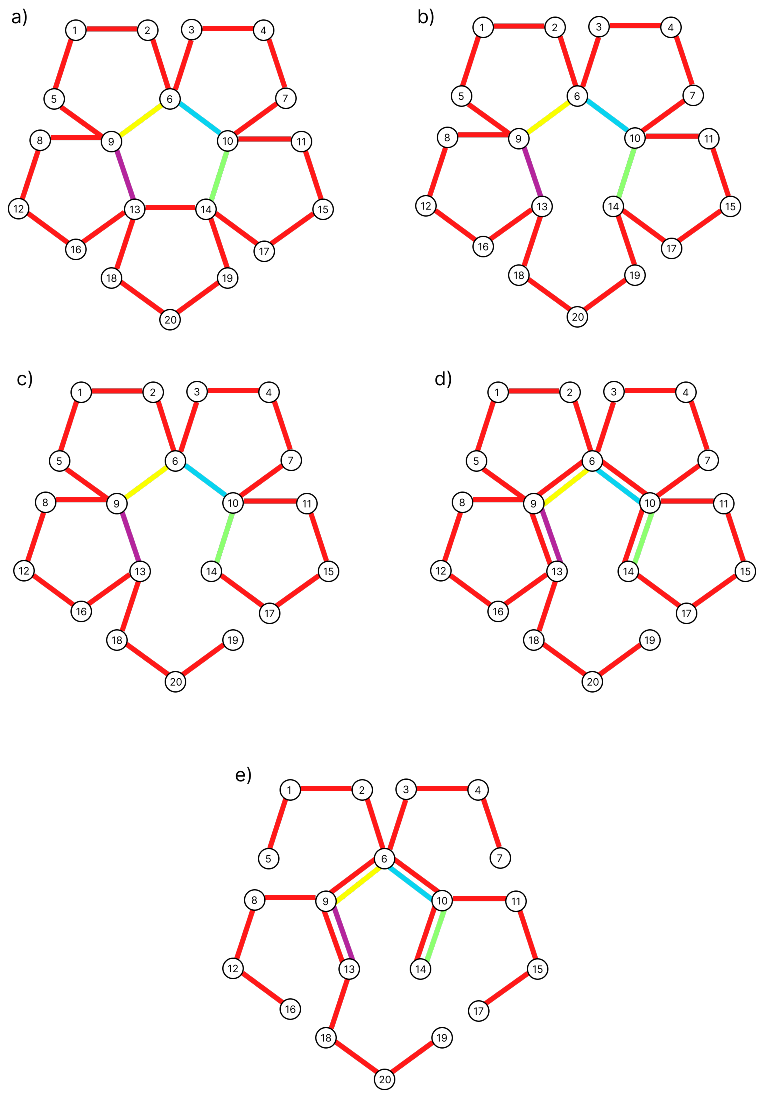

Figure 5: Breaking more non-monochromatic cycles

3. Pre-processing heuristic

In this section, we will describe our pre-processing heuristic. It works as follows: First, we create the graph G' by copying the input graph (line 1). Then, we transform each monochromatic component of G' into a clique (lines 2 to 13). Next, for each label, we apply an algorithm that transforms the spanning subgraph induced by the label into a maximum-weight spanning forest, where each edge has an associated weight equal to the number of labels on that edge in the graph G' (lines 14 to 20). Lastly, we return the reduced graph G' (line 21). It is worth observing that the multigraph resulting from the pre-processing heuristic preserves the same monochromatic components as the original graph.

Algorithm 1 Pre-processing heuristic

```
Input: An ELG $G = (V, E, L, l)$
Output: A reduced ELG $G' = (V, E', L, l')$
 1: $G' \leftarrow G$
 2: for each $label$ in $L$ do
 3:     for each CONNECTED_COMPONENT in $G'[\{label\}]$ do
 4:         for each $v_1$ in $V$(CONNECTED_COMPONENT) do
 5:             for each $v_2$ in $V$(CONNECTED_COMPONENT) do
 6:                 if $v_1 \neq v_2$ and $v_1 < v_2$ and $e = (v_1, v_2) \notin E(G'[\{label\}])$ then
 7:                     $E' \leftarrow E' \cup \{e\}$
 8:                     $l'(e) \leftarrow l'(e) \cup \{label\}$
 9:                 end if
10:             end for
11:         end for
12:     end for
13: end for
14: $w \leftarrow \emptyset$
15: for each $e$ in $E'$ do
16:     $w(e) \leftarrow |\, l'(e) \,|$
17: end for
18: for each $label$ in $L$ do
19:     MAXIMUM_SPANNING_FOREST($G'[\{label\}], w$)
20: end for
21: return $G'$
```

In Figure 6, we show an example of the algorithm's execution.

