

On Generation of Cut Conjunctions, Minimal k -Connected Spanning Subgraphs, Minimal Connected and Spanning Subsets and Vertices

Konrad Borys

`kborys@eden.rutgers.edu`

Rutgers University

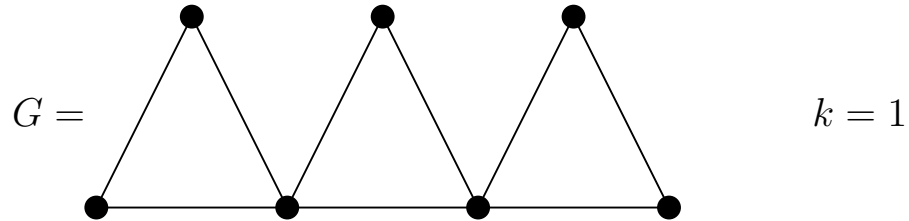
Outline

- Generating k -vertex connected spanning subgraphs
 - Incremental polynomial time algorithm
 - Application: network reliability
- Vertex generation
 - Two representations of polyhedra
 - Proof of NP-hardness
- Monotone generation problem

Generating k -Vertex Connected Spanning Subgraphs

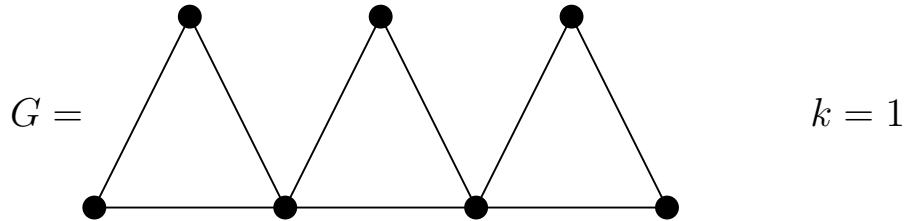
k -Vertex Connected Spanning Subgraphs

Input: k -vertex connected graph G

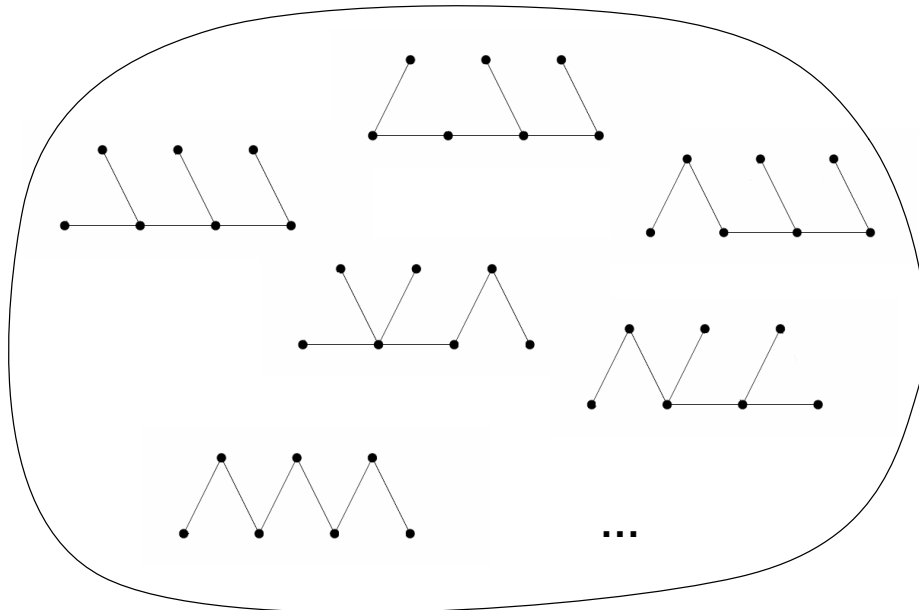


k -Vertex Connected Spanning Subgraphs

Input: k -vertex connected graph G

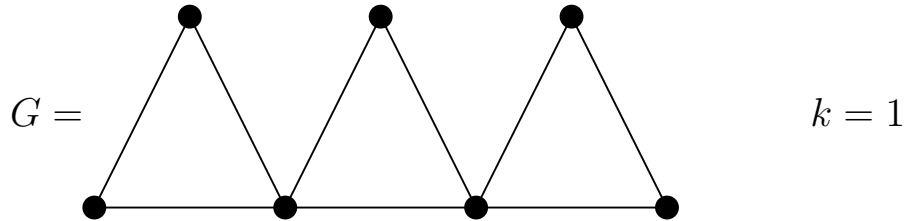


Output: list of all minimal k -vertex connected spanning subgraphs of G

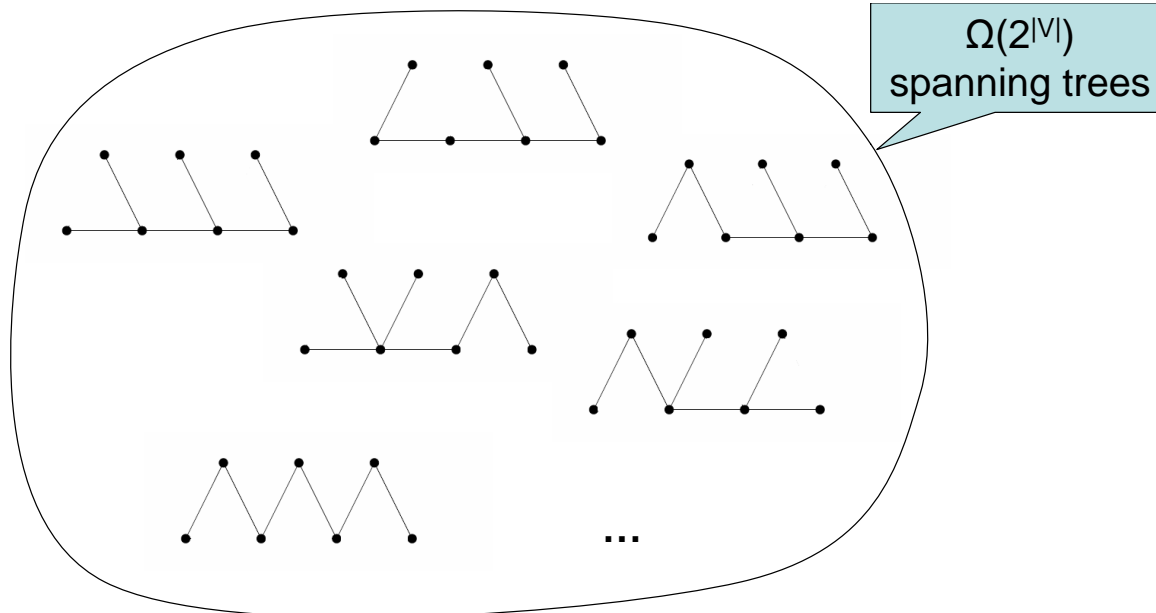


k -Vertex Connected Spanning Subgraphs

Input: k -vertex connected graph G



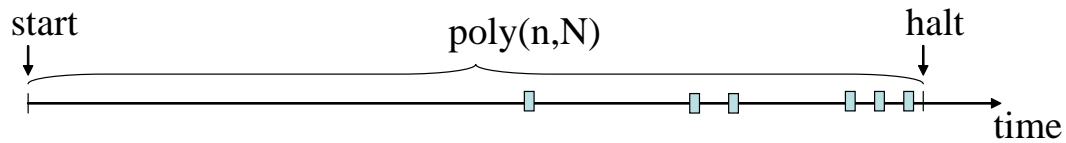
Output: list of all minimal k -vertex connected spanning subgraphs of G



Complexity of Generation Problems

n = input size, N = output size

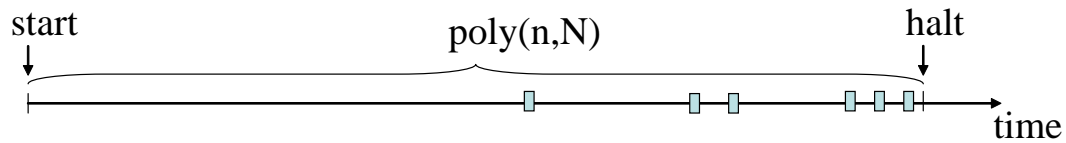
Polynomial Total Time



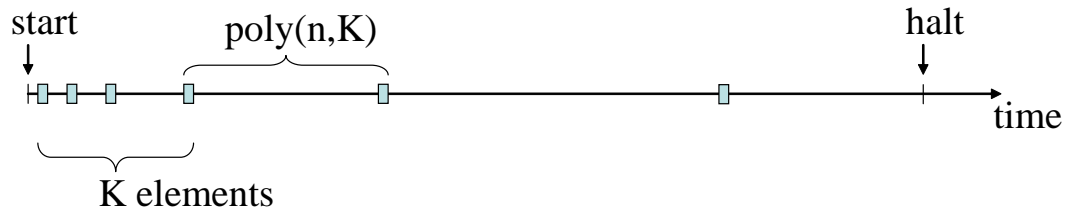
Complexity of Generation Problems

n = input size, N = output size

Polynomial Total Time



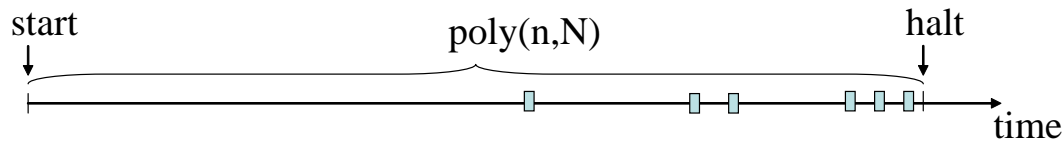
Incremental Polynomial Time



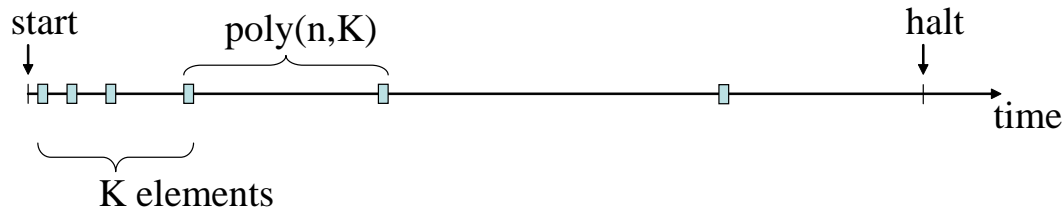
Complexity of Generation Problems

n = input size, N = output size

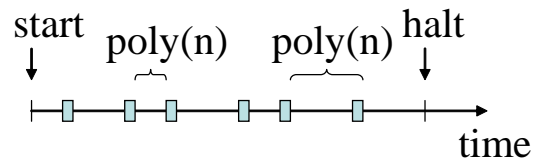
Polynomial Total Time



Incremental Polynomial Time



Polynomial Delay



Previous Results

Polynomial delay algorithms for generating spanning trees ($k = 1$)

$n = \#$ of vertices, $m = \#$ of edges, $N = \#$ of trees

- Read, Tarjan 1975 $O(Nm + n + m)$ time, $O(n + m)$ space
- Gabow, Myers 1978 $O(Nm + n + m)$ time, $O(n + m)$ space (directed and undirected graphs)
- Kapoor, Ramesh 1992 $O(N + n + m)$ time, $O(nm)$ space
- Matsui 1993 $O(Nn + n + m)$ time, $O(n + m)$ space
- Shioura, Tamura 1995 $O(N + n + m)$ time, $O(nm)$ space
- Shioura, Tamura, Uno 1994 $O(N + n + m)$ time, $O(nm)$ space

New Result

We can generate all minimal k -vertex connected spanning subgraphs in incremental polynomial time (for any given k)

Supergraph Approach

Supergraph = directed graph

Vertices = objects to be generated

Supergraph Approach

Supergraph = directed graph

Vertices = objects to be generated

Supergraph strongly connected \implies breadth-first-search
outputs all objects

Supergraph Approach

Supergraph = directed graph

Vertices = objects to be generated

Supergraph strongly connected \implies breadth-first-search
outputs all objects

Generating neighbors in incremental polynomial time
 \implies breadth-first-search in incremental polynomial time

Supergraph Approach

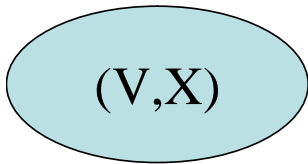
Our tasks:

