Loris Marchal

Équipe Graal - LIP - ENS Lyon

Algorithms for pipelined Multicast

- Groupe de Travail Graal, 2 mars 2004 -

Introduction

- multicast = broadcast to a strict subset of targets in the platform nodes
- lots of studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical netwworks)
- focus on pipelined multicast:
  maximize the throughput of a series of multicast
- same framework as in previous work for other collective communications:

  $$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- multicast = broadcast to a strict subset of targets in the platform nodes
- lots of studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical netwworks)
- focus on pipelined multicast:
  maximize the throughput of a series of multicast
- same framework as in previous work for other collective communications:

  $$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- multicast = broadcast to a strict subset of targets in the platform nodes
- lots of studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical netwworks)
- focus on pipelined multicast:
  maximize the throughput of a series of multicast
- same framework as in previous work for other collective communications:

  scatter, reduce, broadcast $\Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$
- surprisingly, multicast turned out to be more challenging

Introduction

- multicast = broadcast to a strict subset of targets in the platform nodes
- lots of studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical netwworks)
- focus on pipelined multicast:
  maximize the throughput of a series of multicast
- same framework as in previous work for other collective communications:

  $$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$

- surprisingly, multicast turned out to be more challenging

Introduction

- multicast = broadcast to a strict subset of targets in the platform nodes
- lots of studies of multicast:
  - Steiner trees (minimize the cost of a single multicast tree, NP hard problem)
  - for a wide variety of particular architectures and technologies (wormhole-routed, wireless, ad-hoc, optical netwworks)
- focus on pipelined multicast:
  maximize the throughput of a series of multicast
- same framework as in previous work for other collective communications:

  $$\text{scatter, reduce, broadcast} \Rightarrow \begin{cases} \text{optimal throughput,} \\ \text{asymptotically optimal algorithms} \end{cases}$$
- surprisingly, multicast turned out to be more challenging

Outline

1. Framework and Model
2. Some theoretical results: Multicast is hard !
3. Heuristics based on linear programming
4. Tree-based heuristics
5. Experimental results
6. Further on complexity study ?

Outline

1. Framework and Model
2. Some theoretical results: Multicast is hard !
3. Heuristics based on linear programming
4. Tree-based heuristics
5. Experimental results
6. Further on complexity study ?

Outline

1. Framework and Model
2. Some theoretical results: Multicast is hard !
3. Heuristics based on linear programming
4. Tree-based heuristics
5. Experimental results
6. Further on complexity study ?

Outline

1. Framework and Model
2. Some theoretical results: Multicast is hard !
3. Heuristics based on linear programming
4. Tree-based heuristics
5. Experimental results
6. Further on complexity study ?

Outline

1. Framework and Model
2. Some theoretical results: Multicast is hard !
3. Heuristics based on linear programming
4. Tree-based heuristics
5. Experimental results
6. Further on complexity study ?

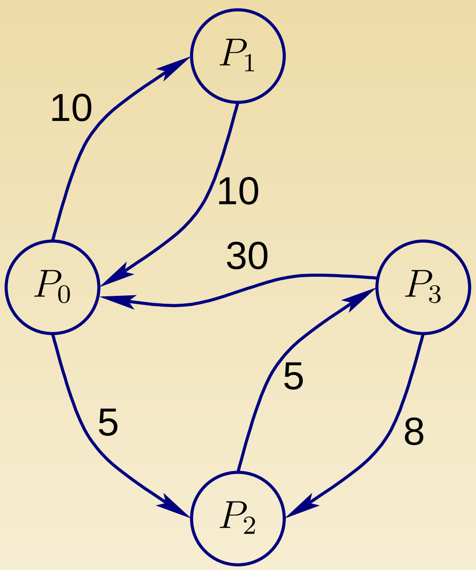
Outline

1. Framework and Model
2. Some theoretical results: Multicast is hard !
3. Heuristics based on linear programming
4. Tree-based heuristics
5. Experimental results
6. Further on complexity study ?

Framework and Model

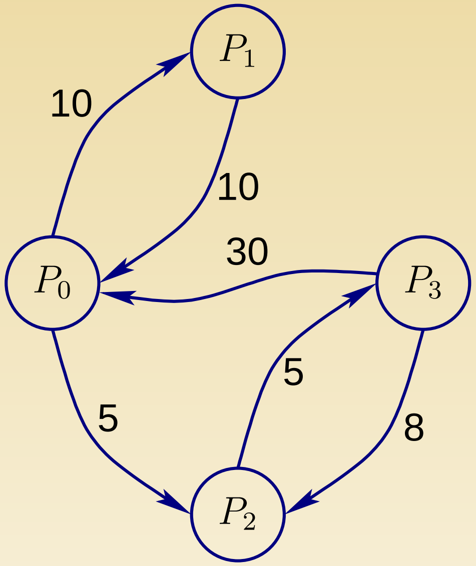
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



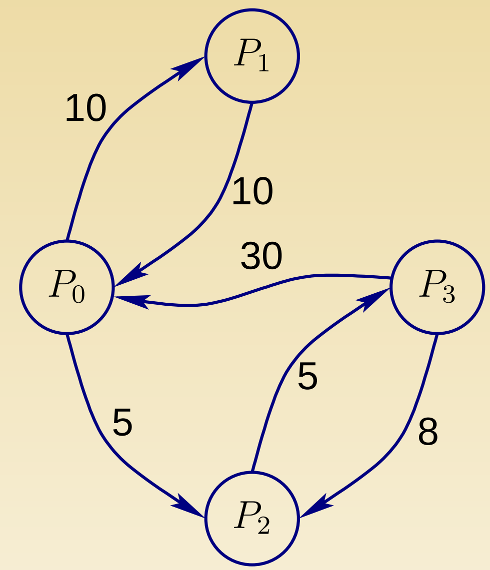
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



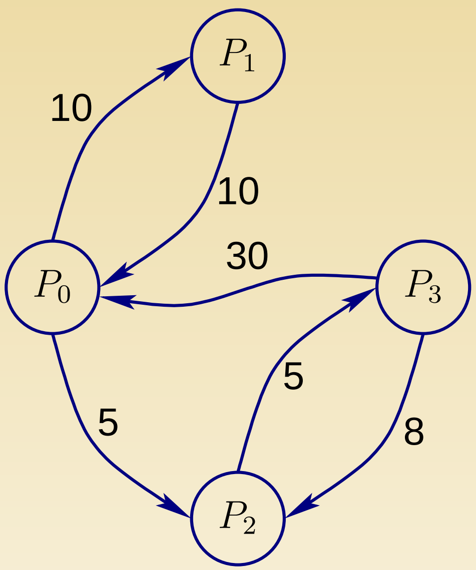
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



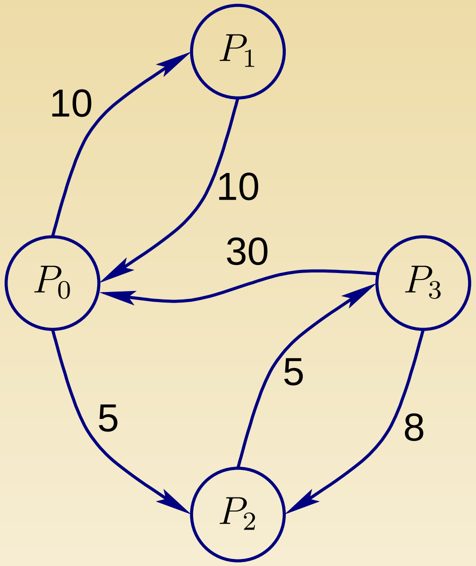
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



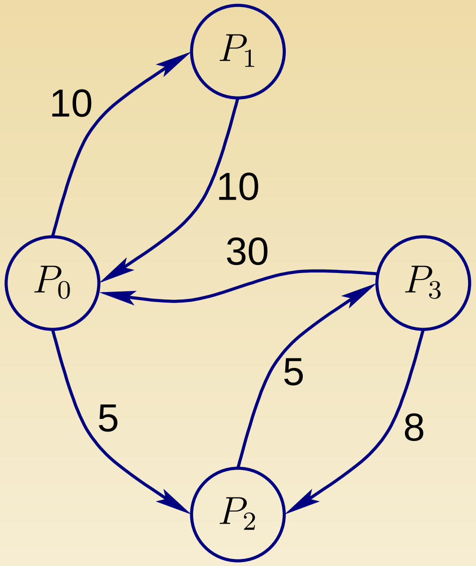
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



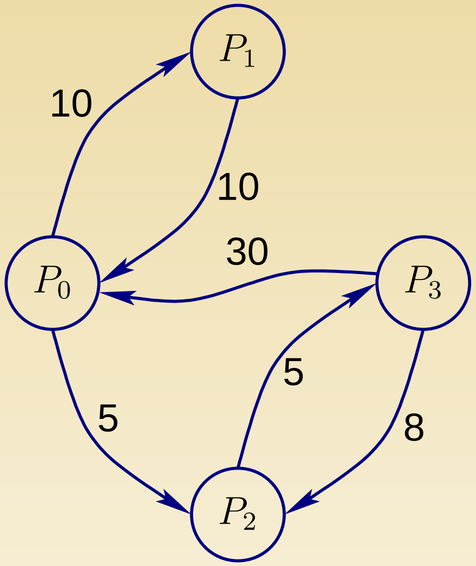
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



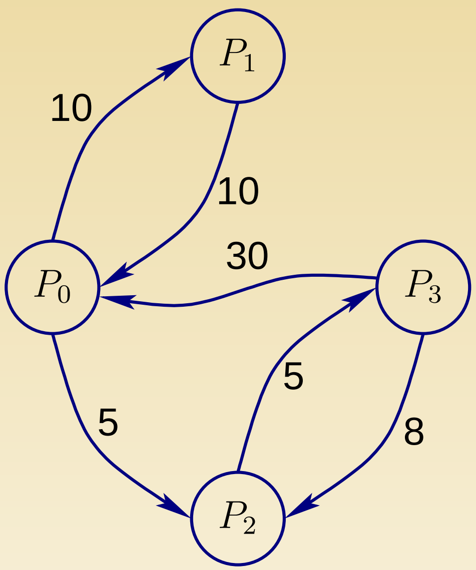
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



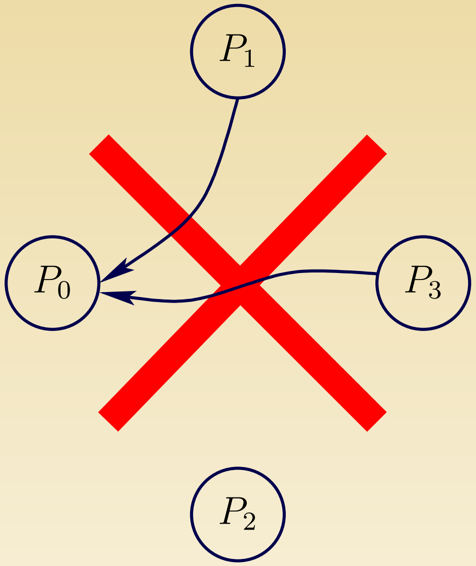
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



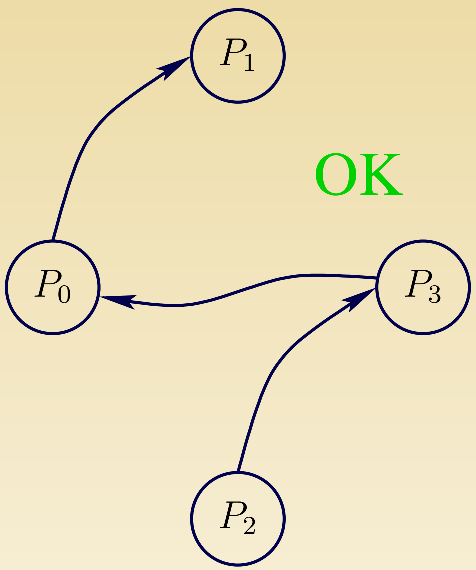
Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



Framework of the platform

- $G = (V, E, c)$
- Let $P_1, P_2, \ldots, P_n$ be the n processors
- P_{source}: processor initiating the multicast
- $\mathcal{P}_{\text{target}}$: set of target processors
- $(j, k) \in E$: communication link between P_i and P_j
- $c(j, k)$: time to transfer one unit message from P_j to P_k
- one-port for incoming communications
- one-port for outgoing communications



Steady-state approach

- Focus on the average quantities, over one time-unit:
  - $t_{i,j}$ average occupation time of edge $P_i \to P_j$
  - $n_{i,j}$ average number of messages going through edge $P_i \to P_j$
  - $x_{i,j}^k$ average number of messages targeting P_k going through $P_i \to P_j$
- Some relations between these quantities:
  - clearly, $0 \leq t_{i,j} \leq 1$
  - a node P_i has a limited sending capacity (one port):
    $$\sum_{j \text{ such that } (i,j)\in E} t_{i,j} \leq 1$$
  - a node P_i has a limited receiving capacity (one port):
    $$\sum_{j \text{ such that } (j,i)\in E} t_{j,i} \leq 1$$
  - link between number of messages and occupation time:
    $$t_{i,j} = n_{i,j} \cdot c_{i,j}$$

Steady-state approach

- Focus on the average quantities, over one time-unit:
  - $t_{i,j}$ average occupation time of edge $P_i \rightarrow P_j$
  - $n_{i,j}$ average number of messages going through edge $P_i \rightarrow P_j$
  - $x_{i,j}^k$ average number of messages targeting P_k going through $P_i \rightarrow P_j$
- Some relations between these quantities:
  - clearly, $0 \leq t_{i,j} \leq 1$
  - a node P_i has a limited sending capacity (one port):
    $$\sum_{j \text{ such that } (i,j) \in E} t_{i,j} \leq 1$$
  - a node P_i has a limited receiving capacity (one port):
    $$\sum_{j \text{ such that } (j,i) \in E} t_{j,i} \leq 1$$
  - link between number of messages and occupation time:
    $$t_{i,j} = n_{i,j} \cdot c_{i,j}$$

Steady-state approach

- Focus on the average quantities, over one time-unit:
  - $t_{i,j}$ average occupation time of edge $P_i \to P_j$
  - $n_{i,j}$ average number of messages going through edge $P_i \to P_j$
  - $x_{i,j}^k$ average number of messages targeting P_k going through $P_i \to P_j$
- Some relations between these quantities:
  - clearly, $0 \leq t_{i,j} \leq 1$
  - a node P_i has a limited sending capacity (one port):
    $$\sum_{j \text{ such that } (i,j) \in E} t_{i,j} \leq 1$$
  - a node P_i has a limited receiving capacity (one port):
    $$\sum_{j \text{ such that } (j,i) \in E} t_{j,i} \leq 1$$
  - link between number of messages and occupation time:
    $$t_{i,j} = n_{i,j} \cdot c_{i,j}$$

Steady-state approach

- Focus on the average quantities, over one time-unit:
  - $t_{i,j}$ average occupation time of edge $P_i \to P_j$
  - $n_{i,j}$ average number of messages going through edge $P_i \to P_j$
  - $x_{i,j}^k$ average number of messages targeting P_k going through $P_i \to P_j$
- Some relations between these quantities:
  - clearly, $0 \leq t_{i,j} \leq 1$
  - a node P_i has a limited sending capacity (one port):
$$\sum_{j \text{ such that } (i,j)\in E} t_{i,j} \leq 1$$
  - a node P_i has a limited receiving capacity (one port):
$$\sum_{j \text{ such that } (j,i)\in E} t_{j,i} \leq 1$$
  - link between number of messages and occupation time:
$$t_{i,j} = n_{i,j} \cdot c_{i,j}$$

Steady-state approach

- Focus on the average quantities, over one time-unit:
  - $t_{i,j}$ average occupation time of edge $P_i \to P_j$
  - $n_{i,j}$ average number of messages going through edge $P_i \to P_j$
  - $x_{i,j}^k$ average number of messages targeting P_k going through $P_i \to P_j$
- Some relations between these quantities:
  - clearly, $0 \leq t_{i,j} \leq 1$
  - a node P_i has a limited sending capacity (one port):
    $$\sum_{j \text{ such that } (i,j) \in E} t_{i,j} \leq 1$$
  - a node P_i has a limited receiving capacity (one port):
    $$\sum_{j \text{ such that } (j,i) \in E} t_{j,i} \leq 1$$
  - link between number of messages and occupation time:
    $$t_{i,j} = n_{i,j} \cdot c_{i,j}$$

