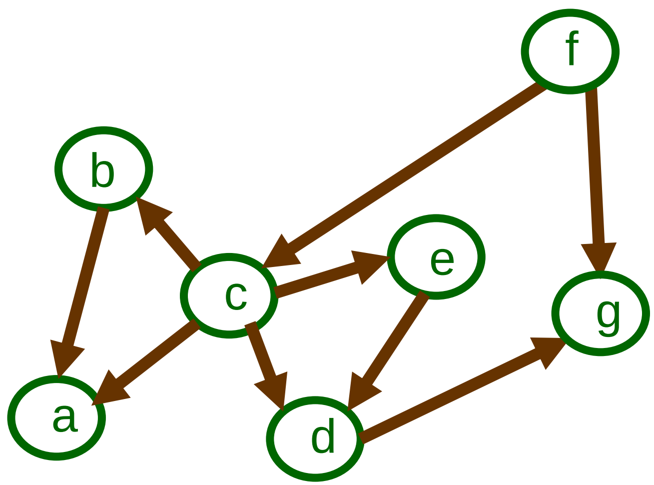
A polynomial time algorithm for bounded directed pathwidth

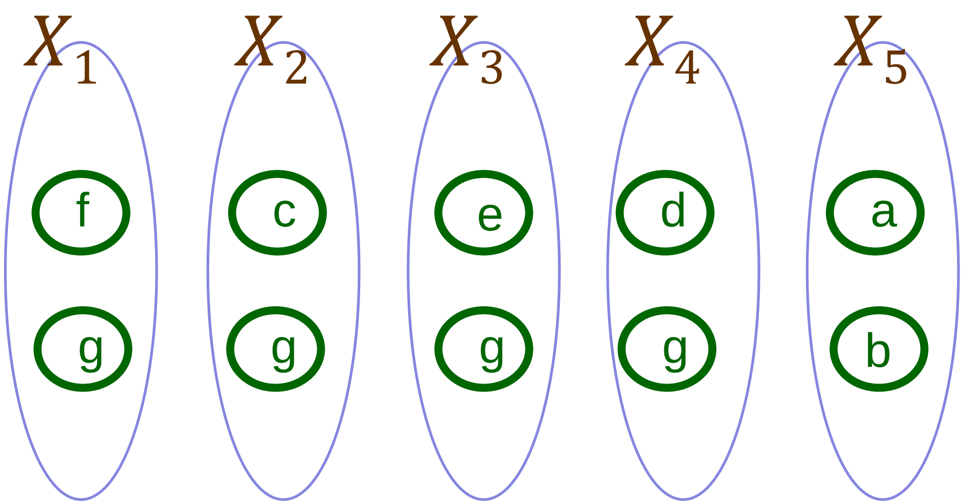
Hisao Tamaki
Meiji University

Directed pathwidth/decomposition

G



A directed path-decomposition of G

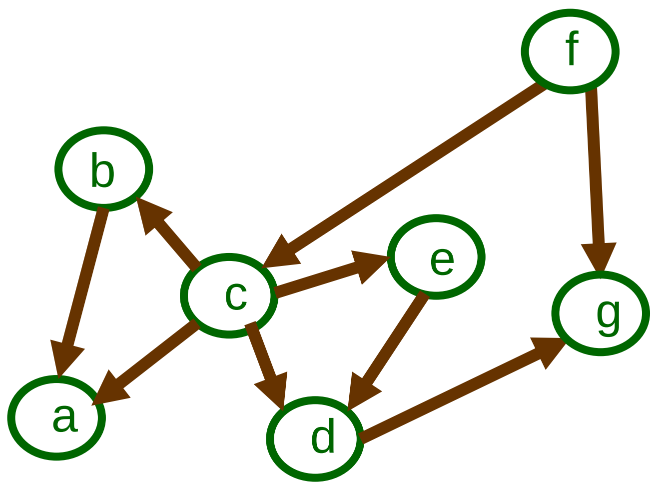


1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \le j$ such that $u \in X_i$ and $v \in X_j$

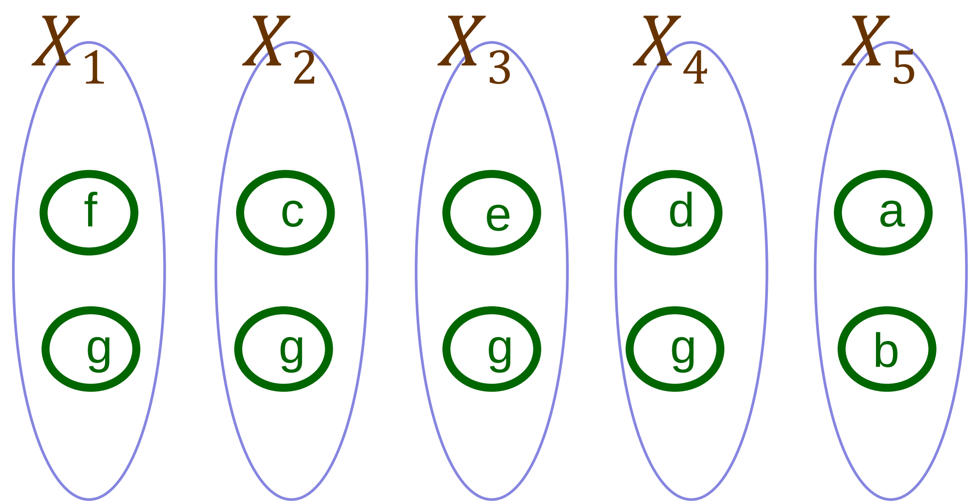
Undirected

Directed pathwidth/decomposition

G



A directed path-decomposition of G



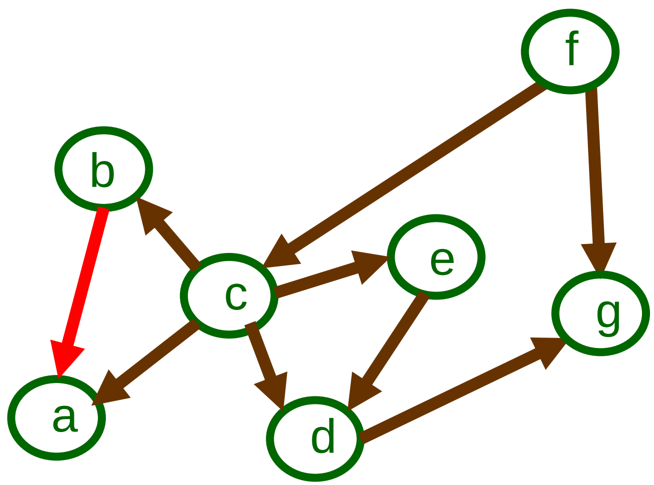
1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval

   $\{u, v\}$ some i

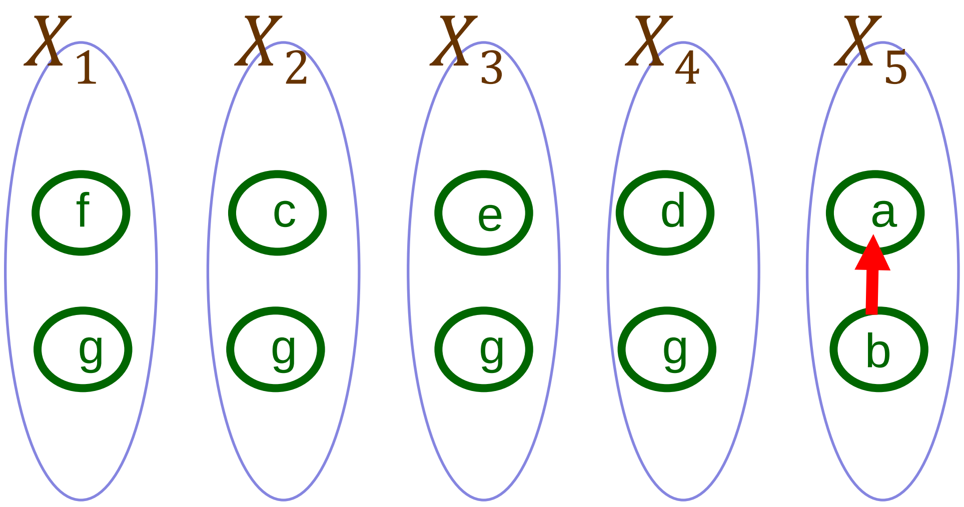
2. for each ~~directed~~ edge (u, v) there is a pair $i \leq j$ such that $u, v \in X_i$ $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

