UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

TIAGO FURTADO DREHMER PINHEIRO

The k-labeled spanning forest problem: complexity, approximability, formulations and algorithms

Dissertation presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Advisor: Prof. Dr. Santiago Valdés Ravelo
Coadvisor: Prof. Dr. Luciana Salete Buriol

Porto Alegre

CIP — CATALOGING-IN-PUBLICATION

Tiago Furtado Drehmer Pinheiro,

The k-labeled spanning forest problem: complexity, approximability, formulations and algorithms / Tiago Furtado Drehmer Pinheiro. – Porto Alegre: PPGC da UFRGS, .

48 f.: il.

Dissertation (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, . Advisor: Santiago Valdés Ravelo; Coadvisor: Luciana Salete Buriol.

1. K-Labeled Spanning Forest. 2. NP-hardness. 3. Inapproximability. 4. Metaheuristic. 5. Matheuristic. 6. Integer Linear Program. 7. Fix-and-optimize. I. Santiago Valdés Ravelo, . II. Luciana Salete Buriol, . III. Título.

“One day I will find the right words and they will be simple”

— JACK KEROUAC

ACKNOWLEDGMENT

My special thanks to my advisor Santiago Ravelo and my coadvisor Luciana Buriol, who over these 2 years, in addition to care about my well being, had a lot of patience with my writing problems. It was hours and hours of meetings, often rewriting what I wrote in a much more appropriate way. I would like to thank my family members for always providing me adequate comfort to focus on my studies, and also give me the love and affection I have always received. Finally, a special thanks to my groups of friends "Ordem Secreta do Pico", "Pico da Neblina", "23 anos", "Papo Pseudosocial", the people from "Badaras" and other friends that are not included in this groups but who are with me on my journey. With these people, there were several conversations about NP problems, the k labeled spanning forest problem, algorithms, metaheuristics, etc... that motivated me to continue studying since I was able to bring the content proposed here to my homies.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS **6**
LIST OF FIGURES **7**
ABSTRACT **8**
1 INTRODUCTION **9**
1.1 Graphs definitions and notations **10**
1.2 A summary of this work's results **12**
1.3 Organization of the work **12**
2 LITERATURE REVIEW **13**
2.1 Bibliographical review on the minimum labeling spanning tree problem **13**
2.2 Bibliographical review on the k-labeled spanning forest **14**
2.2.1 Main algorithmic proposals for the k-labelled spanning forest 14
2.2.2 Comparison of metaheuristics for the k-labeled spanning forest problem 16
2.2.3 The k-labelled spanning forest - Theoretical results and Mixed integer programming study for the problem 17
3 NP-HARDNESS AND APPROXIMABILITY **21**
3.1 NP-Hardness **21**
3.2 Inapproximability **25**
3.3 Approximation for triangle and edges graph cases **29**
4 FIX-AND-OPTIMIZE PROPOSAL **30**
4.1 Component model **30**
4.2 Fix-and-optimize matheuristic **32**
4.3 Computational experiments **35**
4.4 Computing environment and library **36**
4.5 Instances **36**
4.6 Parameters **37**
4.7 Results and analysis **38**
5 CONCLUSION AND FUTURE WORKS **42**
REFERENCES **44**
APPENDIX A — RESUMO ESTENDIDO **46**

LIST OF ABBREVIATIONS AND ACRONYMS

KLSF	K-Labeled Spanning Forest
DFS	Depth First Search
MLST	Minimum Labelling Spanning Tree
MVCA	Maximum Vertex Cover Algorithm
X3C	3 Exact Cover
SA	Simulated Annealing
RTS	Reactive Tabu Search
GA	Genetic Algorithm
VNS	Variable neighborhood search
COVNS	Complementary variable neighborhood search
IVNS	Intelligent variable neighborhood search
GRASP	Greedy randomized adaptive search procedure

LIST OF FIGURES

Figure 1.1 The graph in the left represents an input of the KLSF with set of labels $\{a, b, c, d, e\}$, and considering $k = 2$, the graph in the right is an optimal solution for that instance maintaining only the labels $\{a, b\}$. 9

Figure 2.1 The graph on the left is an input graph for KLSF, on the right the extended graph is illustrated, with the new node s and the edges between s and the rest of the nodes. Notice that different new labels (l_1, l_2, l_3 and l_4) were defined for the edges incident in s. 18

Figure 3.1 Example of construction of an input from X3C to one input for KLSF. The 3 triples define the labels l_1, l_2, and l_3. For each element that is contained in two triples, one triangle is constructed. The first triangle refers to the element c that is contained in the triples 1 and 2 and the other two are related to the elements d and e that share the triples 2 and 3. The edges bellow the triangles are the edges defined in the step 3 of the construction, that considers $m = 3$, the maximum number of edges with the same label, defined from the label l_2. 23

Figure 3.2 An example of all 3 paths generated by the Walecki construction with K_6..26

Figure 3.3 An example of the construction from an instance of X3C to an instance of KLSF, where $|\mathcal{T}| = 2$, $|X| = 5$ and $\theta = 1$. 27

Figure 4.1 Example of how a sequence of labels is fixed. In the first step, the label b is chosen to be fixed, contracting the edges $(2, 6)$ and $(3, 4)$ and producing two merged nodes. In the second step the label c is chosen to be fixed, and the edge between the two previously merged nodes is contracted. 34

Figure 4.2 In this image, the x-axis represents the set of instances defined for each combination of $|X|$ and $|L|$ from the data set. The red squares in the image represent the average of the best known value of each subset of instances, and the blue dots represent the overall average (consider the average value of each subset and compute the average value among them). The lines that extend the blue dots represent the standard deviation of the results. 41

ABSTRACT

In this work, we study the k-labeled spanning forest problem (KLSF). The input of the KLSF is an undirected graph with labeled edges and a positive integer k. The goal is to find a spanning forest of the graph with at most k different labels associated with the edges, that minimizes the number of components. KLSF finds practical applications in different scenarios related to networks design and telecommunications. Its solutions may help to reduce the negative impact of electromagnetic fields exposure on the population health or to increase profits of internet management companies, among others. The interest in the KLSF problem is not only practical but also theoretical since the problem generalizes the best-known NP-hard minimum labeling spanning tree problem (MLST). This work reinforces the NP-hardness of the KLSF and ensures that, even for the simple instances where the components of the original graph are only triangles and edges, the problem is NP-hard. Also as a theoretical result, an inapproximability proof is presented for it, ensuring that unless P = NP there is no polynomial time algorithm with approximation factor polynomial in the number of the labels. To complete the theoretical results a trivial 3-approximation result is presented for the particular case where the input graph components are edges or triangles. From the application side, to approach KLSF, we propose a fix-and-optimize matheuristic that was tested over several instances, achieving high-quality solutions in reasonable computational time. When compared to the best-known algorithms in the literature, our matheuristic outperformed the other proposals in most cases, finding better solutions in less computational time for the most challenging instances.

Keywords: K-Labeled Spanning Forest. NP-hardness. Inapproximability. Metaheuristic. Matheuristic. Integer Linear Program. Fix-and-optimize.

1 INTRODUCTION

Globalization has increasingly demanded the need to keep people connected. This demand for connectivity has been satisfied by computer networks and their technologies which are constantly evolving. This evolution creates the need to find solutions for new problems that end up emerging. One example occurs when studying strategies to reduce the impact in the population health of continuous electromagnetic field exposures. Nowadays, the emergence of wireless networks such as Wifi and cell phone networks have been increasingly exposed the population to electromagnetic fields, which can bring possible negative effects on people's health [Wiart et al. 2019].

This problem of exposure can be modeled as the k-labeled spanning forest (kLSF). The kLSF is defined over the class of graphs labeled on edges, that is, graphs with each edge mapped to one label. The objective is to obtain a spanning forest of the graph with the smallest number of trees (components), such that the number of labels associated to the forest's edges is upper-bounded by an integer k. One example for the kLSF is illustrated by Figure 1.1.

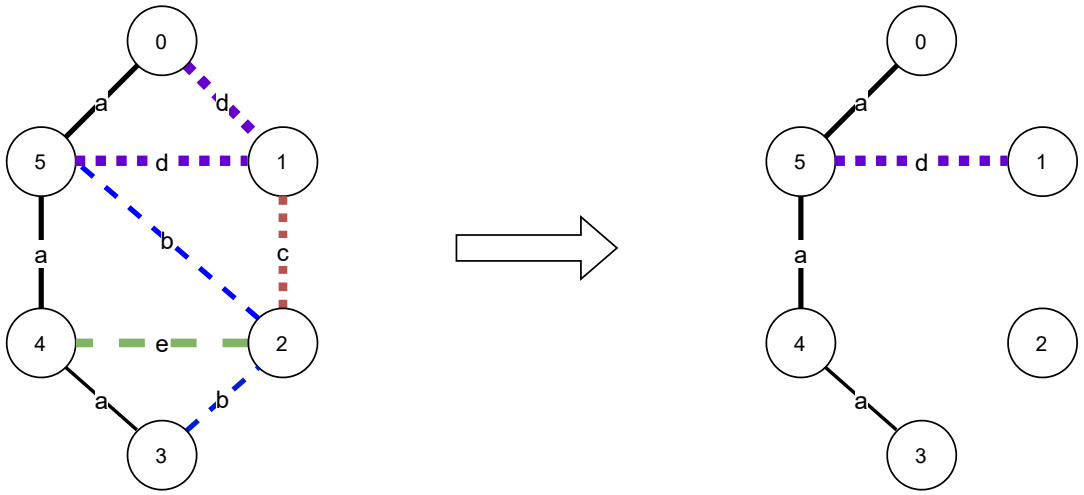

Figure 1.1 – The graph in the left represents an input of the kLSF with set of labels $\{a, b, c, d, e\}$, and considering $k = 2$, the graph in the right is an optimal solution for that instance maintaining only the labels $\{a, b\}$.

In the context of the exposure to electromagnetic fields mentioned before, the nodes of the graph represent the network's devices and the edges the connections between them, being the labels the frequencies of such connections. Guaranteeing that the number of different frequencies is less than or equal to a value k may reduce such a population exposure. To augment the quality of the services, the disconnected coverage areas should be as few as possible, being this the objective of the kLSF.

Another network problem that can be modeled as KLSF is related to business and financial issues of network services. The decentralization of network administration between different types of companies and agencies involves the negotiation of paths to keep the internet connection [Tanenbaum, Wetherall et al. 1996]. In this scenario the topology of graphs can be built, defining again as nodes the devices of the network and as edges the connections between the devices. In this case, the different network providers would be the labels. When planning its network, a company will try to limit the number of connections not managed by it. Thus, considering the usage of at most k different network providers would bring financial and administrative benefits when negotiating and planning the connections.

In the context of routing, we can model a multimodal transport system [Miller et al. 2005] as the KLSF. On this scenario, the different companies of transporting can be represented by labels, while the locals and the paths between them can be, respectively, nodes and edges. In this case, minimizing the number of companies would reduce the service's costs, then limiting the labels to at most k may bring financial and logistical benefits.

Motivated by the practical applicability of the KLSF and by its theoretical challenge this work aims to explore this problem in different ways, approaching it both theoretically and in an applicable way. But first it is necessary to define the problem formally.

1.1 Graphs definitions and notations

The KLSF is defined over simple undirected graphs (or simply graphs), where a **graph** G consists of two sets: a set of nodes denoted by V_G, and a set of edges (unordered pairs of nodes) denoted by E_G. We use V and E instead of V_G and E_G whenever G is apparent from context.

Besides the graph definition, other graph concepts are required before defining KLSF. Below we define the subgraph relation and some special classes of graphs such as paths, cycles, connected graphs, trees, and forests.

Given two graphs G and H, we say that H is **subgraph** of G (denoted by $H \subseteq G$) if the set of nodes of H is a subset of the set of nodes of G (i.e., $V_H \subseteq V_G$) and the set of edges of H is a subset of the set of edges of G (i.e., $E_H \subseteq E_G$). If $H \subseteq G$ and $V_H = V_G$, then H is a **spanning subgraph** of G.

A graph G is a **path** if there exists a sorting $u_1, u_2, \ldots, u_{|V_G|}$ of the nodes, such that

the edges of G are defined by each pair of consecutive nodes (i.e., $E_G = \{(u_i, u_{i+1})\}_{i=1}^{|V_G|-1}$). In such cases it is said that G is a path connecting u_1 to $u_{|V_G|}$.

G is a **connected** graph if for every pair of nodes $u, v \in V_G$ there exists a path $P \subseteq G$ connecting u to v. Any maximal connected subgraph $H \subseteq G$ is a **component** of G (i.e., for each $G' \subseteq G$ if G' is connected and $H \subseteq G'$, then $H = G'$). Given a graph G, the set of its components is denoted by $C(G)$.

