Theoretical Computer Science 540–541 (2014) 40–52

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On the complexity of constructing minimum changeover cost arborescences ☆

Didem Gözüpek [a,*], Mordechai Shalom [b], Ariella Voloshin [c], Shmuel Zaks [c]

[a] Department of Computer Engineering, Gebze Institute of Technology, Kocaeli, Turkey
[b] TelHai Academic College, Upper Galilee, 12210, Israel
[c] Department of Computer Science, Technion, Haifa, Israel

ARTICLE INFO

Article history:
Received 28 September 2012
Received in revised form 23 July 2013
Accepted 24 March 2014

Keywords:
Reload cost
Changeover cost
Approximation algorithms
Network design
Network optimization

ABSTRACT

The reload cost concept refers to the cost that occurs at a vertex along a path on an edge-colored graph when it traverses an internal vertex between two edges of different colors. This cost depends only on the colors of the traversed edges. Reload costs arise in various applications such as transportation networks, energy distribution networks, and telecommunications. Previous work on reload costs focuses on two problems of finding a spanning tree with minimum cost with respect to two different cost measures. In both problems the cost is associated with a set of paths from a given vertex r to all the leaves of the constructed tree. The first cost measure is the sum of the reload costs of all paths from r to the leaves. The second cost measure is the changeover cost, in which the cost of traversing a vertex by using two specific incident edges is paid only once regardless of the number of paths traversing it. The first problem is inapproximable within any polynomial time computable function of the input size [1], and the second problem is inapproximable within $n^{1-\epsilon}$ for any $\epsilon > 0$ [2]. In this paper we show that the first hardness result holds also for the second problem. Given this strong inapproximability result, we study the complexity and approximability properties of numerous special cases of this second problem. We mainly focus on bounded costs, and consider both directed and undirected graphs, bounded and unbounded number of colors, and both bounded and unbounded degree graphs. We also present polynomial time exact algorithms and an approximation algorithm for some special case. To the best of our knowledge, these are the first algorithms with a provable performance guarantee for the problem. Moreover, our approximation algorithm shows a tight bound on the approximability of the problem for a specific family of instances.

☆ This work is supported in part by the Israel Science Foundation (ISF) under grant no. 1249/08, the Scientific and Technological Research Council of Turkey (TUBITAK) under grant no. 113E567 and by the Scientific Research Projects Office of Gebze Institute of Technology (GYTE BAP) under grant no. 2013 A24. Part of the work of Shmuel Zaks was done while visiting Universidad Carlos III de Madrid (UC3M), Departmento de Ingenieria Telematica (supported by a Catedra de Excelencia), and IMDEA Networks, Madrid, Spain.

* Corresponding author.
E-mail addresses: didem.gozupek@gyte.edu.tr (D. Gözüpek), cmshalom@telhai.ac.il (M. Shalom), variella@cs.technion.ac.il (A. Voloshin), zaks@cs.technion.ac.il (S. Zaks).

http://dx.doi.org/10.1016/j.tcs.2014.03.023
0304-3975/

1. Introduction

1.1. Background

Various network optimization problems can be modeled using edge-colored graphs. In these graphs we focus on the *reload costs*, which are associated with the cost incurred while traversing a vertex via two consecutive edges of different colors. The cost associated with the traversal of a vertex depends on the colors of the incident traversed edges. Two models are under concern, the *reload cost* model and the *changeover cost* model. The difference between the two models is that in the reload cost model, the cost is proportional to the amount of commodity flowing through the arcs, whereas in the changeover cost model the cost is fixed. Although these costs have important applications in many areas such as transportation networks, energy distribution networks and telecommunications, they have received little attention in the literature.

Each carrier in an intermodal cargo transportation network can be represented by a color. The cost of transferring cargo from one carrier to another can be modeled by reload costs. In energy distribution networks, transfer of energy from one type of carrier to another has a reload cost. In telecommunications, reload costs arise in many different settings. In heterogeneous networks, routing may necessitate switching from one technology (such as a cellular network) to another (such as a wireless local area network) and this switching cost can be modeled by using the reload cost or changeover cost concept, depending on the specific setting. Even within the same technology, switching between networks of different service providers has different costs corresponding to reload costs [1,3]. Recently, *dynamic spectrum access (DSA)* networks have been studied in the wireless networking literature [4,5]. Assigned frequency bands in an ad hoc DSA network can be significantly far away from one another. Hence, unlike other wireless networks, switching from one frequency band to another in a DSA network can have a significant cost in terms of delay and power consumption [6]. We observe that this frequency switching cost can be modeled using changeover costs, and in this work we investigate this model.

1.2. Related work

The reload cost concept is introduced in [3] where the focus is on the problem of finding a spanning tree in an edge colored undirected graph having minimum diameter with respect to reload costs. It is proven that, in its most general case, the problem is hard to approximate within any polynomial-time computable function $f(n)$, and is hard to approximate within any constant factor less than 3 even when the instance is restricted to graphs of maximum degree 5. A polynomial time algorithm for graphs with maximum degree at most 3 is also provided. In [7] the authors study the minimum diameter reload cost spanning tree problem for graphs with maximum degree 4. In particular, it is proved that if reload costs are unrestricted, the problem cannot be approximated within any constant $c < 2$, and it cannot be approximated within any constant $c < 5/3$ if reload costs satisfy the triangle inequality.

[8] focuses on the minimum reload cost cycle cover problem, aiming to span the vertices of an edge-colored graph by a set of vertex-disjoint cycles.

[9] studies the problems of finding a path, trail or walk connecting two given vertices s, t with minimum total reload cost. It focuses on cases where reload costs are symmetric/asymmetric and do/do not satisfy the triangle inequality. The work in [9] provides a rather complete characterization of polynomial vs. NP-Hard cases. The case of walks is shown to be polynomial as was already pointed out in [3]. Actually this fact is re-proved later, this time for digraphs, in [1].

In all the above problems, the concept of reload cost is closely related to the concept of changeover cost, as will be made more precise in Section 2.

In [10] integer programming formulation is used to find a spanning tree that minimizes the sum of the reload costs of all paths between all pairs of vertices.

Works [1,2] are more closely related to our work. [1] presents numerous path, tour, and flow problems concerning reload costs. In particular, it focuses on the *Minimum Reload Cost Path-Tree* (MinRCPT) problem, which is to find a spanning tree that minimizes the total reload cost from a source vertex to all other vertices. It is shown that, in a directed graph, the problem is inapproximable within any polynomial-time computable function.

In [2] the authors study the *Minimum Changeover Cost Arborescence* problem, which is to find a spanning tree that minimizes the sum of the costs of traversing all pairs of edges. This sum does not depend on the amount of commodity traversing each pair. The authors studied directed graphs and proved that unless $P = NP$, even on graphs with bounded degree and reload costs satisfying the triangle inequality, the problem is not approximable within a) $\beta \log\log n$, for some $\beta > 0$, when the number of colors is 2, b) $n^{\frac{1}{3}-\epsilon}$, for any $\epsilon > 0$ when the number of colors is 3.

1.3. Our contribution

In this work we study the *Minimum Changeover Cost Arborescence* problem, which is to find a spanning tree that minimizes the sum of the costs of traversing paths from r to all nodes. This sum does not depend on the amount of commodity traversing each pair. We refer to the *Minimum Changeover Cost Arborescence* problem on undirected and directed graphs as MinCCA and MinDirectedCCA, respectively. We first observe that the hardness result in [1] obtained for the *Minimum Reload Cost Path-Tree Problem on Directed Graphs* (MinDirectedRCPT) holds also for our problem; i.e., when no restrictions are imposed on the individual costs, MinDirectedCCA is inapproximable within any polynomial-time computable function $f(n)$

Table 1
Summary of results.

