RAIRO-Theor. Inf. Appl. 59 (2025) 12
https://doi.org/10.1051/ita/2025010

RAIRO - Theoretical Informatics and Applications
www.rairo-ita.org

SUPERFLUOUS ARCS AND CONFLUENT REDUCTIONS IN THE MINIMUM FEEDBACK VERTEX SET PROBLEM

Moussa Abdenbi[1,*], Alexandre Blondin Massé[1], Alain Goupil[2] and Odile Marcotte[3]

Abstract. Given a directed graph (digraph) G with vertex set V, a *Feedback Vertex Set* (FVS) is a subset of vertices whose removal eliminates all circuits in G. Finding a minimum feedback vertex set (MFVS) is NP-hard, but digraph reductions can reduce graph size while preserving at least one MFVS. This raises questions about the ordering in which reductions are applied and the existence of an optimal order that maximizes size reduction. The Church-Rosser property (confluence) ensures reductions can be applied in any order, leading to a unique reduced digraph up to isomorphism. In this work, we focus on arc reduction and its confluence within a broader set of known confluent reductions. We introduce *Superfluous Arcs*, which can be removed without affecting MFVS solutions, and propose a new parametrized reduction, $\textsc{chord}_k$, to identify and remove specific superfluous arcs in polynomial time for bounded integer k. We establish the confluence of a set of reductions that includes $\textsc{chord}_k$, creating the largest known confluent reduction system for MFVS, which improves preprocessing techniques for solving the MFVS problem efficiently.

Mathematics Subject Classification. 05C20, 05C85, 68Q25.

Received March 7, 2025. Accepted September 6, 2025.

1. Introduction

Given a directed graph (digraph) G with vertex set V, a *feedback vertex set* is a subset U of V such that every circuit of G visits at least one vertex of U. Although finding a minimum cardinality FVS (MFVS) is NP-hard [1, 2], *digraph reductions* can be applied to help solving the MFVS problem by reducing the digraph size, while preserving the collection of MFVS or at least one of them [3–6]. These reductions either remove some arcs or some vertices, or bypass a vertex by redirecting its incoming arcs to its outgoing targets, effectively connecting each predecessor to each successor and removing the given vertex, while ensuring that at least one MFVS of the original digraph can be constructed in polynomial time from any MFVS of the resulting reduced digraph. This naturally leads us to question the order in which these reductions should be applied. Given a digraph and a set of reductions, does there exist an optimal application order that decreases the size of the original digraph? If so, is this order unique? If multiple application orders exist, do they all lead to the same final reduced digraph? To

Keywords and phrases: Directed graphs, graph reductions, superfluous arcs, Church–Rosser property, confluence, minimum feedback vertex set problem.

[1] Université du Québec À Montréal.
[2] Université du Québec À Trois-Rivières.
[3] Group for Research in Decision Analysis (www.gerad.ca).

[*] Corresponding author: abdenbi.moussa@uqam.ca

address these questions, the Church-Rosser property [7], or confluence [8], if satisfied by the set of reductions, provides a strong answer. Indeed, this property guarantees that regardless of the order in which the reductions from a given set are applied, the final digraph is unique up to isomorphism. As a result, there is no need for costly enumeration-based strategies or heuristic approximations to determine an optimal application order. Instead, with the guarantee of uniqueness for the reduced digraph, the focus is shifted towards the algorithmic efficiency of the reductions to minimize computational resource consumption and towards potential parallelization and speedup of sequential algorithms in graph classes with polynomial algorithms [9, 10].

In this article, we focus on arc reductions and their confluence with a set of confluent reductions introduced in [11]. We first introduce and study a class of arcs that we refer to as *superfluous arcs*. This class encompasses arcs that can be removed from a digraph while ensuring that *all* MFVS of the original digraph are preserved in the reduced digraph. Reductions identifying this type of arcs have already been introduced in previous works, such as PIE and DOME [6], and more recently, DOME++ [3]. We then introduce a new parametrized reduction CHORD$_k$, that identifies and removes special types of superfluous arcs. In particular, the identification of the arcs that can be removed by CHORD$_k$ can be done in polynomial time with respect to a fixed parameter k. Finally, we show that CHORD$_k$ is confluent when included in a set of reductions studied in [11]. To the best of our knowledge, this is the largest known confluent set of reductions for the minimum feedback vertex set problem.

The rest of this article is structured as follows. In Section 2, we review the fundamental graph theory concepts required for this work, along with the formal definition of confluence. In Section 3, we introduce the definition of reduction adopted in this study and recall the set of reductions considered for the confluence analysis. In Section 4, we define the class of superfluous arcs, examining their properties and providing examples. In Section 5, we introduce the CHORD$_k$ reduction, present an algorithm implementing it, prove its correctness, and analyze its complexity. Finally, in Section 6, we establish the confluence of the CHORD$_k$ reduction with the set of reductions introduced in Section 3. We also provide a brief comparison between the CHORD$_k$ reduction and other superfluous arcs reductions.

2. Preliminaries

We briefly recall general definitions about directed graphs.

A *directed graph* (or *digraph*) is an ordered pair $G = (V, E)$ where V is a finite set of *vertices* and $E \subseteq V \times V$ is a set of arcs. Let $G = (V, E)$ be a digraph and $u, v \in V$ and $U \subseteq V$. We denote $N_G^+(u) = \{s \in V \mid (u, s) \in E\}$ and $N_G^-(u) = \{p \in V \mid (p, u) \in E\}$. For a vertex $u \in V$, we denote $G - u$ the digraph where the set of vertices is $V' = V \setminus \{u\}$ and the set of arcs is $E' = E \setminus \{(x, y) \in E \mid x = u \text{ or } y = u\}$. Also for a subset $U \subseteq V$ we write $G - U = (V', E')$ with $V' = V \setminus U$ and $E' = E \setminus \{(x, y) \in E \mid x \in U \text{ or } y \in U\}$.

Similarly, for a given arc $(u, v) \in E$ we denote by $G - (u, v)$ the digraph where the set of vertices is V and the set of arcs is $E' = E \setminus \{(u, v)\}$. Also for a subset $A \subseteq E$ we write $G - A = (V, E \setminus A)$. If $(u, u) \notin E$ then the digraph $G \circ u$ is the digraph where the set of vertices is $V' = V \setminus \{u\}$ and the set of arcs is $E' = (E \setminus \{(x, y) \in E \mid x = u \text{ or } y = u\}) \cup N_G^-(u) \times N_G^+(u)$.

For $e = (u, v) \in E$, we say that e is a *2-way arc* in G if $(v, u) \in E$. The set of all 2-way arcs of G is then $E^{\leftrightarrow} = \{(u, v) \in E \mid (v, u) \in E\}$. Given a digraph $G = (V, E)$ we distinguish two special digraphs $G^{\leftrightarrow} = (V, E^{\leftrightarrow})$ and $G^{\rightarrow} = G - E^{\leftrightarrow}$.

A *(directed) path* of G is a sequence $p = \langle v_1, v_2, \ldots, v_k \rangle$ of vertices $v_i \in V$ for $i = 1, 2, \ldots, k$ such that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \ldots, k - 1$. The length of directed path $p = \langle v_1, v_2, \ldots, v_k \rangle$ is the number of arcs of p. We say that p *starts at* v_i and ends at v_k. Given $u, v \in V$, we denote by $\mathcal{P}_G(u, v)$ the set of all directed paths in G that start at u and end at v. Moreover, p is called a *circuit* if $v_1 = v_k$. We denote by $V(p) = \{v_1, v_2, \ldots, v_k\}$ the set of vertices of p and by $E(p) = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)\}$ the set of arcs of p. For a given circuit $c = (u_0, u_1, u_2, \ldots, u_{k-1}, u_0)$, we say that c is *minimal* if there does not exist another circuit c' with a vertex set $V(c') \subset V(c)$. An arc (u, u) is called a *loop*. We say that G is *acyclic*, if there is no circuit in G. Given $U \subseteq V$, we say that U is a *directed clique* or *diclique* of G if $(u, v) \in E$ and $(u, u) \notin E$ for each $u, v \in U$, $u \neq v$. For a given arc (u, v) in G we say that (u, v) is *acyclic* if there is no circuit in G that passes through the arc (u, v).

Finally, let $U \subseteq V$. The set U is called a *feedback vertex set* (FVS) if $G' = G - U$ is acyclic. A *minimum feedback vertex set* (MFVS) is a feedback vertex set of G with minimum cardinality.

Let $\mathcal{B} \subseteq \mathcal{F} \times \mathcal{F}$ be any binary relation on a set $\mathcal{F}$. We write $x\mathcal{B}y$ whenever $(x, y) \in \mathcal{B}$. The *transitive closure* of the relation $\mathcal{B}$ is defined by $\mathcal{B}^T = \bigcup_{i=1}^{\infty} \mathcal{B}^i$ where $\mathcal{B}^i$ is the composition of $\mathcal{B}$ with itself $i - 1$ times. A pair $(\mathcal{F}, \mathcal{B})$ is called *finite* if for $x, y \in \mathcal{F}$, there is a constant k such that if $x\mathcal{B}^i y$, then $i \leq k$. A pair $(\mathcal{F}, \mathcal{B})$ is said to have the *Church-Rosser finiteness property* if it satisfies the following conditions.

Definition 2.1 (Aho, Sethi and Ullman [12])**.** Let $\mathcal{B}$ be a relation on a set $\mathcal{F}$. Then $(\mathcal{F}, \mathcal{B})$ is *Church-Rosser finite* if and only if $(\mathcal{F}, \mathcal{B})$ is finite and, for all $x, y, z \in \mathcal{F}$, the conditions $x\mathcal{B}y$ and $x\mathcal{B}z$ imply that there exists $w \in \mathcal{F}$ such that $y\mathcal{B}^T w$ and $z\mathcal{B}^T w$.

