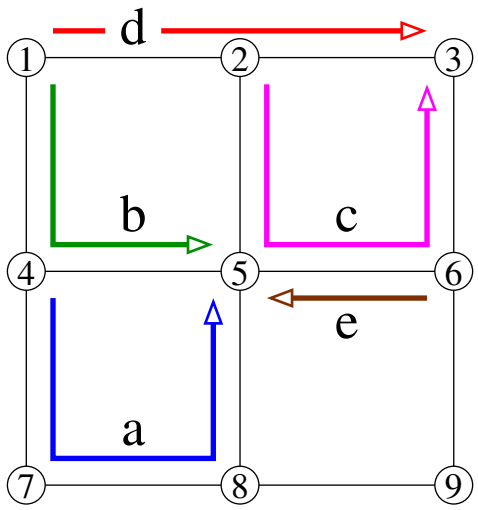
Compromis pour le reroutage et jeu de capture

Nathann Cohen
David Coudert
Dorian Mazauric
Napoleão Nepomuceno
Nicolas Nisse

MASCOTTE, INRIA, I3S, CNRS, Univ. Nice Sophia, Sophia Antipolis, France

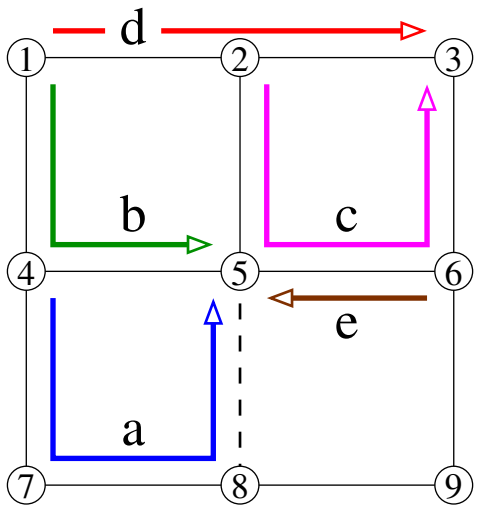
Example: maintenance operation

- Symmetric links, capacity 1
-



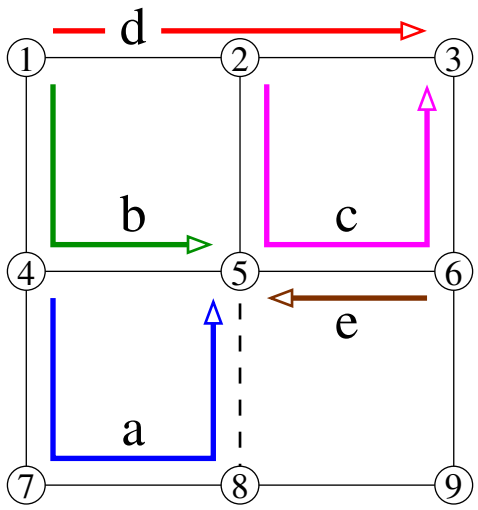
Example: maintenance operation

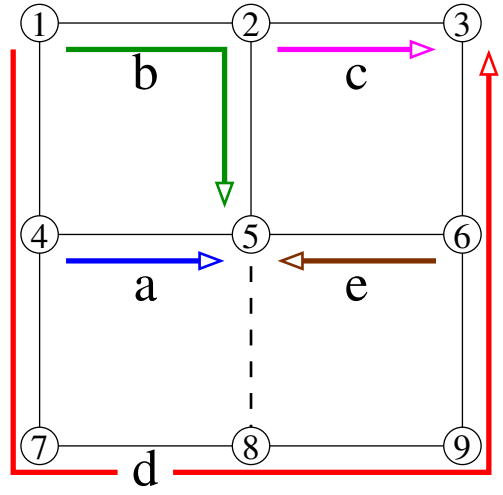
- Symmetric links, capacity 1
- Maintenance on link 5-8



Example: maintenance operation

- Symmetric links, capacity 1
- Maintenance on link 5-8





Context

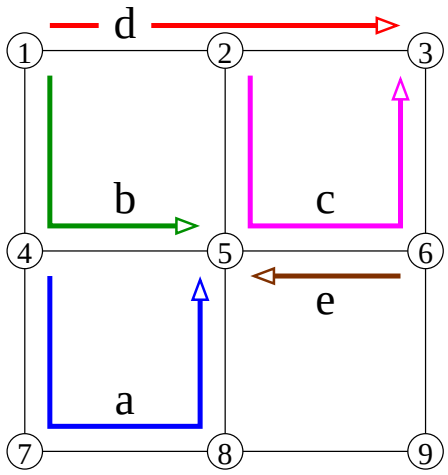
Circuit-switched networks

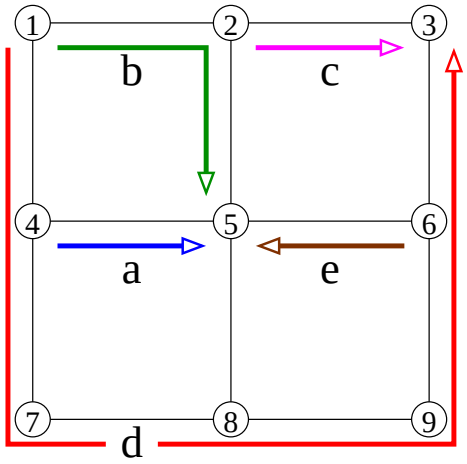
- Telephone: call repacking (70's)
- ATM
- WDM, MPLS

Motivation

- Optimize usage of resources (reduce blocking probability)
- Fault tolerance
- Maintenance operations

Our problem





Inputs: Set of connection requests
+ current **and** new routing

Output: Scheduling for rerouting connection requests from current to new routes

Objectives: Later

Constraint: Reroute requests one by one to their final routes

GMPLS

Make-before-break:

Establish new path before switching the connection

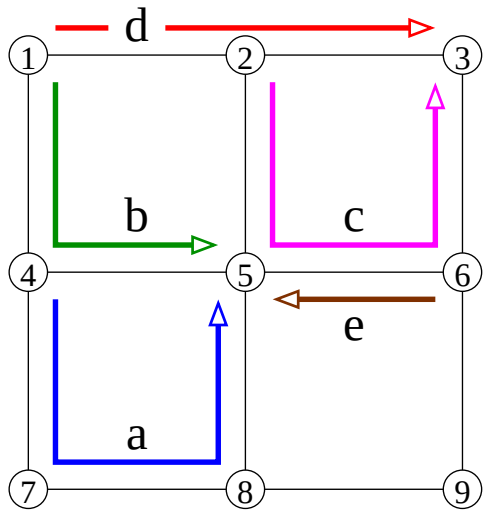
$\Longrightarrow$ Destination resources must be available

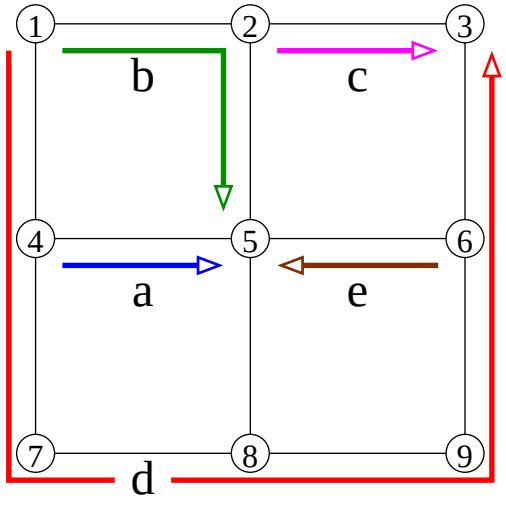
Break-before-make:

Break connection before establishing the new path

$\Longrightarrow$ Traffic stopped while new path not established

Dependency digraph





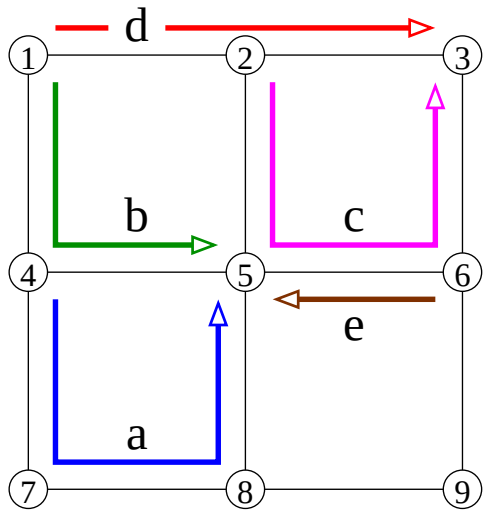
b before *a*

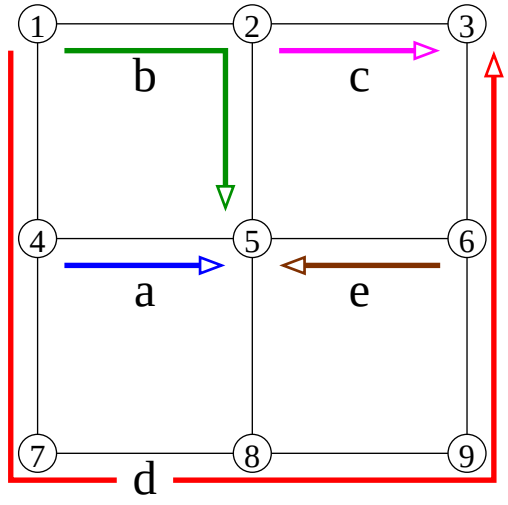
c and *d* before *b*

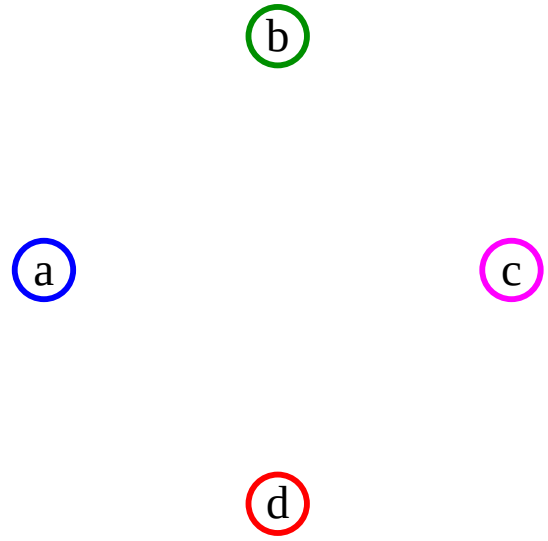
d before *c*

a, *b*, *c* before *d*

Dependency digraph







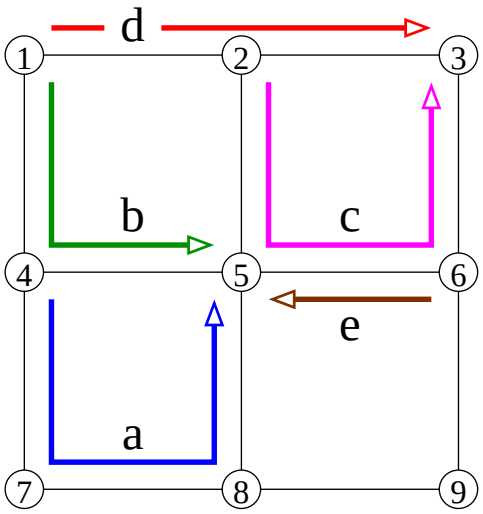
b before a

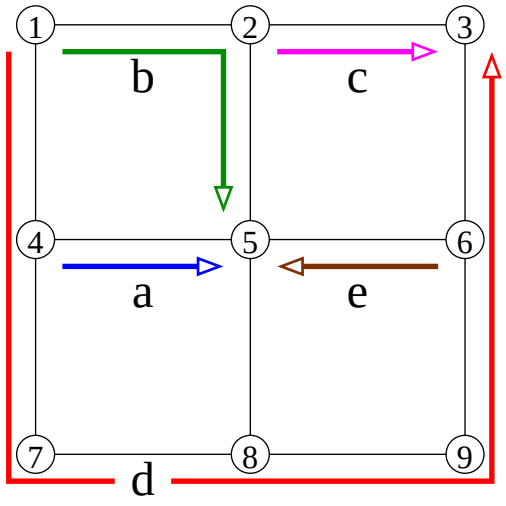
c and d before b

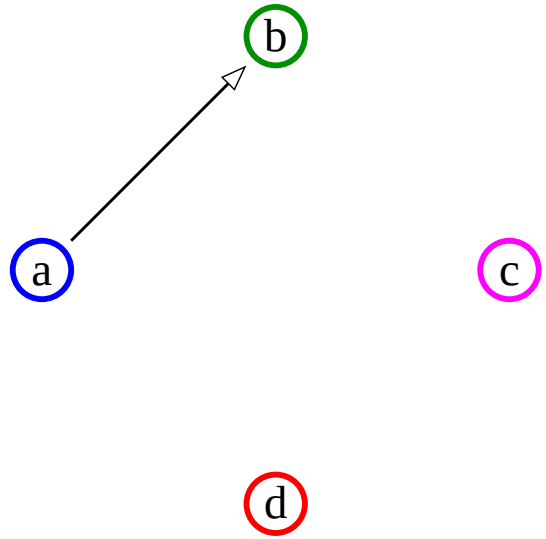
d before c

a, b, c before d

Dependency digraph







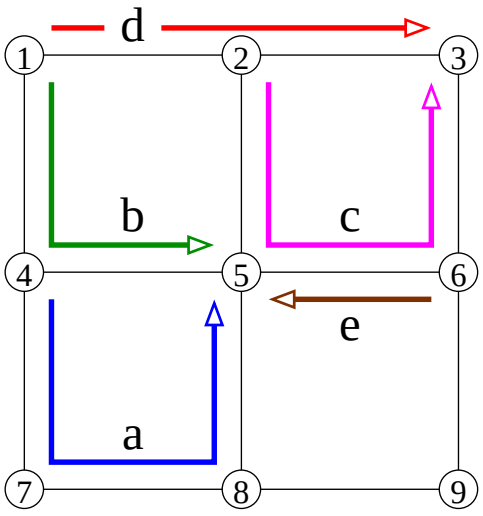
b before *a*

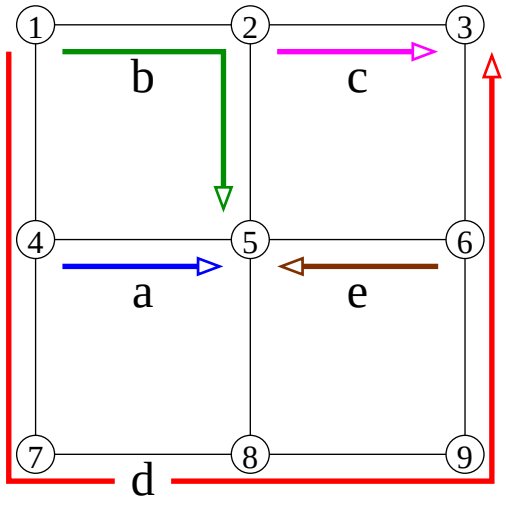
c and *d* before *b*

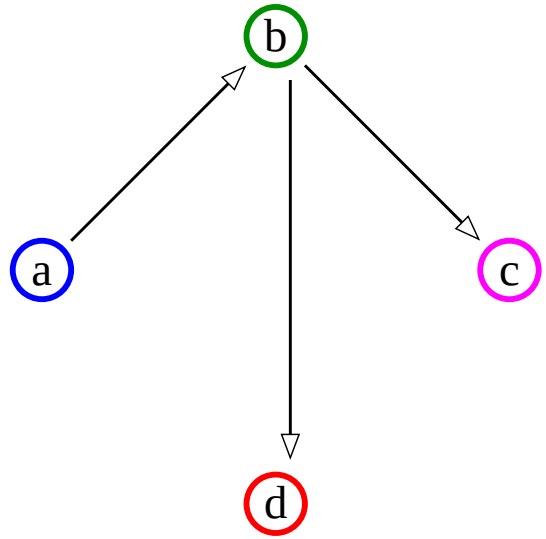
d before *c*

a, *b*, *c* before *d*

Dependency digraph







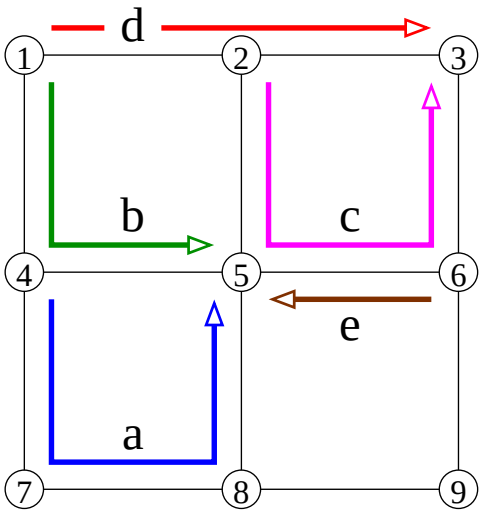
b before *a*

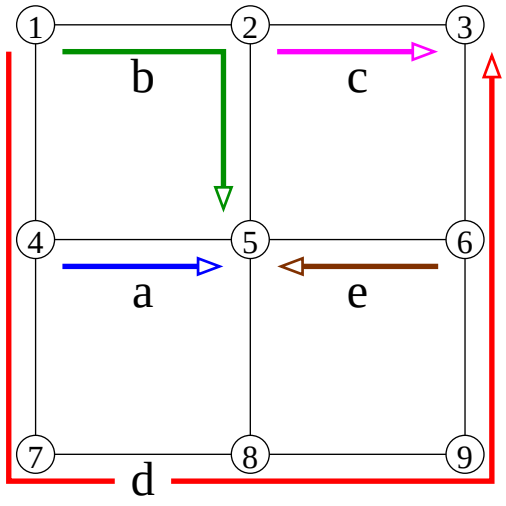
c and *d* before *b*

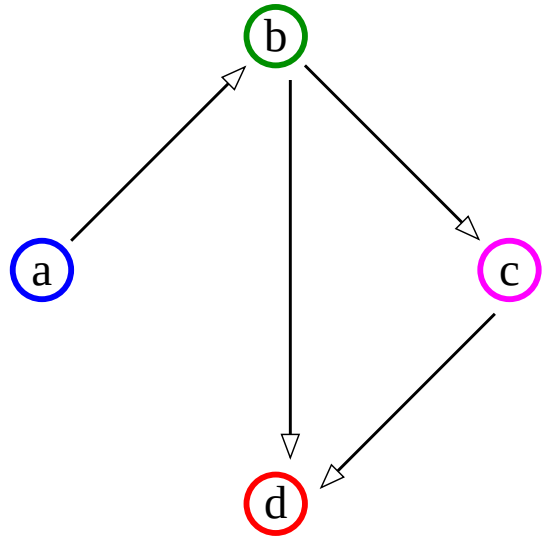
d before *c*

a, *b*, *c* before *d*

Dependency digraph







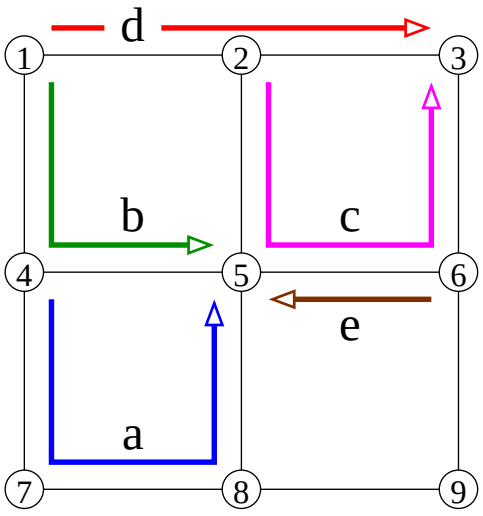
b before *a*

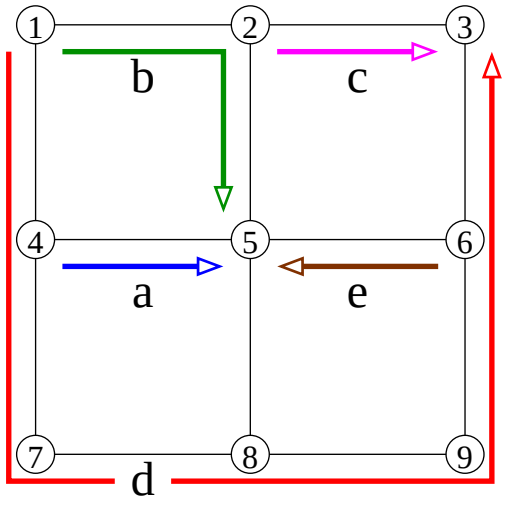
c and *d* before *b*

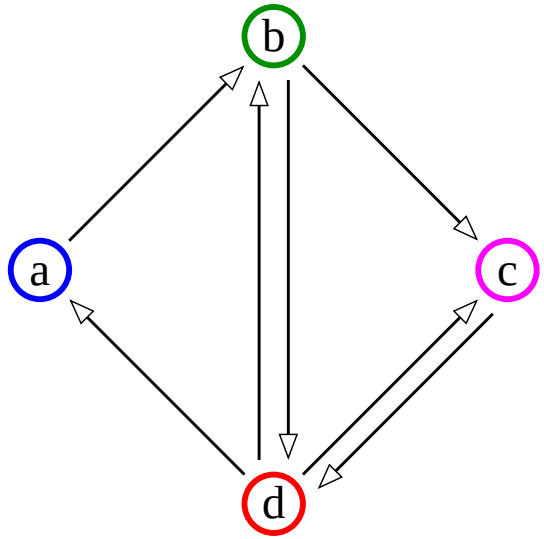
d before *c*

a, *b*, *c* before *d*

Dependency digraph







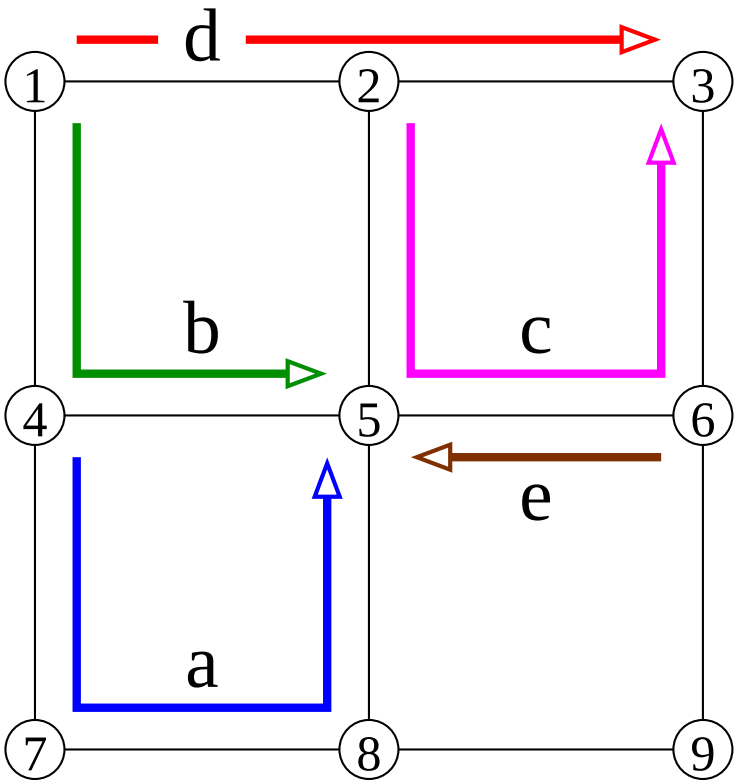
b before a

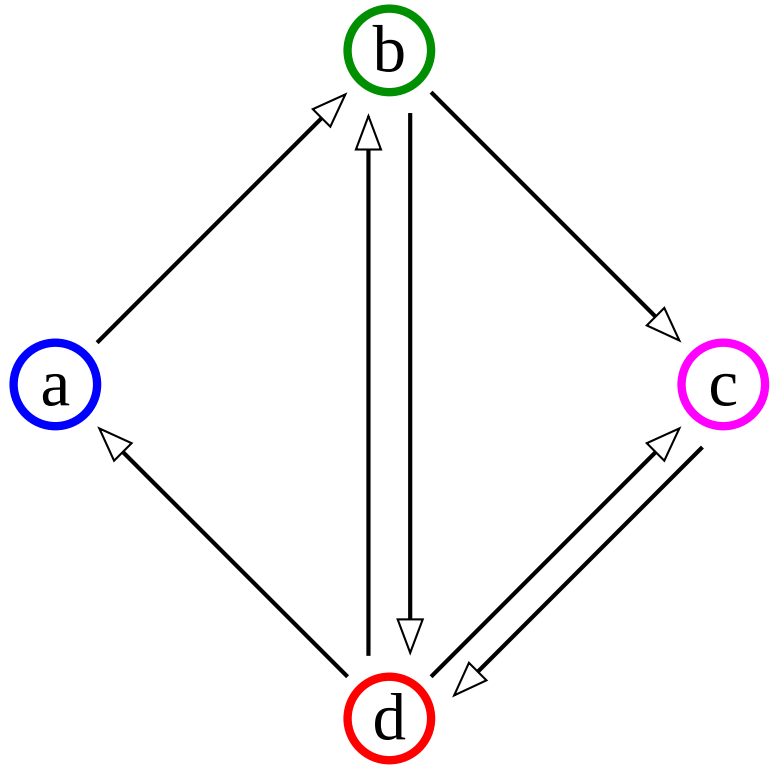
