RAIRO-Oper. Res. 56 (2022) 3393–3404
https://doi.org/10.1051/ro/2022146

RAIRO Operations Research
www.rairo-ro.org

THE EDGE-LABELED SURVIVABLE NETWORK DESIGN PROBLEM: FORMULATIONS AND BRANCH-AND-CUT

Mariem Ben Salem[1], Raouia Taktak[2,*] and Fatmah Almathkour[3]

Abstract. In this paper, we study a variant of the survivable network design problem, that is the survivable network design problem with labels (colors) on the edges. In particular, we address the Generalized Labeled Two Edge Connected Subgraph Problem (GLTECSP) that has many applications in telecommunication and transportation. Given a connected undirected graph G such that with each edge is associated a set of labels (colors), the GLTECSP consists in finding a two-edge connected spanning subgraph of G with a minimum number of distinct labels. We propose two Integer Programming (IP) formulations for the problem, a natural formulation using cuts on the edges, and a compact formulation using color-cuts. We devise Branch-and-Cut algorithms to solve both formulations and compare them on sets of randomly generated instances. Computational results show that the compact formulation outperforms the natural one regarding the linear relaxation and the computational time. Moreover, the compact formulation is able to solve to optimality several instances left unsolved within the time limit by the natural formulation.

Mathematics Subject Classification. 90C10, 90C27, 90C57.

Received January 23, 2022. Accepted August 22, 2022.

1. Introduction

During recent years, designing survivable telecommunication networks has been of great interest. A telecommunication network is said to be *survivable* if it remains always connected even in the case of failure of one or many links (or nodes). A telecommunication network can be modeled as an undirected graph $G = (V, E)$, where the vertices of V represent telecommunication nodes, and the edges of E the possible links between them. In order to have a minimum degree of survivability of this graph (network), one has to ensure that the graph is 2-*edge connected*. That is to say between any two vertices of V there exist at least two-edge-disjoint paths. Recall that two paths are said to be *edge-disjoint* if they do not have any edge in common. Obviously, a 2-edge connected graph remains connected in the case of one-link failure. Given an undirected weighted graph $G = (V, E, \omega)$ where $\omega(.)$ is the weight vector on the edges, the Two Edge Connected Subgraph Problem (TECSP) consists in finding a minimum-cost 2-edge connected subgraph of G.

Keywords. Labeled graphs, survivable network design, two-edge connected subgraph, integer programming, Branch-and-Cut.

[1] FSEGS/MIRACL, Université de Sfax, Sfax, Tunisia.
[2] ISIMS & Sma@rts/CRNS, Université de Sfax, Sfax, Tunisia.
[3] Department of Statistics and Operations Research, Kuwait University, Kuwait City, Kuwait.
[*]Corresponding author: raouia.taktak@gmail.com

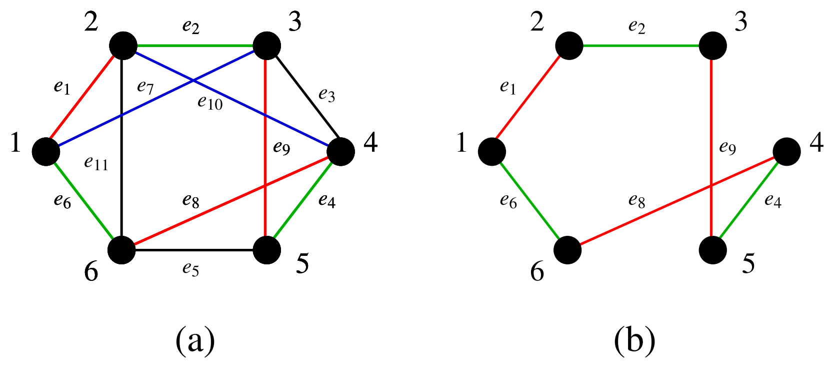

FIGURE 1. Illustrative example for the LTECSP.

In this paper, we are interested in a variant of the TECSP. Assume that instead of weights, with each edge of E is associated a set of *colors*, also called *labels*. Let L be the set of all labels of the graph, such that with each edge $e \in E$ is associated a set of labels $L_e \subseteq L$. The Generalized Labeled Two Edge Connected Subgraph Problem (GLTECSP) consists in finding a minimum-label spanning 2-edge connected subgraph of G, that is a labeled 2-edge connected subgraph with a minimum number of labels. In particular, if each edge $e \in E$ is associated with exactly one color (label), *i.e.*, $|L_e| = 1$, the problem is then called the Labeled Two Edge Connected Subgraph Problem (LTECSP). It is obvious that the GLTECSP is NP-hard. In fact, if with each edge is associated a different single color, that is $|L| = |E|$, the GLTECSP is equivalent to the TECSP known to be NP-hard [27].

In Figure 1a, we give an example of a graph with six vertices denoted from 1 to 6, and eleven edges denoted e_i, $i = 1, 2, \ldots, 11$. There are four labels (colors) in the graph $L = \{$red, blue, green, black$\}$, and with each edge $e_i, i \in \{1, 2, \ldots, 11\}$ is associated exactly one label from L. The objective of the LTECSP is to find a two-edge connected subgraph with a minimum number of labels. Figure 1b illustrates an optimal solution for this example. In fact, in this solution the obtained graph is clearly 2-edge connected, that is between each pair of vertices there are two edge-disjoint paths. Moreover, we are using only two labels, which is obviously the minimum number of labels that we may use in this configuration.

In Figure 2a is illustrated an example for the GLTECSP. It consists of a graph with six nodes, eleven edges and four labels (as previously described). However, in this example, some edges may have more than one label. In Figure 2b an optimal solution of this example is given. The obtained graph is 2-edge connected using two labels, which is the minimum number of labels that can be used. Here, for edge e_2 we may retain both labels (blue and green) since this will not change the number of labels in the final solution.