Define neighborhoods that make the supergraph strongly connected

Show that neighbors can be generated in incremental polynomial time

Generating Neighbors

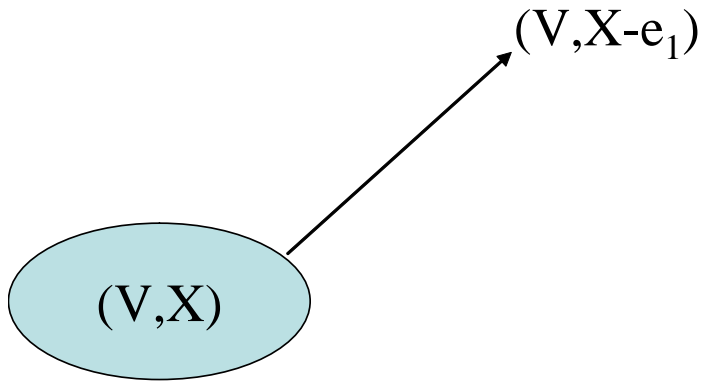
minimal 4-connected



Generating Neighbors

minimal 4-connected

3-connected

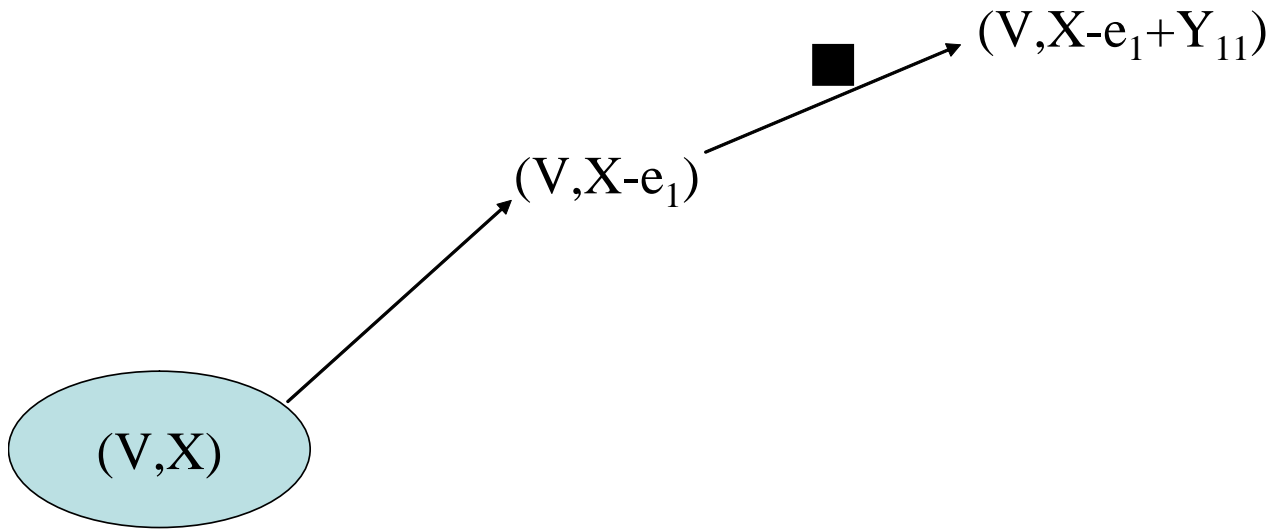


Generating Neighbors

minimal 4-connected

3-connected

4-connected



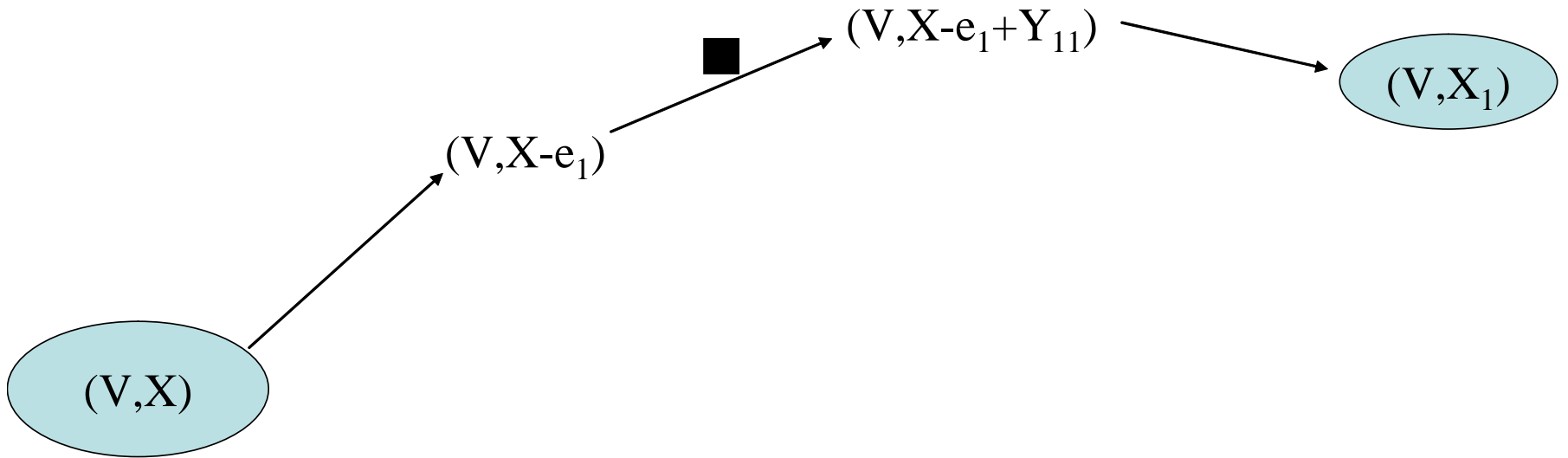
Generating Neighbors

minimal 4-connected

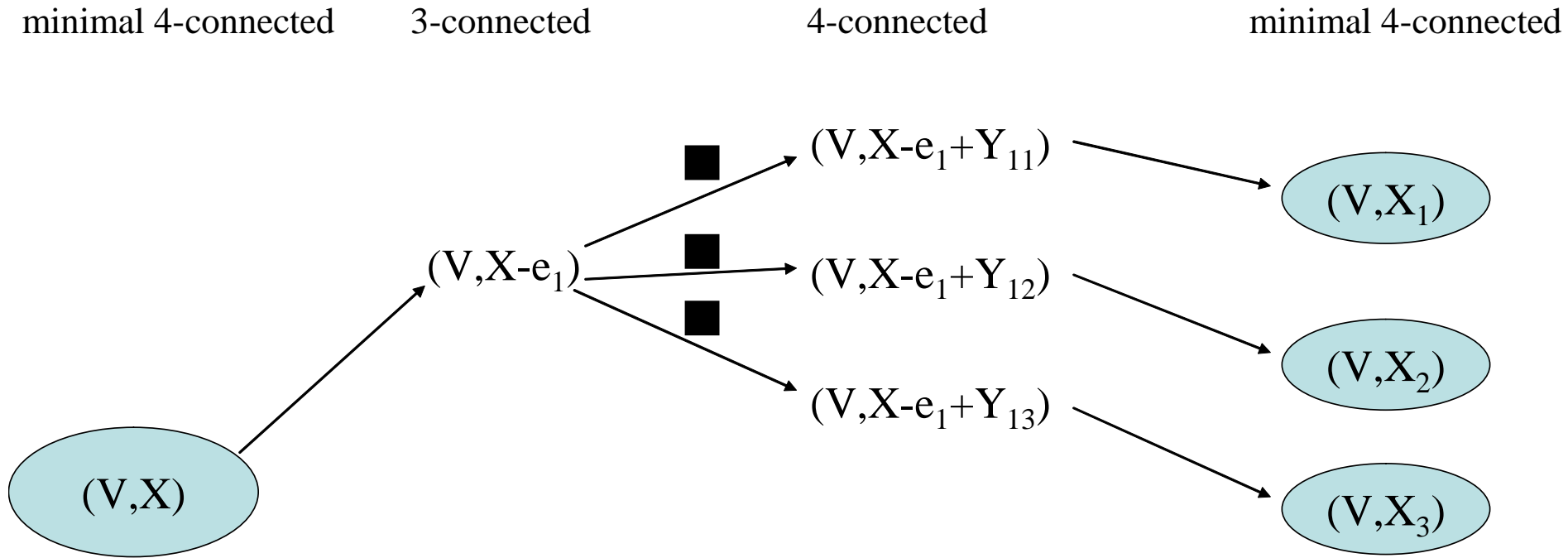
3-connected

4-connected

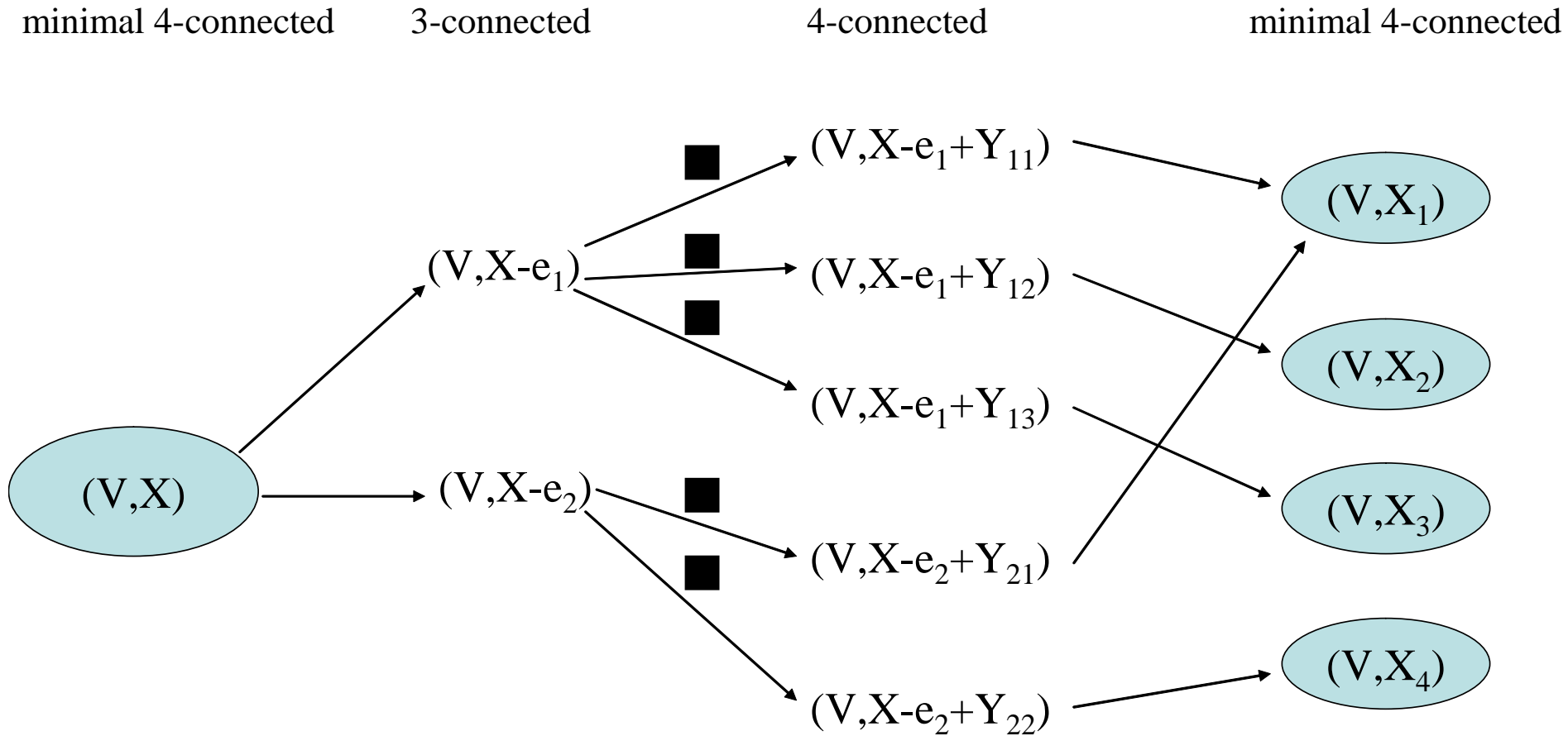
minimal 4-connected



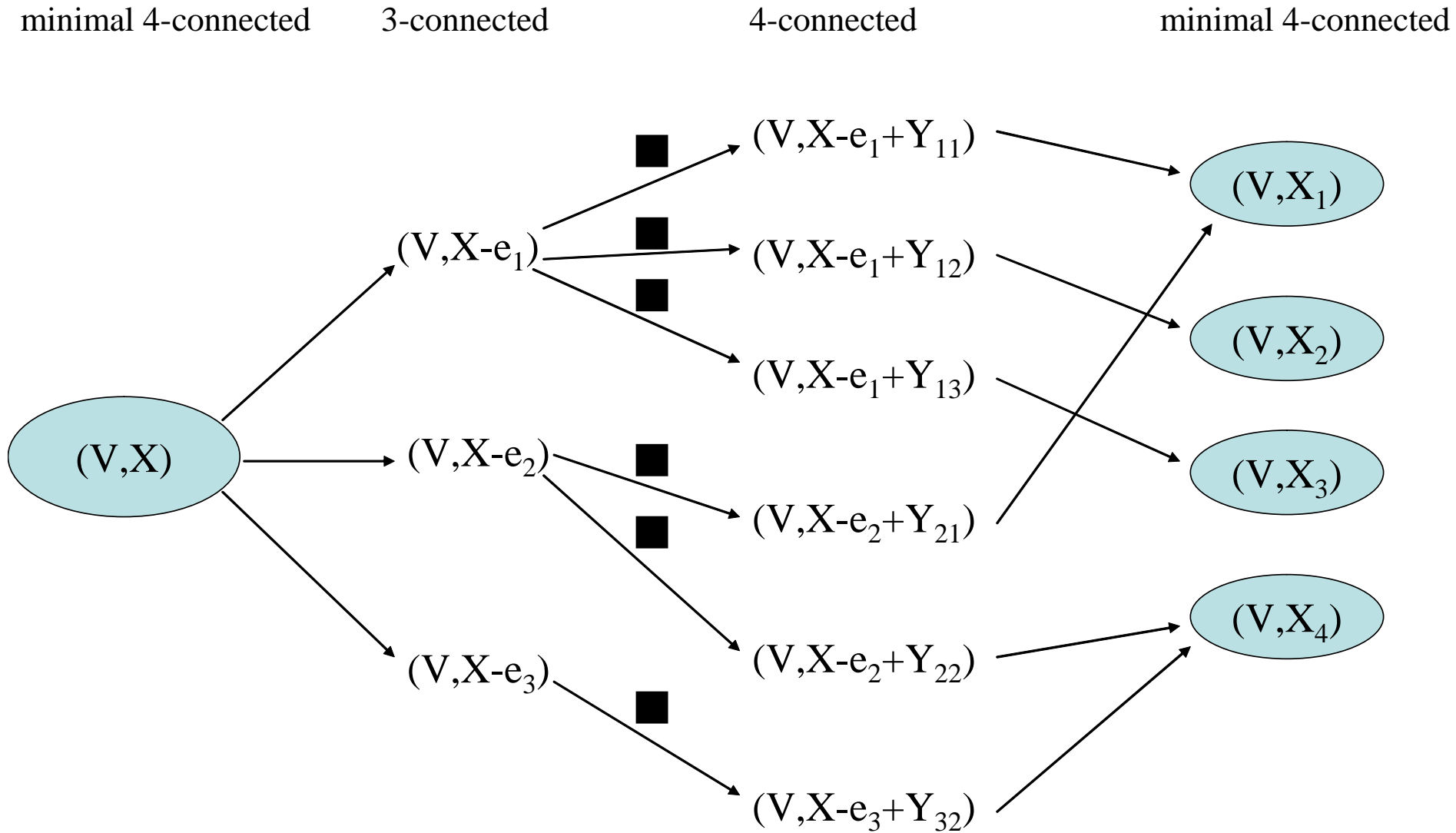
Generating Neighbors



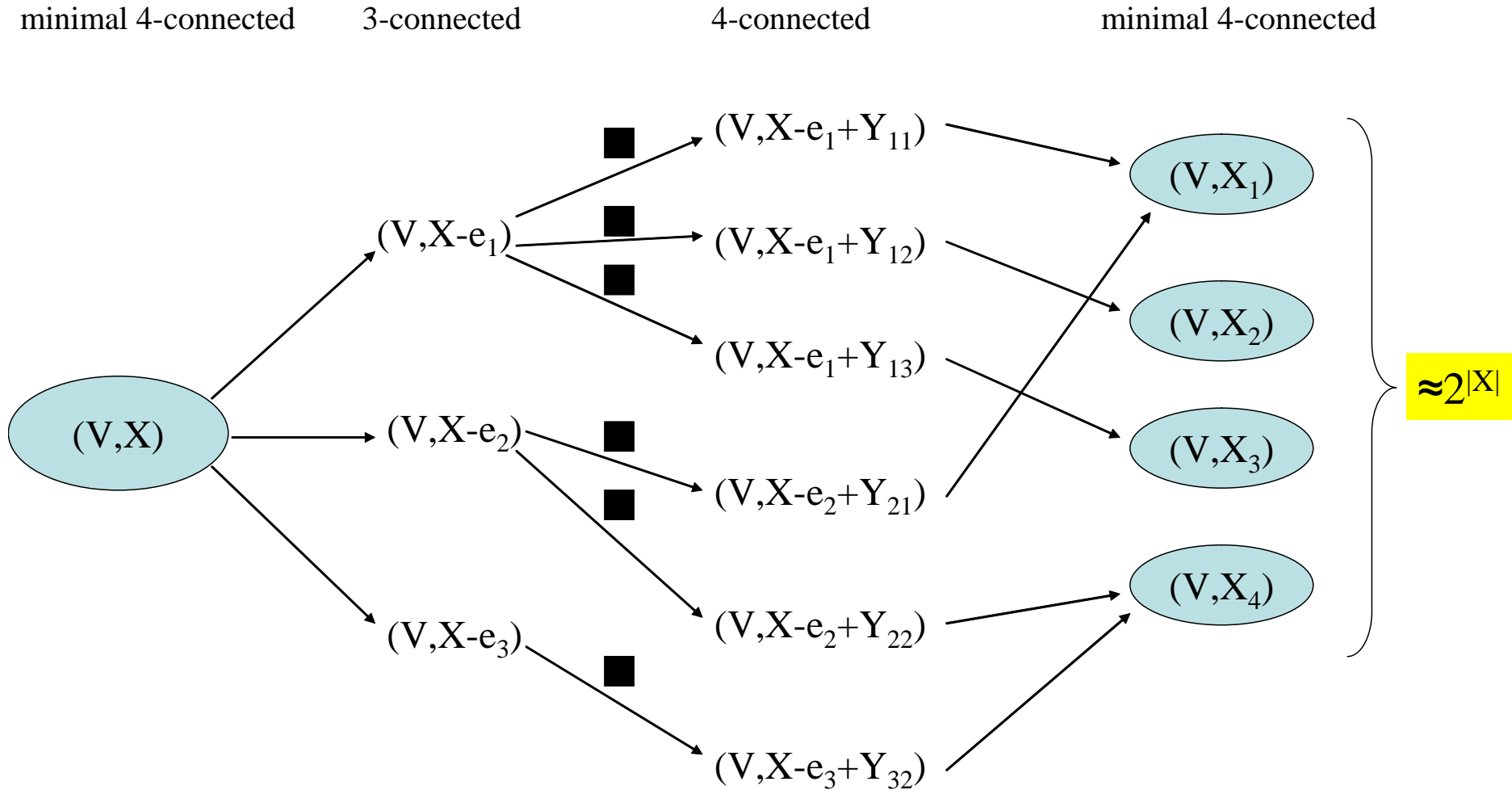
Generating Neighbors



Generating Neighbors



Generating Neighbors



Black Box

$G = (V, E)$ - k -vertex connected graph

(V, X) - minimal k -vertex connected subgraph of G

Input:

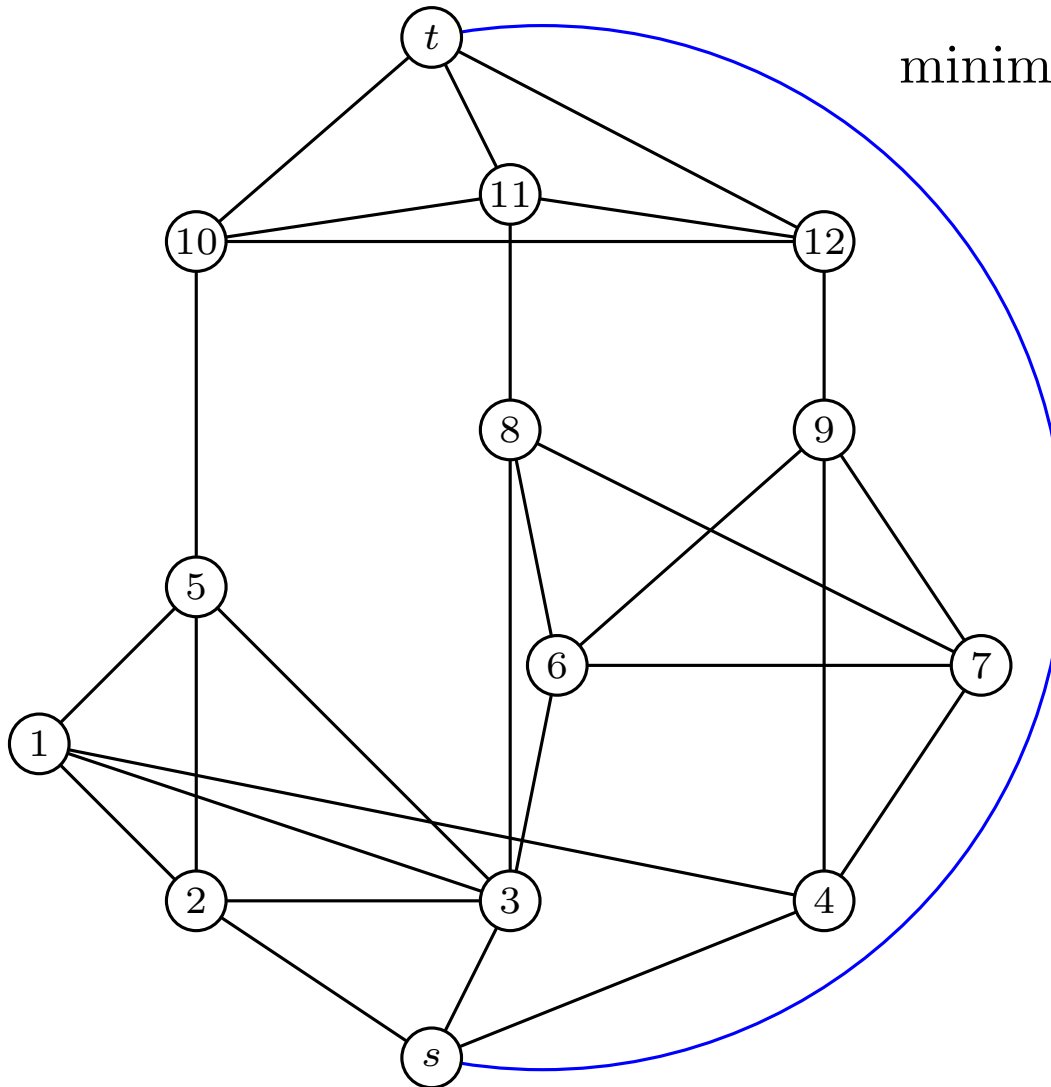
$(k - 1)$ -vertex connected graph $(V, X \setminus e)$

list of edges $E \setminus X$

Output:

list of minimal edge sets $Y \subseteq E \setminus X$ such that
 $(V, (X \setminus e) \cup Y)$ is k -vertex connected

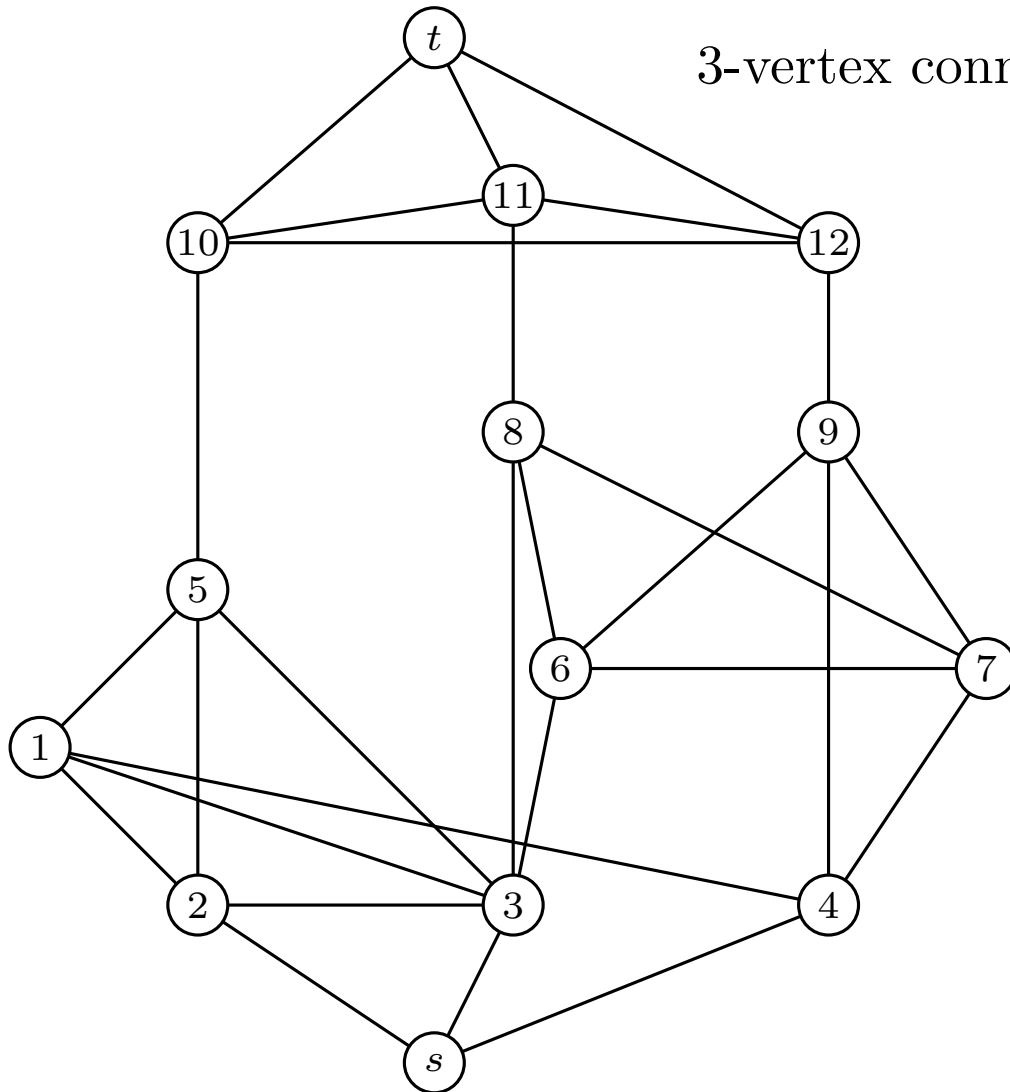
k -Separators



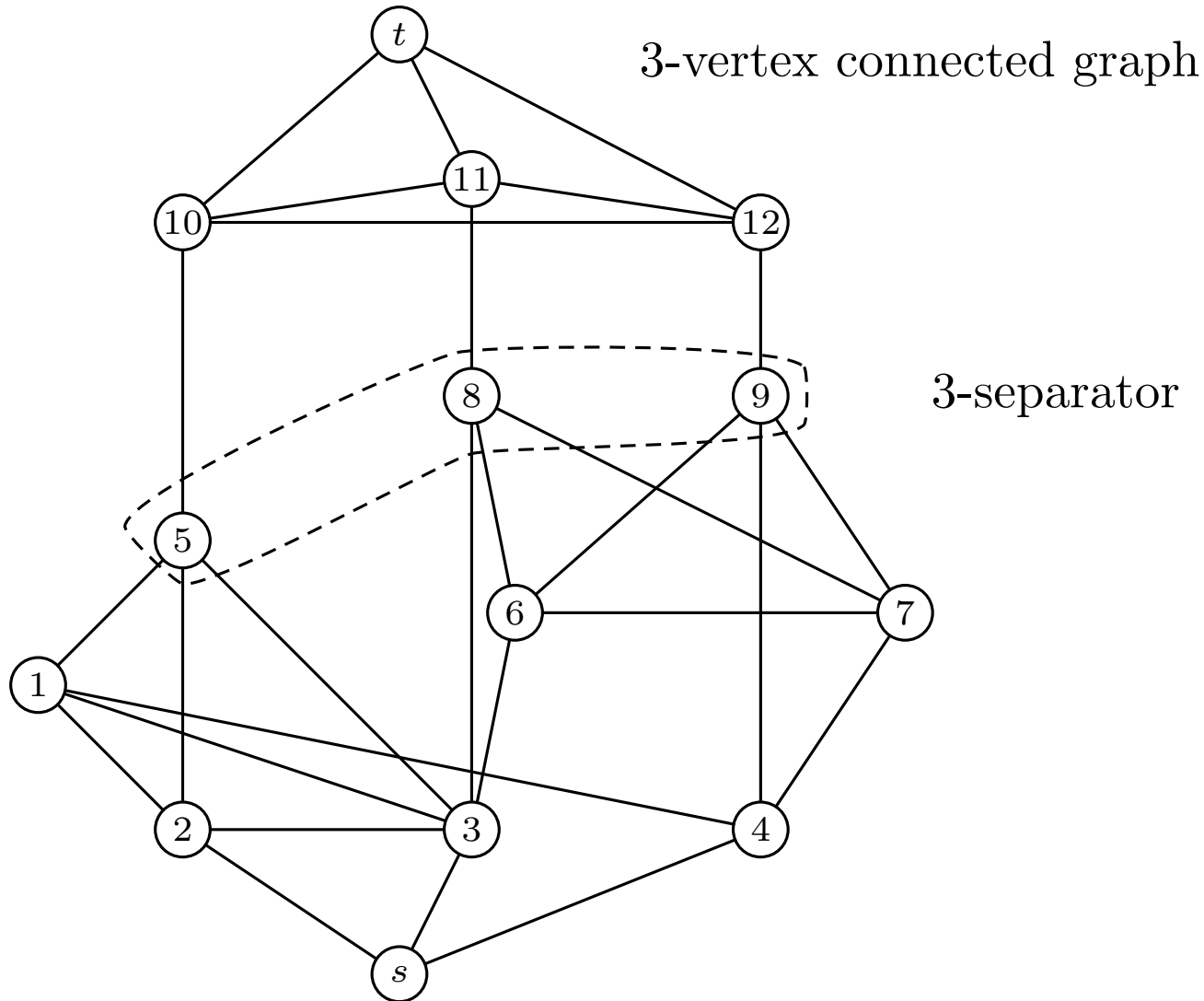
minimal 4-vertex connected graph

k -Separators

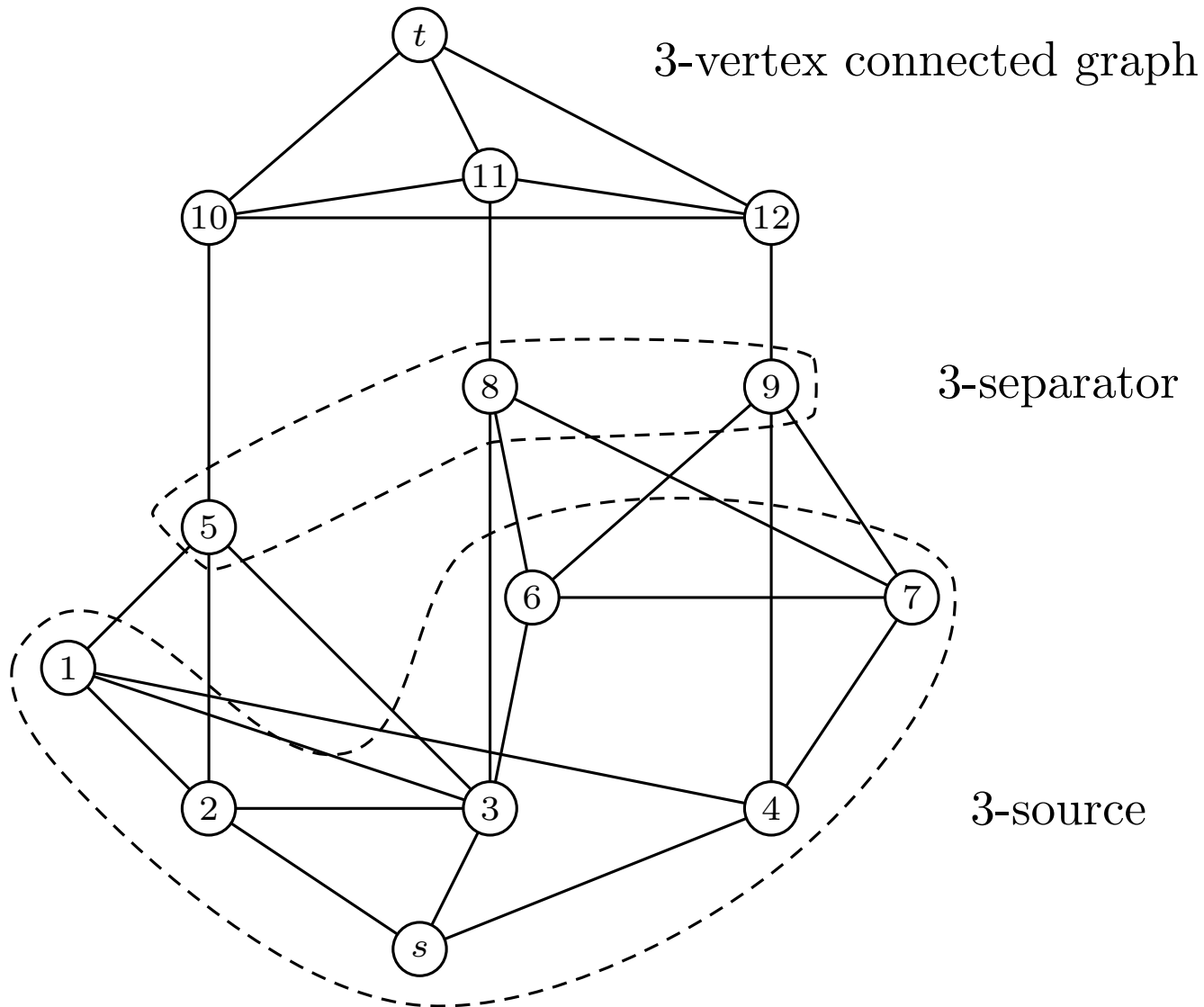
3-vertex connected graph



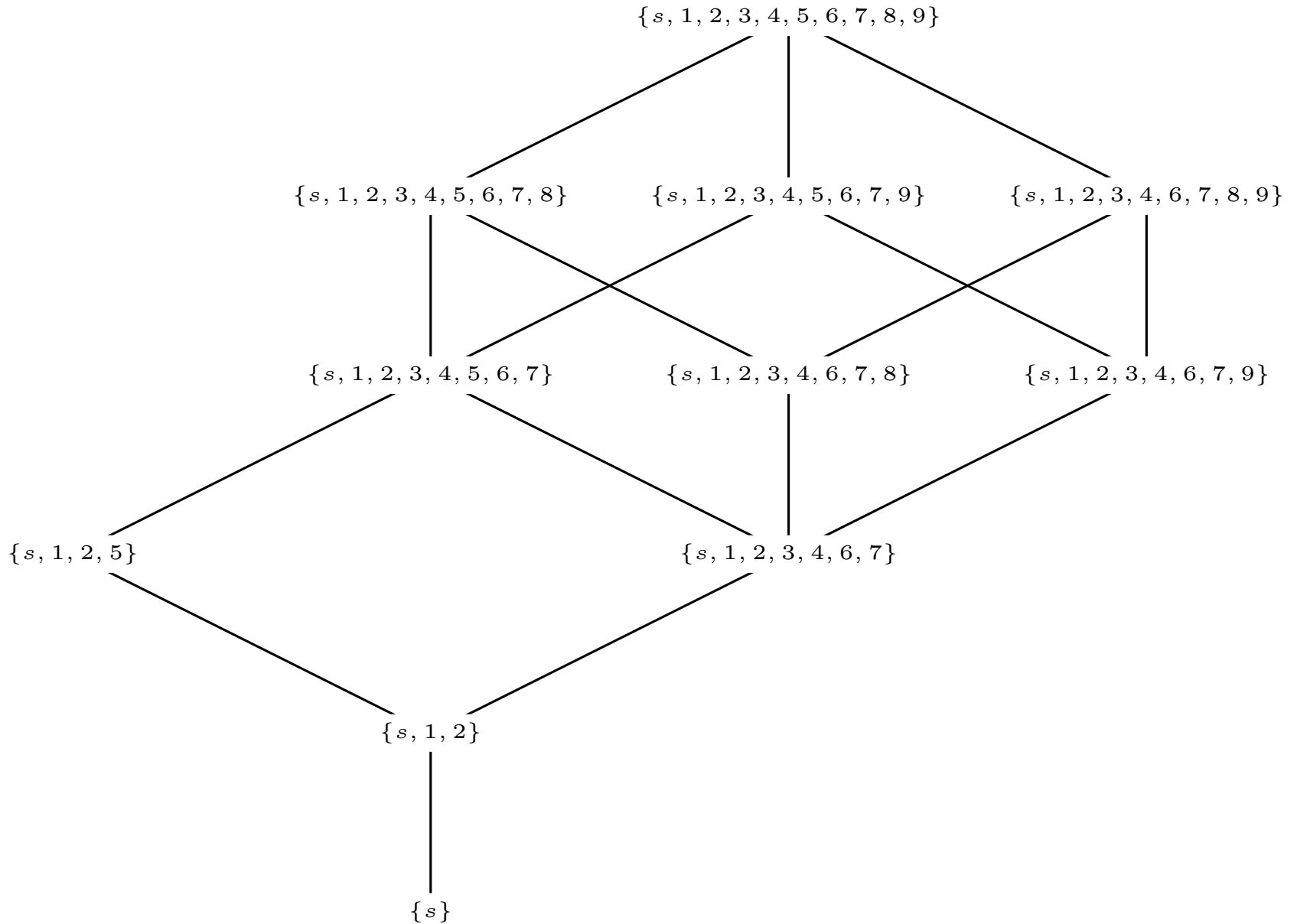
k -Separators



k -Separators



Lattice of k -Sources

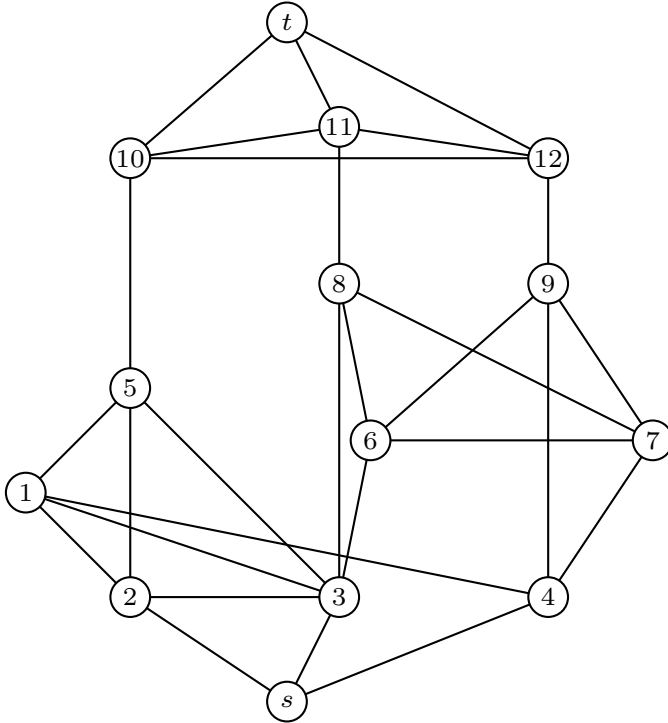


Black Box

Input:

Available edges: $(2,11)$, $(8,t)$, $(s,9)$, $(s,1)$, $(2,12)$

3-vertex connected



3-separators: $\{2,3,4\}$, $\{5,3,4\}$, $\{10,3,4\}$, $\{5,8,9\}$, $\{10,8,9\}$, $\{5,11,9\}$, $\{5,8,12\}$, $\{10,11,9\}$,
 $\{10,8,12\}$, $\{5,11,12\}$, $\{10,11,12\}$

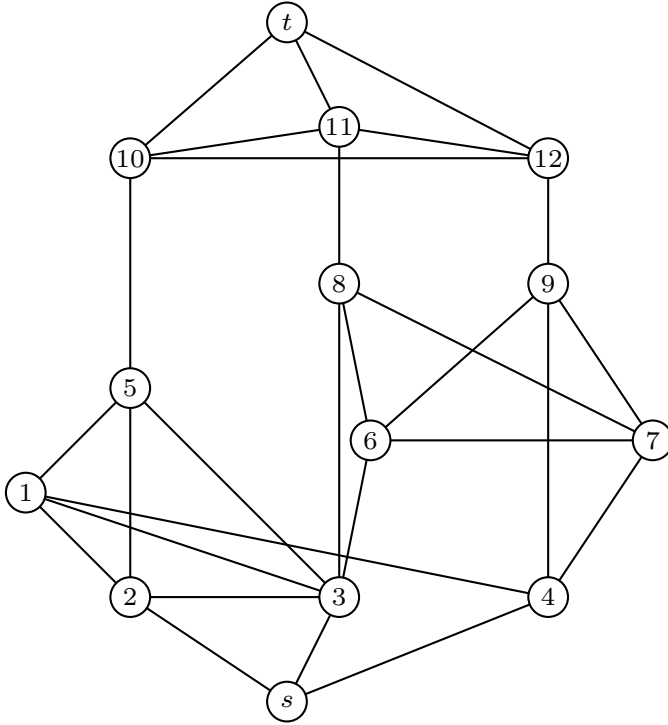
Black Box

Input:

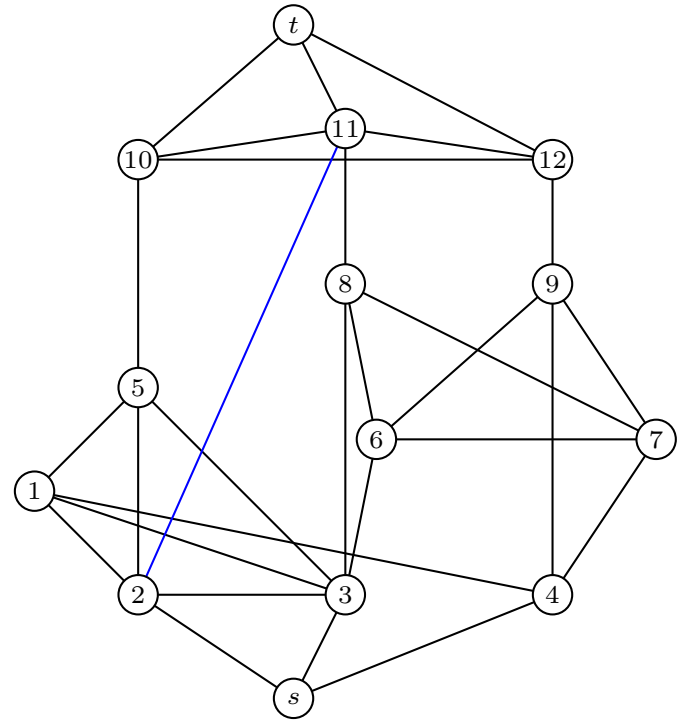
Available edges: (2,11), (8,t), (s,9), (s,1), (2,12)

Add (2,11)

3-vertex connected



3-vertex connected



3-separators: {2,3,4}, {5,11,9}, {10,11,9}, {5,11,12}, {10,11,12}

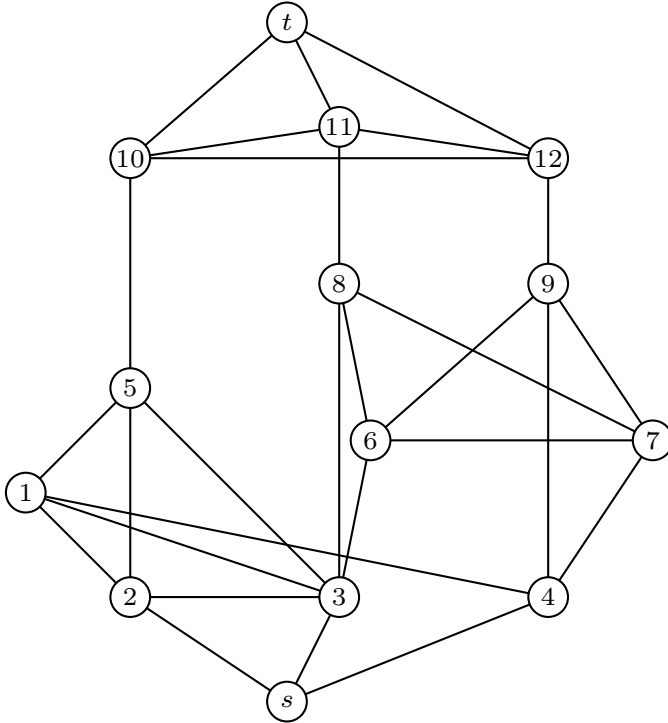
Black Box

Input:

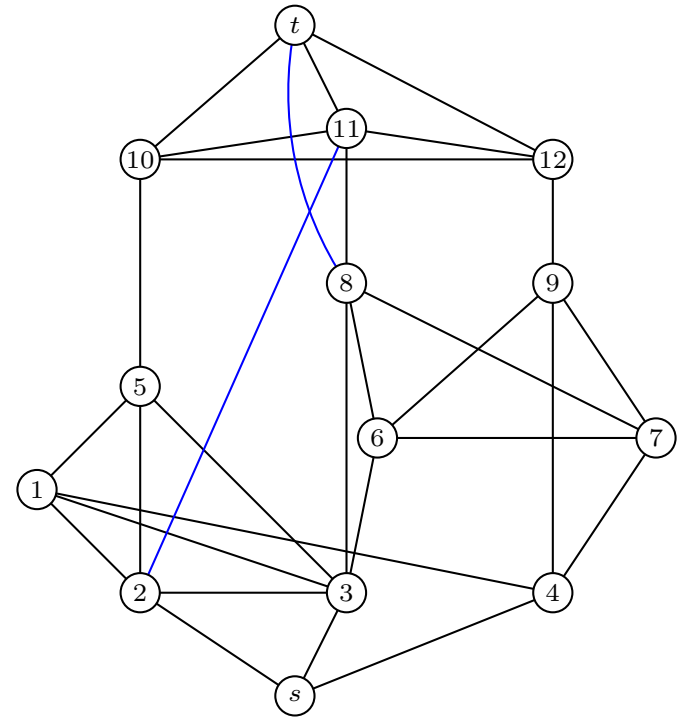
Available edges: $(2,11)$, $(8,t)$, $(s,9)$, $(s,1)$, $(2,12)$

Add $(8,t)$

3-vertex connected



3-vertex connected



3-separators: $\{2,3,4\}$

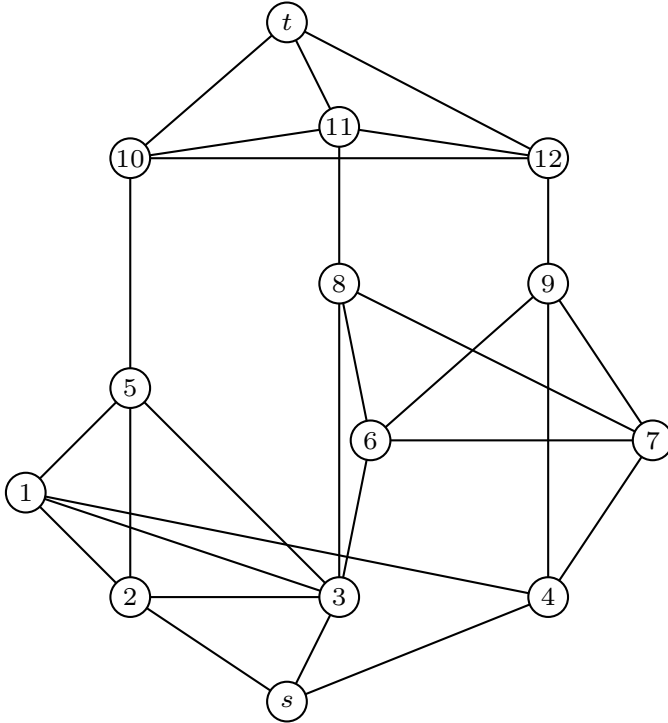
Black Box

Input:

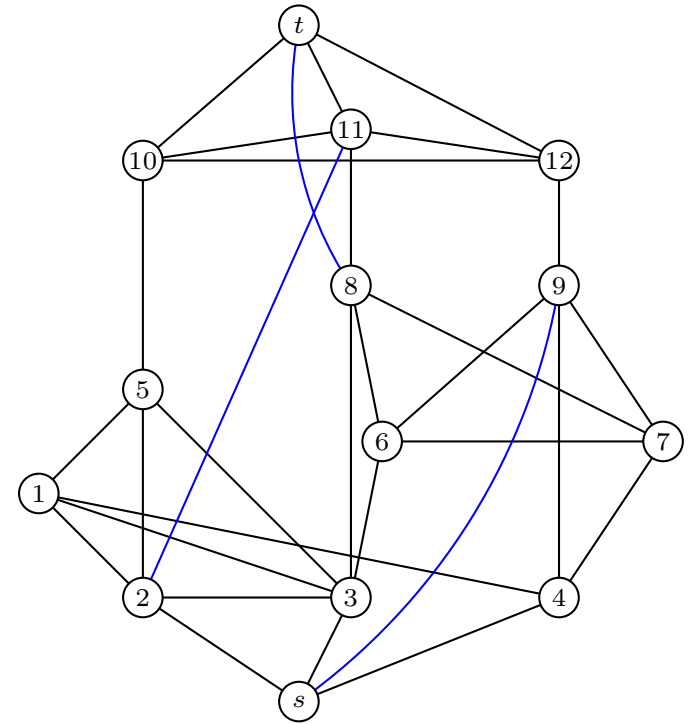
Available edges: $(2,11)$, $(8,t)$, $(s,9)$, $(s,1)$, $(2,12)$

Add $(s,9)$

3-vertex connected



4-vertex connected



3-separators:

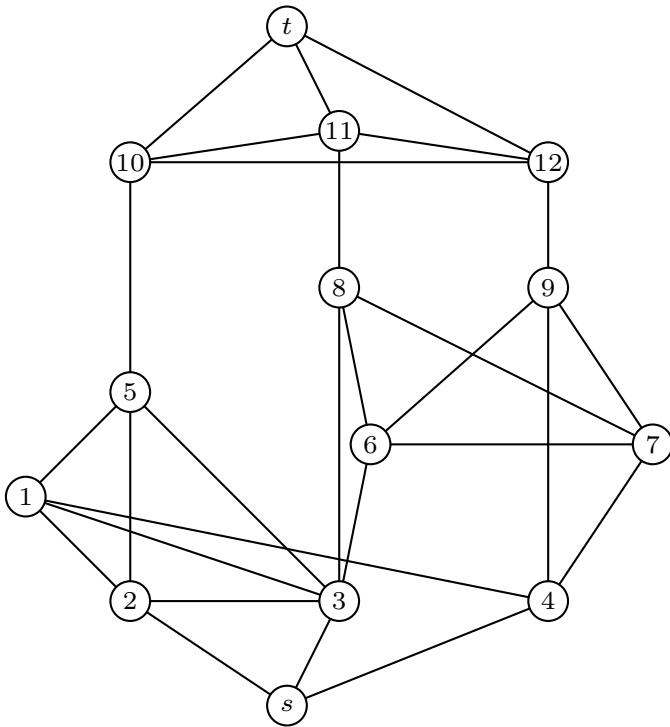
Black Box

Input:

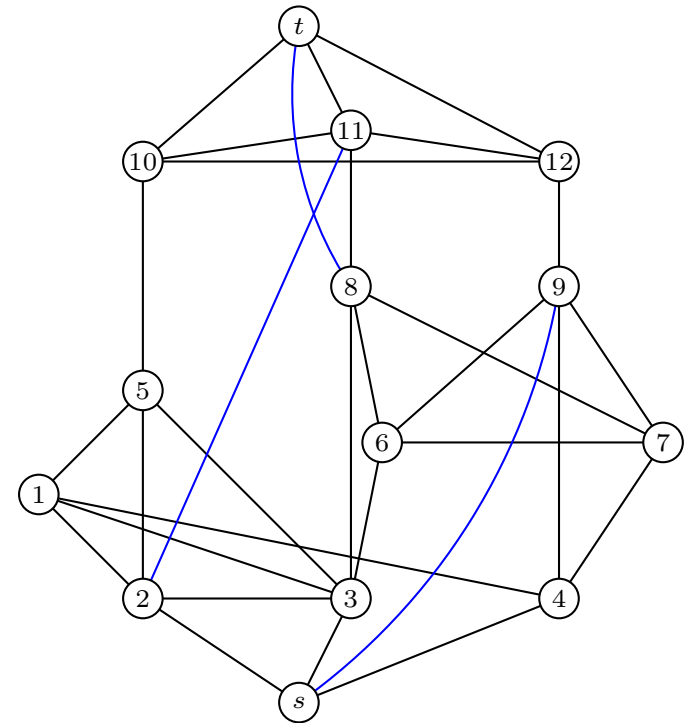
Available edges: $(2,11)$, $(8,t)$, $(s,9)$, $(s,1)$, $(2,12)$