c and d before b

d before c

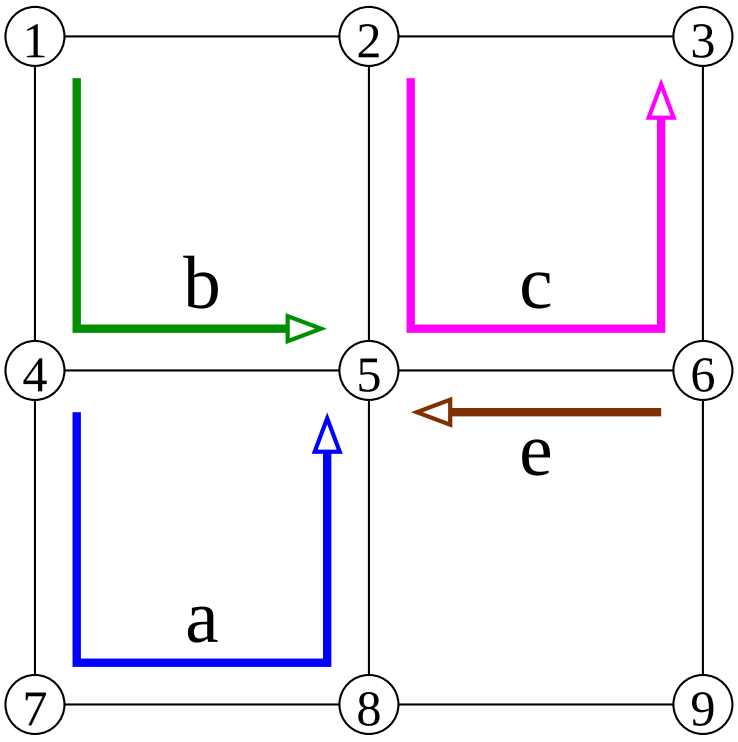
a, b, c before d

Example

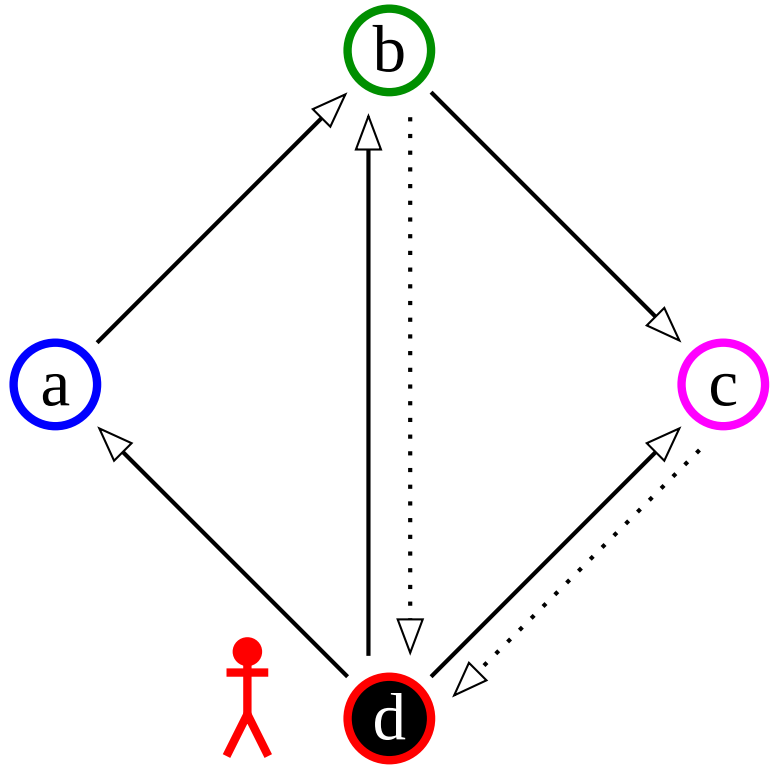




Example

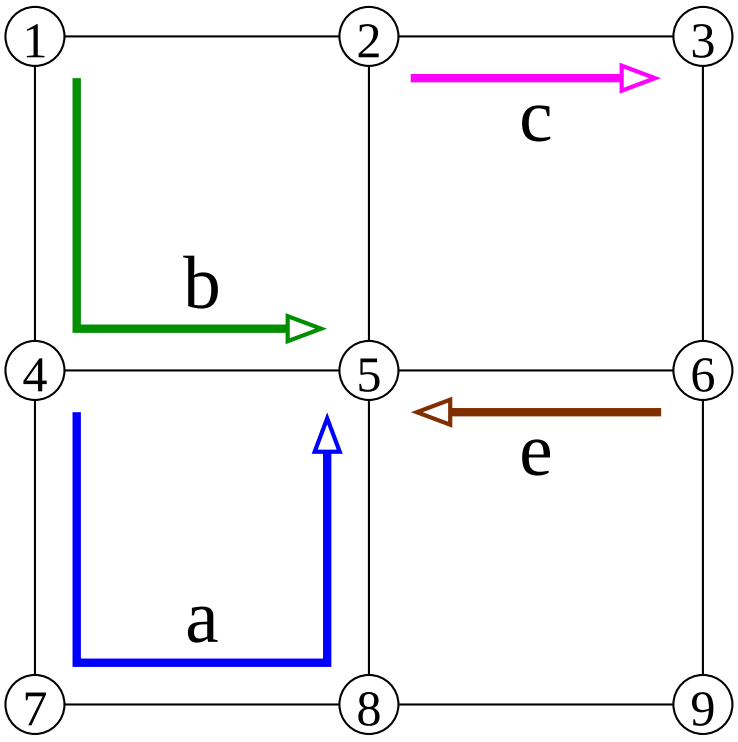


break request *d*

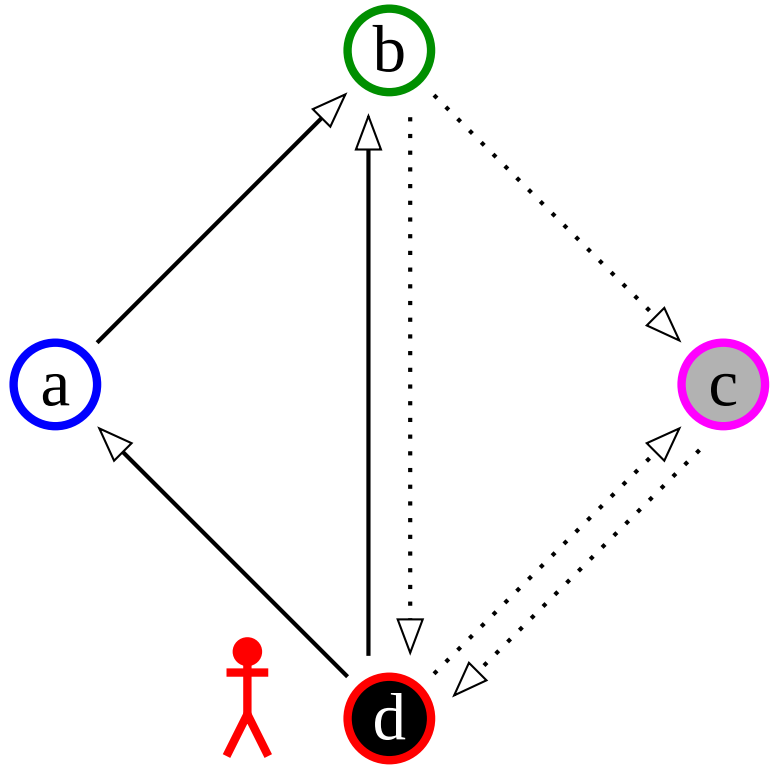


put an agent on node *d*

Example

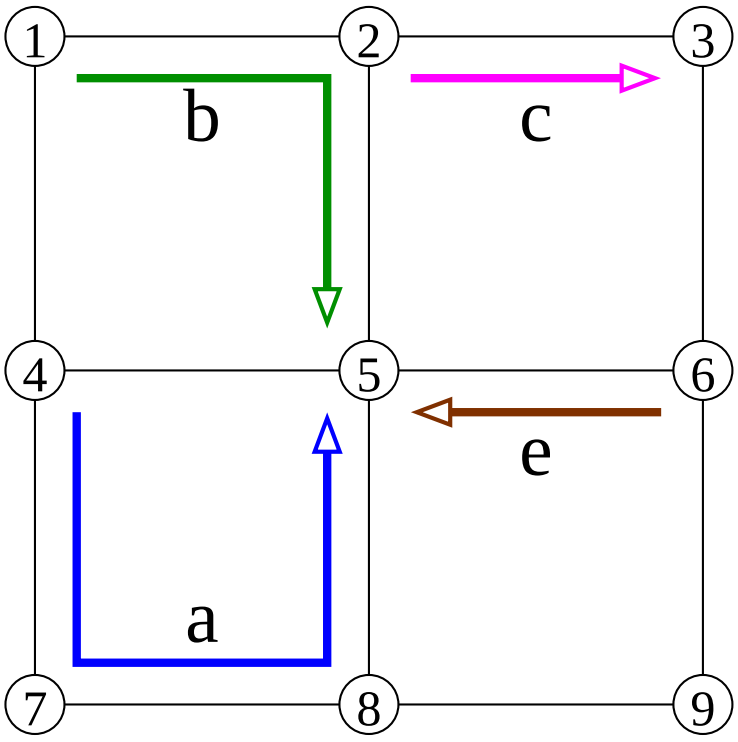


reroute request c

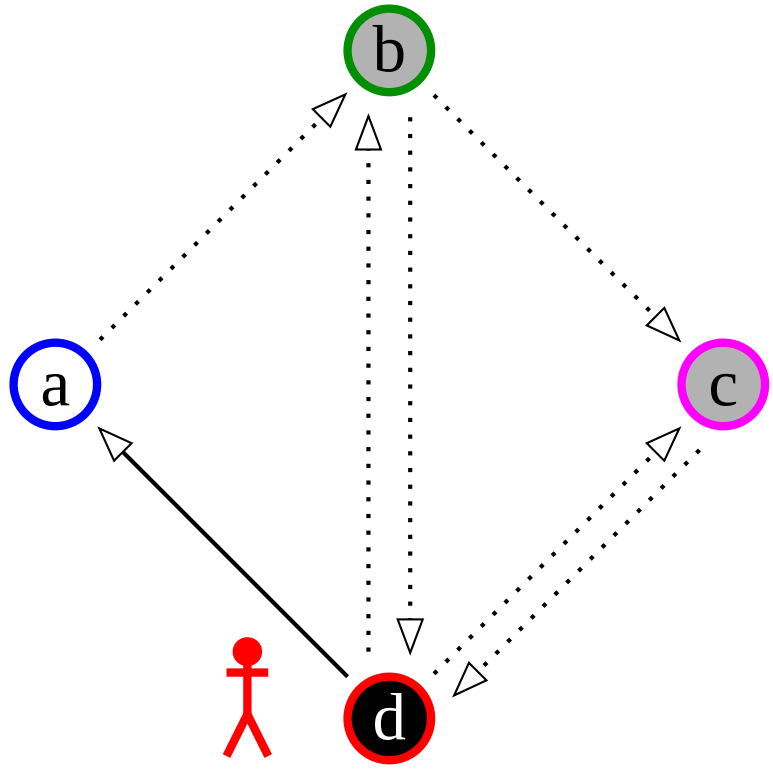


process node c

Example

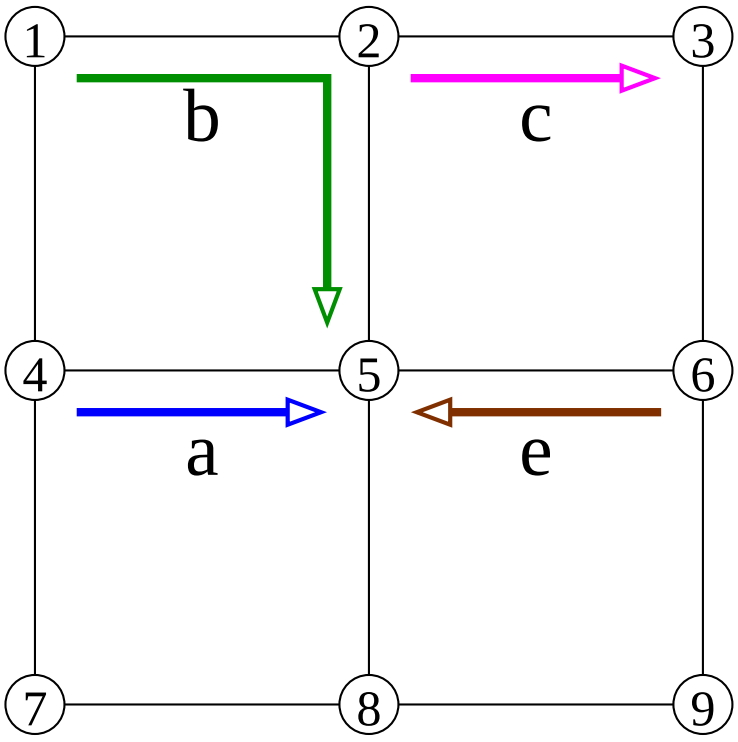


reroute request b

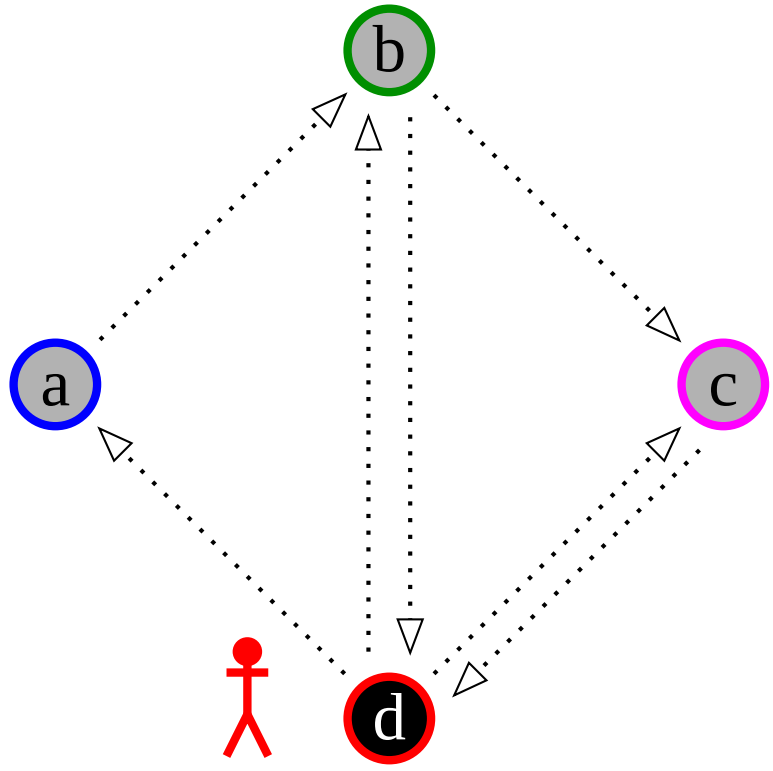


process node b

Example

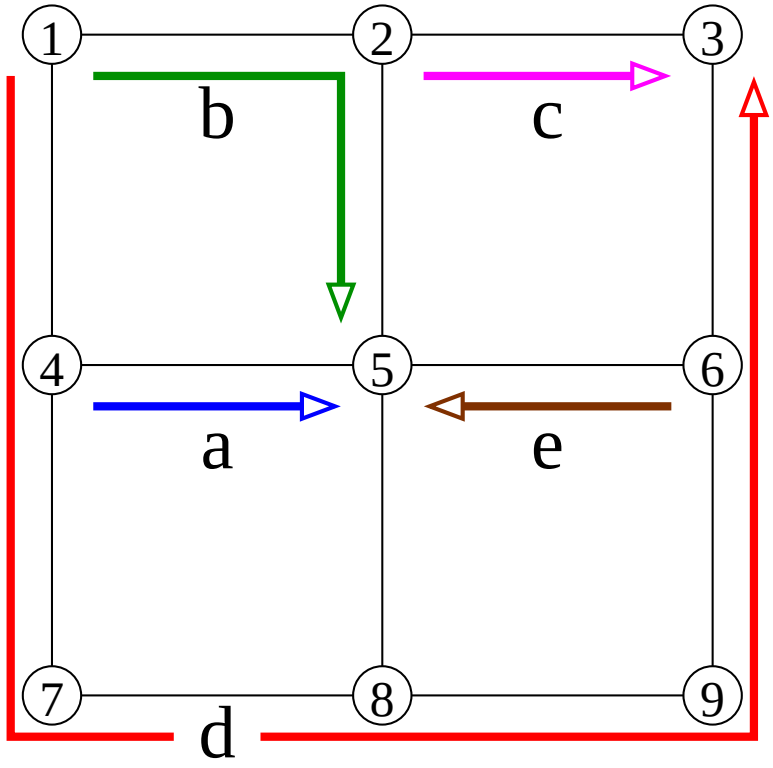


reroute request *a*

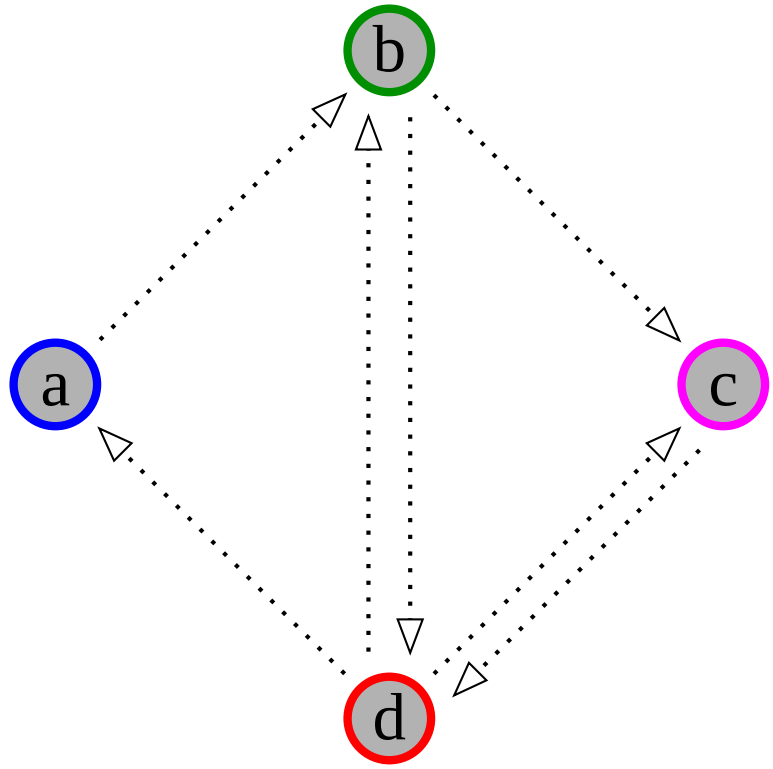


process node *a*

Example



route request *d*



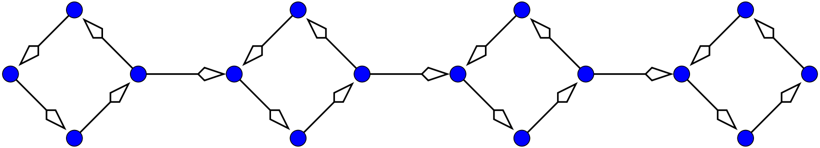
process node *d*
and remove agent

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number, here 1

Gap with MFVS up to $N/2$

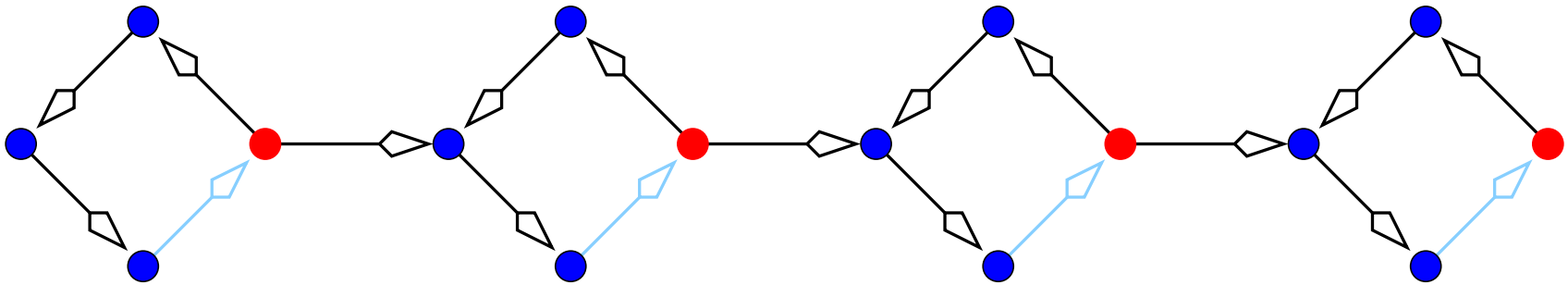
∼ Graph searching problem, cops-and-robber game, pursuit,...

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number here 1

= Cop-with MFVS in the K_2

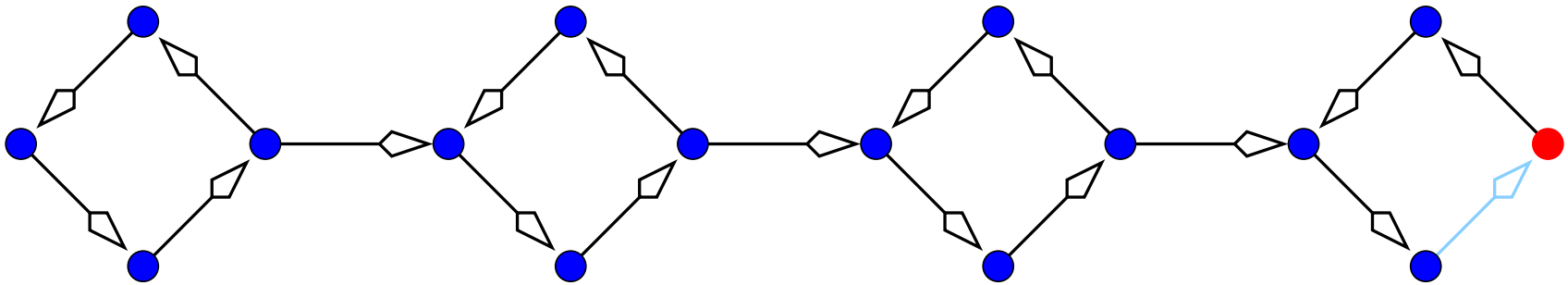
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number, here 1

- Gap with MFVS up to $N/2$

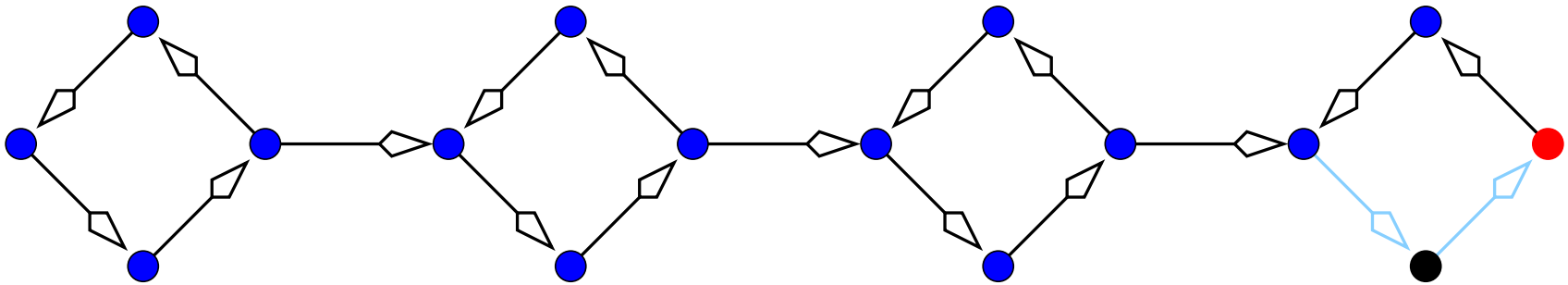
$\sim$ Graph searching problem, cops-and-robber game, pursuit,. . .