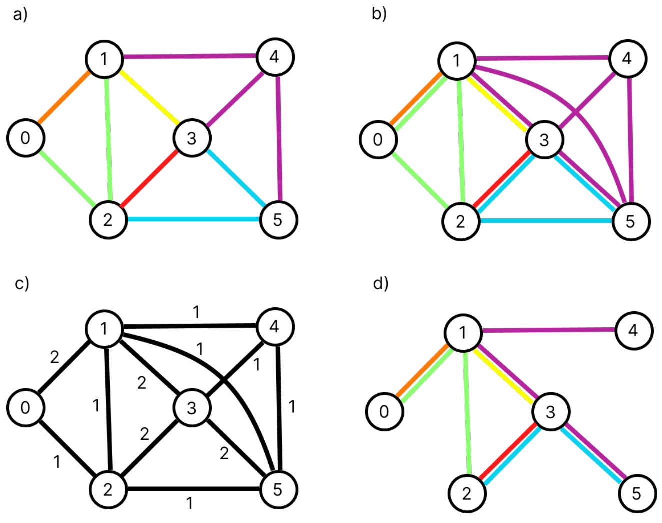

Figure 6: Example of the algorithm's execution

4. Adapting the Flow Model for multigraphs

After applying the pre-processing heuristic on the input graph, we obtain a multigraph. However, the Flow Model shown in Section 1 was defined for simple graphs, so we need to adapt it. The only difference is that now an edge can have multiple labels, so if at least one label of a particular edge was chosen, flow can pass through it; otherwise, the flow passing through it is zero. Therefore, we can define the following adapted flow model (M1) with the same decision variables:

$$(M1)\ \min \sum_{l \in C} z_l \tag{8}$$
$$\text{s.t: } f_{sj} + \sum_{i:(i,j)\in E} f_{ij} - \sum_{i:(i,j)\in E} f_{ji} = 1 \qquad \forall j \in V \tag{9}$$
$$0 \le f_{ij} \le M \cdot \sum_{l \in l(i,j)} z_l \qquad \forall (i,j) \in E \tag{10}$$
$$0 \le f_{ji} \le M \cdot \sum_{l \in l(i,j)} z_l \qquad \forall (i,j) \in E \tag{11}$$
$$0 \le f_{sj} \le M \cdot \sum_{l \in l(s,j)} z_l \qquad \forall j \in V \tag{12}$$
$$\sum_{l \in L} z_l \le k \tag{13}$$
$$z_l \in \{0,1\} \qquad \forall l \in L \cup C \tag{14}$$

A possible drawback of this model is the fact that, when an edge has many labels, the big-M will be multiplied by a potentially bigger value in inequalities (10)-(12) . A way of getting around this is defining new decision variables y_{ij}, where $i,j \in V(G^+)$, which will be the new multipliers of the big-M. This leads to the adapted flow model (M2):

$$(M2)\ \min \sum_{l \in C} z_l \tag{15}$$
$$\text{s.t: } y_{ij} \le \sum_{l \in l(i,j)} z_l \qquad \forall (i,j) \in E \tag{16}$$
$$y_{ji} \le \sum_{l \in l(i,j)} z_l \qquad \forall (i,j) \in E \tag{17}$$
$$y_{sj} \le \sum_{l \in l(s,j)} z_l \qquad \forall j \in V \tag{18}$$
$$f_{sj} + \sum_{i:(i,j)\in E} f_{ij} - \sum_{i:(i,j)\in E} f_{ji} = 1 \qquad \forall j \in V \tag{19}$$
$$0 \le f_{ij} \le M \cdot y_{ij} \qquad \forall (i,j) \in E \tag{20}$$
$$0 \le f_{ji} \le M \cdot y_{ji} \qquad \forall (i,j) \in E \tag{21}$$
$$0 \le f_{sj} \le M \cdot y_{sj} \qquad \forall j \in V \tag{22}$$
$$\sum_{l \in L} z_l \le k \tag{23}$$
$$z_l \in \{0,1\} \qquad \forall l \in L \cup C \tag{24}$$
$$0 \le y_{ij} \le 1 \qquad \forall (i,j) \in E^+ \tag{25}$$

5. Computational experiments

In this section, we present the computational experiments conducted to evaluate and compare the computational times of adapted flow models M1 and M2, before and after pre-processing. Subsequently, we perform a statistical analysis to determine which approach is better.