Output: blue edges

3-vertex connected



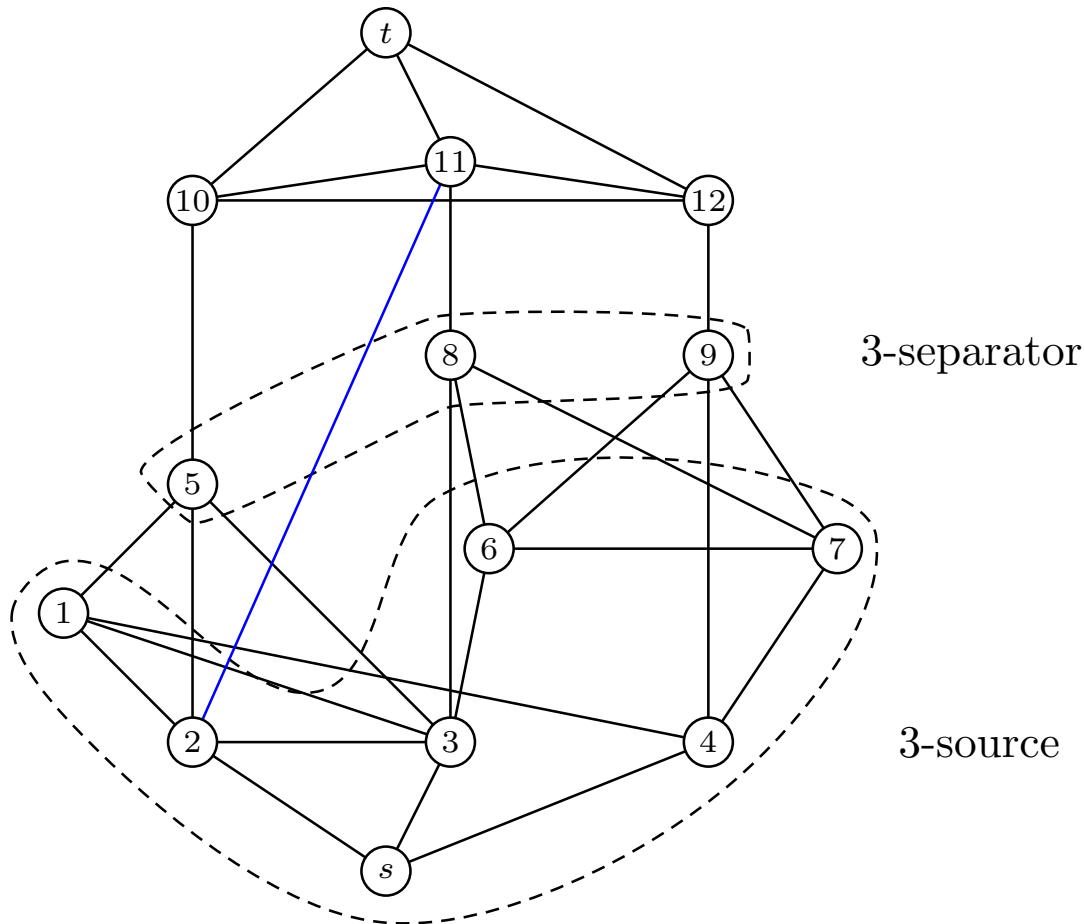
4-vertex connected



Black Box - Key Observation

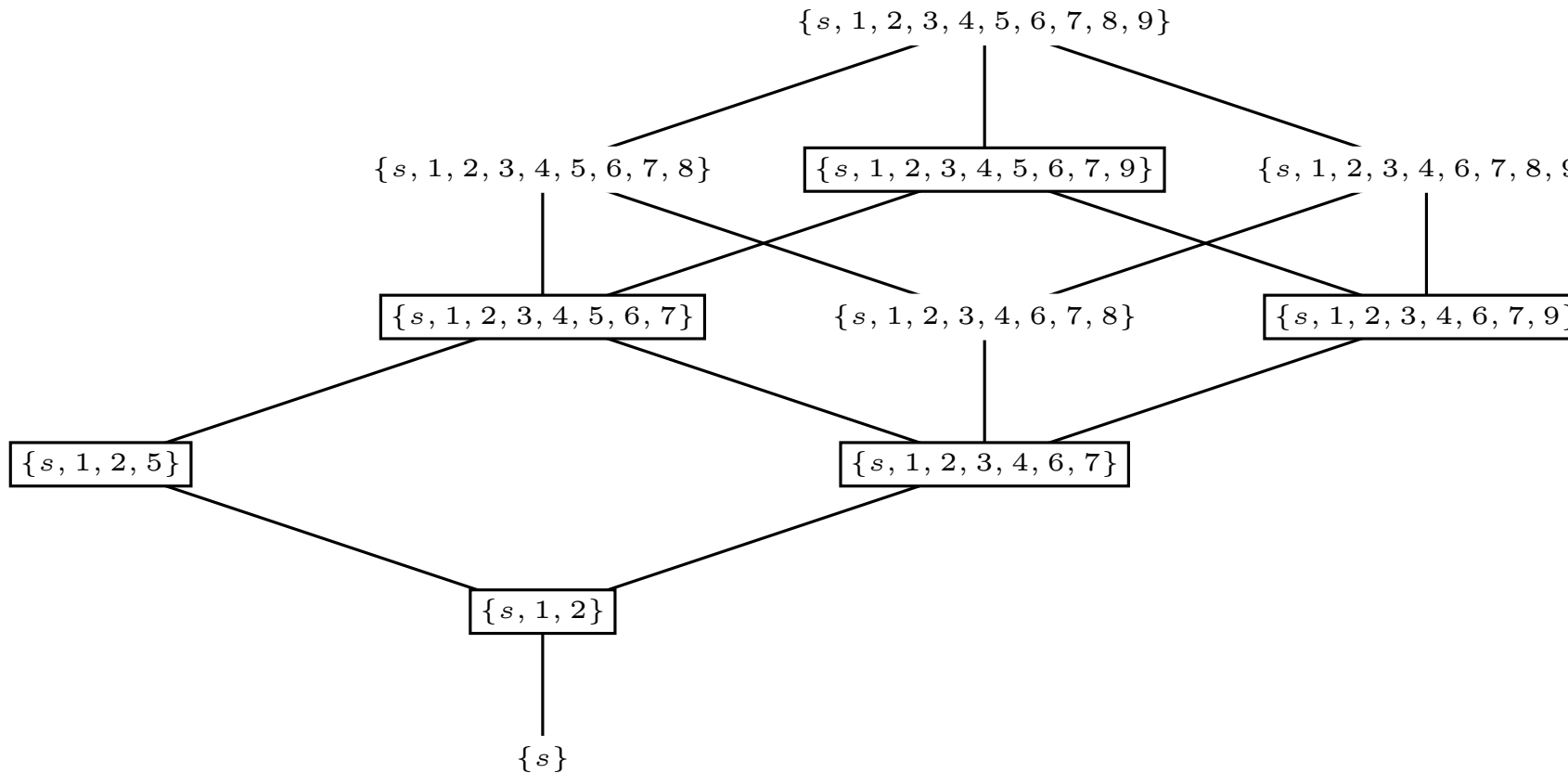
edge \longleftrightarrow sublattice

$\{s, 1, 2, 3, 4, 6, 7\}$ belongs to sublattice corresponding to edge $(2, 11)$



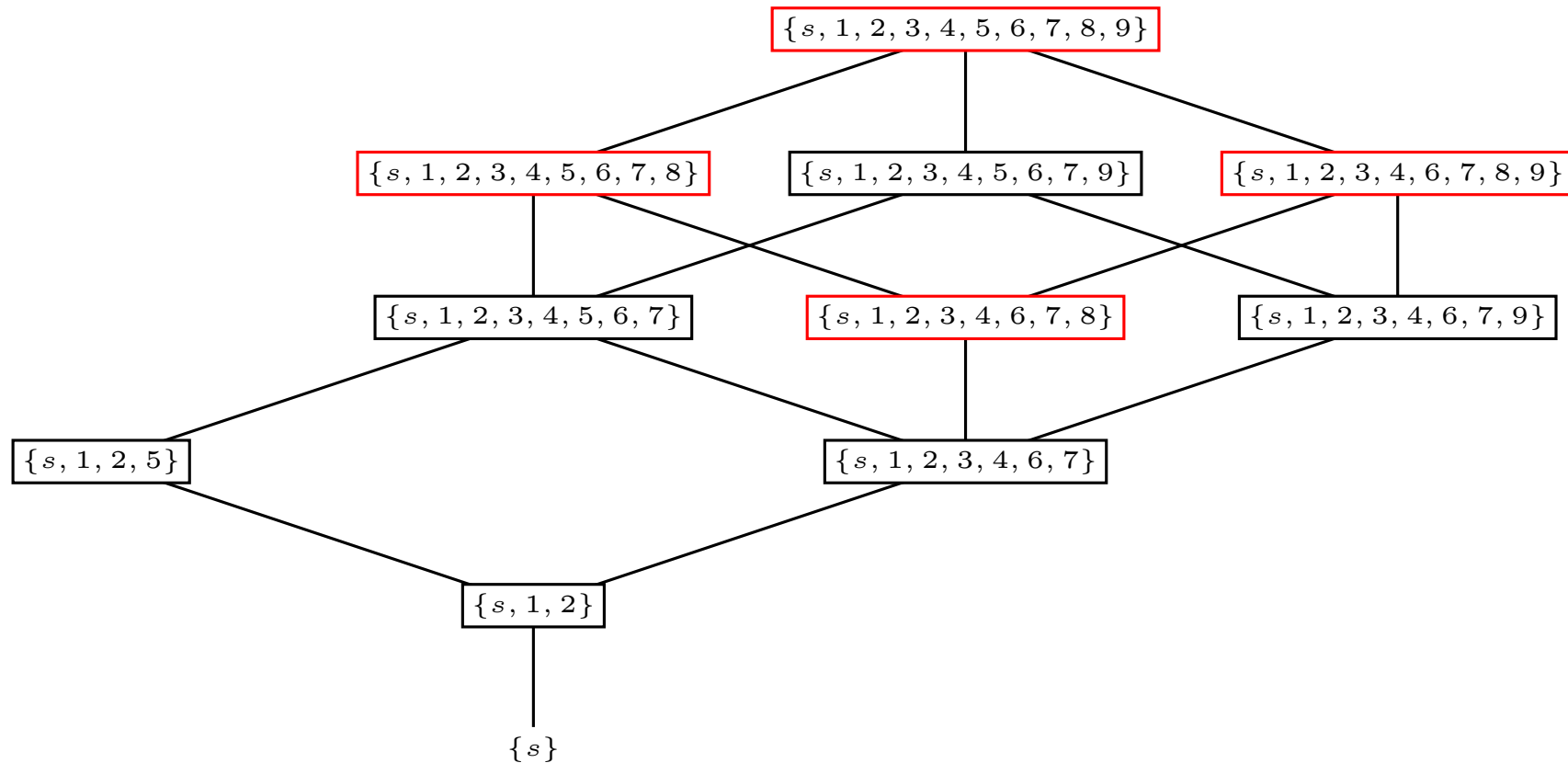
Black Box - Key Observation

edge $(2, 11) \longleftrightarrow$ black sublattice



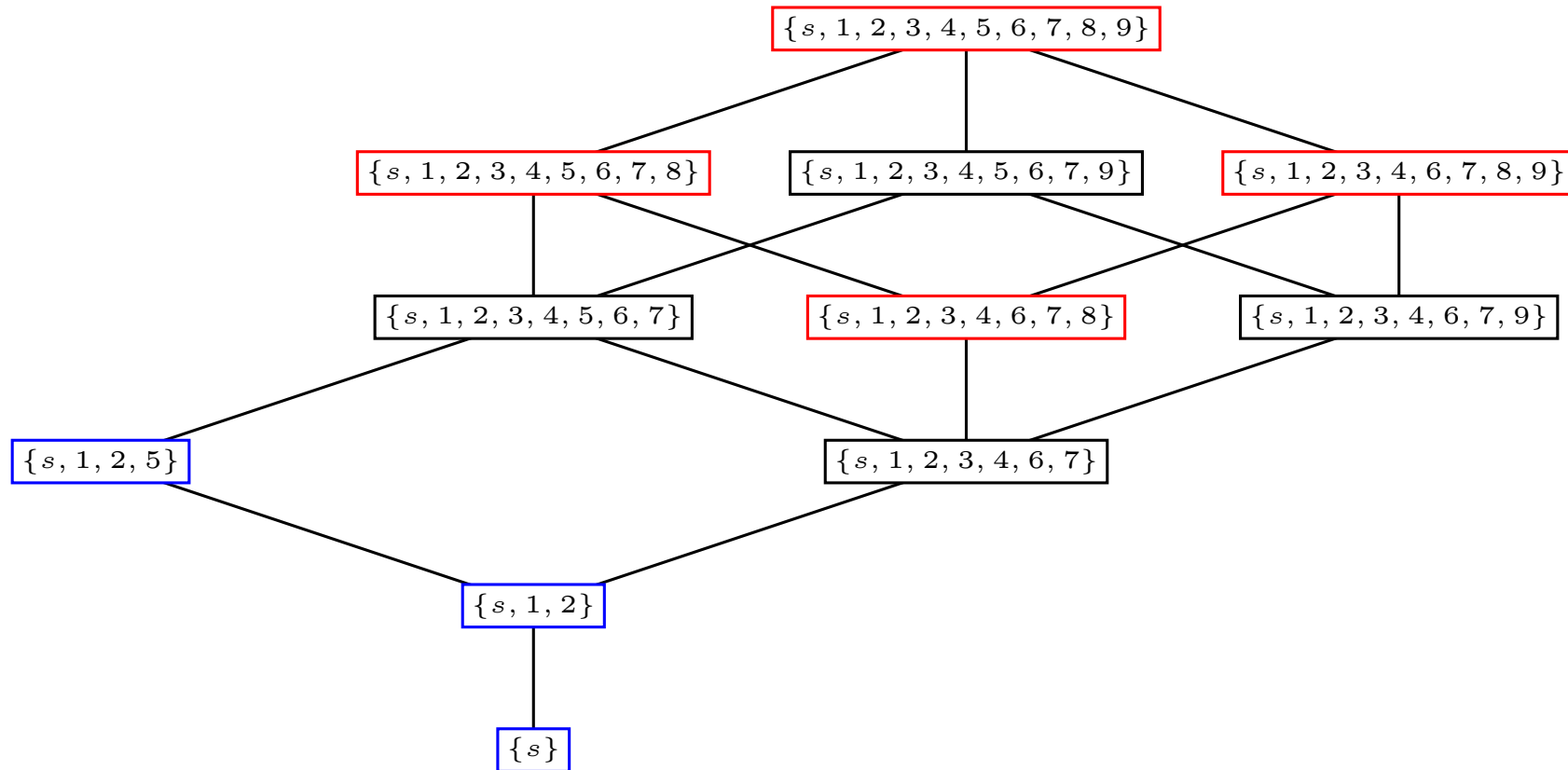
Black Box - Key Observation

edge $(8, t) \longleftrightarrow$ red sublattice

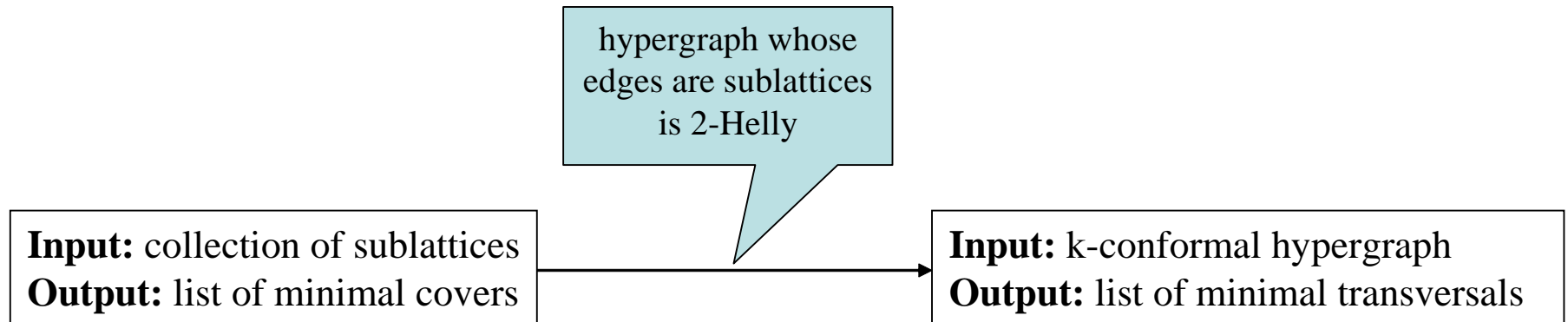


Black Box - Key Observation

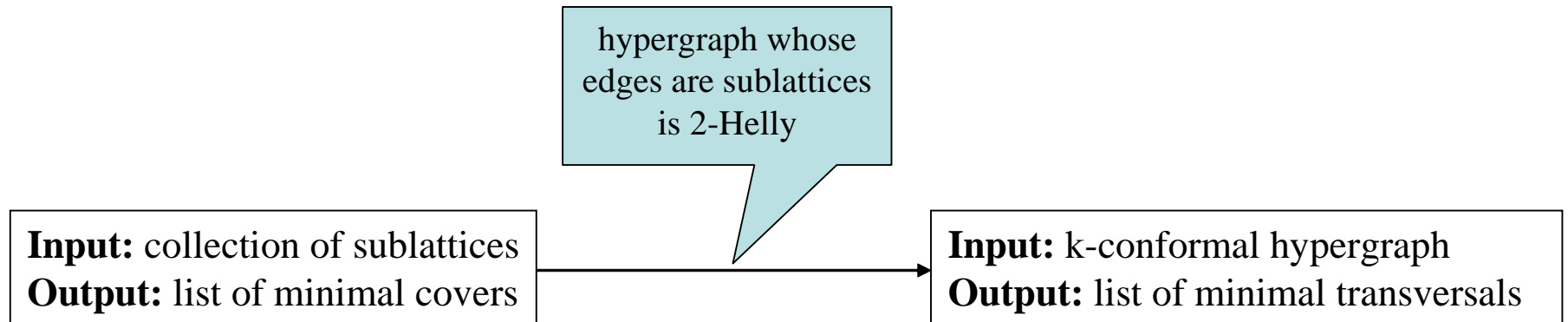
edge $(s, 9) \longleftrightarrow$ blue sublattice



