DAG-width - Connectivity Measure for Directed Graphs

Jan Obdržálek
Masaryk University, Brno

DAG-width - Connectivity Measure for Directed Graphs

Jan Obdržálek
Masaryk University, Brno
Laboratory for Foundations of Computer Science
Edinburgh

DAG-width - Connectivity Measure for Directed Graphs

Jan Obdržálek
Masaryk University, Brno
Laboratory for Foundations of Computer Science
Edinburgh
RWTH Aachen

DAG-width - Connectivity Measure for Directed Graphs

Jan Obdržálek
Masaryk University, Brno
Laboratory for Foundations of Computer Science
Edinburgh
RWTH Aachen

xobdrzal@fi.muni.cz

Miami, 23 January 2006

Note aside

The measure was independently proposed by *Berwanger, Dawar, Hunter and Kreutzer* [STACS2006]

That paper also contains the algorithm for solving parity games.

Note aside

The measure was independently proposed by *Berwanger, Dawar, Hunter and Kreutzer* [STACS2006]

That paper also contains the algorithm for solving parity games.

There will be a combined journal paper soon.

Note aside

The measure was independently proposed by *Berwanger, Dawar, Hunter and Kreutzer* [STACS2006]

That paper also contains the algorithm for solving parity games.

There will be a combined journal paper soon.

Or read my thesis.

Tree-width – a primer

Robertson and Seymour (1984)

Measures how close is a given graph to being a tree.

(Trees have tree-width **one**.)

Low tree-width $\Longrightarrow$ decomposition into small number of subproblems

Some NP-complete problems are linear/polynomial on structures of bounded tree-width, e.g.

- graph colouring *(linear)*
- MSOL *(linear)*
- clique, Hamiltonian cycle, ... *(linear)*
- graph isomorphism *(polynomial)*

Tree-width – a primer

Robertson and Seymour (1984)

Measures how close is a given graph to being a tree.

(Trees have tree-width **one**.)

Low tree-width $\implies$ decomposition into small number of subproblems

Some NP-complete problems are linear/polynomial on structures of bounded tree-width, e.g.

- graph colouring *(linear)*
- MSOL *(linear)*
- clique, Hamiltonian cycle, ... *(linear)*
- graph isomorphism *(polynomial)*
- **parity games *(polynomial)* [O.]**

Tree-width – the definition

A *tree decomposition* of an (undirected) graph G is a pair $(T, \mathcal{X})$, where T is a tree (its vertices are called *nodes* throughout this paper) and $\mathcal{X} = \{X_t \mid t \in T\}$ is a family of subsets of $V(G)$ satisfying the following three conditions:

(T1) $V(G) = \bigcup_{t \in V(T)} X_t$,

(T2) for every edge $\{v, w\} \in E(G)$ there exists $t \in V(T)$ s.t. $\{v, w\} \subseteq X_t$, and

(T3) for all $t, t', t'' \in V(T)$ if t' is on the (unique) path from t to t'' in T, then $X_t \cap X_{t''} \subseteq X_{t'}$.

Tree-width – the definition

A *tree decomposition* of an (undirected) graph G is a pair $(T, \mathcal{X})$, where T is a tree (its vertices are called *nodes* throughout this paper) and $\mathcal{X} = \{X_t \mid t \in T\}$ is a family of subsets of $V(G)$ satisfying the following three conditions:

(T1) $V(G) = \bigcup_{t \in V(T)} X_t$,

(T2) for every edge $\{v, w\} \in E(G)$ there exists $t \in V(T)$ s.t. $\{v, w\} \subseteq X_t$, and

(T3) for all $t, t', t'' \in V(T)$ if t' is on the (unique) path from t to t'' in T, then $X_t \cap X_{t''} \subseteq X_{t'}$.

The *width* of a tree decomposition $(T, \mathcal{X})$ is $\max_{t \in T} |X_t| - 1$.

Tree-width – the definition

A *tree decomposition* of an (undirected) graph G is a pair $(T, \mathcal{X})$, where T is a tree (its vertices are called *nodes* throughout this paper) and $\mathcal{X} = \{X_t \mid t \in T\}$ is a family of subsets of $V(G)$ satisfying the following three conditions:

(T1) $V(G) = \bigcup_{t \in V(T)} X_t$,

(T2) for every edge $\{v, w\} \in E(G)$ there exists $t \in V(T)$ s.t. $\{v, w\} \subseteq X_t$, and

(T3) for all $t, t', t'' \in V(T)$ if t' is on the (unique) path from t to t'' in T, then $X_t \cap X_{t''} \subseteq X_{t'}$.

