Polynomially Recognising Graphs where Saturating Flows are Always Maximum

Daniele Gorla Federica Granese
Sapienza University of Rome, Department of Computer Science

February 6, 2019

Abstract

In this paper we provide a polynomial-time algorithm for deciding whether, given a directed graph with a fixed source and target node, there exists a capacity-to-edges assignment such that the resulting flow network admits a non-maximum saturating flow. This problem has already been studied in the framework of node-capacitated flow networks but, surprisingly, the algorithm developed therein cannot be used in the edge-capacitated model after the traditional 'edge expansion' procedure. Thus, here we follow a similar path and develop the required algorithm, that runs in $O(|E|^3)$. To this aim, we also need to develop a theory for *minimal cutsets*, that, as we prove, are minimal sets of edges whose removal disconnects the source from the target. In particular, we define two order relations on such sets that are used in the main algorithm.

1 Introduction

Flow networks [1] are one of the best known models in computer science and transportation engineering. They are graphs with a source node s, a target node t, and a capacity function that associates an upper bound on the quantity of flow that can pass through every edge or node while traveling from the source to the target. The edge-capacitated model is the one that received more attention in the literature, even though the two models are considered equivalent [1]. In both cases, the basic problem is to determine a maximum flow, i.e. to maximize the amount of information that can be sent from the source to the target node while respecting the constraints put by the capacity function.

Maximum flows have the property of saturating the network, i.e. after a maximum flow nothing can be sent anymore. It is then natural to study the possibility of having non-maximum saturating flows. This is interesting because, if a network only admits maximum saturating flows, then the algorithm for finding a maximum flow is trivial: choose any path from s to t and saturate it, until all paths have at least a saturated edge. This is simply an iterated depth-first search.

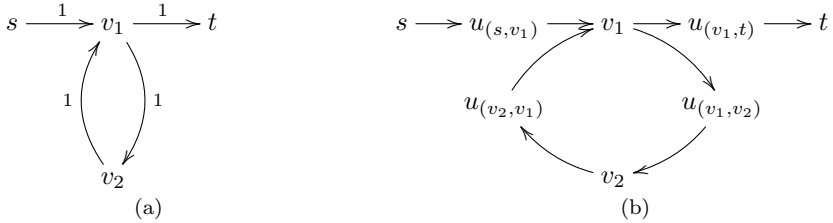


Figure 1: (a) A flow network, with capacities drawn on edges (b) its associated node-capacitated net, where $\eta(s) = \eta(v_1) = \eta(v_2) = \eta(t) = \infty$ and $\eta(u_{(s,v_1)}) = \eta(u_{(v_1,v_2)}) = \eta(u_{(v_2,v_1)}) = \eta(u_{(v_1,t)}) = 1$.

This problem was considered in [7, 8] for the node-capacitated model. In particular, two results are presented: first, checking whether a graph together with a capacity-to-nodes assignment admits a non-maximum saturating flow is an NP-complete problem [7]; second, checking whether a graph admits a capacity-to-nodes assignment that enables a non-maximum saturating flow is a polynomial-time problem [8]. In particular, they provide an algorithm that runs in $O(|V|^2|E|)$ able to check the latter property, called *weakness* therein. The aim of this paper is to study the notion of weakness in the edge-capacitated model; for the sake of clarity, we shall call *edge-weakness* and *node-weakness* the existence of a capacity function that enables non-maximum saturating flows in the edge-capacitated and in the node-capacitated models, respectively. Such notions have already been compared in [9] for directed graphs: for general directed graphs, the two notions turned out to be incomparable; for DAGs, node-weakness implies edge-weakness, but not vice versa.

A first attempt to polynomially check edge-weakness consists in turning an edge-capacitated net into the corresponding node-capacitated one and then use the algorithm for node-weakness. Indeed, there exists a well known procedure [1] for passing from one model to the other that usually preserves all relevant notions, like flow admissibility, flow value and, consequently, saturating and maximum flows. In our case, we can 'edge expand' a flow network to obtain a node-capacitated net (with η a function associating to every node a non-negative capacity): we split every edge $e = (u, v)$ into two edges (u, u_e) and (u_e, v), where u_e is a new node for every edge e; then, we let $\eta(u_e) = c(e)$ and $\eta(u)$ to be a large enough capacity (say, ∞), for every original node u. A simple example of this transformation is given in Figure 1.

However, this approach does not work. For example, the graph underlying the flow network in Figure 1(a) is not edge-weak, because, for every capacity-to-edges assignment c, we have that all saturating flows have value $\min\{c(s, v_1), c(v_1, t)\}$, that hence is the maximum possible flow. By contrast, the graph underlying its edge-expansion (Figure 1(b)) is node-weak, because of the capacity-to-nodes assignment that assigns 2 to every node: in this case, there exists a saturating flow of value 1 (it suffices to send 1 along the path $s \to\to v_1 \to\to v_2 \to\to v_1 \to\to t$), whereas the maximum flow is 2 (send-

able along the path $s \to\to v_1 \to\to t$). So, while edge-weakness of G implies node-weakness of the edge-expansion of G, the converse does not hold.

In this paper, we follow the ideas in [8] and adapt them to the edge-capacitated model. This is done by exploiting the characterizations of both weakness notions given in [8, 9]: *a graph is edge-/node-weak if there exists a path that passes through a MES/MVS twice*, where a MES/MVS (acronyms for Minimal Edge/Vertex Separator) is a minimal set of edges/nodes whose removal disconnects the source and the target. The algorithm provided in [8] essentially finds such a MVS, but requires some ingenuity to be polynomial. As we show in this paper, the very same tricks can be used to polynomially find a MES and a path passing through it twice. To this aim, we first show that MESs coincide with the well-known notion of *minimal cutsets* (MCSs) [4, 15, 21]; then, we prove that they form a poset w.r.t. two order relations adapted from [20].

To sum up, both the algorithm for edge-weakness, together with its soundness proof and its concrete implementation run on a number of sample graphs, and the theory of minimal cutsets are the contributions of this paper.

Related work In the last years, the literature has shown a great deal of interest in modeling network scenarios, where one of the main topics addressed is network efficiency: for example, minimizing energy consumption in the net [18, 23] in order to maximize the net lifetime [6, 10]. Clearly, the current work follows this research line: here minimize the energy consumption for maximizing the network lifetime means to maximize the amount of information passing on it, i.e. finding graphs having only maximum saturating flows.

The model of nets that we adopt in this paper is very basic. For example, it is possible to endow each edge of the *st*-digraph with other functions, like a cost for using the edge or for deleting it. In [2], every edge has a capacity and a deletion cost, and the issue is to study the *maximum flow network interdiction problem*, i.e. to minimize the maximum flow of the graph induced by the edge elimination given a certain budget (so, the aim is to find the greatest damage under that budget). This is an NP-complete problem that generalizes the *network inhibition problem* [16], where the issue is to kill the net by deleting edges with a given budget. In [22], the graph has capacities and usage costs on the edges and the problem is to find the maximum flow of the minimum cost. Then, it is proved that, for acyclic multigraphs with a single source, a single sink, an arbitrary linear cost functions and arbitrary capacities, the problem can be solved by a greedy algorithm (that non-deterministically chooses paths and saturates them) if and only if the graph is *series-parallel* [17]. In [13], the cost on the edges represents the time spent to pass through the edge and the issue to solve is the *minimum-cost dynamic flow problem*, i.e. find a flow that respects an overall time availability (something that resembles the available budget of [2, 16]). Also here, a greedy algorithm solves this problem if and only if the graph is series-parallel.

The idea of solving a flow problem with a greedy algorithm (that simply selects and saturates paths) resembles what happens in our model for non-edge-weak graphs. However, we deal with generic graphs (i.e., not necessarily

DAGs), so the characterization we use is more sophisticated. Incidentally, in [9] edge-weakness is shown to coincide with not-TTSP (*not* being two-terminal series-parallel) in DAGs. Moreover, [9] also compares edge-weakness with the notion of *vulnerability* for traffic networks. These are graphs where edges are equipped with a latency (that is a function of the flow over the edge) and the issue is to minimize the overall delay in traveling from source to target; in this context, a graph is vulnerable if it admits a latency function and a flow at the Nash equilibrium with a non-minimum delay (this notion is strictly related to the *Braess's paradox* [5], according to which network performances can be improved by removing edges). In a general st-digraph, vulnerability implies edge-weakness; moreover, if the graph is a DAG, then also the reverse implication holds.

