Reconfiguration of the routing in WDM networks with two classes of services

D. Coudert[1], F. Huc[1,2], D. Mazauric[1], N. Nisse[1] and J-S. Sereni[3,4]

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

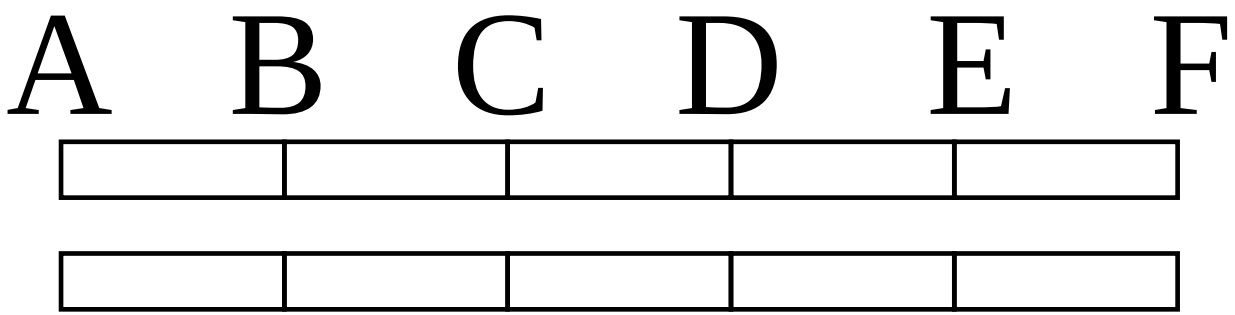
2- TCS-sensor lab, Centre Universitaire d'Informatique, Univ. Genève, Suisse

3- LIAFA, CNRS, Univ. D. Diderot, Paris, France

4- KAM, Faculty of Math. and Physics, Charles Univ., Prague, Czech Republic

WDM networks with dynamic traffic

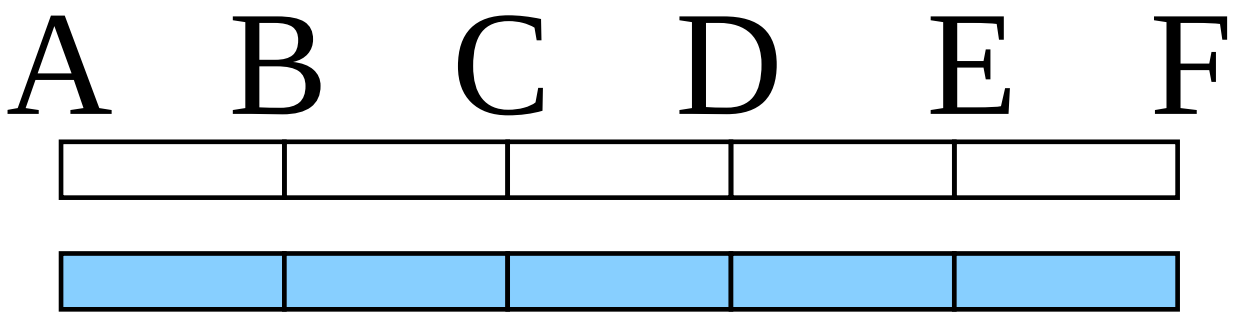
How to handle traffic changes ?



+ $A \rightarrow F$
+ $A \rightarrow C$
+ $E \rightarrow F$
− $A \rightarrow F$
+ $D \rightarrow F$
+ $A \rightarrow B$
+ $B \rightarrow E$

WDM networks with dynamic traffic

How to handle traffic changes ?



$+ \; A \rightarrow F$

$+ \; A \rightarrow C$

$+ \; E \rightarrow F$

$- \; A \rightarrow F$

$+ \; D \rightarrow F$

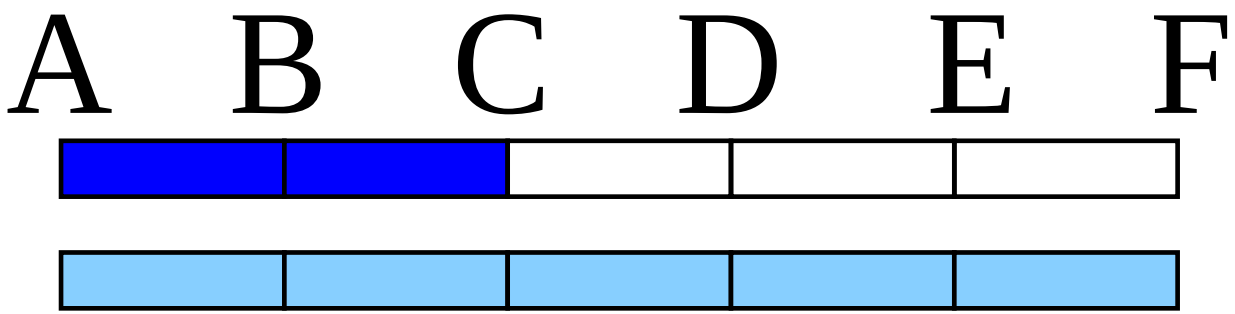
$+ \; A \rightarrow B$

$+ \; B \rightarrow E$

Routing of request: $A \rightarrow F$

WDM networks with dynamic traffic

How to handle traffic changes ?

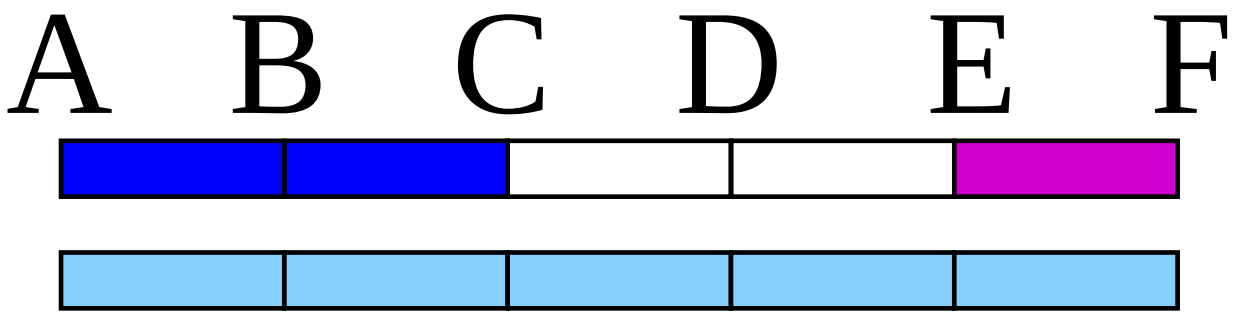


$+ \; A \rightarrow F$
$+ \; A \rightarrow C$
$+ \; E \rightarrow F$
$- \; A \rightarrow F$
$+ \; D \rightarrow F$
$+ \; A \rightarrow B$
$+ \; B \rightarrow E$

Routing of request: $A \rightarrow C$

WDM networks with dynamic traffic

How to handle traffic changes ?

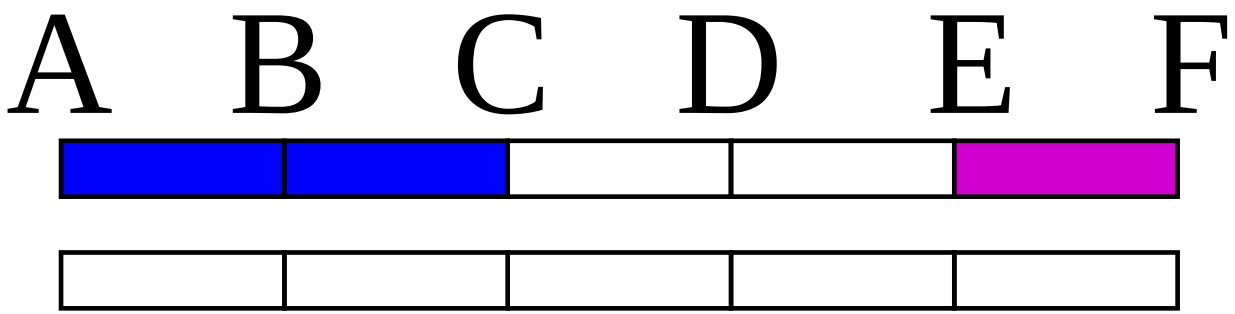


$+ \quad A \rightarrow F$
$+ \quad A \rightarrow C$
$+ \quad E \rightarrow F$
$- \quad A \rightarrow F$
$+ \quad D \rightarrow F$
$+ \quad A \rightarrow B$
$+ \quad B \rightarrow E$

Routing of request: $E \rightarrow F$

WDM networks with dynamic traffic

How to handle traffic changes ?