Steady-state approach

- Focus on the average quantities, over one time-unit:
  - $t_{i,j}$ average occupation time of edge $P_i \to P_j$
  - $n_{i,j}$ average number of messages going through edge $P_i \to P_j$
  - $x_{i,j}^k$ average number of messages targeting P_k going through $P_i \to P_j$
- Some relations between these quantities:
  - clearly, $0 \leq t_{i,j} \leq 1$
  - a node P_i has a limited sending capacity (one port):
    $$\sum_{j \text{ such that } (i,j)\in E} t_{i,j} \leq 1$$
  - a node P_i has a limited receiving capacity (one port):
    $$\sum_{j \text{ such that } (j,i)\in E} t_{j,i} \leq 1$$
  - link between number of messages and occupation time:
    $$t_{i,j} = n_{i,j} \cdot c_{i,j}$$

Steady-state approach

If we consider the message sent to the target node P_k:

- P_{source} sends TP such messages:

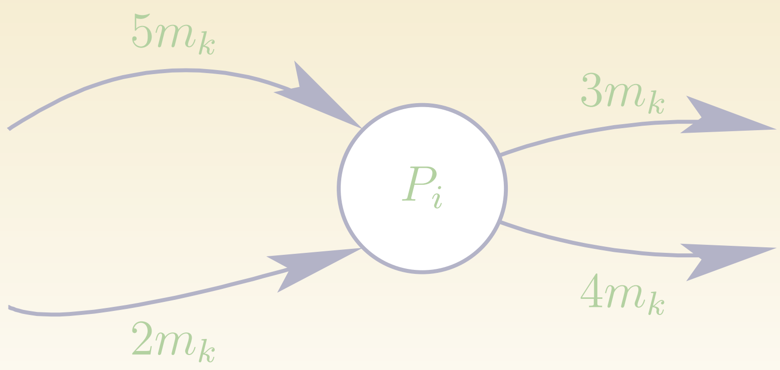
$$\sum_{j/(\text{source},j)\in E} x^k_{\text{source},j} = TP$$

- P_k receives TP such messages:

$$\sum_{j/(j,k)\in E} x^k_{j,k} = TP$$

- On another node P_i, these messages are conserved:

$$\sum_{j/(j,i)\in E} x^k_{j,i} = \sum_{j/(i,j)\in E} x^k_{i,j}$$



Steady-state approach

If we consider the message sent to the target node P_k:

- P_{source} sends TP such messages:

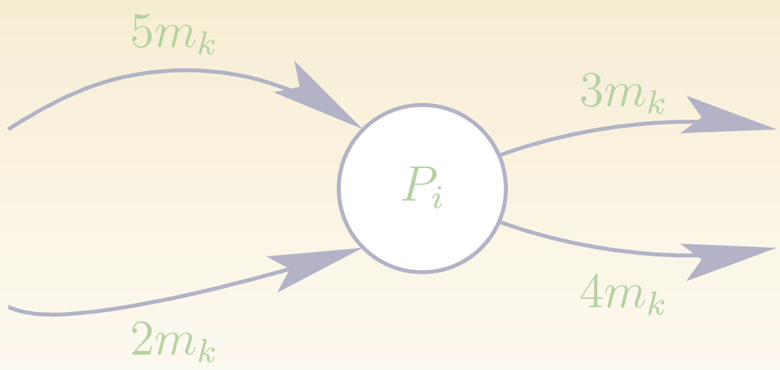
$$\sum_{j/(\text{source},j)\in E} x^k_{\text{source},j} = TP$$

- P_k receives TP such messages:

$$\sum_{j/(j,k)\in E} x^k_{j,k} = TP$$

- On another node P_i, these messages are conserved:

$$\sum_{j/(j,i)\in E} x^k_{j,i} = \sum_{j/(i,j)\in E} x^k_{i,j}$$



Steady-state approach

If we consider the message sent to the target node P_k:

- P_{source} sends TP such messages:

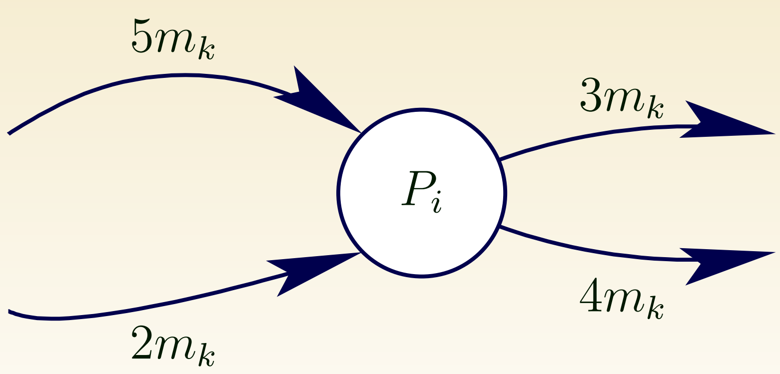
$$\sum_{j/(\text{source},j)\in E} x^k_{\text{source},j} = TP$$

- P_k receives TP such messages:

$$\sum_{j/(j,k)\in E} x^k_{j,k} = TP$$

- On another node P_i, these messages are conserved:

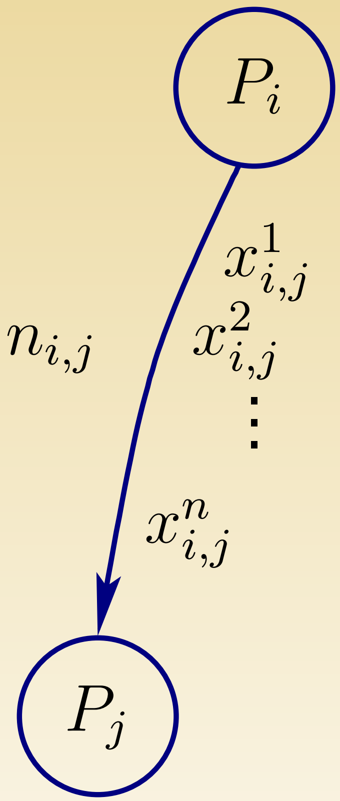
$$\sum_{j/(j,i)\in E} x^k_{j,i} = \sum_{j/(i,j)\in E} x^k_{i,j}$$



Introduction of linear programming

- Objective function: maximize throughput TP
- Which relation between $x_{i,j}^k$ and $n_{i,j}$?

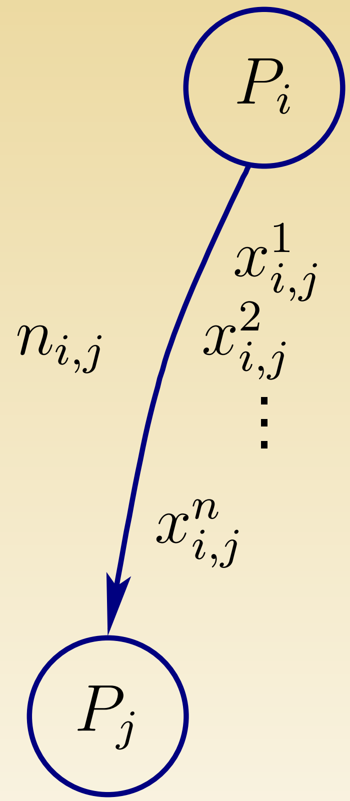
1. Pessimistic view: $n_{i,j} = \sum_k x_{i,j}^k$
   - may be too pessimistic since $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ denotes the same messages
   - as if the source sends differents messages to each target
   - provides a lower bound on the throughput
2. Optimistic view: $n_{i,j} = \max_k x_{i,j}^k$
   - may be too optimistic, if $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ count differents set of messages
   - provides an upper bound on the throughput



Introduction of linear programming

- Objective function: maximize throughput TP
- Which relation between $x_{i,j}^k$ and $n_{i,j}$?

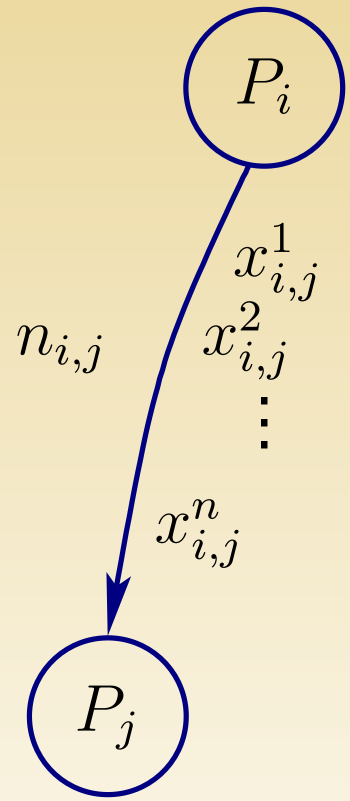
1. Pessimistic view: $n_{i,j} = \sum_k x_{i,j}^k$
   - may be too pessimistic since $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ denotes the same messages
   - as if the source sends differents messages to each target
   - provides a lower bound on the throughput
2. Optimistic view: $n_{i,j} = \max_k x_{i,j}^k$
   - may be too optimistic, if $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ count differents set of messages
   - provides an upper bound on the throughput



Introduction of linear programming

- Objective function: maximize throughput TP
- Which relation between $x_{i,j}^k$ and $n_{i,j}$?

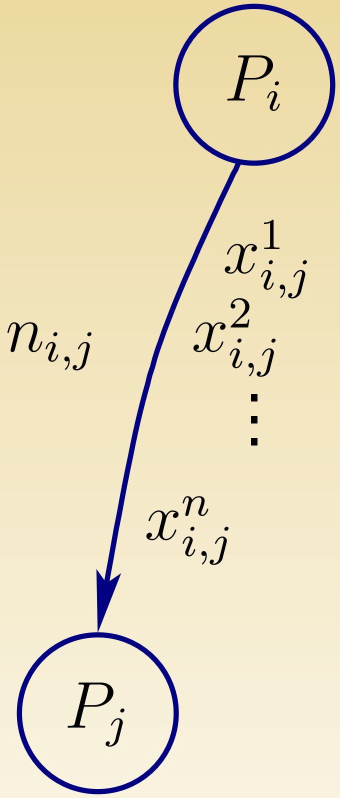
1. Pessimistic view: $n_{i,j} = \sum_k x_{i,j}^k$
   - may be too pessimistic since $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ denotes the same messages
   - as if the source sends differents messages to each target
   - provides a lower bound on the throughput
2. Optimistic view: $n_{i,j} = \max_k x_{i,j}^k$
   - may be too optimistic, if $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ count differents set of messages
   - provides an upper bound on the throughput



Introduction of linear programming

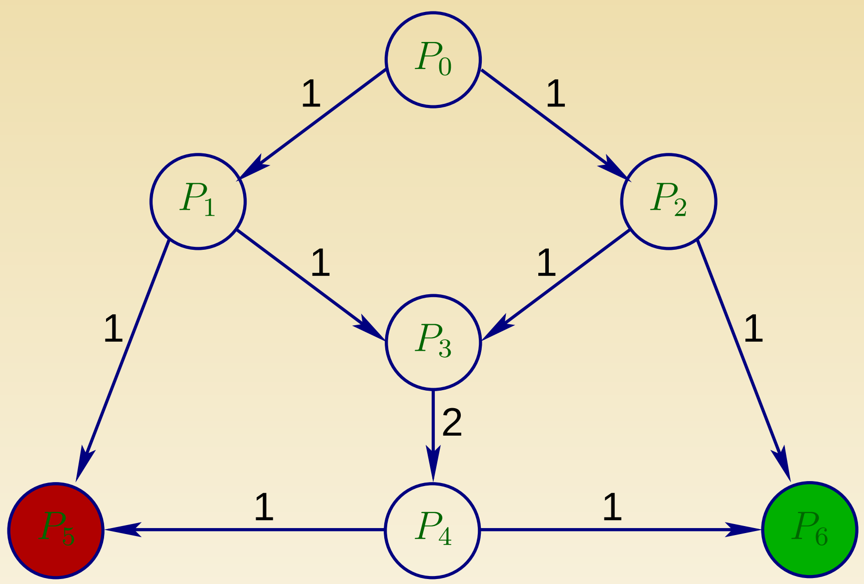
- Objective function: maximize throughput TP
- Which relation between $x_{i,j}^k$ and $n_{i,j}$?

1. Pessimistic view: $n_{i,j} = \sum_k x_{i,j}^k$
   - may be too pessimistic since $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ denotes the same messages
   - as if the source sends differents messages to each target
   - provides a lower bound on the throughput
2. Optimistic view: $n_{i,j} = \max_k x_{i,j}^k$
   - may be too optimistic, if $x_{i,j}^{k_1}$ and $x_{i,j}^{k_2}$ count differents set of messages
   - provides an upper bound on the throughput



Why solution 2 is not always feasible

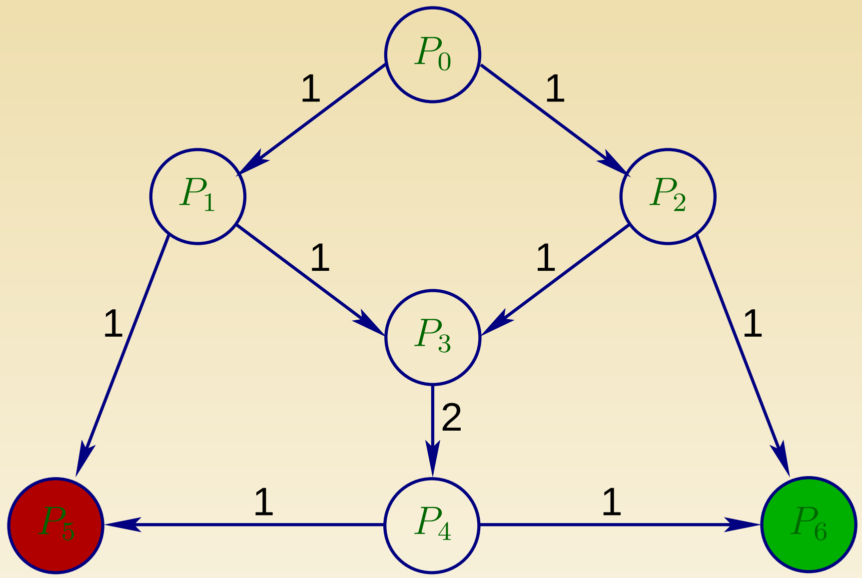
Consider the following platform, where the multicast set consists in the colored nodes:



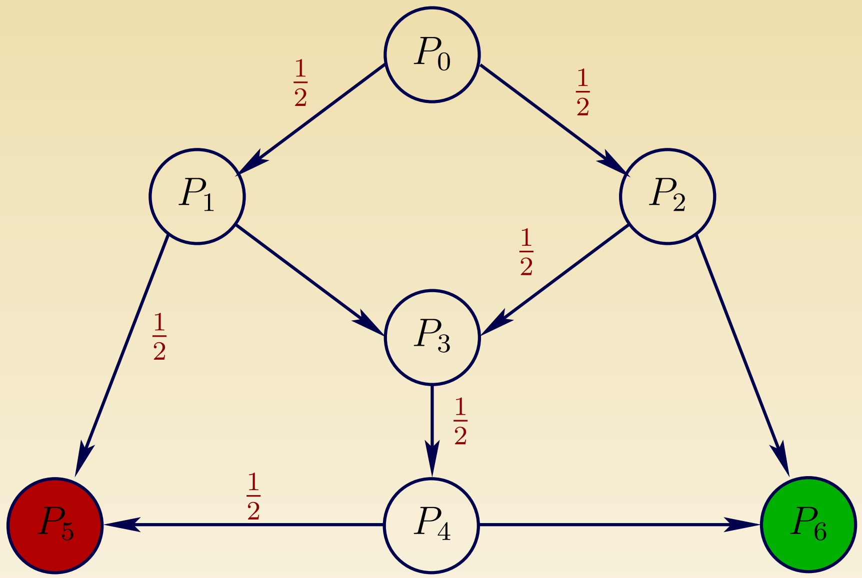
The linear program provides the following solution with throughput 1:

Why solution 2 is not always feasible

Consider the following platform, where the multicast set consists in the colored nodes:

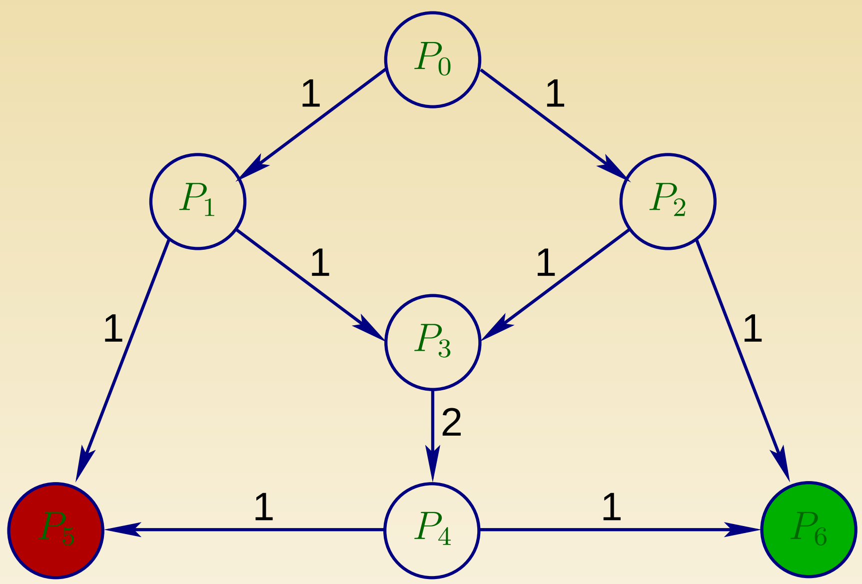


The linear program provides the following solution with throughput 1:

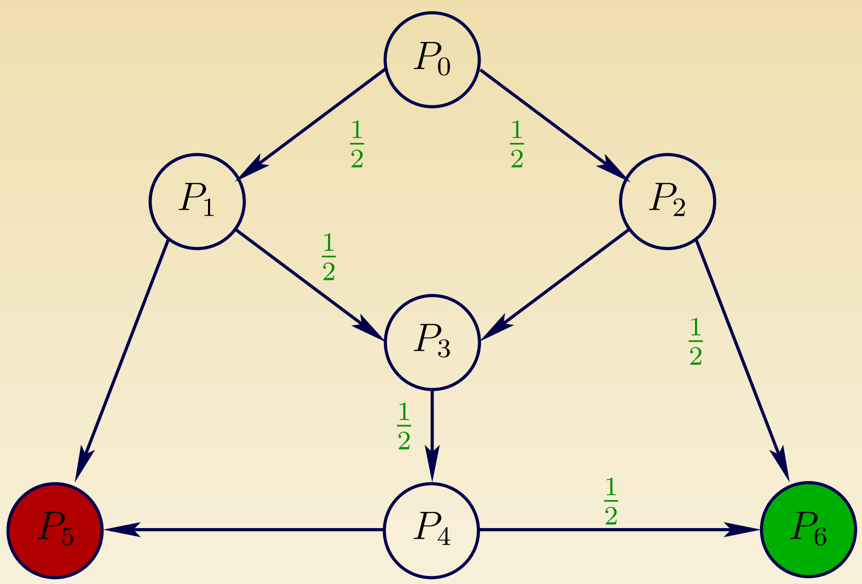


Why solution 2 is not always feasible

Consider the following platform, where the multicast set consists in the colored nodes:

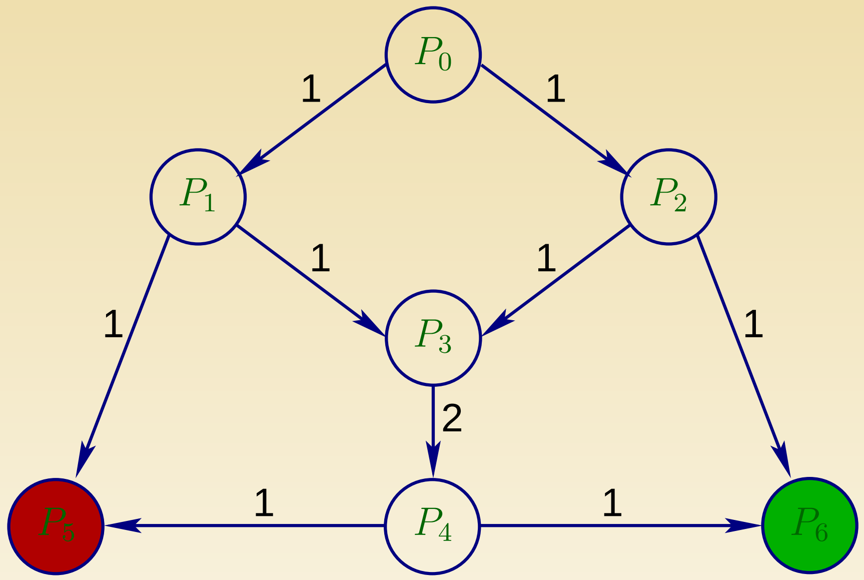


The linear program provides the following solution with throughput 1:

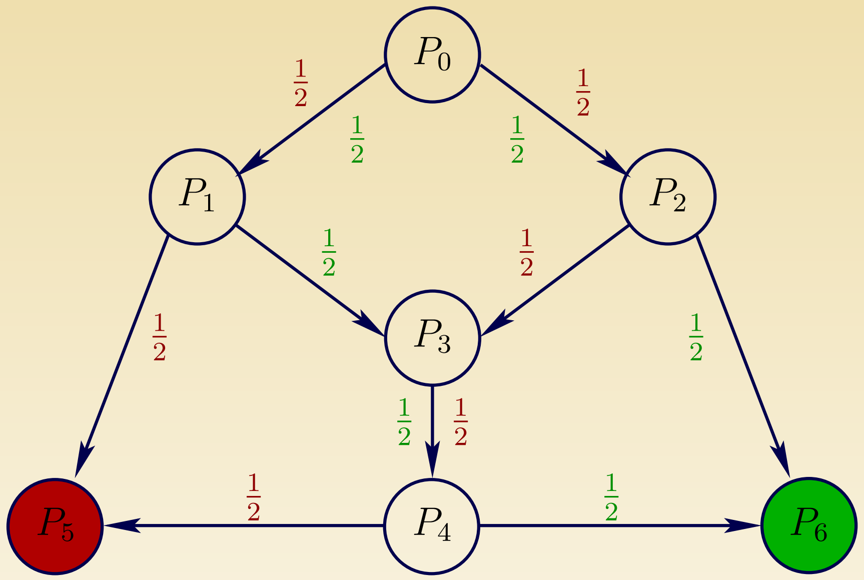


Why solution 2 is not always feasible

Consider the following platform, where the multicast set consists in the colored nodes:

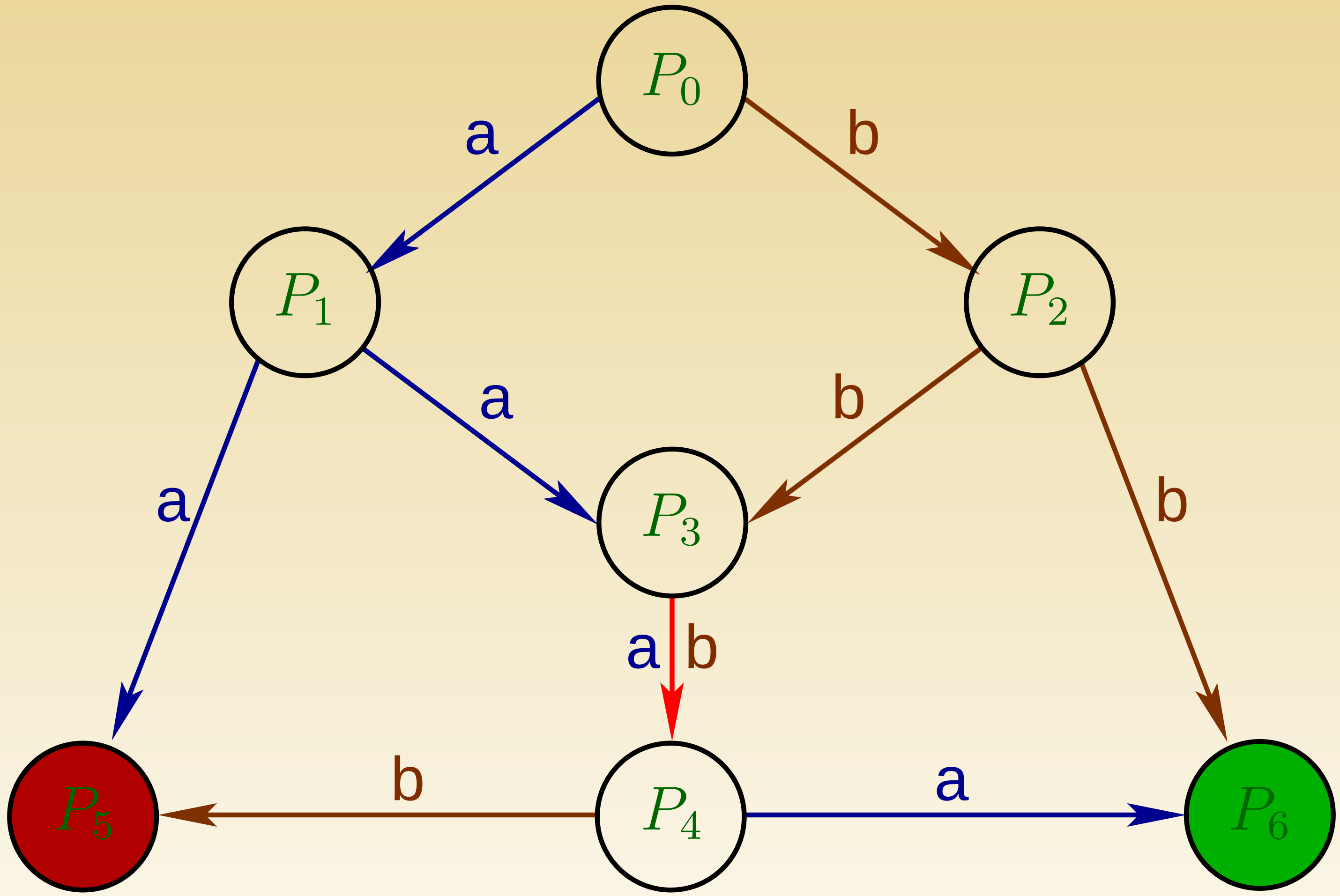


The linear program provides the following solution with throughput 1:



Why solution 2 is not always feasible

Nevertheless, the obtained throughput is not feasible:



Theoretical results

Theoretical results

- Neither linear programs can compute the true optimal throughput
- **theorem :** computing the best throughput for a multicast operation on a given platform is NP-hard
- **definition** multicast tree:
  a tree, rooted in P_{source}, spanning all the nodes of $\mathcal{P}_{\text{target}}$, and made up valid edges from E

Theoretical results

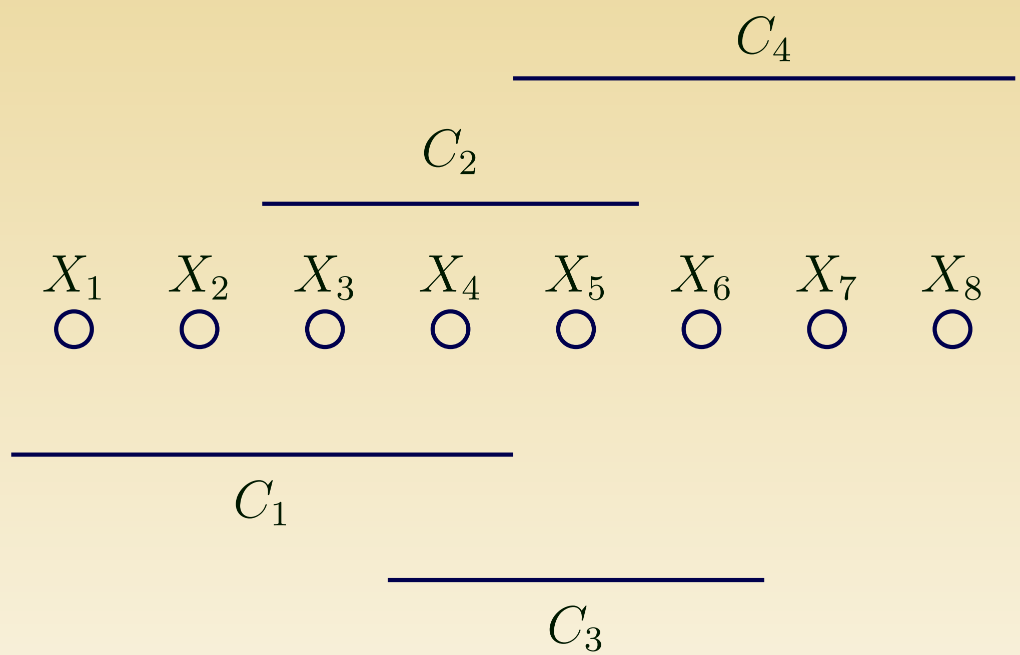
- Neither linear programs can compute the true optimal throughput
- **theorem :** computing the best throughput for a multicast operation on a given platform is NP-hard
- **definition** multicast tree:
  a tree, rooted in P_{source}, spanning all the nodes of $\mathcal{P}_{\text{target}}$, and made up valid edges from E

Theoretical results

- Neither linear programs can compute the true optimal throughput
- **theorem :** computing the best throughput for a multicast operation on a given platform is NP-hard
- **definition** multicast tree:
  a tree, rooted in P_{source}, spanning all the nodes of $\mathcal{P}_{\text{target}}$, and made up valid edges from E

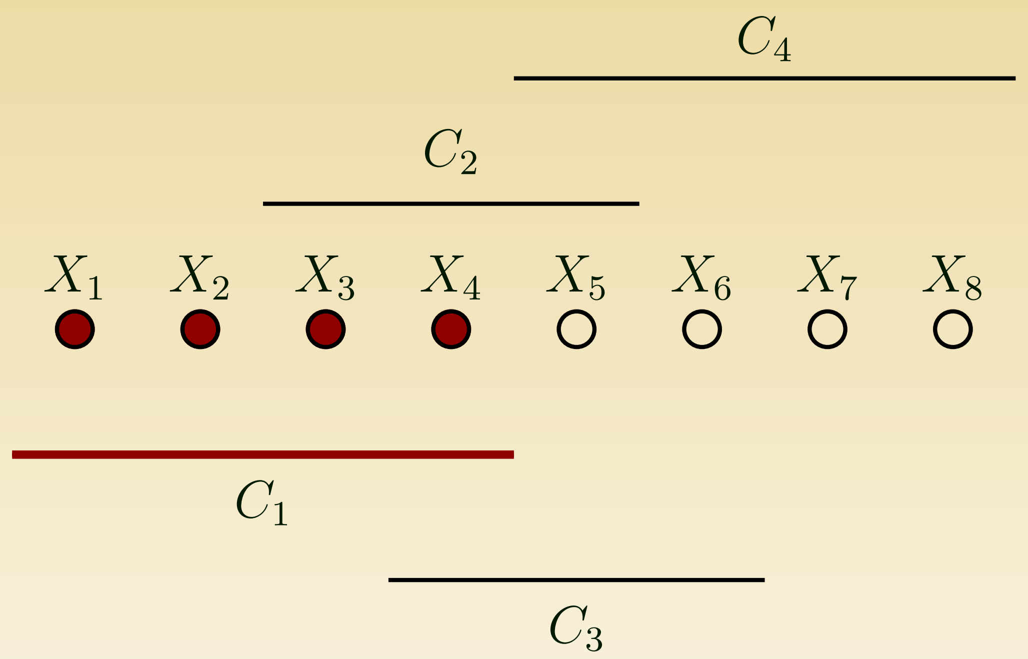
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



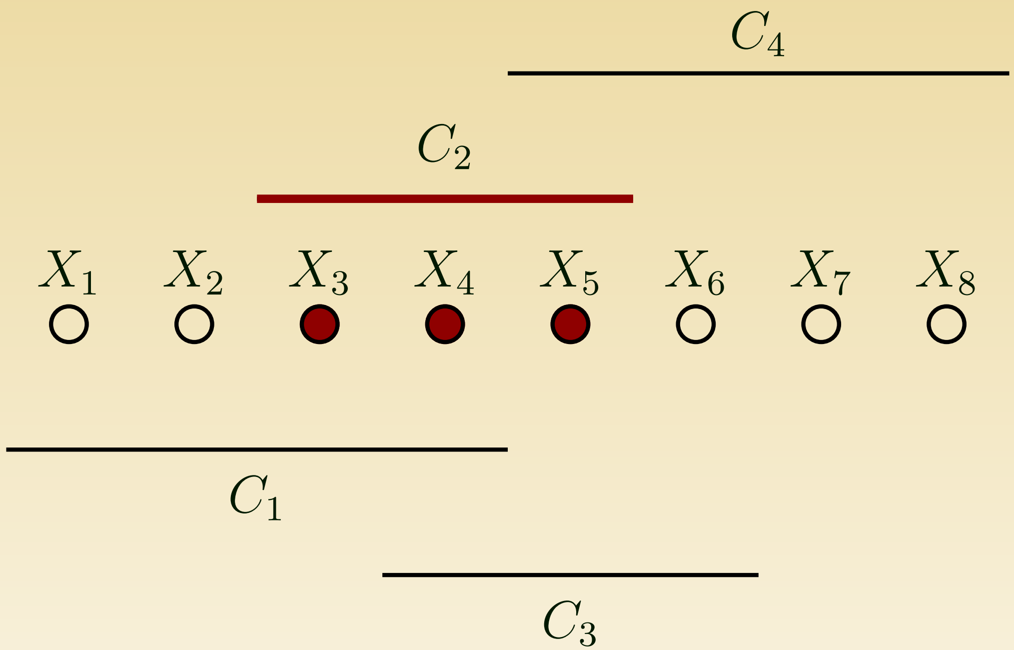
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