In [11] the problem of finding saturating flows in networks is studied. However, the notion of saturation they use is different from ours: for them, a flow is saturating if it saturates every edge it passes through, whereas in our case it suffices to saturate (at least) one edge for every possible st-path. Again, the problem is shown NP-complete by reducing it to 3SAT.

Another related work is [3], where a denial of service attack is expressed in a graph-theoretical way by equipping nodes with a disabling cost and a damage (expressing the impact of removing the node from the net). The objective is to find the attack having the maximum damage with the minimum cost, i.e. choose a subset of nodes to delete (attack), minimizing the cost for deleting nodes (cost of the attack) and creating the highest possible damage (cut the single source apart from as many nodes – i.e., sinks – as possible).

To conclude, notice that some papers deal with the problem of enumerating all the cutsets (minimal or not) separating s and t in a graph. In particular, the problem in a undirected setting for general cutsets has been considered in [21]; the enumeration of all *minimal* cutsets appears in [4] for undirected graphs and in [15] for directed graphs. Like in [20], our ordering relations can be used to enumerate all minimal cutsets in a directed graph in time polynomial w.r.t. the number of the cutsets of the graph that, however, can be exponential in the size of the graph.

Structure of the paper The paper is organized as follows. In Section 2 we set out our model, by recalling some standard notions on flow nets and a few definitions and results from [9]. In Section 3 we develop the basic theory of cutsets, mostly by providing two order relations on them. These orders will be used in the algorithm for edge-weakness, that is fully detailed in Section 4. Section 5 concludes the paper, by also reporting some experimental results.

2 Preliminaries

A directed graph $G = (V, E)$ consists of a set V of *vertices* (or *nodes*) and of a set $E \subseteq V \times V$ of *edges* (or *arcs*). Throughout the paper, we only consider *simple st-digraphs* that are directed graphs without self-loops and parallel edges, with a fixed source node s (without incoming edges) and sink node t (without

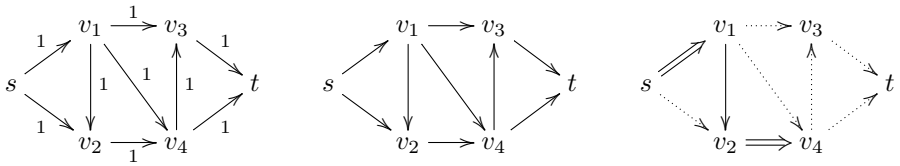


Figure 2: (a) A flow network with edges labeled with their capacities (b) The graph underlying the network (c) a MES for the graph (the edges with a double arrow) and a path that passes through the MES twice (the non-dashed edges).

outgoing edges). We denote with $in(u)$ the set $\{(v,u) \ : \ (v,u) \in E\}$ and with $out(u)$ the set $\{(u,v) \ : \ (u,v) \in E\}$. Then, if $e = (u,v)$, we let $next(e) = out(v)$ and $prev(e) = in(u)$.

A (possibly cyclic) *path* p from a node u to v, notation $u \overset{p}{\rightsquigarrow} v$, is a sequence $u_1 \dots u_n$ $(n > 1)$ of nodes such that $u_1 = u$, $u_n = v$ and, for every $1 \leq i < n$, $(u_i, u_{i+1}) \in E$. We say that (u,v) *belongs to* p (or also that p *passes through* (u,v)) if $u = u_i$ and $v = u_{i+1}$, for some $i \in \{1, \dots n-1\}$. An *st-path* is a path from s to t; we denote with $P(G)$ the set of all st-paths in G.

A *flow network* is a pair (G,c) made up of an st-digraph G and a capacity function, $c : V \times V \to \mathbb{R}^+$, associating to every edge $(u,v) \in E$ a non-negative capacity $c(u,v)$, and a zero capacity to every $(u,v) \notin E$. The next definition describes the notion of *flow* in flow networks; here and in the rest of the paper, we use the following convention: if φ is a function on edges (nodes) and X is a set of edges (nodes), then $\varphi(X) = \sum_{x \in X} \varphi(x)$.

Definition 2.1 (Flow). *A* flow *on* (G,c) *is a function* $f : V \times V \to \mathbb{R}^+$ *satisfying the following three constraints:*

1. *Capacity:* $\forall (u,v) \in V \times V \,.\, f(u,v) \leq c(u,v)$
2. *Conservation:* $\forall u \in V \setminus \{s,t\} \,.\, f(in(u)) = f(out(u))$
3. *Antisymmetry:* $\forall (u,v) \in V \times V \,.\, f(u,v) = -f(v,u)$

The value *of* f *is* $f(out(s))$.

Definition 2.2 (Resulting Net and Saturating Flow). *Given a flow* f *on* (G,c)*, the* resulting net after f *is the flow network* $(G, c-f)$*, where* $(c-f)(e) \triangleq c(e) - f(e)$.

A flow f saturates (G,c) *(or* f *is a saturating flow for* (G,c)*) if the resulting net after* f *has maximum flows of value 0.*

Consider the net in Fig. 2(a). Its maximum flow is 2 and can be obtained by sending one information unit through the paths $s\, v_1\, v_3\, t$ and $s\, v_2\, v_4\, t$. However, it also admits a saturating flow of 1 unit, by sending a unitary flow along $s\, v_1\, v_4\, t$.

Flow networks where every saturating flow is always maximum have the pleasant property that a maximum flow can be easily calculated in a greedy

way: it suffices to pick up an *st*-path, saturate it and iterate the procedure in the resulting net. This essentially corresponds to a visit of the graph. Thus, it is meaningful understanding whether all saturating flows have the same value or not. However, while calculating a maximum flow is polynomial [1], checking the existence of a saturating flow of value smaller than the maximum is an NP-hard problem. This can be easily derived from a result in [7] and is detailed in the Appendix. We now move to characterize those graphs that, for every capacity function, the resulting flow network has a non-maximum saturating flow. Surprisingly, we will show that this problem is simpler than the previous one.

Definition 2.3 (Edge-Weakness). *A graph is* edge-weak *if there exists a capacity-to-edge assignment such that the resulting flow network admits a non-maximum saturating flow.*

For example, the graph in Fig. 2(b) is edge-weak because of the capacity-to-edges assignment depicted in Fig. 2(a) and the saturating unitary flow discussed above. Edge-weakness has been characterized in a graph theoretic way [9], by relying on the notion of *minimal edge separator* (MES, for short), that we now recall.

Definition 2.4 (Edge separator). *An* edge separator *is a set of edges whose removal disconnects s and t.*

Theorem 2.1 ([9])**.** *G is edge-weak if and only if there exists a MES and a path in G that passes through it at least twice.*

Thanks to this result, we can see that the graph in Figure 2(b) is edge-weak by considering the MES $\{(s, v_1), (v_2, v_4)\}$ and the path $s\, v_1\, v_2\, v_4$ (see Fig. 2(c)).

3 A Theory of MCSs

First, we show that MESs coincide with a well-known notion in graph theory, namely *minimal cutsets* [4, 15, 21] (MCSs for short). To define them, given a graph $G = (V, E)$ and $U \subseteq V$, we denote with $w(U, \bar{U})$ the set $\{(u, v) \in E \ : u \in U \wedge v \in \bar{U}\}$.

Definition 3.1 (Cutset). An *st*-cutset (or, simply, a *cutset*) is a set of edges F s.t. there exists a set of nodes U s.t. $F = w(U, \bar{U})$, $s \in U$ and $t \notin U$.

Lemma 3.1. *If F is a cutset, then it is an edge separator.*

Proof. Let $F = w(U, \bar{U})$ and $p : s = u_0, u_1, \ldots, u_k = t$ be an *st*-path. Since $s \in U$ and $t \in \bar{U}$ (by definition of cutset), there exists $i \in \{0, \ldots, k-1\}$ such that $u_i \in U$ and $u_{i+1} \in \bar{U}$; thus, $(u_i, u_{i+1}) \in F$. Since this holds for every p, removing F from G disconnects s and t. □

The converse does not hold: consider, e.g., the graph

$$s \longrightarrow u \longrightarrow v \longrightarrow w \longrightarrow t$$

Indeed, $\{(u,v),(v,w)\}$ is an edge separator but not a cutset. However, if we consider *minimal* cutsets and MESs, the two notions do coincide.

