Tight Localizations of Feedback Sets

MICHAEL HECHT and KRZYSZTOF GONCIARZ, Max Planck Institute of Molecular Cell Biology and Genetics, Center for Systems Biology Dresden
SZABOLCS HORVÁT, Max Planck Institute of Molecular Cell Biology and Genetics, Center for Systems Biology Dresden and Max Planck Institute for the Physics of Complex Systems

The classical NP–hard *feedback arc set problem* (FASP) and *feedback vertex set problem* (FVSP) ask for a minimum set of arcs $\varepsilon \subseteq E$ or vertices $\nu \subseteq V$ whose removal $G \setminus \varepsilon$, $G \setminus \nu$ makes a given multi–digraph $G = (V, E)$ acyclic, respectively. Though both problems are known to be APX–hard, constant ratio approximations or proofs of inapproximability are unknown. We propose a new universal $O(|V||E|^4)$–heuristic for the directed FASP. While a ratio of $r \approx 1.3606$ is known to be a lower bound for the APX–hardness, at least by empirical validation we achieve an approximation of $r \leq 2$. Most of the relevant applications, such as *circuit testing*, ask for solving the FASP on large sparse graphs, which can be done efficiently within tight error bounds with our approach.

CCS Concepts: • **Theory of computation → Network flows**;

Additional Key Words and Phrases: Minimum feedback arc set problem, minimum feedback vertex set problem, maximum linear ordering problem, approximation algorithm

ACM Reference format:
Michael Hecht, Krzysztof Gonciarz, and Szabolcs Horvát. 2021. Tight Localizations of Feedback Sets. *J. Exp. Algorithmics* 26, 1, Article 1.5 (March 2021), 19 pages.
https://doi.org/10.1145/3447652

1 INTRODUCTION

Belonging to R. M. Karp's famous list of 21 NP–complete problems [37], the FVSP & FASP are of central interest in theoretical computer science and beyond.

A very prominent application occurs in *electronic engineering* for *designing processors or computer chips*. The chip design can be represented by a directed graph G, where the direction indicates the possible communication between the chip components. Consistent testing or simulation of the signal process requires to consider sub-designs of feed-forward communication. These sub-designs can be represented by acyclic subgraphs $G' \subseteq G$, which may be derived by solving the FASP. Especially *circuit testing* [22, 27, 32, 41, 43, 50, 56], including *field programmable gate arrays (FPGAs)*

All authors contributed equally to this research.
Authors' addresses: M. Hecht and K. Gonciarz, Max Planck Institute of Molecular Cell Biology and Genetics, Center for Systems Biology Dresden, Pfotenhauerstrasse 108, Dresden, Germany, 01307; emails: {hecht, gonciarz}@mpi-cbg.de; Szabolcs Horvát, Max Planck Institute of Molecular Cell Biology and Genetics, Center for Systems Biology Dresden, Pfotenhauerstrasse 108, Dresden, Germany, 01307, and Max Planck Institute for the Physics of Complex Systems, Nöthnitzerstrasse 38, Dresden, Germany, 01187; email: horvat@mpi-cbg.de

1084-6654/2021/03-ART1.5
https://doi.org/10.1145/3447652

[10, 34, 48], rely on this approach. Further problems and applications include *efficient deadlock resolution in operating systems* [9, 51], *minimum transversals of directed cuts* [44] and *general minimum multi-cuts* [21], *computational biology and neuroscience* [7, 15, 33, 45]. We recommend References [6, 46] for exploring further relations to graph theoretical problems. It is notable that the typical instances of the mentioned applications are represented by graphs of *large and sparse nature*.

While the undirected version of the FASP can be solved efficiently by computing a maximum spanning tree, the undirected FVSP remains NP–complete. The only known (directed) instance classes possessing polynomial time solutions are planar or more general weakly acyclic graphs [26]. Parameter tractable algorithms are given in References [11, 30]. We recommend Reference [6] for an overview.

1.1 Approximation Theory and the Empirical Factor–2–Approximation

Approximations of constrained versions of the problems are presented in Reference [21], but the approximation ratios depend logarithmically on the number of cycles a graph possesses. By reduction from the Hamiltonian cycle problem, counting all cycles is already a #P–hard counting problem [2] and thereby the proposed approximation is not bounded by a constant ratio. Thus, the long-standing question of the APX completeness of the FASP is not resolved by this result nor by the fact that the FASP on tournaments possesses a PTAS [38].

Below, we give an interpretation of our achieved empirical ratio–2–approximation. The directed FVSP & FASP are approximation preserving L-reducible to each other [3, 14, 21, 30, 36]. Consequently, by L–reduction of the *minimum vertex cover problem* (MVCP), both problems are known to be APX–hard [37]. The reduction from MVCP to FVSP in Reference [37] can be adapted also for the undirected FVSP. Due to Reference [18], the MVCP can not be approximated in polynomial time beneath a ratio of $r = 10\sqrt{5} - 21 \approx 1.3606$, unless P = NP. In light of this fact, the FVSP and the FASP are also not polynomial time approximable beneath that ratio. However, the undirected FVSP can be approximated within ratio $r = 2$ [4, 8] and is thereby APX–complete. This crucial difference between the directed and undirected FVSP does not allow to deduce the APX–completeness of the FASP.

The same is true for the complementary problem of finding a *maximum acyclic subgraph* $G_0 = (V_0, E_0) \subseteq G$. It is known that the *maximum acyclic subgraph problem* (MASP) is MAX SNP complete and thereby approximable [29, 46] with constant ratio. However, the MASP is not exactly complementary to the FASP. The MASP measures approximations with respect to the number of arcs, i.e., is efficiently approximated iff

$$C|E'| \geq |E_0|, \quad C \geq 1 \tag{1}$$

holds for the computed acyclic subgraph $G' = (V', E') \subseteq G$. In contrast, the FASP measures approximability with respect to the minimum feedback arc length $|\varepsilon_0| = |E \setminus E_0| = |E| - |E_0|$, i.e., is efficiently approximated iff

$$|\varepsilon| = |E \setminus E'| = |E| - |E'| \leq C(|E| - |E_0|) = C|\varepsilon_0|, \quad C \geq 1, \tag{2}$$

which is equivalent to

$$|E'| \geq |E| - C(|E| - |E_0|) = C|E_0| + (1 - C)|E| = |E_0| - (C - 1)|\varepsilon_0|.$$

While, Equations (1) and (2) represent completely different approximation requirements, neither the approximability of the MASP nor the results of Reference [28] (Corollary 1.2) apply to the FASP. The precise statement is given by Reference [39], stating that conditioned on the unique games conjecture it is NP–hard to approximate the MVCP beneath a ratio of $2 - \epsilon$, $\epsilon > 0$. Due to the mentioned L-reductions, this also applies to the FASP. Thus, ratios $r \geq 2$ might become

achievable by efficient algorithms, with ratio $r = 2$ being optimal. Although we were not able to prove that our approach always achieves a ratio–2–approximation, the empirical validation of that hypothesis makes further investigation in this direction worth to consider.

1.2 Practical Relevance and Contribution

In our work, we focus on finding a universal accurate and reasonable fast performing solution of the FASP. Even though there is a broad variety of heuristics of the FASP that perform very fast and accurate on specific instances (small graphs, tournaments, planar graphs, etc.) [12, 19, 23, 26, 38, 46, 49, 58], the most relevant instance class for applications, namely, large and sparse graphs, either require non–reasonable runtime when applying exact methods or the accuracy performance of the state-of-the-art heuristics becomes insufficient. The validation of the proposed heuristic suggests that this issue can be resolved by our algorithm.

In References [19, 46, 49], an excellent overview of heuristic solutions is given. As shown in Reference [52], among the fast (linear runtime) universal state-of-the-art heuristics the heuristic *Greedy Removal* (GR) [19] performs best. Exact methods use ILP–solvers with modern formulations given in References [5, 55] based on the results of References [26, 58]. Heuristic and exact solutions for the weighted version are discussed in Reference [23]. However, for dense or large (sparse) graphs the ILP–approaches become sensitive to the NP–hardness of the FASP and require infeasible runtimes. In contrast, the heuristic approach GR solves large instances still in seconds, but by the cost of high approximation ratios $r \gg 2$. Consequently, there is a huge class of relevant synthetic and real-world instances that can not be solved accurately by state-of-the-art approaches within reasonable time. As we demonstrate in this article, our proposed heuristic efficiently computes solutions for such instances within very tight approximation ratios $r \ll 2$.

2 GRAPH THEORETICAL CONSIDERATIONS

In this section, we provide the main graph theoretical concepts, which are required throughout the article.