A graph G is a **cycle** if contains at least three nodes (i.e., $|V_G| \geq 3$) that can be ordered as $u_1, u_2, \ldots, u_{|V_G|}, u_1$, such that there exists an edge between each pair of consecutive nodes (i.e., $E_G = \{(u_{|V_G|}, u_1)\} \cup \{(u_i, u_{i+1})\}_{i=1}^{|V_G|-1}$). If no subgraph $H \subseteq G$ is a cycle, then G is **acyclic**.

Any connected and acyclic graph is a **tree**, and a **forest** is a graph whose components are trees. H is a **spanning tree** (**spanning forest**) of a graph G iff H is tree (forest) and H is a spanning subgraph of G.

With the graph notations well defined, we can now formally introduce the KLSF:

Problem 1. *The k-labeled spanning forest (*KLSF*).*

Input: *A tuple $I = \langle G, L, \ell, k \rangle$, where:*

- *G is a graph.*
- *L is a set of labels.*
- *$\ell : E_G \to L$ is a function that maps each edge to a label.*
- *$k \in \mathbb{N}$ is an integer value such that $1 \leq k \leq |L|$, representing an upper bound for the number of labels in a solution.*

Output: *A spanning forest $F \subseteq G$ with minimum number of components, which satisfies $|\ell(E_F)| \leq k$.* [1]

Given an instance $I = \langle G, L, \ell, k \rangle$ of the KLSF, and a subset of labels $S \subseteq L$, the graph $G[S]$ denotes the spanning subgraph of G containing all edges of G with labels in S (i.e., $V_{G[S]} = V_G$ and $E_{G[S]} = \{e | e \in E_G \text{ and } \ell(e) \in S\}$). Also, given $X \subseteq V_G$, its **labeled cut** is the function $\ell(X)$ that returns the set of labels associated to every edge with exactly one endpoint in X (i.e., $\ell_G(X) = \{\ell((u,v)) \,|(u,v) \in E, u \in X \text{ and } v \in V \setminus X\}$).

[1] Given a function f and a subset S of the domain of f, we define $f(S) = \{f(s) | \forall s \in S\}$

1.2 A summary of this work's results

Our main contributions to the KLSF are listed below.

- Proof of NP-hardness for the problem even when the input instance is a graph where the components are only triangles and edges (triangles-edges graph).
- Proof of inapproximability that ensures there are no polynomial approximation for the problem in relation to the set of labels unless P = NP.
- A new mathematical model for the problem,
- A competitive matheuristic to approach the problem.

1.3 Organization of the work

The remainder of this document is organized as follows. First, it is presented a literature review with the main proposals for KLSF in Chapter 2, followed by the theoretical results given by Chapter 3. Chapter 4 presents the description of the fix-and-optimize matheuristic proposed on this work. Finally, Chapter 5 presents a conclusion about the results of the work, and indicates possible future research directions.

2 LITERATURE REVIEW

In this chapter we review the main literature results related to the KLSF. Section 2.1 presents the works developed for the particular case MLST, while Section 2.2 presents the studies for the KLSF.

2.1 Bibliographical review on the minimum labeling spanning tree problem

The MLST is particular case of the KLSF problem, that given a connected graph labelled in the edges, the objective of the problem is find a spanning tree, with the minimum number of labels. The first study for the MLST was proposed by [Chang and Shing-Jiuan 1997], presenting a proof for the problem's NP-hardness and three algorithms. One of these algorithms was a heuristic that starts with a spanning tree and makes swaps of edges, with the objective of reducing the number of labels. The second algorithm was a greedy heuristic named Maximum Vertex Covering Algorithm (MVCA), which starts with a spanning forest containing only the graph's nodes and iteratively chooses a label that ensures the greater reduction in the number of components when the associated edges are included to the solution being constructed. This procedure is executed until the solution has only one component. The last proposal was an exponential-time A^*-algorithm developed for computing an optimal solution of the problem.

Over the years many other approaches were proposed for MLST and related problems [Silva 2018]. In [Consoli et al. 2009], the minimum labeling Steiner tree problem was studied. In that study some heuristic approaches were presented. One of those heuristics is a variable neighborhood algorithm, for it are presented one evolution, that uses the idea of complementary solution. The idea of a complementary solution consists of: construct a new solution before a new exploration of the neighborhood. Such that this the new solution does not use any edge labelled by the current solution explored. In [Xiong, Golden and Wasil 2006], a genetic algorithm and four adaptations of the MVCA was proposed for MLST, and, in [Vaisman 2022], a cross-entropy approach was developed similarly to an evolutionary algorithm that considers static proprieties of the solutions' set.

2.2 Bibliographical review on the k-labeled spanning forest

This section presents a bibliographical review of the main results of the literature for the KLSF. Three main works were reviewed: the presentation of the problem by [Cerulli et al. 2014] where six heuristics were proposed, the work of [Consoli, Perez and Mladenović 2017] were four other heuristics were developed, and the work of [Figueredo 2020] where a study on mathematical models and techniques for solving the problem were conducted. Each of the following subsections presents a review of these studies.

2.2.1 Main algorithmic proposals for the k-labelled spanning forest

In the work of [Cerulli et al. 2014] the KLSF was introduced and a proof of NP-hardness was presented. This proof was obtained by a trivial reduction from the particular case MLST to the KLSF. Additionally, in the same study some heuristics were applied for solving the problem:

1. Maximum Vertex Cover Algorithm (MVCA): Inspired by the heuristic MVCA proposed by [Chang and Shing-Jiuan 1997] for the MLST, this algorithm was modified by [Cerulli et al. 2014] to limit the process of choice of labels to at most k labels. A step-by-step description of this procedure is presented in Algorithm 1.

Algorithm 1 Maximum Vertex Cover Algorithm (MVCA)

Input: An input for KLSF $I = \langle G, L, \ell, k \rangle$
Output: A forest F with at most k labels
1: $F \leftarrow (V = V_G, E = \{\})$
2: **while** $|\ell(E_F)| < k$ **do**
3: $\quad l \leftarrow argmin_{\forall c \in L \setminus \ell(E_F)} |C(G\,[\ell(E_F) \cup \{c\}])|$
4: $\quad F \leftarrow$ Spanning forest from $G[\ell(E_F) \cup \{l\}]$
5: **end while**
6: **return** F

2. Pilot method (PILOT): Also a constructive heuristic, the PILOT approach uses the MVCA as a look-ahead mechanism. At each step of the algorithm, each label not present in the solution yet is evaluated with the already chosen labels. Such evaluation is computed by applying a MVCA with the remaining labels. Then, the label tested with the best result of the objective function after the MVCA is selected to compose the solution. This sequence of steps is executed until k labels

are chosen.

3. Local Search Heuristic (LSH): This heuristic defines a neighborhood for the solutions of the KLSF. This neighborhood was defined by all possible exchanges between a label that was in the solution and one that was not. If a change gives a better objective value, then the solution is updated, otherwise the new solution is discarded and the previous one is kept.
4. Reactive Tabu Search (RTS): This heuristic keeps a reactive tabu list that forbids the algorithm from doing some steps in the neighborhood's exploration. This list has a dynamic size, growing up whenever the algorithm tries to reevaluate a movement to a neighborhood already present in the tabu list. If the algorithm does some steps without hitting the list, then the list's size decreases. This list allows the algorithm to do movements worse than the actual explored since the best movements could be a tabu movement. As the algorithm can get stuck with the tabu list of moves, the author proposed a sequence of random moves as a solution to avoid such a problem. The stopping criterion of the algorithm is given by an input parameter.
5. Simulated Anealing (SA): Also using the idea of neighborhood's exploration, this algorithm keeps a temperature parameter that defines the possibility of a solution being accepted. This temperature allows the algorithm to accept a solution even if it is worse than the current one. This tolerance of acceptance was defined by the Boltzmanm function [Landau 1980] that decays as the iterations occur.
6. Genetic Algorithm (GA): In the genetic algorithm proposed for KLSF a population was initialized with random feasible solutions, let us refer such population as $P = \{p_1, p_2, ..., p_{|P|}\}$. The fitness function used was the number of components of the solution. The parents of the next generation were defined through the sequence $1 \leq i \leq |P|$ of individuals, that is, the individuals p_i and $p_{((i+1)\ mod(|P|))}$ generate the individual t_i with the crossover procedure. The crossover was made by applying an MVCA with the set of labels of the parent solutions. After the crossover, the mutation procedure in the new individual was applied, this procedure was made by adding one more label to the new individual and executing the MVCA with the $k + 1$ labels. From $1 \leq i \leq P$, the next generation was generated by keeping in the population the individual with the best fitness between the generated t_i and the previews one p_i.

In addition to these heuristics, a branch and bound algorithm was presented by

[Cerulli et al. 2014] that exactly solved small instances. The work also proposed a set of random instances for testing the algorithm. This set was provided by the authors and was used for our computational experiments in Chapter 4.

2.2.2 Comparison of metaheuristics for the k-labeled spanning forest problem

The proposal of [Consoli, Perez and Mladenović 2017] consisted of four different metaheuristics for the KLSF:

1. Greedy randomized adaptive search procedure (GRASP): The GRASP proposed for the KLSF starts with a solution with only one random label and creates a set of candidate labels to be included in the solution. The candidates are defined by all the labels that offer the greater reduction for the number of components considering the labels already in the solution. One random label is picked from the candidates. This procedure of choosing labels is executed until k labels have been chosen. After the choice of the k labels, a local search is executed, which consists of removing one label of the solution and adding the label that guarantees the best objective value.
2. Variable neighborhood search (VNS): The VNS defines neighborhoods for the solutions, which in this work were based on the hamming distance. Let $q \geq 1$ be an integer value, and S be a feasible solution, the neighborhood $N_q(S)$ represents all solutions of the space of solutions, such that the hamming distance to S is less than q. The exploration of solutions is made in 2 phases, the shaking phase, that from $q = 1$ to $q = k + k/3$ constructed a new solution of $N_q(S)$. Such construct consists in: remove labels until q if $q \leq k$, and when $q > k$ after make the solution empty removing labels, start to add labels to it until $q - k$. After this procedure the solution has less labels than k, than the algorithm complete such solution applying a MVCA with the labels that are not in the it. To finish the algorithm, it was applied an improvement phase where a LSH is applied in the final solution of the shaking phase.
3. Complementary variable neighborhood search (COVNS): As an update of the above VNS, to ensure a better exploration of the solutions' space, before the construction of the neighbor, the COVNS constructs a complementary solution for exploring the neighborhood. This complementary solution is computed by an MVCA with the labels that were not in the final solution of the LSH (in the complementary space of

labels).

4. Intelligent variable neighborhood search (IVNS): The IVNS was an evolution of COVNS, where instead of using the MVCA to generate the new solution, a probabilistic construction was used. This construction considered an evaluation of the labels of the complementary set and the Boltzmanm function.

2.2.3 The k-labelled spanning forest - Theoretical results and Mixed integer programming study for the problem

Different from the others studies proposed for the KLSF in [Figueredo 2020], the focus were the presentation of theoretical results and exact methods for it. [Figueredo 2020] showed particular cases that admit polynomial time solutions, gave integer programming formulations and applied techniques within mixed integer programming solvers for solving the problem. These techniques included cutting planes, lazy constraints, and bender decomposition. Also, the authors presented a parallel implementation for the branch and bound proposed in [Cerulli et al. 2014]. For testing the techniques developed, more instances were generated following the construction proposed by [Cerulli et al. 2014].

In [Figueredo 2020], three different mathematical models for KLSF were introduced, tested over several instances within a mixed integer programming solver, and ranked by their performance. The formulation with overall better results was the *"labeled Cut Model"*.

The *"labeled Cut Model"* requires the concept of extended instance, which consists of adding an extra node s to the graph, and for each node $u \in V$ an extra edge is also included between s and u with a new label l_u. Figure 2.1 illustrates the construction of an extended instance.