G

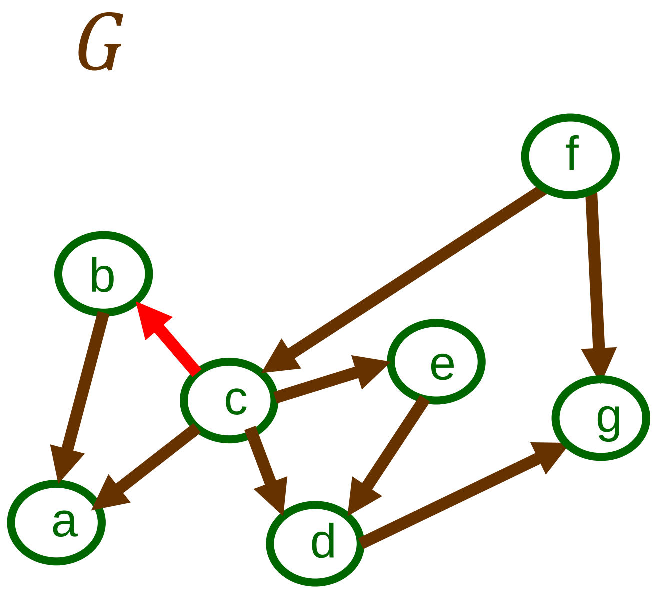


A directed path-decomposition of G

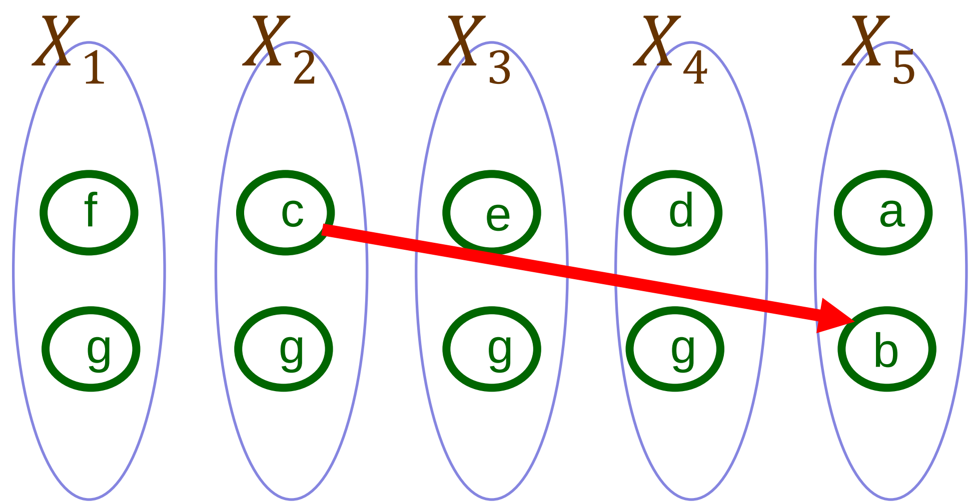


1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition



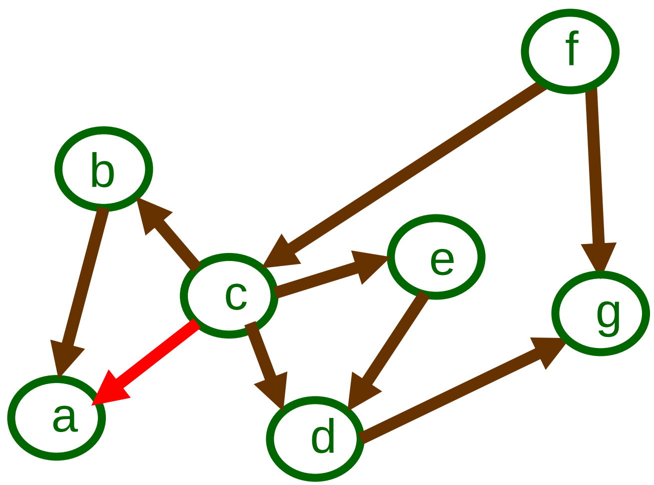
A directed path-decomposition of G



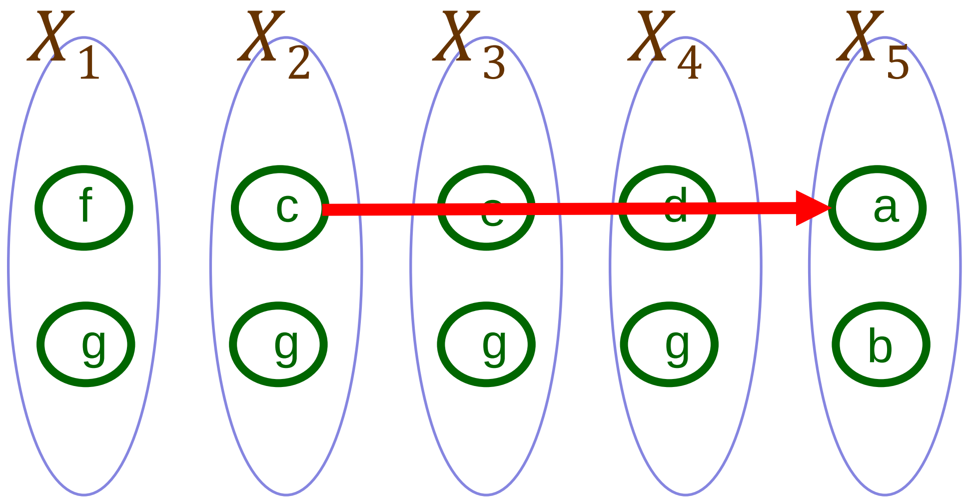
1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

G



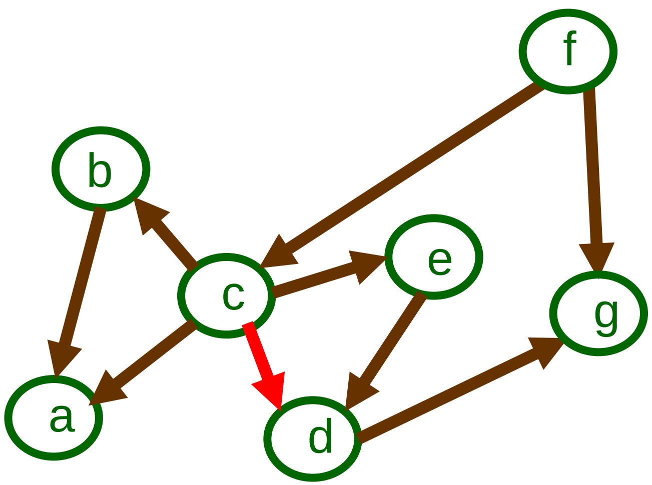
A directed path-decomposition of G



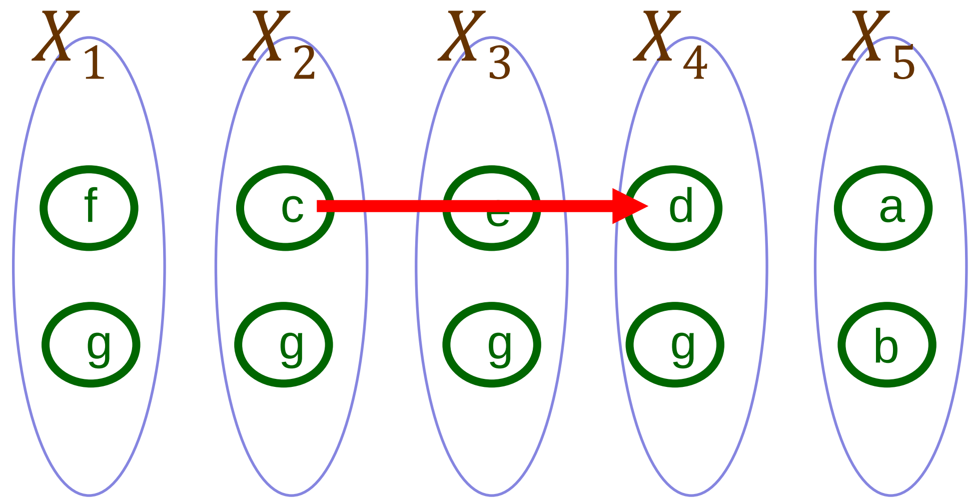
1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

