Discrete Mathematics and Theoretical Computer Science DMTCS vol. **14:2**, 2012, 189–204

Digraph Complexity Measures and Applications in Formal Language Theory†

Hermann Gruber‡

knowledgepark AG, München, Germany

received 29th January 2012, revised 12th July 2012, accepted 26th October 2012.

We investigate structural complexity measures on digraphs, in particular the cycle rank. This concept is intimately related to a classical topic in formal language theory, namely the star height of regular languages. We explore this connection, and obtain several new algorithmic insights regarding both cycle rank and star height. Among other results, we show that computing the cycle rank is $\mathbf{NP}$-complete, even for sparse digraphs of maximum outdegree 2. Notwithstanding, we provide both a polynomial-time approximation algorithm and an exponential-time exact algorithm for this problem. The former algorithm yields an $O((\log n)^{3/2})$-approximation in polynomial time, whereas the latter yields the optimum solution, and runs in time and space $O^*(1.9129^n)$ on digraphs of maximum outdegree at most 2.

Regarding the star height problem, we identify a subclass of the regular languages for which we can precisely determine the computational complexity of the star height problem. Namely, the star height problem for bideterministic languages is $\mathbf{NP}$-complete, and this holds already for binary alphabets. Then we translate the algorithmic results concerning cycle rank to the bideterministic star height problem. We thus obtain a polynomial-time approximation algorithm, as well as a reasonably fast exact exponential algorithm for the bideterministic star height problem.

Keywords: regular expression, star height, digraph, cycle rank, ordered coloring, vertex ranking

MSC: 68Q45 (primary), 68Q25, 05C20 (secondary)
ACM CCS: F4.3 Formal Languages, F2.2 Nonnumerical Algorithms and Problems, G2.2 Graph Theory

1 Introduction

In the theory of undirected graphs, structural complexity measures for graphs, such as treewidth and pathwidth, have gained an important role, both from a structural and an algorithmic viewpoint, see e.g. [11, 14]. However, networks arising in some domains are more adequately modeled as having directed edges. Therefore in recent years, attempts have been made to lift such measures and parts of the

†An early version of this paper was presented at the 4th Workshop on Mathematical and Engineering Methods in Computer Science (MEMICS 2008), see [17].

‡Email: `info@hermann-gruber.com`. Part of the work was done while the author was at Institut für Informatik, Justus-Liebig-Universität Giessen, Germany.

1365–8050 © 2012 Discrete Mathematics and Theoretical Computer Science (DMTCS), Nancy, France

theory of undirected graphs to the case of digraphs. Several recent works show that, while there often exist partial analogues to the undirected case, the picture for digraphs is much more involved [5, 6, 15, 27, 35]. We discuss some of these measures, relate them to each other, and investigate their algorithmic aspects. Interestingly, we are able to show that all these complexity measures bound each other within a factor logarithmic in the order of the digraph, thus paralleling the case of undirected graphs [9]. We focus in particular on the cycle rank, a digraph complexity measure originally motivated by studies in formal languages [12]. Apparently, there is a renewed interest in this measure, as witnessed by recent research efforts [2, 4, 15, 23, 28].

We obtain the following results on computing the cycle rank: The decision version of the problem is $\mathbf{NP}$-complete, and this remains true for graphs of maximum outdegree at most 2. Previously, the problem was known to be $\mathbf{NP}$-complete on undirected symmetric digraphs of unbounded degree, see [8]. On the positive side, we design a polynomial-time $O((\log n)^{3/2})$-approximation algorithm, as well as an exact exponential algorithm algorithm computing the cycle rank of digraphs. If the given digraph is of bounded outdegree, the latter algorithm runs in time and space $O^*((2-\varepsilon)^n)$, where n is the order of the digraph, and ε is a constant depending on the maximum outdegree.

For unbounded outdegree, the running time is still $O^*(2^n)$, whereas for maximum outdegree 2, we even attain a bound of $O^*(1.9129^n)$. As a further application, we also obtain an exact algorithm for the directed feedback vertex set problem on digraphs of maximum outdegree 2, which runs within the same time bound.

Then we present applications to the theory of regular expressions. The star height of a regular language is defined as the minimum nesting depth of stars needed in order to describe that language by a regular expression. Already in the 1960s, Eggan [12] raised the question whether the star height can be determined algorithmically. It was not until 25 years later that Hashiguchi found a rather complicated decidability proof [20]. Even today, the best known algorithm has doubly exponential running time, and is arguably still impractical [26]. Therefore, we study the complexity of the star height problem when restricted to a subclass of the regular languages. We show that the star height problem for bideterministic languages is $\mathbf{NP}$-complete, and this remains true when restricted to binary alphabets. Furthermore, we present both an efficient approximation algorithm and an exact exponential algorithm for this problem. The key to these results are the corresponding algorithms for the cycle rank of digraphs mentioned above; also the above mentioned bounds carry over to this application in formal language theory.

The paper is organized as follows: After this introduction, we recall in Section 2 some basic notions from graph theory and from automata theory. We study structural properties of the cycle rank of digraphs in Section 3. Section 4 is devoted to algorithmic aspects of cycle rank. Afterward, we apply these findings in Section 5 to the star height problem on bideterministic languages. We complete the paper in Section 6 by showing possible directions for further research.

2 Preliminaries

2.1 Asymptotic Notation

Recall that for two functions $f(n)$ and $g(n)$, we write $f(n) = O(g(n))$, if there exists a positive constant c such that for all large enough n holds $f(n) < c \cdot g(n)$. The O^*-notation (also: "soft-O-notation") is a similar notation that is often used in exponential algorithmics [14, 38]. In contrast to O-notation, the latter suppresses not only constant factors, but also larger negligible factors, e.g., polynomial factors accompanying an exponential. More precisely, for functions $f(n)$ and $g(n)$, we write $f(n) = O^*(g(n))$ if $f(n) = O\left(g(n) \cdot (\log g(n))^k\right)$ for some nonnegative constant k. For example, we have $2^n \cdot n^3 = O^*(2^n)$.

2.2 Digraphs

We assume familiarity with basic notions in graph theory, as contained in [11], so we only fix the notation and give a few specialized definitions below. A *digraph* $G = (V, E)$ consists of a finite set of *vertices* V and a set of *edges* $E \subseteq V^2$.

We refer to an edge of the form (v, v) as a *loop*; A digraph without loops is called *loop-free*.