Formally, given an instance $I = \langle G, L, \ell, k\rangle$ of KLSF, the extended instance $I' = \langle G', L', \ell', k'\rangle$ of I is:

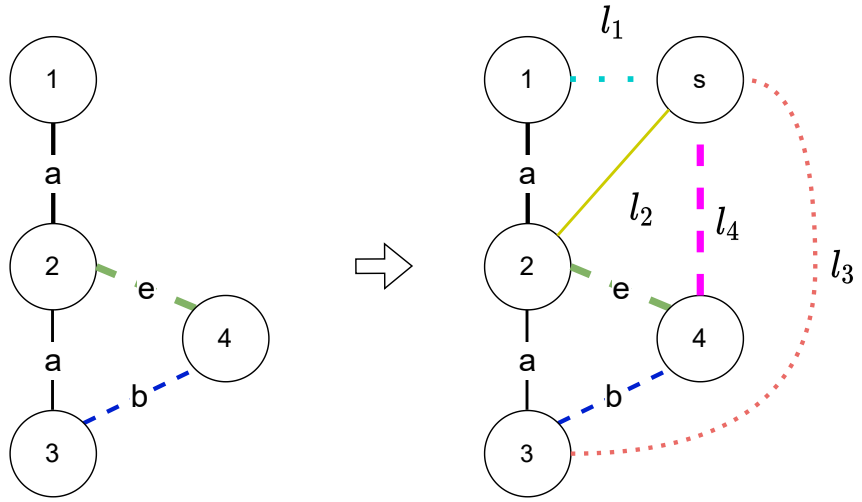

Figure 2.1 – The graph on the left is an input graph for KLSF, on the right the extended graph is illustrated, with the new node s and the edges between s and the rest of the nodes. Notice that different new labels (l_1, l_2, l_3 and l_4) were defined for the edges incident in s.

$$
\begin{aligned}
V_{G'} &= \{s\} \cup V_G \\
E_{G'} &= \{(s,u)\,|u \in V_G\} \cup E_G \\
L' &= \{l_u | u \in V_G\} \cup L \\
\ell'\left((u,v)\right) &= \begin{cases} l_u, \text{ if } v = s \\ \ell\left((u,v)\right), \text{ otherwise} \end{cases} \\
k' &= k.
\end{aligned}
$$

Let $Q = L' \setminus L$. Finding an optimal solution for an instance I of KLSF is equivalent to finding a connected graph $G'[L^* \cup Q^*]$, where $L^* \subseteq L$ and $Q^* \subseteq Q$, respecting $|L^*| \leq k$ and with the objective of minimize $|Q^*|$.

The *"labeled Cut Model"* defines a binary variable z_l for each $l \in L'$ to decide whether the label l belongs to $\ell(E_F)$, where F is a solution for I':

$$
z_l = \begin{cases} 1, \text{ if } l \text{ is in the solution} \\ 0, \text{ if } l \text{ not in the solution} \end{cases}
$$

Since the problem seeks to minimize the cardinality of Q^*, the objective function can be written as:

$$
\min \sum_{l \in Q} z_l
$$

To ensure that $G'[L^* \cup Q^*]$ is connected, for each nonempty proper subset S of nodes of G', there must be an edge connecting S to the nodes in $V_{G'} \setminus S$. The following set of constraints guarantee that condition:

$$\sum_{l \in \ell(S)} z_l \geq 1 \quad \forall\, \emptyset \neq S \subset V_{G'}$$

Finally, the number of labels of L^* cannot be larger than k. This condition can be ensured by the constraint below:

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

From the above discussion, the *"labeled Cut Model"* of [Figueredo 2020] for kLSF is modeled as follows:

$$\begin{aligned}
\min \quad & \textstyle\sum_{l \in Q} z_l && (2.1)\\
s.t.: & \\
& \textstyle\sum_{l \in \ell(S)} z_l \geq 1 \quad \forall\, \emptyset \neq S \subset V_{G'} && (2.2)\\
& \textstyle\sum_{l \in L} z_l \leq k && (2.3)\\
& z_l \in \{0,1\} \quad \forall l \in L' && (2.4)
\end{aligned}$$

The cardinality of the constraints' set 2.2 is exponential, since there exists a constraint for each nonempty proper subset of $V_{G'}$ (i.e., $2^{|V_{G'}|} - 2$ constraints). Hence, writing the complete model is inefficient even for small instances. Therefore, strategies to interactively add those constraints are required, such as the callback functions and lazy constraints mechanisms used in [Figueredo 2020].

Callback functions are used by solvers to avoid writing the complete model and iteratively including unsatisfied constraints. That is, given a relaxed solution explored by the solver, the cut callbacks proposed by [Figueredo 2020] consisted of rounding to 1 every variable with non-zero and non-integer value of this solution, and from this rounding execute a Depth First Search (DFS) procedure to verify which constraints were violated.

Another callback strategy of [Figueredo 2020] was the use of Lazy Constraints callbacks. These callbacks are used when the solution explored is an integer solution for the model, but infeasible by a restriction not added yet in the solver execution. In this

case, the constraint is added, but the addiction of it is only propagated to the branches they were violated. Similar to the cut callback functions, these constraints are identified by a DFS algorithm.

3 NP-HARDNESS AND APPROXIMABILITY

Although the kLSF problem has already been proved to be NP-hard, in this chapter a new proof regarding its NP-hardness is presented. This proof ensures that even in the simplest instances where the components of the input graph are triangles and edges, the problem remains NP-hard. Following the chapter results, it is presented an inapproximability threshold for the problem in terms of the cardinality of labels. Such a result ensures that, unless P = NP, there is no polynomial time $|L|^{\mathrm{O}(1)}$-approximation algorithm for the problem. To finish the chapter it is presented a trivial approximation for the NP-hard case where the graphs' components are triangles and edges. Our proofs of NP-hardness and inapproximability for kLSF use reductions from the 3 Exact Cover problem (X3C).

The X3C receives a set of elements and a set of triples over the elements, being the objective to answer if there exists a set cover of the elements such that each element is contained in exactly one set of the solution. The X3C can be formally defined as follows:

Problem 2. *3 Exact Cover (*X3C*).*

Input: *A tuple $I = \langle X, \mathcal{T} \rangle$, where:*

- *X is a set of elements.*
- *$\mathcal{T} \subseteq \begin{pmatrix} X \\ 3 \end{pmatrix}$[1] is a set of triples over X.*

Output: *A set $T \subseteq \mathcal{T}$, such that each element of X is contained in exactly one element of T (i.e., T is an **exact cover** of X), or to answer that such a set T does not exist.*

For the reductions we will disregard trivial cases of the problem such as, the set of elements is not divisible by 3, or cases where there is at least one element that is not covered by any triple.

3.1 NP-Hardness

In this section it is proved that the kLSF belongs to the NP-hard complexity class of problems even when the input graph is a **triangles-edges** graph (i.e., a graph where each component is an isolated triangle or edge). This result is given by the following theorem, which considers a reduction from X3C.

[1] Given a set S and a value $n \in \mathbb{N}^*$, the $\begin{pmatrix} S \\ n \end{pmatrix}$ notation represents the possible combinations of n elements of the set S

Theorem 1. KLSF *is NP-hard even when restricted to triangles-edges graphs.*

Proof. Let $I = \langle X, \mathcal{T} \rangle$ be an instance of X3C, we construct an instance $I' = \langle G, L, \ell, k \rangle$ of KLSF as described by the following steps (considering the triples are given in some order and $t < t'$ is true if triple t appears before triple t'):

1. We start the label set L with a single special label l_β, and for each triple $t \in \mathcal{T}$ we include in L a new label l_t.
2. For each element $x \in X$, we denote by $\mathcal{T}_x \subseteq \mathcal{T}$ the set of triples containing x, and by $D_x = \{\langle t, t' \rangle \,|t, t' \in \mathcal{T}_x, t < t'\}$ the set of pairs of triples in $\mathcal{T}_x$. Then, for each $x \in X$ and each pair of triples $\langle t, t' \rangle \in D_x$, we construct a triangle, including in V_G the nodes $u^x_{tt'}$, $v^x_{tt'}$ and $w^x_{tt'}$, and in E_G the edges $(u^x_{tt'}, v^x_{tt'})$, $(u^x_{tt'}, w^x_{tt'})$ and $(v^x_{tt'}, w^x_{tt'})$, respectively, with labels l_t, $l_{t'}$, and l_β (i.e., $\ell(u^x_{tt'}, v^x_{tt'}) = l_t$, $\ell(u^x_{tt'}, w^x_{tt'}) = l_{t'}$, and $\ell(v^x_{tt'}, w^x_{tt'}) = l_\beta$).
3. Considering the construction so far, we define the value $m = \max_{t \in \mathcal{T}} \left\{ \left| E_{G[\{l_t\}]} \right| \right\}$, and for each triple $t \in \mathcal{T}$ and from $i = 1$ to $i = m - \left| E_{G[\{l_t\}]} \right|$, we construct a new isolated edge by adding the nodes a^i_t and b^i_t to V_G, connected by the edge (a^i_t, b^i_t) added to E_G with label l_t (i.e., $\ell(a^i_t, b^i_t) = l_t$).
4. Also, we construct an isolated extra edge with the special label l_β, by adding to V_G the nodes a_β and b_β, and defining the edge connecting them in E_G (i.e., $\ell(a_\beta, b_\beta) = l_\beta$).
5. Finally, we set $k = \frac{|X|}{3} + 1$.

Figure 3.1 illustrates an example of the above construction, and f Formally, we define the elements of I' as follows:

$$
\begin{aligned}
k &= \frac{|X|}{3} + 1. \\
V_G &= \{u^x_{tt'}, v^x_{tt'}, w^x_{tt'} | x \in X, \text{ and } \langle t, t' \rangle \in D_x\} \\
&\quad \cup \left\{ a^i_t, b^i_t | t \in \mathcal{T}, \text{ and from } i = 1 \text{ to } i = m - \left| E_{G[\{l_t\}]} \right| \right\} \cup \{a_\beta, b_\beta\}, \\
E_G &= \{(u^x_{tt'}, v^x_{tt'}), (u^x_{tt'}, w^x_{tt'}), (v^x_{tt'}, w^x_{tt'}) |\ x \in X, \text{ and } \langle t, t' \rangle \in D_x\} \\
&\quad \cup \left\{ (a^i_t, b^i_t) | t \in \mathcal{T}, \text{ and from } i = 1 \text{ to } i = m - \left| E_{G[\{l_t\}]} \right| \right\} \cup \{(a_\beta, b_\beta)\},
\end{aligned}
$$

$$
\begin{aligned}
L &= \{l_\beta\} \cup \{l_t |\ t \in \mathcal{T}\}, \\
\ell(e) &= \begin{cases}
l_t, \text{ if } e \in \{(u^x_{tt'}, v^x_{tt'}) |\ x \in X, \langle t, t' \rangle \in D_x\}, \\
l_{t'}, \text{ if } e \in \{(v^x_{tt'}, w^x_{tt'}) |\ x \in X, \langle t, t' \rangle \in D_x\}, \\
l_t, \text{ if } e \in \left\{(a^i_t, b^i_t) |\ t \in \mathcal{T}, \text{ and from } i = 1 \text{ to } i = m - \left|E_{G[\{l_t\}]}\right|\right\}, \\
l_\beta, \text{ if } e \in \{(u^x_{tt'}, w^x_{tt'}) |\ x \in X, \langle t, t' \rangle \in D_x\} \cup \{(a_\beta, b_\beta)\},
\end{cases}
\end{aligned}
$$

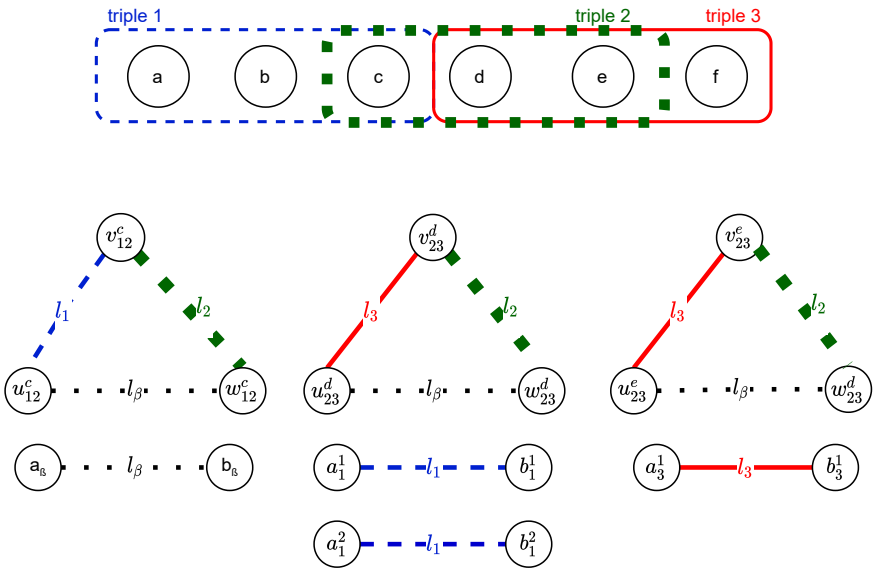

Figure 3.1 – Example of construction of an input from X3C to one input for KLSF. The 3 triples define the labels l_1, l_2, and l_3. For each element that is contained in two triples, one triangle is constructed. The first triangle refers to the element c that is contained in the triples 1 and 2 and the other two are related to the elements d and e that share the triples 2 and 3. The edges bellow the triangles are the edges defined in the step 3 of the construction, that considers $m = 3$, the maximum number of edges with the same label, defined from the label l_2.