2.1 Preliminaries

We address the feedback arc set problem in the most general setup. For this purpose, we introduce a non–classical definition of graphs as follows:

Definition 2.1. Let $G = (V, E, \text{head}, \text{tail})$ be a 4–tuple, where V, E are finite sets and head, tail: $E \longrightarrow V$ are some maps. We call the elements $v \in V$ *vertices* and the elements $e \in E$ *arcs* of G, while $\text{head}(e), \text{tail}(e) \in V$ are called *head* and *tail* of the arc e. An arc e with $\text{head}(e) = \text{tail}(e)$ is called a *loop*. In general, we call G a *multi–digraph*. The following cases are often relevant:

(i) G is called a *(simple) digraph* iff the map $H : E \longrightarrow V \times V$, with $H(e) = (\text{tail}(e), \text{head}(e))$ is injective.
(ii) G is called an *undirected graph* iff G is a digraph and for every $e \in E$ there is $f \in E \setminus \{e\}$ with $\text{head}(e) = \text{tail}(f)$, $\text{tail}(e) = \text{head}(f)$. In this case, we slightly simplify notation by shortly writing e for the pair $e := (f, g) \in E \times E$, which is then called an *edge*. The notion of head, tail can thereby be replaced by link: $E \longrightarrow V$ with $\text{link}(e) = \text{head}(e) \cup \text{tail}(e)$.
(iii) In the special case, where $E \subseteq V \times V$ the maps head, tail are assumed to be canonically given by the relation of E, i.e., $\text{head}((x, y)) = y$, $\text{tail}((x, y)) = x$ for all $(x, y) \in E$.

One readily observes that, in the cases $(i), (ii)$, our definition coincides with the common understanding of graphs. In the general case of multi–digraphs, our definition has the advantage that though *multiple arcs* e, f with $\text{head}(e) = \text{head}(f)$, $\text{tail}(e) = \text{tail}(f)$ are allowed e, f are

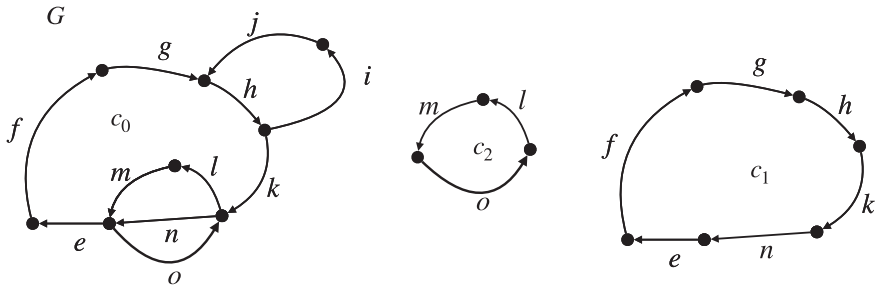

Fig. 1. Elementary and simple cycles.

distinguished. Thus, E is no multi–set, as it is assumed usually, but a simple set, simplifying our considerations. Consequently, the setup allows general and consistent considerations.

For $e \in E$, we denote with $\vec{F}(e) = \{f \in E \mid \text{head}(f) = \text{head}(e), \text{tail}(e) = \text{tail}(f)\}$, $\overleftarrow{F}(e) = \{f \in E \mid \text{tail}(f) = \text{head}(e), \text{head}(f) = \text{tail}(e)\}$, $F(e) = \vec{F}(e) \cup \overleftarrow{F}(e)$ the set of all parallel and anti–parallel arcs and their union, respectively.

Further, two arcs e and f are called *consecutive* if $\text{head}(e) = \text{tail}(f)$ and are called *connected* if $\{\text{head}(e), \text{tail}(e)\} \cap \{\text{head}(f), \text{tail}(f)\} \neq \emptyset$. A *directed path* $p = \{e_1, \ldots, e_n\} \subseteq E$ of length $n \in \mathbb{N}$ from a vertex u to a vertex v is a list of consecutive arcs $e_i \in E$, $i = 1, \ldots, n$ such that $u = \text{tail}(e_1)$ and $v = \text{head}(e_n)$. Thereby, repetition is allowed, i.e., $e_i = e_j$, $1 \leq i < j \leq n$ is possible.

A directed *cycle* is a directed path p from some vertex $v \in V$ to itself, which can also be a *loop*. $O(G)$ shall denote the sets of all directed cycles of G. A cycle is called *simple* or *elementary* if every arc or vertex it contains is passed exactly once, respectively. Certainly, every cycle is given by passing through several elementary cycles. We denote with $\text{O}_{\text{el}}(G) \subseteq O(G)$, the set of all *directed elementary cycles*.

Example 2.2. Consider the graph G in Figure 1 . The cycle $c_0 = \{e, f, g, h, k, l, m, o, n\}$ is a simple and non-elementary cycle, while the cycles $c_1 = \{e, f, g, h, k, n\}$ and $c_2 = \{o, l, m\}$ are elementary. Certainly, c_0 is given by passing through c_1 and c_2.

With $G \setminus e$, $G \setminus v$, we denote the graphs obtained by deleting the arc e or the vertex v and all its connected arcs. Further, $\mathcal{G}(\cdot)$, $\mathcal{E}(\cdot)$, $\mathcal{V}(\cdot)$ denote the graph, the set of all arcs, and the set of all vertices induced by a set of graphs, arcs, and vertices. By $\mathcal{P}(A)$, we denote the power set of a given set A of finite cardinality $|A| \in \mathbb{N}$.

2.2 Problem Formulation

In the following, we formulate the classical optimization problems considered in this article.

Problem 1 (FASP & FVSP). *Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph and $\omega : E \longrightarrow \mathbb{R}^+$ be an arc weight function. Then the* weighted FASP *is to find a set of arcs $\varepsilon \in \mathcal{P}(E)$ such that $G \setminus \varepsilon$ is acyclic, i.e., $O(G \setminus \varepsilon) = \emptyset$ and*

$$\Omega_{G,\omega}(\varepsilon) := \sum_{e \in \varepsilon} \omega(e) \tag{3}$$

is minimized. The weighted minimum feedback vertex set problem (FVSP) is given by considering a vertex weight function $\psi : V \longrightarrow \mathbb{R}^+$ and ask for a set of vertices $\nu \in \mathcal{P}(V)$ such that $G \setminus \nu$ is acyclic, i.e., $O(G \setminus \nu) = \emptyset$ and

$$\Psi_{G,\psi}(\nu) := \sum_{v \in \nu} \psi(v) \tag{4}$$

is minimized. We denote the set of solutions of the FASP & FVSP with $\mathcal{F}_E(G, \omega)$, $\mathcal{F}_V(G, \psi)$, respectively.

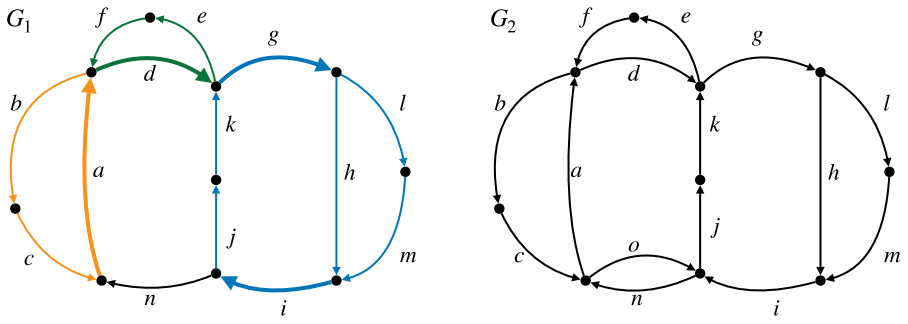

Fig. 2. G_1 with colored isolated cycles $\{a, b, c\}$, $\{d, e, f\}$, $\{g, h, i, j, k\}$, $\{g, l, m, i, j, k\}$, and G_2.

Further, we call $\varepsilon \in \mathcal{F}_E(G, \omega)$ *a* minimum feedback arc set *and* $\nu \in \mathcal{F}_V(G, \psi)$ *a* minimum feedback vertex set *and denote with* $\Omega(G, \omega)$, $\Psi(G, \psi)$ *the* minimum feedback arc/vertex length. *If* ω *or* ψ *are constant functions, then we derive the unweighted versions of the FASP & FVSP, respectively.*

Remark 2.1. Note that checking whether a graph is acyclic or not can be done by *topological sorting* in $O(|E|)$ time [13, 35, 54]. Further, every directed cycle is given by passing through several elementary cycles. Thus, the conditions $O(G) = \emptyset$ and $\mathrm{O_{el}}(G) = \emptyset$ are equivalent. The FVSP & FASP can be also formulated in terms of the *maximum linear ordering problem* see for instance References [5, 46, 58].

2.3 Isolated Cycles

The complexity of an instance G for the FVSP or FASP is certainly correlated to the structure of its cycles. However, by reducing to the Hamiltonian cycle problem, already counting all directed elementary cycles turns out to be a #P-hard problem. This makes it hard to study the structure of $\mathrm{O_{el}}(G)$. Here, we propose to use a technique developed in our previous article [30] to overcome this issue.

Definition 2.3 (Cycle Cover & Isolated Cycles). Let $G = (V, E, \mathrm{head}, \mathrm{tail})$ be a multi–digraph and $e \in E$. We call the subgraph

$$G_e = (V_e, E_e, \mathrm{head}, \mathrm{tail}) := \mathcal{G}\left(\left\{c \in \mathrm{O_{el}}(G) \;\middle|\; \vec{F}(e) \cap c \neq \emptyset\right\}\right) \tag{5}$$

induced by all elementary cycles passing through e or a parallel arcs $f \in \vec{F}(e)$ the *cycle cover* of e. Further, we denote with

$$I_e := \mathcal{G}\left(\left\{c \in \mathrm{O_{el}}(G) \;\middle|\; e \in c, \quad c \cap c' = \emptyset, \;\text{ for all }\; c' \in \mathrm{O_{el}}(G \setminus \vec{F}(e)\right\}\right)$$

the subgraph induced by all *isolated cycles* passing through e, i.e., if c is isolated, then c intersects with no cycle c' passing not through e or some parallel arc of e.

