Price of Anarchy via LP duality

Sayan Bhattacharya (IMSc, Chennai)

Algorithms and Game Theory

Optimization Problem

Input $\longrightarrow$ $\boxed{\text{Algorithm}}$ $\longrightarrow$ Output

Considerations: Computational Efficiency, Approximation Ratio

Algorithms and Game Theory

Optimization Problem

Input → Algorithm → Output

Utility/Cost

Rational Agents

Considerations: Computational Efficiency, Approximation Ratio

Algorithms and Game Theory

Considerations: Computational Efficiency, Approximation Ratio

Algorithms and Game Theory

Input

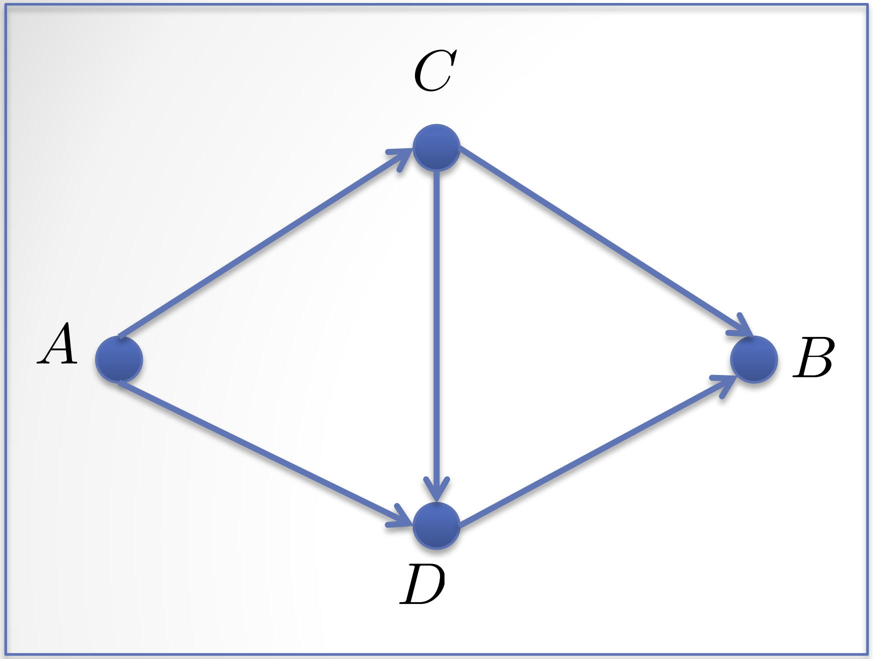
Output

Algorithms and Game Theory

Input

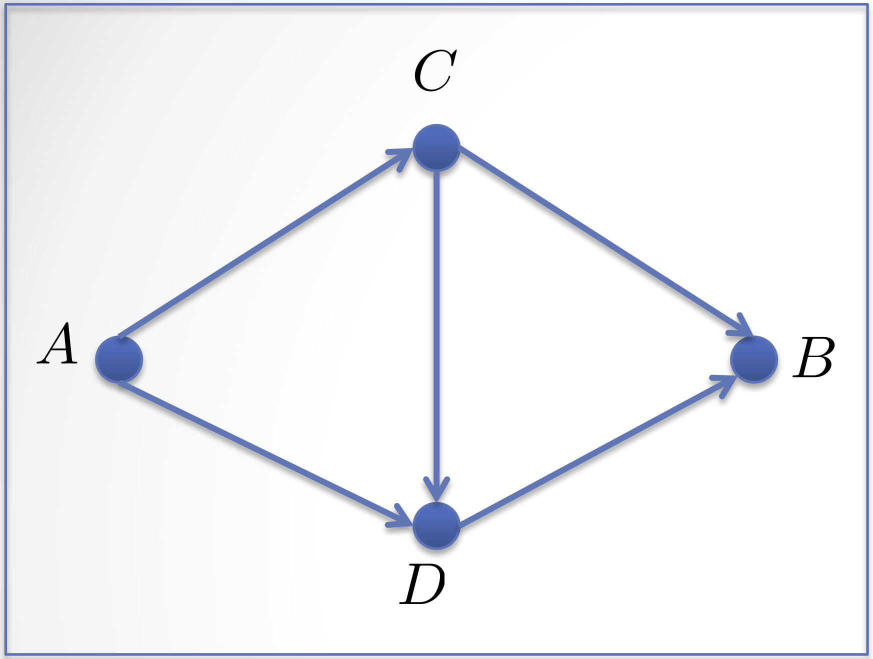
Output

Example: Selfish Routing



400 cars want to go from A to B.

Example: Selfish Routing



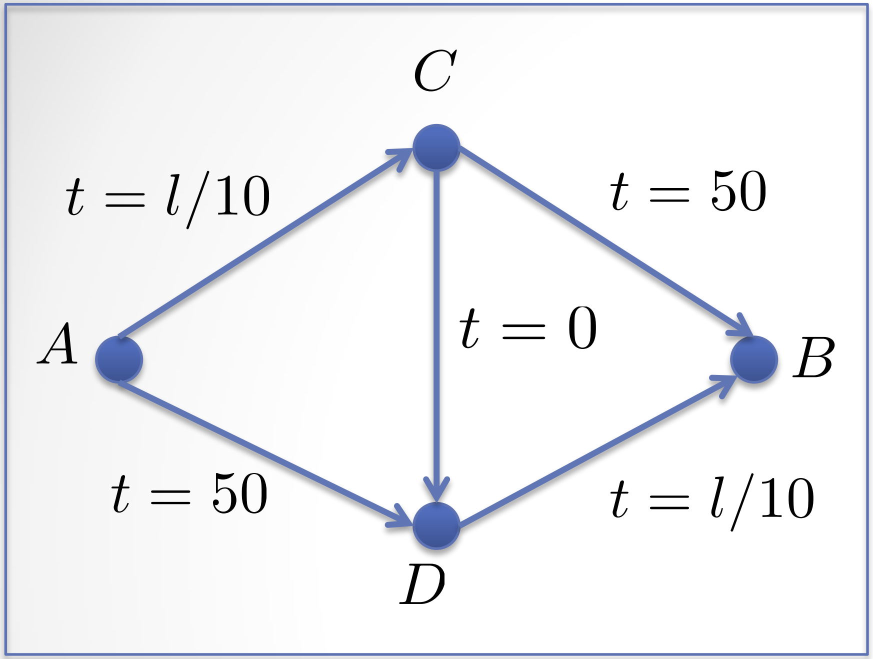
$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

$$t(e) = a \cdot l(e) + b$$

latency function at link e.

Example: Selfish Routing



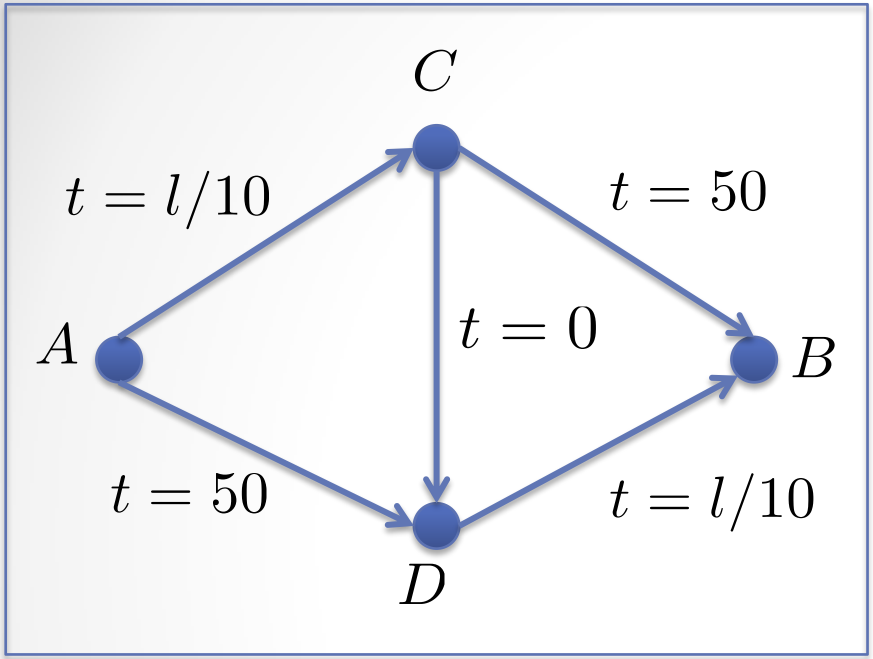
$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

$$t(e) = a \cdot l(e) + b$$

latency function at link e.

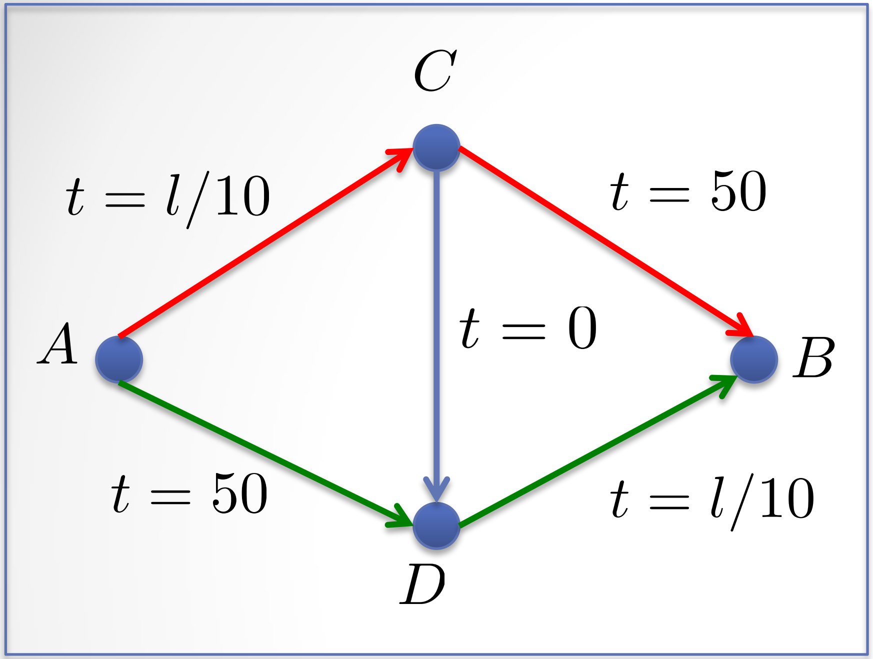
Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

Example: Selfish Routing



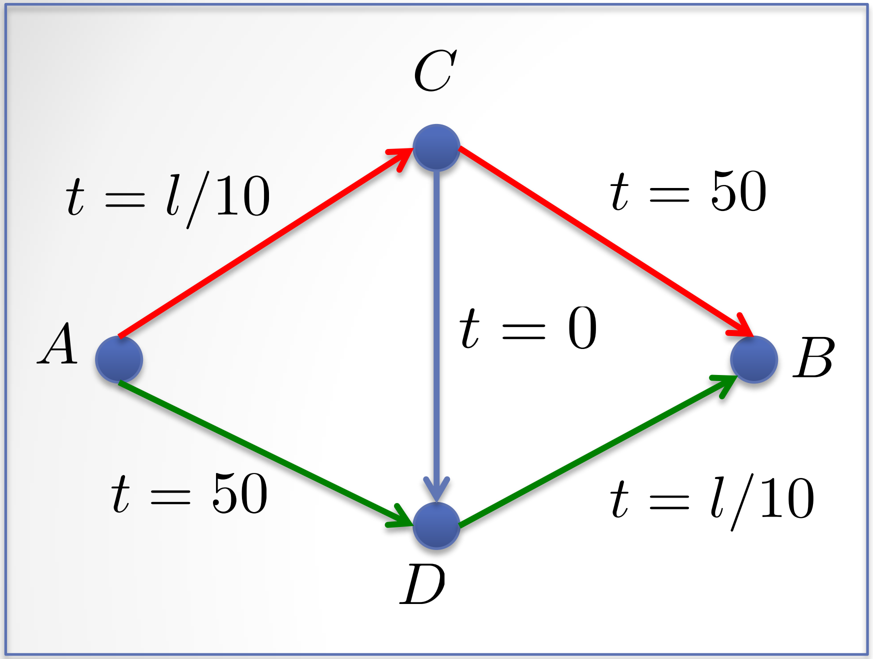
$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

200 cars take the red path.

200 cars take the green path.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

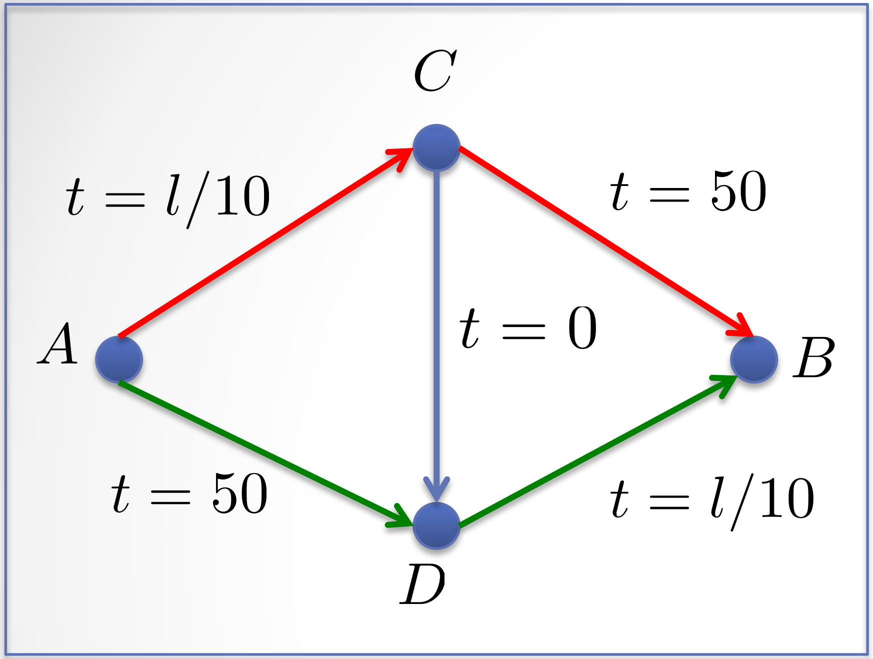
400 cars want to go from A to B.

200 cars take the red path.

200 cars take the green path.

Travel time of a red car $= 200/10 + 50 = 70$.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

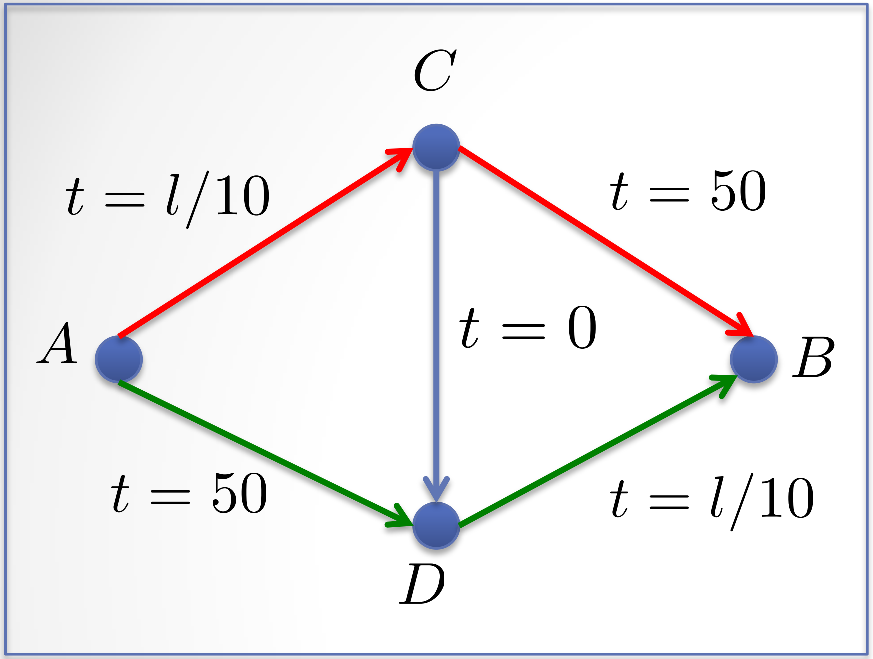
200 cars take the red path.

200 cars take the green path.

Travel time of a red car $= 200/10 + 50 = 70$.

Travel time of a green car $= 50 + 200/10 = 70$.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

Travel time of each car = 70 mins.

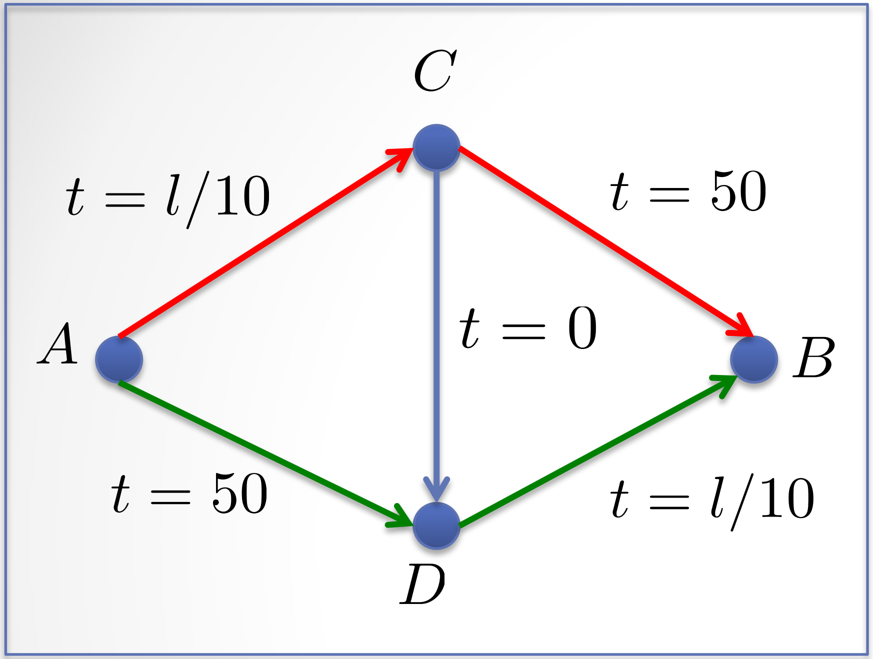
200 cars take the red path.

200 cars take the green path.

Travel time of a red car $= 200/10 + 50 = 70$.

Travel time of a green car $= 50 + 200/10 = 70$.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

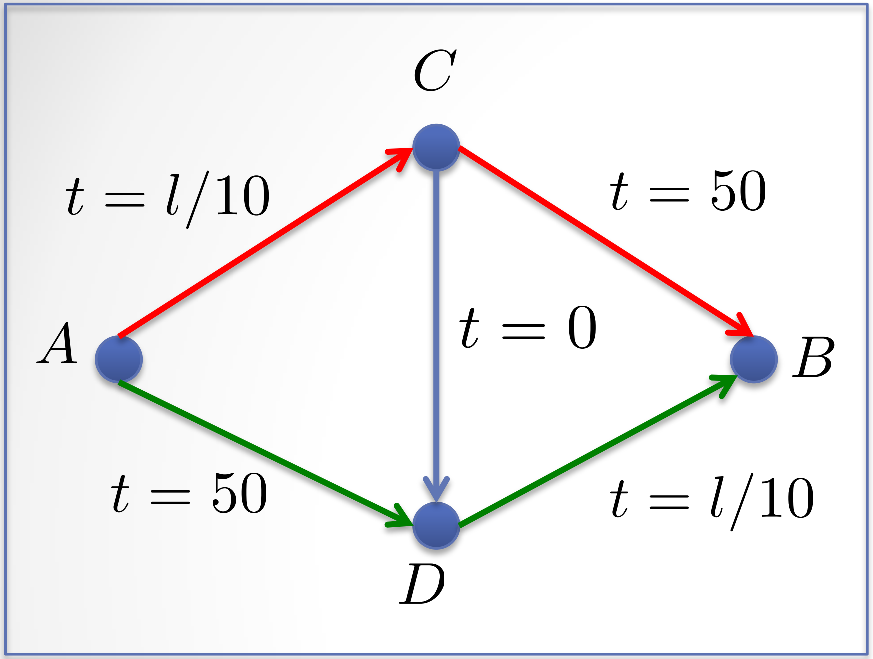
400 cars want to go from A to B.

Travel time of each car = 70 mins.

200 cars take the red path.

200 cars take the green path.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

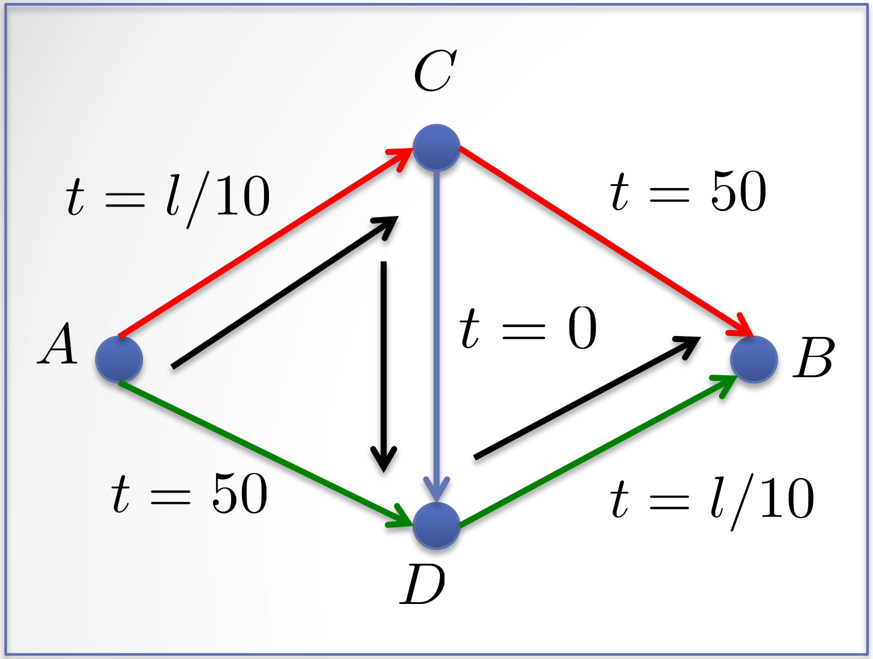
400 cars want to go from A to B.

Travel time of each car $= 70$ mins.

200 cars take the red path. 200 cars take the green path.

Suppose that a red car switches to the path ACDB.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

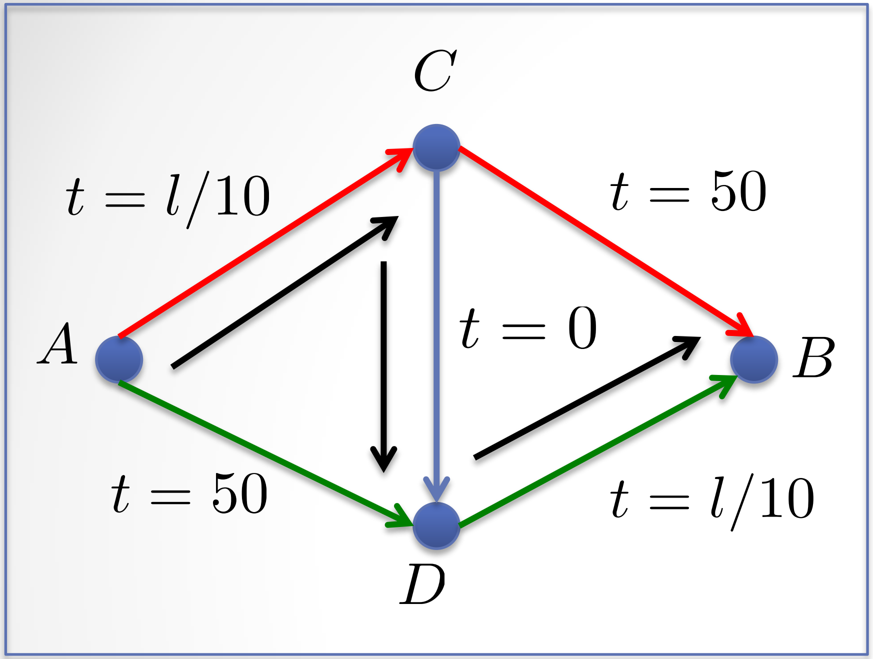
Travel time of each car $= 70$ mins.

200 cars take the red path.

200 cars take the green path.

Suppose that a red car switches to the path ACDB.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

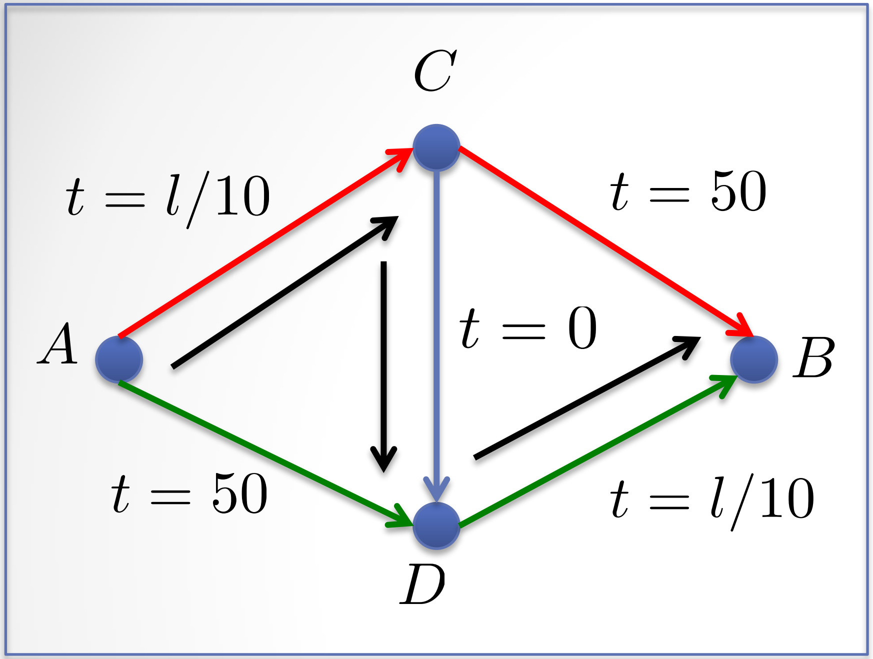
Travel time of each car $= 70$ mins.

200 cars take the red path. 200 cars take the green path.

Suppose that a red car switches to the path ACDB.

Then its travel time $= 200/10 + 0 + 201/10 = 40.1$ mins!

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

Travel time of each car $= 70$ mins.

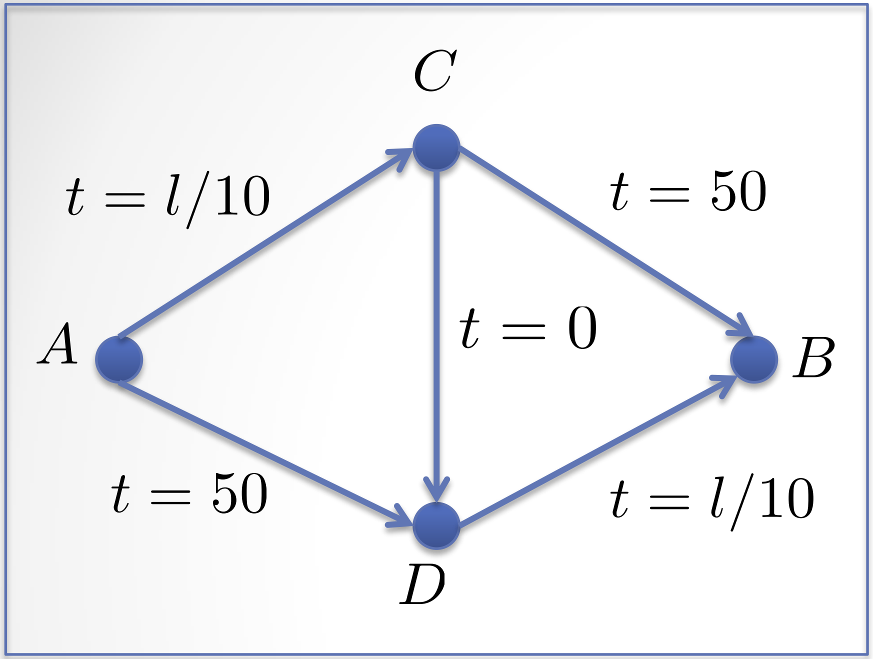
The solution is unstable!

200 cars take the red path. 200 cars take the green path.

Suppose that a red car switches to the path ACDB.

Then its travel time $= 200/10 + 0 + 201/10 = 40.1$ mins!