The *outdegree* of a vertex v is defined as the number of vertices u such that $(u, v) \in E$. The *total degree* of v is defined as the number of distinct vertices u having $(u, v) \in E$ or $(v, u) \in E$.

If the edge relation of a digraph G is symmetric, we say G is an (undirected) *graph*. By taking the symmetric closure of the edge relation of a digraph, we obtain its undirected counterpart—of course, this is a many-to-one correspondence.

For a subset of vertices $U \subseteq V$, let $G[U]$ denote the sub(di)graph *induced by* U, which is obtained by restricting the vertex set of G to U and redefining the edge set E appropriately. In this context, we will often use $G - U$ as a shorthand for $G[V \setminus U]$ and $G - v$ for $G[V \setminus \{v\}]$. A subset of vertices $U \subseteq V$ is *strongly connected* if for every pair of distinct vertices u and v in U, there is both a path from u to v and a path from v to u, and both of these paths visit only vertices from U. Maximal strongly connected subsets of V are called *strongly connected components*; a strongly connected subset S is *nontrivial* if the subdigraph $G[S]$ induced by S contains at least one edge (note that this includes the case $S = \{v\}$ if v has a loop). A digraph is *acyclic* if all of its strongly connected components are trivial.

2.3 Formal Languages

As with digraphs, we only recall some basic notions in formal language and automata theory—for a thorough treatment, the reader might want to consult a textbook such as [22]. In particular, let Σ be a finite alphabet and Σ^* the set of all words over the alphabet Σ, including the empty word λ. The length of a word w is denoted by $|w|$, where $|\lambda| = 0$. A *(formal) language* over the alphabet Σ is a subset of Σ^*.

The *regular expressions* over an alphabet Σ are defined recursively in the usual way:(i) $\emptyset$, λ, and every letter a with $a \in \Sigma$ is a regular expression; and when r_1 and r_2 are regular expressions, then $(r_1 + r_2)$, $(r_1 \cdot r_2)$, and $(r_1)^*$ are also regular expressions. The language defined by a regular expression r, denoted by $L(r)$, is defined as follows: $L(\emptyset) = \emptyset$, $L(\lambda) = \{\lambda\}$, $L(a) = \{a\}$, $L(r_1 + r_2) = L(r_1) \cup L(r_2)$, $L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$, and $L(r_1^*) = L(r_1)^*$. For a regular expression r over Σ, the *star height*, denoted by $\mathrm{h}(r)$, is a structural complexity measure inductively defined by: $\mathrm{h}(\emptyset) = \mathrm{h}(\lambda) = \mathrm{h}(a) = 0$, $\mathrm{h}(r_1 \cdot r_2) = \mathrm{h}(r_1 + r_2) = \max(\mathrm{h}(r_1), \mathrm{h}(r_2))$, and $\mathrm{h}(r_1^*) = 1 + \mathrm{h}(r_1)$. The star height of a regular language L, denoted by $\mathrm{h}(L)$, is then defined as the minimum star height among all regular expressions describing L.

It is well known that regular expressions are exactly as powerful as finite automata, i.e., for every regular expression one can construct an equivalent (deterministic) finite automaton and *vice versa*, see [22]. Finite automata are defined as follows: A *nondeterministic finite automaton* (NFA) is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite set of input symbols, $\delta : Q \times \Sigma \to 2^Q$ is

(i) For convenience, parentheses in regular expressions are sometimes omitted and the concatenation is simply written as juxtaposition. The priority of operators is specified in the usual fashion: concatenation is performed before union, and star before both product and union.

the transition function, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states. The *language accepted* by the finite automaton A is defined as $L(A) = \{ w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \emptyset \}$, where δ is naturally extended to a function $Q \times \Sigma^* \to 2^Q$. A nondeterministic finite automaton $A = (Q, \Sigma, \delta, Q_0, F)$ is *deterministic*, for short a DFA, if $|\delta(q, a)| \leq 1$, for every $q \in Q$ and $a \in \Sigma$. In this case we simply write $\delta(q, a) = p$ instead of $\delta(q, a) = \{p\}$. Two (deterministic or nondeterministic) finite automata are *equivalent* if they accept the same language.

A deterministic finite automaton is *bideterministic*, if it has a single final state, and if the NFA obtained by reversing all transitions and exchanging the roles of initial and final state is again deterministic—notice that, by construction, this NFA in any case accepts the reversed language. A regular language L is *bideterministic* if there exists a bideterministic finite automaton accepting L. These languages form a proper subclass of the regular languages [3].

3 Cycle Rank of Digraphs

3.1 Cycle Rank and Directed Elimination Forests

Originally suggested in the 1960s by Eggan and Büchi in the course of investigating the star height of regular languages [12], the cycle rank is probably one of the oldest structural complexity measures on digraphs. In this section, we delve into the structural foundations of cycle rank.

Definition 1 *The* cycle rank *of a directed graph $G = (V, E)$, denoted by $\mathrm{r}(G)$, is inductively defined as follows: If G is acyclic, then $\mathrm{r}(G) = 0$. If G is strongly connected and $E \neq \emptyset$, then $\mathrm{r}(G) = 1 + \min_{v \in V} \{ \mathrm{r}(G - v) \}$. If G is not strongly connected, then $\mathrm{r}(G)$ equals the maximum cycle rank among all strongly connected components of G.*

We note that the requirement $E \neq \emptyset$ in the above definition allows to differentiate between acyclic digraphs and (otherwise acyclic) digraphs with loops. We also remark that the cycle rank can be equivalently defined using decompositions, compare [30]:

Definition 2 *A* directed elimination tree *for a nontrivially strongly connected digraph $G = (V, E)$ is a rooted tree $T = (\mathcal{T}, \mathcal{E})$ having the following properties:*

a) $\mathcal{T} \subseteq V \times 2^V$, and if $(x, X) \in \mathcal{T}$, then $x \in X$.

b) The root of the tree is (v, V) for some $v \in V$.

c) There is no pair of distinct vertices of the form (x, X) and (y, X) in the forest.

d) If (x, X) is a node in T, and $G[X] - x$ has $j \geq 0$ nontrivial strongly connected components $Y_1, \ldots, Y_j$, then (x, X) has exactly j children of the form $(y_1, Y_1), \ldots (y_j, Y_j)$ for some $y_1, \ldots, y_j \in V$.

