Complexity and Exact Solution Approaches to the Minimum Changeover Cost Arborescence Problem

Giulia Galbiati[a], Stefano Gualandi[b], Emiliano Traversi[c], Frank Baumann[c]

[a]*Dipartimento di Ingegneria Industriale e dell'Informazione*
[b]*Dipartimento di Matematica*
Università degli Studi di Pavia, Via Ferrata 1, 27100 Pavia, Italy
{giulia.galbiati,stefano.gualandi}@unipv.it

[c]*Department of Mathematics, TU Dortmund, Vogelpothsweg 87, D-44227 Dortmund, Germany*
{frank.baumann,emiliano.traversi}@math.tu-dortmund.de

Abstract

We are given a digraph $G = (N, A)$, where each arc is colored with one among k given colors. We look for a spanning arborescence T of G rooted at a given node and having minimum changeover cost. We call this the Minimum Changeover Cost Arborescence problem. To the authors' knowledge, it is a new problem. The concept of changeover costs is similar to the one, already considered in the literature, of reload costs, but the latter depend also on the amount of commodity flowing in the arcs and through the nodes, whereas this is not the case for the changeover costs. Here, given any node j different from the root, if a is the color of the single arc entering node j in arborescence T, and b is the color of an arc (if any) leaving node j, then these two arcs contribute to the total changeover cost of T by the quantity d_{ab}, a non-negative entry of a k-dimensional square matrix D. We first prove that the problem is NPO-complete and very hard to approximate. Then we present a greedy heuristic together with combinatorial lower and upper bounds, a Binary Quadratic Programming formulation, and exact solution approaches based on branch-and-cut solvers. Finally, we report extensive computational results and exhibit a set of very challenging instances.

Keywords: network optimization, changeover cost, reload cost model, computational complexity, integer programming

1. Introduction

The problem that we consider in this paper and that we call Minimum Changeover Cost Arborescence, is formulated on a digraph $G = (N, A)$ having n nodes and m arcs, where each arc comes with a color (or label) out of a set C of k colors. The problem looks for a spanning arborescence T rooted at a specific node, say node 1 (wlog), and having minimum *changeover cost*, a cost that we now describe. We assume that G contains at least one spanning arborescence having node 1 as root, i.e., a cycle-free spanning subgraph containing a directed path from node 1 to all other nodes of G. Let T be any such arborescence; obviously T has only one arc entering each node except node 1, having no entering arc. Consider any node j of T different from the root. A cost at j is paid for each outgoing arc and depends on the colors of this arc and of the one entering j. Such costs are given via a $k \times k$ matrix D of non-negative

rationals: the entry $d_{a,b}$ specifies the cost to be paid at node j for one of the outgoing arcs (if any) colored b, when the incoming arc of j is colored a. We define the *changeover cost at* j, denoted by $d(j)$, as the sum of the costs at j paid for each of its outgoing arcs, and we call the sum of the changeover costs $d(j)$, over all nodes j different from 1, the *changeover cost of* T, denoted by $d(T)$.

Similar problems have recently received some attention in the literature [24, 12, 14, 15, 2, 16], where the entries of our matrix D are called *reload costs*. However, in all these problems the objective functions depend on the amount of commodity flowing in the arcs or edges of the given graph, whereas this is not the case for the problem that we consider, which, to the authors' knowledge, has not been studied in the literature. In the seminal work [24] and in [12] the problem of finding a spanning tree with minimum reload cost diameter is thoroughly analyzed. In [14] the problem of finding a spanning tree which minimizes the sum of the reload costs of all paths between all pairs of nodes is presented, whereas in [15] this problem is generalized and thoroughly discussed together with several variations of reload cost spanning tree problems. The problems of minimum reload cost path-trees, tours and flows are studied in [2], whereas the minimum reload $s-t$-path, trail and walk problems are analyzed in [16].

The motivation for introducing here changeover cost of arborescences comes from the need to model in a real network the fixed costs for installing, in each node, devices to support the changes of carrier, that are modeled with the changes of color.

The Minimum Changeover Cost Arborescence (MinCCA for short from now on) is thoroughly analyzed in this paper, both from a theoretical and a computational point of view. In Section 2 we show that it is NPO-complete and very hard to approximate. We present in Section 3 a greedy algorithm together with combinatorial lower and upper bounds, and in Section 4 a Binary Quadratic Programming formulation together with various exact solution approaches based on branch-and-cut solvers. Section 5 reports extensive computational results and comparisons among the methods, and exhibits a set of very challenging instances. Section 6 presents some conclusions.

A preliminary version of this work has appeared in [13]. Here, in addition to presenting a new theorem (Theorem 2) which states that MinCCA is very hard to approximate also in a very restrictive case, we thoroughly revise our branch-and-cut algorithm, based on a linear formulation of the problem, which now is able to solve instances that were unsolvable in [13]. We also compare our branch-and-cut algorithm with two other branch-and-cut approaches, the first based on a convex formulation of the problem, the second based on the optimization library SCIL (Symbolic Constraints in Integer linear Programming[1]). We show that in practice our method is far better than the two proposed benchmarks.

2. Complexity

In this section we prove that MinCCA is NPO PB-complete and hard to approximate, even in the very restrictive formulations presented in the next two theorems.

[1] `http://www.scil-opt.net/` (last visited August, 6-th, 2013)

The first result, given in Theorem 1 below, is obtained by exhibiting a reduction from the NPO PB-complete problem Minimum Ones, which is defined as follows. Given a set $Z = \{z_1, ..., z_n\}$ of n boolean variables and a collection $C = \{c_1, ..., c_m\}$ of m disjunctive clauses of at most 3 literals, the problem looks for a subset $Z' \subseteq Z$ having minimum cardinality and such that the truth assignment for Z that sets to true the variables in Z' satisfies all clauses in C. The trivial solution, returned when the collection of clauses is unsatisfiable, is set Z.

In [21] it is shown that Minimum Ones is NPO PB-complete, and in [20] that it is not approximable within $|Z|^{1-\epsilon}$ for any $\epsilon > 0$.

We now present the reduction that we use in the proof of Theorem 1. Given any instance I of Minimum Ones we construct the corresponding instance I' of MinCCA as follows (see Figure 2 for an example). Graph $G = (N, A)$ has the vertex set N that contains a vertex $a_0 = 1$, the vertices a_i, b_i, c_i, d_i, for each $i = 1, ..., n$, and the vertices $c_1, ..., c_m$ corresponding to the elements of C. Moreover N contains, for each $i = 1, ..., n$, many couples of vertices $z_i^j, \overline{z}_i^j$, with j having values from 1 to the maximum number of times that either z_i or $\overline{z}_i$ appear in the clauses of C. The arc set A of G contains all arcs (a_i, a_{i+1}), for $i = 0, ..., n-1$, all arcs (a_i, b_i), (a_i, c_i), (b_i, d_i), (c_i, d_i), (d_i, z_i^1), $(d_i, \overline{z}_i^1)$, for $i = 1, ..., n$, and all arcs (z_i^j, z_i^{j+1}) and $(\overline{z}_i^j, \overline{z}_i^{j+1})$, for the increasing values of j, starting from 1. Moreover there is an arc from z_i^j to $\overline{z}_i^j$ and vice versa, for each involved i and j. Finally A contains an arc from z_i^j (resp. $\overline{z}_i^j$) to vertex c_k if the j-th occurrence of variable z_i (resp. $\overline{z}_i$) in the clauses of C, if it exists, appears in clause c_k. The colors for the arcs of G are taken from the set $\{r, g, v\}$. Color r is assigned to all arcs (a_i, a_{i+1}), (a_i, b_i), and (a_i, c_i); color g to all arcs (b_i, d_i), (d_i, z_i^1), (z_i^j, z_i^{j+1}) and all arcs from some vertex z_i^j to some vertex c_k; symmetrically color v is assigned to all arcs (c_i, d_i), $(d_i, \overline{z}_i^1)$, $(\overline{z}_i^j, \overline{z}_i^{j+1})$ and all arcs from some vertex $\overline{z}_i^j$ to some vertex c_k; finally all arcs from z_i^j to $\overline{z}_i^j$ receive color g, whereas all arcs from $\overline{z}_i^j$ to z_i^j receive color v. The matrix D of costs is defined so that $d(r, g) = 1$, $d(g, v) = d(v, g) = M$, with $M > n^2$ being a suitable big integer, and all other costs equal to 0. If we let n' denote the number of nodes of G, it is not difficult to see that $n' \leq \beta n^3$, for some constant β, with $\beta > 1$ and independent from the number n of boolean variables. For this purpose it is enough to convince ourselves that the following inequalities hold:

$$n' \leq 1 + 6n + 7m \leq 1 + 6n + 7\binom{2n}{3} = 1 + 6n + \frac{7}{3}(2n^2 - n)(2n - 2) \leq \beta n^3.$$

It is also not difficult to see that $opt(I) \leq opt(I')$ since to any solution of I' of cost $t < M$ there corresponds a solution to I having the same cost t; if, on the other hand, $opt(I') \geq M$ then $opt(I) \leq n < opt(I')$. Notice that the equality $opt(I) = opt(I')$ holds iff I is a satisfiable instance, since to any non trivial solution of I there corresponds a solution to I' having the same cost, whereas to a trivial solution of I there corresponds a solution to I' having a cost greater than M.

Before presenting Theorem 1 we introduce the notion of costs satisfying the *triangle inequality*.

We say that the costs satisfy the *triangle inequality* if, for any node v having an entering arc of color l and an outgoing arc of color l', if u is any node such that (v, u) and (u, v) are arcs of colors respectively l'' and l''', then $d_{ll'} \leq d_{ll''} + d_{l''l'''} + d_{l'''l'}$ (see Figure 1).

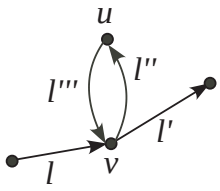

Figure 1: Triangle inequality for costs in directed graphs.

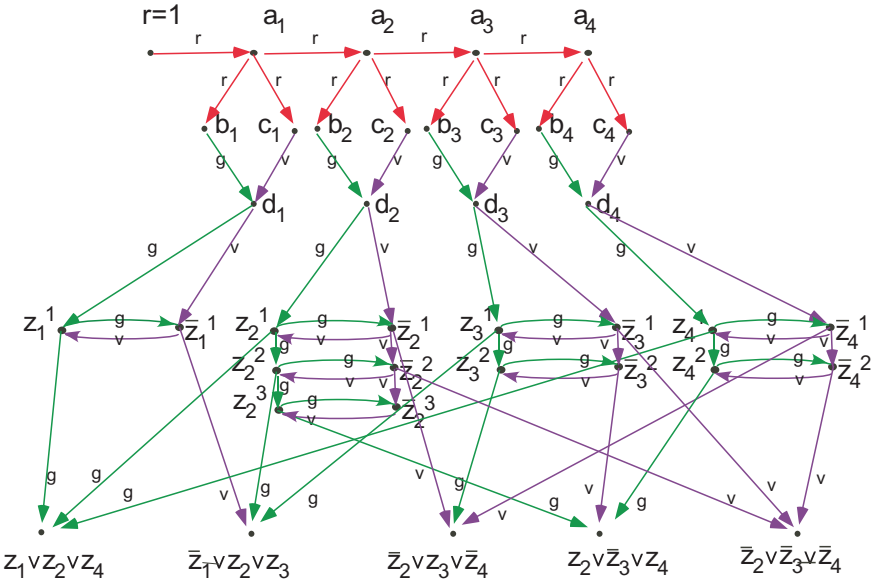

Figure 2: The graph $G = (N, A)$ corresponding to an instance I of Minimum Ones having as collection of clauses the set $C = \{z_1 \vee z_2 \vee z_4, \bar{z}_1 \vee z_2 \vee z_3, \bar{z}_2 \vee z_3 \vee \bar{z}_4, z_2 \vee \bar{z}_3 \vee z_4, \bar{z}_2 \vee \bar{z}_3 \vee \bar{z}_4\}$.

Now we can prove the following theorem.

Theorem 1. *Problem* MinCCA *is NPO-complete. Moreover, there exists a real constant* α, $0 < \alpha < 1$, *such that, unless* P=NP, *the problem is not approximable within* $\alpha\sqrt[3]{n^{1-\varepsilon}}$, *for any* $\varepsilon > 0$, *even if formulated on graphs having bounded in and out degrees, costs satisfying the triangle inequality and 3 colors on the arcs.*

Proof. We use the reduction described at the beginning of this section. Obviously the graph constructed from an instance I of Minimum Ones is a graph $G = (N, A)$ that has maximum in-degree and out-degree equal to 3, uses $k = 3$ colors, and has costs that satisfy the triangle inequality. Moreover we know that the number n' of its nodes satisfies the inequality $n' \leq \beta n^3$, with $\beta > 1$.

We show now that, if we let $\alpha = \frac{1}{\sqrt[3]{\beta}}$, then MinCCA is not approximable within $\alpha\sqrt[3]{n'^{1-\varepsilon}}$, for any $\varepsilon > 0$. Suppose on the contrary that there exists an $\bar{\varepsilon} > 0$ and that MinCCA is approximable within $\alpha\sqrt[3]{n'^{1-\bar{\varepsilon}}}$. The algorithm that we now describe could then be used to approximate Minimum Ones within $n^{1-\bar{\varepsilon}}$, contrary to the result in [20].

The algorithm, given an instance I of Minimum Ones, would construct the corresponding instance I' of MinCCA

and then find an approximate solution for it, i.e. a spanning arborescence T having a changeover cost $d(T)$ satisfying the inequality $\frac{d(T)}{opt(I')} \leq \alpha\sqrt[3]{n'^{1-\bar{\varepsilon}}}$. If $d(T) \geq M$ the algorithm would return the trivial solution, otherwise, if $d(T) < M$, the algorithm would use T to construct and return a solution Z' for instance I having $|Z'| = d(T)$.

Let us verify that this algorithm approximates Minimum Ones within $n^{1-\bar{\varepsilon}}$. If I is not satisfiable this algorithm obviously returns the trivial solution and, following [3] (pag. 254), the behavior of an approximation algorithm is not measured in this case. If I is satisfiable it follows easily that $opt(I') \leq n$; it must also be that $d(T) < M$, otherwise the inequality $d(T) \leq opt(I')\alpha\sqrt[3]{n'^{1-\bar{\varepsilon}}} \leq opt(I')\alpha\sqrt[3]{\beta n^3} \leq opt(I')\, n \leq n^2$ would contradict the inequality $n^2 < M \leq d(T)$. Hence in this case the algorithm uses T to construct and return a solution Z' having $|Z'| = d(T)$ and, as observed at the end of the reduction, in this case we know that $opt(I) = opt(I')$. Hence we can derive the following inequalities

$$\frac{|Z'|}{opt(I)} = \frac{d(T)}{opt(I')} \leq \alpha\sqrt[3]{n'^{1-\bar{\varepsilon}}} \leq \alpha\sqrt[3]{(\beta n^3)^{1-\bar{\varepsilon}}} \leq n^{1-\bar{\varepsilon}}$$

which conclude the proof. □

The second result of this section is reported in Theorem 2 below, which deals with the special case in which only two colors are allowed, showing that even in this restrictive formulation the problem turns out not to be approximable within a constant. In the proof of the theorem we use a reduction from the Minimum Set Cover problem, which has as instance a collection $C = \{S_1, ..., S_c\}$ of c subsets of a set $S = \{e_1, ..., e_n\}$ of n elements and as solution a set cover for S, i.e., a subset $C' \subset C$ such that every element in S belongs to at least one member of C', which has minimum cardinality. It is known that Minimum Set Cover, even in the restricted formulation with instances having a number of subsets bounded by a polynomial in the number of elements, is not approximable within $c \log n$, for some $c > 0$ [9], unless $NP \subseteq DTIME(n^{O(\log \log n)})$.