Trivially I' is a valid instance for KLSF, where the input graph G is a triangles-edges graph. Moreover, since there are at most $|\mathcal{T}|$ triples containing a same element $x \in X$, the number of triangles associated with x is $\mathcal{O}(|\mathcal{T}|^2)$, hence there are no more than $\mathcal{O}(|X| \cdot |\mathcal{T}|^2)$ triangles in G. Also, each triple $t \in \mathcal{T}$ can share a same element with at most $|\mathcal{T}| - 1$ other triples, thus for any triple $\left|E_{G[\{l_t\}]}\right| = \mathcal{O}(|\mathcal{T}|)$ and $m = \mathcal{O}(|\mathcal{T}|)$, implying that the number of isolated edges associated with any triple is upper bounded by $\mathcal{O}(|\mathcal{T}|)$. Therefore, G has $\mathcal{O}(|X| \cdot |\mathcal{T}|^2)$ nodes and edges, and L has $\mathcal{O}(|\mathcal{T}|)$ labels. Consequently I' is polynomial on the size of I.

Below we prove that there exists an exact cover $T \subseteq \mathcal{T}$ for X3C with instance I iff there exists a solution S for KLSF with instance I', such that $|C(S)| \leq |V_G| - (\frac{|X|}{3} \cdot m + |E_{G[\{l_\beta\}]}|)$.

Given an exact cover $T \subseteq \mathcal{T}$ for X3C with instance I, we construct a solution F for KLSF with instance I' by selecting all edges labeled with l_β or with l_t for any triple

$t \in T$ (i.e., $F = G[\{l_t |\ t \in T\} \cup \{l_\beta\}]$). First, we check the properties that F must satisfy to be a feasible solution for KLSF with instance I:

- *F must be a spanning forest of G.* Since the labels $\{l_t | \forall t \in T\}$ are labels from an exact cover, that is no element of X belongs by more than one triple in T. Hence, the edges labeled by T do not form cycles, therefore, F is a forest.
- *F cannot use more than k labels.* Since T is an exact cover, it follows $|T| = \frac{|X|}{3}$, and the number of labels used by F is equal to $\frac{|X|}{3}$ plus one (the special label l_β), hence F does not use more than $k = \frac{|X|}{3} + 1$ labels.

Item 3 of the reduction ensures that, for each triple $t \in \mathcal{T}$, there are exactly m edges of G with label l_t. Since the edges labelled by $\{l_t | \forall t \in T\} \cup \{l_\beta\}$ do not form cycles, then the the number of components of F satisfies:

$$|C(F)| = |V_F| - |E_F| = |V_G| - \left(\sum_{t \in T} m + |E_{G[l_\beta]}| \right) = |V_G| - \left(\frac{|X|}{3} \cdot m + |E_{G[l_\beta]}| \right).$$

In the other direction, consider a feasible solution F for KLSF with instance I', such that $|C(F)| \leq |V_G| - \left(\frac{|X|}{3} \cdot m + |E_{G[l_\beta]}| \right)$. If the special label l_β is not associated with any edge of F, then F contains at most m edges for any used label, and since there are at most $\frac{|X|}{3} + 1$ labels, it follows:

$$|C(F)| = |V_F| - |E_F| \geq |V_G| - \left(\frac{|X|}{3} + 1 \right) \cdot m > |V_G| - \left(\frac{|X|}{3} \cdot m + |E_{G[l_\beta]}| \right).$$

The last inequality is given by the fact that $|E_{G[l_\beta]}| > m$, because m is upper bounded by the number of triangles and l_β appears on each triangle plus one extra edge added by Item 4. The above result denies our assumption on $|C(F)|$, hence F must contain edges labeled with l_β. Moreover, for each triple $t \in T$, there are m edges constructed with label l_t in G, implying that $|C(F)| \leq |V_G| - \left(\frac{|X|}{3} \cdot m + |E_{G[l_\beta]}| \right)$ only if F contains all edges with label l_β and all $m \cdot \frac{|X|}{3}$ edges with labels associated to $\frac{|X|}{3}$ different triples of $\mathcal{T}$. As F is acyclic, the $\frac{|X|}{3}$ triples whose labels are associated with edges of F are disjoint and form an exact cover for I.

Since X3C is an NP-complete problem [Garey and Johnson 1979], it follows that KLSF is NP-hard even when the input graph is a triangles-edges graph.

□

3.2 Inapproximability

In this section we present a strong inapproximability result for KLSF. We give a reduction from X3C to KLSF establishing that, unless P = NP, there does not exists a $|L|^{\mathcal{O}(1)}$-approximation algorithm for the KLSF.

The reduction uses the Walecki decomposition concept [Alspach 2008] to create disjoint Hamiltonian paths. Given a complete graph K_n with n nodes, this decomposition partitions the edges in $\lfloor \frac{n}{2} \rfloor$ edge-disjoint paths (Hamiltonian paths). In Figure 3.2 a visualization of the Walecki construction can be seen, and the Algorithm 2 describes the steps of this construction.

Algorithm 2 Walecki Decomposition

Input: A complete graph K_n with nodes $V_{K_n} = \{v_0, v_1, \ldots, v_{n-1}\}$
Output: A set of Hamiltonian disjoint paths P from K_n.

```
1:  P ← {}
2:  for root = n/2 to n − 1 do
3:      E ← {}
4:      right ← (root + 1)(mod n)
5:      E ← E ∪ {v_root, v_right}
6:      left ← root − 1
7:      for i from 0 to n/2 do
8:          E ← E ∪ {v_right, v_left}
9:          right ← (right + 1)(mod n)
10:         E ← E ∪ {v_left, v_right}
11:         left ← left − 1
12:     end for
13:     Add E to P
14: end for
```

The following theorem not only shows that KLSF is an NP-hard problem, but also proves that it cannot be approximated with a ratio less than or equal to a polynomial function on the number of labels.

Theorem 2. *Unless P = NP,* KLSF *does not admit a polynomial time $|L|^{\mathcal{O}(1)}$-approximation algorithm for every instance $I = \langle G, L, \ell, k \rangle$.*

Proof. Let us take an integer $\theta \geq 1$ and define the value $\alpha = 2 \times |\mathcal{T}|^{\theta} \times |X|$. Given $I = \langle X, \mathcal{T} \rangle$ an instance for X3C, it is constructed an instance $I' = \langle G, L, \ell, k \rangle$ for KLSF as follows:

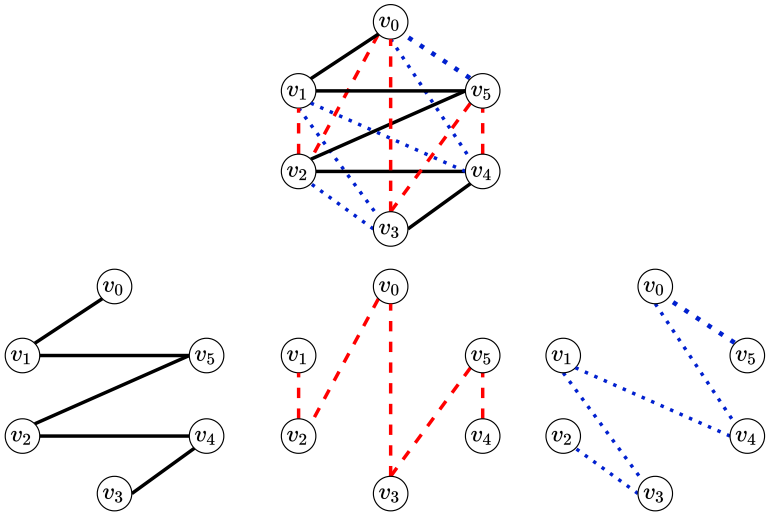

Figure 3.2 – An example of all 3 paths generated by the Walecki construction with K_6.

1. For each element the nodes of a different K_α complete graph are added to V_G. Let us reference these nodes as v_{xi} for each pair of element $x \in X$ and $0 \leq i \leq \alpha - 1$.
2. A total of $|X|$ Walecki decompositions are applied to the complete graph K_α defined for each element. Denote the set of Hamiltonian paths constructed for each element $x \in X$ as P_x. For each triple $t \in \mathcal{T}$ there exists at least one Hamiltonian path in P_x, since the Walecki decomposition ensures $\lfloor \frac{\alpha}{2} \rfloor = |\mathcal{T}| \times |X|$ Hamiltonian paths disjoint, considering this, for each triple $t \in \mathcal{T}$ a disjoint path in P_x is mapped to t. Denote such a path by $p_{xt} \in P_x$.
3. For each triple $t \in \mathcal{T}$ we define a label $l_t \in L$ which we name by triple-label of t.
4. For each triple $t = \{x, y, z\} \in \mathcal{T}$ the paths p_{xt}, p_{yt}, and p_{zt} of the Walecki decompositions are added to E_G. The edges of each of these paths are labeled with label l_t. (i.e., $\ell(e) = l_t$ for all $e \in p_{xt} \cup p_{yt} \cup p_{zt}$).
5. Finally, it is defined $k = \frac{|X|}{3}$

Formally the construction of I' is:

$$
\begin{aligned}
V &= \{v_{xi} \mid \forall x \in X \text{ and } 1 \leq i \leq \alpha\} \\
L &= \{l_t \mid \forall t \in \mathcal{T}\} \\
E &= \{e \mid \forall e \in p_{xt} \cup p_{yt} \cup p_{zt}, \forall t = \{x, y, z\} \in \mathcal{T}\} \\
\ell(e) &= l_t \mid \text{if } e \in p_{xt} \cup p_{yt} \cup p_{zt}, \text{ for some } t = \{x, y, z\} \in \mathcal{T} \\
k &= \frac{|X|}{3}
\end{aligned}
$$

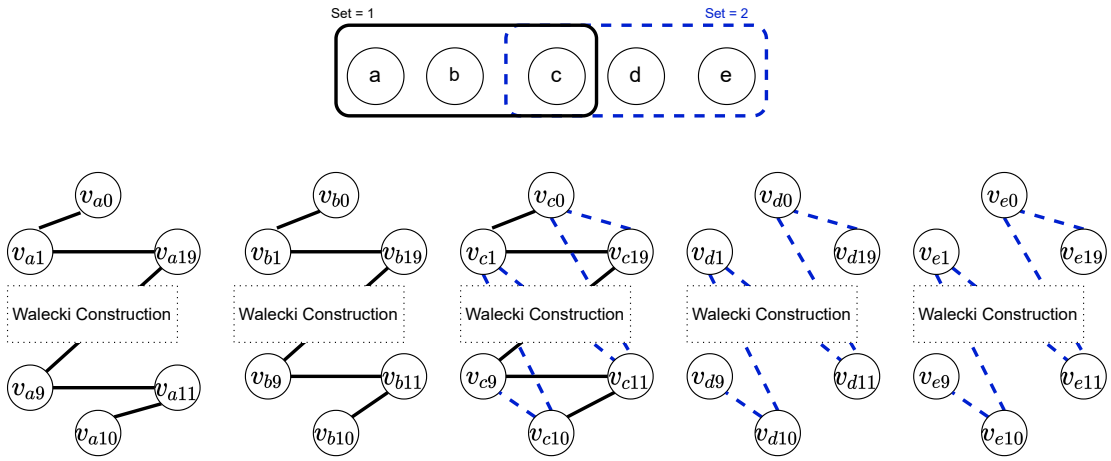

Figure 3.3 – An example of the construction from an instance of X3C to an instance of KLSF, where $|\mathcal{T}| = 2$, $|X| = 5$ and $\theta = 1$.

In Figure 3.3 is presented an example of this construction.

Trivially I' is a graph labeled on edges and a valid instance for KLSF. The complexity of addition of nodes of the complete graph K_α for each element is $\mathcal{O}(\alpha \cdot |X|)$ of time and space. Since each Hamiltonian has $\alpha - 1$ edges, adding the edges of the $3 \cdot |\mathcal{T}|$ paths to E use $\mathcal{O}(\alpha \cdot |\mathcal{T}|)$ also for time and space. Since α is polynomial in function of $|\mathcal{T}|^\theta \cdot |X|$, the whole construction uses $\mathcal{O}(|\mathcal{T}|^\theta \cdot |X|^2 + |\mathcal{T}|^{2\theta} \cdot |X|)$.

Now we prove that a solution $T \subseteq \mathcal{T}$ for X3C with instance I exists iff there exists a solution F for KLSF with instance I' such that the number of components of the solution is $|C(F)| \leq |X|$.

