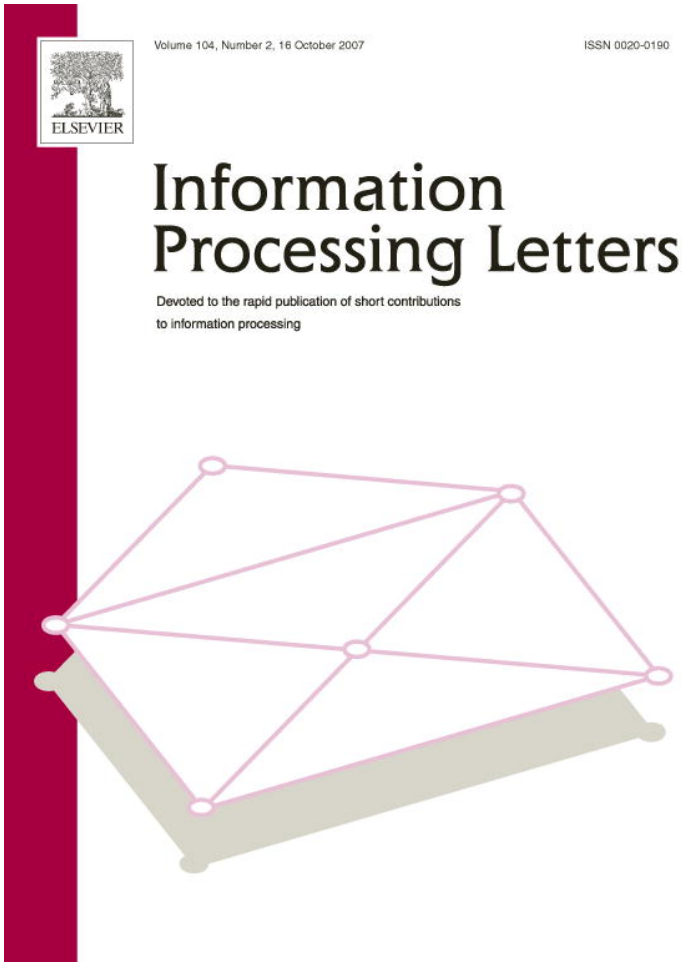

Available online at www.sciencedirect.com
ScienceDirect
Information Processing Letters 104 (2007) 65–72
Information Processing Letters
www.elsevier.com/locate/ipl

Improved fixed parameter tractable algorithms for two "edge" problems: MAXCUT and MAXDAG

Venkatesh Raman, Saket Saurabh *

The Institute of Mathematical Sciences, Chennai 600 113, India

Received 5 October 2006
Available online 5 July 2007
Communicated by F. Dehne

Abstract

We give improved parameterized algorithms for two "edge" problems MAXCUT and MAXDAG, where the solution sought is a subset of edges. MAXCUT of a graph is a maximum set of edges forming a bipartite subgraph of the given graph. On the other hand, MAXDAG of a directed graph is a set of arcs of maximum size such that the graph induced on these arcs is acyclic. Our algorithms are obtained through new kernelization and efficient exact algorithms for the optimization versions of the problems. More precisely our results include:

(i) a kernel with at most αk vertices and βk edges for MAXCUT. Here $0 < \alpha \leqslant 1$ and $1 < \beta \leqslant 2$. Values of α and β depends on the number of vertices and the edges in the graph;
(ii) a kernel with at most $4k/3$ vertices and $2k$ edges for MAXDAG;
(iii) an $O^*(1.2418^k)$ parameterized algorithm for MAXCUT in undirected graphs. This improves the $O^*(1.4143^k)$[1] algorithm presented in [E. Prieto, The method of extremal structure on the k-maximum cut problem, in: The Proceedings of Computing: The Australasian Theory Symposium (CATS), 2005, pp. 119–126];
(iv) an $O^*(2^n)$ algorithm for optimization version of MAXDAG in directed graphs. This is the first such algorithm to the best of our knowledge;
(v) an $O^*(2^k)$ parameterized algorithm for MAXDAG in directed graphs. This improves the previous best of $O^*(4^k)$ presented in [V. Raman, S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments, Theoretical Computer Science 351 (3) (2006) 446–458];
(vi) an $O^*(16^k)$ parameterized algorithm to determine whether an oriented graph having m arcs has an acyclic subgraph with at least $m/2 + k$ arcs. This improves the $O^*(2^k)$ algorithm given in [V. Raman, S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments, Theoretical Computer Science 351 (3) (2006) 446–458].

In addition, we show that if a directed graph has minimum out degree at least $f(n)$ (some function of n) then DIRECTED FEEDBACK ARC SET problem is fixed parameter tractable. The parameterized complexity of DIRECTED FEEDBACK ARC SET is a well-known open problem.

* Corresponding author.
E-mail addresses: vraman@imsc.res.in (V. Raman), saket@imsc.res.in (S. Saurabh).

[1] The O^* notation suppresses polynomial terms. Thus we write $O^*(T(x))$ for a time complexity of the form $O(T(x)) \cdot \text{poly}(|x|)$ where $T(x)$ grows exponentially with $|x|$, the input size. See [G. Woeginger, Exact algorithms for NP-hard problems: A survey, in: Combinatorial Optimization—Eureka! You Shrink! in: Lecture Notes in Comput. Sci., vol. 2570, 2003, pp. 185–207] for a detailed discussion on this.