Given any instance I of Minimum Set Cover we construct the corresponding instance I' of MinCCA as follows (see Figure 3a for an example). Graph $G = (N, A)$ has the vertex set N that contains vertices r and s, and then for each element S_j of the collection C it contains a couple of vertices a_j and s_j, and for each element e_j of S a vertex e_j. The arc set A contains the arcs $\{r, a_1\}$ and $\{a_c, s\}$, all arcs $\{a_i, s_i\}$ and $\{s, s_i\}$, $i = 1, ..., c$, and all arcs $\{a_i, a_{i+1}\}$, $i = 1, ..., c-1$. Moreover if set S_j contains element e_i then arc $\{s_j, e_i\}$ belongs to A. All arcs have color black (b), except the $\{a_i, s_i\}$ and those leaving the s_i, $i = 1, ..., c$, that have color white (w). The matrix D of costs is defined so that all costs are zero except $d(b, w)$ which is equal to one. It is straightforward to see that $opt(I) = opt(I')$ and that, if we let n' be the cardinality of N then $n' = 2 + 2c + n$. If we use the restricted formulation of Minimum Set Cover we can assume that $c \leq n^k$, for some integer k, hence obtaining that $n' \leq 3 \cdot n^k$.

We can now prove the next theorem.

Theorem 2. *Unless $NP \subseteq DTIME(n^{O(\log \log n)})$, problem* MinCCA *is not approximable within $\beta \log n$, for some constant $\beta > 0$, even if formulated on graphs having bounded in and out degrees, costs satisfying the triangle inequality and 2 colors on the arcs.*

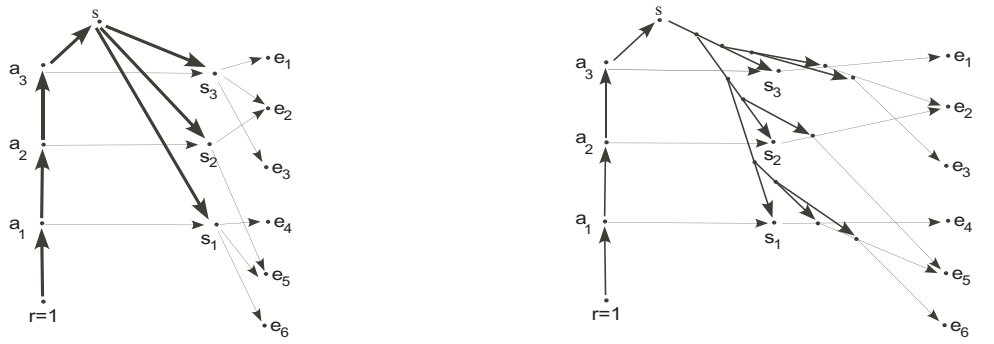

Figure 3: (a) The graph $G = (N, A)$ corresponding to an instance I of Minimum Set Cover having as collection C of subsets of the finite set $S = \{e_1, ..., e_6\}$ the sets $S_3 = \{e_1, e_2, e_3\}$, $S_2 = \{e_2, e_5\}$ and $S_1 = \{e_4, e_5, e_6\}$. (b) Graph G modified to have bounded in and out degrees.

Proof. We use the reduction presented above from the Minimum Set Cover problem. Suppose that problem MinCCA is approximable within $\beta \log n$, for any constant $\beta > 0$. Then, given an instance I of Minimum Set Cover we construct the corresponding instance I' of MinCCA and find an approximate solution within $\beta \log n$ of the optimum, for a value of $\beta > 0$ that we now choose appropriately. If tree T is the approximate solution returned, then it is easy to obtain from T a solution to Minimum Set Cover having a cardinality s bounded above by the changeover cost $d(T)$ of T. We have therefore that $\frac{s}{opt(I)} \leq \frac{d(T)}{opt(I')} \leq \beta \log n' \leq \beta \log(3n^k) = \beta\, d \log n$, for a suitable constant $d > 0$. Since Minimum Set Cover is not approximable within $c \log n$, for some $c > 0$ [9], then choosing β such that $\beta\, d = c$ would yield an absurd. To conclude the proof we only need to observe that graph G can be easily modified to have bounded in and out degrees, without altering the conclusions. Figure 3b shows, for instance, the modifications that could be applied to graph G in Fig. 3a to have bounded in and out degrees.

□

3. Heuristics and Bounds

3.1. Combinatorial Lower and Upper Bounds

A simple lower bound to the objective function of our problem can be derived by applying the technique used by Gilmore and Lawler to derive their well–known lower bound on the Quadratic Assignment Problem [22]. For a review of their technique in a more general setting, see [6].

Let p_e be the smallest possible contribution to the objective function of MinCCA given by arc $e = (i, j)$, if this arc is part of the optimal solution, that is:

$$p_e = p_{(i,j)} = \min_{f \in \delta^-(i)} d_{c(f),c(e)}, \tag{1}$$

where, $\delta^+(i)$ and $\delta^-(i)$ denote the set of outgoing arcs and the set of incoming arcs, respectively. Computing p_e takes time $O(n)$ or $O(d_i)$, where d_i is the in-degree of vertex i.

Once we have collected the values of p_e for every $e \in A$, we can solve the Minimum Weight Arborescence problem (with root node $r = 1$ and the arc weights set to p_e) in time $O(m + n \log n)$ [11]. The weight of the optimal solution of this Minimum Weight Arborescence problem provides a lower bound on the optimal value of MinCCA, whereas the changeover cost of such solution yields an upper bound.

3.2. Greedy Algorithm

Let digraph $G = (N, A)$ be as described in the Introduction and let $A_i \subset A$ be the set of arcs having the same i-th color, out of the set $C = \{1, \ldots, k\}$ of colors. For each $i \in C$ consider the digraph $G_i = (N, A_i)$ and let (N_i, T_i) be a largest (non necessarily spanning) arborescence of G_i rooted at node 1, that is a largest set of nodes N_i reachable from node 1 in G_i. Denote the cardinality of N_i by σ_i. We describe now a constructive greedy heuristic for the MinCCA problem.

Algorithm 1 MinCCA Greedy Heuristic.

1: **for** i:=1 **to** k **do** compute σ_i;
2: let j be such that $\sigma_j = \max\{\sigma_1, \ldots, \sigma_k\}$;
3: $T := T_j$;
4: **while** $(|T| \neq n - 1)$ **do**
5: delete from A all arcs in $A \setminus T$ entering nodes already reached from 1 in (N, T);
6: **forall** $h := 1, \ldots, k$ **do**
7: let A_h be the set of arcs in $A \setminus T$ colored h;
8: find in $(N, T \cup A_h)$ the (not necessarily spanning) arborescence (N, T_h) rooted at node 1, that has minimum changeover cost d_h among those having maximum cardinality;
9: $c_h := \frac{d_h}{|T_h| - |T|}$;
10: **end do**
11: let j be such that $c_j = \min\{c_1, \ldots, c_k\}$;
12: $T := T_j$;
13: **end while**
14: **return** (N, T);

In order to assert that this algorithm is polynomial we only need to show that the very particular case of MinCCA in line 8 is polynomially solvable. In fact, let G' be the subgraph of $(N, T \cup A_h)$ induced by the subset $N' \subseteq N$ of nodes reachable from node 1. Consider any arc a of G' of color h having its tail in a node already reached from 1,

with incoming arc b of T, and set the weight of this arc equal to $d_{c(b),h}$. Set the weights of all other arcs of G' to zero. The minimum weight spanning arborescence of G' rooted in node 1 gives T_h.

