COMP323 – Introduction to Computational Game Theory

Selfish Routing and the Inefficiency of Equilibria

Paul G. Spirakis

Department of Computer Science
University of Liverpool

Outline

1. Modeling network traffic using game theory
2. Network congestion games
3. Selfish load balancing and the Price of Anarchy

Introduction

- Traveling through a transportation network, or sending packets through the Internet, involves fundamentally game-theoretic reasoning:
  Rather than simply choosing a route in isolation, individuals need to evaluate routes in the presence of the congestion resulting from the decisions made by themselves and everyone else.
- We will explore models for network traffic using game-theoretic ideas.

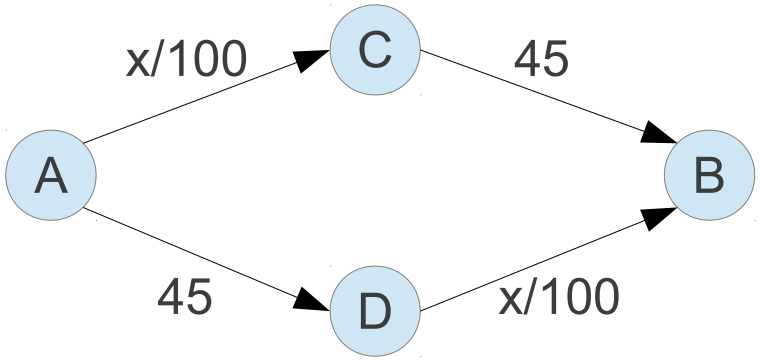
Transportation networks as directed graphs

We represent a transportation network by a directed graph:

- We consider the edges to be highways.
- We consider the nodes to be exits where you can get on or off a particular highway.
- There are two particular nodes, A and B, and we assume everyone wants to drive from A to B.
- Each edge has a designated travel time that depends on the amount of traffic it contains.

Transportation networks as directed graphs

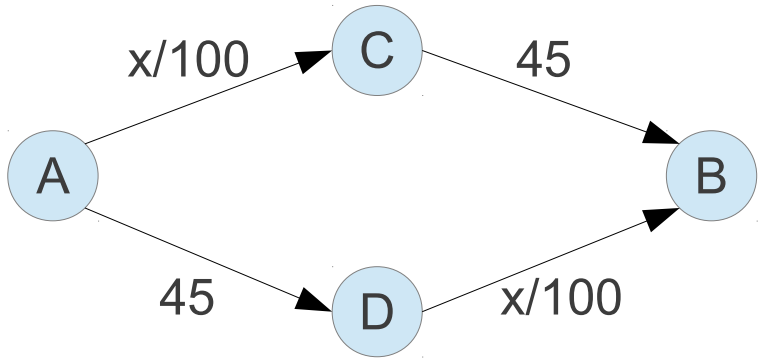
An example



- The label on each edge gives the travel time (in minutes) when there are x cars using the edge.
- The A-D and C-B edges are insensitive to congestion: each takes 45 minutes to traverse regardless of the number of cars on them.
- The A-C and D-B edges are highly sensitive to congestion: for each one, it takes $x/100$ minutes to traverse when there are x cars using the edge.

Transportation networks as directed graphs

An example



- Suppose that 4000 cars want to get from A to B.
- There are two possible routes that each car can choose: the upper route through C, or the lower route through D.
- If each car takes the upper route, then the total travel time for everyone is 85 minutes, since $4000/100 + 45 = 85$.
- The same is true if everyone takes the lower route.
- If the cars divide up evenly between the two routes, so that each carries 2000 cars, then the total travel time for people on both routes is $2000/100 + 45 = 65$.

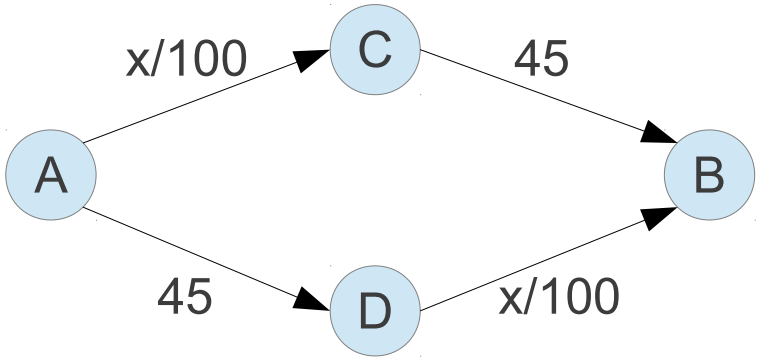
Network traffic as a game

The traffic model weve described is really a game in which

- the players correspond to the drivers,
- each player's possible actions or pure strategies consist of the possible routes from A to B, and the payoff for a player is the negative of his or her travel time (we use the negative since large travel times are bad).