Proposition 3.1. *F is a MCS if and only if it is a MES.*

Proof. (If) By Lemma 2.1 in [9], $F = w(U,\bar{U})$, for $U = \{w \in V \ : \ \exists (u,v) \in F \ \exists s \overset{p}{\rightsquigarrow} u \ s.t. \ w \in p \wedge p \cap F = \emptyset\}$; hence, every MES is a cutset. If it was not minimal, then there would exist $F' \subset F$ that is a cutset and, by Lemma 3.1, F' would also be an edge separator.

(Only If) By Lemma 3.1, F is an edge separator; suppose by contradiction that it is not minimal, i.e. that F properly contains other edge separators. Let F' be a MES contained in F. By Lemma 2.1 in [9], $F' = w(U,\bar{U})$, for $U = \{w \in V \ : \ \exists (u,v) \in F' \ \exists s \overset{p}{\rightsquigarrow} u \ s.t. \ w \in p \wedge p \cap F' = \emptyset\}$; thus, F would properly contain another cutset: contradiction. □

We now give a characterization of MCSs that will be useful in the sequel. Let $X \subset E$, then $G \setminus X$ denotes the graph $(V_G, E_G \setminus X)$. Then,

$$V_s^X = \{u \in V_G \ : \ \exists s \rightsquigarrow u \text{ in } G \setminus X\}$$
$$V_t^X = \{u \in V_G \ : \ \exists u \rightsquigarrow t \text{ in } G \setminus X\}$$

Lemma 3.2. *Let X be a edge-separator of the st-graph $G = (V,E)$. Then X is a MCS of G if and only if every $(u,v) \in X$ is such that $u \in V_s^X$ and $v \in V_t^X$.*

Proof. (Only if) Let $e = (u,v) \in X$ and choose a path p from s to v passing through X only in e (this must exists by minimality of X, seen as a MES by Prop. 3.1). We have that $u \in V_s^X$, otherwise we could not reach u from s in $G \setminus X$ and so p would contain another $e' \in X$, in contradiction with how p was chosen. Similarly, $v \in V_t^X$.

(If) Looking for a contradiction, assume that X is not minimal. Then there exists an edge $e = (u,v)$ which can be removed from X s.t. $G \setminus (X \setminus \{e\})$ still has s and t disconnected. Since by hypothesis $u \in V_s^X$ and $v \in V_t^X$, in G there exists a path from s to t not passing through $X \setminus \{e\}$. Hence X cannot be an edge-separator without e. □

3.1 Ordering MCSs

We now define two order relations on MCSs that will be used in our algorithm for generating a chain of MCSs.

Definition 3.2 (Edge-coverage and Edge-precedence). *An edge $e = (u,v)$ is covered by a set of edges H, written $e \sqsubseteq H$, if all paths $u \rightsquigarrow t$ starting with e touch at least an edge $e' \in H$. A set of edges H' is covered by a set of edges H, written $H' \sqsubseteq H$, if $e \sqsubseteq H, \forall e \in H'$.*

A set of edges H precedes an edge e, written $H \preceq e$, if all paths $s \rightsquigarrow v$ ending with e touch at least an edge $e' \in H$. A set of edges H precedes a set of edges H', written $H \preceq H'$, if $H \preceq e, \forall e \in H'$.

For example, consider the graph in Figure 2(b). Let $A = \{(s, v_1), (s, v_2)\}$ then A is trivially covered by itself. Let $B = \{(v_1, v_3), (v_1, v_4), (v_2, v_4)\}$ and $C = \{(v_3, t), (v_4, t)\}$; then $A \sqsubseteq B$, $B \sqsubseteq C$ and $A \sqsubseteq C$. Clearly $C \not\sqsubseteq B$ and also $C \not\sqsubseteq A$. Similarly, let $D = \{(v_1, v_4), (v_2, v_4), (v_3, t)\}$; then, $D \preceq C$, $A \preceq D$ and $A \preceq C$, whereas $D \not\preceq A$ and $C \not\preceq A$.

Proposition 3.2. *If $e \in X \cap in(t)$ and $X \sqsubseteq Y$, then $e \in Y$; if $e \in X \cap out(s)$ and $Y \preceq X$, then $e \in Y$.*

Proof. If $e \in in(t)$, then $e = (v, t)$, for some $v \in V$; hence, $e \sqsubseteq Y$ only if $e \in Y$. The second claim is similar. □

Lemma 3.3. *The set of MCSs is partially ordered w.r.t. both $\sqsubseteq$ and $\preceq$, whose bottom and top elements are $out(s)$ and $in(t)$, respectively.*

Proof. We only consider $\sqsubseteq$, the same holds for $\preceq$. Clearly $\sqsubseteq$ is reflexive. Moreover, the relation is also transitive: let $X \sqsubseteq X'$ and $X' \sqsubseteq X''$. Thus $e \sqsubseteq X'$, for all $e \in X$, and $e' \sqsubseteq X''$, for all $e' \in X'$; hence $e \sqsubseteq X''$, for all $e \in X$, i.e. $X \sqsubseteq X''$.

We are left with antisymmetry. Looking for a contradiction, let X and X' be two MCSs such that $X \sqsubseteq X'$ and $X' \sqsubseteq X$, but $X \neq X'$. Hence, there exists $e \in X \setminus X'$ (or $e \in X' \setminus X$, that is analogous). By Prop. 3.2, $e \notin in(t)$. By minimality of X, there exists a simple st-path p touching X only in e; let p' be the suffix of p from e to t. Since $X \sqsubseteq X'$, p' passes through X' via some e'. Then, $e' \neq e$ because $e \notin X'$; moreover, $e' \notin in(t)$, otherwise, by Prop. 3.2 and $X' \sqsubseteq X$, we would have $e' \in X$, in contradiction with $p \cap X = \{e\}$. Let p'' be the suffix of p' from e' to t; since $X' \sqsubseteq X$, p'' passes through X via some e''. If $e'' = e$, then p would not be simple; if $e'' \neq e$, then p would not touch X only in e. Both cases lead to a contradiction.

The fact that $out(s)$ and $in(t)$ are the bottom and top elements holds by Definition 3.2. □

3.2 Successor and Predecessor of a MCS

As we said before, relation $\sqsubseteq$ is one of the main ingredients of our algorithm; we now show how a complete chain of MCSs can be iteratively generated. Let $X = out(s)$; a new MCS can be generated from X by replacing any edge e in X with all its immediate successors and deleting all edges that are not necessary to separate t from s (the same can be done backwards from $X = in(t)$). Hence, each X may generate at most $|X|$ new MCSs, through the notions of successor and predecessor of a MCS.

Definition 3.3 (Successor and Predecessor of a MCS w.r.t. any of its edges). *Let X be a MCS and $e \in X$. Then, the successor of X w.r.t. e is*

$$X^e \triangleq ((X \setminus \{e\}) \cup next(e)) \setminus \mathcal{I}_t((X \setminus \{e\}) \cup next(e)) \tag{1}$$

where $\mathcal{I}_t(Y) \triangleq \{e \in Y \mid e \sqsubset Y \setminus \{e\}\}$.

The predecessor of X w.r.t. e is

$$X_e \triangleq ((X \setminus \{e\}) \cup \mathit{prev}(e)) \setminus \mathcal{I}_s((X \setminus \{e\}) \cup \mathit{prev}(e)) \tag{2}$$

where $\mathcal{I}_s(Y) \triangleq \{e \in Y \mid Y \setminus \{e\} \prec e\}$.

Consider again the graph depicted in Figure 2(b). Let X be the MCS $\{(s, v_2), (v_1, v_2), (v_1, v_3), (v_1, v_4)\}$; the successor of X with respect to the edge $e = (s, v_2)$ is the MCS $X^e = \{(v_1, v_3), (v_1, v_4), (v_2, v_4)\}$. Indeed, $\mathit{next}(e) = \{(v_2, v_4)\}$, thus $(X \setminus \{e\}) \cup \mathit{next}(e) = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_4)\}$ and $\mathcal{I}_t((X \setminus \{e\}) \cup \mathit{next}(e)) = \{(v_1, v_2)\}$. Similarly, if $e = (v_1, v_2)$, then $X_e = ((X \setminus \{e\}) \cup \mathit{prev}(e)) \setminus \mathcal{I}_s((X \setminus \{e\}) \cup \mathit{prev}(e)) = \{(s, v_1), (s, v_2), (v_1, v_3), (v_1, v_4)\} \setminus \{(v_1, v_3), (v_1, v_4)\} = \{(s, v_1), (s, v_2)\}$.