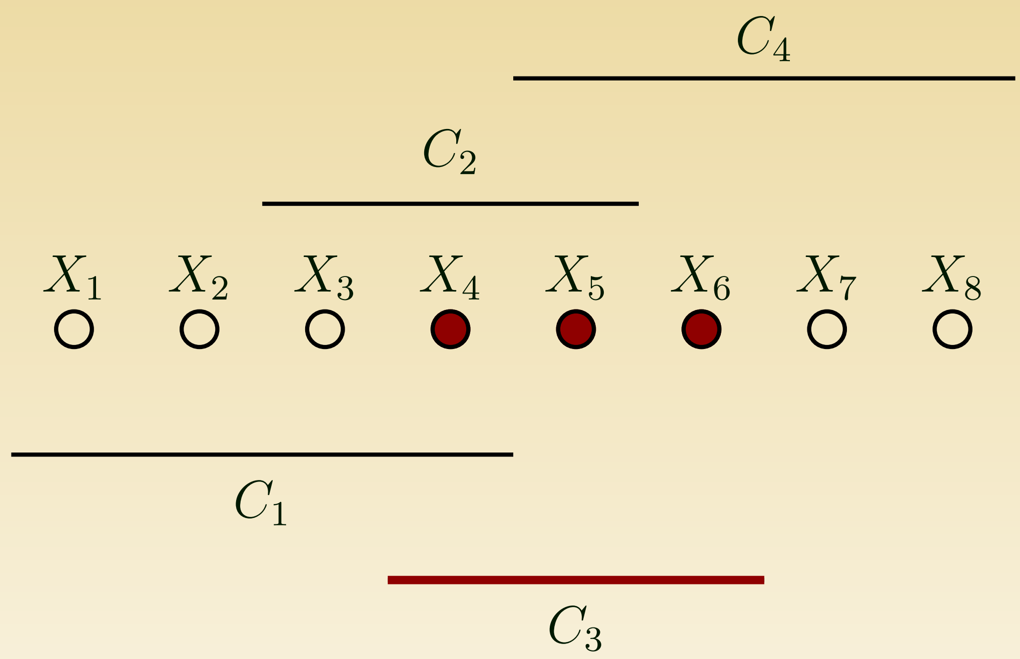
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



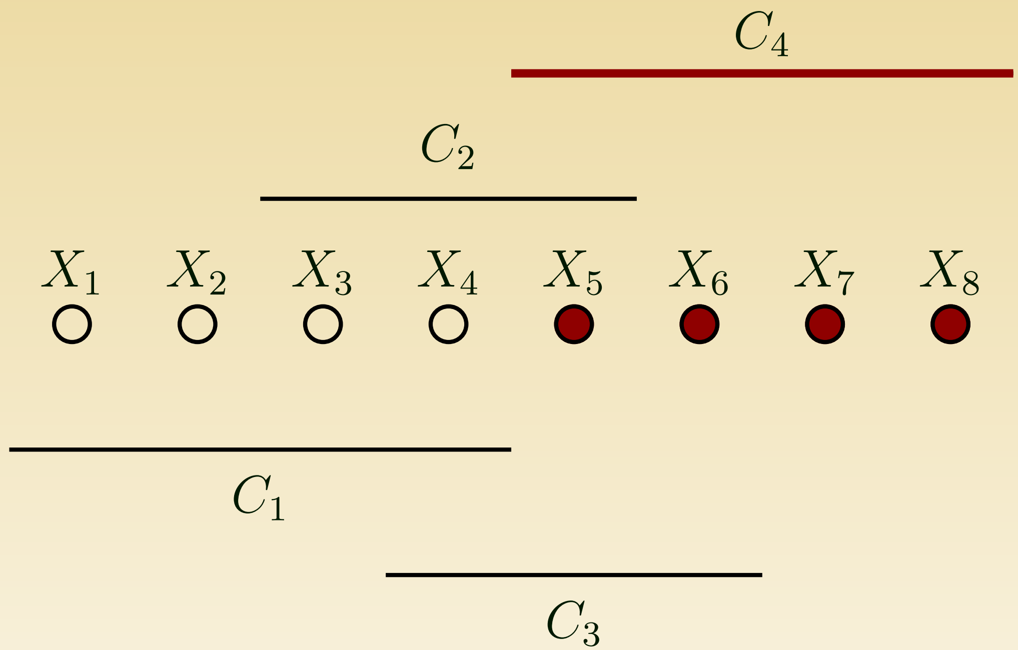
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



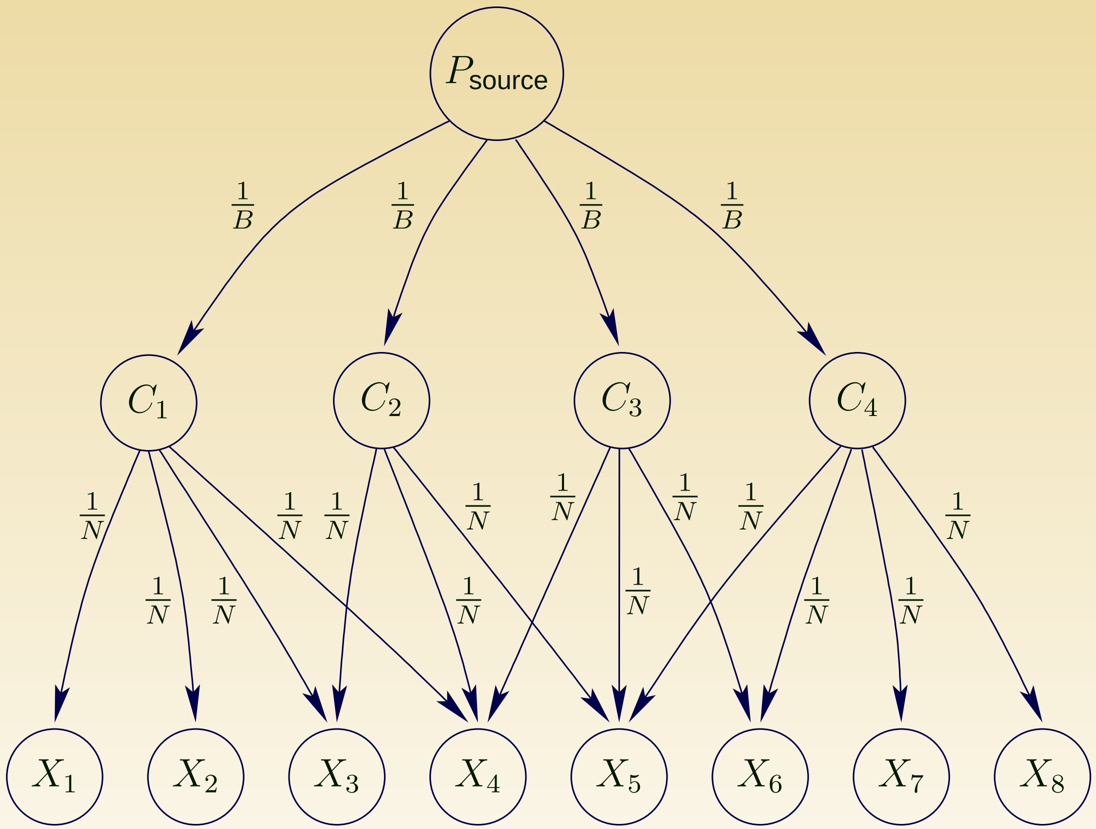
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



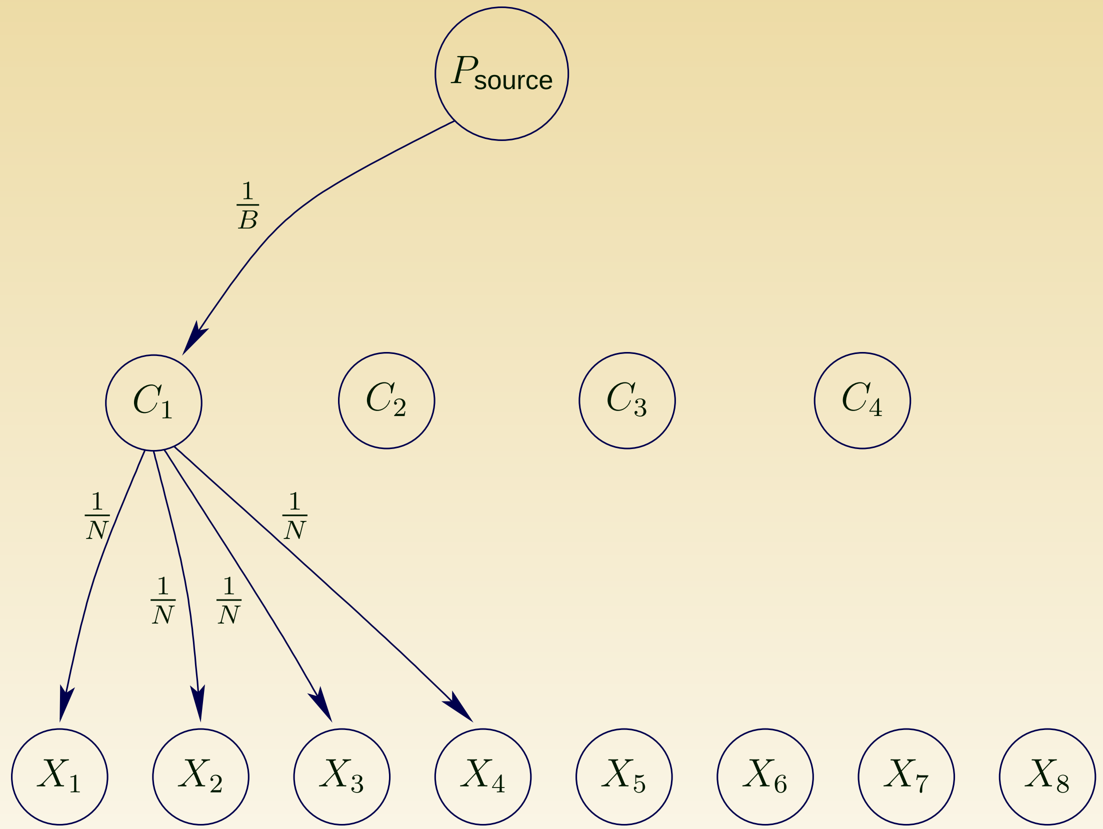
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



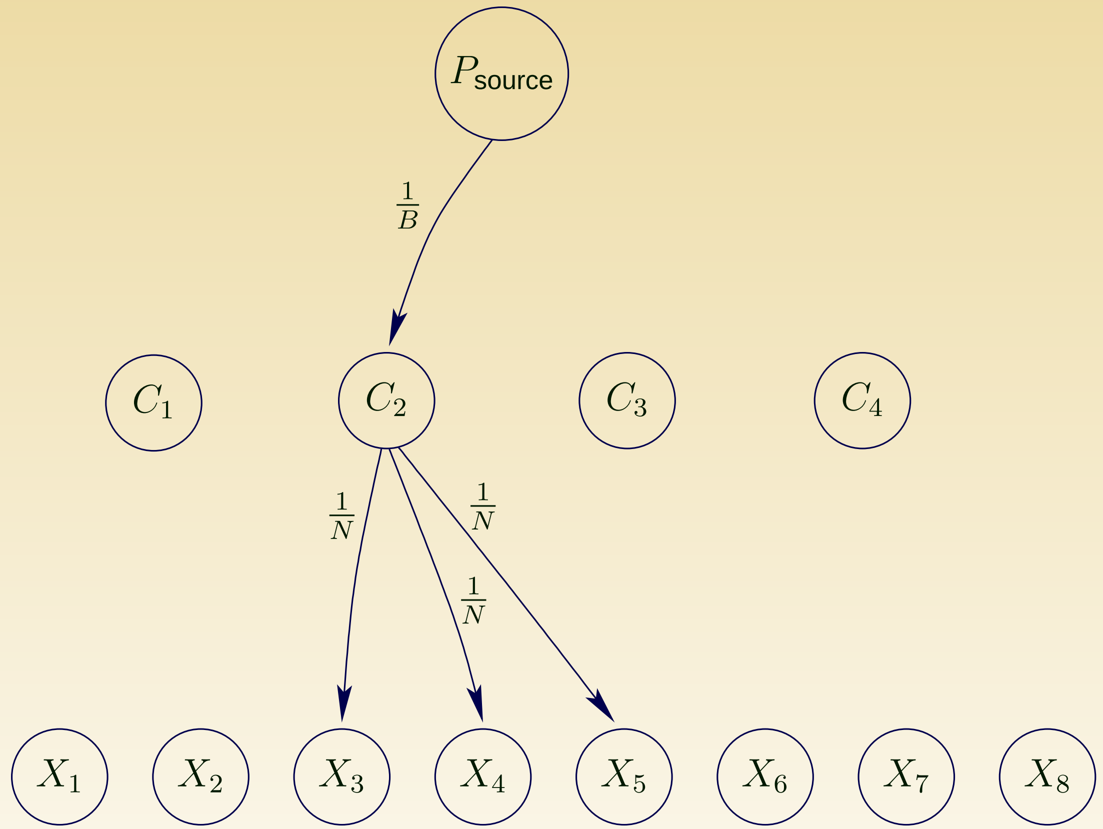
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



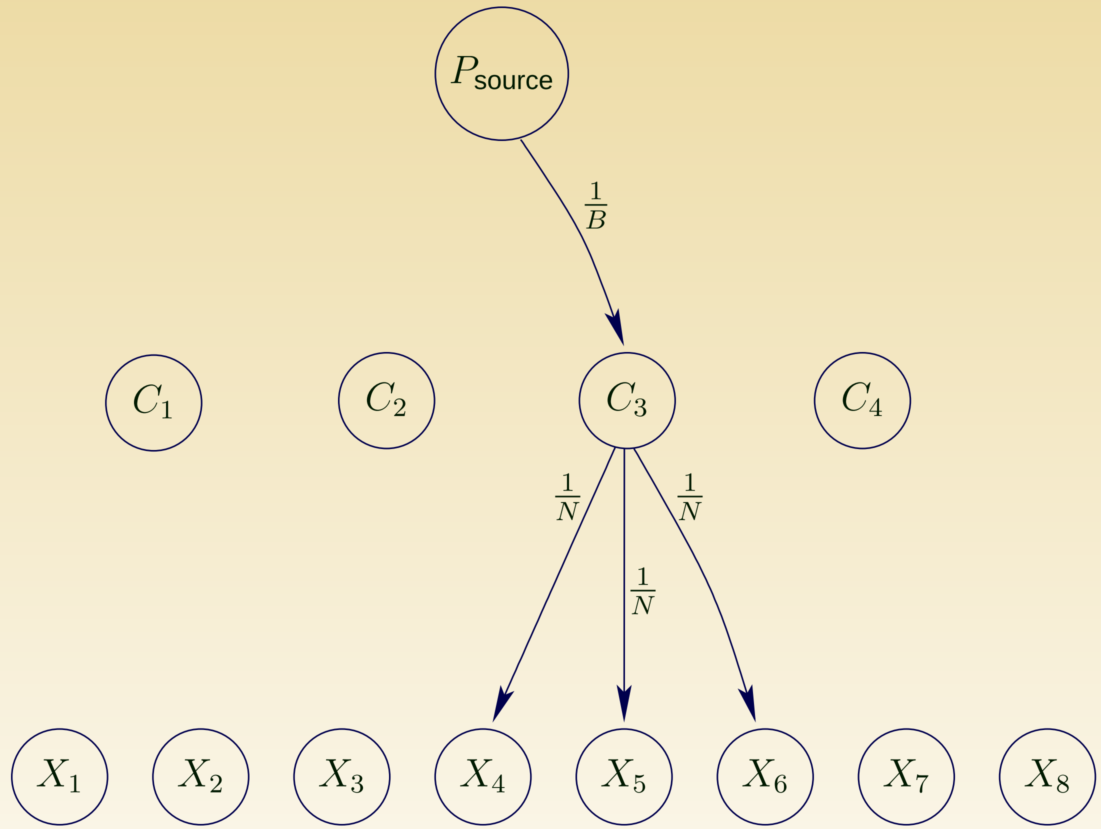
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