A directed elimination forest *for a digraph G with $k \geq 0$ nontrivial strongly connected components $C_1, \ldots C_k$, is a rooted forest consisting of directed elimination trees for $G[C_i]$, $1 \leq i \leq k$.*

Figure 1 illustrates this concept by an example. It is shown in [30] that the minimum height among all directed elimination forests for G equals the cycle rank of G. Interestingly, the concept of elimination forests was rediscovered in the context of sparse matrix factorization, see [36] for the undirected case and [13] for the directed case.

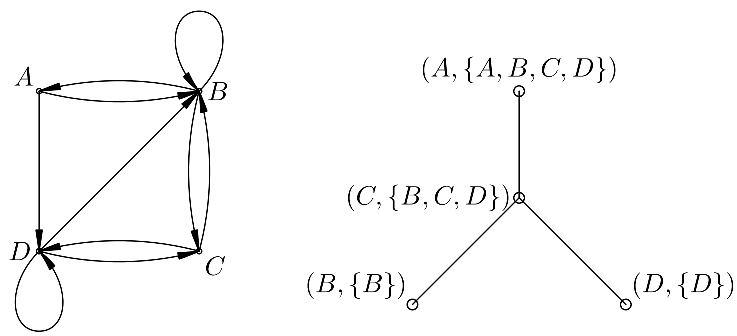

Fig. 1: An example digraph G, and a directed elimination tree for it.

3.2 Cycle Rank and other Digraph Complexity Measures

We compare the cycle rank with two other structural complexity measures, namely weak separator number and directed pathwidth. The first measure is a generalization of separator number (see e.g. [9, 18, 34]) to digraphs:

Definition 3 *Let $G = (V, E)$ be a digraph and let $U \subseteq V$ be a set of vertices. A set of vertices S is a* weak balanced separator *for U if every strongly connected component of $G[U \setminus S]$ contains at most $\left\lceil \frac{1}{2}|U \setminus S| \right\rceil$ vertices. The* weak separator number *of G, denoted by $\mathrm{s}(G)$, is defined as the maximum size, taken over all subsets $U \subseteq V$, among the minimum weak balanced separators for U.*

We illustrate this definition in the following example.

Example 4 *Consider the digraph G in Figure 1. Removing the vertex B results in a digraph with strongly connected components $\{A\}$ and $\{C, D\}$. Since $|\{C, D\}| \leq \left\lceil \frac{1}{2}(4-1) \right\rceil$, the singleton set $\{B\}$ forms a weak balanced separator. Also, each induced subdigraph of G has a weak balanced separator of size 1, so $\mathrm{s}(G) \leq 1$. On the other hand, we must have $\mathrm{s}(G) \geq 1$, since G contains cycles of length greater than 1.*

Some readers will feel that the above definition is a bit contrived because of the ceiling operator $\lceil \cdot \rceil$. But this is an essential detail, as it guarantees that a digraph with a weak balanced separator of size k will always admit a weak balanced separator of size $k + 1$.

In order to relate weak separator number and cycle rank, we need the following recurrence: For integers $k, n \geq 1$, let $R_k(n)$ be given by the recurrence

$$R_k(n) = k + R_k\left(\left\lceil \frac{n-k}{2} \right\rceil\right),$$

with $R_k(r_0) = r_0$ for $r_0 \leq k$.

Lemma 5 *Let G be a loop-free digraph with n vertices and weak separator number at most k. Then $\mathrm{r}(G) \leq R_k(n) - 1$.*

Proof: We generalize a proof given in [18] to the case of digraphs. Let G^ℓ be the digraph obtained from G by adding self-loops to each vertex. Then $\mathrm{r}\big(G^\ell\big) = \mathrm{r}(G) + 1$, so we may prove instead that $\mathrm{r}\big(G^\ell\big) \leq R_k(n)$.

We prove the statement by induction on the order n of G^ℓ. The base cases $n \leq k$ of the induction are easily seen to hold, since the cycle rank of a digraph is always bounded above by its order.

For the induction step, assume $n > k$. As already mentioned, if G^ℓ admits a weak balanced separator of size at most k, then it also has a weak balanced separator of size exactly k. Let X be such a separator. Denote the strongly connected components of $G^\ell - X$ by $C_1, \ldots, C_p$. Then $\mathrm{r}\big(G^\ell\big) \leq k + \mathrm{r}\big(G^\ell - X\big)$, and by definition of cycle rank,

$$\mathrm{r}\big(G^\ell - X\big) \leq \max_{1 \leq i \leq p} \mathrm{r}\big(G^\ell[C_i]\big)\,.$$

As X is a weak balanced separator, we have $|C_i| \leq \lceil \frac{n-k}{2} \rceil$ for $1 \leq i \leq p$. Hence, we can apply the induction hypothesis to obtain

$$\max_{1 \leq i \leq p} \mathrm{r}\big(G^\ell[C_i]\big) \leq R_k(\lceil \frac{n-k}{2} \rceil).$$

Putting these pieces together, we have $\mathrm{r}\big(G^\ell\big) \leq k + R_k(\lceil \frac{n-k}{2} \rceil)$, as desired. □

The recurrence $R_k(n)$ is studied in [18], where also the inequality $R_k(n) \leq k \cdot \log(n/k)$ is derived.(ii) We thus have the following bound:

Corollary 6 *Let G be a loop-free digraph with n vertices and weak separator number at most k. Then* $\mathrm{r}(G) \leq k \cdot \log(n/k) - 1$. □

This inequality is sharp already in the undirected case, see [18]. Previously, a looser bound comparing cycle rank to a similar notion of weak balanced separators was given in [19]. It is easy to see that Corollary 6 improves upon the previous bound.

We turn to the comparison with directed pathwidth. That measure was introduced by Reed, Seymour and Thomas (cf. [5]) as a generalization of pathwidth to digraphs.

Definition 7 *For a digraph $G = (V, E)$, a* directed path decomposition *of G is a sequence $W_1 W_2 \cdots W_r$ of subsets of V, called* bags*, such that*

a) each vertex is contained in at least one bag,

b) $W_i \cap W_k \subseteq W_j$ for all $i < j < k$, and

c) for each edge $(u, v) \in E$, there is a bag containing both endpoints, or there exist i, j with $i < j$ such that the tail u is in W_i and the head v is in W_j.

