

Exact Algorithms for Maximum Acyclic Subgraph on a Superclass of Cubic Graphs

Henning Fernau & Daniel Raible

Univ.Trier, FB 4—Abteilung Informatik, 54286 Trier, Germany
{fernau,raible}@uni-trier.de

Abstract. Finding a maximum acyclic subgraph is on the list of problems that seem to be hard to tackle from a parameterized perspective. We develop two quite efficient algorithms (one is exact, the other parameterized) for $(1, n)$ -graphs, a class containing cubic graphs. The running times are $\mathcal{O}^*(1.1871^m)$ and $\mathcal{O}^*(1.212^k)$, respectively, determined by an amortized analysis via a non-standard measure.

1 Introduction and Definitions

Our problem. The FEEDBACK ARCSET PROBLEM FAS, is on the list of 21 problems that was presented by R.M. Karp [10] in 1972 exhibiting the first \mathcal{NP} -complete problems. It has numerous applications [7], ranging from program verification, VLSI and other network applications to graph drawing, where in particular the re-orientation of arcs in the first phase of the Sugiyama approach to hierarchical layered graph drawing is equivalent to FAS, see [2,17]. More formally, we consider the dual of FAS, namely the following problem:

MAXIMUM ACYCLIC SUBGRAPH MAS

Given a directed graph $G(V, A)$, and the parameter k .

We ask: Is there a subset $A' \subseteq A$, with $|A'| \geq k$, which is acyclic?

In this paper, we deal with finding exact and parameterized algorithms for MAS. Mostly, we focus on a class of graphs that, to our knowledge, has not been previously described in the literature. Let us call a directed graph $G = (V, E)$ $(1, n)$ -graph if, for each vertex $v \in V$, its indegree $d^+(v)$ obeys $d^+(v) \leq 1$ or its outdegree $d^-(v)$ satisfies $d^-(v) \leq 1$ (i.e., $\forall v \in V : \min\{d^+(v), d^-(v)\} \leq 1$). In particular, graphs of maximum degree three are $(1, n)$ -graphs. Notice that MAS, restricted to cubic graphs, is still \mathcal{NP} -complete. For some applications from graph drawing (e.g., laying out “binary decision diagrams” where vertices correspond to yes/no decisions) even the latter restriction is not so severe at all. Having a closer look at the famous paper of I. Nassi and B. Shneiderman [13] where they introduce structograms to aid structured programming (and restricting the use of GOTOs), it can be seen that the resulting class of flowchart graphs is that of $(1, n)$ -graphs.

Cubic graphs also have been discussed in relation to approximation algorithms: A. Newman [14] showed a factor $\frac{12}{11}$ -approximation. Having a closer look at her

algorithm reveals that it also works for $(1, n)$ -graphs with the same approximation factor. This largely improves on the general situation, where only a factor of 2 is known [2]. We point out that finding a minimum feedback arc set (in general graphs) is known to possess a factor $\log n \log \log n$ -approximation, see [7], and hence shows an approximability behavior much worse than MAS.

Our framework: Parameterized Complexity. A *parameterized problem* P is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a fixed alphabet and \mathbb{N} is the set of all non-negative integers. Therefore, each instance of the parameterized problem P is a pair (I, k) , where the second component k is called the *parameter*. The language $L(P)$ is the set of all YES-instances of P . We say that the parameterized problem P is *fixed-parameter tractable* [5] if there is an algorithm that decides whether an input (I, k) is a member of $L(P)$ in time $f(k)|I|^c$, where c is a fixed constant and $f(k)$ is a function independent of the overall input length $|I|$. We will also write $\mathcal{O}^*(f(k))$ for this run-time bound. Equivalently, one can define the class of fixed-parameter tractable problems as follows: strive to find a polynomial-time transformation that, given an instance (I, k) , produces another instance (I', k') of the same problem, where $|I'|$ and k' are bounded by some function $g(k)$; in this case, (I', k') is also called a (*problem*) *kernel*.

Discussion of related results. MAS on general directed graphs can be solved in time $\mathcal{O}^*(2^k)$ and $\mathcal{O}^*(2^n)$, shown by V. Raman and S. Saurabh in [15], with n the number of vertices. Most recently, parallel to our work, I. Razgon, J. Chen *et al.* showed during a workshop [12] that $\text{FAS} \in \mathcal{FPT}$, relying on results from J. Chen *et al.* [4]. In contrast to MAS, it still admits a fairly vast run time of $\mathcal{O}^*(8^k k!)$. Likewise, I. Razgon [16] provided an exact (non-parameterized) $\mathcal{O}^*(1.9977^n)$ -algorithm for FEEDBACK VERTEX SET (FVS), which translates to a FAS-algorithm with the same base, but measured in m .

The complexity picture changes when one considers undirected graphs. The task of removing a minimum number of edges to obtain an acyclic graph can be accomplished in polynomial time, basically by finding a spanning forest. The task of removing a minimum number of vertices to obtain an acyclic graph is (again) \mathcal{NP} -complete, but can be approximated to a factor of two, see V. Bafna *et al.* [1], and is known to be solvable in $\mathcal{O}^*(5^k)$ with J. Chen *et al.* [3] being the currently leading party in a run time race. Also, exact algorithms have been derived for this problem by F. V. Fomin *et al.* [8].

