An Approximation Algorithm for Network Flow Interdiction with Unit Costs and Two Capacities

Jan Boeckmann and Clemens Thielen

TUM Campus Straubing for Biotechnology and Sustainability
Technical University of Munich

16th September 2020

Supported by:

Bundesministerium für Umwelt, Naturschutz und nukleare Sicherheit

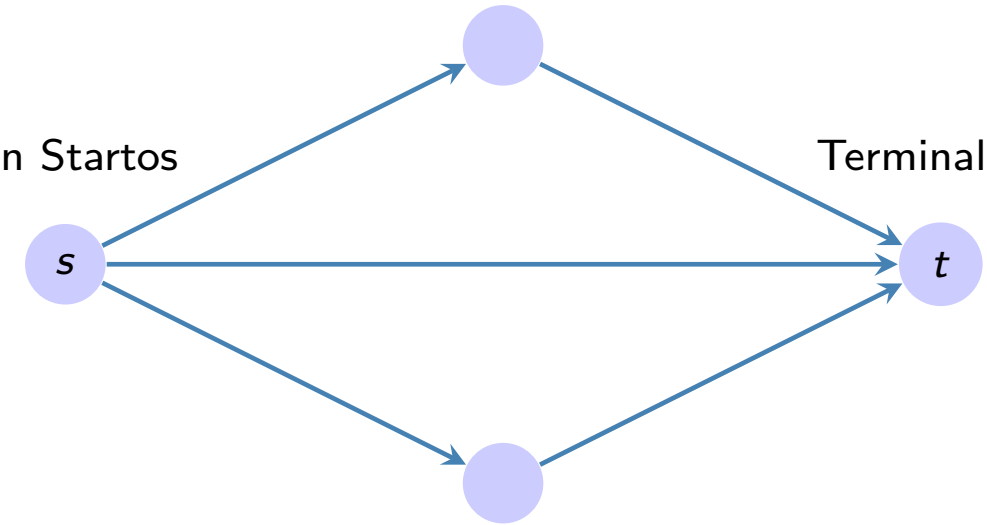
Grant number: 67DAS156C

Motivation

Motivation

Motivation

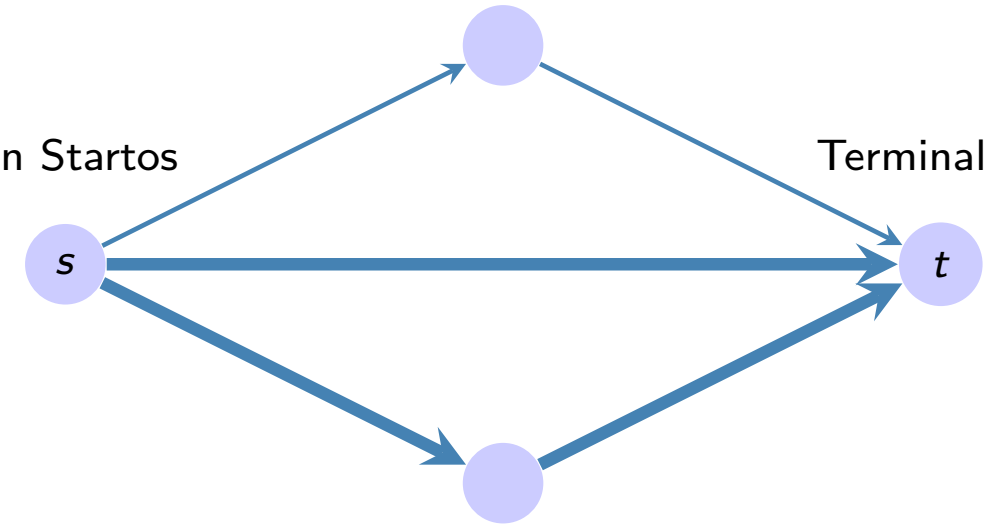
San Startos



Terminal TX

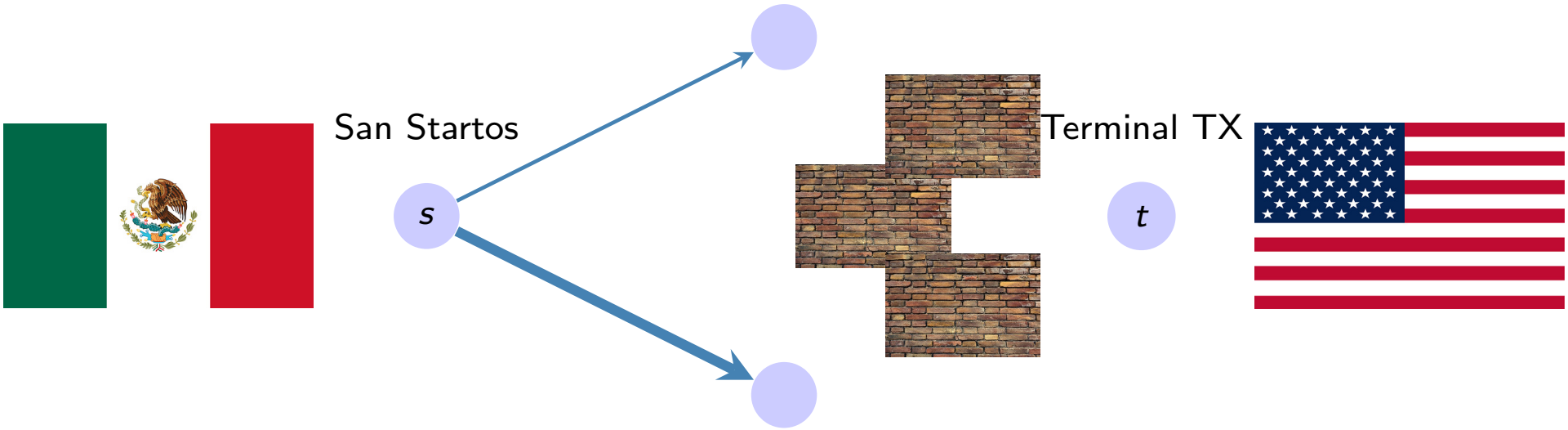
Motivation

San Startos

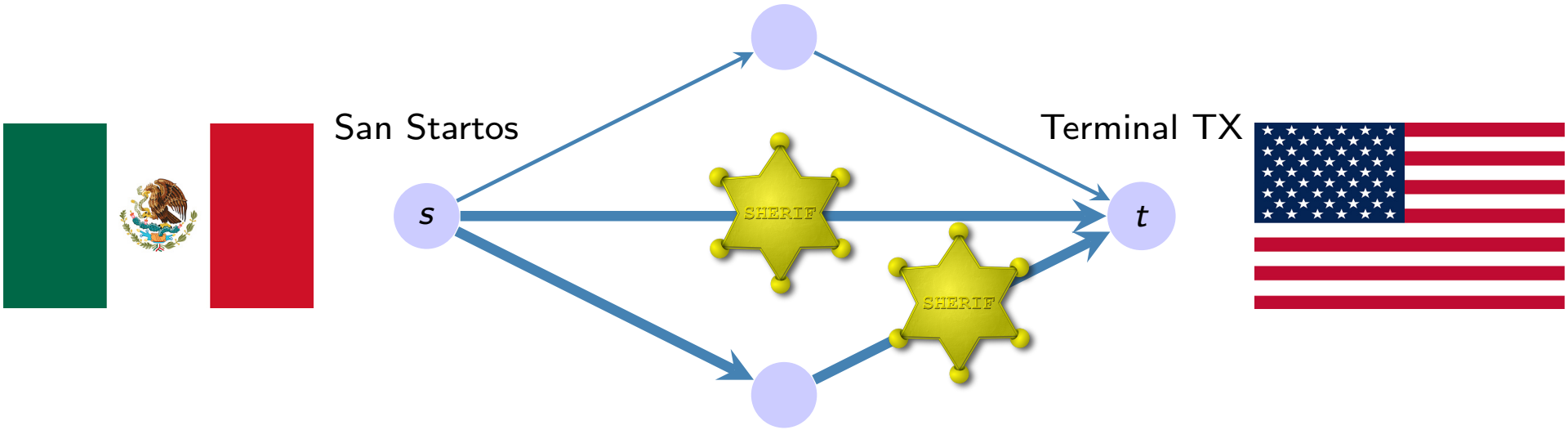


Terminal TX

Motivation



Motivation



Problem Definition

Input

- Directed graph $G = (V, E)$ and two nodes $s \neq t$ in G
- Budget $B \in \mathbb{N}_{>0}$
- Arc capacities $c : E \to \{1, u\}$, where $u \in \mathbb{Q}_{>1}$

Problem Definition – u-NFI

Input

- Directed graph $G = (V, E)$ and two nodes $s \neq t$ in G
- Budget $B \in \mathbb{N}_{>0}$
- Arc capacities $c : E \to \{1, u\}$, where $u \in \mathbb{Q}_{>1}$

Problem Definition – u-NFI

Input

- Directed graph $G = (V, E)$ and two nodes $s \neq t$ in G
- Budget $B \in \mathbb{N}_{>0}$
- Arc capacities $c : E \to \{1, u\}$, where $u \in \mathbb{Q}_{>1}$

Task

Find a removal strategy $R \subseteq E$ of arcs with $|R| \leq B$ such that the value $\mathrm{val}(R)$ of a maximum s-t-flow in the graph $G_R := (V, E \setminus R)$ is minimized.

Problem Definition – u-NFI

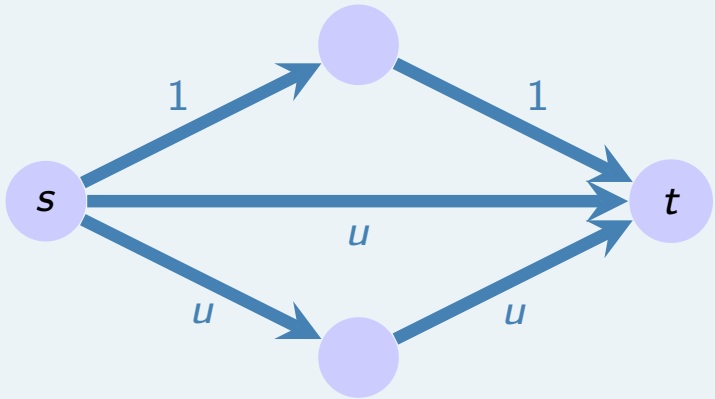
Input

- Directed graph $G = (V, E)$ and two nodes $s \neq t$ in G
- Budget $B \in \mathbb{N}_{>0}$
- Arc capacities $c : E \to \{1, u\}$, where $u \in \mathbb{Q}_{>1}$

Task

Find a removal strategy $R \subseteq E$ of arcs with $|R| \leq B$ such that the value $\text{val}(R)$ of a maximum s-t-flow in the graph $G_R := (V, E \setminus R)$ is minimized.

$B = 2$



Problem Definition – u-NFI

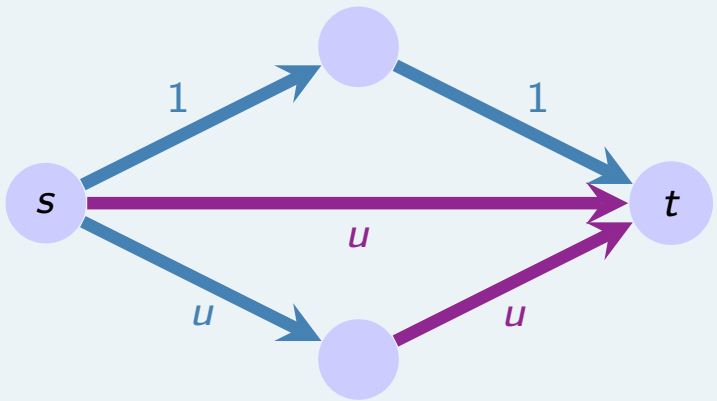
Input

- Directed graph $G = (V, E)$ and two nodes $s \neq t$ in G
- Budget $B \in \mathbb{N}_{>0}$
- Arc capacities $c : E \to \{1, u\}$, where $u \in \mathbb{Q}_{>1}$

Task

Find a removal strategy $R \subseteq E$ of arcs with $|R| \leq B$ such that the value $\mathrm{val}(R)$ of a maximum s-t-flow in the graph $G_R := (V, E \setminus R)$ is minimized.

$B = 2$



Problem Definition – u-NFI

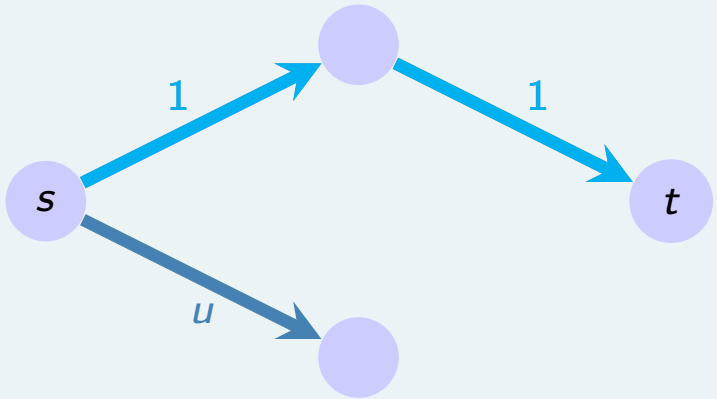
Input

- Directed graph $G = (V, E)$ and two nodes $s \neq t$ in G
- Budget $B \in \mathbb{N}_{>0}$
- Arc capacities $c : E \to \{1, u\}$, where $u \in \mathbb{Q}_{>1}$

Task

Find a removal strategy $R \subseteq E$ of arcs with $|R| \leq B$ such that the value $\mathrm{val}(R)$ of a maximum s-t-flow in the graph $G_R := (V, E \setminus R)$ is minimized.

