Enumeration of s-d Separators in DAGs with Application to Reliability Analysis in Temporal Graphs

Alessio Conte
Università degli Studi di Pisa, Dipartimento di Informatica, Italy
conte@di.unipi.it

Pierluigi Crescenzi
Université de Paris, IRIF, CNRS, France
On-leave from Università degli Studi di Firenze, DiMaI, Firenze, Italy
pierluigi.crescenzi@irif.fr

Andrea Marino
Università degli Studi di Firenze, DiSIA, Firenze, Italy
andrea.marino@unifi.it

Giulia Punzi
Università degli Studi di Pisa, Dipartimento di Informatica, Italy
giulia.punzi@phd.unipi.it

Abstract

Temporal graphs are graphs in which arcs have temporal labels, specifying at which time they can be traversed. Motivated by recent results concerning the reliability analysis of a temporal graph through the enumeration of minimal cutsets in the corresponding line graph, in this paper we attack the problem of enumerating minimal s-d separators in s-d directed acyclic graphs (in short, s-d DAGs), also known as 2-terminal DAGs or s-t digraphs. Our main result is an algorithm for enumerating all the minimal s-d separators in a DAG with $O(nm)$ delay, where n and m are respectively the number of nodes and arcs, and the delay is the time between the output of two consecutive solutions. To this aim, we give a characterization of the minimal s-d separators in a DAG through vertex cuts of an expanded version of the DAG itself. As a consequence of our main result, we provide an algorithm for enumerating all the minimal s-d cutsets in a temporal graph with delay $O(m^3)$, where m is the number of temporal arcs.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases minimal cutset, temporal graph, minimal separator, directed acyclic graph

Digital Object Identifier 10.4230/LIPIcs.MFCS.2020.25

Funding A.C. and A.M. have been partially supported by MIUR under PRIN Project AHeAD (Efficient Algorithms for HArnessing Networked Data). A.M. has been partially supported by University of Florence under Project GRANTED (GRaph Algorithms for Networked TEmporal Data).

Acknowledgements We would like to thank Gaurav Khanna and Lhouari Nourine for several useful discussion by e-mail.

1 Introduction

A fundamental task to be accomplished while analysing the reliability of a network consists of designing, analysing, and implementing efficient algorithms for the extraction of all minimal cutsets [22]. Formally, given a (directed) graph $G = (V, A)$ and two nodes $s, d \in V$, a *minimal s-d cutset* $S \subseteq A$ is a minimal set of arcs whose removal disconnects s from d. The enumeration of minimal cutsets in graphs has been a very active research area since the end

of the sixties [11]. It is now known that listing minimal cutsets can be done, with $O(m)$ delay, both in undirected [25] and directed graphs [21] (thanks to the characterization given in [24]), where m denotes the number of arcs of the graph and the delay is the time between the output of a minimal cutset and the successive one (reporting all the results obtained in this field is clearly out of the scope of this paper).

Recently [7, 12], the minimal cutset technique for determining the reliability of a complex system has been also applied to the case of temporal graphs (in particular, to the case of delay tolerant networks [9, 13]). Several definitions of temporal graphs have been introduced in the literature, even with different notation and terminology (see, for example, [5, 6, 15]): here, we mostly refer to the definitions of [18]. A *temporal graph* is a pair $G = (V, A)$, where V is the set of n *nodes* and E is the set of m *temporal arcs*. A temporal arc $a \in A$ is a triple (u, v, λ), where $u, v \in V$ are the tail and head nodes of the arc, respectively, and $\lambda \in \mathbb{N}$ is its *appearing time*. For example, the temporal graph G in the left part of Figure 1 has five nodes and nine temporal arcs (which are listed below the graph). A *temporal u-v path* in a temporal graph $G = (V, A)$ from a node $u \in V$ to a node $v \in V$ is a sequence $\langle (u, x_1, \lambda_1), (x_1, x_2, \lambda_2), \ldots, (x_{k-1}, v, \lambda_k) \rangle$ of temporal arcs in A such that, for each i with $1 < i \leq k$, $\lambda_i > \lambda_{i-1}$. For example, the temporal graph G in the left part of Figure 1 contains the following four temporal s-d paths: $\pi_1 = \langle (s, a, 1), (a, d, 4) \rangle$, $\pi_2 = \langle (s, a, 1), (a, c, 2), (c, d, 3) \rangle$, $\pi_3 = \langle (s, b, 2), (b, d, 4) \rangle$, and $\pi_4 = \langle (s, b, 2), (b, a, 3), (a, d, 4) \rangle$. We can now adapt the definition of minimal cutset to the case of temporal graphs. Given an s-d temporal graph $G = (V, A)$, a *minimal s-d cutset* is a minimal set $S \subseteq A$ of temporal arcs of G, whose removal breaks all temporal s-d paths. For example, with respect to the temporal graph G in the left part of Figure 1, we have that $S = \{(s, a, 1), (b, a, 3), (b, d, 4)\}$ is a minimal s-d cutset.

In [12], the authors propose to enumerate all minimal s-d cutset in a temporal graph G, by first transforming G into the corresponding s-d line graph (as suggested in [16]). Given a temporal graph $G = (V, A)$ and two nodes $s, d \in V$, its corresponding (static) *s-d line graph* contains a node for each temporal arc $a \in A$, and, for any two nodes $a_1 = (u_1, v_1, \lambda_1)$ and $a_2 = (u_2, v_2, \lambda_2)$, contains the arc (a_1, a_2) if and only if $v_1 = u_2$ and $\lambda_2 > \lambda_1$ (with a little abuse of notation, we denote by the same symbol the temporal arc of the temporal graph and the corresponding node of the corresponding static line graph). Moreover, the line graph contains two nodes s and d, and, for any arc $a = (s, u, \lambda) \in A$ (respectively, $a = (u, d, \lambda) \in A$), it contains the arc (s, a) (respectively, (a, d)). In the middle part of Figure 1, we show the s-d line graph corresponding to the temporal graph in the left part of the figure. It is easy to verify that the s-d line graph of a temporal graph is an *s-d directed acyclic graph* (in short, *s-d DAG*) (also called, in the literature, *st*-digraph [4] or 2-terminal DAG [8]).

Given a (directed) graph $G = (V, A)$ and two nodes $s, d \in V$, a *minimal s-d separator* $S \subseteq V$ is a minimal set of nodes which disconnects s from d. From the definition of line graph, it follows that there is a one-to-one correspondence between the minimal s-d cutsets in a temporal graph and the minimal s-d separators in the corresponding s-d line graph. For example, the s-d line graph in the center of Figure 1 contains the following seven minimal s-d separators: $\{a_1, a_2\}$, $\{a_1, a_4, a_8\}$, $\{a_1, a_7, a_8\}$, $\{a_2, a_3, a_7\}$, $\{a_2, a_6, a_7\}$, $\{a_3, a_7, a_8\}$, and $\{a_6, a_7, a_8\}$. By using the list of arcs, it is easy to find the seven corresponding minimal cutsets in the temporal graph in the left part of the figure: for example, the minimal s-d separator $\{a_1, a_4, a_8\}$ corresponds to the minimal cutset $\{(s, a, 1), (b, a, 3), (b, d, 4)\}$. The problem of enumerating the minimal s-d cutsets in a temporal graph has thus been reduced to the problem of **enumerating the minimal s-d separators in a DAG**. In [12], the authors solve this problem by enumerating all minimal cutsets in the DAG and by then