Remark 2.2. Note that every loop is an isolated cycle. Further, the sets of isolated cycles possess a *flat hierarchy* in the following sense. If $e, f \in E$, with $I_e \neq I_f$, then $I_e \cap I_f = \emptyset$. Vice versa, $I_e \cap I_f \neq \emptyset$ implies $I_e = I_f$ and further $G_e = G_f$.

Example 2.4. Consider the graph G_1 in Figure 2. Then a, d, g, i are all arcs of G_1 with $I_x \neq \emptyset$, $x \in \{a, d, g, i\}$. The corresponding isolated cycles are colored. Indeed, $I_g = I_i$ and $G_g = G_i$ hold according to Remark 2.2. Adding the arc o to G_1 yields the graph G_2. Thereby, all isolated cycles are connected with each other. Thus, $I_e = \emptyset$ for all arcs e of G_2.

ALGORITHM 1: ISO-CUT

```
1: procedure ISO-CUT(G, ω)                                   ▷ ω is an arc weight.
2:     ε = ∅, iso = 1
3:     while G is not acyclic & iso = 1 do                   ▷ Check by topological sorting in O(|E|).
4:         iso = 0
5:         for e ∈ E do
6:             Compute I_e and δ = mincut(head(e), tail(e), I_e, ω_{|I_e})
7:             if I_e ≠ ∅ & Ω_{G,ω}(F⃗(e)) ≤ Ω_{G,ω}(δ) then   ▷ 2nd condition redundant if G is a digraph with
                                                              ω ≡ 1.
8:                 G = G \ e
9:                 ε = ε ∪ F⃗(e)
10:                iso = 1
11:            end if
12:        end for
13:    end while
14:    return (G, ε)
15: end procedure
```

Theorem 2.5. *Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph with arc weight $\omega : E \longrightarrow \mathbb{R}^+$.*

(i) *There exist algorithms computing the subgraph G_e in $O(|E|^2 + |V|)$ and I_e in $O(|E| + |V|)$.*
(ii) *If $\varepsilon \in \mathcal{F}_E(G, \omega)$ is a minimum feedback arc set with $e \in \varepsilon$, then $\vec{F}(e) \subseteq \varepsilon$.*
(iii) *If $I_e \neq \emptyset$ and $\delta = \text{mincut}(\text{head}(e), \text{tail}(e), I_e, \omega_{|I_e}) \in \mathcal{P}(E)$ be a minimum–s–t–cut with source $s = \text{head}(e)$, target $t = \text{tail}(e)$ w.r.t. $I_e, \omega_{|I_e}$ such that:*

$$\Omega_{G,\omega}(\delta) \geq \Omega_{G,\omega}(\vec{F}(e)) \,. \tag{6}$$

Then there is $\varepsilon \in \mathcal{F}_E(G, \omega)$ with $\vec{F}(e) \subseteq \varepsilon$.
(iv) *Checking whether $I_e \neq \emptyset$ and Equation (6) holds can be done in $O(|E||V|)$.*

Proof. For better readability, we recaptured this statement from our previous article [30] in Appendix A. □

Theorem 2.5 applies to unweighted graphs as well by setting the weight $\omega \equiv 1$. Theorem 2.5 allows to localize optimal arc cuts $e \in E$ both in the weighted and unweighted cases, whenever isolated cycles with property (*iii*) exist. We will use this circumstance to propose an algorithm solving the FASP.

3 THE ALGORITHM

The building block of the algorithm relies on applying Theorem 2.5 as presented below.

3.1 Building Block

Given a multi–digraph $G = (V, E, \text{head}, \text{tail})$, we formulate an algorithm termed ISO–CUT, which searches for arcs $e \in E$ that satisfy the assumption (*iii*) of Theorem 2.5. If such an arc $e \in E$ is located, then we store e in a list ε, consider $G = G \setminus e$, and continue the search until either the resulting graph G is acyclic or no desired arc can be localized. In any case, the stored arcs ε are an optimal subsolution for the FASP on G. A formal pseudo-code for the algorithm is given in Algorithm 1.

Lemma 3.1. *Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph with arc weight $\omega : E \longrightarrow \mathbb{R}^+$.*

(i) *The algorithm ISO–CUT requires $O(|E|^3 + |V||E|^2)$ runtime to return an optimal subsolution $\varepsilon \subseteq E$ of the FASP on G and the remaining graph $G \setminus \varepsilon$ in the unweighted case.*
(ii) *The analogous return in the weighted case requires $O(|V||E|^3)$ runtime.*
(iii) *If $G \setminus \varepsilon$ is acyclic, then $\varepsilon \in \mathcal{F}_E(G, \omega)$ is a minimum feedback arc set.*

Proof. As one can verify readily, Algorithm 1 contains two loops over E with line 6 being the bottleneck for each iteration. The runtime estimation thereby follows directly from Theorem 2.5 (i) and (iv). Statement (iii) follows from Theorem 2.5 (iii). □

Note that due to Remark 2.2 the algorithm ISO–CUT removes all loops from G. Further, Theorem 2.5 (iii) is always fulfilled for digraphs with constant weight $\omega \equiv 1$ making the condition in line 7 of Algorithm 1 unnecessary.

3.2 A Good Guess

Though isolated cycles allow to localize optimal cuts, they do not need to exist at all, as the Example 2.4 shows. Thus, the algorithm ISO–CUT might not return an acyclic graph. In this case, we have to develop a concept of a *good guess* for cutting G in a pseudo-optimal way until it possesses isolated cycles and thereby ISO–CUT can proceed. Our idea is based on the following fact:

Proposition 3.2. *Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph with arc weight $\omega : E \longrightarrow \mathbb{R}^+$, $e \in E$, with $G_e \neq \emptyset$ and $I_e = \emptyset$. Denote with $\delta = \text{mincut}(\text{head}(e), \text{tail}(e), G_e, \omega_{|G_e})$ a minimum–s–t–cut and with $\varepsilon \in \mathcal{F}_E(G, \omega)$ a minimum feedback arc set, while $\varepsilon' = \varepsilon \cap \mathcal{E}(G_e)$ shall denote its restriction to G_e. If*

$$\Omega_{G,\omega}(\delta) - \Omega_{G,\omega}(\vec{F}(e)) > \Omega(G \setminus \vec{F}(e), \omega) - \Omega(G \setminus \varepsilon', \omega), \tag{7}$$

then $\vec{F}(e) \subseteq \varepsilon$.

Proof. Assume $e \notin \varepsilon$, then due to Theorem 2.5 (ii) there holds $\varepsilon \cap \vec{F}(e) = \emptyset$. Since $G_e \setminus \varepsilon'$ is acyclic and δ is a minimum s–t–cut, we obtain $\Omega_{G,\omega}(\varepsilon') \geq \Omega_{G,\omega}(\delta)$. Hence, by rewriting Equation (7) we can estimate

$$\Omega_{G,\omega}(\vec{F}(e)) + \Omega(G \setminus \vec{F}(e), \omega) < \Omega_{G,\omega}(\delta) + \Omega(G \setminus \varepsilon', \omega) \leq \Omega_{G,\omega}(\varepsilon') + \Omega(G \setminus \varepsilon', \omega) = \Omega(G, \omega),$$

which yields a contradiction and thereby proves the claim. □

Certainly, the right-hand side of Equation (7) is hard to compute or even to estimate. Intuitively, one could guess that the larger the left-hand side becomes, the more likely it is that the inequality in Equation (7) holds. This intuition is the basic idea of our concept of a *good guess*.

However, maximizing the left-hand side of Equation (7) is too costly for a heuristic guess. Therefore, we restrict our considerations to one cycle $c \in \text{O}_{\text{el}}(G)$ and all arcs $e_1, \ldots, e_n \in \mathcal{E}(c)$ with $\mathcal{G}(c) \subsetneq G_{e_i}$, $i = 1, \ldots, n$ that cut more cycles than c. Now, we choose

$$\text{GOOD–GUESS}(G, \omega, c) = \underset{e_i,\, i=1,\ldots,n}{\text{argmax}} \left(\text{mincut}(\text{head}(e_i), \text{tail}(e_i)) - \Omega_{G,\omega}(\vec{F}(e_i)) \right) \tag{8}$$

as an arc with the most expansive minimum s–t–cut to be the one to cut. By combining References [40] and [47], computing minimum s–t–cuts requires $O(|E||V|)$. Consequently, this heuristical decision can be made efficiently in $O(|c||E||V|)$.

ALGORITHM 2: TIGHT–CUT

```
1: procedure TIGHT–CUT(G, ω))                          ▷ ω is an arc weight.
2:     ε = ∅, δ = ∅
3:     for i = 1, . . . |E| do
4:         (G, ε′) = ISO–CUT(G, ω)
5:         ε = ε ∪ ε′
6:         if G is acyclic then                        ▷ Can be checked by topological sorting in O(|E|)
7:             break
8:         else
9:             Choose c ∈ O_el(G)
10:            h = GOOD–GUESS(G, ω, c)
11:            δ = δ ∪ {h}
12:            G = G \ h
13:        end if
14:    end for
15:    return (ε ∪ δ), Ω_{G,ω}(δ)
16: end procedure
```

3.3 The Global Approach

Now, we combine the algorithms ISO–CUT and GOOD–GUESS to yield an algorithm termed TIGHT-CUT computing feedback arc sets formalized in Algorithm 2.