4. Formulations

The MinCCA problem has the same feasible region of the Minimum Weight Rooted Arborescence problem: The set of spanning arborescences. The latter problem is solvable in polynomial time [8]. It can be solved in polynomial time also by Linear Programming, since we know its convex hull.

Proposition 1. (Schrijver [23], pg. 897) *Let $G = (N, A)$ be a digraph, r a vertex in N such that G contains a spanning arborescence rooted at r. Then the r-arborescence polytope of G is determined by:*

$$\left\{ x \mid x \geq 0, \sum_{a \in \delta^+(X)} x_a \geq 1 \quad \forall X \subset N \text{ with } r \in X, \sum_{a \in \delta^-(i)} x_a = 1 \quad \forall i \in N \setminus \{r\} \right\}.$$

Therefore, to formulate MinCCA as an Integer Program we can use the same polyhedral description of the feasible region. However, since we have a quadratic objective function, the continuous relaxation is no longer integral.

The MinCCA problem has the following Binary Quadratic Programming formulation:

$$\min \quad \sum_{i \in N \setminus \{r\}} \sum_{a \in \delta^-(i)} \sum_{b \in \delta^+(i)} d_{c(a),c(b)} x_a x_b \tag{2}$$

$$\text{s.t.} \quad \sum_{a \in \delta^+(X)} x_a \geq 1 \qquad \forall X \subset N, r \in X \tag{3}$$

$$\sum_{a \in \delta^-(i)} x_a = 1 \qquad \forall i \in N \setminus \{r\} \tag{4}$$

$$x_a \in \{0, 1\} \qquad \forall a \in A. \tag{5}$$

The objective function (2) takes into account the changeover cost occurring at each node i except the root node r. If both the incoming arc a (with color $c(a)$) and the outgoing arc b (with color $c(b)$) are part of the arborescence, that is the corresponding variables x_a and x_b are equal to 1, we pay for the cost $d_{c(a),c(b)}$. Constraints (3) are the well–known r-cut inequalities, which force every proper subset of N that contains the root node r to have at least one outgoing arc. They are exponential in number, but they can be separated in polynomial time by solving a r-min-cut problem. Constraints (4) are the in-degree constraints.

4.1. Linear formulation

The objective function (2) can be linearized in a standard way by introducing binary variables z_{ab} in place of the quadratic term $x_a x_b$ and the linking constraints:

$$z_{ab} \geq x_a + x_b - 1, \quad z_{ab} \leq x_a, \quad z_{ab} \leq x_b.$$

Indeed, since we are minimizing and the costs d are assumed to be non-negative, the last two inequalities can be omitted, obtaining the following Integer Linear Program:

$$\min \quad \sum_{i \in N \setminus \{r\}} \sum_{a \in \delta^-(i)} \sum_{b \in \delta^+(i)} d_{c(a),c(b)} z_{ab} \tag{6}$$

$$(3) - (5)$$

$$z_{ab} \geq x_a + x_b - 1 \qquad \forall i \in N \setminus \{r\}, a \in \delta^-(i), b \in \delta^+(i) \tag{7}$$

$$z_{ab} \geq 0. \tag{8}$$

In practice, the continuous relaxation of (6)–(8) provides rather weak bounds. Several techniques are available in the literature for improving the formulation obtained after linearizing a binary quadratic problem (See for example [6], [5], and [19]). A common idea in these approaches is to strengthen the formulation at the cost of increasing the size of the problem in terms of number of constraints and variables used. Selecting which inequalities to use in practice, among the different classes proposed is not an easy task.

We have exploited the technique where a constraint of the form $a^T x \leq b$ is multiplied on both sides by a variable x_i. Since for $0-1$ variables we have that $x_i^2 = x_i$, the constraint becomes:

$$\sum_{j \neq i} a_j x_j x_i \leq (b - a_i) x_i,$$

whose linearized version is:

$$\sum_{j \neq i} a_j z_{ij} \leq (b - a_i) x_i,$$

where the z_{ij} are either variables already introduced to linearize (2) or are new variables. If we apply this technique to constraints (3), we do not get stronger relaxations, since the constraints have the form $a^T x \geq b$ with all coefficients equal to one, so that we would get $\sum_{j \neq i} x_j x_i \geq 0$, that is useless.

Any criterion for selecting the type of valid inequalities to be added should keep the overall size of the problem under control. In [13] we have multiplied constraints (4) and a redundant constraint on the number of arcs (i.e., $\sum_{a \in A} x_a = n - 1$) with all possible variables x_i, introducing many additional z_{ij} variables that were not appearing in the objective function. As a consequence, we obtained a too heavy formulation. In this work we apply instead a more careful selection of the variables to be multiplied only with constraints (4), and we discard completely the redundant constraint on the number of arcs. More precisely, we multiply every constraint (4) only with the variables associated to an outgoing arc. This yields the following quadratic constraints:

$$\sum_{a \in \delta^-(i)} x_a \, x_b = x_b, \quad \forall i \in N \setminus \{r\}, \forall b \in \delta^+(i).$$

whose linearized version is:

$$\sum_{a \in \delta^-(i)} z_{ab} = x_b, \quad \forall i \in N \setminus \{r\}, \forall b \in \delta^+(i). \tag{9}$$

Note that in (9) only z_{ab} variables already introduced in the objective function appear, so that the overhead of this linearization is of at most m constraints of type (9) per node, but no additional variables are added.

This idea has allowed us to solve to optimality almost all the instances presented as challenging in [13].

4.2. Convex formulation

An alternative to linearizing the quadratic terms in the objective function is to reformulate (2) into a convex function. Note that MinCCA is not convex since all the diagonal terms in (2) are equal to zero (i.e., $d_{c(a),c(a)} = 0, \forall a \in A$). Nowadays, commercial solvers are able to automatically reformulate binary quadratic problems into convex problems only when the complete set of constraints is statically defined. However, our formulation is based on constraints (3) which are exponentially many and are dynamically added only when they are violated.

We have therefore reformulated MinCCA into a convex problem "manually" by using the eigenvalue method introduced in [17], which is briefly presented next. Let Q be the $m \times m$ symmetric cost matrix obtained by carefully expanding the summations in (2), that is:

$$Q_{ab} = Q_{ba} = \begin{cases} \frac{1}{2} d_{c(a),c(b)} & \text{if } \exists\, i \in N, a \in \delta^-(i), b \in \delta^+(i), \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The eigenvalue method consists of first computing the most negative eigenvalue $\lambda_{\min}$ of Q and then using, instead of (2), the following convex objective function:

$$\min \quad x^\top (Q - \lambda_{\min} I) x + \lambda_{\min} e^\top x \tag{11}$$

where e is the all-ones vector and I is the identity matrix. For a proof of the equivalence between (2) and (11) when the variables are restricted to range in $\{0, 1\}$, we refer the reader to the result presented in [17].

4.3. Symbolic Constraints Formulation

As an additional solution approach to MinCCA, we used the optimization library SCIL (Symbolic Constraints in Integer linear Programming[2]), which is based on a efficient branch-and-cut framework. In contrast to other optimization software, SCIL allows to specify *symbolic constraints*, which were introduced in [1]. The symbolic constraints of SCIL provide efficient separation routines for the underlying combinatorial structure of the problem (e.g., it is possible to use cut, matching, or spanning tree symbolic constraints). In addition, SCIL offers several options to handle binary polynomial objective functions. The simplest option for binary quadratic objective functions consists of, firstly, automatically performing a standard linearization and, secondly, strengthening the resulting relaxation by exploiting the equivalence between binary quadratic optimization and the Max-Cut problem, which permits to add cutting planes valid for the Max-Cut polytope [5].