The *width* of a tree decomposition $(T, \mathcal{X})$ is $\max_{t \in T} |X_t| - 1$.

The *tree-width* of a graph G is the minimum width over all possible tree decompositions of G.

Tree-width – the definition

A *tree decomposition* of an (undirected) graph G is a pair $(T, \mathcal{X})$, where T is a tree (its vertices are called *nodes* throughout this paper) and $\mathcal{X} = \{X_t \mid t \in T\}$ is a family of subsets of $V(G)$ satisfying the following three conditions:

(T1) $V(G) = \bigcup_{t \in V(T)} X_t$,

(T2) for every edge $\{v, w\} \in E(G)$ there exists $t \in V(T)$ s.t. $\{v, w\} \subseteq X_t$, and

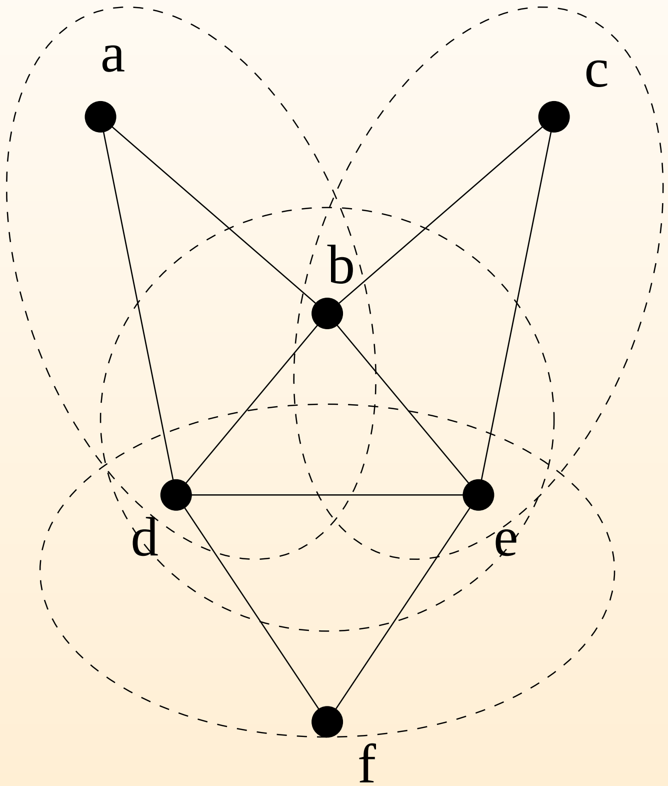
(T3) for all $t, t', t'' \in V(T)$ if t' is on the (unique) path from t to t'' in T, then $X_t \cap X_{t''} \subseteq X_{t'}$.

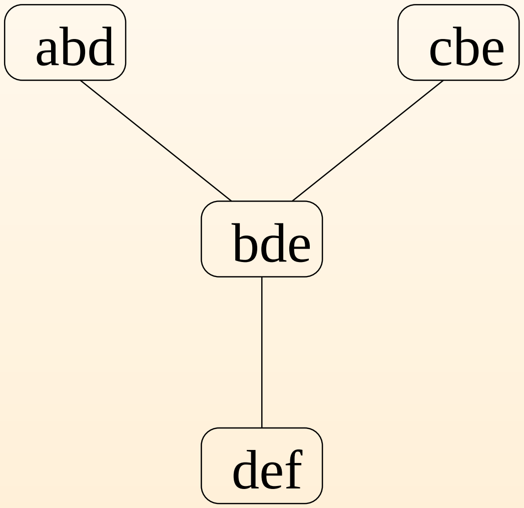
The *width* of a tree decomposition $(T, \mathcal{X})$ is $\max_{t \in T} |X_t| - 1$.

The *tree-width* of a graph G is the minimum width over all possible tree decompositions of G.

Trees have tree-width 1.