5.1. Computing environment

The models were implemented using the JuMP.jl and CPLEX.jl libraries from the Julia 1.10.3 programming language as an interface for the CPLEX 22.1 solver. In the implementation of the models, the greedy heuristic kMVCA was used to obtain an initial upper bound, and the valid inequalities (4.7) and (4.8) from Figueredo [2020] were added. The pre-processing heuristic was also implemented in Julia 1.10.3. The experiments were executed on a computer with an AMD Ryzen 7 4800HS, 2.90 GHz, 16 GB of RAM, running Windows 11 Home 64-bit. The algorithms utilized up to 8 threads.

5.2. Instances

We used 130 instances from group 2 of the ELGs generated by Cerulli et al. [2014]. The parameters of the ELGs are: number of vertices $n = |V| \in \{100, 200\}$, edge densities $d \in \{0.2, 0.5, 0.8\}$, number of labels $|L| \in \{25, 50, 100, 125\}$ and $k \in \{3, 6, 7\}$. Each dataset consists of 10 different graphs for each $n - |L| - d - k$ configuration.

5.3. Results

The results of the experiments are presented in Table 1. The first column represents a dataset with configuration $n - |L| - d - k$, the second column reports the average optimal value of the dataset, and the other columns report the the average running time (in seconds) for the dataset in M1 before pre-processing, M1 after pre-processing, M2 before pre-processing, and M2 after pre-processing, respectively. Note that M1 before pre-processing is exactly the original model M by Figueredo and Campêlo [2020]. The best results of each row are in bold.

Table 1: Results of the experiments

Dataset				opt	M1 before	M1 after	M2 before	M2 after
n	d	$\|L\|$	k		time (s)	time (s)	time (s)	time (s)
100	0.2	25	3	6.3	3.272	1.478	3.247	**1.218**
100	0.2	50	6	2.6	14.498	6.548	10.853	**5.369**
100	0.2	100	6	15.0	14.797	**13.354**	15.517	17.548
100	0.2	125	7	15.3	**31.814**	62.910	36.880	67.992
100	0.5	25	3	1.0	0.167	**0.053**	0.192	0.058
100	0.5	50	6	1.0	0.179	**0.093**	0.198	**0.093**
100	0.5	100	6	1.0	1.318	**0.408**	1.340	0.532
100	0.5	125	7	1.0	0.812	**0.488**	0.854	0.809
100	0.8	25	3	1.0	0.222	0.045	0.248	**0.044**
100	0.8	50	6	1.0	0.323	**0.111**	0.338	0.114
100	0.8	100	6	1.0	0.321	**0.148**	0.336	0.291
100	0.8	125	7	1.0	0.401	**0.175**	0.409	0.214
200	0.2	50	3	17.0	231.821	**23.336**	169.767	39.754

Observing the table, we notice that in almost all rows the pre-processing improved the running time for both model M1 and model M2. Besides, M1 performs slightly better than M2. In Figure 7, we also present a box plot of the running times for all instances. Upon analyzing the box plot, it seems evident that both M1 and M2 benefit from the pre-processing in most of the instances. However, a statistical analysis is necessary to draw a definitive conclusion.