Directed graphs	Bounded degree	APX-Hard for 2 colors
	Unbounded degree	log-APX-Hard for 2 colors log-APX-Complete for directed acyclic graphs without crossing free traversals
Undirected graphs	Bounded degree	APX-Hard for 2 colors
	Unbounded degree	log-APX-Hard Polynomial time solvable if all biconnected components are monochromatic Polynomial time solvable for the special case of cactus graphs where the number of cycles that each edge can belong to is bounded by a constant

where n is the size of the input. Given this strong inapproximability result, a natural research direction is to investigate special cases, especially bounded reload cost values. We explore the complexity of MINCCA and MINDIRECTEDCCA under bounded reload costs and depending on parameters such as bounded vs. unbounded vertex degrees and bounded vs. unbounded number of colors. We also present polynomial time exact algorithms and an approximation algorithm for some special cases. The approximation algorithm shows a tight bound on the approximability of the problem for a specific family of instances. Table 1 summarizes these results.

In Section 2 we introduce some preliminaries including notation, definitions and problem statements. Section 3 presents our inapproximability results for directed graphs, while Section 4 focuses on hardness results for undirected graphs. In Section 5 we present our algorithms for special cases. Finally, in Section 6 we discuss further research directions.

2. Preliminaries

Notations and problem statements: Given an undirected graph $G = (V(G), E(G))$ and a vertex $v \in V(G)$, $\delta_G(v)$ denotes the set of edges incident to v in G, and $d_G(v) \stackrel{def}{=} |\delta_G(v)|$ is the degree of v in G. The minimum and maximum degrees of G are defined as $\delta(G) \stackrel{def}{=} \min\{d_G(v) \mid v \in V(G)\}$ and $\Delta(G) \stackrel{def}{=} \max\{d_G(v) \mid v \in V(G)\}$ respectively. When there is no ambiguity about the graph G we omit it from the notation and write $V, E, \delta(v), d(v), \delta$ and Δ, respectively. Given a subset $\bar{V}$ of $V(G)$, $G[\bar{V}]$ denotes the subgraph of G induced by $\bar{V}$. Given a tree T and two vertices $v_1, v_2 \in V(T)$, we denote by $P_T(v_1, v_2)$ the path between v_1 and v_2 in T. We use the same notation for digraphs wherever applicable. The degree of a vertex in a digraph is its degree in the underlying graph. A digraph is *directed acyclic* (also known as DAG) if it does not contain directed circuits. A digraph is a *rooted tree* or *arborescence* if its underlying graph is a tree and it contains a *root* vertex, such that there is a directed path from it to every other vertex. Note that a rooted tree is a DAG. Given two graphs G and G', their union is $G \cup G' \stackrel{def}{=} (V(G) \cup V(G'), E(G) \cup E(G'))$. For a set A and an element x, we use $A + x$ (resp. $A - x$) as a shorthand for $A \cup \{x\}$ (resp. $A \setminus \{x\}$).

We follow the notation of [3] where the concept of reload cost was defined. We consider edge colored graphs G, where the colors are taken from a set X and $\chi : E(G) \to X$ is the *coloring function*. Given a coloring function χ, we denote by E_x^{χ}, or simply by E_x the set of edges of E colored x, and $G_x = (V(G), E(G)_x)$ is the subgraph of G having the same vertex set as G, but only the edges colored x.

The costs are given by a non-negative function $c : X^2 \to \mathbb{R}^+$ satisfying

- (i) $c(x_1, x_2) = c(x_2, x_1)$ for every $x_1, x_2 \in X$.
- (ii) $c(x, x) = 0$ for every $x \in X$.

We assume w.l.o.g. that the minimum possible non-zero reload cost value is 1. We denote the maximum cost value as $C_{max} \stackrel{def}{=} \max_{x_1, x_2 \in X} c(x_1, x_2)$. The cost of traversing two adjacent edges e_1, e_2 is $c(e_1, e_2) \stackrel{def}{=} c(\chi(e_1), \chi(e_2))$. We say that the instance satisfies the *triangle inequality* if

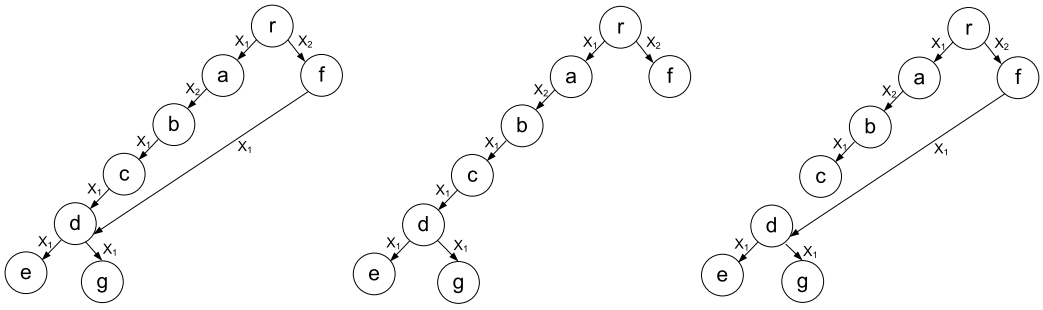
- (iii) $c(e_1, e_3) \leqslant c(e_1, e_2) + c(e_2, e_3)$ whenever e_1, e_2 and e_3 are incident to the same vertex.

Depending on the problem, the cost function will be interpreted as a reload cost or changeover cost.

The *changeover cost* of a path $P = e_1, e_2, \ldots, e_\ell$ of length ℓ is $c(P) \stackrel{def}{=} \sum_{i=2}^{\ell} c(e_{i-1}, e_i)$. Note that $c(P) = 0$ whenever $\ell \leqslant 1$.

We extend this definition to trees as follows: Given a directed tree T rooted at r, (resp. an undirected tree T and a vertex $r \in V(T)$), for every edge e going out of r (resp. incident to r) we define $prev(e) = e$, and for every other edge, $prev(e)$ is the edge before e on the path from r to e. The changeover cost of T with respect to r is $c(T, r) \stackrel{def}{=} \sum_{e \in E(T)} c(prev(e), e)$.

A pair of edges (e_1, e_2) incident to the same vertex v is a *free traversal* of v, if $c(\chi(e_1), \chi(e_2)) = 0$. In digraphs, we use the same terminology when e_1 is an incoming edge of v and e_2 is an outgoing edge of v. Two such free traversals (e_1, e_2) and (e_1', e_2') are *crossing* if they both traverse the same vertex v and $\{e_1, e_2\} \cap \{e_1', e_2'\} = \emptyset$.



(a) $G = (V, E)$ (b) T_1 (c) T_2

Fig. 1. Example graphs.

MinCCA and MinDirectedCCA problems aim to find a spanning tree rooted at r with minimum changeover cost [2]. Formally,

MinCCA (G, X, χ, r, c)
Input: A graph $G = (V, E)$ with an edge coloring function $\chi : E \to X$, a vertex $r \in V$ and a reload cost function $c : X^2 \to \mathbb{R}^+$.
Output: A spanning tree T of G.
Objective: Minimize $c(T, r)$.

MinDirectedCCA(G, X, χ, r, c)
Input: A digraph $G = (V, E)$ with an edge coloring function $\chi : E \to X$, a vertex $r \in V$ and a reload cost function $c : X^2 \to \mathbb{R}^+$.
Output: A directed spanning tree T of G rooted at r.
Objective: Minimize $c(T, r)$.

Both problems have feasible solutions whenever all the vertices of G are reachable from r. As this can be verified easily in polynomial time using any traversal algorithm such as DFS or BFS, we assume without loss of generality that all instances are feasible. Moreover, we assume that every solution has a cost strictly greater than zero, as otherwise an optimal solution can be found in polynomial time, for instance, by traversing the graph G_x for every color x.

Another cost measure for a rooted tree is the reload cost, defined as $c_{RL}(T, r) \stackrel{def}{=} \sum_{v \in V(T)} c(P_T(r, v))$. Problem MinDirectedRCPT is the counterpart of MinDirectedCCA, where the objective function is replaced by $c_{RL}(T, r)$ [1]. When there is no ambiguity about the vertex r, we denote $c_{RL}(T, r)$ and $c(T, r)$ by $c_{RL}(T)$ and $c(T)$, respectively.

Example 1. Consider the digraph $G = (V, E)$ in Fig. 1(a) with $X = \{x_1, x_2\}$ and the cost function c with $c(x_1, x_2) = c(x_2, x_1) = 1$. The changeover cost of the spanning tree T_1 (Fig. 1(b)) is $c(T_1, r) = 2$, whereas its reload cost is $c_{RL}(T_1, r) = 9$. On the other hand, the changeover cost of the spanning tree T_2 of G (Fig. 1(c)) is $c(T_2, r) = 3$, whereas its reload cost is $c_{RL}(T_2, r) = 6$. Actually the tree T_1 (resp. T_2) is an optimum for problem MinDirectedCCA (resp. MinDirectedRCPT), showing that the two problems may lead to different optima.