G



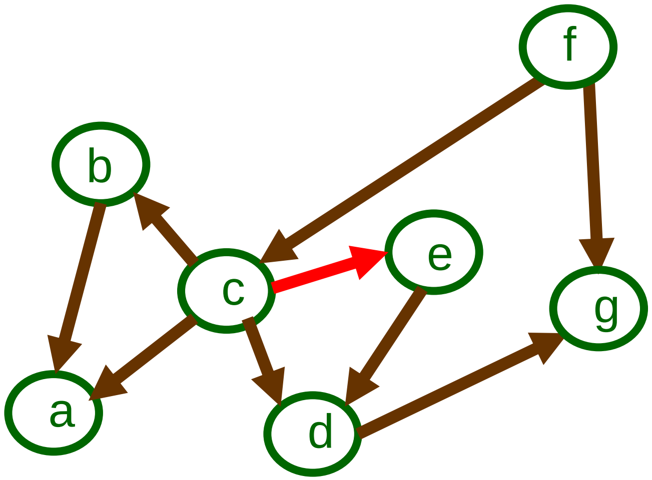
A directed path-decomposition of G



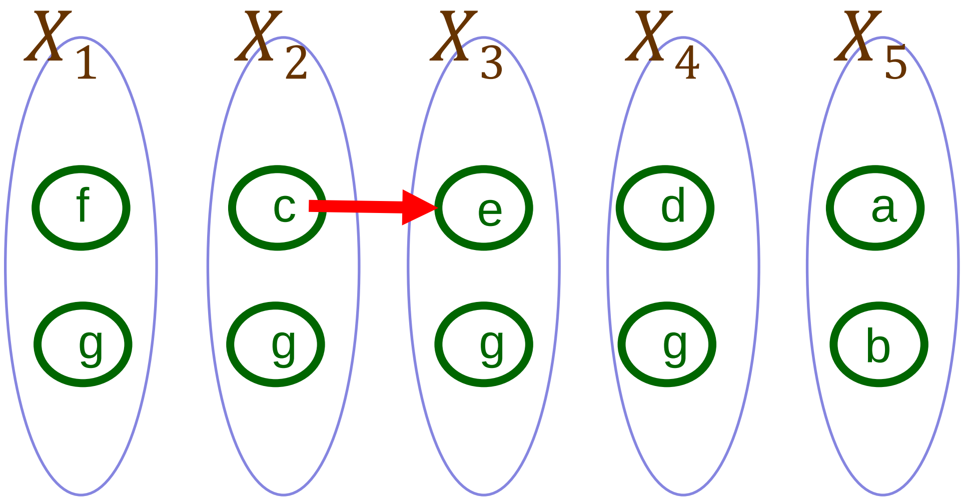
1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

G



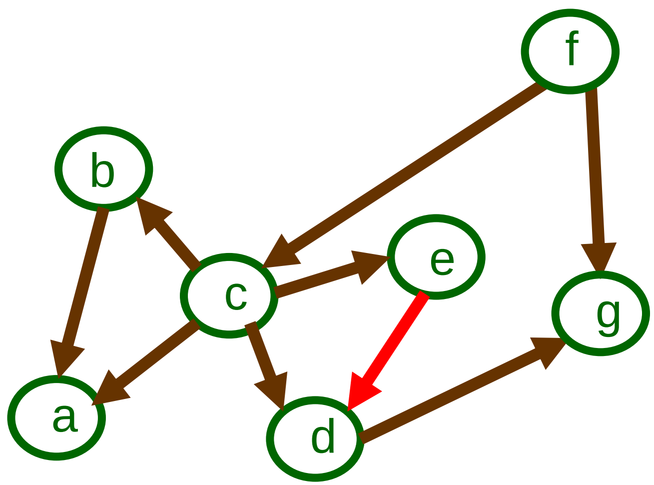
A directed path-decomposition of G



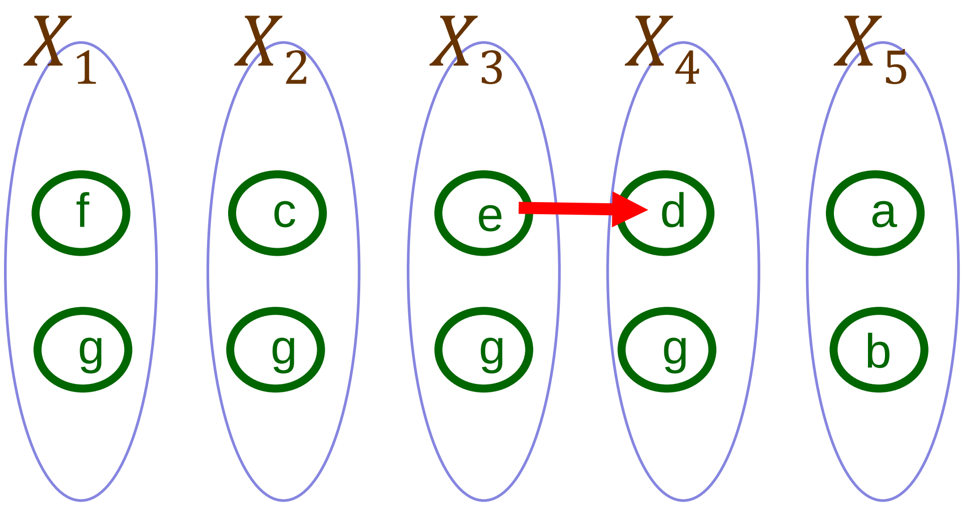
1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

G



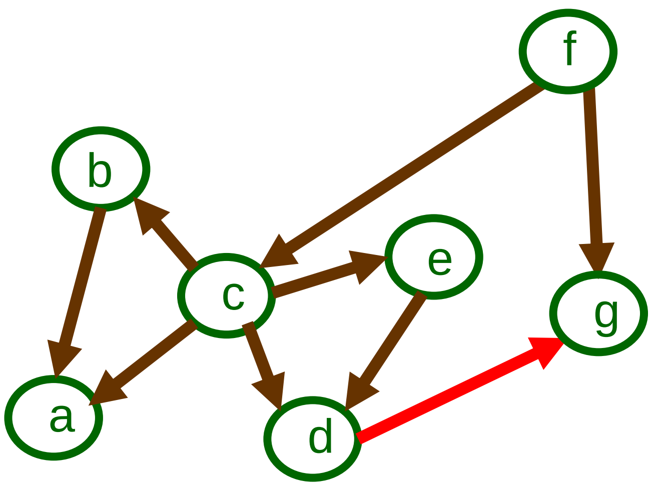
A directed path-decomposition of G



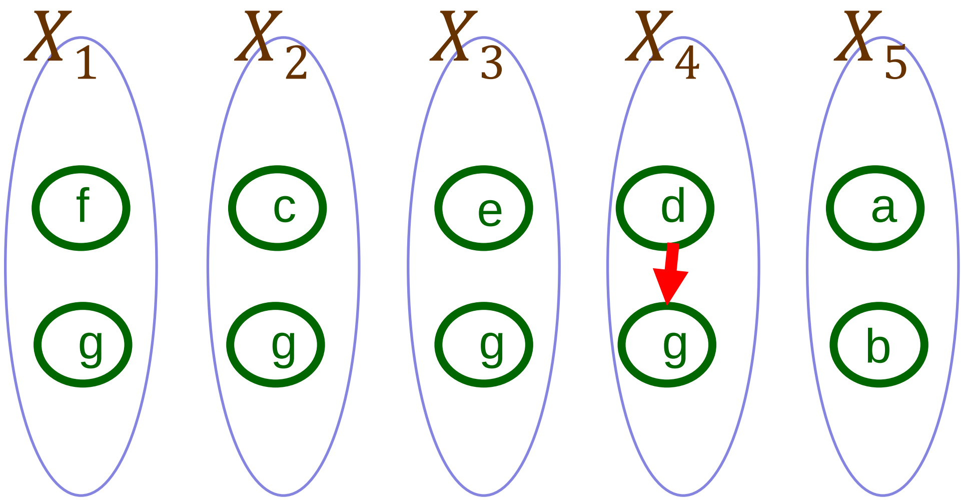
1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

