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 [2,16]. 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 [12] 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 [13] showed a factor $\frac{12}{11}$-approximation. 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 [14], with n the number of vertices. Recently, J. Chen, I. Razgon *et al.* [4] showed that FAS $\in \mathcal{FPT}$. In contrast to MAS, it still admits a fairly vast run time of $\mathcal{O}^*(8^k k!)$. Likewise, I. Razgon [15] 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 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.

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 $a \in A$, $\mathit{init}(a)$ denotes the vertex at the tip of the arc a and $\mathit{ter}(a)$ the end. We distinguish between two kinds of arc-neighborhoods of a vertex v which are $E^+(v) := \{a \in A \mid \mathit{ter}(a) = v\}$ and $E^-(v) := \{a \in A \mid \mathit{init}(a) = v\}$. 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 $N_A(a) := \{a_1, a_2 \in A \mid \mathit{ter}(a_1) = \mathit{init}(a), \mathit{ter}(a) = \mathit{init}(a_2)\}$ and for $A' \subseteq A$ we set $N_A(A') := \bigcup_{a' \in A'} N_A(a')$. For $V' \subseteq V$ we set $A(V') := \{a \in A \mid \exists u, v \in V', \mathit{init}(a) = u, \mathit{ter}(a) = v\}$. 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.

2 Reference Search Trees

We will introduce a new kind of search scheme for combinatorial optimization problems. These problems can usually be modeled as follows. We are given a Triple $(\mathcal{U}, \mathcal{S}, c)$ such that $\mathcal{U} = \{u_1, \ldots, u_n\}$ is called the *universe*, $\mathcal{S} \subseteq \mathcal{P}(\mathcal{U})$ is the *solution space* and $c : \mathcal{P}(\mathcal{U}) \to \mathbb{N}$ is the *value function*. Generally we are looking for a $S \in \mathcal{S}$ such that $c(S)$ is minimum or maximum. We then speak of a combinatorial minimization (maximization, resp.) problem. The general search space is $\mathcal{P}(\mathcal{U})$.
The *set vector (sv_Q)* of a set $Q \in \mathcal{P}(\mathcal{U})$ is a 0/1-vector indexed by the elements of $\mathcal{U}$ such that: $sv_Q[i] = 1 \iff u_i \in Q$. We write $sv_Q \in \mathcal{S}$ when we mean $Q \in \mathcal{S}$ A *solvec* is a $0/1/\star$-vector. We define the following partial order $\preceq$ on solvecs s_1, s_2 of length n:

$$\begin{aligned} s_1 \preceq s_2 \iff & \forall 1 \leq i \leq n : (s_1[i] = \star \Rightarrow s_2[i] = \star) \\ & \wedge (s_1[i] = d \ (d \in \{0, 1\}) \Rightarrow s_2[i] \in \{d, \star\}). \end{aligned}$$

A *branching* is a directed tree $D(V, T)$ with root $r \in V$ such that all arcs are directed from the father-vertex to the child-vertex. For a vertex $u \in V$ the term B_u refers to the sub-tree rooted at u.

Definition 1. *A* reference search tree (rst) *for a combinatorial minimization (maximization, resp.) problem $(\mathcal{U}, \mathcal{S}, c)$ is a directed graph $D(V, T \cup R)$ together with a injective function label : $V \to \{(z_1, \ldots, z_n) \mid z_i \in \{0, 1, \star\}\}$ with the following properties:*

1. *$D(V, T)$ is a branching.*
2. *$D(V, T \cup R)$ is acyclic.*
3. *If u is a child of v in $D(V, T)$ then $\mathit{label}(u) \preceq \mathit{label}(v)$.*

4. *For any set vector sv_Q of a set $Q \in \mathcal{P}(\mathcal{U})$ with $Q \in \mathcal{S}$ and a vertex $v \in V$ such that $sv_Q \preceq label(v)$ we have either one of the following properties:*
   (a) *There exists a leaf $z \in V(B_v)$ such that $c(label(z)) \leq c(sv_Q)$ ($c(label(z)) \geq c(sv_Q)$, resp.) and $label(z) \in \mathcal{S}$.*
   (b) *There exists a vertex $x \in V(B_v)$ such that there is exactly one arc $(x, y) \in R$ and we have that there is a $0/1$-vector h with $h \preceq label(y)$ and $c(h) \leq c(sv_Q)$ ($c(h) \geq c(sv_Q)$, resp.) and $h \in \mathcal{S}$.*

How can a rst be exploited algorithmically? It is important to see that in a rst all the information for finding an optimal solution is included. Ordinary search trees can be defined by skipping item (b) of Definition 1. In a search tree we skip a solution s with $s \preceq u$ for a sub-tree B_u if we can find a solution in B_u which is no worse. In a rst we also have the possibility to make a reference to another subtree B_f where such a solution could be found. In B_f it might also be the case that we have to follow a reference once more. So, the only obstacle seems to be that if we follow reference after reference we end up in a cycle. But this is prevented by item 2. of Definition 1. An algorithm building up an rst can eventually benefit by cutting of branches and introducing references instead.

3 The Algorithm

3.1 Preprocessing

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.

In [7,13] 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 2. *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 [13, 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.*

Proof. Suppose cycles C_1 and C_2 share a path $P = u_1 \ldots u_\ell$. We show that at least one arc of P must be an α-arc. Suppose that any arc (u_i, u_{i+1}) with $1 \leq i \leq \ell - 1$ is not an α-arc. By induction on i ,$1 \leq i \leq \ell - 1$ we show that any (u_i, u_{i+1}) is a join. For $i = 1$ this is clear as C_1 and C_2 both enter P at u_1. This means that two arcs point towards u_1 and by the $(1, n)$-property we have that (u_1, u_2) is a join. Now suppose the claim holds for any $(u_j.u_{j+1})$ with $j \leq i$. Consider the vertex u_{i+1}. Because we have $d(u_{i+1}) \geq 3$ there must be another arc a incident to u_{i+1} with $a \notin A(P)$. As (u_i, u_{i+1}) is a join but not an α-arc it follows $a = (v, u_{i+1})$ for some $v \in V \setminus \{u_1, \ldots, u_\ell\}$. Hence, (u_{i+1}, u_{i+2}) is a join. Especially $(u_{\ell-1}, u_\ell)$ is a join. But on the other hand it must also be a fork. This is due to the two arcs a_1, a_2 leaving u_ℓ $(init(a_1) = init(a_2) = u_\ell)$ such that $a_1 \in A(C_1 \setminus C_2)$ and $a_2 \in A(C_2 \setminus C_1)$. Thus, $(u_{\ell-1}, u_\ell)$ is an α-arc which contradicts our first assumption. □

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.

A Simple Algorithm For $v \in V$ with $E^+(v) = \{a_1, \ldots, a_s\}$ $(E^+(v) = \{c\}$, resp.) and $E^-(v) = \{c\}$ $(E^-(v) = \{a_1, \ldots, a_s\})$, it is always better to delete c than one of $a_1, \ldots, a_s$. Therefore, we adjoin $a_1, \ldots, a_s$ 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. The preprocessing rules gives us another simple brute-force algorithm for MAS: Within the graph with m arcs, there could be at most $m/3$ arcs that are α-arcs. It is obviously sufficient to test all possible $2^{m/3} \leq 1.26^n$ many possibilities of choosing α-arcs into the (potential) feedback arc set.

```
Procedure:
Preprocess($\mathcal{MAS}$,$G(V,A)$,$k$):
1: repeat
2:   cont ← false
3:   apply Pre-1 - Pre-3 exhaustively.
4:   if Pre-4 applies then
5:     cont ← true
6: until cont=false
7: return ($\mathcal{MAS}$,$G(V,A)$,$k$,1)
```

```
Procedure:
Reduce($\mathcal{MAS}$,$G(V,A)$),$k$,$k'$,$w_k$,$w_{k'}$):
1: repeat
2:   cont ← false
3:   for i=1 to 6 do
4:     apply RR-i exhaustively.
5:     if RR-i applied then
6:       cont ← 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().

3.2 Reduction Rules.

The Overall Strategy There is a set of reduction rules from [13] for cubic graphs which also work for $(1,n)$-graphs. We want to use the power of this reduction rules also in our algorithm. 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 very beginning, we have $w_k(a) = w_{k'}(a) = 1$ for all $a \in A$ and in the course of the algorithm $w_k(a) \le w_{k'}(a)$. 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 $w_k(a) = 0$.
α-arcs which we take into $\mathcal{MAS}$ will be called *red*.

Reduction Rules for Weighted Arcs The set of reduction rules from [13] now will be adapted and modified to deal with weighted arcs. Also, we define a (linear time checkable) predicate *contractible* for all $a \in A$.

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

The meaning of this predicate is the following: if $\mathit{contractible}(a) = 0$, 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 $\mathit{contractible}(y) = 1$, 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 $\mathit{contractible}(g) = 0$, 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 $A^U_\alpha := \{a \in A_\alpha \mid a \text{ is non-red}\}$. We classify the arcs of A^U_α 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 $g \in A^U_\alpha$ 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).

In section 4.1 we will show the correctness of the reduction rules. This is due to **RR-2** being dependent on the branching strategy.

3.3 The Concrete Algorithm

We now are putting together the preprocessing rules, the reduction rules and the branching strategy. The obtained algorithm is Algorithm 1.

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