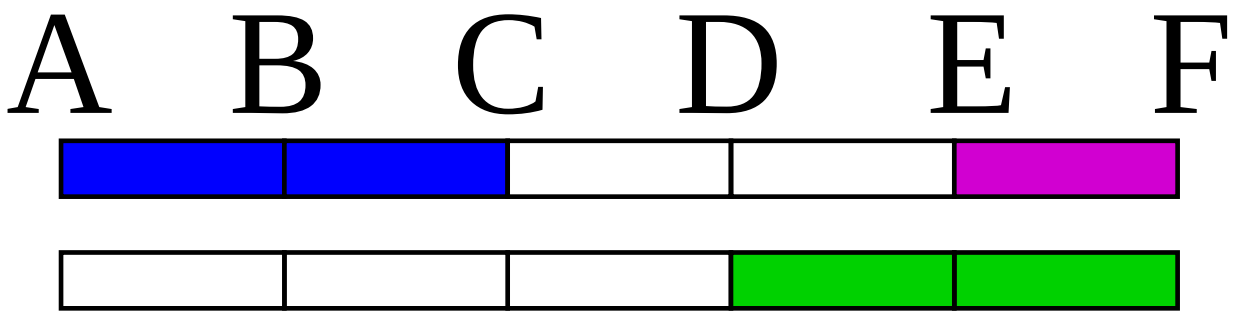


$+ \quad A \rightarrow F$
$+ \quad A \rightarrow C$
$+ \quad E \rightarrow F$
$- \quad A \rightarrow F$
$+ \quad D \rightarrow F$
$+ \quad A \rightarrow B$
$+ \quad B \rightarrow E$

Removal of request: $A \rightarrow F$

WDM networks with dynamic traffic

How to handle traffic changes ?

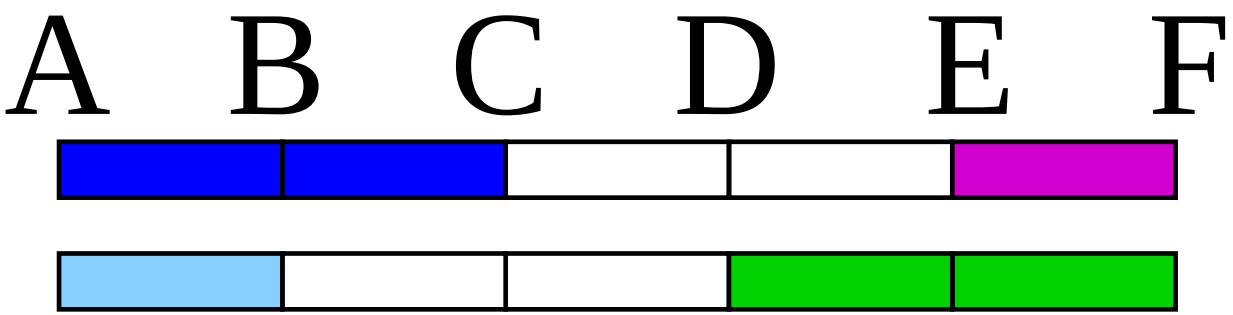


$+ \quad A \rightarrow F$
$+ \quad A \rightarrow C$
$+ \quad E \rightarrow F$
$- \quad A \rightarrow F$
$+ \quad D \rightarrow F$
$+ \quad A \rightarrow B$
$+ \quad B \rightarrow E$

Routing of request: $D \rightarrow F$

WDM networks with dynamic traffic

How to handle traffic changes ?

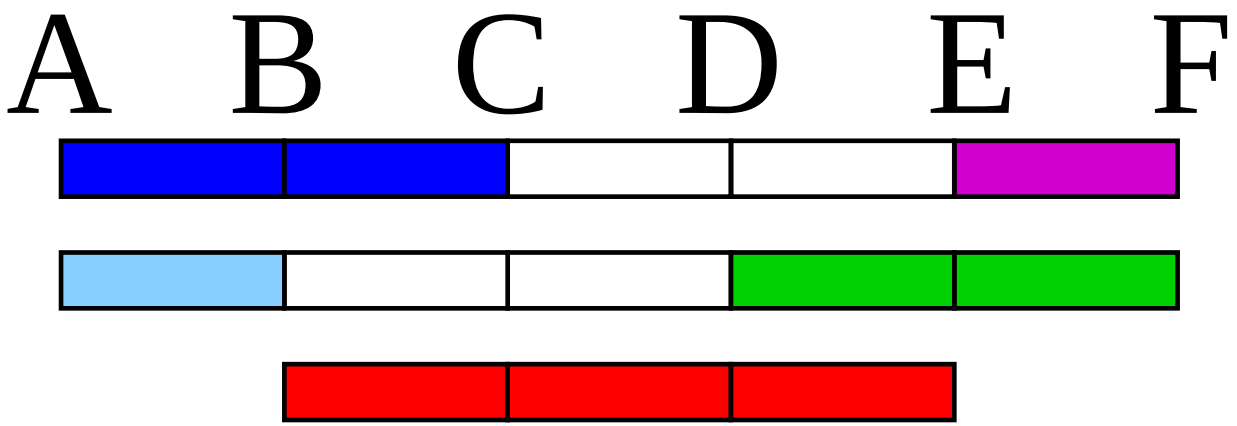


$+ \quad A \rightarrow F$
$+ \quad A \rightarrow C$
$+ \quad E \rightarrow F$
$- \quad A \rightarrow F$
$+ \quad D \rightarrow F$
$+ \quad A \rightarrow B$
$+ \quad B \rightarrow E$

Routing of request: $A \rightarrow B$

WDM networks with dynamic traffic

How to handle traffic changes ?

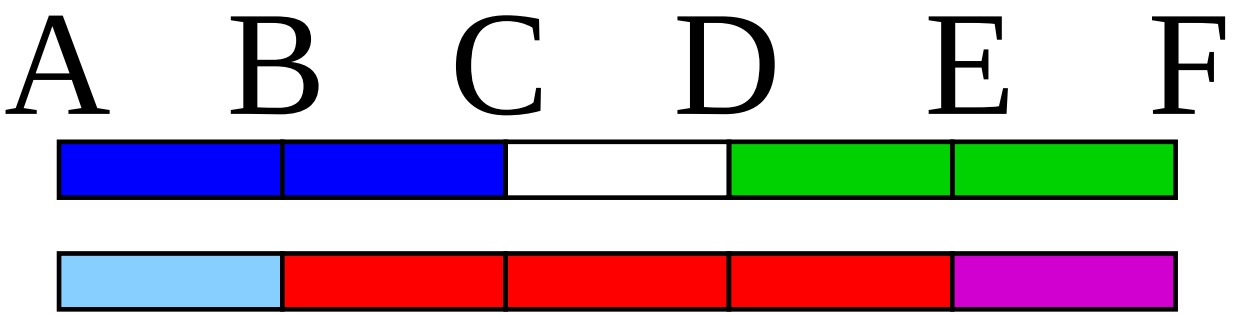


$+ \quad A \rightarrow F$
$+ \quad A \rightarrow C$
$+ \quad E \rightarrow F$
$- \quad A \rightarrow F$
$+ \quad D \rightarrow F$
$+ \quad A \rightarrow B$
$+ \quad B \rightarrow E$

Routing of request: $B \rightarrow E$? ?

WDM networks with dynamic traffic

How to handle traffic changes ?



$$
\begin{aligned}
+ \quad & A \rightarrow F \\
+ \quad & A \rightarrow C \\
+ \quad & E \rightarrow F \\
- \quad & A \rightarrow F \\
+ \quad & D \rightarrow F \\
+ \quad & A \rightarrow B \\
+ \quad & B \rightarrow E
\end{aligned}
$$

What can we do ?

- Reject the new request $\longrightarrow$ *blocking probabilities*
- Stop all requests and restart with new routing

  ...
- Find the most suitable route for incoming request with eventual rerouting of pre-established connections

Our problem:

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

Output: Scheduling for switching connection requests from current to new routes.

What can we do ?

- Reject the new request $\longrightarrow$ *blocking probabilities*
- Stop all requests and restart with new routing

  ...
- Find the most suitable route for incoming request with eventual rerouting of pre-established connections

Our problem:

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

Output: Scheduling for switching connection requests from current to new routes.

What can we do ?

- Reject the new request $\longrightarrow$ *blocking probabilities*
- Stop all requests and restart with new routing

  ...
- Find the most suitable route for incoming request with eventual rerouting of pre-established connections

Our problem:

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

Output: Scheduling for switching connection requests from current to new routes.