[2] `http://www.scil-opt.net/` (last visited August, 6-th, 2013)

In our solution approach based on SCIL, we used the symbolic constraint *Steiner arborescence* and we declared every node except the root as a terminal node. The symbolic constraint for the Steiner arborescence implemented in SCIL relies on several classes of inequalities. In particular, the degree equations for the terminal nodes as well as the degree inequalities for the Steiner nodes are added statically; the cut-constraints are separated with the Hao-Orlin Min-Cut algorithm; additional cut-constraints are obtained by reversing all edges of the graph, as described in [7]. This last class of inequalities ensures that Steiner nodes with an active in-edge are reachable from the root node [10].

5. Computational Results

The greedy heuristic described in Section 3 and the exact solution approaches based on branch-and-cut algorithms presented in Section 4 are evaluated using a wide collection of instances. These instances are generated using the data in the SteinLib, which are a collection of Steiner tree problems on graphs (`http://steinlib.zib.de/`). In particular, we have used the data sets named `B`, `C` `I080`, `I160`, `es20fst`, `es30fst`, and `es40fst`. Since the Steiner library contains undirected graphs, we first introduced for each edge $\{i, j\}$ two arcs (i, j) and (j, i). Each new arc gets a color randomly drawn from $\{1, \ldots, k\}$, where the number k of colors ranges in the set $\{2, 3, 5, 7\}$. We consider two types of changeover costs: *uniform* costs, i.e., all costs equal to 1 whenever there is a change of color, and *non-uniform* costs, randomly drawn from $\{1, \ldots, 10\}$.

We implemented all the algorithms in C++, using the g++ 4.3 compiler and the CPLEX12.3 callable library. Constraints (3) are separated using the Min-Cut algorithm by Hao and Orlin [18] implemented in the COIN-OR LEMON graph library[3]. The tests are run on a standard desktop with 2Gb of RAM and a i686 CPU at 1.8GHz. For all the instances we set a time limit of 3600 seconds. All instances used for the experiments are available at `http://www-dimat.unipv.it/~gualandi/Resources.html`.

5.1. Heuristics and Bounds

The first set of results is used to evaluate the combinatorial and the greedy heuristic described in Section 3. Table 1 gives the lower and upper bound obtained with the combinatorial heuristic (LB_1 and UB_1, respectively) and the upper bound obtained with the greedy heuristic (UG) together with the corresponding computation times in seconds. For each instance, the best upper bound is reported in bold. The combinatorial heuristic has two advantages: first, it is extremely fast, and second, it provides both lower and upper bounds. The greedy heuristic, however, provides better solutions in average, with only a slight increase in the computation time. Therefore, in our branch-and-cut algorithm described next, we always execute both heuristics and we select the best solution as initial upper bound.

5.2. Lower bounds

A second set of results allows us to compare the strength of the proposed lower bounds:

[3] `http://lemon.cs.elte.hu/trac/lemon` (last visited August, 6-th, 2013)

Instance	n	m	LB_1	UB_1	time	UG	time	Instance	n	m	LB_1	UB_1	time	UG	time
b13	100	250	31	**62**	0.0	65	0.01	c01	500	1250	201	385	0.0	**380**	0.06
b14	100	250	33	70	0.0	**59**	0.0	c02	500	1250	183	**357**	0.0	360	0.06
b15	100	250	36	71	0.0	**68**	0.0	c03	500	1250	207	**378**	0.0	**378**	0.05
b16	100	400	6	63	0.0	**42**	0.01	c04	500	1250	175	**341**	0.0	359	0.05
b17	100	400	7	63	0.0	**46**	0.0	c05	500	1250	189	**365**	0.0	372	0.06
i160-001	160	480	40	**105**	0.0	110	0.01	c06	500	2000	47	348	0.0	**270**	0.04
i160-002	160	480	40	101	0.0	**100**	0.01	c07	500	2000	56	364	0.0	**262**	0.04
i160-003	160	480	46	120	0.0	**98**	0.01	c08	500	2000	36	335	0.0	**289**	0.04
i160-004	160	480	32	**96**	0.0	**96**	0.01	c09	500	2000	42	339	0.0	**258**	0.04
i160-005	160	480	33	100	0.0	**93**	0.01	c10	500	2000	41	339	0.0	**276**	0.03

Table 1: Instances with uniform changeover costs and 5 colors: comparison of the combinatorial and the greedy heuristic.

LB_1 is the simple combinatorial lower bound.

LB_2 is the one obtained solving the continuous relaxation of problem (6)–(8).

LB_3 is the one obtained solving the continuous relaxation of problem (6)–(9), i.e., LB_2 plus constraint (9).

LB_4 is the lower bound obtained at the root node of our branch-and-cut algorithm based on problem (6)–(9). It is obtained by allowing CPLEX to strengthen LB_3 with fractional Gomory cuts by setting the CPLEX parameter `FracCut` to 2.

Tables 2, 3, and 4 show the bounds obtained on a set of instances with different graph size. The first two tables refer to instances with uniform costs and 5 colors, while the third table refers to random cost and k colors. For each lower bound, the tables report the gap with the optimum value and the computation time, where we define the *gap* from the optimum value (opt) as the ratio $\frac{opt-\lceil LB \rceil}{opt}$. We use the ceiling of the LB since the objective function takes only integer values. The best gaps are highlighted in bold. For a few instances, for which the optimum value is unknown, we report the best upper bound marked with an asterisk (*), and the gap is computed as $\frac{UB-\lceil LB \rceil}{UB}$, where UB is the best known upper bound.

The combinatorial lower bounds are extremely fast to compute, and they are in average better than those obtained by the standard linearization. However, the best lower bounds are obtained when CPLEX generates fractional Gomory cuts. In a number of instances, the lower bound LB_4 equals the optimum value.

Instance	n	m	opt	LB_1	gap	time	LB_2	gap	time	LB_3	gap	time	LB_4	gap	time
es20fst11	33	72	28	26	0.07	0.0	19.5	0.29	0.1	26.5	**0.04**	0.1	26.5	**0.04**	0.2
es20fst12	33	72	22	18	0.18	0.0	8.0	0.64	0.0	18.5	0.14	0.1	20.5	**0.05**	0.3
es20fst13	35	80	20	16	0.2	0.0	13.5	0.3	0.1	18.5	0.05	0.1	20.0	**0.0**	0.1
es20fst14	36	88	18	13	0.28	0.0	3.3	0.78	0.0	16.2	0.06	0.2	18.0	**0.0**	0.2
es20fst15	37	86	22	16	0.27	0.0	15.8	0.27	0.1	19.0	0.14	0.1	22.0	**0.0**	0.5
es30fst11	79	224	34	22	0.35	0.0	4.5	0.85	1.1	24.9	0.26	1.4	31.5	**0.06**	3.6
es30fst12	46	96	37	30	0.19	0.0	31.5	0.14	0.2	35.0	0.05	0.3	36.0	**0.03**	0.3
es30fst13	65	168	46	43	**0.07**	0.0	16.3	0.63	0.4	37.8	0.17	1.0	42.9	**0.07**	2.0
es30fst14	53	116	37	30	0.19	0.0	29.8	0.19	0.6	35.0	**0.05**	0.5	33.3	0.08	0.3
es30fst15	118	376	^{*57}	32	0.44	0.0	5.8	0.89	4.1	30.7	0.46	6.2	46.0	**0.19**	9.2
es40fst11	97	270	48	35	0.27	0.0	11.1	0.75	0.6	36.8	0.23	1.8	46.5	**0.02**	2.2
es40fst12	67	150	49	41	0.16	0.0	30.0	0.39	0.6	48.5	**0.0**	1.1	49.0	**0.0**	0.8
es40fst13	78	190	47	36	0.23	0.0	24.0	0.49	2.5	39.8	0.15	2.6	41.1	**0.11**	1.6
es40fst14	98	268	51	37	0.27	0.0	13.4	0.73	2.1	41.3	0.18	3.0	44.0	**0.14**	1.6
es40fst15	93	258	54	39	0.28	0.0	10.9	0.8	6.9	40.2	**0.24**	5.3	40.8	**0.24**	2.9

Table 2: Instances with uniform changeover costs and 5 colors: comparison of lower bounds.