Approximation algorithms: Let Π be a minimization problem and $\rho \geqslant 1$. A (feasible) solution S of an instance I of Π is a *ρ-approximation* if its objective function value $O_\Pi(I, S)$ is at most ρ times the optimal objective function value $O^*_\Pi(I)$. An algorithm *ALG* is said to be a *ρ-approximation algorithm* for an optimization problem Π if *ALG* returns a ρ-approximation $ALG(I)$ for every instance I of Π. A problem Π is said to be *ρ-approximable* if there is a polynomial-time ρ-approximation algorithm for it. Π is said to be *ρ-inapproximable* if there is no polynomial-time ρ-approximation algorithm for it unless $P = NP$. Given a real function f, Π is said to be in f-Apx-Hard if there is a constant $c > 0$ such that Π is $(c \cdot f(|I|))$-inapproximable where $|I|$ is the size of the instance I, and Π is said to be in f-Apx if there is a constant $c > 0$ such that Π is $(c \cdot f(|I|))$-approximable. If Π is both in f-Apx-Hard and f-Apx, then it is said to be in f-Apx-Complete. When f is a constant, these complexity classes are called simply Apx-Hard, Apx and Apx-Complete, respectively. Polynomial reductions are the main tools using to prove inapproximability results. An *approximation preserving reduction* of a problem Π to a problem Π', is a polynomial reduction that shows that a ρ-approximation to problem π' implies a ρ-approximation to problem π. A *polynomial time approximation scheme* (PTAS) is an infinite family of algorithms $\{ALG_\epsilon \mid \epsilon > 0\}$ such that ALG_ϵ is a $(1 + \epsilon)$-approximation algorithm with running time $O(|I|^{f(\epsilon)})$ for some function f. A problem Π is said to be in Ptas if there is a PTAS for it [11]. We note that, unless $P = NP$, a problem is in Apx-Hard if and only if it does not admit a PTAS.

The minimum set cover problem: An instance of the MinWeightSetCover problem is a triplet $(U, \mathcal{S}, w)$, where $U = \{u_1, u_2, \ldots, u_n\}$ is a finite ground set of elements, $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is a collection of subsets of U, and $w : \mathcal{S} \to \mathbb{R}$ is a non-negative real *weight function* on the sets. Given such an instance, one has to find a subset $\mathcal{C} \subseteq \mathcal{S}$ that covers U, i.e. $\bigcup \mathcal{C} \stackrel{def}{=} \bigcup_{S_i \in \mathcal{C}} S_i = U$. The goal is to minimize $w(\mathcal{C}) \stackrel{def}{=} \sum_{S_i \in \mathcal{C}} w(S_i)$. The special case in which all weights are 1 (thus $w(\mathcal{C}) = |\mathcal{C}|$) is referred to as the MinCardSetCover problem and we denote its instances as $(U, \mathcal{S})$. The greedy algorithm of [12] is an H_n-approximation to both problems, where $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the n-th harmonic number. It is also shown in [13] that it is hard to approximate MinCardSetCover within a factor $o(\log n)$.

Biconnected components and block trees: An *articulation point*, a *cut vertex* or a *separation vertex* in a connected graph is a vertex whose removal, along with its incident edges, disconnects the remaining vertices. A *bridge* in a connected graph is an edge whose removal disconnects the graph. A graph with no articulation points is called *biconnected* or *nonseparable*. A maximal biconnected subgraph of a graph is called a *biconnected component* or a *block* [14,15]. For any connected graph G we can construct in linear time a tree whose vertices are the biconnected components of G and its articulation points, and the edges join every articulation point to the blocks to which it belongs. This tree is termed the block tree or superstructure of the graph [16,17].

Cactus graphs: A *cactus graph* is a connected graph in which any two simple cycles have at most one vertex in common. Every edge in a cactus graph belongs to at most one simple cycle. Furthermore, every block in a cactus graph is either an edge or a cycle [18].

3. Lower bounds for directed acyclic graphs

In this section we provide hardness results for digraphs. We first observe that MinDirectedCCA is inapproximable within any polynomial time computable function $f(n)$, even if G is a directed acyclic graph-Dag. This hardness result relies heavily on unbounded reload cost values. In view of this result, in the subsequent subsections we focus on bounded reload costs and the case where G is a Dag. We consider the simplest case of two colors with all non-zero reload costs equal to 1, i.e. $|X| = 2$ and $C_{max} = 1$. For these instances we show a lower bound of $\Omega(\log |V|)$ when the degrees can be arbitrarily large, and that the problem is Apx-Hard even when the degrees are bounded. In Section 5.2 we provide an algorithm matching the bound for the unbounded degree case, showing that the bound is tight.

3.1. Unbounded costs

We first observe that the gap proven for MinDirectedRCPT in Theorem 2 of [1] is valid also for the MinDirectedCCA problem, as follows:

Proposition 1. MinDirectedCCA *cannot be approximated within any polynomial time computable function* $f(n)$ *even for instances where* G *is a* Dag.

Proof. We note that for every tree T and root $r \in V(T)$, we have $c(T, r) \leqslant c_{RL}(T, r)$. This is because for every edge e, $(prev(e), e)$ is included in some path $P_T(r, v)$. Moreover, equality holds if T is such that $c(prev(e), e) > 0$ only if e enters a leaf of T.

The rest of the proof is closely related to the proof of Theorem 2 in [1]. (The reader is advised to consult that work for further details.) We observe the following facts about the trees used in the proof:

- If the Set Cover instance is a YES instance: Any optimal tree T_{YES} satisfies $c(T_{YES}, r) = c_{RL}(T_{YES}, r)$ because whenever $c(e) > 0$, e enters a leaf of T_{YES}.
- If the Set Cover instance is a NO instance: It is said that any solution T_{NO} satisfies $c_{RC}(T_{NO}) \geqslant c - h + M$. We observe that $c(T_{NO}) \geqslant c - h + M$ holds, too.

Therefore, the gap proven for MinDirectedRCPT in [1] is valid also for the MinDirectedCCA problem. □

3.2. Bounded costs, unbounded vertex degrees

Theorem 1. *There is a constant* $\rho > 0$, *such that* MinDirectedCCA *is* $(\rho \cdot \log |V|)$*-inapproximable even when* G *is a* Dag *without crossing free traversals,* $|X| = 2$, *and* $C_{max} = 1$.

Proof. We provide an approximation preserving reduction from MinCardSetCover to MinDirectedCCA. It is known (see [19]) that there is a constant $\rho' > 0$ such that MinCardSetCover is $(\rho' \cdot \log |U|)$-inapproximable even if we restrict the problem to instances in which the number of sets is polynomial in the number of elements,[1] i.e. $|\mathcal{S}| \leqslant |U|^k$ for some

[1] This is slight abuse of notation, because the result presented in [19] is valid under the assumption $NP \neq DTIME(n^{O(\log \log n)})$, an assumption stronger than $P \neq NP$. The result is not stated explicitly, but it follows from the proof presented therein.

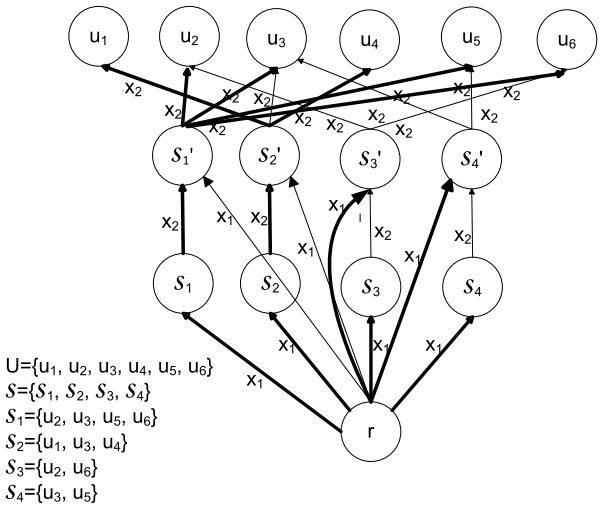


Fig. 2. Digraph G corresponding to an instance of MinCardSetCover with $S_1 = \{u_2, u_3, u_5, u_6\}$, $S_2 = \{u_1, u_3, u_4\}$, $S_3 = \{u_2, u_6\}$, $S_4 = \{u_3, u_5\}$. Bold arcs indicate the spanning tree $T(\mathcal{C}^*)$ corresponding to the optimal set cover $\mathcal{C}^* = \{S_1, S_2\}$.

fixed $k > 0$. Given an instance $(U, \mathcal{S})$ of MinCardSetCover we construct an instance (G, X, χ, r, c) of MinDirectedCCA (see Fig. 2).