GMPLS

Make-before-break:

Establish new path before switching the connection

$\Longrightarrow$ Destination resources must be available

Break-before-make:

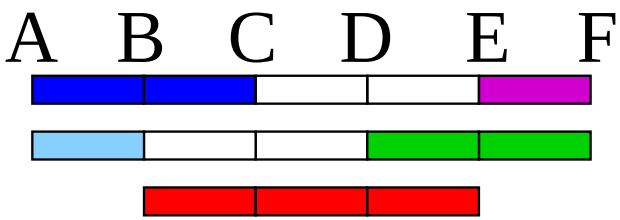
Break connection before establishing the new path

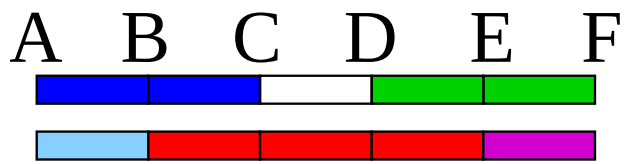
$\Longrightarrow$ Traffic stopped while new path not established

Reconfiguration in WDM networks

Example

Dependency digraph



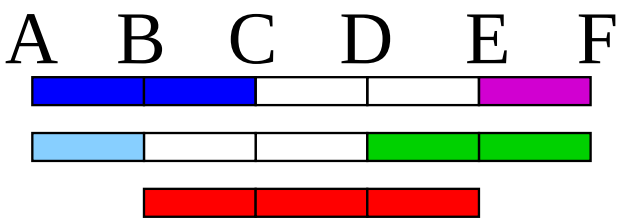


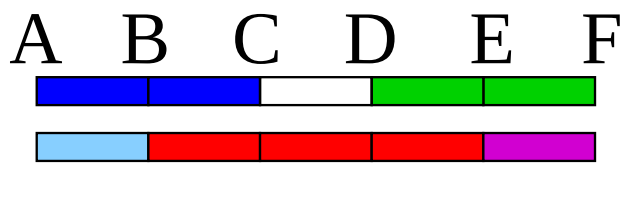
Processing using 1 break-before-make and 1 make-before-break

Reconfiguration in WDM networks

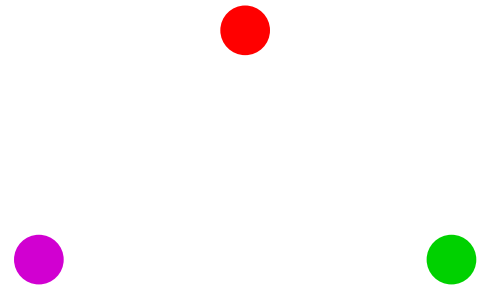
Example

Dependency digraph





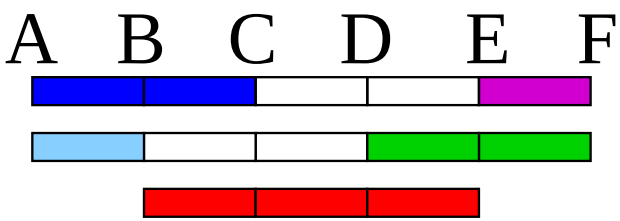
Processing using 1 break-before-make and 1 make-before-break

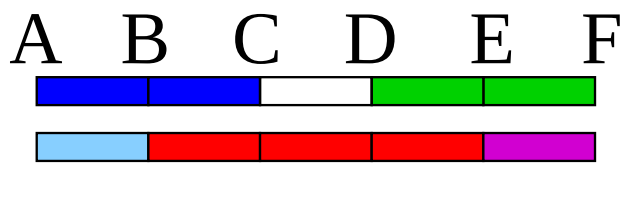


Reconfiguration in WDM networks

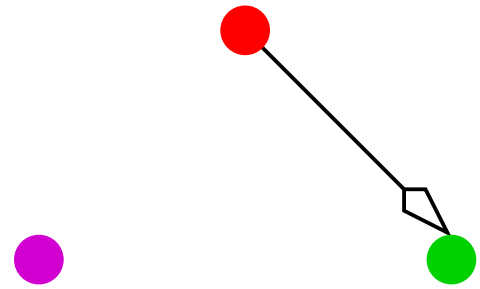
Example

Dependency digraph





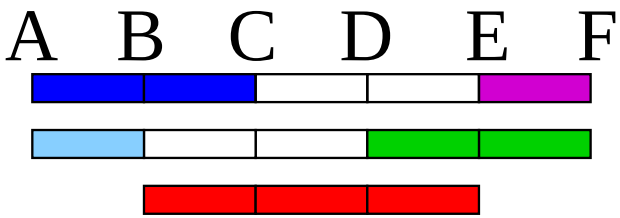
Processing using 1 break-before-make and 1 make-before-break

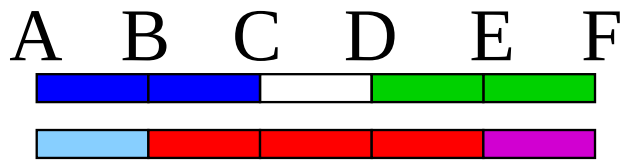


Reconfiguration in WDM networks

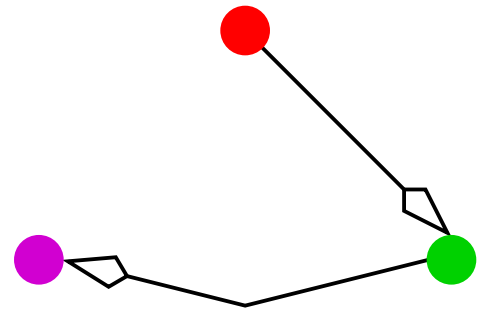
Example

Dependency digraph





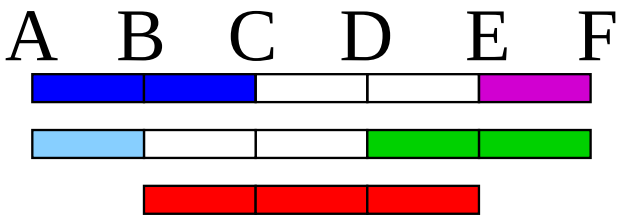
Processing using 1 break-before-make and 1 make-before-break

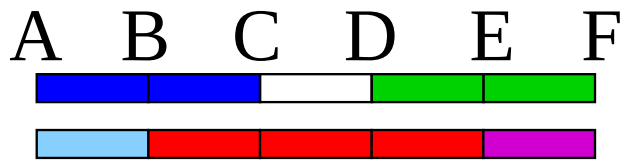


Reconfiguration in WDM networks

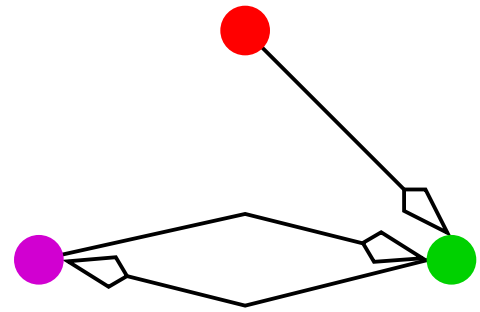
Example

Dependency digraph





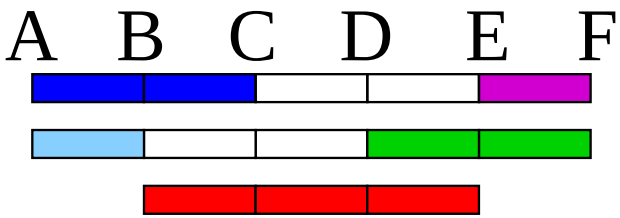
Processing using 1 break-before-make and 1 make-before-break