Both the LTECSP and the GLTECSP have several applications in telecommunication networks, electric networks, multimodal transportation networks, where one aims to ensure connectivity by means of *homogeneous connections* that are generally managed by different and competing companies. This is the case, for example, of telecommunication networks where the different colors can be seen as different operators or different types of media. Recent applications in social networks are depicted in [31], vertices of the graph refers to people, edges represent the relations between them, and different kinds of relationship are labeled with different colors..

The paper is organized as follows. In the next section we overview some related works. We mainly distinguish works that are related to network survivability and those that are related to labeled problems in graphs. In Section 3, we propose Integer Programming (IP) formulations for the GLTECSP, a *natural formulation* based on cuts on the edges, and a *compact formulation* based on cuts on the colors. In Section 4, we devise Branch-and-Cut algorithms for both formulations and present experimental results held on randomly generated instances.

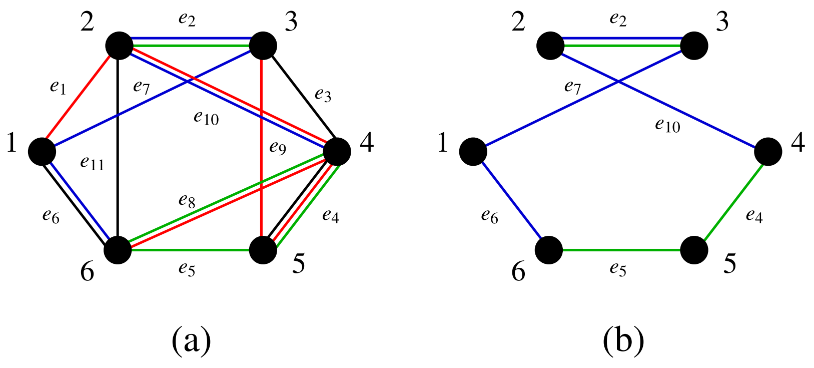

FIGURE 2. Illustrative example for the GLTECSP.

2. Related works

2.1. Survivable network design problem

The TECSP is a classical well known problem in network design. Inspired from the telecommunication context, survivable network design problems have been intensively studied for several years (see [27] for a survey on the problem). Several resolution methods including heuristics, approximation algorithms as well as exact algorithms have been developed to solve various variants of the problem. Consider an undirected graph $G = (V, E)$ representing a telecommunication network and a weight function which associates the weight $\omega(e)$ with every edge $e \in E$. With each node $v \in V$ is associated an integer, denoted by $r(v)$ and called *connectivity type* of v, which can be seen as the minimum number of edges connecting v to the rest of the network. A subgraph $H = (U, F)$ of G where $U \subseteq V$ and $F \subseteq E$ is said to satisfy the edge-connectivity (resp. node-connectivity) requirement if for every pair of nodes $(s, t) \in V \times V$, there exist at least $r(s, t) = \min\{r(s), r(t)\}$ edge-disjoint (resp. node-disjoint) paths between s and t. Recall that two st-paths are *edge-disjoint* if they have no edge in common, and they are *node-disjoint* if they have no internal node in common. The edge-connectivity (resp. node-connectivity) condition ensures that the network remains functional (connected) when some equipments fail. In fact, the traffic can still be routed between two nodes s and t when at most $r(s, t) - 1$ links, in case of edge-connectivity, and at most $r(s, t) - 1$ nodes, in case of node-connectivity, fail. When $r(u) = k$, for every $u \in V$, the subgraph H is k-edge-connected (resp. k-node-connected). In [23], Grötschel *et al.* address the general survivable network design problem that consists in finding a minimum-weight subgraph of G while satisfying the connectivity requirements. In particular, the k-edge-connected subgraph problem, which consists in finding a minimum-weight spanning subgraph of G that is k-edge-connected, has been extensively studied, especially in the case of low connectivity requirement ($k = 2$, *i.e.,* the TECSP). In fact, the 2-connectivity has shown to be a very suitable topology for telecommunication networks. In [28], Mahjoub introduces a general class of valid inequalities for the polytope associated with the TECSP, called F-partition inequalities. Boyd and Hao [4] describe a class of valid inequalities for the TECSP polytope that are "comb"-like inequalities. In [2], Barahona and Mahjoub characterize the TECSP's polytope for the class of Halin graphs. An interesting variant of the problem has also been studied, this is the case when $r \in \{0, 2\}$. The nodes with $r(v) = 2$ are called *terminals* and those with $r(v) = 0$ are called *Steiner nodes.* The terminals need high degree of survivability. The Steiner ones are optional nodes, they may be used if they contribute reducing the cost of the network. This problem is known as the Steiner 2-edge connected subgraph problem (STECSP). In [1], Baïou and Mahjoub study the STECSP's polytope. In parallel, Coullard *et al.* study the node version [11, 12]. In [11], a linear time algorithm for the problem on some special classes of graphs is proposed. In [12], the authors describe the dominant of the relaxed polytope in a special class of graphs.

2.2. Classical labeled problems in graphs

Several classical well-known problems in graphs have been studied within the labeled version. A particular interest has been provided to the Minimum Labeled Spanning Tree Problem (MLSTP). The problem is proved to be NP-hard with different reduction procedures by Chang and Leu [9], and in parallel by Broersma and Li [5]. In [7], Cerruli *et al.* apply several metaheuristic approaches to solve the MLSTP. These metaheuristics are able to improve over existing heuristics presented in the literature. They also provide optimal or close to optimal solutions compared to the exact ones. In [6], Captivo *et al.* introduce some mixed Integer Programming formulations for the MLSTP and compare the different linear relaxations. In [10], Consoli *et al.* propose an intelligent-optimization algorithm to solve the MLSTP. This algorithm is obtained by combining the basic Variable Neighbourhood Search heuristic with some features from machine learning, statistics and experimental algorithmics. In [14], the authors study the Generalized MLSTP, that is when multiple labels can be assigned to an edge. The authors propose a new compact integer programming formulation for the GMLSTP and study the associated polytope. Several families of valid inequalities are identified and their facial aspect is investigated. In [16], the authors introduce a new MIP-based metaheuristic for the MLSTP called the multi-start local branching (MSLB). Computational experiments show that the MSLB outperforms the state-of-the-art metaheuristics with respect to optimality and processing times. In [30], Vaisman proposes a new method to solve the MLSTP called the cross-entropy method, which relies on rigorous developments in the fields of information theory and stochastic simulation. Experimentations indicate that the proposed algorithm can obtain optimal or near-optimal solutions while using a reasonable computational time. Several other connectivity-related problems in edge-labeled graph have been also investigated. In [8], Cerulli *et al.* studied an extension of the MLSTP where the vertices are divided into terminals and Steiner nodes, that is the Steiner Tree Problem with Minimum number of Labels. The problem is solved using several metaheuristics, and numerical results are presented.