0020-0190/$ – see front matter
doi:10.1016/j.ipl.2007.05.014

Keywords: Directed feedback arc set; Max-cut; Parameterized complexity; Exact algorithm; Graph algorithms

1. Introduction

Parameterized Complexity theory is a recently developed framework for a refined analysis of hard algorithmic problems. For decision problems with input size n, and a parameter k (which typically, and in all the problems we consider in this paper, is the solution size), the goal in parameterized algorithms is to design an exact algorithm with runtime $f(k)n^{O(1)}$ where f is a function of k alone, against a trivial $n^{k+O(1)}$ algorithm. Problems having such an algorithm is said to be fixed parameter tractable (FPT), and such algorithms are practical when small parameters cover practical ranges. The book by Downey and Fellows [3] provides a good introduction to the topic of parameterized complexity. For recent developments see books by Flum and Grohe [5] and Niedermeier [15].

An aspect of parameterized complexity theory that has gained importance more recently is its close connection to obtaining efficient exact exponential algorithms for optimization problems. Recent years have seen a lot of growing interest in the field and have led to the development of fast exponential time algorithms for various problems, including SATISFIABILITY [12,22], COLORING [4,2,1,20], MAXIMUM INDEPENDENT SET [21,6], DOMINATING SET [8,9], ODD CYCLE TRANSVERSAL [2,20] and many others. See recent surveys by Fomin et al. [7] and Woeginger [23] for an overview.

In [19,20] it is shown that if we have a fixed parameter algorithm for a problem whose runtime satisfies certain properties, then this algorithm can be turned into an exact algorithm for the optimization version, running in time better than the trivial brute force algorithm. This showed a very clear connection between the parameterized and exact algorithms. Here we demonstrate work in the other direction; we use exact algorithms for optimization versions to develop fast parameterized algorithms, using efficient *kernelization* a well-known method for obtaining parameterized algorithms.

The main idea of kernelization is to replace a given instance (I, k) by a simpler instance (I', k') using some *data reduction rules* in polynomial time such that (I, k) is an yes instance if and only if (I', k') is an yes instance and $|I'|$ is bounded by a function of k alone. The reduced instance, I', is called the *kernel* of the problem. In this paper we first give improved kernelization for MAXCUT and MAXDAG problems. For both these problems, a kernel of size $2k$ [14,18] was known for the number of edges ($m \leqslant 2k$). Prieto [17] gave a kernel for MAXCUT with k vertices and $2k$ edges.

In Section 2, we first give a kernel for MAXCUT having αk vertices and βk edges where α and β are at most 1 and 2, respectively. As the number of edges increases in the graph, β becomes closer to 2 while α gets closer to 0. Then we use known exact algorithms for the optimization version of MAXCUT problem on an "appropriate" sized kernel and obtain an improved fixed parameter tractable algorithm for MAXCUT running in time $O^*(1.2418^k)$. This improves the previously known bound of $O^*(1.4143^k)$ given in [17].

Section 3 deals with the parameterized and optimization version of the MAXDAG problem. We first give an algorithm for an optimization version of MAXDAG problem. More precisely, given a directed graph $G = (V, A)$ on n vertices, we give an exact algorithm with running time $O^*(2^n)$, to find a maximum sized subset of arcs $D \subseteq A$ such that $G' = (V, D)$ is a directed acyclic graph. This is the first such algorithm to the best of our knowledge. Then we give improved parameterized algorithms for MAXDAG and its variants as applications of the exact algorithm developed for its optimization version. We first develop a kernel of size at most $4k/3$ for the number of vertices for MAXDAG. Then applying the $O^*(2^n)$ time algorithm for the optimization version on this kernel, we obtain an $O^*(2^{4k/3})$ algorithm improving the previous $O^*(4^k)$ [18] algorithm. We further improve this to $O^*(2^k)$ algorithm using better branching technique. Then we consider a variant of MAXDAG known as "ABOVE GUARANTEE MAXDAG" problem. Here, the problem is to determine whether an oriented graph having m arcs has an acyclic subgraph with at least $m/2 + k$ arcs. We give an $O^*(16^k)$ parameterized algorithm for this problem, improving the previous known bound of $O^*(2^{k^2})$ [18]. Finally, we show that the DIRECTED FEEDBACK ARC SET (DFAS), where given a directed graph $G = (V, A)$ the problem is to determine whether there exists k arcs whose deletion makes the graph acyclic, is fixed parameter tractable when the graph has minimum out-degree or in-degree at least $f(n)$ (some function of n). The parameterized complexity of DFAS in general directed graphs is a well-known open problem.

We conclude with some remarks and open problems in Section 5. In the rest of the paper, we assume that n

represents the number of vertices and m represents the number of edges, in the given graph.

2. The parameterized MAXCUT problem

The parameterized version of the MAXCUT problem is defined as follows:

MAXCUT. Given a graph $G = (V, E)$ and a positive integer parameter k determine whether there exists a partition of the vertex set into $S \neq \emptyset$ and $V \backslash S$ so that the number of edges with one end point in S and the other in $V \backslash S$ is at least k?