Proposition 3.3. *Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph with arc weight $\omega : E \longrightarrow \mathbb{R}^+$ and vertex weight $\nu : V \longrightarrow \mathbb{R}^+$.*

(i) *The algorithm TIGHT–CUT proposes a feedback arc set $\varepsilon \cup \delta \subseteq E$ in $O(|V||E|^3 + |E|^4)$ in the unweighted case.*
(ii) *In the weighted case the analogous return requires $O(|V||E|^4)$ runtime.*
(iii) *The algorithm TIGHT–CUT can be adapted to propose a feedback vertex set $\nu \subseteq E$ in $O(|E|(\Delta(G)|V|)^3)$ in the unweighted case and $O(|E|(\Delta(G)|V|)^4)$ in the weighted case, where $\Delta(G)$ denotes the maximum degree of G.*

Proof. Obviously TIGHT–CUT runs once through all arcs E in the worst case. Due to the notion of our GOOD–GUESS (8) the bottleneck is thereby ISO–CUT. Thus, due to Lemma 3.1, we obtain (i), (ii). Now (iii) is a consequence of an existing approximation preserving L–reduction from the FVSP to the FASP due to References [3, 14, 21, 30, 36]. □

4 RELAXED VERSION: TIGHT–CUT*

To make the algorithm ISO–CUT faster and more effective, we propose the following relaxation within TIGHT–CUT:

Definition 4.1 (Almost Isolated Cycles). Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph $n \in \mathbb{N}$ and $e \in E$. If there is a set $\mu \subseteq E$ of n arcs, i.e., $|\mu| = n$ such that

$$I_e \neq \emptyset \quad \text{w.r.t.} \quad G \setminus \mu,$$

then we call the cycles $c \in O_{el}(I_e)$ *almost isolated cycles*. If $n = 0$, then we obtain the notion of Definition 2.3.

As long as there are almost isolated cycles for small $n \in \mathbb{N}$ one can hope that the accuracy of TIGHT–CUT remains high. We take this relaxed notion into account as follows: If no isolated cycles were found, then we generate N graphs $H_i = G \setminus \mu_i$ by uniform at random deleting arcs

ALGORITHM 3: TIGHT–CUT*

procedure TIGHT–CUT*(G, ω, n, N) ▷ ω is an arc weight, $n, N \in \mathbb{N}$.
  $\varepsilon = \emptyset, \delta = \emptyset$
  for $i = 1, \ldots |E|$ **do**
    $(G, \varepsilon') =$ ISO–CUT(G, ω)
    $\varepsilon = \varepsilon \cup \varepsilon'$
    if G is acyclic **then** ▷ Can be checked by topological sorting in $O(|E|)$
      break
    else
      Choose $\mu_1, \ldots, \mu_N \subseteq E$ with $|\mu_i| = n$ uniform at random.
      $(H_i, \varrho_i) =$ ISO–CUT$(G \setminus \mu_i)$
      $R = \cup_{i=1}^{N} \varrho_{i,1}$ ▷ $\varrho_{i,1}$ is the first arc cut by ISO–CUT.
      if $R \neq \emptyset$ **then**
        $f = \operatorname{argmax}_{e \in R} \left| \{e = \varrho_{i,1}\}_{1 \leq i \leq N} \right|$ ▷ f is a good choice in most of the H_i.
        $\varepsilon = \varepsilon \cup \{f\}$
        $G = G \setminus f$
      else
        Choose $c \in \mathrm{O}_{\mathrm{el}}(G)$
        $h =$ GOOD–GUESS(G, ω, K)
        $\delta = \delta \cup \{h\}$
      end if
      $G = G \setminus h$
    end if
  end for
  return $(\varepsilon \cup \delta), \Omega_{G,\omega}(\delta)$
end procedure

$\mu_i \subseteq E$, $|\mu_i| = n$, $i = 1, \ldots, N$, $n, N \in \mathbb{N}$ and ask for the existence of almost isolated cycles, i.e., search for arcs f in H_i with $I_f \neq \emptyset$. The arc appearing most in all the explored graphs H_i is assumed to be a good choice for cutting it in the original graph. If no such arc can be found, then we use GOOD–GUESS for making a choice in any case. The relaxation is formalized in Algorithm 3.

5 VALIDATION & BENCHMARKING

We implemented the relaxed version TIGHT–CUT* in C++. For benchmarking the algorithm, we designed several experiments, all of which were run in single-threaded mode on a machine with CPUs: 2× Intel(R) Xeon(R) E5-2660 v3 @ 2.60 GHz; Memory: 128 GB; OS: Ubuntu 16.04.6 LTS using compiler: GCC 9.2.1. The C++ code, as well as all benchmark datasets used in this work, are publicly available at https://git.mpi-cbg.de/mosaic/FaspHeuristic.

The following implementations were used for the experiments:

(I) An exact integer linear programming–based approach implemented as the `feedback_edge_set` function from *SageMath 8.9* [55] with iterative constraint generation, termed *EM*.
(II) The greedy removal approach from Reference [19], termed *GR*, imported from the *igraph library* [16, 31].
(III) The relaxed version TIGHT–CUT* with settings $n = 3$, $N = 20$.

As mentioned in the introduction, among several (linear runtime) universal heuristics, GR produces the most accurate results [52]. EM is similar to the approach from Reference [5] and iteratively increases the *cycle matrix* required for the optimization. Thereby, a sequence $\varepsilon_1, \ldots, \varepsilon_n$,

$n \in \mathbb{N}$ of optimal subsolutions is generated with $\varepsilon_n \in \mathcal{F}_E(G, \omega)$, $\omega \equiv 1$ being a global solution for G. EM was chosen due to its excellent performance, which allowed including relatively large instances in the benchmark set. However, the method can not handle the weighted case. We chose the GLPK back end for SageMath's integer programming solver, which we found to perform significantly better than COIN-OR's CBC or Gurobi for this problem. In light of these facts, we consider our benchmark set to be sufficient to cover the diversity of universal approaches solving the FASP. We have made an implementation of our approach available at https://git.mpi-cbg.de/mosaic/FaspHeuristic so readers would be able to easily test our method and determine if it works sufficiently well for their particular application.

5.1 Synthetic Instances

To compare approximation ratios and runtimes, we generated the following instance classes:

(i) We used the Erdős–Rényi model to generate random digraphs [20].
(ii) We chose uniformly at random a direction for every edge in a complete undirected graph K_n to generate tournaments of size $n \in \mathbb{N}$.
(iii) We generated random maximal planar digraphs, then considered uniform perturbations of planarity by randomly *rewiring*, i.e., removing and re-inserting, a fraction $0 \leq p \leq 1$ of arcs. This construction is similar to the Watts–Strogatz *small-world model* [57].
(iv) We followed Reference [49] to generate large digraphs G of known feedback arc length. Adaptions to treat the weighted case were made.

The broad class of considered instances demonstrates the universality of TIGHT–CUT*. Later on, computations of real-world instances, with similar structure as the instances in (*iv*), yield almost exact solutions. More detailed discussion is given below.

Experiment 5.1. *In total, we generated 1,869 random digraphs. Figure 3 shows the approximation ratios obtained by TIGHT–CUT* and GR on 967 out of these 1,869 graphs plotted once against $|E|/|V|$ and once against the exact minimum feedback arc length. The exact feedback length was determined by EM whose runtime ratio w.r.t. TIGHT–CUT* is plotted in Figure 4. Thereby, the empty region in the left panel reflects the 902 instances that EM could not process within EM–time–out = 30 min. The runtimes of TIGHT–CUT* are plotted for the entire benchmark set of 1,869 random graphs in the left panel of Figure 5.*

Figure 3 validates that TIGHT–CUT approximates the FASP beneath a ratio of at most 1.6 by being much tighter in most of the cases. However, GR reaches ratios up to 2.5. The parameter tractable algorithm of Reference [11] indicates that the feedback length reflects the complexity of a given instance. However, the accuracy of GR decreases quickly with increasing graph size and $|E|/|V|$ regardless of the feedback length. In contrast, the accuracy behavior of TIGHT–CUT* is consistent with the finding of Reference [11]. In any case, TIGHT–CUT* performs significantly better than GR. Notably, when approaching the time-out-region of EM, the approximation ratios of TIGHT–CUT* remain small. Thus, for digraphs located above the red region in Figure 4 the plot in Figure 6 shows that TIGHT–CUT* is up to 100 times faster than EM. Even though TIGHT–CUT* requires up to 3 minutes and GR runs under 1 second on these graphs, TIGHT–CUT* is the only approach producing accurate results in reasonable time.*

Figure 5 shows the absolute runtimes of TIGHT–CUT* in seconds for all instances in the random benchmark set. EM has an advantage over TIGHT–CUT* at runtimes below approximately 10–30 seconds, below the black dashed line. However, more difficult instances, above the dashed line, take hours or days for EM to solve, while TIGHT–CUT* finds a solution in minutes.