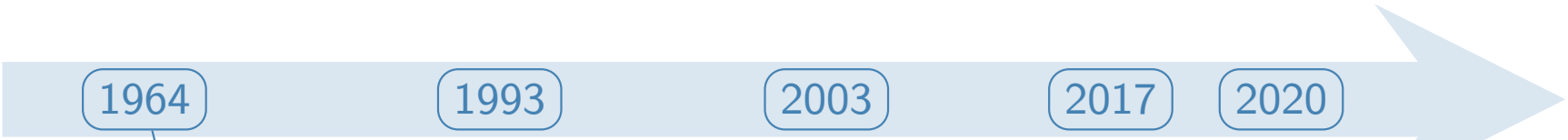
$B = 2$



$\mathrm{val}(R) = 1$

Previous Work

For General Version of NFI with Arbitrary Capacities and Removal Costs

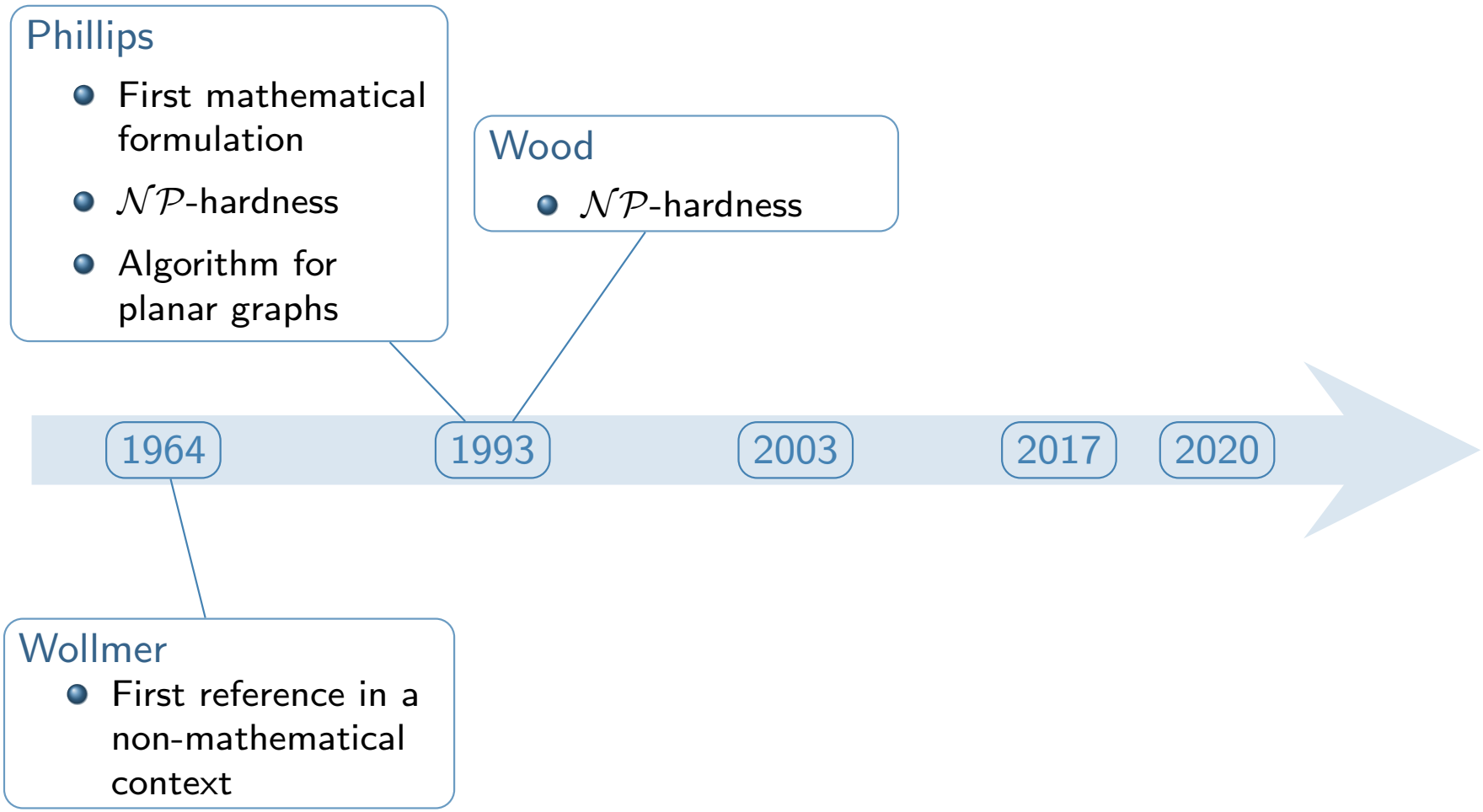


Wollmer

- First reference in a non-mathematical context

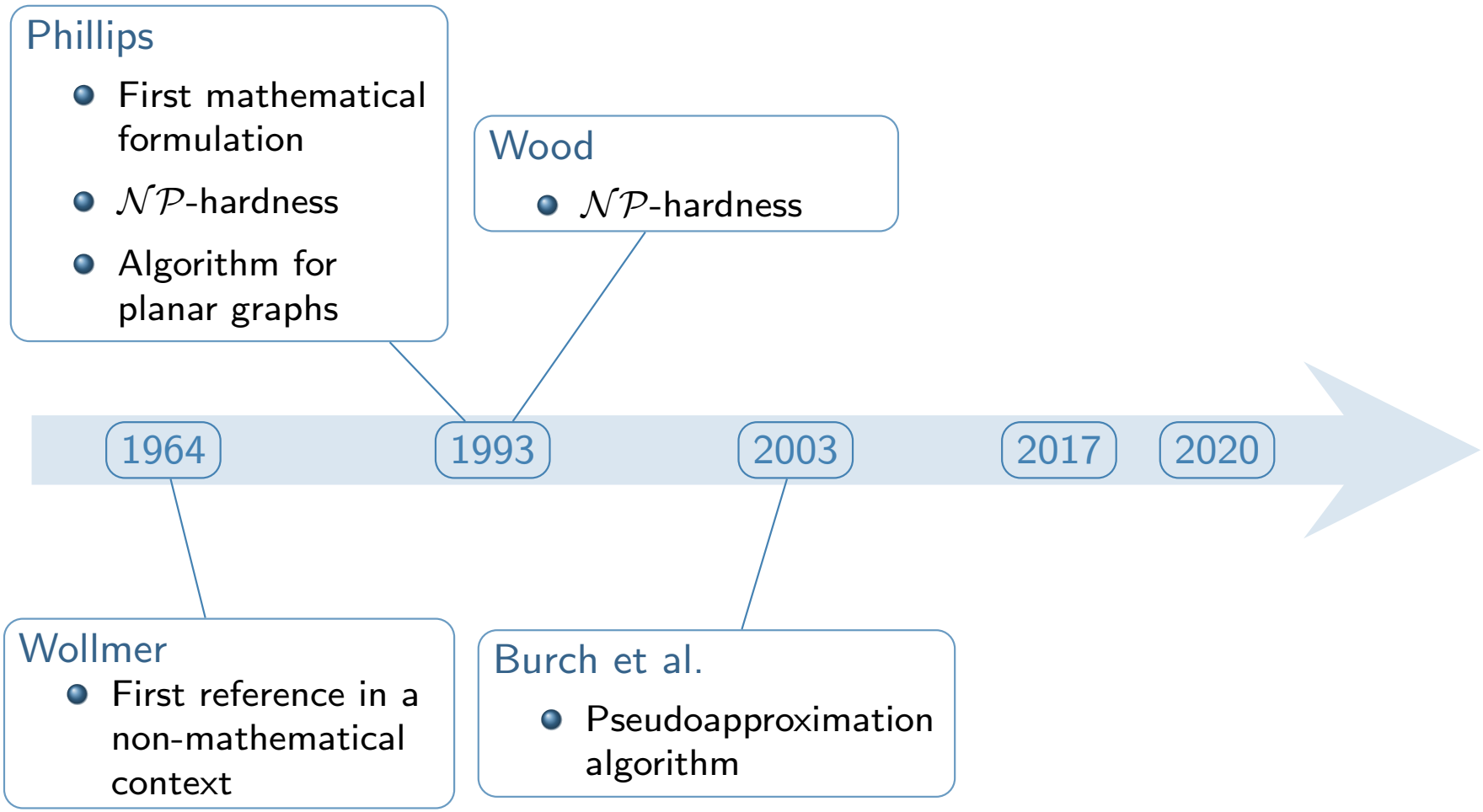
Previous Work

For General Version of NFI with Arbitrary Capacities and Removal Costs



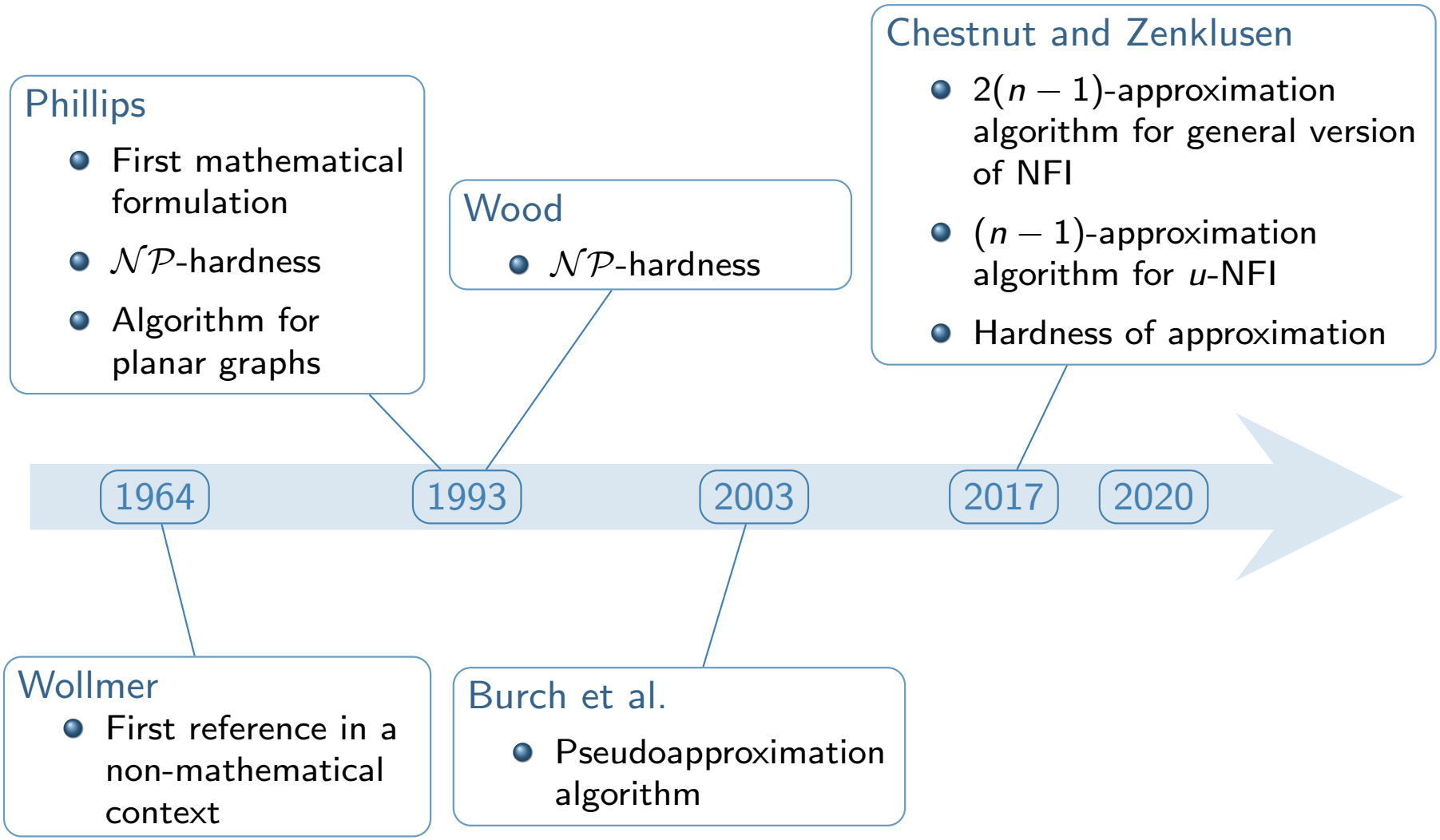
Previous Work

For General Version of NFI with Arbitrary Capacities and Removal Costs



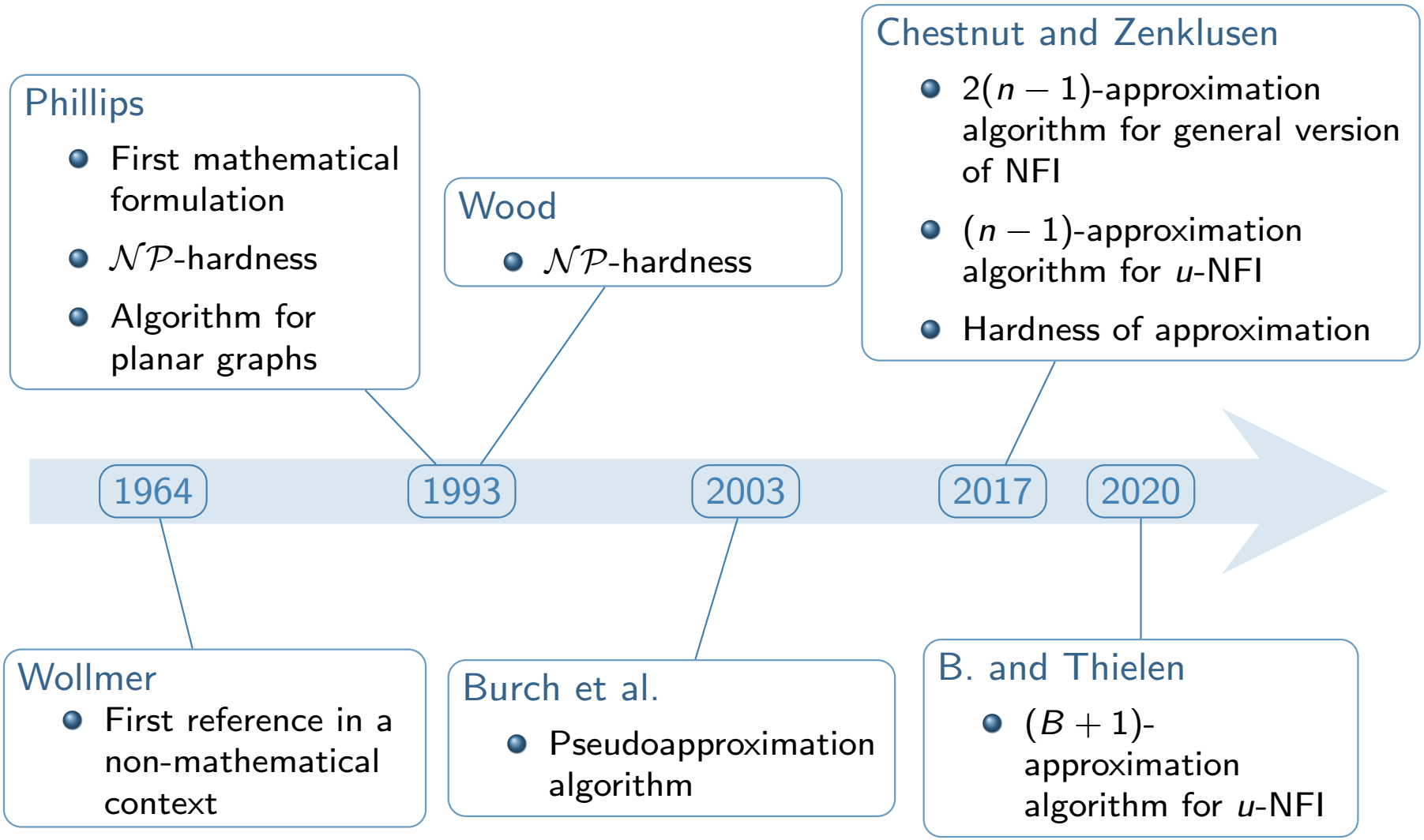
Previous Work

For General Version of NFI with Arbitrary Capacities and Removal Costs



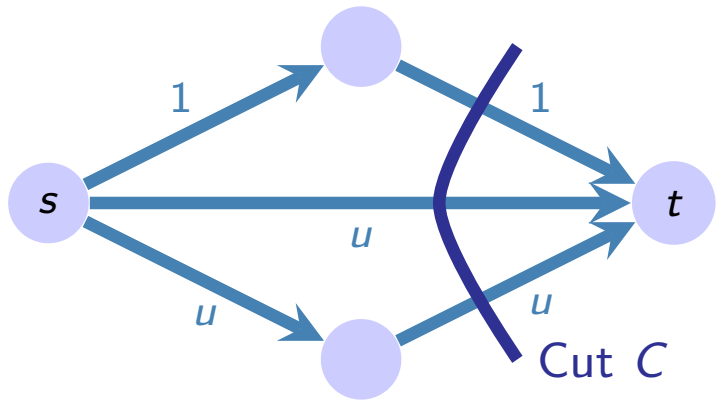
Previous Work

For General Version of NFI with Arbitrary Capacities and Removal Costs



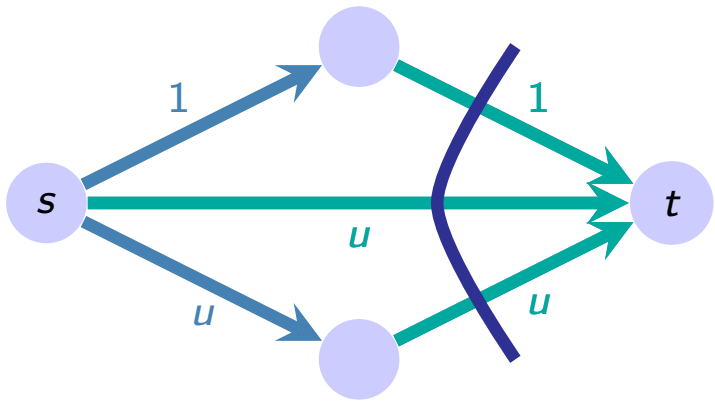
Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.



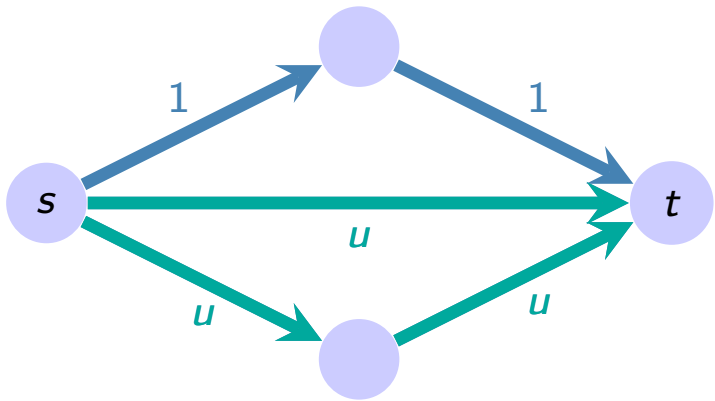
Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$.