This property ensures that for any two sequences of binary relations from $\mathcal{B}$ applied to the same element in $\mathcal{F}$, it they lead to different intermediate values, then these two values can always be joined to a common value. The Church-Rosser finiteness property is also called *confluence* [8] and, from now on, we use this expression for sake of brevity.

3. Reductions

Given a digraph $G = (V, E)$, the problem of finding a minimum feedback vertex set is NP-hard [1]. However, in some cases we can use a set of *transformations* that reduce the size of the input digraph, with the guarantee that at least one minimum feedback vertex set in G can be constructed from a minimum feedback vertex set in the reduced digraph, in polynomial time. These transformations are called *reductions*. More formally, a reduction is a transformation of the digraph $G = (V, E)$ into a digraph $G' = (V', E')$ such that the following two conditions are satisfied.

(1) $(|V'|, |E'|) \prec_{lex} (|V|, |E|)$, where $\prec_{lex}$ is the strict lexicographic order on $\mathbb{N}^2$;
(2) An MFVS of G can be computed in polynomial time from any MFVS of G'.

In the following, we give a brief recall of the reductions studied by Abdenbi, Blondin Massé, Goupil and Marcotte [11], originating from the works of Levy and Low [5], Lemaic [4] and Lin and Jou [6].

In the following, a *precondition* is a predicate that applies to a digraph. It can be a condition on a vertex or an arc, based on its neighborhood or the entire digraph. In a given digraph, there may be multiple vertices or edges that satisfy one or more preconditions. Let $G = (V, E)$ be a digraph and $u, v \in V$.

- The precondition of the reduction LOOP(u) is the predicate "$(u, u) \in E$". This reduction transforms $G = (V, E)$ in $G - u$ and adds u to the MFVS in construction. See Figure 1a for an illustration.
- The precondition of PIE(u, v), for an arc (u, v) of G that is not a 2-way arc, is "there is no circuit in the digraph $G^{\to}$ going through arc (u, v)". The transformation consists of replacing G with $G - (u, v)$. This reduction preserves all the FVS of the original digraph. See Figure 1b for an illustration.
- The precondition of INDICLIQUE(u) is the predicate "$(u, u) \notin E$ and $N_G^-(u)$ forms a diclique in G". The transformation consists of replacing G by $G \circ u$. This reduction does not necessarily preserve all the FVS of the original digraph but every MFVS of the reduced digraph is also an MFVS of the original digraph. See Figure 1c for an illustration.
- The precondition of OUTDICLIQUE(u) is "$(u, u) \notin E$ and $N_G^+(u)$ forms a diclique in G". The transformation consists of replacing G by $G \circ u$. Similarly to INDICLIQUE(u), this reduction does not necessarily preserve all the FVS of the original digraph but every MFVS of the reduced digraph is also an MFVS of the original digraph. See Figure 1d for an illustration.

As introduced in [11], a reduction R, together with its associated precondition P, can be seen as a binary relation $\mathcal{R}$ on $\mathcal{G}$, where $\mathcal{G}$ is the set of all digraphs. Hence, for $G, G' \in \mathcal{G}$ if G satisfies the precondition P and if we can reduce G to G' with the reduction R, then we write $(G, G') \in \mathcal{R}$. We say then that the relation $\mathcal{R}$ is *based* on the reduction R and the precondition P. From now on, we use the following notation for the binary relations based on the reductions introduced above: $\mathcal{R}_L$ for LOOP, $\mathcal{R}_I$ for INDICLIQUE and $\mathcal{R}_O$ for OUTDICLIQUE and $\mathcal{R}_P$

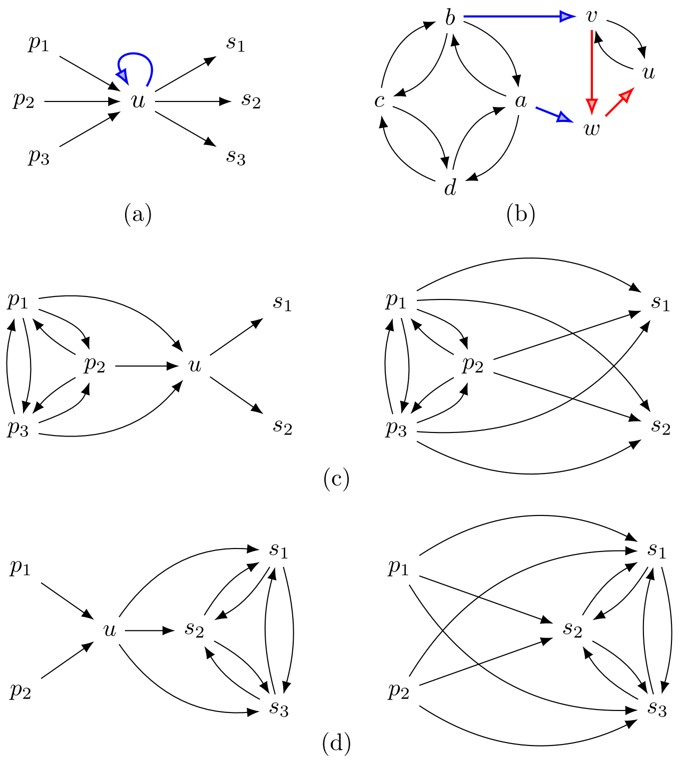


FIGURE 1. Illustration of the reductions. (a) The loop (u, u) in blue means that LOOP(u) is applicable, so we add u to the MFVS and we reduce G to $G - u$. (b) PIE is applicable on the blue and red arcs. Indeed, there is no circuit going through the blue arcs (b, v) and (a, w) in G, the same goes in $G^{\rightarrow}$ and the same goes for the red arcs (v, w) and (w, u) in $G^{\rightarrow}$. So we can remove the blue and red arcs from G. (c) $N^-_G(u)$ is a diclique, so INDICLIQUE(u) is applicable. We remove u and all its incident arcs, and we add the new arcs (p_i, s_j) for $i \in \{1, 2, 3\}$ and $j \in \{1, 2\}$. (d) $N^+_G(u)$ is a diclique, so OUTDICLIQUE(u) is applicable. We remove u and all its incident arcs, and we add the new arcs (p_i, s_j) for $i \in \{1, 2\}$ and $j \in \{1, 2, 3\}$.

for PIE. For a given set of binary relations $\mathcal{S}$ based on a set of reductions, we say that $G \in \mathcal{G}$ is $\mathcal{S}$*-irreducible* (or simply *irreducible* when the context is clear) if for all $\mathcal{R} \in \mathcal{S}$, there does not exist $G' \in \mathcal{G}$ such that $(G, G') \in \mathcal{R}$.

It was proved in [11] that the set

$$\mathcal{S} = \{\mathcal{R}_L, \mathcal{R}_I, \mathcal{R}_O, \mathcal{R}_P\} \tag{3.1}$$

is confluent, up to isomorphism, which means that the order in which a sequence of reductions in $\mathcal{S}$ are applied does not affect the final result, up to isomorphism. More informally, given some $G \in \mathcal{G}$, consider the following procedure:

Step 1: if there is no G' such that $(G, G') \in \mathcal{R}$, then stop;
Step 2: otherwise, pick any $G' \in \mathcal{G}$ such that $(G, G') \in \mathcal{R}$;
Step 3: replace G by G' and repeat the previous steps.

The confluence property means that the resulting $\mathcal{S}$*-irreducible* graph obtained when we stop at Step 1 is unique, up to isomorphism, whatever the choice made for G' at Step 2. In what follows, and to simplify the writing,

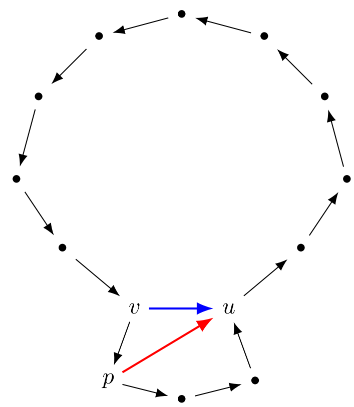


FIGURE 2. Illustration of condition (i) as introduced in the definition of DOME++ [3]. According to the definition of DOME++ and condition (i), the path from v to u goes through p a predecessor of u, and thus the arc (v, u) should be removed. As a result, after this removal, the vertex p becomes a candidate to belong to the MFVS in order to break the circuit passing through (p, u) (or the bigger one). However, this is incorrect, because the original circuit passing through v is not covered by the removal of p. The reason is that (v, u) could potentially be part of a minimal circuit and therefore should not be removed, as shown in Section 4.

we use interchangeably the confluence of a set of *binary relations* and the confluence of the set of *reductions* on which the relations are based.

Two other reductions were introduced respectively by Lin and Jou [6] and Kiesel and Shidler [3]:

- The precondition of DOME(u, v), for an arc (u, v) of G, is "$N^-_{G\rightarrow}(u) \subseteq N^-_G(v)$ or $N^+_{G\rightarrow}(v) \subseteq N^+_G(u)$". The transformation consists of replacing G with $G - (u, v)$. This reduction preserves all the FVS of the original digraph [6].
- The precondition of DOME++(u, v), for an arc (u, v) of G that is not a 2-way arc, is "for any circuit c in the digraph going through arc (u, v), then c has a 2-way arc or there exists $p_v \in N^-_G(v)$ or $s_u \in N^+_G(u)$ such that $p_v \in V(c)$ or $s_u \in V(c)$". The transformation consists of replacing G with $G - (u, v)$. This reduction preserves all the FVS of the original digraph [3].