Our contributions. Our main technical contribution is to derive a parameterized $\mathcal{O}^*(1.212^k)$ -algorithm for MAS on $(1, n)$ -graphs. On cubic graphs the run time reduces to $\mathcal{O}^*(1.1960^k)$ via a novel combinatorial observation. We also derive an exact algorithm for MAS on $(1, n)$ -graphs and as a by-products two other for DIRECTED FEEDBACK VERTEX SET on cubic and planar graphs with running times $\mathcal{O}^*(1.1871^m)$, $\mathcal{O}^*(1.282^n)$ and $\mathcal{O}^*(1.986^n)$, respectively. Besides being a nice combinatorial problem on its own right, we think that our contribution is also interesting from the more general perspective of a development of tools for constructing efficient parameterized algorithms. Namely, the algorithm we present is of a quite simple overall structure, similar in simplicity as, e.g., the recently presented algorithms for HITTING SET [6]. But the analysis is

quite intricate and seems to offer a novel way of amortized search tree analysis that might be applicable in other situations in parameterized algorithmics, as well. It is also one of the fairly rare applications of the “measure & conquer” paradigm [9] in parameterized algorithmics. Due to lack of space, some proofs had to be omitted.

Fixing terminology. We consider directed multigraphs $G(V, A)$ in the course of our algorithm, where V is the vertex set and A the arc set. From A to V we have two kinds of mappings: For $a \in A$, $init(a)$ denotes the vertex at the tip of the arc a and $ter(a)$ the end. We distinguish between two kinds of arc-neighborhoods of a vertex v which are $E^+(v) := \{a \in A \mid ter(a) = v\}$ and $E^-(v) := \{a \in A \mid init(a) = v\}$. We have an in- and outdegree of a vertex, that is $d^+(v) := |E^+(v)|$ and $d^-(v) := |E^-(v)|$. We set $E(v) := E^+(v) \cup E^-(v)$ and $d(v) := |E(v)|$ called the degree of v . We also define a neighborhood for arcs a $N_A(a) := \{a_1, a_2 \in A \mid ter(a_1) = init(a), ter(a) = init(a_2)\}$ and for $A' \subseteq A$ we set $N_A(A') := \bigcup_{a' \in A'} N_A(a')$. For $V' \subseteq V$ we set $A(V') := \{a \in A \mid \exists u, v \in V', init(a) = u, ter(a) = v\}$. We call an arc (u, v) a *fork* if $d^-(v) \geq 2$ (but $d^+(v) = 1$) and a *join* if $d^+(u) \geq 2$ (but $d^-(u) = 1$). With \mathcal{MAS} , we refer to a set of arcs, which is acyclic and is a partial solution. An *undirected cycle* is an acyclic arc set, which is a cycle in the underlying undirected graph.

Kernels via Approximation We first link approximability and parameterized algorithmics by a simple but interesting observation. We call a maximization problem P a *set maximization problem* if its task, given instance I , is to identify a subset S (satisfying additional requirements) of a *ground set* M of maximum cardinality. The natural parameterized problem related to P (denoted by P_{par}) is to find, given (I, k) such a subset S of cardinality at least k .

Proposition 1. *If a set maximization problem P has a c -approximation, where the ratio is measured with respect to the whole ground set M that is part of the input I whose size is measured in terms of the cardinality $|M|$ of M , then P_{par} , parameterized by k , has a kernel of size upper-bounded by ck .*

Both above mentioned approximations exhibit the required properties of Proposition 1. This entails a $2k$ -kernel for the general case of \mathcal{MAS} , as well as a $\frac{12}{11}k$ -kernel when restricted to $(1, n)$ -graphs, based on A. Newman’s result [14].

2 Preprocessing & Reduction Rules

Firstly, we can assume that our instance $G(V, A)$ forms a strongly connected component. Every arc not in such a component can be taken into a solution, and two solutions of two such components can be simply joined.

Preprocessing. In [7,14] a set of preprocessing rules is already mentioned:

Pre-1: For every $v \in V$ with $d^+(v) = 0$ or $d^-(v) = 0$, delete v and $E(v)$, take $E(v)$ into \mathcal{MAS} and decrement k by $|E(v)|$.

Pre-2: For every $v \in V$ with $E(v) = \{(i, v), (v, o)\}$, $v \neq i$ and $v \neq o$, delete v and $E(v)$ and introduce a new arc (i, o) . Decrease k by one.

Pre-3: Remove any loop.

Any preprocessing rule, which applies, will be carried out exhaustively. Afterwards the resulting graph has no vertices of degree less than three.

Definition 1. *An arc g is an α -arc if it is a fork and a join.*

We need the next lemma, which is a sharpened version of [14, Lemma 2.1] and follows the same lines of reasoning.

Lemma 1. *Any two non-arc-disjoint cycles in a $(1, n)$ -graph with minimum degree at least 3 share an α -arc.*

We partition A in A_α containing all α -arcs and $A_{\bar{\alpha}} := A \setminus A_\alpha$. By Lemma 1, the cycles in $G[A_{\bar{\alpha}}]$ must be arc-disjoint. This justifies the next preprocessing rule.

Pre-4 In G delete the arc set of every cycle C contained in $G[A_{\bar{\alpha}}]$. For an arbitrary $a \in C$ adjoin $C \setminus \{a\}$ to \mathcal{MAS} and decrease k by $|C| - 1$.