$G = (V, E)$ is a Dag where $V = \{r\} \cup \mathcal{S} \cup \mathcal{S}' \cup U$, $\mathcal{S}' = \{S_i' \mid S_i \in \mathcal{S}\}$. $E = E_1 \cup E_2 \cup E_3 \cup E_4$ where $E_1 = \{(r, S_i) \mid S_i \in \mathcal{S}\}$, $E_2 = \{(r, S_i') \mid S_i \in \mathcal{S}\}$, $E_3 = \{(S_i, S_i') \mid S_i \in \mathcal{S}\}$, $E_4 = \{(S_i', u_j) | u_j \in S_i, S_i \in \mathcal{S}\}$.

$X = \{x_1, x_2\}$ and the cost function c is such that $c(x_1, x_2) = c(x_2, x_1) = 1$. Recall that $c(x_1, x_1) = c(x_2, x_2) = 0$, by definition. Also

$$\chi(e) = \begin{cases} x_1 & \text{if } e \in E_1 \cup E_2 \\ x_2 & \text{otherwise.} \end{cases}$$

Note that only vertices of $\mathcal{S}'$ have free traversals (namely between edges colored x_2), however all such traversals of the same vertex have a common edge. Therefore there are no crossing free traversals.

Let $\mathcal{C}^*$ be an optimal set cover, and without loss of generality assume $\mathcal{C}^* = \{S_1, S_2, \ldots, S_\ell\}$. Consider the spanning tree $T(\mathcal{C}^*)$ of G rooted at r with the arcs $E(T(\mathcal{C}^*)) = E_1 \cup E_2^* \cup E_3^* \cup E_4^*$ where $E_2^* = \{(r, S_i') \in E_2 \mid i > \ell\}$, $E_3^* = \{(S_i, S_i') \in E_3 \mid 1 \leqslant i \leqslant \ell\}$, and $E_4^* \subseteq E_4$ is such that each u_j has exactly one incoming arc (S_i', u_j) in E_4^* and also $1 \leqslant i \leqslant k$. Note that there is always such an arc because there is at least one set in $S_i \in \mathcal{C}^*$ containing u_j as $\mathcal{C}^*$ is a set cover. The changeover costs of E_1 and E_2^* are zero because $prev(e) = e$ for every edge $e \in E_1 \cup E_2^*$. The changeover cost of an edge $e \in E_3^*$ is 1 and the changeover cost of an edge $e \in E_4^*$ is zero because $\chi(e) = \chi(prev(e)) = x_2$. Therefore, $c(T(\mathcal{C}^*)) = |E_3^*| = \ell = |\mathcal{C}^*|$. The cost c^* of an optimal solution of (G, X, χ, r, c) is at most $c(T(\mathcal{C}^*))$, therefore

$$c^* \leqslant \left|\mathcal{C}^*\right|.$$

Let $T = (V, E_T)$ be a spanning tree of G rooted at r constituting a $\bar{\rho}$-approximation to the constructed instance (G, X, χ, r, c) of MinDirectedCCA. Note that $E_1 \subseteq E_T$, because otherwise some vertex $S_i \in \mathcal{S}$ is not reachable from r in T. Furthermore, for every i, exactly one of $\{(r, S_i'), (S_i, S_i')\}$ is in E_T because otherwise T is not a directed tree rooted at r.

Let $Q \subseteq \mathcal{S}'$ be the set of vertices S_i' such that $(r, S_i') \notin E_T$ and $(S_i, S_i') \in E_T$. Then $\overline{Q} \stackrel{def}{=} \mathcal{S}' \setminus Q$ is the set of vertices S_i' such that $(r, S_i') \in E_T$ and $(S_i, S_i') \notin E_T$. Let U_Q be the set of vertices of U that are reachable from Q in T, and $\overline{U_Q} \stackrel{def}{=} U \setminus U_Q$. Then the vertices of $\overline{U_Q}$ are reachable from $\overline{Q}$ in T.

We now construct a solution $T' = (V, E_{T'})$ with $c(T') \leqslant c(T)$ and $\overline{U_Q} = \emptyset$. Consider a vertex $u_j \in \overline{U_Q}$ and let $S_i' \in \overline{Q}$) be its parent vertex in T. Now consider the subtree of T containing the arcs $(r, S_i), (r, S_i')$ and all the arcs going from S_i' to its children. The changeover cost of this subtree is equal to the number of children of S_i', thus at least 1. On the other hand, the subtree obtained by removing the arc (r, S_i') and adding the arc (S_i, S_i') has changeover cost exactly 1. In the new subtree $S_i' \in Q$, and all the children of S_i' are in U_Q. We can proceed with this transformation until $\overline{U_Q} = \emptyset$ to get a solution T' as claimed. Now $U_Q = U$ and thus Q is a set cover. It can be easily verified that $c(T') = |Q|$. We conclude

$$|Q| = c(T') \leqslant c(T) \leqslant \bar{\rho} \cdot c^* \leqslant \rho \cdot \left|\mathcal{C}^*\right|.$$

Thus, Q constitutes a $\bar{\rho}$-approximation to MinCardSetCover.

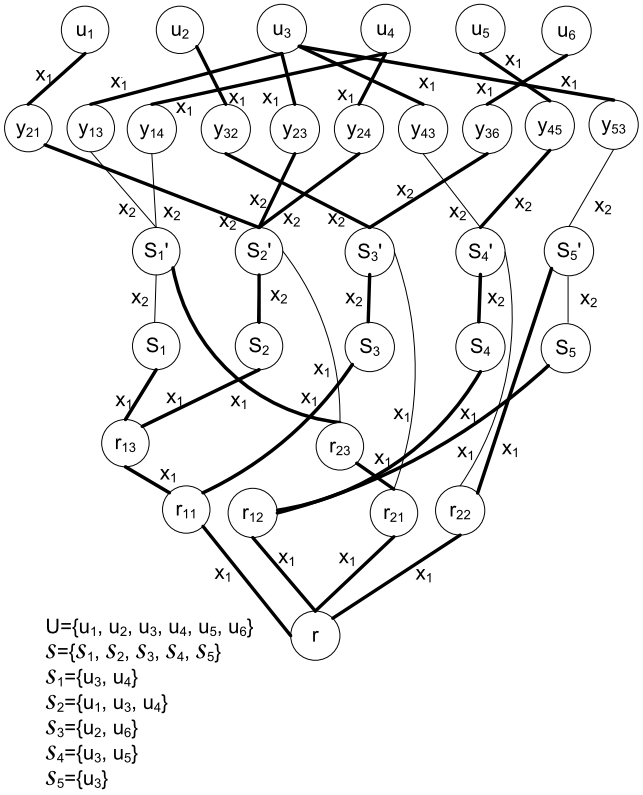


Fig. 3. Graph G corresponding to an instance of MinCardSetCover having $S_1 = \{u_3, u_4\}$, $S_2 = \{u_1, u_3, u_4\}$, $S_3 = \{u_2, u_6\}$, $S_4 = \{u_3, u_5\}$, $S_5 = \{u_3\}$. Bold arcs indicate the spanning tree $T(C^*)$ corresponding to the optimum set cover $C^* = \{S_2, S_3, S_4\}$.