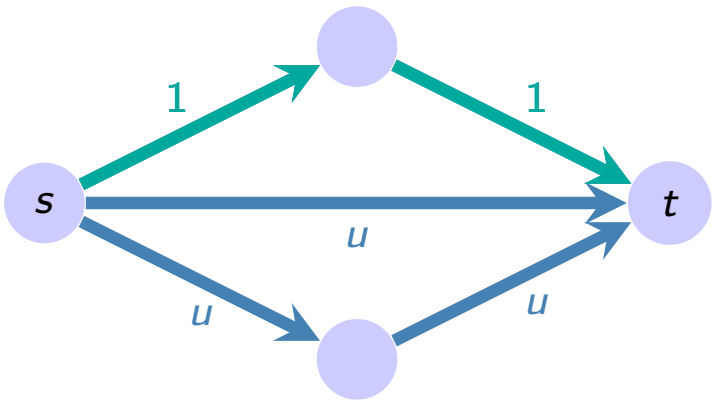
Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$.
- Arcs having capacity u are called large arcs.



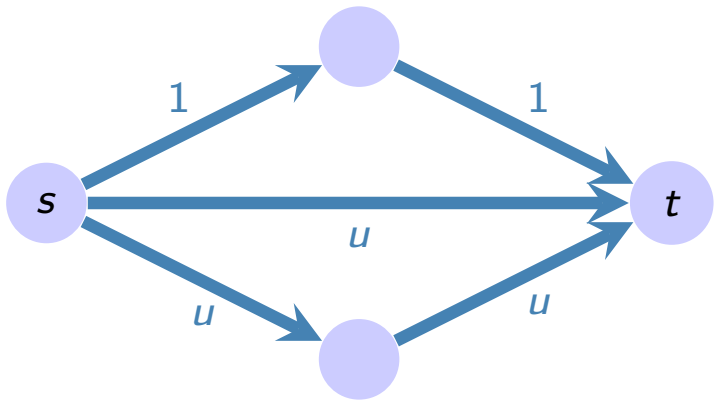
Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$.
- Arcs having capacity u are called large arcs.
- Arcs having capacity 1 are called small arcs.



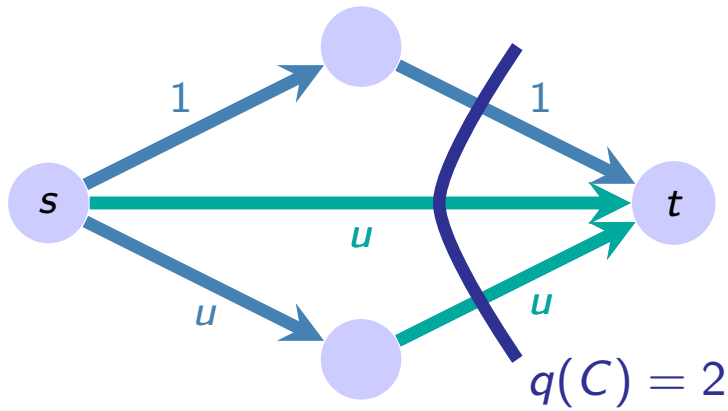
Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$.
- Arcs having capacity u are called large arcs.
- Arcs having capacity 1 are called small arcs.
- $q(C) \coloneqq \#(\text{large arcs in } C)$
- $p(C) \coloneqq \#(\text{small arcs in } C)$



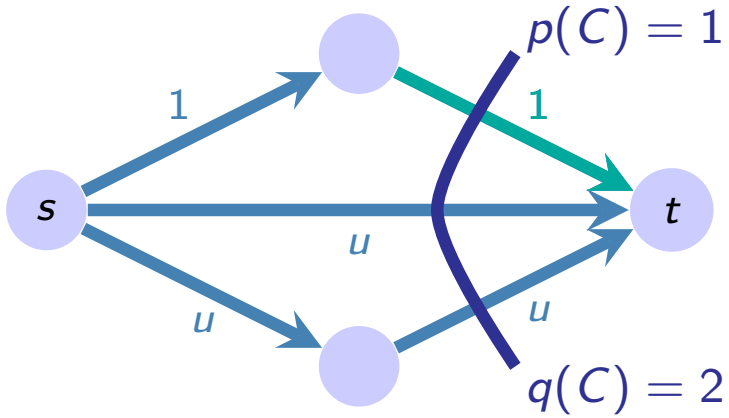
Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$.
- Arcs having capacity u are called large arcs.
- Arcs having capacity 1 are called small arcs.
- $q(C) \coloneqq \#(\text{large arcs in } C)$
- $p(C) \coloneqq \#(\text{small arcs in } C)$



Notation

- A cut or s-t-cut is a partition $V = S \dot{\cup} T$ of the nodes of G such that $s \in S$ and $t \in T$.
- An arc $r \in E$ is in a cut $C = (S, T)$ if $\alpha(r) \in S$ and $\omega(r) \in T$.
- Arcs having capacity u are called large arcs.
- Arcs having capacity 1 are called small arcs.
- $q(C) \coloneqq \#(\text{large arcs in } C)$
- $p(C) \coloneqq \#(\text{small arcs in } C)$



Removal Strategies and Cuts – Part 1

Removal Strategies and Cuts – Part 1

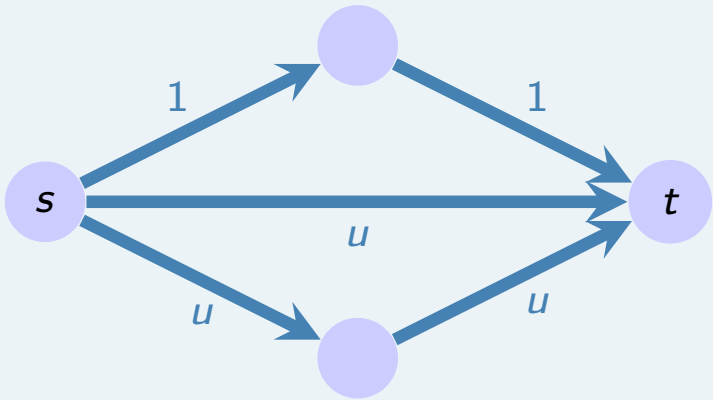
Lemma 1

For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\text{val}(R)$ can be reduced by a simple exchange argument.

Removal Strategies and Cuts – Part 1

Lemma 1

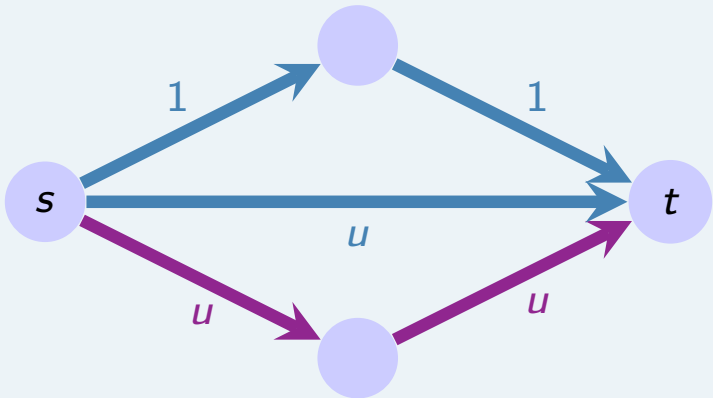
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\text{val}(R)$ can be reduced by a simple exchange argument.



Removal Strategies and Cuts – Part 1

Lemma 1

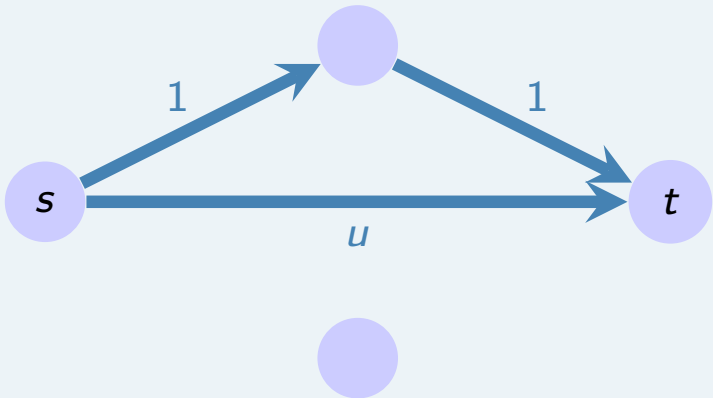
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\mathrm{val}(R)$ can be reduced by a simple exchange argument.



Removal Strategies and Cuts – Part 1

Lemma 1

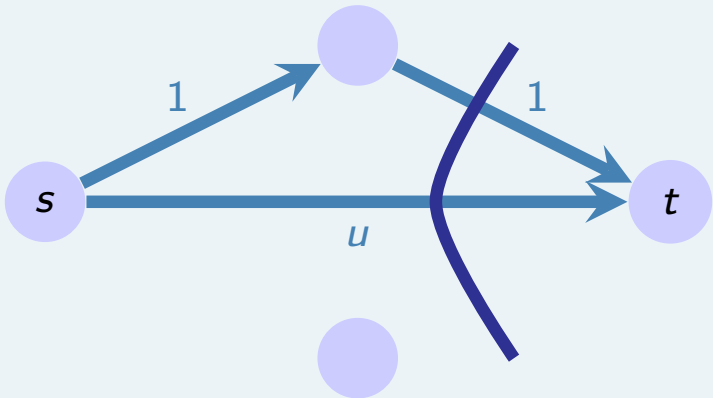
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\text{val}(R)$ can be reduced by a simple exchange argument.



Removal Strategies and Cuts – Part 1

Lemma 1

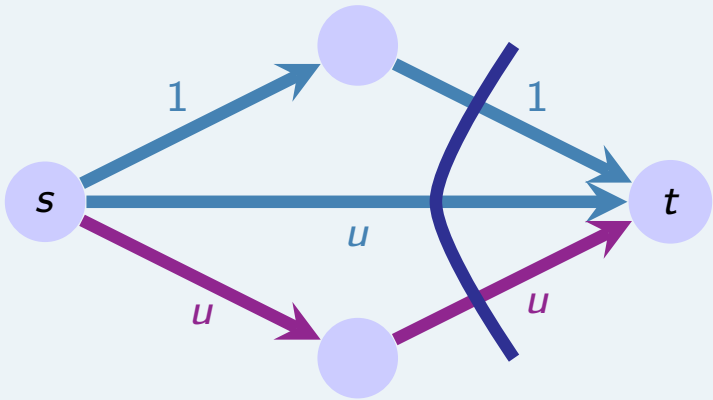
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\mathrm{val}(R)$ can be reduced by a simple exchange argument.



Removal Strategies and Cuts – Part 1

Lemma 1

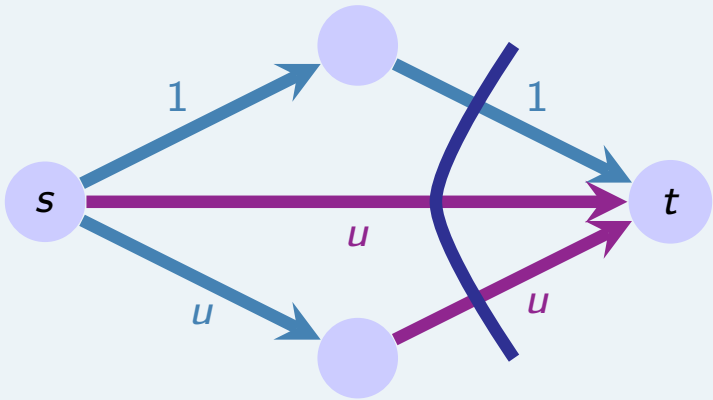
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\mathrm{val}(R)$ can be reduced by a simple exchange argument.



Removal Strategies and Cuts – Part 1

Lemma 1

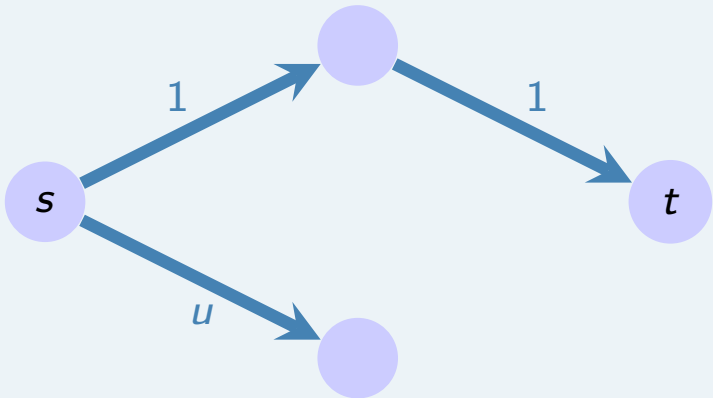
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\text{val}(R)$ can be reduced by a simple exchange argument.



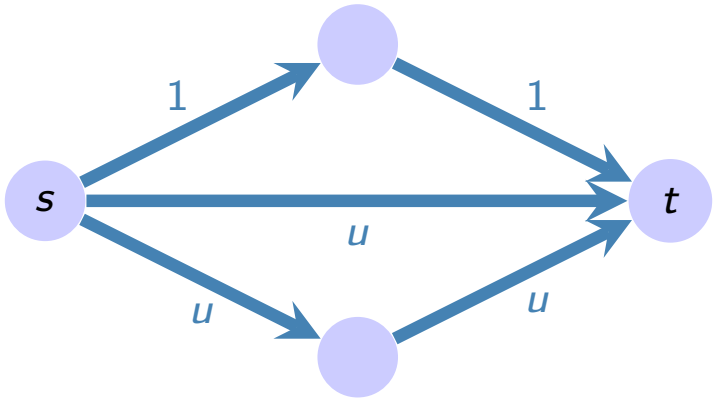
Removal Strategies and Cuts – Part 1

Lemma 1

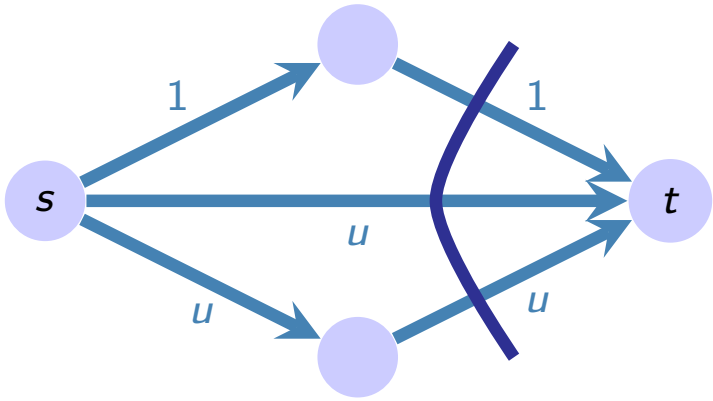
For a solution $R \subseteq E$, either all arcs in R are in one cut, or $\text{val}(R)$ can be reduced by a simple exchange argument.



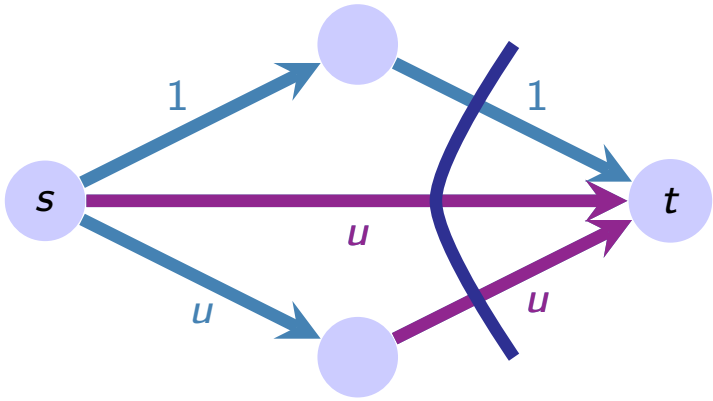
Removal Strategies and Cuts – Part 2



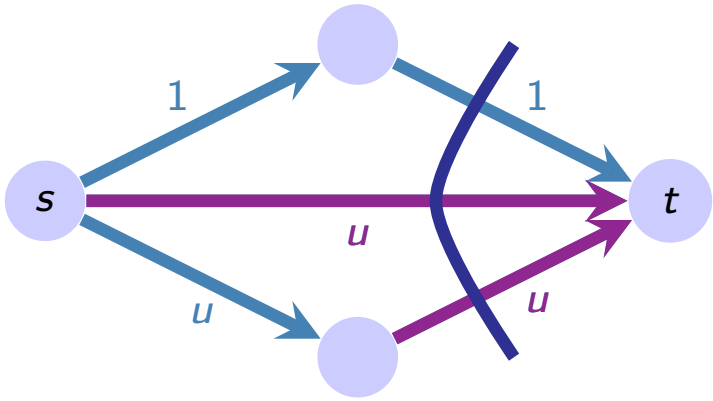
Removal Strategies and Cuts – Part 2



Removal Strategies and Cuts – Part 2



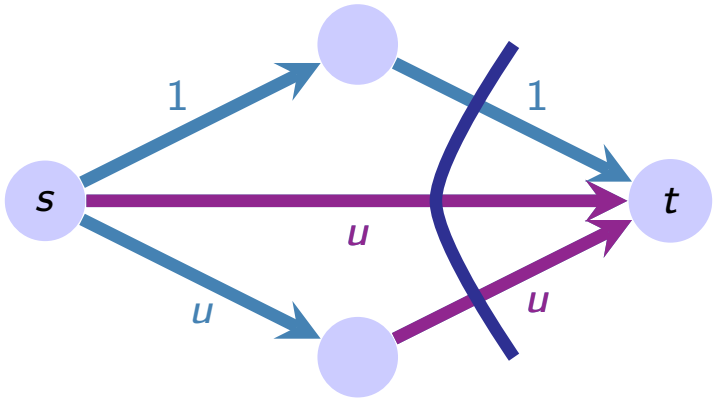
Removal Strategies and Cuts – Part 2



Observation 1

Let C be a cut in G and let $R(C)$ be the set of the B largest arcs in C. If C is a minimum cut in $G_{R(C)}$, then $R(C)$ is a best removal strategy amongst all strategies removing only arcs from C.