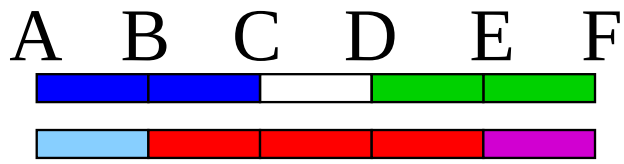


Reconfiguration in WDM networks

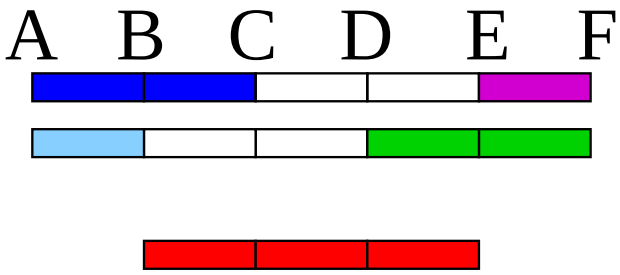
Example

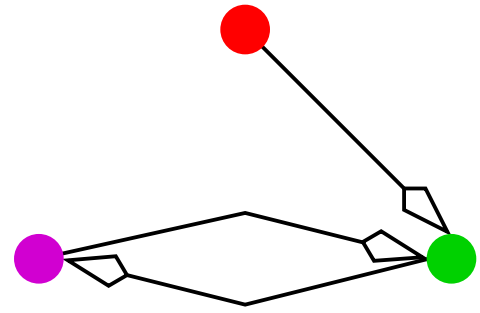
Dependency digraph





Processing using 1 break-before-make and 1 make-before-break

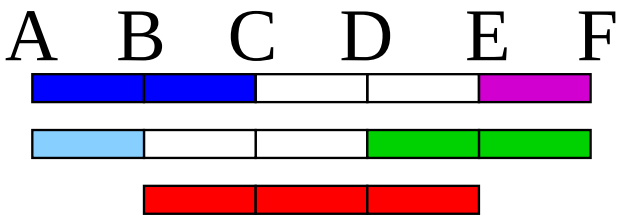


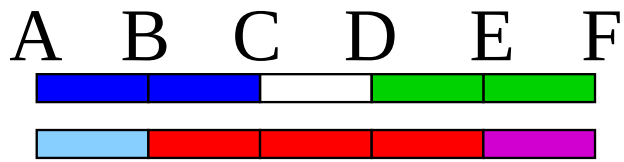


Reconfiguration in WDM networks

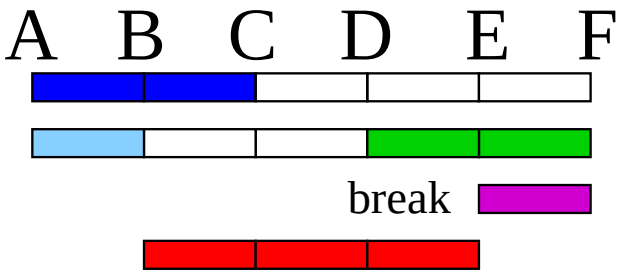
Example

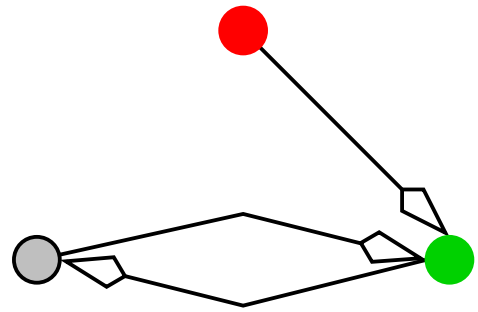
Dependency digraph





Processing using 1 break-before-make and 1 make-before-break

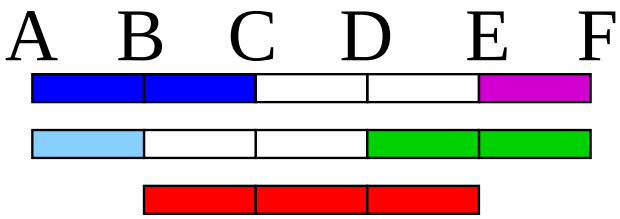


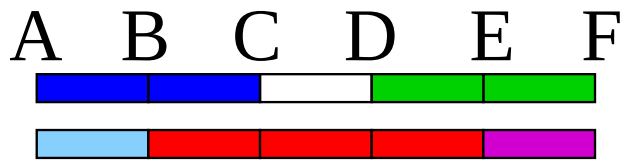


Reconfiguration in WDM networks

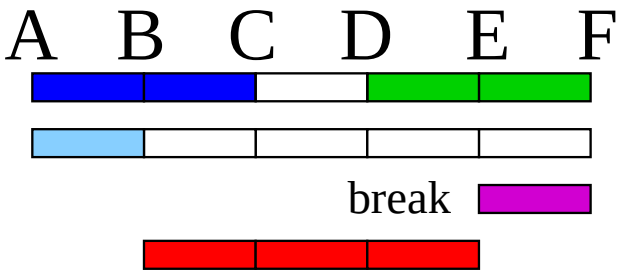
Example

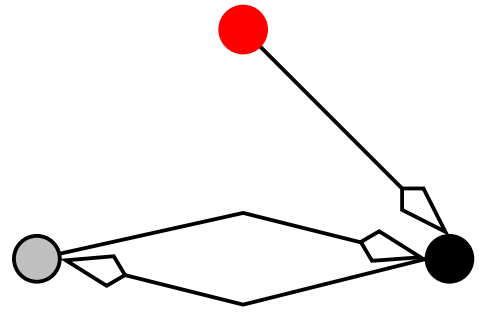
Dependency digraph





Processing using 1 break-before-make and 1 make-before-break

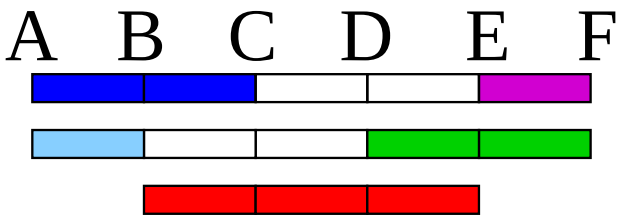


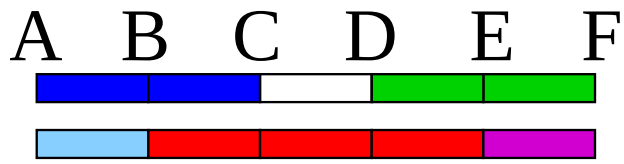


Reconfiguration in WDM networks

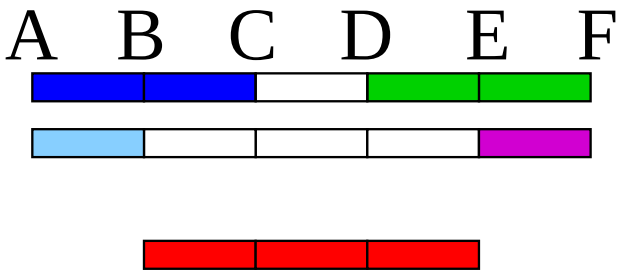
Example

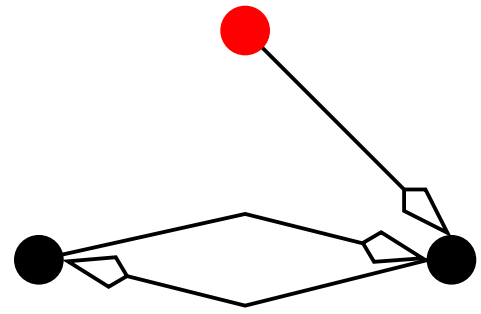
Dependency digraph





Processing using 1 break-before-make and 1 make-before-break

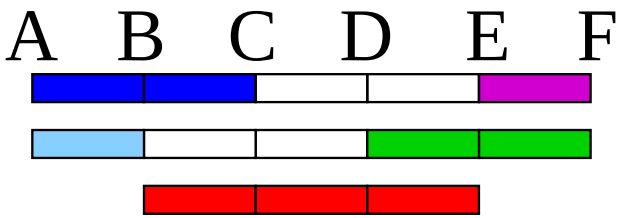


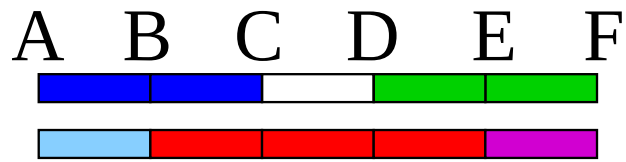


Reconfiguration in WDM networks

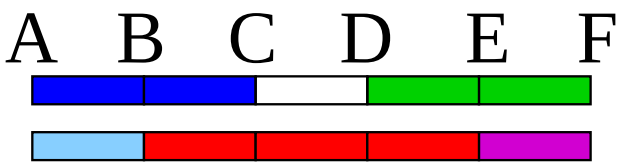
Example

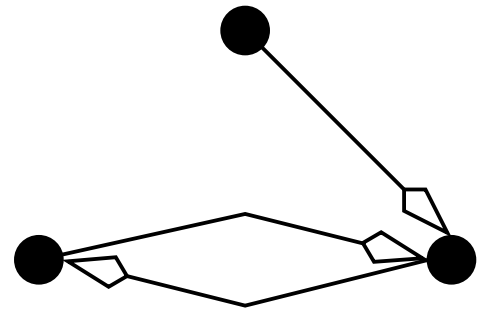
Dependency digraph





Processing using 1 break-before-make and 1 make-before-break

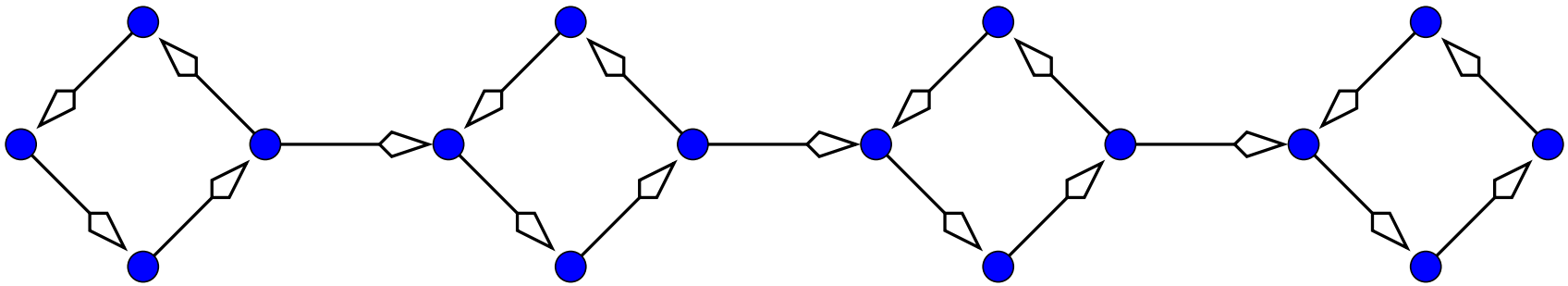




Possible objectives

Minimize overall number of break-before-make

= Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

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

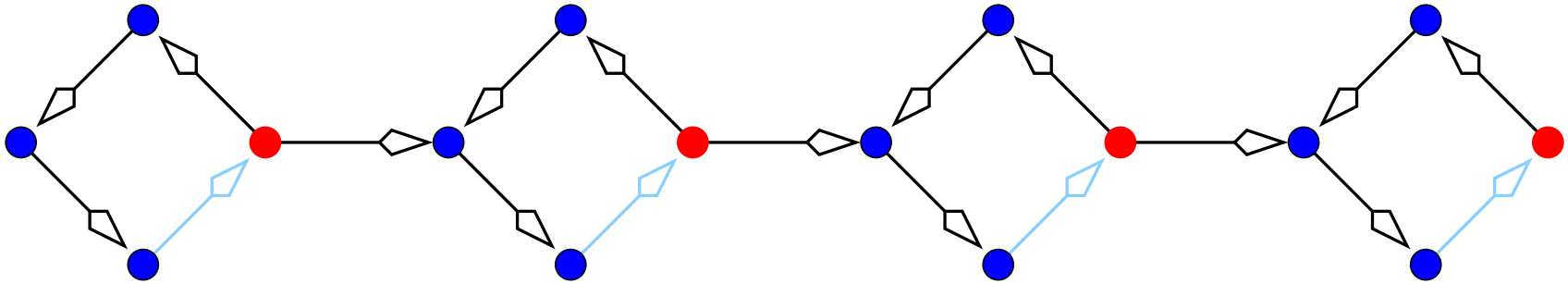
- Process number here 2

$\sim$ Can with MFVS up to $k/2$

Possible objectives

Minimize overall number of break-before-make

$=$ Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

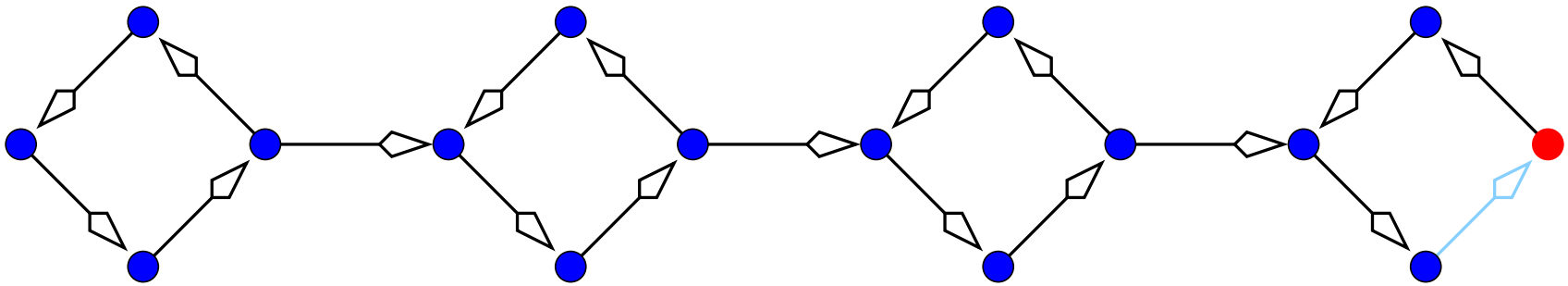
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

- Process number here 2
- $\sim$ Gain with MFVS up to $N/2$

Possible objectives

Minimize overall number of break-before-make

$=$ Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

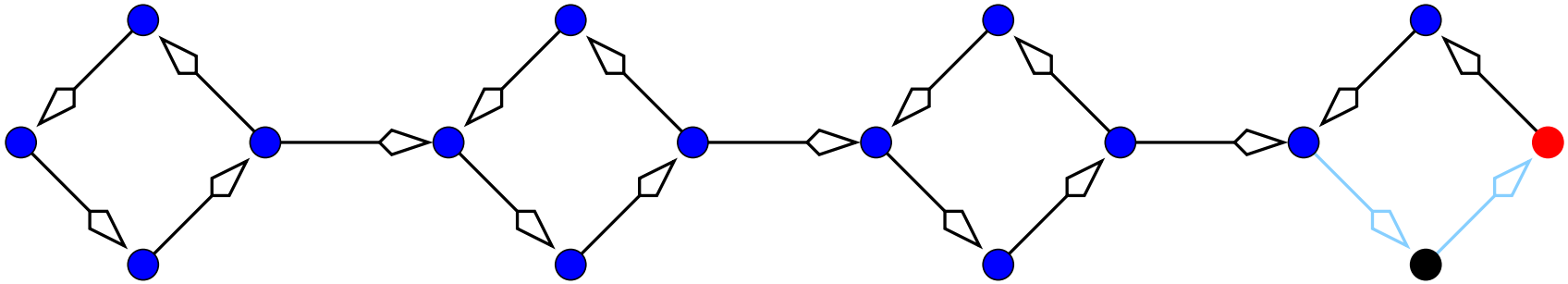
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

- Process number, here 1
- Gap with MFVS up to $N/2$

Possible objectives

Minimize overall number of break-before-make

$=$ Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

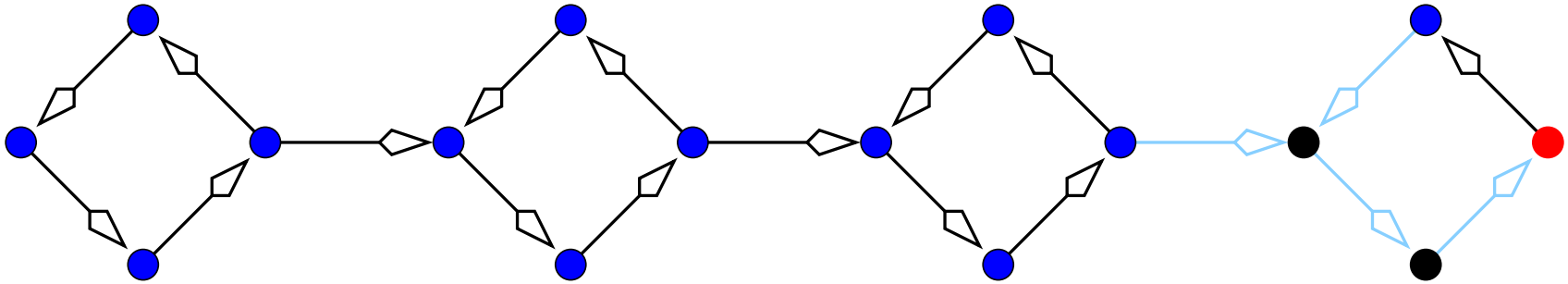
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

- Process number, here 1
- Gap with MFVS up to $N/2$

Possible objectives

Minimize overall number of break-before-make

$=$ Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

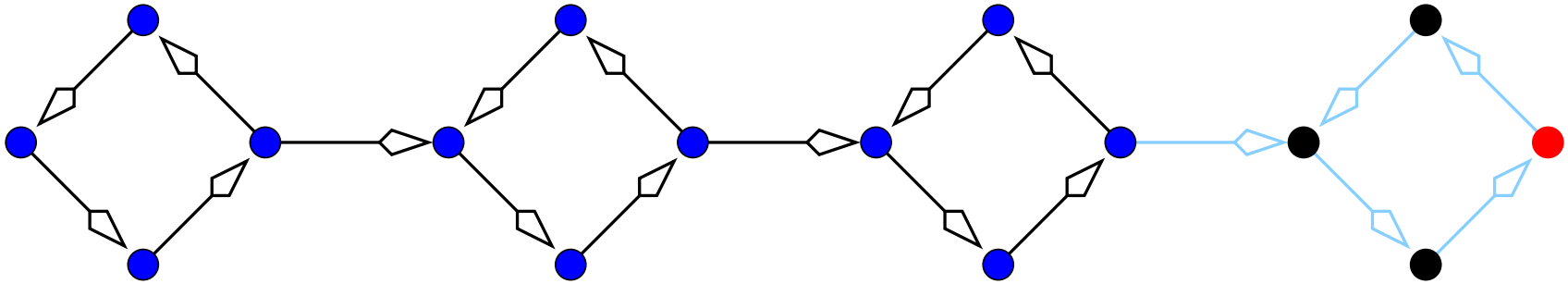
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

- Process number, here 1
- Gap with MFVS up to $N/2$

Possible objectives

Minimize overall number of break-before-make

$=$ Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

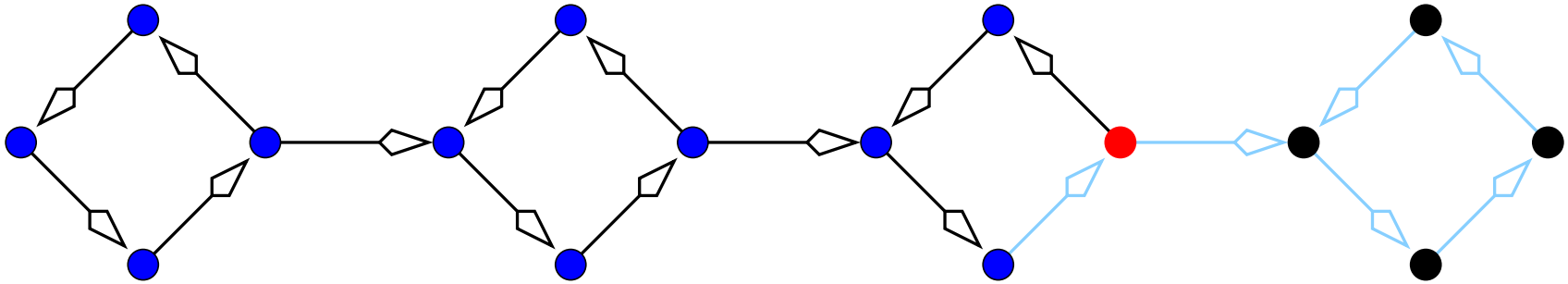
$\sim$ Graph searching problem, cops-and-robber game, pursuit,...