Remark 3.1. The original definition of DOME++, introduced by Kiesel and Shidler [3] (Reduction 1, p. 5), contains, we believe, an error that has not been corrected by the authors at the time of writing. Precisely, the condition (i) from DOME++ definition's stating that an arc (v, u) is identified by DOME++ if "*every path that starts at v and ends at u uses a bi-edge (2 way arc) or a vertex from $N^-_G(u) \setminus \{v\}$*" has no impact on the circuits passing through (v, u). Indeed, a path from v to u has no impact on the circuit passing through the arc (v, u). We believe that such an arc should not be removed, as it may potentially be part of a minimal circuit, as illustrated in Figure 2. Note that in this paper, we generalize the type of arcs that can be removed without affecting the MFVS of a digraph. We do not generalize the DOME and DOME++ reductions, and therefore we will not revise the latter.

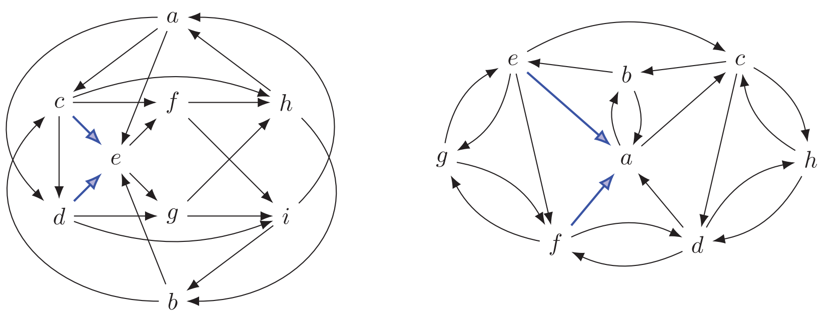
That said, we believe the error is minor, likely due to inattention or a typographical error, and could be easily corrected by considering a path "*from u to v with a bi-edge or a vertex from $N^-_G(u) \setminus \{v\}$*".

It seems difficult to maintain the confluence property with the reductions DOME and DOME++. For instance, the sets

$$\mathcal{S}_1 = \{\text{LOOP}, \text{INDICLIQUE}, \text{OUTDICLIQUE}, \text{PIE}, \text{DOME}\}$$

and

$$\mathcal{S}_2 = \{\text{LOOP}, \text{INDICLIQUE}, \text{OUTDICLIQUE}, \text{PIE}, \text{DOME++}\}$$



(a) (b)

FIGURE 3. Counter-examples of the confluence of DOME and DOME++ with themselves.

are both non-confluent, as illustrated by Example 3.2.

Example 3.2. Consider the two digraphs illustrated in Figure 3.

(a) If $G = (V, E)$ denotes the displayed digraph in Figure 3a, the equalities $N^-_{G\rightarrow}(c) = \{a, b\} \subseteq \{a, b, c, d\} = N^-_G(e)$ and $N^-_{G\rightarrow}(d) = \{a, c\} \subseteq \{a, b, c, d\} = N^-_G(e)$ hold. Hence G can be reduced using DOME(c, e) first or DOME(d, e) first. On one hand, we can apply DOME(d, e) followed by DOME(c, e). On the other hand, if we first apply DOME(c, e), then the precondition of DOME(d, e) is not verified, so that it cannot be applied. Indeed, we have $N^-_{G-(c,e)}(d) = \{a, c\} \nsubseteq \{a, b, d\} = N^-_{G-(c,e)}(e)$ and the graph $(V, E - (c, e))$ is $\mathcal{S}_1$-irreducible.

(b) Let $G = (V, E)$ be the graph depicted in Figure 3b. The arcs (e, a) and (f, a), shown in blue, are identified by DOME++. Indeed, there exists only one path $\langle a, c, b, e\rangle \in \mathcal{P}_G(a, e)$ with no 2-way arc. Since $c \in N^+_G(e)$, the arc (e, a) is identified by DOME++. Similarly, there is a unique path $p = \langle a, c, b, e, f\rangle \in \mathcal{P}_G(a, f)$ with no 2-way arc. Since $e \in N^-_G(a)$, the arc (f, a) is also identified by DOME++. However, if we apply DOME++ to remove the arc (e, a), the arc (f, a) will no longer be identified by DOME++, as the path p is also a path of $G - (e, a)$ and DOME++ preconditions do not apply. As in (a), no other reductions from the set $\mathcal{S}_2$ are applicable on $G - (e, a)$.

In Section 4, we conduct a detailed study of *superfluous* arcs, a category of arcs that can be removed from a digraph while preserving all its MFVS. In particular, arcs identified by DOME and DOME++ are superfluous arcs. We also propose a new reduction that identifies superfluous arcs and preserves confluence when it is added to the set $\mathcal{S}$ defined in (3.1).

4. SUPERFLUOUS ARCS

In this section, we take a closer look at *superfluous arcs*, a generalization we introduce to the types of arcs identified by the PIE or DOME and DOME++ reductions [3, 6].

We start by observing that to verify whether a set of vertices is indeed a feedback vertex set, it suffices to check that it has a non-empty intersection with each minimal circuit. Thus we have the following proposition.

Proposition 4.1. *Let $G = (V, E)$ be a graph and $U \subseteq V$ a subset of vertices. For every minimal circuit c of G, if $U \cap V(c) \neq \emptyset$, then U intersects every circuit of G.*

Proof. For a given circuit c, if c is a non-minimal circuit, then it contains a minimal circuit c', and since $U \cap V(c')$ is non-empty, it follows that $U \cap V(c)$ is also non-empty. ☐

Consequently, when searching for a feedback vertex set, we may disregard all arcs that do not belong to at least one minimal circuit.

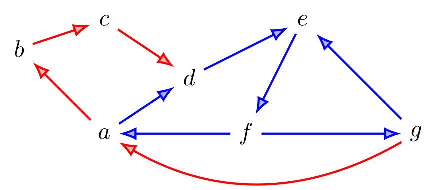


FIGURE 4. Example of minimal circuits, essential and superfluous arcs. Consider the graph $G = (V, E)$ above. The circuits $\langle a, d, e, f, a\rangle$ and $\langle f, g, e, f\rangle$ are minimal circuits. Thus, the blue arcs are essential. Moreover any circuit going through any of the red arcs contains a smaller circuit. Hence, the red arcs are superfluous.

Definition 4.2 (Essential and Superfluous Arcs)**.** An arc (u, v) in a graph $G = (V, E)$ is called *essential* if it belongs to at least one minimal circuit. Otherwise, it is *superfluous*.

Figure 4 illustrates both types of arcs, alongside with the concept of minimal circuit.

Remark 4.3. We observe that a loop, *i.e.*, an arc of the form (u, u), is always essential. Moreover, if the loops (u, u) and (v, v) are absent from the graph but the arcs (u, v) and (v, u) are present, then (u, v) and (v, u) are essential. Finally, if (u, v) is acyclic, then it is superfluous.

Thus in a graph $G = (V, E)$, if $(u, v) \in E$ is a superfluous arc, then removing (u, v) from G does not alter the set of minimal circuits in the graph. Consequently, a subset U of vertices is a FVS in G if and only if it is an FVS in $G - (u, v)$. This implies that the set of MFVS in G is identical to the set of MFVS in $G - (u, v)$. We also note that the order in which superfluous arcs are removed does not matter.

Proposition 4.4. *If a graph $G = (V, E)$ contains two distinct superfluous arcs (u, v) and (x, y), then (x, y) remains superfluous in $G - (u, v)$.*

Proof. We proceed by contradiction. Assume that (u, v) and (x, y) are superfluous in G, but that (x, y) is essential in $G - (u, v)$. Then there exists a minimal circuit c in $G - (u, v)$ that includes (x, y). Since c also exists in G, we consider two possible cases: (i) If c is also minimal in G, then there is a contradiction with the assumption that (x, y) is superfluous in G; (ii) if c is not minimal in G, then removing (u, v) must have caused (x, y) to become essential. In other words, c must contain another circuit c' passing through (u, v) but not (x, y). Since (u, v) is superfluous in G, the circuit c' must contain a smaller circuit c'' that does not include (u, v). Consequently, c'' is a circuit in $G - (u, v)$ and is contained within c, implying that c is not minimal, contradicting our assumption. Hence, our initial assumption is false, and (x, y) remains superfluous in $G - (u, v)$. □

Thus, when searching for an MFVS, our goal is to identify and remove as many superfluous arcs as possible from the graph. However, determining whether an arc is superfluous cannot be done in polynomial time unless $P = NP$. Indeed, the problem of determining whether a given vertex belongs to a minimal circuit was proven to be NP-complete by Lubiw [13], which implies that determining whether an arc belongs to a minimal circuit is also NP-hard. We are thus led to study sufficient conditions for an arc to be superfluous. We first introduce some definitions and remarks.

Definition 4.5. Let $G = (V, E)$ be a digraph, and let $p = \langle u_1, u_2, \ldots, u_\ell\rangle$ be an elementary path of length ℓ, where $\ell \geq 1$, which means that all the vertices u_i are distinct for $1 \leq i \leq \ell$. A *chord* of p is an arc $(u_i, u_j) \in E$ such that $u_i, u_j \in V(p)$ and $(u_i, u_j) \notin E(p)$, where $1 \leq i, j \leq \ell$. We distinguish three types of chords:

(i) If $(i, j) = (\ell, 1)$, then (u_i, u_j) is called a *closing chord*;
(ii) If $(i, j) \neq (\ell, 1)$ and $i \geq j$, then (u_i, u_j) is called a *backward chord* of p;
(iii) If $i < j$, then it is called a *forward chord* of p.

Moreover, if the only chord of p is a closing chord, then p is said to be *minimal*.