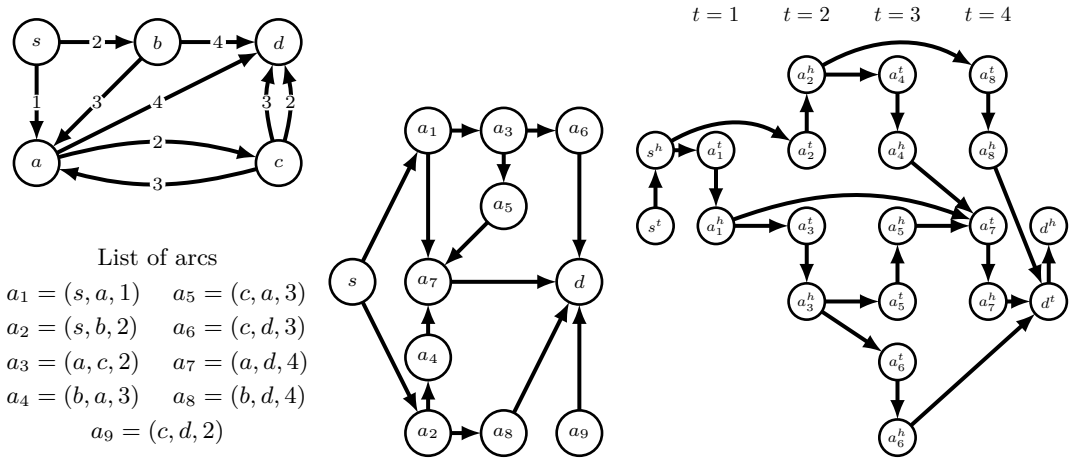

Figure 1 A temporal graph with 5 nodes and 9 temporal arcs (left), its corresponding line graph (center), which is an s-d DAG, and its corresponding expanded s-d DAG (right), without the node a_9 (which cannot participate to any s-d separator).

transforming each cutset into a separator, possibly discarding cutsets that do not correspond to any minimal separator, or that correspond to a minimal separator already listed. This approach, however, does not guarantee any polynomial bound on the delay of the enumeration algorithm. On the other hand, even if there are many results concerning the enumeration of minimal s-d separators in undirected graphs (see [23] for the best delay bound, that is, $O(n^2)$ where n is the number of nodes), as far as we know, no explicit results have been published concerning the enumeration of minimal s-d separators in directed graphs or, more specifically, in DAGs. The techniques of [23] seem to be tailored to undirected graphs and difficult to be applied to directed graphs. It might be that the lattice-based techniques of [3] combined with the algorithm for building lattices described in [20] could be applied to the case of DAGs, but the space complexity of the enumeration algorithm would be exponential in the number of nodes. Moreover, the lattice techniques have been used to enumerate all minimal separators (not just the minimal s-d separators): indeed, in [3], the authors themselves pose as an open problem whether the complexity of the two problems is the same.

Our main result is **an algorithm for enumerating all the minimal s-d separators in a DAG with $O(nm)$ delay**, where m is the number of arcs and n is the number of nodes. To this aim, we basically reduce the problem of enumerating all the minimal s-d separators in a DAG to the problem of enumerating all the minimal s-d cutsets in a DAG, with the additional constraint that the cutsets have to be subsets of a specific set of arcs. More precisely, the enumeration algorithm makes use of a characterization of the minimal s-d separators in a DAG through vertex cuts of an expanded version of the DAG itself. This characterization is similar to the one provided in [24] for enumerating all minimal cutsets in a directed graph.

As a consequence of our main result, we provide **an algorithm for enumerating all the minimal s-d cutsets in a temporal graph with delay $O(m^3)$**, where m is the number of temporal arcs. We also mention that other papers have been devoted to studying other aspects of cuts in temporal graphs, as the complexity of finding one *minimum*, i.e. bounded size, separator [27] and one *minimum* cutset [2]. Our algorithm for enumerating minimal cutsets in temporal graphs contributes to the general field of the enumeration of topological structures in temporal graphs, such as temporal cliques [10, 26], temporal k-plexes [1], temporal paths [14], and bicriteria temporal paths [19].

Notations, definitions, and preliminary results

A *directed graph* is a pair $G = (V, A)$, where V is the set of n *nodes* and A is the set of m *arcs*. An arc $a \in A$ is a pair (u, v), where $u, v \in V$ are the tail and head nodes of the arc, respectively (in the following, we will denote by $\tau(a)$ and $\eta(a)$ the tail and the head of the arc a, respectively). The *out-neighbor* $N_G^+(u)$ (respectively, *in-neighbor* $N_G^-(u)$) of a node $u \in V$ is the set of nodes v such that $(u, v) \in A$ (respectively, $(v, u) \in A$). Given a set $U \subseteq V$, we will denote by $G[U]$ the graph induced by U in G, that is $G[U] = (U, A[U])$, where $A[U] \subseteq A$ and, for any $a \in A$, $a \in A[U]$ if and only if $\tau(a) \in U$ and $\eta(a) \in U$. A u-v *walk* $\mathbb{W}$ from a node $u \in V$ to a node $v \in V$ is a sequence of arcs $\langle a_1, a_2, \ldots, a_k \rangle$ such that $u = \tau(a_1)$, $v = \eta(a_k)$, and, for each i with $1 < i \leq k$, $\tau(a_i) = \eta(a_{i-1})$. A u-v *path* $\mathbb{P}$ is a u-v walk with pairwise distinct nodes. A *directed acyclic graph* (in short, *DAG*) is a directed graph which does not contain any u-u walk, for any node u. Given a DAG $G = (V, A)$ and a node $u \in V$, we will denote by $\texttt{succ}_G(u)$ (respectively, $\texttt{pred}_G(u)$) the set of *successors* (respectively, *predecessors*) of u, that is, the set of nodes v such that G contains a u-v (respectively, v-u) path (note that $\texttt{succ}_G(u) \cap \texttt{pred}_G(u) = \emptyset$).

***s*-*d* DAGs and *s*-*d* separators.** An s-d DAG is a DAG $G = (V, A)$ along with two specified nodes $s, d \in V$ such that (1) there is no arc $a \in A$ such that $\eta(a) = s$ or $\tau(a) = d$, (2) $(s, d) \notin A$, and (3) for any node $u \in V$, there exists an s-u path and a u-d path in G. By applying standard graph visits, given a DAG and two nodes s and d, we can easily compute the corresponding s-d DAG in time $O(m)$. Indeed, after possibly deleting the arc (s, d), it suffices to execute a "forward" traversal from s and a "backward" traversal from d, and to then remove the nodes not "touched" in both traversals. For example, in the case of the DAG of the center part of Figure 1, we have that the node a_9 can be removed, since there is no s-a_9 path in the DAG. Given an s-d DAG $G = (V, A)$ and a subset $S \subseteq V$ of nodes of G, $G \setminus S$ denotes the DAG obtained by removing all nodes in S and all arcs $a \in A$ such that $\{\tau(a), \eta(a)\} \cap S \neq \emptyset$. A subset $S \subseteq V$ is an s-d *separator* of G if $\{s, d\} \cap S = \emptyset$ and there is no s-d path in $G \setminus S$. An s-d separator S is *minimal* if no proper subset of S is an s-d separator. For example, with respect to the s-d DAG G in the center part of Figure 1, we have that $S = \{a_1, a_4, a_7, a_8\}$ is an s-d separator, but it is not minimal, since $S \setminus \{a_4\}$ and $S \setminus \{a_7\}$ are also s-d separators.