The width *of a directed path decomposition is defined as the maximum cardinality among all bags minus* 1*. The directed pathwidth is defined as the minimum width among all directed path decompositions for G.*

For algorithmic constructions, it is often useful to work with a certain normal form of directed path decompositions. A directed path decomposition is *normal*, if adjacent bags may differ in at most one vertex, and it is easy to transform a directed path decomposition into a normal one. We return to our running example to illustrate these concepts.

(ii) Here log denotes the binary logarithm.

Example 8 *Consider again the digraph G in Figure 1. The sequence*

$$\{A, B\}\,\{B, C\}\,\{D\}$$

forms a directed path decomposition, and the sequence

$$\{A, B\}\,\{B\}\,\{B, C\}\,\{C\}\,\emptyset\,\{D\}$$

forms a normal directed path decomposition for G. Furthermore, observe that the intersection of the two adjacent bags $\{B\}$ and $\{B, C\}$ forms a weak balanced separator.

One can show in general that, in a normal directed path decomposition, there exists a pair of adjacent bags "somewhere in the middle", whose intersection forms a weak balanced separator.

Without loss of generality, one can assume that, if W_i and W_{i+1} are adjacent bags for $1 \leq i < n$, then $W_i \neq W_{i+1}$, and thus $W_i \cap W_{i+1} \leq \mathrm{dpw}(G)$. Altogether, this yields the following relation between weak balanced separator number and directed pathwidth:

Lemma 9 *Let G be a digraph. Then $\mathrm{s}(G) \leq \mathrm{dpw}(G)$.* □

How does cycle rank relate to directed pathwidth? We can answer this using directed elimination forests.

Lemma 10 *Let G be a digraph. Then $\mathrm{dpw}(G) \leq \mathrm{r}(G)$.*

Proof: We prove by induction that each directed elimination forest of height k for G can be transformed into a directed path decomposition for G of width at most k.

If $k = 0$, then G is acyclic, and thus clearly admits a directed path decomposition of width 0.

For the induction step, assume the directed elimination forest for G has roots (x_1, C_1), (x_2, C_2),..., (x_r, C_r), with the strongly connected components C_i in topological order. Let $G_i = G[C_i] - x_i$. Then G_i has cycle rank at most $k - 1$. By induction assumption, each digraph G_i admits a directed path decomposition of width at most $k-1$. By adding the vertex x_i to each bag in the respective decomposition for G_i, we obtain a directed path decomposition for $G[C_i]$. Concatenating the r individual directed path decompositions while respecting the above topological order, we obtain a directed path decomposition of width at most k for G, as desired. □

Altogether, we have derived the following chain of inequalities:

Theorem 11 *Let G be a loop-free digraph with n vertices and weak separator number k. Then*

$$k \leq \mathrm{dpw}(G) \leq \mathrm{r}(G) \leq k \cdot \log(n/k) - 1.$$

Quite a few more structural complexity measures on digraphs were studied recently, such as directed tree-width, DAG-width, and Kelly-width. As detailed in [25], each of these measures is bounded below by a function that is linear in the weak separator number(iii). On the other hand, all of those are bounded above by the directed pathwidth (cf. [25]), so Theorem 11 will also serve for comparing them with cycle rank, and with weak separator number.

(iii) The notion used in [25] corresponds to our notion of weak separator number up to a constant factor.

4 Computational Aspects of Cycle Rank

4.1 Computational Complexity

We turn to algorithmic questions. First, we classify the computational complexity of the decision problem **CYCLE RANK**: Given a digraph G and an integer k, determining whether the cycle rank of G is at most k.

Theorem 12 *The* **CYCLE RANK** *problem is* $\mathbf{NP}$*-complete, and this still holds when requiring that the input digraph is strongly connected.*

Proof: Membership in $\mathbf{NP}$ can be seen by the equivalent definition using directed elimination forests: Let $G = (V, E)$ denote the given digraph. Every directed elimination forest for G contains at most $|V|$ tree vertices, and each tree vertex is of size is at most $|V|$. A nondeterministic polynomial-time bounded Turing machine can guess such a witness, and then verify that it indeed constitutes a directed elimination forest of height at most k for G.

For $\mathbf{NP}$-hardness, we use a corresponding result known for the undirected case. Given a symmetric loop-free digraph G, it is easy to see (e.g. by [31, Lem. 2.2]) that an undirected elimination forest of height $k + 1$ in the sense of [9, 31] corresponds to a directed elimination forest of height k in our sense (the term $+1$ accounts for the slightly different notion of *height* used in [31]). However, determining the minimum height among all undirected elimination forests is $\mathbf{NP}$-complete, also for (strongly) connected undirected graphs [9]. □

Using tools from formal language theory, we will prove later that $\mathbf{NP}$-hardness still holds for digraphs of maximum outdegree at most 2 and of maximum total degree at most 4.

4.2 Approximate Computation

How do we cope with this negative result? One possibility is to look for an approximate solution. Indeed, it is known that for undirected graphs, the cycle rank problem admits an input-dependent polynomial-time approximation algorithm [9]. In the following, we devise a more general approximation algorithm, which also addresses the case of unsymmetric digraphs. The basic pattern of our algorithm is divide-and-conquer along separators.

Theorem 13 *The* **CYCLE RANK** *problem admits a polynomial-time approximation within a factor of* $O((\log n)^{3/2})$.

Proof: The following recursive procedure computes a directed elimination forest for the induced subgraph $G[W]$, where $W \subset V$ is passed as parameter to the procedure.

If $G[W]$ consists of several strongly connected components, apply the procedure recursively to each of these; The union of these results gives a directed elimination forest for $G[W]$.

Otherwise, use the polynomial-time algorithm from [25, Corollary 2.25] to find a small vertex subset $S \subseteq W$ in $G[W]$ with the property that every strongly connected component of $G[W] - S$ has at most $\frac{3}{4}|W|$ vertices. Then pass the digraph $G[W] - S$ as parameter to the recursive procedure. Upon returning, the directed elimination forest F returned for $G[W] - S$ is then extended, one vertex at a time, for each vertex s from S.