Along with the Spanning tree related problems, scientists have also addressed other labeled-graph problems. In [32], the authors study the labeled TSP. They prove that the problem is NP-hard, and present a heuristic and a genetic algorithm to solve the problem. In [26], the authors propose a mathematical model, valid inequalities and polyhedral results for the minimum labeled Hamiltonian cycle problem. The authors also introduce two variants of this problem and devise Branch-and-Cut algorithms for the different variants. In [3], the Minimum Coloring Cut Problem (MCCP) is studied. The MCCP is defined as follows: given a connected graph G with colored edges, find an edge cut E' of G (a minimal set of edges whose removal disconnects the graph) such that the number of colors used by the edges in E' is minimum. The authors present two approaches based on Variable Neighborhood Search to solve the problem. The presented algorithms are able to find all the optimal solutions described in the literature. In [18], Faria *et al.* study the maximum labeled cut problem. Given an edge-labeled graph and a parameter k, the question is whether or not G has a nontrivial edge cut using at least k labels (colors). The problem of minimum colorful cut and related problems have also been addressed in [15, 17, 22].

In this work, we address the TECSP in labeled graphs. To the best of our knowledge, the problem has only been initiated in [13, 29], and no algorithmic implementations have been considered for it.

3. IP Formulations

In this section we propose two Integer Programming (IP) formulations for the GLTECSP. The first one is based on *edge-cuts* and is called *edge-cut formulation* (ECut) or *natural formulation*. The second one uses *color-cuts* and is called *color-cut formulation* (CCut) or *compact formulation*. In order to introduce these formulations, we first give some definitions and notations.

3.1. Notations

Let $G = (V, E, L)$ be an edge-labeled undirected graph, where L is a set of labels, such that with each edge $e \in E$ is associated a subset of labels $L_e \subseteq L$. For $W \subset V$ and $W \neq \emptyset$, let $\delta(W)$ denote the set of edges of

G having one node in W and the other in $\overline{W} = V \setminus W$. $\delta(W)$ is called a *cut* or an *edge-cut*. When $W = \{w\}$, $w \in V$, we will write $\delta(w)$.

3.2. Natural formulation

We define three families of binary decision variables. For each label $l \in L$, let $y_l = 1$ if label l is selected in the final solution, and 0 if not. For each edge $e \in E$ and each label $l \in L_e$, let $z_{el} = 1$ if edge e having label l is retained in the final solution, and 0 if not. Finally, for each edge $e \in E$, we let $x_e = 1$ if edge e is selected in the final solution, and 0 if not. y will be called *label variables*, and x and z *design variables*.

The GLTECSP is equivalent to the following integer linear program.

$$\text{(ECut)} \min \sum_{l \in L} y_l \tag{3.1}$$

$$\sum_{e \in \delta(W)} x_e \geq 2 \qquad \text{for all } W \subset V, \text{ and } W \neq \emptyset, \tag{3.2}$$

$$x_e = \sum_{l \in L} z_{el} \qquad \text{for all } e \in E, \tag{3.3}$$

$$z_{el} \leq y_l \qquad \text{for all } e \in E \text{ and } l \in L_e, \tag{3.4}$$

$$0 \leq y_l \leq 1 \qquad \text{for all } l \in L, \tag{3.5}$$

$$0 \leq x_e \leq 1 \qquad \text{for all } e \in E, \tag{3.6}$$

$$0 \leq z_{el} \leq 1 \qquad \text{for all } e \in E, \text{ for all } l \in L_e, \tag{3.7}$$

$$y_l \in \{0,1\} \qquad \text{for all } l \in L, \tag{3.8}$$

$$x_e \in \{0,1\} \qquad \text{for all } e \in E, \tag{3.9}$$

$$z_{el} \in \{0,1\} \qquad \text{for all } e \in E, \text{ for all } l \in L_e. \tag{3.10}$$

By the objective function (3.1), the number of labels is minimized. Inequalities (3.2) are called *edge-cut inequalities*. These ensure that every cut in the graph has a cardinality at least 2, which implies that the solution is 2-edge connected. Note that inequalities (3.2) have an exponential number which means that solving (ECut) to optimality requires their separation which will be detailed in Section 4. Constraints (3.3) are setting the relationship between the design variables x_e and z_{el}. Inequalities (3.4) are giving the relationship between the design variables z_{el} and the label variables y_l. Constraints (3.5)–(3.7) are the trivial constraints (3.8)–(3.10) are the integrality constraints of the decision variables.

3.3. Compact formulation

In [13], the author propose a formulation for the k-edge connected subgraph problem and the 2-node connected subgraph problem based on the so-called *color-cut* or *colorful cut*. In order to adapt this formulation to the GLTECSP, we introduce some definitions and notations that will be useful in the sequel [13].

For a cut $\delta(W)$ let $L(\delta(W)) = \{L_e : e \in \delta(W)\}$ be the set of labels associated with the edges of $\delta(W)$. Denote by $K(W) = L(\delta(W))$ the cut on labels derived from the edge-cut $\delta(W)$. $K(W) \subseteq L$ is called *color-cut* or *colorful cut*. In the sequel, $E(\{l\})$ will denote the set of edges that have label $l \in L$. This will simply be denoted by $E(l)$ for $l \in L$.