G

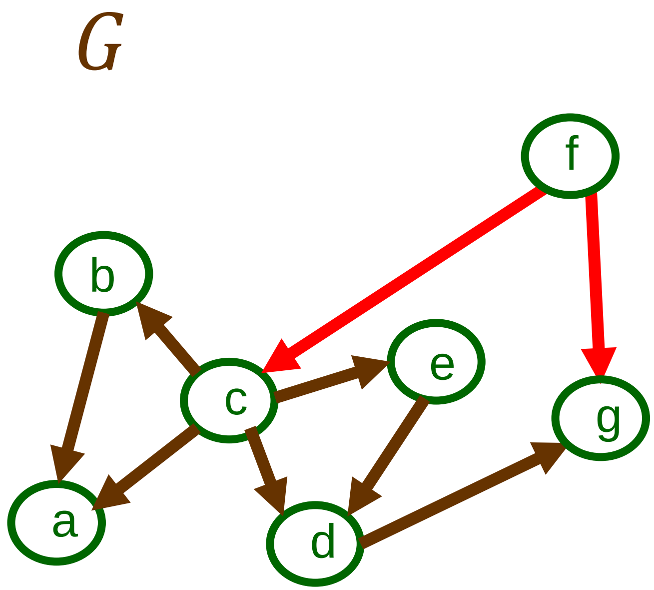


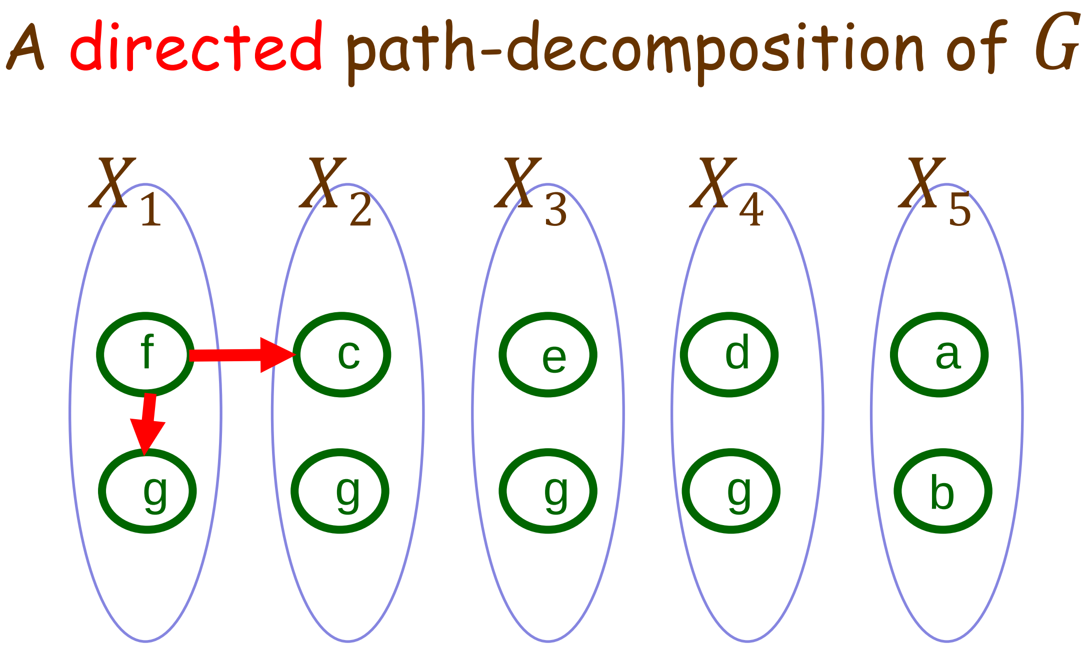
A directed path-decomposition of G



1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

Directed pathwidth/decomposition

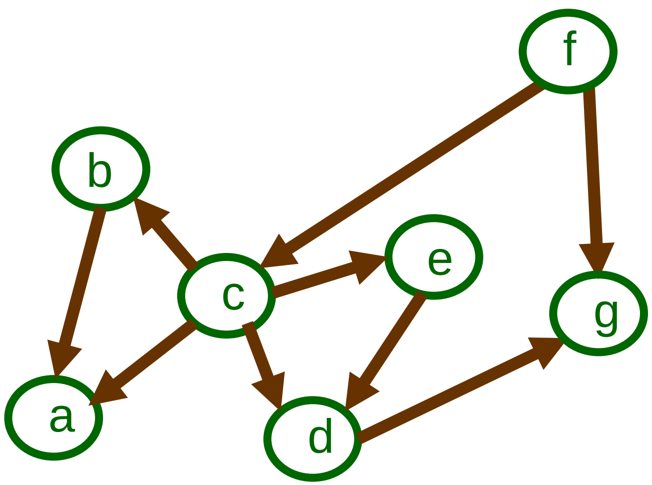




1. for each $v \in V(G)$, $I_v = \{i \mid v \in X_i\}$ is a single non-empty interval
2. for each directed edge (u, v) there is a pair $i \leq j$ such that $u \in X_i$ and $v \in X_j$

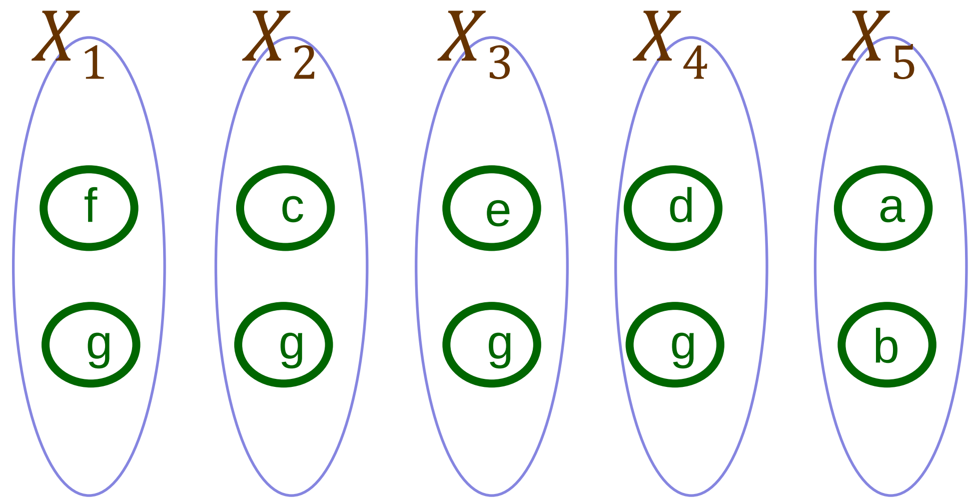
Directed pathwidth/decomposition

G



$\mathbf{dpw}(G) = 1$

A directed path-decomposition of G



width = 1

The width of a directed path-decomposition is $max_i\ |X_i|$ - 1.

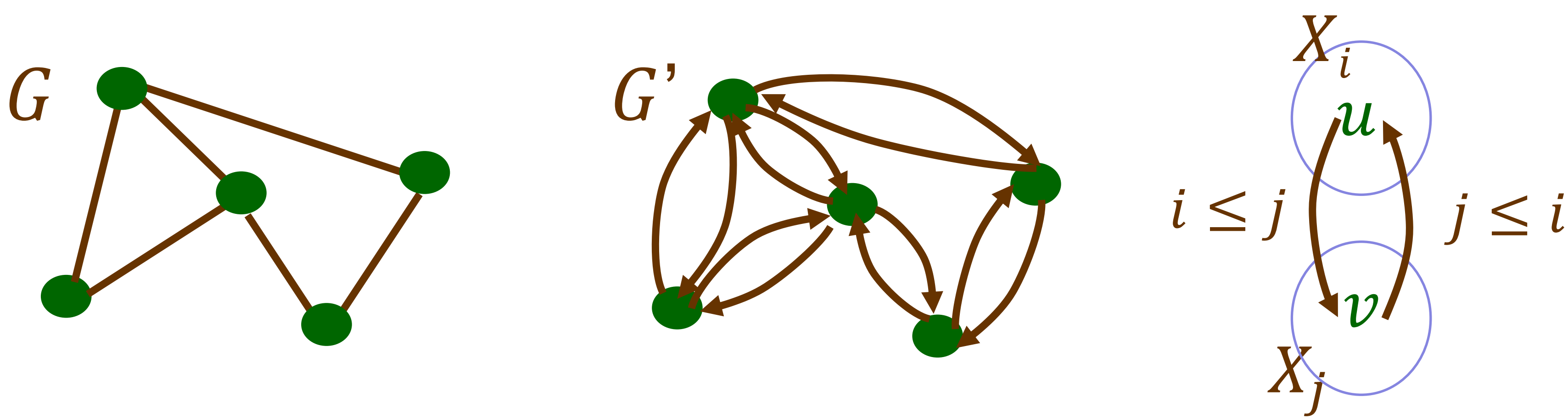
The directed pathwidth of G is the minimum w such that there is a directed path-decomposition of G of width w.

Observation 1

The problem of deciding the directed-pathwidth is a generalization of that of deciding the pathwidth.