Example: Selfish Routing



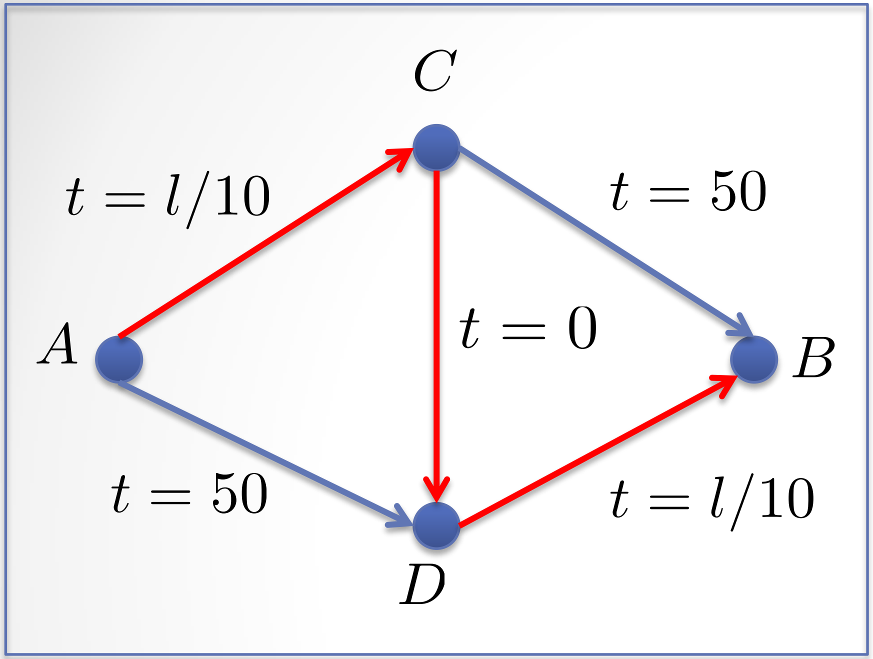
$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

Travel time of each car $= 70$ mins.

The solution is unstable!

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

Travel time of each car = 70 mins.

The solution is unstable!

Every car takes the route ACDB.

Travel time of each car = 400/10+0+400/10 = 80 mins.

This solution is stable! (Time for path ACB = 400/10+50 = 90)

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

(3) Each player j selects a strategy $\theta_j \in \mathcal{S}_j$.

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

(3) Each player j selects a strategy $\theta_j \in \mathcal{S}_j$.

This defines an "outcome/strategy-profile" $\theta = (\theta_1, \ldots, \theta_n)$.

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

(3) Each player j selects a strategy $\theta_j \in \mathcal{S}_j$.

This defines an "outcome/strategy-profile" $\theta = (\theta_1, \ldots, \theta_n)$.

(4) Let $S = \times_j S_j$ be the set of all possible outcomes.

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

(3) Each player j selects a strategy $\theta_j \in \mathcal{S}_j$.

This defines an "outcome/strategy-profile" $\theta = (\theta_1, \ldots, \theta_n)$.

(4) Let $S = \times_j S_j$ be the set of all possible outcomes.

(5) Each player j has a "cost function" $c_j : S \to \mathbf{R}$. (travel-time)

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

(3) Each player j selects a strategy $\theta_j \in \mathcal{S}_j$.

This defines an "outcome/strategy-profile" $\theta = (\theta_1, \ldots, \theta_n)$.

(4) Let $S = \times_j S_j$ be the set of all possible outcomes.

(5) Each player j has a "cost function" $c_j : S \to \mathbf{R}$. (travel-time)

Player j prefers an outcome θ over θ' iff $c_j(\theta) \leq c_j(\theta')$.

Formally defining a ``Game''

(1) A set of "players" $\mathcal{N} = \{1, \ldots, n\}$. (person driving a car)

(2) A set of "strategies" S_j for each player $j \in \mathcal{N}$. (source-destination routes)

(3) Each player j selects a strategy $\theta_j \in \mathcal{S}_j$.

This defines an "outcome/strategy-profile" $\theta = (\theta_1, \ldots, \theta_n)$.

(4) Let $S = \times_j S_j$ be the set of all possible outcomes.

(5) Each player j has a "cost function" $c_j : S \to \mathbf{R}$. (travel-time)

Player j prefers an outcome θ over θ' iff $c_j(\theta) \leq c_j(\theta')$.

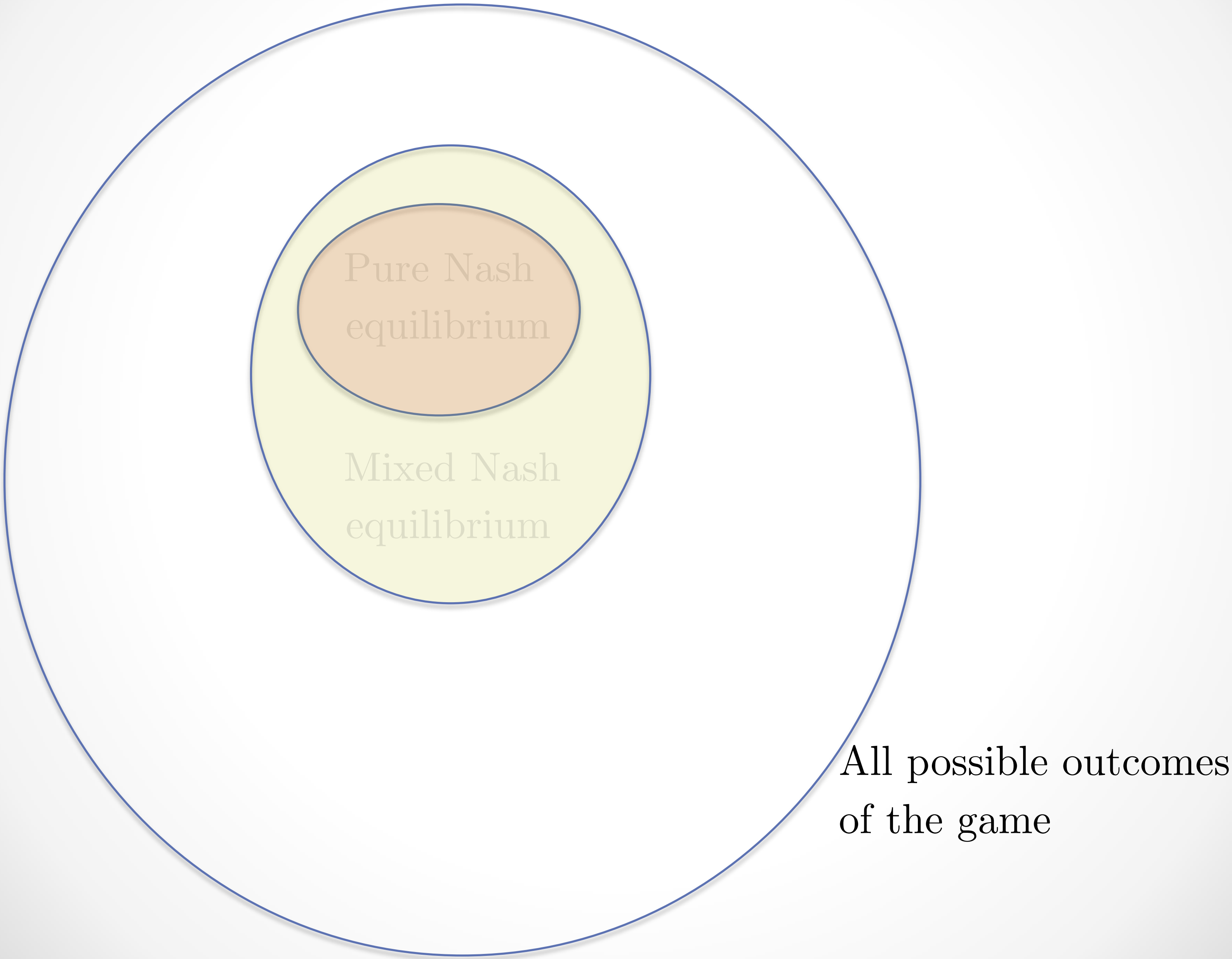
What is a *rational* outcome of such a game?

Pure Nash Equilibrium

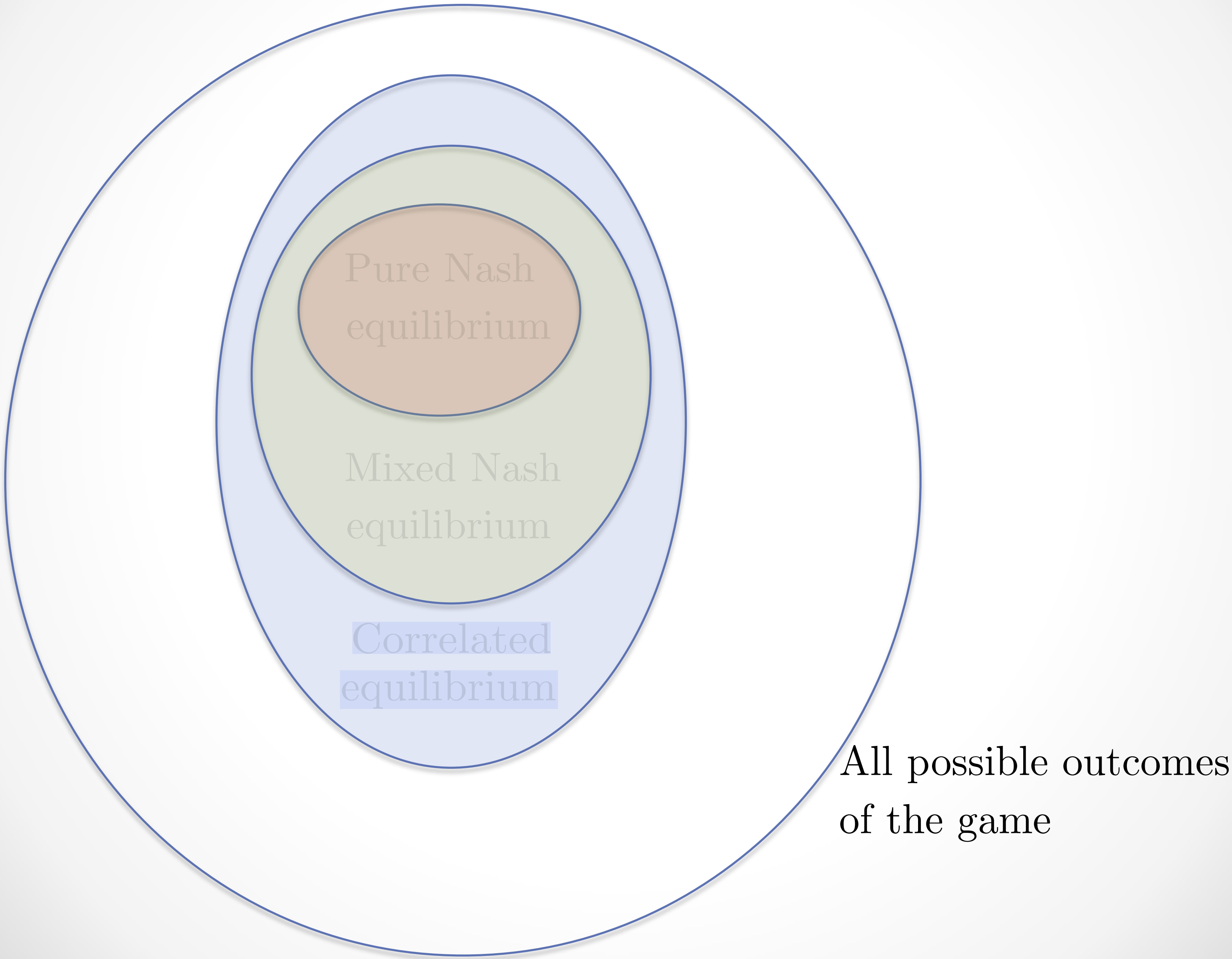
An outcome of a game is a *pure Nash equilibrium* iff no player can reduce her cost by unilaterally switching her strategy.

Different ``Solution Concepts''

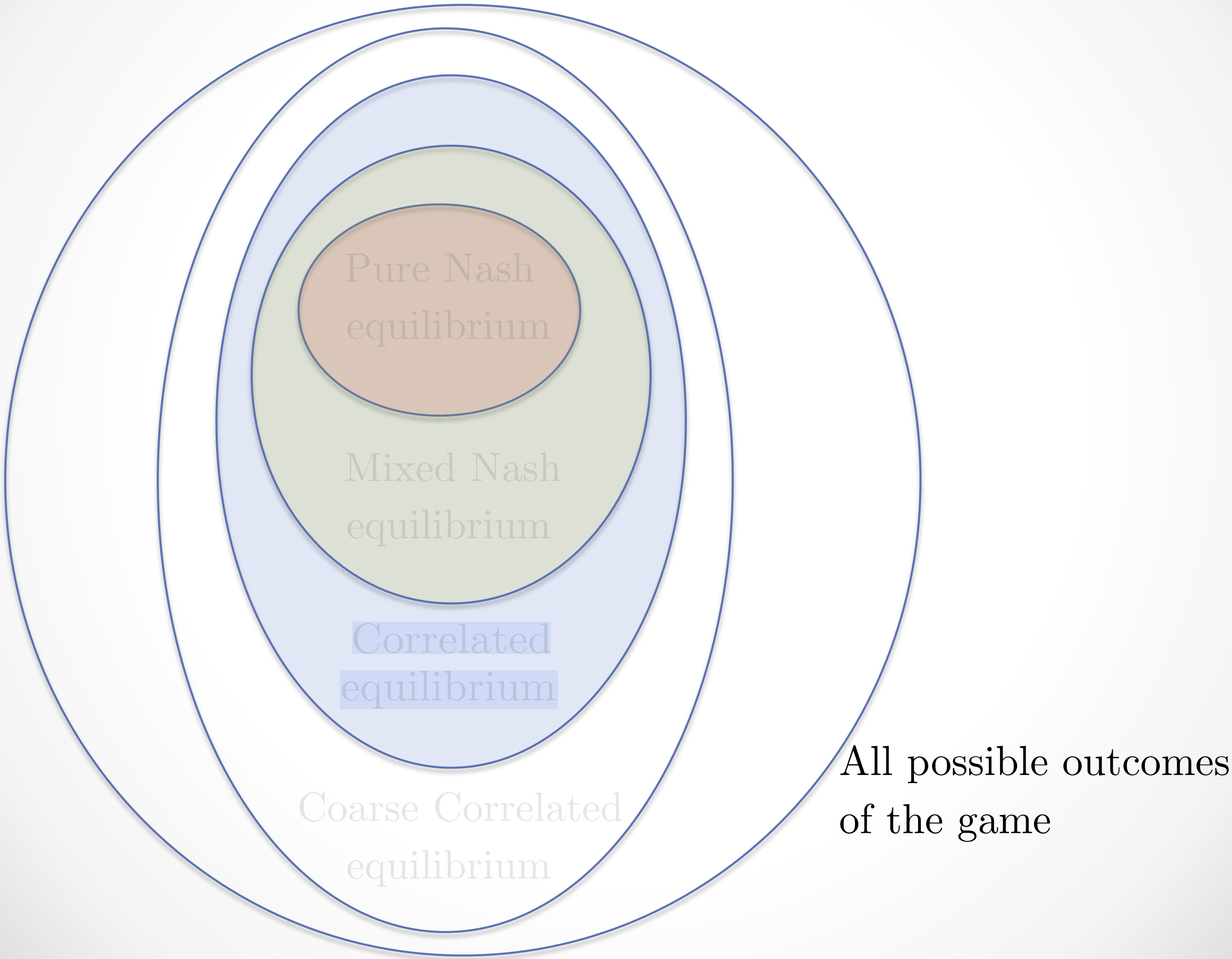
Different ``Solution Concepts''



Different ``Solution Concepts''

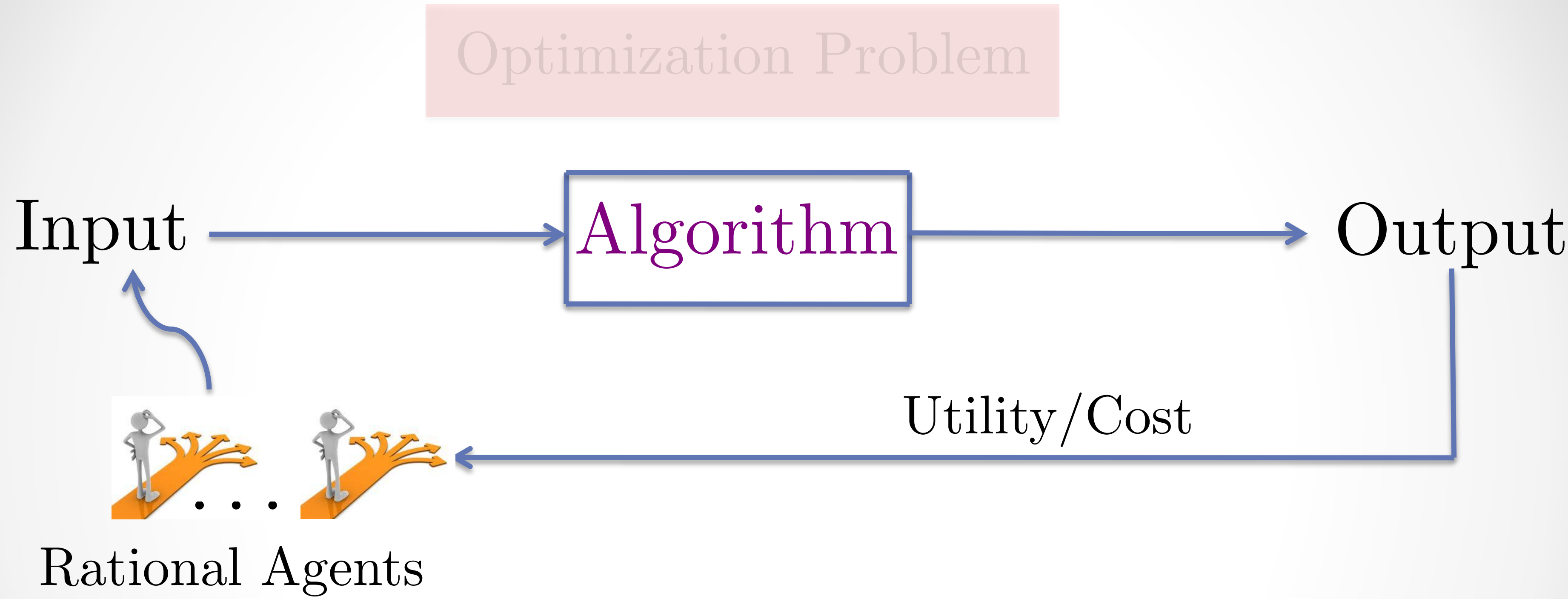


Different ``Solution Concepts''



I: Price of Anarchy (Definition)

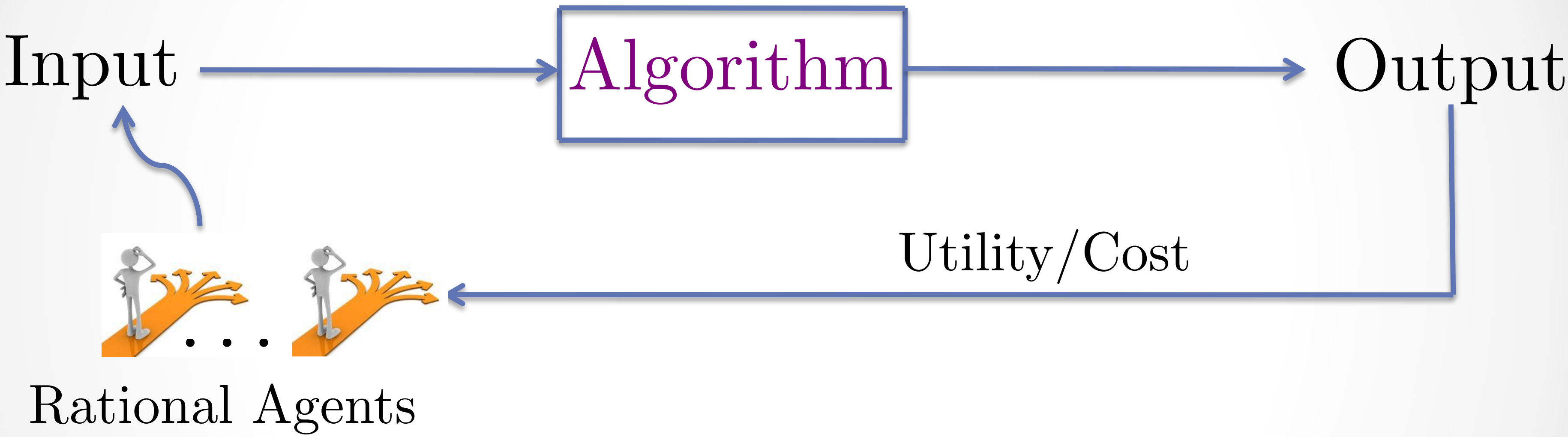
The High Level Idea



How to compare different algorithms for the same problem?

The High Level Idea

Optimization Problem



How to compare different algorithms for the same problem?

(a) Every algorithm defines a game between the agents.

(b) Performance guarantee of an algorithm

$\boxed{= \text{price of anarchy of the corresponding game.}}$

Precise Definition

Consider the underlying optimization problem.

$$\text{Price of Anarchy (PoA)} \ : \ \frac{\text{Objective at the worst equilibrium outcome}}{\text{Optimal objective}}$$

Precise Definition

Consider the underlying optimization problem.

$$\text{Price of Anarchy (PoA)} \; : \; \frac{\text{Objective at the worst equilibrium outcome}}{\text{Optimal objective}}$$

Depends on the solution concept.

(PoA for Pure Nash eq., PoA for Mixed Nash eq. etc.)

Precise Definition

Consider the underlying optimization problem.

$$\text{Price of Anarchy (PoA)} \ : \ \frac{\text{Objective at the worst equilibrium outcome}}{\text{Optimal objective}}$$

Depends on the solution concept.

(PoA for Pure Nash eq., PoA for Mixed Nash eq. etc.)

Will focus on PoA of pure Nash eq.

Past Work

Selfish Routing.

Roughgarden et al. [FOCS 00], Koutsoupias et al. [STACS 99],

Roughgarden [STOC 02, SODA 04, STOC 09], Cole et al. [EC' 03],

Awerbuch et al. [STOC' 05], Christodoulou et al. [ESA' 11] ......

Selfish Scheduling to Minimize Makespan.

Immorlica et al. [WINE 05], Azar et al. [SODA 09],

Caragiannis [SODA 09], Abed et al. [ESA' 12].

Selfish Scheduling for Total Completion Time.

Cole et al. [STOC 11].

Two Techniques

PoA of Smooth Games.

Intrinsic Robustness of the Price of Anarchy.

By Tim Roughgarden. STOC' 09.

PoA via LP/CP duality.

Robust Price of Anarchy Bounds via LP and Fenchel Duality.

Janardhan Kulkarni and Vahab Mirrokni. SODA' 15.

Coordination Mechanism from (almost) all scheduling policies.

B., Im, Kulkarni, Munagala. ITCS' 14.

Two Techniques

PoA of Smooth Games.

Intrinsic Robustness of the Price of Anarchy.

By Tim Roughgarden. STOC' 09.

PoA via LP/CP duality.

Robust Price of Anarchy Bounds via LP and Fenchel Duality.

Janardhan Kulkarni and Vahab Mirrokni. SODA' 15.

Coordination Mechanism from (almost) all scheduling policies.

B., Im, Kulkarni, Munagala. ITCS' 14.

II: Price of Anarchy via LP-duality

Motivation

DNS: Domain Name System

Phone-book of the internet
= DNS servers

www.howstuffworks.com
(domain name)

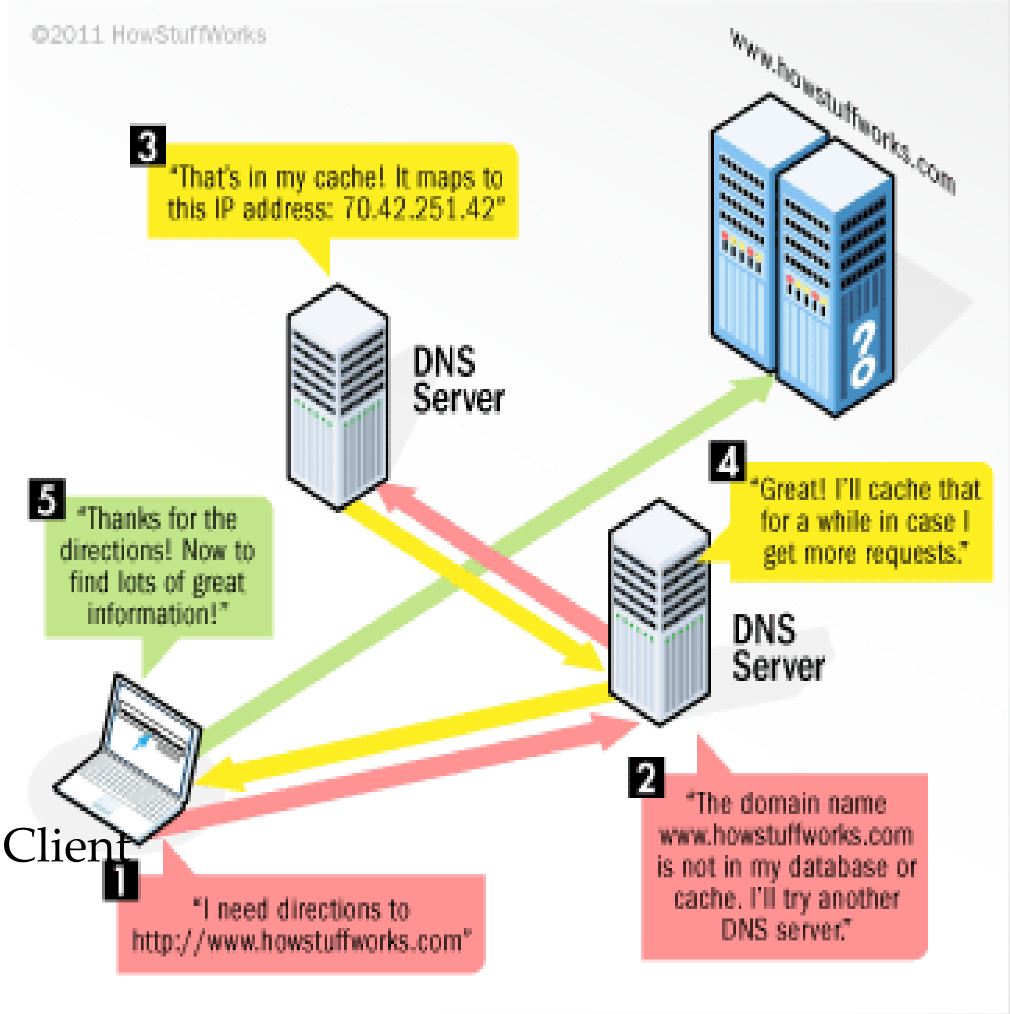
70.42.251.42
(IP address)

DNS: Domain Name System

Phone-book of the internet
= DNS servers

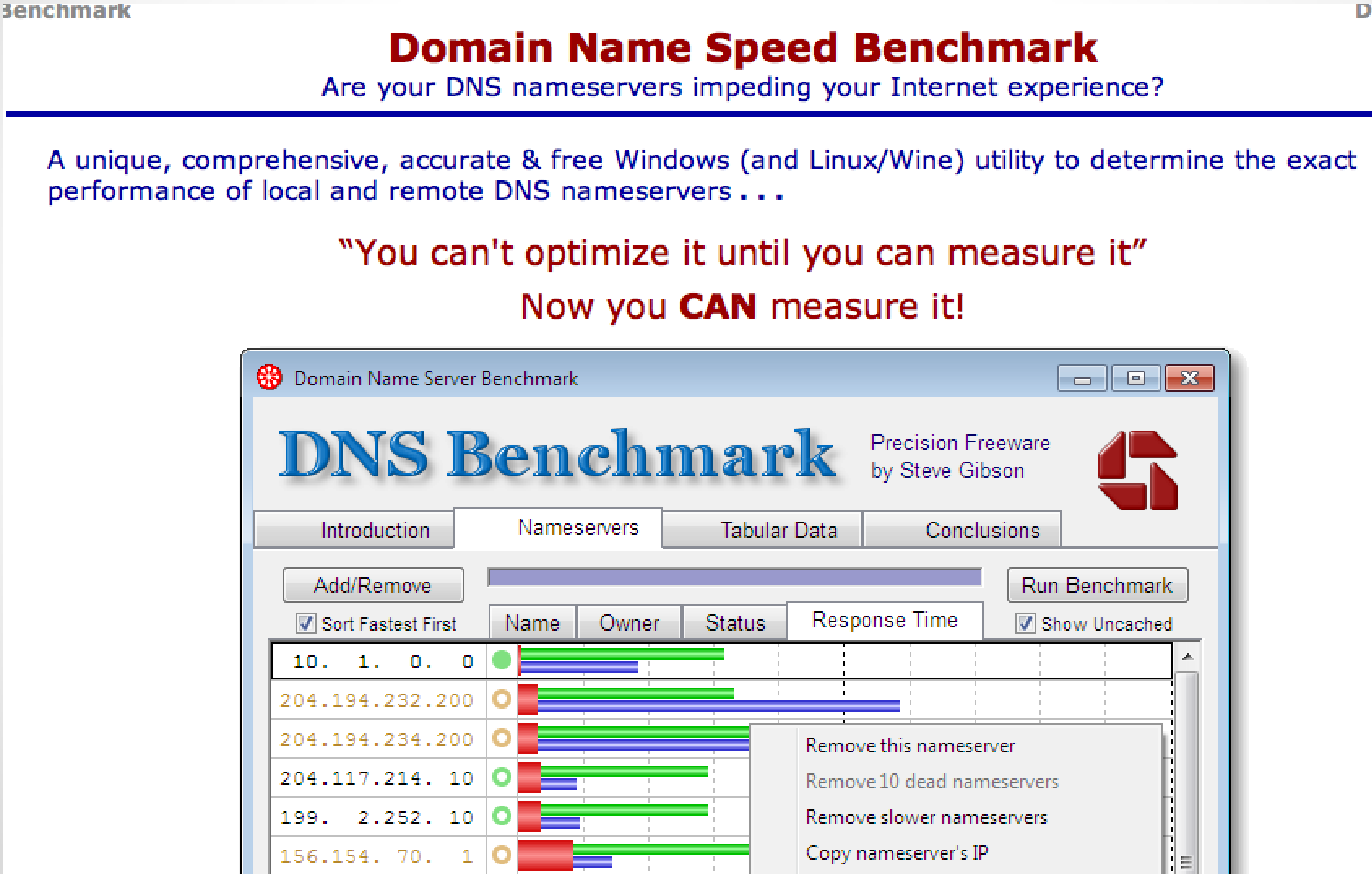
www.howstuffworks.com
(domain name)

70.42.251.42
(IP address)



How does a client select a DNS server?