Expanded *s*-*d* DAGs. Given an s-d DAG $G = (V, A)$, the *expanded* s-d *DAG* $\mathbb{E}(G) = (\mathbb{E}(V), \mathbb{E}(A))$ contains, for each node $u \in V$, two nodes u^t and u^h connected by the arc (u^t, u^h) and, for any arc $(u, v) \in A$, it contains the arc (u^h, v^t) (in the following, the nodes of type u^t will be called tail nodes, while the others will be called head nodes). For example, in the right part of Figure 1 we show the expanded s-d DAG corresponding to the s-d DAG in the center part of the figure. Given a set $U \subseteq V$ of nodes of G, $U^h \subseteq \mathbb{E}(V)$ is the corresponding set of head nodes in $\mathbb{E}(G)$, that is, $U^h = \{u^h : u \in U\}$. The following result can be easily proved (in the following, given a set Q and a subset $P \subseteq Q$, we will denote by $\overline{P}$ the set $Q \setminus P$).

▶ **Lemma 1.** *Given an s-d DAG $G = (V, A)$, let $\mathbb{E}(G) = (\mathbb{E}(V), \mathbb{E}(A))$ be the corresponding expanded s-d DAG, and let $U \subseteq V$. For any $v \in V$, $v \in \texttt{succ}_{G[\overline{U}]}(s)$ if and only if $v^h \in \texttt{succ}_{\mathbb{E}(G)[\overline{U^h}]}(s^t)$.*

2 Minimal s-d separator characterization through vertex cuts

Our algorithm for enumerating the minimal s-d separators of an s-d DAG G is based on the recursive construction of a set X of nodes of the corresponding expanded s-d DAG $\mathbb{E}(G)$. Such a set X will induce an s-d cut set in $\mathbb{E}(G)$ consisting of all arcs whose tail node is in

X and whose head node is not in X. In order to guarantee that this cut set corresponds to a minimal s-d separator in G, the set X has to satisfy some specific properties: the goal of this section is to formally state these properties and to give a characterization of the minimal s-d separators of an s-d DAG G. To this aim, we will refer to the concept of vertex cut, similarly to what has been done in [24] for listing minimal cut sets. Given an s-d DAG $G = (V, A)$ and its corresponding expanded s-d DAG $\mathbb{E}(G) = (\mathbb{E}(V), \mathbb{E}(A))$, let $X \subseteq \mathbb{E}(V)$. The *vertex cut* $\langle X, \overline{X} \rangle$ in G is defined as $\langle X, \overline{X} \rangle = \{u \in V : u^t \in X \wedge u^h \in \overline{X}\}$. For example, with respect to the expanded s-d DAG in the right part of Figure 1, if we choose $X = \{a_1^t, a_4^t, a_7^t, a_8^t\}$, the corresponding vertex cut in the s-d DAG in the center of the figure is given by $\langle X, \overline{X} \rangle = \{a_1, a_4, a_7, a_8\}$, which is an s-d separator. Clearly, not all vertex cuts $\langle X, \overline{X} \rangle$ are also s-d separators: for instance, if we choose $X = \{a_1^t, a_8^t\}$, the corresponding vertex cut is $\langle X, \overline{X} \rangle = \{a_1, a_8\}$, which is not an s-d separator since, after removing a_1 and a_8, there is still an s-d path passing through nodes a_2, a_4, and a_7.

▶ **Definition 2.** *Given an s-d DAG $G = (V, A)$ and its corresponding expanded s-d DAG $\mathbb{E}(G) = (\mathbb{E}(V), \mathbb{E}(A))$, a subset $X \subseteq \mathbb{E}(V)$ is said to be* ***good*** *if*
G1 *$\{s^t, s^h\} \subseteq X$;*
G2 *$\{d^t, d^h\} \subseteq \overline{X}$; and*
G3 *for any $u^h \in X$, $N^+_{\mathbb{E}(G)}(u^h) \subseteq X$.*

▶ **Lemma 3.** *Given an s-d DAG $G = (V, A)$ and its corresponding expanded s-d DAG $\mathbb{E}(G) = (\mathbb{E}(V), \mathbb{E}(A))$, let $X \subseteq \mathbb{E}(V)$ be good. Then $S = \langle X, \overline{X} \rangle$ is an s-d separator in G.*

Proof. Since $\{s^t, s^h\} \subseteq X$ and $\{d^t, d^h\} \subseteq \overline{X}$, we have that $\{s, d\} \cap S = \emptyset$. Let us suppose, by contradiction, that there exists an s-d path in $G \setminus S$. Let $\langle (s, u_1), (u_1, u_2), \ldots, (u_{k-1}, u_k), (u_k, d) \rangle$ be such a path, where, for each i with $1 \leq i \leq k$, $u_i \notin S$. From the definition of expanded s-d DAG, it follows that any arc (x, y) in G corresponds to a pair $(x^t, x^h)(x^h, y^t)$ of arcs in $\mathbb{E}(E)$. Hence, $\langle (s^t, s^h)(s^h, u_1^t), (u_1^t, u_1^h), (u_1^h, u_2^t), (u_2^t, u_2^h), \ldots, (u_{k-1}^h, u_k^t), (u_k^t, u_k^h), (u_k^h, d^t)(d^t, d^h) \rangle$ is a path in $\mathbb{E}(G)$. Because of property G1 of the goodness definition, we have that $s^h \in X$. Because of property G3 and since $u_1^t \in N^+_{\mathbb{E}(G)}(s^h)$, we have that $u_1^t \in X$. Since $u_1 \notin S$, we have that $u_1^h \in X$. By iterating this process, we can conclude that $u_k^h \in X$. Because of property G3 and since $d^t \in N^+_{\mathbb{E}(G)}(u_k^h)$, we have that $d^t \in X$, contradicting property G2. We can thus conclude that there exists no s-d path in $G \setminus S$, and the lemma has been proved. ◀

Note that, for a vertex set X, being good is not a necessary condition, for the corresponding vertex cut $\langle X, \overline{X} \rangle$, for being an s-d separator. For example, with respect to the expanded s-d DAG in the right part of Figure 1, we have already seen that the set $X = \{a_1^t, a_4^t, a_7^t, a_8^t\}$ corresponds to an s-d separator in the s-d DAG in the center of the figure. However, X is not good, since, for example, it contains neither s^t nor s^h and, hence, it does not satisfy property G1 of the goodness definition. The next result shows that, by adding a few more hypotheses, we can obtain a full characterization of minimal s-d separators in terms of good vertex sets (in the following, $\Gamma(X)$ denotes the "neighborhood" of X in $\overline{X}$, that is, $\Gamma(X) = \{u^h \in \overline{X} : u^t \in X\}$).

▶ **Theorem 4.** *Given an s-d DAG $G = (V, A)$, $S \subseteq V$ is a minimal s-d separator if and only if there exists an $X \subseteq \mathbb{E}(V)$ such that the following conditions are satisfied.*
1. *$S = \langle X, \overline{X} \rangle$ and X is good.*
2. *$\mathtt{succ}_{\mathbb{E}(G)[X]}(s^t) \cup \{s^t\} = X$ (that is, the set of the successors of s^t in the graph induced by X in $\mathbb{E}(G)$ is equal to X).*
3. *$\Gamma(X) \subseteq \mathtt{pred}_{\mathbb{E}(G)[\overline{X}]}(d^h) \cup \{d^h\}$ (that is, the set of the predecessors of d^h in the graph induced by $\overline{X}$ in $\mathbb{E}(G)$ includes $\Gamma(X)$).*