G: undirected graph

G': digraph with a pair of anti-parallel edges for each edge of G

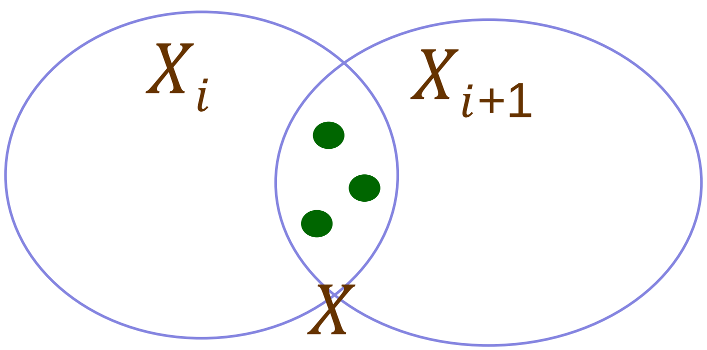


The condition for a path-decomposition of G

= the condition for a directed path-decomposition of G'

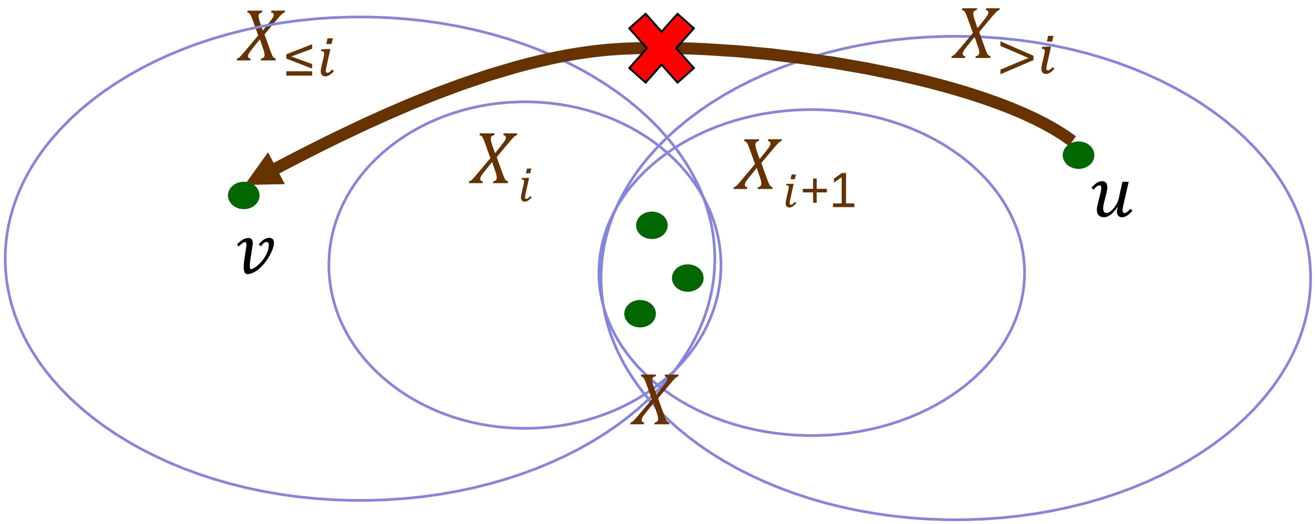
Observation 2

A directed path decomposition represents a linear system of dicuts.



Observation 2

A directed path-decomposition represents a linear system of dicuts of size at most the width.



Some facts on directed pathwidth

Introduced by Reed, Seymour, and Thomas in mid 90's.

Relates to directed treewidth [Johnson, Robertson, Seymour and Thomas 01], D-width [Safari 05], Dag-width [Berwanger, Dawar, Hunter & Kreutzer 05, Obdrzalek 06], and Kelly-width[Hunter & Kreutzer 07] as pathwidth relates to treewidth.

For digraphs of directed pathwidth w, some problems including directed Hamiltonian cycle can be solved in $n^{O(w)}$ time [JRST01] .

Used in a heuristic algorithm for enumerating attractors of boolean networks [Tamaki 10].

Complexity

Input: positive integer k and graph (digraph) G

Question: Is the (directed) pathwidth of G at most k?

NP-complete for the undirected case [Kashiwabara & Fujisawa 79] and hence for the directed case.

Undirected pathwidth is fixed parameter tractable:

$f(k)n^{O(1)}$ time: graph minor theorem

$2^{O(k^3)}\,n$ time: [Bodlaender 96, Bodlaender & Kloks 96]

Directed pathwidth is open for FPT

Even for k = 2, no polynomial time was previously known.

Result

An $O(mn^{k+1})$ time algorithm for deciding if the directed pathwidth is $\leq k$ (and constructing the associated decomposition) for a digraph of n vertices and m edges.

Note This algorithm is extremely simple, easy to implement, and useful even for undirected pathwidth/-decomposition

(the linear time algorithm of Bodlaender depends exponentially on k^3)

Notation

G : digraph, fixed

$n = |V(G)|$

$m = |E(G)|$

$N^-(X) = \{u \in X \mid (u, v) \in E(G), v \in X\}$

: set of in-neighbors of $X \subseteq V(G)$

$d^-(X) = |N^-(X)|$: in-degree of $X \subseteq V(G)$

$\Sigma(G)$: the set of all non-duplicating sequences of vertices of G

$V(\sigma)$: the set of vertices appearing in $\sigma \in \Sigma(G)$

Directed vertex separation number

A vertex sequence $\sigma \in \Sigma(G)$ is *k-feasible* if

$d^-(V(\tau)) \leq k$ for every prefix τ of σ.

The *directed vertex separation number* of G:

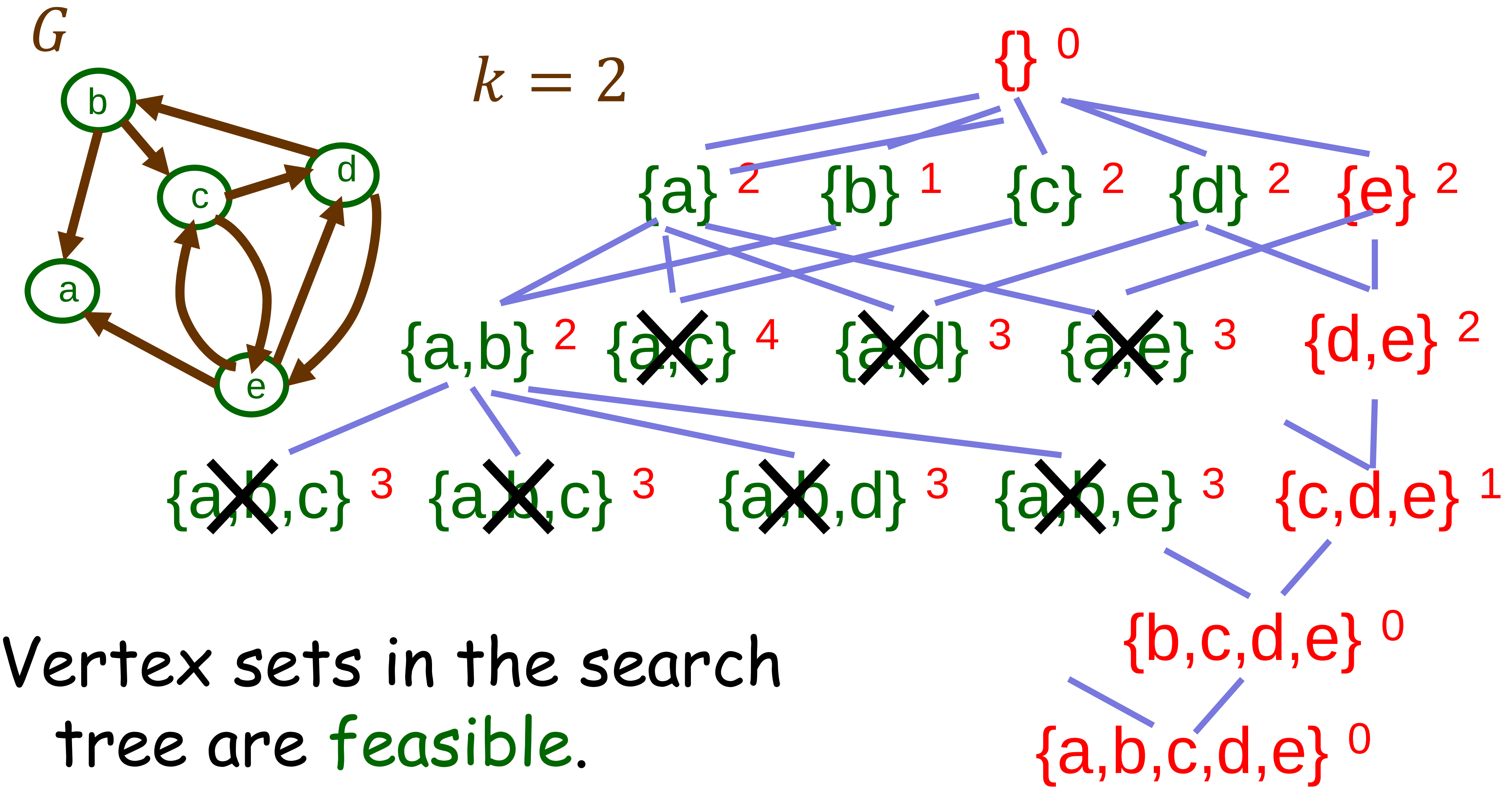
dvsn(G)

$= \min\{ k \mid \sigma \in \Sigma(G): V(\sigma) = V(G)$ and σ is k-feasible$\}$

Fact: **dvsn**(G) = **dpw**(G)

The conversion from a vertex separation sequence to a directed path-decomposition is straightforward.