Instance	n	m	opt	LB_1	gap	time	LB_2	gap	time	LB_3	gap	time	LB_4	gap	time
b01	50	126	28	18	0.36	0.0	18.8	0.32	0.3	25.5	0.07	0.5	27.3	**0.0**	0.7
b02	50	126	29	20	0.31	0.0	9.8	0.66	0.4	22.8	0.21	0.8	25.7	**0.1**	1.6
b03	50	126	28	15	0.46	0.0	11.3	0.57	0.8	22.8	0.18	0.6	26.8	**0.04**	1.8
b04	50	200	20	6	0.7	0.0	1.7	0.9	0.3	8.9	0.55	0.3	13.4	**0.3**	2.3
b05	50	200	13	5	0.62	0.0	3.0	0.69	0.1	6.7	0.46	0.3	9.0	**0.31**	1.9
b06	50	200	15	4	0.73	0.0	1.0	0.93	0.2	6.1	0.53	0.4	10.8	**0.27**	1.4
b07	75	188	45	31	0.31	0.0	20.8	0.53	0.9	37.4	0.16	1.7	43.5	**0.02**	1.5
b08	75	188	48	31	0.35	0.0	22.0	0.54	1.4	39.3	0.17	7.8	44.4	**0.06**	7.1
b09	75	188	42	29	0.31	0.0	21.1	0.48	2.2	34.8	0.17	1.5	39.9	**0.05**	4.9
b10	75	300	23	8	0.65	0.0	5.1	0.74	0.7	12.1	0.43	1.3	17.8	**0.22**	4.5
b11	75	300	*26	8	0.69	0.0	1.7	0.92	0.9	10.0	0.58	1.7	13.6	**0.46**	3.3
b12	75	300	*24	5	0.79	0.0	1.8	0.92	2.2	8.7	0.63	1.4	19.1	**0.17**	3.3
b13	100	250	52	31	0.4	0.0	29.1	0.42	2.3	42.8	0.17	11.8	50.7	**0.02**	16.6
b14	100	250	52	33	0.37	0.0	27.8	0.46	5.6	43.0	0.17	4.9	48.8	**0.06**	7.3
b15	100	250	57	36	0.37	0.0	24.6	0.56	0.9	46.9	0.18	5.6	53.8	**0.05**	15.5
b16	100	400	*26	6	0.77	0.0	2.5	0.88	35.0	10.2	0.58	16.2	16.8	**0.35**	15.3
b17	100	400	*31	7	0.77	0.0	2.4	0.9	1.8	10.5	0.65	0.9	18.4	**0.39**	5.0
b18	100	400	*33	7	0.79	0.0	3.4	0.88	1.9	11.4	0.64	3.4	18.8	**0.42**	6.3
i080-001	80	240	43	28	0.35	0.0	19.9	0.53	0.4	31.8	0.26	0.9	36.5	**0.14**	1.7
i080-002	80	240	39	20	0.49	0.0	11.5	0.69	0.4	27.2	0.28	0.7	36.8	**0.05**	1.6
i080-003	80	240	40	24	0.4	0.0	13.8	0.65	0.2	31.6	0.2	0.3	39.4	**0.0**	0.9
i080-004	80	240	42	22	0.48	0.0	17.5	0.57	0.3	34.0	0.19	0.4	38.0	**0.1**	1.4
i080-005	80	240	31	13	0.58	0.0	9.1	0.68	0.4	21.0	0.32	0.6	29.3	**0.03**	2.0
i160-001	160	480	*79	40	0.49	0.0	22.2	0.71	1.5	50.9	0.35	15.9	64.5	**0.18**	13.0
i160-002	160	480	75	40	0.47	0.0	18.9	0.75	1.7	52.7	0.29	2.2	68.4	**0.08**	6.3
i160-003	160	480	*81	46	0.43	0.0	26.7	0.67	2.3	57.5	0.28	3.5	71.1	**0.11**	4.9
i160-004	160	480	*70	32	0.54	0.0	15.3	0.77	2.7	46.9	0.33	6.2	60.1	**0.13**	11.3
i160-005	160	480	65	33	0.49	0.0	15.9	0.75	1.3	43.0	0.32	2.2	58.8	**0.09**	14.6

Table 3: Instances with uniform costs and 5 colors: comparison of lower bounds.

Instance	n	m	k	opt	LB_1	gap	time	LB_2	gap	time	LB_3	gap	time	LB_4	gap	time
es30fst11	79	224	2	30	12	0.6	0.0	7.0	0.77	1.8	13.8	0.53	2.8	25.4	**0.13**	4.3
			3	100	51	0.49	0.0	32.0	0.68	1.9	57.5	0.42	3.7	81.6	**0.18**	4.3
			5	189	107	0.43	0.0	30.2	0.84	1.6	129.1	0.31	1.6	158.6	**0.16**	3.9
			7	186	96	0.48	0.0	47.3	0.74	0.6	130.6	0.3	1.1	159.9	**0.14**	2.2
es30fst12	46	96	2	99	63	0.36	0.0	93.0	0.06	0.3	96.6	**0.02**	0.2	66.8	0.32	0.1
			3	115	71	0.38	0.0	108.0	0.06	0.2	114.0	0.01	0.3	115.0	**0.0**	0.2
			5	186	143	0.23	0.0	169.0	0.09	0.2	177.3	0.04	0.2	186.0	**0.0**	0.3
			7	188	132	0.3	0.0	170.2	0.09	0.3	180.0	0.04	0.2	188.0	**0.0**	0.2
es30fst13	65	168	2	39	13	0.67	0.0	23.0	0.41	0.5	27.7	0.28	1.2	28.3	**0.26**	1.9
			3	96	68	0.29	0.0	41.7	0.56	0.9	68.3	0.28	1.1	85.5	**0.1**	4.3
			5	235	202	0.14	0.0	94.7	0.6	0.9	184.1	0.21	0.7	216.0	**0.08**	1.1
			7	170	118	0.31	0.0	73.0	0.57	0.6	140.5	0.17	0.7	159.6	**0.06**	1.5
es30fst14	53	116	2	104	56	0.46	0.0	95.0	0.09	0.6	96.0	**0.08**	0.5	96.0	**0.08**	0.7
			3	165	105	0.36	0.0	150.0	0.09	0.7	156.0	0.05	0.6	165.0	**0.0**	0.6
			5	176	109	0.38	0.0	146.8	0.16	0.7	162.8	**0.07**	0.8	163.5	**0.07**	0.4
			7	202	155	0.23	0.0	180.7	0.1	0.4	201.8	**0.0**	0.4	202.0	**0.0**	0.5
es30fst15	118	376	2	57	9	0.84	0.0	22.3	0.6	6.2	26.2	0.53	10.9	40.7	**0.28**	30.8
			3	**103*	36	0.65	0.0	22.3	0.78	4.3	46.7	0.54	4.4	75.8	**0.26**	11.4
			5	**235*	111	0.53	0.0	35.1	0.85	1.6	105.2	0.55	4.8	158.4	**0.32**	11.3
			7	**284*	135	0.52	0.0	43.0	0.85	9.8	150.8	0.47	2.0	222.3	**0.21**	7.3

Table 4: Instances with random costs and k colors: comparison of lower bounds.