Lemma 3.4. *Let X be a MCS and $e \in X$. If $e \notin \mathit{in}(t)$, then X^e is a MCS and $X \sqsubset X^e$; if $e \notin \mathit{out}(s)$, then X_e is a MCS and $X_e \sqsubset X$.*

Proof. Since by hypothesis X is a MCS, then also $X' = (X \setminus \{e\}) \cup \mathit{next}(e)$ is an edge-separator (though not necessarily minimal). To make X' a MES (and thus by Proposition 3.1 a MCS), it is necessary to remove from X' all those edges $e' = (u', v')$ such that either $u' \notin V_s^{X'}$ or $v' \notin V_t^{X'}$ (see Lemma 3.2). We now prove that these edges are $\mathcal{I}_t((X \setminus \{e\}) \cup \mathit{next}(e))$.

First, let $e' \in X \setminus \{e\}$. Since $e' \in X$ and X is a MCS, $u' \in V_s^X$; hence, there exists $s \overset{p}{\rightsquigarrow} u'$ acyclic s.t. $p \cap X = \emptyset$. Moreover, $p \cap \mathit{next}(e) = \emptyset$ otherwise we could combine the prefix of p from s to v' with a $v' \rightsquigarrow t$ (at least one such path exists in $G \setminus X$) and obtain a path from s to t in G that would not touch X. Thus, $p \cap X' = \emptyset$ and so $u' \in V_s^{X'}$. Therefore, it must be that $v' \notin V_t^{X'}$, i.e., for every p' acyclic from v' to t, we have that $p' \cap X' \neq \emptyset$. Thus, $e' \sqsubset X' \setminus \{e'\}$, i.e. $e' \in \mathcal{I}_t((X \setminus \{e\}) \cup \mathit{next}(e))$.

Then, let $e' \in \mathit{next}(e)$, where $e = (u, u')$. Since X is a MCS, $u \in V_s^X$; hence, there exists $s \overset{p}{\rightsquigarrow} u$ acyclic s.t. $p \cap X = \emptyset$. Like before $p \cap \mathit{next}(e) = \emptyset$. Thus, $p' \triangleq s \overset{p}{\rightsquigarrow} u \to u'$ is such that $p' \cap X' = \emptyset$. Hence, $u' \in V_s^{X'}$ and we reason as before.

Finally, $X^e \neq X$, since $e \notin X'$ and hence $e' \notin X^e \subseteq X'$ and $X \sqsubset X' \sqsubseteq X^e$.

The second claim is proved in a similar way. □

Notice that, when $e \in \mathit{in}(t)$, the removal of the edges in X^e from G cannot block paths from s to t via e, since e cannot be replaced with its immediate successors (because edges entering into t have no successors). The same holds for X_e if $e \in \mathit{out}(s)$. Thus X^e and X_e are not edge-separators in these cases.

The next lemma shows that (1) and (2) are inverses of each other. Indeed, by first applying (2) to a MCS X and then (1) to the result, we obtain X. To prove it, we let $h(e)$ denote the length of a shortest path from s to u, for every $e = (u, v) \in E$.

Lemma 3.5. *Let X be a MCS different from $\mathit{out}(s)$; then, there exist $e \in X$ and $f \in \mathit{prev}(e)$ such that $(X_e)^f = X$.*

Proof. Let $e = (u, v) \in X$ be an edge of maximum $h(\cdot)$ among those in X and let $out(u) = A \uplus B$, where $A = out(u) \cap X$ and $B = out(u) \setminus X$. Trivially, $A \subseteq \mathcal{I}_s((X \setminus \{e\}) \cup prev(e))$; we now prove that $\mathcal{I}_s((X \setminus \{e\}) \cup prev(e)) = A \cup P$, for some $P \subset prec(e)$. Indeed, looking for a contradiction, let $e' = (u', v') \in X \setminus A$ (so $u' \neq u$) be such that $((X \setminus \{e\}) \cup prev(e)) \setminus \{e'\} \prec e'$; because X is a MES, it must be that $prev(e) \prec e'$. Since $u' \neq u$, there must be a non-empty path from u to u', against maximality of $h(e)$ in X. Hence

$$\begin{aligned} X_e &\triangleq ((X \setminus \{e\}) \cup prev(e)) \setminus \mathcal{I}_s((X \setminus \{e\}) \cup prev(e)) \\ &= ((X \setminus \{e\}) \cup prev(e)) \setminus (A \cup P) \\ &= (X \setminus A) \cup (prev(e) \setminus P) \end{aligned}$$

since $e \in A$ and $A \cap prev(e) = P \cap X = \emptyset$.

Let $f \in prev(e) \setminus P$; then,

$$\begin{aligned} X' &\triangleq (X_e \setminus \{f\}) \cup next(f) \\ &= (X_e \setminus \{f\}) \cup out(u) \\ &= (((X \setminus A) \cup (prev(e) \setminus P)) \setminus \{f\}) \cup out(u) \\ &= (X \setminus A) \cup (prev(e) \setminus (P \cup \{f\})) \cup out(u) \\ &= X \cup (prev(e) \setminus (P \cup \{f\})) \cup B \end{aligned}$$

Since X is a MES, $B \sqsubseteq X$; thus, $B \subseteq \mathcal{I}_t(X')$. Moreover, since $prev(e) \sqsubseteq out(u) \sqsubseteq X$, also $prev(e) \setminus (P \cup \{f\}) \subseteq \mathcal{I}_t(X')$. Thus, $X' \sqsubseteq X$. We now prove that no edge $e' \in X$ can belong to $\mathcal{I}_t(X')$. By contradiction, assume that every p acyclic from e' to t passes through $X' \setminus \{e'\}$. Since $X' \sqsubseteq X$ and p is acyclic, then p passes through $X \setminus \{e'\}$, in contradiction with minimality of X.

To conclude:

$$\begin{aligned} (X_e)^f &\triangleq X' \setminus \mathcal{I}_t(X') \\ &= (X \cup (prev(e) \setminus (P \cup \{f\})) \cup B) \setminus (prev(e) \cup B) \\ &= X \end{aligned}$$

□

Let X be the MCS $\{(s, v_2), (v_1, v_2), (v_1, v_3), (v_1, v_4)\}$ for the graph in Figure 2(b) and let $e = (v_1, v_2)$; so, $X_e = \{(s, v_1), (s, v_2)\}$. Now, let $f = (s, v_1) \in prev(e)$, then $(X_e)^f = \{(s, v_2), (v_1, v_2), (v_1, v_3), (v_1, v_4)\} \setminus \emptyset = \{(s, v_2), (v_1, v_2), (v_1, v_3), (v_1, v_4)\} = X$.

3.3 Completeness

To conclude, we show that the way of incrementally generating $\sqsubseteq$ through the successor of a MCS (and, dually, the generation of $\preceq$ through the predecessor of a MCS) covers the set of all MCSs. More precisely, we now show that, by starting from $out(s)$, it is possible to generate every MCS by taking successors of already generated MCSs. To this aim, we assign a *level* to every MCS and show that this induces a partition of the set of all MCSs.

Definition 3.4 (Level of a MCS). *The level of a MCS is defined as follows:*

- $lev(out(s)) = 0$
- $lev(X^e) = \begin{cases} k+1 & \text{if } lev(X) = k, e \in X, h(e) = k \\ k & \text{if } lev(X) = k, e \in X, h(e) < k \end{cases}$

Let X^e be the MCS in Figure 2(c), where $X = \{(s, v_1), (s, v_2)\} = out(s)$ and $e = (s, v_2)$. Since X has level 0 and $h(e) = 0 = lev(X)$, then $lev(X^e) = 0+1 = 1$.

Lemma 3.6. *If $lev(X) = k$, then $h(e) \leq k$, $\forall e \in X$.*

Proof. By structural induction on the partial order $\sqsubset$.

Base By Lemma 3.3, $X = out(s)$; then, $h(e) = 0$, $\forall e \in X$.

Inductive step Let $X = Y^{e'}$, for some MCS Y (with $lev(Y) \leq lev(X)$) and $e' \in Y$. By Lemma 3.4, $Y \sqsubset X$. Let $e \in X$. If $e \in Y$, then by induction $h(e) \leq lev(Y) \leq lev(X)$. Otherwise, $e \in next(e')$. If $h(e') < lev(Y)$, then $h(e) \leq lev(Y) \leq lev(X)$. If $h(e') = lev(Y)$, then $h(e) = lev(Y) + 1 = lev(X)$. □

Hence, the main thing to prove is that *every* MCS can be assigned a level i, for $0 \leq i \leq d(t) - 1$ (where $d(t)$ is the length of a shortest path from s to t).