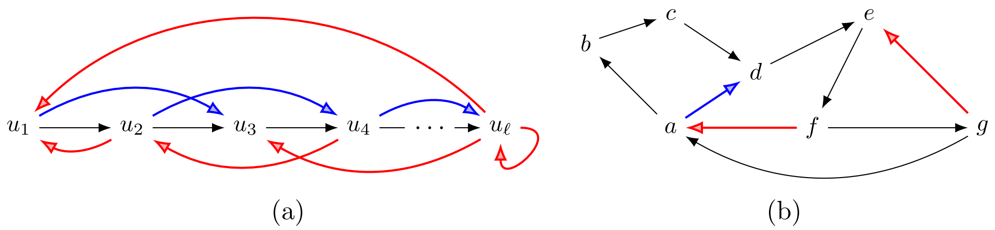


FIGURE 5. Example of elementary path chords. (a) Let $p_1 = \langle u_1, u_2, u_3, u_4, \ldots, u_\ell\rangle$ be an elementary path in a given digraph $G = (V, E)$. The blue arcs are forward chords of p_1, while the red arcs are backward chords of p_1. (b) Let $G = (V, E)$ be the digraph of Figure 4. Let $p_2 = \langle a, b, c, d, e, f, g\rangle$ be the elementary path of G, identified with bold black arcs. Then the arc (a, d) in blue is a forward chord of p_2, while the arcs (f, a) and (g, e) in red are backward chords of p_2.

Figure 5 illustrates the concept of elementary path chords. According to Definition 4.5, given an elementary circuit $c = \langle u_1, u_2, \ldots, u_\ell, u_1\rangle$, where $u_i \in V$ for $i = 1, 2, \ldots, \ell$, it is easy to see that c is minimal if and only if the elementary path $\langle u_2, \ldots, u_\ell, u_1\rangle$ is minimal.

Remark 4.6. Let $G = (V, E)$ be a digraph and p be a nonempty elementary path of G.

(1) If $(u, u) \in E$ for any $u \in V(p)$, then (u, u) is a backward chord of p.
(2) If $p = (u, v)$, $u \neq v$, then the only possible chords of p are the backward chords (u, u) and (v, v) and the closing chord (v, u). In particular, p has no forward chord.
(3) The concept of backward chord is implicitly used in the definitions of PIE given by Lin and Jou [6]. More precisely, for a loop-free digraph G and an arc $(u, v) \in G$, the acyclicity condition of (u, v) in $G^{\rightarrow}$, as introduced with PIE, is equivalent to stating that every elementary path $p = \langle x_1, x_2, \ldots, x_\ell\rangle \in \mathcal{P}_G(x_1, x_\ell)$, where $u = x_\ell$ and $v = x_1$, contains at least one backward chord of the form (x_{i+1}, x_i), where $1 \leq i \leq \ell - 1$.
(4) In the definition of DOME++ of Kiesel and Schidler [3], the condition $N^-_{G^{\rightarrow}}(u) \subseteq N^-_G(v)$ can be reformulated as follows: every elementary path $p = \langle u_1, \ldots, u_\ell\rangle \in \mathcal{P}_G(u_\ell, u_1)$ has a backward chord of the form $(u_{\ell-1}, u_1) = (x, v)$ where $x \in N^-_{G^{\rightarrow}}(u)$. Similarly, the condition $N^+_{G^{\rightarrow}}(v) \subseteq N^+_G(u)$ implies that every elementary path $p = \langle u_1, u_2, \ldots, u_\ell\rangle \in \mathcal{P}_G(u_1, u_\ell)$ has a backward chord of the form $(u_\ell, u_2) = (u, s)$ where $s \in N^+_{G^{\rightarrow}}(v)$.

We conclude this section with a proposition linking chords and superfluous arcs.

Proposition 4.7. *Let $G = (V, E)$ and $(u, v) \in E$. If every elementary path in $\mathcal{P}_G(v, u)$ has a backward chord, then the arc (u, v) is superfluous.*

Proof. Let $c = \langle u_1, u_2, \ldots, u_\ell, u_1\rangle$ be a circuit such that $u_1 = u$ and $u_2 = v$. We show that c is not minimal. Assume that c is minimal. First, notice that c is elementary, otherwise, it would contain a smaller circuit. This implies that the path $p = \langle u_2, u_3, \ldots, u_\ell, u_1\rangle$ is elementary as well. Then, by hypothesis, p has a backward chord (u_i, u_j), with $i \geq j$. Let $c' = \langle u_j, u_{j+1}, \ldots, u_i, u_j\rangle$. Clearly, c' is a circuit of G: we have $(u_k, u_{k+1}) \in E$ for $k = 1, 2, \ldots, i - 1$ and $(u_i, u_j) \in E$. Moreover, $V(c') \subsetneq V(c)$. Hence, c is not minimal. Consequently, (u, v) is superfluous. $\square$

Thus, to determine whether an arc (u, v) is superfluous, it suffices to verify that every elementary path in $\mathcal{P}_G(v, u)$ contains a backward chord.

Note that Definition 4.5 of a chord remains valid even when applied to a non-elementary path. While the resulting circuit may differ in size, the notion of a chord continues to apply.

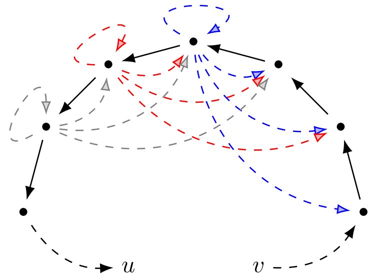


FIGURE 6. Backward chords in CHORD_4 reduction. For each vertex in a path from v to u, CHORD_4 checks if a backward chord forming a circuit of length 1, 2, 3 or 4 exists. CHORD_4 starts with v, checks only for a backward chord of length 1 and increases the length of the circuits up to 4 progressively as the path from v to u is traversed.

5. Superfluous arcs reduction

In this section we introduce a new parametrizable reduction, called CHORD_k, where $k \geq 1$ is an integer, that removes superfluous arcs. This reduction is based on a criterion that can be verified in polynomial time when k is fixed. Let $G = (V, E)$ be a digraph and an arc $(u, v) \in E$ such that $(v, u) \notin E$. Let $p = \langle u_1 = v, u_2, \ldots, u_\ell = u\rangle \in \mathcal{P}_G(v, u)$. We can identify several forms of backward chords in polynomial time, a few examples of which are as follows.

(1) A backward chord of the form (u_i, u_i) for $1 \leq i \leq \ell$. This backward chord forms a circuit of length 1 and causes the vertex u_i to be identified by LOOP.
(2) A backward chord of the form (u_{i+1}, u_i) for $1 \leq i \leq \ell-1$. This backward chord forms, with the arc (u_i, u_{i+1}), a circuit of length 2.
(3) A backward chord of the form (u_{i+1}, u_{i-1}) for $2 \leq i \leq \ell - 1$. This backward chord forms, with the arcs (u_{i-1}, u_i) and (u_i, u_{i+1}), a circuit of length 3.
(4) In general, we can search for backward chords that form a circuit of length $k \geq 1$ with the path p, which will be of the form $(u_{i+(k-1)}, u_i)$ for $1 \leq i \leq \ell - k + 1$.

We define a new reduction based on the identification of backward chords that form a circuit of length at most k.

Definition 5.1 (Chord Reduction CHORD_k)**.** Let $G = (V, E)$ and $(u, v) \in E$ such that $(v, u) \notin E$. If, for every path $p = \langle u_1 = v, u_2, \ldots, u_\ell = u\rangle \in \mathcal{P}_G(v, u)$, with $\ell \geq 2$, p has a backward chord forming a circuit of length at most k for $k \in \mathbb{N}^*$, then the arc (u, v) is superfluous. Hence, the digraph G can be reduced to $G - (u, v)$. We denote this reduction by CHORD_k and write $\text{CHORD}_k(u, v)$ when CHORD_k can be applied on (u, v).

See Figure 6 for an illustration of some of the backward chords that CHORD_4 is searching for given a path in $\mathcal{P}_G(v, u)$. Note that the reduction CHORD_k in Definition 5.1 identifies a subset of superfluous arcs. Indeed, beyond the fact that it only detects backward chords forming circuits of length at most k, it considers *all paths* between two vertices, including non-elementary paths. This provides a weaker criterion for identifying superfluous arcs compared to Definition 4.5. See Figure 7 for an illustration of this difference.

The reduction CHORD_k can be implemented with a polynomial algorithm as long as k is fixed. Indeed, we present Algorithm 1, which implements the reduction CHORD_k. It takes a digraph $G = (V, E)$ and vertices u, v such that $(u, v) \in E$, and returns *true* if (u, v) is identified by CHORD_k as a superfluous arc, which means that each path starting from v has a backward chord that forms a circuit of length at most k, or the path never reaches u. Algorithm 1 returns *false* if the algorithm finds a path from v to u with no backward chord identifiable by CHORD_k. For this algorithm, we associate each vertex u with a set of paths of length $\leq k$ through the map γ_k, representing all paths of length at most k that lead to u.

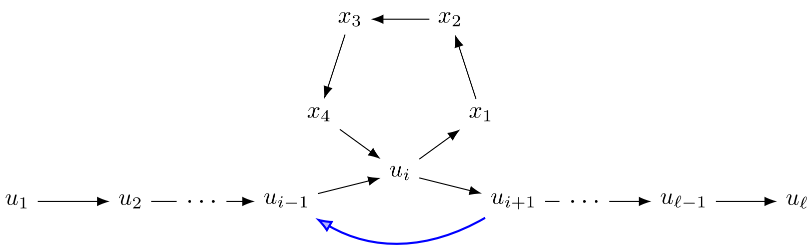


FIGURE 7. Example of a back chord that is not identified by CHORD$_3$. We have two paths in G from u_1 to u_ℓ: an elementary path $p = \langle u_1, u_2, \dots, u_{i-1}, u_i, u_{i+1}, \dots, u_{\ell-1}, u_\ell\rangle$ and a non-elementary path $p' = \langle u_1, u_2, \dots, u_{i-1}, u_i, x_1, x_2, x_3, x_4, u_i, u_{i+1}, \dots, u_{\ell-1}, u_\ell\rangle$. Because in Definition 4.5 we consider only elementary paths, then the arc (u_ℓ, u_1) is superfluous. In Definition 5.1 all possible paths are considered, therefore CHORD$_3$ identifies the backward chord (u_{i+1}, u_{i-1}) in p. However CHORD$_3$ considers that the path p' has no backward chord.