Search tree for k-feasible sequences



Vertex sets in the search tree are feasible.

Those leading to a solution are strongly feasible.

Commitment: a special case

$U^{\,d}$

Adding v does not increase the indegree.

$U \cup \{v\}^{\,\leq d}$

Is it safe to commit to this child?

In other words, is it true that

if U is strongly feasible then $U \cup \{v\}$ is?

Commitment: a special case

$U^{\ d}$

Adding v does not increase the in-degree.

$U \cup \{v\}^{\ \leq d}$

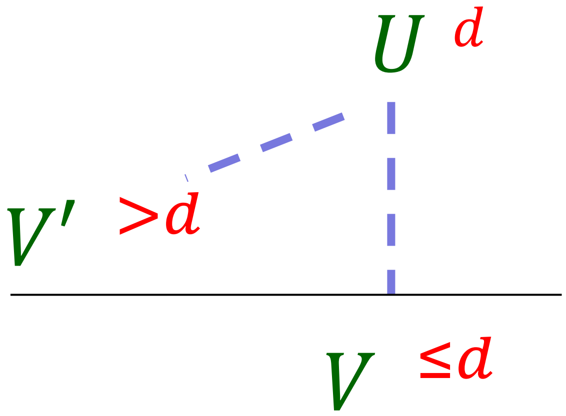
Is it safe to commit to this child?

In other words, is it true that

if U is strongly feasible then $U \cup \{v\}$ is?

YES, in a more genral form

Commitment in general form



First decsendant with the same or smaller in-degree.

Commitment lemma

Let $U \subset V$ both feasible and suppose:

1. $d^{-}(V) \leq d^{-}(U)$,
2. $d^{-}(V') > d^{-}(U)$ for every feasible proper superset V' of U strictly smaller than V, and
3. U is strongly feasible.

Then V is strongly feasible.

Search tree pruning based on commitment

When a node has a descendant to which it can commit, all other descendants are removed from the tree.

Effectively, branching occurs only when the in-degree increases.

The pruned tree behaves like a depth k tree in a fuzzy sense.

Can show, with some technicality, that the size of the pruned tree is at most n^{k+1} .

Proof of the commitment lemma

Fact:

The in-degree function d^{-} is submodular:

for every pair of subsets $X, Y \subseteq V(G)$,

$$d^{-}(X) + d^{-}(Y) \geq d^{-}(X \cap Y) + d^{-}(X \cup Y)$$

Proof of the commitment lemma

Step 1:

Let U and V as in the lemma. Then, $d^{-}(V) \leq d^{-}(X)$ holds for every X such that $U \subseteq X \subseteq V$. (Even if X is not feasible.)

Step 2:

Using the condition established in Step 1, derive the strong feasibility of V from that of U.

Step 2 of the proof

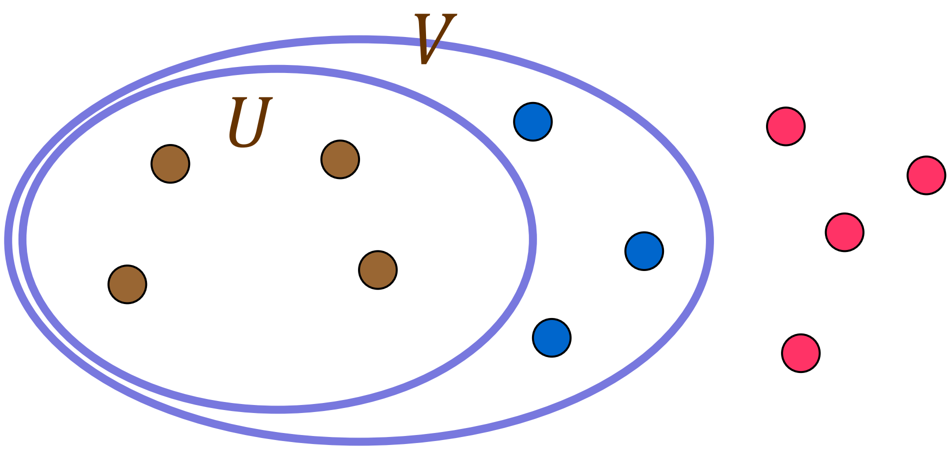
Assumptions:

$U \subset V$,

U is strongly feasible, V is feasible, and

$d^{-}(V) \leq d^{-}(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: V is also strongly feasible



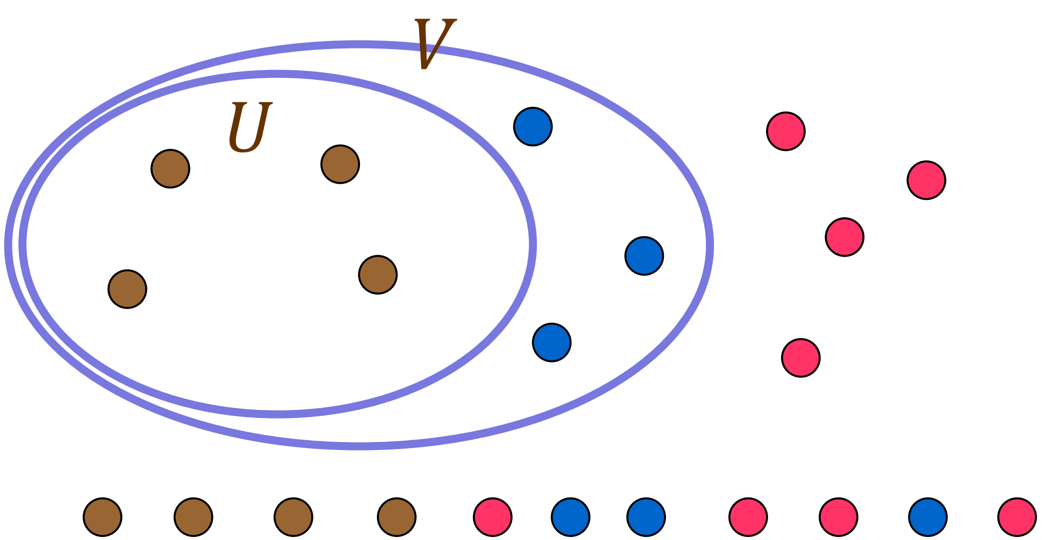
Step 2 of the proof

Assumptions:

U is strongly feasible, V is feasible, and

$d^{-}(V) \leq d^{-}(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: V is also strongly feasible