Definition 3.5 (Clustering of MCSs according to level). *L_i is the set of MCSs at level i i.e., $L_i \triangleq \{X \mid X \text{ is a MCS} \wedge lev(X) = i\}$.*

Theorem 3.1. *If X is a MCS, then there exists $i \in \{0, \ldots, d(t) - 1\}$ such that $i = lev(X)$ and $X \in L_i$.*

Proof. By structural induction on the partial order $\sqsubset$.

Base By Lemma 3.3, $X = out(s)$ is the bottom element of $\sqsubset$ and, by Definitions 3.4 and 3.5, $X \in L_0$.

Inductive Step Let $X \neq out(s)$; because of Lemma 3.5, $X = (X_e)^f$, for some $e \in X$ and $f \in prev(e) \cap X_e$ (hence, $h(f) < d(t) - 1$). By Lemma 3.4, $X_e \sqsubset X$; so, by induction, $X_e \in L_i$, for $i = lev(X_e)$ and $0 \leq i \leq d(t) - 1$. Hence, $X \in L_{i+1}$, if $h(f) = lev(X_e)$ (notice that, thanks to Lemma 3.6, $i < d(t) - 1$ and hence $i + 1 \leq d(t) - 1$), and $X \in L_i$, otherwise. □

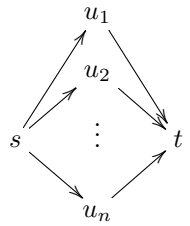
4 Polynomially Checking Edge-Weakness

4.1 The Algorithm in a Nutshell

The basic ideas of our algorithm for edge-weakness are the same as those for node-weakness from [8], but are in terms of MCSs instead of MVSs. The first ingredient is given by the order relation $\sqsubseteq$. A sequence of MCSs $X_0, X_1, \ldots, X_n$

is a *chain* if, for all i $(0 \leq i < n)$, we have $X_i \sqsubset X_{i+1}$; it is *complete* if $X_0 = out(s)$, $X_n = in(t)$ and $X_i \sqsubseteq X \sqsubseteq X_{i+1}$ implies $X = X_i$ or $X = X_{i+1}$, for every MCS X. The second ingredient is the fact that a complete chain contains at most $|E_G|$ MCSs and that every complete chain can be built from the bottom MCS $out(s)$ by proceeding as described in Section 3.2. The third ingredient is Theorem 2.1, according to which a graph is edge-weak iff it contains a MES (i.e. a MCS) X and a path of the form $\xrightarrow{e} \rightsquigarrow \xrightarrow{e'}$ such that $e, e' \in X$. In this case, we call *critical* both the path and e' and X; sometimes, we shall also call X *e'-critical*. A MCS that contains e' and it is minimal (w.r.t. $\sqsubseteq$) satisfying this property is called *e'-minimal*.

The aim of the algorithm is finding a critical MCS. However, the following example (simplified from [14]) shows that we can have exponentially many MCSs:



Indeed, every $S \subseteq \{1, \ldots, n\}$ induces a MCS $\{(s, u_i) : i \in S\} \cup \{(u_i, t) : i \in \{1, \ldots, n\} \setminus S\}$. Thus, this graph has $2n$ edges and 2^n MCSs. So, we need a smart way to find a critical one.

The polynomial algorithm generates a complete chain of MCSs (that contains at most $|E_G|$ MCSs); then, for every new edge e appeared in all such MCSs, the algorithm computes an e-minimal MCS (by backward generating a decreasing chain of at most $|E_G|$ MCSs, all containing e) and it checks if it is e-critical. Soundness of this procedure is proved by showing that every complete chain of MCSs contains a critical edge (Theorem 4.1), and that, if e is critical, then all e-minimal MCSs are e-critical (Theorem 4.2).

In what follows, we always assume the graph G *st-connected*, that is every node and edge belong to a path from s to t. Nodes/edges that do not belong to a path from s to t are not relevant and can be easily removed from G by computing the subgraph that is both reachable from s and backward reachable from t in $O(|E|)$.

4.2 Key results

The first key result underlying our algorithm is that every complete chain of MCSs of an edge-weak graph contains a MCS with a critical edge (Theorem 4.1). For example, consider the complete chain C in Figure 3 for the graph in Figure 2(b). Since G admits the path $s\, v_1\, v_4\, v_3$, the edge (v_4, v_3) is critical in $\{(s, v_1), (v_4, v_3), (v_4, t)\}$ (the third MCS of C). However every chain may contain a different critical edge. For example, consider the chain C' in Figure 3: the critical edge (v_4, v_3) does not belong to any MCS in C'; however, C' contains (v_3, t) that is a critical edge in $\{(v_1, v_4), (v_2, v_4), (v_3, t)\}$ (the fourth MCS of C').

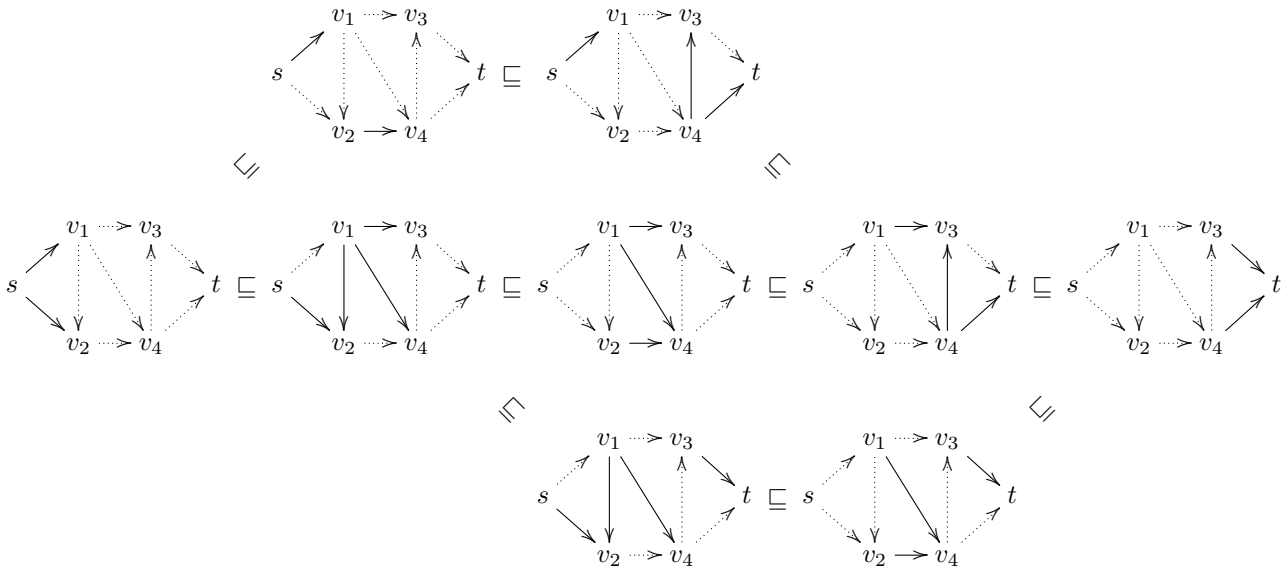


Figure 3: Three chains (C top, C' bottom, C'' middle) of MCSs of the graph in Figure 2(b). The non-dashed edges form the MCS.

To prove this first key result, we need the following lemma:

Lemma 4.1. *Let H be a set of edges and $e \notin H$. If $H \preceq e$ or $e \sqsubseteq H$, then $H \cup \{e\} \not\subseteq X$ for every MCS X.*

Proof. Looking for a contradiction, assume that there is an edge $e = (u, v)$ such that $H \preceq e$ and a MCS X such that $H \cup \{e\} \subseteq X$. Since X is an edge separator, all paths from s to t cross X in some edge of X. Consider a path $s \rightsquigarrow u \to v \rightsquigarrow t$; since X contains H and $H \preceq e$, if e is removed from X, what we obtain would still be an edge separator. Indeed all paths $s \rightsquigarrow v$ ending with e would pass through H. Hence X would not be a MES and, by Prop. 3.1, cannot be a MCS. The same happens if $e \sqsubseteq H$. □

Theorem 4.1. *If G is edge-weak, then in every complete chain of MCSs $X_0, X_1, \ldots, X_n$ there exists at least a X_i that contains a critical edge.*

Proof. Since G is edge-weak, there is an e-critical MCS X, where $e = (u, v)$. By contradiction, assume that there exists a complete chain $X_0, X_1, \ldots, X_n$ such that every X_i does not contain any critical edge. Since G is st-connected, $e \not\sqsubseteq X_0$ and $e \sqsubseteq X_n$ (this happens for every $e \in E \setminus out(s)$); thus, let i be an index such that $e \not\sqsubseteq X_i$ and $e \sqsubseteq X_{i+1}$, By assumption, $e \notin X_i$ and $e \notin X_{i+1}$.