More precisely, put the elements of S in arbitrary order. Then for given s in S, let X denote the set of vertices occurring before s. Assuming we have already computed a directed elimination forest for $G[W \cup X]$, we now show how to extend this to a forest for $G[W \cup X \cup s]$. Initially, the set X is empty, and we proceed for each s until $X = S$. Let $C_1, \dots C_p$ denote those strongly connected components of the digraph $G[W \cup X]$ for which $G[W \cup \{s\} \cup \bigcup_i C_i]$ is strongly connected, and let $D_1, \dots, D_r$ denote the remaining strongly connected components in $G[W \cup X]$. The elimination forest for $G[W \cup X]$ contains an elimination tree for each $G[C_i]$, and for each $G[D_i]$. Make up a new root $(s, X \cup \{s\})$, and attach the directed elimination trees for the digraphs $G[C_i]$ as children to that new root. This gives a directed elimination tree for $G[W \cup \{s\} \cup \bigcup_i C_i]$. The union of this tree with the directed elimination trees for the strongly connected components $D_1, \dots, D_r$ yields a directed elimination forest for $G[W \cup X \cup s]$. This completes the description of the subroutine for extending the forest.

The recursion terminates as soon as the size of W decreases below $\beta(\log n)^{3/2}$. In this case, simply return an (arbitrary) directed elimination forest for $G[W]$.

Here, the number β is a fixed, suitably chosen, constant coming from the analysis below. This completes the description of the algorithm.

It remains to analyze the above algorithm. It is readily checked that the algorithm returns an elimination forest for G. For the performance guarantee, those recursive calls that simply partition the graph into strongly connected components do not add to the height of the resulting forest; if we restrict our attention to these recursive calls that compute a suitable vertex subset S, the depth of the recursion tree is $O(\log n)$. At each such step, we can find in polynomial time a suitable set S of size at most $\beta k\sqrt{\log n}$, where k is the directed pathwidth of G, and β is some known constant (cf. [25, Corollary 2.25]). The recursion terminates with an elimination forest of height at most $\beta \cdot (\log n)^{3/2}$. Thus the overall height is bounded by

$$\beta \cdot k \cdot \sqrt{\log n} \cdot O(\log n) + \beta \cdot (\log n)^{3/2} = O(k \cdot (\log n)^{3/2}),$$

where k is the directed pathwidth of G. By Lemma 10, we have $k \leq \mathrm{r}(G)$. In this way, we have a polynomial-time $O((\log n)^{3/2})$-approximation for cycle rank. □

The above performance guarantee matches the best previous result known for the undirected case [1]. For other digraph complexity measures, such as D-width and directed pathwidth, approximation algorithms in a similar vein were recently given in [25].

4.3 Exact Computation

In certain circumstances, an approximation guarantee within a factor $O((\log n)^{3/2})$ may not suffice. Thus we also take a look at exact algorithms for computing the cycle rank.

The naïve algorithm for determining cycle rank according to Definition 1 entails inspecting $n!$ possibilities on a graph with n vertices. While one may not expect a polynomial-time algorithm, we can still do much better:

Theorem 14 *The cycle rank of an n-vertex digraph can be computed in time and space $O^*(2^n)$.*

Proof: We show how the characterization of the cycle rank of a digraph $G = (V, E)$ in terms of the directed elimination forests from Definition 2 can be turned into a dynamic programming scheme. We only consider the case G itself is nontrivially strongly connected—otherwise, we obtain the cycle rank by taking the minimum among the cycle ranks of the nontrivial strongly connected components of G. For a nontrivial strongly connected subset of vertices $X \subseteq V$ and a vertex $x \in X$, let $\mathrm{r}(x, X)$ denote the minimum height among all elimination forests for G with root (x, X). Then $\mathrm{r}(G) = \min_{v \in V} \mathrm{r}(v, V)$, so it suffices to design an algorithm computing $\mathrm{r}(v, V)$ for each $v \in V$. By inspecting Definition 2, we obtain the recurrence

$$\mathrm{r}(x, X) = \begin{cases} 1 & \text{if } G[X] - x \text{ is acyclic} \\ 1 + \max_Y \min_{y \in Y} \mathrm{r}(y, Y) & \text{otherwise} \end{cases} \tag{1}$$

Here Y runs over all nontrivial strongly connected components of $G[X] - x$ (of which there can be at most $|X| - 1$). Using the classic trick of memoization (see [14]), this recurrence can be easily transformed into a dynamic programming scheme with memoization that runs in time $|\mathfrak{S}| \cdot n^{O(1)}$, where $\mathfrak{S} \subseteq 2^V$ is the set of strongly connected subsets of the digraph G. □

The reader is invited to try out this algorithm for the digraph depicted in Figure 1. The bottleneck in the algorithm is the requirement of computing and storing the cycle rank for all elements of $\mathfrak{S}$, namely of the family of strongly connected subsets in the input digraph. For a complete digraph, we have $|\mathfrak{S}| = 2^n$. But this bound can no longer be reached for digraphs of bounded maximum outdegree. For undirected graphs of maximum degree d, a nontrivial bound on the number of (weakly) connected subsets was established recently in [7]. As it turns out, their bound allows the following generalization to the theory of digraphs, in that the original proof carries over with obvious modifications:

Lemma 15 *Let G be a digraph of order n with maximum outdegree at most d. Then the number of strongly connected subsets of V is at most $\gamma^n + n$, with $\gamma = (2^{d+1} - 1)^{1/(d+1)}$. In particular, for $d = 2$, we have $\gamma \doteq 1.9129$.* □

On digraphs of bounded outdegree, we thus obtain the following improved bound on the running time of the above algorithm:

Theorem 16 *Let G be a digraph of order n with constant maximum outdegree d. Then the cycle rank of G can be computed in time and space $O^*((2 - \varepsilon)^n)$, where ε is a constant depending on d. In particular, for digraphs of maximum outdegree 2, the cycle rank can be computed in time and space $O^*(1.9129^n)$.* □

It seems that Lemma 15 has a host of algorithmic consequences. For illustration, recall that a vertex subset $S \subseteq V$ of a digraph G is a *directed feedback vertex set*, if removing S from G leaves an acyclic digraph. Off the cuff, we can devise an exact algorithm for minimum directed feedback vertex set on sparse digraphs.

Theorem 17 *Let G be a digraph of order n with constant maximum outdegree d. Then a minimum directed feedback vertex set of G can be computed in time and space $O^*((2 - \varepsilon)^n)$, where ε is a constant depending on d. In particular, for digraphs of maximum outdegree 2, a minimum directed feedback vertex set can be computed in time and space $O^*(1.9129^n)$.*

Proof: By duality, the task of enumerating all minimal directed feedback vertex sets is equivalent to enumerating all maximal acyclic subsets, that is, maximal vertex subsets that induce a directed acyclic graph. Here, “minimal” and “maximal” are meant with respect to set inclusion.