```
 1: ($\mathcal{MAS}$,$G(V,A)$,$k$,$w_k$)$\leftarrow$ 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: Sol3MAS($\mathcal{MAS}$,$G(V,A)$,$k$,$k'$,$w_{k'}$,$w_k$)
Procedure: Sol3MAS($\mathcal{MAS}$,$G(V,A)$,$k$,$k'$,$w_k$,$w_{k'}$):
 1: ($\mathcal{MAS}$,$G(V,A)$,$k$,$k'$,$w_k$,$w_{k'}$)$\leftarrow$ 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 $g \in A^U_\alpha$ then
 7:   if not 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 Sol3MAS($\mathcal{MAS}$,$G(V,A)$,$k$,$k'$,$w_k$,$w_{k'}$)
12:   else
13:     return YES
14: else
15:   return NO
```

4 The Analysis

In this section we want to show that Algorithm 1 builds up a reference search tree. Algorithm 1 first applies Preprocess(G). We call the emerging graph G'. Then Algorithm 1 simply branches on α-arcs g of G'. Either it deletes g or it is taken into $\mathcal{MAS}$. In the second case g will be called red. Also we can rely that every time we meet a red α-arc that there has been a branch on it. After the

```
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.

branching the reduction rules will be carried out exhaustively in the subsequent recursive call.
maximum Acyclic Subgraph can be modelled as a combinatorial minimization problem. First, the universe $\mathcal{U}$ is the set A_α which comprises all α-arcs in G'. Let $D(V, T)$ be the ordinary search tree build up by Algorithm 1. The *label*-function is induced by the current partial solution in some $u \in V$. That is any $u \in V$ will be mapped to a solvec sv_u. The vector sv_u has length $|A_\alpha|$. If for $g \in A_\alpha$ we have $sv_u(g) = 1$ then $g \in \mathcal{MAS}$ (i.e., g is red), if $sv_u[g] = 0$ then $g \notin \mathcal{MAS}$ (i.e., g has been deleted) and if $sv_u[g] = \star$ there had been no branching on g yet. The value function will be $c(q) = |\{i \mid q_i = 1\}|$ (i.e. the number of ones in q). The solution space are all subsets $S \subseteq A_\alpha$ which leave $G'[A \setminus S]$ acyclic.

4.1 Analyzing the Reduction Rules

Lemma 2. *1. The reduction rules are sound.*
2. After the application of Reduce(), see Figure 1, we are left with a $(1, n)$-graph with only thick α-arcs, no directed or undirected 2- or 3-cycle and no $v \in V$ with $d(v) = 2$ or $\min\{d^+(v), d^-(v)\} = 0$.

Proof. 1. Basically we show the soundness of the reduction rules that they guarantee item 4(a) of Definition 1. We will not go in to detail in each case as this will follows implicitly from our arguing. An exception will be **RR-2**. This reduction rule will also make use of item 4(b) of the definition. It is the only reduction rule who actually will insert references into the search tree.

RR-1 A vertex $v \in V$ with $d^+(v) = 0$ or $d^-(v) = 0$ cannot be entered and left by a cycle, so the incident arcs are not part of any cycle.

RR-2 For a vertex v with $E(v) = \{a, b\}$, we have to delete at most one arc from $\{a, b\}$ in order to cut a cycle. So we can take one into $\mathcal{MAS}$ and contract it. But additionally we must check if the arc we want to contract is not the last remaining arc on a cycle, which is not in $\mathcal{MAS}$. This check is done by *contractible*(). If so, we have to delete it.
Also **RR-2** differs from the one in [13] by the fact that red arcs are

dominant. It is possible that we create an new α-arc (w, v) by this rule, (w, v) being red. This is justified by the following claim:

Claim. If an α-arc (w, v) was created by merging a red arc (w, t) and a non-red arc (t, v), then w.l.o.g. $(w, v) \in \mathcal{MAS}$ (i.e. (w, v) is red).

Proof. To proof the claim we now are going to proof that Algorithm 1 builds up a reference search tree where the references are inserted by **RR-2**. As the universe $\mathcal{U}$ we choose A_α, the solution space consists of the subsets $L \subseteq A_\alpha$ such that $G'[L \cup A_{\bar{\alpha}}]$ is acylic and the value of $c(L)$ is simply $|L|$. Also we assume that the names of the α-arcs are dominant concerning **RR-2**.
This type of application of **RR-2** to an arc (t, v) and a red arc (w, t) is only possible, because at the time when (w, t) became red, there was, w.l.o.g., a directed path $P = w, t, u_1, \ldots, u_j, t', v$, and during the algorithm we deleted again and again those arcs incident to the u_i's which are not on P, see Figure 3. Due to this $u_1 \ldots u_j$ disappeared by **RR-2** applications in arbitrary order.

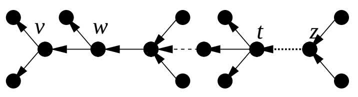


Fig. 3.

In the present search node N_1 of the search tree, just before the merging of (t, v) and (w, t), we have $label(N_1) = (e_1, \ldots, e_y, \star, \ldots, \star)$. W.l.o.g it is of the form where $e_i \in \{0, 1\}$ for $1 \leq i \leq y$. We assume that e_1 corresponds to (w, t) and e_{y+1} to (t, v). At the point when we want to merge (w, t) and (t, v), think of unwrapping P. If (w, v) is not an α-arc in G' then we do not have to branch. Thus (w, v) is such a α-arc.
To destroy any cycle passing through P, we have to delete at most one arc of P. So, deleting (t, v) would be equivalent to deleting (w, t), i.e. $(e_1, \ldots, e_y, 0, \star, \ldots, \star)$ and $(\bar{e}_1, \ldots, e_y, 1, \star, \ldots, \star)$ are equivalent in the sense that any solution $(e_1, \ldots, e_y, 0, e_{y+2}, \ldots, e_\ell)$ can be replaced by $(\bar{e}_1, \ldots, e_y, 1, e_{y+2}, \ldots, e_\ell)$ and vice versa.
Now, if we follow the path from the present node N_1 of the search tree to its root, we find the node N' in which we did a branching with respect to (w, t) (as it is red). Let N_2 be the immediately following node of N' which considers the deletion of (w, t). We have $label(N_2) = (\bar{e}_1, e_2, \ldots, e_x, \star, \ldots, \star)$ where $x \leq y$. Thus, we have
$(\bar{e}_1, \ldots, e_y, 1, \star, \ldots, \star) \preceq label(N_2)$.
Summarizing, we have that if there is some
$\varphi_1 := (e_1, \ldots, e_y, 0, e_{y+2}, \ldots, e_\ell) \in \mathcal{S}$ then it follows that

$\varphi_2 := (\bar{e}_1, \dots, e_y, 1, e_{y+2}, \dots, e_\ell) \in \mathcal{S}$, too. Also $c(\varphi_1) = c(\varphi_2)$ holds. Consequently as $label(\varphi) \preceq label(N_2)$ we do not have to consider the possibility $(e_1, \dots, e_y, 0, \star, \dots, \star)$. The reference which was inserted this way into D is (N_1, N_2).

