Digraph Complexity Measures and Applications in Formal Language Theory

by

Hermann Gruber

Institut für Informatik
Justus-Liebig-Universität Giessen
Germany

November 14, 2008

Overview

1. Introduction and Motivation
2. Measuring Complexity for Digraphs
3. Algorithmic Results on Cycle Rank
4. Applications in Formal Language Theory
5. Discussion

Outline

1. Introduction and Motivation
2. Measuring Complexity for Digraphs
3. Algorithmic Results on Cycle Rank
4. Applications in Formal Language Theory
5. Discussion

Complexity Measures on Undirected Graphs

Important topic in algorithmic graph theory:

- Structural complexity restrictions can speed up algorithms
- Main result: many hard problems solvable in linear time on graphs with bounded treewidth.
- depending on application, also other measures interesting

What about Directed Graphs?

Lifting this successful theory to digraphs?

- Mixed success for treewidth: remarkable positive, but many negative algorithmic results

Here:

- Concentrate on measures whose theory can be lifted more easily: Separator number, pathwidth, cycle rank (=elimination tree height)
- Relations, algorithmic results, applications
- In particular: Cycle rank important for theory of regular expressions

Outline

1 Introduction and Motivation

2 Measuring Complexity for Digraphs

3 Algorithmic Results on Cycle Rank

4 Applications in Formal Language Theory

5 Discussion

Cycle Rank: Definition

- Acyclic graph has $\mathrm{r}(G) = 0$
- If graph is a single SCC and $E \neq \emptyset$, then $\mathrm{r}(G) = \min_{v \in V} \mathrm{r}(G \setminus v) + 1$.
- Otherwise, we compute cycle rank for each component and take maximum

Observe: quite different from vertex feedback number

Intuition: Time needed to break all cycles "in parallel" by removing a single vertex from each component in each round.

Example

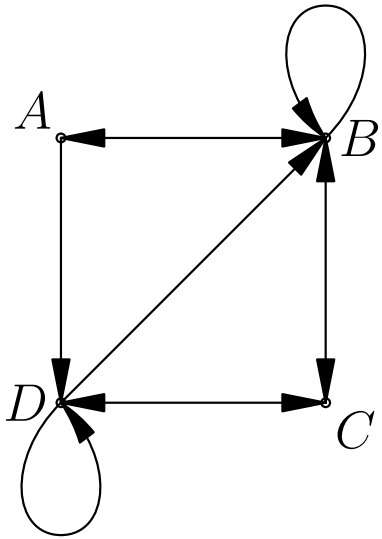
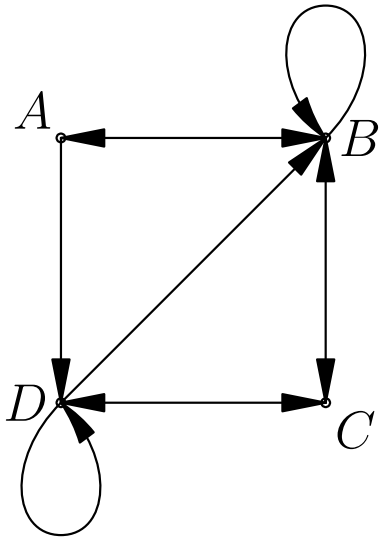


Figure: A recursion tree giving the cycle rank for a toy digraph.

Example



$$(A, \{A, \underset{\odot}{B}, C, D\})$$

Figure: A recursion tree giving the cycle rank for a toy digraph.

Example

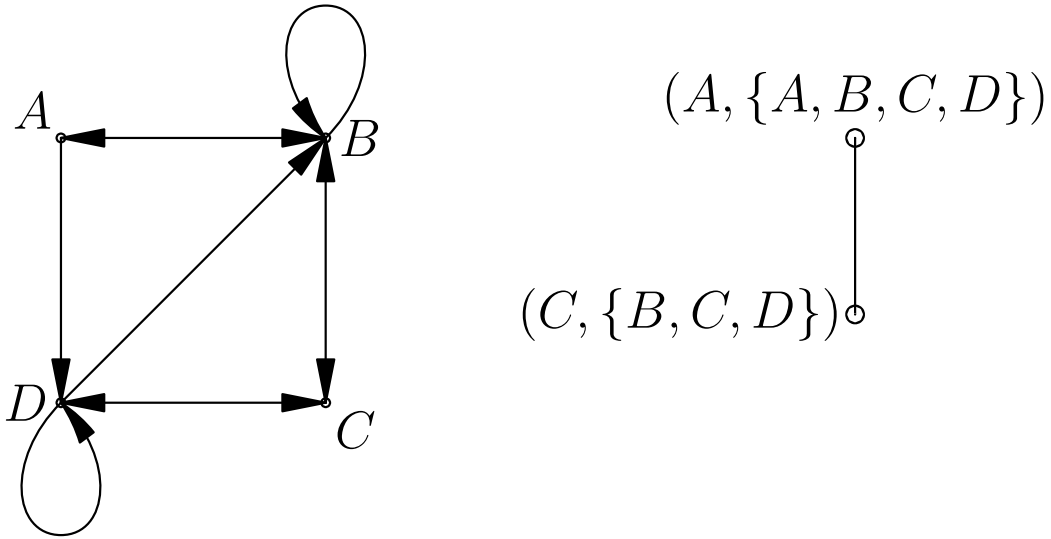


Figure: A recursion tree giving the cycle rank for a toy digraph.