Tree-width – Example





Cops and robber game

Players: k *cops* v *a robber*

Cops and robber game

Players: k *cops* v *a robber*

Positions: $V(G)$ + the helicopter (for cops)

Cops and robber game

Players: k *cops* v *a robber*

Positions: $V(G)$ + the helicopter (for cops)

Rules:

- cops move by a helicopter

Cops and robber game

Players: k *cops* v *a robber*

Positions: $V(G)$ + the helicopter (for cops)

Rules:

- cops move by a helicopter
- robber must follow graph edges

Cops and robber game

Players: *k cops* v *a robber*

Positions: $V(G)$ + the helicopter (for cops)

Rules:

- cops move by a helicopter
- robber must follow graph edges
- robber cannot run through a cop

Cops and robber game

Players: k *cops* v *a robber*

Positions: $V(G)$ + the helicopter (for cops)

Rules:

- cops move by a helicopter
- robber must follow graph edges
- robber cannot run through a cop
- robber can see the helicopter landing and try to escape

Cops and robber game

Players: k *cops* v *a robber*

Positions: $V(G)$ + the helicopter (for cops)

Rules:

- cops move by a helicopter
- robber must follow graph edges
- robber cannot run through a cop
- robber can see the helicopter landing and try to escape

Objectives:
Robber: to evade capture
Cops: land a cop on the vertex occupied by the robber

More games

Monotone strategy for the cop player is a strategy that does not revisit any vertex.

More games

Monotone strategy for the cop player is a strategy that does not revisit any vertex.

Robber's strategy?

More games

Monotone strategy for the cop player is a strategy that does not revisit any vertex.

Robber's strategy?

Haven of size k is a function σ assigning to each set $X \in V(G)$, $|X| \leq k$, a *connected component* $\sigma(X)$ of $G \setminus X$ s.t. for any $X, Y \in V(G)$ of size less than k we have $\sigma(X)$ *touches* $\sigma(Y)$.

More games

Monotone strategy for the cop player is a strategy that does not revisit any vertex.

Robber's strategy?

Haven of size k is a function σ assigning to each set $X \in V(G)$, $|X| \leq k$, a *connected component* $\sigma(X)$ of $G \setminus X$ s.t. for any $X, Y \in V(G)$ of size less than k we have $\sigma(X)$ *touches* $\sigma(Y)$.

Duality theorem [Seymour, Thomas 93]:
The following are equivalent:

1. k cops can search G
2. k cops can monotonely search G
3. G has tree-width at most $k - 1$
4. G has a haven of size k

Tree-width and directed graphs

Tree-width is a measure of *undirected* graphs.

If used as measure of directed graphs, we forget the orientation of edges.

Fact: Clique of size n has tree-width n.

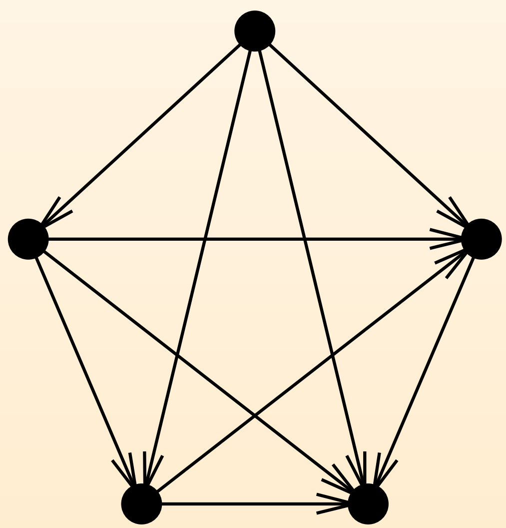
Tree-width and directed graphs

Tree-width is a measure of *undirected* graphs.

If used as measure of directed graphs, we forget the orientation of edges.

Fact: Clique of size n has tree-width n.

Here is a DAG with n vertices and tree-width n (for $n = 5$):



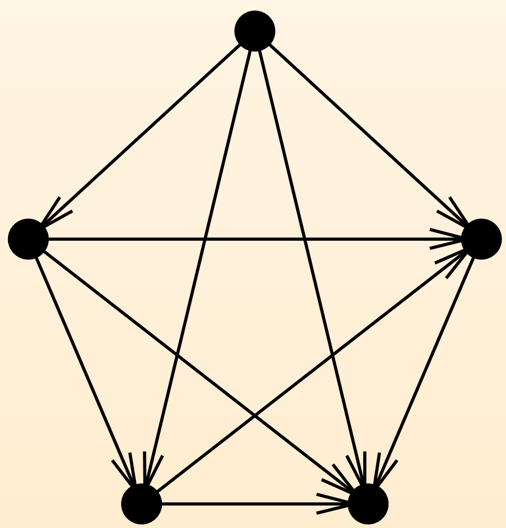
Tree-width and directed graphs

Tree-width is a measure of *undirected* graphs.