The parameterized MAXCUT problem is one of the well studied problem and can be solved in time $O^*(c^k)$, where c is a constant. After a long race of improvements (e.g., [14,10,17]), the current best algorithm of Prieto [17] takes $O^*(1.4143^k)$ time. Here, we give a simple algorithm for parameterized MAXCUT running in time $O^*(1.2418^k)$.

We need the following known results about exact algorithms for the optimization version of MAXCUT and the lower bound on the size of MAXCUT.

Proposition 1. *(See [13].) Let $G = (V, E)$ be a graph on n vertices and m edges. Then the optimization version of* MAXCUT *problem on G can be solved exactly in time $O^*(2^{m/5.217})$ (with polynomial space) and in time $O^*(2^{m/5.769})$ (with exponential space).*

Proposition 2. *(See [22].) Let $G = (V, E)$ be a graph on n vertices and m edges. Then the optimization version of* MAXCUT *problem on G can be solved exactly in time $O^*(2^{\omega n/3})$ (with exponential space). Here $\omega < 2.376$ is the exponent of the best matrix multiplication algorithm.*

Proposition 3. *(See [16].) Let $G = (V, E)$ be a connected graph on n vertices and m edges then there exists a cut M of G with size at least $\frac{m}{2} + \lceil \frac{n-1}{4} \rceil$, and such a cut can be found in polynomial time.*

Lemma 1. *Let $G = (V, E)$ be a connected graph on n vertices and m edges and let $c = \frac{m}{n}$. Further assume that G has a max-cut of size k. Then,*

$$n \leqslant \frac{4k+1}{2c+1} \quad \text{and} \quad m \leqslant f(c,k) = \frac{(4k+1)c}{2c+1},$$

where $f(c, k)$, when viewed as a function of c alone, monotonically increases with c.

Proof. Since G has a maximum cut of size k, by Proposition 3

$$k \geqslant \frac{m}{2} + \frac{n-1}{4} = \frac{cn}{2} + \frac{n-1}{4} \quad \Rightarrow \quad n \leqslant \frac{4k+1}{2c+1}$$

from which it follows that $m \leqslant (4k+1)c/(2c+1)$. It is easy to see that the function $f(c, k)$ monotonically increases with c. □

Lemma 1 gives us the following kernelization lemma.

Lemma 2. *Let $(G = (V, E), k)$ be an instance for* MAXCUT *where G is a connected graph on n vertices and m edges. Then if (G, k) is an "yes" instance then there exist $\alpha \leqslant 1$ and $\beta \leqslant 2$ such that $m \leqslant \beta k + 1$ and $n \leqslant \alpha k + 1$.*

Proof. By Proposition 3, we know that every connected graph has a cut of size at least $\frac{m}{2} + \lceil \frac{n-1}{2} \rceil$ and a cut of this size can be found in polynomial time. So if $k \leqslant m/2$ then the answer to the instance of MAXCUT is always yes. So, we assume that $k \geqslant m/2$. Prieto [17] has given a kernel for MAXCUT with at most k vertices and at most $2k$ edges. Now by Lemma 1 we know that $n \leqslant \frac{4k+1}{2c+1}$ where $c = \frac{m}{n}$. These two together allow us to choose $\alpha = \min\{1, \frac{4}{2c+1}\}$. By Lemma 1 we also know that $m \leqslant f(c, k) = \frac{(4k+1)c}{2c+1}$. Hence we can choose $\beta = \frac{4c}{2c+1}$. □

Given the problem instance $(G = (V, E), k)$ for parameterized MAXCUT we use Propositions 1 or 2 to solve the problem. Using the bounds on m and n obtained in Lemma 1 in terms of k and $c = \frac{m}{n}$, we get the following time complexity for the MAXCUT algorithm.

$$\min\left\{O^*\left(2^{\frac{(4k+1)c}{5.769(2c+1)}}\right), O^*\left(2^{\frac{\omega(4k+1)}{3(2c+1)}}\right)\right\}.$$

For a fixed k the minimum is obtained by setting $c = 4.6$ and the bound is $O^*(2^{0.3124k})$ which is $O^*(1.242^k)$. This gives us the following theorem.

Theorem 1. *Given a graph $G = (V, E)$ and a positive integer k, we can determine whether there exists* MAXCUT *of size at least k (and find one if exists) in $O^*(1.2418^k)$ time.*

3. Parameterized MAXDAG problem and its above guarantee version

In this section we first develop an exact algorithm for the optimization version of MAXDAG. DIRECTED FEEDBACK ARC SET problem is the dual of the optimization version of MAXDAG problem. More precisely the problem statement is:

DIRECTED FEEDBACK ARC SET (DFAS). Given a directed graph $G(V, A)$, find a set of arcs $F \subseteq A$ of minimum size such that $G - F$ is a directed acyclic graph.