Removal Strategies and Cuts – Part 2



Observation 1

Let C be a cut in G and let $R(C)$ be the set of the B largest arcs in C. If C is a minimum cut in $G_{R(C)}$, then $R(C)$ is a best removal strategy amongst all strategies removing only arcs from C.

In the following, write $\text{val}(C)$ instead of $\text{val}(R(C))$.

Attacking a Minimum Cut

Attacking a Minimum Cut

Lemma 2

Let C_m denote a minimum cut in G. Then C_m is optimal if it contains at least B large arcs.

Attacking a Minimum Cut

Lemma 2

Let C_m denote a minimum cut in G. Then C_m is optimal if it contains at least B large arcs.

Proof

$$\text{val}(C_m) = \text{capacity of arcs in the cut} - \text{capacity of removed arcs}$$

Attacking a Minimum Cut

Lemma 2

Let C_m denote a minimum cut in G. Then C_m is optimal if it contains at least B large arcs.

Proof

$$\text{val}(C_m) = \text{capacity of arcs in the cut} - \text{capacity of removed arcs}$$

Attacking a Minimum Cut

Lemma 2

Let C_m denote a minimum cut in G. Then C_m is optimal if it contains at least B large arcs.

Proof

$$\text{val}(C_m) = \text{capacity of arcs in the cut} - \text{capacity of removed arcs}$$

Attacking a Minimum Cut

Lemma 2

Let C_m denote a minimum cut in G. Then C_m is optimal if it contains at least B large arcs.

Proof

$$\text{val}(C_m) = \text{capacity of arcs in the cut} - \text{capacity of removed arcs}$$

Can assume in the following that any minimum cut in G contains at most $B - 1$ large arcs

Attacking a Least Cut

Lemma 3

Let C_l denote a least cut in G, i.e., a cut with least possible number of arcs. Then C_l is optimal if it contains at most B large arcs.

Attacking a Least Cut

Lemma 3

Let C_l denote a least cut in G, i.e., a cut with least possible number of arcs. Then C_l is optimal if it contains at most B large arcs.

Proof

After removal, C_l will have

Attacking a Least Cut

Lemma 3

Let C_l denote a least cut in G, i.e., a cut with least possible number of arcs. Then C_l is optimal if it contains at most B large arcs.

Proof

After removal, C_l will have

- the least possible number of remaining arcs and

Attacking a Least Cut

Lemma 3

Let C_l denote a least cut in G, i.e., a cut with least possible number of arcs. Then C_l is optimal if it contains at most B large arcs.

Proof

After removal, C_l will have

- the least possible number of remaining arcs and
- all of those remaining arcs will be small arcs (least possible capacity)

Attacking a Least Cut

Lemma 3

Let C_l denote a least cut in G, i.e., a cut with least possible number of arcs. Then C_l is optimal if it contains at most B large arcs.

Proof

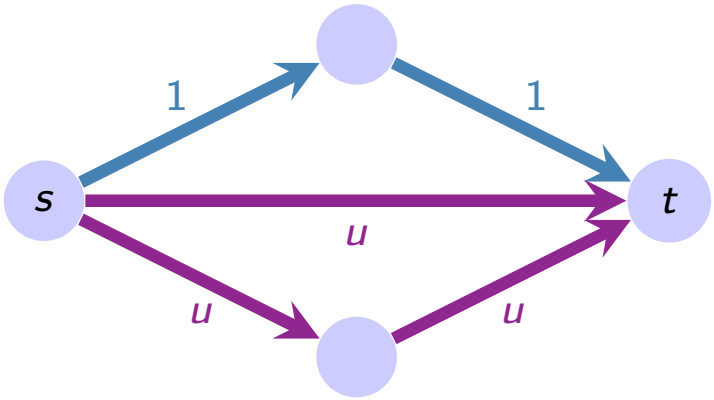
After removal, C_l will have

- the least possible number of remaining arcs and
- all of those remaining arcs will be small arcs (least possible capacity)

Can assume in the following that any least cut in G contains at least $B + 1$ large arcs

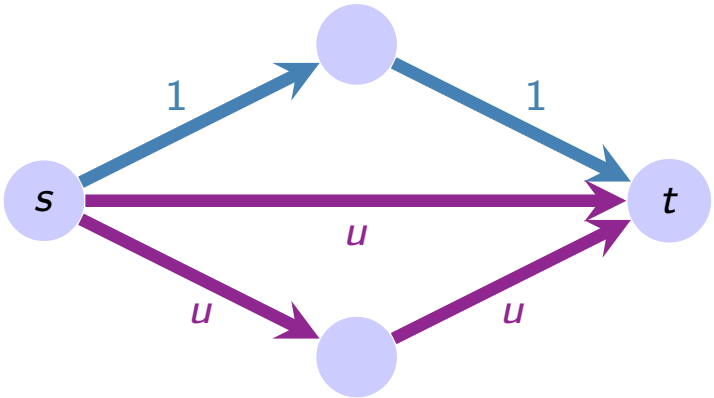
Modifying the Graph

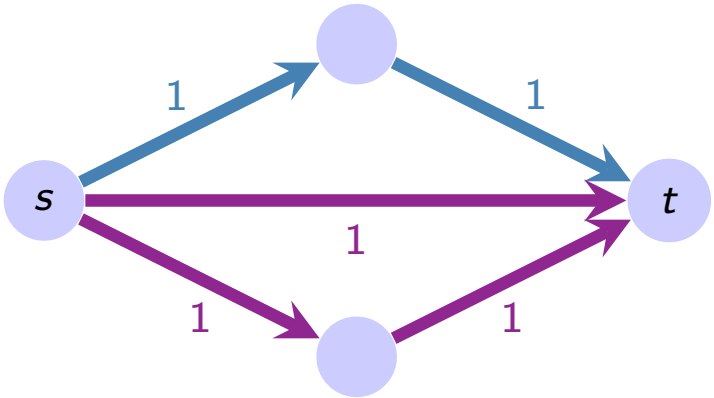
What we already discussed



Modifying the Graph

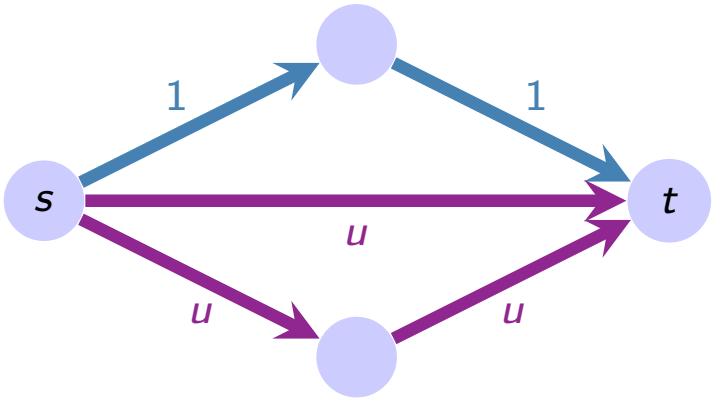
What we already discussed

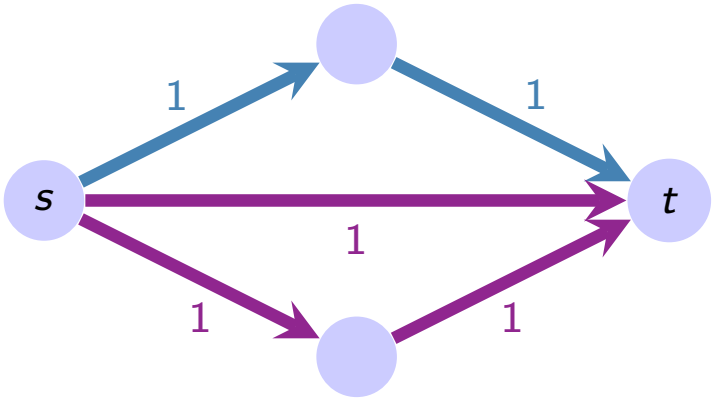




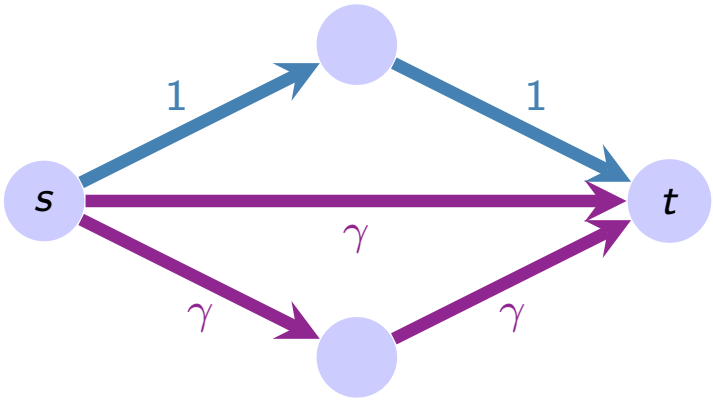
Modifying the Graph

What we already discussed





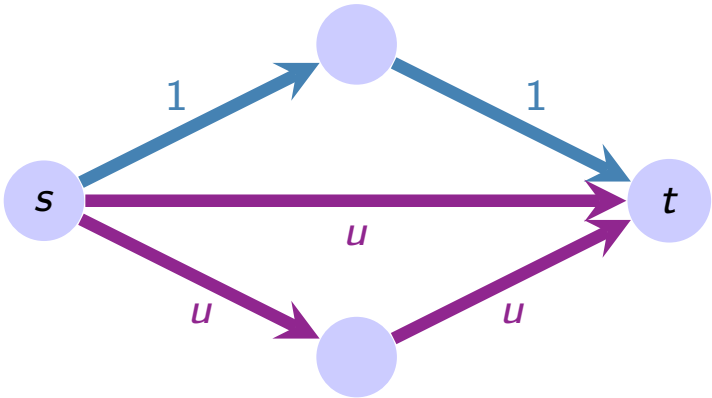
Where we proceed

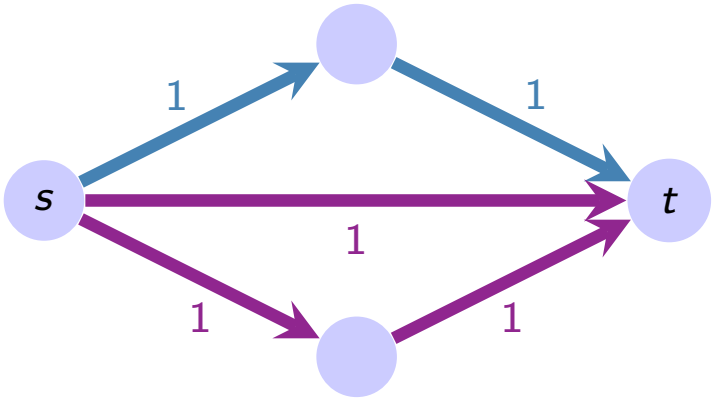


$1 \leq \gamma \leq u$

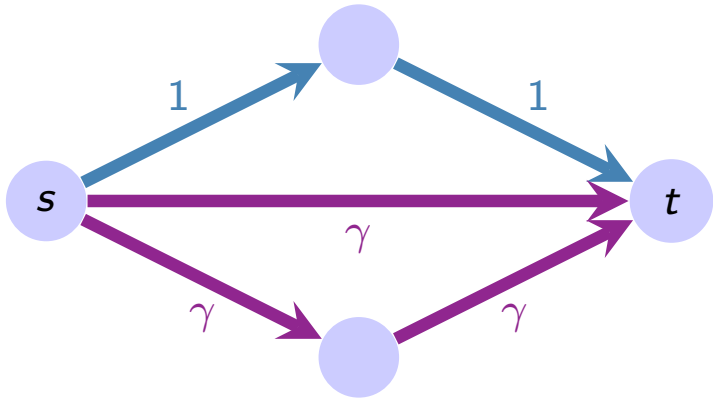
Modifying the Graph

What we already discussed





Where we proceed

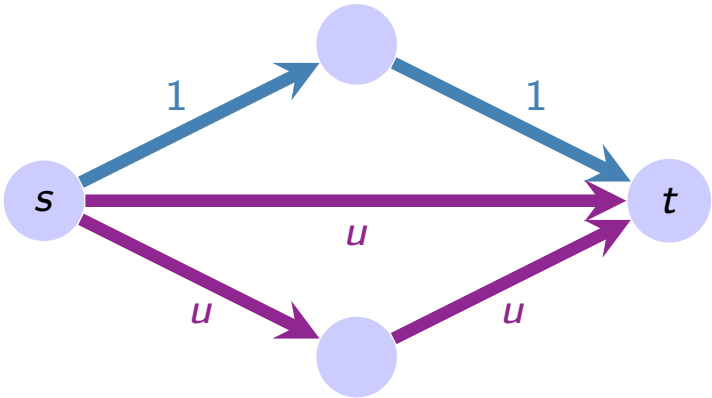


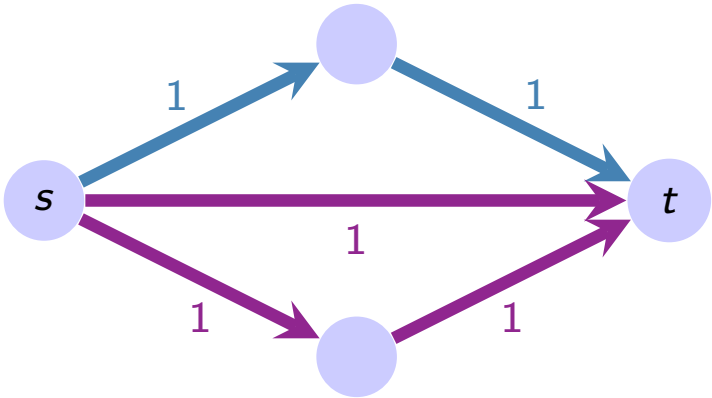
$1 \leq \gamma \leq u$

Call this graph $G(\gamma)$ and denote the capacity of a cut C in $G(\gamma)$ by $\text{cap}^{\gamma}(C)$.