If used as measure of directed graphs, we forget the orientation of edges.

Fact: Clique of size n has tree-width n.

Here is a DAG with n vertices and tree-width n (for $n = 5$):



But it is easy to solve parity games on DAGs!

Z normal sets

For a graph G a set $S \subseteq V \setminus Z$ is *Z-normal* if there is no directed path in $G \setminus Z$ with first and last vertices in S that uses a vertex of $G \setminus (S \cup Z)$. I.e. no path can leave S and then return back to S without passing through a vertex in Z.

Directed Tree-width

(Johnson, Robertson, Seymour and Thomas, 2001)

An *arboreal decomposition* of a graph G is a triple $(R, \mathcal{X}, \mathcal{W})$ where R is a directed tree, and $\mathcal{X} = \{X_e \mid e \in E(R)\}$, $\mathcal{W} = \{W_r \mid r \in V(R)\}$ are sets of vertices of G satisfying:

(R1) $\mathcal{W}$ is a partition of $V(G)$ into nonempty sets

(R2) for $e \in E(R)$, $e = (r_1, r_2)$ the set $\bigcup\{W_r \mid r \in V(R)$ and $r \geq r_2\}$ is X_e-normal.

Directed Tree-width

(Johnson, Robertson, Seymour and Thomas, 2001)

An *arboreal decomposition* of a graph G is a triple $(R, \mathcal{X}, \mathcal{W})$ where R is a directed tree, and $\mathcal{X} = \{X_e \mid e \in E(R)\}$, $\mathcal{W} = \{W_r \mid r \in V(R)\}$ are sets of vertices of G satisfying:

(R1) $\mathcal{W}$ is a partition of $V(G)$ into nonempty sets

(R2) for $e \in E(R)$, $e = (r_1, r_2)$ the set $\bigcup\{W_r \mid r \in V(R)$ and $r \geq r_2\}$ is X_e-normal.

The *width* of $(R, \mathcal{X}, \mathcal{W})$ is the least integer w such that for all $r \in V(R)$, $|W_r \cup \bigcup_{e \sim r} X_e| \leq w$.

Directed Tree-width

(Johnson, Robertson, Seymour and Thomas, 2001)

An *arboreal decomposition* of a graph G is a triple $(R, \mathcal{X}, \mathcal{W})$ where R is a directed tree, and $\mathcal{X} = \{X_e \mid e \in E(R)\}$, $\mathcal{W} = \{W_r \mid r \in V(R)\}$ are sets of vertices of G satisfying:

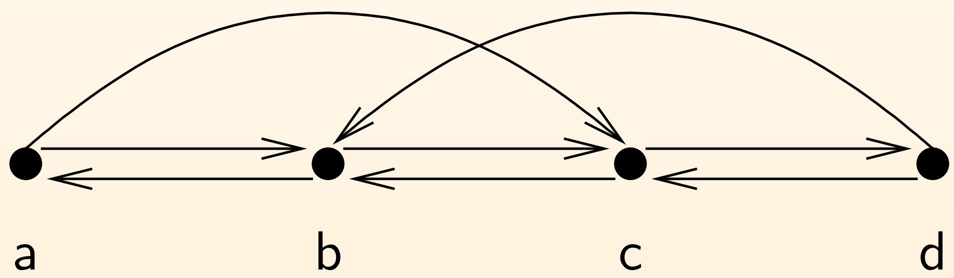
(R1) $\mathcal{W}$ is a partition of $V(G)$ into nonempty sets

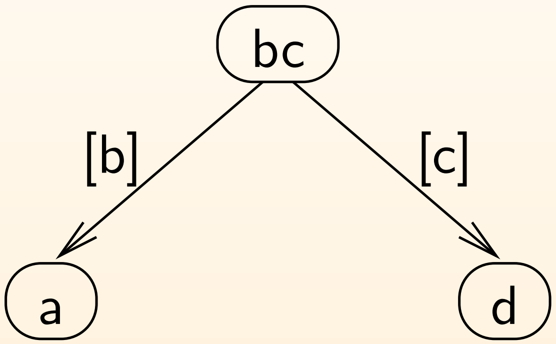
(R2) for $e \in E(R)$, $e = (r_1, r_2)$ the set $\bigcup\{W_r \mid r \in V(R)$ and $r \geq r_2\}$ is X_e-normal.

