Clearing connections by few agents

Christos Levcopoulos[1], Andrzej Lingas[1]
Bengt J. Nilsson[2], and Paweł Żyliński[3]

[1] Lund University, S-221 00 Lund, Sweden
{Christos.Levcopoulos,Andrzej.Lingas}@cs.lth.se
[2] Malmö University, SE-205 06 Malmö, Sweden
bengt.nilsson.TS@mah.se
[3] University of Gdańsk, 80-952 Gdańsk, Poland
zylinski@inf.ug.edu.pl

Abstract. We study the problem of clearing connections by agents placed at some vertices in a directed graph. The agents can move only along directed paths. The objective is to minimize the number of agents guaranteeing that any pair of vertices can be connected by a underlying undirected path that can be cleared by the agents. We provide several results on the hardness, approximability and parameterized complexity of the problem. In particular, we show it to be: NP-hard, 2-approximable in polynomial-time, and solvable exactly in $O(\alpha n^3 2^{2\alpha})$ time, where α is the number of agents in the solution. In addition, we give a simple linear-time algorithm optimally solving the problem in digraphs whose underlying graphs are trees. Finally, we discuss a related problem, where the task is to clear with a minimum number of agents a subgraph of the underlying graph containing its spanning tree. We show that this problem also admits a 2-approximation in polynomial time.

Keywords: clearing paths, NP-hardness, approximation, parametrized complexity.

1 Introduction

Let $D = (V, A)$ be a directed graph whose underlying graph is connected. We say that an agent placed at a vertex of D can *clear* a directed path π in D if and only if it can follow a directed path in D that includes π. The *Agent Clearing Path* problem (ACP) is defined as follows (see Fig. 1).

> Given a directed graph $D = (V, E)$, whose underlying undirected graph is connected, determine a placement of the minimum number α of agents $a_1, \ldots, a_\alpha$ in D such that for any pair of distinct vertices $u, v \in V$, there is a permutation f of $\{1, ..., \alpha\}$ and a path π with endpoints u and v in the underlying graph that is a concatenation of directed paths $\pi_1, \pi_2, ..., \pi_\alpha$ in D, where for $i = 1, ..., \alpha$, the path π_i can be cleared by agent $a_{f(i)}$ or it is empty.

We shall refer to a solution to ACP (as well as any placement of agents) for D as a *placement function* $c\colon S \to V$, where S is a set of agents in the solution, and call the number of agents that solves ACP in D, denoted by $acp(D)$, as the *path clearing number of* D. For simplicity of presentation, we assume that if $D \cong N_1$, that is, D is a trivial 1-vertex graph, then $acp(D) = 1$. And, we shall restrict ourselves only to directed graphs whose underlying graphs are connected.

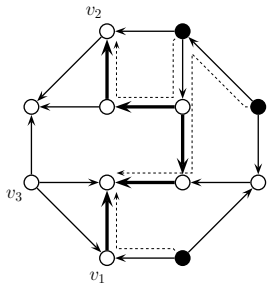

Fig. 1. There is a path with endpoints v_1 and v_2 that is a concatenation of directed paths (marked with (solid arrows) which can be simultaneously cleared by the three agents placed at black vertices, one per each vertex. However, these agents cannot clear any path connecting vertices v_1 and v_3.

The Agent Clearing Path problem seems to have several natural applications. For example, in disaster circumstances, the underground sewage channels in a city can be used as an extraordinary transportation network. The aforementioned channels have slopes (directions) allowing for a continuous flow of sewage. The problem of placing a minimum number of water flushing robots that for any pair of exits could clean a path of channels connecting them can be modelled by a variant of the ACP problem.

Related work. The Agent Clearing Path problem is a variant of the problem of cleaning a graph with brushes, see e.g. [1, 4, 5, 9, 8, 11]. In this problem, given a connected graph, initially with all dirty vertices and edges, a number of agents, called *brushes*, are placed on some vertices of the graph. When a vertex has at least as many brushes as dirty incident edges, it may be cleaned and then 'fired': each dirty incident edge is traversed (i.e. cleaned) by one and only one brush, but brushes can not traverse already cleaned edges. The problem is to determine the initial brush configuration and a corresponding vertex-firing sequence such that the whole graph becomes clean. The minimum number of brushes required to clean a given graph $G = (V, E)$ is denoted by $b(G)$, and for a given integer k, the problem of determining whether $b(G) \leq k$, is NP-complete [4]. Bounds on $b(G)$ are discussed in [9], cleaning random graphs was considered in [1, 8, 11], whereas the parallel version of the model has been studied in [5]. Other variations on this problem have been studied in [6, 10, 12].