- Process number, here 1
- Gap with MFVS up to $N/2$

Possible objectives

Minimize overall number of break-before-make

$=$ Minimum Feedback Vertex Set (MFVS), here 4



Minimize number of simultaneous break-before-make

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

- Process number, here 1
- Gap with MFVS up to $N/2$

Process number, pn

Rules

- R_1 Put an agent on a vertex
  - $=$ break 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-process strategy $=$ strategy to process a (di)graph using at most p agents

Process number $=$ smallest p s.t. G can be p-processed, $pn(G)$

Example: DAG

Rules

R_1 Put an agent on a vertex

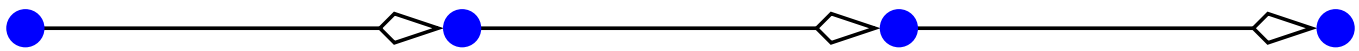
$=$ break a connection

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

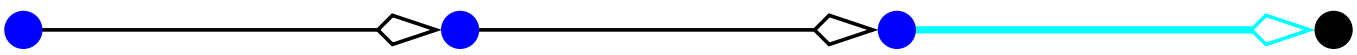
$=$ break a connection

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 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 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

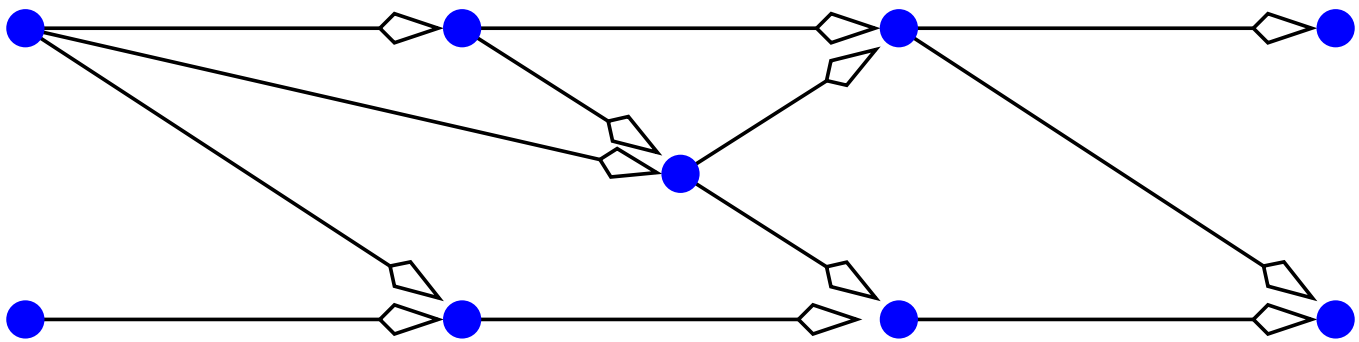
$=$ break a connection

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 $pn(D) = 0$

Digraphs with process number 1

Rules

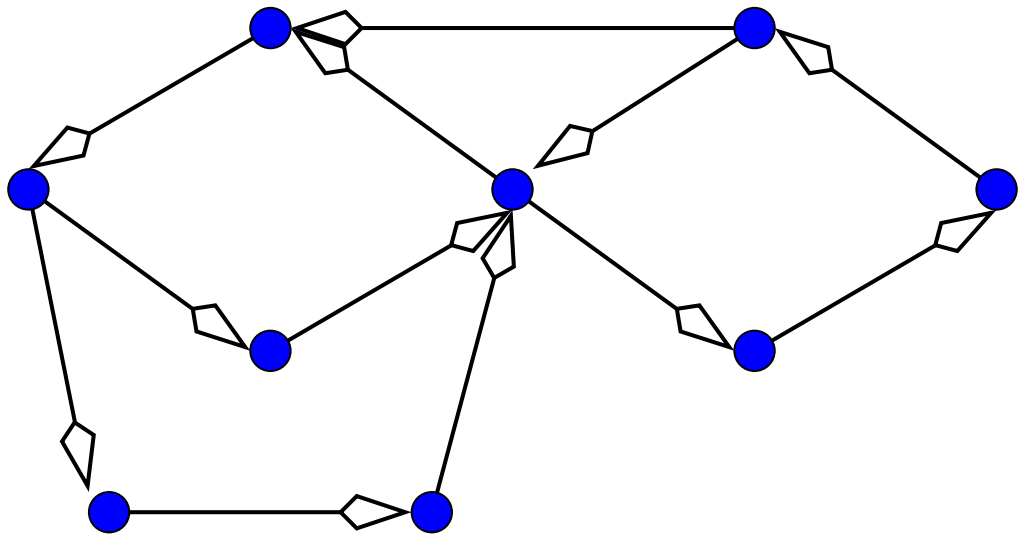
R_1 Put an agent on a vertex

$=$ break 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



Th: $pn(D) = 1 \Leftrightarrow \mathrm{MFVS}(D) = 1$ Recognition in time $O(N + M)$

Digraphs with process number 1

Rules

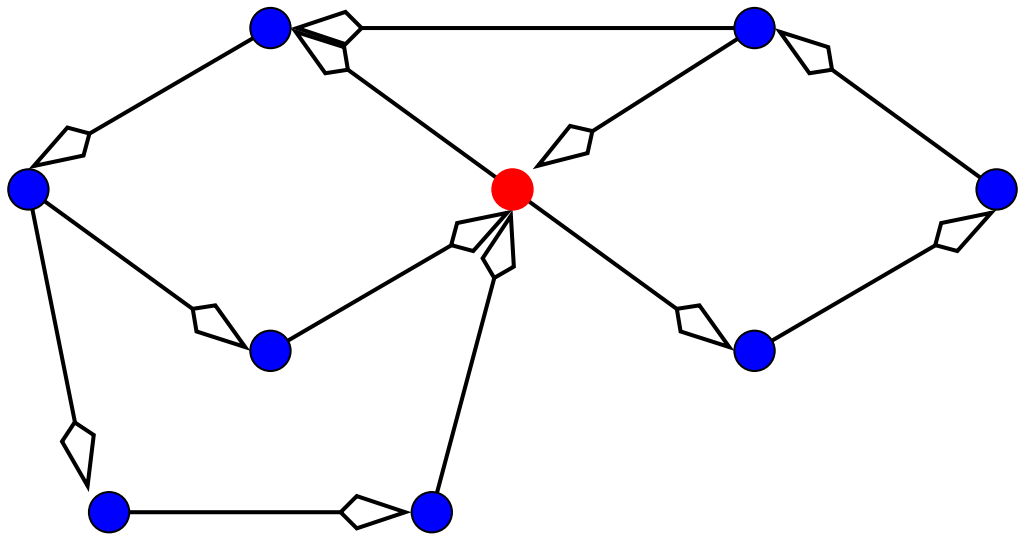
R_1 Put an agent on a vertex

$=$ break 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



Th: $pn(D) = 1 \Leftrightarrow \text{MFVS}(D) = 1$ Recognition in time $O(N + M)$