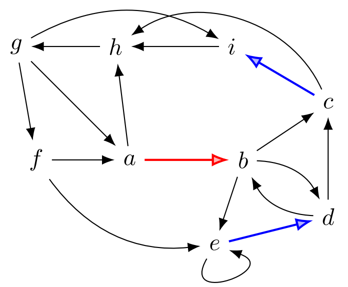


FIGURE 8. Trace of CHORD$_3$. On the digraph G above. We apply CHORD$_3$ on arcs (c, i), (e, d) and (a, b).

TABLE 1. The call K-CHORDLESSPATH$(G, c, i, \langle i\rangle, \gamma_3)$ returns *false* meaning that CHORD(G, c, i) returns *true.*

s	x_ℓ	$\gamma_3(x_\ell)$	
$\langle i\rangle$	i	$i : [\langle\rangle]$	
$\langle i, h\rangle$	h	$h : [\langle i\rangle]$	
$\langle i, h, g\rangle$	g	$g : [\langle i, h\rangle]$	(g, i)

Example 5.2 provides step by step explanations about how Algorithm 1 proceeds.

Example 5.2 (CHORD$_3$ trace)**.** Let G be the digraph depicted in Figure 8. In the following three tables, a row represents a call to the function K-CHORDLESSPATH() from Algorithm 1 with its path s in the first column. Indentation in this column depicts the recursive depth of the call. In the second column we show the current vertex x_ℓ being explored and its γ_3 set after Line 5 of in K-CHORDLESSPATH() from Algorithm 1. A row in green means that we found a backward chord, shown on the right, in the current path and we stop exploring this path. A row in red means that we stop the exploration because this path was already visited, thanks to the map γ_3 and the condition in Line 3 of Algorithm 1. The row in gray means that we reached the target vertex through a path with no backward chord that CHORD$_3$ may identify, which means that the Algorithm 1 ends and returns *false*, so that the given arc is not identified by CHORD$_3$. Table 1 shows the recursive calls to K-CHORDLESSPATH() for the arc (c, i) which is identified by CHORD$_3$ as superfluous.

Algorithm 1 CHORD$_k$ Reduction

Input:
$G = (V, E)$: digraph
u, v: vertices
Output:
true if arc (u, v) is identified by CHORD$_k$ as superfluous
false otherwise

```
function CHORD(G, u, v)
    for x ∈ V(G) do
        γ_k(x) ← {}                                        ▷ an empty set
    return ¬K-CHORDLESSPATH(G, u, v, ⟨v⟩, γ_k)
```

Input:
$G = (V, E)$: digraph
u, v: vertices
s : a path $\langle x_1, x_2, \dots, x_{\ell-1}, x_\ell \rangle$ of length $\ell - 1$, with $\ell \leq k$
γ_k : the mapping associating to a vertex x the set of suffixes through which x has been visited
Output:
true if there exists a path from v to u without any backward chord detectable by CHORD$_k$
false otherwise

```
function K-CHORDLESSPATH(G, u, v, s, γ_k)
    s' ← ⟨x_1, x_2, ..., x_{ℓ-1}⟩                  ▷ remove the last vertex x_ℓ from s
    if s' ∈ γ_k(x_ℓ) then                          ▷ reaching x_ℓ through s' was explored before
        return false
    γ_k(x_ℓ).INSERT(s')
    for y ∈ V(s) do
        if (x_ℓ, y) ≠ (u, v) and (x_ℓ, y) ∈ E then  ▷ a backward chord is found
            return false
    if x_ℓ = u then                                ▷ we found a path with no backward chord
        return true
    for y ∈ N_G^+(x_ℓ) do
        if ℓ = k then
            s'' ← ⟨x_2, ..., x_{ℓ-1}, x_ℓ, y⟩          ▷ remove x_1 and add y to the path s
        else                                       ▷ in this case ℓ < k
            s'' ← ⟨x_1, x_2, ..., x_{ℓ-1}, x_ℓ, y⟩
        if K-CHORDLESSPATH(G, u, v, s'', γ_k) then
            return true
    return false
```

Table 3 shows the recursive calls to K-CHORDLESSPATH() for the arc (a, b) which is not identified by CHORD$_3$ because of the path $\langle b, c, h, g, f, a \rangle$.

In order to prove the correctness of Algorithm 1, we introduce the notion of k-*suffix*. Let $p = \langle p_1, \dots, p_\ell \rangle$ be a path in a digraph $G = (V, E)$. For $k \in \mathbb{N}^*$, the k-*suffix* of p is the subpath that consists of the last k

TABLE 2. The call K-CHORDLESSPATH$(G, e, d, \langle d\rangle, \gamma_3)$ returns *false* which means that CHORD(G, e, d) returns *true*.

s	x_ℓ	$\gamma_3(x_\ell)$	
$\langle d\rangle$	d	$d : [\langle\rangle]$	
$\langle d, b\rangle$	b	$b : [\langle d\rangle]$	(d, b)
$\langle d, c\rangle$	c	$c : [\langle d\rangle]$	
$\langle d, c, i\rangle$	i	$i : [\langle d, c\rangle]$	
$\langle c, i, h\rangle$	h	$h : [\langle c, i\rangle]$	
$\langle i, h, g\rangle$	g	$g : [\langle i, h\rangle]$	(g, i)
$\langle d, c, h\rangle$	h	$h : [\langle c, i\rangle\langle d, c\rangle]$	
$\langle c, h, g\rangle$	g	$g : [\langle i, h\rangle\langle c, h\rangle]$	
$\langle h, g, a\rangle$	a	$a : [\langle h, g\rangle]$	(a, h)
$\langle h, g, f\rangle$	f	$f : [\langle h, g\rangle]$	
$\langle g, f, a\rangle$	a	$a : [\langle h, g\rangle\langle g, f\rangle]$	
$\langle f, a, b\rangle$	b	$b : [\langle d\rangle\langle f, a\rangle]$	
$\langle a, b, c\rangle$	c	$c : [\langle d\rangle\langle a, b\rangle]$	
$\langle b, c, i\rangle$	i	$i : [\langle d, c\rangle\langle b, c\rangle]$	
$\langle c, i, h\rangle$	h	$h : [\langle c, i\rangle\langle d, c\rangle]$	$\langle c, i\rangle$
$\langle b, c, h\rangle$	h	$h : [\langle c, i\rangle\langle d, c\rangle\langle b, c\rangle]$	
$\langle c, h, g\rangle$	g	$g : [\langle i, h\rangle\langle c, h\rangle]$	$\langle c, h\rangle$
$\langle a, b, d\rangle$	d	$d : [\langle\rangle\langle a, b\rangle]$	(d, b)
$\langle a, b, e\rangle$	e	$e : [\langle a, b\rangle]$	(e, e)
$\langle f, a, h\rangle$	h	$h : [\langle c, i\rangle\langle d, c\rangle\langle b, c\rangle\langle f, a\rangle]$	
$\langle a, h, g\rangle$	g	$g : [\langle i, h\rangle\langle c, h\rangle\langle a, h\rangle]$	(g, a)
$\langle g, f, e\rangle$	e	$e : [\langle a, b\rangle\langle g, f\rangle]$	(e, e)
$\langle h, g, i\rangle$	i	$i : [\langle d, c\rangle\langle b, c\rangle\langle h, g\rangle]$	(i, h)

TABLE 3. The call K-CHORDLESSPATH$(G, a, b, \langle b\rangle, \gamma_3)$ which returns *true*, meaning that (a, b) is not identified by CHORD$_3$ as superfluous.

s	x_ℓ	$\gamma_3(p_\ell)$	
$\langle b\rangle$	b	$b : [\langle\rangle]$	
$\langle b, c\rangle$	c	$c : [\langle b\rangle]$	
$\langle b, c, i\rangle$	i	$i : [\langle b, c\rangle]$	
$\langle c, i, h\rangle$	h	$h : [\langle c, i\rangle]$	
$\langle i, h, g\rangle$	g	$g : [\langle i, h\rangle]$	(g, i)
$\langle b, c, h\rangle$	h	$h : [\langle c, i\rangle\langle b, c\rangle]$	
$\langle c, h, g\rangle$	g	$g : [\langle i, h\rangle\langle c, h\rangle]$	
$\langle h, g, a\rangle$	a	$a : [\langle h, g\rangle]$	(a, h)
$\langle h, g, f\rangle$	f	$f : [\langle h, g\rangle]$	
$\langle g, f, a\rangle$	a	$a : [\langle h, g\rangle\langle g, f\rangle]$	

vertices of p if $k \leq \ell$, or the entire path p if $k > \ell$. Thus, the k-suffix of p is the subpath $\langle p_i, p_{i+1}, \ldots, p_\ell\rangle$, where $i = \max(1, \ell - k + 1)$.

Lemma 5.3. *Let $G = (V, E)$ be a digraph, $(u, v) \in E$ an arc and $k \geq 1$ be an integer. Then Algorithm 1 always terminates and explores the k-suffixes of paths starting from v until it reaches u without finding a backward chord identifiable by* CHORD$_k$*, or until all k-suffixes of these paths have been explored.*

Proof. Let γ_k and s be as defined in Algorithm 1.

