Digraph Complexity Measures and Applications in Formal Language Theory

Hermann Gruber

Institut für Informatik, Justus-Liebig–Universität Giessen
Arndtstrasse 2, D-35392 Giessen, Germany
email: hermann.k.gruber@informatik.uni-giessen.de

Abstract. We give an overview of some structural complexity measures and width parameters on digraphs, relate them to each other, and discuss their computational complexity aspects. Particular attention is given to cycle rank, perhaps the oldest of these measures to be studied in the literature, which turns out to be intimately related to structural and descriptional complexity aspects of regular expressions. Some new results on cycle rank thus have immediate implications in formal language 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. [5] for a mainly structural account. 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 theory of undirected graphs to the case of digraphs [1–3, 17, 18]. We discuss some of these measures, relate them to each other, and investigate their algorithmic aspects. Notably, treewidth as the most successful notion admits several competing generalizations to the case of digraphs, and there currently seems to be no commonly accepted generalization of this notion. We decided to let others judge and will discuss these only as far as needed. Interestingly, we are able to show that all discussed complexity measures bound each other within a factor logarithmic in the order of the graph, paralleling the case of undirected graphs [4]. We focus in particular on the cycle rank, a digraph complexity measure originally defined in the context of formal languages [6], and present new applications of this concept to the theory of structural and descriptional complexity of regular expressions.

2 Preliminaries

We assume the reader is familiar with standard notions in graph theory, as contained in [5], so we only fix the notation and a few specialities 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*. If the edge relation of a digraph G is symmetric, we say G is an (undirected) *graph*. For a digraph G, by taking the symmetric closure of the edge relation, we obtain its undirected version $\overleftrightarrow{G}$. For a subset of vertices $U \subseteq V$, let $G[U]$ denote the induced sub(di)graph $(U, E \cap U \times U)$, 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 $v \in V$ there is a (possibly empty) path from v to itself. Maximal strongly connected subsets of V are called strongly connected components (SCCs); a strongly connected subset S is *nontrivial* if the subdigraph $G[S]$ induced by S contains at least one edge (note that this also allows the case $S = \{v\}$ if v has a loop). A digraph is *acyclic* if all of its strongly connected components are trivial.

3 Complexity Measures on Digraphs

This section contains a discussion of different connecitivity measures on digraphs. We begin with an inductive definition of cycle rank, perhaps the oldest digraph complexity measure, defined in the 1960s by Eggan and Büchi [6], which plays a prominent role in this paper.

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 SCCs 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. Next we introduce the weak separator number of a digraph, given in [11] as a generalization of separator number [4, 16] to digraphs.

Definition 2. *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 SCC of $G[U \setminus S]$ contains at most $\frac{1}{2}|U|$ 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.*

The following is shown in [11]:

Lemma 3 ([11]). *Let G be a digraph with n vertices and weak separator number at most k. Then $\mathrm{r}(G) \leq 1 + k \cdot \log n$.*

The last measure we will discuss, directed pathwidth, was according to [1] originally defined by Reed, Seymour and Thomas as a generalization of undirected pathwidth to digraphs.

Definition 4. *Given a digraph $G = (V, E)$, a directed path decomposition of G is a sequence $W_1, W_2, \ldots, W_r$ of subsets of V, called bags, such that a) each*

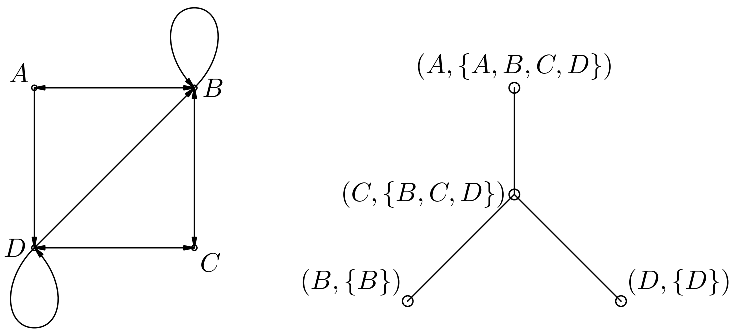

Fig. 1. An example digraph and a directed elimination tree for it.

vertex is contained in at least one bag, b) for all $i < j < k$ holds $W_i \cap W_k \subseteq W_j$, 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.