Sublattice Covers



Sublattice Covers



incremental polynomial time algorithm [Boros,Elbassioni, Gurvich, Khachyian 2004]

Complexity Results

$n = \#$ of vertices, $m = \#$ of edges, $N = \#$ of subgraphs

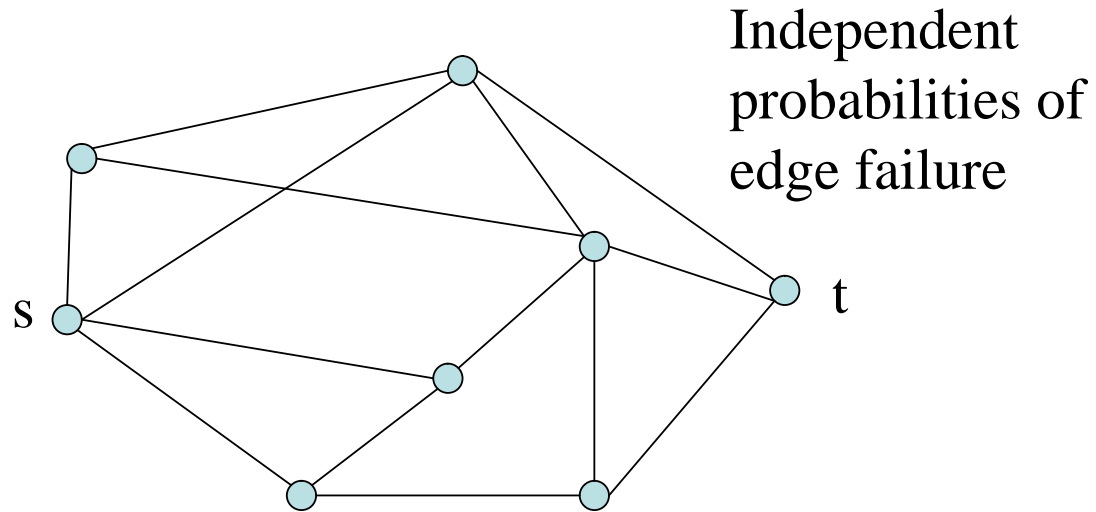
● k -vertex connected: $O(N^3nm^3 + N^2n^4m^5 + Nn^km^2)$

Complexity Results

n = # of vertices, m = # of edges, N = # of subgraphs

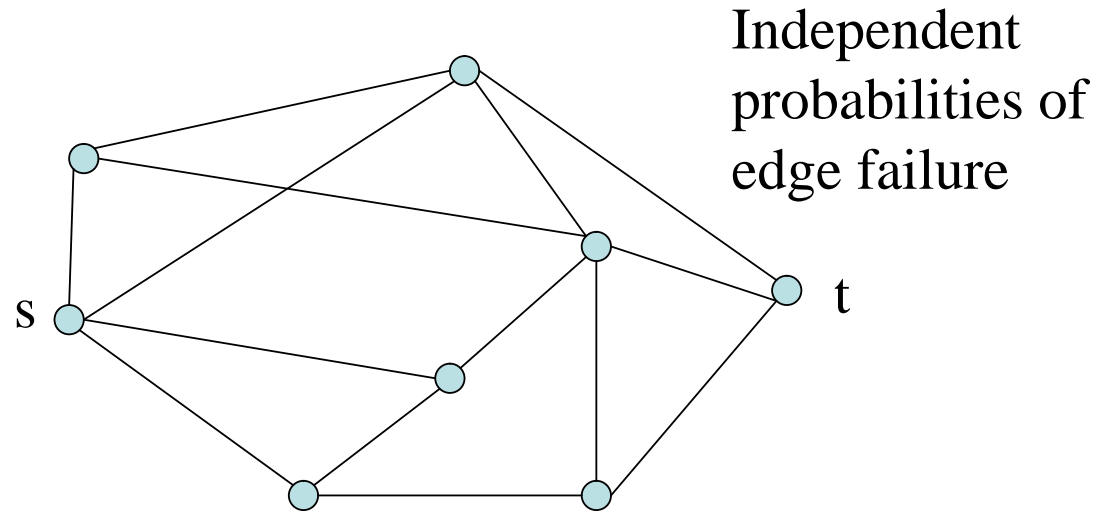
- k -vertex connected: $O(N^3nm^3 + N^2n^4m^5 + Nn^km^2)$
- for $k = 1$ (spanning trees): $O(Nn)$
- for $k = 2, 3$: $O(N^2 \log(N)m^2 + N^2m^3)$
(improvement due to decomposition theory)
- when k is part of the input: OPEN

Two-Terminal Reliability



$Prob(\exists \text{ an operating } s-t \text{ path})$

Two-Terminal Reliability



$Prob(\exists \text{ an operating } s-t \text{ path})$

Hard to compute since counting 1-element, 2-element, ..., $|E|$ -element $s-t$ cuts is hard

How To Compute $Prob(\exists \text{ an operating } s\text{-}t \text{ path})$

1. Generate all N $s\text{-}t$ paths
2. Calculate $Prob(\exists \text{ an operating } s\text{-}t \text{ path})$ using the inclusion-exclusion principle ($A_i = \text{path } i \text{ is operating}$)

$$\begin{aligned} Prob(\exists \text{ an operating } s\text{-}t \text{ path}) &= Prob(A_1 \cup \dots \cup A_N) \\ &= \sum_{k=1}^N (-1)^{k-1} S_k \end{aligned}$$

$$\text{where } S_k = \sum_{l \leq l_1 \leq \dots \leq l_k \leq N} Prob(A_{l_1} \cap \dots \cap A_{l_k})$$

How To Compute $Prob(\exists \text{ an operating } s\text{-}t \text{ path})$

1. Generate all N $s\text{-}t$ paths
2. Calculate $Prob(\exists \text{ an operating } s\text{-}t \text{ path})$ using the inclusion-exclusion principle ($A_i = \text{path } i \text{ is operating}$)

$$\begin{aligned} Prob(\exists \text{ an operating } s\text{-}t \text{ path}) &= Prob(A_1 \cup \dots \cup A_N) \\ &= \sum_{k=1}^N (-1)^{k-1} S_k \end{aligned}$$

$$\text{where } S_k = \sum_{l \leq l_1 \leq \dots \leq l_k \leq N} Prob(A_{l_1} \cap \dots \cap A_{l_k})$$

$\approx 2^{|E|}$

requires 2^N operations

How To Approximate $Prob(\exists \text{ an operating } s\text{-}t \text{ path})$

Generate only $K \ll N$ $s\text{-}t$ paths

Approximate $Prob(\exists \text{ an operating } s\text{-}t \text{ path})$ using only first $L \ll N$ moments S_1, \dots, S_L

Other Generation Problems

Reliability Measures

$Prob(\exists \text{ an operating } s-t \text{ path})$

Generation Problems

$s-t$ paths [Read, Tarjan]

minimal $s-t$ cuts [Tsukiyama et al]

Other Generation Problems

Reliability Measures

$Prob(\exists \text{ an operating } s-t \text{ path})$

$Prob(\text{ all nodes can communicate})$

Generation Problems

$s-t$ paths [Read, Tarjan]

minimal $s-t$ cuts [Tsukiyama et al]

spanning trees [Read, Tarjan]

minimal cuts

k -connected spanning subgraphs

Other Generation Problems

Reliability Measures

$Prob(\exists \text{ an operating } s-t \text{ path})$

$Prob(\text{ all nodes can communicate})$

$Prob(\exists s_1-t_1 \text{ path} \wedge \dots \wedge s_k-t_k \text{ path})$

Generation Problems

$s-t$ paths [Read, Tarjan]

minimal $s-t$ cuts [Tsukiyama et al]

spanning trees [Read, Tarjan]

minimal cuts

k -connected spanning subgraphs

path conjunctions [Boros et al]

cut conjunctions [ISAAC 05]

Other Generation Problems

Reliability Measures

$Prob(\exists \text{ an operating } s-t \text{ path})$

$Prob(\text{ all nodes can communicate})$

$Prob(\exists s_1-t_1 \text{ path} \wedge \dots \wedge s_k-t_k \text{ path})$

Generation Problems

$s-t$ paths [Read, Tarjan]

minimal $s-t$ cuts [Tsukiyama et al]

spanning trees [Read, Tarjan]

minimal cuts

k -connected spanning subgraphs

path conjunctions [Boros et al]

cut conjunctions [ISAAC 05]

Network consists of subnetworks whose edges fail simultaneously

Other Generation Problems

Reliability Measures

$Prob(\exists \text{ an operating } s-t \text{ path})$

$Prob(\text{ all nodes can communicate})$

$Prob(\exists s_1-t_1 \text{ path} \wedge \dots \wedge s_k-t_k \text{ path})$

Generation Problems

$s-t$ paths [Read, Tarjan]

minimal $s-t$ cuts [Tsukiyama et al]

spanning trees [Read, Tarjan]

minimal cuts

k -connected spanning subgraphs

path conjunctions [Boros et al]

cut conjunctions [ISAAC 05]

Network consists of subnetworks whose edges fail simultaneously

$Prob(\text{ all nodes can communicate})$

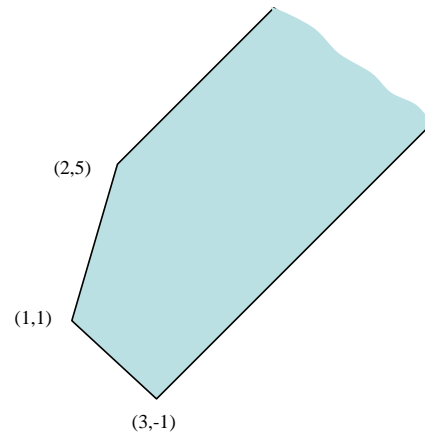
connected spanning sets

in matroids [ESA 06]

Vertex Generation

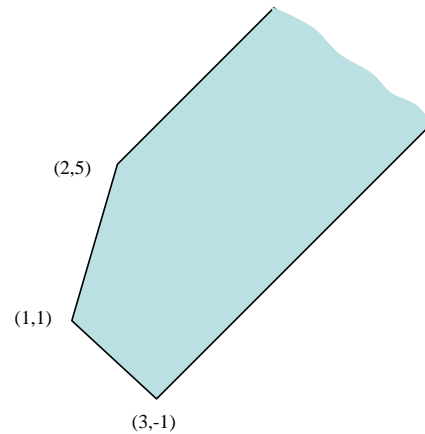
Representation Theorem

Minkowski, Weyl: Two representations of a polyhedron P



Representation Theorem

Minkowski, Weyl: Two representations of a polyhedron P



halfspace representation

$$P = \{x : Ax \leq b\}$$

$$-x_1 + x_2 \leq 3$$

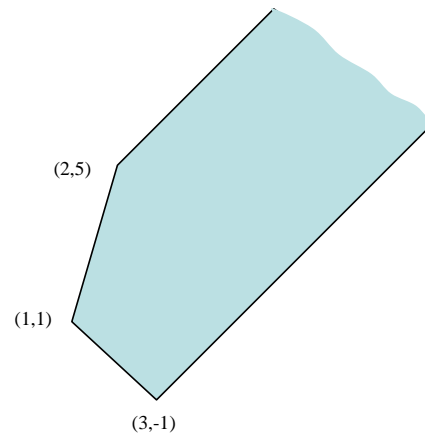
$$-4x_1 + x_2 \leq -3$$

$$-x_1 - x_2 \leq -2$$

$$x_1 - x_2 \leq 4$$

Representation Theorem

Minkowski, Weyl: Two representations of a polyhedron P



halfspace representation

$$P = \{x : Ax \leq b\}$$

$$-x_1 + x_2 \leq 3$$

$$-4x_1 + x_2 \leq -3$$

$$-x_1 - x_2 \leq -2$$

$$x_1 - x_2 \leq 4$$

vertex representation

$$P = \text{conv}\{v_1, \dots, v_k\} + \text{cone}\{d_1, \dots, d_l\}$$

$$(x_1, x_2) = \lambda_1(2, 5) + \lambda_2(1, 1) + \lambda_3(3, -1) + \mu(1, 1)$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

$$\lambda_1, \lambda_2, \lambda_3, \mu \geq 0$$

Two problems

Vertex Generation

Input: halfspace representation

Output: list of vertices and extreme directions

Facet Generation

Input: vertex representation

Output: list of halfspaces

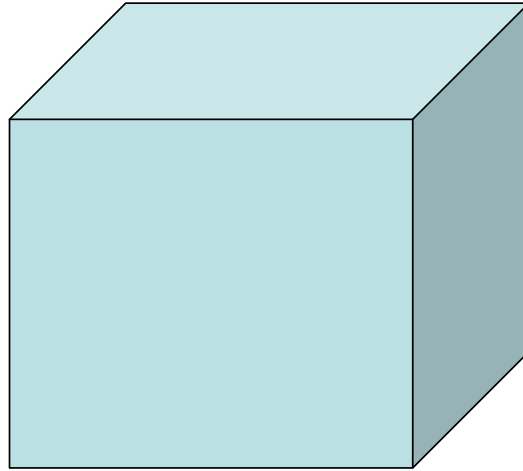
Vertex Generation and Facet Generation are equivalent

$$P^* = \{y : x^T y \leq 1 \text{ for all } x \in P\}$$

vertices of $P \longleftrightarrow$ facets of P^*

facets of $P \longleftrightarrow$ vertices of P^*

Cube



$2n$ inequalities

$$0 \leq x_1 \leq 1$$

$$0 \leq x_2 \leq 1$$

\vdots

$$0 \leq x_n \leq 1$$

2^n vertices

$$\{0, \dots, 0, 0\}, \{0, \dots, 0, 1\},$$

$$\{0, \dots, 1, 0\}, \{0, \dots, 1, 1\},$$

$$\dots, \{1, \dots, 1, 1\}$$

Irredundant Infeasible Subsystems

IIS Generation

Input: infeasible system $Ax \leq b$

Output: list of minimal infeasible subsystems

IIS Generation and Vertex Generation are equivalent
vertices of $\{y : y^T A = 0, y \geq 0, b^T y = 1\} \longleftrightarrow$ minimal infeasible
subsystems of $Ax \geq b$ [Gleeson, Ryan]

Previous Results

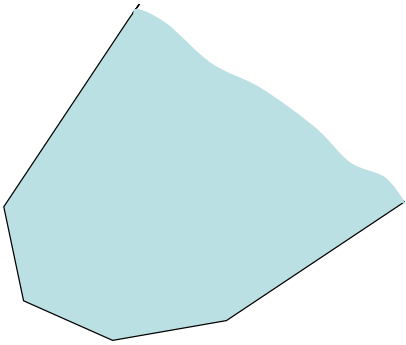
Efficient algorithms for special classes:

- simple polyhedra (every vertex incident with n facets) [Avis, Fukuda 1992]
- simplicial polyhedra (the dual of simple polyhedra) [Bremner, Fukuda, Marzetta 1998]
- network polytopes [Provan 1994]
- polytopes with zero-one vertices [Bussieck, Lubbecke 1998]
- polyhedra in which every facet defining inequality involves at most two nonzero coefficients [Abdullahi 2003]

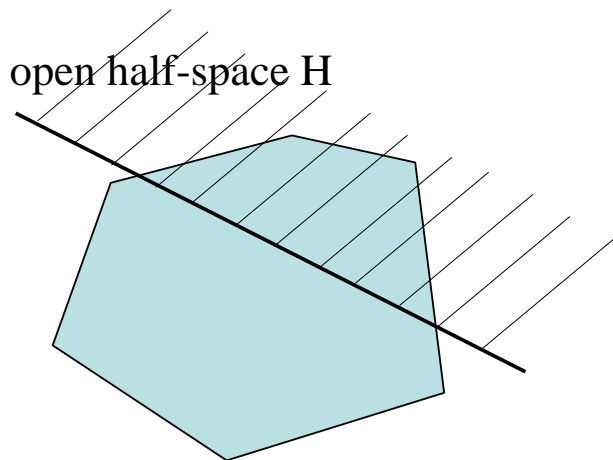
All known algorithms perform poorly in general case
[Fukuda, Bremner, Seidel 1995]

New Results

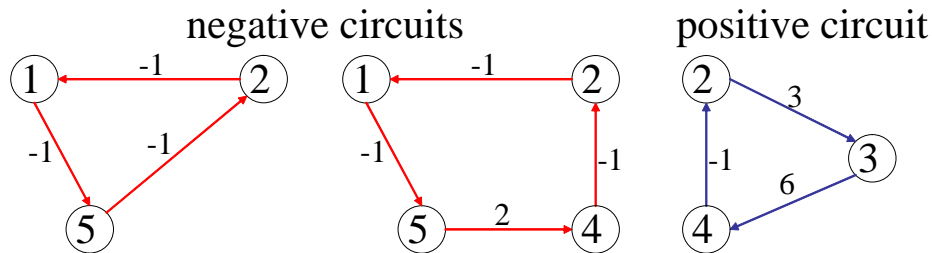
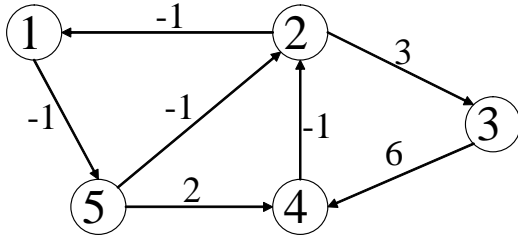
Generating vertices of an unbounded polyhedron is NP-hard [SODA 06, DCG].