Finally, we note that $|V| = |U| + 2|\mathcal{S}| + 1 \leqslant 4|U|^k$, thus $\log|V| \leqslant k \log|U| + 2$. Let ρ' be a constant for which MinCardSetCover is $(\rho' \cdot \log|U|)$-inapproximable, and let $\rho < \rho'/k$. Assume that there is a $(\rho \cdot \log|V|)$-approximation algorithm for MinDirectedCCA. Then we can use it to find a $(\rho' \cdot \log|U| + O(1))$-approximation to MinCardSetCover, a contradiction. □

A similar reduction has been presented independently in [2]. Though, our result is stronger ($O(\log|V|)$ vs. $O(\log\log|V|)$ inapproximability).

3.3. Bounded costs and bounded vertex degrees

Theorem 2. *For any integer constant $B \geqslant 5$,* MinDirectedCCA *is in* Apx*-Hard even when G is a* Dag *without crossing free traversals, $\Delta(G) = B$, $|X| = 2$, and $C_{max} = 1$.*

Proof. It was shown in [20] that for every $k \geqslant 3$, MinCardSetCover is in Apx-Hard even for instances where the sets are of cardinality at most k and each element appears in at most 2 sets. Given such an instance $(U, \mathcal{S})$ of MinCardSetCover we construct an instance (G, X, χ, r, c) of MinDirectedCCA in two steps. In the first step, we construct an instance of MinDirectedCCA as we did in the proof of Theorem 1 (see Fig. 2). In the second step we replace the tree induced by $\{r\} \cup \mathcal{S}$ with a directed tree rooted at r with edges colored x_1, out-degree at most 2, whose leaves are $\mathcal{S}$ (for example, see the tree induced by the vertices $\{r, r_{11}, r_{12}, r_{13}\} \cup \mathcal{S}$ in Fig. 3). We do the same transformation to the tree induced by $\{r\} \cup \mathcal{S}'$. Note that this does not change the changeover costs of the original edges. Furthermore, the changeover costs of the new edges are zero, i.e. same as the changeover costs of the removed edges. By the choice of the instance $(U, \mathcal{S})$, every vertex in U has in-degree at most 2. Moreover, the degree of every vertex in $\mathcal{S}'$ is at most $k + 2$, the degree of r is at most 4, and the degree of all the other vertices is at most 3. Therefore $\Delta(G) \leqslant k + 2$. As the changeover costs did not change, using the same proof as in Theorem 1 we can show that a ρ-approximation to MinDirectedCCA implies a ρ-approximation to the instance under consideration of MinCardSetCover.

Assume, by way of contradiction, that for some $B \geqslant 5$, there exists a PTAS for the MinDirectedCCA problem on graphs with $\Delta(G) \leqslant B$. Using our reduction, this implies a PTAS for MinCardSetCover instances for which the sets have at most $B - 2 \geqslant 3$ elements and every element appears in at most 2 sets, a contradiction. □

4. Lower bounds for undirected graphs

4.1. Bounded number of colors

Theorem 3. *For any integer constant $B \geqslant 5$,* MinCCA *is in* Apx*-Hard even when $\Delta(G) = B$, $|X| = 2$, $C_{max} = 1$, and G_1 and G_2 are forests.*

Proof. Let $B \geqslant 5$ an integer constant and let $k = B - 2 \geqslant 3$. We recall that for every $k \geqslant 3$, MinCardSetCover is in Apx-Hard even for instances where the sets are of cardinality at most k and each element appears in at most 2 sets [20]. Given such an instance $(U, \mathcal{S})$ of MinCardSetCover we construct an instance (G, X, χ, r, c) of MinCCA as follows (see Fig. 3):

$G = (V, E)$ is a graph where $V = \{r\} \cup R_1 \cup R_2 \cup \mathcal{S} \cup \mathcal{S}' \cup Y \cup U$, $\mathcal{S}' = \{S'_i \mid S_i \in \mathcal{S}\}$, and $Y = \{y_{ij} | S_i \in \mathcal{S}, u_j \in S_i\}$. $E = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5$. R_1, R_2, E_1, and E_2 are constructed such that the subgraph $(\{r\} \cup R_1 \cup \mathcal{S}, E_1)$ of G is a directed binary tree with root r leaves $\mathcal{S}$; and R_2 and E_2 are such that the subgraph $(\{r\} \cup R_2 \cup \mathcal{S}', E_2)$ of G is a tree with root r, leaves $\mathcal{S}'$, and out-degree 2. $E_3 = \{\{S_i, S'_i\} \mid S_i \in \mathcal{S}\}$, $E_4 = \{\{S'_i, y_{ij}\} \mid u_j \in S_i, S_i \in \mathcal{S}\}$, and $E_5 = \{\{y_{ij}, u_j\} \mid u_j \in S_i, S_i \in \mathcal{S}\}$. Note that $\Delta(G) \leqslant B$ and this maximum is attained for vertices of $\mathcal{S}'$ corresponding to sets with k elements. $X = \{x_1, x_2\}$ and the cost function c is such that $c(x_1, x_2) = c(x_2, x_1) = 1$. Also

$$\chi(e) = \begin{cases} x_1 & \text{if } e \in E_1 \cup E_2 \cup E_5 \\ x_2 & \text{otherwise.} \end{cases}$$

To see that the above construction can be performed in polynomial time, we note that a binary tree with k leaves has $2k-1$ vertices and can be easily constructed in time linear in k.

Let $\mathcal{C}^*$ be an optimal set cover, and without loss of generality assume $\mathcal{C}^* = \{S_1, S_2, \ldots, S_\ell\}$. Consider the spanning tree $T(\mathcal{C}^*)$ of G with the edges $E(T(\mathcal{C}^*)) = E_1 \cup E_2^* \cup E_3^* \cup E_4^* \cup E_5$ where $E_2^* = E_2 \setminus \{\{r_2, S'_i\} \in E_2 \mid i \leqslant k, r_2 \in R_2\}$, $E_3^* = \{\{S_i, S'_i\} \in E_3 \mid 1 \leqslant i \leqslant \ell\}$, and $E_4^* \subseteq E_4$ is such that for each u_j there is exactly one edge $\{S'_i, y_{ij}\}$ in E_4^* and also $1 \leqslant i \leqslant \ell$. Note that there is always such an edge because there is at least one set in $S_i \in \mathcal{C}^*$ containing u_j as $\mathcal{C}^*$ is a set cover. Whenever an element is covered more then once we choose an edge to E_4^* arbitrarily. The changeover costs of E_1 and E_2^* are zero because for every edge $e \in E_1 \cup E_2^*$ either $prev(e) = e$ or $\chi(prev(e)) = \chi(e) = x_1$. The changeover cost of an edge $e \in E_3^*$ is 1, the changeover cost of an edge $e \in E_4^*$ is zero because $\chi(e) = \chi(prev(e)) = x_2$, and finally the changeover cost of an edge of E_5 is 1 if it is incident to an edge in E_4^* and zero otherwise. Therefore, $c(T(\mathcal{C}^*)) = |E_3^*| + |E_4^*| = \ell + |U| = |\mathcal{C}^*| + |U|$. The cost c^* of an optimal solution of (G, X, χ, r, c) is at most $c(T(\mathcal{C}^*))$, therefore

$$c^* \leqslant |\mathcal{C}^*| + |U|.$$

In the following claim a subtree of T means a tree of the forest obtained by the removal of $\{r\} \cup R_1 \cup R_2$ from T.

Claim 1. *Every solution of T of (G, X, χ, r, c) can be transformed in polynomial time to a solution T' with $c(T') \leqslant c(T)$ satisfying:*

- *(i) Every subtree of T' contains at most one vertex of $\mathcal{S}'$.*
- *(ii) $E_5 \subseteq E(T')$.*
- *(iii) If there is a path between r and S'_i in T' that does not traverse S_i then S'_i is a leaf of T'.*

Proof. Suppose that T has a subtree containing more than one vertex of $\mathcal{S}'$. Assume without loss of generality that S'_1 and S'_2 are the closest and second closest, respectively, to r among these vertices. The path between these two vertices in T does not use $\{r\} \cup R_1 \cup R_2$ because they are in the same subtree, therefore it must use some vertex $u_j \in U$. We conclude that there is a path $r - \cdots - S_1 - y_{1j} - u_j - y_{2j} - S_2$ in T (with possibly the vertex S_1 and vertices of $R_1 \cup R_2$ between r and S'_1). We consider two cases: (a) $\{S_2, S'_2\} \notin E(T)$. In this case we replace $\{y_{2j}, S'_2\}$ by $\{S_2, S'_2\}$ to obtain a new tree with the same cost and less $\mathcal{S}'$ vertices sharing a subtree. (b) $\{S_2, S'_2\} \in E(T)$. In this case we replace $\{y_{2j}, S'_2\}$ by an edge $\{r_1, S_2\}$ where $r_1 \in \{r\} \cup R_1$ to obtain a tree with at most the same changeover cost and less $\mathcal{S}'$ vertices sharing a subtree. By applying this procedure as long as there are $\mathcal{S}'$ vertices sharing a subtree, we get a tree having the same changeover cost or less, and satisfying (i).