First, because we keep the length of s less or equal to $k-1$, then s is always the k-suffix of some path. In Algorithm 1, for a given vertex x_ℓ, if the condition in Line 3 is *false* and s does not have a backward chord and does not reach u, then we explore all vertices in $N^+_G(x_\ell)$ and, with each one of them, we extend the k-suffix s to create new paths. Otherwise, the current vertex x_ℓ was already processed following exactly the same k-suffix s'. In all cases, the vertices in $N^+_G(x_\ell)$ are visited at least once. Therefore, Algorithm 1 explores all paths starting from v and stops only if it finds a path to u with a k-suffix with no backward chord.

Because we keep the length of paths less than k with the condition of Line 12, all paths considered in Algorithm 1 are k-suffixes of paths starting from v to vertices in $N^+_G(x_\ell)$. Therefore, for each vertex x_ℓ reachable from v, we keep all different k-suffixes in the mapping $\gamma_k(x_\ell)$, thanks to Line 5. Thus, using the condition of Line 3, we do not explore and extend a k-suffix if it was already processed. Next, if we consider $\Delta = \max\{|N^-_G(y)| \mid y \in V\}$, then we are certain that the number of k-suffixes leading to a given vertex x_ℓ in G is bounded by $\underbrace{\Delta \cdot \ldots \cdot \Delta}_{k \text{ times}} = \Delta^k$. In other words, $|\gamma_k(x_\ell)| \leq \Delta^k$. Hence, each vertex of G is considered at most Δ^k times to extend the k-suffixes in the **for** loop of Line 11, which means that the K-CHORDLESSPATH() function is called at most $|V| \cdot \Delta^k$ times. This implies that Algorithm 1 always terminates. □

We now prove the correctness of Algorithm 1.

Lemma 5.4. *Let $G = (V, E)$ be a digraph and $(u, v) \in E$ an arc. Then* CHORD$_k$ *is applicable on (u, v) if and only if Algorithm 1 returns true.*

Proof. Let γ_k and s be as defined in Algorithm 1.

If (u, v) is identified by CHORD$_k$, then all paths from v to u have a backward chord identified by CHORD$_k$. From Lemma 5.3 we know that Algorithm 1 explores k-suffixes of all possible paths starting from v unless it reaches u without finding a backward chord and, thanks to the **for** loop of Line 6 and the **if** condition of Line 7, Algorithm 1 checks all possible backward chords of the form (x_ℓ, y) with y in the current k-suffix s. Since s is a k-suffix, it has at most k vertices, meaning that the largest circuit that a backward chord could create has length k. The smallest circuit is a loop, in the case where $y = x_\ell$. Therefore, the **for** loop at Line 6 searches for a backward chord identifiable by CHORD$_k$ in k-suffixes of all possible paths starting from v, including the paths leading to u if they exist. This means that the call to Algorithm 1 returns *true*.

Conversely, assume that a call to Algorithm 1 returns *true*. From Lemma 5.3, we know that all k-suffixes of all possible paths starting from v leading to u are explored. If a backward chord is found by the **for** loop of Line 6, then the exploration stops at the current vertex x_ℓ. If no chord is found, then u is not reached, otherwise Algorithm 1 returns *false*. Thanks again to Lemma 5.3 we know that the recursive calls to K-CHORDLESSPATH() are bounded and therefore Algorithm 1 returns *true* indicating that (u, v) is identified by CHORD$_k$. □

Thanks to the map γ_k we are sure that Algorithm 1 does not visit a given vertex more then Δ^k times, as stated in the proof of Lemma 5.3. Therefore, the complexity of Algorithm 1 is $\mathcal{O}(|V| \cdot \Delta^{2k})$, which remains polynomial as long as k is fixed. Finally, it is worth noting that Algorithm 1 can be interpreted as a search in a graph formed by the k-suffixes associated with each vertex u in G.

6. CHORD$_k$ AND CONFLUENCE

We first notice that the reduction CHORD$_k$ subsumes the PIE reduction for $k \geq 2$, meaning that, if PIE is applicable on an arc (u, v), then CHORD$_k$ is also applicable on (u, v). The same goes for the reduction LOOP with a slight difference. Indeed, the reduction CHORD$_k$ removes all arcs incident to a vertex with a loop (for $k \geq 1$). It therefore isolates it in the graph while preserving its loop. Thus, in a certain sense, a vertex with a loop will be excluded from subsequent computations.

Before analyzing the confluence of reduction sets that include CHORD$_k$, we first establish some properties and lemmas that will facilitate the arguments for proving confluence later. First, a property of interest for CHORD$_k$ is formalized in the following proposition.

Proposition 6.1. *Given a digraph $G = (V, E)$ and two arcs $(u, v), (x, y) \in E$ identified by* CHORD$_k$*, then* CHORD$_k$ *identifies (x, y) in $G - (u, v)$.*

Proof. Note that this proof is similar to the proof of Proposition 4.4.

The removal of the arc (u, v) with the reduction CHORD$_k$ may reduce the number of paths from y to x, so the applicability of CHORD$_k$ on the arc (x, y) in $G - (u, v)$ should be maintained. The only problematic case arises when (u, v) is a backward chord in a given path $p = \langle x_1 = y, x_2, \ldots, x_\ell = x\rangle$ from y to x. Then there exists $1 \leq j \leq k$ such that $p = \langle x_1 = y, x_2, \ldots, x_i = v, \ldots, x_{i+(j-1)} = u, \ldots, x_\ell = x\rangle$ with an arc $(x_{i+(j-1)}, x_i) = (u, v)$. Thus, if CHORD$_k$ removes the arc (u, v) identified as superfluous, is it possible that the path p loses its backward chord? When (u, v) is a backward chord in p then there is a subpath $p' = \langle x_i = v, \ldots, x_{i+(j-1)} = u\rangle$ from v to u and since (u, v) is identified by CHORD$_k$ and every path, including p', has a backward chord, this backward chord of p' is also a backward chord of p. Therefore, CHORD$_k$ remains applicable on (x, y) in $G - (u, v)$. □

Because a loop is a backward chord, it is now easy to see that the set {LOOP, PIE, CHORD$_k$} is confluent. The following lemma characterizes the interaction of the reductions INDICLIQUE and OUTDICLIQUE with CHORD$_k$.

Lemma 6.2. *Let $G = (V, E)$, $(u, v) \in E$ and $x \in V$ such that $(x, x) \notin E$. Assume that* CHORD$_k(u, v)$ *is applicable.*

(a) If $x \neq u$ and $x \neq v$, then CHORD$_k$ *is applicable on (u, v) in $G \circ x$.*
(b) If $x = u$ and for $p_x \in N^-_G(x)$, $(p_x, v) \notin E$, then CHORD$_k$ *is applicable on (p_x, v) in $G \circ x$.*
(c) If $x = v$ and for $s_x \in N^+_G(x)$, $(u, s_x) \notin E$, then CHORD$_k$ *is applicable on (u, s_x) in $G \circ x$.*

Proof. **(a)** Assume that $x \neq u$ and $x \neq v$.

Let $p = \langle u_1 = v, u_2, \ldots, u_\ell = u\rangle$ be a path from v to u in $G \circ x$. We have the following cases.

(i) $E(p) \cap N^-_G(x) \times N^+_G(x) = \emptyset$, in which case the path p exists in G, and since CHORD$_k$ in applicable on (u, v) in G, p has a backward chord identifiable by CHORD$_k$ in G, which will also be identifiable by CHORD$_k$ in $G \circ x$ because $x \notin p$ so the backward chord in G is not incident to x.

(ii) $E(p) \cap N^-_G(x) \times N^+_G(x) \neq \emptyset$, which means that there exists $p_x \in N^-_G(x)$ and $s_x \in N^+_G(x)$ such that $(p_x, s_x) \in p$. Thus, the path p can be written as $p = \langle u_1 = v, \ldots, u_i = p_x, u_{i+1} = s_x, \ldots, u_\ell = u\rangle$. The path p being in $G \circ x$, there is a path p' in G obtained from p by adding the vertex x and substituting the arc (p_x, s_x) with the two arcs (p_x, x) and (x, s_x). Therefore, we have the following path in G,

$$p' = \langle u_1 = v, \ldots, u_i = p_x, x, u_{i+1} = s_x, \ldots, u_\ell = u\rangle$$

To simplify the following arguments, we label the vertices of p' as follows.

$$\begin{array}{lllllll} p' = \langle u_1 = v, & \ldots, & u_i = p_x, & x, & u_{i+1} = s_x, & \ldots, & u_\ell = u\rangle \\ p' = \langle x_1 = v, & \ldots, & x_i = p_x, & x_{i+1} = x, & x_{i+2} = s_x, & \ldots, & x_{\ell+1} = u\rangle \end{array}$$

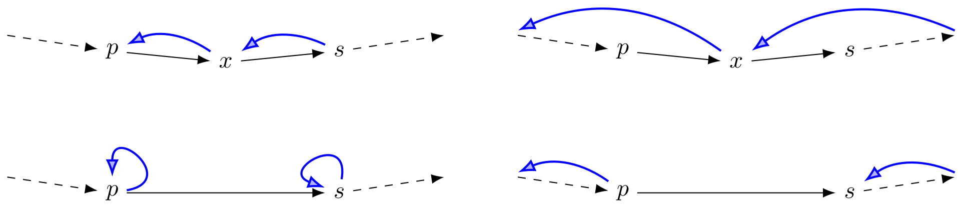
Since p' is a path in G from v to u and (u, v) is identified by CHORD$_k$ in G, p' has a backward chord, and we have several possibilities.

– The backward chord is a loop, and since $(x, x) \notin E$, this loop exists in $G \circ x$. Hence, one of the vertices in p' other than x has a loop, and consequently, this same vertex in p will keep its loop in $G \circ x$.