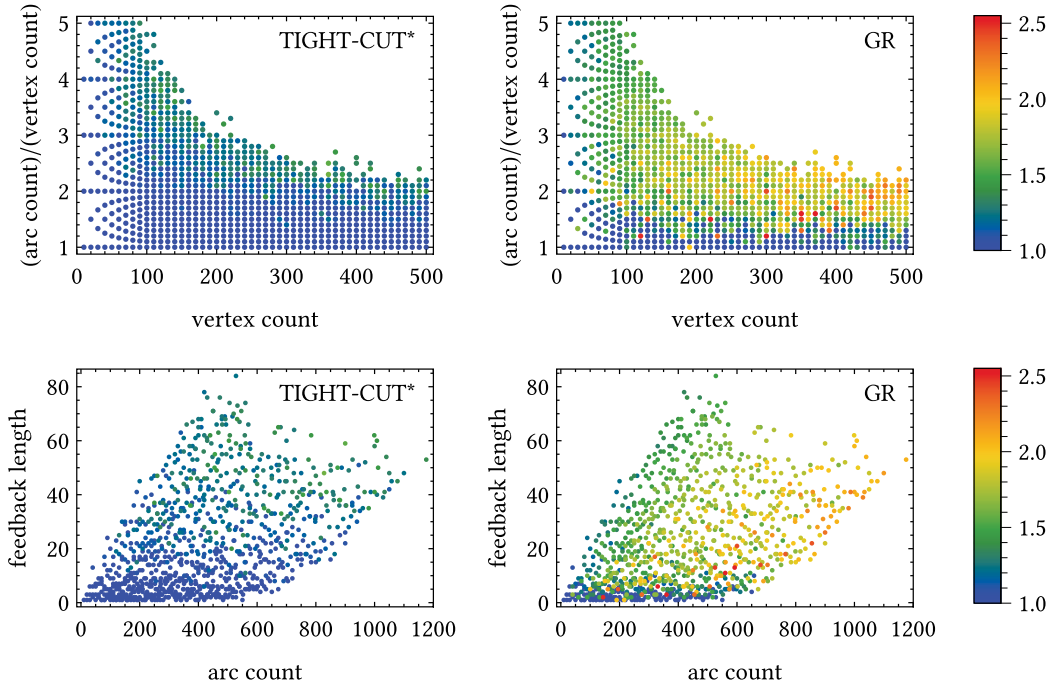

Fig. 3. Approximation ratios of TIGHT–CUT* (left) and GR (right) for random graphs.

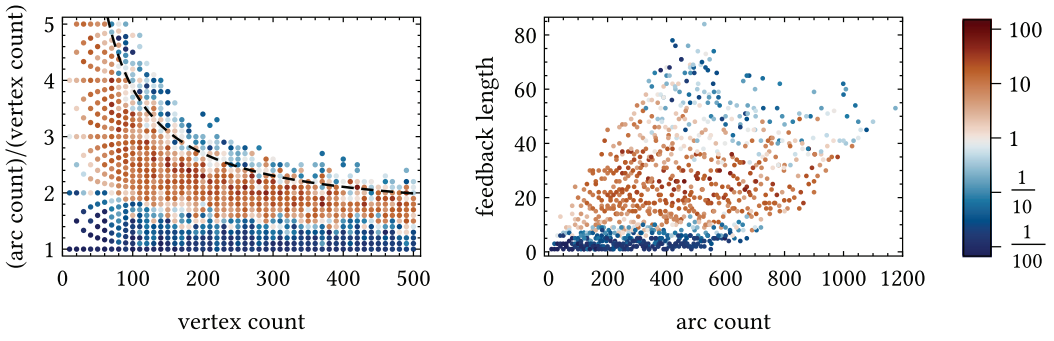

Fig. 4. Runtime ratios of TIGHT–CUT* / EM plotted against $|E|/|V|$ (left) and feedback length (right) for random graphs. TIGHT–CUT* performs better than EM above the dashed line.

Experiment 5.2. *We focus our considerations on tournaments. In Figure 7 the results for* $|V| = 1, \ldots, 27$ *with 10 instances for each size are shown.* $|V| = 27$ *is thereby the maximum size for EM not running into time–out = 30 min. GR seems to perform only slightly worse than TIGHT–CUT*. However, the feedback arc length for tournaments averages about 25% of its high arc count* $|E| = (|V|^2 + |V|)/2$. *Thus, the improvement in accuracy TIGHT–CUT* gains compared to GR is as significant. Again the NP-hardness of the FASP becomes visible for the runtime ratios in Figure 7 and Figure 6 (left). As expected, the feedback arc length is correlated to the complexity of the cycle structure of G. Regardless of the type of the graphs, we thereby reach intractable instances for EM beyond a feedback arc length of* $\Omega(G, \omega) \geq 50$. *Thus, for instances allowing* $\Omega(G, \omega) \geq 50$, *EM might run into time out while TIGHT–CUT* processes them efficiently.*

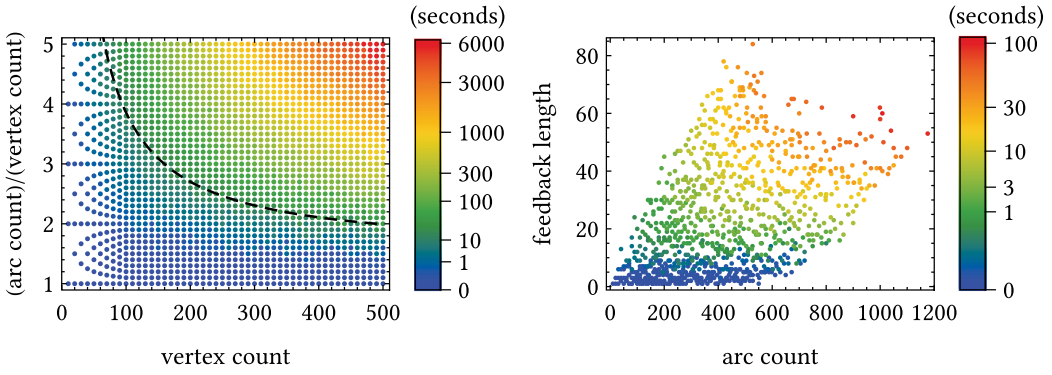

Fig. 5. Runtime of TIGHT–CUT* in seconds plotted against $|E|/|V|$ for all random instances (left) and plotted against feedback length for instances solvable by EM (right). The color bars are scaled to a power 1/4 of the runtimes for better visibility and in accordance with the complexity of the algorithm. As in Figure 4, the dashed line indicates the boundary above which TIGHT–CUT* performs faster than EM. It corresponds to approximately 10–30 seconds.

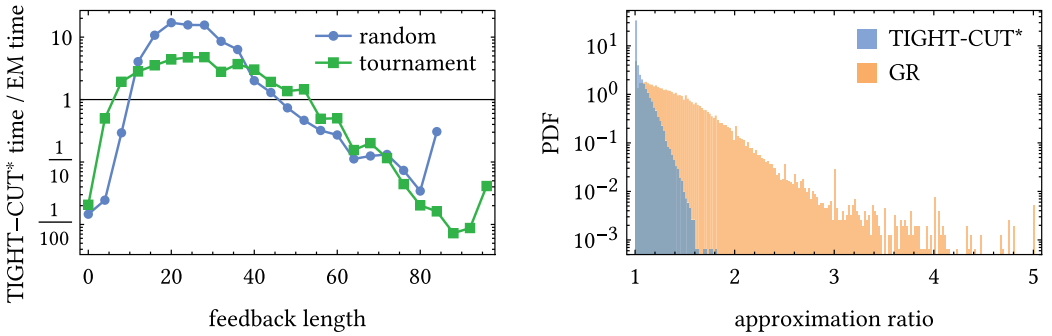

Fig. 6. Runtime ratios of TIGHT–CUT* / EM (left) and distribution of TIGHT–CUT* and GR on weighted digraphs with logarithmic y–scale (right).

Although there are specific algorithms for the FASP on tournaments [12, 38] that might perform better than GR, these algorithms are limited to this special instance class. Therefore, an explicit comparison with these non-universal approaches is omitted here.

Experiment 5.3. *Since EM can not handle the weighted FASP, we adapted the method of Reference* [49] *to generate 77,700 weighted multi–digraphs* (G, ω) *of integer weights* $\omega(e) = 1 \sim 10$ *with known feedback arc length* $\Omega(G, \omega) = 1 \sim 287$ *and sizes from* $|V| = 100 \sim 500$ *and* $|E|/|V| = 1.5 \sim 5$. *Figure 6 (right) illustrates the results. To merge the ratio distributions of GR and TIGHT–CUT* on one plot, we chose a logarithmic scaling for the y–axis. Indeed, TIGHT-CUT* approximates the FASP beneath a ratio of* 2 *and solves more than* 50% *exactly and* 95% *beneath a ratio of* 1.18. *In contrast, GR is spread over ratios from* 1 *to* 5, *producing exact solutions only for* 8.8% *and* 95% *beneath a ratio of* 1.96. *Thus, though GR runs beneath* 1 *sec and the runtimes of TIGHT–CUT* vary from seconds to* 3 *minutes, this accuracy improvement justifies the larger amount of time.*

Experiment 5.4. *In Figure 8, the EM runtimes and TIGHT–CUT* approximation ratios for 541 small–world (perturbed planar digraphs) are plotted. As one can observe already for small perturbations, a similar behavior as for random graphs occurs. In applications, one can rarely guarantee planarity. At best, one can hope for planar-like instances, i.e., graphs that are planar when*