How does cycle rank relate to directed pathwidth? We first note that the cycle rank can be equivalently defined using decompositions, compare [15]:

Definition 5. *A* directed elimination tree *for a nontrivially strongly connected digraph G 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 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 SCCs $C_1, \ldots C_k$, 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 [15] that the minimum height among all directed elimination forests for G equals the cycle rank of G. Interestingly, the concept of elimination forests was apparently rediscovered in the context of sparse matrix factorization, see [19] for the undirected case and [7] for the directed case. The equivalence of these concepts is best seen by [7, Thm. 3.3]. We can now relate cycle rank to directed pathwidth.

Theorem 6. *Let G be a digraph. Then* $\mathrm{dpw}(G) \leq \mathrm{r}(G)$.

Proof (Sketch). If G is acyclic, then $\mathrm{r}(G) = 0$ and $\mathrm{dpw}(G) = 0$. Otherwise, we transform a directed elimination forest of minimum height into a directed path decomposition recursively as follows. If the forest has roots (x_1, C_1), (x_2, C_2),..., (x_k, C_k), order the SCCs topologically, and recursively compute a directed path-decomposition of width $\mathrm{r}(G) - 1$ for each digraph $G[C_i] - x_i$. Add the vertex x_i to each bag in the respective decomposition to get each a decomposition of width $\mathrm{r}(G)$. Concatenating the k individual directed path decompositions while respecting the above topological order, we obtain a directed path decomposition of width $\mathrm{r}(G)$ for G. □

Currently, there are a few competing candidates for generalizing treewidth to digraphs, among them D-width [18] and DAG-width [3, 17]. Quite a few more proposals are discussed in these references; all proposed candidates share the property that they specialize on symmetric digraphs to undirected treewidth. For formal definitions, see the given references. We obtain the following relation:

Theorem 7. *Let G be an n-vertex digraph. Then*

$$\mathrm{s}(G) - 1 \leq \text{D-width}(G) \leq \text{DAG-width}(G) \leq \mathrm{dpw}(G) \leq \mathrm{r}(G) \leq 1 + \mathrm{s}(G) \cdot \log n.$$

Proof. We establish the inequalities from left to right. In the undirected case, the separator number of a graph is bounded above by its treewidth plus one. An analogous theorem for D-width is proved by the author in [10].

To compare D-width and DAG-width, we use the fact that both admit an equivalent formulation as the number of cops required to catch the robber in certain cops and robber games. In the game for DAG-width k, the k cops have to catch a visible robber on the graph G. In each turn, some cops use helicopters to change their position, while others stay at a vertex. The robber can flee in each step along a directed path in G not blocked by a cop, and the cops are not allowed to revisit any vertex once vacated [3, 17]. The game for D-width k has the same rules, but the robber player is not allowed to leave his current strongly connected component induced by the non-blocked vertices [8], thus it is no harder for the cops to catch the robber than in the previous game.

The third inequality follows from the fact that every path decomposition can be seen as a DAG-decomposition [3], and the last two inequalities are literally Lemma 3 and Theorem 6. □

We turn to the question of computational complexity.

Theorem 8. *Given a digraph G and an integer k, determining whether the cycle rank of G is at most k is* **NP***-complete. This even holds if G is strongly connected.*

Proof. Membership in **NP** can be seen by the equivalent definition using directed elimination trees: every such tree contains at most $|V|$ tree vertices, and each tree vertex is of size is at most $|V|$. Such a witness can be guessed, and it can be verified in polynomial time that it is indeed an elimination tree of height at most k.

For **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 [16, Lem. 2.2]) that an undirected elimination tree of height k in the sense of [4, 16] corresponds to a directed elimination tree of height $k-1$ in our sense (the minus one is only due to a different definition of height). However, determining undirected elimination tree height is **NP**-complete [4], even for (strongly) connected graphs. □

How to cope with this negative result? A straightforward implementation determining cycle rank according to Definition 1 requires inspecting $n!$ possibilities on a graph with n vertices, as witnessed by the complete graph K_n. Clearly, we cannot expect a polynomial-time algorithm, but still we can do much better:

Theorem 9. *The cycle rank of an n-vertex digraph can be computed in time and space $2^n \cdot n^{O(1)}$ in the uniform random access model.*