Figure 3a illustrates the concept of color-cuts. In this example, we have a graph with six vertices denoted from 1 to 6, and ten edges denoted e_i, $i = 1, 2, \ldots, 10$. There are four labels (colors) $L = \{\text{red, blue, green, black}\}$ in the graph, and with each edge $e_i, i \in \{1, 2, \ldots, 11\}$ it is associated exactly one label. In Figure 3b, we illustrate a cut separating $W = \{1,2\}$ from $V \setminus W = \{3,4,5,6\}$. The edge-cut induced by W is $\delta(W) = \{e_2, e_6, e_7, e_{10}\}$ and the corresponding color-cut is $K(W) = \{\text{green, blue}\}$. Note that at least one of the elements of $K(W) =$

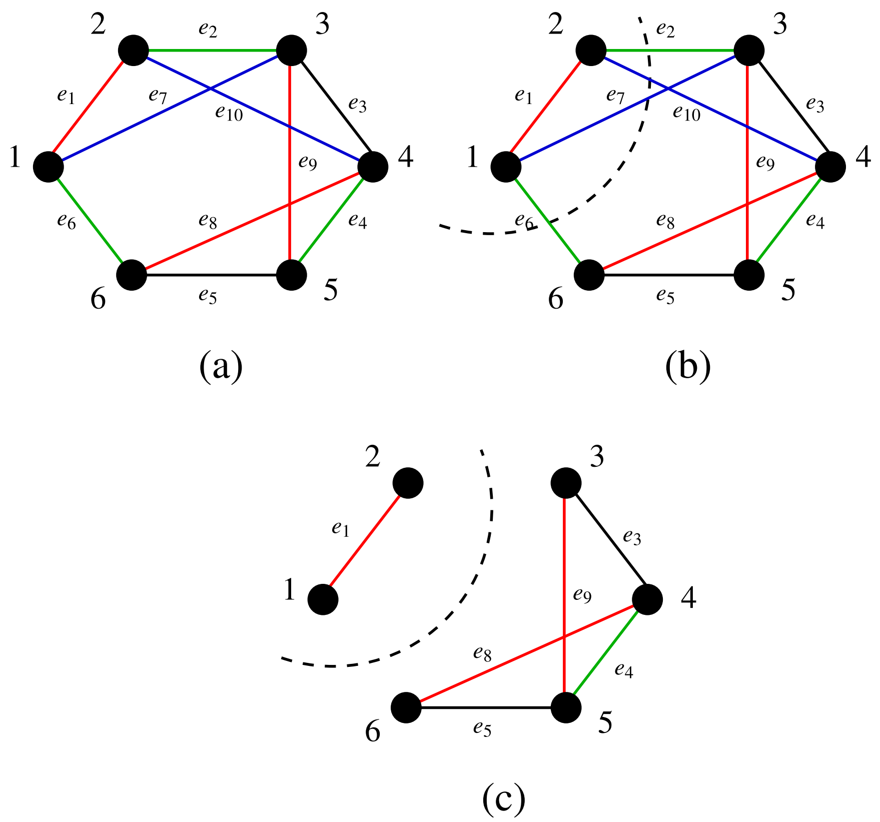

FIGURE 3. Illustrative example for color-cuts.

{green, blue} must be considered in the optimal solution, otherwise the resulting graph will be disconnected as shown in Figure 3c.

The GLTECSP is equivalent to the following IP.

$$\text{(CCut)}\quad \min \sum_{l \in L} y_l \tag{3.11}$$

$$\sum_{l \in K(W)} \min(2, |E(l) \cap \delta(W)|)\; y_l \geq 2 \qquad \text{for all } W \subset V, \text{ and } W \neq \emptyset, \tag{3.12}$$

$$0 \leq y_l \leq 1 \qquad \text{for all } l \in L, \tag{3.13}$$

$$y_l \in \{0, 1\} \qquad \text{for all } l \in L. \tag{3.14}$$

The objective function (3.11) minimizes the number of labels. Inequalities (3.12) are called *color-cut inequalities*. They ensure that the solution graph is 2-edge connected by requiring at least 2 edges for every cut in the graph. Basically, the nonlinear term $\min(2, |E(l) \cap \delta(W)|)$ will take into consideration cuts where at least two edges have the same label $l \in L$. In this case $|E(l) \cap \delta(W)| \geq 2$. Constraints (3.13) are the trivial constraints, and (3.14) are the integrality constraints of the decision variables.

Obviously, (CCut) has significantly less variables than (ECut), and that is why we will call it *compact formulation* even though inequalities (3.12) are in exponential number and need to be separated. Note also that (CCut) enables having more than one color per edge in the final solution, which is not the case for the (ECut) formulation due to constraints (3.3). This means that the solution space of (ECut) is included in the (CCut) one. The optimal solution has though always the same number of labels. Moreover, (CCut) would clearly have a better linear relaxation than (ECut), which will be proved by experimentations in the following section.

4. Experimental results

In this section, we devise Branch-and-Cut algorithms for the edge-cut formulation (ECut) and the color-cut formulation (CCut) previously introduced for the GLTECSP. The Branch-and-Cut algorithms are implemented in C++ using CPLEX 12.10[1] as a linear solver. Computations are performed on an Intel-Core-i7 2.80 GHz with 16 GB of RAM, running under Linux (Ubuntu 21.0). Tests are computed on complete-graph instances that have been randomly generated, and time limit is set to 1 h.