Suppose that there is an edge $\{y_{ij}, u_j\} \in E_5 \setminus E(T)$. Then $\{S'_i, y_{ij}\} \in E(T)$, because otherwise y_{ij} is isolated. By removing this edge and adding the edge $\{y_{ij}, u_j\}$ we obtain a tree with at most the same cost and containing one more edge of E_5. By applying this transformation as long as there are edges in $E_5 \setminus E(T)$ we obtain a tree with at most the same changeover cost, satisfying (ii). Note that this transformation does not violate (i).

Suppose that there is a path P between r and S'_i in T that does not traverse S_i, and that S'_i is not a leaf. If S'_i has $k \geqslant 1$ adjacent Y vertices in T, then by removing the edge of P adjacent to S'_i from $E(T)$ and adding the missing edge among $\{\{r, S_i\}, \{S_i, S'_i\}\}$ we can reduce the changeover cost by at least $k - 1 \geqslant 0$ and get a tree with one less path violating (iii). If S'_i has no adjacent Y vertices in T, then it is adjacent to S_i, because otherwise it is a leaf of T. By removing $\{S_i, S'_i\}$ and adding an edge $\{r_1, S_i\}$ where $r_1 \in R_1$ we get a tree with smaller cost and one less edge of $R_1 \times \mathcal{S}'$ violating (iii). Clearly, these transformations violate neither (i) nor (ii), and applying them as long as T contains violating edges we get a tree T' as claimed. □

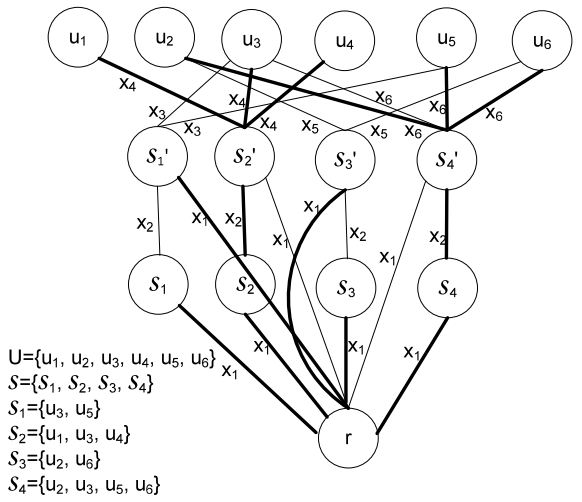


Fig. 4. Graph G corresponding to an instance of MinCardSetCover having $S_1 = \{u_3, u_5\}$, $S_2 = \{u_1, u_3, u_4\}$, $S_3 = \{u_2, u_6\}$, $S_4 = \{u_2, u_3, u_5, u_6\}$. Bold arcs indicate the spanning tree $T(C^*)$ corresponding to the optimum set cover $C^* = \{S_2, S_4\}$.

We proceed with the proof of the theorem. Let $T = (V, E_T)$ be a spanning tree of G rooted at r constituting a ρ-approximation to the constructed instance (G, X, χ, r, c) of MinCCA. By the above claim, it can be transformed in polynomial time into a ρ-approximation $T' = (V, E_{T'})$ satisfying conditions (i, ii, iii). In such a solution, all the E_2 edges connect r to a leaf S'_i, thus do not incur any changeover cost. E_1 edges connect r to vertices S_i, some of which are leaves of T'. Let $Q \subseteq \mathcal{S}$ be the set of vertices S_i such that there is an edge $\{r_1, S_i\} \in E_{T'}$ for some $r_1 \in R_1$ and S_i is not a leaf of T'. Every vertex $S_i \in Q$ is connected to S'_i in T' by an edge with a changeover cost of 1. Thus, the changeover cost of these edges is $|Q|$. All the vertices $u_j \in U$ are distributed among the subtrees rooted at a vertex of Q. By (i), every S'_i is unique in its subtree, thus the u_j vertices in each subtree are connected to S'_i by a path $S'_i - y_{ij} - u_j$ of length 2. Therefore $u_j \in S_i$. We conclude that Q constitutes a set cover of $(U, \mathcal{S})$. The cost of each such path is 1, incurring a total changeover cost of $|U|$. The remaining E_5 edges do not incur a changeover cost because their previous edges are also in E_5. Therefore $c(T') = |Q| + |U|$. We conclude

$$\begin{aligned} |Q| &= c(T') - |U| \leqslant c(T) - |U| \leqslant \rho \cdot c^* - |U| \\ &\leqslant \rho\big(|C^*| + |U|\big) - |U| = (\rho - 1)|U| + \rho \cdot |C^*| \\ &\leqslant (\rho - 1)k|C^*| + \rho \cdot |C^*| = \big((k+1)\rho - k\big)|C^*| \end{aligned}$$

(in the last line we used the fact that each set contains at most k elements). Therefore, Q constitutes a $((k+1)\rho - k)$-approximation to the MinCardSetCover problem, which is known to be in Apx-Hard. Let $\rho' > 1$ be a constant such that MinCardSetCover is ρ'-inapproximable. Then $(k+1)\rho - k > \rho'$ and $\rho > (\rho' + k)/(k+1)$, implying that for $\rho = (\rho' + k)/(k+1) > 1$ MinCCA is ρ-inapproximable. □

4.2. Unbounded number of colors

Theorem 4. *There is a constant $\rho > 0$ such that* MinCCA *is $(\rho \cdot \log |V|)$-inapproximable even when $C_{max} = 1$.*

Proof. Similar to Theorem 1, we provide an approximation preserving reduction from MinCardSetCover to MinCCA. We show the reduction for the case where all nonzero costs are equal to 1. Given an instance $(U, \mathcal{S})$ of MinCardSetCover, we build an instance (G, X, χ, r, c) of MinCCA. Consult Fig. 4 for the construction.

The construction is the same as in Theorem 1 except that each edge $\{S'_i, u_j\} \in E_4$ is colored with x_{i+2}. The cost function is

$$c(x_i, x_j) = \begin{cases} 0 & \text{if } i = j \\ 0 & \text{if } \min(i, j) = 2 \text{ and } \max(i, j) > 2 \\ 1 & \text{otherwise.} \end{cases}$$

The impact of having directed edges between $\{S'_i, u_j\}$ in Fig. 2 is achieved in Fig. 4 by having undirected edges with different colors, between which the cost is 1. Hence, the same reduction in Theorem 1 can be applied here. □

Algorithm 1 Algorithm for monochromatic biconnected components.

Require: For every biconnected component B of G, the edges $E(B)$ are colored with the same color.
1: Calculate the set $\mathcal{B}$ of biconnected components of G.
2: Let $\mathcal{B}_r$ the set of biconnected components of G that contain r
3: **for all** $B \in \mathcal{B} \setminus \mathcal{B}_r$ **do**
4: $v_B \leftarrow$ the (unique) vertex in B closest to r
5: $\bar{T}_B \leftarrow$ a spanning tree of $B \setminus \{v_B\}$
6: Connect v_B to $\bar{T}_B$ as a leaf
7: **end for**
8: **for all** $B \in \mathcal{B}_r$ **do**
9: $\bar{T}_B \leftarrow$ a spanning tree of B
10: **end for**
11: $\bar{T} \leftarrow \bigcup_{B \in \mathcal{B}} \bar{T}_B$
12: **return** $\bar{T}$.