Generating vertices of a bounded polyhedron which belong to an open half-space H is NP-hard [SODA 06, DCG].



Proof - Negative Circuits in Digraphs



Negative Circuits Generation Problem

Input: directed graph with edge weights

Output: list of all negative circuits

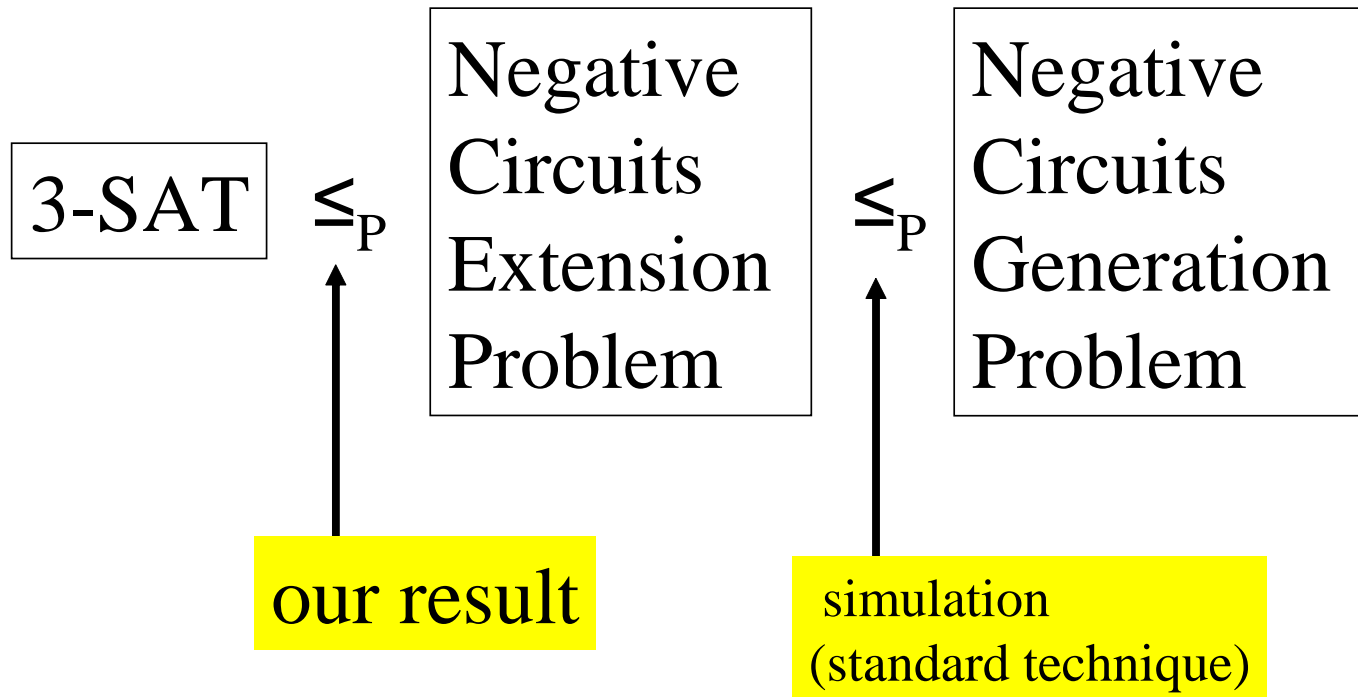
Negative Circuits Extension Problem

Input: directed graph with edge weights, \mathcal{X} - a collection of negative circuits

Output: **Yes**, if there is negative circuit which does not belong to \mathcal{X} **No**, if \mathcal{X} contains all negative circuits

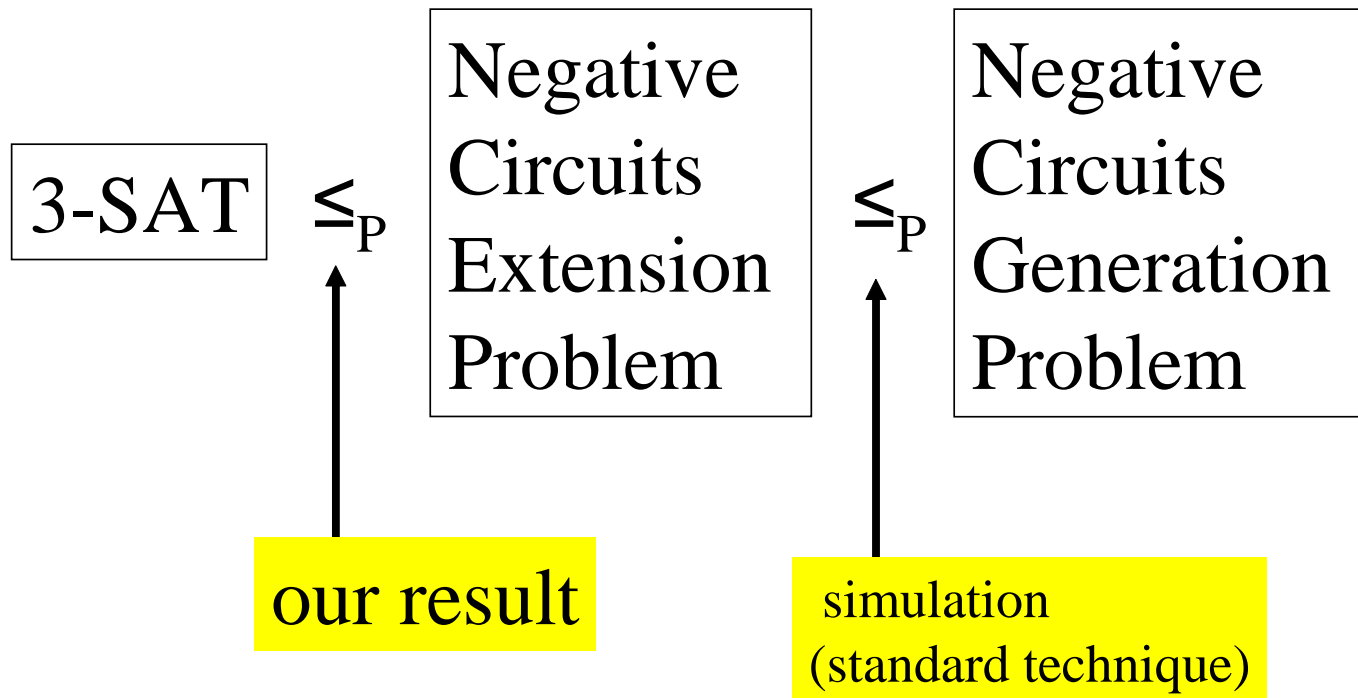
Proof

Strategy



Proof

Strategy



Thus generating negative circuits is NP-hard

Simulation

n = input size, N = total number of negative circuits

\mathcal{X} a collection of negative circuits

Suppose: algorithm **A** generates all N negative circuits in time $\phi(n, N)$, where $\phi(n, N)$ is *poly*(n, N)

Run **A** and interrupt it after $\phi(n, |\mathcal{X}|) + 1$ time (if it did not stop before)

Simulation

n = input size, N = total number of negative circuits
 \mathcal{X} a collection of negative circuits

Suppose: algorithm **A** generates all N negative circuits
in time $\phi(n, N)$, where $\phi(n, N)$ is *poly*(n, N)

Run **A** and interrupt it after $\phi(n, |\mathcal{X}|) + 1$ time (if it did not
stop before)

A outputs a negative circuit not in $\mathcal{X} \Rightarrow$ **Yes**

Simulation

n = input size, N = total number of negative circuits

\mathcal{X} a collection of negative circuits

Suppose: algorithm **A** generates all N negative circuits in time $\phi(n, N)$, where $\phi(n, N)$ is *poly*(n, N)

Run **A** and interrupt it after $\phi(n, |\mathcal{X}|) + 1$ time (if it did not stop before)

A outputs a negative circuit not in $\mathcal{X} \Rightarrow$ **Yes**

A does not output a negative circuit not in \mathcal{X} and it does not halt after $\phi(n, |\mathcal{X}|) + 1$ time (interrupted) \Rightarrow **Yes**

Simulation

n = input size, N = total number of negative circuits
 \mathcal{X} a collection of negative circuits

Suppose: algorithm **A** generates all N negative circuits in time $\phi(n, N)$, where $\phi(n, N)$ is *poly*(n, N)

Run **A** and interrupt it after $\phi(n, |\mathcal{X}|) + 1$ time (if it did not stop before)

A outputs a negative circuit not in $\mathcal{X} \Rightarrow$ **Yes**

A does not output a negative circuit not in \mathcal{X} and it does not halt after $\phi(n, |\mathcal{X}|) + 1$ time (interrupted) \Rightarrow **Yes**

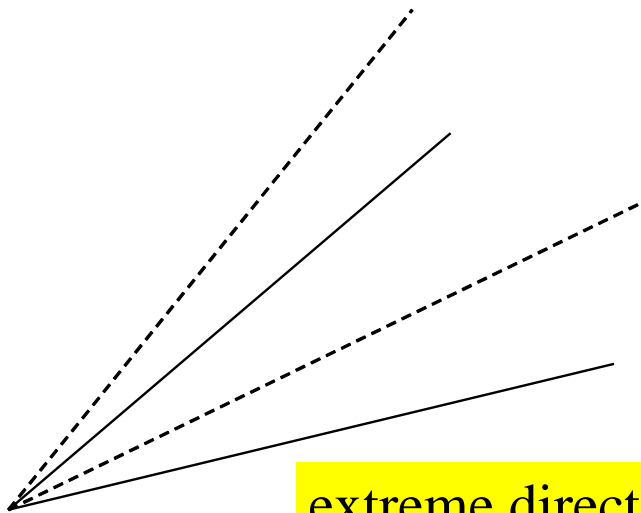
A outputs all negative circuits of \mathcal{X} and halts \Rightarrow **No**

Our Result - Proof

Proof - Circulation Cone

A - incidence matrix of a digraph

$$\{y : y^T A = 0, y \geq 0\}$$



extreme directions \leftrightarrow circuits

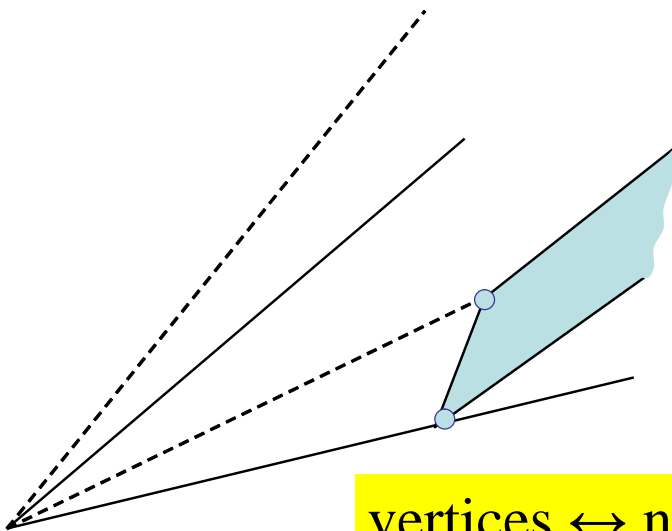
extreme direction $y \mapsto$ circuit $\{(u, v) \in E : y_{uv} \neq 0\}$

circuit $C \mapsto$ extreme direction: characteristic vector of C

Proof - Circulation Polyhedron

A - incidence matrix of a digraph, b - vector of edge weights

$$P = \{y : y^T A = 0, y \geq 0, b^T y = -1\}$$

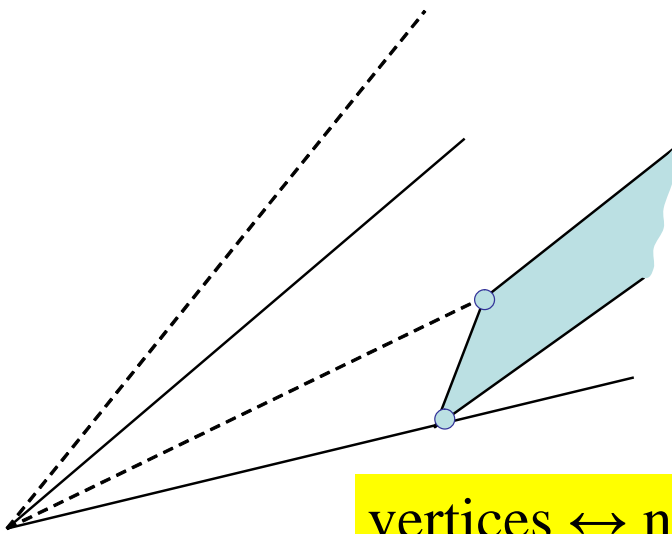


vertices ↔ negative circuits

Proof - Circulation Polyhedron

A - incidence matrix of a digraph, b - vector of edge weights

$$P = \{y : y^T A = 0, y \geq 0, b^T y = -1\}$$



vertices ↔ negative circuits

P is unbounded

y_1 negative circuit ($b^T y_1 = -1$)

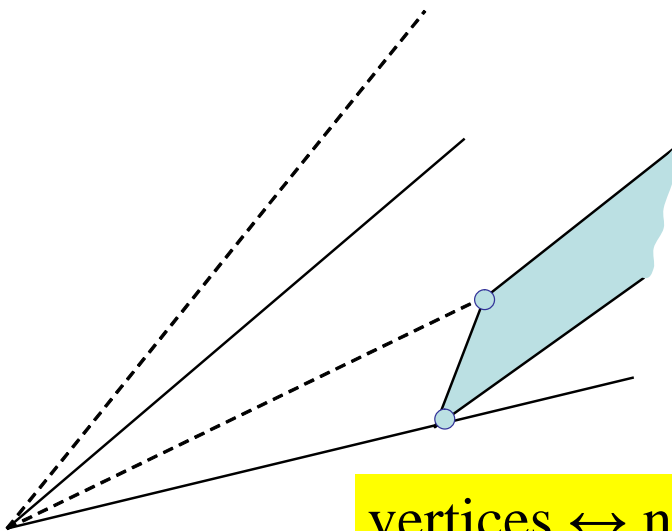
y_2 positive circuit ($b^T y_2 = 1$)

$y_1 + t(y_1 + y_2) \in P \quad \forall t \geq 0$

Proof - Circulation Polyhedron

A - incidence matrix of a digraph, b - vector of edge weights

$$P = \{y : y^T A = 0, y \geq 0, b^T y = -1\}$$



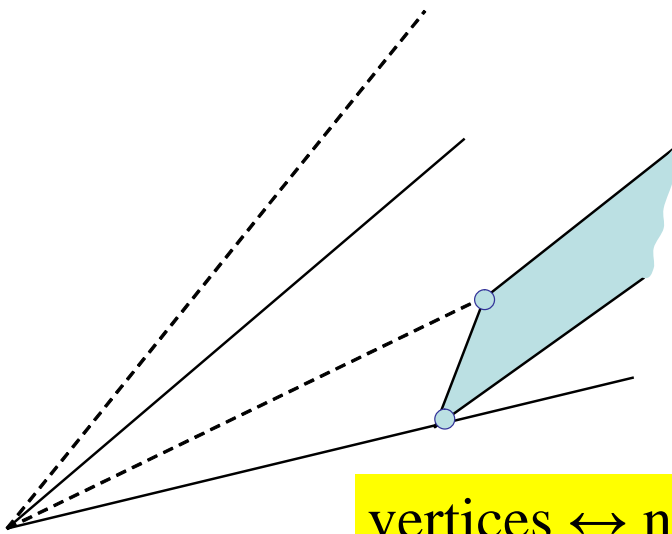
vertices \leftrightarrow negative circuits

Corollary 1. *Generating vertices of an unbounded polyhedron is NP-hard.*

Proof - Circulation Polyhedron

A - incidence matrix of a digraph, b - vector of edge weights

$$P = \{y : y^T A = 0, y \geq 0, b^T y = -1\}$$



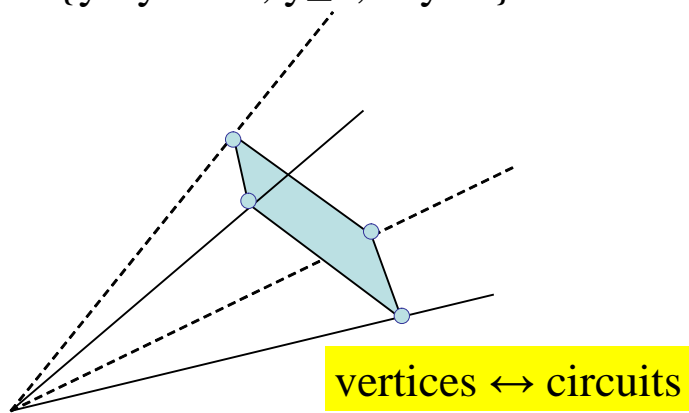
vertices ↔ negative circuits

vertices of $\{y : y^T A = 0, y \geq 0, b^T y = -1\} \longleftrightarrow$ minimal infeasible subsystems of $Ax \geq -b$