Proof. We show how the alternative characterization of the cycle rank of a digraph $G = (V, E)$ in terms of the directed elimination forests from Definition 5 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 SCCs of G. For a nontrival strongly connected subset of vertices $X \subseteq V$ and a vertex $x \in X$, let $cr(x, X)$ 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 5, 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 SCCs of $G[X] - x$ (of which there can be at most $|X| - 1$). This recurrence can be transformed into a dynamic programming scheme with memoization, requiring large stack memory, or into a botton-up lookup scheme, as follows:

1. Build a list L of all pairs (x, X) with $G[X]$ a strongly connected subgraph and $x \in X$, sorted by the size of X.
2. Let D be an (initially empty) dictionary for storing the value $\mathrm{r}(x, X)$ for all keys (x, X) with $G[X]$ a strongly connected subgraph and $x \in X$.
3. For all $x \in (x, X)$, compute $cr(x, X)$ according to Eqn. (1), using the (at most $|X|^2$) known values $\mathrm{r}(y, Y)$, which are already stored in D. Store the result in D.

Using reasonably efficient standard algorithms for sorting according to a small number of keys, maintaining dictionaries supporting insert and find, and computing all SCCs of a graph, the details can be implemented such that the algorithm runs in time and space $2^n \cdot n^{O(1)}$. □

The reader is invited to try the above algorithm on the digraph depicted in Figure 1. Obviously, the bottleneck in the above algorithm is the requirement of enumerating and and storing all strongly connected subsets of a digraph. We expect that the number of these sets is often much smaller than 2^n, a number which is reached by a complete graph. An interesting question is whether there is an algorithm listing the set $\mathcal{S}$ of all strongly connected subsets of an n-vertex digraph in time $|\mathcal{S}| \cdot n^{O(1)}$.

Known computational complexity aspects for the other digraph complexity measures under consideration are summarized in Table 1.

Complexity measure	Decision problem	Size of decomposition for width k	References
$\mathrm{s}(G)$	?	n. a.	[11]
D-width(G)	**NP**-hard	$n^{O(k)}$	[8, 18]
DAG-width(G)	**NP**-hard	$n^{O(k)}$	[3, 17]
$\mathrm{dpw}(G)$	**NP**-complete	$n^{O(1)}$	[2]
$\mathrm{r}(G)$	**NP**-complete	$n^{O(1)}$	Here

Table 1. Computational complexity aspects of digraph complexity measures.

In addition to the open problems suggested by the gaps in Table 1, we note that the undirected counterpart of Theorem 7 in [4] has impressive consequences on the approximability of the (undirected counterparts of) all these measures because of the existence of an approximation algorithm for the separator number of an undirected graph. An apparent question is now whether the weak separator number admits a similar polynomial-time approximation.

4 Cycle Rank and Regular Expressions

We turn to applications. 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:

Theorem 10 (Eggan's Theorem). *Let L be a regular language. Then*

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

Here, $\mathrm{r}(A)$ denotes the cycle rank of the digraph underlying the transition structure of A. Recently, Eggan's Theorem was used in combination with Lemma 3 to obtain a lower bound technique for the minimum required size of regular expressions for a given language [11]—for further applications of that technique, see [9]. For convenience, we briefly sketch the idea.

Theorem 11 ([11]). *Assume L is a regular language described by a regular expression of size n. Then $n = 2^{\Omega(\mathrm{h}(L))}$.*

Proof (Idea.). The standard construction for recursively transforming a regular expression of size n into a finite automaton gives an NFA A with $O(n)$ states. The undirected transition structure $\overleftrightarrow{A}$ of this automaton is essentially series-parallel and thus the latter graph has undirected separator number of constant size. Note that for any digraph G, the undirected separator number of $\overleftrightarrow{G}$ can be never smaller than $\mathrm{s}(G)$. Thus using Eggan's Theorem and Lemma 3, we have $\mathrm{h}(L) \leq \mathrm{h}(A) \leq 1 + O(1) \cdot \log O(n) = O(\log n)$, or equivalently $n = 2^{\Omega(\mathrm{h}(L))}$. □

Of course, the minimum in Eggan's Theorem is taken over infinitely many NFAs, and indeed for quite some time, 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 [6]. Although the problem is now known to be decidable, the best known upper bound[1] to date is exponential space [13]. To the best of our knowledge, nontrivial lower bounds are known only for the case where the input is specified succinctly[2], as an NFA.