Up to here we have shown items 1, 3, and 4. We now proof that $D(V, T \cup R)$ is acyclic. We do is in a graphical way. Draw $D(V, T)$ in the plane with x- and y-coordinates. If u is a point in the plane then $pos_x(u)$ denotes its x- and $pos_y(u)$ its y-coordinate. It is possible to draw $D(V, T)$ such that we have two properties.

- First, if $v \in V(D)$ is a father of u then $pos_y(v) > pos_y(u)$.
- Second, any vertex $v \in V(D)$ has two children u_{v_0}, u_{v_1}. u_{v_0} corresponds to the branch where we deleted some α-arc g, in u_{v_1} we decided $g \in \mathcal{MAS}$. We want D to be drawn such that for all $z \in B_{u_{v_0}}$ we have $pos_x(v) < pos_x(z)$ and for all $z \in B_{u_{v_1}}$ we have $pos_x(z) < pos_x(v)$.

Hence, we have $pos_x(N_1) < pos_x(N')$ and $pos_x(N') < pos_x(N_2)$. Therefore the arc (N_1, N_2) is pointing from the left to the right in the drawing. This is true for any reference as they are only inserted by **RR-2**. Because $D(V, T)$ is also a branching it must be the case that $D(V, T \cup R)$ is acyclic as otherwise there should be a reference pointing form right to left. □

The proof of the claim was quite lengthy but is needed to analyze the interactions between the reduction rules and the branching strategy later on. It surely also applies in the case when we merge a red arc (t, v) with a non-red arc (w, t) such that (w, v) becomes an α-arc.

RR-3 If an α-arc is not contractible, it must be deleted because it is the only arc not in $\mathcal{MAS}$ for some cycle, so **RR-3** is correct.

RR-4 If $g \in A^U_\alpha$ is thin, it can cut only one cycle C. Because it is contractible (**RR-3** was carried out before), there must be another α-arc g' which is able to cut C (possibly also some other cycle). We take g into $\mathcal{MAS}$ because it is no worse to delete g' than to delete g.

RR-5 Let $u, v \in V$ be the endpoints of an undirected 2-cycle. W.l.o.g., there are arcs a_1, a_2 with $init(a_1) = init(a_2) = u$ and $ter(a_1) = ter(a_2) = v$. Because having no vertices of degree less than three, the $(1, n)$-property and **RR-1**, there are distinct arcs (c, u) and (v, b). Clearly, it is better to delete one of these arcs than to delete a_1 and a_2 (we have to delete both because they form an undirected 2-cycle). So by deleting, w.l.o.g., a_1, we trigger **RR-2** possibly on a_2. If, w.l.o.g, (c, u) will be contracted and a_2 deleted in a later step, we also should exchange these properties in a post-processing step, i.e., we delete (c, u) and take a_2 into $\mathcal{MAS}$.

RR-6 If we have $(u, v), (v, w), (u, w) \in A$, there must be also $(a, u), (w, b) \in A$ and w.l.o.g., $(c, v) \in A$, because of the absence of vertices of degree less than three. It is always better to delete (w, b) or (a, u) than to delete $(u, v), (v, w), (u, w)$. Also, any cycle C passing through (u, w) passes also through $(u, v), (v, w)$. If take care only of cycles passing through

$(u, v), (v, w)$ we also cover those passing through (u, w). This justifies the deletion of (u, w). By deleting (u, w), **RR-2** possibly will be applied to v or w. Again, if eventually (v, w) will be deleted and (w, b) taken into $\mathcal{MAS}$, we should exchange these properties in a post-processing step. The same is true for (u, v) and (a, u).

2. Due to **RR-1** theres is no $v \in V$ with $\min\{d^+(u), d^-(v)\} = 0$. By **RR-2** we have $d(u) \geq 3$. Note that after the application of preprocess any cycle has at least one α-arc. Also observe that any directed 2- or 3- cycle has at most one α-arc. Then by **RR-3** these α-arcs will be deleted. This happens also to any thin α-arc which is the only α-arc of some cycle. Any other thin α-arc will be taken into $\mathcal{MAS}$ due to **RR-4**. After the exhaustive application of **RR-5** and **RR-6** there are no undirected 2- or 3-cycles, respectively, left. Since deleting arcs preserves the $(1, n)$-property the only critical reduction rules is **RR-2**. Here we contract an arc $a = (u, v)$ with $d^+(u) = 1$ or $d^-(u) = 1$. Both cases do not violate the $(1, n)$-property.

□

4.2 Analyzing 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 can be also due to triggering* **RR-3** *after putting g into $\mathcal{MAS}$.*

Proof. If there is a cycle C with $w_k(C \setminus q) = 1$ then there must be another arc $b \in C \setminus g$ with $w_k(b) = 1$. By taking g into $\mathcal{MAS}$ we trigger **RR-3**, which deletes b. If such a cycle C does not occur we never trigger any reduction rule by taking g into $\mathcal{MAS}$. In the case where g is deleted we immediately trigger **RR-2** as we always have graph with minimum degree 3 in any node of the search tree. Subsequently, other reduction rules might be triggered. □

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 $g \in A^U_\alpha$ and for all $a \in N_A(g)$, we have: $w_{k'}(a) > 0$.*
6. *For all $g \in A^U_\alpha$ we have $w_{k'}(g) = 1$.*

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.

As induction hypothesis, we assume that the claim is true for all search tree nodes up to depth n. Let us discuss a certain search tree node s at depth $n+1$. Let $G = (V, A)$ be the graph instance associated with s. Let k and k' be the parameter values at node s. Let $\bar{s}$ be the immediate predecessor node of s in the search tree. We will refer with $\bar{G} = (\bar{V}, \bar{A})$, $\bar{k}$, $\bar{k}'$ to the corresponding instance and parameter values. 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.

Exemplarily, we will give a very detailed proof of the very first assertion. The other parts can be similarly shown, so that we only indicate the basic steps of a complete formal proof.

1. Consider an arc $a = (u, v) \in A_{gr}$ with $w_{k'}(a) = (1-\omega)$. This situation could have been due to three reasons:
   (A) In node $\bar{s}$, we branched at an arc $\bar{d} \in N_A(\bar{a})$ with $w_{\bar{k}'}(\bar{a}) = 1$, see Figure 4 (where $d = (w, u)$). We consider here the case that $\bar{d}$ is turned red. Namely, according to case b. of the procedure "Adjust", $w_{k'}(a) = (1-\omega)$. Since we only branch at α-arcs, $\bar{d}$ is even both a fork and a join. As detailed in (B), $\bar{a}$ could give rise to $a \in V$ by a sequence of **RR-2**-applications in possible combination with other rule applications, such that $w_{k'}(a) = (1-\omega)$. As described in (C), $\bar{d}$ will yield, as a red neighbor of $\bar{a}$, again by a (possibly empty) sequence of reduction rule applications, in particular of **RR-2**-applications, a fork or join that is neighbor of a in G as required.
   (B) In node $\bar{s}$, we branched at some arc c. We consider here the case that (possibly due to an application of **RR-3**) some arc b is deleted (possibly $b = c$). This triggers some reduction rules. How could the situation have been created by reduction rule applications ? The only possibility is to (eventually) use **RR-2**. In that case, there would have been two neighbored arcs a', a'' in $\bar{G}$ with $\max\{w_{\bar{k}'}(a'), w_{\bar{k}'}(a'')\} = (1-\omega)$. Hence, at least one of these arcs, say a', actually carried the weight $(1-\omega)$. Moreover, since $a \in A_{gr}$, $a', a'' \in \bar{A}_{gr}$.