Let $prev^*(e) = \{e' \in E \mid \exists s \rightsquigarrow u \to v \text{ s.t. } e' \in s \rightsquigarrow u \to v\}$ be the set of many-steps predecessors of e, and let $P = prev^*(e) \cap X_i$ be the set of many-steps predecessors of e in X_i.

(i) By construction, $e \not\sqsubseteq X_i$; this implies that $X_i \preceq e$. Indeed, since X_i is a MCS, all paths from the source node to the target node must cross X_i; so either every path from s to v ending with e passes through X_i or every path from u to t starting with e passes through X_i. Since $X_i \preceq e$ and P is built by removing from X_i those edges that do not appear in the paths from s to v ending with e, we have that $P \preceq e$; thus trivially $P \neq \emptyset$.

(ii) Since $P \subseteq X_i$ and by hypothesis $X_i \sqsubseteq X_{i+1}$, also $P \sqsubseteq X_{i+1}$. Thus, for every $e' \in P$, $e' \sqsubseteq X_{i+1}$ and $e' \notin X_{i+1}$. Hence, $\mathit{next}(e') \sqsubseteq X_{i+1}$ and so $X_i^{e'} \sqsubseteq X_{i+1}$. However, $X_i^{e'} = X_{i+1}$ otherwise $X_0, X_1, \ldots, X_n$ would not be a complete chain (there would exist a MCS $X_i^{e'}$ such that $X_i \sqsubset X_i^{e'} \sqsubset X_{i+1}$). So $X_i^{e'} = X_{i+1}$, for all $e' \in P$.

(iii) Let $U = X_{i+1} \setminus X_i$ be the set of new edges added from X_i to X_{i+1}.Then, $e \sqsubseteq U$, otherwise there would be a path from u to t starting with e passing either through an edge $e' \in X_i \setminus X_{i+1}$ or an edge $e' \in X_i \cap X_{i+1}$; both these cases are not possible otherwise by point (i) X_i would be critical.

(iv) If some $e' = (u', v') \in P$ belonged to X_{i+1}, we could find a path $u' \to v' \rightsquigarrow u \to v \rightsquigarrow u'' \to v''$ where $(u'', v'') \in X_{i+1}$; this would make X_{i+1} a critical MCS. So, $P \cap X_{i+1} = \emptyset$. Moreover, because $P \cap U = \emptyset$ (trivial consequence of what we have just shown) and because of point (ii), every edge in U is an immediate successor of every edge in P, since $U \subseteq \mathit{next}(e')$, for all $e' \in P$.

We can now contradict the assumption that e is critical. Consider all the paths of the form $s \rightsquigarrow x \to y \to z \rightsquigarrow t$, for all $(x, y) \in P$ and $(y, z) \in U$. Every MCS, being an edge separator, must contain either a set of edges $P' \preceq P$ or a set of edges $U' \sqsupseteq U$. In both cases, since $P' \preceq e$ and $e \sqsubseteq U'$ (because of (i) and (iii)), e cannot belong to any MCS (see Lemma 4.1) and hence cannot be critical. □

Notice that Theorem 4.1 is not enough to conclude that any complete chain of MCSs in an edge-weak graph contains at least a critical MCS. An example for the graph in Figure 2(b) is the chain C'' in Figure 3: no critical MCS is present, even though its third MCS contains the critical edge (v_2, v_4). However, if we take its (v_2, v_4)-minimal predecessor, we obtain a critical MCS (viz., the second MCS of C). This is not incidental, as the following key result proves.

Theorem 4.2. *If e is a critical edge, then all e-minimal MCSs are e-critical.*

Proof. Let X be an e-critical MCS, with $e = (u, v)$. Then, there exists $e' = (u', v') \in X$ and a path $u' \to v' \rightsquigarrow u \to v$. By contradiction, assume the existence of a MCS X^* that is both e-minimal and not e-critical. Clearly, $e' \notin X^*$.

Since X^* is an edge separator, all paths from s to t cross X^*. If $X^* \preceq e'$ then there exists a path that starts in X^* and reaches e'; because of the existence of $u' \to v' \rightsquigarrow u \to v$, X^* would be an e-critical MCS. Therefore, it must be $e' \sqsubseteq X^*$.

Let $A = \{(x,y) \in X^* \mid \exists u' \to v' \rightsquigarrow x \to y \ \wedge \ (x,y) \neq e\} = \{f_1, \ldots, f_n\}$. If $A = \emptyset$ then $e' \sqsubseteq \{e\}$, since X^* is an e-minimal MCS, but this is impossible for Lemma 4.1. So, $A \neq \emptyset$.

For each element $f_i \in A$, consider the MCS $X^*_{f_i} \sqsubset X^*$. By e-minimality of X^*, e does not belong to $X^*_{f_i}$. This implies that there exists a set of edges $P_i \subseteq \mathit{prev}(f_i)$ such that $P_i \preceq e$. Indeed, by construction $X^*_{f_i}$ is made up from (some of) X^*'s edges (except for f_i) and f_i's predecessors. Considering that e cannot be preceded by edges in X^*, e must be preceded by edges that are only in $X^*_{f_i}$, i.e. some of those in $\mathit{prev}(f_i)$.

Note that $P_i \neq \{e'\}$ otherwise $\{e'\} \preceq e$ and so e and e' could not belong to the same MCS because of Lemma 4.1. Therefore, for each $i \in \{1, \ldots, n\}$, consider the set of edges $B_i \triangleq P_i \setminus \{e'\}$. Trivially, $B_i \neq \emptyset$ otherwise all paths from s to v ending with e would pass through e'.

Fix an $i \in \{1, \ldots, n\}$ where $f_i = (x_{f_i}, y_{f_i}) \in A$ and consider all paths of the form $s \rightsquigarrow u_{b_{i_j}} \to v_{b_{i_j}} \rightsquigarrow x_{f_i} \to y_{f_i} \rightsquigarrow t$, for all $b_{i_j} = (u_{b_{i_j}}, v_{b_{i_j}}) \in B_i$. Notice that $v_{b_{i_j}} = x_{f_i}$ because $b_{i_j} \in \mathit{prev}(f_i)$. Given that $b_{i_j} \notin X$ and $f_i \notin X$ (because in the former case $e \notin X$ and in the latter case $e' \notin X$), all these paths must cross X in some way.

(i) If X is crossed in the path $s \rightsquigarrow u_{b_i} \to v_{b_i}$, then there exists $L_j \subseteq X$ such that $L_j \preceq b_{i_j}$. Since this argument works for every b_{i_j}, consider $L \triangleq \bigcup_j L_j \subseteq X$; then, $L \preceq B_i$. Since $L \cup \{e'\} \preceq B_i \cup \{e'\} = P_i \preceq e$ and $L \cup \{e'\} \subseteq X$, because of Lemma 4.1, $e \notin X$.

(ii) If X is crossed in the path $x_{f_i} \to y_{f_i} \rightsquigarrow t$, then there exists $R_i \subseteq X$ such that $f_i \sqsubseteq R_i$. Since this argument works for every i, consider $R \triangleq \bigcup_i R_i \subseteq X$; then, $A \sqsubseteq R$. Since $e' \sqsubseteq A \cup \{e\} \sqsubseteq R \cup \{e\}$ and $R \cup \{e\} \subseteq X$, because of Lemma 4.1, $e' \notin X$.