4.1. Branch-and-cut algorithms

The IP formulations proposed for the GLTECSP are given with a huge number of edge-cut inequalities for the (ECut) formulation (resp. color-cut inequalities for the (CCut) formulation). We first consider a restricted version of the linear program corresponding to the (ECut) formuation (resp. the (CCut) formulation). A restricted number of edge-cut inequalities (resp. color-cut inequalities) is generated in the first LP. For the (ECut), we generate only edge-cut inequalities on the vertices. Similarly, only color-cut inequalities on the vertices are generated for the (CCut) formulation. Therefore, the initial linear program for the (ECut) (resp. (CCut)) formulation denoted by LP(ECut) (resp. LP(CCut)) solved in the first step is given by inequalities (3.2) (resp. inequalities (3.12)) written for $W = \{v\}$ for each $v \in V$.

Algorithm 1: Branch-And-Cut Algorithm for the (ECut) formulation.

```
   Data: An edge-labeled undirected graph $G = (V, E, L)$
   Result: Optimal solution for the GLTECSP
 1 LP ← LP(ECut);
 2 Solve the linear program LP and denote by $(\overline{x}, \overline{y}, \overline{z})$ the optimal solution of LP;
 3 if edge-cut inequality (3.2) is violated by $(\overline{x}, \overline{y}, \overline{z})$ then
 4     go to 8;
 5 else
 6     Add all possible violated inequalities by $(\overline{x}, \overline{y}, \overline{z})$ to LP;
 7     go to 2;
 8 if $(\overline{x}, \overline{y}, \overline{z})$ is integer then
 9     $(\overline{x}, \overline{y}, \overline{z})$ is an optimal solution for the GLTECSP. Stop;
10 else
11     Create two sub-problems by branching on a fractional variable.
12 forall the open sub-problem do
13     go to 2;
14 return the best optimal solution of all the sub-problems.
```

Denote by $(\overline{x}, \overline{y}, \overline{z}) \in \mathbb{R}^E \times \mathbb{R}^L \times \mathbb{R}^{L\times E}$ the solution of the linear relaxation LP(ECut). The obtained solution $(\overline{x}, \overline{y}, \overline{z})$ is optimal for the restricted LP if and only if it satisfies all the edge-cut inequalities (3.2). In general, this is not the case. Therefore, violated edge-cut inequalities are added to the restricted LP by solving a subproblem called *separation problem* (see Sect. 4.2 for details). The process is repeated until no more violated inequality is found. The final solution is hence optimal for the linear relaxation LP(ECut). If the solution is integer, then

[1]https://www.ibm.com/products/ilog-cplex-optimization-studio.

it is optimal for the GLTECSP. If not, then we create new subproblems by branching on a fractional variable. The separation routine is then considered at each node of the tree and the process continues. These steps are summarized in Algorithm 1.

Similarly, denote by $\overline{y} \in \mathbb{R}^L$ the optimal solution of the linear relaxation LP(CCut). For the (CCut) formulation, as the color-cut inequalities (3.12) separation is an open question, we only choose to separate on integer solutions (see Sect. 4.3). Consequently, if $\overline{y}$ is fractional then we branch on one of the fractional variables, creating hence two subproblems. Otherwise, we solve the separation problem corresponding to the color-cut inequalities (3.12) as it will be explained in Section 4.3. Algorithm 2 gives the Branch-and-Cut steps for formulation (CCut).

Algorithm 2: Branch-And-Cut Algorithm for the (CCut) formulation.

```
   Data: An edge-labeled undirected graph $G = (V, E, L)$
   Result: Optimal solution for the GLTECSP
1  LP ← LP(CCut);
2  Solve the linear program LP and denote by $\overline{y}$ the optimal solution of LP;
3  if $\overline{y}$ is fractional then
4      Create two sub-problems by branching on a fractional variable;
5  else if color-cut inequality is violated by $\overline{y}$ then
6      Add all possible violated inequalities by $\overline{y}$ to LP;
7      go to 2;
8  else
9      $\overline{y}$ is an optimal solution for the GLTECSP. Stop;
10 forall the open sub-problem do
11     go to 2;
12 return the best optimal solution of all the sub-problems.
```

4.2. Separation of the edge-cut inequalities

The separation of inequalities (3.2) can be done in polynomial time. The problem is to find one or more cut inequalities (3.2) that are violated by the current solution $(\overline{x}, \overline{y}, \overline{z})$. This can be done exactly using the algorithm of Gomory–Hu [21]. This algorithm gives back the so-called Gomory–Hu tree having the property that between two nodes $s, t \in V$, the minimum cut separating s and t in the graph G is nothing but the minimum cut separating s and t in the cut tree. To compute Gomory–Hu tree, we use the efficient implementation of Gusfield [24, 25]. Recall that, for a graph $G = (V, E)$, this implementation consists of $|V| - 1$ maximum flow problems in G. By the maximum flow – minimum cut theorem [19], the minimum cut problem can be solved in polynomial time. Thus, in our Branch-and-Cut algorithm, for all the problems of minimum cut (and hence maximum flow), we use the algorithm of Goldberg and Tarjan [20], which is one of the most powerful implementations of this problem. Note that the complexity of the separation is $O(mn^2 \log(\frac{n}{m}))$ where $n = |V|$ and $m = |E|$.

4.3. Separation of the color-cut inequalities

For an optimal solution $\overline{y} \in \mathbb{R}^L$, the separation of the color-cut inequalities (3.12) consists in finding a cut violated by $\overline{y}$ or showing that such a cut does not exist. The complexity of this problem is an open question. As a consequence, we choose to separate these inequalities using a heuristic approach. We first consider three scenarios: (i) separating fractional and integer solutions only in the root node; (ii) separating fractional and integer solutions for all the Branch-and-Cut tree's nodes and (iii) separating only integer solutions for all the Branch-and-Cut tree's nodes. Experimentations show that the last scenario is the best regarding the number of generated cuts, complexity and computational time. Consequently, in our Branch-and-Cut algorithm, we choose

TABLE 1. Branch-and-Cut results for the GLTECSP (1).