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number, here 1

- Gap with MFVS up to $N/2$

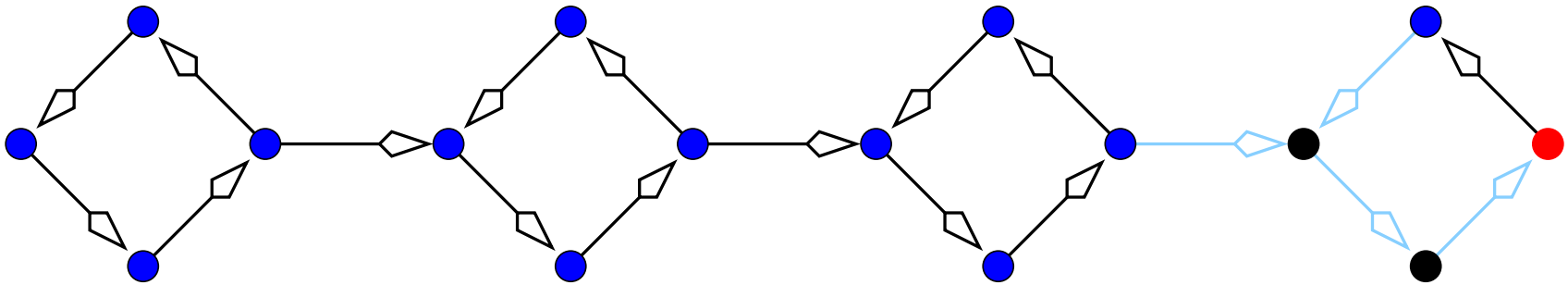
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number, here 1

- Gap with MFVS up to $N/2$

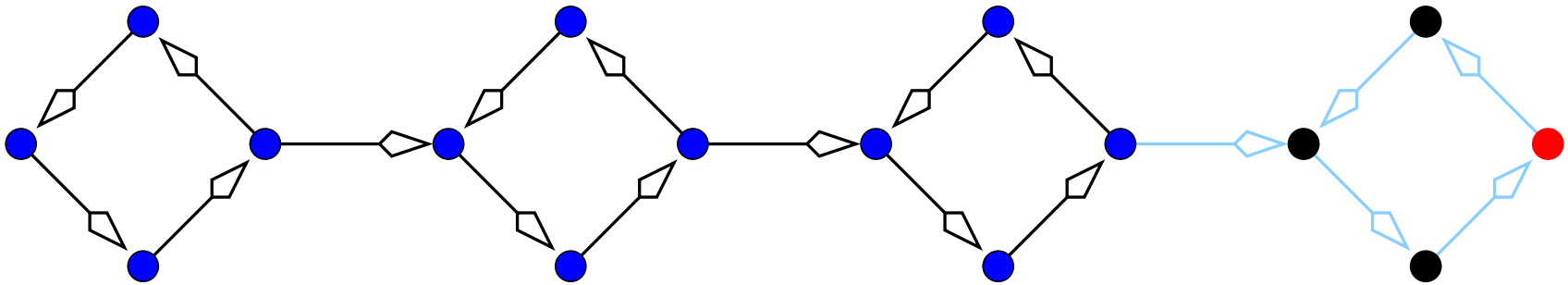
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number, here 1

- Gap with MFVS up to $N/2$

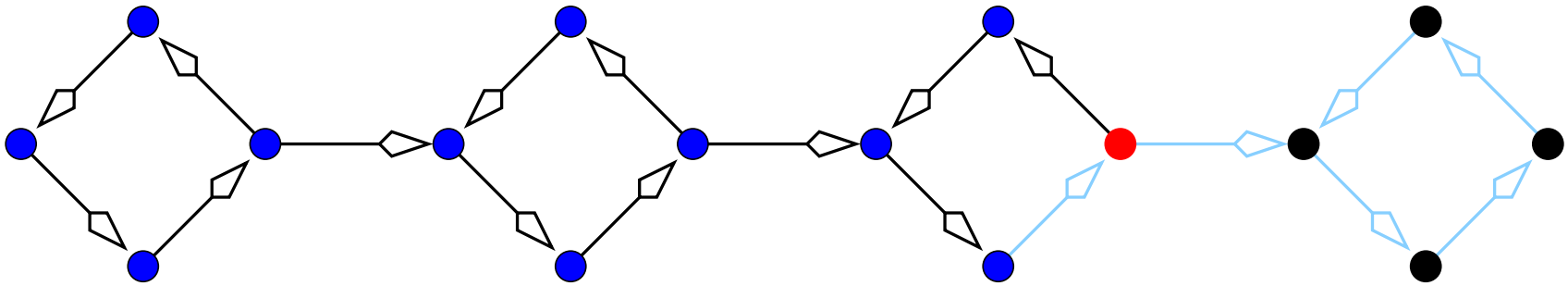
$\sim$ Graph searching problem, cops-and-robber game, pursuit,. . .

Objective Functions

MIN-TOTAL-DISRUPTION:

= Minimize overall number of interrupted connections

= Minimum Feedback Vertex Set (MFVS), here 4



MIN-MAX-DISRUPTION

= Minimize number of simultaneous interrupted connections

= Process Number, here 1

- Gap with MFVS up to $N/2$

$\sim$ Graph searching problem, cops-and-robber game, pursuit,. . .

Strategy

Rules

R_1 Put an agent on a vertex
  $=$ break/interrupt/route on temporary resources a connection

R_2 Process a vertex if all its out-neighbors are either processed or occupied by an agent
  $=$ (Re)route a connection when final resources are available

R_3 An agent can be re-used after the processing of the vertex

(p, q)-process strategy $=$ strategy to process D using p agents and q vertices covered by agents.

MIN-MAX-DISRUPTION $=$ minimize p, $pn(D)$

MIN-TOTAL-DISRUPTION $=$ minimize q, $mfvs(D)$

Example: DAG

Rules

R_1 Put an agent on a vertex

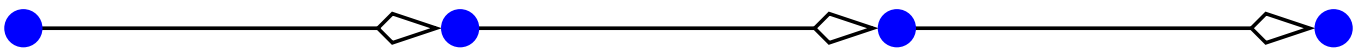
$=$ break/interrupt/route on temporary resources a connection

R_2 Process a vertex if all its out-neighbors are either processed or occupied by an agent

$=$ (Re)route a connection when final resources are available

R_3 An agent can be re-used after the processing of the vertex

Direct path, DAG



Th: If D is a DAG, then $pn(D) = 0$

Example: DAG

Rules

R_1 Put an agent on a vertex

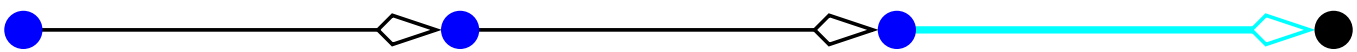
= break/interrupt/route on temporary resources a connection

R_2 Process a vertex if all its out-neighbors are either processed or occupied by an agent

= (Re)route a connection when final resources are available

R_3 An agent can be re-used after the processing of the vertex

Direct path, DAG



Th: If D is a DAG, then $pn(D) = 0$

Example: DAG

Rules

R_1 Put an agent on a vertex

= break/interrupt/route on temporary resources a connection

R_2 Process a vertex if all its out-neighbors are either processed or occupied by an agent

= (Re)route a connection when final resources are available

R_3 An agent can be re-used after the processing of the vertex

Direct path, DAG

Th: If D is a DAG, then $pn(D) = 0$

Example: DAG

Rules

R_1 Put an agent on a vertex

$=$ break/interrupt/route on temporary resources a connection

R_2 Process a vertex if all its out-neighbors are either processed or occupied by an agent

$=$ (Re)route a connection when final resources are available

R_3 An agent can be re-used after the processing of the vertex

Direct path, DAG

Th: If D is a DAG, then $pn(D) = 0$

Example: DAG

Rules

R_1 Put an agent on a vertex

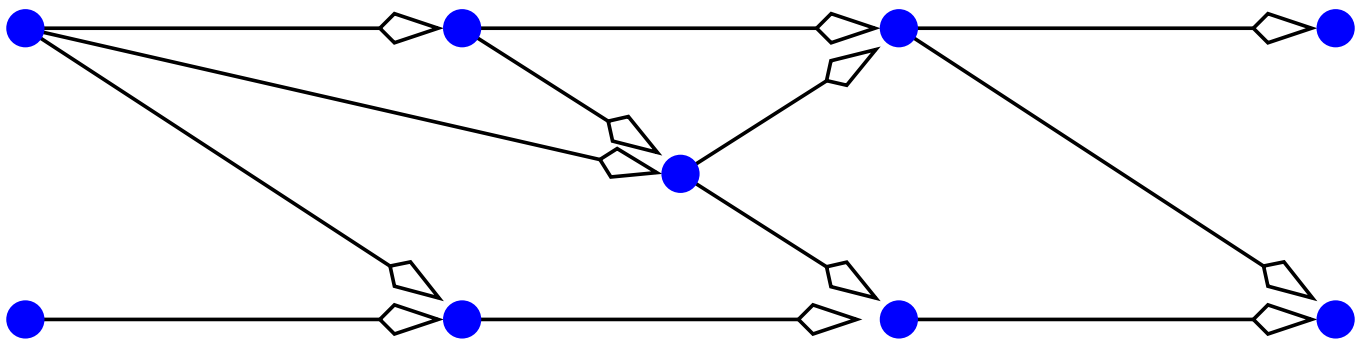
$=$ break/interrupt/route on temporary resources a connection

R_2 Process a vertex if all its out-neighbors are either processed or occupied by an agent

$=$ (Re)route a connection when final resources are available

R_3 An agent can be re-used after the processing of the vertex

Direct path, DAG



Th: If D is a DAG, then $pn(D) = 0$

What is known?

Parameters related to *pn*

- Pathwidth, *pw* [Robertson & Seymour, JCTB, 1983]
- Node search number, *ns* [Kirousis & Papadimitriou, TCS, 1986]
- Vertex separation, *vs* [Kinnersley, IPL, 1992]

Relations with *pn*

- $pw(G) = vs(G) = ns(G) - 1$
- $vs(D) \leq pn(D) \leq vs(D) + 1$ [Coudert & Sereni, 2007]

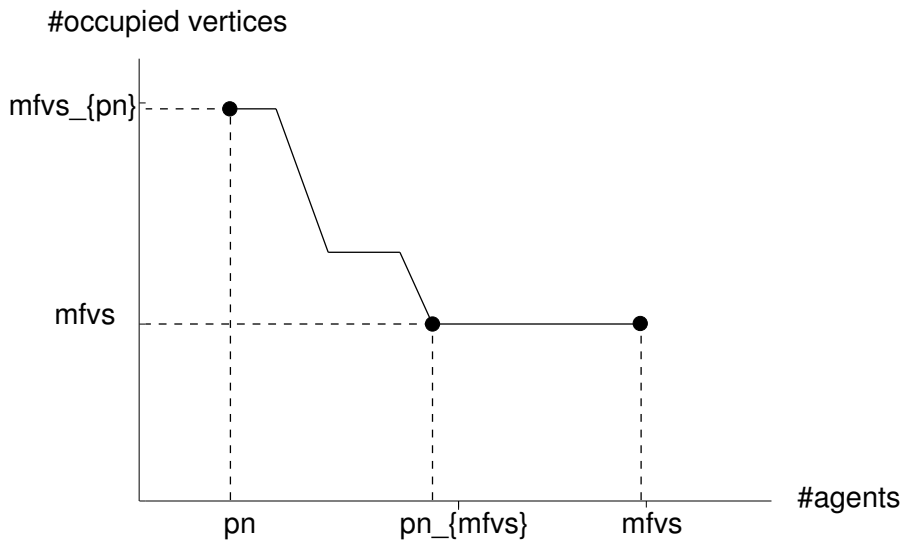
Complexity

- NP-Hard (*pn* and *mfvs*)
- Not APX (*pn* and *mfvs*)
  - = No polynomial time constant factor approximation algorithm

- Characterization of digraphs with process number 0, 1, 2
- Heuristic algorithms [Coudert, Huc, M, Nisse, Sereni, ONDM 09]

Tradeoff

- Smallest number of agents such that the number of occupied vertices is minimum $= pn_{mfvs}(D)$
- Smallest total number of occupied vertices such that the number of agents is minimum $= mfvs_{pn}(D)$



Tradeoff

- Smallest number of agents such that the number of occupied vertices is minimum $= pn_{mfvs}(D)$
- $\mu = \frac{pn_{mfvs}(D)}{pn(D)}$

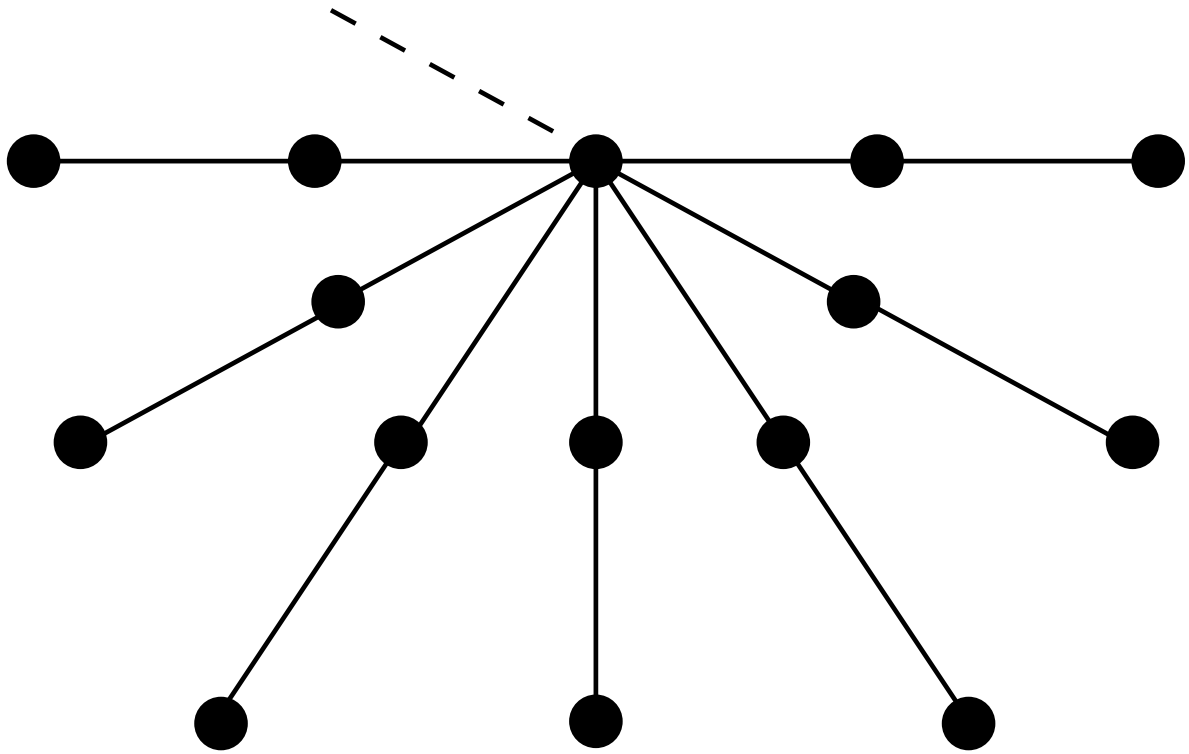
- Smallest total number of occupied vertices such that the number of agents is minimum $= mfvs_{pn}(D)$
- $\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$

Theorem

The problems of determining $pn_{mfvs}(D)$, $mfvs_{pn}(D)$, μ, and λ are NP-Complete and not APX.