– The backward chord is a circuit of length 2, and in this case, we have two possibilities, either x is part of the length 2 circuit, that is, one of the two length 2, (p_x, x, p_x) or (s_x, x, s_x), so either p_x or s_x will have a loop in $G \circ x$, and thus p will have a loop; or it could be a length 2 circuit that does not pass through x, in which case, it will exist in $G \circ x$, and p will have a 2-way arc. Figure 9a illustrates this case.

– The backward chord is an arc (x_{l+k-1}, x_l) with $k \geq 3$, thus forming a circuit of length k. We have three possible cases.

(1) $(x_{l+k-1}, x_l) = (x_{i+k}, x)$ (for $l = i + 1$ and $x_{i+1} = x$). Since $k \geq 3$ and the vertex $x_{i+k} \in N^-_G(x)$, this arc will be transformed into (x_{i+k}, s_x), which is a backward chord of p that exists in $G \circ x$ and forms a circuit of length $k - 1$.



(a) (b)

FIGURE 9. Backward chords illustration in the path p' and p for the case $x \neq u$ and $x \neq v$. For the segment of the path p' (at the top) from v to u that goes through x, and the corresponding segment in the path p (at the bottom). (a) There are two cases of backward chords incident to x that form a circuit of length 2 in p'. In both cases, we observe that a backward chord of length 1 is created in p in $G \circ x$. (b) For the two cases of backward chords incident to x forming a circuit of length k in p', in both cases, a backward chord of length $k-1$ is obtained in p in $G \circ x$.

(2) $(x_{l+k-1}, x_l) = (x, x_{i-k+2})$ (for $l+k-1 = i+1$ and $x_{i+1} = x$). Since $k \geq 3$ and $x_{i-k+2} \in N_G^+(x)$, this arc will be transformed into (p_x, x_{i-k+2}), which is a backward chord of p that exists in $G \circ x$ and forms a circuit of length $k-1$.
(3) x is not incident to this backward chord, in which case, this backward chord will exist in $G \circ x$ and will be part of p, thus forming a circuit of length $k-1$ or k.

In all cases, the arc (x_{l+k-1}, x_l) forming a backward chord of p' in G will form a backward chord of p in $G \circ x$. Figure 9b illustrates Cases (1) and (2).

Thus, the path p will have a backward chord identifiable by CHORD$_k$ in $G \circ x$, so (u, v) will be identified by CHORD$_k$ in $G \circ x$.

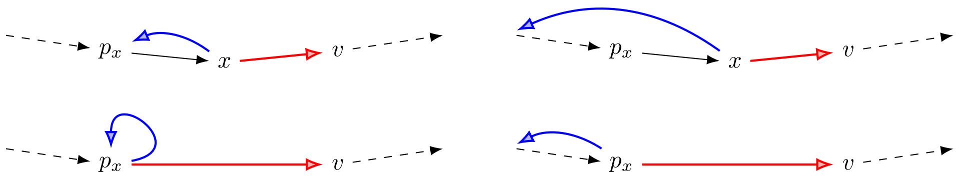
(b) $x = u$ and $p_x \in N_G^-(x)$ such that $(p_x, v) \notin E$.

Let $p = \langle u_1 = v, u_2, \ldots, u_\ell = p_x \rangle$ be a path from v to p_x in $G \circ x$. Since the path p is in $G \circ x$, we can construct a path p' from v to x in G by adding the vertex x and substituting the arc (p_x, v), which does not exist in G, with the arc (p_x, x). Thus, we have:

$$p' = \langle u_1 = v, \ldots, u_\ell = p_x, u_{\ell+1} = x \rangle$$

Since p' is a path from v to x and (x, v) is identified by CHORD$_k$ in G, p' has a backward chord, and we have several possibilities.

(i) The backward chord could be a loop, and since $(x, x) \notin E$, this loop exists in $G \circ x$. Hence, one of the vertices in p' other than x has a loop, and consequently, this same vertex in p will keep its loop in $G \circ x$.
(ii) The backward chord could be a 2-cycle, and in this case, we have two possibilities: either x is part of the 2-cycle, that is, the 2-cycle (p_x, x, p_x), so p_x will have a loop in $G \circ x$, and thus p will have a loop; or it could be a 2-cycle that does not pass through x, in which case, it will exist in $G \circ x$, and p will have a 2-cycle. See Figure 10a for an illustration of this case.
(iii) It could be an arc (u_{l+k-1}, u_l) with $k \geq 3$, thus forming a cycle of length k. We have two possible cases:
   (1) $(u_{l+k-1}, u_l) = (x, u_{\ell-k+2})$ (for $l+k-1 = \ell+1$ and $u_{\ell+1} = x$). Since $k \geq 3$ and $u_{\ell-k+2} \in N_G^+(x)$, this arc will be transformed into $(p_x, u_{\ell-k+2})$, which is a backward chord of p that exists in $G \circ x$ and forms a cycle of length $k-1$. See Figure 10b for an illustration of this case.
   (2) x is not incident to this backward chord, in which case, this backward chord will exist in $G \circ x$ and will be part of p, thus forming a cycle of length $k-1$.



(a) (b)

FIGURE 10. Backward chords illustration in the path p' and p for the case $x = u$. Consider the segment of the path p' (at the top) from v to x that goes through p_x, and the corresponding segment in the path p (at the bottom). (a) A possible backward chord incident to x that form a circuit of length 2 in p'. In this case, a backward chord of length 1 is created in p in $G \circ x$. (b) Similarly, a possible backward chord incident to x forming a circuit of length k in p'. In this case, a backward chord of length $k-1$ is obtained in p in $G \circ x$.

In all cases, the arc (u_{l+k-1}, u_l) forming a backward chord of p' in G will form a backward chord of p in $G \circ x$.

Thus, the path p has a backward chord identifiable by CHORD$_k$ in $G \circ x$, so (p_x, v) is identified by CHORD$_k$ in $G \circ x$.

(c) $x = v$ and $s_x \in N_G^+(x)$ such that $(u, s_x) \notin E$.

Let $p = \langle u_1 = s_x, u_2, \ldots, u_\ell = u\rangle$ be a path from s_x to u in $G \circ x$. Since the path p is in $G \circ x$, we can construct a path p' from x to u in G by adding the vertex x and substituting the arc (u, s_x), which does not exist in G, with the arc (x, s_x). Thus, we have:

$$p' = \langle u_0 = x, u_1 = s_x, u_2, \ldots, u_\ell = u\rangle$$

Since p' is a path from x to u and (u, x) is identified by CHORD$_k$ in G, p' has a backward chord, and we have several possibilities.

(i) The backward chord could be a loop, and since $(x, x) \notin E$, this loop exists in $G \circ x$. Hence, one of the vertices in p' other than x has a loop, and consequently, this same vertex in p will keep its loop in $G \circ x$.

(ii) The backward chord could be a 2-cycle, and in this case, we have two possibilities: either x is part of the 2-cycle, that is, the 2-cycle (s_x, x, s_x), so s_x will have a loop in $G \circ x$, and thus p will have a loop; or it could be a 2-cycle that does not pass through x, in which case, it will exist in $G \circ x$, and p will have a 2-cycle. See Figure 11a for an illustration of this case.

(iii) It could be an arc (u_{l+k-1}, u_l) with $k \geq 3$, thus forming a cycle of length k. We have two possible cases:
   (1) $(u_{l+k-1}, u_l) = (u_{k-1}, x)$ (for $l = 0$ and $u_0 = x$). Since $k \geq 3$ and $u_{k-1} \in N_G^-(x)$, this arc will be transformed into (u_{k-1}, s_x), which is a backward chord of p that exists in $G \circ x$ and forms a cycle of length $k-1$. See Figure 11b for an illustration of this case.
   (2) x is not incident to this backward chord, in which case, this backward chord will exist in $G \circ x$ and will be part of p, thus forming a cycle of length $k-1$.

   In all cases, the arc (u_{l+k-1}, u_l) forming a backward chord of p' in G forms a backward chord of p in $G \circ x$.

Thus, the path p has a backward chord identifiable by CHORD$_k$ in $G \circ x$, so (u, s_x) is identified by CHORD$_k$ in $G \circ x$. □

We can now expand the set of confluent reductions $\mathcal{S}$ introduced in Section 3 with the reduction CHORD$_k$. First, for a given $k \in \mathbb{N}^*$, let the relation

$$\mathcal{R}_C = \{(G, G') \mid G \text{ is reduced to } G' \text{ by the reduction CHORD}_k\}$$

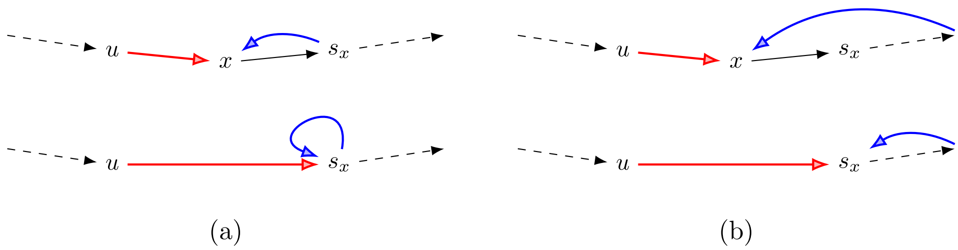


FIGURE 11. Backward chords illustration in the path p' and p for the case $x = v$. Lets consider the segment of the path p' (at the top) from x to u that goes through s_x, and the corresponding segment in the path p (at the bottom). (a) A possible case of backward chord incident to x that form a circuit of length 2 in p'. In this cases, we observe that a backward chord of length 1 is created in p in $G \circ x$. (b) Similarly, a possible case of backward chord incident to x forming a circuit of length k in p', we see that in this case, a backward chord of length $k-1$ is obtained in p in $G \circ x$.