After exhaustively applying Preprocess() (shown in Figure 1), every cycle has an α -arc. For $v \in V$ with $E^+(v) = \{a_1, \dots, a_s\}$ ($E^+(v) = \{c\}$, resp.) and $E^-(v) = \{c\}$ ($E^-(v) = \{a_1, \dots, a_s\}$), it is always better to delete c than one of a_1, \dots, a_s . Therefore, we adjoin a_1, \dots, a_s to \mathcal{MAS} , adjusting k accordingly. Having applied this rule on every vertex, we adjoined $A_{\bar{\alpha}}$ to \mathcal{MAS} , and the remaining arcs are exactly A_α .

So, the next task is to find $S \subseteq A_\alpha$ with $|\mathcal{MAS} \cup S| \geq k$ so that $G[\mathcal{MAS} \cup S]$ is acyclic. We have to branch on the α -arcs, deciding whether we take them into \mathcal{MAS} or if we delete them. α -arcs which we take into \mathcal{MAS} will be called *red*. For the purpose of measuring the complexity of the algorithm, we will deal with two parameters k and k' , where k measures the size of the partial solution and k' will be used for purposes of run-time estimation: We do not account the arcs in $A_{\bar{\alpha}}$ immediately into k' . For every branching on an α -arc, we count only a portion of them into k' . More precisely, upon first seeing an arc $b \in A_{\bar{\alpha}}$ within the neighborhood $N_A(g)$ of an α -arc g we branch on, we will count b only by an amount of ω , where $0 < \omega < 0.5$ will be determined later. So, we will have two weighting functions w_k and $w_{k'}$ for k and k' with $w_k(a) \in \{0, 1\}$ and $w_{k'}(a) \in \{0, (1 - \omega), 1\}$ for $a \in A$, indicating each how much of the arc has not been counted into k , or k' respectively, yet. In the course of the algorithm, we always have $w_k(a) \leq w_{k'}(a)$. In the very beginning, we have $w_k(a) = w_{k'}(a) = 1$ for all $a \in A$. For a set $A' \subseteq A$, we define $w_{k'}(A') := \sum_{a' \in A'} w_{k'}(a')$ and $w_k(A')$ accordingly. Observe that for $a \in A$ we have $a \in \mathcal{MAS}$ iff $w_k(a) = 0$.

The preprocessing rules, together with the mentioned kernel of $\frac{12}{11}k$ arcs, gives us another simple brute-force algorithm for MAS: Within the kernel with m arcs, there could be at most $m/3$ arcs that are α -arcs. It is obviously sufficient to test all possible $2^{m/3} \leq 2^{(4/11)k} \approx 1.288^k$ many possibilities of choosing α -arcs into the (potential) feedback arc set.

Reduction Rules. There is a set of reduction rules from [14], which we adapted and modified to deal with weighted arcs. Also, we define a (linear time checkable)

<hr/> Procedure: Preprocess($\mathcal{MAS}, G(V, A), k$): 1: repeat 2: cont \leftarrow false 3: apply Pre-1 - Pre-3 exhaustively. 4: if Pre-4 applies then 5: cont \leftarrow true 6: until cont= false 7: return ($\mathcal{MAS}, G(V, A), k, 1$)	<hr/> Procedure: Reduce($\mathcal{MAS}, G(V, A), k, k', w_k, w_{k'}$): 1: repeat 2: cont \leftarrow false 3: for i=1 to 6 do 4: apply RR-i exhaustively. 5: if RR-i applied then 6: cont \leftarrow true 7: until cont= false 8: return ($\mathcal{MAS}, G(V, A), k, k', w_k, w_{k'}$)
--	--

Fig. 1. The procedures Preprocess() and Reduce().

predicate *contractible* for all $a \in A$.

$$contractible(a) = \begin{cases} 0 & : w_k(a) = 1, \exists \text{ cycle } C \text{ with } a \in C \text{ and } w_k(C \setminus \{a\}) = 0 \\ 1 & : \text{else} \end{cases}$$

The meaning of this predicate is the following: if $contractible(a) = 0$, then a is the only remaining arc of some cycle, which is not already determined to be put into \mathcal{MAS} . Thus, a has to be deleted. In the following, **RR-(i-1)** is always carried out exhaustively before **RR-i**.

RR-1 For $v \in V$ with $d^+(v) = 0$ or $d^-(v) = 0$, take $E(v)$ into \mathcal{MAS} , delete v and $E(v)$ and decrease k by $w_k(E(v))$ and k' by $w_{k'}(E(v))$.

RR-2 For $v \in V$ with $E(v) = \{a, b\}$ let $z = \arg \max\{w_{k'}(a), w_{k'}(b)\}$ and $y \in E(v) \setminus \{z\}$. If $contractible(y) = 1$, then contract y , decrement k by $w_k(y)$, k' by $w_{k'}(y)$. If y was red, then z becomes red.

RR-3 If for $g \in A$, we have $contractible(g) = 0$, then delete g .

We point out that due to **RR-2** also non- α -arcs can become red. But it is still true for a α -arc a that $a \in \mathcal{MAS}$ iff a is red. Let $A_\alpha^U := \{a \in A_\alpha \mid a \text{ is non-red}\}$. We classify the arcs of A_α^U in *thin* α -arcs, which are contained in exactly one cycle, and *thick* α -arcs, which are contained in at least two cycles. Because G is strongly connected, there are no other α -arcs. We can distinguish them as follows: For every α -arc g , find the smallest cycle C_g which contains g via BFS. If g is contained in a second cycle C'_g , then there is an arc $a \in C_g$ with $a \notin C'_g$. So for all $a \in C_g$, remove a and restart BFS, possibly finding a second cycle.