In our example, this means that each player only has two strategies, but in larger networks, there could be many strategies for each player (possibly exponential to the number of nodes of the network!).

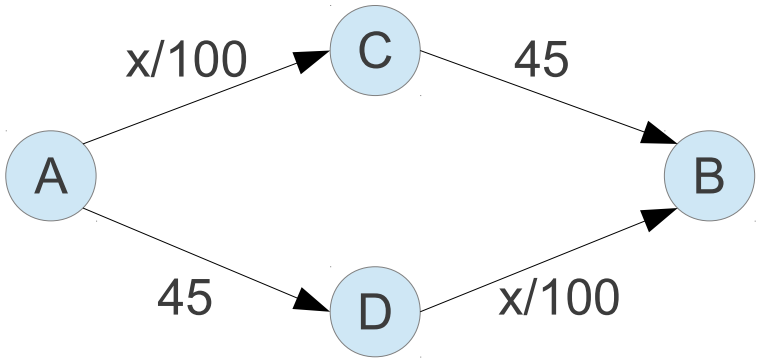
Equilibrium traffic



So what do we expect will happen?

- In this traffic game, there is generally no dominant strategy.
- Either route has the potential to be the best choice for a player if all the other players are using the other route.
- The game does have pure Nash equilibria, however: any list of strategies in which the drivers balance themselves evenly between the two routes (2000 on each) is a Nash equilibrium, and these are the only Nash equilibria.

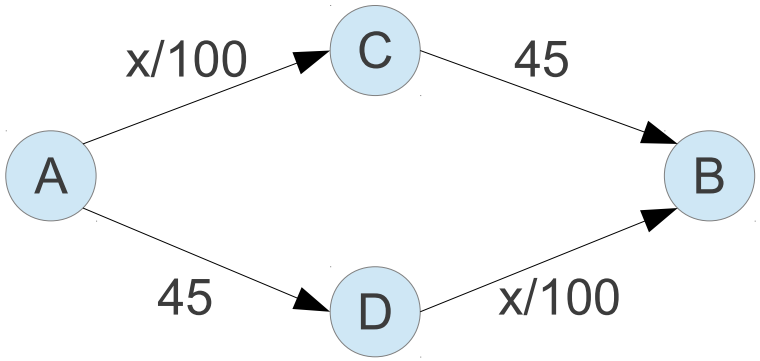
Equilibrium traffic



Why does equal balance yield a Nash equilibrium?

With an even balance between the two routes, no driver has an incentive to switch over to the other route.

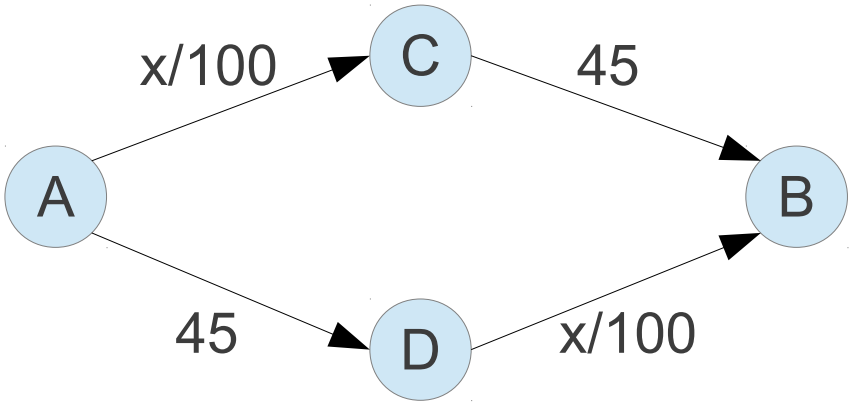
Equilibrium traffic



Why do all Nash equilibria have equal balance?