The *width* of $(R, \mathcal{X}, \mathcal{W})$ is the least integer w such that for all $r \in V(R)$, $|W_r \cup \bigcup_{e \sim r} X_e| \leq w$.

The *directed tree-width* of a graph G is the minimum width over all possible tree decompositions of G.

Directed Tree-width – an example





Directed Tree-width – the game

Essentially the same "cops and robber" game as for tree-width, with the following two restrictions on robber movements:

Directed Tree-width – the game

Essentially the same “cops and robber” game as for tree-width, with the following two restrictions on robber movements:

1. the robber must respect the *orientation of edges* of G

Directed Tree-width – the game

Essentially the same "cops and robber" game as for tree-width, with the following two restrictions on robber movements:

1. the robber must respect the *orientation of edges* of G
2. when the robber moves, he must stay in the *same strongly connected component*

Directed Tree-width – the game

Essentially the same "cops and robber" game as for tree-width, with the following two restrictions on robber movements:

1. the robber must respect the *orientation of edges* of G
2. when the robber moves, he must stay in the *same strongly connected component*

While (1) is natural, (2) does not "feel right".

DTW - game equivalence

We have:

- $dtw(G) \leq k-1 \implies k$ cops can search G [JRST01]
- Opposite implication **does not** hold. [O.]
- Monotone and non-monotone strategies for cops are not equivalent. [JRST01]

DTW - game equivalence

We have:

- $dtw(G) \leq k-1 \implies k$ cops can search G [JRST01]
- Opposite implication **does not** hold. [O.]
- Monotone and non-monotone strategies for cops are not equivalent. [JRST01]

Strategies for the robber:

Haven is a function σ assigning each set $X \in V(G)$ s.t. $|X| \leq k$ a *strongly connected component* $\sigma(X)$ of $G \setminus X$ s.t. for any $X, Y \in V(G)$ of size less than k we have $\sigma(X)$ *touches* $\sigma(Y)$.

DAG-width

A *DAG-decomposition* of a (directed) graph G is a pair $(D, \mathcal{X})$ where D is a DAG and $\mathcal{X} = \{X_d \mid d \in V(D)\}$ is a family of subsets of $V(G)$ satisfying:

(D1) $V(G) = \bigcup_{d \in V(D)} X_d$

(D2) if $(d, d') \in E(D)$, then for each $(v, w) \in E(G)$ s.t. $v \in X_{\geq d'} \setminus X_d$ we have $w \in X_{\geq d'}$, where $X_{\geq c} = \bigcup_{c' \geq c} X_{c'}$. If d' is a root we replace X_d with $\emptyset$.

(D3) for all $d, d', d'' \in D$ if d' lies on (some) path from d to d'', then $X_d \cap X_{d''} \subseteq X_{d'}$.

DAG-width

A *DAG-decomposition* of a (directed) graph G is a pair $(D, \mathcal{X})$ where D is a DAG and $\mathcal{X} = \{X_d \mid d \in V(D)\}$ is a family of subsets of $V(G)$ satisfying:

(D1) $V(G) = \bigcup_{d \in V(D)} X_d$

(D2) if $(d, d') \in E(D)$, then for each $(v, w) \in E(G)$ s.t. $v \in X_{\geq d'} \setminus X_d$ we have $w \in X_{\geq d'}$, where $X_{\geq c} = \bigcup_{c' \geq c} X_{c'}$. If d' is a root we replace X_d with $\emptyset$.

(D3) for all $d, d', d'' \in D$ if d' lies on (some) path from d to d'', then $X_d \cap X_{d''} \subseteq X_{d'}$.

The *width* of a DAG decomposition $(D, \mathcal{X})$ is $\max_{d \in D} |X_d| - 1$.

DAG-width

A *DAG-decomposition* of a (directed) graph G is a pair $(D, \mathcal{X})$ where D is a DAG and $\mathcal{X} = \{X_d \mid d \in V(D)\}$ is a family of subsets of $V(G)$ satisfying:

(D1) $V(G) = \bigcup_{d \in V(D)} X_d$

(D2) if $(d, d') \in E(D)$, then for each $(v, w) \in E(G)$ s.t. $v \in X_{\geq d'} \setminus X_d$ we have $w \in X_{\geq d'}$, where $X_{\geq c} = \bigcup_{c' \geq c} X_{c'}$. If d' is a root we replace X_d with $\emptyset$.