5.3. Implementation Details

Our improved branch-and-cut algorithm is based on problem (6)–(9), and embeds the combinatorial upper bound and the greedy heuristic. The separation procedure for constraints (3) is implemented exploiting the CPLEX callbacks. Constraints (3) are separated both by using the fractional LP relaxation and by using the integer solutions found by the solver primal heuristics (that might be infeasible). In the first case, the inequalities are added as user cuts that strengthen the lower bounds (see `UserCutCallback` in the CPLEX manuals). In the second case, they are added as model constraints (see `LazyConstraintCallback` in the CPLEX manuals). The fractional Gomory cuts are automatically separated by CPLEX; we just set the parameter `FracCut` to 2, which forces the solver to generate a large number of such inequalities. The best solution found between the combinatorial and the greedy heuristic is passed to the solver as initial integer solution. The branching phase is handled by the standard CPLEX procedures.

Similarly, we solve by branch-and-cut the convex reformulation of MinCCA given by problem (11),(3)–(5), (7)–(9). We separate inequalities (3) only when needed by adding them as model constraints (see `LazyConstraintCallback` in the CPLEX manuals). Note that in this case the continuous relaxation, which is a convex program, is solved by CPLEX using its interior point algorithm. The quadratic solver of CPLEX is a branch-and-cut that "convexifies" the quadratic matrix Q with an approach similar to the one introduced in [4] and then uses its continuous relaxation to get valid lower bounds. Unfortunately, this convexification process is included in the preprocessing, and the preprocessing has to be deactivated in order to be able to use the `LazyConstraintCallback`. This is the reason why we used the engine value method to reformulated MinCCA into a convex problem, as presented in Section 4.2.

In the SCIL implementation, for the quadratic terms of the objective function we chose the standard linearization plus the separation of cycle-constraints of the cut polytope. In the root node we applied an exact separation routine, while in all other nodes of the branch-and-bound tree a heuristic method was used. Quadratic reformulation of linear constrains are disabled.

5.4. Comparison of Exact Solution Approaches

Tables 5 and 6 compare the results obtained with (i) an exact solution approach based on the convex reformulation solved with CPLEX, (ii) the implementation based on the symbolic constraints solved with SCIL, and (iii) our branch-and-cut algorithm that is an improved version of the implementation published in [13]. Indeed, they are three different branch-and-cut implementations. Table 5 refers to random instances with k colors, and Table 6 to uniform instances with 5 colors.

Each table is divided vertically into four blocks. In the first block we report the name of the instance, together with its size, in terms of number of nodes n and of arcs m. The second block provides the best upper bound (UB_{CP}) and lower bound (LB_{CP}) and the solution time (*time*) obtained by CPLEX when solving the convex formulation presented in Section 4.2. The third block contains the same information, but relative to SCIL. Finally, the last block provides the results obtained with our method; we report the best upper bound (UB) and lower bound (LB) obtained within the time limit of 3600 seconds, the open gap (*gap*, computed as in Section 5.2), the time needed for solving the instances

Instance	n	m	k	LB_{CP}	UB_{CP}	$time_{CP}$	LB_{SC}	UB_{SC}	$time_{SC}$	LB	UB	gap	$time$	$cuts$	$nodes$
es30fst11	79	224	2	-174.7	30	3600	18.5	44	3600	30.0	30	0	4.8	242	75
			3	-316.3	100	3600	60.0	172	3600	100.0	100	0	7.8	415	954
			5	-376.2	192	3600	89.2	361	3600	189.0	189	0	7.9	317	1341
			7	-438.9	186	3600	101.1	298	3600	186.0	186	0	4.5	240	561
es30fst12	46	96	2	99.0	99	3600	99.0	99	0.46	99.0	99	0	0.1	34	3
			3	115.0	115	0.19	115.0	115	0.63	115.0	115	0	0.2	41	0
			5	186.0	186	0.19	186.0	186	0.84	186.0	186	0	0.3	44	2
			7	188.0	188	0.13	188.0	188	0.79	188.0	188	0	0.2	38	0
es30fst13	65	168	2	-52.7	39	3600	37.0	44	3600	39.0	39	0	2.5	263	140
			3	-64.9	96	3600	81.3	168	3600	96.0	96	0	4.7	292	55
			5	1.7	235	3600	192.9	335	3600	235.0	235	0	1.6	151	107
			7	-46.6	170	3600	138.2	289	3600	170.0	170	0	2.1	179	162
es30fst14	53	116	2	104.0	104	0.66	104.0	104	1.43	104.0	104	0	0.7	85	3
			3	165.0	165	0.43	165.0	165	1.84	165.0	165	0	0.6	79	0
			5	176.0	176	0.58	176.0	176	3.54	176.0	176	0	0.4	70	7
			7	202.0	202	0.53	202.0	202	3.33	202.0	202	0	0.4	54	0
es30fst15	118	376	2	-817.5	64	3600	15.3	120	3600	57.0	57	0	484	7725	8849
			3	-1281.8	103	3600	22.3	266	3600	98.0	103	0.05	3600	12569	76727
			5	-1208.5	235	3600	37.0	486	3600	193.8	235	0.17	3600	14079	58834
			7	-1306.7	295	3600	50.3	505	3600	263.4	284	0.07	3600	7051	170679

Table 5: Instances with random costs and k colors: results of the three branch-and-cut algorithms.

(*time*), the number of added inequalities of type (3) (*cuts*), and the number of processed nodes of the branch-and-cut tree (*nodes*).

It is evident from the tables that our method is far better than the other two benchmarks proposed. We are able to solve 37 out of 48 instances to optimality, against the 7 solved by CPLEX and the 9 solved by SCIL. Moreover, among the 11 instances where all three methods go in time limit, our method always provides the best lower bound and provides the best upper bound in 9 cases (for instances `b11` and `i160-001` the best upper bound is provided by CPLEX).

Finally, we would like to stress the fact that in our previous work [13] none of the `bXX` and `i080-XX` instances were solved to optimality. This big improvement is definitely due to the more accurate strategy used in the selection of the valid inequalities to consider.