Digraphs with process number 1

Rules

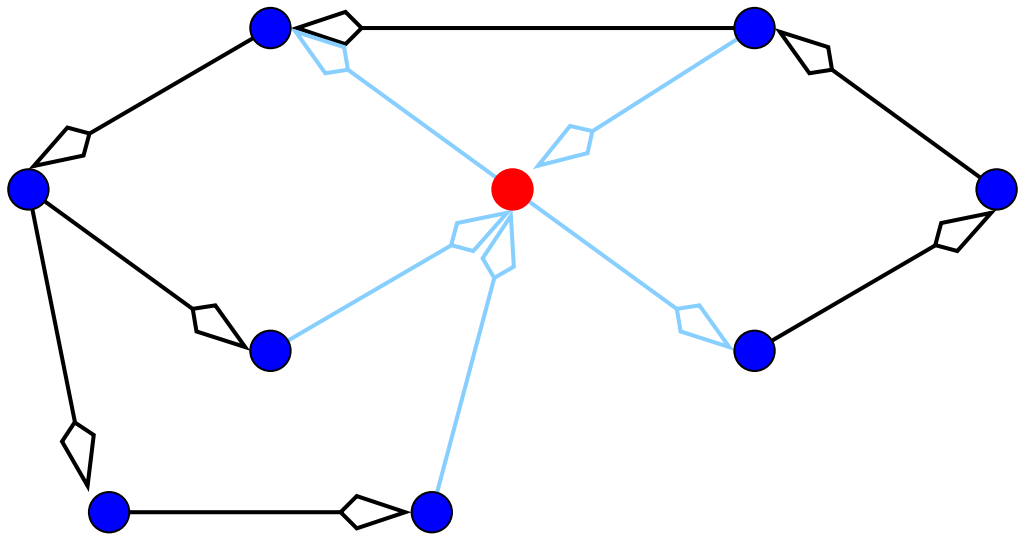
R_1 Put an agent on a vertex

$=$ break 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



Th: $pn(D) = 1 \Leftrightarrow \text{MFVS}(D) = 1$ Recognition in time $O(N + M)$

Process number: what is known

Related parameters

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

Relations

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

Complexity

- NP-Hard
- Not APX
  - = No polynomial time constant factor approximation algorithm
- Characterization of digraphs with process number 0, 1, 2

What's next

- How to handle *priority connections* (no break-before-make)
- Heuristic algorithms
  - Previous heuristic by Jose & Somani, DRCN 03
  - New heuristic using the process number
- Simulation results

Two classes of services

Priority connections

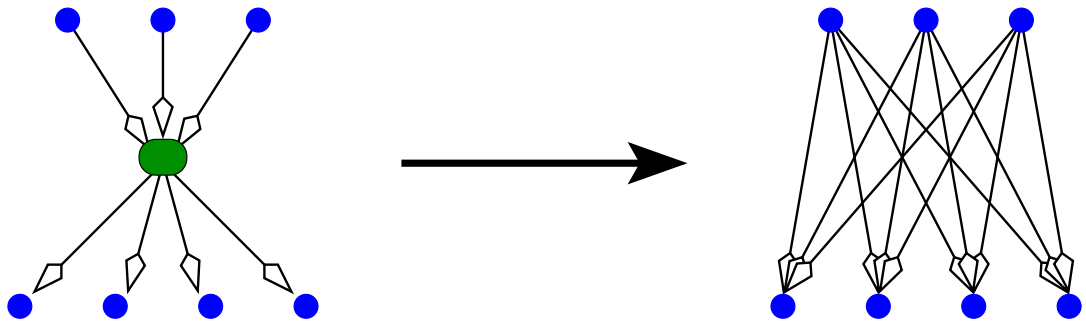
- Refuse *by contract* break-before-make

Impossiblitity

- Direct cycle of priority connections in the dependency digraph

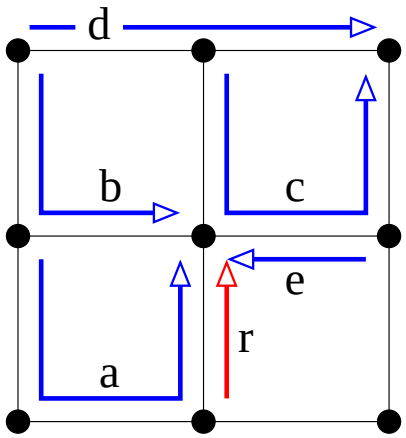
$\Rightarrow$ *Small* number of such connections

Transformation

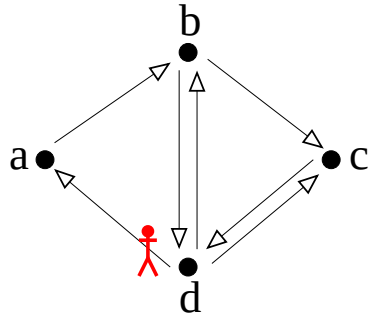


$\Rightarrow$ Same problem to solve

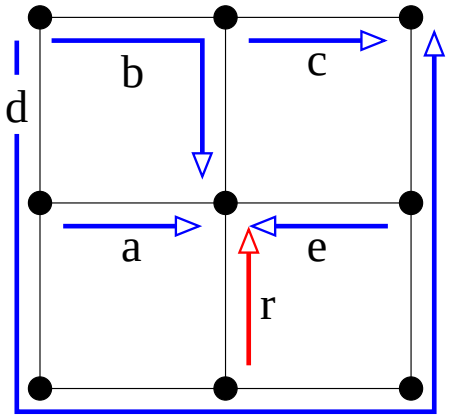
Example with priority connection *d*



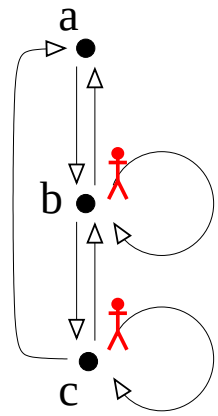
Routing 1



Dependency digraph



Routing 2



Without *d*

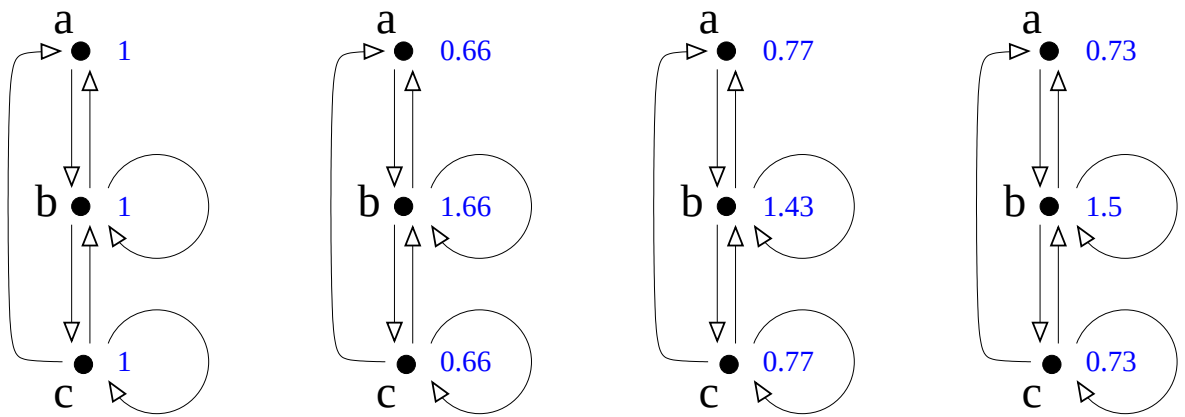
Previous heuristic [Jose and Somani, DRCN 03]

1. Compute all directed cycles using Johnson's algorithm
2. Choose the vertex that belongs to the maximum number of cycles
3. Remove that vertex and update set of cycles
4. Repeat 2-3 until remaining digraph is a DAG
5. Process DAG
6. Process removed vertices

- Heuristic for MFVS
- Complexity in $O((n+m)(c+1))$
- Exponential number of cycles $\Rightarrow$ only for small digraphs

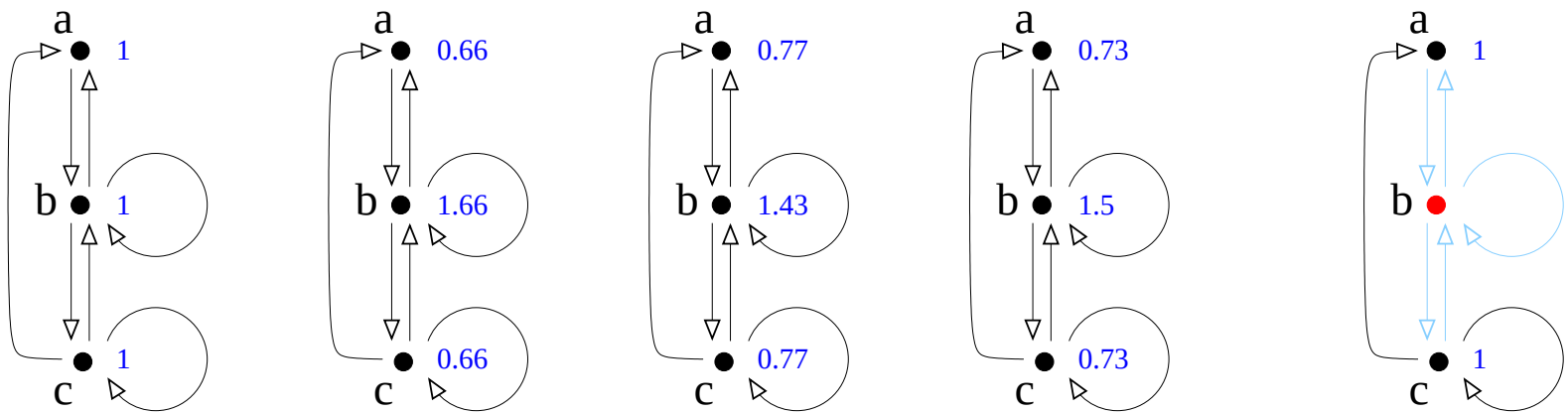
Our heuristic / process number

1. Priority connections: impossibility and transformation
2. Choose of a candidate vertex to receive an agent (to be removed) using a flow circulation method
3. Remove that vertex and process all possible vertices including removed vertices
4. Repeat 2-3 until processing of all vertices