RR-4 If $g \in A_\alpha^U$ is thin and $contractible(g) = 1$, then take g into \mathcal{MAS} and decrease k by $w_k(g)$, k' by $w_{k'}(g)$ and set $w_k(g) \leftarrow 0$, $w_{k'}(g) \leftarrow 0$.

RR-5 If $a, b \in A$ form an undirected 2-cycle then let $z = \arg \min\{w_{k'}(a), w_{k'}(b)\}$, decrease k by $w_k(z)$, k' by $w_{k'}(z)$, take z into \mathcal{MAS} and delete z .

RR-6 Having $(u, v), (v, w), (u, w) \in A$ (an undirected 3-cycle), decrease k by $w_k((u, w))$, k' by $w_{k'}((u, w))$, take (u, w) into \mathcal{MAS} and delete (u, w) .

Lemma 2. a) *The reduction rules are sound.* b) *After the application of Reduce(), see Figure 1, we are left with a $(1, n)$ -graph with only thick α -arcs and no directed or undirected 2- or 3-cycle.*

Algorithm 1 A parameterized algorithm for MAXIMUM ACYCLIC SUBGRAPH on $(1, n)$ -graphs

```

1:  $(\mathcal{MAS}, G(V, A), k, w_k) \leftarrow \text{Preprocess}(\emptyset, G(V, A), k)$ .
2:  $\mathcal{MAS} \leftarrow A_{\bar{\alpha}} \cup \mathcal{MAS}, k' \leftarrow k, k \leftarrow k - w_k(A_{\bar{\alpha}}), w_k(A_{\bar{\alpha}}) \leftarrow 0$ 
3:  $\text{Sol3MAS}(\mathcal{MAS}, G(V, A), k, k', w_{k'}, w_k)$ 
Procedure:  $\text{Sol3MAS}(\mathcal{MAS}, G(V, A), k, k', w_k, w_{k'})$ :
1:  $(\mathcal{MAS}, G(V, A), k, k', w_k, w_{k'}) \leftarrow \text{Reduce}(\mathcal{MAS}, G(V, A), k, k', w_k, w_{k'})$ 
2: if  $k \leq 0$  then
3:   return YES
4: else if there is a component  $C$  with at most 9 arcs then
5:   Test all possible solutions for  $C$ .
6: else if there is a  $\alpha$ -arc  $g \in A_{\alpha}^U$  then
7:   if not  $\text{Sol3MAS}(\mathcal{MAS}, G[A \setminus \{g\}], k, k', w_k, w_{k'})$  then
8:      $k \leftarrow k - 1, k' \leftarrow k' - w_{k'}(g), w_k(g) \leftarrow w_{k'}(g) \leftarrow 0, \mathcal{MAS} \leftarrow \mathcal{MAS} \cup N_A(g) \cup \{g\}$ .
9:     for all  $a \in N_A(g)$  do
10:      Adjust  $w_{k'}$ , see Figure 2.
11:     return  $\text{Sol3MAS}(\mathcal{MAS}, G(V, A), k, k', w_k, w_{k'})$ 
12:   else
13:     return YES
14: else
15:   return NO

```

Adjust $w_{k'}$:

```

1: if  $w_{k'}(a) = 1$  then
2:   if  $\exists b \in (N_A(a) \setminus (N_A(g) \cup \{g\}))$  with  $w_{k'}(b) = 0$  then
3:      $k' \leftarrow k' - 1, w_{k'}(a) \leftarrow 0,$ 
        $k \leftarrow k - w_k(a), w_k(a) \leftarrow 0$  (case a.)
4:   else
5:      $k' \leftarrow k' - \omega, w_{k'}(a) \leftarrow (1 - \omega),$ 
        $k \leftarrow k - w_k(a), w_k(a) \leftarrow 0$  (case b.)
6:   else
7:      $k' \leftarrow k' - (1 - \omega), w_{k'}(a) \leftarrow 0.$  (case c.)

```

Fig. 2. In case a . we set $w_{k'}(a) = 0$, because there will not be any other neighboring non-red α -arc of a . In case b ., this might not be the case, so we count only a portion of ω . In case c ., we will prove that $w_{k'}(a) = (1 - \omega)$ and that there will be no other non-red neighboring α -arc of a , see Theorem 1.5.

3 The Algorithm and its Analysis

3.1 The Algorithm

We are ready now to state our main Algorithm 1; observe that the handling of the second parameter k' is only needed for the run-time analysis and could be avoided when implementing the algorithm. Therefore, the branching structure of the algorithm is quite simple, as expressed in the following:

Lemma 3. *Branching in Alg. 1 either puts a selected α -arc g into \mathcal{MAS} , or it deletes g . Only if arcs are deleted, reduction rules will be triggered in the subsequent recursive call. This can be also due to triggering **RR-3** after putting a into \mathcal{MAS} .*

3.2 Analysis

Basic Combinatorial Observations. While running the algorithm, $k \leq k'$. Now, substitute in line 2 of Sol3MAS of Algorithm 1 k by k' . If we run the algorithm, it will create a search tree $T_{k'}$. The search tree T_k of the original algorithm must be contained in $T_{k'}$, because $k \leq k'$. If $|T_{k'}| \leq c^{k'}$, then it follows that also $|T_k| \leq c^{k'} = c^k$, because in the very beginning, $k = k'$. So in the following, we will state the different recurrences derived from Algorithm 1 in terms of k' . For a good estimate, we have to calculate an optimal value for ω .