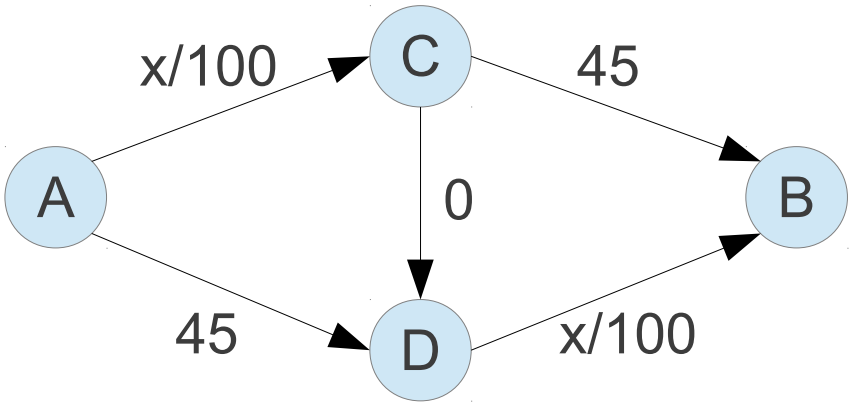
- Consider a list of strategies in which x drivers use the upper route and $4000 - x$ drivers use the lower route.
- If x is not equal to 2000, the two routes will have unequal travel times, and any driver on the slower route would have an incentive to switch to the faster one.
- Hence any list of strategies in which x is not equal to 2000 cannot be a Nash equilibrium.

Braess's paradox



- In this example, everything works out very cleanly: self-interested behavior by all drivers causes them, at equilibrium, to balance perfectly between the available routes.
- But with only a small change to the network, we can quickly find ourselves in truly counter-intuitive territory.

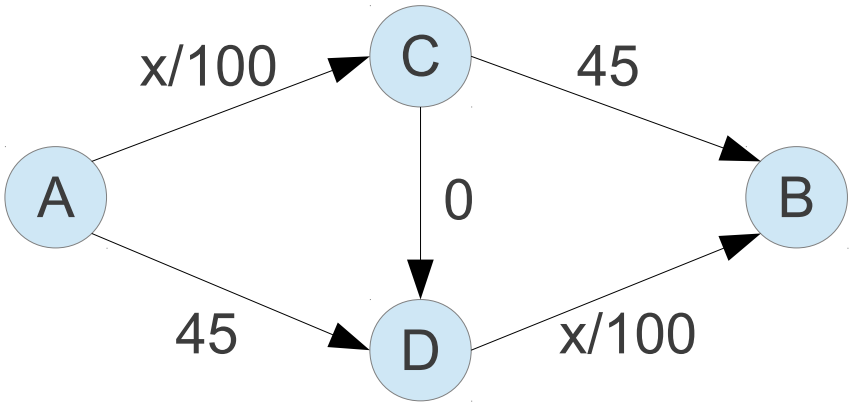
Braess's paradox



The change is as follows:

- We decide to build a new, very fast highway from C to D.
- We model its travel time as 0, regardless of the number of cars on it.
- It would stand to reason that people's travel time from A to B ought to get better after this edge from C to D is added.

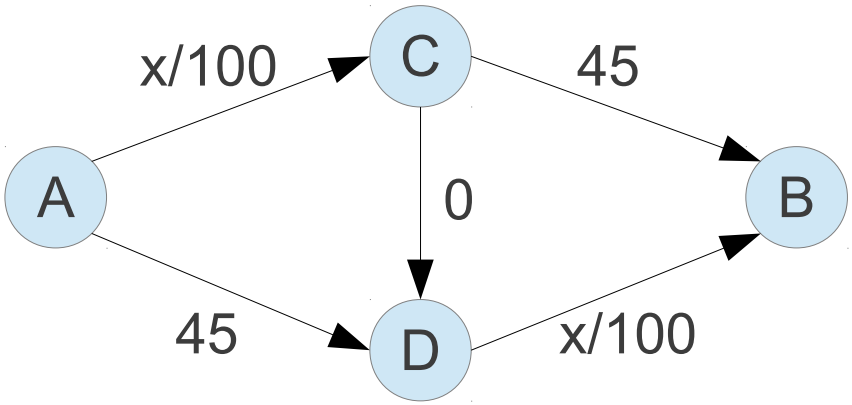
Braess's paradox



Here is the surprise:

- There is a unique Nash equilibrium in this new network, but it leads to a worse travel time for everyone.
- At equilibrium, every driver uses the route through both C and D.
- As a result, the travel time for every driver is $4000/100 + 0 + 4000/100 = 80$

Braess's paradox



- At equilibrium, every driver uses the route through both C and D: No driver can benefit by changing their route, since any other route would now take 85 minutes.
- This is the only equilibrium: the creation of the edge from C to D has in fact made the route through C and D a dominant strategy for all drivers: regardless of the current traffic pattern, you gain by switching your route to go through C and D.