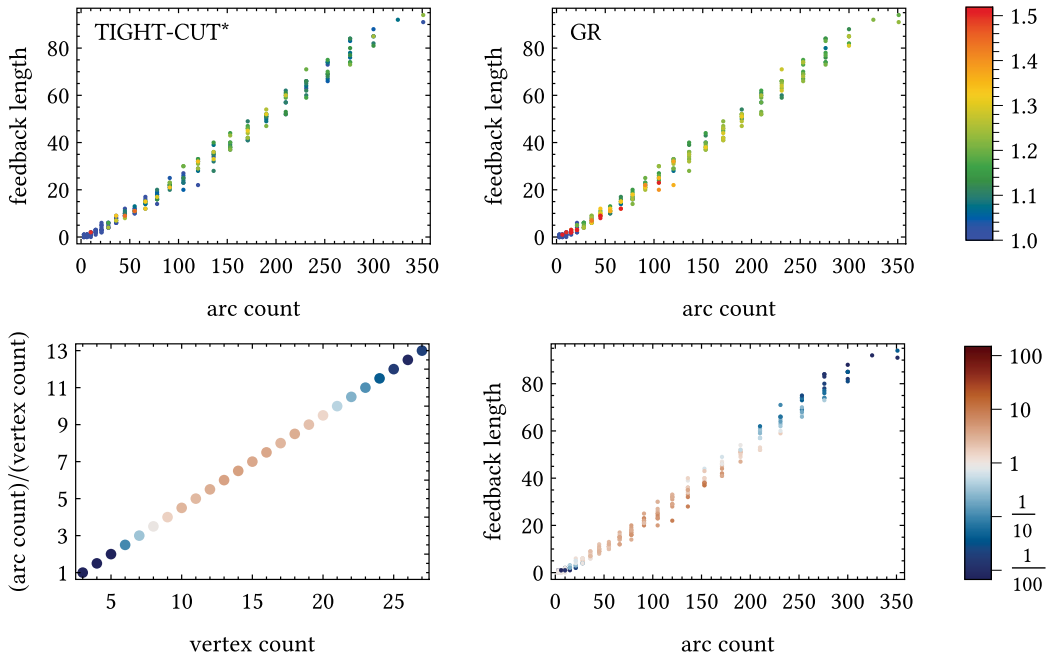

Fig. 7. Approximation ratios (above) of TIGHT–CUT* (left) and GR (right) and runtime ratios (below) of TIGHT–CUT*/EM vs. vertex count (left) and arc count (right) on tournaments.

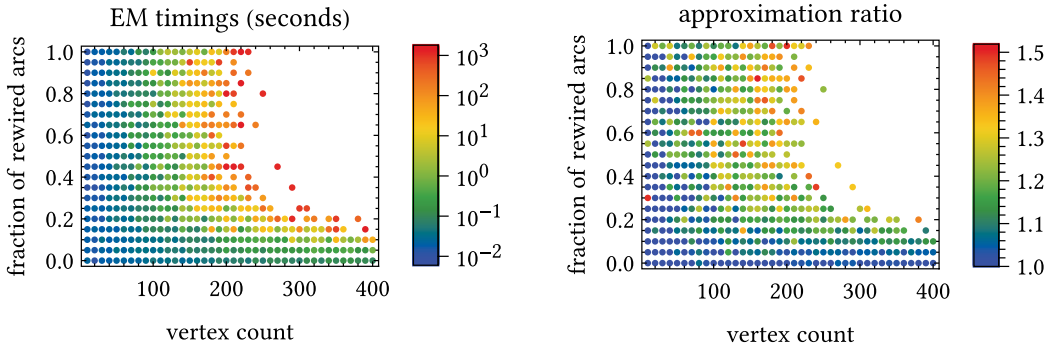

Fig. 8. Runtimes of EM for perturbed planar graphs (left) and approximation ratio of TIGHT–CUT* (right).

rearranging/removing a few arcs. Consequently, real-world instances hinder the efficiency of ILP–Solvers on planar graphs to come into effect. Therefore, TIGHT–CUT is an alternative to EM worth considering even for planar-like graphs.*

Experiment 5.5. *We measured the accuracy behavior of GR and TIGHT–CUT* in the time–out region of EM. Therefore, we used the method of Reference* [49] *to generate very large unweighted digraphs with known feedback arc length* $\Omega(G, \omega) = 20$ *and varying vertex size* $|V| = 100, 200, \ldots, 1000$*, ten instances for each size, with 5% density, i.e.,* $|E|/(|V|(|V| - 1)) = 5\%$*. In Figure 9, the computed feedback lengths of both approaches are plotted with error bars indicating the standard deviation. TIGHT-CUT* delivers almost exact solutions within minutes. In contrast, GR computes feedback sets being more than 1,000 times larger than the minimum. We note that these*

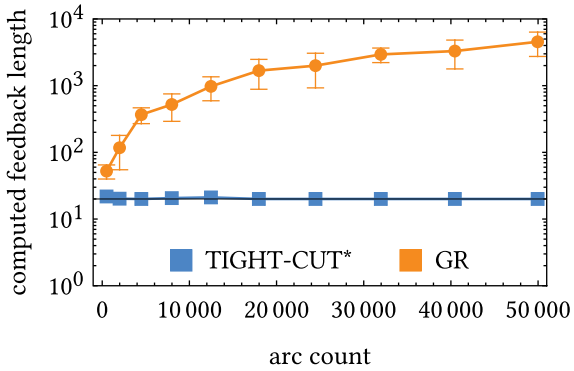

Fig. 9. Accuracy for TIGHT–CUT* and GR on very large digraphs of density $|E|/(|V|(|V|-1)) = 5\%$. The minimum feedback arc length of each instance is 20.

Table 1. Examples of High Feedback Arc Length for Validating the Ratio Approximation

vertices	arcs	$\Omega(G, \omega)$	TIGHT–CUT*	approx. ratio
100	990	200	321	1.61
100	990	200	279	1.40
200	3,980	200	280	1.40
500	1,500	200	334	1.67
501	1,501	200	345	1.73

graphs have a very small feedback length relative to their arc count, i.e., they are "nearly acyclic." Such graphs are very likely to contain arcs that possess isolated cycles, which explains the excellent performance of TIGHT–CUT. Thus, TIGHT–CUT* can detect that a graph is close to acyclic, while GR is unsuitable for this task, regardless of its speed (executed in seconds).*

Experiment 5.6. *We generated a few very large and dense unweighted graphs with large feedback length* $\Omega(G, \omega) = 200$. *The results are listed in Table 1 and validate that TIGHT–CUT* approximates the FASP beneath a ratio of* 2.

5.2 Real-world Datasets

Feedback problems find applications in *circuit testing*, as efficient testing requires the elimination of feedback cycles [22, 27, 32, 41, 43, 50, 56]. Here, we consider graphs generated from the IS-CAS circuit testing datasets, made available in Reference [17] and at https://github.com/alidasdan/graph-benchmarks. The results of these usually non–planar graphs are summarized in Table 2. All examples were solved in runtime comparable to that of EM, except for "dsip," which could not be solved by EM within a computation time of 1 day and was solved by TIGHT–CUT* in 10 minutes. The runtime of the other examples ranged from milliseconds to minutes.

We also considered circuits from the ISPD98 benchmark [1, 17]. These graphs are much larger, with arc counts ranging from 36, 000 to 670, 000. The exact solution could only be obtained for one graph ("ibm05") by EM with a runtime limit of 1 day. Thereby, EM took 30 minutes and TIGHT–CUT* obtained an exact solution for "ibm05" in 2 minutes. Without limiting timeout, TIGHT–CUT* could process "ibm01," "ibm02" in 6 and 13 days, respectively. The resulting feedback sizes are 1.25 ∼ 1.85 times smaller than the solutions proposed by GR (see again Table 2).

Table 2. Feedback Arc Set Size Results for Graphs Generated from the ISCAS and ISPD98 Circuit Benchmarks

circuit name	vertices	arcs	$\Omega(G, \omega)$	TIGHT–CUT*	GR
s27	55	87	2	2	2
s208	83	119	5	5	5
s420	104	178	1	1	1
mm4a	170	454	8	8	16
s382	273	438	15	15	29
s344	274	388	15	15	23
s349	278	395	15	15	24
s400	287	462	15	15	28
s526n	292	560	21	21	29
mult16a	293	582	16	16	23
s444	315	503	15	15	20
s526	318	576	21	21	31
mult16b	333	545	15	15	22
s641	477	612	11	11	16
s713	515	688	11	11	16
mult32a	565	1,142	32	32	45
mm9a	631	1,182	27	27	29
s838	665	941	32	32	37
s953	730	1,090	6	6	11
mm9b	777	1,452	26	27	31
s1423	916	1,448	71	71	112
sbc	1,147	1,791	17	17	21
ecc	1,618	2,843	115	115	137
phase decoder	1,671	3,379	55	55	64
daio receiver	1,942	3,749	83	83	123
mm30a	2,059	3,912	60	60	62
parker1986	2,795	5,021	178	178	313
s5378	3,076	4,589	30	30	75
s9234	3,083	4,298	90	91	163
bigkey	3,661	12,206	224	224	224
dsip	4,079	6,602	—	153	165
s38584	20,349	34,562	1,080	1,080	1,601
s38417	24,255	34,876	1,022	1,022	1,638
ibm01	12,752	36,048	—	1,761	3,254
ibm02	19,601	57,753	—	3,820	5,726
ibm05	29,347	98,793	4,769	4,769	5,979

TIGHT–CUT* is exact except in "mm9b" and "s9234," with failure 1. For "dsip," no exact solution is available. Results for "ibm01," "ibm02," "ibm05" are given in the last three lines.