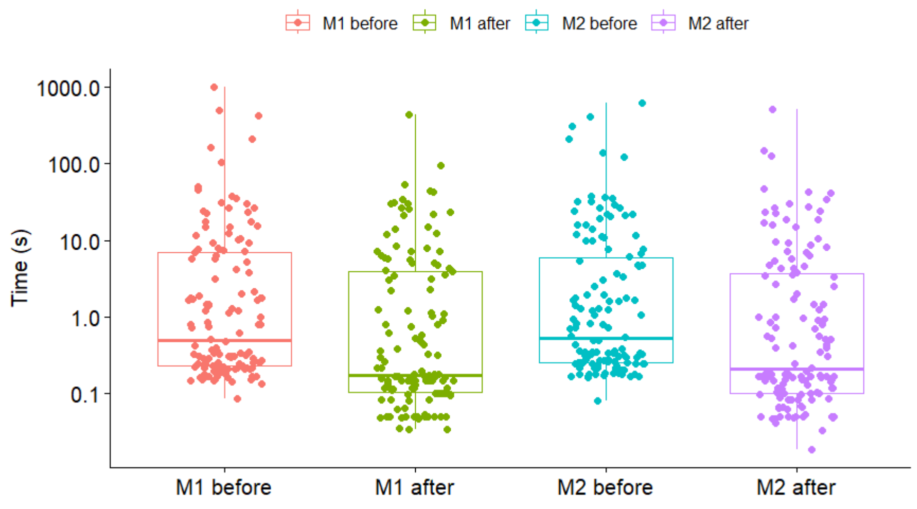

Figure 7: Box plot of running times for all instances

5.4. Statistical analysis

To conduct the statistical analysis, we relied on a similar analysis conducted by da Silva et al. [2019] for the minimum labeling spanning tree problem (MLSTP), which, in turn, was based on the recommendations of Demšar [2006] regarding which statistical tests to use for comparing algorithms over multiple datasets. Our tests were performed using the software R version 4.4.0 and the PMCMRplus package.

For the first test, the null hypothesis is that there is no significant statistical difference between the methods, with any observed deviations being the result of chance. To verify this hypothesis, we used the Friedman test. The ranking of the methods for the Friedman test was calculated from Table 1, where each method received a rank from 1 to 4 for each dataset. Using the `friedman.test` method, we obtained $\chi^2_F = 21.462$ and p-value = 0.00008442. Therefore, considering a significance level $\alpha = 0.01$, the null hypothesis can be rejected with 99% confidence.

To detect the significant differences between the methods on a pairwise comparison, we used a post-hoc test. Specifically, we employed the Nemenyi test using `frdAllPairsNemenyiTest` method from PMCMRplus. The p-values obtained from the Nemenyi test are shown in Table 2.

Table 2: p-values obtained from the Nemenyi test

	M1 after	M1 before	M2 after
M1 before	0.00776	-	-
M2 after	0.52003	0.26243	-
M2 before	0.00012	0.71187	0.02036

The p-values indicate that the performance of M1 after pre-processing is significantly better than both M1 and M2 before pre-processing at a significance level of $\alpha = 0.01$ and 99% confidence. We can also see that the performance of M2 after pre-processing is significantly better than M2 before pre-processing at a significance level of $\alpha = 0.05$ and 95% confidence. Furthermore, there is no statistically significant difference between M1 and M2 both before pre-processing, between M1 and M2 both after pre-processing, or between M1 before pre-processing and M2 after pre-processing.

DOI: 10.59254/sbpo-2024-193802
https://proceedings.science/p/193802?lang=pt-br

6. Concluding remarks

We have proposed a pre-processing heuristic for the k-labeled spanning forest problem (kLSFP). As the input graph may become a multigraph (or equivalently, some edges may be assigned more than one label), we have adapted the flow model by Figueredo and Campêlo [2020] to deal with such generalization. By the conducted computational experiments, we can conclude that the pre-processing heuristic was effective in improving the computational running time for solving kLSFP.

Acknowledgment Yasser Djalo was supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq 404479/2023-5). Manoel Campêlo was partially supported by: Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq 442977/2023-9, CNPq 404479/2023-5, CNPq 312417/2022-5), and Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico (PS1-0186-00155.01.00/21, PGP-0192-00040.01.00/22, ITR-0214-00096.01.00/23). This support is gratefully acknowledged.

References

Cerulli, R., Fink, A., Gentili, M., and Raiconi, A. (2014). The k-labeled spanning forest problem. *Procedia - Social and Behavioral Sciences*, 108:153–163. ISSN 1877-0428. Operational Research for Development, Sustainability and Local Economies.

da Silva, T. G., Queiroga, E., Ochi, L. S., dos Anjos Formiga Cabral, L., Gueye, S., and Michelon, P. (2019). A hybrid metaheuristic for the minimum labeling spanning tree problem. *European Journal of Operational Research*, 274(1):22–34. ISSN 0377-2217.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30. ISSN 1532-4435.

Figueredo, P. J. d. A. (2020). O problema da floresta geradora k-rotulada. Master's thesis, Universidade Federal do Ceará, Fortaleza.

Figueredo, P. J. d. A. and Campêlo, M. (2020). O problema da floresta geradora k-rotulada. In *Anais do Simpósio Brasileiro de Pesquisa Operacional*, João Pessoa.

Krumke, S. O. and Wirth, H.-C. (1998). On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81–85. ISSN 0020-0190.

Pinheiro, T. F. D. (2022). The k-labeled spanning forest problem : complexity, approximability, formulations and algorithms. Master's thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre.