LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

The k-labeled spanning forest problem: instance analysis and effective heuristic solution

Marcus Ritt
Instituto de Informática, Dept. de Informática Teórica, UFRGS
Av. Bento Gonçalves, 8500, Porto Alegre, Brazil
marcus.ritt@inf.ufrgs.br

ABSTRACT

We consider the problem of selecting a fixed number of labels from an edge-labeled, undirected graph such as to minimize the number of connected components in the graph over the edges of the selected labels. We contribute some theoretical considerations, an analysis of instances from the literature, and propose a new heuristic solution procedure for this problem combining a form of GRASP with a cyclic best-first search (CBFS). In computational experiments, we derive some optimal solutions for instances from the literature and compare to exact and heuristic solution procedures from the literature. The experiments suggest that the combination of GRASP and CBFS performs favorably compared to existing solution approaches.

KEYWORDS. **k-labeled spanning forest. GRASP. CBFS.**

Paper topics (indicate in order of PRIORITY the paper topic(s))

- **OC – Combinatorial Optimization**
- **MH – Metaheurístics**

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

1. Introduction

Consider an edge-labeled, undirected graph $G = (V, E)$ with labels $l : E \to L$, where L is the set of labels. (For a non-negative integer n, $[n]$ denotes the set $1, 2, \ldots, n$.) For a given number of labels k, the k-labeled spanning forest (kLSF) problem is to choose at most k from labels L, such that the graph with only edges of chosen labels has the minimal number of connected components. More formally, let $E_l \subseteq E$ be the set of edges labeled with $l \in L$, and $E[S] = \bigcup_{l \in S} E_l$ the edges of labels $S \subseteq L$. Denote by $G[S] = (V, E[S])$ the graph over edges of labels S, and $c(G)$ the number of connected components of a graph G. Then, we seek a set

$$S^* = \operatorname*{argmin}_{S \in \binom{L}{k}} c(G[S]), \tag{1}$$

where $\binom{L}{k}$ is the set of all k-subsets of L. Note that the minimum can be always achieved choosing exactly k labels, i.e. we need not consider cases with less than k labels.

A related problem is the minimum labeling spanning tree (MLST), where we seek to choose the smallest number of labels such that the graph is connected. The MLST can be reduced to the kLSF in polynomial time, by a binary search for the smallest k such that minimization (1) yields a set S with $c(G[S]) = 1$. This problem has obvious applications in communication problems, where we seek to maximize the possibility of communication subject to using only a limited number of communication types (e.g. frequencies).

Since $|\binom{L}{k}| = \binom{|L|}{k} = \Theta(|L|^k)$ kLSF and MLST clearly can be solved in polynomial time for fixed k, and thus are fixed-parameter tractable for parameter k. With k being part of the input kLSF is NP-hard, since the decision version of MLST (i.e. to decide if there is a subset of at most k labels that makes a graph connected) can be (Karp-)reduced to the decision version of kLSF (i.e. to decide if there is a subset of at most k labels that yields a graph of at most c components) [Cerulli et al., 2014].

For completeness we give here a mixed-integer programming model for the problem introduced by Figueredo and Campêlo Neto (2020). It works on a auxiliary graph G' that has an additional, new vertex s connected to every vertex $v \in V$ by a new edge with a new label l'_v. Let L' be the set of all new labels. Then minimizing the number of components in G is equivalent to minimizing the selected labels in L' in G' such that the graph is connected, and such that no more than k labels from L are selected. This leads to the following model that has for each label $l \in L \cup L'$ a decision variable $z_l \in \{0, 1\}$.

$$\begin{aligned}
\text{(C):}\quad & \textbf{minimize} && \sum_{l \in L'} z_l && && (2)\\
& \textbf{subject to} && \sum_{l \in L} z_l \leqslant k, && && (3)\\
& && \sum_{l \in L(\delta(S))} z_l \geqslant 1, && \forall \varnothing \subset S \subset V \cup \{s\}, && (4)\\
& && z_l \in \{0, 1\}, && \forall l \in L \cup L'. && (5)
\end{aligned}$$

In this model (2) minimizes the number of auxiliary labels (and thus edges), and constraint (3) limits the number of selected edges from L to at most k. Constraint set (4) are connectivity constraints that require for each non-empty and non-complete subset of vertices S to select at least one label that contains an edge connecting it to the remainder of the graph, where $\delta(S) = \{e \in E' \mid |e \cap S| = 1\}$ are edges incident to S, and $L(E) = \{l_e \mid e \in E\}$ the labels of edge set E. Note that constraint set (4) has size 2^{n-1}, so model (C) has to be solved by cutting plane techniques.

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Table 1: Overview on the literature about the kLSF.

Reference	Methods
Cerulli et al. (2014)	GC, PM, TS, SAnn, GA
Consoli, Pérez, and Mladenović (2017)	PM, GA, GRASP, VNS
Figueredo and Campêlo Neto (2020)	MIP
Pinheiro et al. (2022)	MIP, FO

FO: Fix-and-optimize heuristic. GA: Genetic algorithm. GC: greedy construction. MIP: Mixed-integer programming model. PM: pilot method. SAnn: Simulated Annealing. TS: Tabu search. VNS: Variable neighborhood search.

In the remainder of this paper we first discuss related work in Section 2, then propose a new heuristic solution for the problem in Section 3, which will be evaluated experimentally in Section 4. We conclude in Section 5.

2. Related work

The kLSF has been introduced by Cerulli et al. (2014), who have also shown its NP-completeness, and introduced several heuristic solutions and an exact solution procedure. The heuristics are a greedy construction, an extended greedy construction using the Pilot method, and the application of three meta-heuristics, namely tabu search, simulated annealing, and genetic algorithms. The greedy construction repeatedly selects the next free label that minimizes the number of connected components. A corresponding heuristic for the MLST has been proposed by Chang and Shing-Jiuan (1997) under the name Maximum Vertex Coverage Algorithm (MVCA), and the difference here is only to stop after k labels have been selected. The Pilot method [Voß et al., 2005] is an extension of a greedy construction that chooses the next element among all candidates elements not by a simple greedy criterion, but by constructing a complete solution greedily and choosing the candidate of best solution value. The heuristics based on local search are mostly standard and use a neighborhood that exchanges a selected with an unselected label. We will call this neighborhood the swap neighborhood $\mathcal{N}_s$. Here $s \geqslant 1$ denotes the number of exchanged labels. In $\mathcal{N}_1$ each solution has $k(|L| - k)$ neighbors. The genetic algorithm recombines solutions by applying the greedy algorithm to their joint labels, and mutates them by adding a random unselected label, and dropping a previously selected one which leads to the least increase of the number of components.

A series of further heuristics have been proposed to solve the kLSF. They are listed in Table 1 together with the proposed solution method. We discuss them here briefly, with a focus on novel, or problem-specific ideas.

Consoli, Pérez, and Mladenović (2017) present a comparison of several heuristics methods. These include the above-mentioned genetic algorithm and the Pilot method. They further introduce a greedy adaptive randomized search procedure (GRASP, Feo and Resende (1995)) and a variable neighborhood search (VNS, Mladenović and Hansen (1997)). This paper also subsumes a set of previous papers of the same content [Consoli and Pérez, 2015b; Consoli and Pérez, 2015a; Consoli, Pérez, and Mladenović, 2015]. The GRASP is canonical: it first selects a random label, applies the greedy algorithm explained above to obtain a complete solution, and then applies a local search that drops a label and replaces it greedily by the label the yields the least number of components, equal to the construction phase. The basic VNS is similarly canonical: a shake is a random move from the neighborhood $\mathcal{N}_s$ described above, and the local search is the same as used in the GRASP.

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Consoli, Pérez, and Mladenović (2017) propose an extension of this VNS, dubbed "complementary" and "intelligent" VNS. This means the following. Given a solution $S \subseteq L$ with $|S| = k$ labels, the complementary search constructs greedily a solution using only labels from $L \backslash S$. Its "intelligent" part amounts to select not one of the best labels greedily, but according to a Boltzmann distribution in the difference of the number of components. Specifically, for a current solution P, let l^* be a label that minimizes the number of components $c(G[P \cup \{l^*\}])$ and l any other label. Then, in the greedy procedure above, label $l \in L \backslash S$ is selected with probability proportional to $e^{-\Delta(l)}$, where $\Delta(l) = c(G[P \cup \{l\}]) - c(G[P \cup \{l^*\}])$ is the increase in the number of components when using label l instead of label l^*. The overall VNS using this procedure then is structured as follows. Given a current solution, it repeatedly constructs a complementary solution using the above procedure while this leads to a new incumbent solution. Then follows the VNS phase, as described above. This phase uses an extended neighborhood $\mathcal{N}_s$ for shaking: for $s \leqslant k$, s random elements are removed, for $s > k$ a random solution with $s - k$ elements is generated. In both cases the solution is then rebuilt greedily. Finally, if the current incumbent could be improved in an iteration, the maximum shake is decreased, and otherwise increases, but always kept within $[k/2, k]$.

Figueredo and Campêlo Neto (2020) and Abreu Figueredo (2020) introduce two mathematical models, one based on colored cuts, which is the model (C) presented above, and a flow model, as well as a parallel version of the backtracking search of Cerulli et al. (2014). They also show that the kLSF can be reduced to the MLST by introducing auxiliary graph explained above with the description of MIP model (C), with the additional restriction that at most k of the original colors can be used.

Pinheiro et al. (2022) propose a matheuristic to solve the problem. It starts from a greedy solution S and tries to improve it by fixing some labels from S, and optimizing over a reduced set of remaining labels. More specifically, a random subset $L_1 \subseteq S$ of size $\rho_1|S|$ of labels is fixed (i.e. selected). From the remaining labels another random subset $L_2 \subseteq S \backslash L_1$ of size $\rho_2|S|$ is selected, and joined with a random subset L_3 of the unselected labels $U = L \backslash S$ of size $\rho_3|U|$. All other labels are dropped, and the MIP model (C) presented above is solved to find the subset of $L_2 \cup L_3$ that together with L_1 leads to the smallest number of components. A solution is accepted if it improves the current solution or after $2k$ iterations. This is repeated for $75k$ iterations. (Parameters are set to $\rho_1 = 0.8$, $\rho_2 = \min\{0.8, |L|/n\}$, and $\rho_3 = 0.8$.)

3. Heuristic solution

A first step in the heuristic solution is pre-processing. There are two obvious instance reductions. First, since all edges of a certain label will be selected, it is enough for connectivity to consider a spanning forest of these edges. These can be found in time $O(k(n + m))$, and as a consequence a pre-processed graph can have at most $(n - 1)|L|$ edges. Second, we can define a notion of label dominance, namely for label sets $L_1, L_2 \subseteq L$ we say that L_1 dominates L_2 if $G[L_1]$ is a coarsening of $G[L_2]$, in the sense that each pair of vertices u, v that is connected in $G[L_2]$ is also connected in $G[L_1]$. Two sets of labels are equivalent if they dominate each other. Of particular interest are dominations between single labels. These will be analyzed in the experimental part.

We now propose two heuristic algorithms, a GRASP [Feo and Resende, 1995] and a cyclic best-first search (CBFS, Kao et al. (2009)). A GRASP has two phases, a randomized construction followed by a local search. For the randomized construction, it repeatedly constructs a so-called restricted candidate list (RCL), and selects a random element from that list. In our case, the RCL contains the first κ unselected labels in order of increasing number of resulting connected components when added to the current partial solution. Ties are broken by choosing the label with the larger number of edges. The following local search phase uses the neighborhood $\mathcal{N}_1$ and accepts the first improvement. GRASP repeats this for a fixed number of iterations. We use $\kappa = 1$ in

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

the first iteration. This GRASP has three important differences to the GRASP proposed earlier by Consoli, Pérez, and Mladenović (2017). First, it does not select the first label randomly, which produces much more varied solutions, and seems to be a mistake; second, it uses a cardinality-constrained RCL, instead of only selecting among the candidates that lead to the least number of connected components; and third, the local search does not greedily drop a single label, and rebuild the solution, but considers all exchanges of a selected and an unselected label.

We now turn to the cyclic best-first search. CBFS is a tree search, with a specific search order that can be also used in backtracking or a branch-and-bound algorithms. For the kLSF we consider the constructive space of all partial solutions, organized as a search tree, with the empty partial solution at its root. In this tree, each solution has as its children all partial solutions with an additional label. To avoid symmetries, only labels higher than the highest current label are added. The leafs of this tree are complete solutions with k labels. This search tree is explored in a cyclic best-first order. This means that cyclically partial solutions with $\kappa = 0, 1, 2, \ldots, k-1$ labels are visited, if any. From the current solutions with κ labels, the solution of the least number of components is chosen, with ties broken by selecting the solution with the highest number of edges. This solution is removed, and its descendants with $\kappa + 1$ labels are generated. Thus, in each complete pass of (at most) k iterations, CBFS produces a complete solution. Algorithmically, this is implemented by maintaining priority queues for each level κ, removing in each iterations the best element from the priority queue in level κ and adding its descendants to the priority queue in level $\kappa+1$. Note that CBFS in this form is complete and will eventually find the optimal solution. We use CBFS in a heuristic manner, with a limited number of passes $\overline{p}$ and return the best solution found. We also apply a local search in neighborhood $\mathcal{N}_1$ whenever a new best solution is found. When passes are limited to $\overline{p}$, we can also truncate all priority queues to at most $\overline{p}$ elements, to reduce the memory requirements. The complete steps are given in Algorithm 1.

initialize priority queues $q_0 = \{\varnothing\}, q_1 = \cdots = q_{k-1} = \varnothing$
maintain the best solution S^* during the search
$\kappa := 0$
while $q_\kappa \neq \varnothing$ **do**
  $S :=$ deletemin(q_κ)
  $\chi := \max S$
  $S^+ := \{S \cup \{l\} \mid l \in L \backslash S \backslash [\chi]\}$
  if $\kappa + 1 = k$ **then**
    evaluate all solutions in S^+ and update the best solution S^*
    apply a local search in $\mathcal{N}_1$ to a new best solution, if any
    stop if $c(G[S^*]) = 1$
  else
    insert all solutions in S^+ into $q_{\kappa+1}$
    truncate $q_{\kappa+1}$ to at most $\overline{p}$ solutions
  advance κ cyclically to the next value such that $q_\kappa \neq \varnothing$, if any
  stop after $\overline{p}$ cycles

Algorithm 1: A cyclic best-first search for finding all valid labelings.

Finally, we propose to combine both algorithms as follows. We first run GRASP for I iterations, followed by CBFS with $\overline{p} = I$ complete passes. We set $I = \iota \log \binom{|L|}{k}$ for some multiplier ι. This is done to account for different search spaces sizes $\binom{|L|}{k}$.

In terms of the implementation of the above algorithms, evaluating the objective function,

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Table 2: Characteristics of the instances: number of vertices n, number of edges m, label factor λ, number of labels k, estimated number of labels $\hat{k}$, and edge reduction Δm (in percent).

n	m	λ	k	$\hat{k}$	Δm	n	m	λ	k	$\hat{k}$	Δm
100	990	0.25	3	3	1.29	200	3,980	1.00	12	12	0.02
100	990	0.50	6	6	0.11	200	3,980	1.25	15	15	0.02
100	990	1.00	6	12	0.01	400	15,960	0.25	3	3	0.53
100	990	1.25	7	15	0.01	400	15,960	0.50	6	6	0.04
150	2,235	0.25	4	4	1.48	400	15,960	1.00	12	12	0.01
150	2,235	0.50	4	4	0.14	400	15,960	1.25	15	15	0.00
150	2,235	1.00	9	9	0.04	500	24,950	0.25	4	3	0.68
150	2,235	1.25	11	11	0.01	500	24,950	0.50	7	7	0.05
200	3,980	0.25	3	3	1.30	500	24,950	1.00	15	15	0.01
200	3,980	0.50	6	6	0.12	500	24,950	1.25	19	9	0.00

i.e. determining the number of components, is one of the most expensive steps. Instead of running a BFS on selected edges, it can be sped up by using a union-find data structure, that when adding a label l, applies the union operation to all edges in E_l as observed by e.g. Chwatal and Raidl (2010). This is convenient in constructive algorithms, but for removing a label, for example in a local search, the data structure has to be rebuilt. This could be avoided by dynamic connectivity algorithms (see e.g. Hanauer et al. (2022)), but the practical performance gains of using such data structures remains open.

4. Computational Experiments

In this section we present the instances used, the experimental methodology, and discuss the results of some computational experiments.

4.1. Instances: characteristics, model, pre-processing

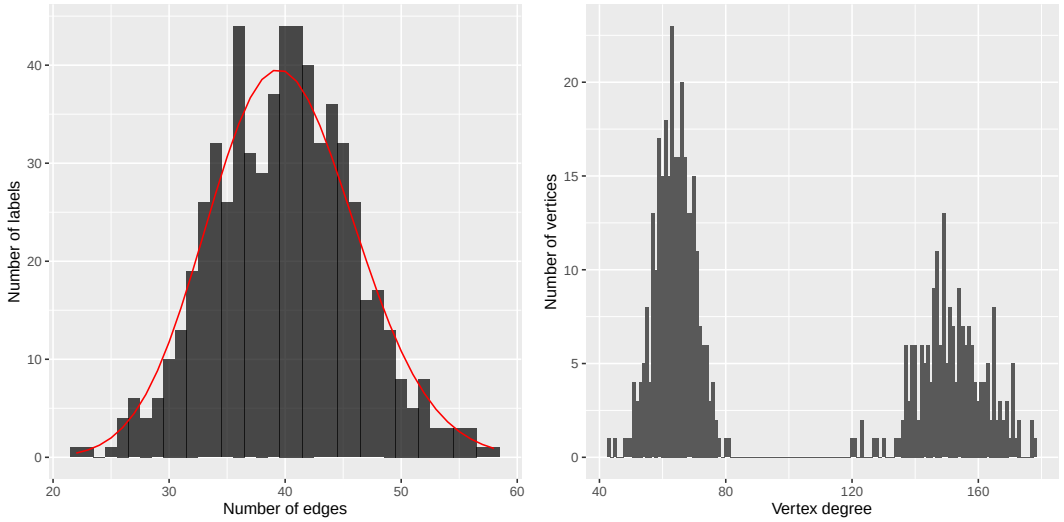
Existing instances for the kLSF have been generated randomly, according to the literature, following the Erdős–Rényi (ER) random graph model $G(n, M)$ which selects uniformly at random one of the graphs with n vertices and M edges. These graphs have been generated with a fixed density d, and therefore with $M = d\binom{n}{2}$ edges. The instances of Cerulli et al. (2014) have $n \in \{100, 150, 200, 400, 500\}$ vertices, density $d = 0.2$, and total number of labels $|L| = \lfloor \lambda n \rfloor$, with $\lambda \in \{0.25, 0.50, 1.00, 1.25\}$; edges are randomly assigned a label from L. For each of the $5 \times 4 = 20$ combinations of n and λ, ten instances have been generated, for a total of 200 instances. We further use 43 selected instances from Figueredo and Campêlo Neto (2020). These instances also have $\lambda \in \{0.25, 0.50, 1.00, 1.25\}$, but $n \in \{100, 200, 300, 400, 500\}$ and densities $d \in \{0.2, 0.5, 0.8\}$ in different combinations. These two instance sets are called "Set 1" and "Set 2" in the following.

For all instances the number of available labels has been set to $k = \lfloor |L|/2^i \rfloor$ where i is the smallest integer such that the greedy algorithm discussed in Section 2 does not produce a single component, as proposed by Cerulli et al. (2014). The motive for this is to avoid instances that are trivially solvable. (A problem with this choice is that the number of components $c(G[S^*])$ in an optimal solution S^* varies significantly depending on how close to a threshold value of $|L|/2^i$ the number of labels required to produce a single component is, as can be seen in the experimental results below.)

Since the graphs are random, a very simple model can predict the number of available labels k. We know from Janson et al. (1993) that $n/2$ edges are a threshold for ER graphs: if we have $(1+\mu)n/2$ edges with $\mu > 0$ we have almost surely a unique giant component of asymptotic size αn for $\mu = -\alpha^{-1} \ln(1-\alpha) - 1$. For example, for the largest component to almost surely have $\alpha = 1 -$

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Figure 1: Exemplary statistics for instance for instance "500_24950_625_19_1" [Cerulli et al., 2014]. Left: Number of labels per number of edges. Right: Distribution of vertex degrees.



$e^{-4} \approx 98\%$ of the vertices, we need $\mu \approx 3.07$ and thus $M_\mu \approx 2.04n$ edges. Furthermore, since each edge receives random label, the number of edges of a fixed label l follows a binomial distribution $M_l = |E_l| \sim B(M; p)$, with $p = |L|^{-1}$. Thus for any label l the probability of having x edges is $\Pr(x) = \binom{m}{x} p^x (1-p)^{m-x}$, and we can expect overall $|L| \Pr(x)$ labels with x edges. From this, the number of labels to reach M_μ edges, and thus almost surely a single component can be determined. Namely define $M : [0, |L|] \to [0, m]$ by $M(l_i) = m_i$, for $0 \leqslant i \leqslant m$, where $l_i = \sum_{0 \leqslant j \leqslant i} |L| \Pr(j)$ and $m_i = \sum_{0 \leqslant j \leqslant i} j |L| \Pr(j)$, and interpolate linearly between points (l_i, m_i). Function $M(x)$ gives the (interpolated) number of edges when selecting x labels in increasing number of expected edges. Since M is monotone, it has an inverse and $k_1 = \left\lceil |L| - M^{-1}(m - M_\mu) \right\rceil$ is the expected number of labels needed to select at least M_μ edges, when selecting labels in decreasing number of expected edges. Therefore we expect that the process above will choose $k = \left\lfloor |L|/2^i \right\rfloor$ such that $\left\lfloor |L|/2^i \right\rfloor < k_1$, i.e. $i = \lfloor \log_2 |L|/k_1 \rfloor + 1$. Column $\hat{k}$ in Table 2 shows that the estimates obtained in this manner are in good agreement with the actually chosen values of k, with exception of three instances, where we consequently observe the double or the half of the actually chosen values of k.

We finally turn to pre-processing. In the instances we have, dominance has a negligible effect, since the labels partition the edges. Reducing labels to spanning forests makes a difference in 146 of 200 instances. Column Δm in Table 2 shows the reduction in the number of edges (in percent). We can see that reduction is small and in average removes only about 0.3 % of the edges. The reason for this is that most labels have few edges. A typical distribution is shown in Figure 1. Note that the number of edges per label does not exceed 60 for a graph with 500 vertices.

As a final note we add that the graphs proposed by Cerulli et al. (2014) actually are not ER random graphs, as can be seen in the right part of Figure 1 which shows an example of a degree distribution. All graph graphs generated by Cerulli et al. (2014) show a similar bimodal distribution. It is not clear how these graphs have been generated. The graphs introduced by Figueredo and Campêlo Neto (2020) do not show this behaviour.

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Table 3: Instances of Set 1 that can be solved by exhaustion. Average number of components c, and time to solve t compared to a branch-and-bound approach.

				Enum		B&B	
n	m	λ	k	c	t (s)	C	t (s)
100	990	0.25	3	6.3	0.00	6.3	0.05
100	990	0.50	6	2.6	20.21	2.6	4.59
100	990	1.00	6	–	–	15.0	0.25
100	990	1.25	7	–	–	15.7	0.57
150	2,235	0.25	4	3.5	0.12	3.5	0.85
150	2,235	0.50	4	22.3	1.50	22.3	0.76
150	2,235	1.00	9	–	–	7.1	1,518.00
200	3,980	0.25	3	17.0	0.04	–	–
400	15,960	0.25	3	35.6	0.67	–	–
500	24,950	0.25	4	22.4	59.69	–	–

4.2. Methodology

We have implemented all algorithms in C++. They have been compiled with GCC 13.2 and maximum optimization. The experiments have been run on a PC with a Core i7-1360P CPU and 16 GB of main memory running Ubuntu 24.04, using only a single processor in each experiment. Source code and detailed experimental data will be made available at `https://github.com/mrpritt/klsf`. For the GRASP phase of the heuristic, we have chosen a limit for the RCL of $\kappa = 3$. The number of iterations in both phases uses a multiplier $\iota = 10$.

4.3. Exact solution by enumeration

A weakness of the instances from the literature is that many have few labels combined with a low number k of labels to select. As mentioned in the introduction, for a fixed number of labels, the problem is fixed-parameter tractable. Namely, we can enumerate all possible solutions, when the number of combinations is relatively low. This is clearly not a good approach to solve the problem, but allows to eliminate trivial instances, and gauge the effectiveness of other exact approaches to the problem. Therefore, in this section we give some results on this kind of exhaustive solution. We use the "revolving door" algorithm to enumerate all $\binom{|L|}{k}$ possible combinations of labels [Knuth, 2011, Algorithm 7.2.1R].

Tables 3 and 4 present the results on instance sets 1 and 2. In Table 3 we show only instance groups for which either the enumeration or the branch-and-bound approach of Cerulli et al. (2014) has obtained solutions. Each group is identified by the number of vertices n, edges m, label factor λ, and number of labels k. In Table 4 we list only the 24 instances where the solutions could be enumerated and compare to the overall best approach CCut4 (a branch-and-cut algorithm) of Figueredo and Campêlo Neto (2020). Since these are single instances they are additionally identified by the replication number r. Both tables show the (mean) number of components c, and the solving time t in seconds.

We can see that enumeration, as expected, solves smaller instances, but is not competitive when a large number of solutions exists. However, we also find that 70 of the 200 instances of Set 1, and 23 of the 43 instances of Set 2 can be solved by a simple enumeration in a very short time.

We finally add a note on the solution value distribution. Figure 2 shows for two exemplary instance groups the distribution of solution values over all feasible solutions obtained by the enumeration above. To compensate for a different number of optimal components, solution values are expressed as a standard score z, i.e. $z = (c - \mu_c)/\sigma_c$, where μ_c is the average number of

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Table 4: Instances of Set 2 that can be solved by exhaustion. Number of components c, and time to solve t, compared to a branch-and-cut approach.

					Enum		CCut4							Enum		CCut4	
n	m	r	λ	k	c	t (s)	C	t (s)	n	m	r	λ	k	c	t (s)	C	t (s)
100	990	6	0.50	6	1	23.17	1	0.00	300	35,880	4	0.50	2	3	0.02	3	3.18
100	2,475	7	1.00	3	9	0.13	9	0.08	300	35,880	7	1.00	2	41	0.11	41	1.62
100	3,960	1	1.25	3	3	0.35	3	1.14	400	15,960	3	0.25	3	15	0.83	15	10.78
200	3,980	6	0.25	3	6	0.05	6	0.35	400	39,900	3	1.00	3	60	37.95	60	13.47
200	9,950	3	1.00	3	26	2.29	26	1.02	400	39,900	5	0.50	3	5	7.02	5	96.03
200	9,950	8	0.50	3	1	0.43	1	0.20	400	63,840	6	1.00	3	12	55.54	12	1,200.07
200	15,920	5	1.00	3	3	3.41	3	27.59	400	63,840	7	1.25	3	31	96.04	31	1,200.12
200	15,920	6	1.25	3	11	5.88	11	33.05	500	24,950	7	0.25	3	21	2.06	21	30.54
300	8,970	3	0.50	4	37	59.34	37	10.51	500	62,375	5	0.50	3	6	17.24	6	403.06
300	8,970	7	0.25	4	1	5.65	1	0.22	500	62,375	5	1.00	3	81	92.62	81	40.58
300	22,425	6	0.50	2	23	0.03	23	2.71	500	99,800	6	1.00	3	15	135.82	17	1,200.36
300	22,425	8	0.25	2	1	0.00	1	0.16	–	–	–	–	–	–	–	–	–

components over all solutions, and σ_c the standard deviation. We can see that solution values are approximately normal, with the optimal value never less than two standard deviations below the mean. (The range over all instances in Set 1 is $[-1.58, -1.90]$.) These observations support the idea that many instances in the current instance sets are too regular and too easy to solve.

4.4. Heuristic solution

In this section we evaluate the quality of the heuristic proposed in Section 3. We have run a single replication of the combined GRASP with CBFS, dubbed GC, and compare its results to the Simulated Annealing, Reactive Tabu Search, genetic algorithm and the Pilot method as reported by Cerulli et al. (2014), and to the fix-and-optimize matheuristics of Pinheiro et al. (2022). The results are shown in Table 5. It shows for each of the heuristics the mean number of components over all 10 instance per group c, as well as the computation time t (in seconds). Pinheiro et al. (2022) additionally report c_{30}, the best solution over 30 replications (note that the effective time for c_{30} is consequently 30 times higher). The computation times are rescaled from the original times reported by Cerulli et al. (2014), as reported by Pinheiro et al. (2022), to adjust for machine differences, and should be taken, as always, with a grain of salt.

By comparison with Table 3 we can see that all algorithms find the optimal solutions for the easy instances where exact values are known. Exceptions are $n = 100, \lambda = 1$ which could be solved by the branch-and-bound approach of Cerulli et al. (2014), and $n = 500, \lambda = 0.25$ which has been solved by complete enumeration, but both take considerably more time. When comparing solution qualities, it is clear that GC finds the overall best values, and is best in 18 of 20 instance groups. Solution times are also clearly better, with more than a factor three faster than the second fastest approach of Pinheiro et al. (2022) on comparable machines.

We finally turn to a brief comparison to the exact algorithm of Figueredo and Campêlo Neto (2020). The motive here is only to judge solution quality of GC and not a direct comparison, since both algorithms have a different purpose. Table 6 has the results. We can see that with the exception of one instance ($n = 300, m = 22425, k = 4, r = 3, \lambda = 1$) GC finds the optimal solution for all 31 instances where CCut4 could prove optimality (i.e. the instances with $t < 1200$ seconds). On the remaining instances we observe 5 ties, and 7 better solutions. Given the short solving times, GC could useful for seeding exact solution approaches.

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Figure 2: Exemplary solution value distributions for instances from Set 1 with $n \in \{200, 400\}$ vertices and $\lambda = 0.25$. Both graphs show the histogram of standard scores z for all solutions over all 10 replications.

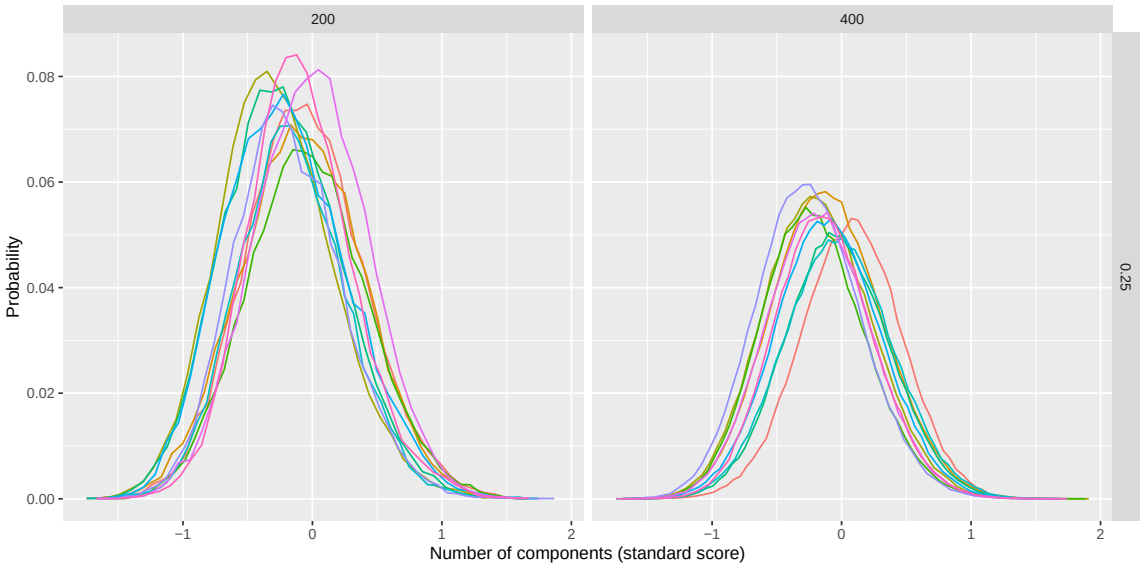


Table 5: Comparison of heuristic solutions on Set 1 to the approaches of Cerulli et al. (2014), Pinheiro et al. (2022).

				SimAnn		RTS		GA		PM		FO			GC	
n	m	λ	k	c	t	c	t	c	t	c	t	c	t	c_{30}	c	t
100	990	0.25	3	**6.3**	0.1	**6.3**	0.1	**6.3**	0.2	**6.3**	0.0	**6.3**	0.2	**6.3**	**6.3**	0.0
100	990	0.50	6	2.7	0.2	2.9	0.1	**2.6**	0.4	2.7	0.1	2.7	0.7	**2.6**	**2.6**	0.0
100	990	1.00	6	**15.0**	0.4	**15.0**	0.4	**15.0**	0.7	15.6	0.3	15.1	8.1	**15.0**	**15.0**	0.1
100	990	1.25	7	**15.7**	0.5	**15.7**	0.5	**15.7**	0.9	**15.7**	0.6	15.8	7.1	**15.7**	**15.7**	0.2
150	2235	0.25	4	**3.5**	0.2	**3.5**	0.1	**3.5**	0.5	**3.5**	0.0	3.6	0.1	3.6	**3.5**	0.0
150	2235	0.50	4	**22.3**	0.4	**22.3**	0.4	**22.3**	0.8	**22.3**	0.1	22.3	1.1	**22.3**	**22.3**	0.1
150	2235	1.00	9	7.7	1.0	**7.3**	0.8	8.1	2.9	8.3	1.3	7.4	3.2	7.5	7.4	0.4
150	2235	1.25	11	6.1	1.3	6.2	1.1	6.2	3.3	6.1	1.9	5.8	5.7	**5.7**	**5.7**	0.7
200	3980	0.25	3	**17.0**	0.4	**17.0**	0.4	17.2	0.9	**17.0**	0.1	17.2	0.5	17.2	**17.0**	0.0
200	3980	0.50	6	**9.3**	0.8	**9.3**	0.8	9.6	2.6	9.6	0.9	9.4	5.0	9.4	**9.3**	0.2
200	3980	1.00	12	2.6	1.6	3.1	1.1	3.1	5.0	3.0	2.7	2.5	3.1	**2.4**	**2.4**	0.9
200	3980	1.25	15	1.8	1.7	1.6	1.2	1.5	4.9	**1.2**	5.3	1.3	1.5	1.3	1.3	1.1
400	15960	0.25	3	**35.6**	5.2	**35.6**	5.2	**35.6**	4.5	**35.6**	1.4	35.8	1.9	**35.6**	**35.6**	0.2
400	15960	0.50	6	24.4	11.2	23.7	11.3	25.1	12.3	24.0	16.6	23.9	34.7	23.7	**23.3**	0.9
400	15960	1.00	12	12.1	25.4	11.4	21.7	12.5	37.5	11.1	81.8	10.6	33.5	10.6	**9.9**	5.1
400	15960	1.25	15	8.7	34.2	8.2	21.2	8.9	48.9	7.7	163.0	7.3	13.8	7.0	**6.8**	11.4
500	24950	0.25	4	22.7	13.1	22.7	13.2	22.9	10.7	22.7	8.3	23.4	1.1	23.0	**22.6**	0.3
500	24950	0.50	7	22.4	27.5	21.5	27.8	22.8	24.8	21.9	33.1	21.8	28.1	21.7	**21.3**	1.9
500	24950	1.00	15	6.3	47.2	5.3	42.0	6.6	75.6	5.0	251.8	5.4	19.1	4.9	**4.5**	11.1
500	24950	1.25	19	3.4	52.7	3.0	38.6	3.4	111.0	2.1	214.1	2.4	22.1	2.1	**2.0**	21.8
				12.3	11.3	12.1	9.4	12.4	17.4	12.1	39.2	12.0	9.5	11.9	**11.7**	2.8

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Table 6: Comparison of heuristic solutions on Set 2 to the exact values of Abreu Figueredo (2020). Number of components c, and time to solve t.

					CCut4		CBFS							CCut4		CBFS	
n	m	k	r	λ	t (s)	c	t (s)	c	n	m	k	r	λ	t (s)	c	t (s)	c
100	2,475	3	7	1.00	0.1	9	0.0	9	400	15,960	3	3	0.25	10.8	15	0.2	15
100	3,960	3	1	1.25	1.1	3	0.1	3	300	35,880	4	3	1.25	1,200.0	3	1.1	2
100	990	6	6	0.50	0.0	1	0.0	1	400	15,960	6	4	0.50	1,200.0	8	1.1	8
100	990	6	5	1.00	0.2	13	0.1	13	300	8,970	9	1	1.00	1,200.0	11	2.5	9
100	990	6	5	1.25	0.1	17	0.1	17	400	39,900	3	5	0.50	96.0	5	0.4	5
200	15,920	3	5	1.00	27.6	3	0.2	3	400	15,960	12	6	1.00	15.6	1	2.8	1
200	3,980	6	5	0.50	100.3	2	0.2	2	300	8,970	9	7	1.25	1,200.0	29	3.4	29
200	3,980	3	6	0.25	0.3	6	0.0	6	400	39,900	3	3	1.00	13.5	60	0.9	60
200	15,920	3	6	1.25	33.0	11	0.3	11	400	63,840	3	6	1.00	1,200.1	12	1.1	12
200	9,950	3	8	0.50	0.2	1	0.1	1	500	24,950	3	7	0.25	30.5	21	0.3	21
200	9,950	3	3	1.00	1.0	26	0.2	26	400	63,840	3	7	1.25	1,200.1	31	1.3	31
200	3,980	6	7	1.00	0.7	36	0.5	36	400	39,900	6	4	1.25	1,200.1	5	3.4	4
300	22,425	2	6	0.50	2.7	23	0.1	23	500	24,950	7	3	0.50	1,200.0	2	2.1	2
200	9,950	6	7	1.25	1.1	1	0.4	1	500	62,375	3	5	0.50	403.1	6	0.7	6
300	22,425	2	8	0.25	0.2	1	0.0	1	500	62,375	3	5	1.00	40.6	81	1.7	81
300	35,880	2	4	0.50	3.2	3	0.1	3	500	24,950	7	3	1.00	1,200.1	72	5.4	72
200	3,980	12	4	1.25	0.3	1	0.7	1	500	99,800	3	6	1.00	1,200.4	17	1.7	15
300	35,880	2	7	1.00	1.6	41	0.3	41	500	62,375	7	6	1.25	131.6	1	3.2	1
300	22,425	4	3	1.00	645.3	14	0.8	15	400	15,960	12	2	1.25	1,200.0	9	11.6	4
300	8,970	4	3	0.50	10.5	37	0.3	37	500	24,950	7	3	1.25	18.7	114	6.3	114
300	22,425	4	3	1.25	233.9	29	1.0	29	500	99,800	3	1	1.25	1,200.3	41	2.2	41
300	8,970	4	7	0.25	0.2	1	0.1	1	–	–	–	–	–	–	–	–	–

5. Conclusions

We have proposed a new heuristic for the kLSF which combines GRASP and CBFS. It compares favorably with existing heuristic procedures, finding better solution in a shorter time. On the instances for which an optimal solution is known the heuristic produces optimal solutions.

We have further taken a closer look at existing instances, found that many of them are too regular and too easy to solve. Therefore, useful future work would be to define a set of new instances that is more diverse, using different random graph models, and possible with an embedded, hidden optimal solution, to facilitate the comparison of algorithms.

Acknowledgements

We acknowledge support from Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, Brasil (CAPES), Finance Code 001, and CNPq (Grant 310259/2022-3).

References

Abreu Figueredo, Pedro Jorge de (2020). “O problema da floresta geradora k-rotulada”. MA thesis. Universidade Federal do Ceará.

Cerulli, R. et al. (Jan. 2014). “The k-labeled Spanning Forest Problem”. In: *Procedia - Social and Behavioral Sciences* 108. DOI: 10.1016/j.sbspro.2013.12.828.

Chang, Ruay-Shiung and Leu Shing-Jiuan (1997). “The minimum labeling spanning trees”. In: *Information Processing Letters* 63.5, pp. 277–282. DOI: 10.1016/s0020-0190(97)00127-0.

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br

Chwatal, Andreas M and Günther R Raidl (2010). "Solving the Minimum Label Spanning Tree Problem by Ant Colony Optimization." In: *Gem*, pp. 91–97.

Consoli, Sergio and José Andrés Moreno Pérez (2015a). "An intelligent extension of Variable Neighbourhood Search for labelling graph problems". In: *arXiv*. DOI: 10.48550/ARXIV.1509.08792.

— (Feb. 2015b). "Variable neighbourhood search for the k-labelled spanning forest problem". In: *Electronic Notes in Discrete Mathematics* 47. DOI: 10.1016/j.endm.2014.11.005.

Consoli, Sergio, José Andrés Moreno Pérez, and Nenad Mladenović (2015). "Towards an intelligent VNS heuristic for the k-labelled spanning forest problem". In: *arXiv*. DOI: 10.48550/ARXIV.1503.02009.

— (Nov. 2017). "Comparison of metaheuristics for the k-labeled spanning forest problem". In: *International Transactions in Operational Research* 24.3 (3), pp. 559–582. DOI: 10.1111/itor.12217.

Feo, Thomas A. and Mauricio G. C. Resende (1995). "Greedy Randomized Adaptive Search Procedures". In: *J. Glob. Optim.* 6.2, pp. 109–133. DOI: 10.1007/BF01096763.

Figueredo, Pedro Jorge de Abreu and Manoel Bezerra Campêlo Neto (2020). "O Problema da Floresta Geradora k-Rotulada". In: *Anais do Simpósio Brasileiro de Pesquisa Operacional*. SBPO 2020. Galoa. DOI: 10.59254/sbpo-2020-122578.

Hanauer, Kathrin, Monika Henzinger, and Christian Schulz (Dec. 2022). "Recent Advances in Fully Dynamic Graph Algorithms – A Quick Reference Guide". In: *ACM Journal of Experimental Algorithmics* 27, pp. 1–45. ISSN: 1084-6654. DOI: 10.1145/3555806.

Janson, Svante et al. (Jan. 1993). "The birth of the giant component". In: *Random Structures & Algorithms* 4.3, pp. 233–358. ISSN: 1098-2418. DOI: 10.1002/rsa.3240040303.

Kao, Gio K., Edward C. Sewell, and Sheldon H. Jacobson (2009). "A branch, bound, and remember algorithm for the $1 \mid r_i \mid \sum t_i$ scheduling problem". In: *J. Sched.* 12.163. DOI: 10.1007/s10951-008-0087-3.

Knuth, Donald Ervin (2011). *The art of computer programming*. Vol. IVa, Combinatorial Algorithms, Part 1. Addison-Wesley.

Mladenović, Nenad and Pierre Hansen (Nov. 1997). "Variable neighborhood search". In: *Computers & Operations Research* 24.11, pp. 1097–1100. ISSN: 0305-0548. DOI: 10.1016/s0305-0548(97)00031-2.

Pinheiro, Tiago F.D., Santiago V. Ravelo, and Luciana S. Buriol (July 2022). "A fix-and-optimize matheuristic for the k-labelled spanning forest problem". In: *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. DOI: 10.1109/cec55065.2022.9870342.

Voß, Stefan, Andreas Fink, and Cees Duin (Apr. 2005). "Looking Ahead with the Pilot Method". In: *Annals of Operations Research* 136.1, pp. 285–302. ISSN: 1572-9338. DOI: 10.1007/s10479-005-2060-2.

DOI: 10.59254/sbpo-2024-193454
https://proceedings.science/p/193454?lang=pt-br