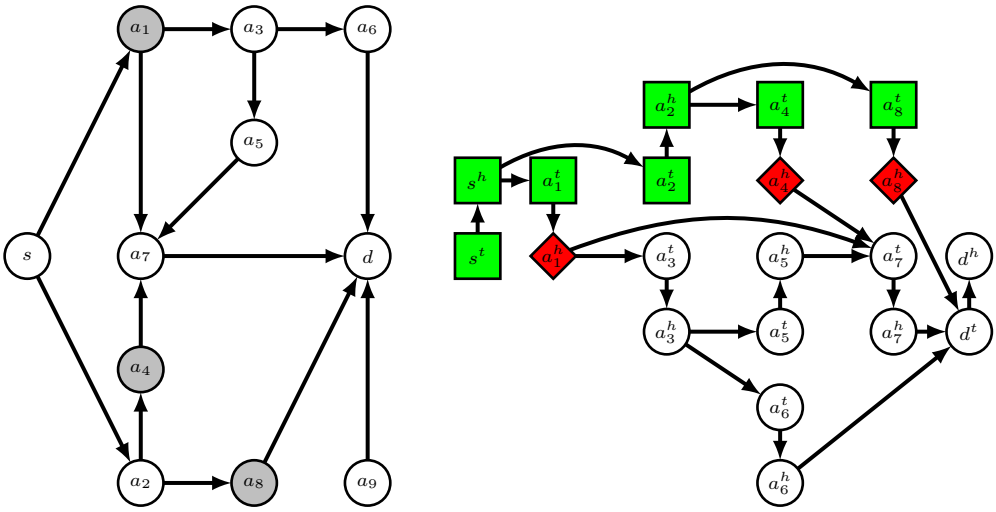

Figure 2 The proof of the necessity in Theorem 4: the gray nodes are in the minimal separator S, the red diamond nodes are in S^h, and the green rectangle nodes are in X (hence, $\overline{X}$ consists of the white circle and red diamond nodes).

Proof. Let us first prove the necessity. To this aim, let $S \subseteq V$ be a minimal s-d separator. We then define $X = \{s^t\} \cup \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$, where $S^h = \{u^h : u \in S\}$ (see Figure 2). We will now prove that this vertex set satisfies the required conditions.

- X is good. Since $s \notin S$, we have that $s^h \notin S^h$: since $(s^t, s^h) \in \mathbb{E}(A)$, it follows that $s^h \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. Hence, both s^t and s^h belongs to X and property G1 is satisfied. Since $d \notin S$, we have that $\{d^t, d^h\} \cap S^h = \emptyset$: if by contradiction $d^t \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$, then $d^h \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$ (since $(d^t, d^h) \in \mathbb{E}(A)$). From Lemma 1, it follows that $d \in \mathtt{succ}_{\mathbb{E}(G)[V \setminus S]}(s)$, contradicting the fact that S is an s-d separator. Hence, neither d^t nor d^h belongs to X, and property G2 is satisfied. Finally, if $u^h \in X$, then $u^h \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. From the definition of S^h and since all out-neighbors of a head node are tail nodes, we have that, for any $u^h \in X$, $N^+_{\mathbb{E}(G)}(u^h) \cap S^h = \emptyset$. Hence, for any $v^t \in N^+_{\mathbb{E}(G)}(u^h)$, we have that $v^t \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$: that is, $v^t \in X$. Hence, $N^+_{\mathbb{E}(G)}(u^h) \subseteq X$ and property G3 is also satisfied. We can conclude that X is good.
- $S = \langle X, \overline{X} \rangle$. First, we show that $S \subseteq \langle X, \overline{X} \rangle$. Let $u \in S$, and, by contradiction, suppose that $u \notin \langle X, \overline{X} \rangle$. Note that by definition of S^h, we have that $u^h \in S^h$ and, hence, that $u^h \notin \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$, that is, $u^h \in \overline{X}$. Since $u \notin \langle X, \overline{X} \rangle$, this implies that $u^t \notin X$, that is, $u^t \notin \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. Since $\mathbb{E}(G)$ is a DAG, there is no u^h-u^t path in $\mathbb{E}(G)$: hence, we have that $u^t \notin \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h} \cup \{u^h\}]}(s^t)$. Since u^t is the only in-neighbor of u^h, this implies that $u^h \notin \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h} \cup \{u^h\}]}(s^t)$. From Lemma 1, it follows that $u \notin \mathtt{succ}_{G[(V \setminus S) \cup \{u\}]}(s)$, that is, reintegrating u in G does not produce any s-d path (since u is not a successor of s in $G[(V \setminus S) \cup \{u\}]$). This contradicts the minimality of S. Hence, u must belong to $\langle X, \overline{X} \rangle$: we have thus proved that $S \subseteq \langle X, \overline{X} \rangle$. In order to prove the opposite inclusion, let $u \in \langle X, \overline{X} \rangle$, that is, $u^t \in X$ and $u^h \in \overline{X}$. By contradiction, suppose that $u \notin S$, which implies that $u^h \in \overline{S^h}$. Since $u^t \in X$, we have that $u^t \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. By using the arc (u^t, u^h), we have that $u^h \in \mathtt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^h)$: that is, $u^h \in X$, which contradicts the fact that $u^h \in \overline{X}$. Thus, we have that $\langle X, \overline{X} \rangle \subseteq S$, and, hence, that $S = \langle X, \overline{X} \rangle$.

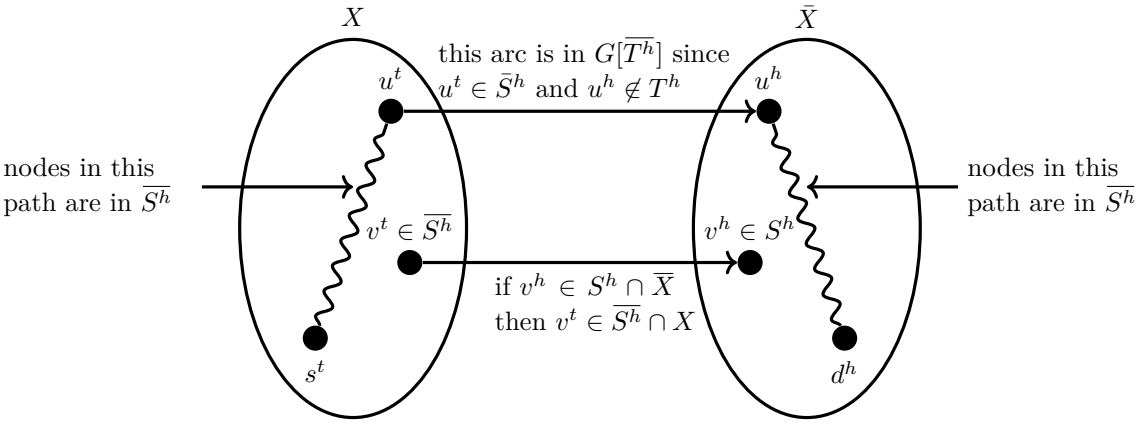

Figure 3 The proof of the sufficiency in Theorem 4.