5. Upper bounds

In the previous sections we presented inapproximability results. An interesting question is to characterize the instances that admit polynomial time algorithms. Though we do not have such a characterization, we provide in this section polynomial-time optimal algorithms for two special cases of the MinCCA problem. In the first case the graph is arbitrary but we impose a restriction on the coloring χ. In the second case χ is arbitrary and we impose restrictions on G. Finally, we provide a $(C_{max} \cdot \log|V|)$-approximation algorithm for MinDirectedCCA, which matches the lower bound we proved in Section 3.2.

5.1. Polynomial cases for undirected graphs

Theorem 5. MinCCA *is solvable in polynomial time for graphs whose biconnected components are monochromatic.*

Proof. Algorithm 1 starts by decomposing the graph into its biconnected components in Line 1. For each biconnected component B that does not contain r the algorithm finds a spanning tree of B such that the vertex of B closest to r is a leaf (note that this vertex is unique and it is the only cut vertex in B that separates the rest of B from r). For each component that contains r it finds an arbitrary spanning tree. Finally, it returns the union of all these trees.

The running time of the Algorithm 1 is polynomial because (a) Step 1 can be performed in linear time [16], (b) the number of iterations of the loops is sublinear, and (c) a spanning tree can be calculated in polynomial time. In the sequel we show that the graph $\bar{T}$ returned by the algorithm is a tree and its cost is minimum whenever the biconnected components are monochromatic. In this discussion we use the same symbol to denote both a biconnected component and its vertex set.

$\bar{T}$ is a spanning tree because it is the union of spanning trees of biconnected components. To see that such a graph is connected we note that for any vertices $u, v \in V(G)$ such that $u \in B_u$, $v \in B_v$ for biconnected components B_u, B_v. The path in the block tree from B_u to B_v can be used to find a path between u and v in G. A cycle in this graph spans at least two biconnected components because in each biconnected component we have tree. The sequence of articulation points in this cycle imply a cycle in the block tree, a contradiction.

Let T be any spanning tree of G, and B a biconnected component of G. Then the induced subgraph $T[B]$ is a tree. Clearly $T[B]$ is acyclic, because T is acyclic. Assume, by contradiction, that $T[B]$ is not connected. Then there are two vertices $u, v \in V(B)$ such that $p_T(u, v)$ contains vertices not in B. Then the sequence of articulation points of the path $p_T(u, v)$ implies a cycle in the block tree of G, a contradiction.

Consider the block tree of G. Let $\mathcal{B}_r$ be the set of biconnected components containing r. Note that $|\mathcal{B}_r| > 1$ if r is an articulation point. In this case all the biconnected components of $\mathcal{B}_r$ are adjacent in the block tree. A biconnected component B of G that does not contain r, has exactly one vertex v_B closest to r. Moreover, v_B is the articulation point adjacent to B on the path from B to any vertex of $\mathcal{B}_r$ in the block tree. Let $prev(B)$ be the third vertex (i.e. adjacent to v_B) on this path. As we consider monochromatic biconnected components, we denote by $\chi(B)$ the common color of all the edges of B. Consider a spanning tree T of G rooted at r. Only the first edges of every subtree $T[B]$ incur a changeover cost. The number of such edges is $d_{T[B]}(v_B)$ and the cost of each one is $c(\chi(prev(B)), \chi(B))$. Therefore,

$$c(T, r) = \sum_{B \notin B_r} d_{T[B]}(v_B) \cdot c\big(\chi\big(prev(B)\big), \chi(B)\big) \geqslant \sum_{B \notin B_r} c\big(\chi\big(prev(B)\big), \chi(B)\big).$$

For the tree $\bar{T}$ returned by the algorithm, $\bar{T}[B]$ is a tree where v_B is a leaf. Therefore

$$c(\bar{T}, r) = \sum_{B \notin B_r} c\big(\chi\big(prev(B)\big), \chi(B)\big).$$

We conclude that $\bar{T}$ is optimal. □

Algorithm 2 Algorithm for a subfamily of cactus graphs.

Require: G is a Cactus Graph with every vertex in at most one simple cycle.
1: $E_T \leftarrow \{e \in E \mid e \text{ is a bridge}\}$
2: **for all** cycles C of G **do**
3: $v_C \leftarrow$ the vertex of C that is closest to r
4: **if** $v_C = r$ **then**
5: $v'_C \leftarrow r$
6: **else**
7: $v'_C \leftarrow$ the neighbor of v_C that is not in C
8: **end if**
9: $bridges_C \leftarrow \{\{u, v\} \in E \mid u \in V(C), v \notin V(C)\}$
10: $cc \leftarrow \infty$
11: **for all** $e \in E(C)$ **do**
12: $T' \leftarrow C \cup bridges_C - e$
13: $cc_e \leftarrow c(T', v'_C)$
14: **if** $cc_e < cc$ **then**
15: $e'_C \leftarrow e$
16: $cc \leftarrow cc_e$
17: **end if**
18: **end for**
19: $E_T \leftarrow E_T \cup E(C) - e'_C$
20: **end for**
21: **return** $T = (V, E_T)$

Theorem 6. MinCCA *is solvable in polynomial time for the special case of cactus graphs in which every vertex belongs to at most one simple cycle.*

Proof. We claim that Algorithm 2 solves in polynomial time the special case of cactus graphs in which every vertex belongs to at most one simple cycle. Step 1 adds all the bridges to the MinCCA solution because they have to be in any spanning tree of G. The remaining graph consists of disjoint cycles. In order to find a spanning tree, exactly one edge has to be removed from each cycle. The crucial point is that for any two such cycles C_1 and C_2, the decision of the edges to be removed can be taken independently of each other. Lines 3–19 find this edge by exhaustive search. □

5.2. Approximation algorithm for directed acyclic graphs

In this section we present a $(C_{max} \cdot \log|V|)$-approximation algorithm for DAGs without crossing free traversals, proving that the bound of Theorem 1 is tight up to a constant factor.

Given a digraph $G = (V, E)$ with a coloring function χ on its arcs, a cost function c, and an arc $e = (u, v) \in E$ directed from u to v, $R(e)$ is the set of all vertices of G reachable from v at a cost of zero. Note that this set can be calculated in polynomial time. The *balloon* $B(e)$ is the subgraph $G[R(e)] + e$ of G. A balloon is maximal if it is not contained in any other balloon. Note that for the arc $e = (u, v) \in E$ we have (i) $u, v \in V(B(e))$, (ii) $v \in R(e)$, (iii) $u \notin R(e)$ (because otherwise there is a directed circuit in G).

Theorem 7. *Algorithm 3 is a* $(C_{max} \cdot \log|V|)$*-approximation algorithm for* MinDirectedCCA *instances where* G *is a DAG without crossing free traversals.*

Proof. Consider any solution to MinDirectedCCA, in particular an optimal solution T^*. Consider the set of arcs $E_1^* \stackrel{def}{=} \{e \in E(T^*) \mid c(e) > 0\}$. Clearly, $|E_1^*| \leqslant c(T^*)$. Let $E_2^* \stackrel{def}{=} E_1^* \cup \delta_G^{(out)}(r)$. From every vertex $v \in V - r$, we can find a unique arc $e \in E_2^*$ by following the (reverse) path from v to r until we reach either a vertex that is traversed with a non-zero cost, or r. Then v is reachable from e with zero cost, i.e. $v \in R(e)$. Therefore, the set $\mathcal{C}^* = \{R(e) | e \in E_2^*\}$ is a partition of $V - r$. We associate a weight w with each set of this partition such that $w(R(e)) = 0$ if $e \in \delta_G^{(out)}(r)$, and $w(R(e)) = 1$ otherwise (i.e. when $e \in E_1^*$). Then the weight $w(\mathcal{C}^*)$ associated with $\mathcal{C}^*$ is $|E_1^*|$. Therefore $w(\mathcal{C}^*) \leqslant c(T^*)$. In fact, Algorithm 3 is based on this observation. Although the observation holds for every instance of MinDirectedCCA, the algorithm works only for *DAG*s without crossing free traversals.

In Step 8 we find a set cover $\mathcal{C}$ of $V - r$ using maximal sets $R(e)$. Using the greedy algorithm from [12] provides us with an approximation guarantee of $w(\mathcal{C}) \leqslant w(\mathcal{C}^*) \cdot \log|V|$.