Corollary 2. *Generating minimal infeasible subsystems of a system of linear inequalities is NP-hard.*

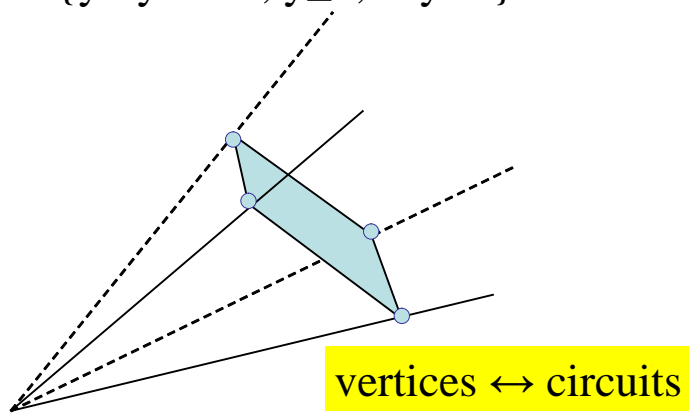
Proof - Circulation Polytope

$P = \{y : y^T A = 0, y \geq 0, 1^T y = 1\}$ is bounded

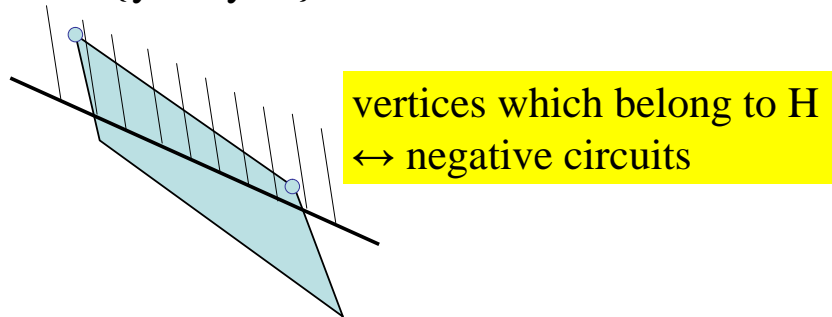


Proof - Circulation Polytope

$P = \{y : y^T A = 0, y \geq 0, 1^T y = 1\}$ is bounded

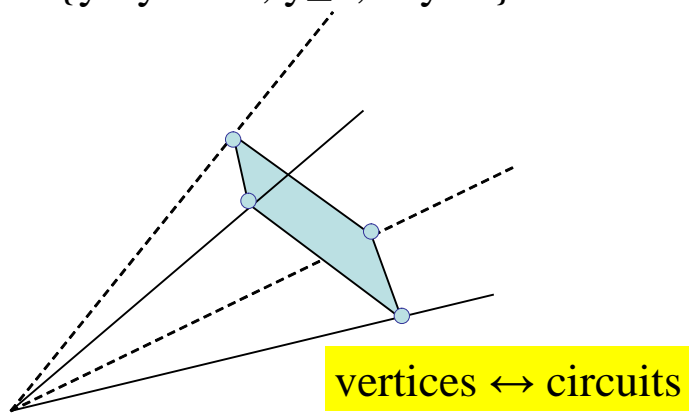


$H = \{y : b^T y < 0\}$

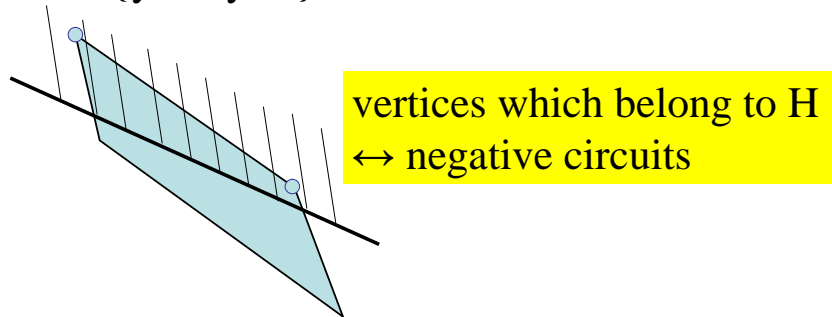


Proof - Circulation Polytope

$P = \{y : y^T A = 0, y \geq 0, 1^T y = 1\}$ is bounded

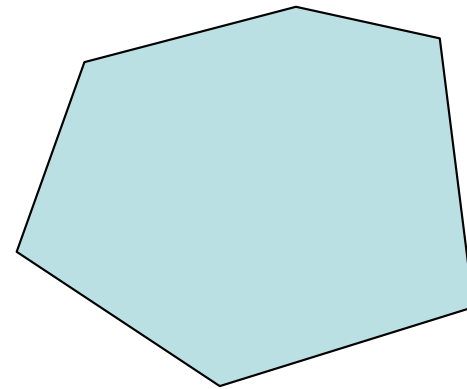
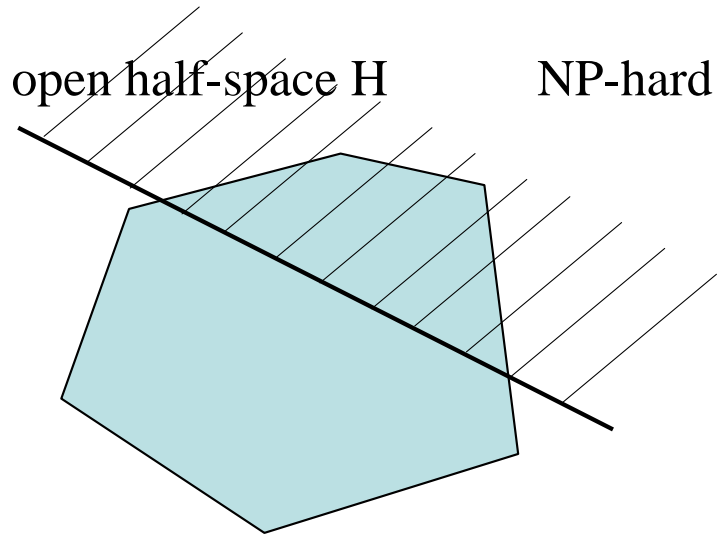
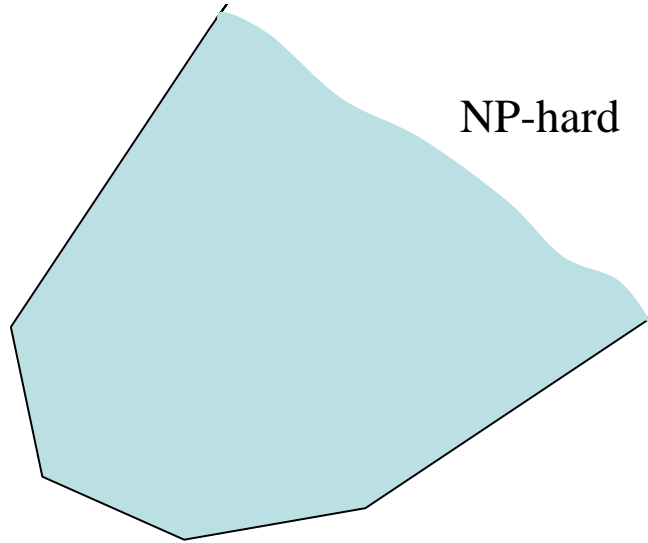


$H = \{y : b^T y < 0\}$



Corollary 5. *Generating vertices of P not in H is NP-hard.*

Vertex Generation For Bounded Polyhedra



Monotone Generation Problem

Monotone Generation Problem

Input: set E , monotone Boolean function $\pi : 2^E \rightarrow \{0, 1\}$

Output: list of all minimal subsets of E satisfying π

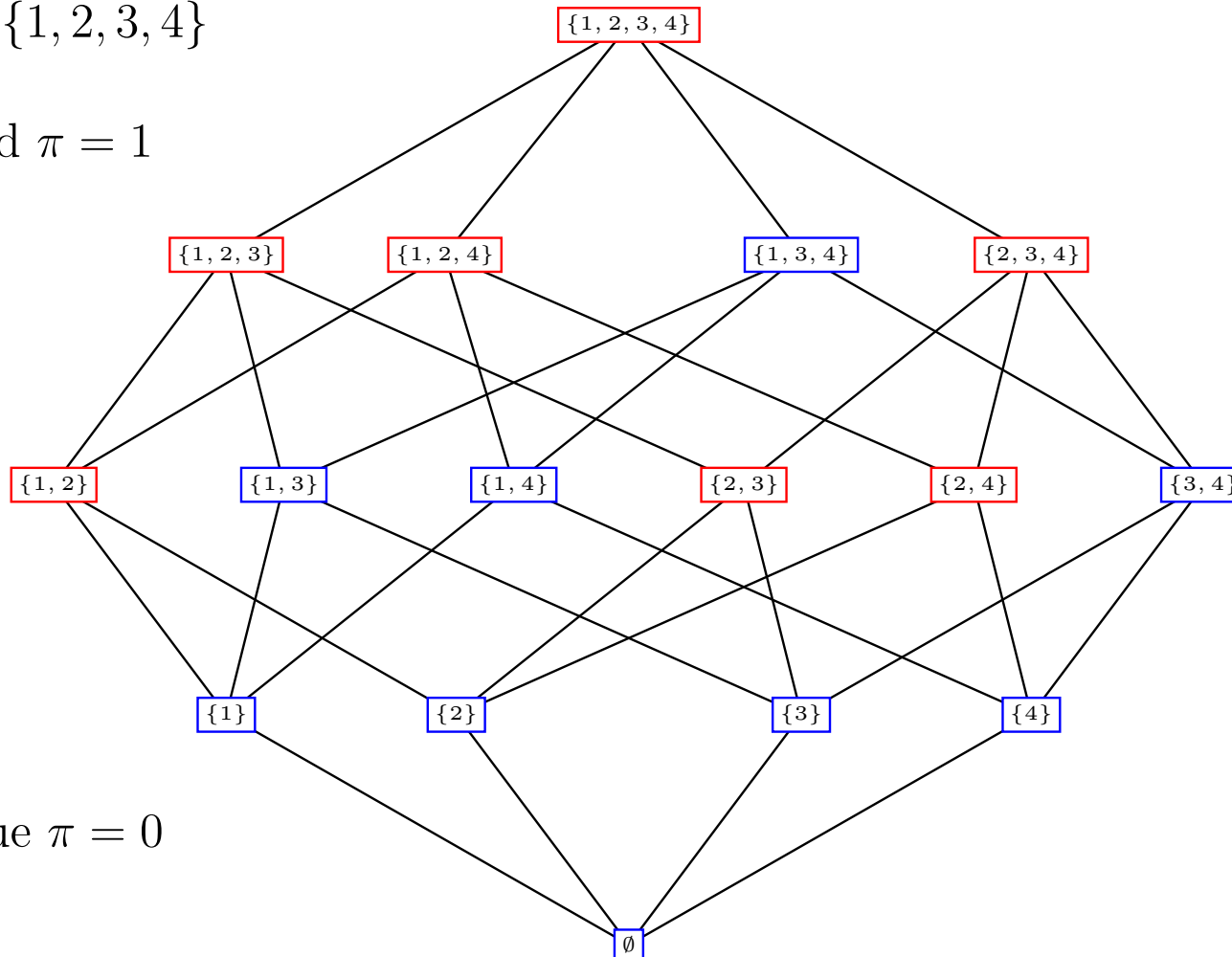
Monotone Generation Problem

Input: set E , monotone Boolean function $\pi : 2^E \rightarrow \{0, 1\}$

Output: list of all minimal subsets of E satisfying π

$$E = \{1, 2, 3, 4\}$$

red $\pi = 1$



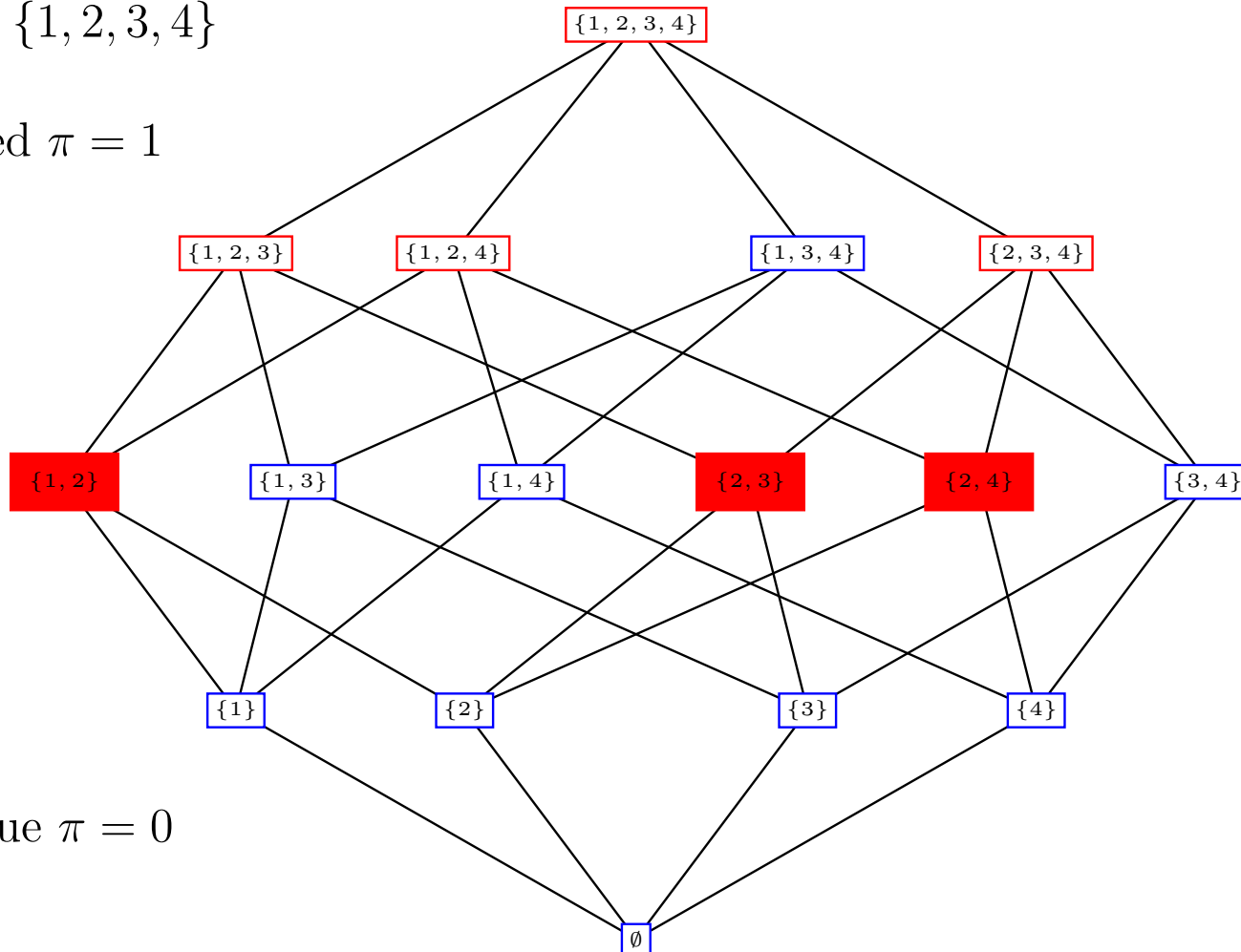
Monotone Generation Problem

Input: set E , monotone Boolean function $\pi : 2^E \rightarrow \{0, 1\}$

Output: list of all minimal subsets of E satisfying π

$$E = \{1, 2, 3, 4\}$$

red $\pi = 1$



Monotone Generation Problem

• graph $G = (V, E)$, subset X of edges

$$\pi(X) = \begin{cases} 1, & \text{if } (V, X) \text{ is } k\text{-vertex connected;} \\ 0, & \text{otherwise.} \end{cases}$$

Monotone Generation Problem

- graph $G = (V, E)$, subset X of edges

$$\pi(X) = \begin{cases} 1, & \text{if } (V, X) \text{ is } k\text{-vertex connected;} \\ 0, & \text{otherwise.} \end{cases}$$

- infeasible system $Ax \geq b$, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$

matrix $A(X)$ of rows of A whose indices belong to $X \subseteq \{1, \dots, m\}$

$$\pi(X) = \begin{cases} 1, & \text{if } A(X)x \geq b \text{ is an infeasible subsystem;} \\ 0, & \text{otherwise.} \end{cases}$$