Given an exact cover $T \subseteq \mathcal{T}$ for X3C, a forest F for KLSF is constructed using the edges of the referent triple labels in L (i.e., $F = G\left[\{l_t | t \in T\}\right]$). To validate this solution we check two properties that F must satisfy.

- *F must be a spanning forest of G.* Since the labels $\{l_t | \forall t \in T\}$ are from an exactly cover, that is no one element in X is shared for more than one triple in T. This means that for each element exists only one path along its nodes mapped for a label in this set, therefore the edges of F do not form any cycle, thus F is a forest.
- *F cannot use more than k labels.* Since the labels used are defined from T and $|T| = |X|/3 \leq k$ this property is trivially satisfied.

Proved that F is a valid solution for KLSF, now we prove that the cardinality of the components that compose F satisfies $|C(F)| = |X|$. For each element $x \in X$, all of its α nodes will be connected in F with the edges labeled by at least one triple label in

$\{l_t | \forall t \in T\}$ (i.e. for each $x \in X$ exist a $t \in T$ such that $p(x,t) \in E_F$). Since the sets of nodes defined for the elements in X are disconnected from each other, the number of components of F will be:

$$|C(F)| = |X|$$

For the other direction, let F be a feasible solution for KLSF with I' such that $|C(F)| \leq |X|$. Suppose that there are two different labels whose edges in F connect the nodes of a same element in X. This means that these edges will connect at most a set of nodes corresponding to 5 elements. Consequently, there are at most $k - 2 = \frac{|X|}{3} - 2$ remaining labels to connect all $|X| - 5$ remaining sets for $|C(F)| \leq |X|$. However, the edges associated with $\frac{|X|}{3} - 2$ labels can connect at most the nodes of $(\frac{|X|}{3} - 2) \cdot 3 = |X| - 6$ sets of elements, hence at least the nodes of one of the elements of X will be disconnected. Therefore, F would have more than $(|X| - 1) + \alpha$ components.

The above discussion proves that the edges of two labels of $\ell(F)$ cannot connect the nodes of a same element. Thus, the triples associated to the labels of $\ell(F)$ are disjoint. Furthermore, since $|C(F)| \leq |X|$, the nodes of each element set are connected and the triples to the labels of $\ell(F)$ form an exact cover of X.

For the inapproximability result, suppose that KLSF admits a polynomial time $|L|^{\beta}$-approximation algorithm, for some constant $\beta > 0$. Considering $\theta = \lceil \beta \rceil$, such an algorithm computes a solution S for KLSF with instance I', satisfying:

$$|C(S)| \leq |L|^{\beta} \times |C(F)| = |\mathcal{T}|^{\beta} \times |C(F)|$$

If there exists an exact cover for X3C with instance I', then $|C(F)| \leq |X|$, implying that:

$$|C(S)| \leq |\mathcal{T}|^{\beta} \times |C(F)| \leq |\mathcal{T}|^{\beta} \times |X|$$

If the nodes of an element are not connected by the edges of a triple label in S, then each of the $\alpha = 2 \times |\mathcal{T}|^{\theta} \times |X|$ nodes in that set will form a single component and:

$$|C(S)| \geq \alpha + (|X| - 1) = 2 \times |\mathcal{T}|^{\theta} \times |X| + (|X| - 1) > |\mathcal{T}|^{\beta} \times |X|$$

Consequently, for $|C(S)| \leq |\mathcal{T}|^{\beta} \times |X|$ to be satisfied, for each element the set of

nodes must be connected by the edges associated with one triple in $\ell(S)$. Thus, the triples related to $\ell(S)$ form a cover for X3C with instance I. This implies that a polynomial time $|L|^{\beta}$-approximation algorithm for KLSF would be a polynomial time algorithm for X3C since the reduction would be solved by this algorithm when considered $\theta = \lceil \beta \rceil$. Therefore, unless P = NP, the KLSF does not admit a polynomial time algorithm with approximation ratio less than or equal to $|L|^{\beta}$, for any constant $\beta > 0$.

□

3.3 Approximation for triangle and edges graph cases

In this section a trivial approximation is given for triangles-edges graphs.

Theorem 3. *Any solution for the* KLSF *over triangles-edges graphs is a* 3*-approximation of the optimal value.*

Proof. Let $I = \langle G, L, \ell, k \rangle$ be an instance of KLSF and F^* an optimal solution for KLSF with I. A lower bound for the number of components of F^* is the number of components of G:

$$|C(F^*)| \geq |C(G)|$$

Considering that the input graph G for KLSF is a triangles-edges graph, each component of G has at most three nodes and:

$$|C(F^*)| \geq |C(G)| \geq \frac{|V_G|}{3}$$

The number of components of any feasible solution F of KLSF with I is bounded by the number of nodes of G:

$$|C(F)| \leq |V_G|$$

Hence,

$$|C(F^*)| \geq \frac{|C(F)|}{3}$$

Concluding that any solution of KLSF is a 3-approximation of the optimal value.

□

4 FIX-AND-OPTIMIZE PROPOSAL

Heuristics and metaheuristics offer practical strategies for dealing with the scalability of difficult problems. However, for simpler instances and subproblems, the exact methods are capable of reaching optimal solutions in short time. Hence, several examples in the literature used the combination of exact methods with the capability of heuristics to explore the solution space. Among these strategies, the fix-and-optimize matheuristics emerged as a highly competitive approach.

A fix-and-optimize algorithm iteratively fixes the values of some variables by heuristic decisions, generating subproblems that can be exactly solved by mixed-integer linear programming tools. Some examples are fix-and-optimize approaches for a production chain problem [Pochet and Wolsey 2006], a multi-level capacitated lot-sizing problem [Helber and Sahling 2010, Li, Song and Wu 2015], and a school timetabling problem [Dorneles, de Araújo and Buriol 2014, Lindahl, Sørensen and Stidsen 2018]. Those results motivated us to explore the applicability of a fix-and-optimize strategy for the KLSF.

Our fix-and-optimize matheuristic consists of fixing some variables associated with labels through heuristics decisions, in order to generate a simpler subproblem that can be exactly solved by a mixed-integer linear programming solver (e.g., CPLEX [Cplex 2009], Gurobi [Gurobi Optimization, LLC 2021], GLPK [Makhorin]). This process of fixing variables and solving subproblems is done iteratively with the objective of exploring the solution space.

This chapter is divided in four sections. Section 4.1 presents a model for the problem that we proposed for being inserted in the matheuristic. Section 4.2 presents the description of the fix-and-optimize proposal for the KLSF followed by Section 4.3 that describes the experimental results obtained by our algorithm and Section 4.7 that compares the results with other algorithms from the literature.

4.1 Component model

The model here proposed is a binary model with three sets of variables. The first set of variables are the variables z_l, defined for each $l \in L$ and representing if the label l is in the solution.

$$z_l = \begin{cases} 1, \text{ if } l \text{ is in the solution} \\ 0, \text{ if } l \text{ not in the solution} \end{cases}$$

The second set contains the variables x_{uw}, defined for all combinations $(u, w) \in V_G^2$ and representing if the node u is in the component constructed from the node w.

$$x_{uw} = \begin{cases} 1, \text{ if the node } u \text{ is in the component } w \\ 0, \text{ if } u \text{ is not in the component } w \end{cases}$$

The last set of binary variables are defined over each component w and consists of the variables y_w, representing if the component is empty or not.

$$y_w = \begin{cases} 1, \text{ if the component } w \text{ has node} \\ 0, \text{ if } w \text{ is a empty component} \end{cases}$$

The objective of the problem is to minimize the number of trees (components) in the forest, then, by using the above variables the objective function can be written as:

$$\min \sum_{w \in V} y_w$$

The first set of constrains of the model ensures that if a subset of nodes $X \subseteq V_G$ belong to the same component $w \in V_G$, then there exist a label in the colored cut of $\ell(X)$ in the solution:

$$\sum_{u \in V} x_{uw} - \sum_{v \in X} x_{vw} \leq |V| \sum_{l \in \ell(X)} z_l \; \forall w \in V_G \text{ and } \forall X \subset V_G$$

The second set of constraints ensures that if a component $w \in V_G$ has nodes then y_w is equal to 1:

$$y_w |V| \geq \sum_{u \in V} x_{uw} \; \forall w \in V_G$$

The last constraints' set of the problem limits the number of labels in the solution to the upper bound value k:

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

From the above discussion, the "Component Model" for KLSF is formulated as

follows:

$$\min \quad \sum_{w \in V} y_w \tag{4.1}$$

$$s.t.:$$

$$\sum_{u \in V} x_{uw} - \sum_{v \in X} x_{vw} \leq |V| \sum_{l \in \ell(X)} z_l \quad \forall w \in V_G \text{ and } \forall X \subset V_G \tag{4.2}$$

$$y_w |V| \geq \sum_{u \in V} x_{uw} \quad \forall w \in V_G \tag{4.3}$$

$$\sum_{l \in L} z_l \leq k \tag{4.4}$$

$$z_l \in \{0, 1\} \quad \forall l \in L \tag{4.5}$$

$$x_{uw} \in \{0, 1\} \quad \forall u, w \in V_G \tag{4.6}$$

$$y_w \in \{0, 1\} \quad \forall w \in V_G \tag{4.7}$$

Since the above model has many symmetries, we also consider the following set of constraints:

$$y_w > y_c \; \forall w, c \in V_G \text{ and } w < c$$

In those constraints, $w < c$ represents the precedence between two components on a sorted set, and ensures a component will only exist if the previous one already have nodes (avoiding solutions with empty components at the beginning).

The "Component Model" was tested with the matheuristic, but it did not present good results, so it was discarded and replaced by the *labelled cut model* proposed by [Figueredo 2020] (see Section 2.2.3).

4.2 Fix-and-optimize matheuristic

In this section we describe our fix-and-optimize proposal for the KLSF and in Algorithm 3 it is presented each step of this proposal.

Algorithm 3 receives as input an instance $I = \langle G, L, \ell, k \rangle$ of KLSF, returning at line 26 a feasible solution F^* with the least number of components among the explored solutions. The strategy to explore the solution space consists of constructing a subproblem from a current solution F. This solution is initialized at line 1 through the greedy algorithm MVCA. After the initialization, the value of F is only updated at line 17 if a new solution with fewer components is found or if the number of iterations without

Algorithm 3 Fix-and-optimize

Input: $I = \langle G, L, \ell, k \rangle$ 55555
Output: Feasible solution F^* for KLSF with instance I

$F \leftarrow$ solution of MVCA with I.
$F^* \leftarrow F$
$i \leftarrow 0$
for j from 0 to MAX_ITER and while $|C(F^*)| > 1$ **do**
    $L_\alpha \leftarrow \lfloor \alpha \cdot |\ell(F)| \rfloor$ random labels of $\ell(F)$
    $G_{con} \leftarrow$ contract in G the edges of $E[L_\alpha]$
    $L_\beta \leftarrow \lfloor \beta \cdot |\ell(F) \setminus L_\alpha| \rfloor$ random labels of $\ell(F) \setminus L_\alpha$
    $L_\theta \leftarrow \lfloor \theta \cdot |L \setminus \ell(F)| \rfloor$ random labels of $L \setminus \ell(F)$
    $L_{con} \leftarrow \{L_\beta \cup L_\theta\}$
    $G_{con} \leftarrow$ remove from G_{con} the edges of $E[L \setminus L_{con}]$
    $k_{con} \leftarrow k - |L_\alpha|$
    $I_{con} \leftarrow \langle G_{con}, L_{con}, \ell, k_{con} \rangle$
    $I' \leftarrow$ extended instance of I_{con}
    $L' \leftarrow$ solver's solution for I'
    $F_{aux} \leftarrow$ spanning forest of $G[L_\alpha \cup L']$
    if $|C(F_{aux})| < |C(F)|$ or $i = IN\_SOL$ **then**
        $F \leftarrow F_{aux}$
        $i \leftarrow 0$
    else
        $i \leftarrow i + 1$
    end if
    if $|C(F_{aux})| < |C(F^*)|$ **then**
        $F^* \leftarrow F_{aux}$
    end if
end for
return F^*

changing its value reaches the given boundary parameter IN_SOL (see line 16).

At each iteration of the main loop (between lines 5-15), a new feasible solution F_{aux} is constructed. The computation of F_{aux} depends on solving a subproblem obtained from the current solution F and the input instance I. The construction of the subproblem starts with a random selection of the labels to be fixed, which depends on a rational parameter of the algorithm $\alpha \in [0, 1)$. At line 5, the subset L_α (with the labels to be fixed) is obtained by randomly choosing $\lfloor \alpha \cdot |\ell(F)| \rfloor$ labels of $\ell(F)$ (the labels in the current solution F). Fixing a label l consists of contracting the edges in E_G associated with l (line 6). An edge contraction is the operation of merging its endpoints in a single node and removing the edge. This operation generates parallel edges whenever the two merged nodes share common neighbors. Hence, the generated subproblem considers multigraphs (i.e., graphs with parallel edges). Figure 4.1 illustrates an example of a sequence of contractions.