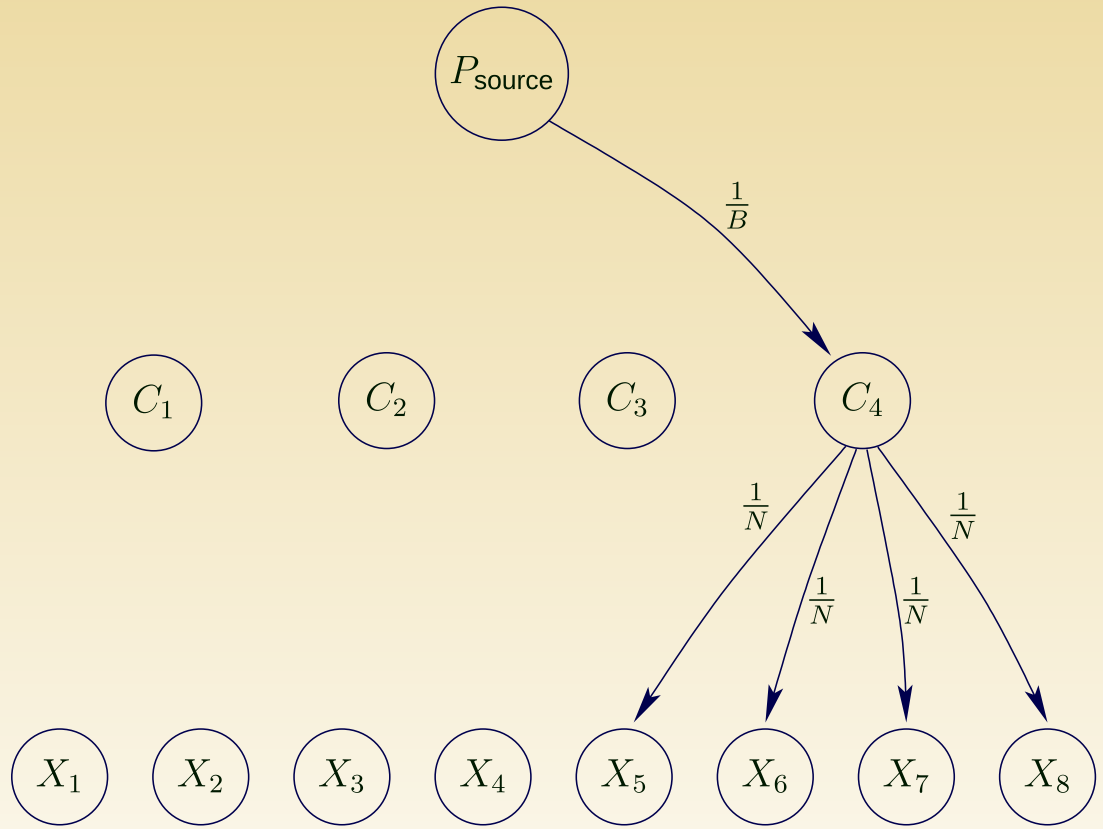
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



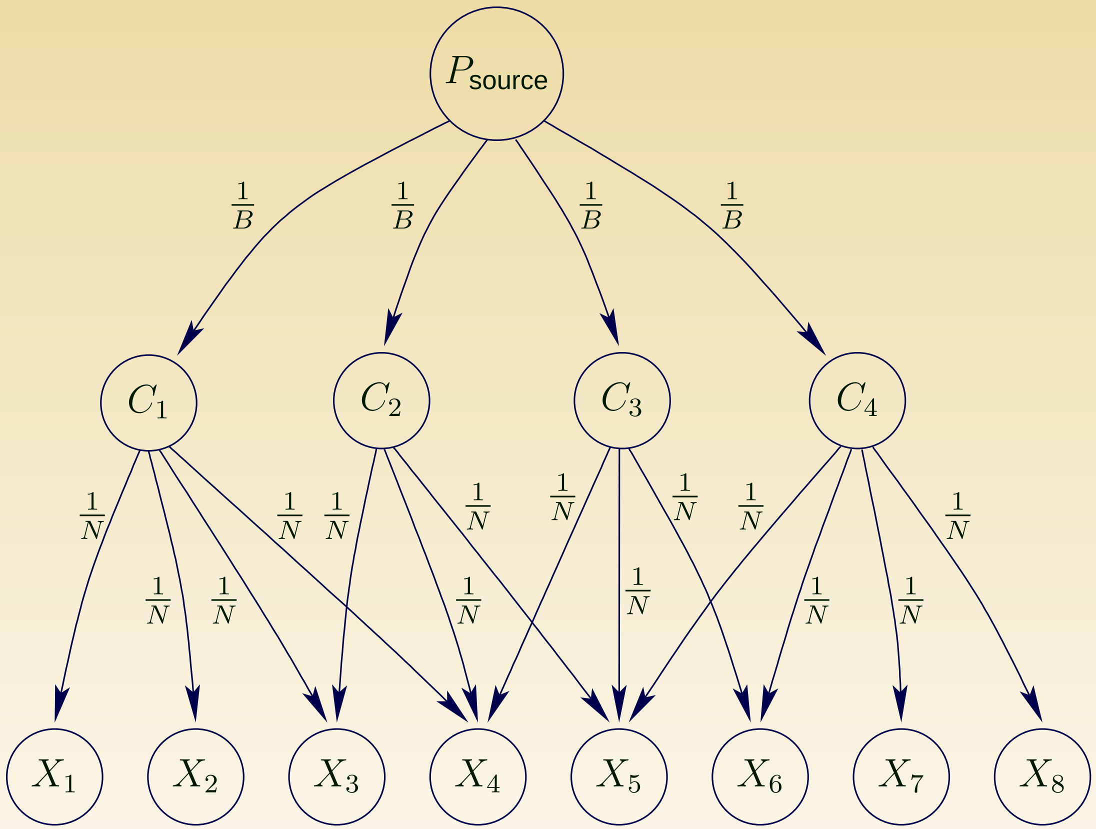
NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



NP-Completeness

- reduction from MINIMUM-SET-COVER:
  C is a collection of subsets of X, a B is a bound
  does C contain a cover of X of size at most B ?



Heuristic based on linear programming

Straightforward heuristics

- lower bound on the throughput $\rightarrow$ scatter heuristic
  $|\mathcal{P}_{\text{target}}|$: cardinal of the target set
  scatter has a guarantee factor of $|\mathcal{P}_{\text{target}}|$:

$$\text{throughput}(\text{scatter}) \geq \frac{\text{upper bound}}{|\mathcal{P}_{\text{target}}|}$$

- broadcast on the whole platform
  the optimistic view ($n_{i,j} = \max_k x_{i,j}^k$) leads to a feasible schedule in this case

Straightforward heuristics

- lower bound on the throughput $\rightarrow$ scatter heuristic

  $|\mathcal{P}_{\text{target}}|$: cardinal of the target set

  scatter has a guarantee factor of $|\mathcal{P}_{\text{target}}|$:

  $$\text{throughput}(\text{scatter}) \geq \frac{\text{upper bound}}{|\mathcal{P}_{\text{target}}|}$$

- broadcast on the whole platform

  the optimistic view ($n_{i,j} = \max_k x_{i,j}^k$) leads to a feasible schedule in this case

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:

$$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$

  3. set $V = V \setminus \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
  $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \setminus \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
  $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \setminus \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
  $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \backslash \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
  $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \setminus \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
     $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \backslash \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
     $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \setminus \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Reduced Broadcast:
  1. compute the solution of the broadcast
  2. choose the node P_{min} not in the original $\mathcal{P}_{\text{target}}$ which forwards the minimum of messages:
  $$\text{MIN} \sum_{i \in \mathcal{P}_{\text{target}}} \sum_{P_j \in \mathcal{N}^{\text{in}}(P_m)} x_i^{j,m}$$
  3. set $V = V \setminus \{P_{\text{min}}\}$ and start again until the throughput is not improved
- Augmented Multicast:
  1. compute the solution of the scatter
  2. choose the node P_{max} not in the original $\mathcal{P}_{\text{target}}$ which forwards the maximum of messages
  3. add this node to $\mathcal{P}_{\text{target}}$ if it improves the throughput of a broadcast on the set of nodes $\{P_{\text{source}}\} \cup \mathcal{P}_{\text{target}}$

Refined heuristics

- Multisource Multicast

  1. start from the solution of a scatter
  2. compute the node which forwards the maximum of messages
  3. add this node as secondary source:

     it receives all the messages from the previous sources

     it sends part of the messages to the target nodes

Refined heuristics

- Multisource Multicast

1. start from the solution of a scatter
2. compute the node which forwards the maximum of messages
3. add this node as secondary source:
   it receives all the messages from the previous sources
   it sends part of the messages to the target nodes

Refined heuristics

- Multisource Multicast

1. start from the solution of a scatter
2. compute the node which forwards the maximum of messages
3. add this node as secondary source:
   it receives all the messages from the previous sources
   it sends part of the messages to the target nodes

Refined heuristics

- Multisource Multicast

1. start from the solution of a scatter
2. compute the node which forwards the maximum of messages
3. add this node as secondary source:
   it receives all the messages from the previous sources
   it sends part of the messages to the target nodes

Tree-based heuristic

Steiner tree

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- Minimum Steiner Tree: NP-complete
- some heuristics exist, among other the Minimum Cost Path Heuristic:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Steiner tree

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- Minimum Steiner Tree: NP-complete
- some heuristics exist, among other the Minimum Cost Path Heuristic:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Steiner tree

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- Minimum Steiner Tree: NP-complete
- some heuristics exist, among other the Minimum Cost Path Heuristic:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Steiner tree

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- Minimum Steiner Tree: NP-complete
- some heuristics exist, among other the Minimum Cost Path Heuristic:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Steiner tree

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- Minimum Steiner Tree: NP-complete
- some heuristics exist, among other the Minimum Cost Path Heuristic:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Steiner tree

- problem: find a low-cost multicast tree
- cost: sum of the weights of the edges in the tree
- Minimum Steiner Tree: NP-complete
- some heuristics exist, among other the Minimum Cost Path Heuristic:
  - grow a tree until it spans all the target nodes
  - at each step, find the target which could be added with minimum cost to the current tree

Minimum Cost Path Heuristic

- we adapt the previous heuristic to our metric:

$$\max_i(\sum \text{cost of all edges } P_i \rightarrow P_j \text{ in the tree } T)$$

1. $T = (P_{\text{source}}, \emptyset)$
2. choose the target node P_t wich minimizes
$$max(\text{cost of the edges on the path } P_{\text{source}} \rightsquigarrow P_t)$$
3. add the path $P_{\text{source}} \rightsquigarrow P_t$ to the tree T
4. update the cost of the edges: if (i, j) is a new edge of the tree,
   - $\forall$ edge (i, k) $c(i, k) \leftarrow c(i, k) + c(i, j)$
   - $c(i, j) \leftarrow 0$

Minimum Cost Path Heuristic

- we adapt the previous heuristic to our metric:

$$\max_i(\sum \text{cost of all edges } P_i \to P_j \text{ in the tree } T)$$

1. $T = (P_{\text{source}}, \emptyset)$
2. choose the target node P_t wich minimizes

   $$max(\text{cost of the edges on the path } P_{\text{source}} \rightsquigarrow P_t)$$
3. add the path $P_{\text{source}} \rightsquigarrow P_t$ to the tree T
4. update the cost of the edges: if (i, j) is a new edge of the tree,
   - $\forall$ edge (i, k) $c(i, k) \leftarrow c(i, k) + c(i, j)$
   - $c(i, j) \leftarrow 0$

Minimum Cost Path Heuristic

- we adapt the previous heuristic to our metric:

$$\max_i(\sum \text{cost of all edges } P_i \rightarrow P_j \text{ in the tree } T)$$

1. $T = (P_{\text{source}}, \emptyset)$
2. choose the target node P_t wich minimizes

$$max(\text{cost of the edges on the path } P_{\text{source}} \rightsquigarrow P_t)$$

3. add the path $P_{\text{source}} \rightsquigarrow P_t$ to the tree T
4. update the cost of the edges: if (i,j) is a new edge of the tree,
   - $\forall$ edge (i,k) $c(i,k) \leftarrow c(i,k) + c(i,j)$
   - $c(i,j) \leftarrow 0$

Minimum Cost Path Heuristic

- we adapt the previous heuristic to our metric:

$$\max_i(\sum \text{cost of all edges } P_i \to P_j \text{ in the tree } T)$$

1. $T = (P_{\text{source}}, \emptyset)$
2. choose the target node P_t wich minimizes

$$max(\text{cost of the edges on the path } P_{\text{source}} \rightsquigarrow P_t)$$

3. add the path $P_{\text{source}} \rightsquigarrow P_t$ to the tree T
4. update the cost of the edges: if (i, j) is a new edge of the tree,
   - $\forall$ edge (i, k) $c(i, k) \leftarrow c(i, k) + c(i, j)$
   - $c(i, j) \leftarrow 0$

Minimum Cost Path Heuristic

- we adapt the previous heuristic to our metric:

$$\max_i(\sum \text{cost of all edges } P_i \rightarrow P_j \text{ in the tree } T)$$

1. $T = (P_{\text{source}}, \emptyset)$
2. choose the target node P_t wich minimizes

$$max(\text{cost of the edges on the path } P_{\text{source}} \rightsquigarrow P_t)$$

3. add the path $P_{\text{source}} \rightsquigarrow P_t$ to the tree T
4. update the cost of the edges: if (i, j) is a new edge of the tree,
   - $\forall$ edge (i, k) $c(i, k) \leftarrow c(i, k) + c(i, j)$
   - $c(i, j) \leftarrow 0$

Experimental results

Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

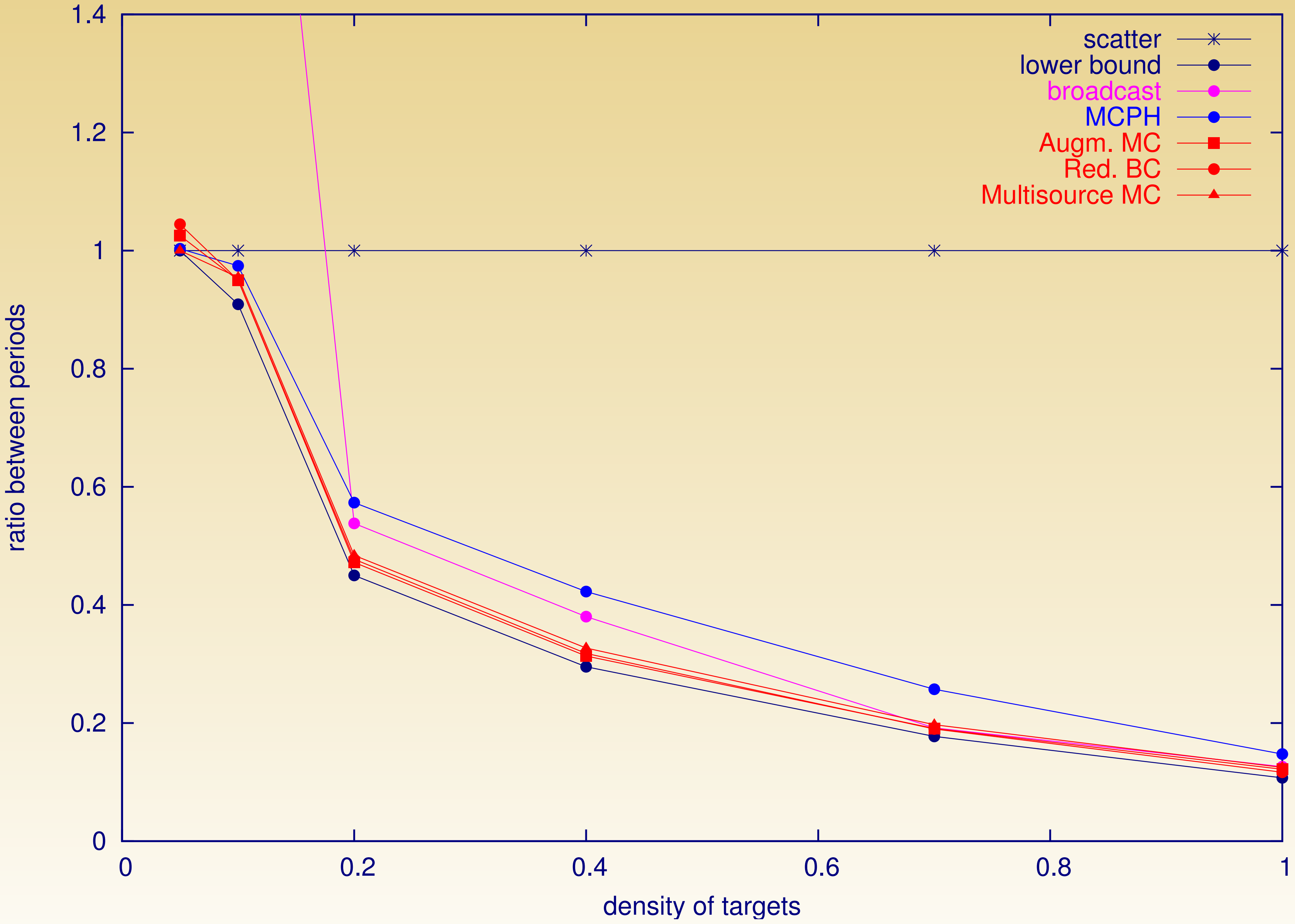
Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