The accuracy improvement gained by TIGHT–CUT* makes circuit testing much more efficient and robust for these graphs. Therefore, we aim to speed up our implementation such that runtimes under 1 day can be achieved for the ISPD98 instances. How these aims may become achievable and other remaining issues can be resolved is discussed in the final section.

6 CONCLUSION

We presented a new $O(|V||E|^4)$–heuristic termed TIGHT–CUT of the FASP that is adaptable for the FVSP in $O(|E|(\Delta(G)|V|)^4)$ processing even-weighted versions of the problems. At least by validation, the ratio r of the implemented relaxation TIGHT–CUT* is shown to be bounded by $r \leq 2$ (in the unweighted case) on all (more than 6,000) instances with known minimum feedback length and is much tighter in most of the cases. Though we followed several ideas, we can not deliver a proof of the APX–completeness for the directed FVSP & FASP at this time. Nevertheless, we are optimistic that a deeper understanding of isolated cycles may provide a path for proving the ratio r to be bounded by 2 for all possible instances. In any case, while the FVSP & FASP can be L-reduced to each other, either both problems are APX–complete or none of them.

Regardless of these theoretical questions, validation and benchmarking with the heuristic GR [19] and the ILP–method EM from Reference [55] demonstrated the high–quality performance of TIGHT–CUT* even in the weighted case. Though of runtime complexity $O(|V||E|^4)$, the approach makes it possible to compute solutions with tight approximation ratios within minutes for instances that can be solved by exact methods only in hours or days. Especially when considering the real-world instances, such as the graphs from the ISCAS circuit benchmarks, the results suggest that we have found a convenient almost exact solution for this relevant use case.

In light of this fact, we consider it worthwhile to spend further resources on improving the runtime of the implementation to be able to solve more instances from the ISPD98 circuit dataset. Runtime improvements of TIGHT–CUT* are certainly possible by parallelization and by using more efficient implementations of subroutines, e.g., using a dynamic decremental computation of strongly connected components [42] and using the improved minimum–s–t–cut algorithms from Reference [25]. Besides these ideas, dynamically adapting the hyper-parameter settings of TIGHT–CUT* is expected to yield the most significant gains. A fast implementation of the L–reduction from the FVSP to the FASP is in progress allowing to solve the FVSP by TIGHT–CUT* with the same accuracy in similar time.

We hope that many of the applications, even those that are not mentioned here, will benefit from our approach.

APPENDIX

A PREVIOUS RESULTS

We deliver the outstanding proofs of the statements in Section 2.3. As already mentioned, these statements were already proven in our previous work [30] and are given here in a simplified version.

THEOREM A.1. *Let $G = (V, E, \text{head}, \text{tail})$ be a multi–digraph with arc weight $\omega : E \longrightarrow \mathbb{R}^+$.*

(i) *There exist algorithms computing the subgraph G_e in $O(|E|^2 + |V|)$ and I_e in $O(|E| + |V|)$.*
(ii) *If $\varepsilon \in \mathcal{F}_E(G, \omega)$ is a minimum feedback arc set with $e \in \varepsilon$, then $\vec{F}(e) \subseteq \varepsilon$.*
(iii) *If $I_e \neq \emptyset$ and $\delta = \text{mincut}(\text{head}(e), \text{tail}(e), I_e, \omega_{|I_e}) \in \mathcal{P}(E)$ be a minimum–s–t–cut with $s = \text{head}(e)$, $t = \text{tail}(e)$ w.r.t. $I_e, \omega_{|I_e}$ such that:*

$$\Omega_{G,\omega}(\delta) \geq \Omega_{G,\omega}(\vec{F}(e)) \,. \tag{9}$$

Then there is $\varepsilon \in \mathcal{F}_E(G, \omega)$ with $\vec{F}(e) \subseteq \varepsilon$.
(iv) *Checking whether $I_e \neq \emptyset$ and Equation (9) holds can be done in $O(|E||V|)$.*

PROOF. We show (i). Certainly, there has to be a directed path p from head(e) to tail(j) and from head(j) to tail(e) for every arc $j \in \mathcal{E}(G_e)$. If $c \in O(G) \setminus \text{O}_{\text{el}}(G)$ with $e \in \mathcal{E}(c)$ is a non–elementary cycle passing through e, then there is at least one arc $j \in \mathcal{E}(c) \setminus \{e\}$ such that head(j) or tail(j) are

passed twice by c. Hence, either there is no directed path p from head(e) to tail(j) in $G \setminus \vec{N}(\text{head}(j))$ or there is no directed path p from head(j) to tail(e) in $G \setminus \overleftarrow{N}(\text{tail}(j))$. We denote with $J(c)$ all such arcs. If f is an arc of an elementary cycle $c \in \mathrm{O}_{\mathrm{el}}(G)$, then none of the cases occur, i.e., $f \notin J(c)$ (see Figure 1). Thus, determining $J(c)$ can be done by running depth first search (DFS) at most $|E|$ times requiring $\mathcal{O}(|E|^2)$ operations. The strongly connected component $G' = \mathrm{SCC}_e(G)$ of $G \setminus J(c)$ that includes head(e) and tail(e) therefore coincides with G_e and can be be determined in $\mathcal{O}(|E| + |V|)$, [53]. Now, we consider the set $H(e) = \{f \in E_e \mid \mathrm{O}_{\mathrm{el}}(f) \neq \emptyset \text{ w.r.t. } G \setminus e\}$, which can be determined by computing the SCCs of $G \setminus e$. The SCC of $(G \setminus H(e)) \cup \{e\}$ that includes e yields I_e finishing the proof.

We prove (ii). Let $\varepsilon \in \mathcal{F}_E(G, \omega)$ and $e \in \varepsilon$. Assume there is $f \in \vec{F}(e) \setminus \varepsilon$ then certainly $\varepsilon \cap \overleftarrow{F}(e) = \overleftarrow{F}(e)$ otherwise there would be a two–cycle that is not cut. Since $\varepsilon \in \mathcal{F}_E(G, \omega)$, we have $\mathcal{E}(c) \cap \varepsilon \neq \emptyset$ for all $c \in \mathrm{O}_{\mathrm{el}}(f)$ implying $\mathcal{E}(c) \cap (\varepsilon \setminus \{e\}) \neq \emptyset$ for all $c \in \mathrm{O}_{\mathrm{el}}(e)$ contradicting the minimality of ε. Thus, $\varepsilon \cap \vec{F}(e) = \vec{F}(e)$.

We prove (iii). Indeed Equation (9) implies that there is no arc $f \in E$ with $G_f \supseteq I_e$ and $\Omega_{G,\omega}(\vec{F}(f)) < \Omega_{G,\omega}(\vec{F}(e))$. However, due to Remark 2.2, every arc $h \in E$ with $I_h = I_e$ satisfies $G_e = G_h$. Due to $G_e \supseteq I_e$, this implies that $G_e \supseteq G_\delta$. Thus, if $\Omega_{G,\omega}(\delta) \geq \Omega_{G,\omega}(\vec{F}(e))$, then

$$\Omega(G \setminus \delta, \omega) \geq \Omega(G \setminus \vec{F}(e), \omega),$$

which implies (iii).

An exhaustive list of polynomial time algorithms with runtime complexity contained in $\mathcal{O}(|E|^3)$ computing minimum–s–t–cuts is given in References [24, 25]. Especially for the unweighted case in Reference [25], an algorithm with $\mathcal{O}(|E||V|)$ or even faster is presented. A combination of References [40] and [47] ensures that complexity also for the weighted version. Due to (i), this shows (iv). □

ACKNOWLEDGMENTS

We thank Ivo F. Sbalzarini and Christian L. Müller for inspiring discussions and suggestions.

REFERENCES

[1] Charles J. Alpert. 1998. The ISPD98 circuit benchmark suite. In *Proceedings of the International Symposium on Physical Design (ISPD'98)*. ACM Press, New York, New York, 80–85. DOI: https://doi.org/10.1145/274535.274546

[2] Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity: A Modern Approach* (1st ed.). Cambridge University Press, New York, NY.

[3] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. 1980. Structure preserving reductions among convex optimization problems. *J. Comput. Syst. Sci.* 21, 1 (1980), 136–153.

[4] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. 1999. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. Disc. Math.* 12, 3 (1999), 289–297.

[5] Ali Baharev, Hermann Schichl, Arnold Neumaier, and Tobias Achterberg. 2015. An exact method for the minimum feedback arc set problem. *University of Vienna* 10 (2015), 35–60. https://www.mat.univie.ac.at/~neum/ms/minimum_feedback_arc_set.pdf.

[6] Jürgen Bang-Jensen and Gregory Z. Gutin. 2008. *Digraphs: Theory, Algorithms and Applications* (2nd ed.). Springer Publishing Company, Inc.

[7] Yu Bao, Morihiro Hayashida, Pengyu Liu, Masayuki Ishitsuka, Jose C. Nacher, and Tatsuya Akutsu. 2018. Analysis of critical and redundant vertices in controlling directed complex networks using feedback vertex sets. *J. Comput. Biol.* 25, 10 (2018), 1071–1090.