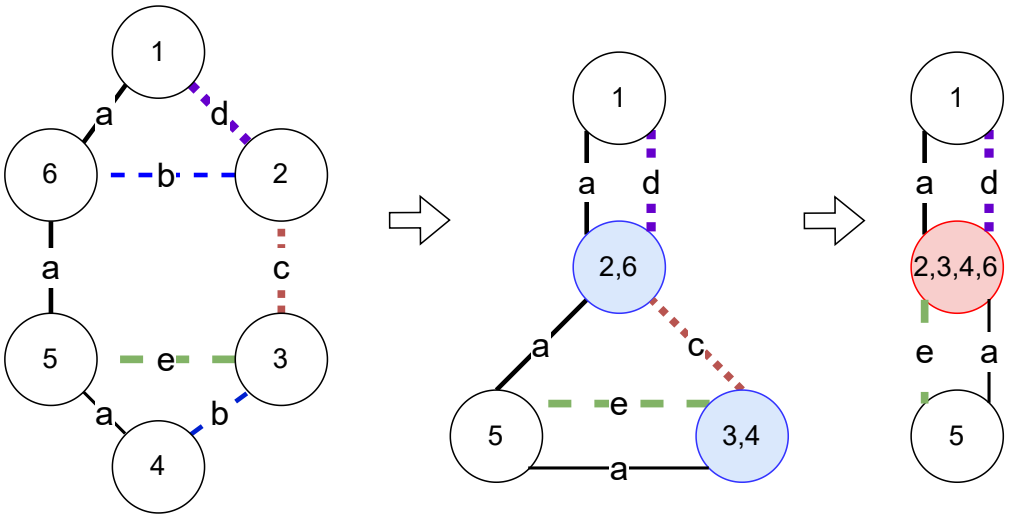

Figure 4.1 – Example of how a sequence of labels is fixed. In the first step, the label b is chosen to be fixed, contracting the edges $(2, 6)$ and $(3, 4)$ and producing two merged nodes. In the second step the label c is chosen to be fixed, and the edge between the two previously merged nodes is contracted.

Despite reducing the complexity after the graph compression at line 6, the subproblem may still be hard to solve. Thus, we apply a second reduction, involving the removal of a set of labels from the instance and the corresponding edges from the graph. Instead of selecting the labels to remove, we define (at line 9) the set $L_{con} = L_\beta \cup L_\theta$ containing the labels that will be maintained in the subproblem. Notice that L_{con} depends on the sets L_β and L_θ which, respectively, contain the labels of $\ell(F) \setminus L_\alpha$ and $L \setminus \ell(F)$ that will not be removed from G_{con} and their computation size is analogous to the set L_α, but depending on the rational algorithm's parameters $\beta, \theta \in [0, 1)$ (lines 7 and 8). Following

the construction of L_{con}, the second reduction is applied at line 10, removing every edge of the compressed graph with labels outside L_{con}.

To ensure that the number of labels in a subproblem's solution joined to those in L_α do not exceed the input value k, a new upper bound for the cardinality of the labels in a feasible solution of the subproblem is defined at line 11. After defining all the subproblem parameters, a new simpler instance I_{con} and its extended counterpart I' are defined at lines 12 and 13. To obtain the solution L', the extended instance I' is solved exactly at line 14 by using the model and techniques mentioned in Section 4.2.

Since the algorithm accepts only a better solution than the current F in the steps where $i < IN\_SOL$, for these steps we add the following extra constraint to the model:

$$\sum_{l \in Q} z_l \leq |C(F)|$$

The above constraint addition refines the feasible solutions' space, considering only those solutions capable of improving the current one. Consequently, the space search is reduced and the solver gains efficiency. Finally, at line 15, the resulting forest of a DFS on $G\left[L' \cup L_\alpha\right]$ is computed and assigned to the new solution F_{aux}.

The stop criteria for the main loop of the algorithm (at line 4) considers a maximum number of iterations (given by the algorithm's parameter MAX_ITER) and a lower bound for the number of components in the best solution found (where any solution must have at least one component).

4.3 Computational experiments

To assess our fix-and-optimize proposal, this section presents computational experiments over instances from the literature, comparing the results with previous algorithmic approaches for the KLSF. In the following subsections we describe the computing environment, the instances, and the choice of parameters. Finally, we present and analyze the results of our tests.

All our implementations and the tests results are available at a repository in the GitHub platform [Github].

4.4 Computing environment and library

The proposed fix-and-optimize method was implemented in the C programming language using the compiler g++ 9.3.0, and all tests were executed in a processor AMD Ryzen 9 3900X with 12 cores of 3.8 GHz each, and 32 GB of RAM, under Ubuntu Linux 20.04 LTS 64 bits. To generate pseudo-random values we used the default C function rand(). Despite the machine having 12 cores, only one core was used to run the proposed algorithm for each test.

To generate a comparative result with the algorithms proposed by [Cerulli et al. 2014] in their machine, we adjusted the running times. For this purpose, we used the PassMark Benchmark dataset [PassMark CPU benchmark dataset] that allows us to estimate and compare the running times on different machines. The machine specified by [Cerulli et al. 2014] was an Intel Xeon 2.8 GHz processor, which can be compared with our machine based on the values in Table 4.1, available at [PassMark CPU benchmark dataset].

Table 4.1 – Result of the comparison between the machine of [Cerulli et al. 2014] and the machine used in this work. The column "PassMark Score" is the value of the machine in the [PassMark CPU benchmark dataset], while the column "Factor" represents the factor difference between the two machines

Machine	PassMark Score	Factor
Intel Xeon 2.8 GHz	2727	7.1015625
AMD Ryzen 9 3900X	384	1.0000000

Since our algorithm uses pseudo-random generation, for each instance we executed each algorithm 30 times, with different integer seeds from 1 to 30.

4.5 Instances

For the experiments, we considered the dataset from [Cerulli et al. 2014] with 200 instances organized in five groups. The groups were identified by the number of nodes $|V| \in \{100, 150, 200, 400, 500\}$ and divided in four subsets, each one with 10 instances. Each subset was identified by their group ($|V|$) and the number of labels ($|L| \in \left\{\frac{|V|}{4}, \frac{|V|}{2}, |V|, 1.25|V|\right\}$). The instances' generation was finalized by adding $\frac{(|V|-1)|V|}{10}$ edges to their graphs and setting k equal to $\lfloor |V|/2^i \rfloor$, where i represents the smallest value such that the MVCA's solution for the instance is greater than 1.

4.6 Parameters

Our fix-and-optimize algorithm requires the setting of five correlated parameters: 2 integer parameters (MAX_ITER and IN_SOL), and 3 rational parameters (α, β, and θ).

The MAX_ITER and IN_SOL parameters determine, respectively, the algorithm's maximum number of iterations and maximum number of iterations with the same current solution. Since instances with lower values of k (usually easier to solve) require fewer iterations to achieve *good enough* solutions, we defined these parameters to be directly proportional to the value of k, depending on a sixth and seventh parameters (M_1 and M_2) as follows:

$$\begin{aligned} MAX\_ITER &= k \cdot M_1 \\ IN\_SOL &= k \cdot M_2 \end{aligned}$$

The value of θ defines the number of new labels (not used in the current solution) added to the subproblem, whose complexity depends on the graph's edge density. Therefore, for a solver to find a subproblem's solution within reasonable time, the value of θ should guarantee to generate graphs with low density. For dense graphs $|E| = O(|V|^2)$, hence by setting $\theta = \frac{|L|}{|V|}$, there will be selected at most $\frac{|L|^2}{|V|}$ labels, and assuming an uniform distribution of edges per labels, the average number of edges in the subproblems will be bounded by $O(|V| \cdot |L|)$. To ensure the low density of the subproblem's graphs, we set the value of $\theta = 0.8$ for larger values of $|L|$ ($|L| > 0.8 \cdot |V|$):

$$\theta = \begin{cases} \frac{|L|}{|V|}, \text{ if } \frac{|L|}{|V|} < 0.8 \\ 0.8, \text{ otherwise.} \end{cases}$$

To define the parameters α and β we executed several tests with manual and automatic values selection. The automatic tests were run with the Irace package [López-Ibáñez et al. 2016], and both parameters values were selected from the set $\{0.1, 0.2, ..., 0.8, 0.9\}$. All the experiments showed that the algorithm achieved better results when both parameters' values were equal to 0.8 ($\alpha = \beta = 0.8$).

Finally, we used the Irace package for defining the two remaining parameters M_1 and M_2. Table 4.2 shows the results of these tests, highlighting on gray background the

chosen configuration for the computational experiments we discuss in the next section.

Table 4.2 – Result of combination of parameters for the fix-and-optimize algorithm, in gray background the configuration chosen to represent in Section 4.7

M_1	M_2	sol.	time(s)
75	2.0	11.99	9.53
80	2.0	11.98	10.26
80	3.0	12.01	9.16
70	1.5	11.99	10.44
100	1.5	11.91	14.53
200	1.5	11.80	27.47

4.7 Results and analysis

This section presents and analyzes the experimental results generated by our fix-and-optimize algorithm, as well as it presents results of other algorithms from the literature that used the same set of instances. Table 4.3 summarizes these results. The first set of columns ($|V|$, $|E|$, $|L|$, and k) presents characteristics of the instances. The next four groups of columns, named SA, RTS, GA and PILOT, present average solution values (sol.) and computational time (time (s)) for the algorithms from [Cerulli et al. 2014]. Our results are under the column FO, which presents the average solution value (sol.) and the computational time in seconds (time (s)) of the the executions with all seeds, and the average solution value with the seed with best average values (seed=1, sol.(1)).

Table 4.3 – Results of the 5 algorithms developed for the KLSF.

Dataset				SA		RTS		GA		PILOT		FO		
$\|V\|$	$\|E\|$	$\|L\|$	k	sol.	time (s)	sol.	time (s)	sol.	time (s)	sol.	time (s)	sol.	time (s)	sol. (1)
100	990	25	3	**6.30**	0.08	**6.30**	0.08	**6.30**	0.25	**6.30**	0.00	**6.30**	0.18	**6.30**
100	990	50	6	2.70	0.16	2.90	0.11	**2.60**	0.42	2.70	0.08	2.66	0.65	**2.60**
100	990	100	6	**15.00**	0.38	**15.00**	0.39	**15.00**	0.67	15.60	0.31	15.12	8.12	**15.00**
100	990	125	7	**15.70**	0.53	**15.70**	0.54	**15.70**	0.87	**15.70**	0.59	15.76	7.07	**15.70**
150	2235	37	4	**3.50**	0.19	**3.50**	0.13	**3.50**	0.52	**3.50**	0.04	3.58	0.12	3.60
150	2235	75	4	**22.30**	0.40	**22.30**	0.40	**22.30**	0.80	**22.30**	0.13	22.32	1.07	**22.30**
150	2235	150	9	7.70	0.97	**7.30**	0.84	8.10	2.93	8.30	1.31	7.42	3.18	7.50
150	2235	187	11	6.10	1.31	6.20	1.10	6.20	3.28	6.10	1.86	5.78	5.73	**5.70**
200	3980	50	3	**17.00**	0.37	**17.00**	0.37	17.20	0.93	**17.00**	0.05	17.22	0.52	17.20
200	3980	100	6	**9.30**	0.84	**9.30**	0.84	9.60	2.56	9.60	0.87	9.42	5.01	9.40
200	3980	200	12	2.60	1.56	3.10	1.07	3.10	5.03	3.00	2.72	2.48	3.09	**2.40**
200	3980	250	15	1.80	1.66	1.60	1.20	1.50	4.87	**1.20**	5.35	1.27	1.46	1.30
400	15960	100	3	**35.60**	5.15	**35.60**	5.17	**35.60**	4.47	**35.60**	1.38	35.75	1.85	**35.60**
400	15960	200	6	24.40	11.25	**23.70**	11.35	25.10	12.32	24.00	16.55	23.87	34.73	**23.70**
400	15960	400	12	12.10	25.42	11.40	21.66	12.50	37.47	11.10	81.83	**10.60**	33.55	**10.60**
400	15960	500	15	8.70	34.22	8.20	21.24	8.90	48.92	7.70	163.04	7.28	13.84	**7.00**
500	24950	125	4	**22.70**	13.11	**22.70**	13.18	22.90	10.66	**22.70**	8.34	23.41	1.12	23.00
500	24950	250	7	22.40	27.54	**21.50**	27.75	22.80	24.75	21.90	33.10	21.84	28.15	21.70
500	24950	500	15	6.30	47.23	5.30	42.05	6.60	75.62	5.00	251.83	5.36	19.10	**4.90**
500	24950	625	19	3.40	52.67	3.00	38.56	3.40	111.03	**2.10**	214.08	2.43	22.06	**2.10**
average				12.28	11.25	12.08	9.40	12.45	17.42	12.07	39.17	11.99	9.53	**11.88**

For a fair comparison of the results, the original times from [Cerulli et al. 2014] for algorithms SA, RTS, GA, and PILOT were divided by 7.11 (see Table 4.1). Hence, the computational times in Table 4.3 for those algorithms are much smaller than those reported in [Cerulli et al. 2014]. After normalizing times it is still clear that the results found by FO are promising. Next we analyse and discuss these results in details.