Instance	n	m	LB_{CP}	UB_{CP}	$time_{CP}$	LB_{SC}	UB_{SC}	$time_{SC}$	LB	UB	gap	$time$	cuts	nodes
b01	50	126	26.0	28	3600	28.0	28	368.9	28	28	0	0.8	58	10
b02	50	126	9.4	29	3600	23.3	36	3600	29	29	0	2	170	121
b03	50	126	19.7	28	3600	23.0	35	3600	28	28	0	2	133	16
b04	50	200	-87.9	20	3600	4.3	33	3600	20	20	0	1701	5765	71760
b05	50	200	-79.3	13	3600	4.2	27	3600	13	13	0	20.5	1145	3340
b06	50	200	-94.1	15	3600	2.7	27	3600	15	15	0	6.8	493	1398
b07	75	188	-18.6	45	3600	32.2	58	3600	45	45	0	1.6	72	15
b08	75	188	-13.5	48	3600	32.8	58	3600	48	48	0	8.8	470	266
b09	75	188	-4.1	42	3600	31.4	56	3600	42	42	0	5	236	17
b10	75	300	-180.5	24	3600	8.3	48	3600	23	23	0	2109	11439	17504
b11	75	300	-167.5	25	3600	4.2	48	3600	17.4	26	0.31	3600	17849	22600
b12	75	300	-187.8	25	3600	3.7	50	3600	22.3	24	0.04	3600	16841	37181
b13	100	250	-52.1	52	3600	38.7	68	3600	52	52	0	17.3	388	25
b14	100	250	-38.6	52	3600	37.9	76	3600	52	52	0	10.1	436	542
b15	100	250	-39.6	57	3600	35.5	74	3600	57	57	0	17.6	630	141
b16	100	400	-264.8	26	3600	3.8	63	3600	20.6	26	0.19	3600	15342	44400
b17	100	400	-287.0	31	3600	3.2	55	3600	23	31	0.26	3600	13096	59505
b18	100	400	-250.0	33	3600	4.3	51	3600	22.6	33	0.3	3600	17805	19100
i080-001	80	240	-60.9	43	3600	23.4	59	3600	43	43	0	22	367	12468
i080-002	80	240	-87.5	39	3600	17.5	48	3600	39	39	0	2.5	117	237
i080-003	80	240	-90.5	40	3600	17.5	47	3600	40	40	0	1	41	15
i080-004	80	240	-77.6	42	3600	20.3	57	3600	42	42	0	2.7	88	752
i080-005	80	240	-74.2	31	3600	16.5	52	3600	31	31	0	5.8	460	790
i160-001	160	480	-267.5	78	3600	26.3	107	3600	69.5	79	0.11	3600	18220	23163
i160-002	160	480	-218.9	75	3600	20.7	111	3600	75	75	0	2126	4669	131187
i160-003	160	480	-262.1	81	3600	27.1	108	3600	80.6	81	0	3600	1469	618589
i160-004	160	480	-209.8	72	3600	19.1	100	3600	66.7	70	0.04	3600	11052	96016
i160-005	160	480	-269.3	65	3600	20.4	105	3600	65	65	0	122	1571	9416

Table 6: Instances with uniform costs and 5 colors: results of the three branch-and-cut algorithms.

Instance	n	m	UB_0	LB_0	LB	UB	gap	time	cuts	nodes
c01	500	1250	380	288.5	288.5	380	0.24	3600	4273	0
c02	500	1250	357	203.2	203.2	357	0.43	3600	4706	0
c03	500	1250	378	252.7	252.7	378	0.33	3600	4568	0
c04	500	1250	341	264.3	264.3	341	0.22	3600	5972	130
c05	500	1250	365	207.2	207.2	365	0.43	3600	3696	0
c06	500	2000	270	106.5	107.5	270	0.6	3600	10105	416
c07	500	2000	262	95.6	96.6	262	0.63	3600	11946	40
c08	500	2000	289	64.8	64.8	289	0.78	3600	2850	0
c09	500	2000	258	102.0	106.0	184	0.42	3600	7959	13202
c10	500	2000	276	96.8	97.5	209	0.53	3600	9950	1552

Table 7: Challenging instances: results of our improved branch-and-cut algorithm.

5.5. Challenging instances

We present in Table 7 the results obtained with our improved branch-and-cut algorithm on a set of instances that we consider very demanding for MinCCA. These instances are obtained from the `C` data set of the SteinerLib. Though these instances are easy for Steiner tree problems, they are quite demanding for the MinCCA problem. Note that for 5 instances out of 10, after one hour the solver is still trying to solve the LP relaxation of the problem. For a single instance, namely, `c09`, our branch-and-cut algorithm is able to improve the initial upper bound. Hence, we propose these as "new" challenging instances of the MinCCA problem.

6. Conclusions

In this paper, we have proposed a new arborescence problem with a quadratic cost function that depends on so-called *changeover* costs, and we have shown that this new problem is indeed challenging both theoretically and computationally. Theoretically, we have proved that it is NPO-complete and very hard to approximate, even in the very restrictive case of digraphs with bounded degree, costs satisfying the triangle inequality, and arcs colored only with two colors. Computationally, we have shown that in practice the problem is very hard to solve to optimality with state-of-the-art commercial optimization software, even on small graphs with only hundreds of nodes.

Acknowledgements

The first two authors first approached the subject of the present paper together with Francesco Maffioli. His premature departure deprived them of his support and enthusiasm, which will be certainly remembered by all those who had the good fortune of knowing him.

[1] E. Althaus, A. Bockmayr, M. Elf, M. Jnger, T. Kasper, and K. Mehlhorn. SCIL – Symbolic Constraints in Integer Linear Programming. In *Proc of European Symposium on Algorithms*, volume LNCS 2461, pages 76–87, 2002.

[2] E. Amaldi, G. Galbiati, and F. Maffioli. On minimum reload cost paths, tours, and flows. *Networks*, 57(3):254–260, 2011.

[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 1999.

[4] A. Billionnet, S. Elloumi, and M. Plateau. Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method. *Discrete Applied Mathematics*, 157(6):1185–1197, 2009.

[5] C. Buchheim, F. Liers, and M. Oswald. Speeding up IP-based algorithms for constrained quadratic 0–1 optimization. *Mathematical Programming (B)*, 124(1-2):513–535, 2010.

[6] A. Caprara. Constrained 0–1 quadratic programming: Basic approaches and extensions. *European Journal of Operational Research*, 187:1494–1503, 2008.

[7] S. Chopra, E.R. Gorres, and M.R. Rao. Solving the Steiner tree problem on a graph using branch and cut. *ORSA Journal on Computing*, 4(3):320–335, 1992.

[8] J. Edmonds. Optimum branchings. *J. Res. Nat. Bur. Stand. (B)*, 71:233–240, 1967.

[9] Uriel Feige. private communication.

[10] M. Fischetti. Facets of two steiner arborescence polyhedra. *Mathematical Progrogramming (A)*, 51(3):401–419, 1991.

[11] H.N. Gabow, Z. Galil, T. Spencer, and R.E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122, 1986.

[12] G. Galbiati. The complexity of a minimum reload cost diameter problem. *Discrete Applied Mathematics*, 156(18):3494–3497, 2008.

[13] G. Galbiati, S. Gualandi, and F. Maffioli. On the minimum changeover cost arborescences. In *Proc of International Symposium on Experimental Algorithms, LNCS 6630*, pages 112–123, 2011.

[14] I. Gamvros, L. Gouveia, and S. Raghavan. Reload cost trees and network design. In *Proc. International Network Optimization Conference*, page paper n.117, 2007.

[15] I. Gamvros, L. Gouveia, and S. Raghavan. Reload cost trees and network design. *Networks*, 59(4):365–379, 2012.

[16] L. Gourvès, A. Lyra, C. Martinhon, and J. Monnot. The minimum reload s–t path, trail and walk problems. *Discrete Applied Mathematics*, 158(13):1404–1417, 2010.

[17] P.L. Hammer and A.A. Rubin. Some remarks on quadratic programming with 0-1 variables. *RAIRO*, 3:67–79, 1970.

[18] J. Hao and J.B. Orlin. A faster algorithm for finding the minimum cut in a graph. In *Proc of the 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 165–174, 1992.

[19] C. Helmberg, F. Rendl, and R. Weismantel. A semidefinite programming approach to the quadratic knapsack problem. *Journal of Combinatorial Optimization*, 4(2):197–215, 2000.

[20] P. Jonsson. Near-optimal nonapproximability results for some NPO PB-complete problems. *Inform. Process. Lett.*, 68:249–253, 1997.

[21] V. Kann. Polynomially bounded minimization problems that are hard to approximate. *Nordic J. Comp.*, 1:317–331, 1994.

[22] E.L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.

[23] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*. Springer, 2003.

[24] H. Wirth and J. Steffan. Reload cost problems: minimum diameter spaning tree. *Discrete Applied Mathematics*, 113:73–85, 2001.