[8] Ann Becker and Dan Geiger. 1996. Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. *Artif. Intell.* 83, 1 (1996), 167–188.

[9] Lubomir Bic and Alan C. Shaw. 1988. *The Logical Design of Operating Systems*. Prentice-Hall, Inc.

[10] Michael Caffrey, Manuel Echave, Charles Fite, Tony Nelson, Anthony Salazar, and Steven Storms. 2002. A space-based reconfigurable radio. In *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA'02)*. Citeseer, 49–53.

[11] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. 2008. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM* 55, 5 (2008), 21:1–21:19.
[12] Tom Coleman and Anthony Wirth. 2010. Ranking tournaments: Local search and a new algorithm. *J. Exper. Alg.* 14 (2010), 2–6.
[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms*. The MIT Press and McGraw-Hill , 636–640.
[14] Pierluigi Crescenzi, Viggo Kann, M. Halldórsson, and M. Karpinski. 1995. A compendium of NP-optimization problems. http://www.nada.kth.se/~viggo/index-en.html, emailviggo@nada.kth.se.
[15] Francis Crick and Christof Koch. 1998. Constraints on cortical and thalamic projections: The no-strong-loops hypothesis. *Nature* 391, 6664 (1998), 245–250. DOI: https://doi.org/10.1038/34584
[16] Gábor Csárdi and Tamás Nepusz. 2006. The igraph software package for complex network research. *InterJ. Complex Syst.* 1695, 5 (2006), 1–9.
[17] Ali Dasdan. 2004. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Trans. Des. Automat. Electron. Syst.* 9, 4 (2004), 385–418. DOI: http://doi.acm.org/10.1145/1027084.1027085
[18] Irit Dinur and Samuel Safra. 2004. On the hardness of approximating minimum vertex cover. *Ann. Math.* 162 (2004), 439–485.
[19] Peter Eades, Xuemin Lin, and W. F. Smyth. 1993. A fast and effective heuristic for the feedback arc set problem. *Inf. Proc. Lett.* 47, 6 (Oct. 1993), 319–323. DOI: https://doi.org/10.1016/0020-0190(93)90079-O
[20] Pál Erdős and Alfréd Rényi. 1959. On random graphs I. *Pub. Math.* 6 (1959), 290–297.
[21] G. Even, J. (Seffi) Naor, B. Schieber, and M. Sudan. 1998. Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica* 20, 2 (1998), 151–174.
[22] Pietro Fezzardi, Marco Lattuada, and Fabrizio Ferrandi. 2017. Using efficient path profiling to optimize memory consumption of on-chip debugging for high-level synthesis. *ACM Trans. Embed. Comput. Syst.* 16, 5s (2017), 1–19.
[23] Merrill M. Flood. 1990. Exact and heuristic algorithms for the weighted feedback arc set problem: A special case of the skew-symmetric quadratic assignment problem. *Networks* 20, 1 (1990), 1–23.
[24] Andrew V. Goldberg and Robert E. Tarjan. 1988. A new approach to the maximum-flow problem. *J. ACM* 35, 4 (1988), 921–940.
[25] Andrew V. Goldberg and Robert E. Tarjan. 2014. Efficient maximum flow algorithms. *Commun. ACM* 57, 8 (2014), 82–89. DOI: https://doi.org/10.1145/2628036
[26] Martin Grötschel, Michael Jünger, and Gerhard Reinelt. 1985. On the acyclic subgraph polytope. *Math. Prog.* 33, 1 (1985), 28–42.
[27] Rajesh Gupta and Melvin A. Breuer. 1989. BALLAST: A methodology for partial scan design. In *Proceedings of the 19th International Symposium on Fault-Tolerant Computing*. IEEE, 118–125.
[28] Venkatesan Guruswami, Rajsekar Manokaran, and Prasad Raghavendra. 2008. Beating the random ordering is hard: Inapproximability of maximum acyclic subgraph. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*. IEEE, 573–582.
[29] Refael Hassin and Shlomi Rubinstein. 1994. Approximations for the maximum acyclic subgraph problem. *Inf. Proc. Lett.* 51, 3 (1994), 133–140.
[30] Michael Hecht. 2018. Exact localisations of feedback sets. *Theor. Comput. Syst.* 62, 5 (2018), 1048–1084.
[31] Szabolcs Horvát. 2020. IGraph/M—the igraph interface for Mathematica (version 0.4). DOI: https://doi.org/10.5281/zenodo.1134932
[32] Anand V. Hudli and Raghu V. Hudli. 1994. Finding small feedback vertex sets for VLSI circuits. *Microproc. Microsyst.* 18, 7 (1994), 393–400.
[33] Iaroslav Ispolatov and Sergei Maslov. 2008. Detection of the dominant direction of information flow and feedback links in densely interconnected regulatory networks. *BMC Bioinf.* 9, 1 (2008), 424.
[34] Jonathan Johnson and Michael Wirthlin. 2009. Voter insertion techniques for fault tolerant FPGA design. (2009). https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1113&context=spacegrant&httpsredir=1&referer.
[35] Arthur B. Kahn. 1962. Topological sorting of large networks. *Commun. ACM* 5, 11 (1962), 558–562.
[36] Viggo Kann. 1992. *On the approximability of NP-complete optimization problems*. Ph.D. Dissertation. Royal Institute of Technology, Stockholm.
[37] Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (Eds.). Springer, 85–103.
[38] Claire Kenyon-Mathieu and Warren Schudy. 2007. How to rank with few errors. In *Proceedings of the 39th ACM Symposium on Theory of Computing (STOC'07)*. ACM, New York, NY, 95–103. DOI: https://doi.org/10.1145/1250790.1250806
[39] Subhash Khot and Oded Regev. 2008. Vertex cover might be hard to approximate to within $2-\varepsilon$. *J. Comput. Syst. Sci.* 74, 3 (2008), 335–349.

[40] Valerie King, Satish Rao, and Rorbert Tarjan. 1994. A faster deterministic maximum flow algorithm. *J. Alg.* 17, 3 (1994), 447–474.
[41] Arno Kunzmann and Hans-Joachim Wunderlich. 1990. An analytical approach to the partial scan problem. *J. Electron. Test.* 1, 2 (1990), 163–174.
[42] Jakub Łącki. 2013. Improved deterministic algorithms for decremental reachability and strongly connected components. *ACM Trans. Alg.* 9, 3 (June 2013), 1–15. DOI: https://doi.org/10.1145/2483699.2483707
[43] Charles E. Leiserson and James B. Saxe. 1991. Retiming synchronous circuitry. *Algorithmica* 6, 1–6 (1991), 5–35.
[44] C. L. Lucchesi and D. H. Younger. 1978. A minimax theorem for directed graphs. *J. London Math. Soc. s2*-17, 3 (1978), 369–374.
[45] Nikola T. Markov, Julien Vezoli, Pascal Chameau, Arnaud Falchier, René Quilodran, Cyril Huissoud, Camille Lamy, Pierre Misery, Pascale Giroud, Shimon Ullman, Pascal Barone, Colette Dehay, Kenneth Knoblauch, and Henry Kennedy. 2014. Anatomy of hierarchy: Feedforward and feedback pathways in macaque visual cortex. *J. Compar. Neurol.* 522, 1 (2014), 225–259. DOI: https://doi.org/10.1002/cne.23458
[46] R. Martí and G. Reinelt. 2011. *The Linear Ordering Problem: Exact and Heuristic Methods in Combinatorial Optimization.* Springer.
[47] James B. Orlin. 2013. Max flows in O(nm) time, or better. In *Proceedings of the 45th ACM Symposium on Theory of Computing*. ACM, 765–774.
[48] David Ratter. 2004. FPGAs on Mars. *Xcell J.* 50 (2004), 8–11.
[49] Youssef Saab. 2001. A fast and effective algorithm for the feedback arc set problem. *J. Heurist.* 7, 3 (2001), 235–250.
[50] H. C. Shih, Predrag G. Kovijanic, and Rahul Razdan. 1990. A global feedback detection algorithm for VLSI circuits. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors.* IEEE, 37–40.
[51] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. 2008. *Operating System Concepts* (8th ed.). Wiley Publishing.
[52] Michael Simpson, Venkatesh Srinivasan, and Alex Thomo. 2016. Efficient computation of feedback arc set at web-scale. *Proc. VLDB Endow.* 10, 3 (2016), 133–144.
[53] Robert Tarjan. 1972. Depth-first search and linear graph algorithms. *SIAM J. Comput.* 1, 2 (1972), 146–160.
[54] Robert Endre Tarjan. 1976. Edge-disjoint spanning trees and depth-first search. *Acta Inform.* 6, 2 (1976), 171–185.
[55] The Sage Developers. 2019. *SageMath, the Sage Mathematics Software System (Version 8.9).* https://www.sagemath.org
[56] Stephen H. Unger. 1957. A study of asynchronous logical feedback networks. Massachusetts Institute of Technology, Research Laboratory of Electronics.
[57] Duncan J. Watts and Steven H. Strogatz. 1998. Collective dynamics of "small-world" networks. *Nature* 393, 6684 (1998), 440–442. DOI: https://doi.org/10.1038/30918
[58] D. Younger. 1963. Minimum feedback arc sets for a directed graph. *IEEE Trans. Circ. Theor.* 10, 2 (1963), 238–245.

Received June 2020; revised January 2021; accepted January 2021