Our heuristic / process number

1. Priority connections: impossibility and transformation
2. Choose of a candidate vertex to receive an agent (to be removed) using a flow circulation method
3. Remove that vertex and process all possible vertices including removed vertices
4. Repeat 2-3 until processing of all vertices

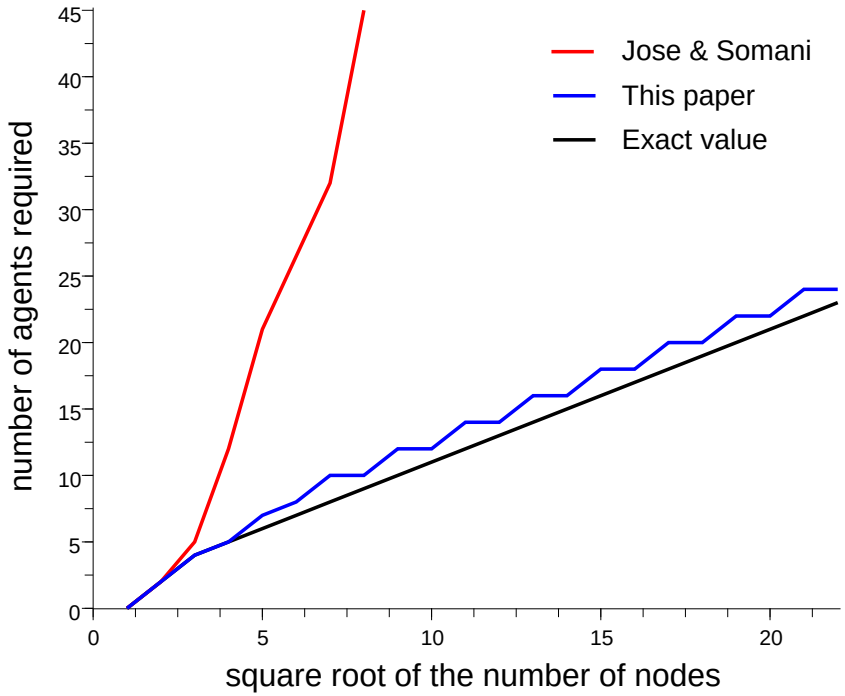


Our heuristic / process number

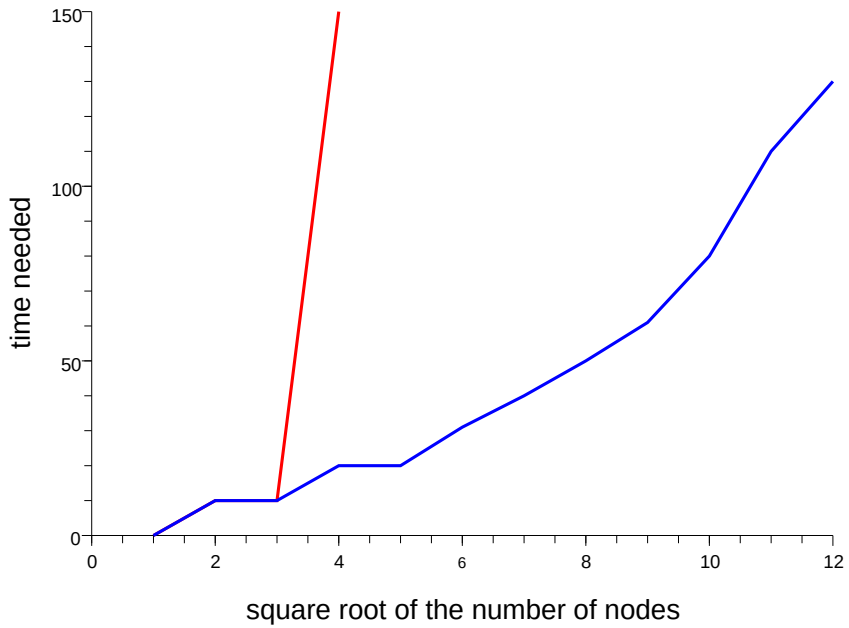
1. Priority connections: impossibility and transformation
2. Choose of a candidate vertex to receive an agent (to be removed) using a flow circulation method
3. Remove that vertex and process all possible vertices including removed vertices
4. Repeat 2-3 until processing of all vertices

- Heuristic for the process number
- Complexity in $O(n^2(n+m)) \Rightarrow$ large digraphs

Simulation results: $n \times n$ grids

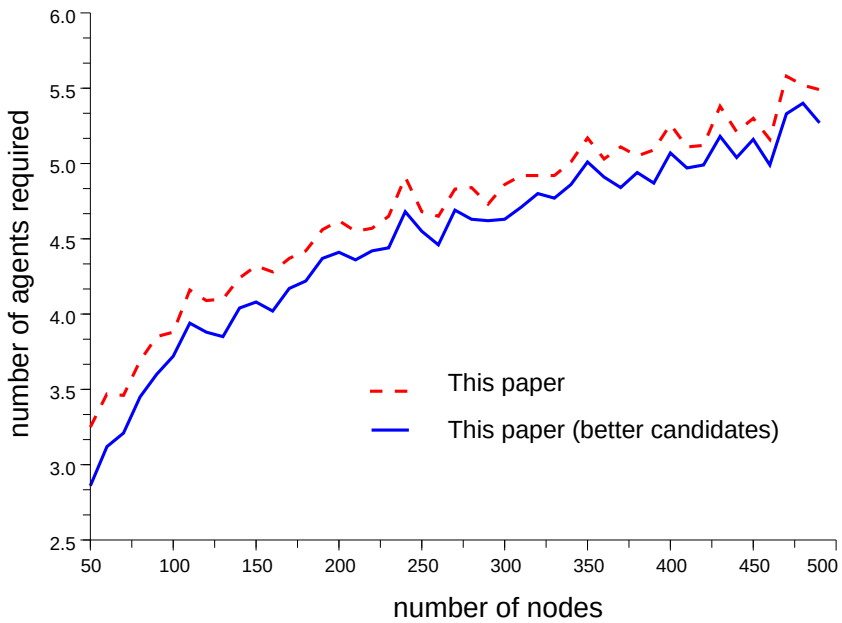


Number of simultaneous agents (break-before-make)

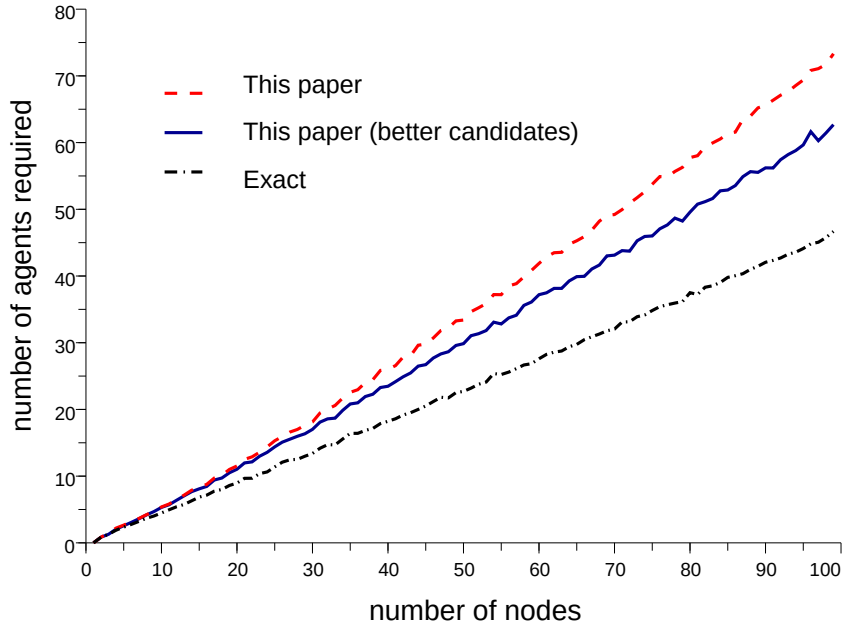


Computation time

Simulation results



2-digraphs



Circular arc graphs

Conclusion & perspectives

Routing reconfiguration in WDM networks

- Modeling using of tools from graph theory
  - Process number & cops-and-robber games
- Take into account priority connections
- New heuristic algorithm

Future work

- More simulations on more realistic scenarios
- Multiple classes of services
- Other objectives, e.g. minimizing interruption time of connections
- Distributed algorithms