V	L	Optimal	Sub-EC	Gap-EC	Cuts-EC	Opt-EC	CPU-EC	Sub-CC	Gap-CC	Cuts-CC	Opt-CC	CPU-CC
20	20	2.6	23.2	38.22	0	5/5	0.126	14	29.78	0	5/5	0.009
20	30	3.2	36	32.26	0	5/5	0.207	21.4	28.30	0	5/5	0.009
20	40	4	30.2	28.71	0.6	5/5	0.196	16.6	27.85	0	5/5	0.007
20	50	4	45	22.04	1.2	5/5	0.280	19.2	20.64	0	5/5	0.006
20	60	4.8	38	19.87	4.6	5/5	0.261	14.2	18.74	0	5/5	0.005
20	70	4.8	26.4	16.89	3.8	5/5	0.173	15.2	15.73	0	5/5	0.005
20	80	5.2	35.8	15.13	3	5/5	0.237	17.6	14.23	21.2	5/5	0.008
20	90	5.8	27.4	15.83	6	5/5	0.194	21	15.23	24	5/5	0.009
20	100	6	30.6	16.71	7.4	5/5	0.224	31.4	16.63	46.6	5/5	0.009
30	50	3.8	83.4	36.17	0.2	5/5	1.840	33	32.69	0	5/5	0.020
30	100	6	127.2	27.64	1	5/5	2.720	54.6	26.14	0	5/5	0.018
30	200	7.8	233.8	19.01	4.8	5/5	7.343	58.8	18.80	107.2	5/5	0.038
30	300	8.8	226	12.35	7.6	5/5	7.266	124.2	12.05	53.2	5/5	0.067
30	400	10	215.2	11.00	13	5/5	8.137	105.4	10.92	83.2	5/5	0.079
40	100	5	202	32.16	0	5/5	17.538	158	30.62	0	5/5	0.080
40	200	7	768.6	18.54	4.6	5/5	68.855	451.8	17.64	0	5/5	0.165
40	300	8.8	1325.4	18.03	12.4	5/5	142.348	321.6	17.90	0	5/5	0.138
40	400	9.8	225.2	16.62	3.4	5/5	17.184	171.4	16.14	0	5/5	0.134
40	500	10.8	1372.8	16.29	20.8	5/5	139.203	186	16.25	143.8	5/5	0.120
40	600	11.6	1368.8	9.68	24.2	5/5	141.046	159.2	9.59	168.2	5/5	0.189
40	700	12.6	849.8	12.43	22.4	5/5	95.934	140.6	12.43	296.6	5/5	0.302
50	100	4.6	677.8	36.30	1	5/5	115.609	485	34.13	0	5/5	0.279
50	200	7	768.2	26.31	1.4	5/5	190.203	1843.8	25.34	0	5/5	0.941
50	300	8.6	2620.8	22.56	4.2	5/5	530.595	1623.4	22.00	0	5/5	0.686
50	400	9.8	2279	20.04	12	5/5	521.228	567.4	19.77	245	5/5	0.362
50	500	10.8	3804.8	18.95	15.6	5/5	705.247	1364.2	18.79	132.6	5/5	0.731
50	600	11.2	2996.8	16.77	20.4	5/5	743.684	675	16.81	239.4	5/5	0.541
50	700	12.6	5349.4	14.12	36.8	5/5	1064.751	3408.2	13.94	232.2	5/5	1.994
50	800	13	6441.4	14.38	41.2	5/5	1186.349	1767.4	13.19	0	5/5	1.256
50	900	13.8	6121.8	12.93	46.8	5/5	1201.532	2235.6	12.79	103.8	5/5	1.810
50	1000	14	8470.8	13.44	64.2	5/5	1836.224	1679.4	12.22	36.8	5/5	1.415

to apply separation only for integer solutions (see Algorithm 2). The separation procedure can be described as follows. As $\overline{y}$, the obtained optimal solution for the LP(CCut), is integer, we will apply a maximum flow algorithm to look for a violated cut. To this end, we use LEMON[2] to compute these minimum cuts.

4.4. Computational results

The experimental results are reported in Tables 1 and 2. Entries of the tables are averages values as follows:

$\|V\|$	:	Number of vertices in G ranging from 20 to 100;
$\|L\|$	:	Number of labels ranging from 20 to 1000;
Optimal	:	Best found solution;
Sub-EC/Sub-CC	:	Number of nodes in the Branch-and-Cut tree (number of treated subproblems) for (ECut)/(CCut);
Gap-EC/GapCC (%)	:	Relative error between the best upper bound and the lower bound obtained at the root for (ECut)/(CCut);
EC-cuts/CC-cuts	:	Number of separated edge-cuts/color-cuts;
CPU-EC/CPU-CC (s)	:	Total time of execution in seconds for (ECut)/(CCut).

[2]https://lemon.cs.elte.hu/trac/lemon.

TABLE 2. Branch-and-Cut results for the GLTECSP (2).