Here we settle the complexity of the star height problem for the class of bideterministic languages, a certain subclass of regular languages: Namely, a regular language is bideterministic if it can be accepted by a finite automaton that is both forward and backwards deterministic, and has a single start and a single accepting state. The following theorem is due to McNaughton [14]:

Theorem 12 (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 $\mathrm{h}(L) = \mathrm{r}(A)$.*

Using this theorem and the **NP**-completeness result for cycle rank, we can prove:

Theorem 13. *The star height problem for bideterministic languages is* **NP***-complete.*

Proof. Membership in **NP** is immediate from Theorem 12 and Theorem 8.

To establish **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

[1] 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. Note that in the case of a given NFA, the size of complemented NFA can be exponential in the size of the original NFA, thus potentially blowing up the "actual" size of the problem instance.

[2] Determining the star height of a language specified as an NFA is **PSPACE**-hard [12]; but, as illustrated in [12], a large multitude of natural questions about the language accepted by a given NFA is that hard, but often these questions become computationally easy if a DFA is given. Admittedly, such hardness results render more service to understanding succinctness of NFAs over DFAs than to understanding the nature of the actual problem under consideration. This is why here we deliberately stick to the convention to specify the input as a DFA.

is at most k, which is **NP**-hard by Theorem 8. For a vertex v in V, let $L(G, v)$ denote the language $\{ w \in E^* \mid w \text{ walk starting and ending in } v \}$. 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 12. □

References

1. J. Bárat. Directed Path-width and Monotonicity in Digraph Searching. *Graphs and Combinatorics* 22(2): 161–172, 2006.
2. B. Yang and Y. Cao. Digraph searching, directed vertex separation and directed pathwidth. *Discrete Applied Mathematics* 156(10): 1822–1837, 2008.
3. D. Berwanger, A. Dawar, P. Hunter, and S. Kreutzer. DAG-width and parity games. In *STACS 2006*, LNCS 3884, pp. 524–536. Springer, 2006.
4. H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, T. Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *Journal of Algorithms* 18(2): 238–255, 1995.
5. R. Diestel. Graph Theory. 3rd edition, Springer, 2005.
6. L. C. Eggan. Transition graphs and the star height of regular events. *Michigan Mathematical Journal*, 10(4):385–397, 1963.
7. S. Eisenstat and J. W. H. Liu. The theory of elimination trees for sparse unsymmetric matrices. *SIAM Journal on Matrix Analysis and Applications*, 26(3):686-705, 2005.
8. W. Evans, P. Hunter and M. A. Safari, D-Width and cops and robbers, 2007, preprint.
9. W. Gelade. Succinctness of regular expressions with interleaving, intersection and counting. In *MFCS 2008*, LNCS 5162, pp. 363–374. Springer, 2008.
10. H. Gruber. On the D-width of directed graphs. Manuscript, 2008.
11. H. Gruber and M. Holzer. Finite automata, digraph connectivity, and regular expression size. In *ICALP 2008*, LNCS 5126, pp. 39–50. Springer, 2008.
12. H. B. Hunt III and D. J. Rosenkrantz. Computational parallels between the regular and context-Free languages. *SIAM Journal on Computing*, 7(1):99–114, 1978
13. D. Kirsten. Desert automata V. On the complexity of the relative inclusion star height problem. Manuscript, 2007.
14. R. McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1/2):167–176, 1967.
15. R. McNaughton. The loop complexity of regular events. *Information Sciences* 1(3):305–328, 1969.
16. J. Nešetřil and P. Ossona de Mendez. Tree-depth, subgraph coloring and homomorphism bounds. *European Journal of Combinatorics*, 27(6):1022–1041, 2006.
17. J. Obdržálek. Dag-width: Connectivity measure for directed graphs. In *SODA 2006*, pp. 814–821. ACM Press, 2006.
18. M. A. Safari. D-Width: A more natural measure for directed tree width. In *MFCS 2005*, LNCS 3618, pp. 745–756. Springer, 2005.
19. R. Schreiber. A new implementation of sparse Gaussian elimination. *ACM Transactions on Mathematical Software*, 8(3):256–276, 1982