□

4.3 A Polynomial-time Algorithm

Algorithm 1 receives an st-connected graph G and returns *true* if G is edge-weak, *false* otherwise. To check edge-weakness, a complete chain of MCSs is built by following the approach presented in Section 4.1: we start with X initially equal to $\mathit{out}(s)$ and the *while* is executed until X is different from $\mathit{in}(t)$ or a critical MCS is found. For each iteration of the loop, in Step 3 we generate a new MCS X' that is an immediate successor of the MCS X. This is done by choosing an edge from X and producing a new MCS following equation (1). However, the new MCS is not necessarily an immediate successor (w.r.t. $\sqsubset$) of X. To obtain this, it suffices to consider a minimal (w.r.t. $\sqsubset$) MCS among all the X^e's.

To see why minimality is necessary, consider again the chains in Figure 3. Imagine that we are building C'' (so, we started with $\mathit{out}(s)$ and then generated a new MCS by using (s, v_1)). If we now select (v_1, v_4), we will directly go to the fourth MCS of C'', thus jumping the third one. By contrast, if we select (s, v_2)

Algorithm 1 Checking Edge-Weakness

Input: A directed st-graph $G = (V, E)$, s is the source and t is the sink
function *isEdgeWeak*(G)

```
1: X ← out(s);
2: while X ≠ in(t) do
3:    X' ← immediateMcsRight(X)
4:    if X' is critical then
5:       return TRUE
6:    for all e ∈ X' \ X do
7:       X* ← minimalMcs(X', e)
8:       if X* is critical then
9:          return TRUE
10:   X ← X'
11: return FALSE
```

Algorithm 2 Generating an e-minimal MCS smaller (w.r.t. $\sqsubset$) than X

Input: A mcs X, an edge $e \in X$
function *minimalMcs*(X, e)

```
1: X' ← X
2: while X' ≠ ∅ do
3:    choose f ∈ X
4:    X' ← X' \ {f}
5:    if e ∈ X_f then
6:       X' ← X' ∪ (X_f \ X)
7:       X ← X_f
8: return X
```

or (v_1, v_2), we obtain the third MCS. Notice also that, if we select (v_1, v_3), we still obtain an immediate successor, but we then build C'.

Now, if X' is critical (hence there exists a path between two edges in X'), then G is edge-weak and Algorithm 1 terminates by giving in output *true*. Otherwise, we invoke Algorithm 2 to build a (not necessarily complete) backwards chain $X' = X_0 \sqsupset X_1 \sqsupset \ldots$, for every $e \in X' \setminus X$ (edges in X have already been controlled) and find X^*, an e-minimal predecessor of X'. Finally, criticality of every such X^* is checked. If any of them critical, then Algorithm 1 returns *true*; otherwise the next iteration of the while is executed with X' replacing X. If the main algorithm terminates at Step 11, then G has a chain formed by MCSs without critical edges; hence, it is not edge-weak.

Correctness of Algorithm 1 is given by Theorems 4.1 and 4.2, whereas correctness of Algorithm 2 is given by the following result.

Proposition 4.1. *Let X be a MCS and $e \in X$; then, X is e-minimal if and only if $e \notin X_f$, for every $f \in X$.*

Proof. By observing that all immediate predecessors of a MCS X are of the form X_f, for some $f \in X$. If e does not belong to anyone of them, X is e-minimal. □

Analysis In a precomputation phase, we can calculate the reachability relation for every pair of nodes in $O(|V|^3)$ and fill a $|V| \times |V|$ reachability binary matrix that will allow us to check whether X is critical in $O(|V|^2)$ within Algorithm 1. Indeed, checking whether (u, v) reaches (u', v') corresponds to checking whether v reaches u'.

Function *immediateMcsRight* costs $O(|E|^2)$: for every $f \in X$ (there are at most $|E|$ such edges), we have to build X^e (by paying $O(|E|)$) and checking minimality w.r.t. $\sqsubset$ (again in $O(|E|)$).

Function *minimalMcs* costs $O(|E|^2)$: there are at most $O(|E|)$ iterations of the *while* at Step 2 in Alg. 2 and each iteration costs $O(|E|)$ (see Step 5).

So, function *isEdgeWeak* costs $O(|E|^3)$. Indeed:

1. function *immediateMcsRight* is invoked at most $|E|$ times, that is the maximum length of an increasing chain of MCSs;

2. function *minimalMcs* is invoked at most once for each edge in E. Indeed, if e appears as a new edge in $X' \setminus X$ (line 6 Algorithm 1), it cannot have already appeared in $X'' \sqsubset X'$ of the chain, otherwise there would exists a path from e to e and hence X'' (as well as X') would be critical; consequently, function *isEdgeWeak* would have terminated in line 5 returning *true* as result.

5 Conclusions

In this paper we provide a polynomial-time algorithm for deciding whether, given a directed graph with a fixed source and target node, there exists a

capacity-to-edges assignment such that the resulting flow network admits a non-maximum saturating flow. To this aim, we also defined two order relations and used them to build up a complete chain of MCSs. In doing so, we also proved that MCSs are minimal sets whose removal disconnects source and target.

In [8], an algorithm for node-weakness is given, by finding a critical MVS in a node-capacitated net. As we discussed in the introduction, it is not possible to exploit the algorithm for node-weakness for checking edge-weakness: while edge-weakness of G implies node-weakness of the edge-expansion of G, the converse does not hold. This happens because not all the MVSs in the edge-expanded graph have their counterpart in the original graph. Indeed, the MVS $\{v_1\}$ in Figure 1(b) (which causes node-weakness) has not a corresponding MCS in Figure 1(a).

It is worth saying that we could adapt the algorithm in [8] to check edge-weakness: to this aim, we have to consider only the MVSs that are formed by 'new' nodes (i.e., those arising from the edge-expansion procedure). However, this essentially consists in running the algorithm we provide in this paper: indeed, there is a one-to-one correspondence between the MCSs in the original graph and the MVSs formed only by 'new' nodes in the edge-expanded graph. For this reason we prefer to develop the algorithm for edge-weakness directly on the original graph; this saves us the burden of calculating the edge-expansion, of adjusting the algorithm for node-weakness to only consider 'new' nodes, and of running the modified algorithm on the edge-expanded graph (that is bigger than the original one, since it has $|V|+|E|$ nodes and $2|E|$ edges). Furthermore, adapting the algorithm for node-weakness would have required proving its correctness: indeed, the soundness proofs developed in [8] would not hold anymore and should have been adapted, essentially by providing the proofs we gave in Section 4.2.

The only real advantage in modifying the algorithm for node-weakness would be that the theory of MVSs (their ordering and their enumeration) is already known [12, 20], whereas we have to adapt this theory to MCSs. However, working with the MVSs of the edge-expanded graph would have been computationally much more expensive, since MVSs are exponential in the number of nodes and the edge-expansion can have quadratically many more nodes than the original graph. Furthermore, we believe that the theory of MCSs is also an interesting contribution of our paper on its own.

To conclude, we implemented our algorithms in Python and run some experiments. The overall results are given in Figure 4; in particular, such a diagram shows the results of the experiments conducted on 2.109.801 graphs obtained by fixing the number of nodes (from 2 to 6 in the picture) and building all the possible directed *st*-graphs with such node sets. The test consists in running the function `is_edge_weak(G)` for measuring the relation between edge-weakness and the number of nodes and edges. For every fixed number of nodes, graphs with few arcs are always non-edge-weak; the more edges we add, the higher the likelihood that the graph becomes edge-weak, until a number of edges after which all resulting graphs are edge-weak.

These tests confirm the intuition that sparse graphs are usually non-edge-

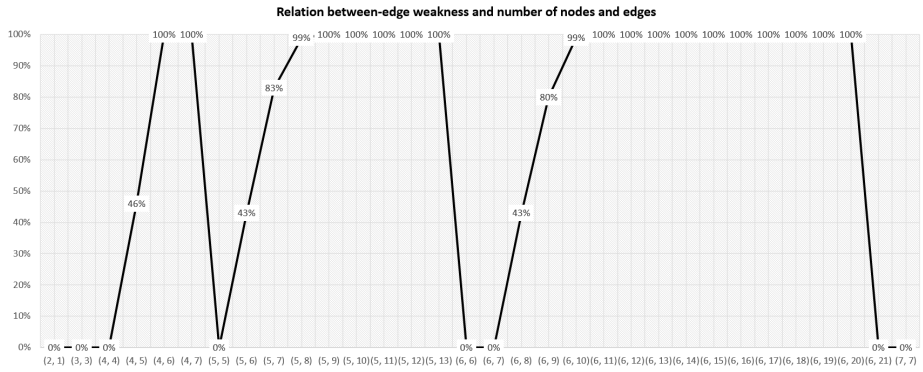


Figure 4: Diagram representing the relation between edge-weakness and amount of nodes and edges. The horizontal axis is labeled with pairs of nodes-edges and the vertical axis is labeled with the probabilities (in percentages) of the graph to be edge weak

weak. A challenging direction for future research is to find the best topology for a fixed number of nodes, i.e. to determine the best set of connections among nodes so that the resulting graph only admits maximum saturating flows, independently of the capacity chosen.