DNS: Domain Name System

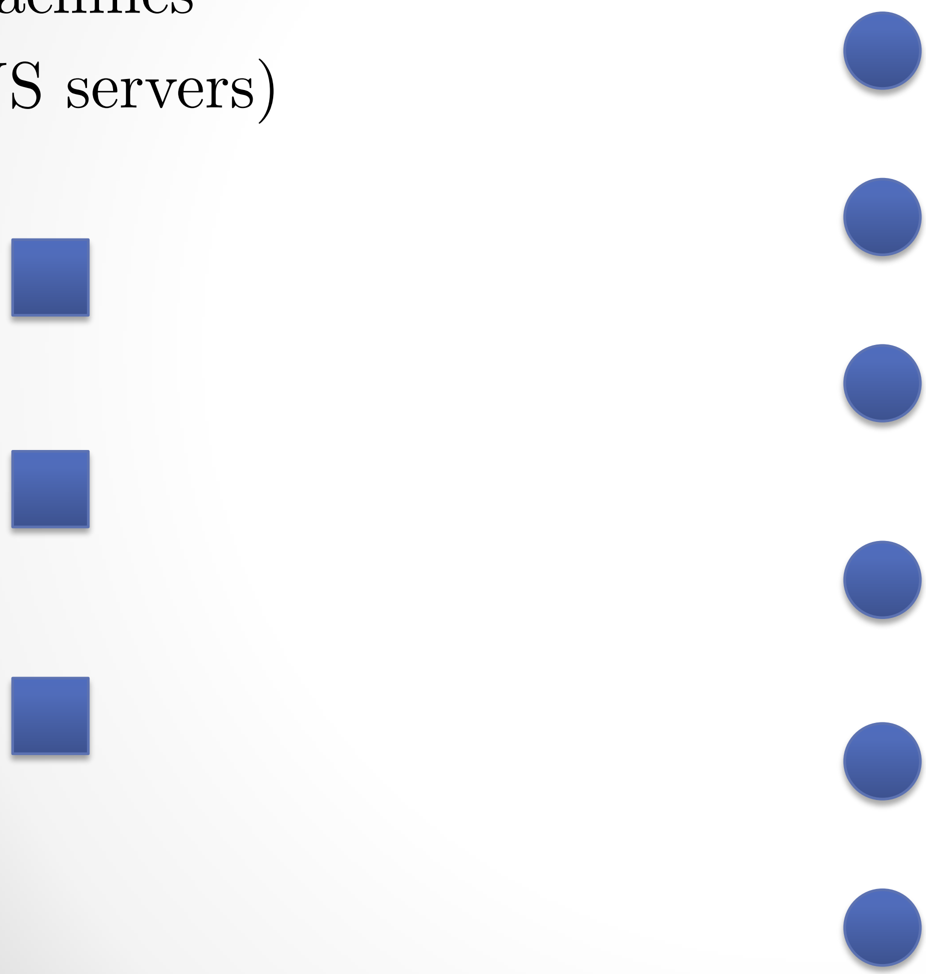


The model

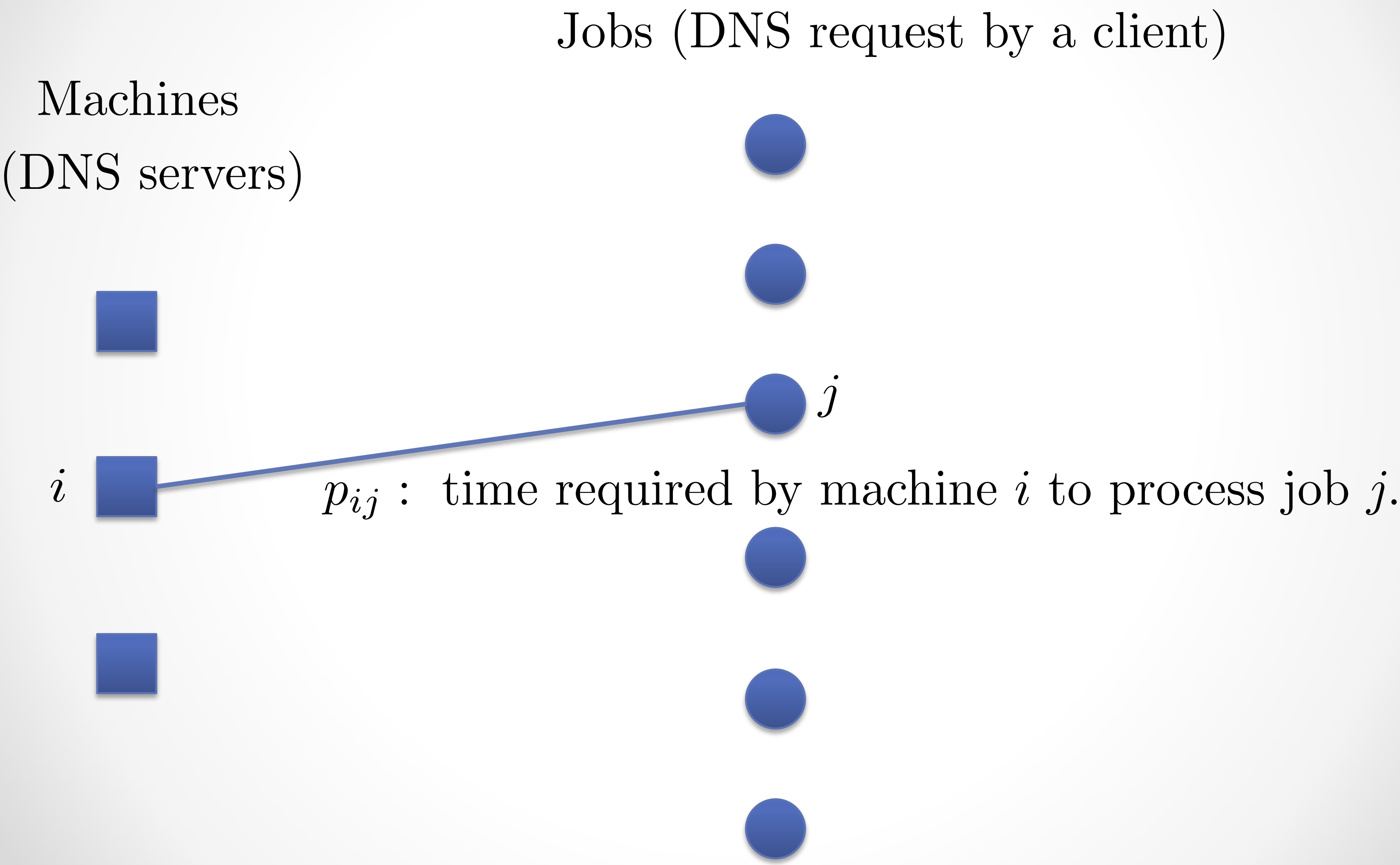
The Scheduling Problem

Jobs (DNS request by a client)

Machines
(DNS servers)



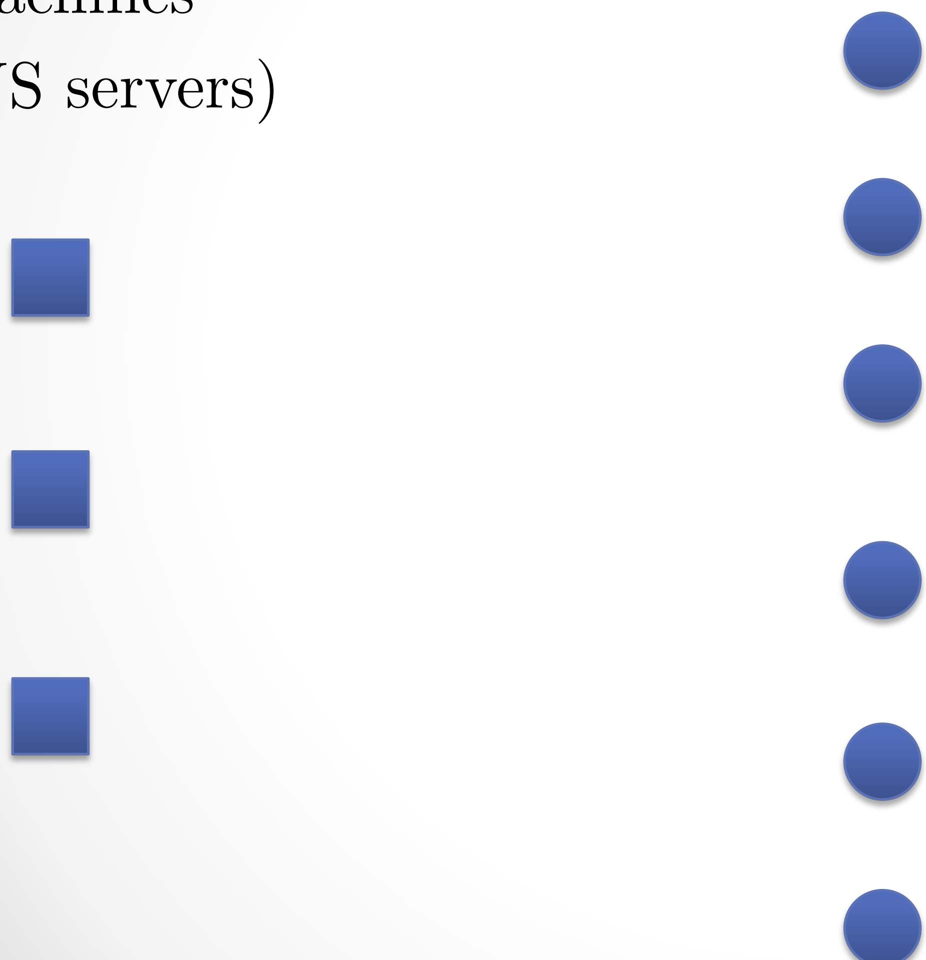
The Scheduling Problem



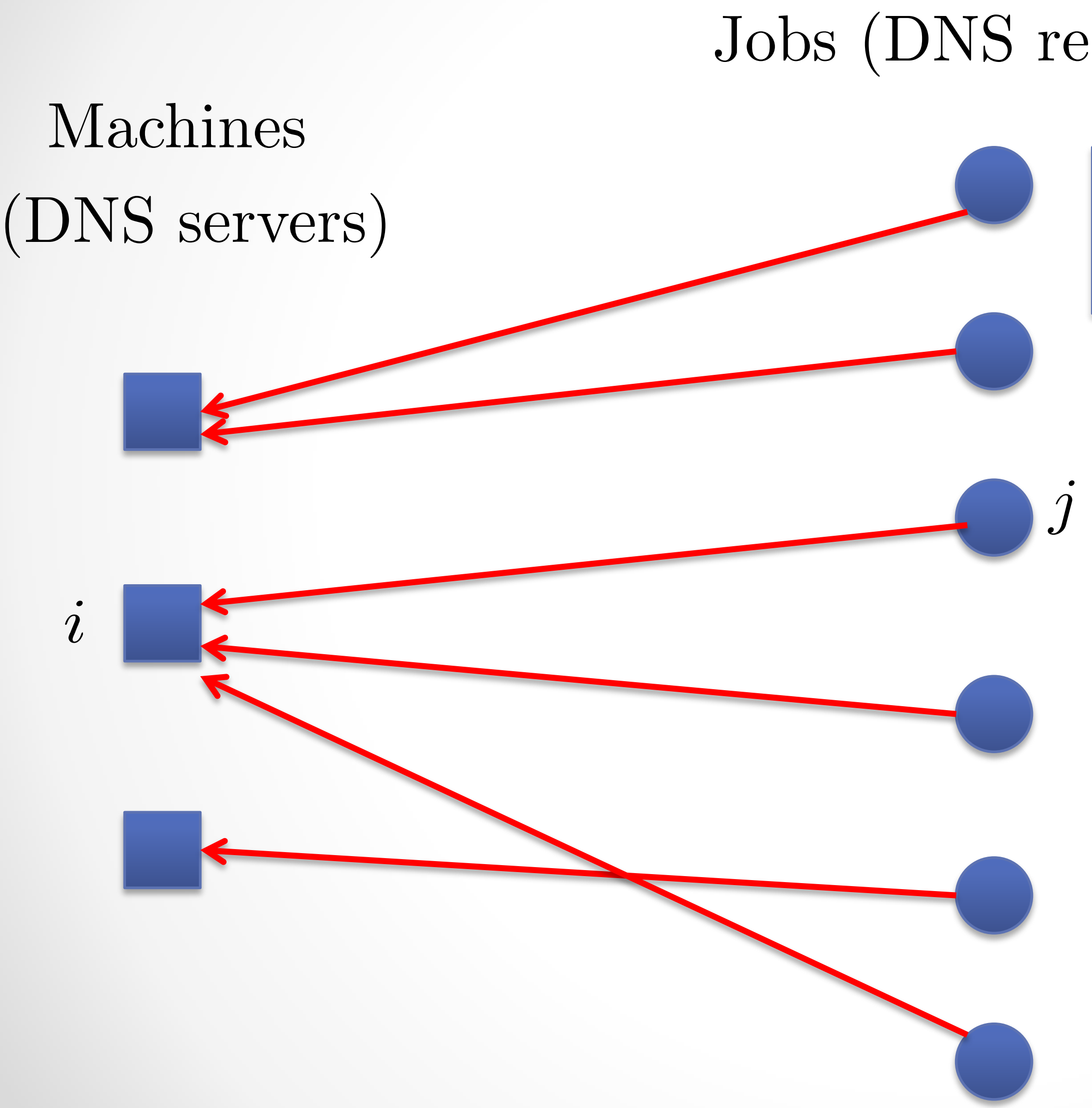
The Scheduling Problem

Jobs (DNS request by a client)

Machines
(DNS servers)



The Scheduling Problem

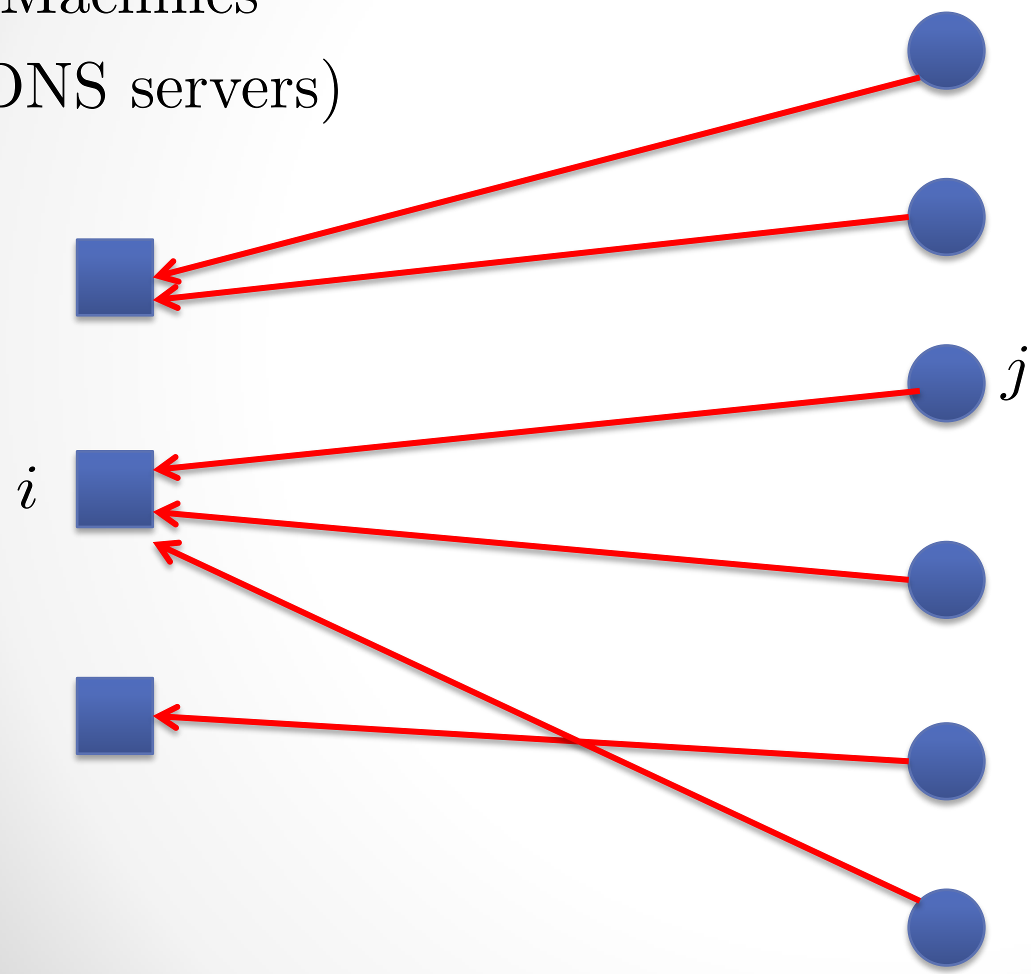


Step 1: Assign each job to a machine.

The Scheduling Problem

Jobs (DNS request by a client)

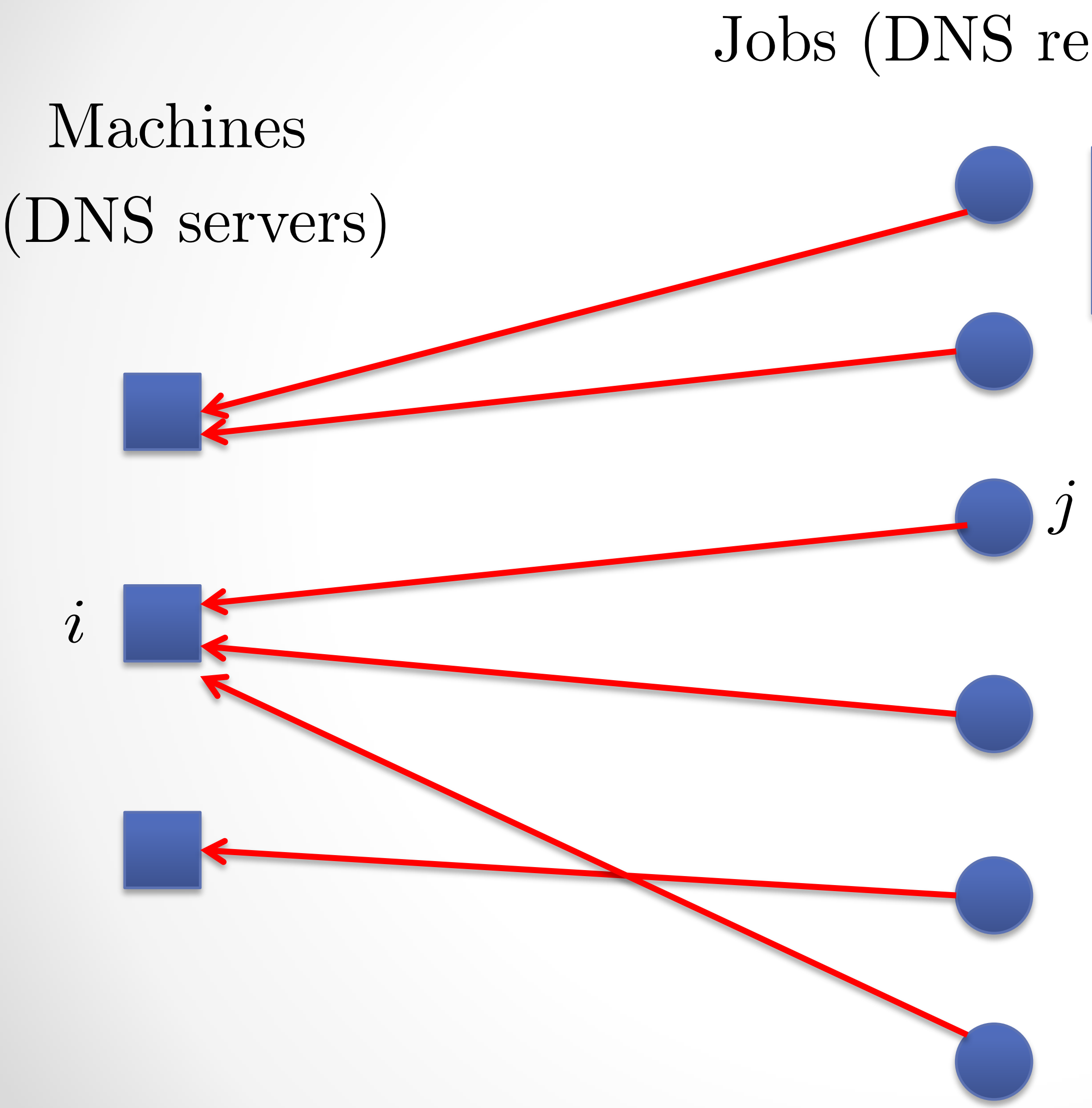
Machines
(DNS servers)



Step 1: Assign each job to a machine.

Step 2: Each machine processes the jobs assigned to it according to some scheduling policy.

The Scheduling Problem

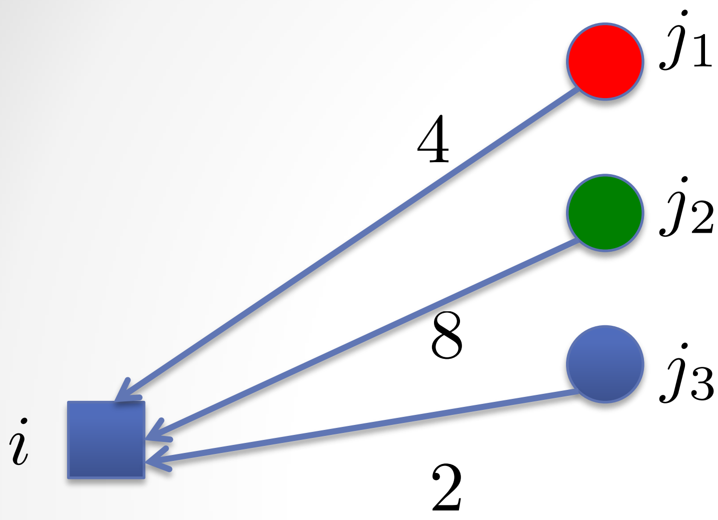


Step 1: Assign each job to a machine.

Step 2: Each machine processes the jobs assigned to it according to some scheduling policy.

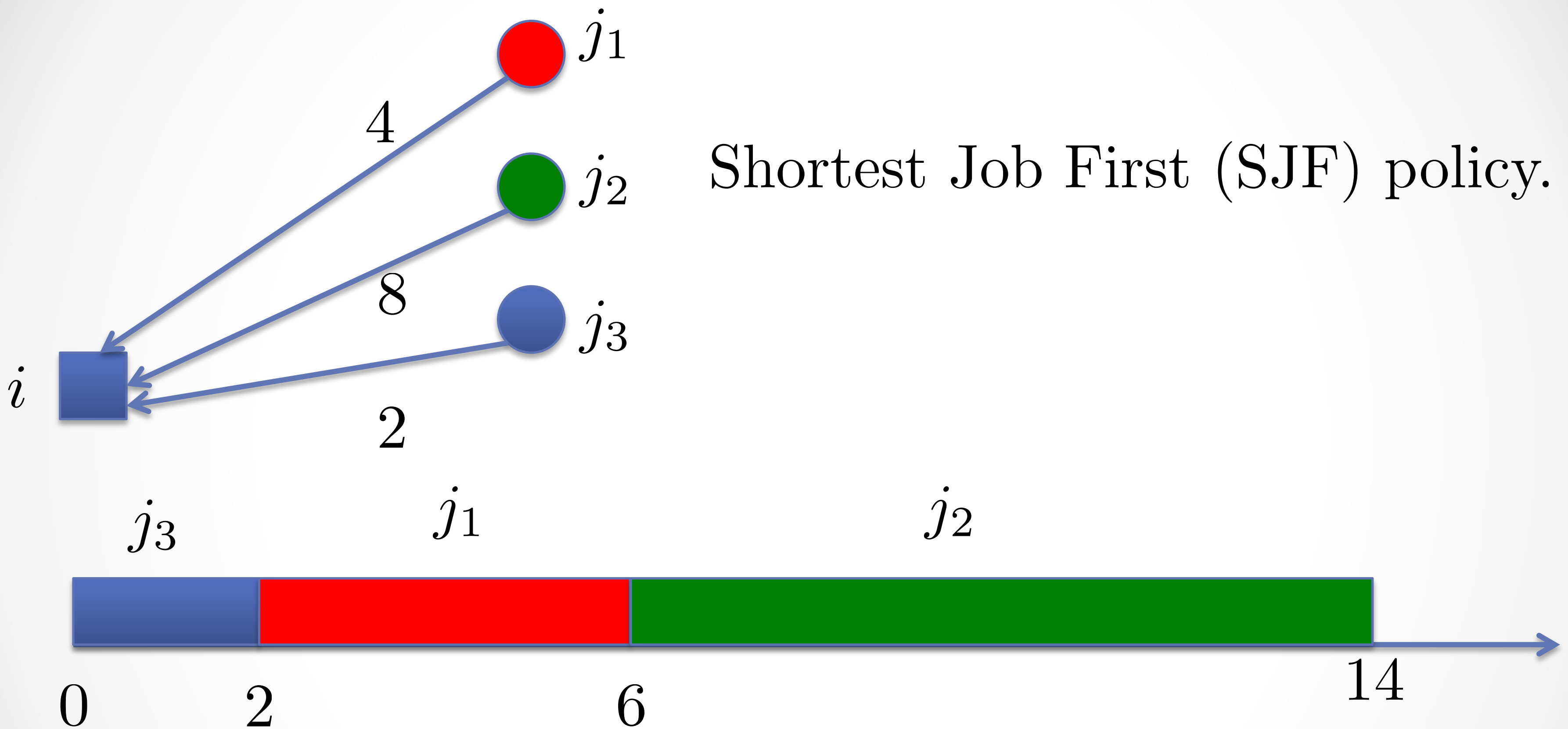
Goal: Minimize the sum of completions times of jobs.

Example of a Scheduling Policy



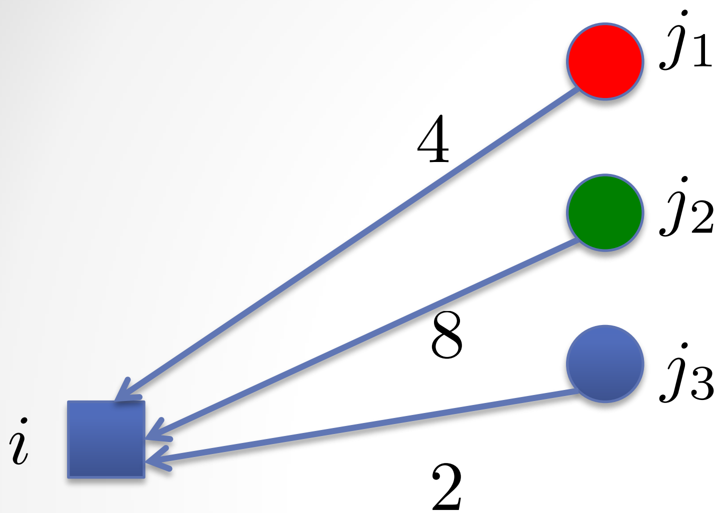
Shortest Job First (SJF) policy.

Example of a Scheduling Policy

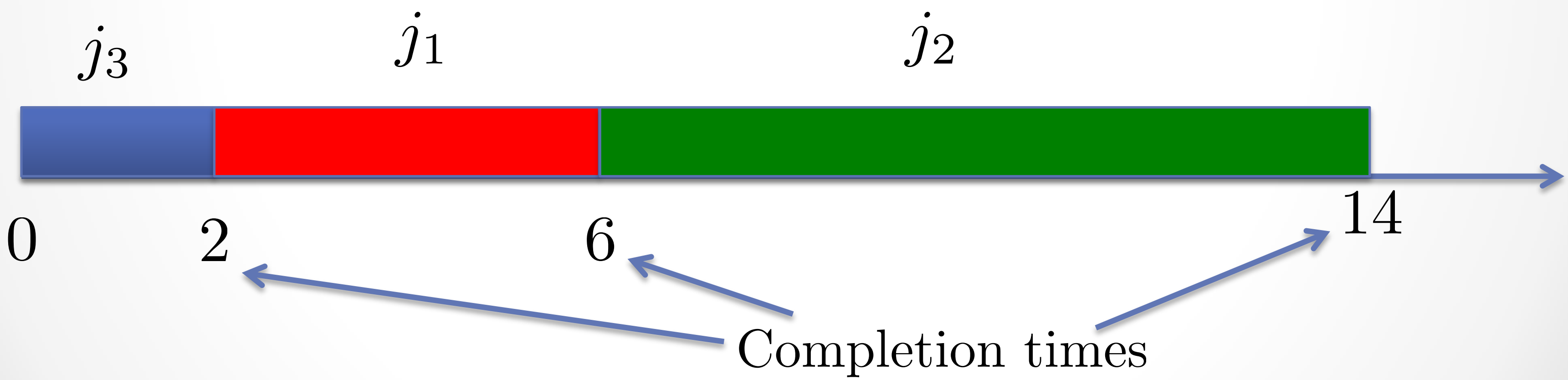


Shortest Job First (SJF) policy.