Regarding the execution times, unlike the algorithms from [Cerulli et al. 2014] whose running times grow with the value of k, FO running times do not exhibit the same behavior. This can be seen in all instance groups. Also, when $|V| \geq 150$, $|L| \geq 100$ and $k \geq 11$, at least in 85% of the occasions (6 of 7 subsets), Table 4.3 reports better average solution values from our fix-and-optimize matheuristic than those proposed by [Cerulli et al. 2014], except for the PILOT method. Thus, for the large instances, FO shows a better compromise between solution quality and running times when comparing with [Cerulli et al. 2014].

Table 4.4 – Percentage of tests where the fix-and-optimize found a better solution for all seeds in relation to the results of [Cerulli et al. 2014].

$\|V\|$	$\|E\|$	$\|L\|$	k	SA(%)	RTS(%)	GA(%)	PILOT(%)
100	990	25	3	100.00	100.00	100.00	100.00
100	990	50	6	86.21	100.00	51.72	86.21
100	990	100	6	72.41	72.41	72.41	89.66
100	990	125	7	79.31	79.31	79.31	79.31
150	2235	37	4	51.72	51.72	51.72	51.72
150	2235	75	4	79.31	79.31	79.31	79.31
150	2235	150	9	100.00	17.24	100.00	100.00
150	2235	187	11	93.10	93.10	93.10	93.10
200	3980	50	3	10.34	10.34	51.72	10.34
200	3980	100	6	34.48	34.48	96.55	96.55
200	3980	200	12	82.76	100.00	100.00	100.00
200	3980	250	15	100.00	100.00	100.00	44.83
400	15960	100	3	55.17	55.17	55.17	55.17
400	15960	200	6	96.55	31.03	100.00	82.76
400	15960	400	12	100.00	96.55	100.00	86.21
400	15960	500	15	100.00	100.00	100.00	89.66
500	24950	125	4	0.00	0.00	0.00	0.00
500	24950	250	7	100.00	3.45	100.00	68.97
500	24950	500	15	100.00	48.28	100.00	13.79
500	24950	625	19	100.00	100.00	100.00	3.45
average				77.07	63.62	81.55	66.55

Table 4.4 shows the percentage of executions in which our algorithm found a solution with value better than or equal to those obtained by the other algorithms. According to the average value shown in the last row of the table, our FO outperformed all other

algorithms in more than 60% of the tests. That percentage increases above 90% when considering $k > 10$ and the algorithms SA, RTS and GA. These gains can be justified by our choice of $\alpha = 0.8$, guaranteeing that at most 20% of k labels will be chosen by a subproblem solution. This gains in relation to the growth of k can be justified by the fact that for smaller instances (e.g., $k < 10$) only 1 or 2 labels would be chosen by the subproblem, while for the larger values of k the subproblems complexity increase choosing from 3 to 5 labels.

In Table 4.5 we report the best known solution (BK) for each subset of 10 instances, the average solution value (sol.) and standard deviation (σ) for the 30 execution of our fix-and-optimize algorithm, besides the percentage of runs FO found the best known solution (FBK).

Table 4.5 – Best known combination of solutions for each group of instances (BK), average of the 30 runs (sol.) and the mean standard deviation of the 30 runs (σ) of the fix-and-optimize algorithm. The percentage of different seeds that FO found the best known combination of solutions is presented in column FBK(%).

$\|V\|$	$\|E\|$	$\|L\|$	k	BK	sol.	σ	FBK(%)
100	990	25	3	6.30	6.30	0.00	100.00
100	990	50	6	2.60	2.66	0.11	51.72
100	990	100	6	15.00	15.12	0.47	72.41
100	990	125	7	15.70	15.76	0.28	79.31
150	2235	37	4	3.50	3.58	0.14	51.72
150	2235	75	4	22.30	22.32	0.08	79.31
150	2235	150	9	7.10	7.42	0.39	0.00
150	2235	187	11	5.70	5.78	0.31	65.52
200	3980	50	3	17.00	17.22	0.34	10.34
200	3980	100	6	9.30	9.42	0.22	34.48
200	3980	200	12	2.10	2.48	0.39	3.45
200	3980	250	15	1.20	1.27	0.15	44.83
400	15960	100	3	35.60	35.75	0.45	55.17
400	15960	200	6	23.30	23.87	0.73	0.00
400	15960	400	12	9.60	10.60	1.03	0.00
400	15960	500	15	6.20	7.28	0.76	0.00
500	24950	125	4	22.40	23.41	0.76	0.00
500	24950	250	7	21.10	21.84	0.67	0.00
500	24950	500	15	4.00	5.36	0.70	0.00
500	24950	625	19	1.80	2.43	0.39	0.00
average				11.59	11.99	0.42	32.41

From the percentages in Table 4.5 we conclude that our FO matheuristic is consistent with an average standard deviation below 0.5 for more than 6000 executions. Furthermore, FO improved the best known solution for some sets of instances, and even in the executions where the best known values were not reached, the found solutions were

extremely close of the best known values. This analysis reinforces the consistency of our proposal, and these results are also illustrated by Figure 4.2. A consistency comparison with [Cerulli et al. 2014] was not possible, because the authors only reported results of a single seed execution for each instance.

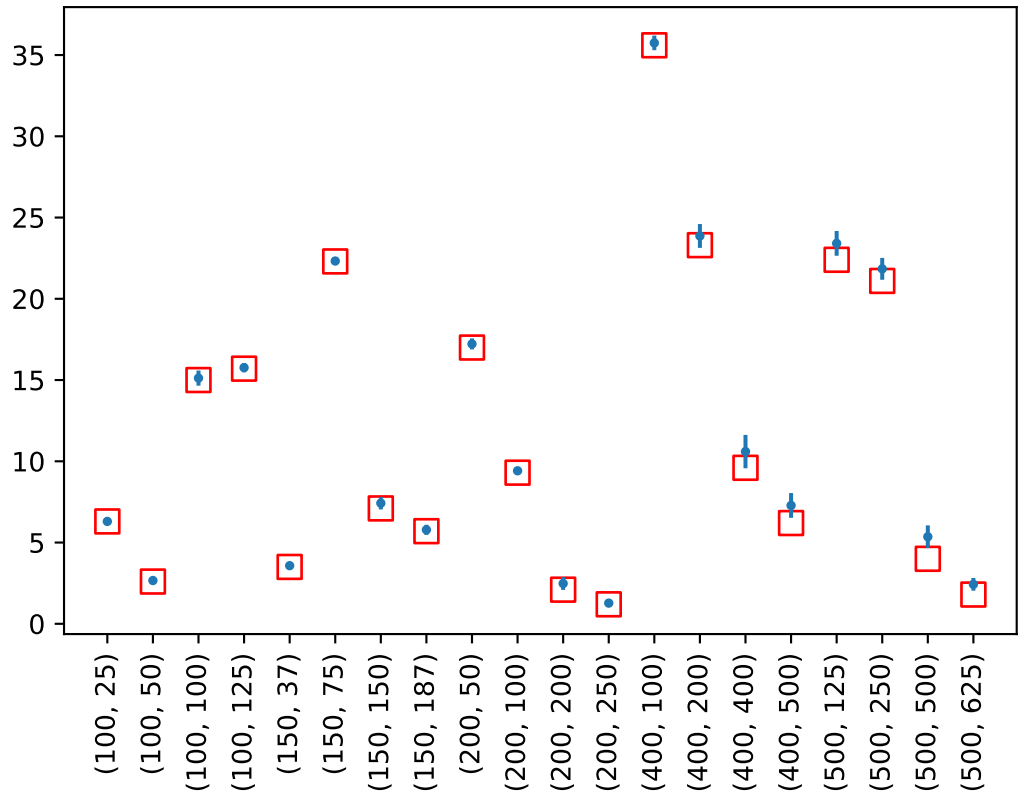

Figure 4.2 – In this image, the x-axis represents the set of instances defined for each combination of $|X|$ and $|L|$ from the data set. The red squares in the image represent the average of the best known value of each subset of instances, and the blue dots represent the overall average (consider the average value of each subset and compute the average value among them). The lines that extend the blue dots represent the standard deviation of the results.

5 CONCLUSION AND FUTURE WORKS

Problems related to labelled graphs have been present in some studies over the last few years [Silva 2018] what motivates the study of the the k labelled spanning forest. The KLSF has a lot of real world applicability from the network context to the routing context. In the literature, there are few studies in relation to it, which leaves open many possibilities to be explored.

From the theoretical part, two strong results were given: the NP-hardness of the problem even for simple instances and the inapproximability in relation to the number of labels. Additionally, a trivial approximation result was also presented for the triangles-edges graphs. As future plannings for the theoretical part, we intend to conclude some studies that include:

- A parameterized algorithm.
- A better approximation factor for triangles-edges graphs, and the possibility of approximating more general classes such as forests of cacti.

From the results obtained for the fix-and-optimize implementation it was possible to conclude that the combination of heuristic exploration and exact solutions of subproblems should be carefully analyzed. Some tests for applying such a combination of solver and heuristic during our study that were discarded for their low performance or solution quality include:

- A genetic algorithm for the heuristic part with an embedded formulation for the genetic operators.
- A branch and bound for solving the subproblem as part of the matheuristic here proposed.
- Probabilistic odds in the choice of labels that go to the subproblem or that are fixed.
- The idea of complementary solution proposed in [Consoli, Perez and Mladenović 2017].

Furthermore, the final version of the algorithm was tested with different seeds over 200 instances (more than 6,000 computational experiments). With these tests we empirically verified our proposal consistency, Although our fix-and-optimize was outperformed by previous approaches in some of the smaller instances, the algorithm presented a good scalability, since for the large tested instances our algorithm presented better solutions in shorter computational times.

For future works we consider to hybridize our fix-and-optimize matheuristic with other strategies, such the RTS, since it was the algorithm with better relation between execution time and solution quality proposed by [Cerulli et al. 2014]. Besides studying improvements for the heuristic part of the algorithm through hybridizations, exploring the exact part may bring benefits. In that direction, we will propose new mathematical formulations attempting to obtain more suitable models for solving the subproblems generated during the search process, also we consider developing an exact algorithm for computing optimal solutions of these subproblems. Nevertheless, we also intend to test our algorithm on real-world instances extracted from large computer networks applications, and on other datasets from the literature such as the ones in [Figueredo 2020] and [Consoli, Perez and Mladenović 2017].

REFERENCES

ALSPACH, B. The wonderful walecki construction. **Bull. Inst. Combin. Appl**, v. 52, p. 7–20, 2008.

CERULLI, R. et al. The k-labeled spanning forest problem. **Procedia-Social and Behavioral Sciences**, Elsevier, v. 108, p. 153–163, 2014.

CHANG, R.-S.; SHING-JIUAN, L. The minimum labeling spanning trees. **Information Processing Letters**, v. 63, n. 5, p. 277–282, 1997. ISSN 0020-0190. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0020019097001270>.

CONSOLI, S. et al. Variable neighbourhood search for the minimum labelling steiner tree problem. **Annals of Operations Research**, Springer, v. 172, n. 1, p. 71–96, 2009.

CONSOLI, S.; PEREZ, J. A. M.; MLADENOVIĆ, N. Comparison of metaheuristics for the k-labeled spanning forest problem. **International Transactions in Operational Research**, Wiley Online Library, v. 24, n. 3, p. 559–582, 2017.

CPLEX, I. I. V20. 1: User's manual for cplex. **International Business Machines Corporation**, v. 46, n. 53, p. 157, 2009.

DORNELES Árton P.; de Araújo, O. C.; BURIOL, L. S. A fix-and-optimize heuristic for the high school timetabling problem. **Computers Operations Research**, v. 52, p. 29–38, 2014. ISSN 0305-0548. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0305054814001816>.

FIGUEREDO, P. J. d. A. O problema da floresta geradora k-rotulada. 2020.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability**. [S.l.]: freeman San Francisco, 1979.

GITHUB. <https://github.com/tiagodrehmer/klsf_tests>. Accessed: 2022-03-02.