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



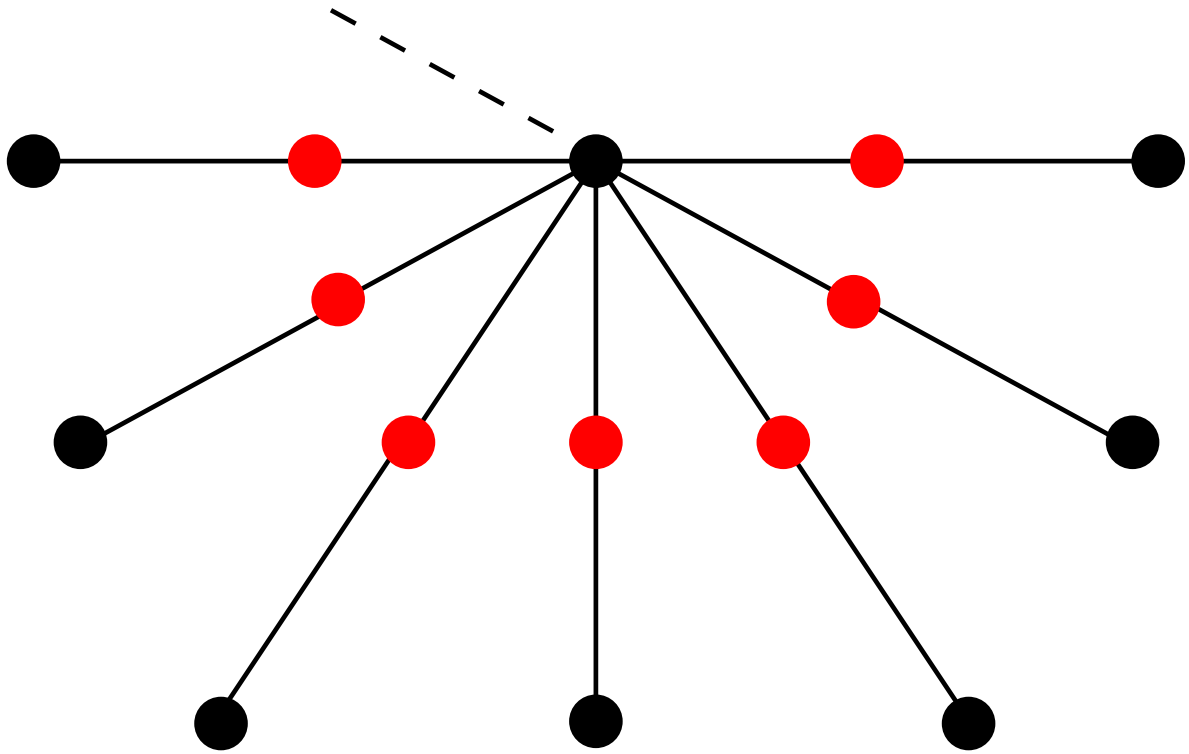
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



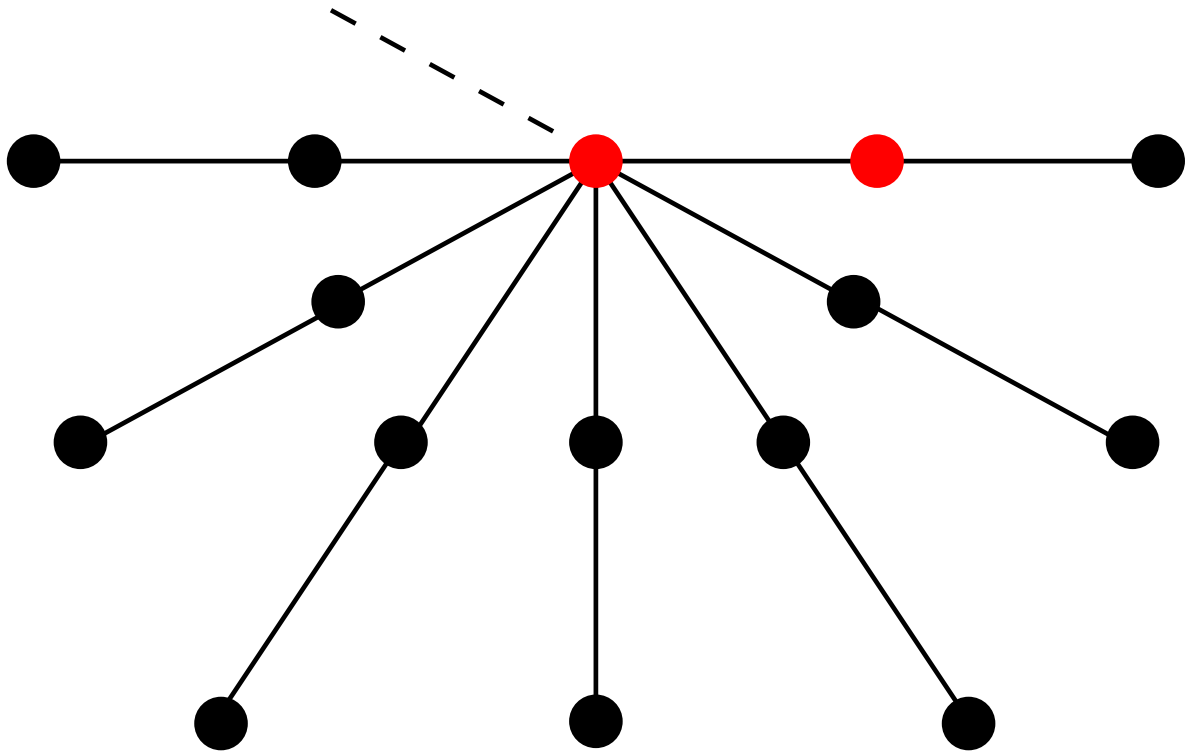
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



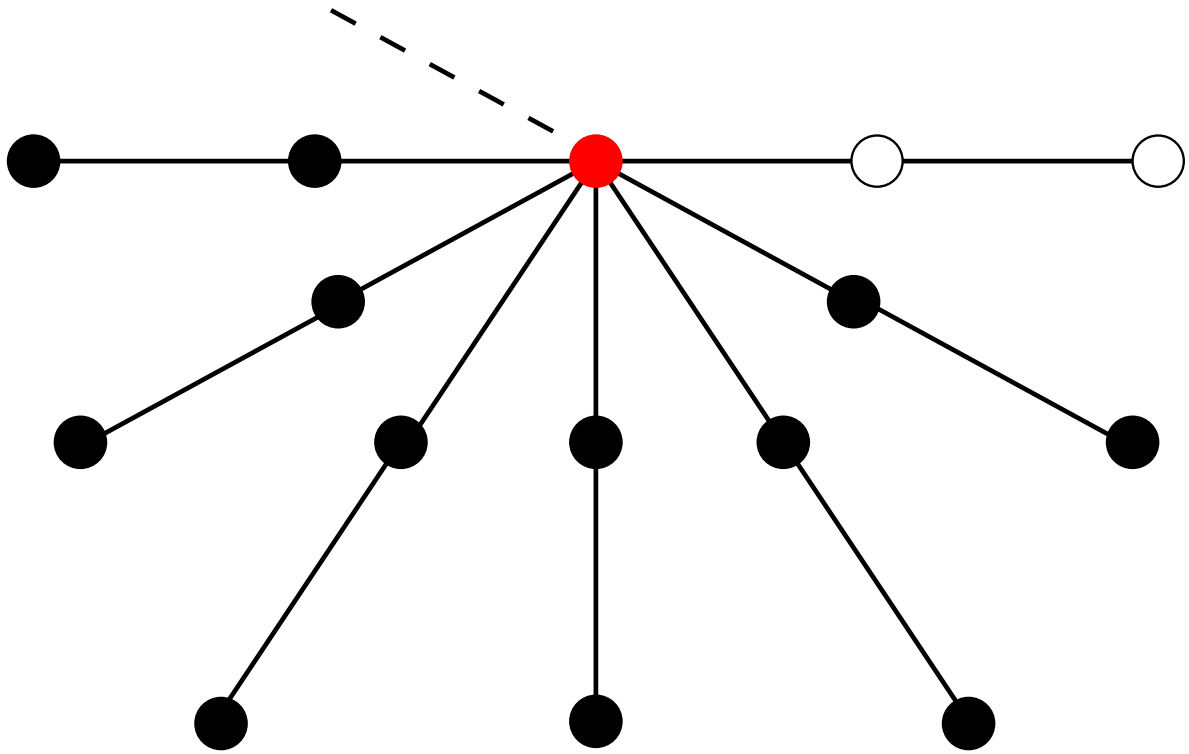
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



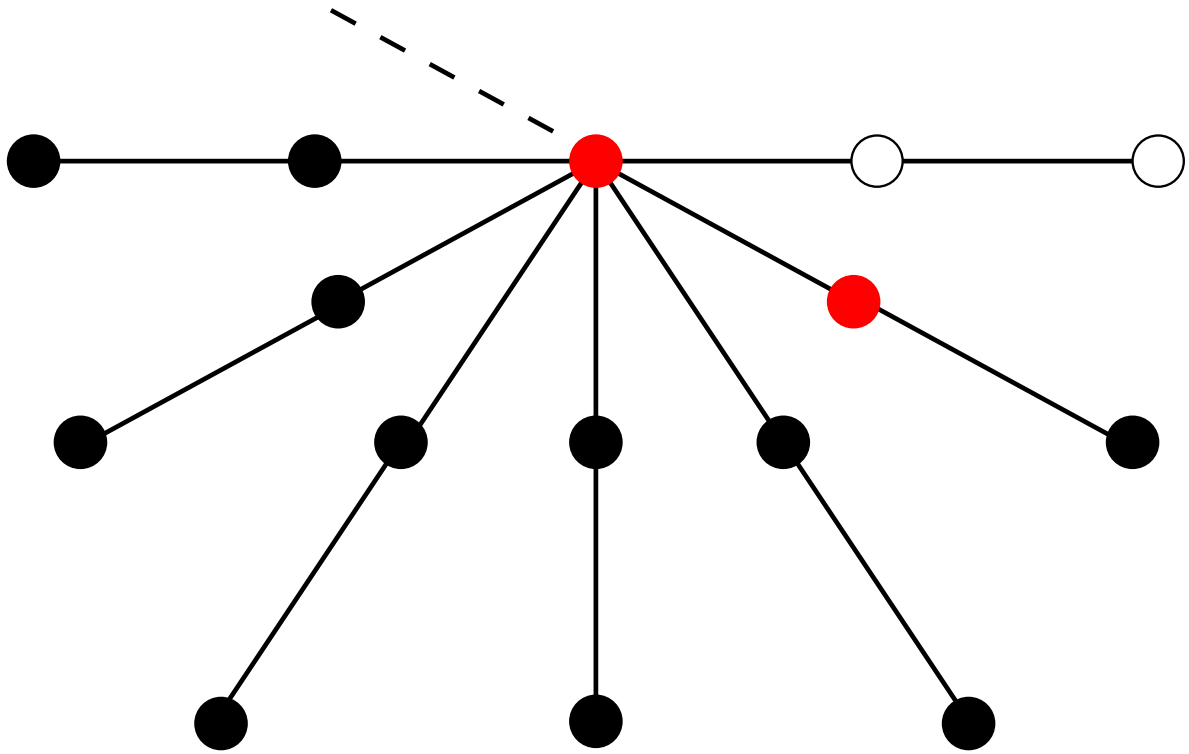
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



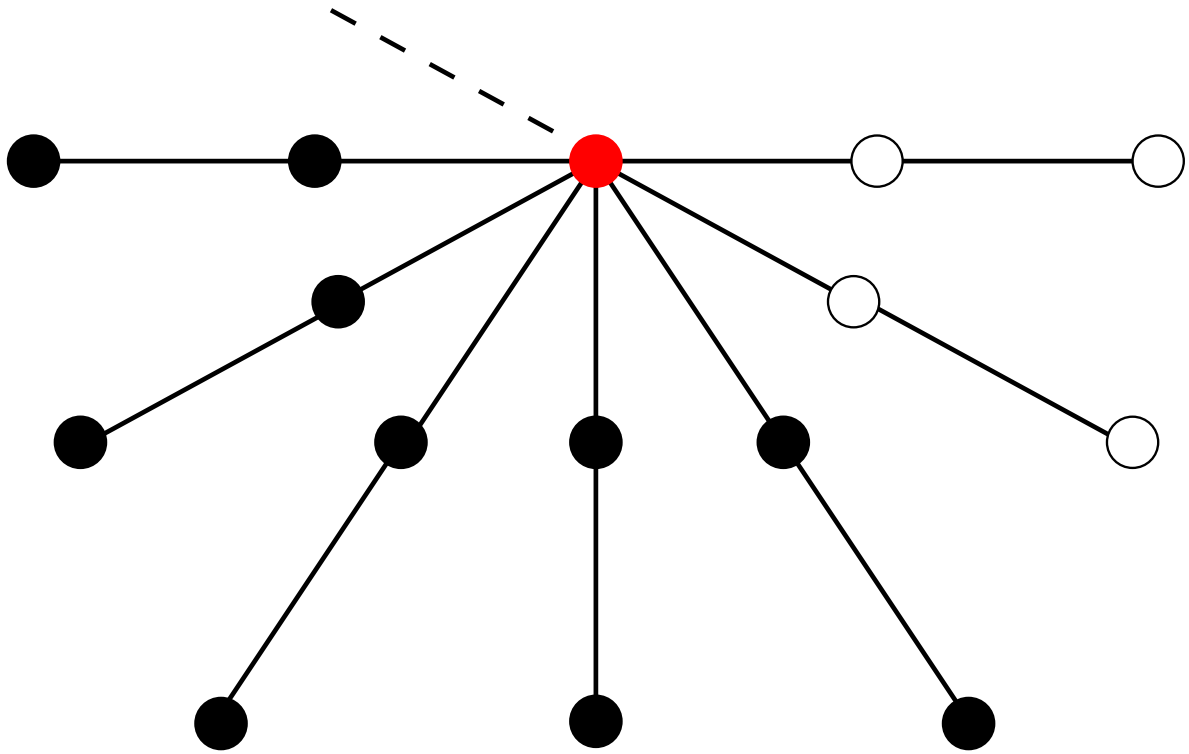
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



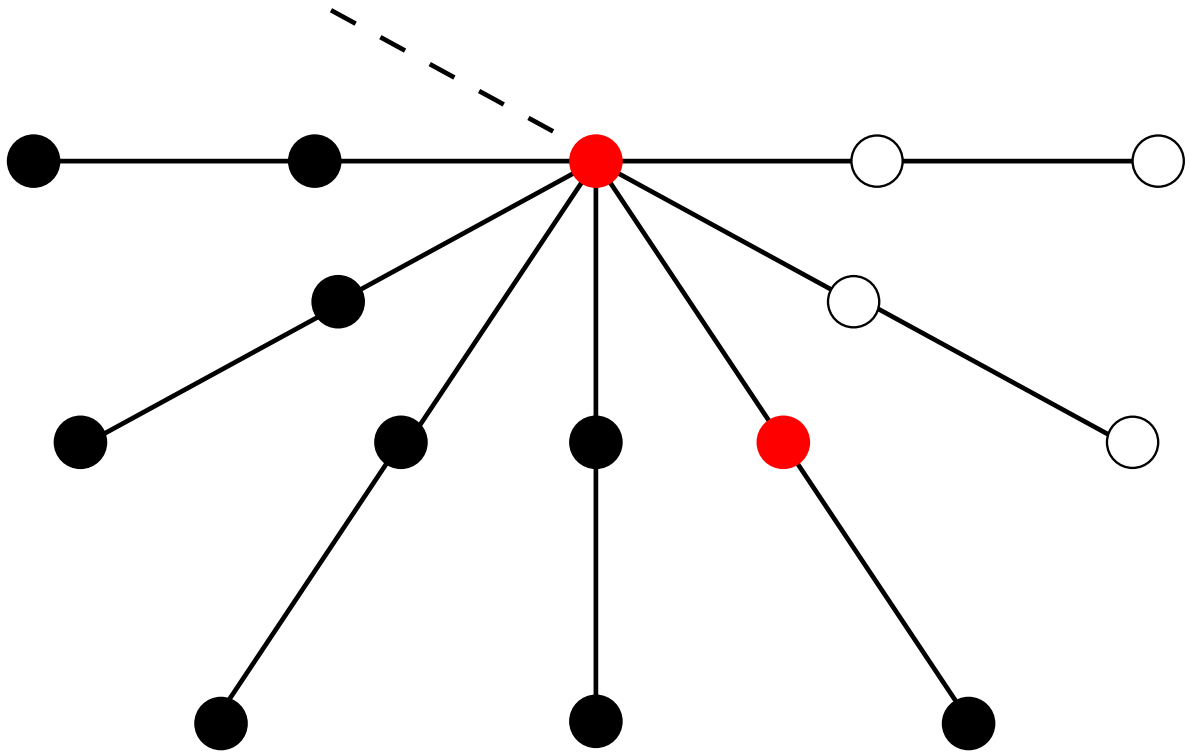
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