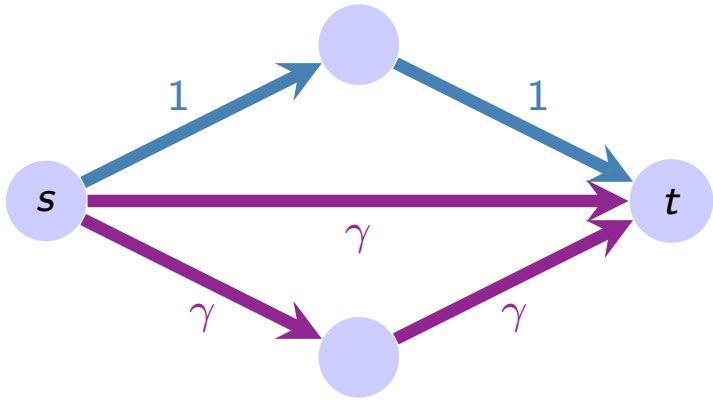
Modifying the Graph

What we already discussed





Where we proceed

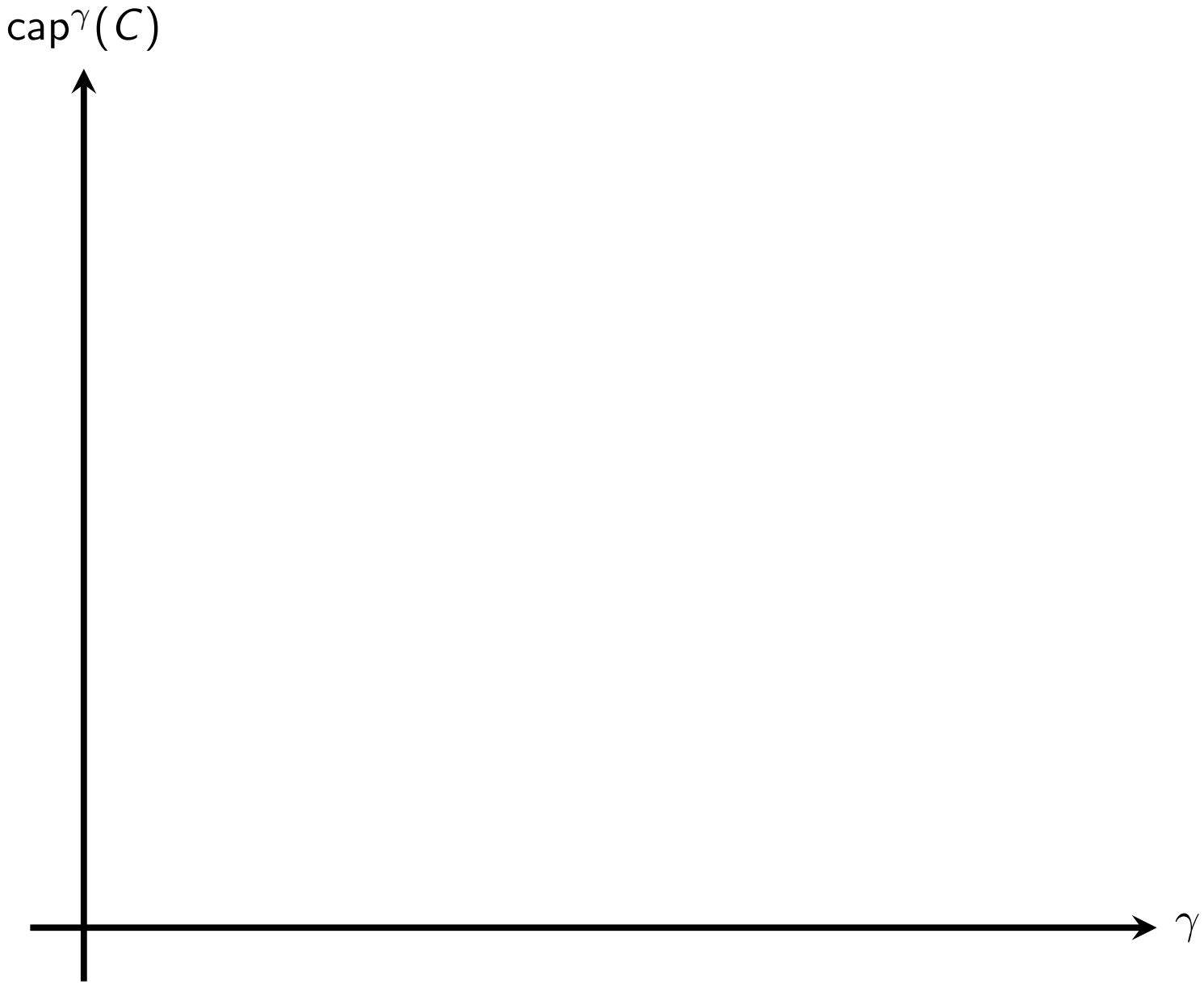


$1 \leq \gamma \leq u$

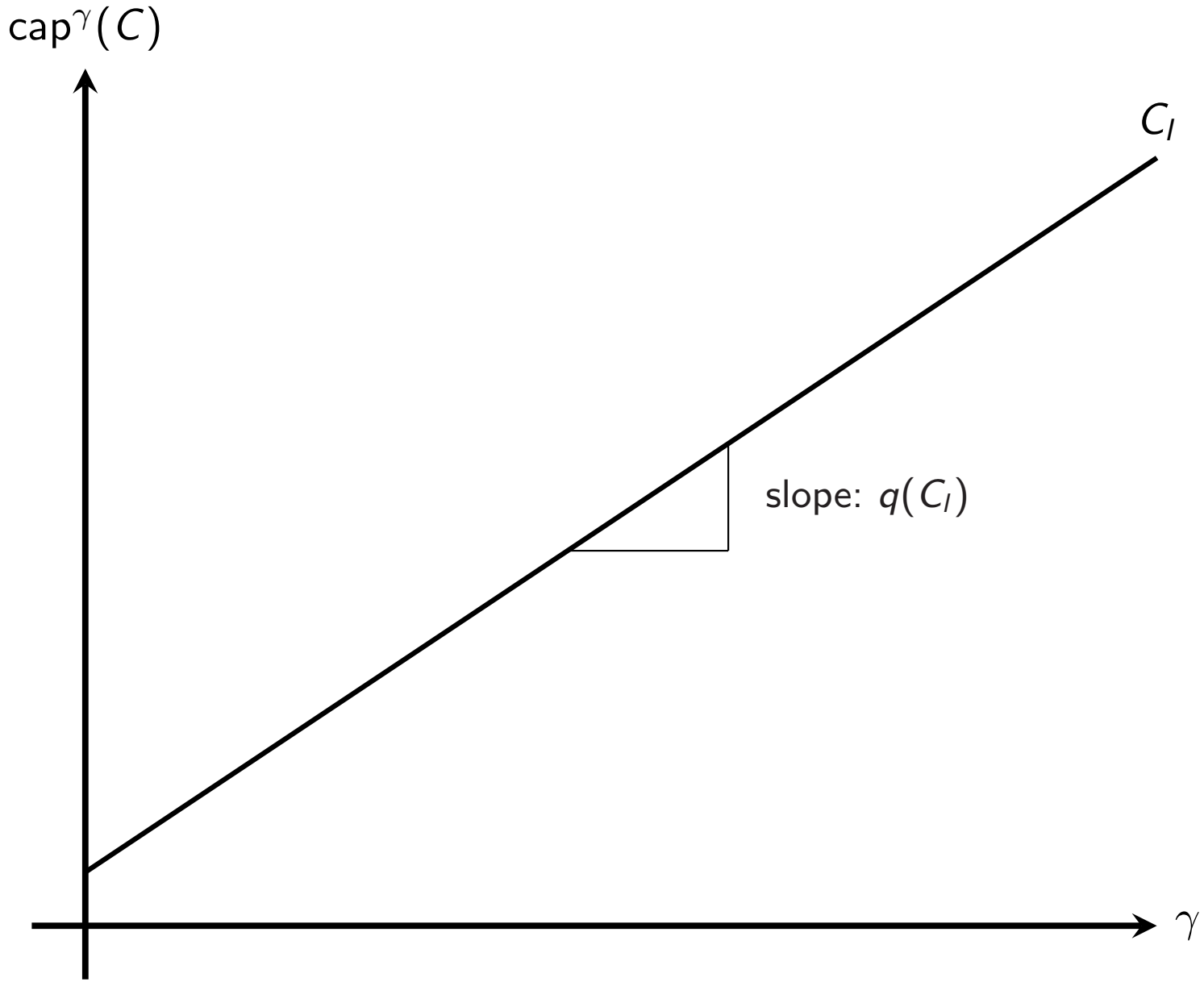
Call this graph $G(\gamma)$ and denote the capacity of a cut C in $G(\gamma)$ by $\text{cap}^{\gamma}(C)$.

But how do we choose γ?