σ : k-feasible

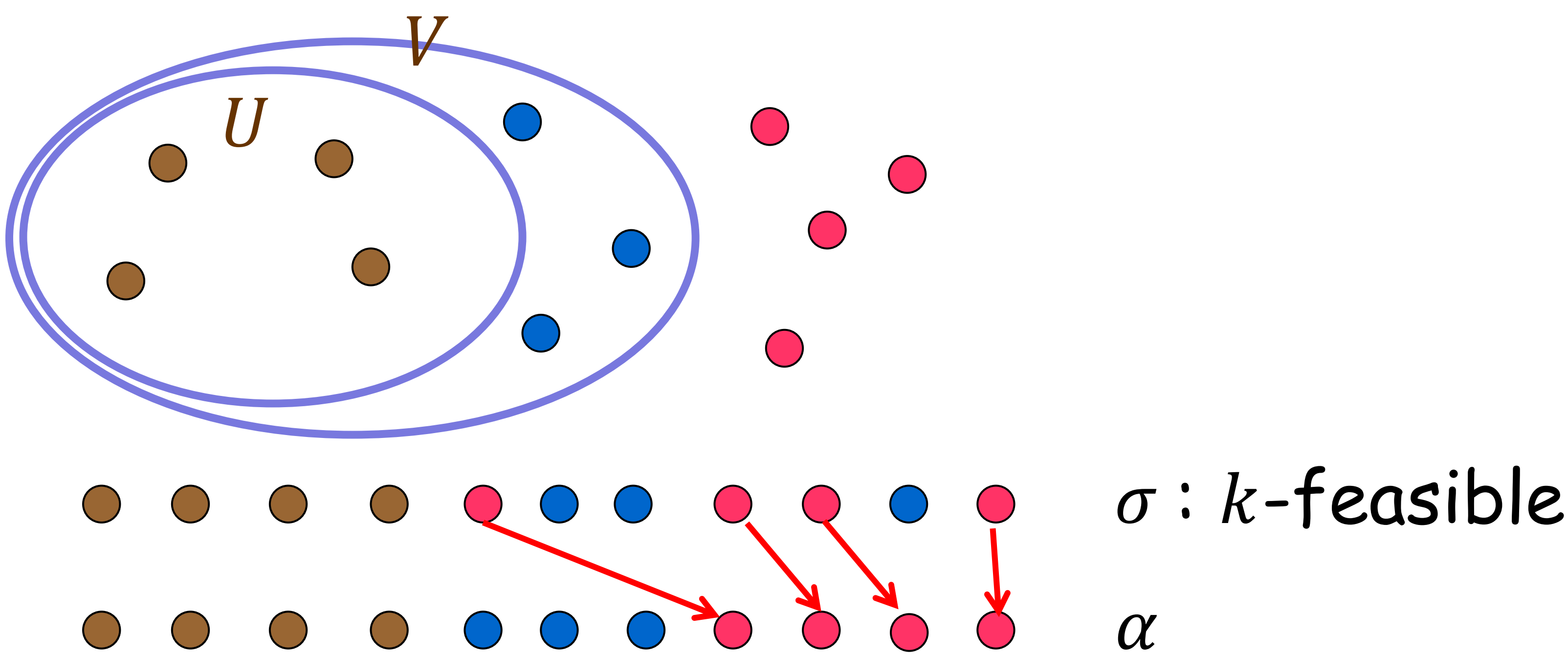
Step 2 of the proof

Assumptions:

V is feasible, and

$d^-(V) \leq d^-(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: α is k-feasible



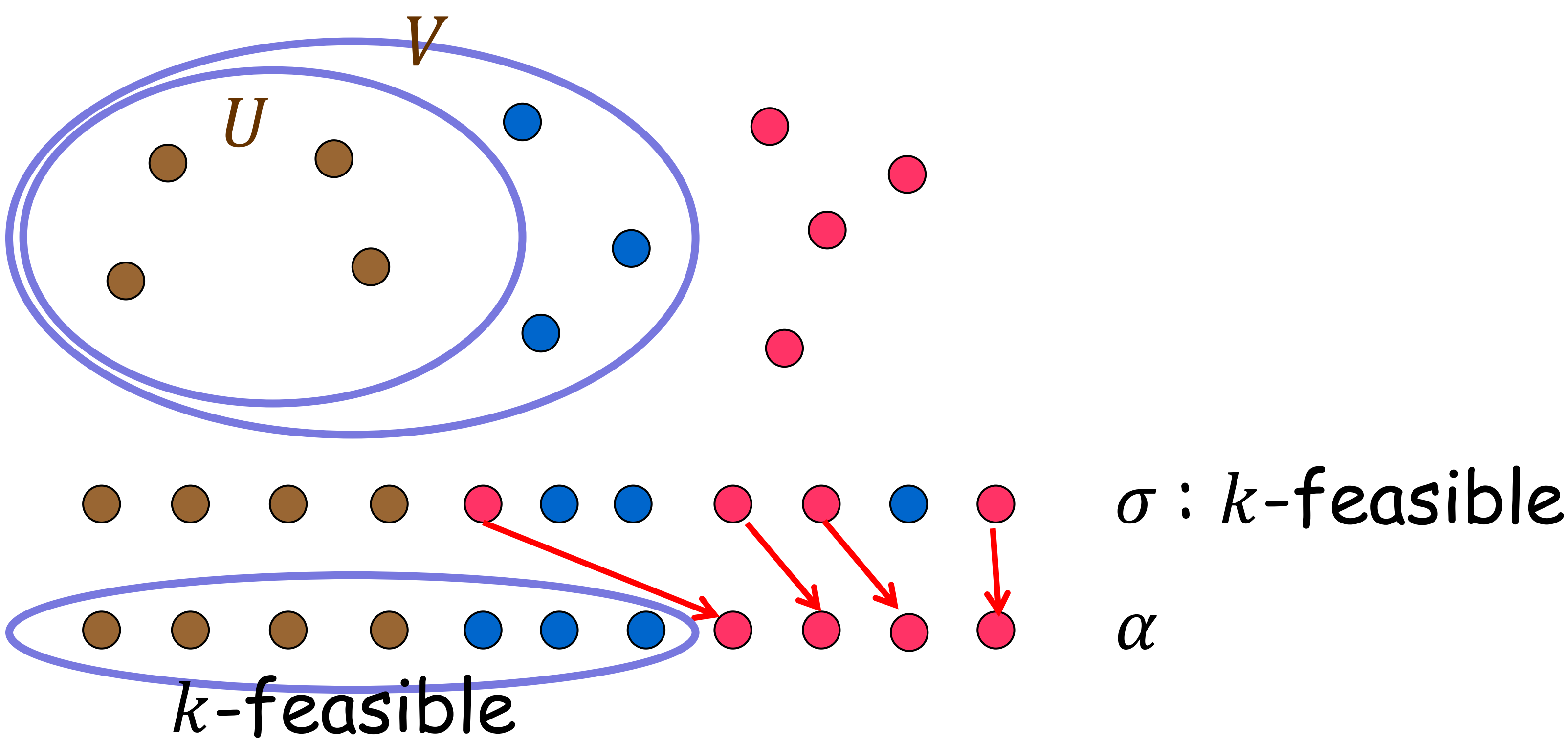
Step 2 of the proof

Assumptions:

V is feasible, and

$d^-(V) \leq d^-(X)$ for every X such that $U \subseteq X \subseteq V$.

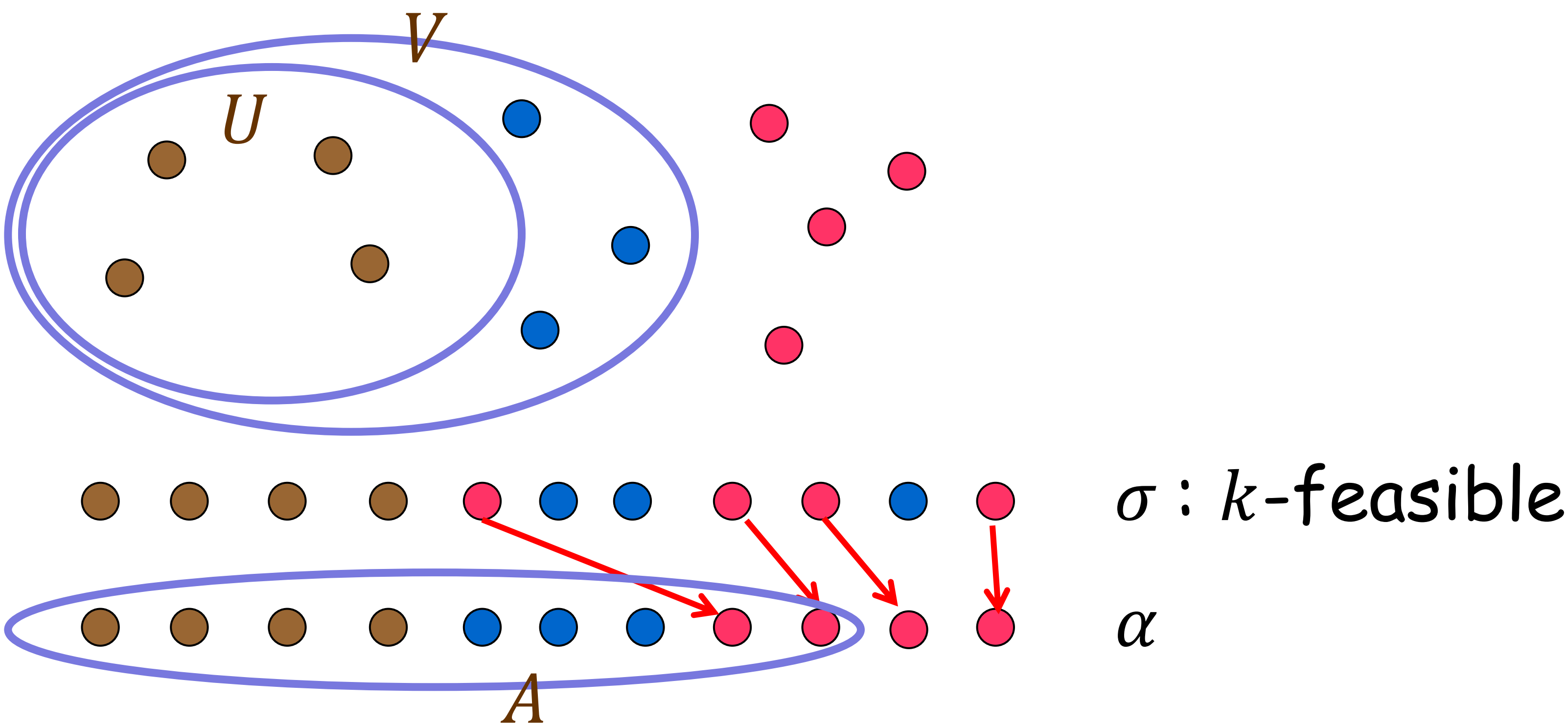
Goal: α is k-feasible



Step 2 of the proof

$d^-(V) \leq d^-(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: $d^-(A) \leq k$ for each superset A of V that is the vertex set of some prefix of α