In actual fact, there could have been a whole cascade of **RR-2**- applications along a path P in $\bar{G}$ (P consists of a sequence of subsequently neighbored arcs from $\bar{A}_{gr}$), eventually leading to a, but by an easy inductive argument one can see that there must have been some $\bar{a} \in \bar{A}_{gr}$ within this cascade to which the induction hypothesis applies, so that we conclude that, in $\bar{G}$, $\bar{a}$ has a neighboring arc $\bar{d}$ that is a fork or a join. Since $\bar{d}$ is red, $\bar{d}$ is not on the path P. Since d is a fork or a join, it cannot be neighbor to two subsequent arcs from the path P. Therefore, w.l.o.g., $\bar{a}$ is the first arc on P (without predecessors on P), and $\bar{d}$ is a fork. After the sequence of **RR-2**-applications on P (possibly interrupted by reduction rules not affecting P), a has been created with $\bar{d}$ as a neighboring red fork. We will show in (C) that the fork $\bar{d}$ will eventually lead to a fork d that is neighbor of a in G.
(C) We consider the scenario that already in node $\bar{s}$, $w_{\bar{k}'}(a) = (1 - \omega)$. By induction hypothesis, assume that (w.l.o.g.) $a = (u, v)$ has a neighbored red fork $\bar{d} = (u', u)$. If $\bar{d}$ is deleted by using reduction rules, then u would have (intermediately) in-degree zero, so that **RR-1** triggers on a, contradicting our very scenario in G that we are discussing. Therefore, the local situation could only change by applications of **RR-2** involving $\bar{d}$. If those mergers refer to neighbors of $\bar{d}$ via the tip of $\bar{d}$, then either a is directly deleted or merged with $\bar{d}$. Both possibilities would destroy the scenario we discuss, since a would disappear. Therefore, such mergers could be only via the tail of $\bar{d}$. Since $\bar{d}$ is red, a merger with $\bar{d}$ will be red, as well. Moreover, this merger would be also a fork. Again by an easy induction, one can conclude that the neighbor d of a in G that results from a sequence of mergers using rule **RR-2** on a path ending at $\bar{d}$ in $\bar{G}$ would be a red fork as required.

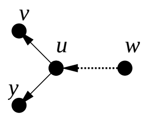


Fig. 4.

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 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

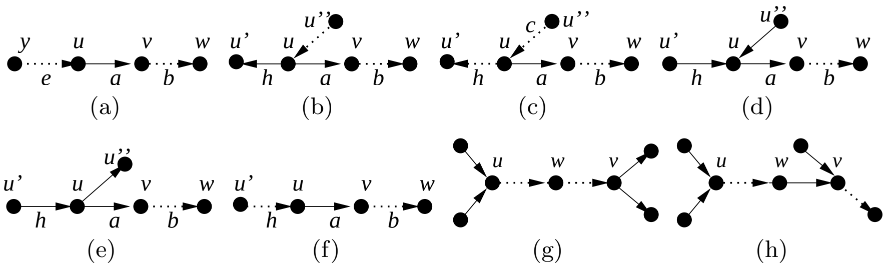


Fig. 5. Dotted lines indicate red arcs.

of our property list, see Figure 5(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 5(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 5(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 5(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 5(e). This contradicts item 2., since there is no red join protecting h.
(d) $h = (u', u)$ is red, see Figure 5(f). Suppose h is not a fork. Then all other arcs beside a and h are of the form $(\tilde{u}, u)$. 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

5(g), then the claim holds by induction. If (w, v) is not red, see Figure 5(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 $g \in A^U_\alpha$. 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.
6. Here we must consider α-arcs which are created by **RR-2**. As a matter of principle this situation has the following property: We have two arcs (u, t) and (t, v) such that u is a join and v is a fork. This **RR-2** application became possible because an arc a incident to t had been deleted in a previous reduction step. Now a must be either of the form (r, t) or (t, r). Thus, either (u, t) or (t, v) was an α-arc before a's deletion. W.l.o.g. we assume (u, t) was this α-arc.
   We now make the number n of **RR-2** applications involving an α-arc. Clearly, for $n = 0$ the claim holds. Suppose now it is also true for some n. Now (u, t) is an α-arc. If (u, t) is not red then by induction hypothesis we have $w_{k'}((u, t)) = 1$. Hence, after the **RR-2** application we have $w_{k'}((u, v)) = 1$. If (u, t) was red then the emerging α-arc (u, v) is also red and hence the premise does not apply. We would like to point out that at this point the dominance of the red arcs is crucial. As otherwise it would be possible the generate α-arcs by **RR-2** such that their $w_{k'}$-weight is smaller than one.

□

Estimating the Running Time for Max-Degree-3 Graphs. For an arc $a \in N_a(g)$ where g is an α-arc let $N_{ecl}(a) := N_A(a) \setminus (N_A(g) \cup \{g\})$. 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 item 5. of 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 $d, e \in N_{ecl}(a)$ 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 $V(d), V(e) \subset V(N_A(g))$ as we will argue in the following claim.

Proposition 1. *Let $a \in N_A(g)$ for some $g \in A^U_\alpha$ which matches case b. Then we get a reduction of at least $2-\omega$ with respect to a in case of deleting g.*

Proof. Note that if for all $a \in N_A(g)$ we have that $V(N_{ecl}(a)) \subseteq V(N_A(g))$, then $A(V(N_A(g)))$ is a component of 9 arcs, see Figure 6(a), which are handled separately.
We now examine the case where there are $a_1, a_2 \in N_A(g)$ with $N_{ecl}(a_1) \cap N_{ecl}(a_2) \neq \emptyset$ and to both applies case b.. Let $a_1, \ldots, a_\ell$ (with $2 \leq \ell \leq 4$) be a maximal sequence of arcs from $N_A(g)$ such that $N_{ecl}(a_i) \cap N_{ecl}(a_{i+1}) \neq \emptyset$ $(1 \leq i \leq \ell - 1)$. The arcs in $\cup_{i=1}^{\ell} N_{ecl}(a_i)$ form an directed or undirected path P as indicated in Figure 6 (for $\ell = 2$ see Figure 6(b), for $\ell = 3$ see Figure 6(c) and for $\ell = 4$ see Figure 6(d)).
Let $s_0, s_1, \ldots, s_\ell, s_{\ell+1}$ be the vertices of P. Observe that we must have $s_0 \neq s_{\ell+1}$ for any $\ell \in \{2,3,4\}$. In case $\ell \in \{2,3\}$ $s_0 = s_{\ell+1}$ would imply a directed or undirected 2- or 3-cycle which contradicts item 2. of Lemma 2. If $\ell = 4$ then $A(V(N_A(g)))$ is a component which was already excluded. Summarizing P is a path of $\ell+1$ arcs having each $w_{k'}$-weight at least $(1-\omega)$.
Suppose there is a vertex of P which is a source or a sink after deleting $N_A(g) \cup \{g\}$. Then is rather obvious that **RR-1** will delete all arcs of P. This yields a reduction of $(\ell+1) \cdot (1-\omega)$ with respect to P. Thus, we can say that we get a reduction of at least $2-\omega$ for each a_i.
If no vertex of P is a source or a sink after the deletion of $N_A(g) \cup \{g\}$. Then **RR-2** yields a reduction of $\ell(1-\omega)$ for P. This finally proofs the proposition. □

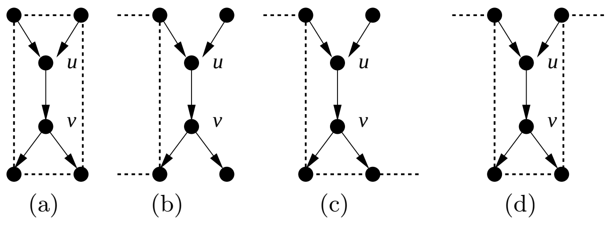


Fig. 6. Dotted lines indicate arcs which can be directed in both ways.

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 $T_1, \ldots, T_{15}$ 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 $\max\{c_1(\omega), \ldots, c_{15}(\omega)\}$. We numerically obtained $\omega = 0.1687$ so that $\max\{c_1(\omega), \ldots, c_{15}(\omega)\}$ evaluates to 1.201.

α-arc g	$a.$	$b.$	$c.$	$b'.$
$\mathcal{MAS}$	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 $\mathcal{MAS}$ or delete g and to the case applying to a.

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 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-arc, we can decrement m by one more. By adjusting $T_1, \ldots, T_{15}$ according to this and by choosing $\omega = 0.2016$, we derive an upper bound of $\mathcal{O}^*(1.1798^m)$.

Theorem 2. MAS *can be solved in* $\mathcal{O}^*(1.1798^m)$ *on* $(1, n)$*-graphs.*

A Speed-Up for the Max-Degree-3 Case 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 $l := |A^U_\alpha|$.