V	L	Optimal	Sub-EC	Gap-EC	Cuts-EC	Opt-EC	CPU-EC	Sub-CC	Gap-CC	Cuts-CC	Opt-CC	CPU-CC
60	100	4	711.6	40.08	0.2	5/5	408.691	352	36.67	0	5/5	0.526
60	200	7	3307	33.29	1.6	5/5	1155.846	1997.2	32.23	0	5/5	0.988
60	300	8	1500.6	25.46	3.2	5/5	843.591	3697.6	24.86	156	5/5	2.328
60	400	9.4	5394.6	22.50	9	3/5	2014.688	3872.8	22.18	0	5/5	2.317
60	500	10.2	4575.8	17.87	22.2	3/5	1721.630	30 711.4	17.74	319.8	5/5	22.747
70	100	4	726.2	47.41	0.2	5/5	534.214	186	43.68	0	5/5	0.352
70	200	6	2336.6	35.69	0.2	5/5	1963.173	2080.8	34.49	0	5/5	2.606
70	300	8	5149.6	29.50	1.2	1/5	3523.594	4534.8	28.75	0	5/5	3.794
70	400	9	4194.4	24.01	6.4	2/5	2671.340	26 134.4	23.80	0	5/5	20.803
70	500	10.2	2786.4	22.97	3.6	3/5	2001.895	59 357.2	22.59	0	5/5	45.603
80	100	4	2300.8	52.77	0	5/5	3072.642	767	48.37	0	5/5	1.410
80	200	6	2865	40.71	0.2	0/5	3600	7330.6	39.46	0	5/5	5.631
80	300	8	3007	34.57	0.2	0/5	3600	30843.8	33.78	0	5/5	20.033
80	400	9	3019	28.04	2	0/5	3600	670 003.4	27.37	0	5/5	226.621
80	500	10	2943.8	25.62	2	0/5	3600	205 048.4	25.16	0	5/5	183.336
90	100	3.4	1218	48.61	0.2	2/5	2836.722	1585	42.50	0	5/5	5.570
90	200	5.2	1191.4	37.42	0	2/5	2382.604	19 792.8	35.69	0	5/5	22.439
90	300	7.4	1854.6	34.84	0.2	0/5	3600	196 019.4	33.92	0	5/5	124.167
90	400	8.6	1843	31.32	0.4	0/5	3600	173 563.4	30.58	0	5/5	140.159
90	500	10	1920.8	29.50	0.2	0/5	3600	129 278.4	29.08	0	5/5	89.523
100	100	3	1036.8	47.85	0	1/5	3038.317	766.6	40.29	0	5/5	11.589
100	200	5	1266.6	39.87	0	0/5	3600	87 480.4	37.47	0	5/5	139.910
100	300	7	1222	36.11	0	0/5	3600	193 825	34.92	0	5/5	203.539
100	400	8.2	1244.2	32.18	0.4	0/5	3600	796 982.4	31.38	547.2	5/5	840.724
100	500	9.8	1205.8	32.33	0	0/5	3600	984 722	31.97	0	5/5	918.873
100	600	10.8	1272.2	30.45	0.4	0/5	3600	3 069 592	30.02	0	5/5	1847.090
100	700	11.6	1309.4	28.51	1	0/5	3600	3 933 963.2	28.19	0	3/5	2655.668
100	800	12.6	1268.2	27.20	0.6	0/5	3600	8 216 310	26.88	0	2/5	2577.845
100	900	13	1193.4	25.50	0.4	0/5	3600	1 373 011	25.12	58.2	5/5	885.151
100	1000	13.8	1216	24.86	1.4	0/5	3600	3 111 876	24.61	449.8	3/5	2819.580

For each size of $|V|$ and $|L|$, we generate 5 random instances. Consequently, each line of the tables reports the average values over the 5 tested instances of the same size. Columns opt-EC (resp. opt-CC) report the number of instances over 5 that have been solved to optimality within the time limit. Overall we tested 61×5 instances (61 sizes/lines).

From the results, it is clear that the (CCut) formulation performs better than (ECut) formulation. In fact, over the 61 sizes, the (CCut) formulation was able to solve 58 to optimality. That is to say for 58 lines of the tables, all the 5/5 instances have been solved to optimality. However, the (ECut) formulation left 24 lines incomplete, which means that for each size (line) among the 24, at least one instance could not be solved to optimality within the time limit. And among these 24 lines, 16 have 0/5 solved instances, which says that for all these sizes, the (ECut) formulation was unable to solve the generated instances to optimality.

Along with optimality, the (CCut) formulation outperforms the (ECut) formulation with respect to the computational time. In fact, the results for 39 lines over the 61 show an average computational time less than 2 seconds. In addition, for some sizes where the (ECut) formulation was unable to solve any of the generated instances, the (CCut) formulation provides 5/5 optimal solutions within only some seconds. This is mainly the case of instances with $|V| = 80$ and $|L| \geq 200$.

Now concerning the gaps to the root LP relaxation, one can clearly see that the (CCut) gap is slightly better than the (ECut) gap for all the instances (except for line $|V| = 50$ and $|L| = 600$). However, for both formulations the gaps are quite high. In fact, these reach an average value of 52% for the (ECut) formulation (see line $|V| = 80$ and $|L| = 100$), and 48% for the (CCut) formulation (see line $|V| = 80$ and $|L| = 100$). This can be explained by the fact that the average optimal values obtained for all the instances are quite small, not exceeding 14 labels. At this stage, in order to improve the linear relaxations of both (Ecut) and (CCut) formulations, one has to identify new valid inequalities for both, which may tighten the gap and improve the resolution.

We note also that, even having optimality for the majority of the instances, the (CCut) formulation spreads through huge Branch-and-Cut trees compared to the (ECut) formulation. This is clearly depicted in column "Sub-CC" that gives the number of explored subproblems reaching huge values particularly for the instances with $|V| \in \{90, 100\}$. This is mainly due to the fact that the separation phase is ensured only for integer solutions. This is also clear through the number of separated color-cuts that is equal to 0 for 41 out of 61 lines of the tables. As a consequence, improving the separation phase mainly by devising efficient routines has to be considered for future works.

5. Concluding remarks

In this paper, we study the Generalized Labeled Two Edge Connected Subgraph Problem (GLTECSP) that is a variant of the Two Edge Connected Subgraph Problem, a well-studied problem in network design. In this variant, a set of labels (colors) is associated with each edge in the graph, and the objective is to look for a minimum-label 2-edge connected subgraph. We propose two IP formulations for the GLTECSP, the first is based on edge-cuts and the second uses color-cuts. We discuss the separation routines and devise Branch-and-Cut algorithms for both formulations. The experimentations show that the color-cut formulation performs better than the edge-cut one in terms of linear relaxation and computational time. Several instances that could not be solved within the time limit by the edge-cut formulation were solved to optimality by the color-cut formulation within some seconds. In the future, we aim at improving the linear relaxations for both formulations mainly by adding valid inequalities. In fact, even if the formulations are able to solve the majority of the instances, the value of the gaps to the root bounds is significantly large and needs to be reduced.