- $\texttt{succ}_{\mathbb{E}(G)[X]}(s^t) \cup \{s^t\} = X$. Clearly, $\texttt{succ}_{\mathbb{E}(G)[X]}(s^t) \subseteq X$. Let us show the opposite inclusion. By contradiction, suppose that there exists $x \in X \setminus \texttt{succ}_{\mathbb{E}(G)[X]}(s^t)$. By definition of X, there is an s^t-x path in $\mathbb{E}(G)[\overline{S^h}]$. Since $x \notin \texttt{succ}_{\mathbb{E}(G)[X]}(s^t)$, this path must contain a node of $\overline{X}$: let y the first such node. We have then found a node $y \in \overline{X}$ such that $y \in \texttt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t) \subseteq X$, which is a contradiction.
- $\Gamma(X) \subseteq \texttt{pred}_{\mathbb{E}(G)[\overline{X}]}(d^h) \cup \{d^h\}$. Suppose, by contradiction, that there exists $u^h \in \Gamma(X)$ such that $d^h \notin \texttt{succ}_{\mathbb{E}(G)[\overline{X}]}(u^h)$. We now show that $T = S \setminus \{u\}$ is an s-d separator, thus contradicting the minimality of S. Suppose, by contradiction, that T is not an s-d separator: that is, $d \in \texttt{succ}_{G[\overline{T}]}(s)$. From Lemma 1, it follows that there is an s^t-d^h path π in $\mathbb{E}(G)[\overline{T^h}]$, where $T^h = S^h \setminus \{u^h\}$. On the other hand, since S is an s-d separator, from Lemma 1 it also follows that $d^h \notin \texttt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. This implies that π must pass through the node u^h. All the nodes following u^h in π have to be in $\overline{S^h}$ and they cannot be in X, since otherwise they would reachable from s^t in $\mathbb{E}(G)[\overline{S^h}]$ thus contradicting the fact that $d^h \notin \texttt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. Hence, there exists an u^h-d^h path in $\mathbb{E}(G)[\overline{X}]$, contradicting the fact that $d^h \notin \texttt{succ}_{\mathbb{E}(G)[\overline{X}]}(u^h)$. This implies that all $u^h \in \Gamma(X)$ are in $\texttt{pred}_{\mathbb{E}(G)[\overline{X}]}(d^h)$.

We have thus concluded the proof of the necessity. Let us now prove the sufficiency (see Figure 3). By Lemma 3, we have that S is an s-d separator. We only need to show that it is minimal: to this aim, we now prove that, for any node $u \in S = \langle X, \overline{X} \rangle$, $d \in \texttt{succ}_{G[\overline{T}]}(s)$, where $T = S \setminus \{u\}$. From the definition of vertex cut, we have that $u^t \in X$ and $u^h \in \overline{X}$, that is, $u^h \in \Gamma(X)$. Since $u^t \in X$, condition 2 implies that $u^t \in \texttt{succ}_{\mathbb{E}(G)[X]}(s^t)$: note that, for any $v \in S$, $v^h \in \overline{X}$, so that $u^t \in \texttt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(s^t)$. Since $u^h \in \Gamma(X)$, condition 3 implies that $d^h \in \texttt{succ}_{\mathbb{E}(G)[\overline{X}]}(u^h)$: note that, for any $v \in S$, $v^t \in X$ and v^t is the only in-neighbor of v^h, so that $d^h \in \texttt{succ}_{\mathbb{E}(G)[\overline{S^h}]}(u^h)$. In summary, by using the arc (u^t, u^h) we then obtain an s^t-d^h path in $\mathbb{E}(G)[\overline{T^h}]$, where $T^h = S^h \setminus \{u^h\}$. From Lemma 1, it follows that $d \in \texttt{succ}_{G[\overline{T}]}(s)$. The minimality of S has thus been proved, and the proof of theorem is complete. ◀

Algorithm 1 Enumeration of all vertex cuts $\langle X, \overline{X}\rangle$ satisfying conditions 1-3 of Theorem 4.

```
1  Function CUTS(X, F)
2      if Γ(X) ⊆ F then output ⟨X, X̄⟩
3      else
4          u^h ← arbitrary element in Γ(X) \ F
5          CUTS(X, F ∪ {u^h})
6          Z ← CLOSURE(X ∪ {u^h}, F)
7          if Γ(Z) ⊆ pred_{E(G)[Z̄]}(d^h) ∪ {d^h} then CUTS(Z, F)
8  Function CLOSURE(X, F)
9      C ← haug_G(taug_G(X), F)
10     while C ≠ X do
11         X ← C; C ← haug_G(taug_G(X), F)
12     return X
13 Function MAIN(G = (V, E), s, d)
14     F ← {d^h} ∪ N^-_{E(G)}(d^t); X ← CLOSURE({s^t, s^h}, F)
15     CUTS(X, F)
```

3 Minimal s-d separator enumeration

In this section, we will provide an algorithm for the enumeration of minimal s-d separators in a given s-d DAG $G = (V, A)$. The algorithm (see Algorithm 1) will employ a binary partition scheme (see, for example, [17]), and its correctness proof will be based on the characterization of Theorem 4. Before explicitly describing the algorithm, we need a few preliminary definitions.

Given $X, F \subseteq \mathbb{E}(V)$ with $X \cap F = \emptyset$, we define the following two operations.

Tail augmentation: $\texttt{taug}_G(X) = X \cup \bigcup_{u^h \in X} N^+_{\mathbb{E}(G)}(u^h)$.

Head augmentation: $\texttt{haug}_G(X, F) = X \cup \Big(\Gamma(X) \setminus (\texttt{pred}_{\mathbb{E}(G)[\overline{X}]}(d^h) \cup F)\Big)$.

Intuitively, the tail augmentation enforces property G3 of the definition of goodness, while the head augmentation identifies nodes in $\Gamma(X) \setminus F$ that do not have valid paths to d in $\mathbb{E}(G)[\overline{X}]$, and that must, hence, be added to X in order to satisfy condition 3 of Theorem 4. With this in mind, we define the *closure* $\mathcal{C}_G(X, F)$ of $X \subseteq \mathbb{E}(V)$, for a fixed set $F \subseteq \mathbb{E}(V)$, as the smallest set containing X closed with respect to the two operations $\texttt{taug}_G(\cdot)$ and $\texttt{haug}_G(\cdot, \cdot)$ (that is, $\texttt{taug}_G(\mathcal{C}_G(X, F)) = \mathcal{C}_G(X, F)$ and $\texttt{haug}_G(\mathcal{C}_G(X, F), F) = \mathcal{C}_G(X, F)$). A direct computation of the closure of X is shown in function CLOSURE of Algorithm 1 (a significantly more efficient implementation can be made).

We are now ready to describe our algorithm to enumerate all vertex cuts $\langle X, \overline{X}\rangle$ satisfying conditions 1-3 of Theorem 4 (see Algorithm 1). The algorithm maintains both the set X and a set of *forbidden* head nodes $F \subseteq \mathbb{E}(V)$, such that $X \cap F = \emptyset$. We start with $F = \{d^h\} \cup N^-_{\mathbb{E}(G)}(d^t)$ and $X = \mathcal{C}_G(\{s^t, s^h\}, F)$. The algorithm makes X grow by considering the possible choices in $\Gamma(X) \setminus F$. Specifically, for each u^h in $\Gamma(X) \setminus F$, it applies the binary partition technique, by partitioning the set of vertex cuts between the ones in which $u^h \in \overline{X}$ and the ones in which $u^h \in X$. In the first case, u^h is added to F and the recursion continues. In the second case, u^h is added to X, the closure of the new set is performed, and the

recursion proceeds only if condition 3 of Theorem 4 is satisfied (indeed, it is not difficult to show that the closure operation can invalidate this condition). In both cases, the recursion (if executed) continues generating all the vertex cuts $\langle Y, \overline{Y} \rangle$ such that $X \subseteq Y$ and $F \subseteq \overline{Y}$. The invariant is that in any branch of the algorithm, $\langle X, \overline{X} \rangle$ is always a *solution* (that is, it is a vertex cut satisfying conditions 1-3 of Theorem 4), so that there are no dead-ends in the recursion (that is, any branch will produce at least one solution).

Figure 4 shows one subtree of the execution of the algorithm with input the expanded *s-d* DAG in the right part of Figure 1, while the other subtree is shown in Figure 5 (see the caption for the semantics of the colors). Note that the leaves of the execution tree correspond to the seven minimal *s-d* separators of the *s-d* DAG in the center part of Figure 1, that we already listed in the introduction.