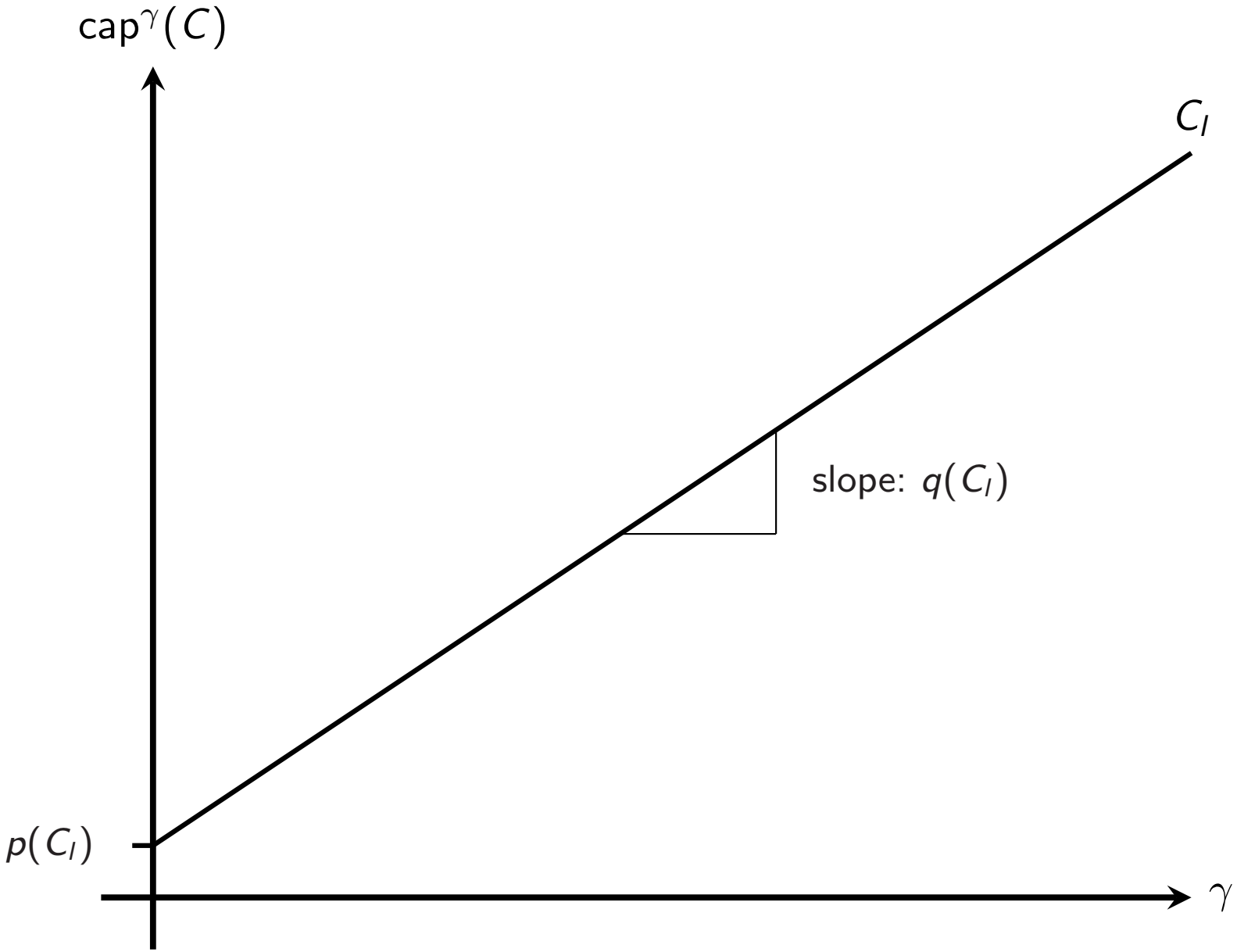
The Bisection Method



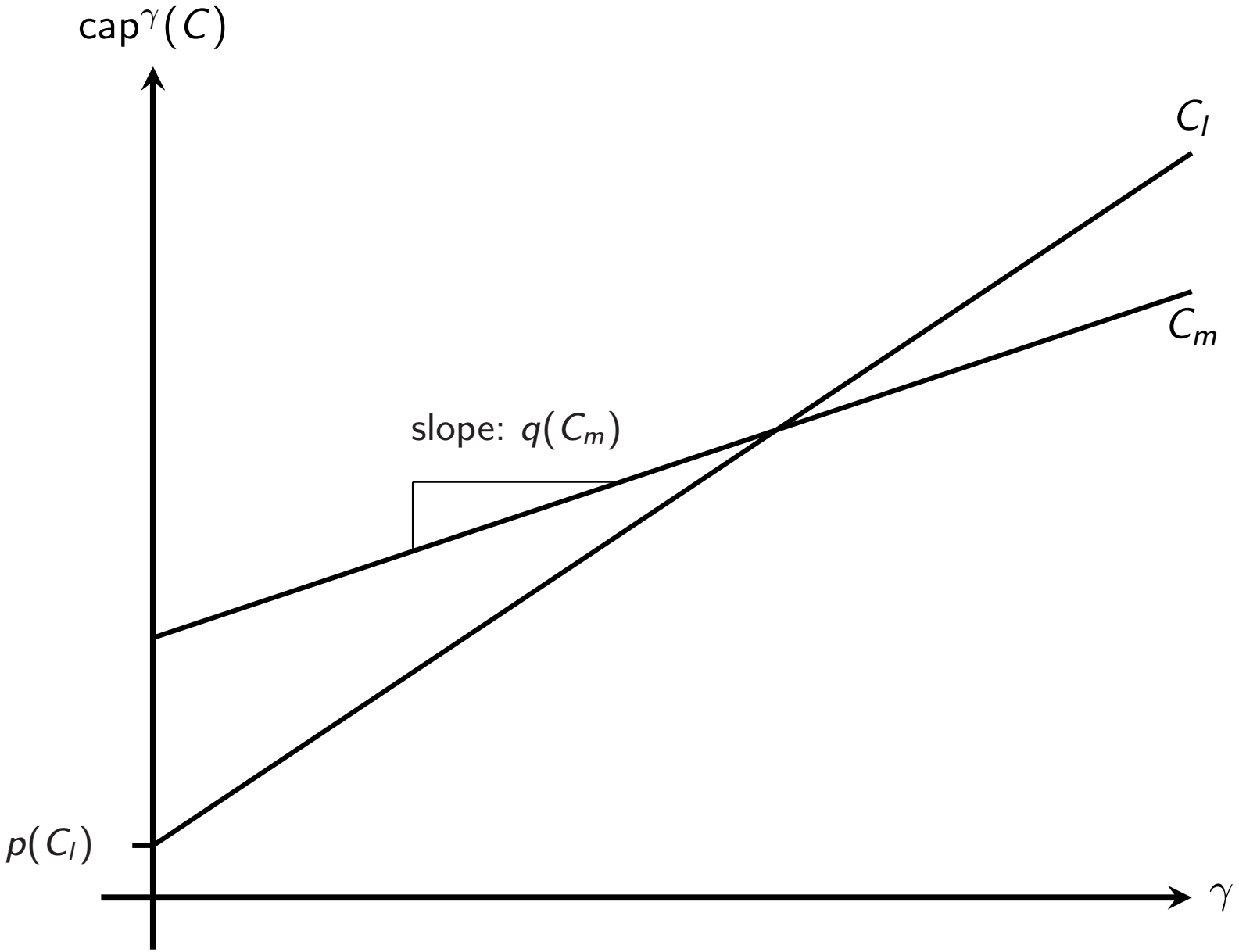
The Bisection Method



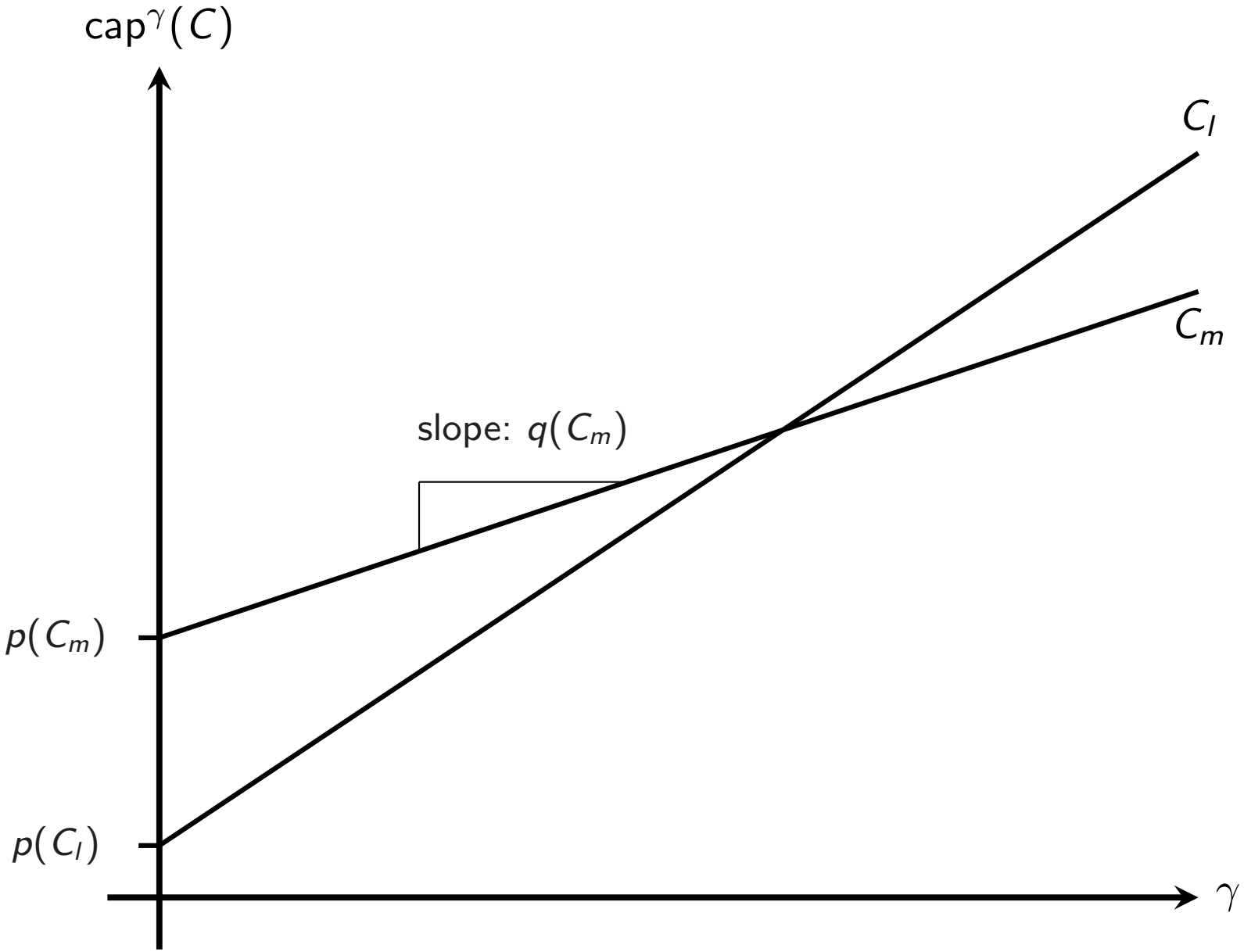
The Bisection Method



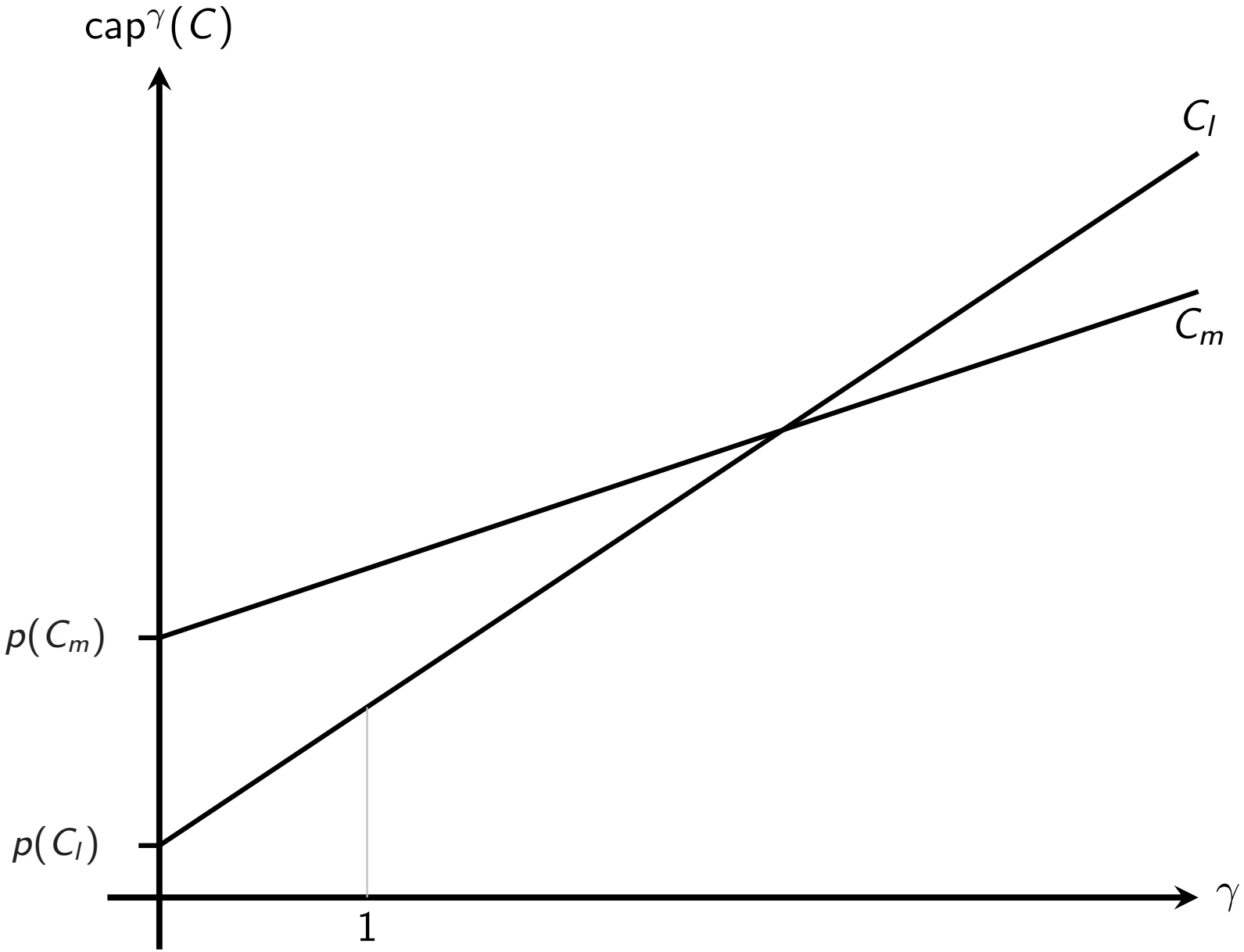
The Bisection Method



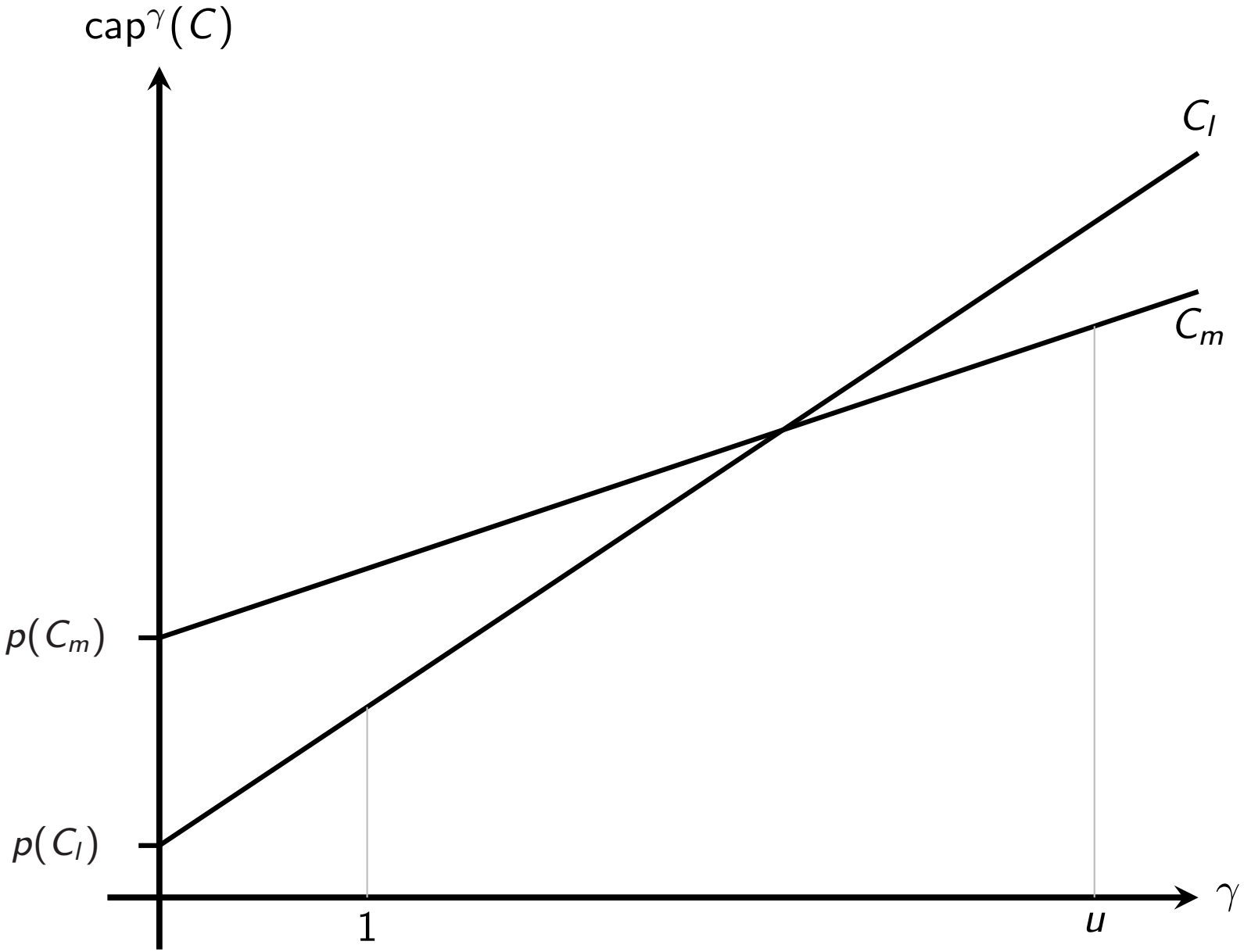
The Bisection Method



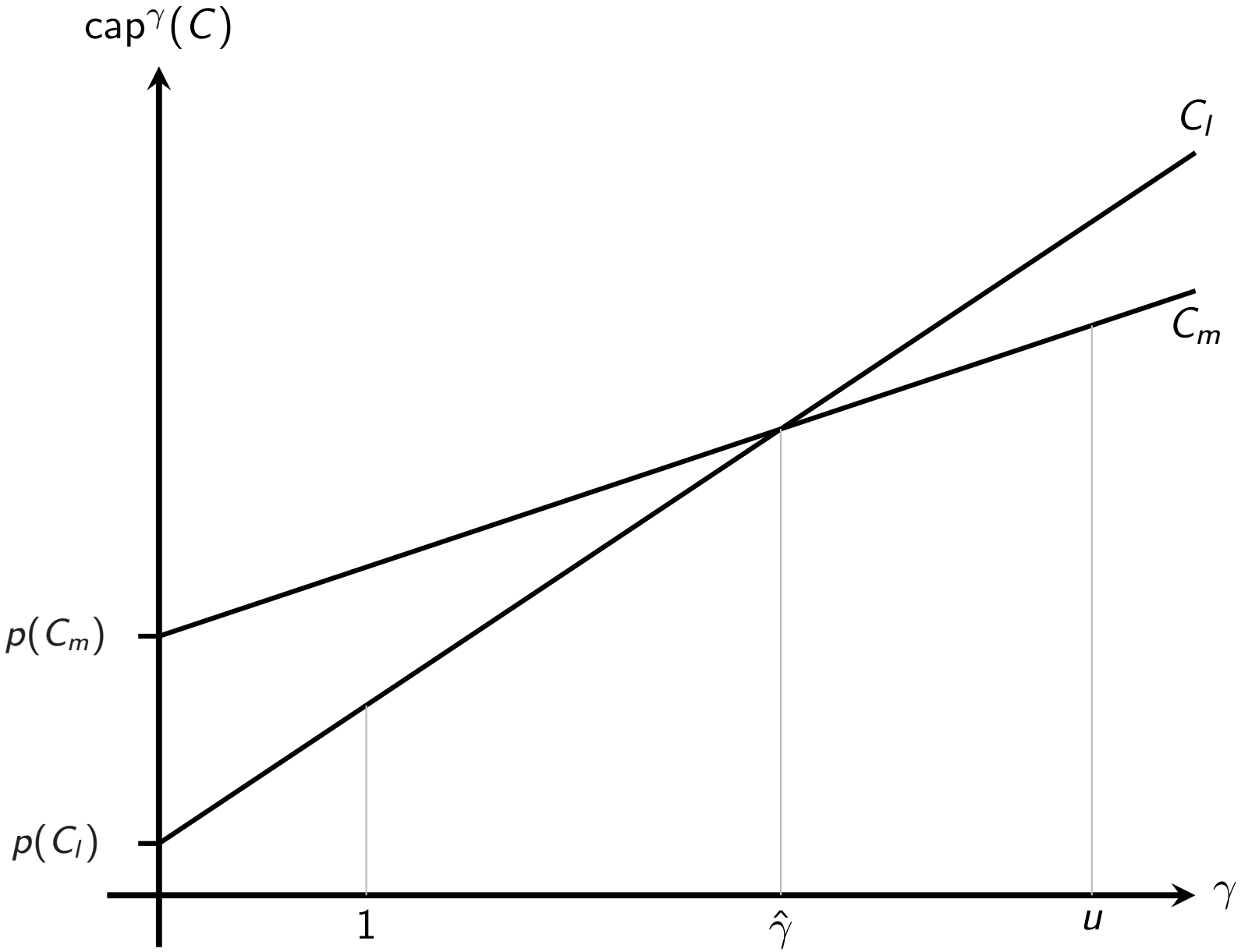
The Bisection Method



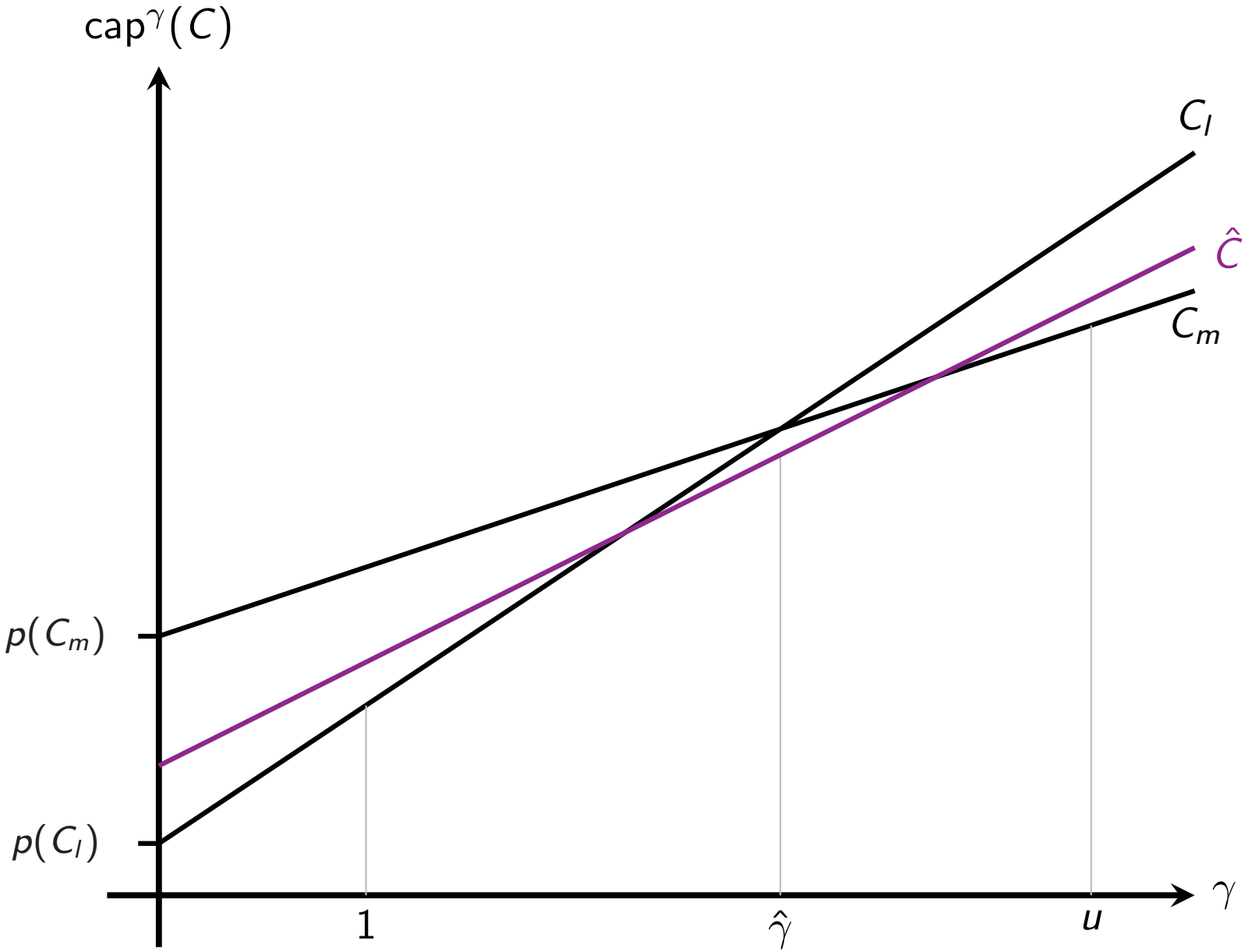
The Bisection Method



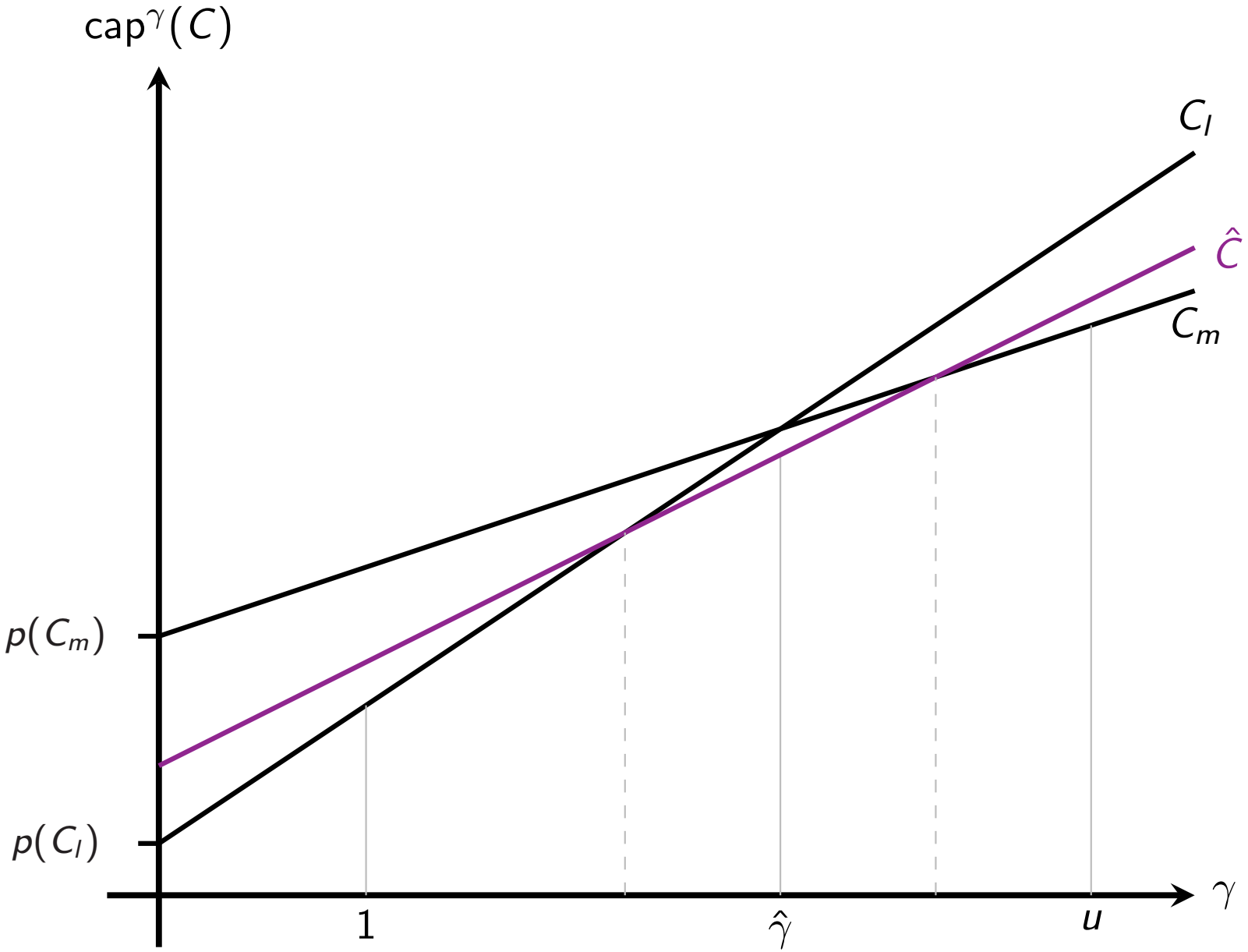
The Bisection Method



The Bisection Method



The Bisection Method



The Algorithm

bisection (C_1, C_2)
  Compute $\hat{\gamma}$ and $\hat{C}$
  if $q(\hat{C}) \notin \{q(C_1), q(C_2)\}$ **then**
    return bisection$(C_1, \hat{C}) \cup$ bisection$(\hat{C}, C_2)$
  else
    return $\{C_1, C_2\}$
  end

The Algorithm

```
1 bisection (C_1, C_2)
2 |   Compute γ̂ and Ĉ
3 |   if q(Ĉ) ∉ {q(C_1), q(C_2)} then
4 |   |   return bisection(C_1, Ĉ) ∪ bisection(Ĉ, C_2)
5 |   else
6 |   |   return {C_1, C_2}
7 |   end
```

Theorem 1