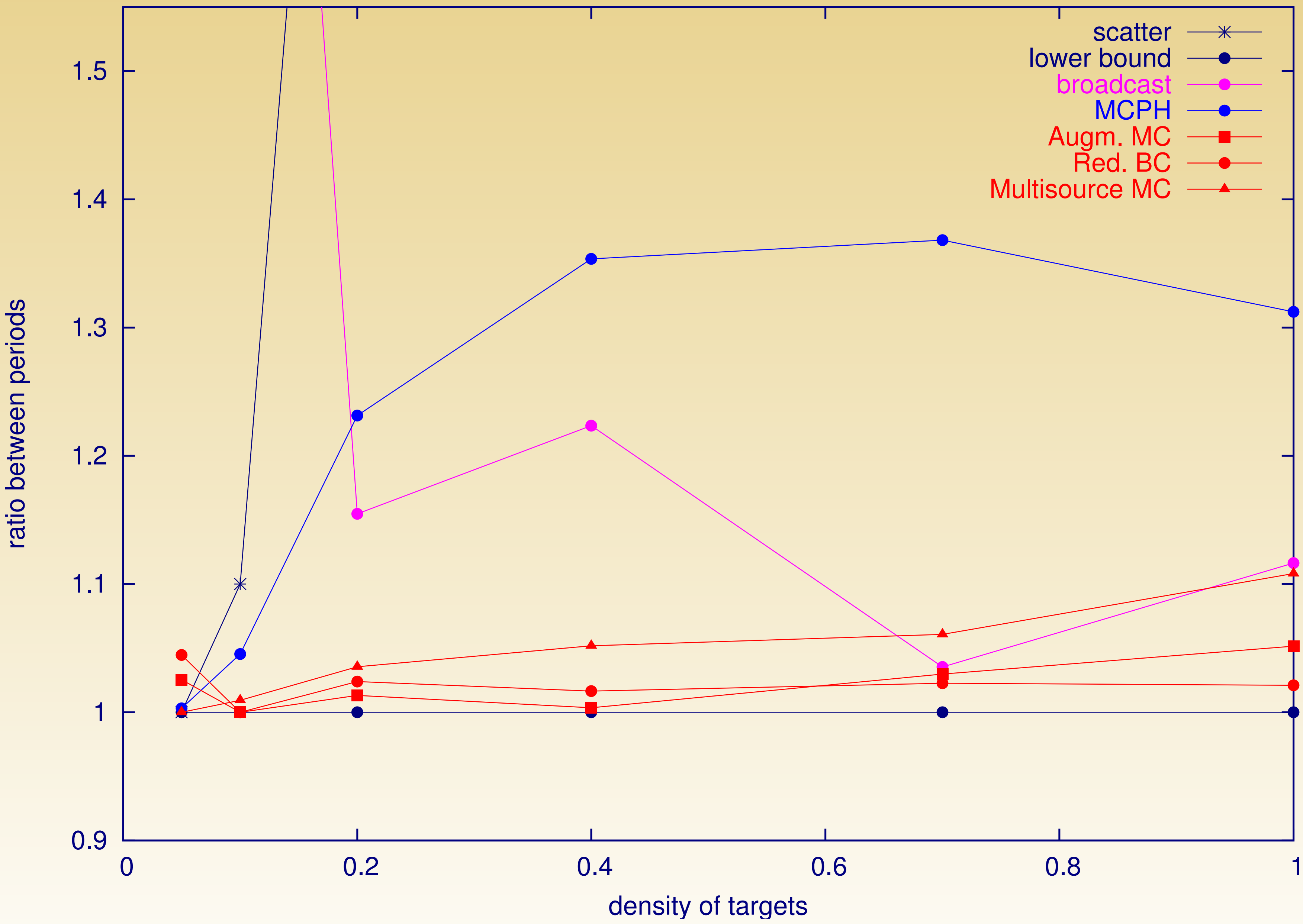
Experimental results

- we perform experiments on platforms generated by Tiers
- two types of platforms:
  - one “big”: 65 nodes
  - one “small”: 30 nodes
- results: comparison of the throughput of our heuristics over the two bounds:
  - over the lower bound (scatter operation)
  - over the theoretical upper bound

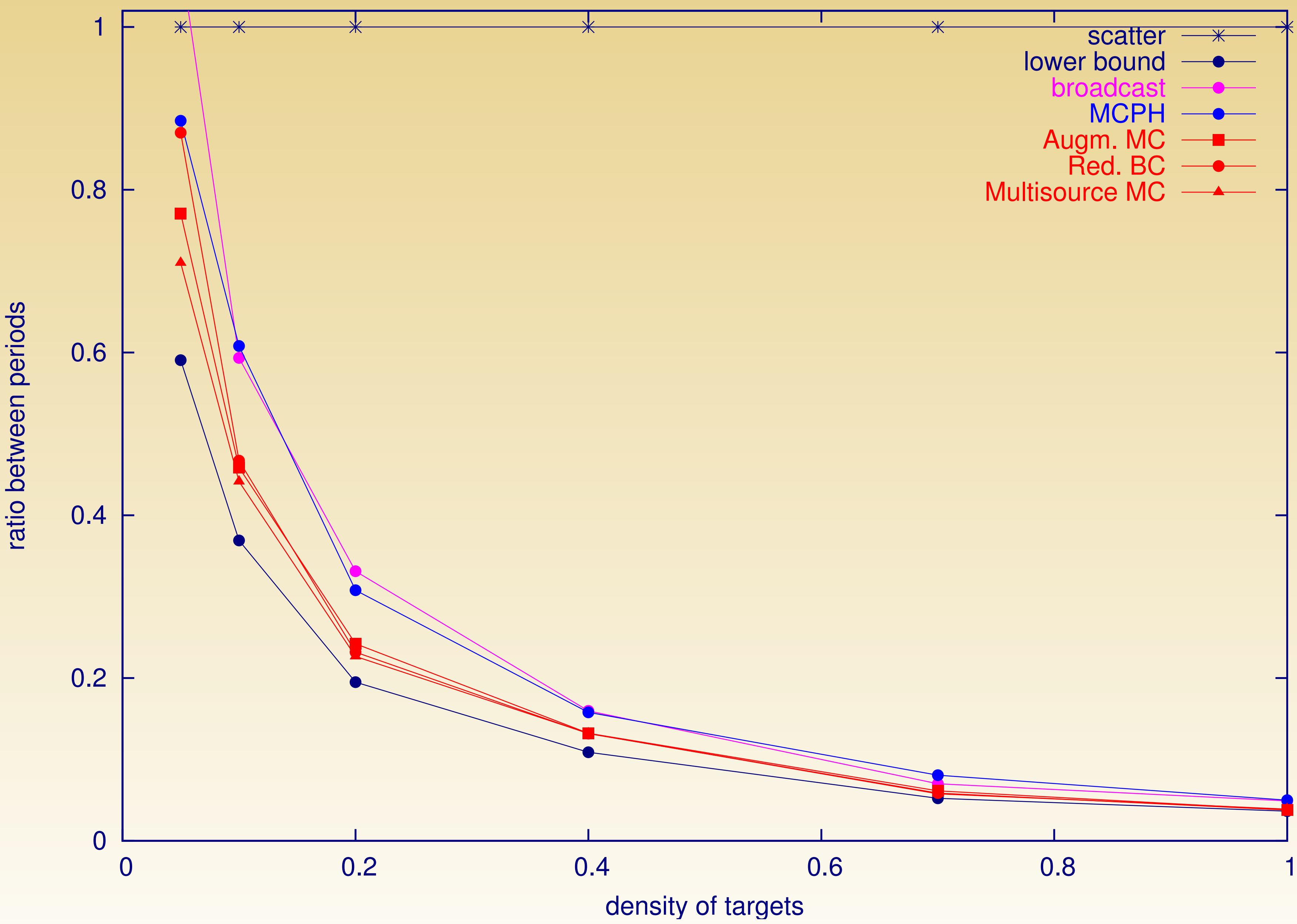
Small platform - comparison over scatter



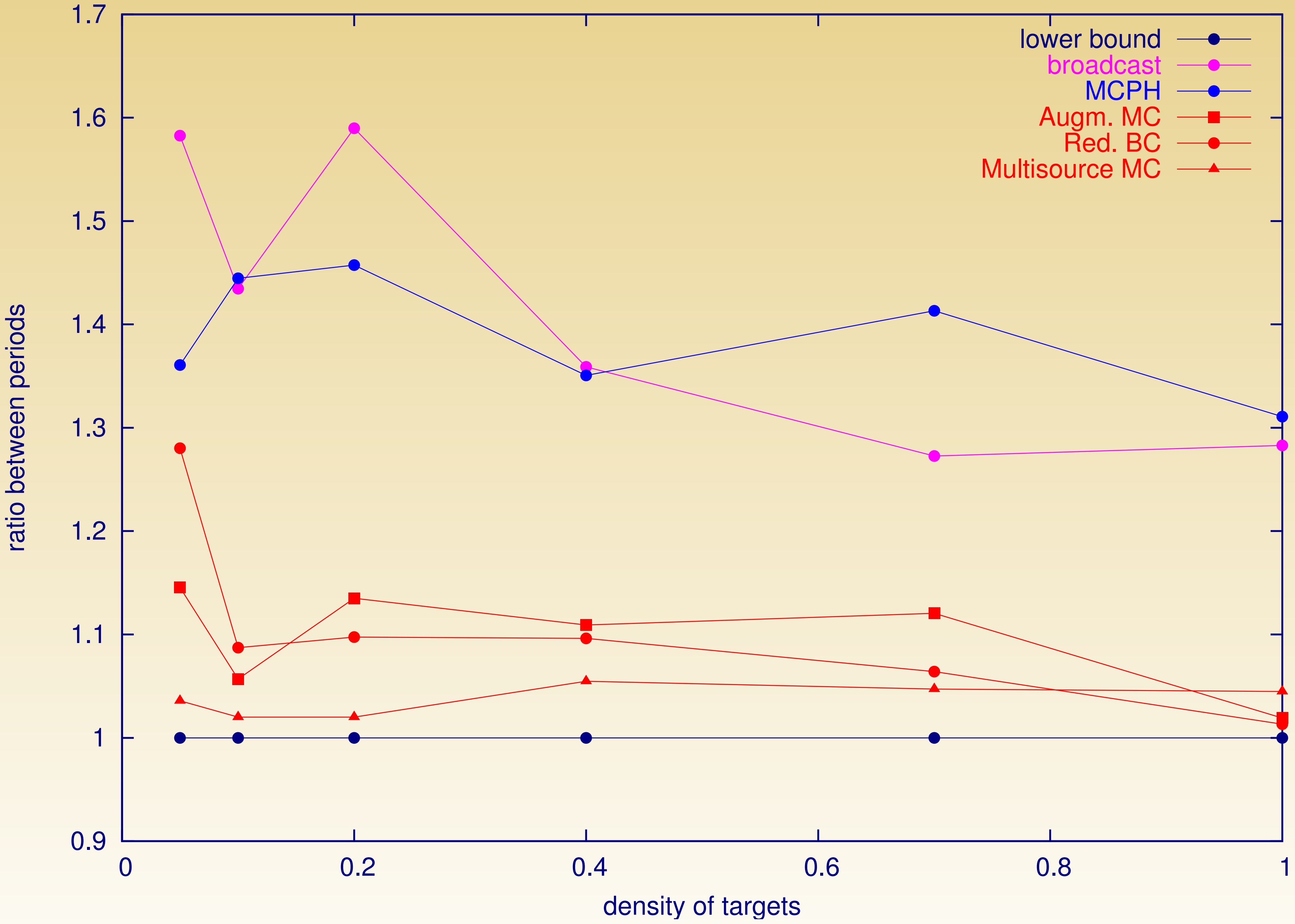
Small platform - comparison the lower bound



Big platform - comparison scatter



Big platform - comparison the lower bound



More on complexity ?

Does Multicast belong to NP?

- we have shown that multicast is NP-hard, but multicast $\in$ NP?
- problem: check that a set of multicast trees is a valid solution
  $\rightarrow$ time linear in the size of the set, potentially in $2^{|V|}$
  $\rightarrow$ check if all communications can be orchestrated
- we prove that at most $2 \times |V|$ of those trees are useful (with throughput $\neq 0$)
- suppose the solution is made of weighted trees (T_i, y_i)
- write the one-port constraints for all the communications involved in these trees

Does Multicast belong to NP?

- we have shown that multicast is NP-hard, but multicast $\in$ NP?
- problem: check that a set of multicast trees is a valid solution
  $\longrightarrow$ time linear in the size of the set, potentially in $2^{|V|}$
  $\longrightarrow$ check if all communications can be orchestrated
- we prove that at most $2 \times |V|$ of those trees are useful (with throughput $\neq 0$)
- suppose the solution is made of weighted trees (T_i, y_i)
- write the one-port constraints for all the communications involved in these trees

Does Multicast belong to NP?

- we have shown that multicast is NP-hard, but multicast $\in$ NP?
- problem: check that a set of multicast trees is a valid solution
  $\longrightarrow$ time linear in the size of the set, potentially in $2^{|V|}$
  $\longrightarrow$ check if all communications can be orchestrated
- we prove that at most $2 \times |V|$ of those trees are useful (with throughput $\neq 0$)
- suppose the solution is made of weighted trees (T_i, y_i)
- write the one-port constraints for all the communications involved in these trees

Does Multicast belong to NP?

- we have shown that multicast is NP-hard, but multicast $\in$ NP?
- problem: check that a set of multicast trees is a valid solution
  $\rightarrow$ time linear in the size of the set, potentially in $2^{|V|}$
  $\rightarrow$ check if all communications can be orchestrated
- we prove that at most $2 \times |V|$ of those trees are useful (with throughput $\neq 0$)
- suppose the solution is made of weighted trees (T_i, y_i)
- write the one-port constraints for all the communications involved in these trees

Does Multicast belong to NP?

- we have shown that multicast is NP-hard, but multicast $\in$ NP?
- problem: check that a set of multicast trees is a valid solution
  - $\rightarrow$ time linear in the size of the set, potentially in $2^{|V|}$
  - $\rightarrow$ check if all communications can be orchestrated
- we prove that at most $2 \times |V|$ of those trees are useful (with throughput $\neq 0$)
- suppose the solution is made of weighted trees (T_i, y_i)
- write the one-port constraints for all the communications involved in these trees

Does Multicast belong to NP?

- we get the following linear program:

$$\textbf{Maximize} \sum_k y_k,$$

$$\textbf{subject to}$$

$$\begin{cases} (1)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{t_k \ni (P_j, P_i)} y_k \cdot c(j,i) \leq 1 \\ (2)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{t_k \ni (P_i, P_j)} y_k \cdot c(i,j) \leq 1 \\ (3)\ \ \forall t_k \in \mathcal{T}, & y_k \geq 0 \end{cases}$$

- one vertex V of the polyhedron is an optimal solution
- V is the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system such that at V, at least $\mathcal{T}$ inequalities (among $\mathcal{T} + 2|V|$) are tight
- at most $2|V|$ tree T_k such that $y_k > 0$
- these weights are solution of the linear system:

$$\log(a_i) \text{ and } \log(b_i) \leq 2|E|(\log(|E|) + \log(\max c(i,j)))$$

Does Multicast belong to NP?

- we get the following linear program:

$$\textbf{Maximize} \sum_k y_k,$$

$$\textbf{subject to}$$

$$\begin{cases} (1)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{t_k \ni (P_j,P_i)} y_k \cdot c(j,i) \leq 1 \\ (2)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{t_k \ni (P_i,P_j)} y_k \cdot c(i,j) \leq 1 \\ (3)\ \ \forall t_k \in \mathcal{T}, & y_k \geq 0 \end{cases}$$

- one vertex V of the polyhedron is an optimal solution
- V is the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system such that at V, at least $\mathcal{T}$ inequalities (among $\mathcal{T} + 2|V|$) are tight
- at most $2|V|$ tree T_k such that $y_k > 0$
- these weights are solution of the linear system:

$$\log(a_i) \text{ and } \log(b_i) \leq 2|E|(\log(|E|) + \log(\max c(i,j)))$$

Does Multicast belong to NP?

- we get the following linear program:

$$\textbf{Maximize} \sum_k y_k,$$

$$\textbf{subject to}$$

$$\begin{cases} (1)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{t_k \ni (P_j,P_i)} y_k \cdot c(j,i) \leq 1 \\ (2)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{t_k \ni (P_i,P_j)} y_k \cdot c(i,j) \leq 1 \\ (3)\ \ \forall t_k \in \mathcal{T}, & y_k \geq 0 \end{cases}$$

- one vertex V of the polyhedron is an optimal solution
- V is the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system such that at V, at least $\mathcal{T}$ inequalities (among $\mathcal{T} + 2|V|$) are tight
- at most $2|V|$ tree T_k such that $y_k > 0$
- these weights are solution of the linear system:

$$\log(a_i) \text{ and } \log(b_i) \leq 2|E|(\log(|E|) + \log(\max c(i,j)))$$

Does Multicast belong to NP?