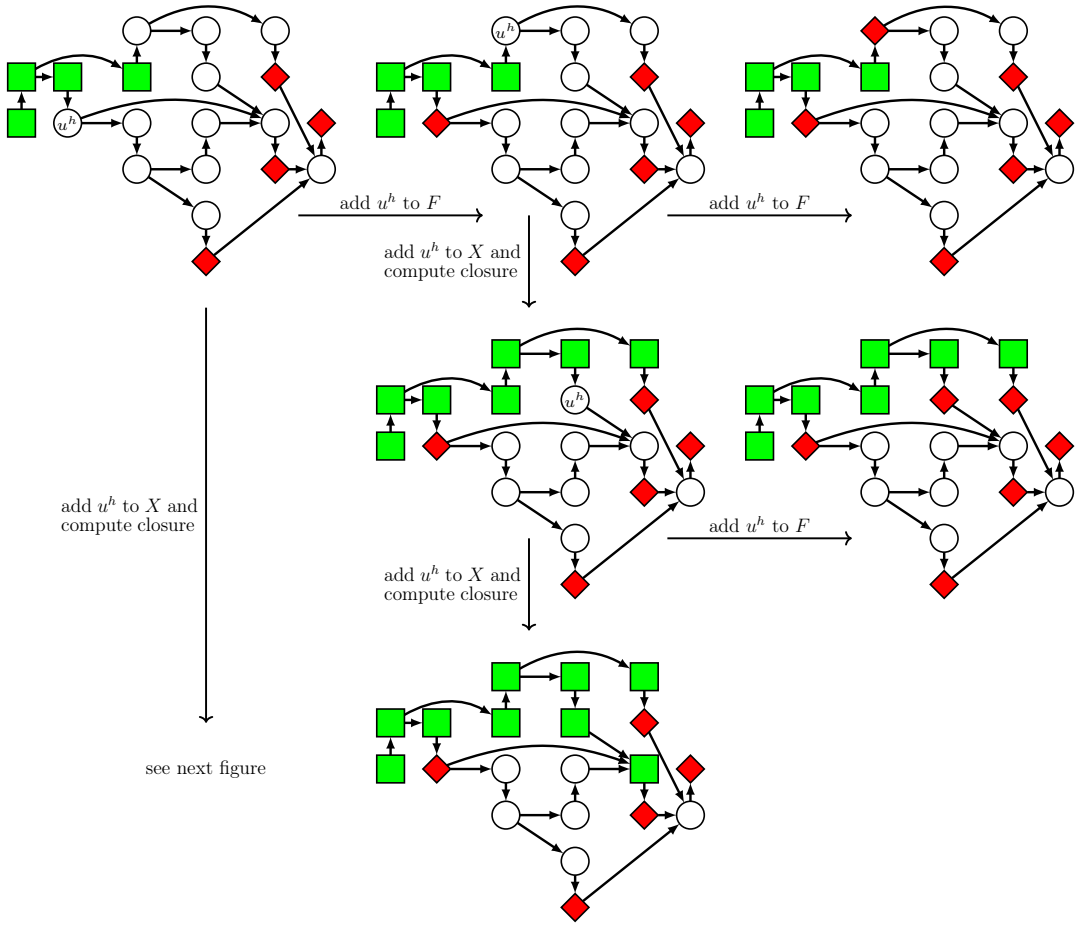

Figure 4 The recursion tree of our algorithm for the expanded *s-d* DAG in the right part of Figure 1 (left subtree is shown in Figure 5). Green rectangle nodes are in X while red diamond nodes are in F: hence, $\overline{X}$ includes white circle and red diamond nodes. A solution $\langle X, \overline{X} \rangle$ is reached when no green rectangle node is connected to a white circle node: the corresponding minimal *s-d* separator for the *s-d* DAG in the center part of Figure 1 includes all nodes a_i such that a_i^t is a green rectangle node and a_i^h is a red diamond node.

Recall that $\overline{X} = \mathbb{E}(V) \setminus X$ and that, for $\langle X, \overline{X} \rangle$ to be a solution, the three conditions of Theorem 4 must be satisfied. We will show in the proof of Theorem 6 that these conditions hold at the beginning of the computation, and that they are preserved when making the recursive calls. To this aim, let us first notice that, from the definition of Algorithm 1, it follows that, at any invocation of the function CUTS, $X \cap F = \emptyset$, $X = \mathcal{C}_G(X, F)$, and $\Gamma(X) \subseteq \texttt{pred}_{\mathbb{E}(G)[\overline{X}]}(d^h) \cup \{d^h\}$. In order to prove Theorem 6, we first show that the goodness of the set X is preserved by the closure operation.

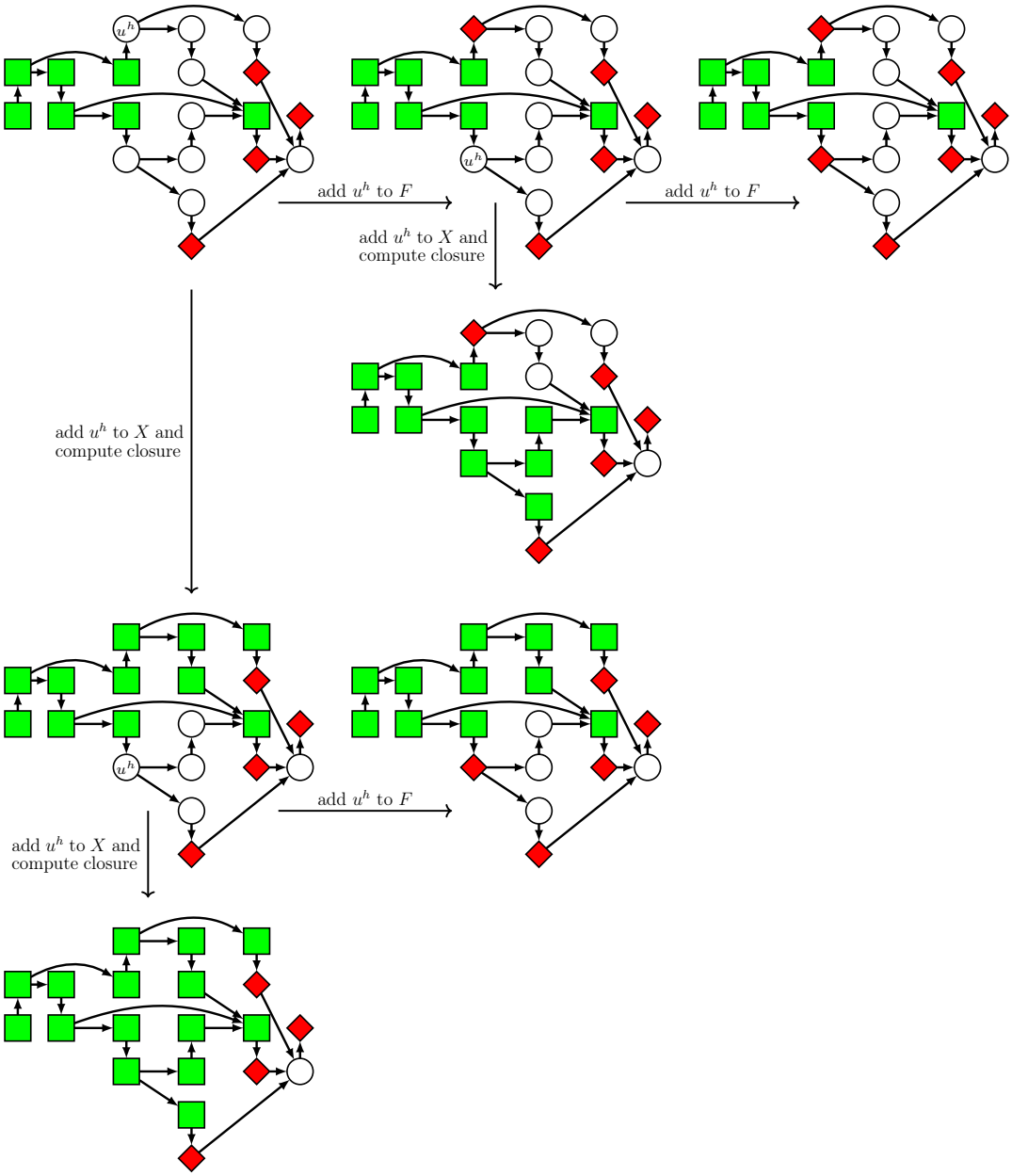

Figure 5 The left subtree of the recursion tree of our algorithm for the expanded s-d DAG in the right part of Figure 1.