Our results. Some elementary properties are discussed in Section 2. In particular, we show that, up to the time of constructing the condensation (acyclic) digraph of a given digraph, we may restrict ourselves to consider only directed acyclic graphs. In Section 3, we provide a simple linear-time 2-approximation for ACP in acyclic digraphs, and an exact algorithm with the running time of $O(sn^3 2^{2s})$, where s is the number of source vertices in the input n-vertex digraph. These results yield a polynomial-time 2-approximation algorithm as well as an $O(\alpha n^3 2^{2\alpha})$-time exact algorithm for ACP for an arbitrary directed graph on n vertices, where α is the number of agents in the solution. We continue our study of ACP in Section 4, with an NP-hardness proof valid even for bipartite acyclic digraphs and in Section 5 with a simple linear-time algorithm optimally solving ACP in digraphs whose underlying graphs are trees. Finally, we discuss a related problem, where the task is to clear with a minimum number of agents a subgraph of the underlying graph containing its spanning tree. We show that this problem also admits a 2-approximation in polynomial time.

Notation. Let $D = (V, A)$ be a directed graph. For two distinct vertices $u, v \in V$, a directed path (resp. walk) connecting u and v (in any direction) is called the (u, v)-*path* (resp. (u, v)-*walk*). For a vertex $v \in V$, $\Pi(v)$ denotes the set of paths in D that can be cleared by an agent placed at v, and $\mathcal{C}(v)$ denotes the strongly connected component in D that v belong to.

2 Preliminaries

The first crucial property that allow us to simplify our analysis is that for an arbitrary digraph $D = (V, A)$, ACP for D can be reduced (in polynomial time) to ACP for its condensation digraph $D^{\downarrow}$. Recall that the *condensation digraph* $D^{\downarrow} = (V^{\downarrow}, A^{\downarrow})$ *of* D is the (acyclic) digraph whose vertices correspond to strongly connected components in D and there is an arc $(\mathcal{C}', \mathcal{C}'') \in A^{\downarrow}$ if and only if there is an arc $(v', v'') \in A$, where v' (resp. v'') is a vertex that belongs to component $\mathcal{C}'$ (resp. $\mathcal{C}''$).

Lemma 1. *For an arbitrary digraph $D = (V, A)$, $acp(D) = acp(D^{\downarrow})$ holds, where $D^{\downarrow}$ is the condensation digraph of D.*

Proof. Due to space limits, we omit the proof.

From now on, taking into account the above lemma, we shall restrict ourselves only to non-trivial directed acyclic graphs (DAGs) whose underlying graphs are connected. Another crucial property is established by the following lemma.

Lemma 2. *For an arbitrary DAG $D = (V, A)$, one may assume without loss of generality that in a solution to ACP for D, all agents are placed at the source vertices of D.*

Proof. It follows from the fact that for any source vertex u such that there exists a directed path from u to v in D, $\Pi(u) \subset \Pi(v)$ holds. □

Consequently, if $s = s(D)$ denotes the number of source vertices in a directed acyclic graph $D = (V, A)$, we have the following corollary.

Corollary 1. *For an arbitrary DAG $D = (V, A)$, we have $acp(D) \geq s$.*

Proof. It follows from Lemma 2 and the fact that at least one agent has to be placed on each source vertex in order to clear the edge leading to a successor of the source vertex. □

3 Approximation and exact algorithms

In this section we propose two algorithms for solving ACP: an approximation of factor 2 and an exact parameterized one. Both are based upon the following lemma. (Recall, for a directed acyclic graph $D = (V, E)$, $s = s(D)$ denotes the number of source vertices in D.)

Lemma 3. *Let $c\colon S \to V$ be a solution to ACP in a DAG $D = (V, A)$. Then, at most two agents are placed at any source vertex s in D, that is, $|c^{-1}(s)| \leq 2$.*

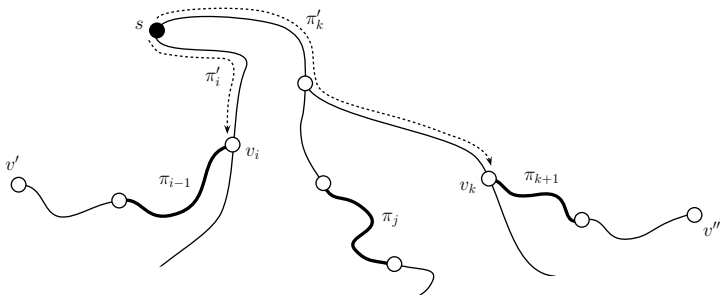

Fig. 2. If three agents at s clears three (directed) subpaths, then there is a shortcut via s that can be cleared only by two agents.