Since there is an algorithm enumerating all minimal directed feedback vertex sets (or, equivalently, all maximal acyclic subsets) with polynomial delay [37], it only remains to derive a combinatorial bound on the number of such sets. A strongly connected subset $S \subset V$ in G is called a *minimal strongly connected* *subset*, if S contains a vertex v such that $S - v$ is an acyclic subset. Clearly, in this case, $S - v$ is a maximal acyclic subset. Thus, each minimal strongly connected subset S will give rise to at most $|S| \leq n$ maximal acyclic subsets; and each maximal acyclic subset can be obtained in this way from a minimal strongly connected subset. Thus the total number of maximal acyclic subsets in G is at most n times the number of (minimal) strongly connected subsets in G. The result now follows with Lemma 15. □

The above running time appears reasonable if we consider the following facts: First, even on digraphs of maximum outdegree at most 2, the problem is $\mathbf{NP}$-complete [16, Problem GT7]. Second, the fastest known exact algorithm for digraphs of unbounded outdegree [33] runs in time $O^*(1.9977^n)$. Even for digraphs of maximum total degree at most 4, the best previous result [32] was a running time of $O^*(1.945^n)$. Observing that maximum total degree at most 4 implies maximum indegree at most 2 or maximum outdegree at most 2, our algorithm is both faster than the one given in [32], and encompasses a larger class of digraphs. Third, easy examples show that digraphs with outdegree 2 can have at least 1.4142^n minimal directed feedback vertex sets [37].

5 Star Height of Regular Expressions

As it turns out, the cycle rank of digraphs is intimately related to structural and descriptional complexity aspects of regular expressions. The star height of a regular language L, denoted by $\mathrm{h}(L)$, is defined as the minimum nesting depth of stars in any regular expression describing L. The following relation between star height and the cycle rank of nondeterministic finite automata (NFAs) was shown already in the seminal paper on star height [12].

Theorem 18 (Eggan's Theorem) *Let L be a regular language. Then*

$$\mathrm{h}(L) = \min\{\, \mathrm{r}(A) \mid A \textit{ is an NFA accepting } L \,\}.$$

Here, $\mathrm{r}(A)$ denotes the cycle rank of the digraph underlying the transition structure of A.

As an aside, Eggan's Theorem was recently used to obtain a powerful lower bound technique for the minimum required length of regular expressions for a given regular language:

Lemma 19 (Star Height Lemma, [19]) *Let L be a regular language. If L admits a regular expression of length n, then $n \geq 2^{\Omega(\mathrm{h}(L))}$.*

The gist of the proof is that each regular expression can be converted into an equivalent NFA of comparable size, but whose transition structure is only poorly connected. The result then follows using Eggan's Theorem. In [19], this method was used to prove the unexpected result that complementing regular languages can cause a doubly-exponential blow-up in the minimum required regular expression length.

Of course, the minimum in Eggan's Theorem is taken over infinitely many NFAs, and indeed for more than two decades, it was unknown whether there exists an algorithm deciding the **STAR HEIGHT** problem: given a deterministic finite automaton (DFA) A and an integer k, determine whether the star height of $L(A)$ is at most k, a question raised in [12]. Although the problem is now known to be decidable, the best known upper bound[iv] to date is $\mathbf{EXPSPACE}$ [26]. To the best of our knowledge, nontrivial

[iv] The noted upper bound holds more generally for a given NFA if also an NFA accepting the complement language is provided as part of the input. Recall that complementing a DFA does not affect its size, whereas complementing an NFA can cause an exponential blow-up in the required number of states [21].

lower bounds are known *only* for the case where the input is specified succinctly, as an NFA: Determining the star height of a language specified as an NFA is $\mathbf{PSPACE}$-hard [24]. Yet, as illustrated in [24], a large multitude of natural questions about the language accepted by a given NFA is $\mathbf{PSPACE}$-hard, whereas the corresponding questions often become computationally easy if a DFA is given. Therefore, such a hardness result renders more service to understanding the effect of succinct input descriptions than to understanding the computational nature of the core problem at hand. That is why we deliberately stick to the convention to specify the input as a DFA.

Here we settle the complexity of the star height problem for a subclass of the regular languages, namely the bideterministic languages. The decision problem **BIDETERMINISTIC STAR HEIGHT** is defined as follows: Given a bideterministic finite automaton A and an integer k, decide whether the star height of $L(A)$ is at most k.

Bideterministic finite automata have the special property that the star height problem of bideterministic languages boils down to determining the cycle rank of a digraph. The following theorem is proved in [29]:

Theorem 20 (McNaughton's Theorem) *Let L be a bideterministic language, and let A be the minimal trim (i.e., without a dead state) DFA accepting L. Then* $\mathrm{h}(L) = \mathrm{r}(A)$.

On the positive side, the algorithmic results from the previous section easily translate to a formal language setup using McNaughton's Theorem. For approximating **STAR HEIGHT**, we have to resort to Eggan's Theorem, giving only an $O(n)$-approximation. In the bideterministic case, we have the following counterpart to Theorem 13:

Theorem 21 *The* **BIDETERMINISTIC STAR HEIGHT** *problem admits a polynomial-time approximation within a factor of* $O((\log n)^{3/2})$. □

We also have a natural counterpart to Theorem 16:

Theorem 22 *Let A be a bideterministic finite automaton with n states over an input alphabet of size k. Then the star height of $L(A)$ can be computed exactly, in time and space $O^*\left((2-\varepsilon)^n\right)$, where ε is a constant depending on k. In particular, for the case of binary input alphabets, the star height can be computed in time and space $O^*(1.9129^n)$.* □

On the negative size, also the $\mathbf{NP}$-hardness result for **CYCLE RANK** translates to its language-theoretic counterpart. Moreover, we show that already the case of binary input alphabets is that hard:

Theorem 23 *The* **BIDETERMINISTIC STAR HEIGHT** *problem is $\mathbf{NP}$-complete, and this still holds when restricted to bideterministic automata over binary input alphabets.*

Proof: We first show $\mathbf{NP}$-completeness for the case of unbounded alphabet size, and then provide a polynomial-time reduction to the case of binary alphabets.