Example of a Scheduling Policy



Shortest Job First (SJF) policy.



Sum of completion times $= 2 + 6 + 14 = 22$.

Past Work: Approximation Algorithms

Total Completion Time on a Single Machine

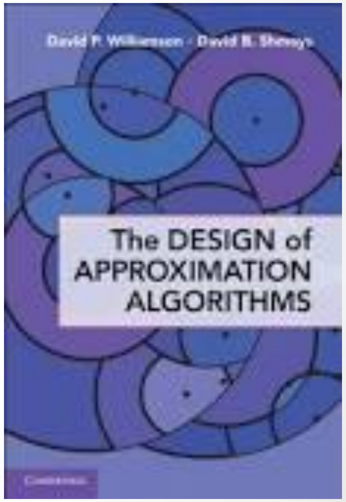
Smith [Naval Res. 56], Hall et al. [SODA 96],

Phillips et al. [Math. Prog. 98], Queyranne [Math. Prog. 93], ……

Total Completion Time on Multiple Machines

Chekuri et al. [SODA 97], Afrati et al. [FOCS 99],

Sethuraman et al. [SODA 99], Skutella [JACM 01], …



Jobs Arriving Online

Garg et al. [STOC 06], Chadhha et al. [STOC 09],

Anand et al. [SODA 12] ……

A decentralized environment

Each job selects *its own* machine.

A decentralized environment

Each job selects *its own* machine.

Each machine executes a *local* scheduling policy.

> * It only sees those jobs that come to it.

A decentralized environment

Each job selects *its own* machine.

Each machine executes a *local* scheduling policy.

* It only sees those jobs that come to it.

Each job wants to minimize *its own* completion time.

* It is a selfish, rational agent.

A decentralized environment

Each job selects *its own* machine.

* The choice depends on (a) the scheduling policies, and
  (b) the strategies of the other jobs.

Each machine executes a *local* scheduling policy.

* It only sees those jobs that come to it.

Each job wants to minimize *its own* completion time.

* It is a selfish, rational agent.

A tug of war

Selfish jobs

System-designer

Price of Anarchy

The scheduling policies define a *game* between the jobs.

The *strategy* of a job is the machine it selects.

Price of Anarchy

The scheduling policies define a *game* between the jobs.

The *strategy* of a job is the machine it selects.

A strategy-profile is in Nash equilibrium iff no job can reduce its completion time by switching to another machine.

Price of Anarchy

The scheduling policies define a *game* between the jobs.

The *strategy* of a job is the machine it selects.

A strategy-profile is in Nash equilibrium iff no job can reduce its completion time by switching to another machine.

Price of anarchy (PoA) : $\dfrac{\text{Objective at the worst Nash equilibrium}}{\text{Optimal objective}}$

Price of Anarchy

The scheduling policies define a *game* between the jobs.

The *strategy* of a job is the machine it selects.

A strategy-profile is in Nash equilibrium iff no job can reduce its completion time by switching to another machine.

Price of anarchy (PoA) : $\frac{\text{Objective at the worst Nash equilibrium}}{\text{Optimal objective}}$

Goal: Design the scheduling policies so as to minimize PoA.

Characterization of Scheduling Polices

A scheduling policy has "fairness" α, iff the delay of any job j due to any other job j' is at most $\alpha \times p_j$.

If α is small, then the policy is fair to every job.

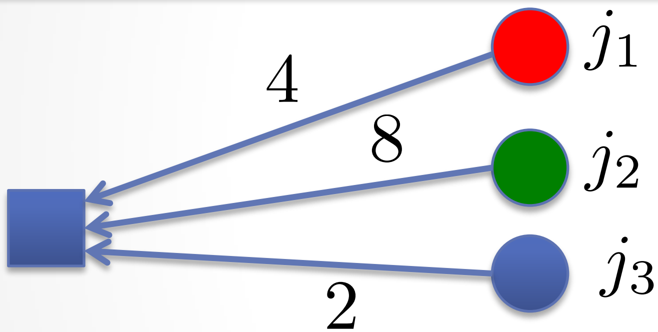
Characterization of Scheduling Polices

A scheduling policy has "fairness" α, iff the delay of any job j due to any other job j' is at most $\alpha \times p_j$.

Shortest Job First (SJF) policy has $\alpha = 1$.

Characterization of Scheduling Polices

A scheduling policy has "fairness" α, iff the delay of any job j due to any other job j' is at most $\alpha \times p_j$.



Shortest Job First (SJF) policy has $\alpha = 1$.

The result

If the machines follow (possibly different) scheduling policies that are α – fair, then the price of anarchy of the induced game is at most 4α. B., Im, Kulkarni, Munagala. *ITCS' 14*.

The result

If the machines follow (possibly different) scheduling policies that are $\alpha-$ fair, then the price of anarchy of the induced game is at most 4α. B., Im, Kulkarni, Munagala. *ITCS' 14*.

Message: Fair policies have small price of anarchy.
Nice guys finish first!

The result

If the machines follow (possibly different) scheduling policies that are α – fair, then the price of anarchy of the induced game is at most 4α. B., Im, Kulkarni, Munagala. *ITCS' 14.*

The result

If the machines follow (possibly different) scheduling policies that are $\alpha-$fair, then the price of anarchy of the induced game is at most 4α. B., Im, Kulkarni, Munagala. *ITCS' 14.*

In the talk, we will only show that the Price of Anarchy of Shortest Job First (SJF) policy is at most 4.

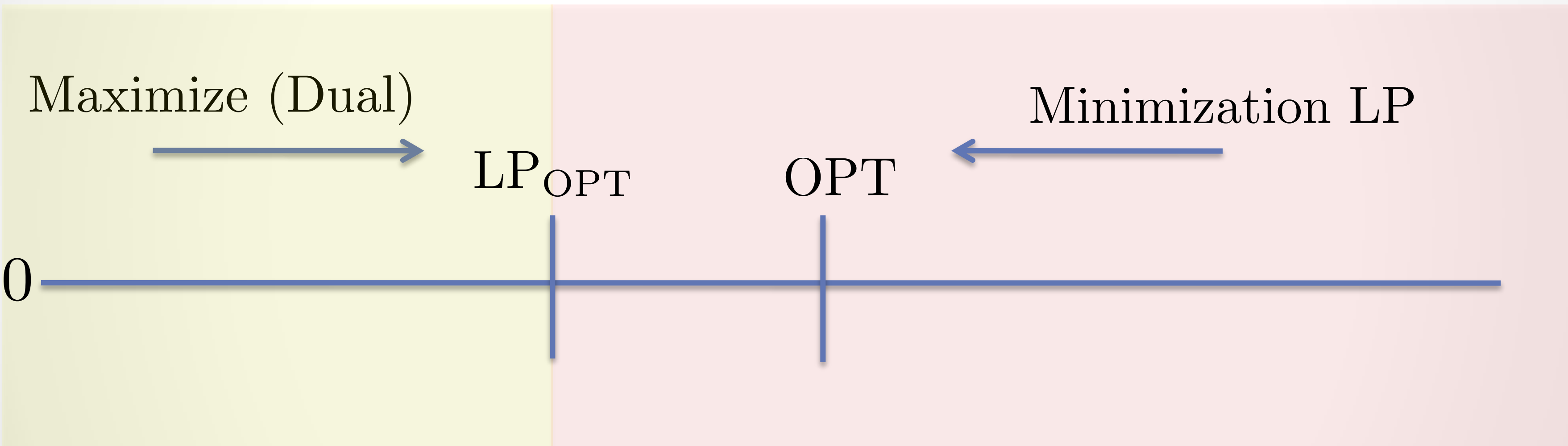
The Technique

Price of Anarchy via Dual Fitting

Find a linear program (LP) relaxation for the optimization problem.

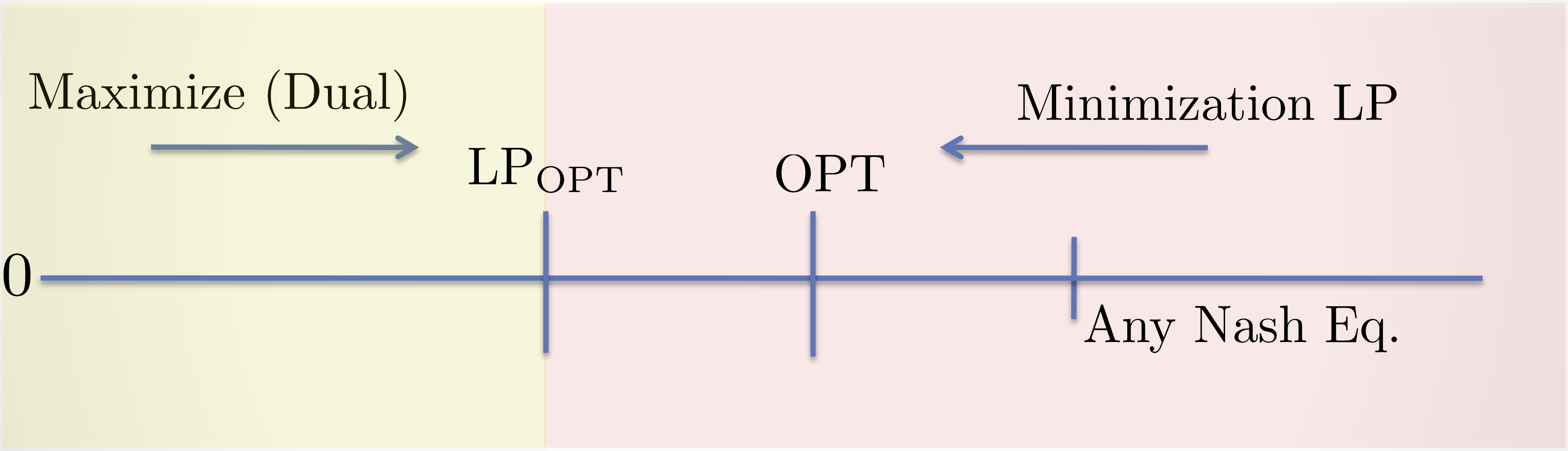
Price of Anarchy via Dual Fitting

Find a linear program (LP) relaxation for the optimization problem.



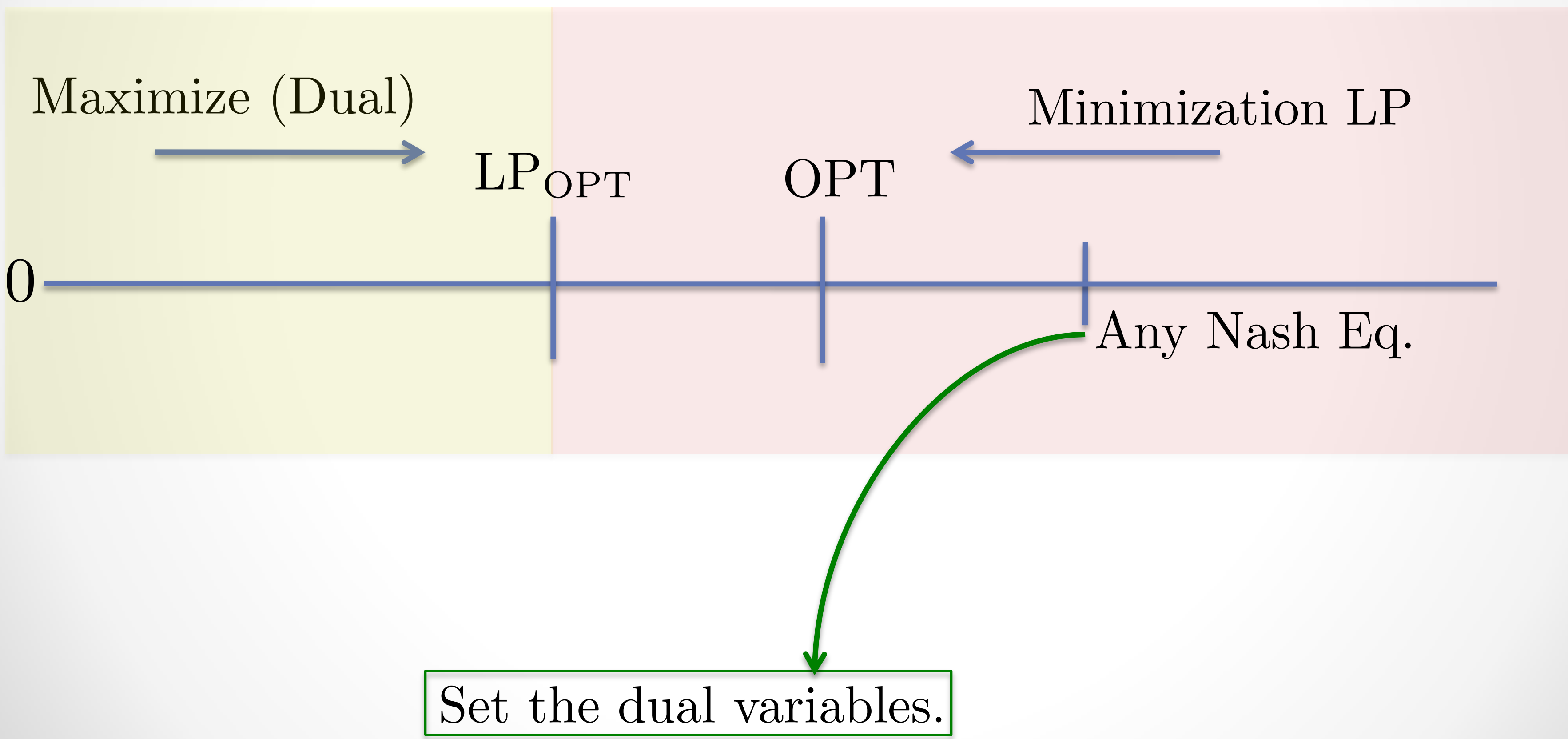
Price of Anarchy via Dual Fitting

Find a linear program (LP) relaxation for the optimization problem.



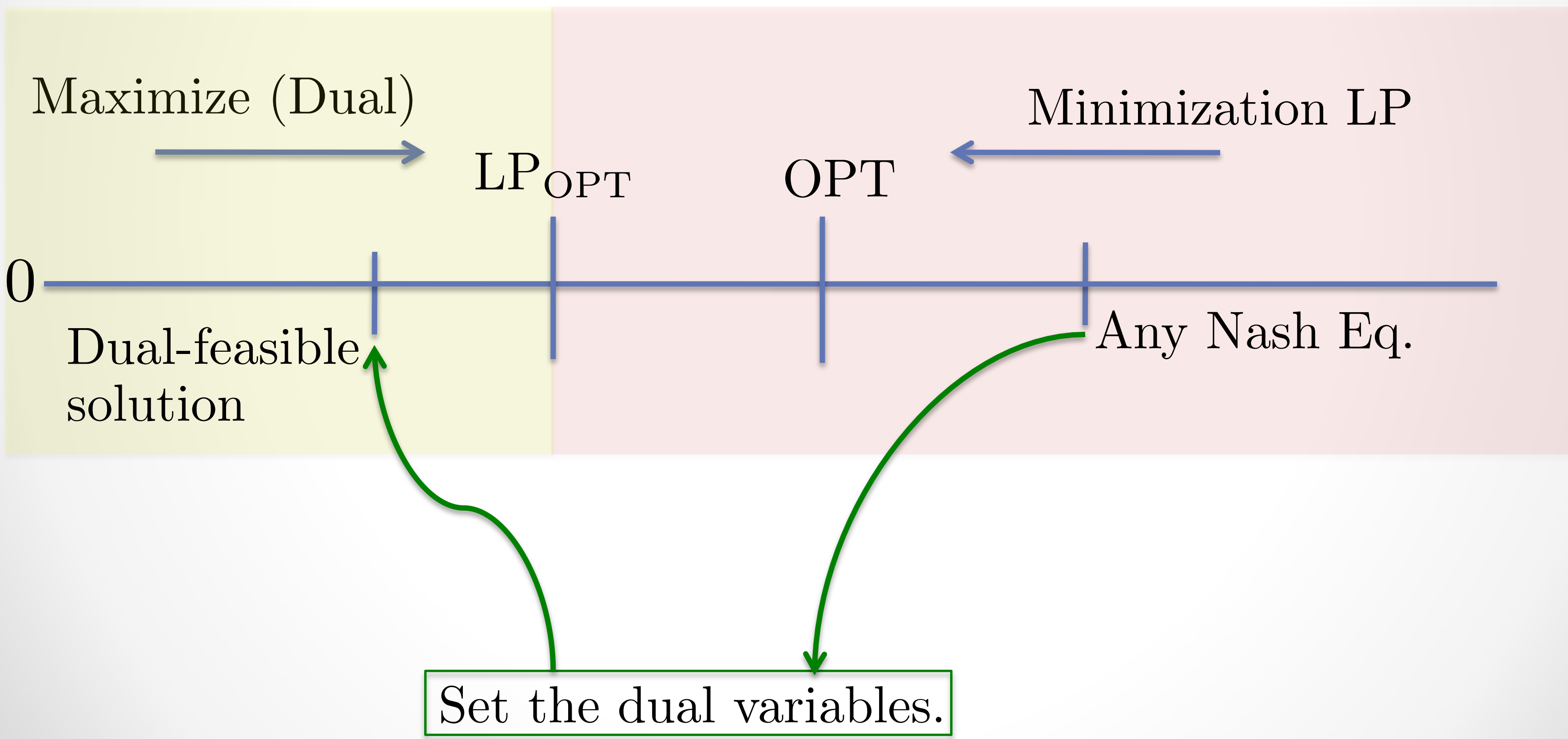
Price of Anarchy via Dual Fitting

Find a linear program (LP) relaxation for the optimization problem.



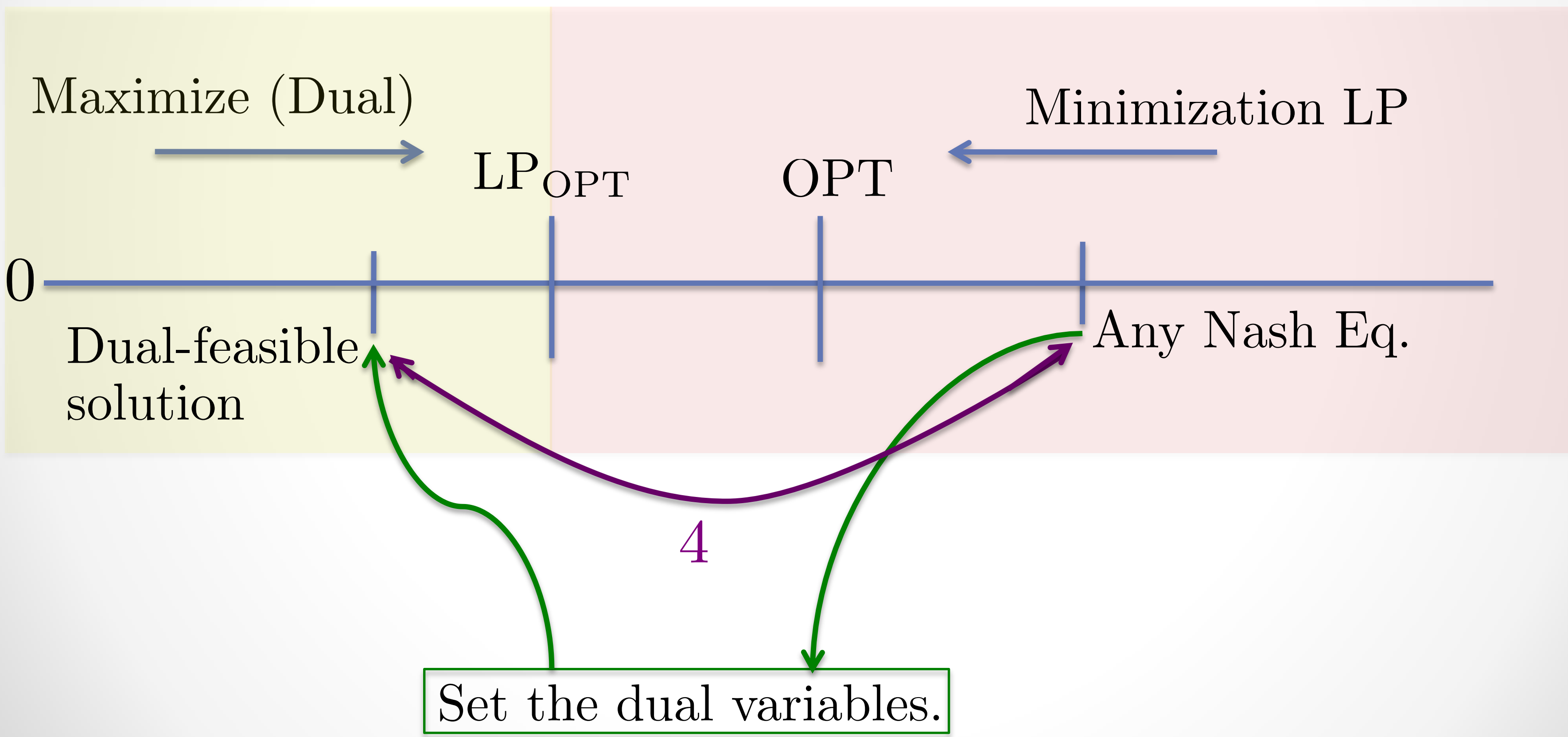
Price of Anarchy via Dual Fitting

Find a linear program (LP) relaxation for the optimization problem.



Price of Anarchy via Dual Fitting

Find a linear program (LP) relaxation for the optimization problem.



Greedy Algorithm: Anand et al. [SODA' 12]

Optimization version. No selfish jobs.

Every machine executes Shortest Job First (SJF) policy.

Greedy Algorithm: Anand et al. [SODA' 12]

Optimization version. No selfish jobs.

Every machine executes Shortest Job First (SJF) policy.

The Algorithm

Only need to find the assignment of the jobs to the machines.

Greedy Algorithm: Anand et al. [SODA' 12]

Optimization version. No selfish jobs.

Every machine executes Shortest Job First (SJF) policy.

The Algorithm

Only need to find the assignment of the jobs to the machines.

* Consider the jobs in arbitrary order.

Greedy Algorithm: Anand et al. [SODA' 12]

Optimization version. No selfish jobs.

Every machine executes Shortest Job First (SJF) policy.

The Algorithm

Only need to find the assignment of the jobs to the machines.

* Consider the jobs in arbitrary order.

* While considering a job j, assign it to a machine which increases the overall objective by the least amount.

Greedy Algorithm: Anand et al. [SODA' 12]

Optimization version. No selfish jobs.

Every machine executes Shortest Job First (SJF) policy.

Analysis by Dual Fitting!!

The Algorithm

Only need to find the assignment of the jobs to the machines.

* Consider the jobs in arbitrary order.

* While considering a job j, assign it to a machine which increases the overall objective by the least amount.

A New Take on Price of Anarchy

Greedy Algorithm

Price of Anarchy

System Designer

A New Take on Price of Anarchy

Greedy Algorithm

System Designer

Price of Anarchy

I minimize the sum of costs incurred by all of you, greedily.

Jobs

A New Take on Price of Anarchy

Greedy Algorithm

I minimize the sum of costs incurred by all of you, greedily.

System Designer

Price of Anarchy

Jobs

A New Take on Price of Anarchy

Greedy Algorithm

I minimize the sum of costs incurred by all of you, greedily.

System Designer

Price of Anarchy

I set the rules of the game. Now you are free.

Jobs

Jobs

A New Take on Price of Anarchy

Greedy Algorithm

I minimize the sum of costs incurred by all of you, greedily.

System Designer

Price of Anarchy

I set the rules of the game. Now you are free.

Jobs

Locally greedy algorithm!

Free agents

Greedy (selfish)

Jobs

A New Take on Price of Anarchy

Greedy Algorithm

I minimize the sum of costs incurred by all of you, greedily.

System Designer

Price of Anarchy

I set the rules of the game. Now you are free.

Locally greedy algorithm!

Jobs

Free agents

Greedy (selfish)

Jobs

Proof sketch

The LP [Anand et al., SODA' 12]

x_{ijt} : Denotes if machine i works on job j at time t.

The LP [Anand et al., SODA' 12]

x_{ijt} : Denotes if machine i works on job j at time t.

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j. A job is fully processed.

The LP [Anand et al., SODA' 12]

x_{ijt} : Denotes if machine i works on job j at time t.

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j. A job is fully processed.

$\sum_j x_{ijt} \leq 1 \;\; \forall$ machines i, times t. A machine finishes at most one unit of the jobs per unit time-step.

The LP [Anand et al., SODA' 12]

x_{ijt} : Denotes if machine i works on job j at time t.

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j. A job is fully processed.

$\sum_j x_{ijt} \leq 1 \;\; \forall$ machines i, times t. A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \;\; \forall \; i, j, t.$

The LP [Anand et al., SODA' 12]

x_{ijt} : Denotes if machine i works on job j at time t.

$$\text{Min. } \sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \sum_t (1/2) \cdot x_{ijt} \right\}$$

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j. — A job is fully processed.

$\sum_j x_{ijt} \leq 1 \;\; \forall$ machines i, times t. — A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \;\; \forall\, i, j, t.$

The LP [Anand et al., SODA' 12]

x_{ijt} : Denotes if machine i works on job j at time t.

fractional completion time

total processing time

$$\text{Min. } \sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \sum_t (1/2) \cdot x_{ijt} \right\}$$

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j. A job is fully processed.

$\sum_j x_{ijt} \leq 1 \;\; \forall$ machines i, times t. A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \;\; \forall\, i, j, t.$

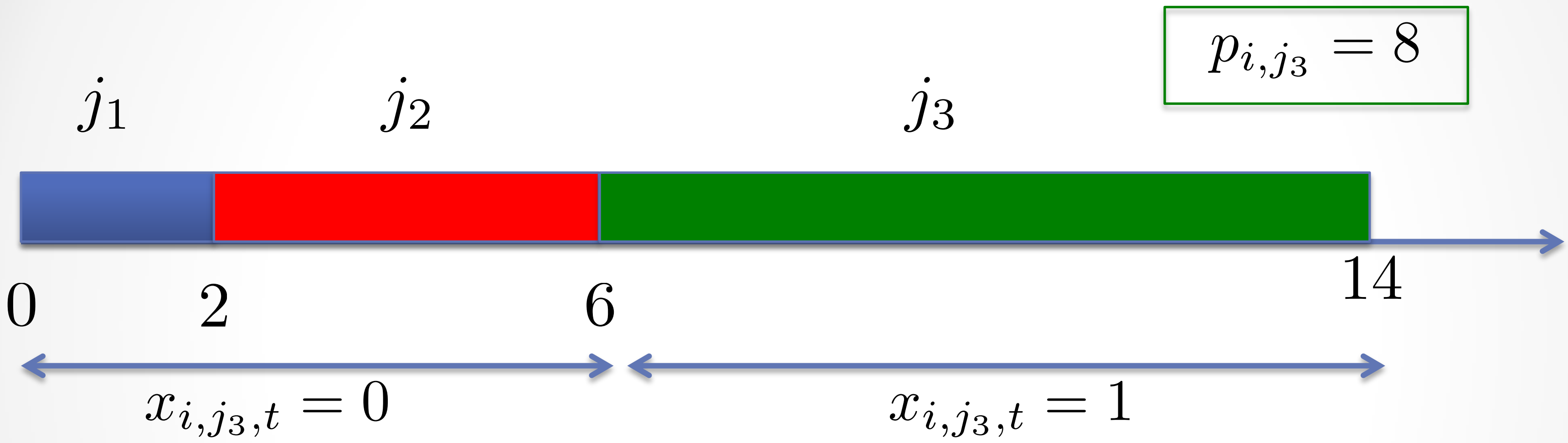
Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$$\boxed{p_{i,j_3} = 8}$$

Fractional Completion Time

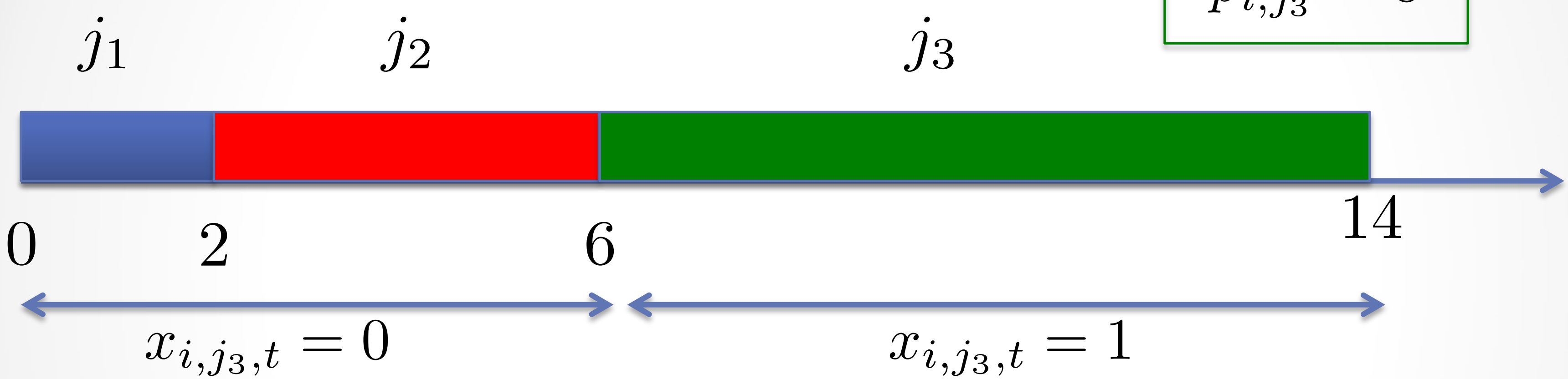
x_{ijt} : Denotes if machine i works on job j at time t.



Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$$p_{i,j_3} = 8$$

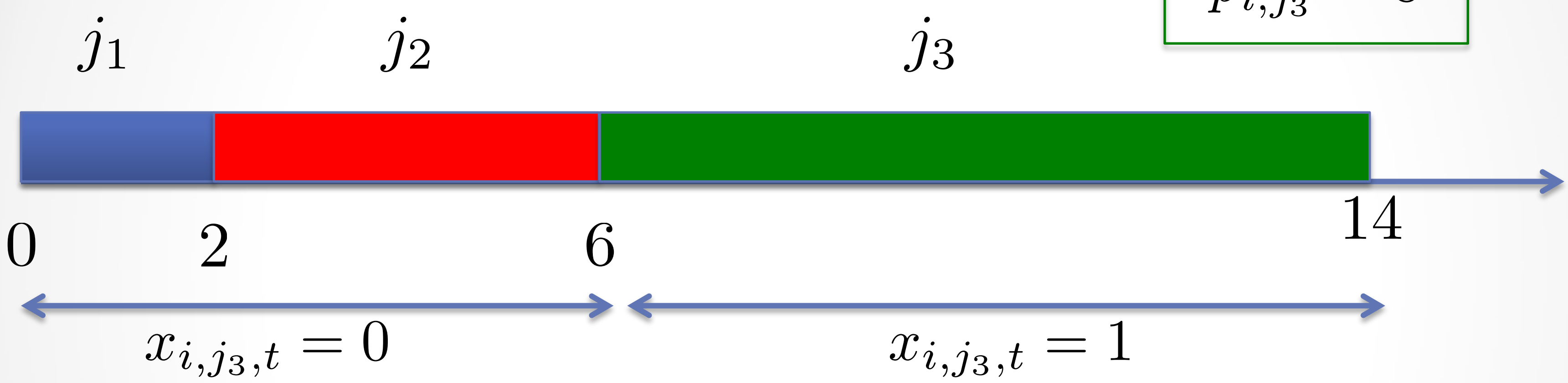


$$\text{Fractional completion time of } j_3 = \frac{\sum_t (x_{i,j_3,t}) \cdot t}{p_{i,j_3}}$$

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$p_{i,j_3} = 8$



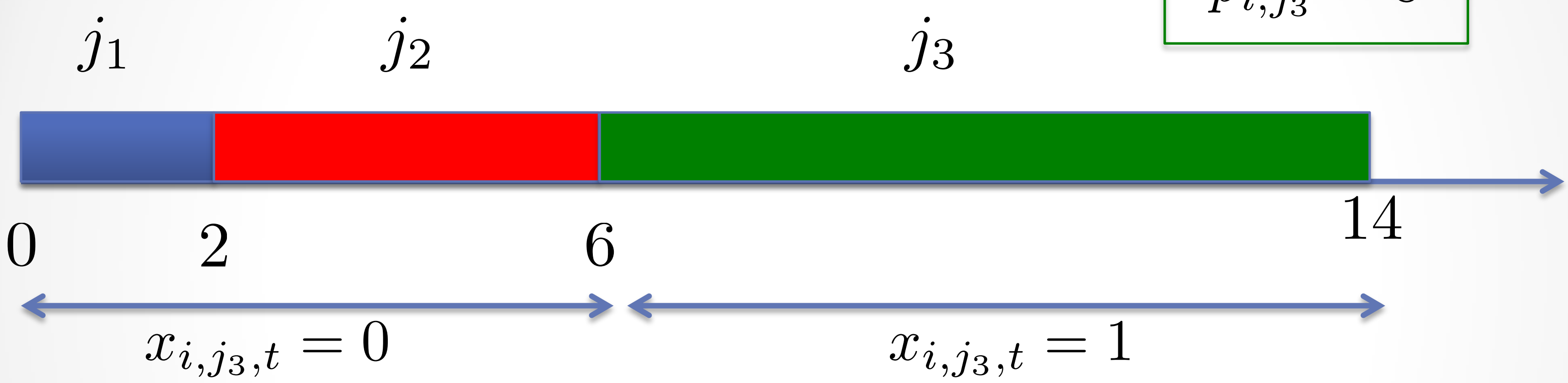
$$\text{Fractional completion time of } j_3 = \frac{\sum_t (x_{i,j_3,t}) \cdot t}{p_{i,j_3}}$$

$$= \frac{7+8+9+10+11+12+13+14}{8}$$

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$p_{i,j_3} = 8$



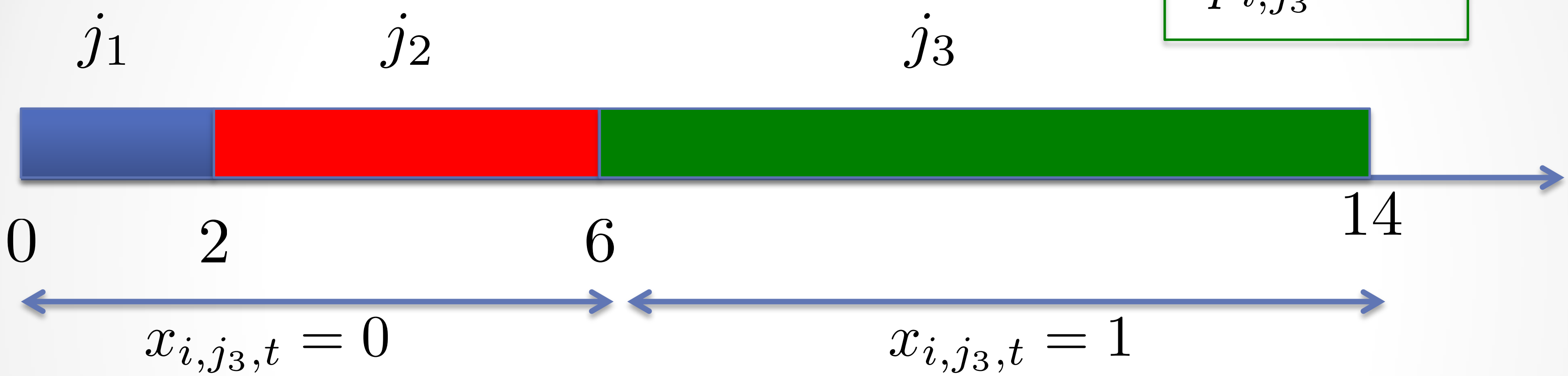
$$\text{Fractional completion time of } j_3 = \frac{\sum_t (x_{i,j_3,t}) \cdot t}{p_{i,j_3}}$$

$$= \frac{7+8+9+10+11+12+13+14}{8} = 10.5$$

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$\boxed{p_{i,j_3} = 8}$



$$\text{Fractional completion time of } j_3 = \frac{\sum_t (x_{i,j_3,t}) \cdot t}{p_{i,j_3}}$$

$$= \frac{7+8+9+10+11+12+13+14}{8} = 10.5 \leq \text{Completion time of } j_3.$$

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

fractional completion time   total processing time

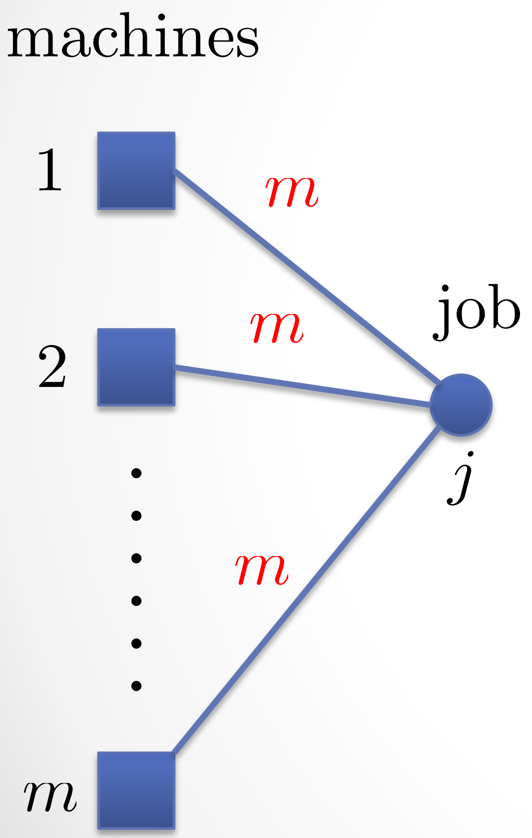
$$\text{Min.} \ \sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \sum_t (1/2) \cdot x_{ijt} \right\}$$

Why do we need the second term in the LP-objective?

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$p_{i,j} = m$ for every machine i.

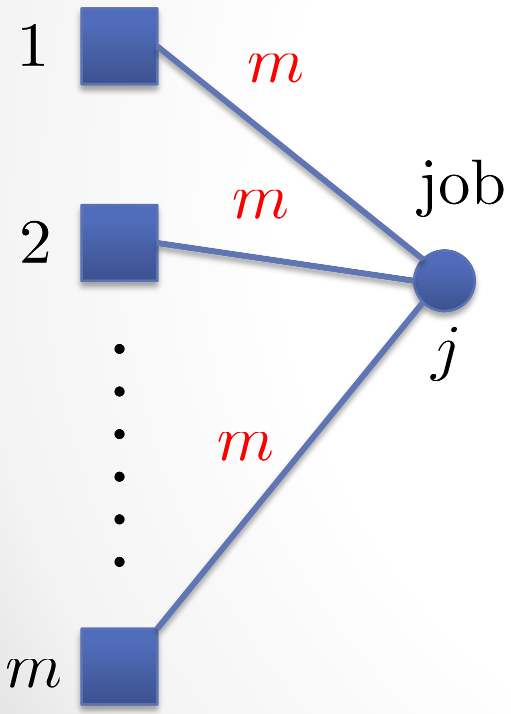


Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

machines

$p_{i,j} = m$ for every machine i.



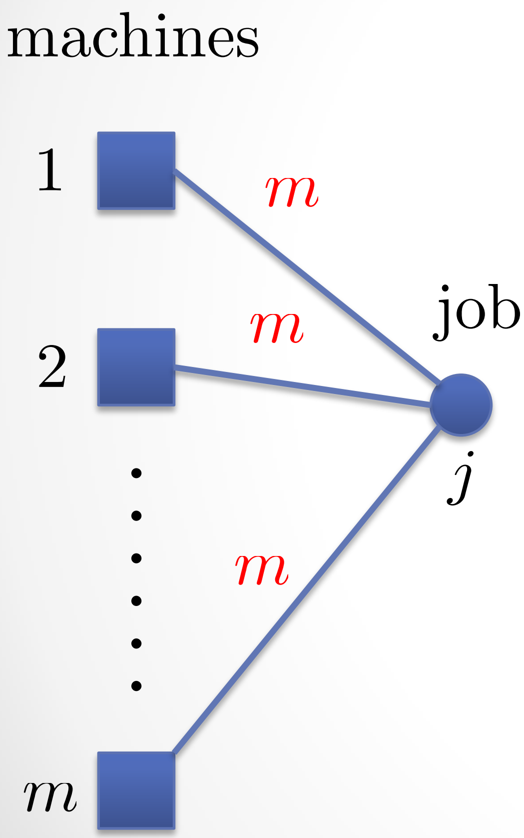
Divide the job equally among the machines.

$x_{i,j,1} = 1$ for every machine i.

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$p_{i,j} = m$ for every machine i.



Divide the job equally among the machines.

$x_{i,j,1} = 1$ for every machine i.

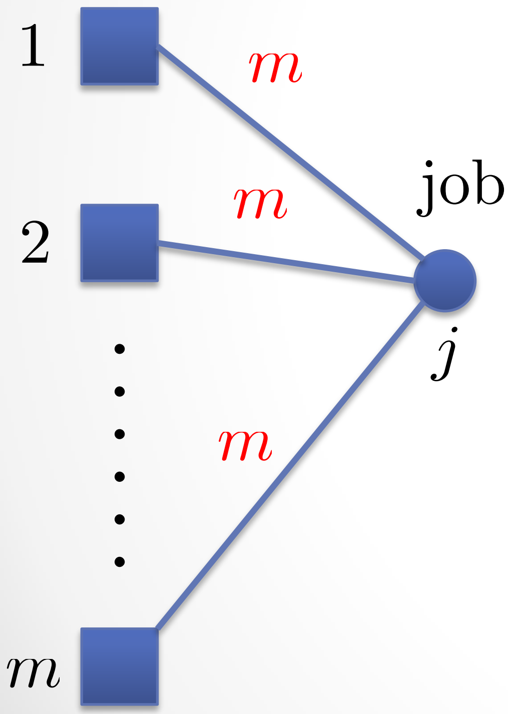
$\sum_{i,t} x_{i,j,t} = m$ (the job is fully processed)

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

machines

$p_{i,j} = m$ for every machine i.



Divide the job equally among the machines.

$x_{i,j,1} = 1$ for every machine i.

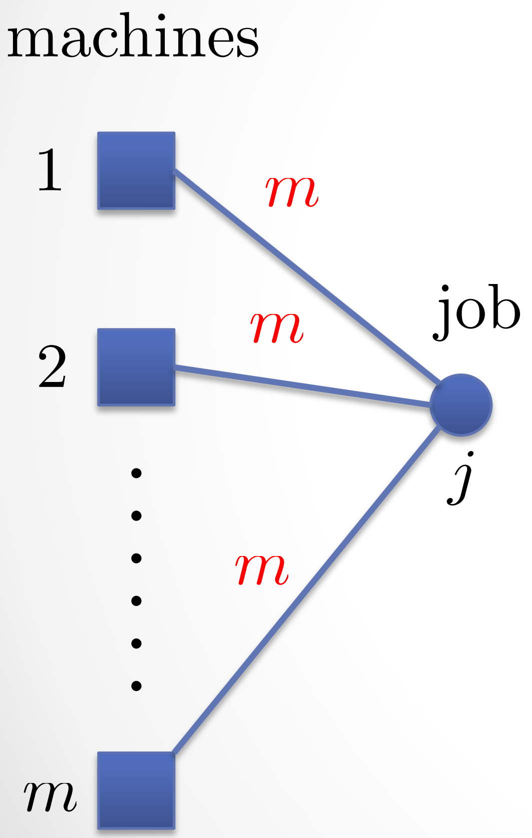
$\sum_{i,t} x_{i,j,t} = m$ (the job is fully processed)

$$\frac{\sum_{i,t}(x_{i,j,t}) \cdot t}{p_{i,j}}$$

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.

$p_{i,j} = m$ for every machine i.



Divide the job equally among the machines.

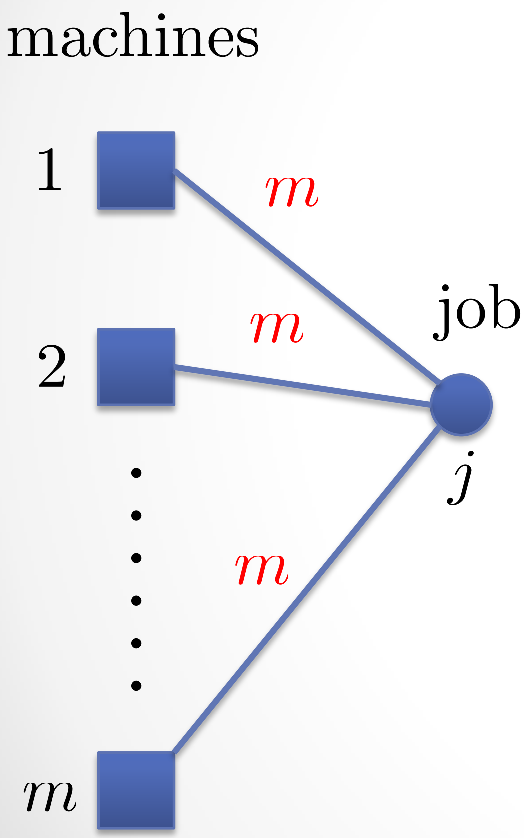
$x_{i,j,1} = 1$ for every machine i.

$\sum_{i,t} x_{i,j,t} = m$ (the job is fully processed)

$$\frac{\sum_{i,t}(x_{i,j,t}) \cdot t}{p_{i,j}} = \frac{\sum_{i=1}^{m} 1 \cdot 1}{m} = 1$$

Fractional Completion Time

x_{ijt} : Denotes if machine i works on job j at time t.



$p_{i,j} = m$ for every machine i.

Fractional completion time $= 1$!

Divide the job equally among the machines.

$x_{i,j,1} = 1$ for every machine i.

$\sum_{i,t} x_{i,j,t} = m$ (the job is fully processed)

$$\frac{\sum_{i,t}(x_{i,j,t}) \cdot t}{p_{i,j}} = \frac{\sum_{i=1}^{m} 1 \cdot 1}{m} = 1$$

LP-relaxation

x_{ijt} : Denotes if machine i works on job j at time t.

fractional completion time

total processing time

$$\text{Min. } \sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \sum_t (1/2) \cdot x_{ijt} \right\}$$

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j. A job is fully processed.

$\sum_j x_{ijt} \leq 1 \;\; \forall$ machines i, times t. A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \;\; \forall\, i, j, t.$

The LP on a machine with ½ speed

x_{ijt} : Denotes if machine i works on job j at time t.

fractional completion time

total processing time

$$\text{Min. } \sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \boxed{\sum_t x_{ijt}} \right\}$$

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \ \ \forall$ jobs j.

A job is fully processed.

$\sum_j x_{ijt} \leq \boxed{1/2} \ \forall$ machines i, times t.

A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \ \ \forall\ i, j, t.$

The LP on a machine with ½ speed

x_{ijt} : Denotes if machine i works on job j at time t.

fractional completion time

total processing time

Min. $\sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \sum_t x_{ijt} \right\}$

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \quad \forall$ jobs j.

A job is fully processed.

$\sum_j x_{ijt} \leq 1/2 \quad \forall$ machines i, times t.

A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \quad \forall\ i, j, t.$

The LP on a machine with ½ speed

x_{ijt} : Denotes if machine i works on job j at time t.

fractional completion time

total processing time

C_j

Min. $\sum_j \sum_i \left\{ \sum_t x_{ijt} \cdot (t/p_{ij}) \right\} + \sum_j \sum_i \left\{ \sum_t x_{ijt} \right\}$

$N_{i,t}$

$\sum_i \sum_t (x_{ijt}/p_{ij}) \geq 1 \;\; \forall$ jobs j.

A job is fully processed.

$\sum_j x_{ijt} \leq 1/2 \;\; \forall$ machines i, times t.

A machine finishes at most one unit of the jobs per unit time-step.

$x_{ijt} \geq 0 \;\; \forall\, i, j, t.$

The dual LP

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \; \forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \;\; \forall \; i, j, t.$

Setting the dual variables

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}$ $\forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \ \ \forall\ i, j, t.$

Setting the dual variables

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}\ \forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$

$*$ Fix any Nash equilibrium $\vec{\theta} = (\vec{\theta}_1, \ldots, \vec{\theta}_j, \ldots, \vec{\theta}_m)$.

Here, $\vec{\theta}_j$ denotes the machine chosen by the job j.

$*$ $C_j \leftarrow$ Completion time of job j under $\vec{\theta}$.

$*$ $N_{it} \leftarrow$ #Unfinished jobs on machine i at time t, under $\vec{\theta}$.

Dual Objective

$$\text{Max.} \quad \sum_j C_j - 1/2 \sum_i \sum_t N_{it}$$

$$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \ \forall \text{ jobs } j, \ \text{machines } i, \ \text{times } t.$$

$$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$$

Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}$ $\forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.

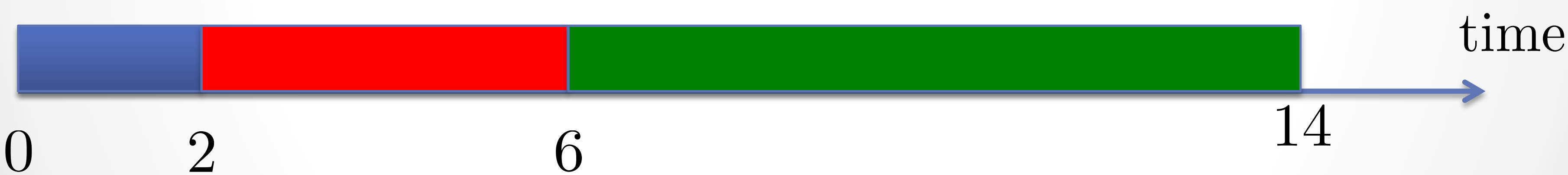
Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}\ \forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



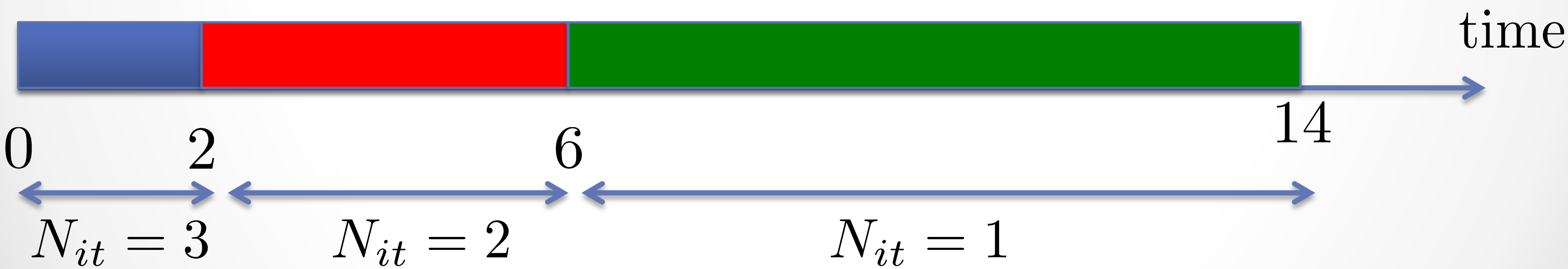
Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \;\forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\; i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



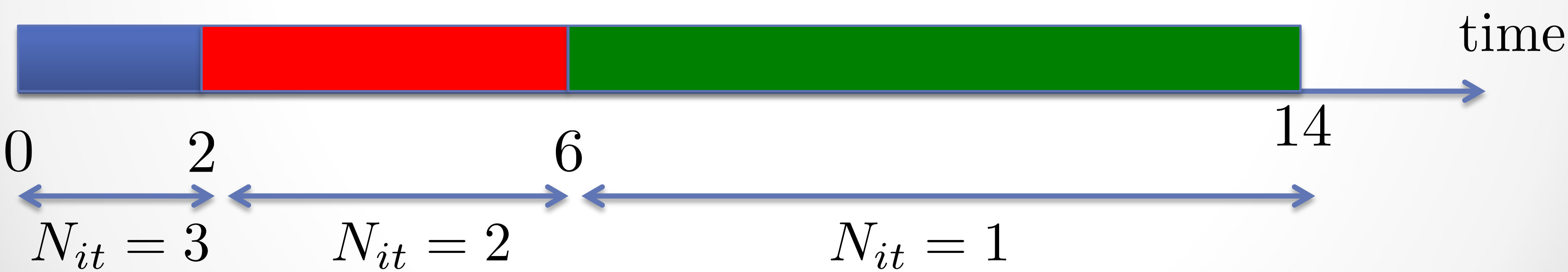
Dual Objective

$$\text{Max. } \sum_j C_j - 1/2 \sum_i \sum_t N_{it}$$

$$C_j - t \le p_{ij} + p_{ij} \cdot N_{it} \;\forall \text{ jobs } j, \text{ machines } i, \text{ times } t.$$

$$C_j, N_{it} \ge 0 \quad \forall\, i, j, t.$$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



$$\sum_t N_{it} = (3 \times 2) + (2 \times 4) + (8 \times 1) = 22 = \sum_j C_j$$

Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}\ \forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.

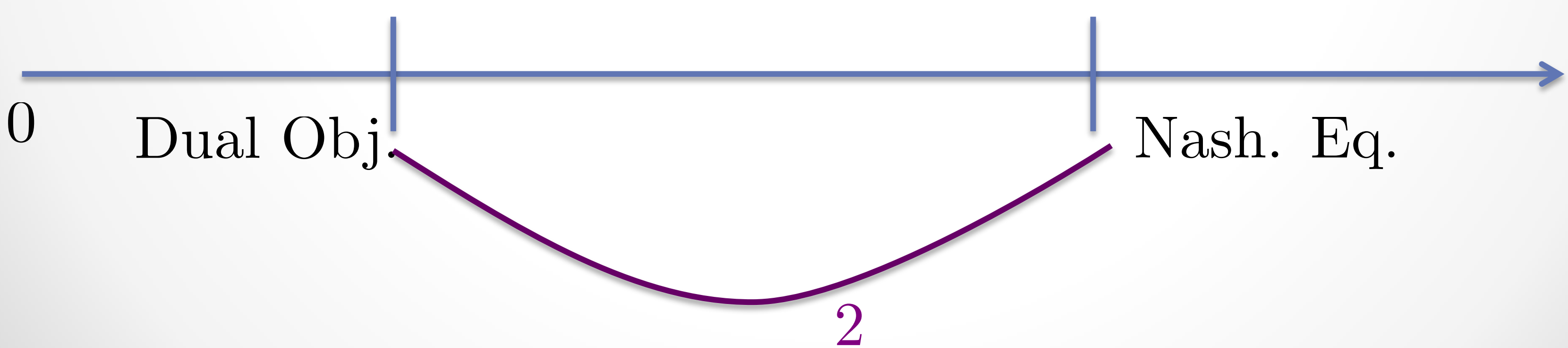
Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}$ $\forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



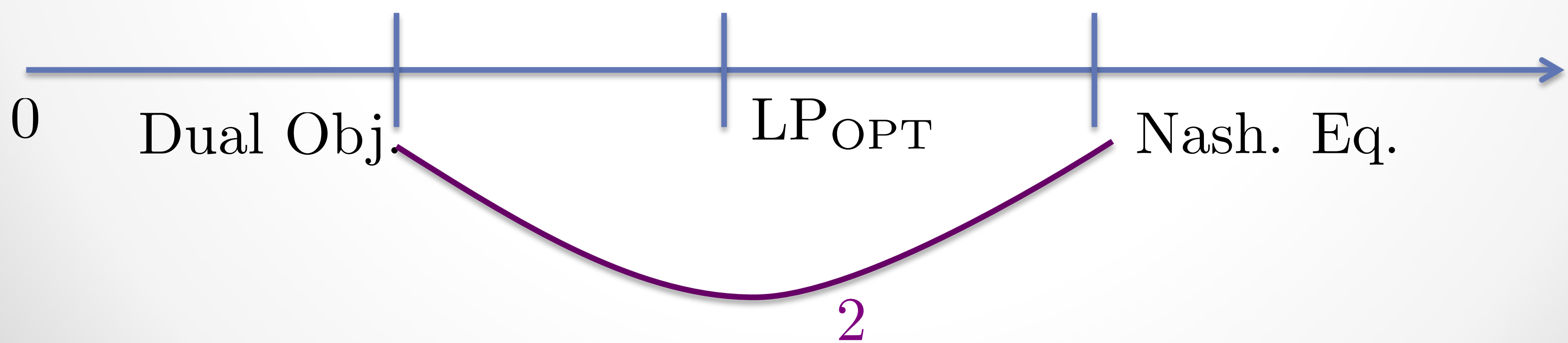
Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \le p_{ij} + p_{ij} \cdot N_{it}$ $\forall$ jobs j, machines i, times t.