Proof. Suppose on the contrary that $|c^{-1}(s)| > 2$ for some source vertex s in D. Since the placement function c is optimal, there exist two disjoint vertices $v', v'' \in V$ such that any path π connecting v' and v'' requires at least three agents from s. Consider any such path π and assume that π consists of $l \geq 3$ directed paths $\pi_1, \ldots, \pi_l$; set $\pi_0 := \{\{v'\}, \emptyset\}$ and $\pi_{l+1} := \{\{v''\}, \emptyset\}$. Then there exist three paths π_i, π_j, π_k, $1 \leq i < j < k \leq l$, that are cleared by three agents, say a_i, a_j and a_k, from s; assume that i is minimal and k is maximal with respect to this property. Let $\pi'_i \subset \pi_i$ be a (s, v_i)-path in D, where v_i is the vertex that π_{i-1} and π_i have in common, and let $\pi'_k \subset \pi_k$ be a (s, v_k)-path in D, where v_k is the vertex that π_k and π_{k+1} have in common, see Fig. 2 for an illustration.

Then the path π can be replaced with the path π' that consists of $l' < l$ paths $\pi_1, \ldots, \pi_{i-1}, \pi'_i, \pi'_k, \pi_{k+1}, \ldots, \pi_l$ and uses only two agents a_i and a_k from s.

By applying a similar argument to any source vertex s such that $|c^{-1}(s)| > 2$, we obtain a contradiction with minimality of c. □

The above lemma immediately results in the following upper bound on the path clearing number.

Corollary 2. *For an arbitrary directed acyclic graph $D = (V, A)$, we have $acp(D) \leq 2s(D)$.*

3.1 2-approximation algorithm

Taking into account Corollary 1 and the proof of Lemma 3, by placing two agents at each source vertex in D, we obtain a simple 2-approximation algorithm to ACP for D.

Theorem 1. *There exists a linear-time 2-approximation algorithm to ACP in directed acyclic graphs.*

3.2 Exact parameterized algorithm

Keeping in mind Corollary 1 and Lemma 3, following the dynamic programming based approach for the Traveling Salesman Problem [7], the idea is to try all possible placements for at most $s = s(D)$ additional agents at s source vertices in a directed acyclic graph $D = (V, A)$ and check, whether a given placement is *valid*, that is, for any pair of distinct vertices $u, v \in V$, there is a path π with endpoints v and u that is a concatenation of directed paths in D which can be simultaneously cleared by the agents. The valid placement that uses the fewest number of agents constitutes a solution to ACP in D.

Theorem 2. *A solution to ACP in a directed acyclic graph $D = (V, A)$ on n vertices can be computed in $O(sn^3 2^{2s})$ time, where $s = s(D)$.*

Proof. Consider $s = s(D)$ pairs of agents, numbered 1 through $2s$, on the s source vertices of the input directed acyclic graph $D = (V, A)$, respectively. Assume that $V = \{1, 2, \ldots, n\}$. In $O(sn^3)$ time, form an edge-colored multigraph $M = (V, E)$ such that an edge $\{i, j\}$ with color c in $\{1, ..., 2s\}$ occurs in M if and only if the c-th agent can traverse some (i, j)-path. Let $M(c)$ denote the subset of edges in M colored with c. (When constructing the graph M, we also pre-compute the relevant data that allows answering the following query in a constant time: "For a fixed color $c \in C$, does $\{i, j\} \in M(c)$?")

Now, for a subset $C \subseteq \{1, ..., 2s\}$ of colors and two vertices $i, k \in V$, let $P(i, C, k)$ be the (sub)problem of determining whether there is a path connecting i with k in D that can be cleared by agents from C, or equivalently, whether there is a path connecting i with k in M colored with distinct colors. The dynamic programming recursion (Bellman equation) is defined as follows. The base of the

recursion is $C = \{c\}$: $f(i, C, k) =$ 'yes' if $\{i, k\} \in M(c)$; otherwise, $f(i, C, k) =$ 'no'. Next, for a set of colors C, $|C| \geq 2$, we define

$$P(i, C, k) = \bigvee_{j \in V} \bigvee_{c \in C} P(i, C \setminus \{c\}, j) \wedge (\{j, k\} \in M(c)).$$

Clearly, there are $O(n^2 2^{2s})$ subproblems $P(i, C, k)$. If C is a singleton set, $P(i, C, k)$ can be solved in $O(1)$ time using the information gathered during the construction of the graph M. If $|C| > 1$ then $P(i, C, k)$ can be solved by the aforementioned recurrence on the basis of solutions to the subproblems with smaller color subsets in $O(sn)$ time. Thus, the overall time required to solve the subproblems is $O(sn^3 2^{2s})$.

Afterwards, in the order of increasing size of $C \subseteq \{1, ..., 2s\}$, we check if for all $i, k \in V$, and the given C, the subproblems $P(i, C, k)$ are answered positively. If so, we can answer that $|C|$ agents are sufficient to solve ACP for D. Overall, this postprocessing takes $O(n^2 2^{2s})$ time. □

By combining Lemma 1 and Corollary 1 with Theorem 2, we obtain the following parametrized upper bound on ACP.