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

Proof. If $l \geq \left\lceil \frac{1}{4(1-\omega)} k' \right\rceil$ then by deleting A^U_α, we decrement k' by at least $l \cdot 4(1-\omega) \geq k'$, returning YES. If $l < \left\lfloor \frac{1}{1+4(1-\omega)} k' \right\rfloor$ then by taking A^U_α into $\mathcal{MAS}$ we decrement k' by at most $l \cdot (1 + 4(1-\omega)) < k'$, returning NO. □

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

Proof. So using Lemma 4, in general we can find $b \geq 1$ such that $l = \left\lceil \frac{1}{4(1-\omega)} k' - b \right\rceil$. Again, if we decided to delete A^U_α we decrement k' by at least $l \cdot 4(1-\omega)$, so that afterwards $k' \leq b4(1-\omega)$. If $k' \geq k > 0$, we have to step back and take some arcs of A^U_α into $\mathcal{MAS}$. For any such arc we can decrement k' by one more than by deleting it. Finally, we have to find at most $\lceil b4(1-\omega) \rceil$ arcs from A^U_α, which we can take in to $\mathcal{MAS}$ without causing any cyclicity. For this we have $\binom{l}{\lceil b4(1-\omega) \rceil}$ choices, which is biggest for $l = 2 \lceil b4(1-\omega) \rceil$. So for $b = \frac{1}{4(1-\omega)(9-8\omega)} k'$ this can be upper bounded asymptotically by

$$\mathcal{O}^* \left(\binom{\frac{2}{4(1-\omega)(9-8\omega)} k'}{\frac{1}{4(1-\omega)(9-8\omega)} k'} \right) \subseteq \mathcal{O}^* \left(4^{\frac{1}{4(1-\omega)(9-8\omega)} k'} \right) .$$

We mention that we have to take care of the case where $k' = l$. In this case we have to check whether $G[\mathrm{MAS} \cup A_\alpha^U]$ is acyclic and give the appropriate answer. Then the above mentioned run time for recurrence T_5 can be assumed. For $\omega = 0.2012$ we get an improved run time of $\mathcal{O}^*(1.1960^k)$, where recurrences T_4 and T_{15} are dominating. Further note that we can not make use of Lemma 4 when we measure the running time in terms of m. □

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. □

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 [13].

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 $w_{k'}(g') \leftarrow 0$.

We also add the next lines to Algorithm 1.

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.*

Proof. It should be rather obvious that applying **RR-8** is sound and does not interfere with Theorem 1.
Let us discuss the following scenario: Assume arcs (u,v) and (v,w) such that v and w have indegree one. Hence, **RR-7** could apply. Before applying **RR-7**, we find: (u,v) is a fork. (v,w) is not a join. (v,w) is a fork.

Hence, after applying **RR-7**, (u, v) is a fork. It will be an α-arc iff (u, v) was an α-arc before applying **RR-7**.

The soundness of the rule follows by induction. As well as the fact that the assertions of Theorem 1 will also hold after applying **RR-7**. Namely, if some arc (that is not removed by rule **RR-7**) was neighbor of some fork or of some join or of some red arc before applying **RR-7**, this will be true after applying **RR-7**. Moreover, observe that through applying **RR-7** or **RR-8**, no other reduction rule (numbered up to 6) could be triggered. □

Lemma 6. *We can omit branching on arcs $g \in A^U_\alpha$ with*

1. $|N_A(g)| = 5$ *and 5 occurrences of case b' or*
2. $|N_A(g)| = 4$ *with an occurrence of case b'.*

Proof. Let $g = (u, v)$.

1. Suppose the contrary holds. W.l.o.g. $d^-(v) = 2$. For the arc (v, a) case b' must match, see Figure 7. This means that $d^-(a) = 1$ and $d^+(a) \geq 3$ or otherwise **RR-1** or **RR-2** could be applied to a after the deletion of (v, a). For the distinct arc $r = (a, y)$ we must have $d^-(y) > 1$ or otherwise **RR-7** could be applied. Thus r must be an non-red α-arc (otherwise **RR-8** could be applied due to having case b' and therefore $w_{k'}(r) > 0$.). Also we must have $|N_A(r)| = 5$ and with (v, a) due to Theorem 1.5) at least one case b. occurrence. This contradicts the choice of the α-arc g, because we would have preferred r.
2. Suppose the contrary holds. W.l.o.g. take an arc (v, a). As in item 1. we can deduce that $d^-(a) = 1$ and $d^+(v) \geq 3$ and that there is a distinct non-red α-arc (a, y) with $|N_A((a, y))| \geq 5$. This is a contradiction to the choice of g. □

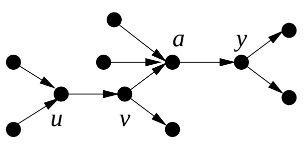


Fig. 7. The contradicting situation in the proof of Lemma 6.1)

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 6.1). Additionally we have to cover the case where we have 4 occurrences of case b' and one of case b. ($T[k] \leq T[k-(1+5\omega)]+T[k-(6-\omega)]$ and $T[m] \leq T[m - (1 + 5\omega)] + T[m - (7 - \omega)]$, resp.). 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 6.2).

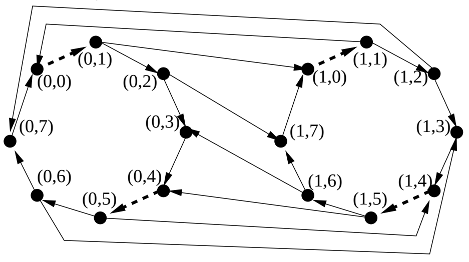
Theorem 4. *On $(1, n)$-graphs with m arcs,* MAS *is solvable in time $\mathcal{O}^*(1.1871^m)$ $(\omega = 0.2392)$ and $\mathcal{O}^*(1.212^k)$ $(\omega = 0.21689)$, respectively.*

Corollary 2. *We solve* Feedback Vertex Set *on planar graphs in $\mathcal{O}^*(1.986^n)$.*

Proof. Given a directed planar graph $G(V, A)$, use the well-known transformation form Feedback Vertex Set to Feedback Arc Set to obtain an instance $G'(V', A')$. That is every $v \in V$ will be substituted by $v_1.v_2$. Arcs a with $ter(a) = v$ now point towards v_1, those with $tail(v)$ start at v_2. Also add an arc (v_1, v_2). Hence, every $v \in V$ has a corresponding arc $a_v \in A'$. Note that G' has the $(1, n)$-property and $|A'| \leq 4n$. By running Algorithm 1 on G' we can solve Feedback Vertex Set in $\mathcal{O}^*(1.1871^{4n})$. □

5 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 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 $0 \leq i \leq r$ 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. $(((i + 1) \bmod r, (1 - j) \bmod 8, (i, j))$ 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 $\left(\frac{11}{12} + \epsilon\right) |E| + k$ many arcs ?*

6 Conclusion

We presented a simple algorithm for solving Maximum Acyclic Subgraph on $(1, n)$-graphs, which is a superclass of cubic graphs. In contrast to the algorithm

the analysis concerning correctness and run time is rather involved. Nevertheless, the analysis was accomplished in an amortized fashion showing a quite better run time than analyzed naively. This kind of analysis is also applied in terms of 'parameterized' run time. This is one of the few occasion where this is possible. We think that at least for problems on special graph classes (e.g., cubic graphs) one can find further examples.
We introduce the concept of a reference search tree which generealizes ordinary search trees. Using this concept we showed the correctness of a reduction rule (**RR-2**) which is crucial for the run time improvement. We like to point out that the validity of this rule depends on the algorithm. It is not valid on its own. We are looking forward solve more problems using this concept. It seems that especially problems are suited where we cannot simply delete a vertex once the decision has been made whether to take it into the solution or not. For example problems where the solution must be connected (e.g., Max Internal Spanning Tree or Connected Dominating Set) or (directed) feedback problems on general graphs.

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, S. Lu., B. O'Sullivan and I. Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem In *STOC*, pages 177–186, 2008
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. I. Nassi and B. Shneiderman. Flowchart techniques for structured programming. *ACM SIGPLAN Notices*, 12, 1973.
13. A. Newman. The maximum acyclic subgraph problem and degree-3 graphs. In *RANDOM-APPROX*, volume 2129 of *LNCS*, pages 147–158. Springer, 2001.