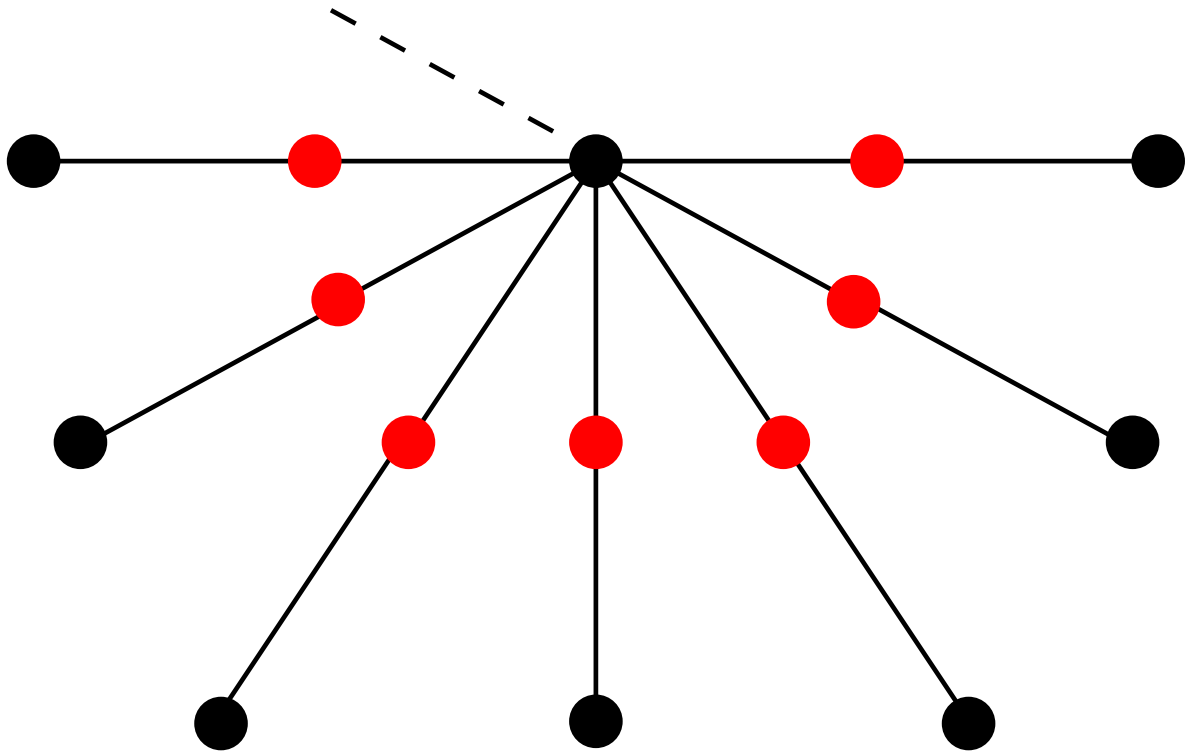
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$$\mu = \frac{pn_{mfvs}(D)}{pn(D)}.$$



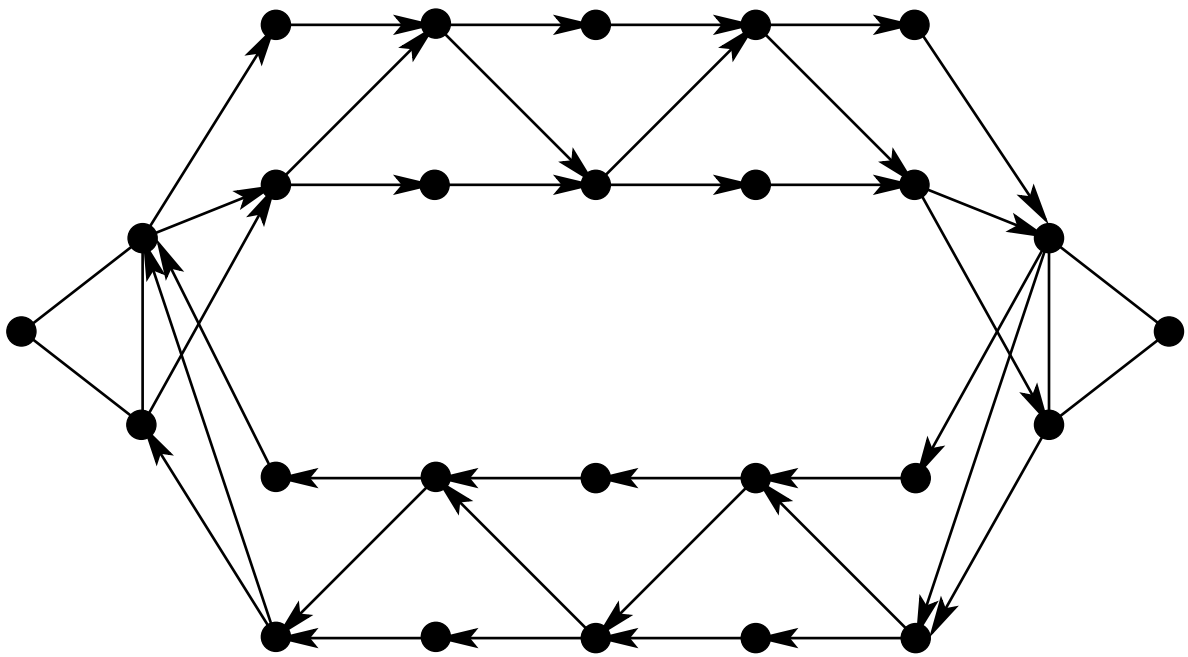
$mfvs(D) = n$

$pn(D) = 2$

$pn_{mfvs}(D) = n$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



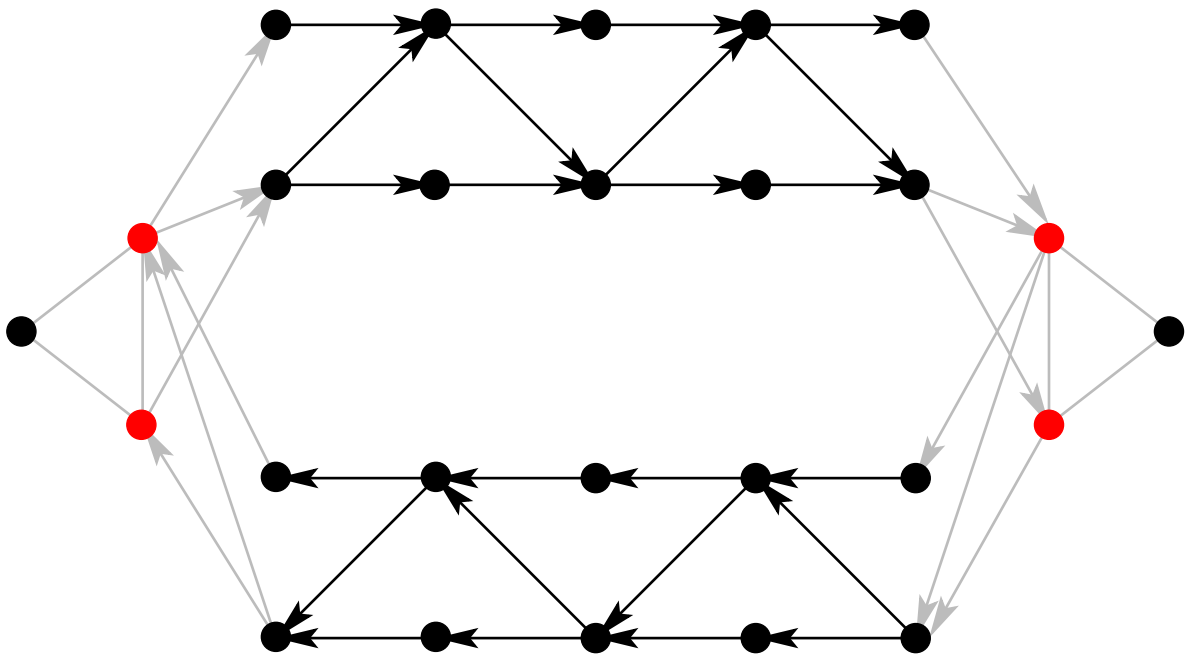
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



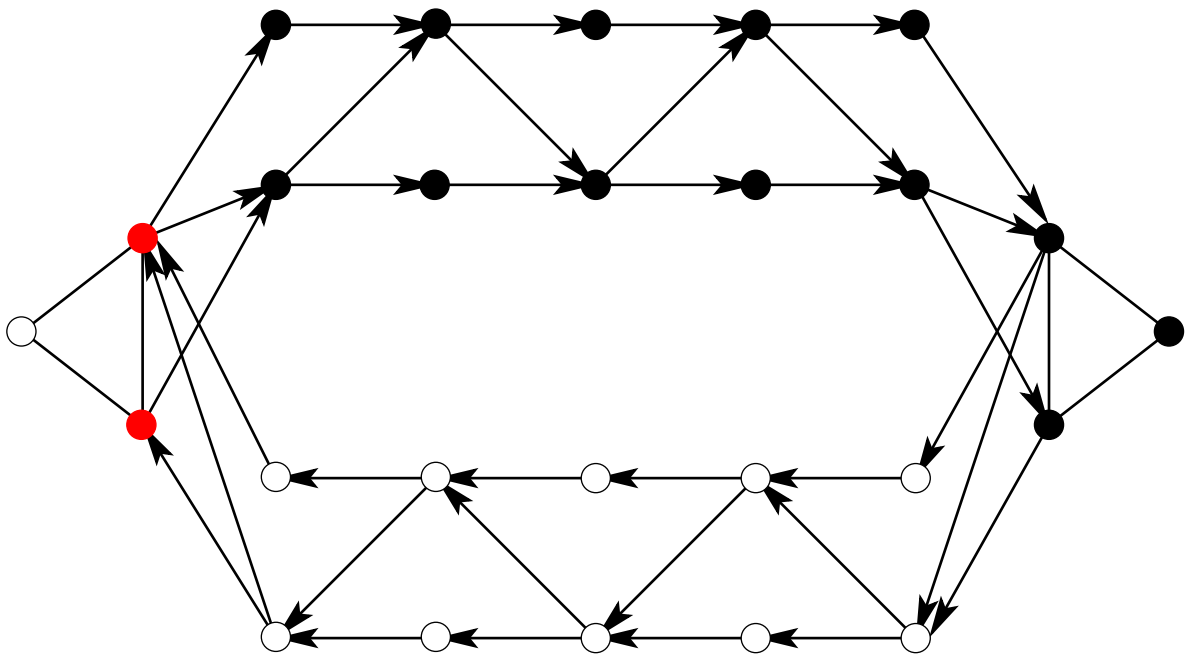
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



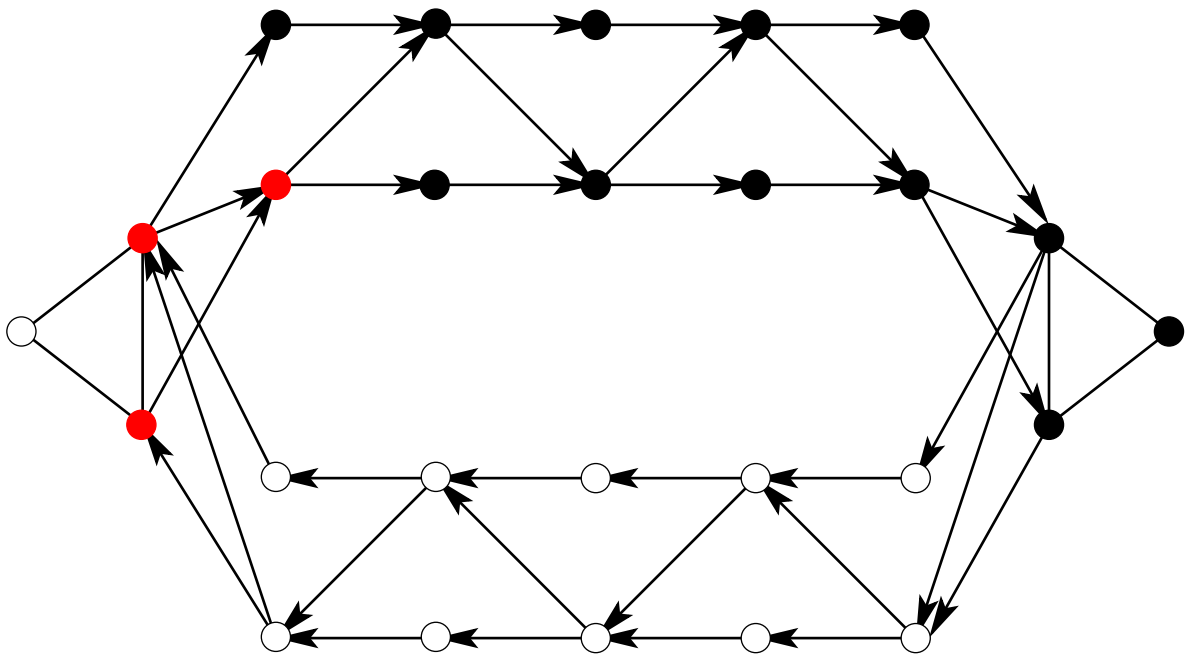
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



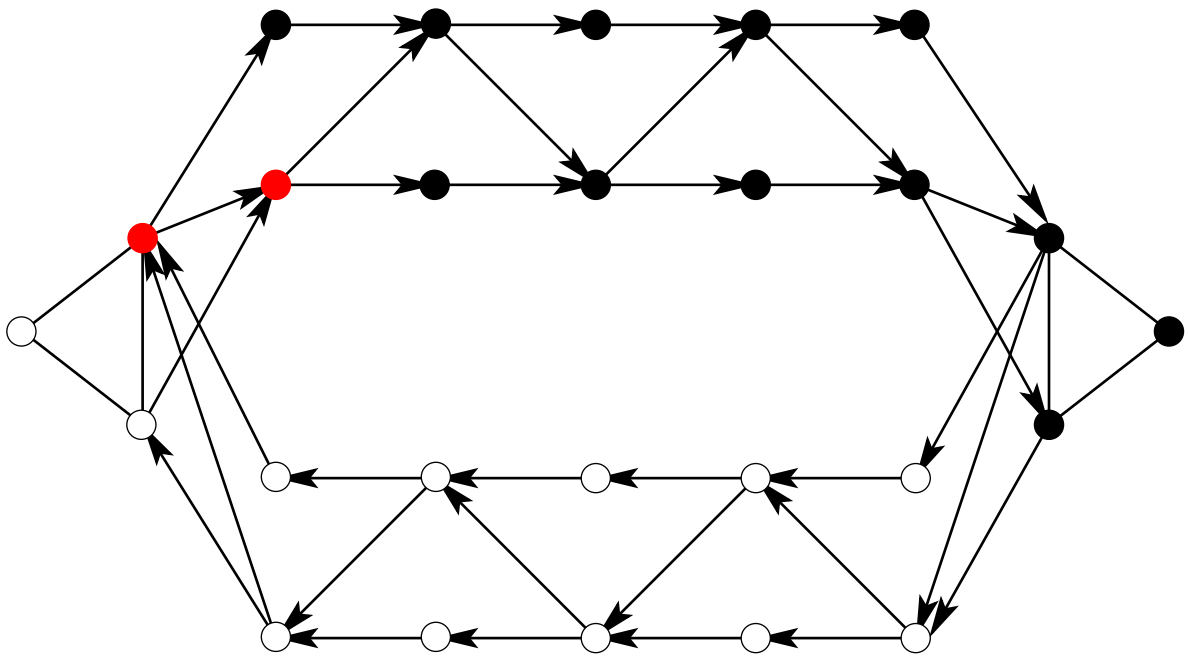
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