Corollary 3. *A solution to ACP in a directed graph on n vertices can be computed in $O(\alpha n^3 2^{2\alpha})$ time, where α is the size of the solution, i.e., the number of agents in the solution.*

4 NP-hardness

In this section we present a proof of NP-hardness of ACP. Our proof is a reduction from the connected set cover problem (CSC) [13, 14], which is known to be NP-hard; its decision version can be formulated as follows.

Let V be a finite set of elements, let $\mathcal{F}$ be a family of non-empty subsets of V, and let $G = (\mathcal{F}, \mathcal{E})$ be a graph. A *connected set cover* $\mathcal{S} \subseteq \mathcal{F}$ is a set cover of V such that $\mathcal{S}$ induces a connected subgraph of G. The size of $\mathcal{S}$, that is, the number of sets in $\mathcal{S}$, is denoted by $|\mathcal{S}|$.

The connected set cover problem (CSC)
Given a triple $(V, \mathcal{F}, G)$ and a positive integer k, does there exists a connected set cover of size at most k?

The connected set cover problem is NP-complete even if at most one vertex of the auxiliary graph G has degree greater than two [14]. In addition, the reduction of the set cover problem to the connected set cover problem in [14] implies that the variant of CSC when the adjacency relation $\mathcal{E}$ in the auxiliary graph G is determined by having an element in common, that is, $\{S', S''\} \in \mathcal{E}$ if and only if $S' \cap S'' \neq \emptyset$, is also NP-complete. We shall use this fact in our proof of NP-hardness of ACP.

Specifically, let V be a set of $m \geq 2$ elements, and let $\mathcal{F}$ be a family of subsets of V. Define a graph $G = (\mathcal{F}, \mathcal{E})$ where two disjoint vertices/sets $S', S'' \in \mathcal{F}$ are

adjacent if and only if $S' \cap S'' \neq \emptyset$ (Fig. 3(a)). We also define a bipartite DAG $D = (\mathcal{F} \cup V, A)$, with the elements of $\mathcal{F}$ on one side and the elements of V on the other (Fig. 3(b)); we have that for any $(S, v) \in \mathcal{F} \times V$, $(S, v) \in A$, if and only if $v \in S$. Observe that all source vertices in D correspond to sets in $\mathcal{F}$ ($s(D) = |\mathcal{F}|$), and we shall use the terminology 'vertex' and 'set' (in G or D) interchangeably. The following lemma is crucial.

Lemma 4. *Given a positive integer k, there exists a connected set cover of size at most k for the triple $(V, \mathcal{F}, G)$ if and only if $acp(D) \leq s(D) + k$.*

Proof. The direct implication. Let $\mathcal{S}$ be a connected set cover for $(V, \mathcal{F}, G)$ with $|\mathcal{S}| \leq k$. Define a placement function c: $\{a_1, \ldots, a_{s(D)+|\mathcal{S}|}\} \rightarrow (\mathcal{F} \cup V)$ as follows: place $s(D)$ agents $a_1, \ldots, a_{s(D)}$ at all source vertices of D, and $|\mathcal{S}|$ agents $a_{s(D)+1}, \ldots, a_{s(D)+|\mathcal{S}|}$ at the source vertices of D that constitute $\mathcal{S}$ (Fig. 3(c)). We claim that the placement function c is valid, that is, for any pair of distinct vertices $u, v \in \mathcal{F} \cup V$, there is a path π with endpoints v and u that is a concatenation of directed paths in D which can be simultaneously cleared by the agents. (And thus $acp(D) \leq s(D) + k$.)

Consider a pair of distinct vertices $u, v \in \mathcal{F} \cup V$. There are three cases to consider.

- Case 1: $u \in V$ and $v \in V$ (Fig. 3(d). Since $\mathcal{S}$ is a connected set cover of V, taking into account the definition of D, there exists a path π in D connecting u and v that consists of arcs $(s_1, u), (s_1, v_1), (s_2, v_1), \ldots, (s_{l-1}, v_{l-1}), (s_l, v_{l-1}), (s_l, v)$, $l \geq 1$, where $v_i \in V$, $i = 1, \ldots, l-1$, and $s_i \in \mathcal{S}$, $i = 1, \ldots, l$. And, since $|c^{-1}(s_i)| = 2$, $i = 1, \ldots, l$, path π can be cleared by agents placed at source vertices $s_1, \ldots, s_l$.
- Case 2: $u \in V$ and $v \in \mathcal{F}$ (Fig. 3(e)). By the same argument as above, taking into account that each set in $\mathcal{F} \setminus \mathcal{S}$ has an element in common with some set in $\mathcal{S}$, there exists a path π in D connecting u and v that consists of arcs $(s_1, u), (s_1, v_1), (s_2, v_1), (s_2, v_2), \ldots, (s_l, v_{l-1}), (s_l, v_l), (v, v_l)$, where $v_i \in V$ and $s_i \in \mathcal{S}$, $i = 1, \ldots, l$. Again, since $|c^{-1}(v)| \geq 1$, and $|c^{-1}(s_i)| = 2$, $i = 1, \ldots, l$, path π can be cleared by agents placed at source vertices $s_1, \ldots, s_l$ and v.
- Case 3: $u \in \mathcal{F}$ and $v \in \mathcal{F}$ (Fig. 3(f)). By the same argument as in Case 2, there exists a path π in D connecting u and v that consists of a sequence of arcs $(u, v_1), (s_1, v_1), (s_1, v_2), \ldots, (s_l, v_l), (s_l, v_{l+1}), (v, v_{l+1})$, where $v_i \in V$, $i = 1, \ldots, l+1$, and $s_i \in \mathcal{S}$, $i = 1, \ldots, l$, $l \geq 0$. And, since $|c^{-1}(u)| \geq 1$ and $|c^{-1}(v)| \geq 1$, and $|c^{-1}(s_i)| = 2$, $i = 1, \ldots, l$, path π can be cleared by agents $a_1, \ldots, a_{s(D)+|\mathcal{C}|}$ placed at source vertices $s_1, \ldots, s_l$ and source vertices u and v.