Given an ordering π of vertices of G, if there is an arc (i, j) such that $\pi(i) < \pi(j)$ then we call it a forward arc, else we call it a backward arc. Given a directed graph $G = (V, A)$, DFAS can also be viewed as linear ordering of vertices such that the number of backward arcs is minimized. Or equivalently:

DIRECTED FEEDBACK ARC SET (DFAS). Given a directed graph $G(V, A)$, find a permutation $\pi : V \to \{1, 2, \ldots, |V|\}$ such that $\sum_{(e=(u,v)\in A,\ \pi(u)>\pi(v))} \mathbf{1}$ is minimized.

Given a permutation π, let $s(\pi)$ denote the number of backward arcs with respect to the permutation π. Our dynamic programming algorithm is based on the fact that minimum feedback arc set possess an optimal substructure. Given a graph $G = (V, E)$, for a subset $V' \subseteq V$, by $G[V']$ we mean the subgraph of G induced on V'. Let $X(S)$ denote the number of backward arcs in an optimal ordering minimizing backward arcs on the graph $G[S]$ where $S \subseteq V$. Then $X(S)$ is recursively defined as follows:

$$X(S) = \min_{u\in S}\left\{X(S-u) + \sum_{((u,v)\in A\ \&\ v\in(S-u))} \mathbf{1}\right\}. \qquad (1)$$

The correctness of the above recurrence is easy to verify. From now on by the phrase *optimal permutation* we mean a permutation π such that the number of backward arcs is minimized.

3.1. Optimization version

Now we give our algorithm FASD to find a minimum sized directed feedback arc set. The detailed algorithm is presented in Fig. 1.

We have a multi-dimensional array Y of size $2^n \times 2$ which for every subset $S \subseteq V$ stores the value of an optimum permutation and a set of vertices which are possible last vertices of optimal permutations for the induced graph $G[S]$. More precisely, given a subset $S \subseteq V$, we have:

- $Y[S, 1] = X(S)$,
- $Y[S, 2] = \{v \mid v \in V$ such that $X(S)$ is minimized in Eq. (1)$\}$ or set of vertices which are possible last vertices in optimal permutation for $G[S]$.

Algorithm FASD(G)
Input: A directed graph G.
Output: Size of a minimum feedback arc set of G.

Step 1: Let Y be a $2^n \times 2$ multi-dimensional array indexed from 0 to $2^n - 1$, initialized to $Y[S, 1] = \infty$, $Y[S, 2] = 0$ for all subsets $S \subseteq V$ and $S \neq \emptyset$. $Y[\emptyset, 1] = Y[\emptyset, 2] = 0$.
Step 2: for $S \subseteq V$ enumerated in increasing order of cardinality do
  Step 3: For every vertex $u \in V - S$:
    Let $P = Y[S, 1] + \sum_{((u,v)\in A\ \&\ v\in(S-u))} \mathbf{1}$.
    Step 4a: If $P = Y[S \cup \{u\}, 1]$ then
      (i) $Y[S \cup \{u\}, 2] = Y[S \cup \{u\}, 2] \cup \{u\}$.
    Step 4b: If $P < Y[S \cup \{u\}, 1]$ then
      (i) $Y[S \cup \{u\}, 1] = P$.
      (ii) $Y[S \cup \{u\}, 2] = u$.
Step 5: return $Y[V, 1]$.

Fig. 1. Exact algorithm for finding a minimum size feedback arc set in a directed graph.

The correctness of the algorithm follows from Eq. (1). To see the time complexity of the algorithm observe that for every subset $S \subseteq V$ the algorithm takes $O(n)$ time. This gives the following theorem:

Theorem 2. *Let $G = (V, E)$ be a directed graph with n vertices and m arcs. Then the size of a minimum feedback arc set in G can be found in $O^*(2^n)$ time and $O^*(2^n)$ space.*

In fact we can also count all optimal permutations in the same time as finding the size of an optimal permutation by keeping an extra entry $Y[S, 3]$ for every $S \subseteq V$ and making some simple modifications in algorithm FASD. $Y[S, 3]$ stores the number of optimal permutation for $G[S]$. Initialize $Y[S, 3] = 0$ for all $S \subseteq V$ and $S \neq \emptyset$ and $Y[\emptyset, 3] = 1$. Whenever $P = Y[S \cup \{u\}, 1]$ in Step 4a of FASD then do $Y[S \cup \{u\}, 3] = Y[S \cup \{u\}, 3] + Y[S, 3]$ and if $P < Y[S \cup \{u\}, 1]$ in Step 4b of FASD then do $Y[S \cup \{u\}, 3] = Y[S, 3]$. The value in $Y[V, 3]$ gives us the total number of optimal permutations for G.

Theorem 3. *Let $G = (V, E)$ be a directed graph with n vertices and m arcs. Then we can count the number of minimum sized feedback arc sets in G in $O^*(2^n)$ time and $O^*(2^n)$ space.*

Observe that if we actually want an optimal permutation then we can obtain this by following the values stored at $Y[S, 2]$. We start from $Y[V, 2]$, which stores a set of vertices which are last vertices of optimal permutations for G, and trace back following the list stored in $Y[S, 2]$. Suppose we want to enumerate all optimal permutations. We can do this recursively by trying every

vertex v in $Y[V, 2]$ as the last vertex of an optimal permutation and then looking for an optimal permutations of $G[V - \{v\}]$. Observe that after we have filled the array in the algorithm $\mathsf{FASD}(G)$, we can enumerate all optimal permutations of G in polynomial delay.