$C_j, N_{it} \ge 0 \quad \forall\ i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



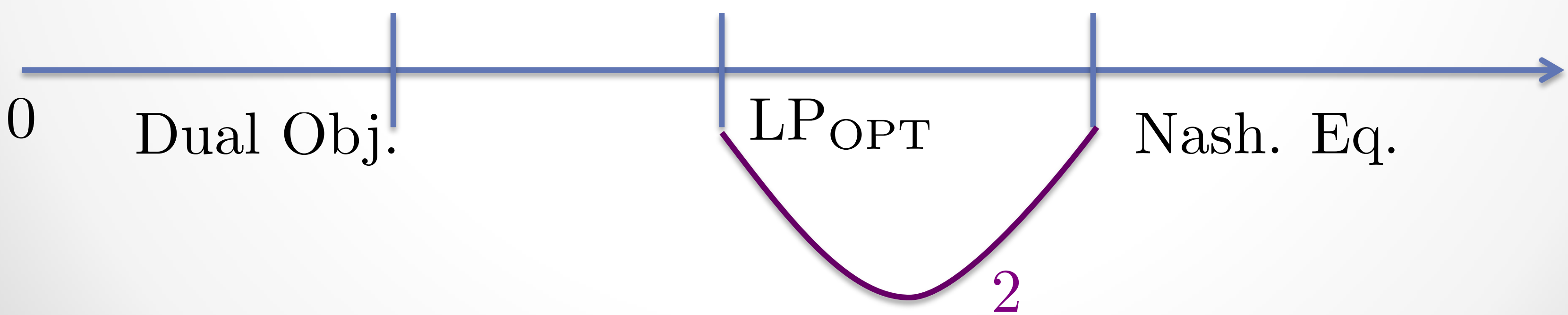
Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \;\forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \;\; \forall\; i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



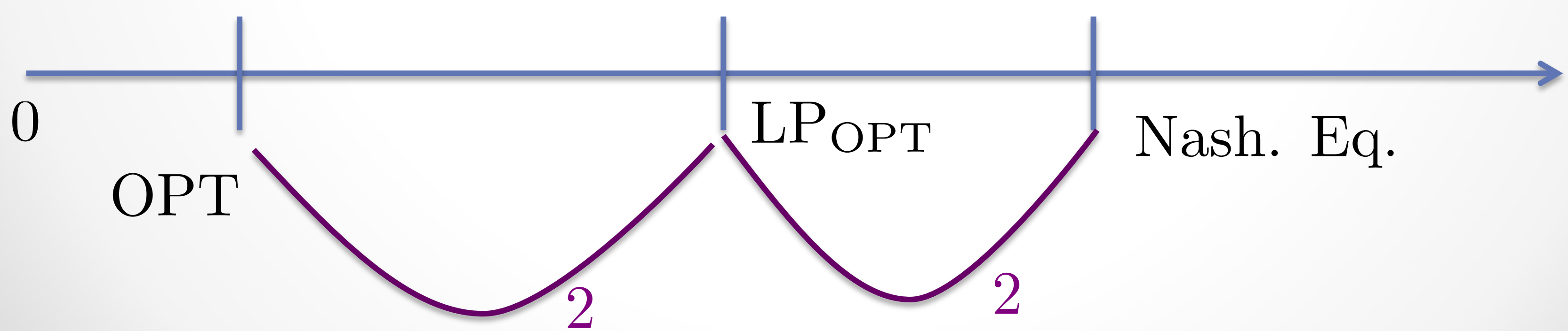
Dual Objective

Max. $\sum_j C_j - 1/2 \sum_i \sum_t N_{it}$

$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it}$ $\forall$ jobs j, machines i, times t.

$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$

$\sum_j C_j =$ Total completion time of the jobs $= \sum_{it} N_{it}$.



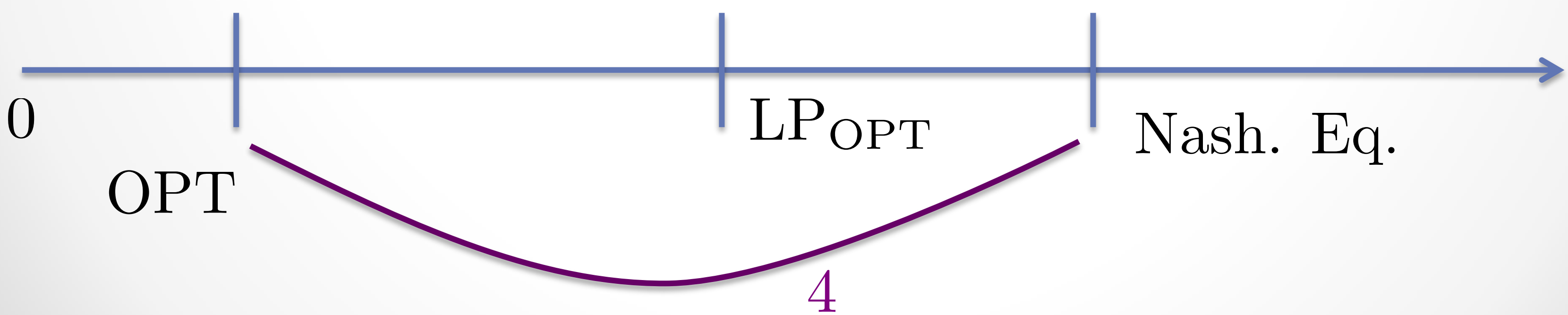
Dual Objective

$$\text{Max. } \sum_j C_j - 1/2 \sum_i \sum_t N_{it}$$

$$C_j - t \le p_{ij} + p_{ij} \cdot N_{it} \ \forall \text{ jobs } j, \text{ machines } i, \text{ times } t.$$

$$C_j, N_{it} \ge 0 \quad \forall\ i, j, t.$$

$$\sum_j C_j = \text{ Total completion time of the jobs } = \sum_{it} N_{it}.$$



Dual Objective

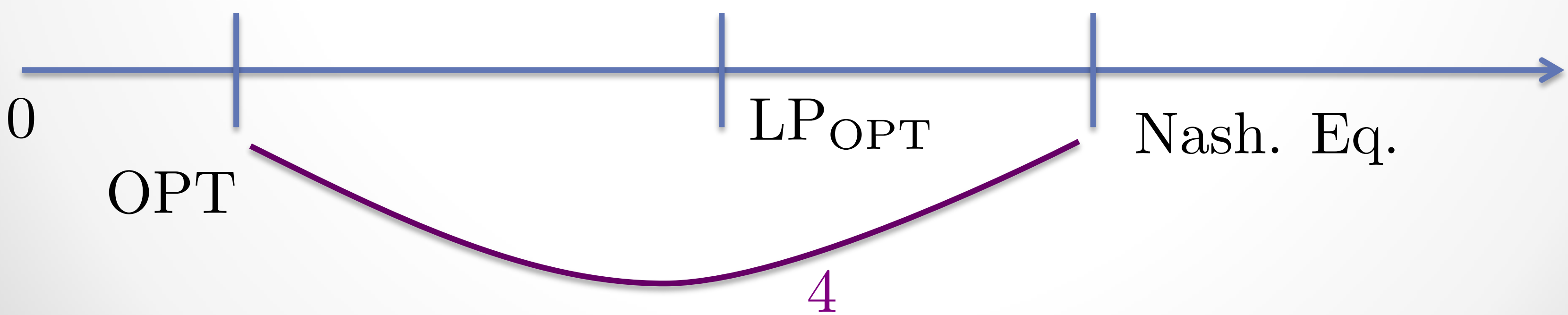
$$\text{Max.} \quad \sum_j C_j - 1/2 \sum_i \sum_t N_{it}$$

$$C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \ \forall \text{ jobs } j, \text{ machines } i, \text{ times } t.$$

$$C_j, N_{it} \geq 0 \quad \forall\ i, j, t.$$

$$\sum_j C_j = \text{ Total completion time of the jobs } = \sum_{it} N_{it}.$$

Feasibility of Dual soln. $\Longrightarrow$ Price of Anarchy ≤ 4



Dual Constraints

$$\boxed{C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \; \forall \text{ jobs } j, \text{ machines } i, \text{ times } t.}$$

Dual Constraints

$$\boxed{C_j - t \le p_{ij} + p_{ij} \cdot N_{it} \; \forall \text{ jobs } j, \text{ machines } i, \text{ times } t.}$$

Define $C_j(i) \leftarrow$ Job j's completion time if it switches to machine i.

$C_j \le C_j(i)$ Nash equilibrium condition

Dual Constraints

$$\boxed{C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \; \forall \text{ jobs } j, \text{ machines } i, \text{ times } t.}$$

Define $C_j(i) \leftarrow$ Job j's completion time if it switches to machine i.

$C_j \leq C_j(i)$ Nash equilibrium condition

$$\boxed{C_j - t \leq C_j(i) - t}$$

Dual Constraints

$$\boxed{C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \;\forall \text{ jobs } j,\ \text{machines } i,\ \text{times } t.}$$

Define $C_j(i) \leftarrow$ Job j's completion time if it switches to machine i.

$C_j \leq C_j(i)$ Nash equilibrium condition

$$\boxed{C_j - t \leq C_j(i) - t}$$

Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints

$$\boxed{C_j - t \leq p_{ij} + p_{ij} \cdot N_{it} \ \forall \text{ jobs } j, \text{ machines } i, \text{ times } t.}$$

Define $C_j(i) \leftarrow$ Job j's completion time if it switches to machine i.

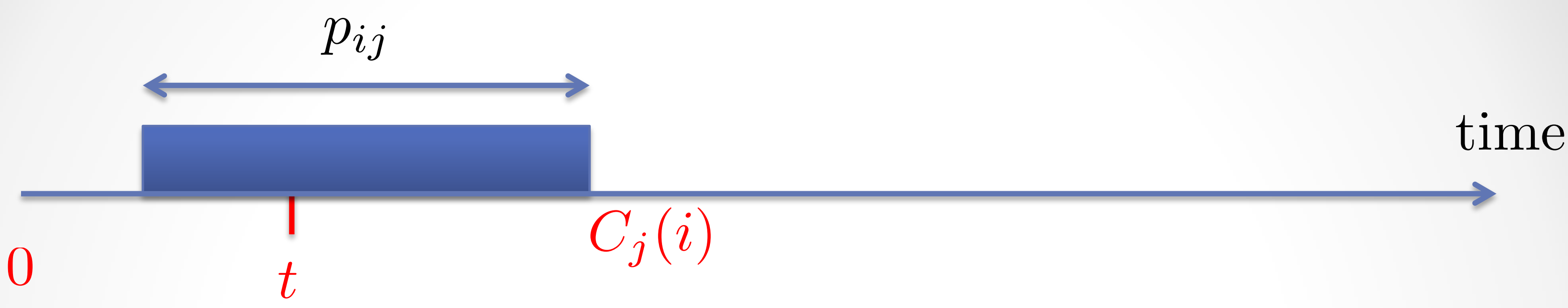
$C_j \leq C_j(i)$

Nash equilibrium condition

$$\boxed{C_j - t \leq C_j(i) - t}$$

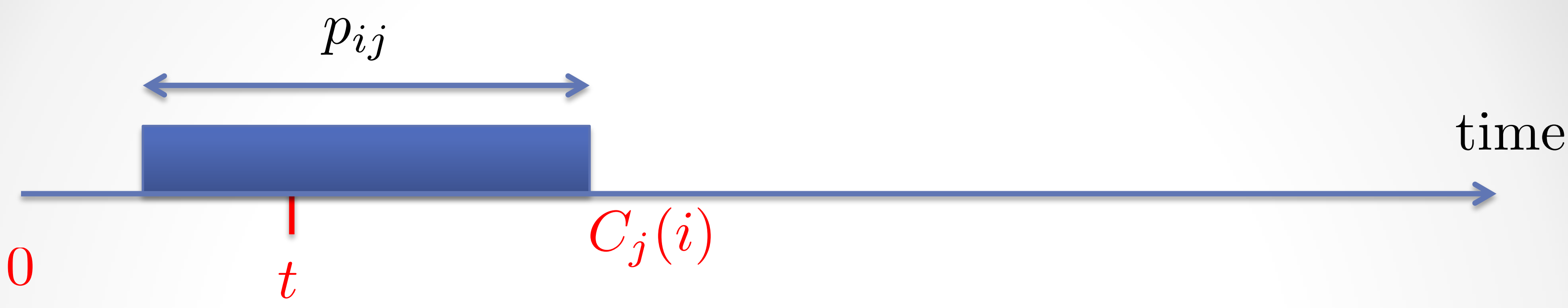
Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints



Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

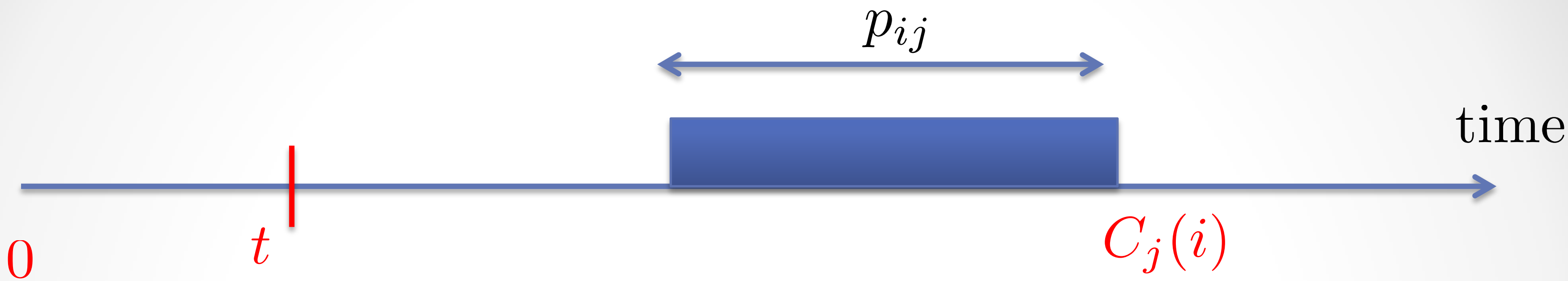
Dual Constraints



$$C_j(i) - t \le p_{ij}$$

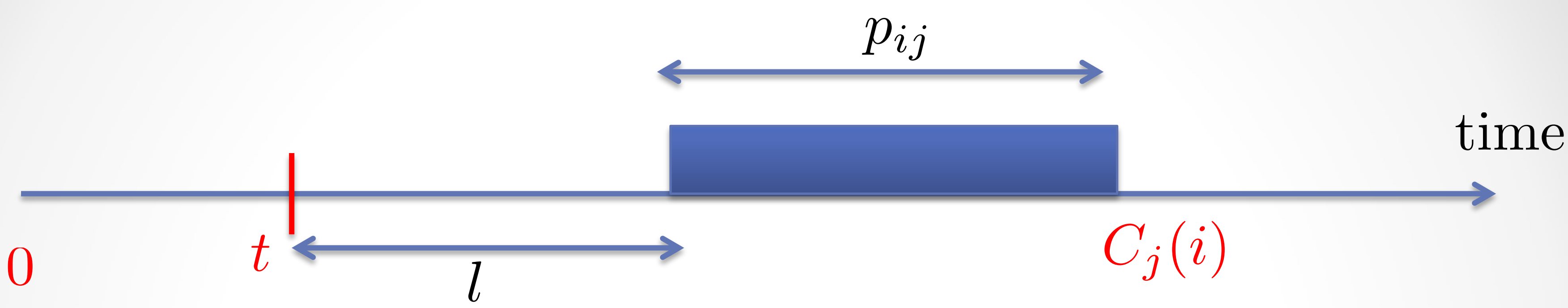
Will show: $\boxed{C_j(i) - t \le p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints



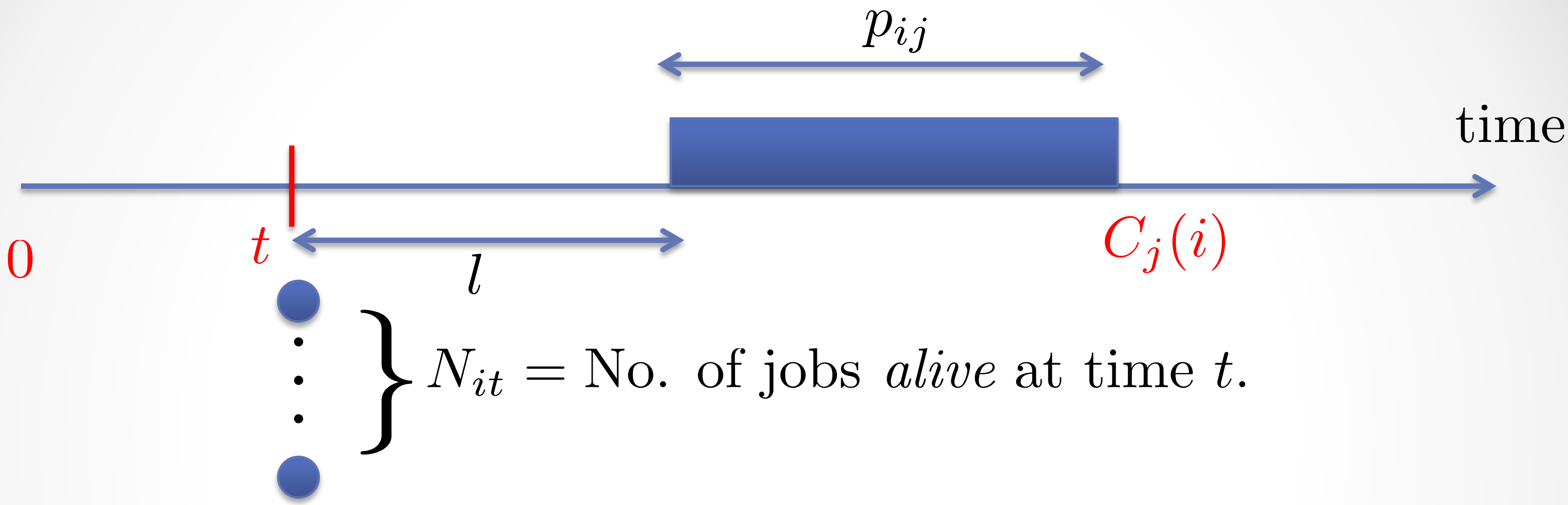
Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints



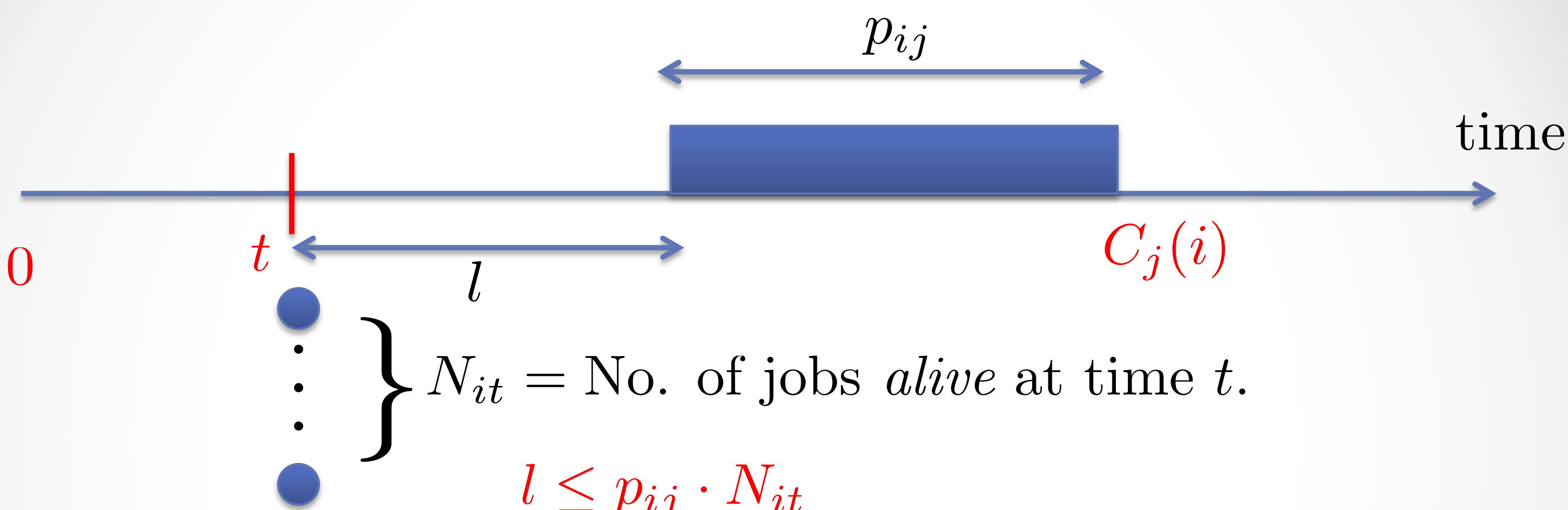
Will show: $\boxed{C_j(i) - t \le p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints



Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints

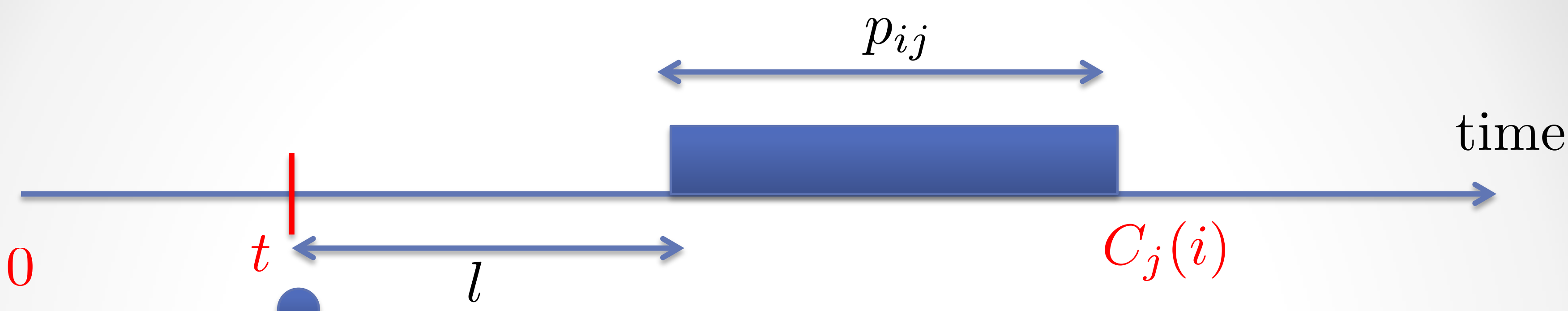


N_{it} = No. of jobs *alive* at time t.

$l \leq p_{ij} \cdot N_{it}$

Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints



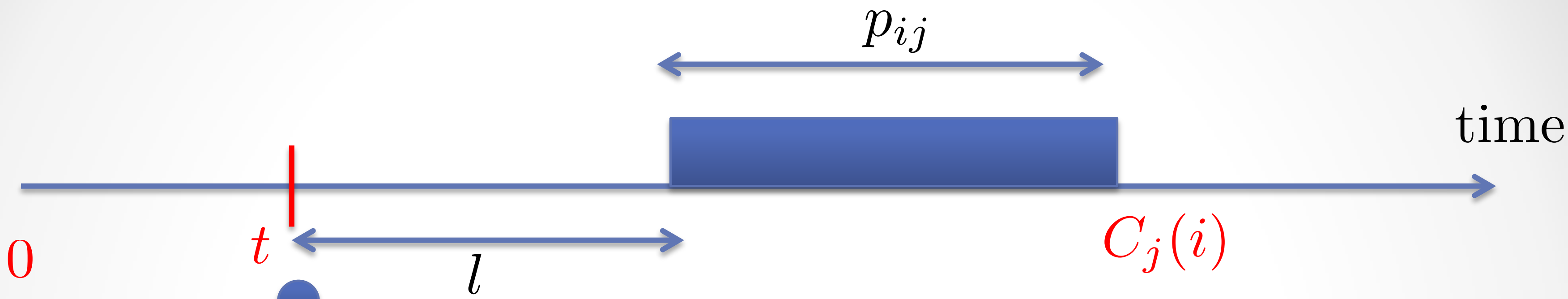
$N_{it} =$ No. of jobs *alive* at time t.

$$l \leq p_{ij} \cdot N_{it}$$

$$C_j(i) - t = p_{ij} + l$$

Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Dual Constraints



N_{it} = No. of jobs *alive* at time t.

$$l \leq p_{ij} \cdot N_{it}$$

$$C_j(i) - t = p_{ij} + l \leq p_{ij} + p_{ij} \cdot N_{it}$$

Will show: $\boxed{C_j(i) - t \leq p_{ij} + p_{ij} \cdot N_{it}}$

Conclusion

Price of Anarchy via Linear Programs

Scheduling, routing, connectivity, ...

Optimization Problem

NP-hardness

$$\text{Max.} \frac{\text{Objective at algorithm's output}}{\text{Optimal objective}}$$

Game Theoretic Variant

Strategic interactions

$$\text{Max.} \frac{\text{Objective at a Nash Eq.}}{\text{Optimal objective}}$$

Price of Anarchy via Linear Programs

Scheduling, routing, connectivity, ...

Optimization Problem

NP-hardness

$$\text{Max.}\ \frac{\text{Objective at algorithm's output}}{\text{Optimal objective}}$$

Linear programs

lower bounds optimal objective

Game Theoretic Variant

Strategic interactions

$$\text{Max.}\ \frac{\text{Objective at a Nash Eq.}}{\text{Optimal objective}}$$

Combinatorial

Problem specific

Price of Anarchy via Linear Programs

Scheduling, routing, connectivity, ...

Optimization Problem

NP-hardness

$$\text{Max.}\ \frac{\text{Objective at algorithm's output}}{\text{Optimal objective}}$$

Linear programs

lower bounds optimal objective

Game Theoretic Variant

Strategic interactions

$$\text{Max.}\ \frac{\text{Objective at a Nash Eq.}}{\text{Optimal objective}}$$

Combinatorial

Problem specific

Thank you.

III: Price of Anarchy of Smooth Games

Smooth Games

Consider a game with $\text{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \text{Obj}(\theta)$.

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \le \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \ge 1, 0 \le \mu < 1$

Theorem: A $(\lambda, \mu) -$ smooth game has PoA at most $\lambda/(1-\mu)$.

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Theorem: A (λ, μ) – smooth game has PoA at most $\lambda/(1-\mu)$.

Proof: Let $\theta^* =$ optimal outcome, $\theta =$ any pure Nash eq. Then we have:

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Theorem: A $(\lambda, \mu) -$ smooth game has PoA at most $\lambda/(1-\mu)$.

Proof: Let $\theta^* =$ optimal outcome, $\theta =$ any pure Nash eq. Then we have:

$\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Theorem: A $(\lambda, \mu) -$ smooth game has PoA at most $\lambda/(1-\mu)$.

Proof: Let $\theta^* =$ optimal outcome, $\theta =$ any pure Nash eq. Then we have:

$\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta) \leq \sum_{j \in N} c_j(\theta_j^*, \theta_{-j})$

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Theorem: A $(\lambda, \mu) -$ smooth game has PoA at most $\lambda/(1-\mu)$.

Proof: Let $\theta^* =$ optimal outcome, $\theta =$ any pure Nash eq. Then we have:

$\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta) \leq \sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$.

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \le \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\quad \lambda \ge 1, 0 \le \mu < 1$

Theorem: A (λ, μ) – smooth game has PoA at most $\lambda/(1-\mu)$.

Proof: Let $\theta^* =$ optimal outcome, $\theta =$ any pure Nash eq. Then we have:

$\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta) \le \sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \le \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$.

Rearranging the terms, we get: $(1-\mu) \cdot \mathrm{Obj}(\theta) \le \lambda \cdot \mathrm{Obj}(\theta^*)$

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Theorem: A (λ, μ) – smooth game has PoA at most $\lambda/(1-\mu)$.

Proof: Let $\theta^* =$ optimal outcome, $\theta =$ any pure Nash eq. Then we have:

$$\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta) \leq \sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

Rearranging the terms, we get: $(1-\mu) \cdot \mathrm{Obj}(\theta) \leq \lambda \cdot \mathrm{Obj}(\theta^*)$

$$\Rightarrow \quad \frac{\mathrm{Obj}(\theta)}{\mathrm{Obj}(\theta^*)} \leq \frac{\lambda}{1-\mu}.$$

Smooth Games

Consider a game with $\mathrm{Obj}(\theta) = \sum_{j \in N} c_j(\theta)$ at each outcome θ.

Underlying optimization problem is: $\min_\theta \mathrm{Obj}(\theta)$.

Such a game is (λ, μ)-smooth iff for every two outcomes θ, θ^*, we have

$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta)$. $\lambda \geq 1, 0 \leq \mu < 1$

Theorem: A $(\lambda, \mu) -$ smooth game has PoA at most $\lambda/(1-\mu)$.

Proof:

The proof can be extended to all other solution concepts!

Example: Selfish Routing

Directed graph $G = (V, E)$.

Player j selects a path from $u_j \in V$ to $v_j \in V$.

θ_j denotes the strategy of player j(i.e., it is a $u_j \to v_j$ path).

Example: Selfish Routing

Directed graph $G = (V, E)$.

Player j selects a path from $u_j \in V$ to $v_j \in V$.

θ_j denotes the strategy of player j(i.e., it is a $u_j \to v_j$ path).

Under a given outcome $\theta = (\theta_1, \ldots, \theta_n)$, the *load* on an edge $e \in E$ is $l_e(\theta) = \{j \in N : e \in \theta_j\}$: the number of players using the edge.

$c_j(\theta) = \sum_{e \in \theta_j} l_e(\theta)$: cost function of player j.

Example: Selfish Routing

Directed graph $G = (V, E)$.

Player j selects a path from $u_j \in V$ to $v_j \in V$.

θ_j denotes the strategy of player j(i.e., it is a $u_j \to v_j$ path).

Under a given outcome $\theta = (\theta_1, \ldots, \theta_n)$, the *load* on an edge $e \in E$ is $l_e(\theta) = \{j \in N : e \in \theta_j\}$: the number of players using the edge.

$c_j(\theta) = \sum_{e \in \theta_j} l_e(\theta)$: cost function of player j.

$\mathrm{Obj}(\theta) = \sum_j c_j(\theta)$.

Example: Selfish Routing

Directed graph $G = (V, E)$.

Player j selects a path from $u_j \in V$ to $v_j \in V$.