For membership in $\mathbf{NP}$, we use McNaughton's Theorem (Theorem 20) to reduce the problem to **CYCLE RANK**, and the latter is in $\mathbf{NP}$ by Theorem 12.

To establish $\mathbf{NP}$-hardness, we reduce from the problem of determining for a strongly connected digraph $G = (E, V)$ and an integer k whether the cycle rank is at most k, which is $\mathbf{NP}$-hard by Theorem 12. For a vertex v in V, define

$$L(G, v) = \{\, w \in E^* \mid w \text{ is a walk in } G \text{ starting and ending in } v \,\}.$$

A deterministic finite automaton A accepting $L(G, v)$ has V as set of states and for each edge $(x, y) \in E$ a transition labeled (x, y) from x to y. The start and only accepting state is v. It is readily verified that A accepts $L(G, v)$, is bideterministic, and that A is the minimal trim DFA for this language. By construction, $\mathrm{r}(A) = \mathrm{r}(G)$, and $\mathrm{r}(A) = \mathrm{h}(L)$ by Theorem 20. This completes the $\mathbf{NP}$-completeness proof for unbounded alphabet size.

We turn to the case of binary alphabets. Given an instance (A, k) of **BIDETERMINISTIC STAR HEIGHT**, we construct in polynomial time a bideterministic finite automaton B over the alphabet $\{a, b\}$, such that the star height of B equals the star height of A. Assume the input alphabet of A is $\Sigma = \{a_1, a_2, \ldots, a_r\}$. The automaton B will accept the homomorphic image of $L(A)$ under the homomorphism $\rho : \Sigma \to \{a, b\}$ given by $\rho(a_i) = a^i b^{r+1-i}$, for $1 \leq i \leq r$. It is known [30] that ρ preserves star height, that is, for every regular language L, the image of L under ρ is of the same star height as L. It remains to construct, in polynomial time, a bideterministic automaton B accepting $\rho(L(A))$: automaton B will have the states of A, plus some extra states. For each state q copied from A, we add r states $q_1^+, q_2^+, \ldots q_r^+$ and r more states $q_1^-, q_2^-, \ldots q_r^-$ to the state set of B. The transition relation of B is given by requiring that whenever there is a transition $p \stackrel{a_i}{\to} q$ in A, then B admits the sequence of transitions

$$p \stackrel{a}{\to} p_1^+ \stackrel{a}{\to} p_2^+ \cdots \stackrel{a}{\to} p_i^+ \stackrel{b}{\to} q_{r-i}^- \stackrel{b}{\to} \cdots q_2^- \stackrel{b}{\to} q_1^- \stackrel{b}{\to} q.$$

There are no other transitions in B. By construction, B accepts $\rho(L(A))$. It is easily verified that if A is bideterministic, then so is B. □

Returning again to **CYCLE RANK**, we observe that the digraph underlying a bideterministic automaton over a binary alphabet always has maximum outdegree at most 2 and maximum total degree at most 4. The correspondence given by McNaughton's Theorem between bideterministic automata and digraphs yields the following consequence:

Corollary 24 *The* **CYCLE RANK** *problem restricted to digraphs of maximum outdegree at most* 2 *and total degree at most* 4 *remains* $\mathbf{NP}$*-complete.* □

6 Conclusion

In this work, we explored measures for the complexity of digraphs, and their applications. We paid particular attention to the cycle rank of digraphs and its relation to other digraph complexity measures, as well as its connection to the star height of regular languages. A tabular summary of our main algorithmic results is given in the Appendix.

Regarding cycle rank, the undirected case seems to be much better understood than the general case. An intriguing open question is whether the cycle rank problem is fixed-parameter tractable. This is known to be true on undirected graphs, see [8].

Regarding the star height problem, the picture is even less clear. The main problem, namely the decidability status, has been settled for more than 20 years. Still, the computational complexity of this problem is not well understood. From the viewpoint of a computational complexity, we studied the "easiest hard case", and showed that (the non-succinct version of) this problem is $\mathbf{NP}$-hard. Currently the best upper bound [26] is $\mathbf{EXPSPACE}$. Tightening the eminent gap between these bounds is certainly a challenging theme for further research.

Acknowledgements

The author would like to thank Markus Holzer for carefully reading an earlier draft of this paper, and for providing some valuable suggestions. Also thanks to the referee and to the editor for some useful hints.

References

[1] Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $o(\sqrt{(}\log n))$ approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 573–581, 2005.

[2] Hannah Alpert. Rank numbers of grid graphs. *Discrete Mathematics*, 310(23):3324–3333, 2010.

[3] Dana Angluin. Inference of reversible languages. *Journal of the ACM*, 29(3):741–765, 1982.

[4] Amotz Bar-Noy, Panagiotis Cheilaris, Michael Lampis, Valia Mitsou, and Stathis Zachos. Ordered coloring grids and related graphs. *Theoretical Computer Science*, 444:40–51, 2012.

[5] János Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.

[6] Dietmar Berwanger, Anuj Dawar, Paul Hunter, Stephan Kreutzer, and Jan Obdrzálek. The DAG-width of directed graphs. *Journal of Combinatorial Theory, Series B*, 102(4):900–923, 2012.

[7] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. The traveling salesman problem in bounded degree graphs. *ACM Transactions on Algorithms*, 8(2):18, 2012.

[8] Hans L. Bodlaender, Jitender S. Deogun, Klaus Jansen, Ton Kloks, Dieter Kratsch, Haiko Müller, and Zsolt Tuza. Rankings of graphs. *SIAM Journal on Discrete Mathematics*, 11(1):168–181, 1998.

[9] Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms*, 18(2):238–255, 1995.

[10] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *Journal of the ACM*, 55(5), 2008.

[11] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 3rd edition, 2006.

[12] Lawrence C. Eggan. Transition graphs and the star height of regular events. *Michigan Mathematical Journal*, 10(4):385–397, 1963.

[13] Stanley C. Eisenstat and Joseph W. H. Liu. The theory of elimination trees for sparse unsymmetric matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3):686–705, 2005.

[14] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[15] Robert Ganian, Petr Hlinený, Joachim Kneis, Alexander Langer, Jan Obdrzálek, and Peter Rossmanith. On digraph width measures in parameterized algorithmics. In Jianer Chen and Fedor V. Fomin, editors, *4th International Workshop on Parameterized and Exact Computation*, volume 5917 of *Lecture Notes in Computer Science*, pages 185–197. Springer, 2009.