We have the following binary relation $\mathcal{S}'$ on the set $\mathcal{G}$ of all finite digraphs,

$$\mathcal{S}' = \mathcal{R}_I \cup \mathcal{R}_O \cup \mathcal{R}_C$$

Since CHORD$_k$ subsumes LOOP and PIE for $k \geq 2$, then the relation $\mathcal{S}$ is included in $\mathcal{S}'$, and we have the following theorem.

Theorem 6.3. $(\mathcal{G}, \mathcal{S}')$ *is confluent.*

Proof. According to Theorem 2.1 it is enough to prove that $(\mathcal{G}, \mathcal{S}')$ is finite and that the following property stands for any digraphs and relation in $\mathcal{S}'$, for $G, G_1, G_2 \in \mathcal{G}$ if $(G, G_1) \in \mathcal{S}'$ and $(G, G_2) \in \mathcal{S}'$, then there exist $G' \in \mathcal{G}$ such that $(G_1, G') \in \mathcal{S}'^T$ and $(G_2, G') \in \mathcal{S}'^T$. Since $\mathcal{S}$ is known to be confluent [11], it is sufficient to prove the above property only for $\mathcal{R}_C$ and with all the relations in $\mathcal{S}'$.

Let $G = (V, E) \in \mathcal{G}$ be a digraph, $x \in V$ and $(u, v) \in E$. CHORD$_k$ being a reduction, its successive applications are bounded by $|E|$ or, at most, $|V|^2$. Hence $(\mathcal{G}, \mathcal{S}')$ is finite.

Now, assume that CHORD$_k(u, v)$ is applicable and we have $G_1 = G - (u, v)$.

Assume that INDICLIQUE(x) is applicable and we have $G_2 = G \circ x$. We have three cases to consider.

(1) $x = u$. Then the arc (x, v) does not exist in G_2, although, CHORD$_k$ is applicable on arcs (p_x, v) for $p_x \in N_G^-(x)$ if $(p_x, v) \notin E$, thanks to Lemma 6.2-(b). We obtain the reduced digraph,

$$G_2 - \bigcup_{p_x \in N_G^-(x) - N_G^-(v)} \{(p_x, v)\}$$

which is equal to $(G - (x, v)) \circ x$, because CHORD$_k$ removes all arcs substituting (x, v) in G_2, which is equivalent to the removal of (x, v) from the beginning.

On the other hand, INDICLIQUE(x) is still applicable on G_1 because CHORD$_k$ does not identify an arc from a diclique and we obtain the digraph $G_1 \circ x = (G - (x, v)) \circ x$. Therefore, it is sufficient to consider $G' = ((G - (x, v)) \circ x$.

(2) $x = v$. Then the arc (u, v) does not exist in G_2, although, CHORD$_k$ is applicable on arcs (u, s_x) for $s_x \in N_G^+(x)$ if $(u, s_x) \notin E$, thanks to Lemma 6.2-(c). We obtain the reduced digraph $G_2 - \bigcup_{s_x \in N_G^+(x) - N_G^+(u)} \{(u, s_x)\}$ which is equal to $(G - (u, x)) \circ x$.

As in the precedent case, on the other hand, INDICLIQUE(x) is still applicable on G_1 because CHORD$_k$ does not identify an arc from a diclique and we obtain the digraph $G_1 \circ x = (G - (u, x)) \circ x$. Therefore, it is sufficient to choose $G' = ((G - (u, x)) \circ x$.

(3) (u, v) is not incident to x. Then thanks to Lemma 6.2-(a) CHORD$_k(u, v)$ remains applicable in G_2 and we obtain $G_2 - (u, v)$.

On the other hand, as stated before, CHORD$_k$ does not identify an arc from a diclique, so INDICLIQUE(x) is applicable on G_1. Therefore, it is sufficient to consider $G' = (G - (u, v)) \circ x = (G \circ x) - (u, v)$.

Hence, in all cases, there exists $G' \in \mathcal{G}$ such that $(G_1, G') \in \mathcal{S}'^T$ and $(G_2, G') \in \mathcal{S}'^T$.

The case OUTDICLIQUE(x) is symmetric to the case INDICLIQUE(x), and we omit the details of the proof for this case, as the arguments supporting the proof are almost identical to the previous case. The reduced digraphs are identical to the previous case and we can conclude that there exists $G' \in \mathcal{G}$ such that $(G_1, G') \in \mathcal{S}'^T$ and $(G_2, G') \in \mathcal{S}'^T$.

Finally, from Proposition 6.1 we can consider $G' = (G - (u, v)) - (x, y) = (G - (x, y)) - (u, v)$, if CHORD$_k(x, y)$ is applicable for a given $(x, y) \in E$ and we have $(G_1, G') \in \mathcal{S}'^T$ and $(G_2, G') \in \mathcal{S}'^T$.

Therefore, we conclude that $(\mathcal{G}, \mathcal{S}')$ is confluent. □

The reductions CHORD$_k$ and DOME++, both being based on backward chords, naturally lead to the question of when they identify the same arcs. The following digraph properties, if satisfied, imply that CHORD$_k$ subsumes DOME++, meaning that every arc identified by DOME++ will also be identified by CHORD$_k$. Given a digraph $G = (V, E)$ and an integer $k \in \mathbb{N}^*$, we have.

(1) The longest path in G has length at most k.
(2) The longest cycle in G has length at most k.
(3) For a given vertex $u \in V$, the following paths have length less than k.
  - A path from u to p, for every $p \in N_G^-(u)$.
  - A path from s to u, for every $s \in N_G^+(u)$.

These conditions are sufficient but not necessary for CHORD$_k$ to cover DOME++. Consequently, they represent strong constraints for such coverage.

7. Conclusion

In this article, we explored arc reductions for preprocessing the minimum feedback vertex set problem. We introduced the notion of superfluous arcs and the parametrized reduction CHORD$_k$, which eliminates a subset of such arcs in polynomial time for bounded k. We showed that CHORD$_k$ is confluent within an established reduction set, ensuring order independence in the final reduced digraph. This expands the largest known set of confluent reductions for MFVS, strengthening the theoretical foundations of digraph transformations. In future work, we will focus on a global implementation of the CHORD$_k$ reduction using a dynamic programming algorithm. Leveraging confluence, we will also explore the possibility of implementing parallel algorithms that apply one or more reductions from a confluent set.

Funding

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant in the Computer Science Group at Université du Québec à Montréal, and a Discovery Grant in the Mathematics and Statistics Group at Université du Québec à Trois-Rivières.

Conflict of interest

The authors have nothing to disclose.

Data availability statement

This article has no associated data generated and/or analyzed.

Author contribution statement

- Conceptualization: Moussa Abdenbi, Alexandre Blondin Massé, Alain Goupil and Odile Marcotte;
- Methodology: Moussa Abdenbi;
- Software: N/A;
- Validation: Moussa Abdenbi, Alexandre Blondin Massé and Alain Goupil;
- Formal Analysis: Moussa Abdenbi and Odile Marcotte;
- Investigation: Moussa Abdenbi;
- Resources: N/A;
- Data Curation: N/A;
- Writing – Original Draft Preparation: Moussa Abdenbi.;
- Writing – Review & Editing: Moussa Abdenbi, Alexandre Blondin Massé, Alain Goupil and Odile Marcotte;
- Visualization: Moussa Abdenbi;
- Supervision: Alexandre Blondin Massé and Alain Goupil;
- Project Administration: N/A;
- Funding Acquisition: Alexandre Blondin Massé and Alain Goupil.

References

[1] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, 1st edn. W. H. Freeman (1979).

[2] R. Karp, Reducibility Among Combinatorial Problems, **40** (1972) 85–103.

[3] R. Kiesel and A. Schidler, A dynamic maxsat-based approach to directed feedback vertex sets (2022) https://arxiv.org/abs/2211.06109v1.

[4] M. Lemaic, *Markov-Chain-Based Heuristics for the Feedback Vertex Set Problem for Digraphs*. PhD thesis, Universität zu Köln (2008).

[5] H. Levy and D.W. Low, A contraction algorithm for finding small cycle cutsets. *J. Algorithms* **9** (1988) 470–493.

[6] H.-M. Lin and J.-Y. Jou, On computing the minimum feedback vertex set of a directed graph by contraction operations. *IEEE Trans, Comput. Aided Des. Integr. Circuits Syst.* **19** (2000) 295–307.

[7] A. Church and J.B. Rosser, Some properties of conversion. *Trans. Am. Math. Soc.* **39** (1936) 472–482.

[8] F. Baader and T. Nipkow, Term Rewriting and All That. Cambridge University Press (1998).

[9] H. Ehrig and B.K. Rosen, Parallelism and concurrency of graph manipulations. *Theoret. Comput. Sci.* **11** (1980) 247–275.

[10] B.K. Rosen, Correctness of parallel programs: the Church-Rosser approach. *Theoret. Comput. Sci.* **2** (1976) 183–207.

[11] M. Abdenbi, A. Blondin Massé, A. Goupil and O. Marcotte, On the confluence of directed graph reductions preserving feedback vertex set minimality, in Proceedings of the 13th Edition of the Conference on Random Generation of Combinatorial Structures. Polyominoes and Tilings, edited by S. Brlek and L. Ferrari Bordeaux, France, 24–28th June 2024, vol. 403 of *Electronic Proceedings in Theoretical Computer Science*. Open Publishing Association (2024) 6–11.

[12] A.V. Aho, R. Sethi and J.D. Ullman, Code optimization and finite Church–Rosser systems, in *Design and Optimization of Compilers*. Prentice-Hall, Englewood Cliffs, NJ (1972).

[13] A. Lubiw, A note on odd/even cycles. *Discrete Appl. Math.* **22** (1988) 87–92.