θ_j denotes the strategy of player j(i.e., it is a $u_j \to v_j$ path).

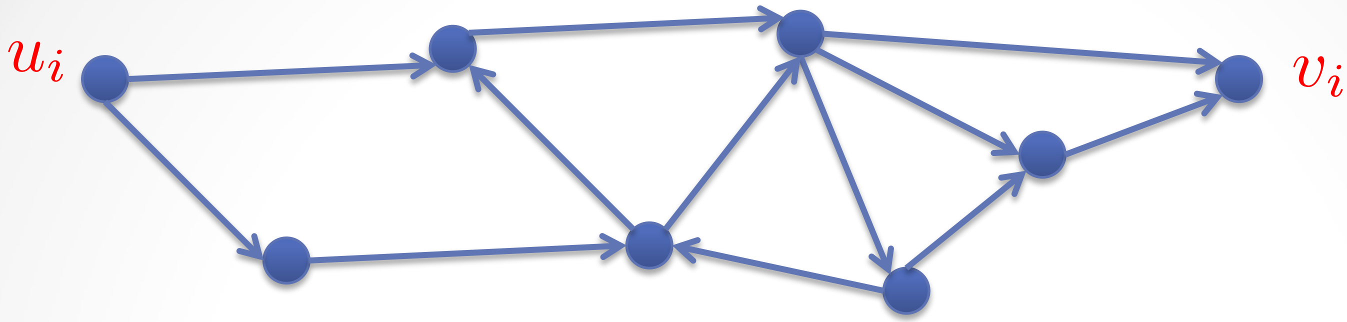
Under a given outcome $\theta = (\theta_1, \ldots, \theta_n)$, the *load* on an edge $e \in E$ is $l_e(\theta) = \{j \in N : e \in \theta_j\}$: the number of players using the edge.

$c_j(\theta) = \sum_{e \in \theta_j} l_e(\theta)$: cost function of player j.

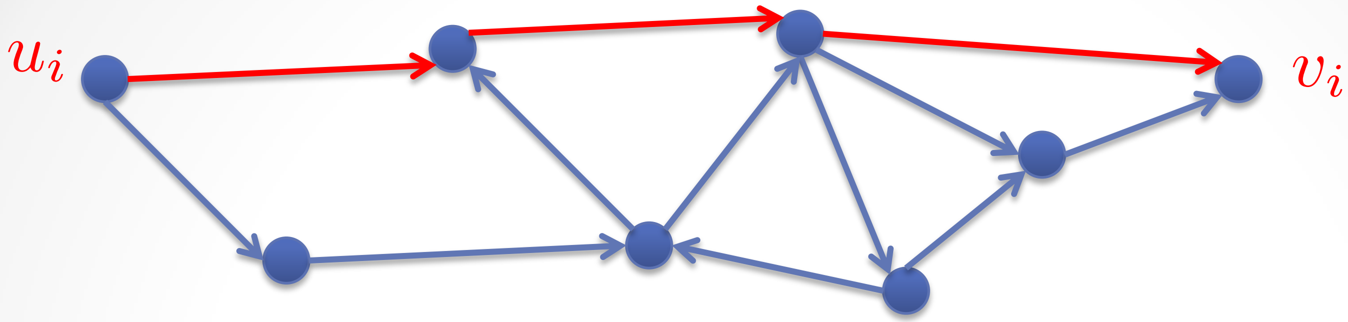
$\mathrm{Obj}(\theta) = \sum_j c_j(\theta)$.

$$\text{Price of Anarchy (PoA)} = \frac{\max_{\theta \text{is in equilibrium}} \mathrm{Obj}(\theta)}{\min_\theta \mathrm{Obj}(\theta)}$$

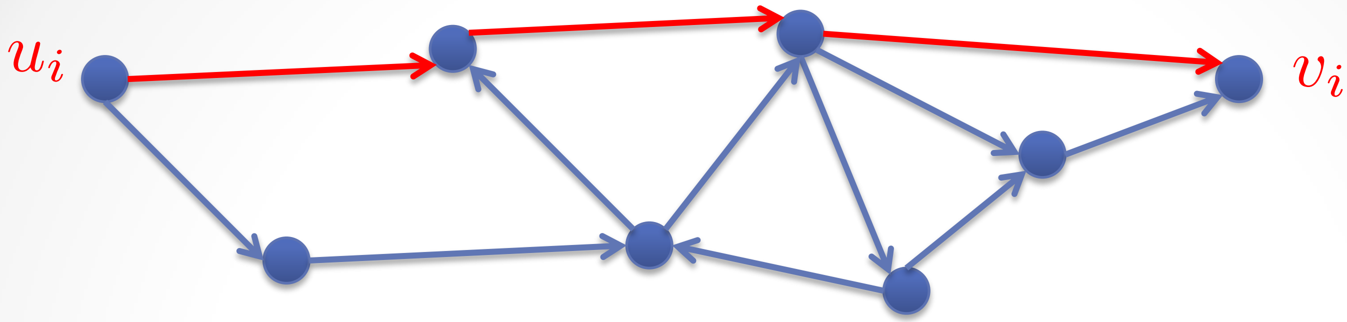
Robust PoA of Selfish Routing



Robust PoA of Selfish Routing



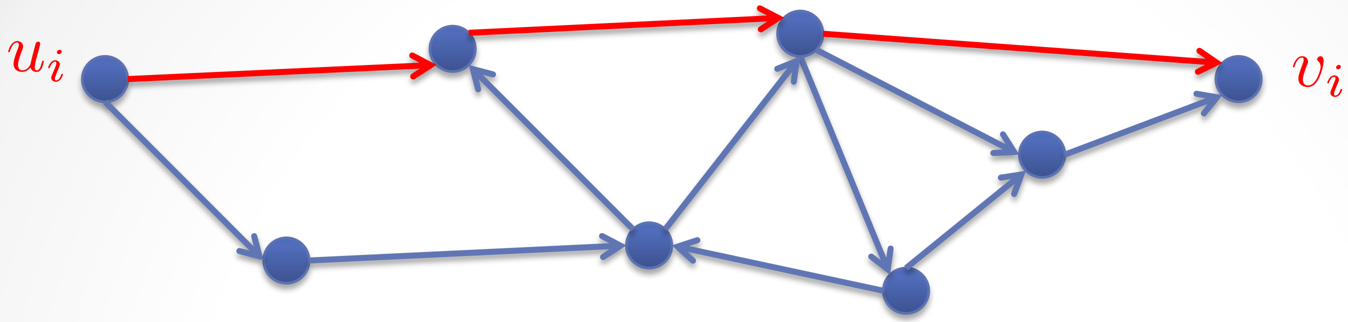
Robust PoA of Selfish Routing



$l_e(\theta) =$ no. of players using the edge e under θ.

$c_i(\theta) =$ sum of the loads in the path chosen by i.

Robust PoA of Selfish Routing

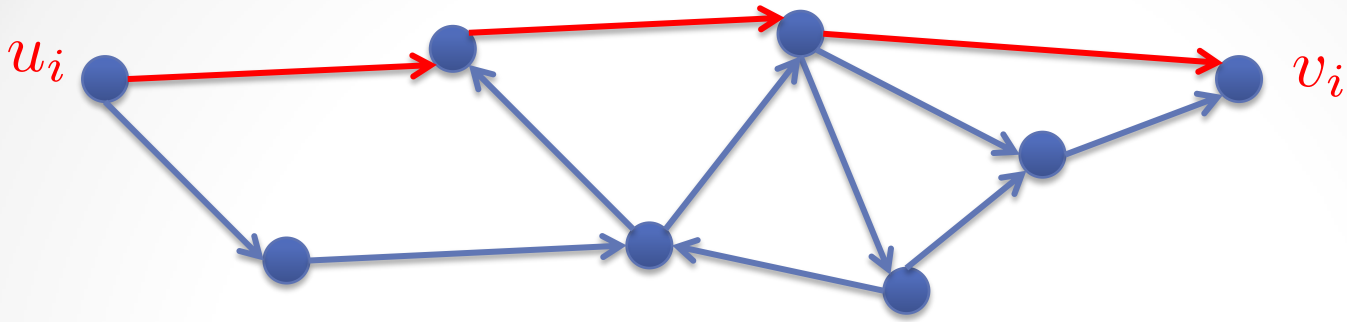


$l_e(\theta) = \text{no. of players using the edge } e \text{ under } \theta.$

$c_i(\theta) = \text{sum of the loads in the path chosen by } i.$

$$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

Robust PoA of Selfish Routing



$l_e(\theta) =$ no. of players using the edge e under θ.

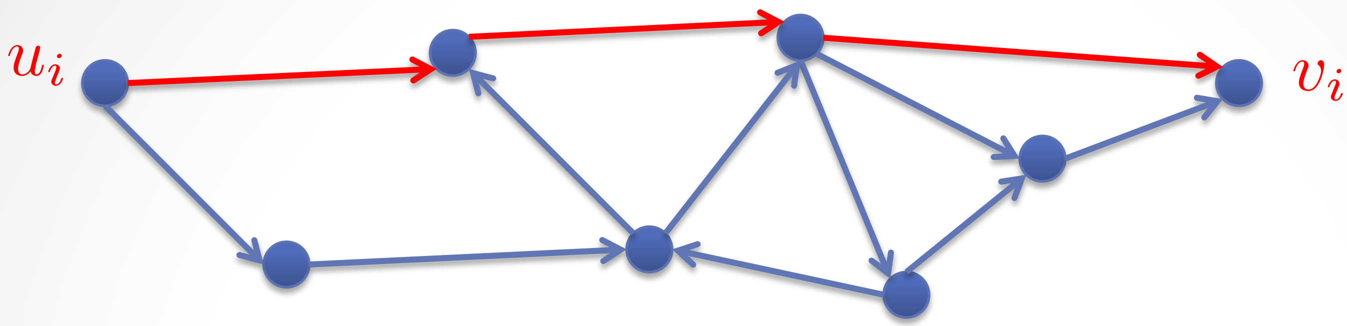
$c_i(\theta) =$ sum of the loads in the path chosen by i.

$$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

$$\Updownarrow$$

$$\sum_{e \in E} l_e(\theta^*) \cdot (l_e(\theta) + 1) \leq \lambda \cdot \sum_{e \in E} l_e(\theta^*)^2 + \mu \cdot \sum_{e \in E} l_e(\theta)^2$$

Robust PoA of Selfish Routing



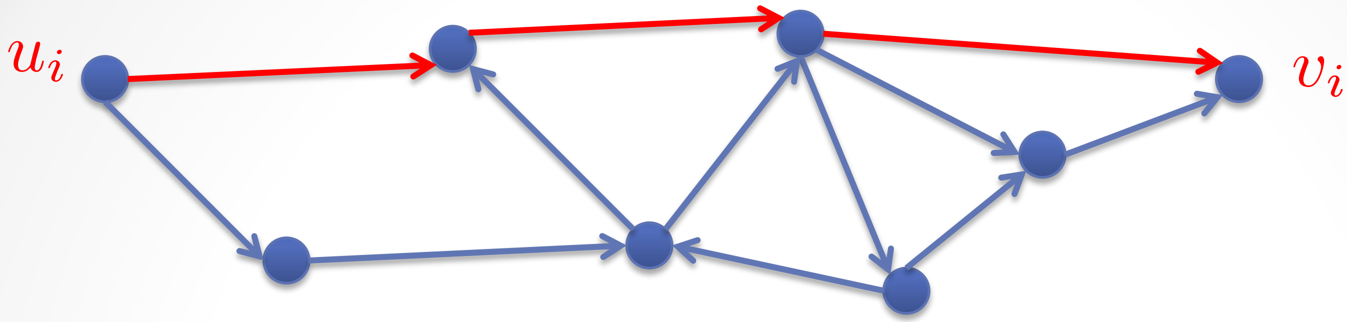
$l_e(\theta)$ = no. of players using the edge e under θ.

$c_i(\theta)$ = sum of the loads in the path chosen by i.

$$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

$$\boxed{\sum_{e \in E} l_e(\theta^*) \cdot (l_e(\theta) + 1)} \leq \lambda \cdot \boxed{\sum_{e \in E} l_e(\theta^*)^2} + \mu \cdot \boxed{\sum_{e \in E} l_e(\theta)^2}$$

Robust PoA of Selfish Routing



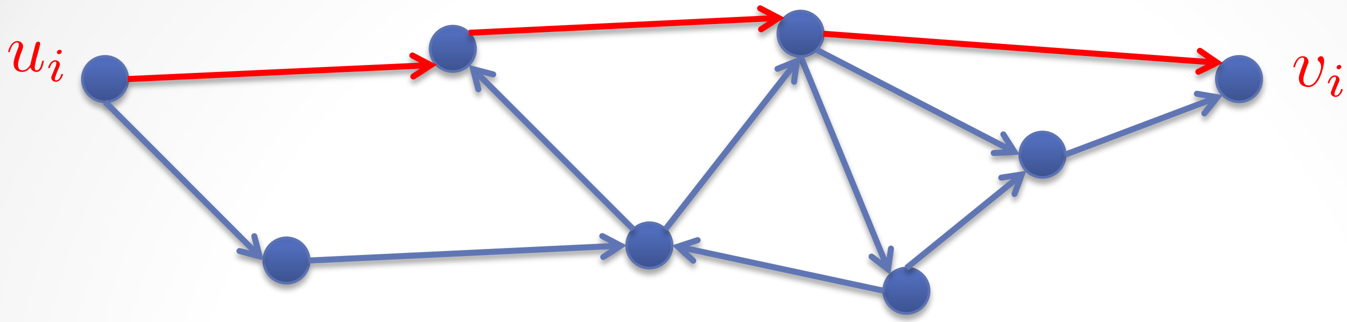
$$\boxed{\begin{array}{l} l_e(\theta) = \text{no. of players using the edge } e \text{ under } \theta. \\ c_i(\theta) = \text{sum of the loads in the path chosen by } i. \end{array}}$$

$$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

$$\boxed{\sum_{e \in E} l_e(\theta^*) \cdot (l_e(\theta) + 1)} \leq \lambda \cdot \boxed{\sum_{e \in E} l_e(\theta^*)^2} + \mu \cdot \boxed{\sum_{e \in E} l_e(\theta)^2}$$

$$x(y+1) \leq \lambda \cdot x^2 + \mu \cdot y^2$$

Robust PoA of Selfish Routing



$l_e(\theta) =$ no. of players using the edge e under θ.

$c_i(\theta) =$ sum of the loads in the path chosen by i.

$$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

$$\boxed{\sum_{e \in E} l_e(\theta^*) \cdot (l_e(\theta) + 1)} \leq \lambda \cdot \boxed{\sum_{e \in E} l_e(\theta^*)^2} + \mu \cdot \boxed{\sum_{e \in E} l_e(\theta)^2}$$

$$x(y+1) \leq \lambda \cdot x^2 + \mu \cdot y^2 \Longrightarrow \lambda = 5/3, \mu = 1/3.$$

Robust PoA of Selfish Routing

Hence, PoA of selfish routing $\leq \lambda/(1-\mu) = (5/3)/(1-1/3) = 5/2$.

$$\sum_{j \in N} c_j(\theta_j^*, \theta_{-j}) \leq \lambda \cdot \mathrm{Obj}(\theta^*) + \mu \cdot \mathrm{Obj}(\theta).$$

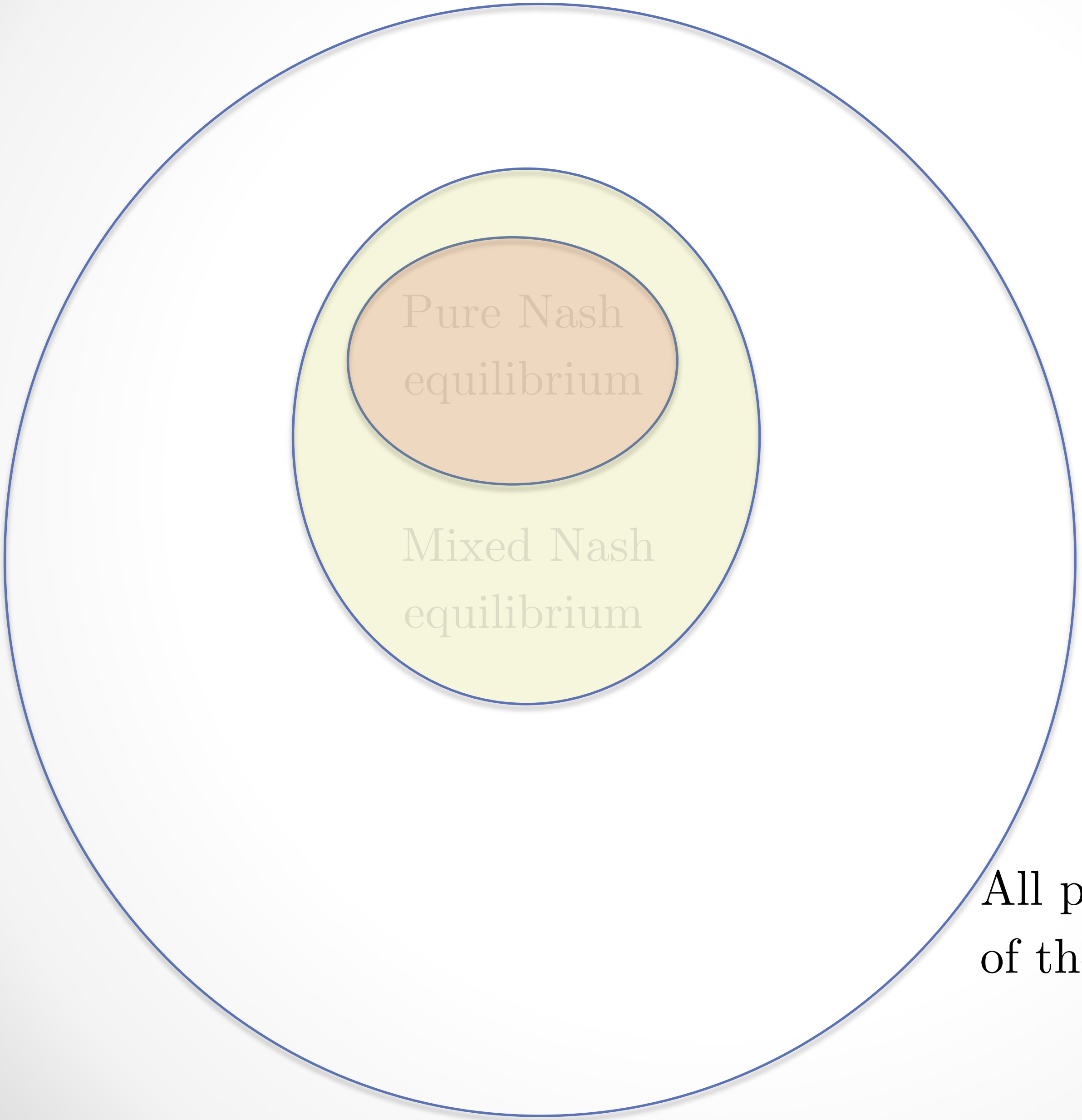
$$\boxed{\sum_{e \in E} l_e(\theta^*) \cdot (l_e(\theta)+1)} \leq \lambda \cdot \boxed{\sum_{e \in E} l_e(\theta^*)^2} + \mu \cdot \boxed{\sum_{e \in E} l_e(\theta)^2}$$

$$x(y+1) \leq \lambda \cdot x^2 + \mu \cdot y^2 \Longrightarrow \lambda = 5/3, \mu = 1/3.$$

IV: Review of Basic Solution Concepts

Different ``Solution Concepts''

Different ``Solution Concepts''



All possible outcomes of the game

Mixed Nash Equilibrium

Mixed Nash Equilibrium

Each player $j \in \mathcal{N}$ has a set of strategies $\mathcal{S}_j$.

Mixed Nash Equilibrium

Each player $j \in \mathcal{N}$ has a set of strategies $\mathcal{S}_j$.

An element of $\mathcal{S}_j$ is a "pure strategy" for player j.

Mixed Nash Equilibrium

Each player $j \in \mathcal{N}$ has a set of strategies $\mathcal{S}_j$.

An element of $\mathcal{S}_j$ is a "pure strategy" for player j.

A "mixed strategy" π_j for player j is a probability distribution over $\mathcal{S}_j$.

Mixed Nash Equilibrium

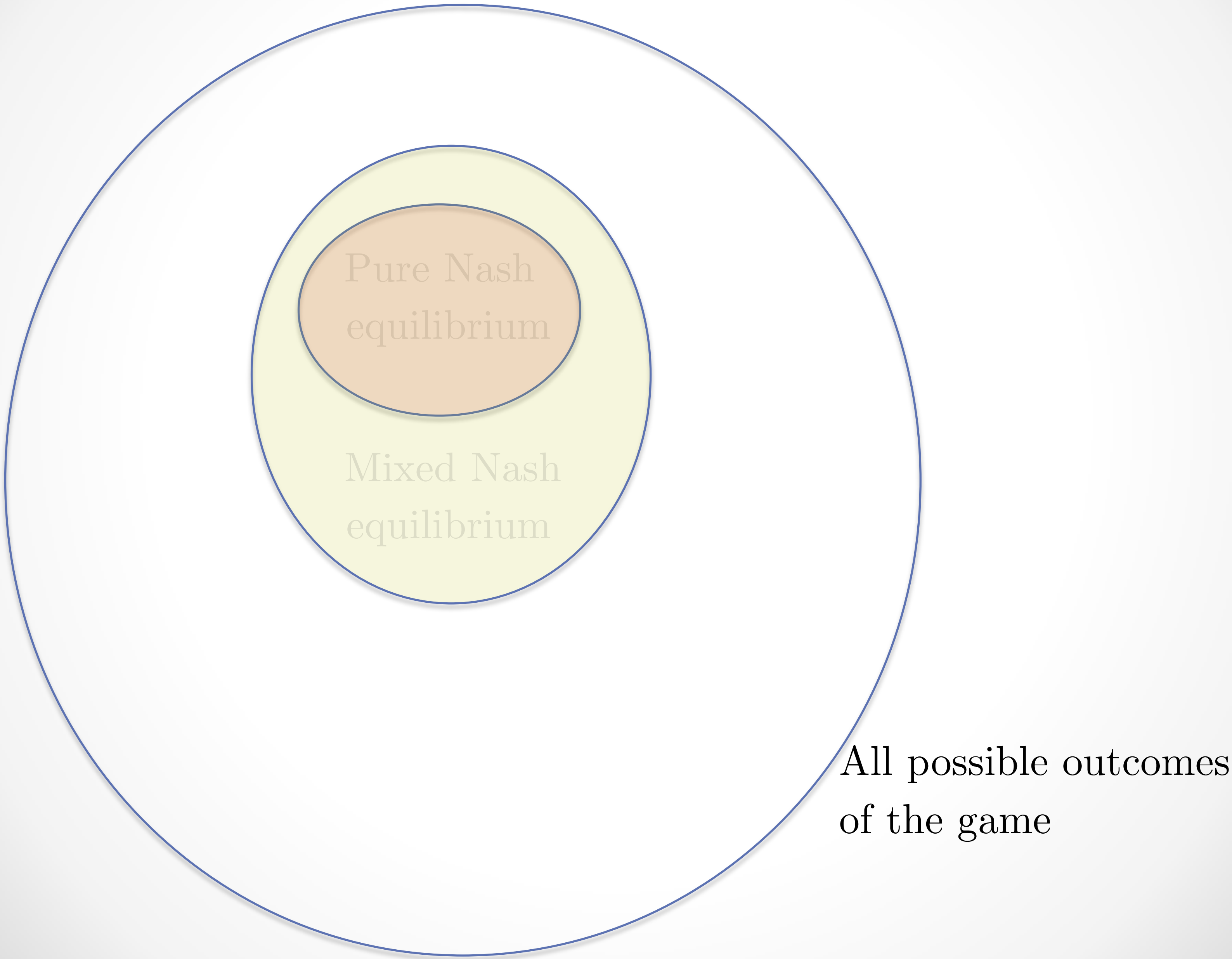
Each player $j \in \mathcal{N}$ has a set of strategies $\mathcal{S}_j$.

An element of $\mathcal{S}_j$ is a "pure strategy" for player j.

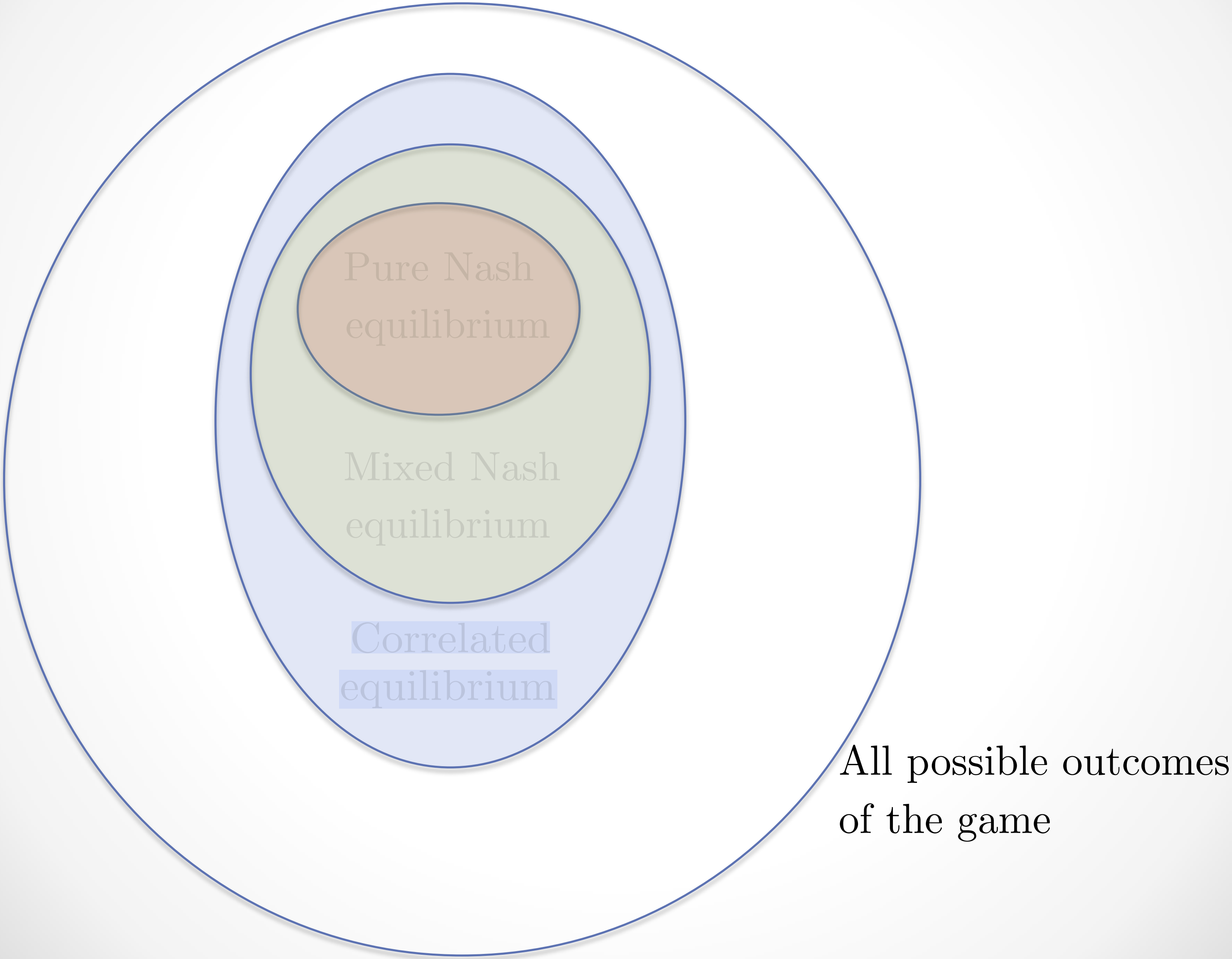
A "mixed strategy" π_j for player j is a probability distribution over $\mathcal{S}_j$.

A profile $\pi = (\pi_1, \ldots, \pi_n)$ is a Mixed Nash equilibrium iff no player can decrease her expected cost by unilaterally switching her strategy.

Different ``Solution Concepts''



Different ``Solution Concepts''



Correlated Equilibrium

Traffic Light

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross		
Stop		

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility for crossing safely: $+1$.

Utility Matrix

Bob / Alice	Cross	Stop
Cross		
Stop		

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility for crossing safely: +1.

Utility for stopping: 0.

Utility Matrix

Bob / Alice	Cross	Stop
Cross		
Stop		

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility for crossing safely: +1.

Utility for stopping: 0.

Utility Matrix

Bob Alice	Cross	Stop
Cross		0 +1
Stop	+1 0	

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility for crossing safely: $+1$.

Utility for stopping: 0.

Utility Matrix

Bob / Alice	Cross	Stop
Cross		0 $+1$
Stop	$+1$ 0	0 0

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility for crossing safely: +1.

Utility for stopping: 0.

Utility for being involved in a crash: -100.

Utility Matrix

Bob Alice	Cross	Stop
Cross		0 +1
Stop	+1 0	0 0

Traffic Light

Two player, driving different cars, arrive at an intersection at the same time.

Utility for crossing safely: $+1$.

Utility for stopping: 0.

Utility for being involved in a crash: -100.