Consequently, the placement function c is valid, and thus $acp(D) \leq s(D) + k$ as required.

The converse implication. Let c: $S \rightarrow (\mathcal{S} \cup V)$ be a solution to ACP in D. By Lemma 2, we may assume that all agents are placed at source vertices of D.

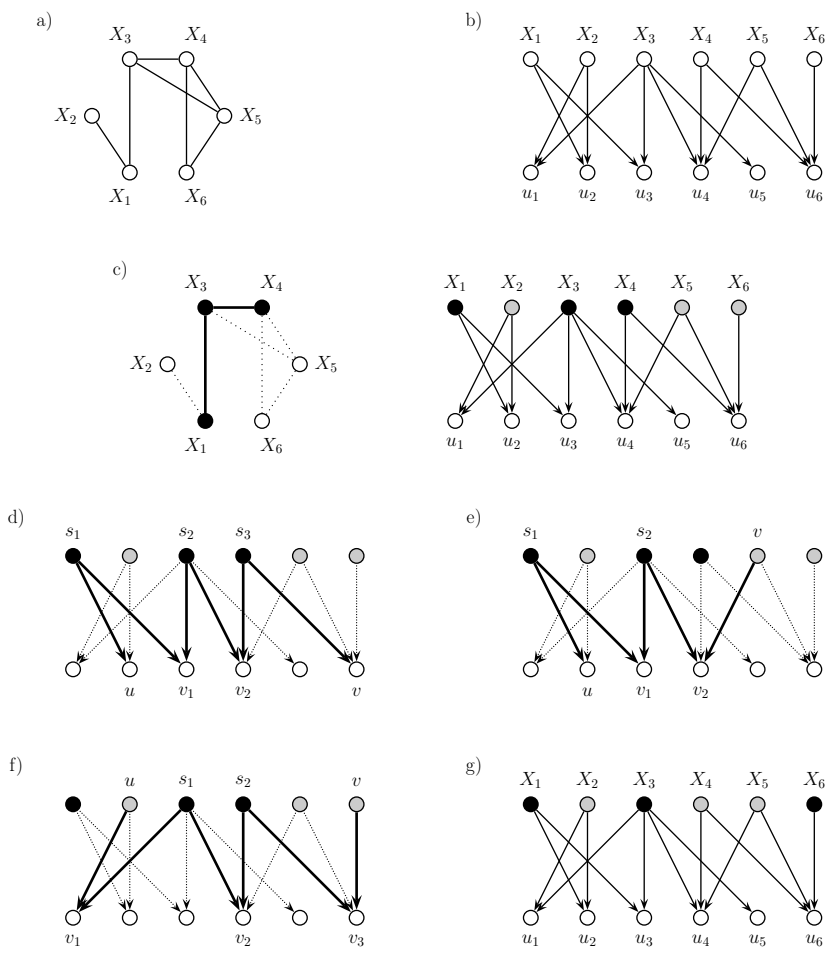

Fig. 3. $V = \{u_1, u_2, \ldots, u_6\}$, $\mathcal{F} = \{X_1, X_2, \ldots, X_3\}$, where $X_1 = \{u_2, u_3\}$, $X_2 = \{u_1, u_2\}$, $X_3 = \{u_1, u_3, u_4, u_5\}$, $X_4 = \{u_4, u_6\}$, $X_5 = \{u_4, u_6\}$, and $X_6 = \{u_6\}$. (a) The graph $G = (\mathcal{F}, \mathcal{E})$, where two disjoint vertices/sets $X', X'' \in \mathcal{F}$ are adjacent if and only if $X' \cap X'' \neq \emptyset$. (b) The bipartite digraph $D = (\mathcal{F} \cup V, A)$. (c) A connected set cover $\mathcal{S}$ for $(V, \mathcal{F}, G)$ and the relevant agent placement in D: there is one agent at any gray vertex, and two agents at any black vertex. (d) Case 1: clearing path between $u \in V$ and $v \in V$. (e) Case 2: clearing path between $u \in V$ and $v \in \mathcal{F}$. (f) Case 3: clearing path between $u \in \mathcal{F}$ and $v \in \mathcal{F}$. (g) If the set cover $\mathcal{S}$ is not connected, then placing two agents only at vertices corresponding the sets in $\mathcal{S}$ (and one agent at any other source vertex) does not imply a solution to ACP in D: here, $\{X_1, X_3, X_6\}$ is a set cover for $(V, \mathcal{F})$, and there is no path between u_1 and u_6 that can be cleared by agents.