Theorem 1. *In every node of the search tree, after applying Reduce(), we have*

1. For all $a = (u, v) \in A_{\bar{\alpha}}$ with $w_{k'}(a) = (1 - \omega)$, there exists a red fork (u', u) or a red join (v, v') .
2. For all non-red $a = (u, v) \in A_{\bar{\alpha}}$ with $w_{k'}(a) = 0$, we find a red fork (u', u) and a red join (v, v') . We will also say that a is protected (by the red arcs).
3. For all red arcs $d = (u, v)$ with $w_{k'}(d) = 0$, if we have only non-red arcs in $E(u) \setminus \{d\}$ ($E(v) \setminus \{d\}$, resp.), then d is a join (d is a fork, resp.).
4. For each red arc $d = (u, v)$ with $w_{k'}(d) = 0$ that is not a join (fork, resp.), if there is at least one red arc in $E(u) \setminus \{d\}$ (in $E(v) \setminus \{d\}$, resp.), then there is a red fork (red join, resp.) in $G[E(u)]$ ($G[E(v)]$, resp.).
5. For all $g \in A_{\bar{\alpha}}^U$ and for all $a \in N_A(g)$, we have: $w_{k'}(a) > 0$.

Proof. We use induction on the depth of the search tree. Clearly, all claims are trivially true for the original graph instance, i.e., the root node. Notice that each claim has the form $\forall a \in A : X(a) \implies Y(a)$. Here, X and Y express local situations affecting a . Therefore, we have to analyze how $X(a)$ could have been created by branching. According to Lemma 3, we have to discuss what happens (1) if a certain α -arc had been put into \mathcal{MAS} and (2) if reduction rules were triggered. As a third point, we must consider the possibility that $X(a)$ is true both in the currently observed search tree node s and in its predecessor, but that $Y(a)$ was possibly affected upon entering s .

1. (*sketch*) Such an arc (u, v) can be created in two ways: (1) By taking a neighboring α -arc g into \mathcal{MAS} . Then g is a join and a fork. (2) By applying **RR-2** on arcs $(u, z), (z, v)$. W.l.o.g. $w_{k'}((u, z)) = (1 - \omega)$. Then by induction there is a fork (u', u) . How can this situation be affected by the reduction rules? Any **RR-2**-application leaves it unchanged as red arcs are 'dominant'. If w.l.o.g. (u', u) is a red fork then this situation may be changed by deletions in the case where $d(u)$ becomes two. However, (u, v) disappears by a subsequent **RR-2**-application.
2. We will actually prove points 2. through 4. by a parallel induction. To improve readability of our main argument, we refrain from giving all possible details how the employment of **RR-2** may affect (but not drastically change) the situation in particular. How can $a = (u, v) \in A_{\bar{\alpha}}$ with $w_{k'}(a) = 0$ have been created? Firstly, it could be due to a **RR2**-contraction with a non-red arc t with $w_{k'}(t) = 0$. But then a was not protected, which is a contradiction to the induction hypothesis. Secondly, it could be due to branching on a

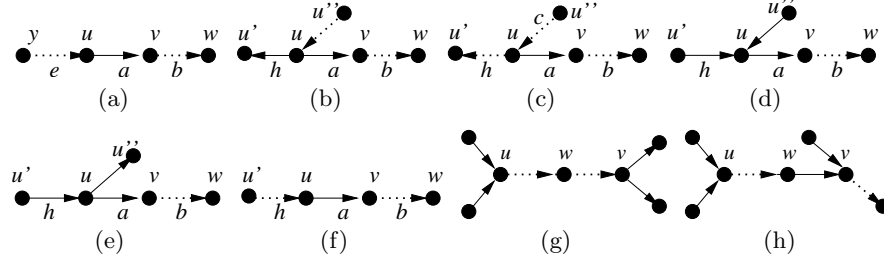


Fig. 3. Dotted lines indicate red arcs.

neighboring α -arc b , say $b = (v, w)$ with b a join, in two different ways:

- (1) either we branched at b at a point of time when $w_{k'}(a) = (1 - \omega)$ (case c . of Procedure “Adjust”), or (2) we branched at b when $w_{k'}(a) = 1$ (case a .)

In case (1), there must have been another red arc e incident to a by item 1 of our property list, see Figure 3(a). e is not incident to v , since it is a fork. Hence, $e = (y, u)$. This displays the two required red arcs (namely b and e) in this case. In case (2), a was created by case a . of Procedure “Adjust.” Obviously, b is red after branching. Since we have branched according to case a ., there is another arc h incident with a (but not with b) such that $w_{k'}(h) = 0$. There are four subcases to be considered:

- (a) $h = (u, u')$ is not red, see Figure 3(b). By induction (item 2.), there must be a red fork arc (u'', u) . Hence, a is protected.

- (b) $h = (u, u')$ is red, see Figure 3(c). Consider all other arcs incident to u . Since we are dealing with reduced instances and by the $(1, n)$ -property, there must be exactly one of the form $c = (u'', u)$, since otherwise u would be a sink. Suppose h is the only red arc in $E(u)$. Then this contradicts item 3. Now, suppose c is not red. Then there is a red arc (u, \bar{u}) . By item 4. c must be also red, a contradiction, Therefore, c is the red fork, which protects a .