A NP-completeness

The problem of checking whether a given flow network has a non-maximum saturating flow is an NP-hard problem. We show this by reducing the analogous problem defined in [7] for channels (that, in *loc.cit.* has been proved to be NP-hard by reduction from maximal matching in a bipartite graph) to the edge-weakness problem defined here for flow nets.

To this aim, recall that a *channel* is a pair (G, η) where $G = (V, E)$ is an st-graph and $\eta : V \mapsto \mathbb{N}$. A flow for a channel is a function $\phi : E \to \mathbb{N}$ such that $\phi(in(v)) = \phi(out(v)) \leq \eta(v)$, for all $v \in V \setminus \{s, t\}$. Again, the value of ϕ is $\phi(out(s))$. The resulting channel after ϕ is $(G, \eta - \phi)$, where $(\eta - \phi)(v) \triangleq \eta(v) - \phi(out(v))$, for all $v \in V \setminus \{s, t\}$. A flow ϕ inhibits channel (G, η) if the resulting channel after η has maximum flows of value 0.

In [7] the following problem is defined:

> **DMIF**: Given a channel and an integer k, is there an inhibiting flow of value at most k?

and it is proved to be NP-complete. Let fnDMIF be the corresponding problem in the context of flow networks; we now easily prove that also fnDMIF is NP-complete.

Theorem A.1. *fnDMIF is NP-complete*

Proof. Easily, fnDMIF is in NP: this condition holds because, given a flow, we can polynomially check whether its value is smaller than k and it is saturating. Then, fnDMIF is NP-hard, because of the following polynomial reduction of DMIF to it.

Let (G, η) be a channel and consider the following well known transformation of the node-capacitated in the edge-capacitated model, called *node splitting* [1, Section 2.4]. Every vertex v different from s and t of a channel (G, η) is replaced by two new vertices v_ι (*input vertex*) and v_o (*output vertex*), connected by the edge (v_ι, v_o) whose capacity is $\eta(v)$; edges of the form (u, v), (s, v), and (v, t) in the original graph are replaced with (u_o, v_ι), (s, v_ι), and (v_o, t) respectively, and all have capacity ∞. This construction yields the flow network (G', c). We now show that (G, η) has an inhibiting flow of value k iff (G', c) has a saturating flow of value k.

(Only if) Assume ϕ is an inhibiting flow of value k for (G, η) and let $f : E_{G'} \to \mathbb{N}$ be such that:

$$\begin{array}{lcllcl} f((u_\iota, u_o)) & = & \phi(out(u)) & f((u_o, v_\iota)) & = & \phi((u, v)) \\ f((s, u_\iota)) & = & \phi((s, u)) & f((u_o, t)) & = & \phi((u, t)) \end{array}$$

Trivially, f is a flow of value k for (G', c). Furthermore, f saturates (G', c): suppose by contradiction that there exists in $(G', c - f)$ a non-saturated path $su_\iota^1 u_o^1 \dots u_\iota^n u_o^n t$. By construction, this implies that $su^1 \dots u^n t$ is a non-saturated path in $(G, \eta - \phi)$, in contradiction with the fact that ϕ inhibits (G, η).

(If) Assume that f is an saturating flow of value k for (G', c). This means that there exists an edge separator X in G' saturated by f; so, X is formed only by arcs of kind (v_ι, v_o), since these are the only ones with a finite capacity. By [19, Corollary 11.2c], there exists a flow f' of value k for (G', c) such that $\lfloor f(e) \rfloor \leq f'(e) \leq \lceil f(e) \rceil$, for every $e \in E_{G'}$. Furthermore, f' still saturates X, since $f'(e) = f(e) = \eta(v) \in \mathbb{N}$, for every $e = (v_\iota, v_o) \in X$. Now, let $\phi : E_G \to \mathbb{N}$ be such that

$$\begin{array}{lcl} \phi((s, u)) & = & f'((s, u_\iota)) \\ \phi((u, t)) & = & f'((u_o, t)) \\ \phi((u, v)) & = & f'((u_o, v_\iota)) \end{array}$$

Like before, ϕ is an inhibiting flow of value k for (G, η). □

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows, theory, algorithms, and applications.* Prentice-Hall, New Jersey, 1993.

[2] D. S. Altner, O. Ergun, and N. A. Uhan. The maximum flow network interdiction problem: Valid inequalities, integrality gaps, and approximability. *Oper. Res. Lett.*, 38(1):33–38, Jan. 2010.

[3] T. Aura, M. Bishop, and D. Sniegowski. Analyzing single-server network inhibition. In *Proceedings 13th IEEE Computer Security Foundations Workshop. CSFW-13*, pages 108–117, 2000.

[4] M. Benaddy and M. Wakrim. Cutset enumerating and network reliability computing by a new recursive algorithm and inclusion exclusion principle. *International Journal of Computer Applications*, 45(16):22–25, 2012.

[5] D. Braess, A. Nagurney, and T. Wakolbinger. On a paradox of traffic planning. *Transportation Science*, 39(4):446–450, 2005.

[6] T. X. Brown, H. N. Gabow, and Q. Zhang. Maximum flow-life curve for a wireless ad hoc network. In *Proceedings of the 2Nd ACM International Symposium on Mobile Ad Hoc Networking &Amp; Computing*, MobiHoc '01, pages 128–136. ACM, 2001.

[7] P. Cenciarelli, D. Gorla, and I. Salvo. Depletable channels: Dynamics and behaviour. In *Proc. of FCT*, pages 50–61. Springer, 2009.

[8] P. Cenciarelli, D. Gorla, and I. Salvo. Graph theoretic investigations on inefficiencies in network models. *CoRR*, abs/1603.01983, 2016.

[9] P. Cenciarelli, D. Gorla, and I. Salvo. Inefficiencies in network models: A graph-theoretic perspective. *Information Processing Letters*, 131:44–50, 2018.

[10] J.-H. Chang and L. Tassiulas. Maximum lifetime routing in wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(4):609–619, 2004.

[11] B. S. Chlebus, M. Chrobak, and K. Diks. Saturating flows in networks. In *International Conference on Fundamentals of Computation Theory*, pages 82–91. Springer, 1987.

[12] F. Escalante. Schnittverbände in graphen. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 38(1):199–220, 1972.

[13] B. Klinz and G. J. Woeginger. Minimum-cost dynamic flows: the series-parallel case. *Networks*, 43(3):153–162, 2004.

[14] T. Kloks and D. Kratsch. Listing all minimal separators of a graph. *SIAM Journal on Computing*, 27(3):605–613, 1998.

[15] H.-Y. Lin, S.-Y. Kuo, and F.-M. Yeh. Minimal cutset enumeration and network reliability evaluation by recursive merge and bdd. In *Proceedings of the Eighth IEEE Symposium on Computers and Communications. ISCC 2003*, pages 1341–1346 vol.2, 2003.

[16] C. A. Phillips. The network inhibition problem. In *Proc. of STOC*, pages 776–785. ACM Press, 1993.

[17] J. Riordan and C. Shannon. The number of two-terminal series-parallel networks. *Journal of Mathematics and Physiscs*, 21:83–93, 1942.

[18] V. Rodoplu and T. H. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, 1999.

[19] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, 2003.

[20] H. Shen and W. Liang. Efficient enumeration of all minimal separators in a graph. *Theor. Comput. Sci.*, 180(1-2):169–180, 1997.

[21] H. O. Shuji Tsukiyama, Isao Shirakawa and H. Ariyoshi. An algorithm to enumerate all cutsets of a graph in linear time per cutset. *Journal of the ACM (JACM)*, 27:619–632, 1980.

[22] P. B. Wolfgang W. Bein and A. Tamir. Minimum Cost Flow Algorithms for Series-Parallel Networks. *Discrete Applied Mathematics 10*, pages 117–124, 1985.

[23] Y. Wu, P. A. Chou, and S.-Y. Kung. Minimum-energy multicast in mobile ad hoc networks using network coding. *IEEE Transactions on Communications*, 53(11):1906–1918, 2005.