14. 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.
15. I. Razgon. Computing minimum directed feedback vertex set in $O(1.9977^n)$. In *ICTCS*, 2007. to appear.
16. 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.

A Recursions and running times

A.1 The maximum degree three case

In Table 2(a) we state the 15 recurrences necessary for solving the parameterized version of MAS on maximum degree 3 graphs, which we needed to estimate the run time in Theorem 3. They are derived from the positive integer solutions of $i + j + q = 4$. Similarly, Table 2(b) displays the recurrences for the exact, non-parameterized case (measured in m).

A.2 The $(1, n)$-case

We state Tables 3(a) which covers all recursions derived from integer solutions of $x + y + z = 4$ where x, y and z are the number of occurrences of cases a,b and c in Table 1. Tables 3(b) and 3(c) cover all recursions derived from integer solutions of $x + y + z = 5$ and $x + y + z = 6$, respectively. Here x, y and z are the number of occurrences of cases a,b' and c in Table 1. Both tables refer to the parameterized version of $\mathcal{MAS}$ on $(1, n)$-graphs. To derive the corresponding tables when we measure in m, simply transform any $T[k] \leq T[k - a] + T[k - b]$ to $T[m] \leq T[m - (a + 1)] + T[m - b]$. The corresponding are tables are given in Table 4(a), 4(b) and 4(c).
We also mention that we have to consider the recursion $T[k] \leq T[k - (1 + 5\omega)] + T[k - (6 - \omega)]$ when measuring in k and $T[m] \leq T[m - (1 + 5\omega)] + T[m - (7 - \omega)]$ when measuring in m. Both refer to the situation where an we have four occurences of case b' and one case b occurence.

Notice again that these tables are correct upper bounds in particular because we have shown in point 5. of Theorem 1 that we will always find some non-null value still attached to neighbors of arcs we branch on.

Table 2. Recursions for the max-degree 3 case. In 2(a) the measure is k, in 2(b) it is m.

(a)

No.	j	q	i	Derived recursion	Upper bound
1	0	0	4	$T[k] \leq T[k-4] + T[k-5]$	$\mathcal{O}^*(1.1674^k)$
2	0	1	3	$T[k] \leq T[k-(4-\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.1745^k)$
3	0	2	2	$T[k] \leq T[k-(4-2\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.1822^k)$
4	0	3	1	$T[k] \leq T[k-(4-3\omega)] + T[k-(5-3\omega)]$	$\mathcal{O}^*(1.191^k)$
5	0	4	0	$T[k] \leq T[k-(4-4\omega)] + T[k-(5-4\omega)]$	$\mathcal{O}^*(1.2^k)$
6	1	0	3	$T[k] \leq T[k-(5-\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.167^k)$
7	1	1	2	$T[k] \leq T[k-(5-2\omega)] + T[k-4]$	$\mathcal{O}^*(1.174^k)$
8	1	2	1	$T[k] \leq T[k-(5-3\omega)] + T[k-(4-\omega)]$	$\mathcal{O}^*(1.1817^k)$
9	1	3	0	$T[k] \leq T[k-(5-4\omega)] + T[k-(4-2\omega)]$	$\mathcal{O}^*(1.191^k)$
10	2	0	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.171^k)$
11	2	1	1	$T[k] \leq T[k-(6-3\omega)] + T[k-(3+\omega)]$	$\mathcal{O}^*(1.179^k)$
12	2	2	0	$T[k] \leq T[k-(6-4\omega)] + T[k-3]$	$\mathcal{O}^*(1.187^k)$
13	3	0	1	$T[k] \leq T[k-(7-3\omega)] + T[k-(2+3\omega)]$	$\mathcal{O}^*(1.181^k)$
14	3	1	0	$T[k] \leq T[k-(7-4\omega)] + T[k-(2+2\omega)]$	$\mathcal{O}^*(1.19^k)$
15	4	0	0	$T[k] \leq T[k-(8-4\omega)] + T[k-(1+4\omega)]$	$\mathcal{O}^*(1.2^k)$

(b)

No.	j	q	i	Derived recursion	Upper Bound
1	0	0	4	$T[m] \leq T[m-5] + T[m-5]$	$\mathcal{O}^*(1.1487^m)$
2	0	1	3	$T[m] \leq T[m-(5-\omega)] + T[m-(5-\omega)]$	$\mathcal{O}^*(1.156^m)$
3	0	2	2	$T[m] \leq T[m-(5-2\omega)] + T[m-(5-2\omega)]$	$\mathcal{O}^*(1.168^m)$
4	0	3	1	$T[m] \leq T[m-(5-3\omega)] + T[m-(5-3\omega)]$	$\mathcal{O}^*(1.1709^m)$
5	0	4	0	$T[m] \leq T[m-(5-4\omega)] + T[m-(5-4\omega)]$	$\mathcal{O}^*(1.1798^m)$
6	1	0	3	$T[m] \leq T[m-(6-\omega)] + T[m-(4+\omega)]$	$\mathcal{O}^*(1.151^m)$
7	1	1	2	$T[m] \leq T[m-(6-2\omega)] + T[m-4]$	$\mathcal{O}^*(1.158^m)$
8	1	2	1	$T[m] \leq T[m-(6-3\omega)] + T[m-(4-\omega)]$	$\mathcal{O}^*(1.165^m)$
9	1	3	0	$T[m] \leq T[m-(6-4\omega)] + T[m-(4-2\omega)]$	$\mathcal{O}^*(1.173^m)$
10	2	0	2	$T[m] \leq T[m-(7-2\omega)] + T[m-(3+2\omega)]$	$\mathcal{O}^*(1.155^m)$
11	2	1	1	$T[m] \leq T[m-(7-3\omega)] + T[m-(3+\omega)]$	$\mathcal{O}^*(1.163^m)$
12	2	2	0	$T[m] \leq T[m-(7-4\omega)] + T[m-3]$	$\mathcal{O}^*(1.171^m)$
13	3	0	1	$T[m] \leq T[m-(8-3\omega)] + T[m-(2+3\omega)]$	$\mathcal{O}^*(1.164^m)$
14	3	1	0	$T[m] \leq T[m-(8-4\omega)] + T[m-(2+2\omega)]$	$\mathcal{O}^*(1.173^m)$
15	4	0	0	$T[m] \leq T[m-(9-4\omega)] + T[m-(1+4\omega)]$	$\mathcal{O}^*(1.1798^m)$

Table 3. Recurrences for $(1,n)$-graphs where the run time is measured in terms of k.

(a)

No.	z	y	x	Derived recursion	Upper bound
1	0	0	4	$T[k] \leq T[k-4] + T[k-5]$	$\mathcal{O}^*(1.1674^k)$
2	0	1	3	$T[k] \leq T[k-(4-\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.1766^k)$
3	0	2	2	$T[k] \leq T[k-(4-2\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.187^k)$
4	0	3	1	$T[k] \leq T[k-(4-3\omega)] + T[k-(5-3\omega)]$	$\mathcal{O}^*(1.1986^k)$
5	0	4	0	$T[k] \leq T[k-(4-4\omega)] + T[k-(5-4\omega)]$	$\mathcal{O}^*(1.2118^k)$
6	1	0	3	$T[k] \leq T[k-(5-\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.167^k)$
7	1	1	2	$T[k] \leq T[k-(5-2\omega)] + T[k-4]$	$\mathcal{O}^*(1.176^k)$
8	1	2	1	$T[k] \leq T[k-(5-3\omega)] + T[k-(4-\omega)]$	$\mathcal{O}^*(1.1862^k)$
9	1	3	0	$T[k] \leq T[k-(5-4\omega)] + T[k-(4-2\omega)]$	$\mathcal{O}^*(1.1978^k)$
10	2	0	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.171^k)$
11	2	1	1	$T[k] \leq T[k-(6-3\omega)] + T[k-(3+\omega)]$	$\mathcal{O}^*(1.18^k)$
12	2	2	0	$T[k] \leq T[k-(6-4\omega)] + T[k-3]$	$\mathcal{O}^*(1.191^k)$
13	3	0	1	$T[k] \leq T[k-(7-3\omega)] + T[k-(2+3\omega)]$	$\mathcal{O}^*(1.1179^k)$
14	3	1	0	$T[k] \leq T[k-(7-4\omega)] + T[k-(2+2\omega)]$	$\mathcal{O}^*(1.19^k)$
15	4	0	0	$T[k] \leq T[k-(8-4\omega)] + T[k-(1+4\omega)]$	$\mathcal{O}^*(1.195^k)$