- (c) $h = (u', u)$ is not red. All other arcs incident to u could be of the form (u'', u) , see Figure 3(d). Since h must be protected, by induction, a should be red, contradicting our assumption on a . Thus, all these arcs are of the form (u, u'') , see Figure 3(e). This contradicts item 2., since there is no red join protecting h .

- (d) $h = (u', u)$ is red, see Figure 3(f). Suppose h is not a fork. Then all other arcs beside a and h are of the form (\bar{u}, u) . If none of them is red we have a contradiction concerning item 3. If one of them is red then by item 4. a is red, which contradicts our assumptions. Hence, a will be protected.

3. How could d have been created? If it had been created by branching, then there are two cases: (1) d was put into \mathcal{MAS} ; (2) d was neighbor of an arc b which we put into \mathcal{MAS} .

In case (1), the claim is obviously true. In case (2), let, w.l.o.g., u be the common neighbor of b and d . After putting b into \mathcal{MAS} , there will be a red arc (namely b), incident to u , so that there could be only non-red arcs

incident with v that have the claimed property by induction. If d has been created by reduction rules, it must have been through **RR-2**. So, there have been (w.l.o.g.) two arcs (u, w) and (w, v) with $w_{k'}$ -weights zero. One of them must be red. W.l.o.g., assume that (u, w) is red. If (w, v) is red, see Figure 3(g), then the claim holds by induction. If (w, v) is not red, see Figure 3(h), then (w, v) must be protected due to item 2. Hence, the premise is falsified for vertex v .

4. We again discuss the possibilities that may create a red d with $w_{k'}(d) = 0$. If d was created by taking it into \mathcal{MAS} during branching, then d would be both fork and join in contrast to our assumptions. If we branch in the neighborhood of d , then the claim could be easily verified directly. Finally, d could be obtained from merging two arcs $e = (u, w)$, $f = (w, v)$ with $w_{k'}(e) = w_{k'}(f) = 0$. If both e and f are red, the claim follows by induction. If only f is red and e is non-red, then there is a red fork, which protects e by item 2. Again, by induction the claim follows. The case where only e is red is symmetric.
5. Assume the contrary. Discuss a neighbor arc a of g with $w_{k'}(a) = 0$. If a is not red, then g must be red due to item 2., contradicting $g \in A_\alpha^U$. If a is red, then discuss another arc b that is incident to the common endpoint of a and g . If there is no red b , then the situation contradicts item 3. So, there is a red b . This picture contradicts item 4. \square

Estimating the running time for maximum degree 3 graphs. In Algorithm 1, depending in which case of Figure 2 we end up, we decrement k' by a different amount for each arc $a \in N_A(g)$ in the case that we put g into \mathcal{MAS} . We can be sure that we may decrement k' by at least $(1 - \omega)$ for each neighbor $a \in N_A(g)$ due to the last property of the Theorem 1. If we do not put g into \mathcal{MAS} , we delete g and $N_A(g)$ immediately afterwards by **RR-1**, decrementing k' accordingly (by $w_{k'}(N_A(g))$). Moreover, if case *b.* applies to $a \in N_A(g)$, we know that the two arcs $d, e \in (N_A(a) \setminus (N_A(g) \cup g))$ obey $w_{k'}(d)w_{k'}(e) > 0$ (observe that we do not have triangles). By deleting a , no matter whether **RR-1** or **RR-2** applies to d and e (this depends on the direction of the arcs) we can decrement k' by an extra amount of at least $(1 - \omega)$, cf. the handling of k' by these reduction rules. This is true even if $V(d), V(e) \subset V(N_A(g))$. Note that if $V(N_A(N_A(g))) \subseteq V(N_A(g))$, then $A(V(N_A(g)))$ is a component of 9 arcs (which are handled separately). Let i denote the number of arcs $a \in N_A(g)$ for which case *a.* applies. In the analogous sense j stands for the case *b.* and q for *c.* For every positive integer solution of $i + j + q = 4$, we can state a total of 15 recursions T_1, \dots, T_{15} according to Table 1 depending on ω (ignoring the last column for the moment). For every T_i and for a fixed ω , we can calculate a constant $c_i(\omega)$ such that $T_i[k] \in \mathcal{O}^*(c_i(\omega)^k)$. We want to find a ω with subject to minimize $\max\{c_1(\omega), \dots, c_{15}(\omega)\}$. We numerically obtained $\omega = 0.1687$ so that $\max\{c_1(\omega), \dots, c_{15}(\omega)\}$ evaluates to 1.201. The dominating cases are when $i = 0, j = 0, q = 4$ (T_5) and $i = 0, j = 4, q = 0$ (T_{15}). We conclude that \mathcal{MAS} on graphs G with $\Delta(G) \leq 3$ can be solved in $\mathcal{O}^*(1.201^k)$. Measuring the run time in terms of $m := |A|$ the same way is also possible. Observe that if we delete an α -

α -arc g	$a.$	$b.$	$c.$	$b'.$
MAS	1	ω	$(1 - \omega)$	ω
Deletion	1	$(2 - \omega)$	$(1 - \omega)$	1

Table 1. Summarizes by which amount k' can be decreased for $a \in N_A(g)$, subject to if we take g into MAS or delete g and to the case applying to a .

arc, we can decrement m by one more. By adjusting T_1, \dots, T_{15} according to this and by choosing $\omega = 0.2016$, we derive an upper bound of $\mathcal{O}^*(1.1798^m)$. Note that if we run this exact algorithm on the $\frac{12}{11}k$ -kernel, we finish in $\mathcal{O}^*(1.1977^k)$, being slightly better than the pure parameterized algorithm.