Braess's paradox

- Once the fast highway from C to D is built, the route through C and D acts like a "vortex" that draws all drivers into it, to the detriment of all.
- In the new network there is no way, given individually self-interested behavior by the drivers, to get back to the even-balance solution that was better for everyone.
- This phenomenon, that adding resources to a transportation network can sometimes hurt performance at equilibrium, was first articulated by Dietrich Braess in 1968, and it has become known as Braess's Paradox.

Braess's paradox

Reflections

But is there actually something really "paradoxical" about Braess's paradox? Recall the Prisoner's Dilemma:

	Quiet	*Fink*
Quiet	(2,2)	(0,3)
Fink	(3,0)	(1,1)

- If the only strategy for each player were *Quiet*, then both players would be better off compared with the game where *Fink* is an option.
- Still, we have an informal sense that offering more options, such as "upgrading" a network, has to be a good thing, and so it is surprising when it turns out to make things worse.

Braess's paradox is actually the starting point for a large body of work on game-theoretic analysis of network traffic.

Network traffic as a game

We will focus on several issues regarding games modeling network traffic, including:

- Which are the appropriate models to study selfish behavior of network users?
- Do such games possess a pure strategy Nash equilibrium?
- Can we compute one efficiently?
- Are there "better" and "worse" equilibria, with respect to some global objective, such as the maximum delay over a path?
- How do the delay functions (travel times) on the edges affect the answer to the above questions?
- What happens if the network users (travellers) are not identical (i.e., have different weights)?

1. Modeling network traffic using game theory

2. Network congestion games
   - Network congestion models
   - Pure Nash equilibria and potential functions

3. Selfish load balancing and the Price of Anarchy

Congestion models

Recall our definition of a congestion model: it is defined by

1. a set of players $N = \{1, 2, \ldots, n\}$;
2. a set of resources $M = \{1, 2, \ldots, m\}$;
3. for each player $i \in N$, a set Σ_i of pure strategies of a player i, where each $A_i \in \Sigma_i$ is a non-empty subset of resources;
4. for each resource $j \in M$, a non-decreasing delay function $d_j() : \{1, \ldots, n\} \to \mathbb{R}_+$, where $d_j(k)$ denotes the cost (delay) to each user of resource j, if there are exactly k players using j.

Congestion models

Recall our definition of a congestion model: it is defined by

1. a set of players $N = \{1, 2, \ldots, n\}$;
2. a set of resources $M = \{1, 2, \ldots, m\}$;
3. for each player $i \in N$, a set Σ_i of pure strategies of a player i, where each $A_i \in \Sigma_i$ is a non-empty subset of resources;
4. for each resource $j \in M$, a non-decreasing delay function $d_j() : \{1, \ldots, n\} \to \mathbb{R}_+$, where $d_j(k)$ denotes the cost (delay) to each user of resource j, if there are exactly k players using j.

Observe that network traffic models are congestion models such that the resources correspond to the edges of a network and the set of pure strategies of each player correspond to source-destination paths in the network.

Network congestion games

More specifically, a general network congestion game is defined by:

- A directed network $G = (V, E)$ with the edges playing the role of resources.
- A set of players $N = \{1, 2, \ldots, n\}$ (the network users).
- For each player i, a source-destination pair of nodes $(s_i, t_i) \in V^2$.
- For each player i, a weight $w_i \in \mathbb{R}_+$, representing her load.
- For each edge $e \in E$, a non-decreasing delay function $d_e : \mathbb{R}_+ \to \mathbb{R}_+$.

Network congestion games

- Then, the action (pure strategy) set of player i is the set P_i of all network paths from s_i to t_i.
- We denote by $\varpi = (\varpi_i)_{i \in N} \in \times_{i \in N} P_i$ a configuration or pure strategy profile, where each user has chosen a path.
- We denote by $P = \times_{i \in N} P_i$ the set of all configurations.
- The delay $\theta_e(\varpi)$ on edge $e \in E$ in configuration $\varpi \in P$ is

$$\theta_e(\varpi) = d_e \left(\sum_{i : e \in \varpi_i} w_i \right) .$$

- The cost for user i in configuration ϖ is the total delay on her path:

$$\lambda_i(\varpi) = \sum_{e \in \varpi_i} d_e(\theta_e(\varpi)) .$$

Network congestion games

- The objective of each player i is to choose an $s_i - t_i$ path so that her cost is minimized.
- A network congestion game where all users have equal weights ($w_i = 1$ for all $i \in N$) is called unweighted.
- If all origin-destination pairs (s_i, t_i) of the players coincide with a unique pair (s, t) we have a *single-commodity network congestion game* and then all users share the same strategy set.
- An unweighted single-commodity network congestion game is a symmetric game (all users are identical and share the same strategy set).