Define $\mathcal{S} := \{s \in \mathcal{F} : |c^{-1}(s)| = 2\}$; notice that $|\mathcal{S}| = k$ by Corollary 1 and Lemma 3. We claim that $\mathcal{S}$ is a connected set cover for $(V, \mathcal{F}, G)$.

Consider an element $v \in V$. Since c solves ACP, there exists a path π connecting v and some $u \in V$, $u \neq v$, in D that can be cleared by agents. By the definition of graph D, π must visit a source vertex $s \in \mathcal{F}$, more precisely, π must traverse two arcs (s, u') and (s, u''). Since these arcs can be only cleared from s, we have $|c^{-1}(s)| = 2$, and thus $s \in \mathcal{S}$. Consequently, v is covered by $\mathcal{S}$, and since v is an arbitrary element in V, $\mathcal{F}$ is a set cover of V.

To finalize the proof, we have to show that $\mathcal{S}$ is connected. Suppose on the contrary that $\mathcal{S}$ is not connected. W.l.o.g. assume that the induced graph $G[\mathcal{S}]$ has two connected components, induced by two disjoint families $\mathcal{S}_1$ and $\mathcal{S}_2$, $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$. Consider $S_1 \in \mathcal{S}_1$ and $S_2 \in \mathcal{S}_2$. Again, since c solves ACP, there exists a path π connecting sets/vertices S_1 and S_2 in D that can be cleared by agents. By the definition of the graph D, π consists of a sequence of arcs $(S_1, v_1), (s_1, v_1), (s_1, v_2), \ldots, (s_l, v_l)$, $(s_l, v_{l+1}), (S_2, v_{l+1})$, where $v_i \in V$, $i = 1, \ldots, l+1$, and $s_i \in \mathcal{S}$, $i = 1, \ldots, l$. Since $|c^{-1}(s_i)| = 2$, we have $s_i \in \mathcal{S}$, $i = 1, \ldots, l$. And, since $v_1 \in S_1 \cap s_1$, $v_i \in s_{i-1} \cap s_i$, $i = 2, \ldots, l$, and $v_{l+1} \in S_2 \cap s_l$, there exists a path connecting S_1 and S_2 in $G[\mathcal{S}]$ — a contradiction with S_1 and S_2 lying in two different connected components of $G[\mathcal{S}]$. Consequently, $G[\mathcal{S}]$ is connected, and thus $\mathcal{S}$ is a connected set cover for $(V, \mathcal{F}, G)$ of size k. □

With the result of Lemma 4, we obtain the following theorem.

Theorem 3. *ACP in bipartite directed acyclic graphs is NP-hard.*

5 Trees

In view of the NP-hardness of ACP for general directed graphs, it is natural and interesting to analyze the complexity of ACP for dags whose underlying undirected graph are trees. We shall term such dags *tree dags*.

Theorem 4. *ACP for tree dags can be solved in linear time.*

Proof. Recall, for a vertex $v \in V$ in a directed graph $D = (V, A)$, $N^-(v)$ (resp. $N^+(v)$) denotes the set of all vertices $u \in V$ such that $(u, v) \in A$ (resp. $(v, u) \in A$); the number of elements in $N^-(v)$, denoted by $\deg_{\text{in}}(v)$, is called the *indegree* of v, while the number of element in $N^+(v)$, denoted by $\deg_{\text{out}}(v)$, is called the *outdegree* of v.

For a tree dag $T = (V, A)$, let $l(T)$ and $s(T)$ be the set of leaves and the set of source vertices in T, respectively. Suppose that $c : S \to s(T)$ is a solution to ACP in T, where S is the set of agents, with $|S| = acp(T)$. Since T is a tree dag, observe that for any $s \in s(T)$, if $\deg_{\text{out}}(s) \geq 2$ then c must place two agents at s, that is, $|c^{-1}(s)| = 2$; otherwise, $|c^{-1}(s)| \in \{1, 2\}$ (by Corollary 1 and Lemma 3). Consequently, to compute $acp(T)$, all we need is to determine all such source vertices s in $s(T) \cap l(T)$ such that for any solution c to ACP in T, $|c^{-1}(s)| = 2$ holds.