We will obtain a better bound for the search tree by a precedence rule, aiming to improve recurrence T_5 . If we branch on an α -arc g according to this recurrence, for all $a \in N_A(g)$ we have $w_{k'}(a) \geq (1 - \omega)$. Such α -arcs will be called α_5 -arcs. We add the following rule: branch on α_5 -arcs with least priority. Let $l := |A_\alpha^U|$.

Lemma 4. *Branching on an α_5 -arc, we can assume: $\lfloor \frac{1}{5-4\omega} k' \rfloor \leq l < \lceil \frac{1}{4-4\omega} k' \rceil$.*

Employing this lemma, we can find a good combinatorial estimate for a brute-force search at the end of the algorithm. This allows us to conclude:

Theorem 2. *MAS is solvable in time $\mathcal{O}^*(1.1798^m)$ and $\mathcal{O}^*(1.1960^k)$ on maximum-degree-3-graphs.*

Corollary 1. *FEEDBACK VERTEX SET on cubic graphs is solvable in $\mathcal{O}^*(1.282^n)$.*

Proof. We argue that MAS and FAS are equivalent for graphs of degree at most three as follows. Namely, if A is a feedback arc set, then we can remove instead the set S of vertices the arcs in A are pointing to in order to obtain a directed feedback vertex set with $|S| \leq |A|$. Conversely, if S is a directed feedback vertex set, then we can assume that each vertex $v \in S$ has one ingoing and two outgoing arcs or two ingoing and one outgoing arc; in the first case, let a_v be the ingoing arc, and in the second case, let a_v be the outgoing arc. Then, $A = \{a_v \mid v \in S\}$ is a feedback arc set with $|A| \leq |S|$. With $m \leq \frac{3}{2}n$ the claim follows. \square

Estimating the running time for $(1, n)$ -graphs. There is a difference to maximum degree 3 graphs, namely the entry for case $b.$ in case of deletion in Table 1. For $a \in N_A(g)$ it might be the case that $|N_A(a) \setminus (N_A(g) \cup \{g\})| \geq 3$, so that when we delete g and afterwards a by **RR-1** that whether **RR-1** nor **RR-2** applies (due to the lack of sources, sinks or degree two vertices). We call this case $b'.$ Remember, case $b.$ refers to the same setting but with $|N_A(a) \setminus (N_A(g) \cup \{g\})| = 2$. Thus the mentioned entry should be 1 for $b'.$ As long as $|N_A(g)| \geq 6$ the reduction in k' is great enough for the modified table, but for the other cases we must argue more detailed. We introduce two more reduction rules, the first already mentioned in [14], and add items **a)-c)** to Algorithm 1.

RR-7 Contract and adjoin to \mathcal{MAS} any $(u, v) \in A$ with $d^+(u) = d^+(v) = 1$ ($d^-(u) = d^-(v) = 1$, resp.). If (u, v) was red the unique arc $a := (x, u)$ ((v, y) , resp.) will be red. Decrease k' by $\min\{w_{k'}((u, v)), w_{k'}(a)\}$ and set $w_{k'}(a) \leftarrow \max\{w_{k'}((u, v)), w_{k'}(a)\}$. Proceed similarly with (v, y) .

RR-8 For a red $g' \in A_\alpha$ with $w_{k'}(g') > 0$, set $k' \leftarrow k' - w_{k'}(g')$ and $w_{k'}(g') \leftarrow 0$.

a) After Reduce(), first apply **RR-7** and then **RR-8** exhaustively.

b) Prefer α -arcs g such that $|N_A(g)|$ is maximal for branching.

c) Forced to branch on $g \in A_\alpha^U$ with $|N_A(g)| = 5$, choose an α -arc with the least occurrences of case b' .

Lemma 5. **RR-7** and **RR-8** are sound and do not violate Theorem 1.

Lemma 6. We can omit branching on arcs $g \in A_\alpha^U$ with a) $|N_A(g)| = 5$ and 5 occurrences of case b' or b) $|N_A(g)| = 4$ with an occurrence of case b' .

Let x, y, z denote the occurrences of cases a, b' and c . To upperbound the branchings according to α -arcs g with $|N_A(g)| \geq 6$, we put up all recurrences resulting from integer solutions of $x + y + z = 6$. Note that we also use the right column of Table 1. To upperbound branchings with $|N_A(g)| = 5$ we put up all recurrences obtained from integer solutions of $x + y + z = 5$, except when $x = z = 0$ and $y = 5$ due to Lemma 6.a). Additionally we have to cover the case where we have 4 occurrences of case b' and one of case b . ($T[k] \leq T[k - 5\omega] + T[k - (6 - \omega)]$). To upperbound the case where $|N_A(g)| = 4$ the recurrences derived from Table 1 for the integer solutions of $x + y + z = 4$ suffice due to Lemma 6.b).

Theorem 3. On $(1, n)$ -graphs with m arcs, \mathcal{MAS} is solvable in time $\mathcal{O}^*(1.1871^m)$ ($\omega = 0.2392$) and $\mathcal{O}^*(1.212^k)$ ($\omega = 0.21689$), respectively.