Gurobi Optimization, LLC. **Gurobi Optimizer Reference Manual**. 2021. Available from Internet: <https://www.gurobi.com>.

HELBER, S.; SAHLING, F. A fix-and-optimize approach for the multi-level capacitated lot sizing problem. **International Journal of Production Economics**, v. 123, n. 2, p. 247–256, 2010. ISSN 0925-5273. Available from Internet: <https://www.sciencedirect.com/science/article/pii/S0925527309003107>.

LANDAU, L. Em lifshitz statistical physics. **Course of Theoretical Physics**, v. 5, p. 396–400, 1980.

LI, L.; SONG, S.; WU, C. Solving a multi-level capacitated lot sizing problem with random demand via a fix-and-optimize heuristic. In: IEEE. **2015 IEEE Congress on Evolutionary Computation (CEC)**. [S.l.], 2015. p. 2721–2728.

LINDAHL, M.; SØRENSEN, M.; STIDSEN, T. R. A fix-and-optimize matheuristic for university timetabling. **Journal of Heuristics**, Springer, v. 24, n. 4, p. 645–665, 2018.

LóPEZ-IBáñEZ, M. et al. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016.

MAKHORIN, A. **GLPK (GNU Linear Programming Kit)**.
Available at `http://www.gnu.org/software/glpk/glpk.html`.

MILLER, J. S. et al. **Multimodal statewide transportation planning: a survey of state practices.** [S.l.], 2005.

PASSMARK CPU benchmark dataset. <https://www.cpubenchmark.net/>. Accessed: 2022-03-02.

POCHET, Y.; WOLSEY, L. A. **Production planning by mixed integer programming**. [S.l.]: Springer Science & Business Media, 2006.

SILVA, T. G. da. **The Minimum Labeling Spanning Tree and Related Problems**. Thesis (PhD) — Université d'Avignon; Universidade Federal do Estado do Rio de Janeiro (Brésil), 2018.

TANENBAUM, A. S.; WETHERALL, D. J. et al. **Computer networks**. [S.l.]: Prentice hall, 1996. I–XVII p.

VAISMAN, R. Finding minimum label spanning trees using cross-entropy method. **Networks**, Wiley Online Library, v. 79, n. 2, p. 220–235, 2022.

WIART, J. et al. **Electromagnetic fields (EMF) exposure**. [S.l.]: Springer, 2019. 1–3 p.

XIONG, Y.; GOLDEN, B.; WASIL, E. Improved heuristics for the minimum label spanning tree problem. **IEEE Transactions on Evolutionary Computation**, v. 10, n. 6, p. 700–703, 2006.

APPENDIX A — RESUMO ESTENDIDO

Neste trabalho foi estudado o problema da floresta geradora k-rotulada (kLSF). Este problema recebe como entrada um grafo simples não direcionado com arestas rotuladas e um número inteiro k. O objetivo do problema consiste em encontrar uma floresta geradora com no máximo k rótulos, tal que esta floresta possua o menor número de árvores (componentes). O kLSF pertence à classe de problemas de complexidade NP-completos e generaliza o problema mais conhecido da Árvore Geradora Minimamente Rotulada (MLST) [Cerulli et al. 2014].

O interesse em estudar o kLSF não se justifica apenas por sua dificuldade teórica, mas também pelos cenários práticos onde o problema encontra aplicações. Um exemplo ocorre ao estudar estratégias para reduzir o impacto na saúde da população em relação a exposições contínuas a campos eletromagnéticos. Atualmente, o surgimento de redes sem fio, como Wifi e redes de telefonia celular, vem expondo cada vez mais a população a campos eletromagnéticos, com possíveis efeitos negativos na saúde das pessoas [Wiart et al. 2019]. Uma abordagem para reduzir a exposição ao campo eletromagnético é limitar o número de frequências possíveis a um número máximo k. No entanto, para aumentar a qualidade dos serviços, as áreas de cobertura desconectadas devem ser o menor possível. Esse problema pode ser modelado como o kLSF definindo-se um nó para cada dispositivo da rede e uma aresta para cada par de conexões de nós, sendo os rótulos as frequências de tais conexões.

Outra aplicação de rede para o kLSF surge de empresas que gerenciam conexões de internet, onde muitas vezes são utilizados serviços de diferentes provedores para manter a conectividade dos clientes [Tanenbaum, Wetherall et al. 1996]. Neste cenário, para obter alguns benefícios em termos de negociação e custos, o número de diferentes fornecedores de aluguel é geralmente limitado por um valor k, selecionando os fornecedores que garantem menor número de áreas desconectadas. Analogamente ao exemplo anterior, os dispositivos e as conexões podem ser representados por nós e arestas de um grafo, enquanto para cada provedor pode ser definido um rótulo e associado às arestas correspondentes, sendo possível resolver este problema através de técnicas para o kLSF.

Ao longo do estudo feito neste trabalho, o problema kLSF foi abordado tanto de forma teórica quanto na parte aplicável. Das provas da parte teórica é reforçado a NP-dificuldade do problema não só para os casos mais gerais mas como também para instâncias mais simples. Isto é, no trabalho é provado que o problema é NP-dificil mesmo

para as instâncias em que o grafo de entrada são apenas triângulos e arestas. Esta prova é feita através da redução do problema NP-completo da 3-cobertura exata (X3C) para o KLSF tal que a não ser que P = NP não existe algoritmo de tempo polinomial que resolva estas instâncias. Outra prova teórica apresentada no trabalho é a inaproximabilidade do problema em relação ao conjunto de rótulos L, tal que novamente a não ser que P = NP não existe um algoritmo que garanta uma $|L|^{\mathcal{O}}(1)$-aproximação para o problema. Esta prova é feita também através da redução do problema X3C para os casos gerais do KLSF, tal que esta redução se existe uma $|L|^{\mathcal{O}(1)}$-aproximação para o problema então esta aproximação resultaria em uma solução ótima para o KLSF e consequentemente solucionaria o problema NP-completo X3C. Para finalizar os resultados teóricos, é apresentada uma 3-aproximação trivial para o problema, quando o grafo de entrada é um grafo em que suas componentes são apenas triângulos e arestas.

Seguindo o trabalho para a parte de aplicação uma matheurística é apresentada para solucionar as instâncias da literatura. Para esta matheuristica inicialmente um novo modelo matemático de programação linear inteira foi apresentado. Como este modelo não apresentou bons resultados foi logo descartado, e através de uma revisão bibliográfica foi encontrado o trabalho de [Figueredo 2020] que apresentou diferentes modelos matemáticos e técnicas de programação linear inteira para serem usadas. Deste trabalho foi usado o modelo do *"corte colorido"* e as técnicas de cut callbacks e lazzy constraints foram usadas na matheuristica.

A matheuristica desenvolvida, considera o uso de atalhos através de decisões heurísticas para reduzir o problema original e construir subproblemas que podem ser resolvidos de maneira eficiente por solvers de programação linear inteira. Esta matheuristica consiste em interativamente explorar o espaço de soluções do problema, tal que nestas interações é mantida uma solução corrente. A partir desta solução corrente e através de decisões heurística é feita a escolha de rótulos para serem fixados e rótulos para pertencerem ao subproblema. Tal que estas decisões reduzem a complexidade do grafo, permitindo então que o solver resolva o subproblema de maneira mais eficiente. A partir da solução do solver e os rótulos fixados é construído uma solução para o grafo original e feito um teste para aceitação da mesma para a atualização da solução corrente. Claro que, este processo iterativo sempre mantém a melhor solução explorada para ser o resultado final do algoritmo.

Esta matheuristica foi aplicada às instâncias propostas por [Cerulli et al. 2014] a fim de gerar resultados comparativos com os demais algoritmos propostos para o prob-

lema. Nestas instâncias foram feitos mais de 6000 testes do algoritmo. Os resultados apresentados demonstraram a escalabilidade que o algoritmo apresenta, obtendo uma solução melhor para as instâncias mais complexas da base de dados do que os demais algoritmos. Além disso foi feito uma análise da consistência do algoritmo sendo o primeiro a apresentar um estudo sobre esse tema para o KLSF na literatura. Esta análise consistiu em rodar com 30 sementes diferentes, e verificar quão variado ficaram os resultados. Os resultados destes testes mostraram que o algoritmo tem consistência, apresentando um desvio padrão menor que 0.5 em média.

Na tabela A.1 é possível encontrar os resultados obtidos para o algoritmo em relação aos propostos por [Cerulli et al. 2014]. Dividia em 6 colunas principais, a tabela apresenta na primeira coluna as caracteristicas do conjunto de instâncias tal que $|V|$ representa a quantidade de nós, $|E|$ a quantidade de arestas, $|L|$ a quantidade de rótulos e k o valor inteiro que limita o número de rótulos na solução final. Cada linha destas combinações representa um conjunto de 10 instâncias. Seguindo as colunas são apresentados os resultados obtidos pelos algoritmo desenvolvido por [Cerulli et al. 2014] (SA, RTS, GA e PILOT), no qual a baixo deles se tem duas coluna representando a média do resultado obtido (sol.) para as 10 instâncias de cada conjunto e o tempo médio da execução (tempo (s)). Na última coluna é apresentado os resultados do algoritmo fix-and-optimize, onde tal coluna está dividida em 3 sub colunas que definem a solução média das 30 execução do algoritmo (sol.), o tempo médio destas execuções (tempo (s)) e a solução que apresentou a melhor média de resultados (sol. (1)).

Table A.1 – Resultados dos 5 algoritmos testado para o KLSF

Dataset				SA		RTS		GA		PILOT		FO		
$\|V\|$	$\|E\|$	$\|L\|$	k	sol.	time (s)	sol.	time (s)	sol.	time (s)	sol.	time (s)	sol.	time (s)	sol. (1)
100	990	25	3	**6.30**	0.08	**6.30**	0.08	**6.30**	0.25	**6.30**	0.00	**6.30**	0.18	**6.30**
100	990	50	6	2.70	0.16	2.90	0.11	**2.60**	0.42	2.70	0.08	2.66	0.65	**2.60**
100	990	100	6	**15.00**	0.38	**15.00**	0.39	**15.00**	0.67	15.60	0.31	15.12	8.12	**15.00**
100	990	125	7	**15.70**	0.53	**15.70**	0.54	**15.70**	0.87	**15.70**	0.59	15.76	7.07	**15.70**
150	2235	37	4	**3.50**	0.19	**3.50**	0.13	**3.50**	0.52	**3.50**	0.04	3.58	0.12	3.60
150	2235	75	4	**22.30**	0.40	**22.30**	0.40	**22.30**	0.80	**22.30**	0.13	22.32	1.07	**22.30**
150	2235	150	9	7.70	0.97	**7.30**	0.84	8.10	2.93	8.30	1.31	7.42	3.18	7.50
150	2235	187	11	6.10	1.31	6.20	1.10	6.20	3.28	6.10	1.86	5.78	5.73	**5.70**
200	3980	50	3	**17.00**	0.37	**17.00**	0.37	17.20	0.93	**17.00**	0.05	17.22	0.52	17.20
200	3980	100	6	**9.30**	0.84	**9.30**	0.84	9.60	2.56	9.60	0.87	9.42	5.01	9.40
200	3980	200	12	2.60	1.56	3.10	1.07	3.10	5.03	3.00	2.72	2.48	3.09	**2.40**
200	3980	250	15	1.80	1.66	1.60	1.20	1.50	4.87	**1.20**	5.35	1.27	1.46	1.30
400	15960	100	3	**35.60**	5.15	**35.60**	5.17	**35.60**	4.47	**35.60**	1.38	35.75	1.85	**35.60**
400	15960	200	6	24.40	11.25	**23.70**	11.35	25.10	12.32	24.00	16.55	23.87	34.73	**23.70**
400	15960	400	12	12.10	25.42	11.40	21.66	12.50	37.47	11.10	81.83	**10.60**	33.55	10.60
400	15960	500	15	8.70	34.22	8.20	21.24	8.90	48.92	7.70	163.04	7.28	13.84	**7.00**
500	24950	125	4	**22.70**	13.11	**22.70**	13.18	22.90	10.66	**22.70**	8.34	23.41	1.12	23.00
500	24950	250	7	22.40	27.54	**21.50**	27.75	22.80	24.75	21.90	33.10	21.84	28.15	21.70
500	24950	500	15	6.30	47.23	5.30	42.05	6.60	75.62	5.00	251.83	5.36	19.10	**4.90**
500	24950	625	19	3.40	52.67	3.00	38.56	3.40	111.03	**2.10**	214.08	2.43	22.06	**2.10**
average				12.28	11.25	12.08	9.40	12.45	17.42	12.07	39.17	11.99	9.53	11.88