- we get the following linear program:

$$\textbf{Maximize} \sum_k y_k,$$

$$\textbf{subject to}$$

$$\begin{cases} (1)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{t_k \ni (P_j, P_i)} y_k \cdot c(j,i) \leq 1 \\ (2)\ \ \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{t_k \ni (P_i, P_j)} y_k \cdot c(i,j) \leq 1 \\ (3)\ \ \forall t_k \in \mathcal{T}, & y_k \geq 0 \end{cases}$$

- one vertex V of the polyhedron is an optimal solution
- V is the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system such that at V, at least $\mathcal{T}$ inequalities (among $\mathcal{T} + 2|V|$) are tight
- at most $2|V|$ tree T_k such that $y_k > 0$
- these weights are solution of the linear system:

$$\log(a_i) \text{ and } \log(b_i) \leq 2|E|(\log(|E|) + \log(\max c(i,j)))$$

Does Multicast belong to NP?

- we get the following linear program:

$$\textbf{Maximize} \sum_k y_k,$$

$$\textbf{subject to}$$

$$\begin{cases} (1) \;\; \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \; \sum_{t_k \ni (P_j, P_i)} y_k \cdot c(j,i) \leq 1 \\ (2) \;\; \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \; \sum_{t_k \ni (P_i, P_j)} y_k \cdot c(i,j) \leq 1 \\ (3) \;\; \forall t_k \in \mathcal{T}, & y_k \geq 0 \end{cases}$$

- one vertex V of the polyhedron is an optimal solution
- V is the solution of a $|\mathcal{T}| \times |\mathcal{T}|$ linear system such that at V, at least $\mathcal{T}$ inequalities (among $\mathcal{T} + 2|V|$) are tight
- at most $2|V|$ tree T_k such that $y_k > 0$
- these weights are solution of the linear system:

$$\log(a_i) \text{ and } \log(b_i) \leq 2|E|(\log(|E|) + \log(\max c(i,j)))$$

Does Multicast belong to NP?

- how to orchestrate communications?
- for now:
  - one-port constraints are satisfied
  - weights are polynomial in the size of the platform
- is this enough to build a valid schedule?
- use of a bipartite graph

Does Multicast belong to NP?

- how to orchestrate communications?
- for now:
  - one-port constraints are satisfied
  - weights are polynomial in the size of the platform
- is this enough to build a valid schedule?
- use of a bipartite graph

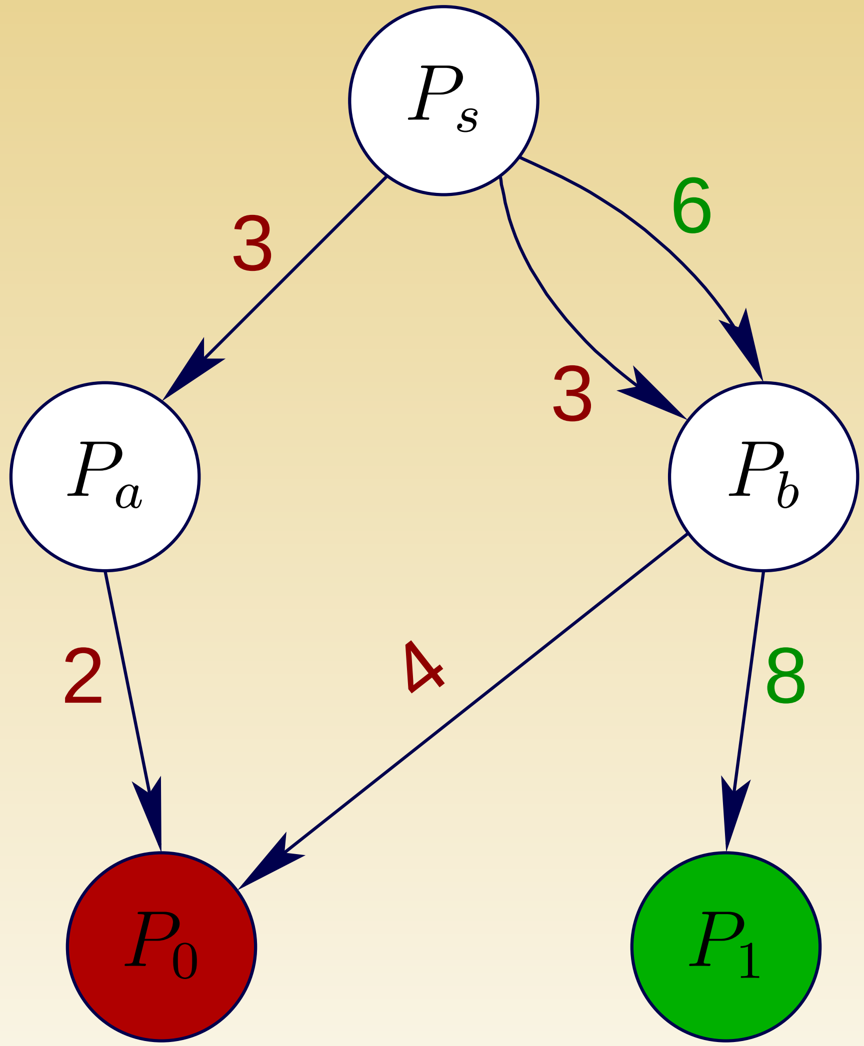
Does Multicast belong to NP?

- how to orchestrate communications?
- for now:
  - one-port constraints are satisfied
  - weights are polynomial in the size of the platform
- is this enough to build a valid schedule?
- use of a bipartite graph

Does Multicast belong to NP?

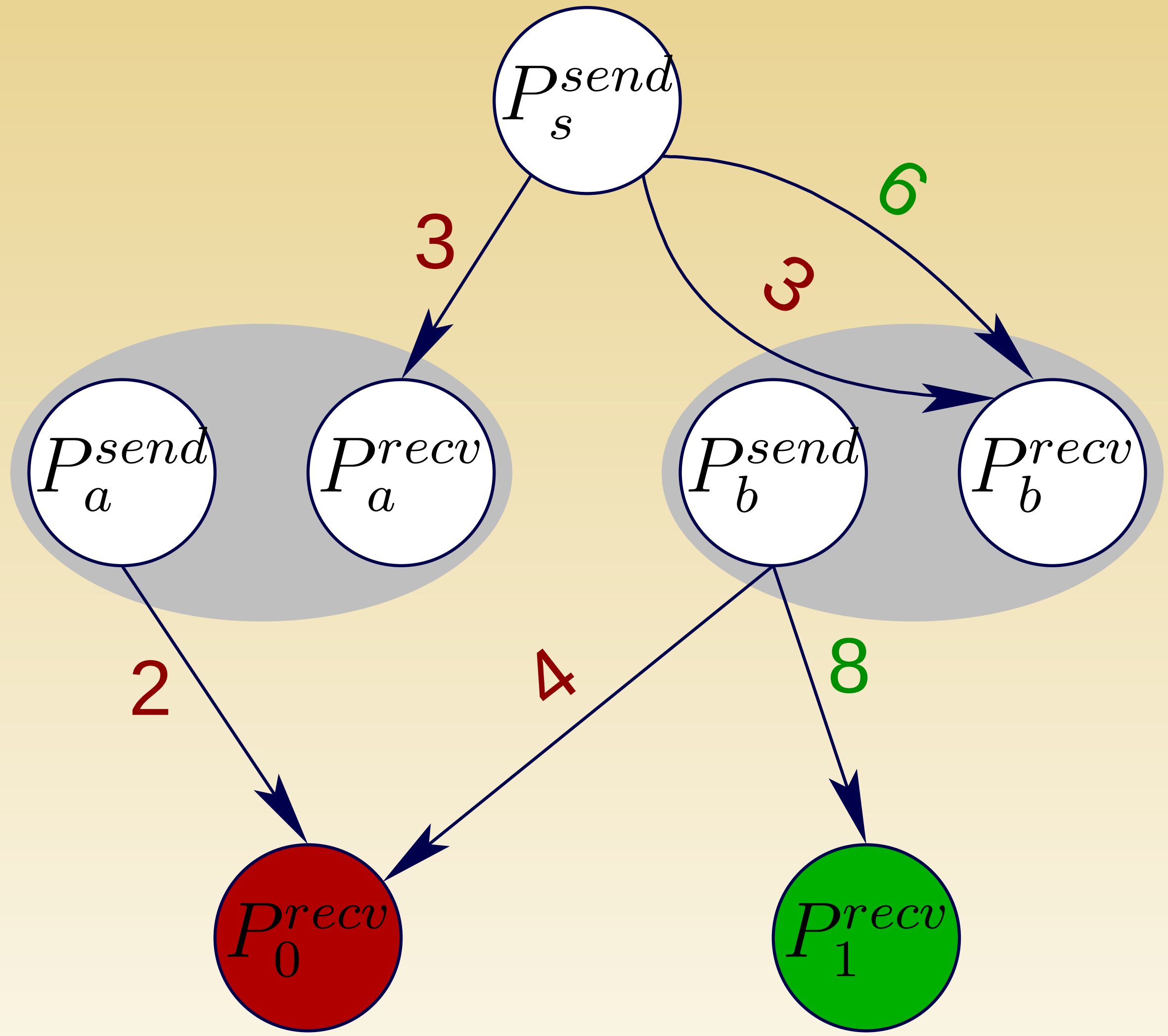
- how to orchestrate communications?
- for now:
  - one-port constraints are satisfied
  - weights are polynomial in the size of the platform
- is this enough to build a valid schedule?
- use of a bipartite graph