Utility Matrix

Bob Alice	Cross	Stop
Cross	-100 -100	0 $+1$
Stop	$+1$ 0	0 0

Traffic Light

Utility Matrix

Bob Alice	Cross	Stop
Cross	-100 -100	0 $+1$
Stop	$+1$ 0	0 0

Traffic Light

Two pure Nash eq. in this game (one player stops, the other crosses).

Utility Matrix

Bob / Alice	Cross	Stop
Cross	-100 -100	0 $+1$
Stop	$+1$ 0	0 0

Traffic Light

Two pure Nash eq. in this game (one player stops, the other crosses).

– None of them is "fair".

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100 (Alice), -100 (Bob)	$+1$ (Alice), 0 (Bob)
Stop	0 (Alice), $+1$ (Bob)	0 (Alice), 0 (Bob)

Traffic Light

Two pure Nash eq. in this game (one player stops, the other crosses).

– None of them is "fair".

One mixed Nash equilibrium.

$\Pr[\text{Alice Crosses}] = 1/101$

$\Pr[\text{Alice Stops}] = 100/101$

$\Pr[\text{Bob Crosses}] = 1/101$

$\Pr[\text{Bob Stops}] = 100/101$

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100 (Alice), -100 (Bob)	$+1$ (Alice), 0 (Bob)
Stop	0 (Alice), $+1$ (Bob)	0 (Alice), 0 (Bob)

Traffic Light

Two pure Nash eq. in this game (one player stops, the other crosses).

– None of them is "fair".

One mixed Nash equilibrium.

– Positive chance of a crash.

$$\Pr[\text{Alice Crosses}] = 1/101$$

$$\Pr[\text{Alice Stops}] = 100/101$$

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100, -100	0, $+1$
Stop	$+1$, 0	0, 0

$$\Pr[\text{Bob Crosses}] = 1/101$$

$$\Pr[\text{Bob Stops}] = 100/101$$

Traffic Light

Utility Matrix

Bob / Alice	Cross	Stop
Cross	-100 -100	0 $+1$
Stop	$+1$ 0	0 0

Traffic Light

Assume that there is a "mediator" (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	Alice: -100, Bob: -100	Alice: $+1$, Bob: 0
Stop	Alice: 0, Bob: $+1$	Alice: 0, Bob: 0

Traffic Light

Assume that there is a "mediator" (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

$\sigma(\theta) = \Pr[\text{mediator picks the outcome } \theta]$.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100 (Alice), -100 (Bob)	$+1$ (Alice), 0 (Bob)
Stop	0 (Alice), $+1$ (Bob)	0 (Alice), 0 (Bob)

Traffic Light

Assume that there is a “mediator” (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

$\sigma(\theta) = \Pr[\text{mediator picks the outcome } \theta]$.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100, -100	$+1$, 0
Stop	0, $+1$	0, 0

outcome θ	$\sigma(\theta)$
Cross , Cross	
Cross , Stop	
Stop , Cross	
Stop , Stop	

Traffic Light

Assume that there is a "mediator" (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

$\sigma(\theta) = \Pr[\text{mediator picks the outcome } \theta]$.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100, -100	$+1$, 0
Stop	0, $+1$	0, 0

outcome θ	$\sigma(\theta)$
Cross , Cross	0
Cross , Stop	
Stop , Cross	
Stop , Stop	0

Traffic Light

Assume that there is a "mediator" (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

$\sigma(\theta) = \Pr[\text{mediator picks the outcome } \theta]$.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100, -100	$+1$, 0
Stop	0, $+1$	0, 0

outcome θ	$\sigma(\theta)$
Cross , Cross	0
Cross , Stop	p
Stop , Cross	$1-p$
Stop , Stop	0

Traffic Light

Assume that there is a "mediator" (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

$\sigma(\theta) = \Pr[\text{mediator picks the outcome } \theta]$.

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100 (Alice), -100 (Bob)	$+1$ (Alice), 0 (Bob)
Stop	0 (Alice), $+1$ (Bob)	0 (Alice), 0 (Bob)

Correlated equilibrium

outcome θ	$\sigma(\theta)$
Cross , Cross	0
Cross , Stop	p
Stop , Cross	$1-p$
Stop , Stop	0

Traffic Light

Assume that there is a "mediator" (traffic light) that picks a probability distribution σ over the set of all possible outcomes.

$\sigma(\theta) = \Pr[\text{mediator picks the outcome } \theta]$.

For $p = 1/2$, the solution is "fair".

Utility Matrix

Alice \ Bob	Cross	Stop
Cross	-100 (Bob), -100 (Alice)	0 (Bob), $+1$ (Alice)
Stop	$+1$ (Bob), 0 (Alice)	0 (Bob), 0 (Alice)

Correlated equilibrium

outcome θ	$\sigma(\theta)$
Cross , Cross	0
Cross , Stop	p
Stop , Cross	$1-p$
Stop , Stop	0

Correlated Equilibrium: Definition

Correlated Equilibrium: Definition

Consider a distribution σ over the set of possible outcomes $\mathcal{S}$.

Correlated Equilibrium: Definition

Consider a distribution σ over the set of possible outcomes $\mathcal{S}$.

A "mediator" draws an outcome $\theta = (\theta_1, \ldots, \theta_n)$ from $\mathcal{S}$, and suggests to each player j that she should play strategy θ_j.

Correlated Equilibrium: Definition

Consider a distribution σ over the set of possible outcomes $\mathcal{S}$.

A "mediator" draws an outcome $\theta = (\theta_1, \ldots, \theta_n)$ from $\mathcal{S}$, and suggests to each player j that she should play strategy θ_j.

Player j only knows σ and θ_j. In a correlated equilibrium, she will follow the mediator's suggestion, *provided that others do the same.*

Correlated Equilibrium: Definition

Consider a distribution σ over the set of possible outcomes $\mathcal{S}$.

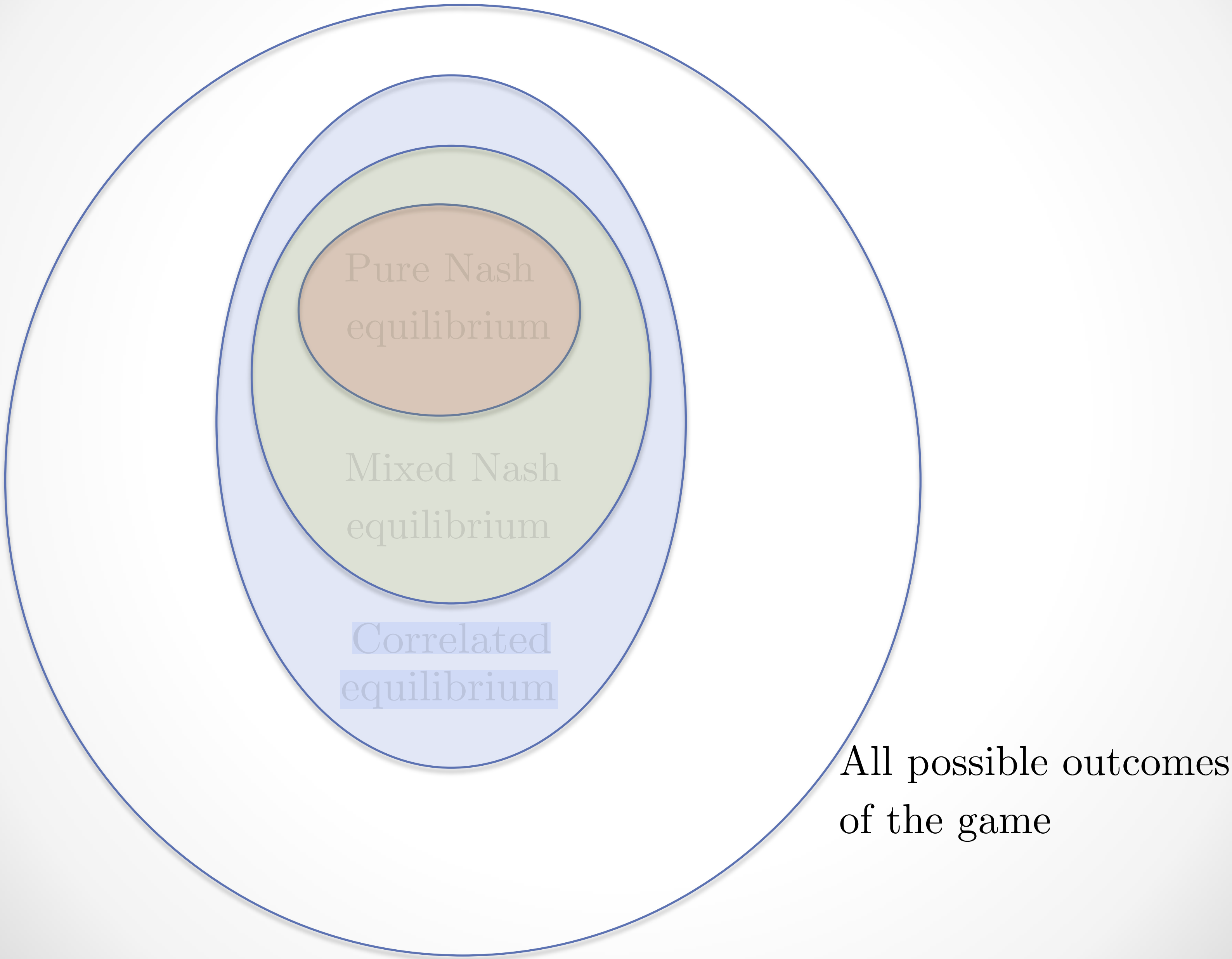
A "mediator" draws an outcome $\theta = (\theta_1, \ldots, \theta_n)$ from $\mathcal{S}$, and suggests to each player j that she should play strategy θ_j.

Player j only knows σ and θ_j. In a correlated equilibrium, she will follow the mediator's suggestion, *provided that others do the same.*

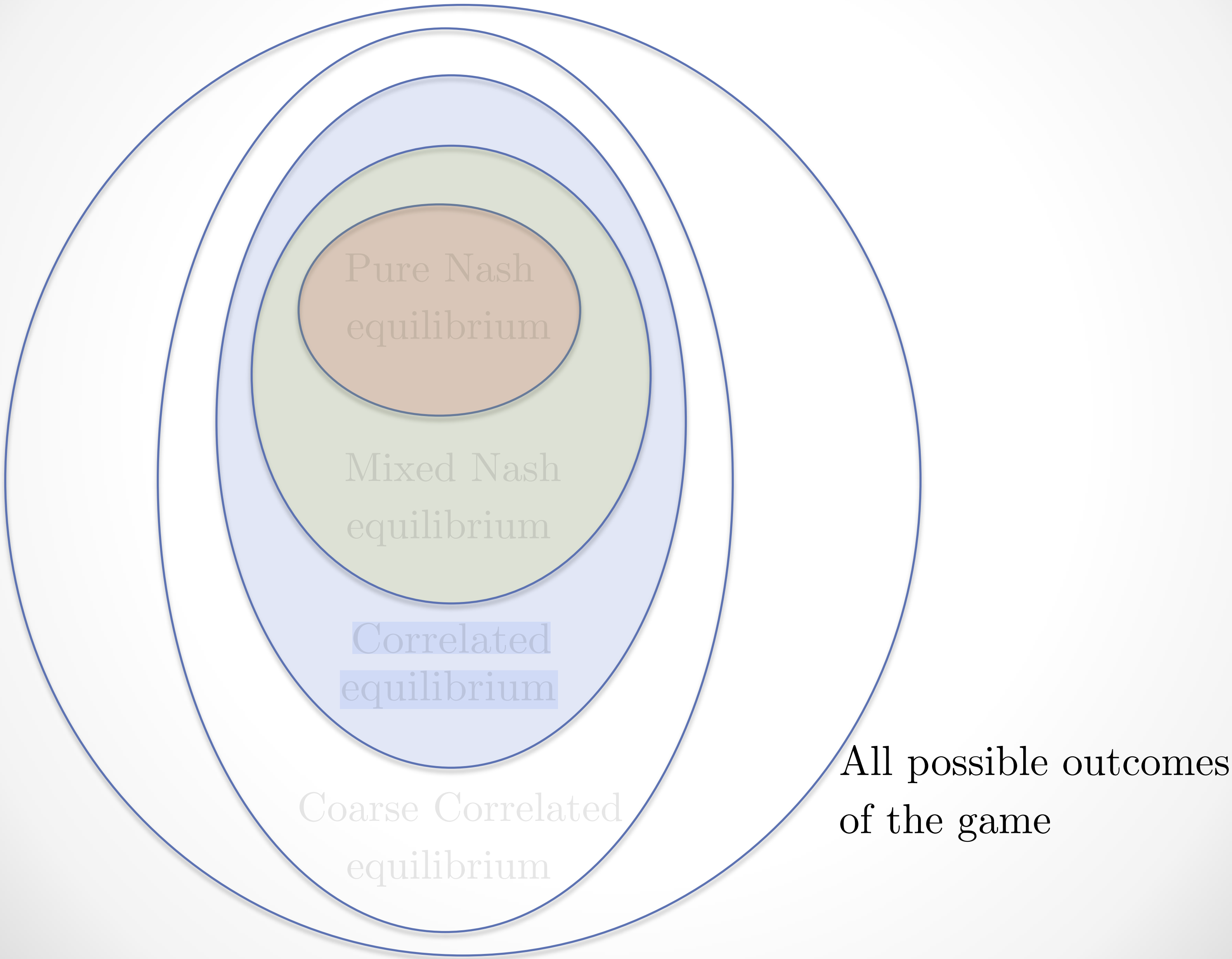
The distribution σ is a correlated equilibrium iff for every player $j \in \mathcal{N}$,

$E_{\theta\sim\sigma}\left[c_j(\theta) \,|\, \theta_j\right] \leq E_{\theta\sim\sigma}\left[c_j(x, \theta_{-j}) \,|\, \theta_j\right]$ for all strategies $x, \theta_j \in \mathcal{S}_j$.

Different ``Solution Concepts''



Different ``Solution Concepts''



Coarse Correlated Equilibrium

Coarse Correlated Equilibrium

Consider a distribution σ over the set of all outcomes $\mathcal{S}$.

Coarse Correlated Equilibrium

Consider a distribution σ over the set of all outcomes $\mathcal{S}$.

The distribution σ is a coarse correlated eq. iff for every player $j \in \mathcal{N}$,

$E_{\theta \sim \sigma}\left[c_j(\theta)\right] \leq E_{\theta \sim \sigma}\left[c_j(x, \theta_{-j})\right]$ for all strategies $x \in \mathcal{S}_j$.

Coarse Correlated Equilibrium

Consider a distribution σ over the set of all outcomes $\mathcal{S}$.

The distribution σ is a coarse correlated eq. iff for every player $j \in \mathcal{N}$,

$E_{\theta \sim \sigma}\left[c_j(\theta)\right] \leq E_{\theta \sim \sigma}\left[c_j(x, \theta_{-j})\right]$ for all strategies $x \in \mathcal{S}_j$.

$E_{\theta \sim \sigma}[c_j(\theta) \mid \theta_j] \leq E_{\theta \sim \sigma}[c_j(x, \theta_{-j} \mid \theta_j)]$ $\Leftarrow$ Correlated equilibrium

Coarse Correlated Equilibrium

Consider a distribution σ over the set of all outcomes $\mathcal{S}$.

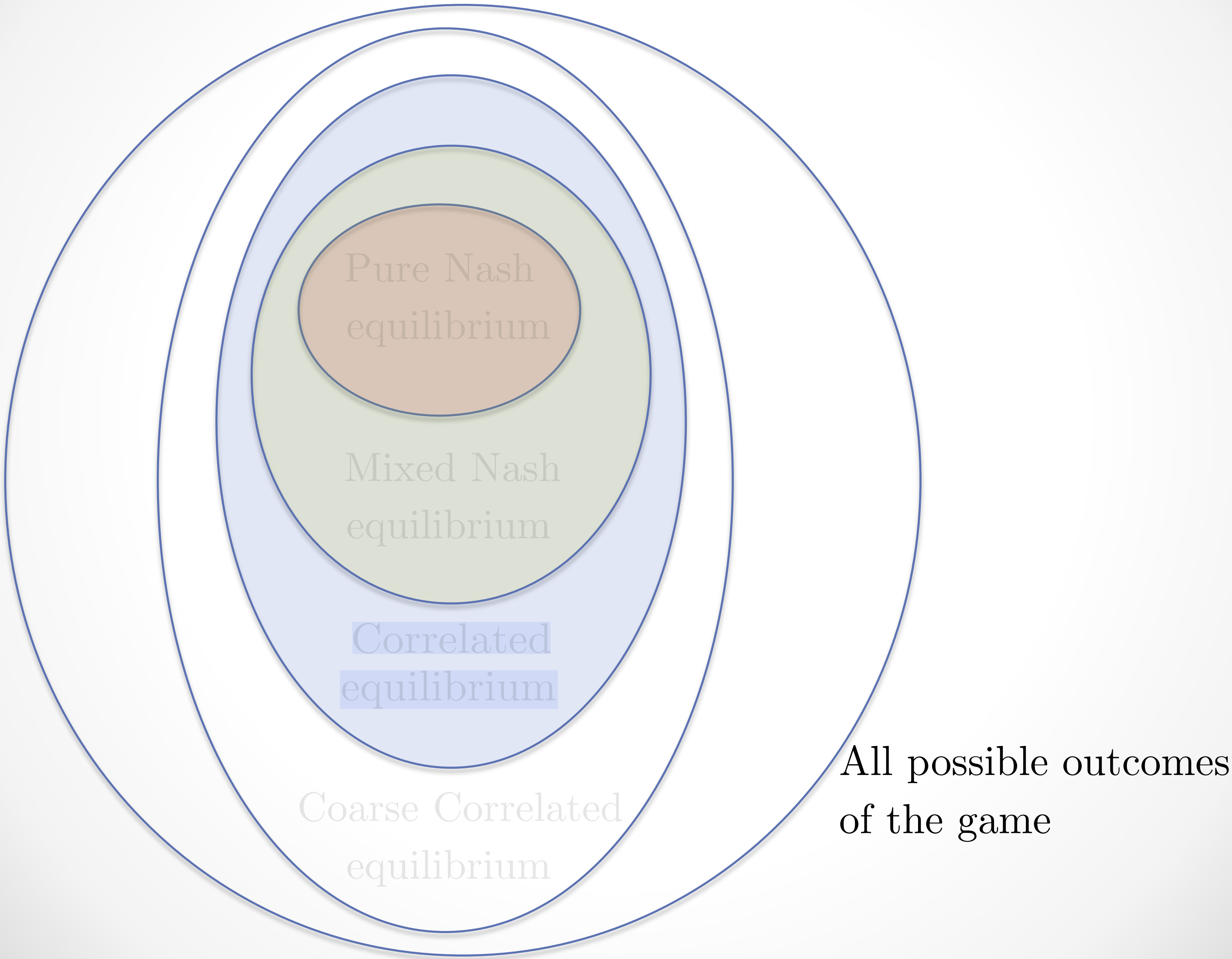
The distribution σ is a coarse correlated eq. iff for every player $j \in \mathcal{N}$,

$E_{\theta\sim\sigma}\left[c_j(\theta)\right] \leq E_{\theta\sim\sigma}\left[c_j(x,\theta_{-j})\right]$ for all strategies $x \in \mathcal{S}_j$.

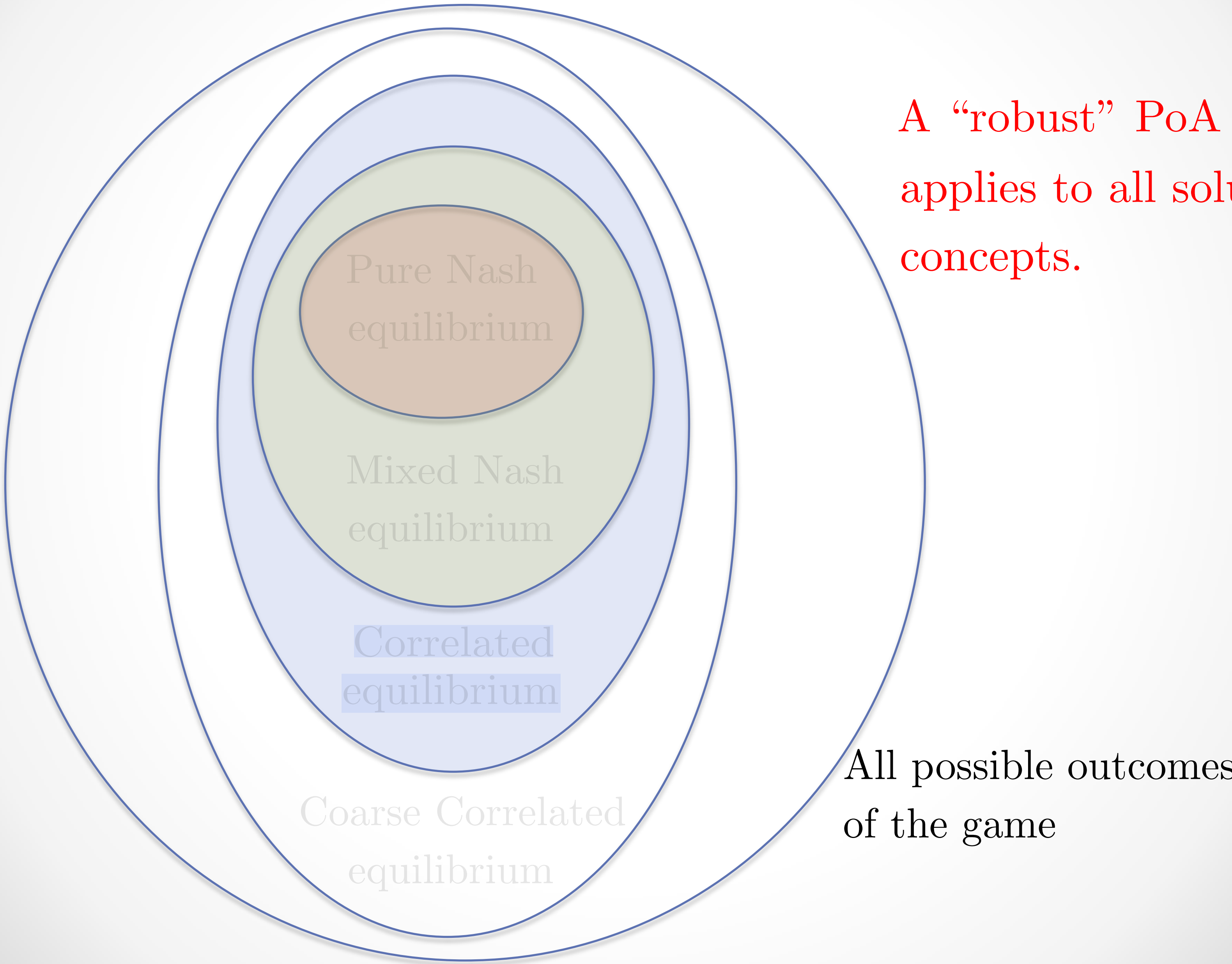
$E_{\theta\sim\sigma}[c_j(\theta)\,|\,\theta_j] \leq E_{\theta\sim\sigma}[c_j(x,\theta_{-j}\,|\,\theta_j)]$ $\Leftarrow$ Correlated equilibrium

In contrast with correlated eq., here the switching strategy of a player j does note depend on the suggestion θ_j received from the mediator.

Different ``Solution Concepts''



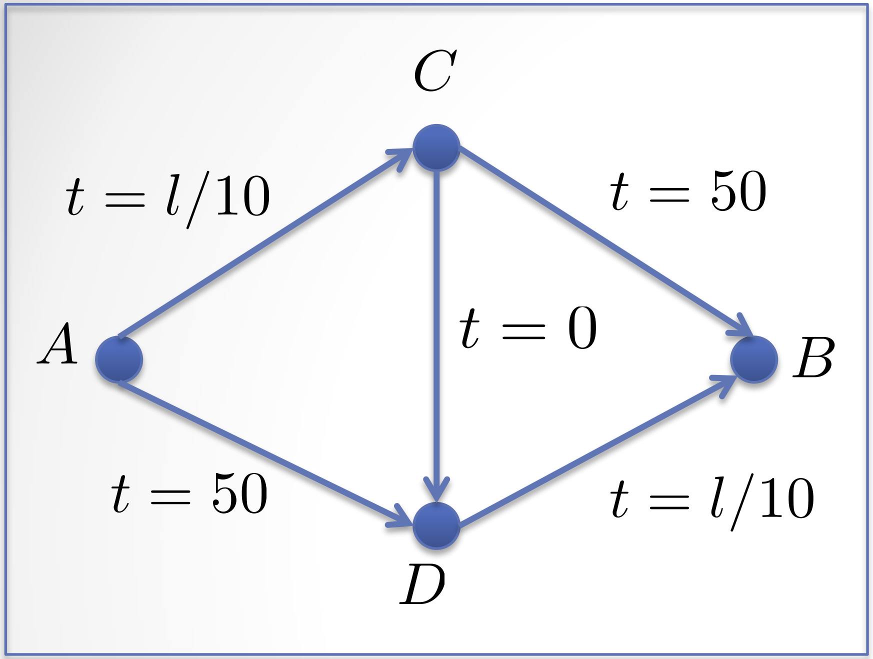
Robust Price of Anarchy



A "robust" PoA bound applies to all solution concepts.

Thank You.

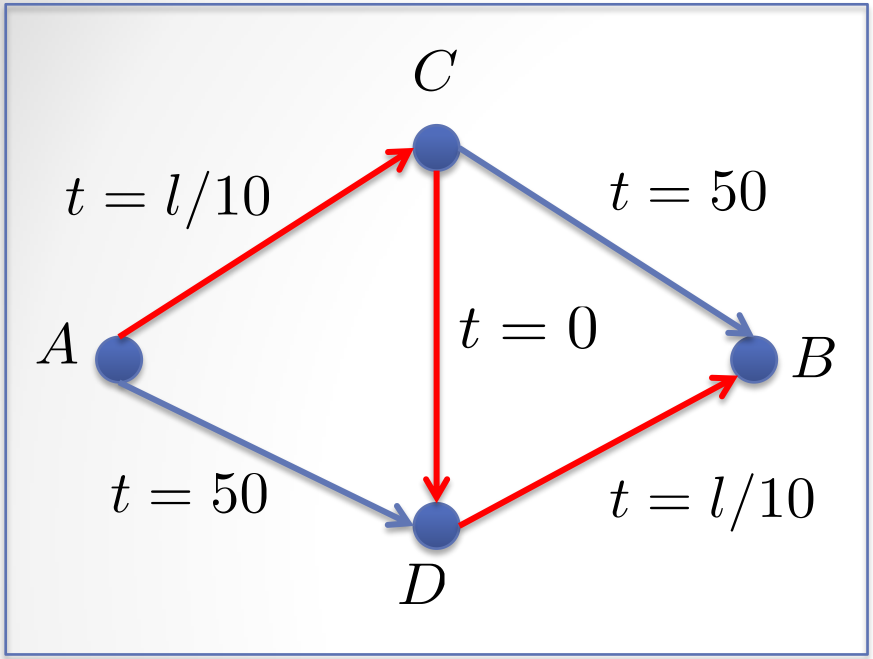
Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

Example: Selfish Routing



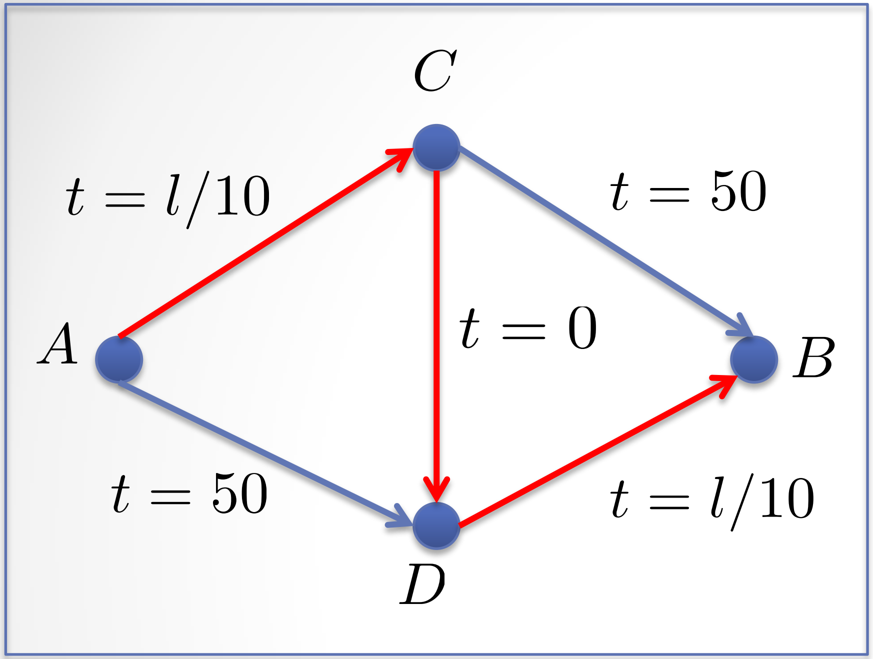
$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

All 400 cars take the route $ACDB$.

Travel time of each car $= 80$ mins.

Example: Selfish Routing



$t =$ time to traverse a link (mins)
$l =$ no. of cars taking the link

400 cars want to go from A to B.

All 400 cars take the route $ACDB$.

Travel time of each car $= 80$ mins.

This outcome is a pure Nash eq.