References

[1] M. Baïou and A.R. Mahjoub, Steiner 2-edge connected subgraph polytopes on series-parallel graphs. *SIAM J. Discrete Math.* **10** (1997) 505–514.

[2] F. Barahona and A.R. Mahjoub, On two-connected subgraph polytopes. *Discrete Math.* **147** (1995) 19–34.

[3] A. Bordini, F. Protti, T.G. da Silva, and G.F. de Sousa Filho, New algorithms for the minimum coloring cut problem. *Int. Trans. Oper. Res.* **26** (2019) 1868–1883.

[4] S.C. Boyd and T. Hao, An integer polytope related to the design of survivable communication networks. *SIAM J. Discrete Math.* **6** (1993) 612–630.

[5] H. Broersma and X. Li, Spanning trees with many or few colors in edge-colored graphs. *Discuss. Math. Graph Theory* **17** (1997) 259–269.

[6] M.E. Captivo, J.C. Clímaco and M.M. Pascoal, A mixed integer linear formulation for the minimum label spanning tree problem. *Comput. Oper. Res.* **36** (2009) 3082–3085.

[7] R. Cerulli, A. Fink, M. Gentili and S. Voß, Metaheuristics comparison for the minimum labelling spanning tree problem. In: The Next Wave in Computing, Optimization, and Decision Technologies. Springer (2005) 93–106.

[8] R. Cerulli, A. Fink, M. Gentili and S. Voß, Extensions of the minimum labelling spanning tree problem. *J. Telecommun. Inf. Technol.* (2006) 39–45.

[9] R.-S. Chang and L. Shing-Jiuan, The minimum labeling spanning trees. *Inf. Process. Lett.* **63** (1997) 277–282.

[10] S. Consoli, J.A.M. Pérez and N. Mladenović, Intelligent optimization for the minimum labelling spanning tree problem. In: International Conference on Learning and Intelligent Optimization. Springer (2013) 19–23.

[11] C.R. Coullard, A. Rais, R.L. Rardin and D.K. Wagner, Linear-time algorithms for the 2-connected steiner subgraph problem on special classes of graphs. *Networks* **23** (1993) 195–206.

[12] C.R. Coullard, A. Rais, R.L. Rardin and D.K. Wagner, The dominant of the 2-connected-steiner-subgraph polytope for w4-free graphs. *Discrete Appl. Math.* **66** (1996) 33–43.

[13] T.G. Da Silva, The Minimum Labeling Spanning Tree and Related Problems. Avignon University, France (2018).

[14] T.G. Da Silva, S. Gueye, P. Michelon, L.S. Ochi and L.D.A.F. Cabral, A polyhedral approach to the generalized minimum labeling spanning tree problem. *EURO J. Comput. Optim.* **7** (2019) 47–77.

[15] T.G. Da Silva, L.S. Ochi, P. Michelon, S. Gueye and L.A. Cabral, Solving the minimum labeling global cut problem by mathematical programming. Preprint arXiv:1903.04319 (2019).

[16] T.G. Da Silva, E. Queiroga, L.S. Ochi, L.D.A.F. Cabral, S. Gueye, and P. Michelon, A hybrid metaheuristic for the minimum labeling spanning tree problem. *Eur. J. Oper. Res.* **274** (2019) 22–34.

[17] T. Dutta, L.S. Heath, V.A. Kumar and M.V. Marathe, Labeled cuts in graphs. *Theor. Comput. Sci.* **648** (2016) 34–39.

[18] L. Faria, S. Klein, I. Sau, U.S. Souza and R. Sucupira, Maximum cuts in edge-colored graphs. *Discrete Appl. Math.* **281** (2020) 229–234.

[19] L.R. Ford and D.R. Fulkerson, Maximal flow through a network. *Can. J. Math.* **8** (1956) 399–404.
[20] A.V. Goldberg and R.E. Tarjan, A new approach to the maximum-flow problem. *J. Assoc. Comput. Mach.* **35** (1988) 921–940.
[21] R.E. Gomory and T.C. Hu, Multi-terminal network flows. *J. Soc. Ind. Appl. Math.* **9** (1961) 551–570.
[22] D. Granata, R. Cerulli, M.G. Scutella and A. Raiconi, Maximum flow problems and an np-complete variant on edge-labeled graphs. *Handb. Comb. Optim.* (2013) 1913–1948.
[23] M. Grötschel, C.L. Monma and M. Stoer, Design of survivable networks. *Handb. Oper. Res. Manage. Sci.* **7** (1995) 617–672.
[24] D. Gusfield, Very simple algorithms and programs for all pairs network flow analysis. Technical report, Computer Science Division, University of California, Davis (1987).
[25] D. Gusfield, Very simple methods for all pairs network flow analysis. *SIAM J. Comput.* **19** (1990) 143–155.
[26] N. Jozefowiez, G. Laporte and F. Semet, A branch-and-cut algorithm for the minimum labeling hamiltonian cycle problem and two variants. *Comput. Oper. Res.* **3** (2011) 1534–1542.
[27] H. Kerivin and A.R. Mahjoub, Design of survivable networks: a survey. *Networks: Int. J.* **46** (2005) 1–21.
[28] A.R. Mahjoub, Two-edge connected spanning subgraphs and polyhedra. *Math. Program.* **64** (1994) 199–208.
[29] J. Perez and S. Consoli, On the minimum labelling spanning bi-connected subgraph problem. Preprint arXiv:1505.01742 (2015).
[30] R. Vaisman, Finding minimum label spanning trees using cross-entropy method. *Networks* **79** (2022) 220–235.
[31] X. Wen, H. Broersma, Z.-B. Zhang and X. Zhang, On the complexity of edge-colored subgraph partitioning problems in network optimization. *Discrete Math. Theor. Comput. Sci.* **17** (2016). DOI: 10.46298/dmtcs.2159.
[32] Y. Xiong, B. Golden and E. Wasil, The colorful traveling salesman problem. In: Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies. Springer (2007) 115–123.