(D3) for all $d, d', d'' \in D$ if d' lies on (some) path from d to d'', then $X_d \cap X_{d''} \subseteq X_{d'}$.

The *width* of a DAG decomposition $(D, \mathcal{X})$ is $\max_{d \in D} |X_d| - 1$.

The *DAG-width* of a graph G is the minimum width over all possible DAG decompositions of G.

DAG-width

A *DAG-decomposition* of a (directed) graph G is a pair $(D, \mathcal{X})$ where D is a DAG and $\mathcal{X} = \{X_d \mid d \in V(D)\}$ is a family of subsets of $V(G)$ satisfying:

(D1) $V(G) = \bigcup_{d \in V(D)} X_d$

(D2) if $(d, d') \in E(D)$, then for each $(v, w) \in E(G)$ s.t. $v \in X_{\geq d'} \setminus X_d$ we have $w \in X_{\geq d'}$, where $X_{\geq c} = \bigcup_{c' \geq c} X_{c'}$. If d' is a root we replace X_d with $\emptyset$.

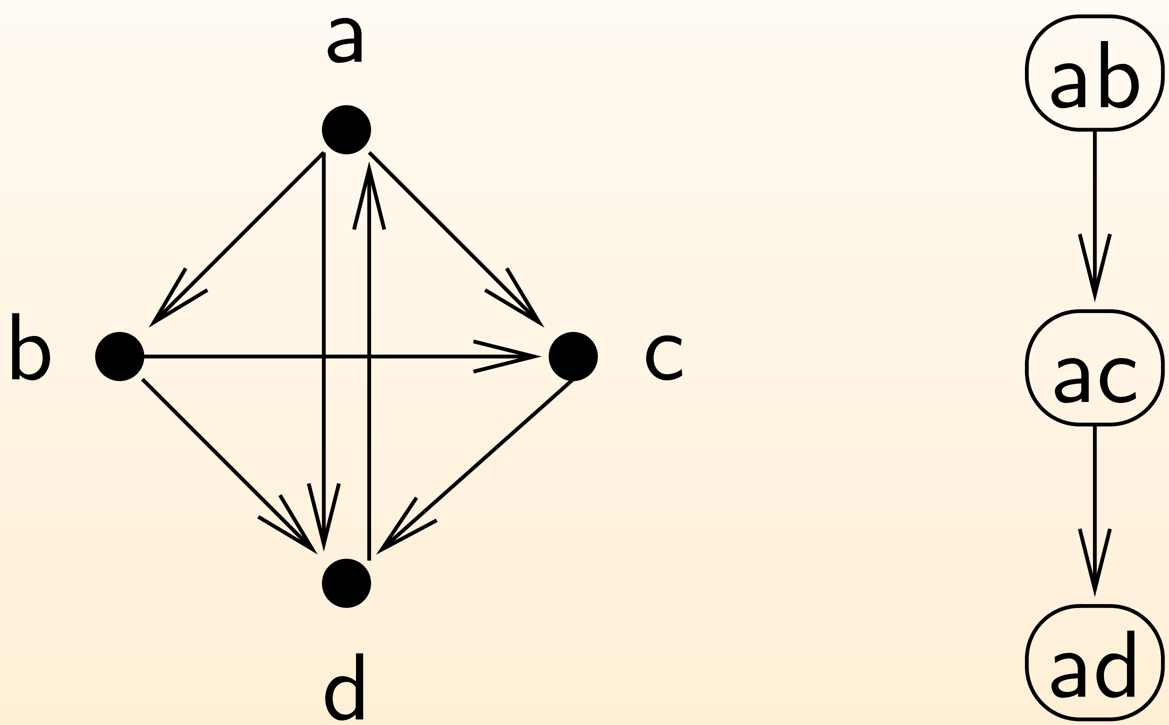
(D3) for all $d, d', d'' \in D$ if d' lies on (some) path from d to d'', then $X_d \cap X_{d''} \subseteq X_{d'}$.

The *width* of a DAG decomposition $(D, \mathcal{X})$ is $\max_{d \in D} |X_d| - 1$.