Use of a bipartite graph - example



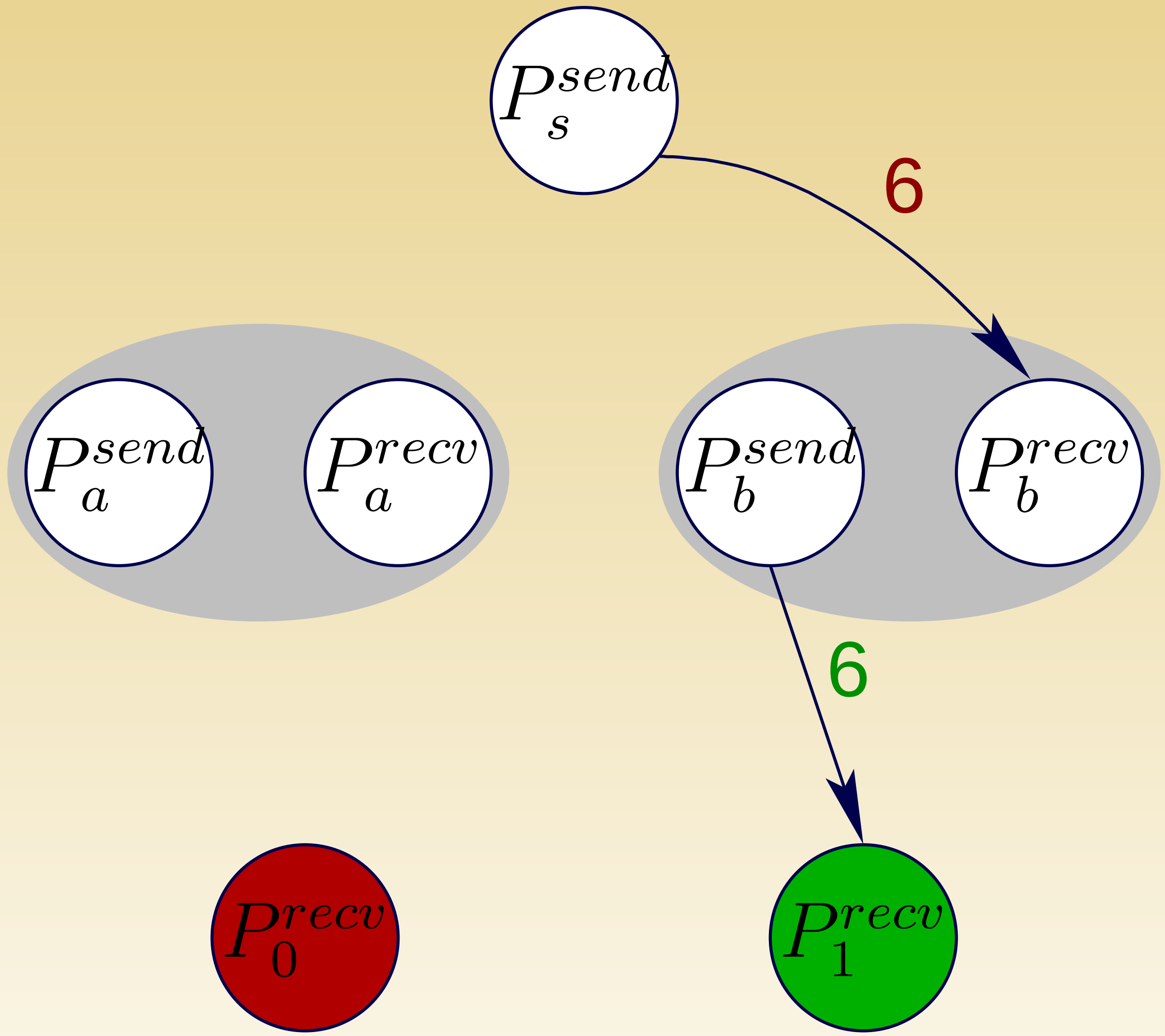
Communications

Use of a bipartite graph - example



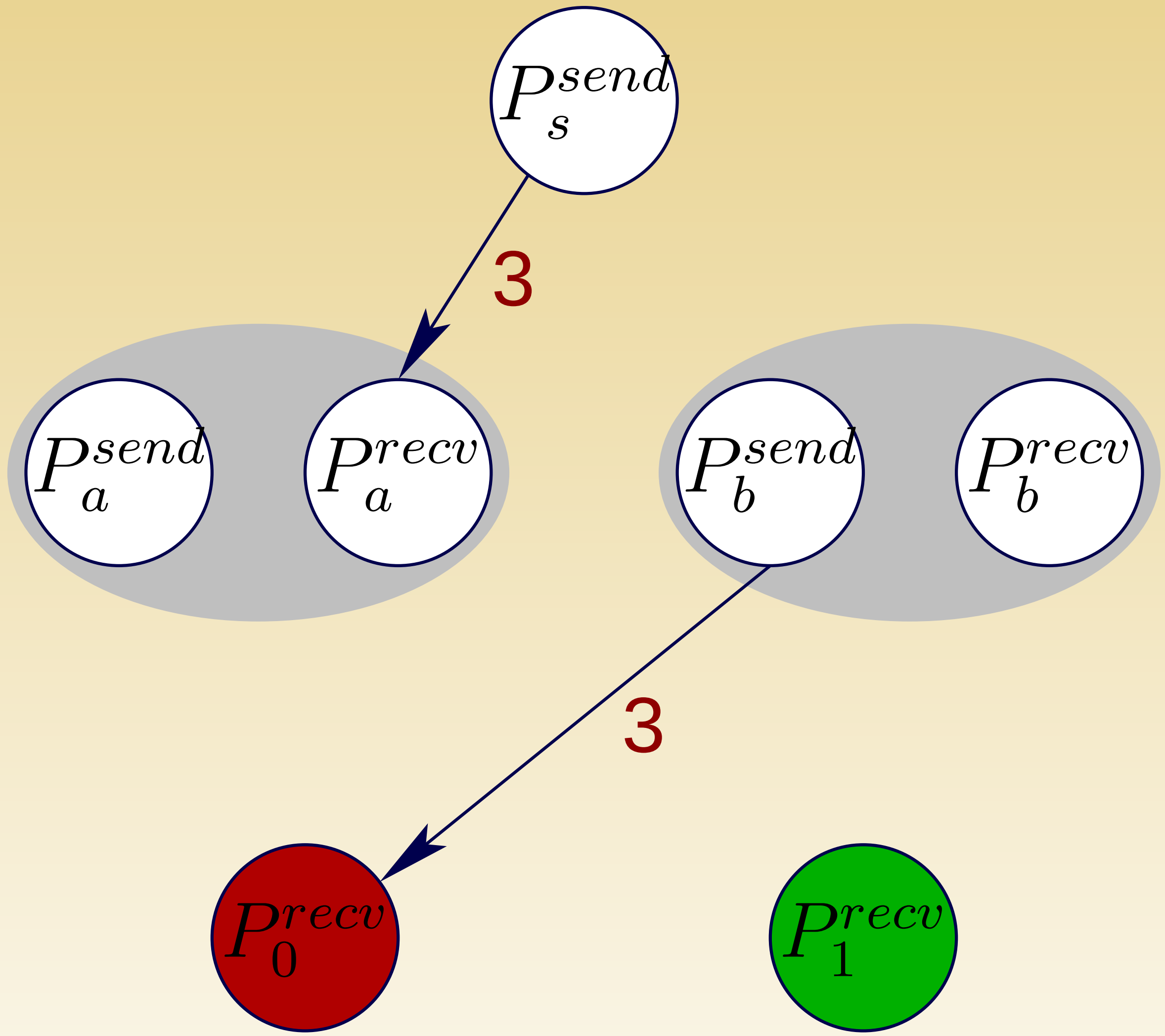
bipartite graph

Use of a bipartite graph - example



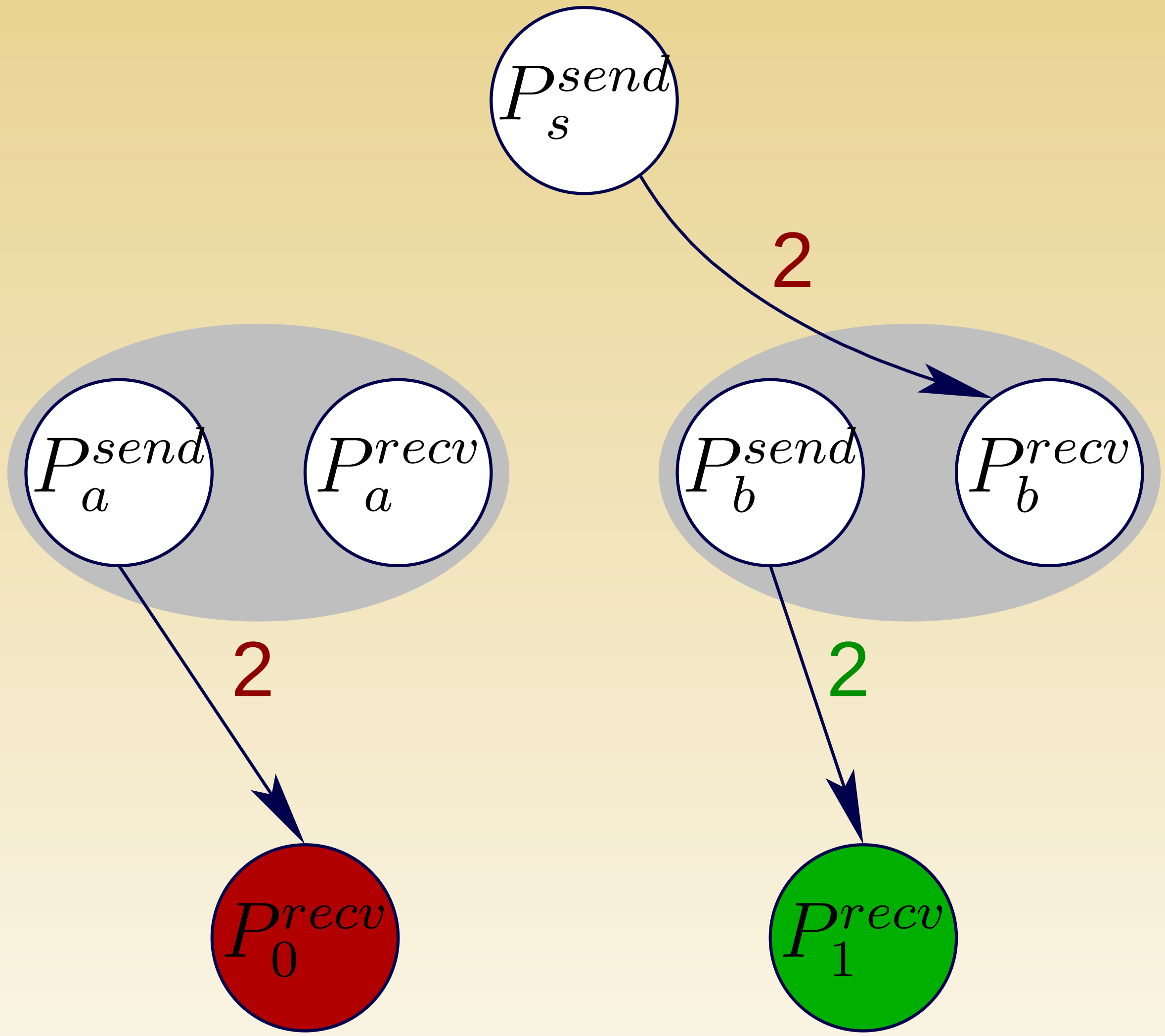
matching 1 (weight 6)

Use of a bipartite graph - example



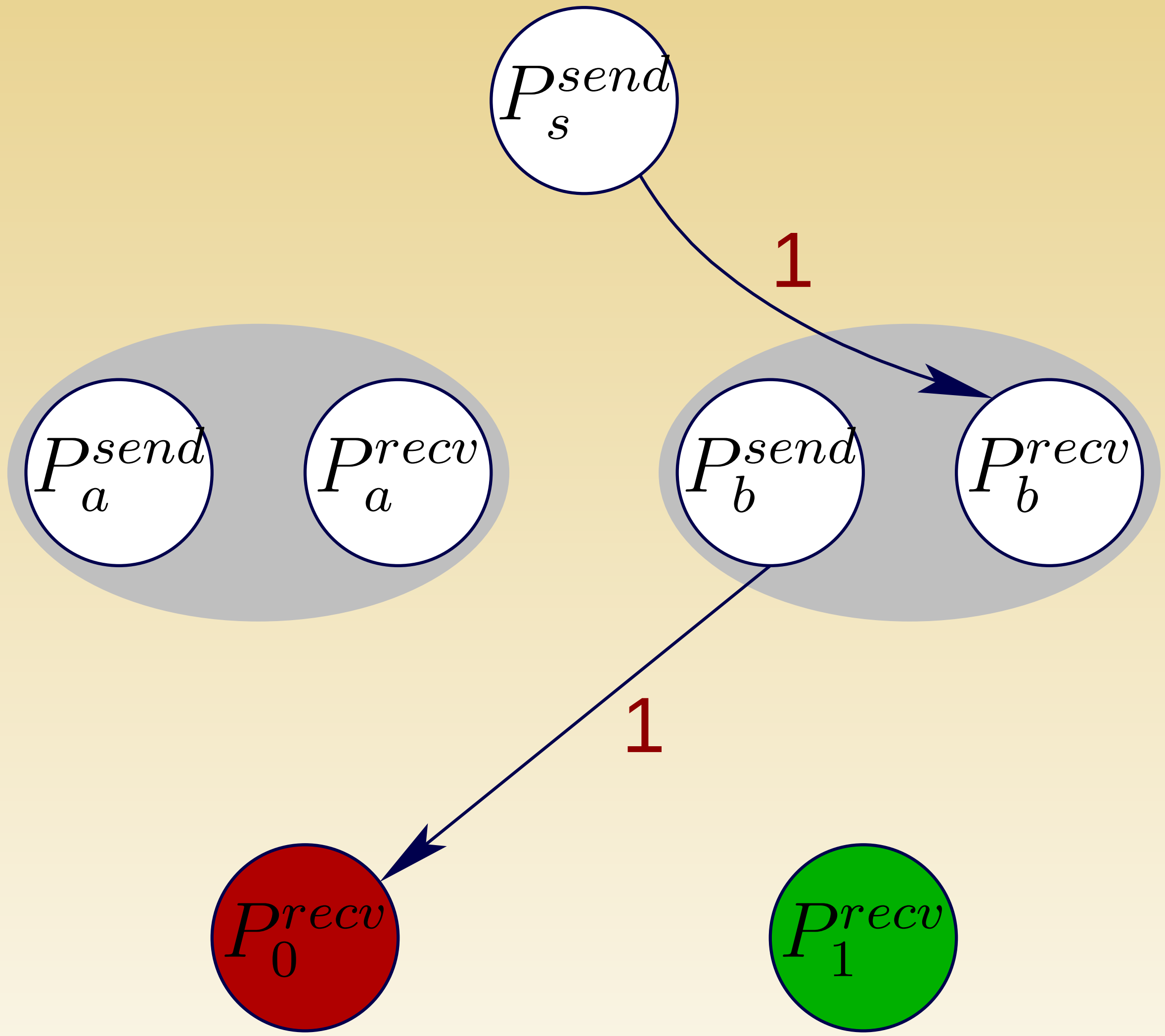
matching 2 (weight 3)

Use of a bipartite graph - example



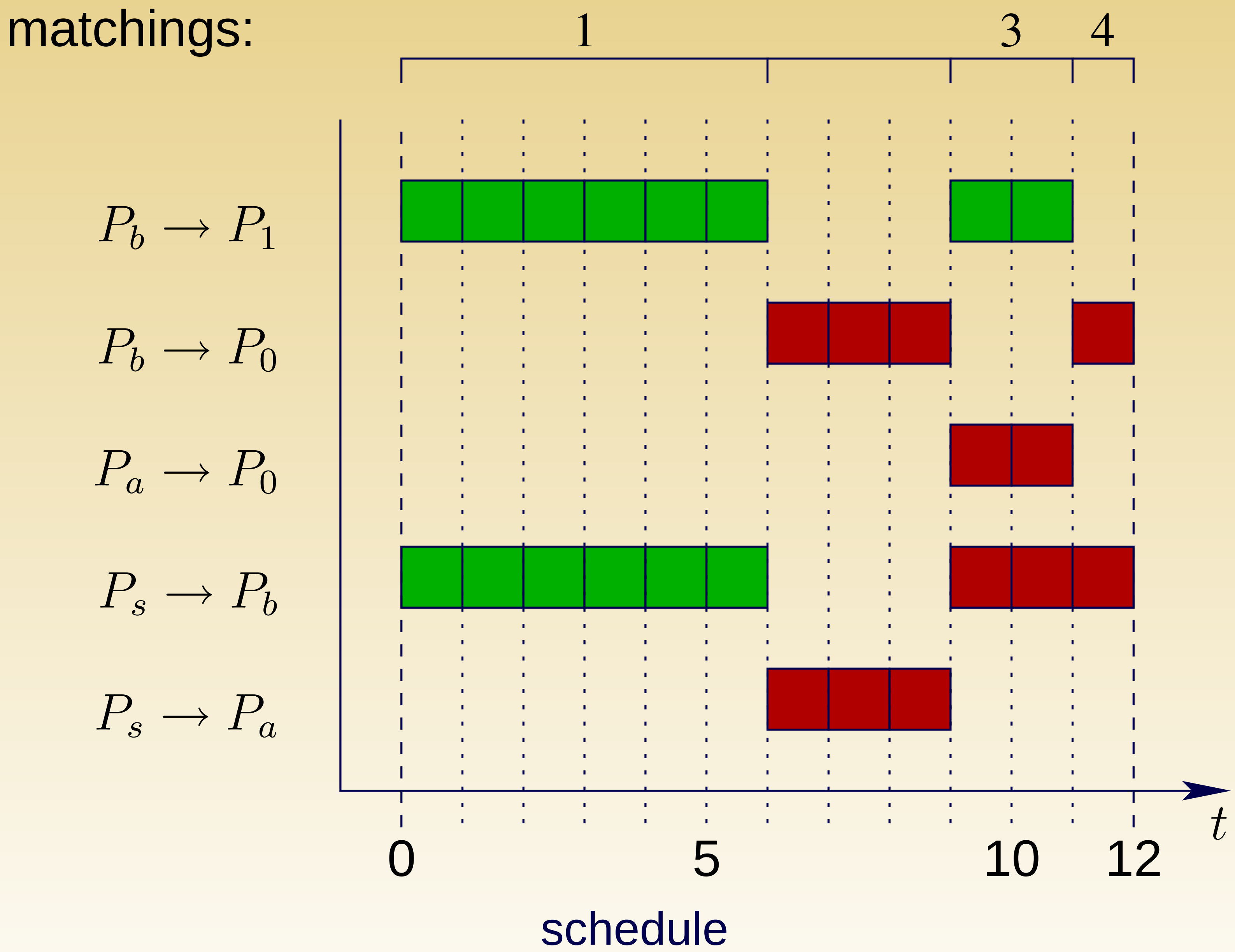
matching 3 (weight 2)

Use of a bipartite graph - example



matching 4 (weight 1)

Use of a bipartite graph - example



Does multicast belong to NP?

- if we restrict the solution to have:
  - a polynomial number of tree
  - a description polynomial in G

  then the problem (COMPACT-WEIGHTED-MULTICAST) is in NP
- this restriction does not affect the optimality of a solution

Does multicast belong to NP?

- if we restrict the solution to have:
  - a polynomial number of tree
  - a description polynomial in G

  then the problem (COMPACT-WEIGHTED-MULTICAST) is in NP
- this restriction does not affect the optimality of a solution

new linear program

- $$\begin{array}{l}
\textbf{Maximize} \sum_k y_k, \\
\textbf{subject to} \\
\left\{\begin{array}{lll}
(1) & \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{in}}(P_i)} \sum_{t_k \ni (P_j, P_i)} y_k \cdot c(j,i) \leq 1 \\
(2) & \forall P_i, & \displaystyle\sum_{P_j \in \mathcal{N}^{\text{out}}(P_i)} \sum_{t_k \ni (P_i, P_j)} y_k \cdot c(i,j) \leq 1 \\
(3) & \forall t_k \in \mathcal{T}, & y_k \geq 0
\end{array}\right.
\end{array}$$

new linear program

- dual:

$$\textbf{Minimize} \sum_i w_i^{in} + \sum_i w_i^{out},$$

$$\textbf{subject to}$$

$$\left\{ (tree) \quad \forall T_k, \qquad \sum_{(i,j)\in T_k} c(i,j) \cdot (w_j^{in} + w_i^{out}) \geq 1 \right.$$

- given an allocation (w_i^{in}, w_i^{out}),
  find a constraint that is not fulfilled
  $\Leftrightarrow$ find a tree, spanning the targets, with minimum weight (aka Steiner)

new linear program

- dual:

$$\textbf{Minimize} \sum_i w_i^{in} + \sum_i w_i^{out},$$

$$\textbf{subject to}$$

$$\left\{ (tree) \quad \forall T_k, \qquad \sum_{(i,j) \in T_k} c(i,j) \cdot (w_j^{in} + w_i^{out}) \geq 1 \right.$$

- given an allocation (w_i^{in}, w_i^{out}),
  find a constraint that is not fulfilled
  $\Leftrightarrow$ find a tree, spanning the targets, with minimum weight (aka Steiner)

new linear program

Grötschel, Lavasz, Schrijver:

- *There exists an oracle-polynomial time algorithm that solves the weak violation problem for every circumscribed convex body* (K, n, R) *given by a weak separation oracle*
  (using the ellipsoid method)
- *There exists an oracle-polynomial time algorithm that solves the weak separation problem for every convex body given by a weak optimization oracle*
- optimal throughput for multicast $\Leftrightarrow$ Minimum Steiner Tree (NP)
- optimal throughput for broadcast $\Leftrightarrow$ Minimum Spanning Tree (P)

new linear program

Grötschel, Lavasz, Schrijver:

- *There exists an oracle-polynomial time algorithm that solves the weak violation problem for every circumscribed convex body* (K, n, R) *given by a weak separation oracle*
  (using the ellipsoid method)
- *There exists an oracle-polynomial time algorithm that solves the weak separation problem for every convex body given by a weak optimization oracle*
- optimal throughput for multicast $\Leftrightarrow$ Minimum Steiner Tree (NP)
- optimal throughput for broadcast $\Leftrightarrow$ Minimum Spanning Tree (P)

new linear program

Grötschel, Lavasz, Schrijver:

- *There exists an oracle-polynomial time algorithm that solves the weak violation problem for every circumscribed convex body* (K, n, R) *given by a weak separation oracle*
  (using the ellipsoid method)
- *There exists an oracle-polynomial time algorithm that solves the weak separation problem for every convex body given by a weak optimization oracle*
- optimal throughput for multicast $\Leftrightarrow$ Minimum Steiner Tree (NP)
- optimal throughput for broadcast $\Leftrightarrow$ Minimum Spanning Tree (P)

new linear program

Grötschel, Lavasz, Schrijver:

- *There exists an oracle-polynomial time algorithm that solves the weak violation problem for every circumscribed convex body* (K, n, R) *given by a weak separation oracle*
  (using the ellipsoid method)
- *There exists an oracle-polynomial time algorithm that solves the weak separation problem for every convex body given by a weak optimization oracle*
- optimal throughput for multicast $\Leftrightarrow$ Minimum Steiner Tree (NP)
- optimal throughput for broadcast $\Leftrightarrow$ Minimum Spanning Tree (P)