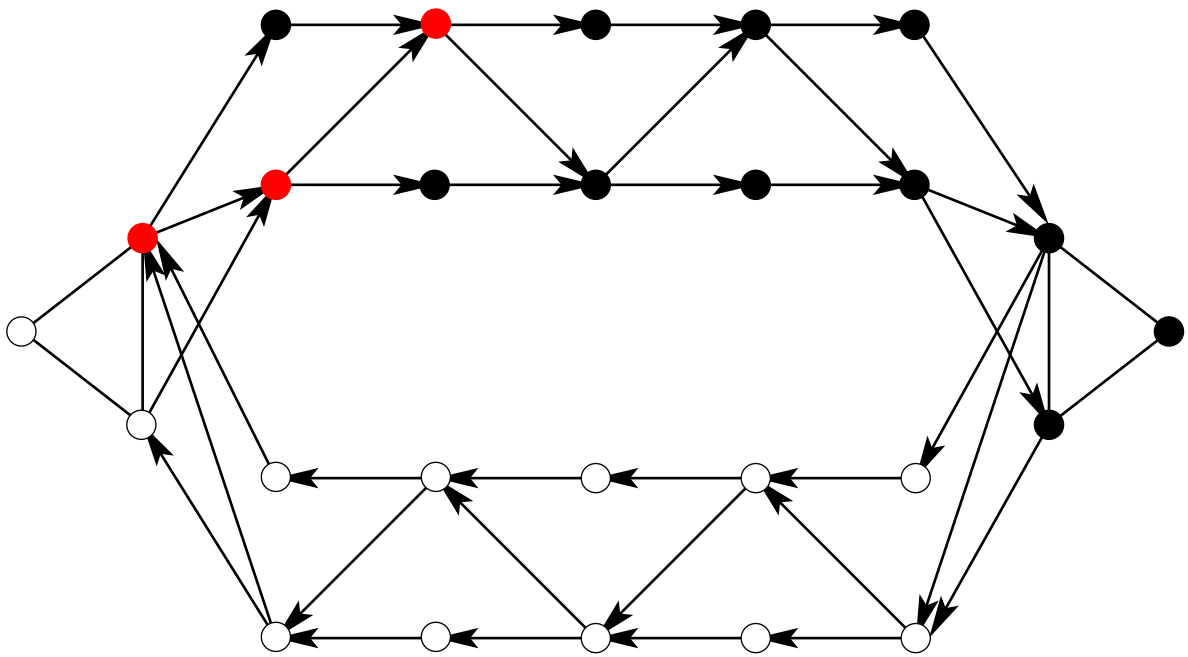
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



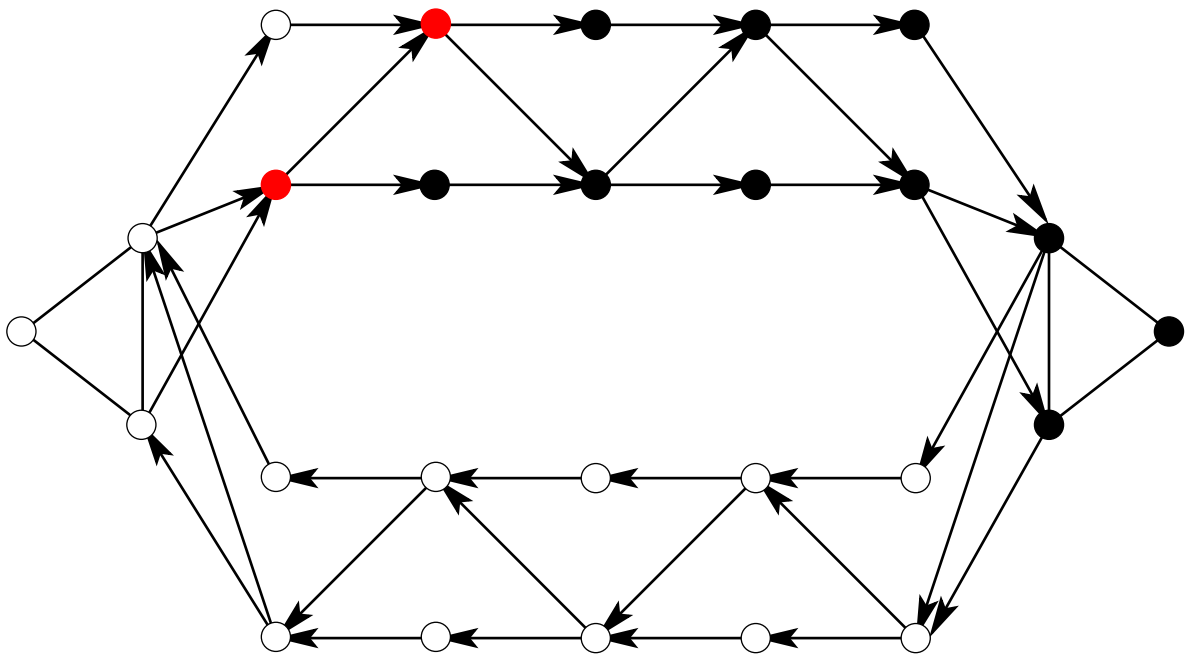
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



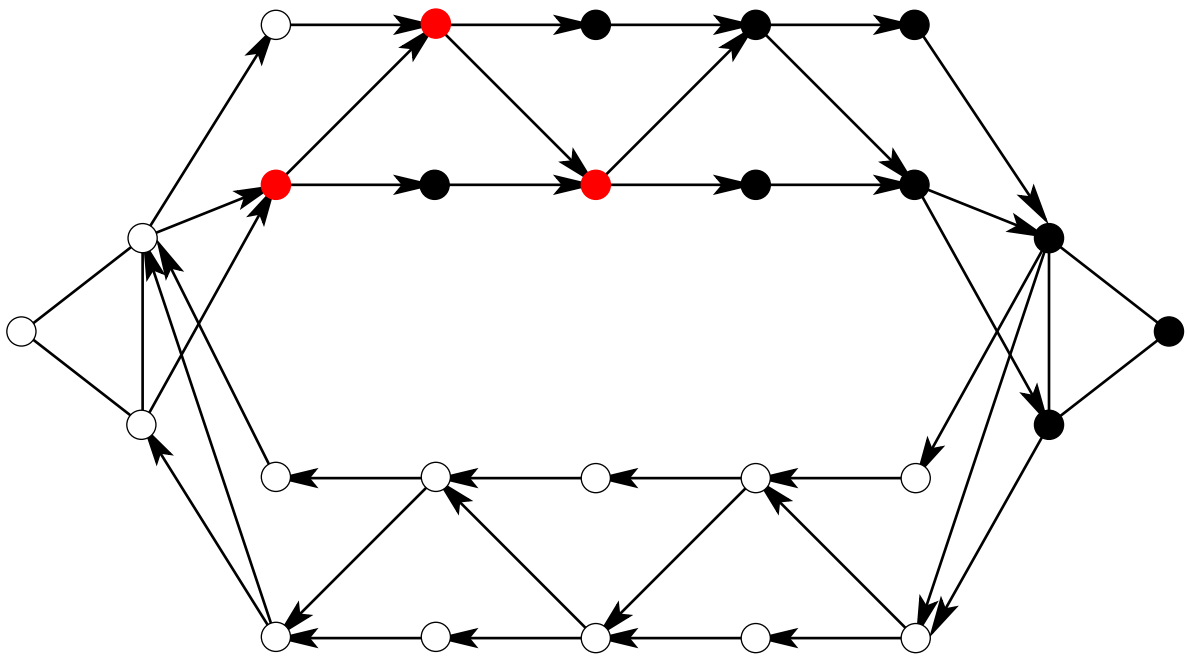
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)} \text{ is not bounded.}$$



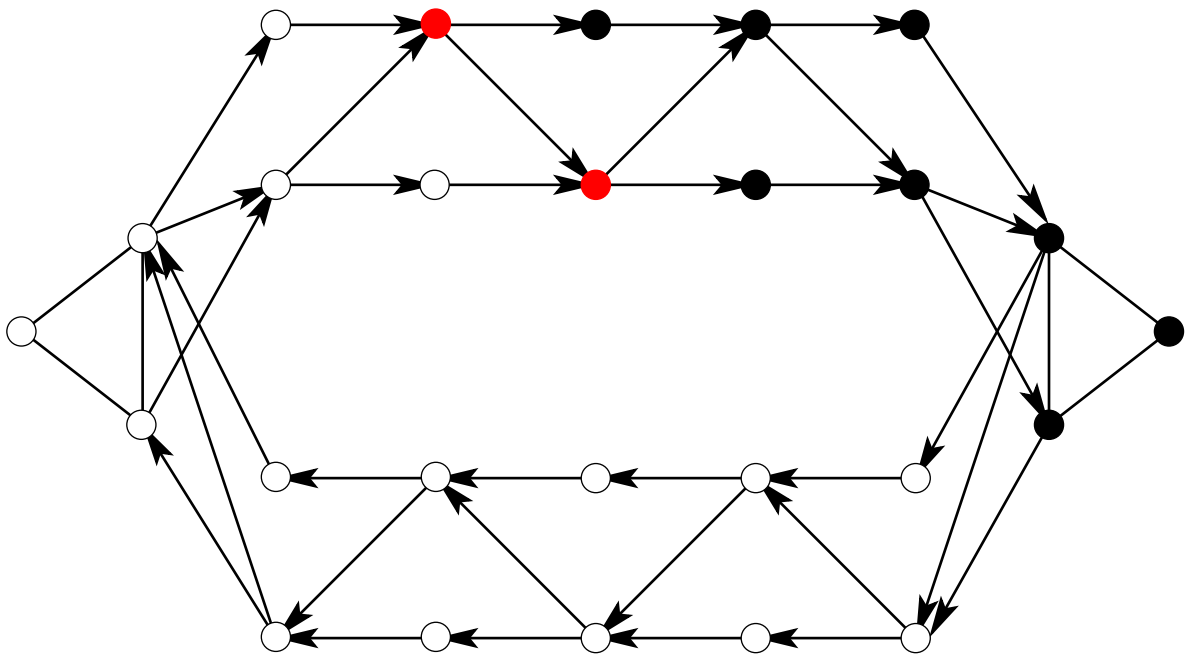
$$mfvs(D) = 4$$

$$pn(D) = 3$$

$$mfvs_{pn}(D) = n + 4$$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



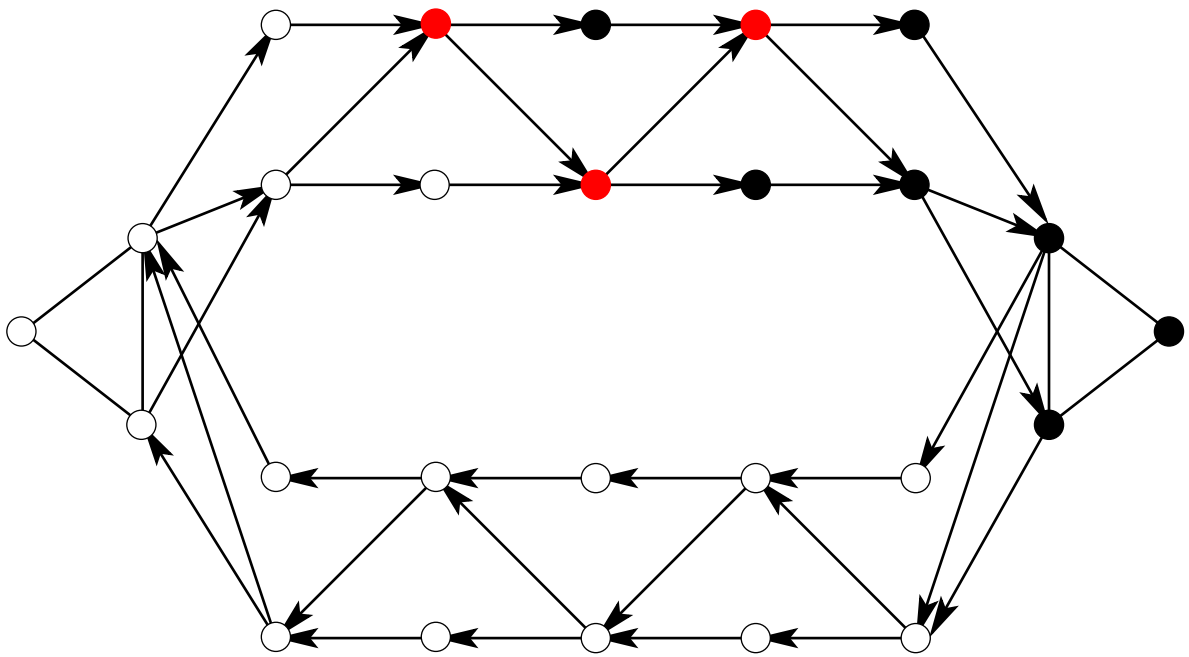
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)} \text{ is not bounded.}$$



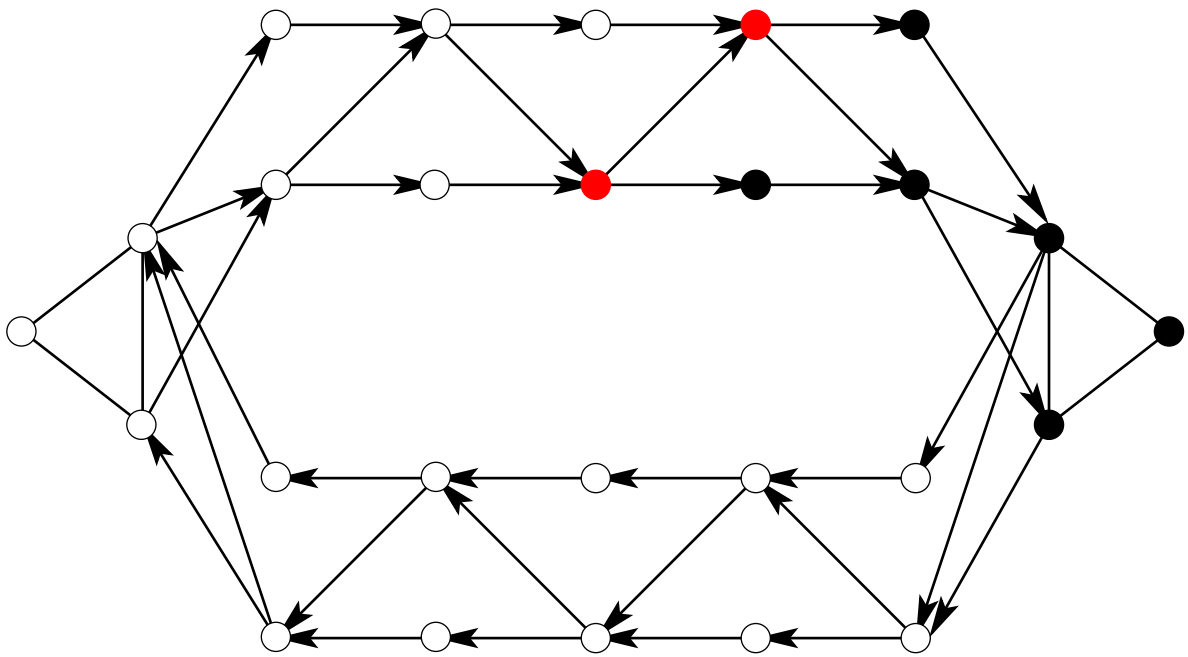
$$mfvs(D) = 4$$

$$pn(D) = 3$$

$$mfvs_{pn}(D) = n + 4$$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



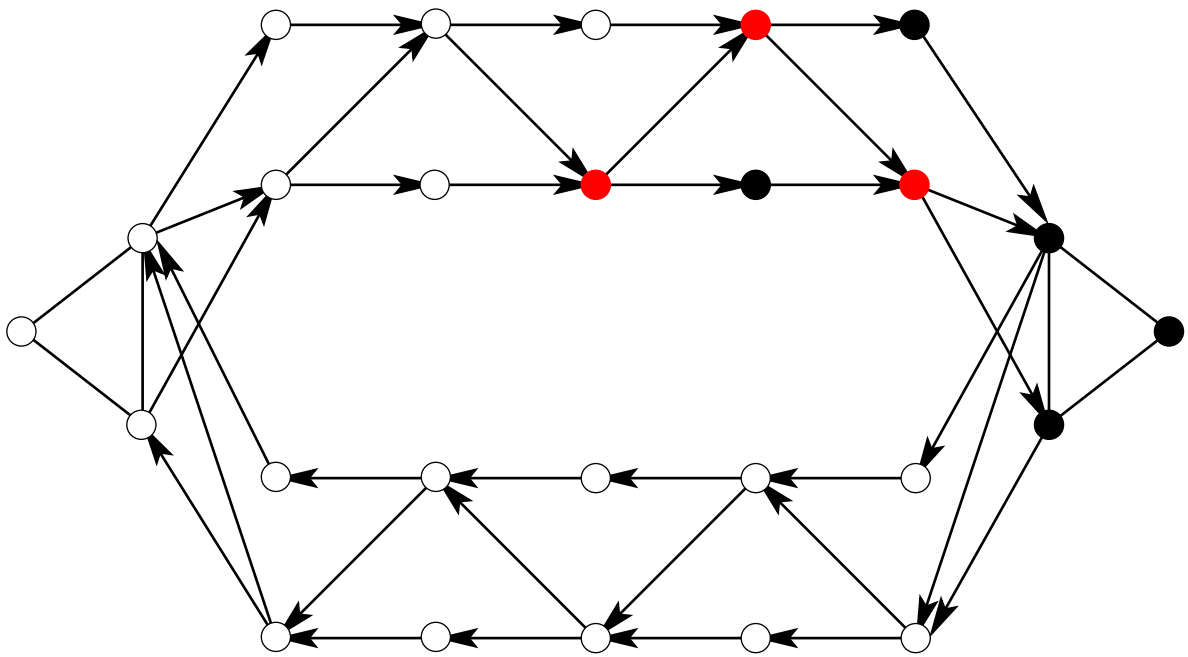
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)} \text{ is not bounded.}$$