[16] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. W. H. Freeman, 1979.

[17] Hermann Gruber. Digraph complexity measures and applications in formal language theory. In David Antoš, Milan Češka, Zdeněk Kotásek, Mojmír Křetínský, Luděk Matyska, and Tomáš Vojnar, editors, *4th Workshop on Mathematical and Engineering Methods in Computer Science, Znojmo, Czech Republic*, pages 60–67, 2008.

[18] Hermann Gruber. On balanced separators, treewidth, and cycle rank. Preprint, 2010. Available online as arXiv:1012.1344v1 [cs.DM].

[19] Hermann Gruber and Markus Holzer. Finite automata, digraph connectivity, and regular expression size. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walkuwiewicz, editors, *35th International Colloquium on Automata, Languages and Programming (Part II)*, volume 5126 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2008.

[20] Kosaburo Hashiguchi. Algorithms for determining relative star height and star height. *Information and Computation*, 78(2):124–169, 1988.

[21] Markus Holzer and Martin Kutrib. Nondeterministic descriptional complexity of regular languages. *International Journal of Foundations of Computer Science*, 14(6):1087–1102, 2003.

[22] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Series in Computer Science. Addison-Wesley, 1979.

[23] Paul Hunter. LIFO-search on digraphs: A searching game for cycle-rank. In Olaf Owe, Martin Steffen, and Jan Arne Telle, editors, *18th International Symposium on Fundamentals of Computation Theory*, volume 6914 of *Lecture Notes in Computer Science*, pages 217–228. Springer, 2011.

[24] Harry B. Hunt III and Daniel J. Rosenkrantz. Computational parallels between the regular and context-free languages. *SIAM Journal on Computing*, 7(1):99–114, 1978.

[25] Shiva Kintali, Nishad Kothari, and Akash Kumar. Approximation algorithms for directed width parameters. Preprint, 2011. Available online as arXiv:1107.4824v1 [cs.DS].

[26] Daniel Kirsten. On the complexity of the relative inclusion star height problem. *Advances in Computer Science and Engineering*, 5(3):173–211, 2010.

[27] Stephan Kreutzer and Sebastian Ordyniak. Digraph decompositions and monotonicity in digraph searching. *Theoretical Computer Science*, 412(35):4688–4703, 2011.

[28] Michael Lampis, Georgia Kaouri, and Valia Mitsou. On the algorithmic effectiveness of digraph decompositions and complexity measures. *Discrete Optimization*, 8(1):129–138, 2011.

[29] Robert McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1/2):167–176, 1967.

[30] Robert McNaughton. The loop complexity of regular events. *Information Sciences*, 1(3):305–328, 1969.

[31] Jaroslav Nešetřil and Patrice Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, 2006.

[32] Venkatesh Raman, Saket Saurabh, and Somnath Sikdar. Efficient exact algorithms through enumerating maximal independent sets and other techniques. *Theory of Computing Systems*, 41(3):563–587, 2007.

[33] Igor Razgon. Computing minimum directed feedback vertex set in $O^*(1.9977^n)$. In Giuseppe F. Italiano, Eugenio Moggi, and Luigi Laura, editors, *Proceedings of the 10th Italian Conference on Theoretical Computer Science*, pages 70–81. World Scientific, 2007.

[34] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[35] Mohammad Ali Safari. D-Width: A more natural measure for directed tree width. In Joanna Jedrzejowicz and Andrzej Szepietowski, editors, *30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*, pages 745–756. Springer, 2005.

[36] Robert Schreiber. A new implementation of sparse Gaussian elimination. *ACM Transactions on Mathematical Software*, 8(3):256–276, 1982.

[37] Benno Schwikowski and Ewald Speckenmeyer. On enumerating all minimal solutions of feedback problems. *Discrete Applied Mathematics*, 117(1-3):253–265, 2002.

[38] Joachim von zur Gathen. Irreducibility of multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):225–264, 1985.

Appendix

CYCLE RANK	
Instance.	A digraph G and an integer k.
Question.	Is the cycle rank of G at most k?
Good news.	Approximable within $O((\log n)^{3/2})$ in polynomial time (Thm. 13). Exact solution can be computed in time $O^*(1.9129^n)$ for digraphs with maximum outdegree at most 2; and for unbounded outdegree in time $O^*(2^n)$ (Thm. 16).
Bad news.	$\mathbf{NP}$-complete (Thm. 12). Problem is $\mathbf{NP}$-hard already for digraphs of maximum outdegree 2 and maximum total degree 4 (Cor. 24); $\mathbf{NP}$-hard also for some classes of undirected graphs (e.g., bipartite and cobipartite) [8].

DIRECTED FEEDBACK VERTEX SET	
Instance.	A digraph G and an integer k.
Question.	Does G admit a directed feedback vertex set of cardinality at most k?
Good news.	For digraphs with maximum outdegree at most 2, exact solution can be computed in time $O^*(1.9129^n)$ (Thm. 17); and in time $O^*(1.9977^n)$ for unbounded outdegree [33]. Problem is fixed-parameter tractable [10].
Bad news.	$\mathbf{NP}$-complete, already for digraphs of maximum outdegree 2 [16, Problem GT7].

BIDETERMINISTIC STAR HEIGHT	
Instance.	A bideterministic finite automaton A and an integer k.
Question.	Is the star height of $L(A)$ at most k?
Good news.	Approximable within $O((\log n)^{3/2})$ in polynomial time (Thm. 21). Exact solution can be computed in time $O^*(1.9129^n)$ for binary alphabets; and for unbounded alphabet size in time $O^*(2^n)$ (Thm. 22).
Bad news.	$\mathbf{NP}$-complete; $\mathbf{NP}$-hardness holds already for binary alphabets (Thm. 23).

STAR HEIGHT	
Instance.	A deterministic finite automaton A and an integer k.
Question.	Is the star height of $L(A)$ at most k?
Good news.	Problem is decidable [20]. Exact solution can be computed within exponential space and doubly exponential time [26].
Bad news.	$\mathbf{NP}$-hard, already for binary alphabets (Thm. 23). Problem is $\mathbf{PSPACE}$-hard if input given by an nondeterministic finite automaton in place of a deterministic one [24].