There is at least one cut C in the set returned by bisection(C_l, C_m) such that the solution $R(C)$ is a $(B+1)$-approximate solution for u-NFI on G.

The Algorithm

```
1 bisection (C_1, C_2)
2 |   Compute γ̂ and Ĉ
3 |   if q(Ĉ) ∉ {q(C_1), q(C_2)} then
4 |   |   return bisection(C_1, Ĉ) ∪ bisection(Ĉ, C_2)
5 |   else
6 |   |   return {C_1, C_2}
7 |   end
```

Theorem 1

There is at least one cut C in the set returned by bisection(C_l, C_m) such that the solution $R(C)$ is a $(B+1)$-approximate solution for u-NFI on G.

Theorem 2

The algorithm bisection(C_l, C_m) runs in polynomial time.

The Algorithm

```
1 bisection (C_1, C_2)
2 |   Compute γ̂ and Ĉ
3 |   if q(Ĉ) ∉ {q(C_1), q(C_2)} then
4 |   |   return bisection(C_1, Ĉ) ∪ bisection(Ĉ, C_2)
5 |   else
6 |   |   return {C_1, C_2}
7 |   end
```

Theorem 1

There is at least one cut C in the set returned by bisection(C_l, C_m) such that the solution $R(C)$ is a $(B+1)$-approximate solution for u-NFI on G.

Theorem 2

The algorithm bisection(C_l, C_m) runs in polynomial time.

Properties of New Cuts

Lemma 4

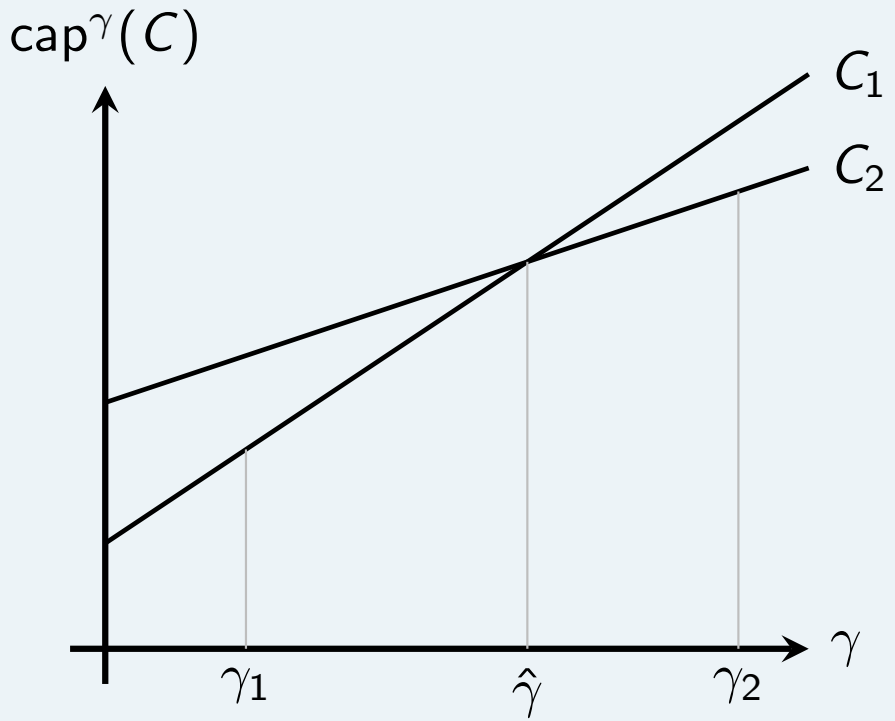
If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof

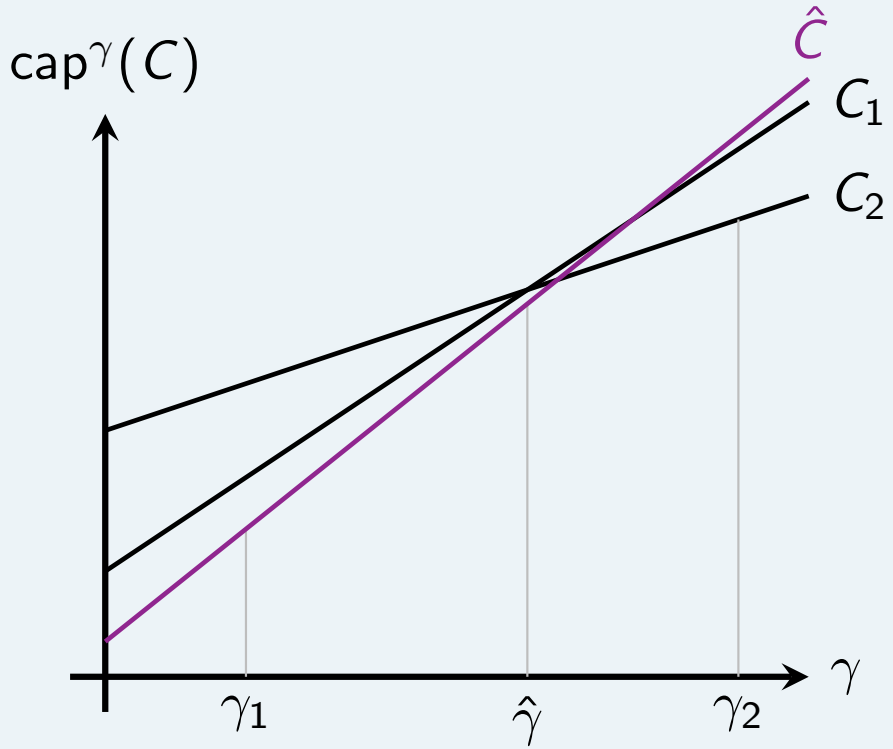


Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof

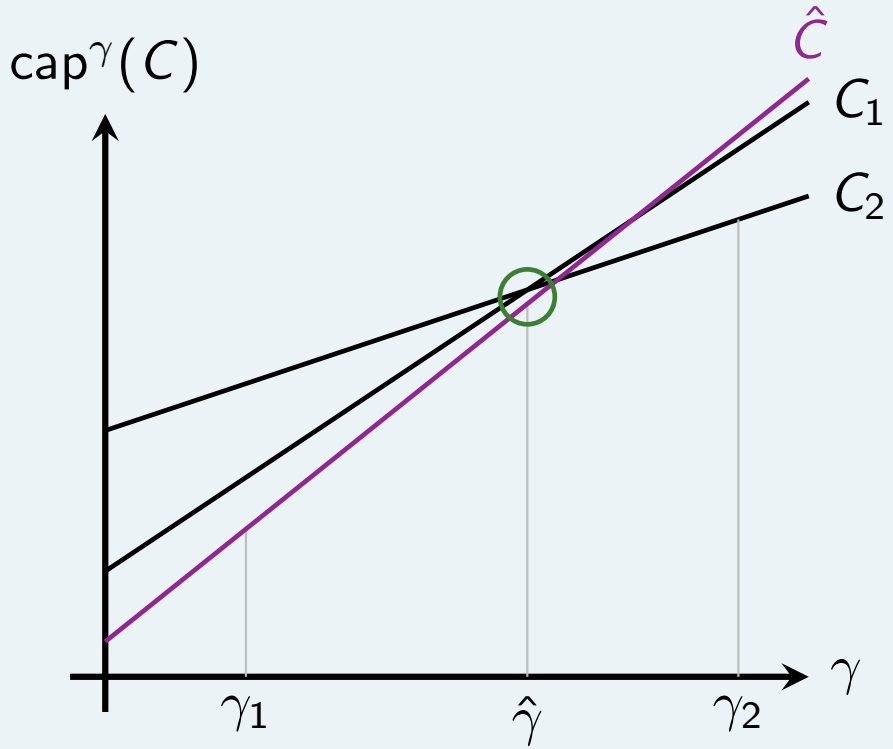


Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof

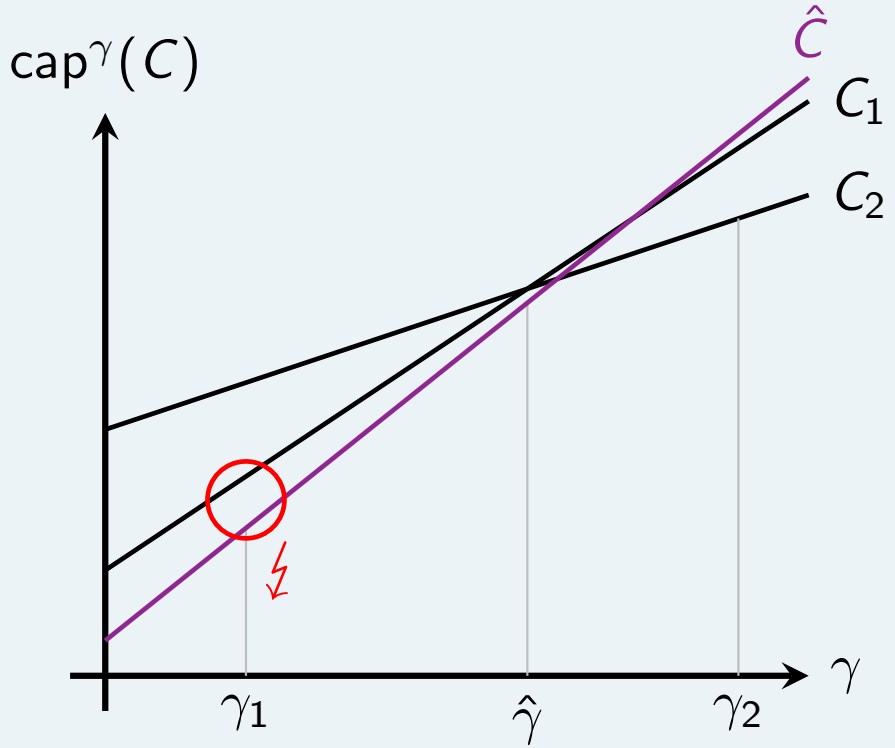


Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof

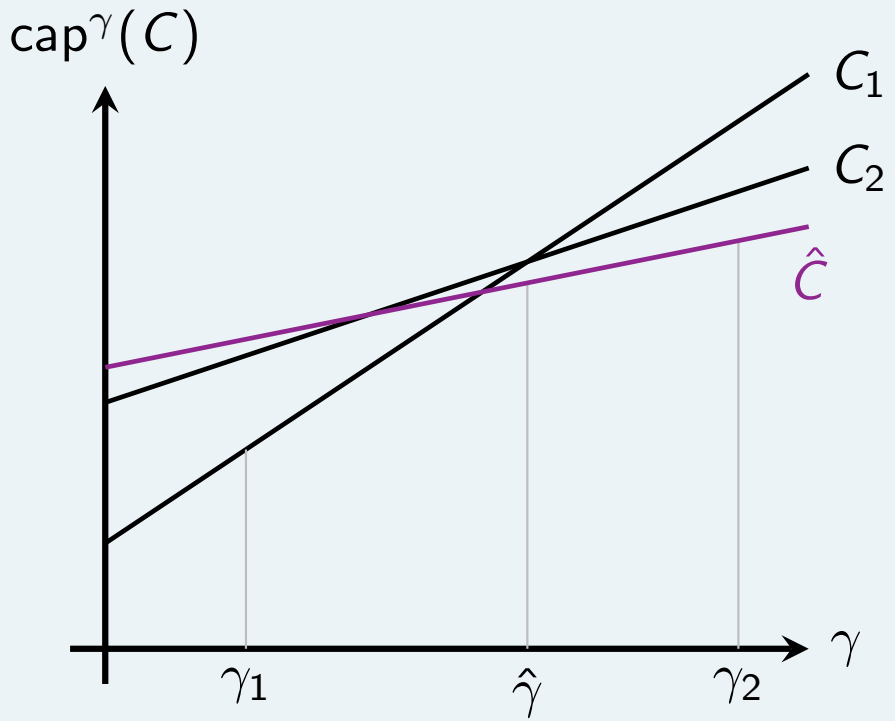


Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof

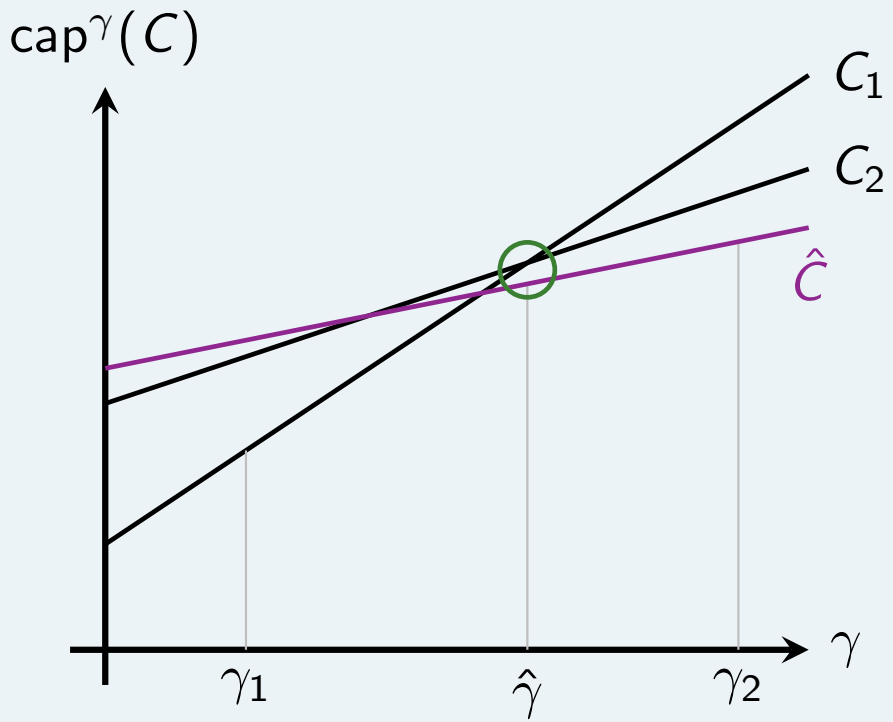


Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof

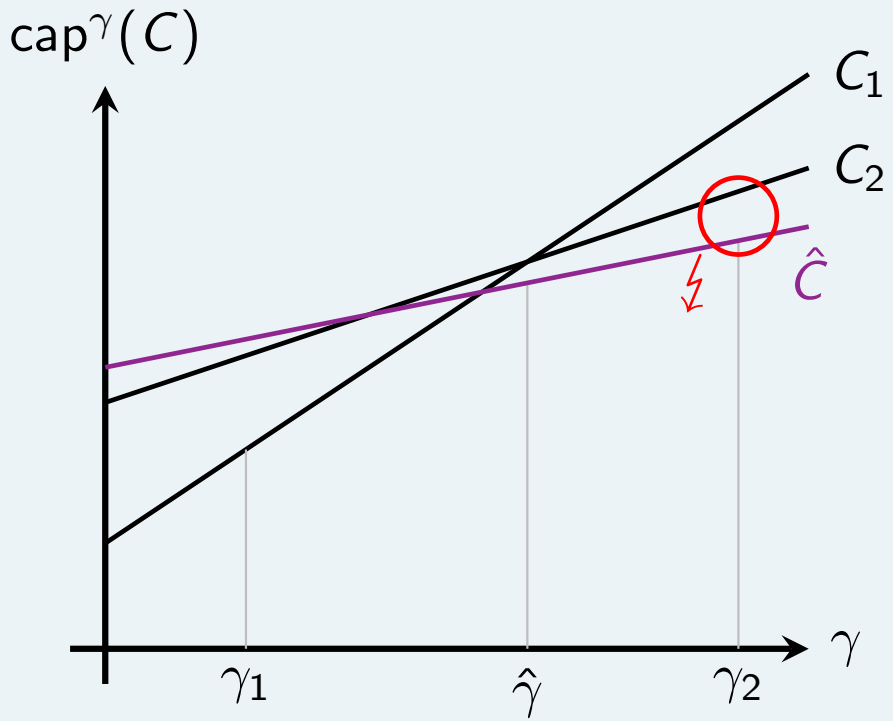


Properties of New Cuts

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Proof



Polynomial Running Time

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

Polynomial Running Time

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

- For each $k \in \{0, \ldots, m\}$, the algorithm computes at most three cuts with exactly k large arcs.

Polynomial Running Time

Lemma 4

If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

- For each $k \in \{0, \ldots, m\}$, the algorithm computes at most three cuts with exactly k large arcs.
- The algorithm has at most $3(m+1)$ recursion steps.

Polynomial Running Time

Lemma 4

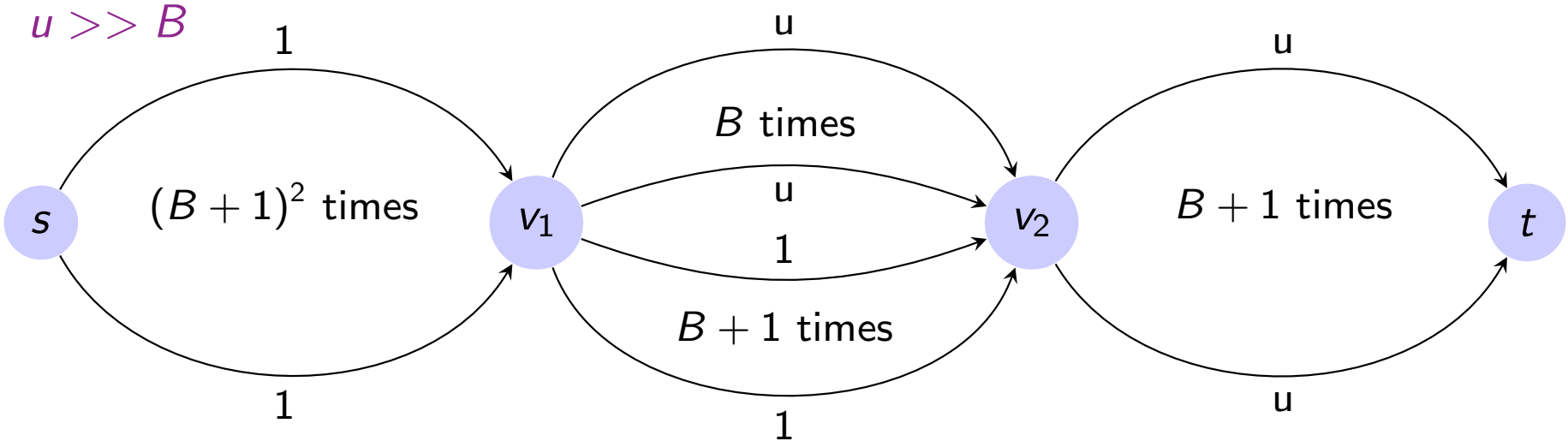
If the bisection method finds a new cut $\hat{C}$ by calling bisection(C_1, C_2), then $q(C_1) > q(\hat{C}) > q(C_2)$.

- For each $k \in \{0, \ldots, m\}$, the algorithm computes at most three cuts with exactly k large arcs.
- The algorithm has at most $3(m+1)$ recursion steps.

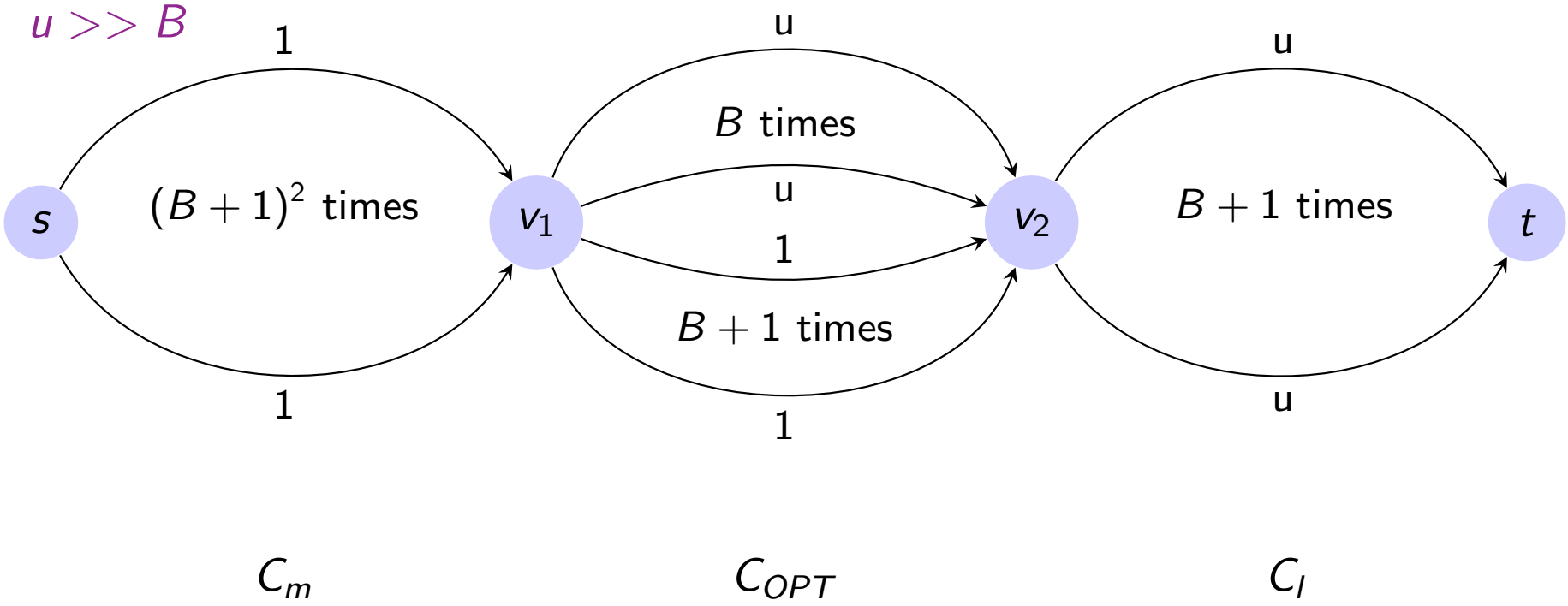
Theorem 2

The algorithm bisection(C_l, C_m) runs in polynomial time.

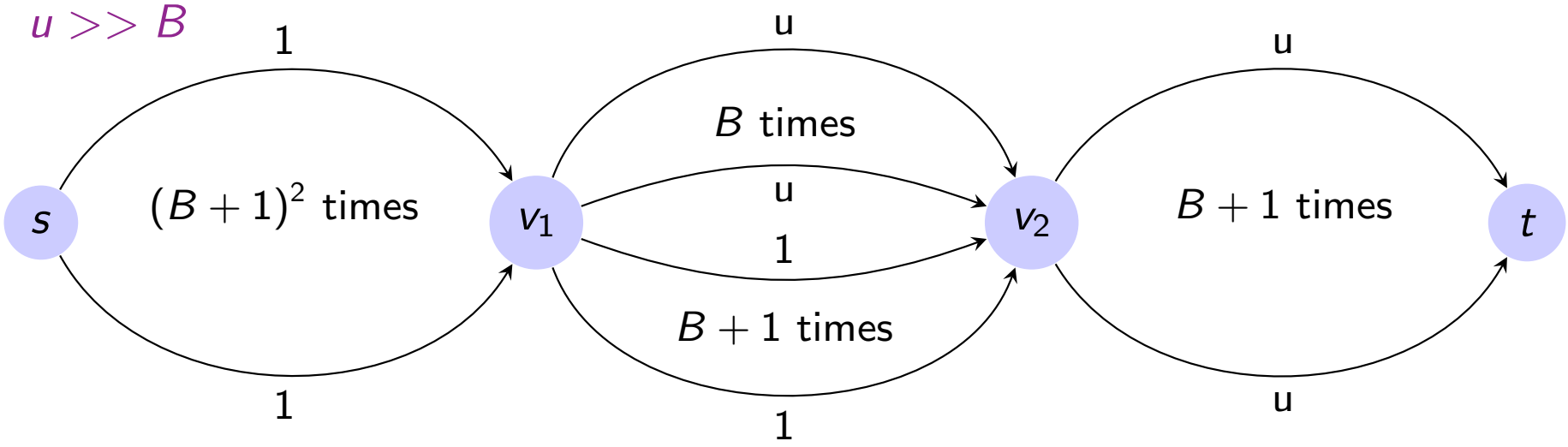
Almost-Tightness of Approximation Ratio



Almost-Tightness of Approximation Ratio



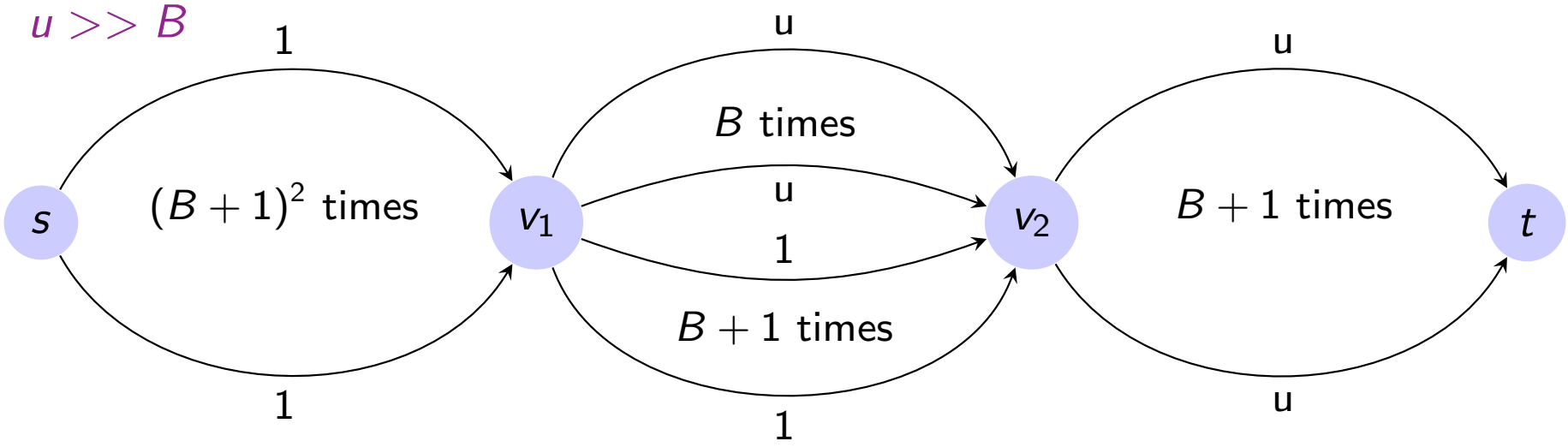
Almost-Tightness of Approximation Ratio



C_m $\quad$ C_{OPT} $\quad$ C_l

$\text{val}(C_m) = (B+1)^2 - B$ $\quad$ $\text{val}(C_{OPT}) = B+1$ $\quad$ $\text{val}(C_l) = (B+1)^2$

Almost-Tightness of Approximation Ratio



C_m $\quad$ C_{OPT} $\quad$ C_l

$\text{val}(C_m) = (B+1)^2 - B$ $\quad$ $\text{val}(C_{OPT}) = B+1$ $\quad$ $\text{val}(C_l) = (B+1)^2$

Observation 2

The algorithm is not a B-approximation algorithm for u-NFI as $\text{val}(C_m) = B^2 + B + 1 > B^2 + B = B \cdot \text{val}(C_{OPT})$.

Things I want you to say

After this presentation, I know ...

Things I want you to say

After this presentation, I know ...

- ... that the crucial part of u-NFI is to find the right cut.

Things I want you to say

After this presentation, I know ...

- ... that the crucial part of u-NFI is to find the right cut.
- ... how to find good candidates for the right cut.

Things I want you to say

After this presentation, I know ...

- ... that the crucial part of u-NFI is to find the right cut.
- ... how to find good candidates for the right cut.
- ... how to be a great president.

JAN BOECKMANN, M.Sc.

TUMCS

Am Essigberg 3
94315 Straubing

jan.boeckmann@tum.de