Initially the sets $\bar{B}(e)$ are the balloons associated with $\mathcal{C}$. In Lines 9–21 we transform $\mathcal{C}$ to a partition by choosing for each vertex v exactly one set from $\mathcal{C}$, and modifying the sets $\bar{B}(e)$ appropriately. This step succeeds, i.e. the graphs $\bar{B}(e)$ remain reachable from e, due to our assumption of free traversals being pairwise non-crossing. Indeed, if v is a sink in some balloon, then it can be safely removed from it. Assume that v is a non-sink in two balloons B, B'. Let e_1, e_2 (resp. e'_1, e'_2 be a pair of edges of B (resp. B') traversing v. By definition of a balloon, these are free traversals. Moreover, by our assumption, they are not crossing. Then $e_1 = e'_1$ or $e_2 = e'_2$. In the first case e_1 is the only incoming edge of v in both balloons. Then the set of vertices reachable from v in B and B' are the same. In the second case e_2 is the only outgoing edge of v in both

Algorithm 3 Algorithm for DAGs without crossing free traversals.

Require: G is a DAG.
Require: G does not have crossing free traversals.
1: **for all** maximal balloons $B(e)$ **do**
2: **if** $e = (r, v)$ for some $v \in V$ **then**
3: $w(R(e)) = 0$
4: **else**
5: $w(R(e)) = 1$
6: **end if**
7: **end for**
8: $\mathcal{C} \leftarrow$ a cover of $V - r$ using the sets $R(e)$ with associated weights $w(R(e))$ ▷ Use the greedy algorithm [12].
9: **for all** $R(e) \in \mathcal{C}$ **do**
10: $\bar{B}(e) \leftarrow B(e)$
11: **end for**
12: **for all** $v \in V - r$ **do**
13: **if** v is in $k > 1$ sets $R(e_1), \ldots, R(e_k) \in \mathcal{C}$ **then**
14: **if** v is a sink in all the graphs $\bar{B}(e_i)$ **then**
15: $i \leftarrow 1$
16: **else**
17: Choose i such that v is not a sink in $\bar{B}(e_i)$ ▷ i is unique
18: **end if**
19: Remove v from all the graphs $\bar{B}(e)$ except $\bar{B}(e_i)$
20: **end if**
21: **end for**
22: **for all** $R(e_i) \in \mathcal{C}$ **do**
23: Convert the $\bar{B}(e_i)$ to a tree rooted at the start vertex of e_i
24: **end for**
25: **return** $\bigcup_{R(e_i)\in\mathcal{C}} \bar{B}(e_i)$

balloons, and we observe that the claim holds. Therefore, removing v and all the vertices reachable from it, from one of these balloons is a safe operation, because the vertices remain reachable in the other.

Line 23 transforms each graph $\bar{B}(e)$ to a tree. This is always possible because all the vertices are reachable from e. Finally, the algorithm returns the union T of all these trees. Every vertex except r is a non-root vertex of exactly one of these trees, thus has exactly one incoming arc in T, and there are no directed circuits in T, because G is a DAG. Therefore, T is a tree rooted at r.

In addition, we observe that each tree $\bar{B}(e)$ that is added to the final solution has a reload cost of 0 because it is part of a balloon. However, adding each $\bar{B}(e)$ when e does not start at r to the solution tree T incurs a changeover cost of at most C_{max} for the arc e, i.e. $c(T) \leqslant C_{max} \cdot w(\mathcal{C})$ and therefore we get that Algorithm 3 is a $(C_{max} \cdot \log|V|)$-approximation algorithm. □

Corollary 1. MINDIRECTEDCCA *is* log*-*APX*-Complete even when G is a DAG without crossing free traversals, and C_{max} is bounded by some constant.*

We note that the number of colors does not affect the hardness of the problem in this case. The lower bound holds for two colors and it is achievable for any number of colors.

6. Summary and future work

In this paper we studied the complexity and approximability properties of MINCCA and MINDIRECTEDCCAproblems. We started our investigation by observing that for the general case, in which the costs can be arbitrarily large, MINDIRECTEDCCA cannot be approximated within any polynomial time computable function $f(n)$. Therefore, we studied special cases of the problem in directed and undirected graphs. We developed hardness results for cases such as bounded and unbounded vertex degrees, bounded and unbounded number of colors, and bounded cost values. Our results show that MINCCA and MINDIRECTEDCCA are inherently difficult to approximate even in special cases. We introduced a tight $C_{max} \cdot \log|V|$-approximation algorithm for MINDIRECTEDCCA in the special case of *DAG*s without crossing free traversals and bounded costs. We showed that in this case the number of colors does not affect the hardness of the problem.

We also presented two special cases solvable in polynomial time (to the best of our knowledge there are no other families of instances for which polynomial-time optimal algorithms have been presented): (a) The graph is arbitrary and the coloring is such that every biconnected component is monochromatic, (b) When the graph is a special case of cactus graphs and the coloring is arbitrary.

Two main research directions are of interest: developing approximation algorithms for the remaining special cases considered in this work, and getting hardness results for other cases. In particular, it would be interesting to find approximation algorithms for more general cases, as well as stronger inapproximability results for directed graphs. Another interesting research direction is to find other families of instances for which the problem is polynomial, beyond the above mentioned two families.

References

[1] E. Amaldi, G. Galbiati, F. Maffioli, On minimum reload cost paths, tours, and flows, Networks 57 (3) (2011) 254–260.
[2] G. Galbiati, S. Gualandi, F. Maffioli, On minimum changeover cost arborescences, in: Experimental Algorithms – 10th International Symposium, SEA, Kolimpari, Chania, Crete, Greece, 2011, pp. 112–123.
[3] H. Wirth, J. Steffan, Reload cost problems: minimum diameter spanning tree, Discrete Appl. Math. 113 (1) (2001) 73–85.
[4] I. Akyildiz, W. Lee, M. Vuran, S. Mohanty, Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey, Computer Networks 50 (13) (2006) 2127–2159.
[5] B. Wang, K. Liu, Advances in cognitive radio networks: a survey, IEEE J. Sel. Top. Signal Process. 5 (1) (2011) 5–23.
[6] D. Gözüpek, S. Buhari, F. Alagöz, A spectrum switching delay-aware scheduling algorithm for centralized cognitive radio networks, IEEE Trans. Mob. Comput. 12 (7) (2013) 1270–1280.
[7] G. Galbiati, The complexity of a minimum reload cost diameter problem, Discrete Appl. Math. 156 (18) (2008) 3494–3497.
[8] G. Galbiati, S. Gualandi, F. Maffioli, On minimum reload cost cycle cover, Discrete Appl. Math. 164 (1) (2014) 112–120.
[9] L. Gourvès, A. Lyra, C. Martinhon, J. Monnot, The minimum reload s–t path, trail and walk problems, Discrete Appl. Math. 158 (13) (2010) 1404–1417.
[10] I. Gamvros, L. Gouveia, S. Raghavan, Reload cost trees and network design, Networks 59 (4) (2012) 365–379.
[11] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation, Combinatorial Optimization Problems and Their Approximability Properties, Springer Verlag, 1999.
[12] V. Chvátal, A greedy heuristic for the set covering problem, Math. Oper. Res. 4 (1979) 233–235.
[13] R. Raz, S. Safra, A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP, in: Proceedings of the 29th ACM Symposium on Theory of Computing, STOC, 1997, pp. 475–484.
[14] E. Reingold, J. Nievergelt, N. Deo, Combinatorial Algorithms: Theory and Practice, Prentice Hall, 1977.
[15] J. Bondy, U. Murty, Graph Theory, Springer, 2008.
[16] J. Hopcroft, R. Tarjan, Efficient algorithms for graph manipulation, Commun. ACM 16 (6) (1973) 372–378.
[17] S. Even, Graph Algorithms, Computer Science Press, 1979.
[18] A. Brandstädt, V. Le, J. Spinrad, Graph Classes: A Survey, Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics, 1999.
[19] V.V. Vazirani, Approximation Algorithms, Springer Verlag, 2003.
[20] R. Duh, M. Fürer, Approximation of k-set cover by semi-local optimization, in: ACM Symposium on Theory of Computing, STOC, 1997, pp. 256–264.