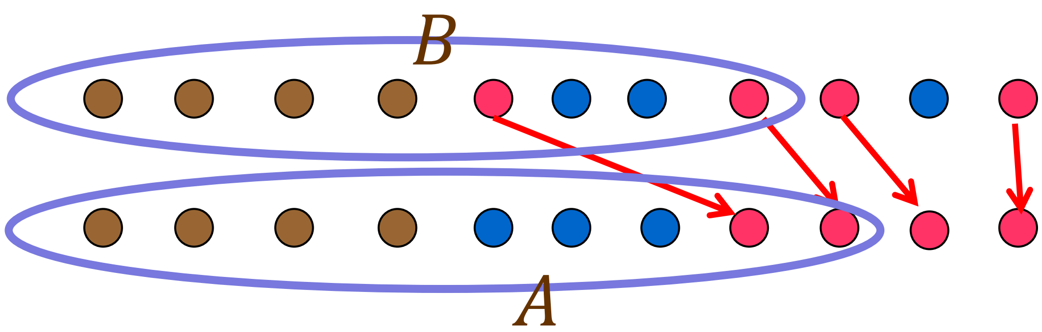
Step 2 of the proof

$d^-(V) \leq d^-(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: $d^-(A) \leq k$

B: the vertex set of the minimal prefix of σ such that

$$B \setminus V = A \setminus V$$



σ : k-feasible

α

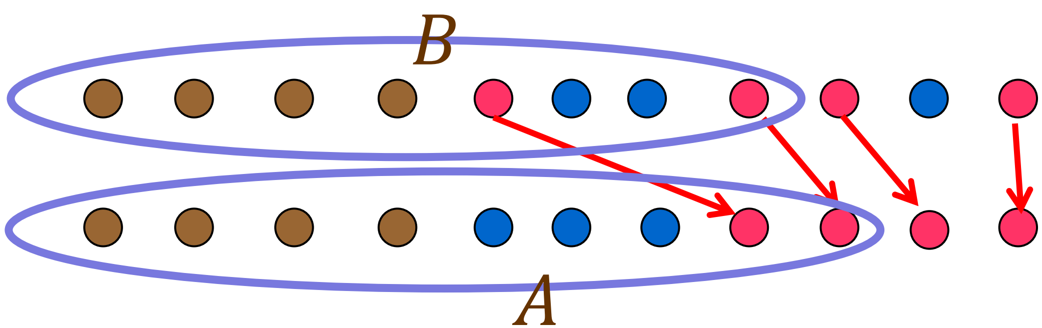
Step 2 of the proof

$d^{-}(V) \leq d^{-}(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: $d^{-}(A) \leq k$

By submodularity:

$$d^{-}(V) + d^{-}(B) \geq d^{-}(\overset{X}{V \cap B}) + d^{-}(\overset{A}{V \cup B})$$



σ : k-feasible

α

Step 2 of the proof

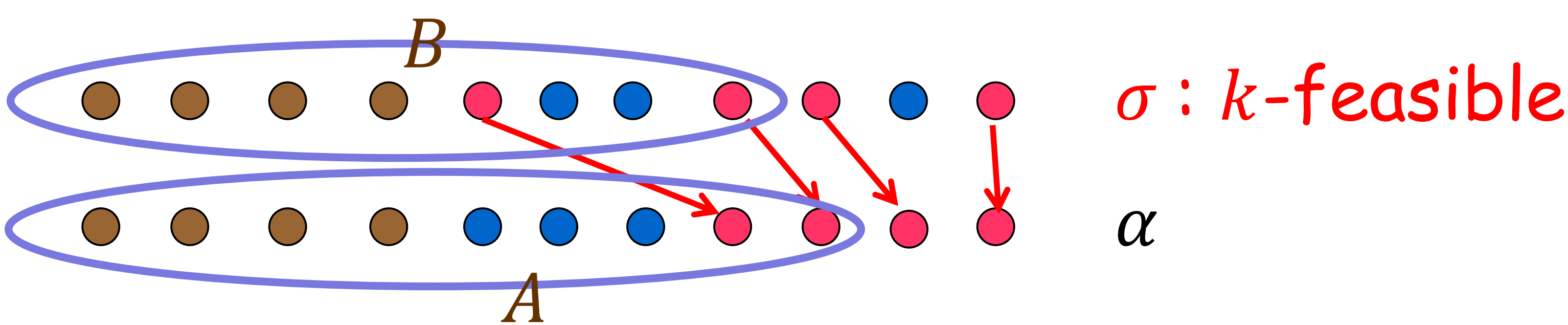
$d^{-}(V) \leq d^{-}(X)$ for every X such that $U \subseteq X \subseteq V$.

Goal: $d^{-}(A) \leq k$

By submodularity:

$$\underbrace{d^{-}(V)}_{} + \underbrace{d^{-}(B)}_{\leq k} \geq d^{-}(\overset{X}{V \cap B}) + \underbrace{d^{-}(\overset{A}{V \cup B})}_{}$$

$$\leq k \qquad \geq d^{-}(V)$$



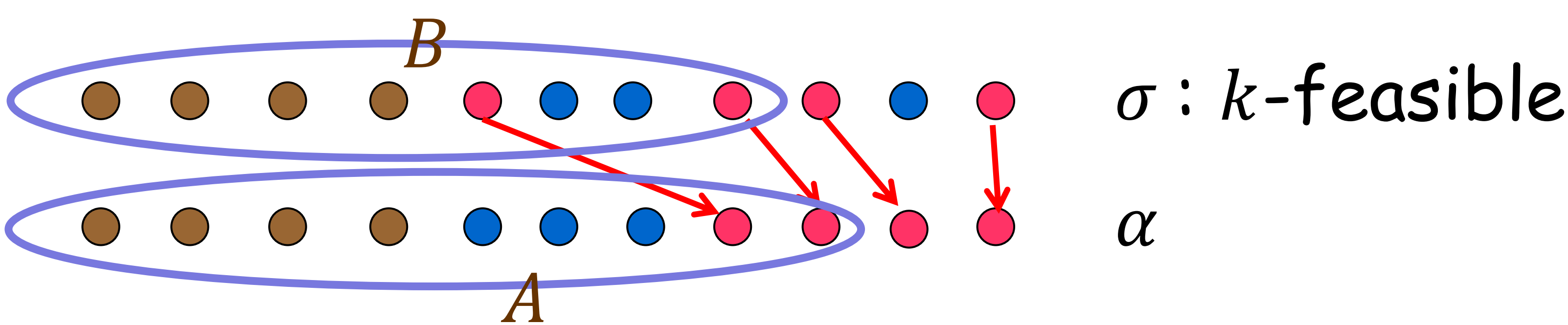
Step 2 of the proof

Goal: $d^-(A) \le k$: established

$\Rightarrow$ α is k-feasible $\Rightarrow$ V is strongly feasible

By submodularity:

$$d^-(V) + \underbrace{d^-(B)}_{\le k} \ge \overbrace{\underbrace{d^-(V \cap B)}_{\ge d^-(V)}}^{X} + \overbrace{d^-(V \cup B)}^{A}$$



Open problems

Is directed-pathwidth FPT, i.e., has an $f(k)n^{O(1)}$ time algorithm?

Are directed-treewidth, D-width, Dag-width, and Kelly-width in XP, i.e., has an $n^{O(k)}$ time algorithm?

More applications?