(b)

No.	x	y	z	Derived recursion	Upper bound
1	5	0	0	$T[k] \leq T[k-5] + T[k-6]$	$\mathcal{O}^*(1.135^k)$
2	4	1	0	$T[k] \leq T[k-5] + T[k-(5+\omega)]$	$\mathcal{O}^*(1.146^k)$
3	4	0	1	$T[k] \leq T[k-(5-\omega)] + T[k-(6-\omega)]$	$\mathcal{O}^*(1.141^k)$
4	3	2	0	$T[k] \leq T[k-5] + T[k-(4+2\omega)]$	$\mathcal{O}^*(1.159^k)$
5	3	1	1	$T[k] \leq T[k-(5-\omega)] + T[k-5]$	$\mathcal{O}^*(1.153^k)$
6	3	0	2	$T[k] \leq T[k-(5-2\omega)] + T[k-(6-2\omega)]$	$\mathcal{O}^*(1.148^k)$
7	2	3	0	$T[k] \leq T[k-5] + T[k-(3+3\omega)]$	$\mathcal{O}^*(1.176^k)$
8	2	2	1	$T[k] \leq T[k-(5-\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.167^k)$
9	2	1	2	$T[k] \leq T[k-(5-2\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.16^k)$
10	2	0	3	$T[k] \leq T[k-(5-3\omega)] + T[k-(6-3\omega)]$	$\mathcal{O}^*(1.155^k)$
11	1	4	0	$T[k] \leq T[k-5] + T[k-(2+4\omega)]$	$\mathcal{O}^*(1.199^k)$
12	1	3	1	$T[k] \leq T[k-(5-\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.186^k)$
13	1	2	2	$T[k] \leq T[k-(5-2\omega)] + T[k-4]$	$\mathcal{O}^*(1.176^k)$
14	1	1	3	$T[k] \leq T[k-(5-3\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.169^k)$
15	1	0	4	$T[k] \leq T[k-(5-4\omega)] + T[k-(6-4\omega)]$	$\mathcal{O}^*(1.163^k)$
16	0	5	0	$T[k] \leq T[k-5] + T[k-(1+5\omega)]$	$\mathcal{O}^*(1.232^k)$
17	0	4	1	$T[k] \leq T[k-(5-\omega)] + T[k-(2+3\omega)]$	$\mathcal{O}^*(1.2118^k)$
18	0	3	2	$T[k] \leq T[k-(5-2\omega)] + T[k-(3+\omega)]$	$\mathcal{O}^*(1.198^k)$
19	0	2	3	$T[k] \leq T[k-(5-3\omega)] + T[k-(4-\omega)]$	$\mathcal{O}^*(1.187^k)$
20	0	1	4	$T[k] \leq T[k-(5-4\omega)] + T[k-(5-3\omega)]$	$\mathcal{O}^*(1.178^k)$
21	0	0	5	$T[k] \leq T[k-(5-5\omega)] + T[k-(6-5\omega)]$	$\mathcal{O}^*(1.171^k)$

(c)

No.	x	y	z	Derived recursion	Upper bound
1	6	0	0	$T[k] \leq T[k-6] + T[k-7]$	$\mathcal{O}^*(1.1128^k)$
2	5	1	0	$T[k] \leq T[k-6] + T[k-(6+\omega)]$	$\mathcal{O}^*(1.1202^k)$
3	5	0	1	$T[k] \leq T[k-(6-\omega)] + T[k-(7-\omega)]$	$\mathcal{O}^*(1.117^k)$
4	4	2	0	$T[k] \leq T[k-6] + T[k-(5+\omega)]$	$\mathcal{O}^*(1.1291^k)$
5	4	1	1	$T[k] \leq T[k-(6-\omega)] + T[k-6]$	$\mathcal{O}^*(1.125^k)$
6	4	0	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(7-2\omega)]$	$\mathcal{O}^*(1.1214^k)$
7	3	3	0	$T[k] \leq T[k-6] + T[k-(4+3\omega)]$	$\mathcal{O}^*(1.14^k)$
8	3	2	1	$T[k] \leq T[k-(6-\omega)] + T[k-(5+\omega)]$	$\mathcal{O}^*(1.135^k)$
9	3	1	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(6-\omega)]$	$\mathcal{O}^*(1.13^k)$
10	3	0	3	$T[k] \leq T[k-(6-3\omega)] + T[k-(7-3\omega)]$	$\mathcal{O}^*(1.127^k)$
11	2	4	0	$T[k] \leq T[k-(6)] + T[k-(3+4\omega)]$	$\mathcal{O}^*(1.154^k)$
12	2	3	1	$T[k] \leq T[k-(6-\omega)] + T[k-(4+2\omega)]$	$\mathcal{O}^*(1.145^k)$
13	2	2	2	$T[k] \leq T[k-(6-2\omega)] + T[k-5]$	$\mathcal{O}^*(1.141^k)$
14	2	1	3	$T[k] \leq T[k-(6-3\omega)] + T[k-(6-2\omega)]$	$\mathcal{O}^*(1.136^k)$
15	2	0	4	$T[k] \leq T[k-(6-4\omega)] + T[k-(7-4\omega)]$	$\mathcal{O}^*(1.132^k)$
16	1	5	0	$T[k] \leq T[k-6] + T[k-(2+5\omega)]$	$\mathcal{O}^*(1.172^k)$
17	1	4	1	$T[k] \leq T[k-(6-\omega)] + T[k-(3+3\omega)]$	$\mathcal{O}^*(1.162^k)$
18	1	3	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.154^k)$
19	1	2	3	$T[k] \leq T[k-(6-3\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.147^k)$
20	1	1	4	$T[k] \leq T[k-(6-4\omega)] + T[k-(6-3\omega)]$	$\mathcal{O}^*(1.142^k)$
21	1	0	5	$T[k] \leq T[k-(6-5\omega)] + T[k-(7-5\omega)]$	$\mathcal{O}^*(1.137^k)$
22	0	6	0	$T[k] \leq T[k-6] + T[k-(1+6\omega)]$	$\mathcal{O}^*(1.198^k)$
23	0	5	1	$T[k] \leq T[k-(6-\omega)] + T[k-(2+4\omega)]$	$\mathcal{O}^*(1.182^k)$
24	0	4	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.171^k)$
25	0	3	3	$T[k] \leq T[k-(6-3\omega)] + T[k-4]$	$\mathcal{O}^*(1.162^k)$
26	0	2	4	$T[k] \leq T[k-(6-4\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.154^k)$
27	0	1	5	$T[k] \leq T[k-(6-5\omega)] + T[k-(6-4\omega)]$	$\mathcal{O}^*(1.148^k)$
28	0	0	6	$T[k] \leq T[k-(6-6\omega)] + T[k-(7-6\omega)]$	$\mathcal{O}^*(1.144^k)$

Table 4. Recurrences for $(1,n)$-graphs where the run time is measured in terms of m.