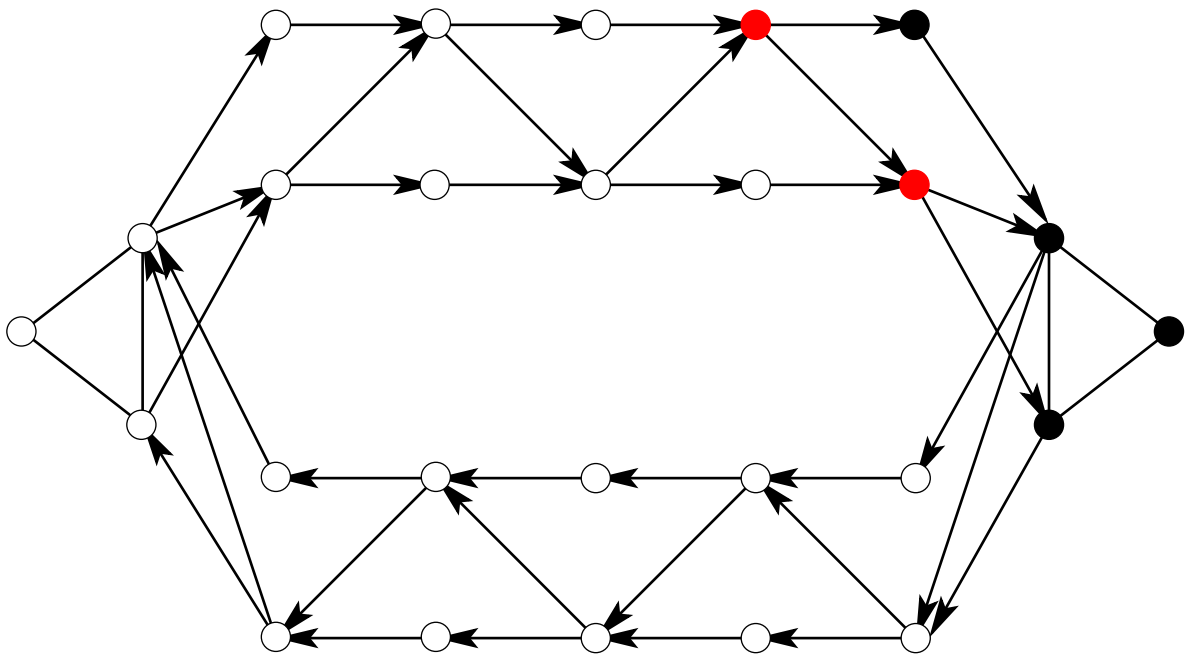
$$mfvs(D) = 4$$

$$pn(D) = 3$$

$$mfvs_{pn}(D) = n + 4$$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



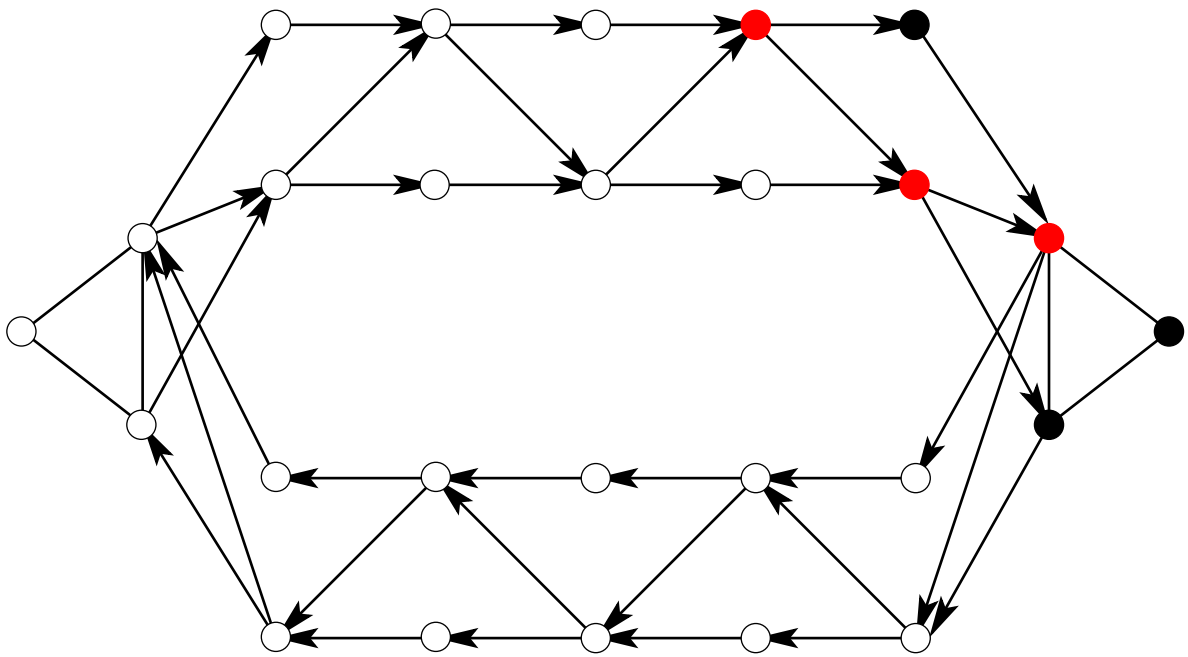
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



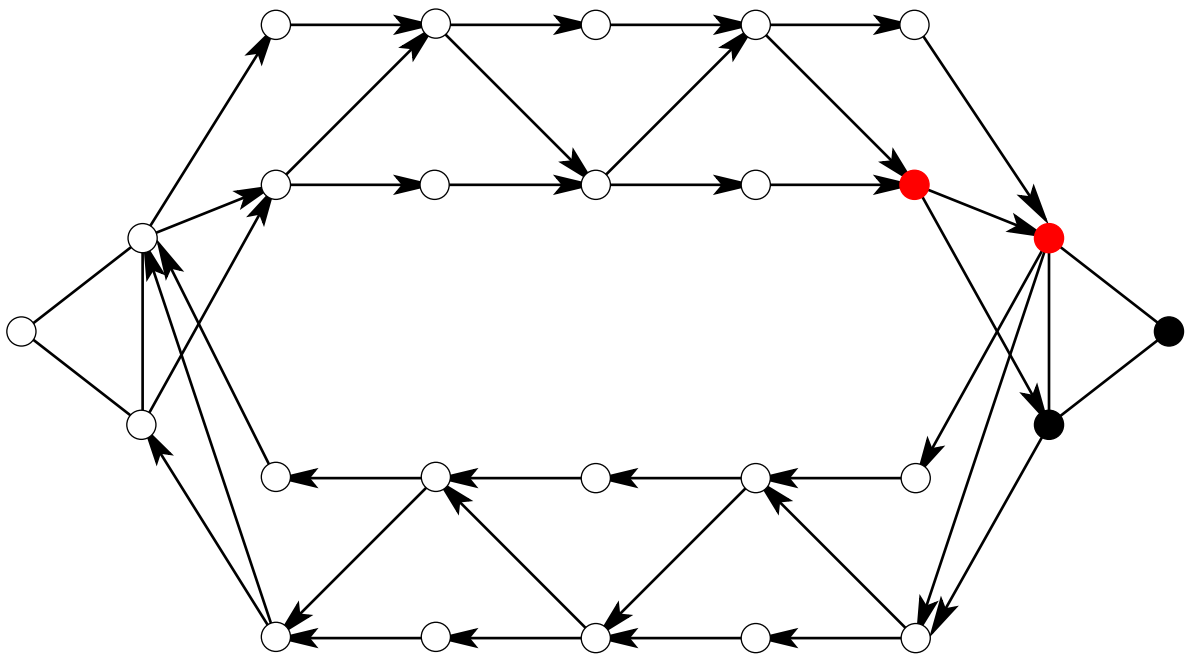
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)} \text{ is not bounded.}$$



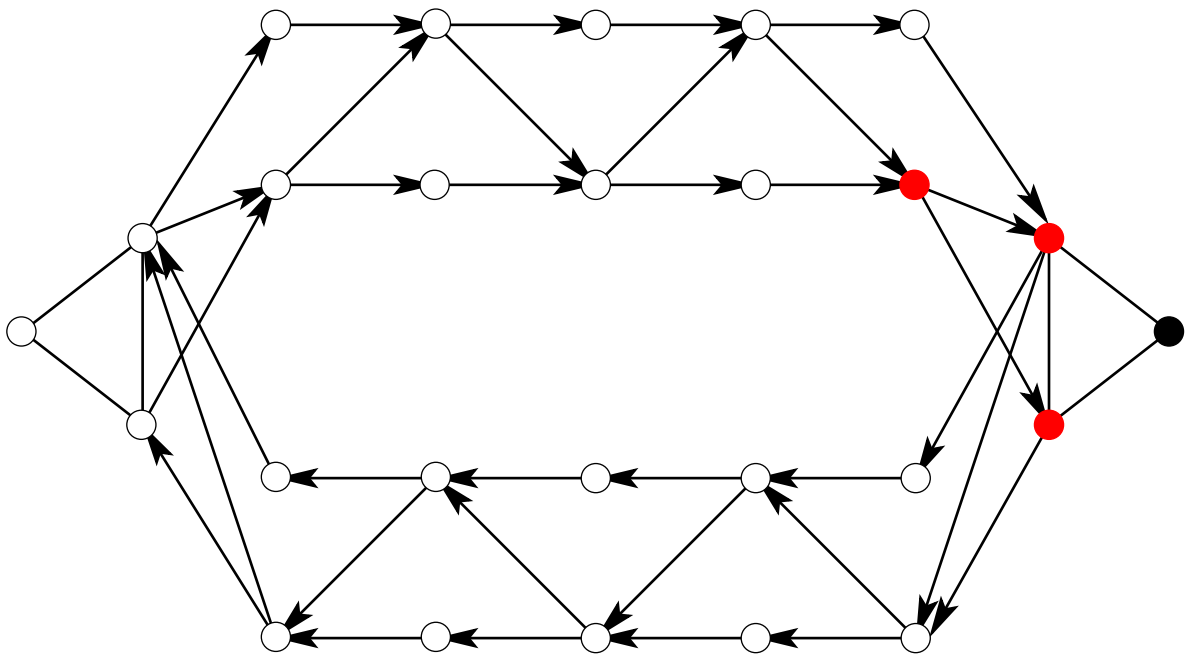
$$mfvs(D) = 4$$

$$pn(D) = 3$$

$$mfvs_{pn}(D) = n + 4$$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



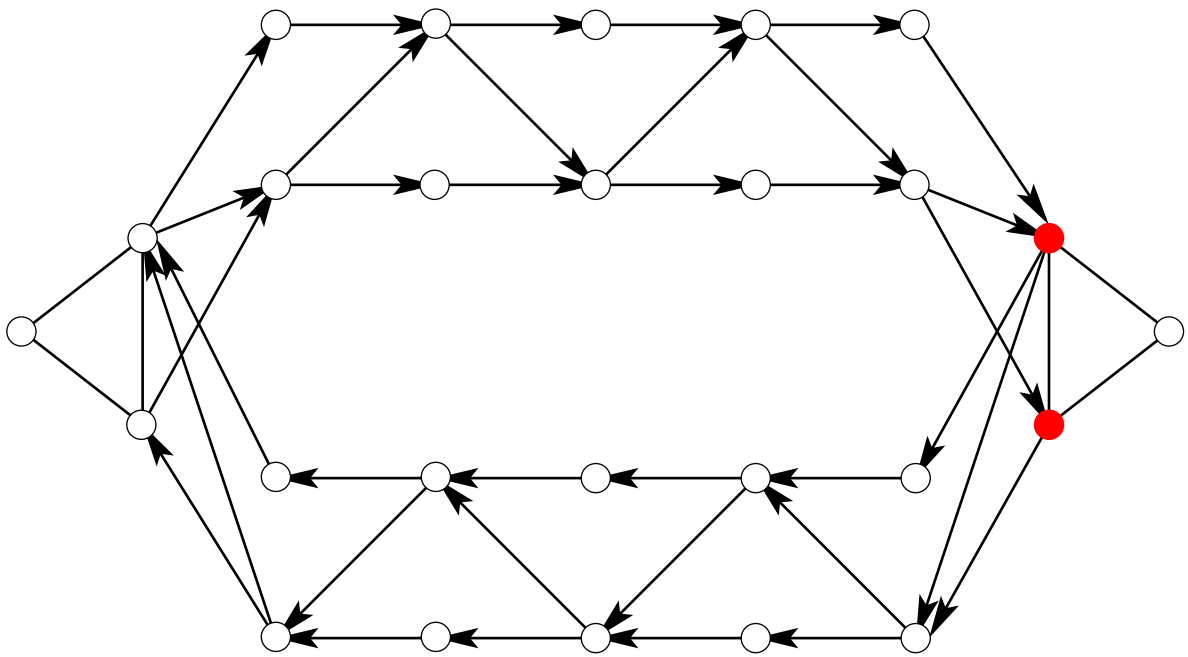
$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Digraphs with Arbitrary Large Ratios

$$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)} \text{ is not bounded.}$$



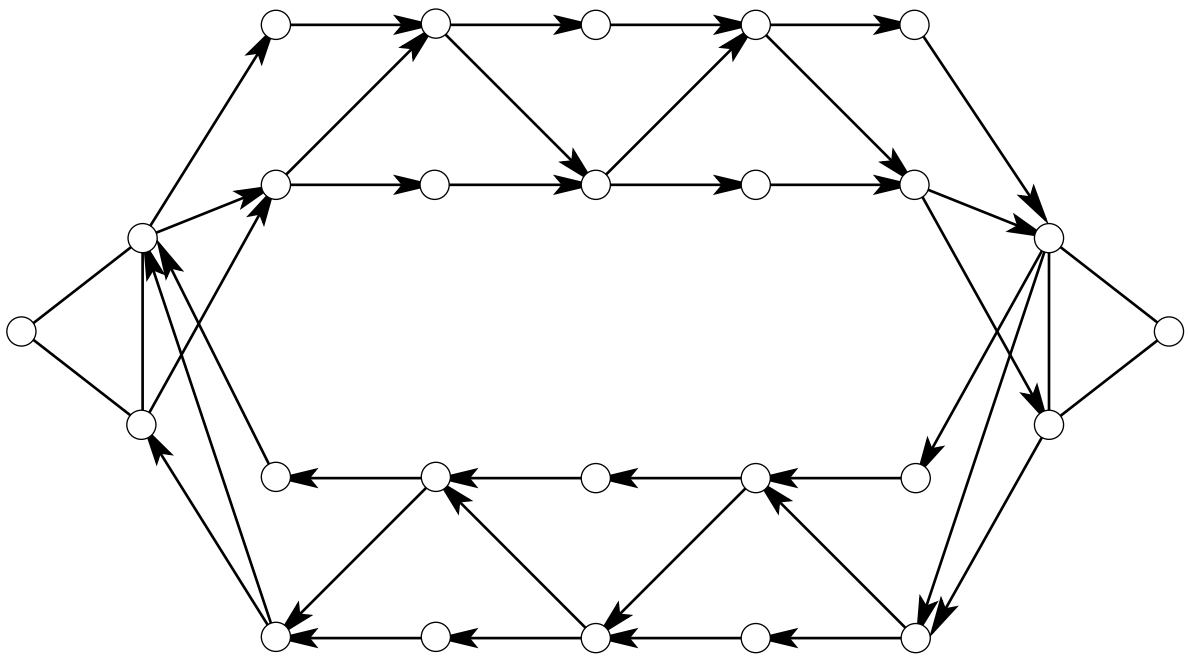
$$mfvs(D) = 4$$

$$pn(D) = 3$$

$$mfvs_{pn}(D) = n + 4$$

Digraphs with Arbitrary Large Ratios

$\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ is not bounded.



$mfvs(D) = 4$

$pn(D) = 3$

$mfvs_{pn}(D) = n + 4$

Some open questions

- Maximum value of $\lambda = \frac{mfvs_{pn}(D)}{mfvs(D)}$ for symmetric digraphs?

We have a graph with $\lambda > 3 - \varepsilon$ and we proved that $\lambda \leq pn(D)$.

- Approximation and Heuristic algorithms for these parameters

Merci