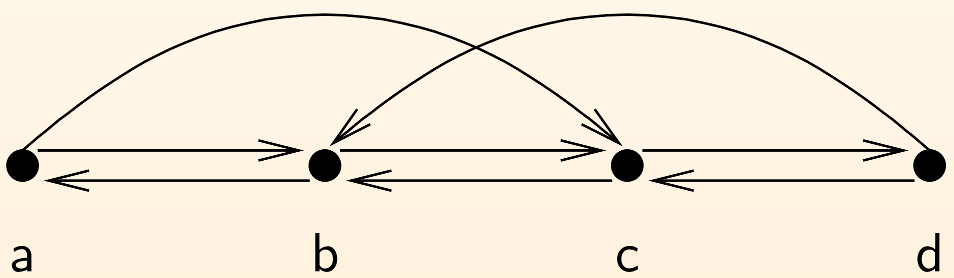
The *DAG-width* of a graph G is the minimum width over all possible DAG decompositions of G.

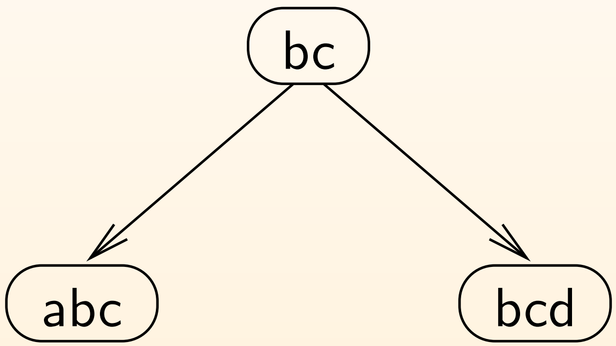
DAGs have DAG-width zero.

DAG-width – an example



DAG-width – an example





DAG-width – the game

The same "cops and robber" game as for tree-width, except for the fact that the robber must respect the orientation of edges of G.

Duality theorem:
The following are equivalent:

1. k cops can search G
2. G has DAG-width at most $k - 1$

DAG-width – the game

The same "cops and robber" game as for tree-width, except for the fact that the robber must respect the orientation of edges of G.

Duality theorem:
The following are equivalent:

1. k cops can search G
2. G has DAG-width at most $k-1$

We do not know whether monotone and general strategies are equivalent.

DAG-width – the game

The same "cops and robber" game as for tree-width, except for the fact that the robber must respect the orientation of edges of G.

Duality theorem:
The following are equivalent:

1. k cops can search G
2. G has DAG-width at most $k-1$

We do not know whether monotone and general strategies are equivalent.

Havens cannot be connected components.

DAG-width and tree-width

$$\text{DAG-width}(G) \leq \text{tree-width}(G)$$

Easy to prove using games.

The inequivalence may be sharp – e.g. for directed cliques.

DAG-width and directed tree-width

$$\text{directed tree-width}(G) \leq \text{DAG-width}(G)$$

- This is up to a constant factor.
- We cannot use the game characterization directly here, as the cops strategies for directed tree-width are not cop monotone.
- Conversion of a DAG-decomposition into an arboreal decomposition of the same width is not possible in a *strict* sense.
- Inequivalence can be sharp.

Algorithms

The following in general NP-complete problems can be solved in polynomial time on graphs of bounded DAG-width:

- Hamiltonian cycle, Hamiltonian path

Algorithms

The following in general NP-complete problems can be solved in polynomial time on graphs of bounded DAG-width:

- Hamiltonian cycle, Hamiltonian path

- even cycle through a specified vertex

Algorithms

The following in general NP-complete problems can be solved in polynomial time on graphs of bounded DAG-width:

- Hamiltonian cycle, Hamiltonian path
- even cycle through a specified vertex
- ...

Algorithms

The following in general NP-complete problems can be solved in polynomial time on graphs of bounded DAG-width:

- Hamiltonian cycle, Hamiltonian path
- even cycle through a specified vertex
- ...
- *parity games*

Algorithms

The following in general NP-complete problems can be solved in polynomial time on graphs of bounded DAG-width:

- Hamiltonian cycle, Hamiltonian path
- even cycle through a specified vertex
- ...
- *parity games*

Note: In the case of tree-width the algorithms usually run in linear time.

Future work

Are monotone and non-monotone strategies equivalent?

Future work

Are monotone and non-monotone strategies equivalent?

Is there a notion of decomposition linear in the size of G?

Future work

Are monotone and non-monotone strategies equivalent?

Is there a notion of decomposition linear in the size of G?

Which otherwise NP-complete problems can be solved in polynomial time on graphs of bounded DAG-width?

Thanks for your attention.