(a)

No.	z	y	x	Derived recursion	Upper bound
1	0	0	4	$T[k] \leq T[k-4] + T[k-5]$	$\mathcal{O}^*(1.149^m)$
2	0	1	3	$T[k] \leq T[k-(4-\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.157^m)$
3	0	2	2	$T[k] \leq T[k-(4-2\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.166^m)$
4	0	3	1	$T[k] \leq T[k-(4-3\omega)] + T[k-(5-3\omega)]$	$\mathcal{O}^*(1.1176^m)$
5	0	4	0	$T[k] \leq T[k-(4-4\omega)] + T[k-(5-4\omega)]$	$\mathcal{O}^*(1.1.1871^m)$
6	1	0	3	$T[k] \leq T[k-(5-\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.15^m)$
7	1	1	2	$T[k] \leq T[k-(5-2\omega)] + T[k-4]$	$\mathcal{O}^*(1.159^m)$
8	1	2	1	$T[k] \leq T[k-(5-3\omega)] + T[k-(4-\omega)]$	$\mathcal{O}^*(1.168^m)$
9	1	3	0	$T[k] \leq T[k-(5-4\omega)] + T[k-(4-2\omega)]$	$\mathcal{O}^*(1.178^m)$
10	2	0	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.155^m)$
11	2	1	1	$T[k] \leq T[k-(6-3\omega)] + T[k-(3+\omega)]$	$\mathcal{O}^*(1.164^m)$
12	2	2	0	$T[k] \leq T[k-(6-4\omega)] + T[k-3]$	$\mathcal{O}^*(1.174^m)$
13	3	0	1	$T[k] \leq T[k-(7-3\omega)] + T[k-(2+3\omega)]$	$\mathcal{O}^*(1.163^m)$
14	3	1	0	$T[k] \leq T[k-(7-4\omega)] + T[k-(2+2\omega)]$	$\mathcal{O}^*(1.173^m)$
15	4	0	0	$T[k] \leq T[k-(8-4\omega)] + T[k-(1+4\omega)]$	$\mathcal{O}^*(1.176^m)$

(b)

No.	x	y	z	Derived recursion	Upper bound
1	5	0	0	$T[k] \leq T[k-5] + T[k-6]$	$\mathcal{O}^*(1.123^m)$
2	4	1	0	$T[k] \leq T[k-5] + T[k-(5+\omega)]$	$\mathcal{O}^*(1.132^m)$
3	4	0	1	$T[k] \leq T[k-(5-\omega)] + T[k-(6-\omega)]$	$\mathcal{O}^*(1.128^m)$
4	3	2	0	$T[k] \leq T[k-5] + T[k-(4+2\omega)]$	$\mathcal{O}^*(1.143^m)$
5	3	1	1	$T[k] \leq T[k-(5-\omega)] + T[k-5]$	$\mathcal{O}^*(1.138^m)$
6	3	0	2	$T[k] \leq T[k-(5-2\omega)] + T[k-(6-2\omega)]$	$\mathcal{O}^*(1.134^m)$
7	2	3	0	$T[k] \leq T[k-5] + T[k-(3+3\omega)]$	$\mathcal{O}^*(1.157^m)$
8	2	2	1	$T[k] \leq T[k-(5-\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.15^m)$
9	2	1	2	$T[k] \leq T[k-(5-2\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.145^m)$
10	2	0	3	$T[k] \leq T[k-(5-3\omega)] + T[k-(6-3\omega)]$	$\mathcal{O}^*(1.141^m)$
11	1	4	0	$T[k] \leq T[k-5] + T[k-(2+4\omega)]$	$\mathcal{O}^*(1.176^m)$
12	1	3	1	$T[k] \leq T[k-(5-\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.166^m)$
13	1	2	2	$T[k] \leq T[k-(5-2\omega)] + T[k-4]$	$\mathcal{O}^*(1.159^m)$
14	1	1	3	$T[k] \leq T[k-(5-3\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.153^m)$
15	1	0	4	$T[k] \leq T[k-(5-4\omega)] + T[k-(6-4\omega)]$	$\mathcal{O}^*(1.148^m)$
16	0	5	0	$T[k] \leq T[k-5] + T[k-(1+5\omega)]$	
17	0	4	1	$T[k] \leq T[k-(5-\omega)] + T[k-(2+3\omega)]$	$\mathcal{O}^*(1.1871^m)$
18	0	3	2	$T[k] \leq T[k-(5-2\omega)] + T[k-(3+\omega)]$	$\mathcal{O}^*(1.177^m)$
19	0	2	3	$T[k] \leq T[k-(5-3\omega)] + T[k-(4-\omega)]$	$\mathcal{O}^*(1.168^m)$
20	0	1	4	$T[k] \leq T[k-(5-4\omega)] + T[k-(5-3\omega)]$	$\mathcal{O}^*(1.161^m)$
21	0	0	5	$T[k] \leq T[k-(5-5\omega)] + T[k-(6-5\omega)]$	$\mathcal{O}^*(1.156^m)$

(c)

No.	x	y	z	Derived recursion	Upper bound
1	6	0	0	$T[k] \leq T[k-6] + T[k-7]$	$\mathcal{O}^*(1.105^m)$
2	5	1	0	$T[k] \leq T[k-6] + T[k-(6+\omega)]$	$\mathcal{O}^*(1.12^m)$
3	5	0	1	$T[k] \leq T[k-(6-\omega)] + T[k-(7-\omega)]$	$\mathcal{O}^*(1.11^m)$
4	4	2	0	$T[k] \leq T[k-6] + T[k-(5+\omega)]$	$\mathcal{O}^*(1.12^m)$
5	4	1	1	$T[k] \leq T[k-(6-\omega)] + T[k-6]$	$\mathcal{O}^*(1.12^m)$
6	4	0	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(7-2\omega)]$	$\mathcal{O}^*(1.12^m)$
7	3	3	0	$T[k] \leq T[k-6] + T[k-(4+3\omega)]$	$\mathcal{O}^*(1.128^m)$
8	3	2	1	$T[k] \leq T[k-(6-\omega)] + T[k-(5+\omega)]$	$\mathcal{O}^*(1.124^m)$
9	3	1	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(6-\omega)]$	$\mathcal{O}^*(1.12^m)$
10	3	0	3	$T[k] \leq T[k-(6-3\omega)] + T[k-(7-3\omega)]$	$\mathcal{O}^*(1.12^m)$
11	2	4	0	$T[k] \leq T[k-(6)] + T[k-(3+4\omega)]$	$\mathcal{O}^*(1.14^m)$
12	2	3	1	$T[k] \leq T[k-(6-\omega)] + T[k-(4+2\omega)]$	$\mathcal{O}^*(1.14^m)$
13	2	2	2	$T[k] \leq T[k-(6-2\omega)] + T[k-5]$	$\mathcal{O}^*(1.129^m)$
14	2	1	3	$T[k] \leq T[k-(6-3\omega)] + T[k-(6-2\omega)]$	$\mathcal{O}^*(1.125^m)$
15	2	0	4	$T[k] \leq T[k-(6-4\omega)] + T[k-(7-4\omega)]$	$\mathcal{O}^*(1.122^m)$
16	1	5	0	$T[k] \leq T[k-6] + T[k-(2+5\omega)]$	$\mathcal{O}^*(1.16^m)$
17	1	4	1	$T[k] \leq T[k-(6-\omega)] + T[k-(3+3\omega)]$	$\mathcal{O}^*(1.15^m)$
18	1	3	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(4+\omega)]$	$\mathcal{O}^*(1.14^m)$
19	1	2	3	$T[k] \leq T[k-(6-3\omega)] + T[k-(5-\omega)]$	$\mathcal{O}^*(1.14^m)$
20	1	1	4	$T[k] \leq T[k-(6-4\omega)] + T[k-(6-3\omega)]$	$\mathcal{O}^*(1.14^m)$
21	1	0	5	$T[k] \leq T[k-(6-5\omega)] + T[k-(7-5\omega)]$	$\mathcal{O}^*(1.13^m)$
22	0	6	0	$T[k] \leq T[k-6] + T[k-(1+6\omega)]$	$\mathcal{O}^*(1.175^m)$
23	0	5	1	$T[k] \leq T[k-(6-\omega)] + T[k-(2+4\omega)]$	$\mathcal{O}^*(1.164^m)$
24	0	4	2	$T[k] \leq T[k-(6-2\omega)] + T[k-(3+2\omega)]$	$\mathcal{O}^*(1.155^m)$
25	0	3	3	$T[k] \leq T[k-(6-3\omega)] + T[k-4]$	$\mathcal{O}^*(1.15^m)$
26	0	2	4	$T[k] \leq T[k-(6-4\omega)] + T[k-(5-2\omega)]$	$\mathcal{O}^*(1.15^m)$
27	0	1	5	$T[k] \leq T[k-(6-5\omega)] + T[k-(6-4\omega)]$	$\mathcal{O}^*(1.14^m)$
28	0	0	6	$T[k] \leq T[k-(6-6\omega)] + T[k-(7-6\omega)]$	$\mathcal{O}^*(1.144^m)$