Theorem 4. *Let $G = (V, E)$ be a directed graph with n vertices and m arcs. Then all the permutations π of V corresponding to minimum size feedback arc sets in G can be enumerated in $O^*(2^n + Z)$ time where Z is the total number of such permutations for G. After initial $O^*(2^n)$ time to find an optimal permutation, every other optimal permutation is enumerated after polynomial delay.*

In Theorems 6 and 3, we used an array of size $O^*(2^n)$ time to find an optimal permutation for G. We can reduce the exponential space requirement to find a minimum size feedback arc set or to count all minimum size feedback arc sets to polynomial space at the expense of increased running time. The usual trick is to apply divide and conquer paradigm to reduce the space requirement. It has been used in [11,1] for other problems.

The idea is to guess the middle vertex $v \in V$ of the optimal permutation and recurse on all possible partitions P_1, P_2 of $V - \{v\}$ such that $\|P_1| - |P_2\| \leqslant 1$. Observe that there are at most 2^n such partitions and given a partition P_1, P_2 either P_1 or P_2 could be left of v. Hence we have at most 2^{n+1} possible legal partitions. This gives us the following recurrence for the polynomial space algorithm:

$$T(n) = 2^{n+1} n^{O(1)} T\left(\frac{n}{2}\right).$$

This recurrence solves to $O^*(4^n n^{O(\log n)})$ which gives us the following theorem:

Theorem 5. *Let $G = (V, E)$ be a directed graph with n vertices and m arcs. Then the size of a minimum feedback arc set and total number of minimum size feedback arc sets in G can be found in $O^*(4^{n+o(n)})$ time and polynomial space.*

We remark that Theorems 6, 4 and 5 can be generalized for weighted case where every arc has been assigned a positive real weight and the objective is to find a maximum weight arc induced acyclic subgraph or to count or enumerate all the maximum weight arc induced acyclic graphs. We state the following theorem without proof.

Theorem 6. *Let $G = (V, E)$ be a directed graph with n vertices and m arcs and let w be a weight function $w : A \to \mathbb{R}^+$. Then*

(i) *a minimum weight feedback arc set can be found in $O^*(2^n)$ time and $O^*(2^n)$ space;*
(ii) *total number of minimum weight feedback arc set can be counted in $O^*(2^n)$ time and $O^*(2^n)$ space;*
(iii) *all the minimum weight feedback arc set can be enumerated in $O^*(2^n + Z)$ time where Z is the total number of minimum weight feedback arc set of G. After initial $O^*(2^n)$ time to find a minimum weight feedback arc set, every other minimum weight feedback arc set is enumerated after polynomial delay.*

Now we give various applications of Theorem 6 in obtaining improved parameterized algorithms for variants of MAXDAG.

3.2. Parameterized MAXDAG

The parameterized version of MAXDAG is defined as follows:

MAXDAG. Given a directed graph $G = (V, A)$ and a positive integer parameter k, determine whether there exists a set of at least k arcs $D \subseteq A$ such that $G' = (V, D)$ is a directed acyclic graph.

In this section, we first find a kernel for MAXDAG as a function of n and then give several applications of the algorithm for parameterized versions of MAXDAG.

3.2.1. Kernel for MAXDAG

Given a graph $G = (V, A)$, we preprocess it by doing the following steps:

(R0) If $k \leqslant |A|/2$ then answer YES.
(R1) Remove vertices of indegree or outdegree 0.
(R2) Let v be a vertex of indegree = outdegree = 1 having u as the inneighbor and w as its outneighbor. Then remove v and add the arc (u, w).

By Lemma 9 of [18], we know that every directed graph has an acyclic subgraph of size at least $\lceil |A|/2 \rceil$ and can be obtained in polynomial time. This ensures the soundness of (R0). The soundness of (R1) and (R2) follows from following lemma which is easy to show.

Lemma 3. *Let $G = (V, A)$ be a directed graph and $G' = (V', A')$ be the directed graph obtained after applying one reduction step. Then*

(1) *if we apply* (R1) *on v then G has an acyclic subgraph of size k if and only if G' has an acyclic subgraph of size $k - \deg(v)$. Here the* $\deg(v)$ *is either an indegree or an outdegree of v depending on whether v is a vertex of outdegree* 0 *or indegree* 0.
(2) *if we apply* (R2) *on v then G has an acyclic subgraph of size k if and only if G' has an acyclic subgraph of size $k - 1$.*

Lemma 3 gives us the following kernelization lemma.

Lemma 4. *Let $(G = (V, A), k)$ be an "yes" instance of* MAXDAG *and $(G' = (V', A'), k')$ be the reduced instance of $(G = (V, A), k)$ after applying the rules* (R0)–(R2) *until no longer possible. Then $|V'| \leqslant 4k'/3$.*

Proof. Observe that when we can not apply reduction rules (R1) and (R2) then every vertex in G' has total degree (indegree + outdegree) at least 3. This implies that $m \geqslant 3|V'|/2$ and (R0) implies that $m \leqslant 2k'$. Combining these two inequalities we get $|V'| \leqslant 4k'/3$. □

3.2.2. Parameterized algorithm for MAXDAG

The following corollary follows from Theorem 6 and Lemma 4.