Consider a source vertex $s \in s(T) \cap l(T)$. A vertex $v \in \Pi(s)$ is called *essential with respect to* s if $\deg_{\text{out}}(v) \geq 2$ and all vertices on the (unique directed) (s, v)-path in T are of indegree at most one. We need the following lemma.

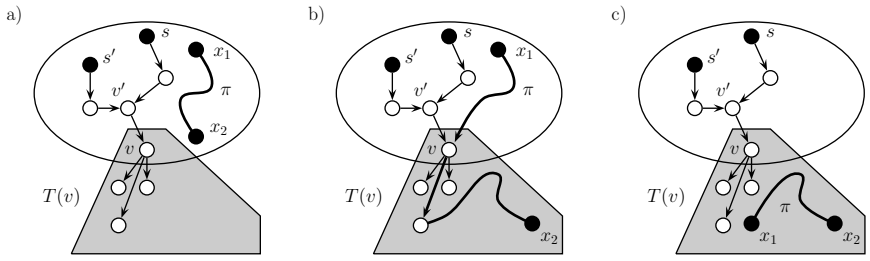

Fig. 4. Lemma 5, Case 2.

Lemma 5. *Any solution to ACP in T places two agents at s if and only if there is an essential vertex with respect to s.*

Proof (of Lemma). The direct implication. Suppose on the contrary that there exists a solution c to ACP in T that places two agents a_1 and a_2 at some $s \in s(T) \cap l(T)$ such that there is no essential vertex with respect to s. We claim that c is not optimal, i.e., one of the agents at s is superfluous.

Case 1: Vertices in $\Pi(s)$ form a directed path, that is, there is no vertex $v \in \Pi(s) \subseteq V \setminus s(T)$ such that $\deg_{\text{out}}(v) \geq 2$. Then, in any path connecting two vertices x_1 and x_2 in T that can be cleared by agents, there is at most one directed path induced by some vertices in $\Pi(s)$, which requires only one agent at s — a contradiction to the optimality of c.

Case 2: There is a vertex $v \in \Pi(s) \subseteq V \setminus s(T)$ such that $\deg_{\text{out}}(v) \geq 2$. (Notice $v \neq s$.) Since v is not essential with respect to s (according to our assumption), there is a vertex v' on the (s, v)-path, $v' \neq s$ such that $\deg_{\text{in}}(v') \geq 2$. Since T is a tree dag, there is a source $s' \in s(T)$, $s' \neq s$, such that $v' \in \Pi(s')$ and hence $v \in \Pi(s')$. (Fig. 4.) Let a' be one of the agents placed at s'.

Deleting all arcs $(u, v) \in A$ results in several tree subdags of T; let $T(v) = (V_v, A_v)$ denote the relevant tree subdag such that $v \in V_v$. Observe that v is a source vertex in $T(v)$. Consider now some (unique) path π between two vertices x_1 and x_2 in T that can be cleared with agents placed by our placement function c.

Subcase 2.*a*: $x_1 \notin V_v$ and $x_2 \notin V_v$ (Fig. 4(a)). By the choice of v, all but v vertices on the unique (s, v)-path in T are of outdegree one, and thus, vertices in $\Pi(s)$ contribute at most one directed path in π, which requires only one agent at s.

Subcase 2.*b*: either $x_1 \in V_v$ and $x_2 \notin V_v$ or $x_1 \notin V_v$ and $x_2 \in V_v$ (Fig. 4(b)). Similarly as above, since all but v vertices on the unique (s, v)-path in T are of outdegree one, vertices in $\Pi(s)$ contribute at most one directed path in π, which requires only one agent at s.

Subcase 2.*c*: $x_1 \in V_v$ and $x_2 \in V_v$ (Fig. 4(c)). Since agent a' can reach v only through vertices in $V \setminus V_v$, vertex v is "supported" with at least

three agents (a_1, a_2 and a') that may take a part in clearing $\pi \subseteq T(v)$. Since T is a tree dag, vertices in $\Pi(v) \subseteq V_v$ contribute only at most two directed paths in π, and thus a_2 is useless for clearing π.

Consequently, in Case 2 as well, any solution to ACP requires only one agent at s — a contradiction to the optimality of c.

The converse implication. It follows from the fact that T is a tree dag. Namely, consider the essential vertex v with respect to $s \in s(T) \cap l(T)$. Any path π connecting two successors v' and v'' of v that can be cleared by agents must use arcs $e' = (v, v')$ and $e'' = (v, v'')$, respectively. Since all vertices on the (unique directed) (s, v)-path are of inner degree at most one, the arcs e' and e'' can be cleared only from agent(s) in s (by Lemma 2), which requires two agents at s. □

Continuing the proof of Theorem 4.
Given Lemma 5, all we need to determine a solution to ACP in T is to check whether there is an essential vertex with respect to s, for each $s \in s(T) \cap l(T)$. This can be done by a standard DFS-based approach, starting from any element in $s(T) \cap l(T)$. □

6 Extentions

A natural extension of ACP, more closely related to the problem of cleaning a graph with brushes, is the following variant where we want to clear some connections between all vertices.