Corollary 2. We solve FEEDBACK VERTEX SET on planar graphs in $\mathcal{O}^*(1.986^n)$.

Proof. Reduce [7] FVS to FAS. The resulting graph G' has no more than $4n$ arcs. Then run Algorithm 1 on G' . \square

4 Reparameterization

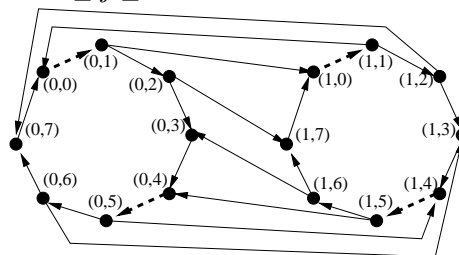
M. Mahajan, V. Raman and S. Sidkar [11] have discussed a rather general setup for re-parameterization of problems according to a “guaranteed value.” In order to use their framework, we only need to exhibit a family of example graphs where Newman’s approximation bound for \mathcal{MAS} is sharp. Consider $G_r(V_r, A_r)$, $r \geq 2$, with $V_r = \{(i, j) \mid 0 \leq i < r, 0 \leq j \leq 7\}$, and A_r contains two types of arcs:

1. $((i, j), (i, (j + 1) \bmod 8))$ for $0 \leq i < r$ and $0 \leq j \leq 7$.

2a. $((i, j), ((i + 1) \bmod r, (1 - j) \bmod 8))$ for $0 \leq i < r$ and $j = 1, 2$.

2b. $((i, j), (((i + 1) \bmod r, (1 - j) \bmod 8), (i, j)))$ for $0 \leq i < r$ and $j = 3, 4$.

For $r = 2$ we find an example to the right. G_r is cubic with $|V_r| = 8r$ and $|A_r| = 12r$. Its α -arcs are $((i, 0), (i, 1))$ and $((i, 4), (i, 5))$ for $0 \leq i < r$.



Since we have to destroy all ‘rings’ as described by the arcs from 1., any feasible solution to these instances require r arcs to go into the feedback arc set. Also r arcs suffice, namely $((0, 4), (0, 5))$ and $((i, 0), (i, 1))$ for $0 < i < r$, giving the ‘tight example’ as required in [11] to conclude:

Corollary 3. *For any $\epsilon > 0$, the following question is not fixed-parameter tractable unless $\mathcal{P} = \mathcal{NP}$: Given a cubic directed graph $G(V, E)$ and a parameter k , does G possess an acyclic subgraph with at least $(\frac{11}{12} + \epsilon) |E| + k$ many arcs ?*

References

1. V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal of Discrete Mathematics*, 12:289–297, 1999.
2. B. Berger and P. W. Shor. Approximation algorithms for the maximum acyclic subgraph problem. In *SODA*, pages 236–243, 1990.
3. J. Chen, F. V. Fomin, Y. Liu, S. Lu, and Y. Villanger. Improved algorithms for the feedback vertex set problems. In *WADS*, pages 422–433, 2007.
4. J. Chen, Y. Liu, and S. Lu. An improved parameterized algorithm for the minimum node multiway cut problem. In *WADS*, pages 495–506, 2007.
5. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
6. H. Fernau. Parameterized algorithms for HITTING SET: the weighted case. In *CIAC*, volume 3998 of *LNCS*, pages 332–343. Springer, 2006.
7. P. Festa, P. M. Pardalos, and M. G. C. Resende. Feedback set problems. In *Handbook of Combinatorial Optimization*, volume Supplement Volume A, pages 209–258. Kluwer Academic Publishers, 1999.
8. F. V. Fomin, S. Gaspers, and A. V. Pyatkin. Finding a minimum feedback set in time $\mathcal{O}(1.7548^n)$. In *IWPEC*, volume 4169 of *LNCS*, pages 184–191. Springer, 2006.
9. F. V. Fomin, F. Grandoni, and D. Kratsch. Measure and conquer: domination – a case study. In *ICALP*, volume 3580 of *LNCS*, pages 191–203. Springer, 2005.
10. R.M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Plenum Press, 1972.
11. M. Mahajan, V. Raman, and S. Sikdar. Parameterizing MAXNP problems above guaranteed values. In *IWPEC*, volume 4169 of *LNCS*, pages 38–49. Springer, 2006.
12. *Seminar 07281: Structure Theory and FPT Algorithmics for Graphs, Digraphs and Hypergraphs, Dagstuhl, Germany, July 8-13, 2007, Proceedings, to appear*, <http://drops.dagstuhl.de/portals/index.php?semnr=07281>, 2007. See also <http://uk.arxiv.org/pdf/0707.0282.pdf> for a pre-version.
13. I. Nassi and B. Shneiderman. Flowchart techniques for structured programming. *ACM SIGPLAN Notices*, 12, 1973.
14. A. Newman. The maximum acyclic subgraph problem and degree-3 graphs. In *RANDOM-APPROX*, volume 2129 of *LNCS*, pages 147–158. Springer, 2001.
15. V. Raman and S. Saurabh. Improved fixed parameter tractable algorithms for two edge problems: MAXCUT and MAXDAG. *Inf. Process. Lett.*, 104(2):65–72, 2007.
16. I. Razgon. Computing minimum directed feedback vertex set in $\mathcal{O}(1.9977^n)$. In *ICTCS*, 2007. to appear.
17. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Systems Man Cybernet.*, 11(2):109–125, 1981.