Corollary 1. *Let $G = (V, A)$ be a directed graph and k be a positive integer then we can determine whether G has an acyclic subgraph of size k or not in $O^*(2^{4k/3}) = O^*(2.5198^k)$.*

Corollary 1 already improves the previous result of $O^*(4^k)$ presented in [18]. We further improve this to $O^*(2^k)$ by obtaining a kernel with at most k vertices after a branching technique.

We give two more reduction rules.

(R3) Let v be a vertex of indegree 1 and outdegree r (> 1) having u as the inneighbor and $w_1, w_2, \ldots, w_r$ as its outneighbors. Then remove v and add arcs (u, w_i), $1 \leqslant i \leqslant r$.
(R4) Let v be a vertex of outdegree 1 and indegree r (> 1) having w as the outneighbor and $u_1, u_2, \ldots, u_r$ as its inneighbors. Then remove v and add arcs (u_i, w), $1 \leqslant i \leqslant r$.

Lemma 5. *Let $G = (V, A)$ be a directed graph and v be a vertex of indegree* 1 (*or outdegree* 1). *Let G' be the graph obtained from G by applying reduction rule* (R3) [(R4)]. *Then, G has an acyclic subgraph of size k containing $(u, v)[(v, w)]$ if and only if G' has an acyclic subgraph of $k - 1$.*

Algorithm MADS(G, k, D)
Input: A directed graph $G = (V, A)$.
Output: A subset of arcs $D \subseteq A$ of size at least k such that the induced subgraph on D is acyclic if exists or NO otherwise.

Step 0: If $k \leqslant |A|/2$ then find the set of arcs of size $|A|/2$ forming an acyclic subgraph in polynomial time and return it as D.
Step 1: Obtain a graph (G', k') by applying reduction rules (R1) and (R2) recursively on (G, k). Now $k \leftarrow k'$.
Step 2: If there exists a vertex v of indegree 1 with inneighbor u then branch as in Steps 2a and 2b and return the solution of larger size.
  Step 2a: $D \leftarrow D \cup \{(u, v)\}$. Apply (R3) on G' and call MADS$(G', k - 1, D)$.
  Step 2b: $D \leftarrow D \cup \{(v, w_i) \mid (v, w_i) \in A\}$. Call MADS$(G - \{v\}, k - \text{outdeg}(v), D)$.
Step 3: If there exists a vertex v of indegree 1 with outneighbor w then branch as in Steps 3a and 3b and return the solution of larger size.
  Step 3a: $D \leftarrow D \cup \{(u, v)\}$. Apply (R4) on G' and call MADS$(G', k - 1, D)$.
  Step 3b: $D \leftarrow D \cup \{(w_i, v) \mid (w_i, v) \in A\}$. Call MADS$(G - \{v\}, k - \text{indeg}(v), D)$.
Step 4: If $|V'| > k$ then return NO else apply Theorem 6 on G and obtain a D and return it.

Fig. 2. Improved parameterized algorithm for MAXDAG.

Proof. Let $D \subseteq A$ of size at least k containing (u, v) such that $G(V, D)$ is acyclic. Let $B = \{(u, v), (v, w_1), \ldots, (v, w_l)\}$ be the set of arcs in D which contain v as one of its endpoint. Then take $D' = D - B + \{(u, w_1), \ldots, (v, w_l)\}$ as an acyclic subgraph of size at least $k - 1$ for $G' = G - \{v\}$.

Let D' be the subset of arcs of size at least $k - 1$ of $G' = G - \{v\}$ such that it forms an acyclic subgraph of G'. Consider the topological ordering of $G''(V - \{v\}, D')$. Now place the vertex v on the right side of u. Let $X = \{(u, w_i) \mid (u, w_i) \in D', w_i$ outneighbor of $v\}$. Now take $D = D' - X + \{(v, w_i) \mid (u, w_i) \in X\} + (u, v)$ as an acyclic subgraph of size k containing (u, v) in G. □

We conjure all that we have developed so far and use it to give an improved algorithm for parameterized MAXDAG. The algorithm is given in Fig. 2.

Now we argue about the correctness and time complexity of the algorithm. Correctness of Steps 0 and 1 is clear. In Steps 2 and 3 we branch on the arc (u, v) and (v, w), respectively. In each of these steps we further branch on two cases that is either $(u, v) \in D$ or

$(u, v) \notin D$ and return the larger size solution. Correctness of Steps 2a and 3a follows from Lemma 5. In Steps 2b and 3b, we are looking for an acyclic subgraph without (u, v) or (v, w). This implies that (u, v) or (v, w) is part of the directed feedback arc set and hence can be deleted from G which makes v either a vertex of indegree or outdegree 0. This implies that there exists a solution D of size k containing all out arcs or in arcs of v.