The Agent Clearing Tree problem (ACT)

Given a directed graph $D = (V, A)$ whose underlying graph $G = (V, E)$ is connected, determine a placement of minimum number of agents in D such that agents can simultaneously clear some subgraph of D whose underlying graph includes a spanning tree of G.

The complexity status of ACT remains open, however, there is a simple 2-approximation algorithm for solving ACT.

Theorem 5. *For a given n-vertex DAG $D = (V, A)$, ACT is 2-approximable in polynomial time.*

Proof. Let Π be a minimum path cover of D. (A *path cover Π of D* is a set of directed paths in D such that for every $v \in V$, there exists at least one path $\pi \in \Pi$ visiting v.) Recall that by considering the reflexive transitive closure of D, Π can be computed in polynomial time by reduction to the maximum matching problem in a bipartite graph [3]. Initially, set $\mathcal{S} = \Pi$. Now, since the underlying graph of D is connected and paths in Π visits all vertices in V, by adding at most $|\Pi| - 1$ single arcs to $\mathcal{S}$, we obtain a set of at most $2|\Pi| - 1$ directed paths that constitute the required connected spanning subgraph of D, and so at most $2|\Pi| - 1$ are enough to solve ACT in D. On the other hand, any solution for ACT uses at least $|\Pi|$ agents, which concludes the proof of the theorem. □

7 Final remarks

There are several interesting generalizations, variants of ACP and ACT and problems related to them. For instance, for each placement of an agent on the underlying graph, one could specify a set of paths that can be cleared by the agent and then ask for a minimum number of agents that for any given pair of vertices could clear (not necessarily in the directed fashion) a path between them, or that could clear a spanning tree of the underlying graph.

Also, the complexity status of ACP and ACT and their variants where the underlying graph is restricted to some special graph class substantially larger than trees, e.g., graphs of bounded treewidth, are interesting open problems.

Acknowledgments. The authors thank Adrian Kosowski for valuable remarks and interesting discussions on the topic.

References

1. N. Alon, P. Prałat, N. Wormald, Cleaning regular graphs with brushes, *SIAM Journal on Discrete Mathematics* 23(1), pp. 233-250 (2008).
2. E. W. Dijkstra, A note on two problems in connexion with graphs, *Numerische Mathematik* 1, pp. 269-271 (1959).
3. L. R. Ford, D. R. Fulkerson, Flows in Networks, Princeton University Press (1962).
4. S. Gaspers, M.-E. Messinger, R. J. Nowakowski, P. Prałat, Clean the graph before you draw it!, *Information Processing Letters* 109(10), pp. 463-467 (2009).
5. S. Gaspers, M.-E. Messinger, R. J. Nowakowski, P. Prałat, Parallel cleaning of a network with brushes, *Discrete Applied Mathematics* 158(5), pp. 467-478 (2010).
6. P. Gordinowicz, R. J. Nowakowski, P. Prałat, Polish — Let us play the cleaning game, *Theoretical Computer Science* 463, pp. 123-132 (2012).
7. M. Held, R. M. Karp, A dynamic programming approach to sequencing problems, *Journal of the Society for Industrial and Applied Mathematics* 10(1), pp. 196-210 (1961).
8. M.-E. Messinger, R. J. Nowakowski, P. Prałat, N. Wormald, Cleaning random d-regular graphs with brushes using a degree-greedy algorithm, Proceedings of the 4th Workshop on Combinatorial and Algorithmic Aspects of Networking, *Lecture Notes in Computer Science* 4852, pp. 13-26 (2007).
9. M.-E. Messinger, R. J. Nowakowski, P. Prałat, Cleaning a network with brushes, *Theoretical Computer Science* 399, pp. 191-205 (2008).
10. M.-E. Messinger, R. J. Nowakowski, P. Prałat, Cleaning with Brooms, *Graphs and Combinatorics* 27(2), pp. 251-267 (2011).
11. P. Prałat, Cleaning random graphs with brushes, *Australasian Journal of Combinatorics* 43, pp. 237-251 (2009).
12. P. Prałat, Cleaning random d-regular graphs with Brooms, *Graphs and Combinatorics* 27(4), pp. 567-584 (2011).
13. W. Ren, Q. Zhao, A note on Algorithms for connected set cover problem and fault-tolerant connected set cover problem, *Theoretical Computer Science* 412(45), pp. 6451-6454 (2011).
14. T. P. Shuai, X.-D. Hu, Connected set cover problem and its applications, Proceeding of the 2nd International Conference on Algorithmic Aspects in Information and Management, *Lecture Notes in Computer Science* 4041, pp. 243-254 (2006).