▶ **Lemma 5.** *If $\{d^h\} \cup N^-_{\mathbb{E}(G)}(d^t) \subseteq F$, and if either $X = \{s^t, s^h\}$ or X is good, then, for any $u^h \in \Gamma(X) \setminus F$, $\mathcal{C}_G(X \cup \{u^h\}, F)$ is also good.*

Proof. Note first that $d^t \notin \mathcal{C}_G(X \cup \{u^h\}, F)$. Indeed, d^t can never be added by a head augmentation (which adds only head nodes). Moreover, d^t can never be added by a tail augmentation since all its in-neighbors are in F and, hence, they will never be added to X. Let us now prove that $\mathcal{C}_G(X \cup \{u^h\}, F)$ is good. Condition G1 is satisfied either because $X = \{s^t, s^h\}$, or because X is good and adding nodes cannot break condition G1. Furthermore, because of the assumption on F and because $d^t \notin \mathcal{C}_G(X \cup \{u^h\}, F)$, condition G2 is also satisfied. Finally, note that condition G3 is satisfied whenever $X = \mathtt{taug}_G(X)$ (that is, whenever X already contains all neighbors of its head nodes), and, thus, it is guaranteed by the fact that $\mathcal{C}_G(X \cup \{u^h\}, F)$ is closed with respect to the operation $\mathtt{taug}_G(\cdot)$. ◀

▶ **Theorem 6** (Correctness). *At any invocation of function* CUTS *of Algorithm 1, $\langle X, \overline{X}\rangle$ is a solution.*

Proof. At the beginning of the computation, the call $\textsc{cuts}(X, F)$ is performed with $X = \mathcal{C}_G(\{s^t, s^h\}, F)$ and $F = \{d^h\} \cup N^-_{\mathbb{E}(G)}(d^t)$. Note that $\mathcal{C}_G(\{s^t, s^h\}, F) = \mathtt{taug}_G(\{s^t, s^h\})$, since the graph is acyclic. From Lemma 5, it follows that X is good (that is, condition 1 is satisfied). Moreover, since the tail augmentation only adds neighbors of s^h, which is the only neighbor of s^t, we have that $X = \mathtt{succ}_{\mathbb{E}(G)[X]}(s^t) \cup \{s^t\}$: hence, condition 2 is also satisfied. Finally, $\Gamma(X) \subseteq \mathtt{pred}_{\mathbb{E}(G)[\overline{X}]}(d^h) \cup \{d^h\}$, since otherwise the graph would contain a cycle.

Let us now consider a generic call of function CUTS, and assume that the corresponding X satisfies all conditions of Theorem 4. Let $u^h \in \Gamma(X) \setminus F$ be the node chosen for the two recursive calls. We will show that the properties are kept in the recursive calls. In the call $\textsc{cuts}(X, F \cup \{u^h\})$, the properties trivially hold since we do not change X. Let us then consider the call $\textsc{cuts}(Z, F)$ with $Z = \mathcal{C}_G(X \cup \{u^h\}, F)$. This invocation is executed only if $\Gamma(Z) \subseteq \mathtt{pred}_{\mathbb{E}(G)[\overline{Z}]}(d^h) \cup \{d^h\}$ (line 7 of Algorithm 1): hence, condition 3 of Theorem 4 is satisfied. By Lemma 5, it follows that Z is good: hence, also condition 1 of Theorem 4 is satisfied. Moreover, since the two augmentation operations add only neighbors of nodes which are already in their argument, and since $s^t \in \{s^t, s^h\}$, we have that $X \subseteq \mathtt{succ}_{\mathbb{E}(G)[X]}(s^t) \cup \{s^t\}$. The opposite inclusion is trivial, thus implying that $\mathtt{succ}_{\mathbb{E}(G)[X]}(s^t) \cup \{s^t\} = X$ and, hence, that condition 2 is also satisfied. The theorem is thus proved. ◀

▶ **Theorem 7** (Completeness). *Algorithm 1 outputs all solutions without duplicates.*

Proof. Recall that $\langle Y, \overline{Y}\rangle$ is a solution if and only if it satisfies the three conditions of Theorem 4. We will show that, for any solution $\langle Y, \overline{Y}\rangle$, there is a path in the recursion tree which outputs $\langle Y, \overline{Y}\rangle$ such that, at each node of the path corresponding to some call $\textsc{cuts}(X, F)$, the two conditions $X \subseteq Y$ and $F \subseteq \overline{Y}$ hold.

At the beginning of the algorithm, it is easy to show that both inclusions hold because of the goodness of Y: conditions G1 and G3 imply $X = \mathcal{C}(\{s^t, s^h\}, F) = \mathtt{taug}_G(s^t, s^h) \subseteq Y$, while $F = \{d^h\} \cup N^-_{\mathbb{E}(G)}(d^t) \subseteq \overline{Y}$ since, by conditions G2 and G3, Y cannot contain the in-neighbors of d^t. We now show that if the inclusions hold at one given call $\textsc{cuts}(X, F)$ (that is, $X \subseteq Y$ and $F \subseteq \overline{Y}$), then they also hold in exactly one of the two subsequent recursive calls. Indeed, let $u^h \in \Gamma(X) \setminus F$ be the node considered for the recursive calls. We have two possibilities: either $u^h \in Y$, or not. If $u^h \notin Y$, then the two inclusions still hold for the call $\textsc{cuts}(X, F \cup \{u^h\})$, since we did not modify X, and the element added to F is in $\overline{Y}$. On the other hand, if $u^h \in Y$, we show that the call $\textsc{cuts}(Z, F)$ for $Z = \mathcal{C}_G(X \cup \{u^h\}, F)$ is indeed performed, and retains the inclusions. Note first that by definition of closure, and since

$X \cup \{u^h\} \subseteq Y$, we have the inclusion $Z = \mathcal{C}_G(X \cup \{u^h\}, F) \subseteq \mathcal{C}_G(Y, F) = Y$. Assume now by contradiction that $\Gamma(Z) \not\subseteq \texttt{pred}_{\mathbb{E}(G)[\overline{Z}]}(d^h) \cup \{d^h\}$, thus failing the check on line 7. We remark that such a check can only fail for nodes in $\Gamma(Z) \cap F$, as nodes not belonging to F with such a property would have been included in Z by the closure, by definition. By contradiction hypothesis, we are assuming that there exists at least one v^h in $\Gamma(Z) \setminus \texttt{pred}_{\mathbb{E}(G)[\overline{Z}]}(d^h) \cup \{d^h\}$. Since $\Gamma(Y) \subseteq \texttt{pred}_{\mathbb{E}(G)[\overline{Y}]}(d^h) \cup \{d^h\} \subseteq \texttt{pred}_{\mathbb{E}(G)[\overline{Z}]}(d^h) \cup \{d^h\}$. This implies that $v^h \notin \Gamma(Y)$: v^t, v^h are either both in Y, or in $\overline{Y}$. Since $v^h \in \Gamma(Z)$, we have $v^t \in Z \subseteq Y$, meaning that $v^h \in Y$. Recall now that we remarked that $v^h \in F \subseteq \overline{Y}$, leading us to a contradiction. Therefore, as $u^h \in Y$, the check at line 7 gives a positive response, and the subsequent call retains the inclusions (we showed that $Z \subseteq Y$; while F is clearly still contained in $\overline{Y}$).