Observe that when we apply either Step 2 or Step 3 the total degree of every vertex v is at least 3. Hence, after we branch in Steps 2a and 3a the parameter k reduces by 1 while in Steps 2b and 3b the parameter k at least reduces by 2. This gives the following recurrence on the parameter k:

$$T(k) \leqslant T(k-1) + T(k-2). \tag{2}$$

When we reach Step 4 of the algorithm then every vertex of G has indegree as well as outdegree at least 2 and hence $|A| \geqslant 2|V|$. By Step 0 we know that $k \geqslant |A|/2$. This gives us that

$$2|V| \leqslant |A| \leqslant 2k \quad \Rightarrow \quad |V| \leqslant k.$$

This implies that we have obtained a graph G with at most k vertices and $2k$ arcs. Now we apply Theorem 6 on G and solve the maximum acyclic subgraph problem in $O^*(2^k)$ time. The recurrence 2 solves to $O^*(1.62^k)$ and hence the running time of the algorithm is bounded by $O^*(2^k)$. This gives us the following theorem:

Theorem 7. *Let $G = (V, A)$ be a directed graph and k be a positive integer. We can determine whether G has an acyclic subgraph of size k or not in $O^*(2^k)$ time.*

3.3. Above guarantee MAXDAG

Now we apply Theorem 6 to obtain an improved parameterized algorithm for a special version of MAXDAG.

We call a directed graph *oriented directed graph* if there is at most one directed arc between every pair of vertices. The following proposition was shown in [18] about the size of maximum acyclic subgraph in an oriented directed graph.

Proposition 4. *(See [18].) Any oriented directed graph $G = (V, E)$ with m arcs and n vertices, with the underlying undirected graph having c components, has an acyclic subgraph with at least $\frac{m}{2} + 1/2\lceil (n-c)/2 \rceil$ arcs and such a subgraph can be found in $O(n^3)$ time.*

This was used to give a fixed parameter algorithm of time complexity $O^*(2^{k^2})$ for the question 'whether the given oriented directed graph has a set of at least $\frac{m}{2} + k$ arcs that forms an acyclic subgraph'. Call this problem ABOVE GUARANTEE MAXDAG.

Here we give an algorithm of time complexity $O^*(16^k)$ as an application of Theorem 6. First, find all the c components of the underlying undirected graph corresponding to G. If $k \leqslant 1/2\lceil (n-c)/2 \rceil$, then G has an acyclic subgraph with at least $\frac{m}{2} + k$ arcs, else $k > 1/2\lceil (n-c)/2 \rceil \geqslant (n-c)/4 - 1$ or $n \leqslant 4k + 4 + c$. Thus n_i, the number of vertices in the ith component is at most $n - (c-1) \leqslant 4k + 5$. So now apply the Theorem 6 on every connected component containing at most $4k + 5$ vertices. This gives us the following theorem:

Theorem 8. *Let G be an oriented directed graph on n vertices and m arcs. Then given an integer k, we can determine whether or not G has at least $\frac{m}{2} + k$ arcs which forms an acyclic subgraph in $O^*(16^k)$ time.*

3.4. Directed graphs with minimum outdegree $f(n)$

In this section we show that if the minimum outdegree or indegree of a graph is at least $f(n)$, for any function of n, then the directed feedback arc set problem is fixed parameter tractable in such graphs. Here $f(n)$ could be as slow growing function as $\log^* n$. Our algorithm depends on the following combinatorial lemma which relates the size of a minimum feedback arc set and the minimum outdegree or indegree of the graph.

Lemma 6. *Let $G = (V, A)$ be a directed graph with minimum outdegree t (or minimum indegree t). Then any feedback arc set of G must contain at least $\binom{t+1}{2}$ arcs.*

Proof. Without loss of generality, assume that the minimum outdegree of the graph is at least t. Let F be any feedback arc set of G. Then by definition of F, $G - F$ is a directed acyclic graph and hence there exists a topological ordering of $G - F$. Let this ordering of vertices be $P = v_1, v_2, \ldots, v_n$. We know that for every arc (u, v) of $G - F$, u occurs before v in the ordering P. Let S be the set of the last t vertices of P. Since v_n has out degree 0 in $G - F$, all arcs out of v_n in G are part of F. Similarly for v_{n-k}, $0 \leqslant k \leqslant t-1$, at least $t - k$ arcs emanating from v_{n-k} in G are part of F. Hence

$$|F| \geqslant \sum_{k=0}^{t-1} (t-k) = \binom{t+1}{2}.$$

This shows that any minimum feedback arc set of G must contain at least $\binom{t+1}{2}$ arcs. □

Lemma 6 directly gives a fixed parameter tractable algorithm for the directed feedback arc set problem in graphs with minimum outdegree or indegree $f(n)$ by kernelization technique. Given a graph G with minimum outdegree or indegree $f(n)$ and a positive integer k, we check whether $k < \binom{f(n)+1}{2}$. If $k < \binom{f(n)+1}{2}$ then we return NO. Otherwise $k \geqslant \binom{f(n)+1}{2}$ and hence $n \leqslant g(k)$ for some function g which is a function of k alone. Now we apply Theorem 6 and solve the parameterized directed feedback arc set problem in the time $O^*(2^{g(k)})$. This gives the following theorem.