Example

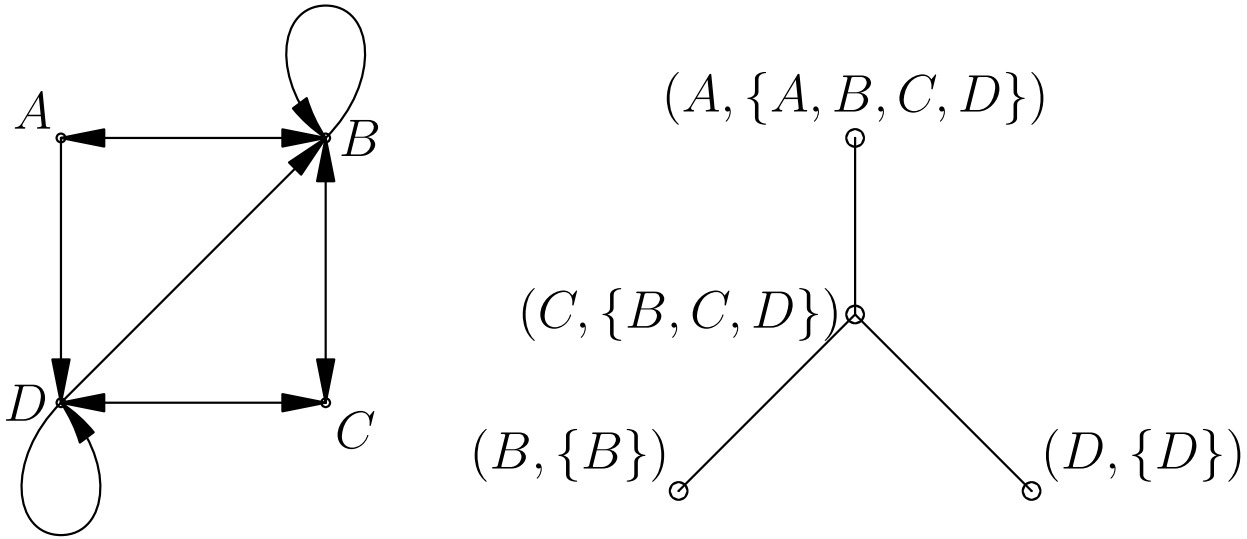


Figure: A recursion tree giving the cycle rank for a toy digraph.

Cycle Rank: Some historical notes

- Defined in 1960s by Eggan and Büchi.
  Motivation: formal languages
- remained largely unnoticed outside formal language community
- For undirected graphs rediscovered in 1980s as elimination tree height
  Motivation: Fast computation on sparse symmetric matrices
- Generalized to digraphs by Eisenstat and Liu in 2005
  Motivation: sparse matrices, again

Relation to other measures

We compared cycle rank other complexity measures
Result in short:

- the inspected measures admit a linear ordering
- Cycle rank is upper bound for all these measures
- $r(G)$ can exceed all other measures by at most logarithmic factor, i.e. $O(\log |V|)$.

Generalizes an influential result from undirected theory

Outline

1 Introduction and Motivation

2 Measuring Complexity for Digraphs

3 Algorithmic Results on Cycle Rank

4 Applications in Formal Language Theory

5 Discussion

Complexity Results for Cycle Rank

Theorem

The decision version of determining the cycle rank of a given digraph G is **NP***-complete.* □

Hardness not unexpected, some comparable measures not known to be in **NP**!
How to cope with? Naive algorithm runs in time $O(n!)$.

Theorem

The cycle rank of an n-vertex graph can be determined in time and space $\widetilde{O}(2^n)$. □

Outline

1 Introduction and Motivation

2 Measuring Complexity for Digraphs

3 Algorithmic Results on Cycle Rank

4 Applications in Formal Language Theory

5 Discussion

Cycle Rank and Regular Expressions

- Cycle rank closely related to star height of regular languages
- **NP**-completeness of cycle rank can be used to show:

Theorem

The star height problem for bideterministic finite automata is **NP***-complete.* □

General case, i.e. deterministic finite automaton input, largely open: **EXPSPACE** (Kirsten 2007) vs **NP**-hard (here)

Cycle Rank and Regular Expressions (2)

Another application:

- Predicting minimum regular expression size often difficult
- Can bound required size from below using star height (joint work with M. Holzer, presented at ICALP '08)
- Proof essentially relies on cycle rank

Outline

1 Introduction and Motivation

2 Measuring Complexity for Digraphs

3 Algorithmic Results on Cycle Rank

4 Applications in Formal Language Theory

5 Discussion

Discussion

- Approximation algorithm for cycle rank? Would imply (a bit weaker) approximation for D-width, DAG-width, directed path-width, ...
- Better (exact/fixed-parameter) algorithms for cycle rank? In particular, presented algorithm uses exponential space
- What other results from undirected theory can be more easily lifted to directed graphs?