It remains to prove that the execution path arrives at outputting the solution $\langle Y, \overline{Y} \rangle$. To this aim, we show that, at each invocation of the function CUTS, either $X = Y$, or there is $u^h \in \Gamma(X)$ belonging to Y. If $X \neq Y$, let $y \in Y \setminus X$. By condition 2 of Theorem 4, there exists an s^t-y path in $\mathbb{E}(G)[Y]$: let u be the first node in this path which does not belong to X. Note that u cannot be a tail node, because of property G3 of the goodness definition (recall that X is good). Hence, u is a head node in $\Gamma(X) \cap Y$. This implies the algorithm eventually reaches a recursion node where $X = Y$ and $F \subseteq \overline{Y}$. From this, $\langle Y, \overline{Y} \rangle$ can be found by recursively taking the call on Line 5 until we reach a leaf, as X is not modified and eventually F will contain $\Gamma(X)$.

Finally, absence of duplication is guaranteed by the fact that, on a recursive call considering a certain u^h, all solutions in one recursive subtree will have $u^h \in X$, while all the ones in the other subtree will have $u^h \notin X$. ◀

▶ **Theorem 8** (Complexity). *Given a DAG G with n nodes and m arcs, Algorithm 1 has $O(nm)$ delay and $O(m)$ space.*

Proof. Since each leaf of the recursion tree outputs a solution, because of Theorem 6, the delay between two consecutive solutions is bounded by the cost of a root-to-leaf path in the recursion tree. The depth of the tree is $O(n)$ as at each step a node is added to either X or F. Thus, the cost per solution is equal to $O(n)$ times the cost of one recursive call. This latter cost depends on the execution time of the function CLOSURE. If we use the naive implementation shown in Algorithm 1, this execution takes $O(m^2)$ time and $O(m)$ space. A more efficient implementation whose time and space are both bounded by $O(m)$ can be easily given. Furthermore, the space to store the recursion stack amortizes to $O(m)$ if we store just the differences of X and F with respect to their parent recursive call. The theorem thus follows. ◀

4 Application to temporal graphs and conclusion

In the introduction we have seen how the problem of enumerating all minimal *s-d* cutsets in a temporal graph can be reduced to the problem of enumerating all minimal *s-d* separators in the corresponding *s-d* line graph. It is easy to verify that the number nodes and arcs in the line graph are respectively $O(m)$ and $O(m^2)$, where m is the number of temporal arcs. This observation along with Theorems 6-8 gives us the following result.

▶ **Theorem 9.** *Given a temporal graph G with m temporal arcs and given two nodes s and d, all minimal s-d cutsets in G can be computed with $O(m^3)$ delay and $O(m^2)$ space.*

A natural open problem is whether the cost per solution for enumerating all minimal *s-d* separators in a DAG in Theorem 8 can be reduced to $O(m)$ in order to improve also Theorem 9.

References

1 Matthias Bentert, Anne-Sophie Himmel, Hendrik Molter, Marco Morik, Rolf Niedermeier, and René Saitenmacher. Listing all maximal k-plexes in temporal graphs. In *ASONAM*, pages 41–46, 2018.

2 Kenneth A Berman. Vulnerability of scheduled networks and a generalization of menger's theorem. *Networks: An International Journal*, 28(3):125–134, 1996.

3 Anne Berry, Jean-Paul Bordat, and Olivier Cogis. Generating all the minimal separators of a graph. In *WG*, pages 167–172, 1999.

4 Paola Bertolazzi, Giuseppe Di Battista, and Walter Didimo. Quasi-upward planarity. *Algorithmica*, 32(3):474–506, 2002.

5 Binh-Minh Bui-Xuan, Afonso Ferreira, and Aubin Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.

6 Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

7 S.K. Chaturvedi, Gaurav Khanna, and Sieteng Soh. Reliability evaluation of time evolving delay tolerant networks based on sum-of-disjoint products. *Reliability Engineering & System Safety*, 171:136–151, 2018.

8 Qizhi Fang, Rudolf Fleischer, Jian Li, and Xiaoxun Sun. Algorithms for core stability, core largeness, exactness, and extendability of flow games. In *COCOON*, pages 439–447, 2007.

9 Longxiang Gao, Shui Yu, Tom H. Luan, and Wanlei Zhou. *Delay Tolerant Networks*. Springer, 2015.

10 Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, and Manuel Sorge. Adapting the Bron–Kerbosch algorithm for enumerating maximal cliques in temporal graphs. *Social Network Analysis and Mining*, 7(1):35, 2017.

11 P. A. Jensen and M. Bellmore. An algorithm to determine the reliability of a complex system. *IEEE Transactions on Reliability*, R-18(4):169–174, 1969.

12 Gaurav Khanna, Sanjay K. Chaturvedi, and Sieteng Soh. Two-terminal reliability analysis for time-evolving and predictable delay-tolerant networks. *Recent Advances in Electrical & Electronic Engineering*, 13(2):236–250, 2020.

13 Gaurav Khanna and Sanjay Kumar Chaturvedi. A comprehensive survey on multi-hop wireless networks: Milestones, changing trends and concomitant challenges. *Wireless Personal Communications*, 101(2):677–722, 2018.

14 Rohit Kumar and Toon Calders. 2SCENT: an efficient algorithm to enumerate all simple temporal cycles. *Proceedings of the VLDB Endowment*, 11(11):1441–1453, 2018.

15 Matthieu Latapy, Tiphaine Viard, and Clémence Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(1):61:1–61:29, 2018.

16 Q. Liang and E. Modiano. Survivability in time-varying networks. *IEEE Transactions on Mobile Computing*, 16(9):2668–2681, 2017.

17 Andrea Marino. *Analysis and Enumeration: Algorithms for Biological Graphs*. Springer, 2015.

18 Othon Michail. An introduction to temporal graphs: An algorithmic perspective. *Internet Mathematics*, 12(4):239–280, 2016.

19 Petra Mutzel and Lutz Oettershagen. On the enumeration of bicriteria temporal paths. In *TAMC*, pages 518–535, 2019.

20 Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Information Processing Letters*, 71(5-6):199–204, 1999.

21 J. Scott Provan and Douglas R. Shier. A paradigm for listing (s, t)-cuts in graphs. *Algorithmica*, 15(4):351–372, 1996.

22 Marvin Rausand. *Reliability of Safety-Critical Systems: Theory and Applications*. John Wiley & Sons, Ltd, 2014.

23 Hong Shen, Keqin Li, and Si-Qing Zheng. Separators are as simple as cutsets. In *ASIAN*, pages 347–358, 1999.

24 Douglas R. Shier. *Network Reliability and Algebraic Structures*. Clarendon Press, 1991.

25 S. Tsukiyama, I. Shirakawa, H. Ozaki, and H. Ariyoshi. An algorithm to enumerate all cutsets of a graph in linear time per cutset. *Journal of the ACM*, 27(4):619–632, 1980.

26 Tiphaine Viard, Matthieu Latapy, and Clémence Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

27 Philipp Zschoche, Till Fluschnik, Hendrik Molter, and Rolf Niedermeier. The complexity of finding small separators in temporal graphs. *J. Comput. Syst. Sci.*, 107:72–92, 2020.