Theorem 9. *Let $G = (V, A)$ be a directed graph such that the minimum outdegree or indegree of the graph is at least $f(n)$, for some fixed function f. Then the directed feedback arc set problem is fixed parameter tractable in G.*

4. Conclusion

We have given improved parameterized algorithms for MAXCUT and MAXDAG through new kernelization and efficient exact algorithms for their optimization versions. For both these "edge" problems we obtain an improved kernel by bounding n as a function of k which is crucial for the improved algorithms. It will be interesting to find some other "edge" problems where bounding n, the number of vertices, as a function of k and then using the exact algorithms for the optimization version of these problems gives an improved parameterized algorithm.

References

[1] A. Björklund, T. Husfeldt, Exact algorithms for exact satisfiability and number of perfect matchings, in: The Proceedings of International Colloquium on Automata, Languages and Programming (ICALP), in: Lecture Notes in Comput. Sci., vol. 4051, 2006, pp. 548–559.

[2] J.M. Byskov, Enumerating maximal independent sets with applications to graph colouring, Operations Research Letters 32 (6) (2004) 547–556.

[3] R. Downey, M. Fellows, Parameterized Complexity, Springer-Verlag, 1999.

[4] D. Eppstein, 3-coloring in time $O(1.3289^n)$, Journal of Algorithms 54 (2) (2005) 168–204.

[5] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer-Verlag, 2006.

[6] F.V. Fomin, F. Grandoni, D. Kratsch, Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm, in: The Proceedings of ACM–SIAM Symposium on Discrete Algorithms (SODA), 2006, pp. 18–25.

[7] F.V. Fomin, F. Grandoni, D. Kratsch, Some new techniques in design and analysis of exact (exponential) algorithms, Bulletin of the European Association for Theoretical Computer Science 87 (2005) 47–77.

[8] F.V. Fomin, F. Grandoni, D. Kratsch, Measure and conquer: domination—a case study, in: The Proceedings of International Colloquium on Automata, Languages and Programming (ICALP), in: Lecture Notes in Comput. Sci., vol. 3580, 2005, pp. 191–203.

[9] F.V. Fomin, F. Grandoni, A.V. Pyatkin, A.A. Stepanov, Bounding the number of minimal dominating sets: a measure and conquer approach, in: The Proceedings of International Symposium on Algorithms and Computation (ISAAC), in: Lecture Notes in Comput. Sci., vol. 3827, 2005, pp. 573–582.

[10] J. Gramm, E.A. Hirsch, R. Niedermeier, P. Rossmanith, Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT, Discrete Applied Mathematics 130 (2) (2003) 139–155.

[11] Y. Gurevich, S. Shelah, Expected computation time for Hamiltonian path problem, SIAM Journal on Computing 16 (3) (1987) 486–502.

[12] K. Iwama, S. Tamaki, Improved upper bounds for 3-SAT, in: The Proceedings of ACM–SIAM Symposium on Discrete Algorithms (SODA), 2004, p. 328.

[13] J. Kneis, D. Mölle, S. Richter, P. Rossmanith, Algorithms based on the treewidth of sparse graphs, in: The Proceedings of Workshop on Graph-Theoretic Concepts in Computer Science (WG), in: Lecture Notes in Comput. Sci., vol. 3787, 2005, pp. 385–396.

[14] M. Mahajan, V. Raman, Parameterizing above guaranteed values: MaxSat and MaxCut, Journal of Algorithms 31 (2) (1999) 335–354.

[15] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford Lecture Series in Mathematics and its Applications, Oxford University Press, 2006.

[16] S. Poljak, D. Turzik, A polynomial algorithm for constructing a large bipartite subgraph, with an application to a satisfiability problem, Canad. J. Math. 34 (3) (1982) 519–524.

[17] E. Prieto, The method of extremal structure on the k-maximum cut problem, in: The Proceedings of Computing: The Australasian Theory Symposium (CATS), 2005, pp. 119-126.

[18] V. Raman, S. Saurabh, Parameterized algorithms for feedback set problems and their duals in tournaments, Theoretical Computer Science 351 (3) (2006) 446–458.

[19] V. Raman, S. Saurabh, S. Sikdar, Improved exact exponential algorithms for vertex bipartization and other problems, in: The proceedings of Italian Conference on Theoretical Computer Science (ICTCS), in: Lecture Notes in Comput. Sci., vol. 3701, 2005, pp. 375–389.

[20] V. Raman, S. Saurabh, S. Sikdar, Efficient exact algorithms through enumerating maximal independent sets and other techniques, Theory of Computing Systems, Springer-Verlag.

[21] J.M. Robson, Finding a maximum independent set in time $O^*(2^{n/4})$?, Technical Report 1251-01, LaBRI, Université Bordeaux I, 2001.

[22] R. Williams, A new algorithm for optimal 2-constraint satisfaction and its implications, in: The Proceedings of International Colloquium on Automata, Languages and Programming (ICALP), in: Lecture Notes in Comput. Sci., vol. 3142, 2004, pp. 1227–1237.

[23] G. Woeginger, Exact algorithms for NP-hard problems: A survey, in: Combinatorial Optimization—Eureka! You Shrink! in: Lecture Notes in Comput. Sci., vol. 2570, 2003, pp. 185–207.