Pure Nash equilibrium

Definition (PNE of a network congestion game)

A configuration $\varpi \in P$ is a pure Nash equilibrium of the network congestion game iff, for all $i \in N$,

$$\lambda_i(\varpi) \leq \lambda_i(\pi_i, \varpi_{-i}) \quad \forall \pi_i \in P_i$$

where (π_i, ϖ_{-i}) is the same configuration as ϖ except for user i that has now been assigned to path π_i.

Do pure equilibria exist?

Existence of PNE

Recall:

Theorem (Rosenthal, 1973)

Every unweighted congestion game admits an exact potential.

Proof. Rosenthal's potential, using our notation for network congestion games, is:

$$\Phi(\varpi) = \sum_{e \in E} \sum_{k=1}^{\sigma_e(\varpi)} d_e(k) \ ,$$

where

$$\sigma_e(\varpi) = |\{i \in N : e \in \varpi_i\}|$$

is the total number of users using edge e. □

Existence of PNE

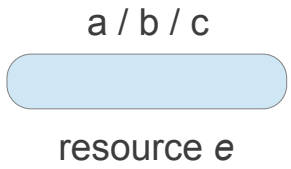
- Thus unweighted (network) congestion games are exact potential games.
- Therefore they possess pure strategy Nash equilibria.
- What about weighted network congestion games?
  - Are they exact potential games?
  - If not, do they admit some other kind of potential?

An potential does not exist in general

Q: Weighted network congestion game: does it have a potential?

A: Not in general.

- Two players $N = \{1, 2\}$ with loads $w_1 = 1$, $w_2 = 2$.
- Resource e has 2-wise linear delays:.
  - delay a when its total load is 1.
  - delay b when total load is 2.
  - delay c when total load is 3.



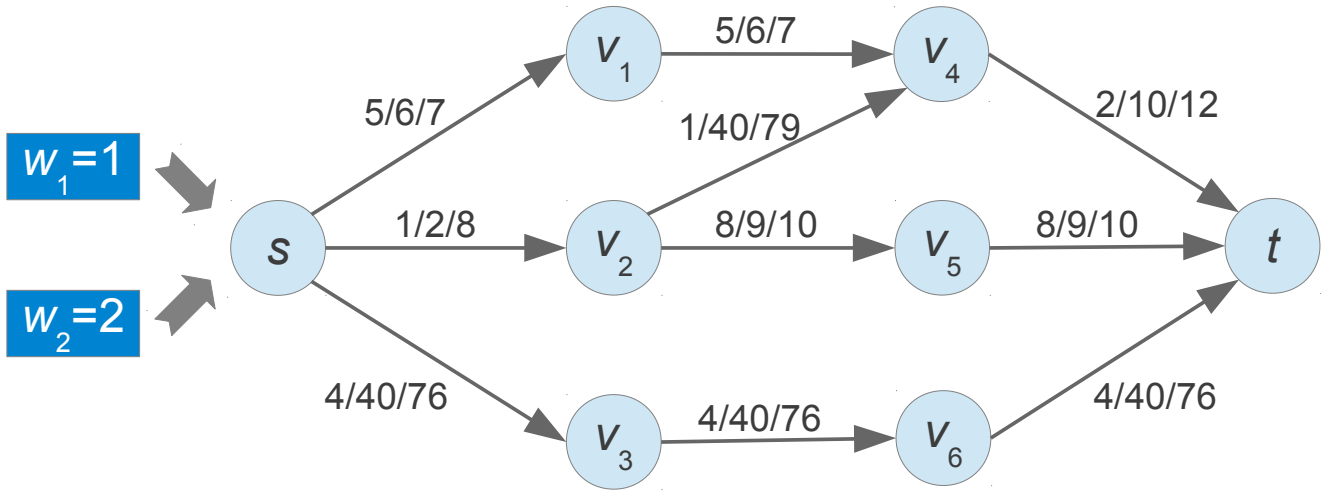
Theorem

There exist weighted network congestion games (even of 2 players) with 2-wise linear delays, for which there exists not even an ordinal potential (and no PNE).

Weighted congestion games

A potential does not exist in general

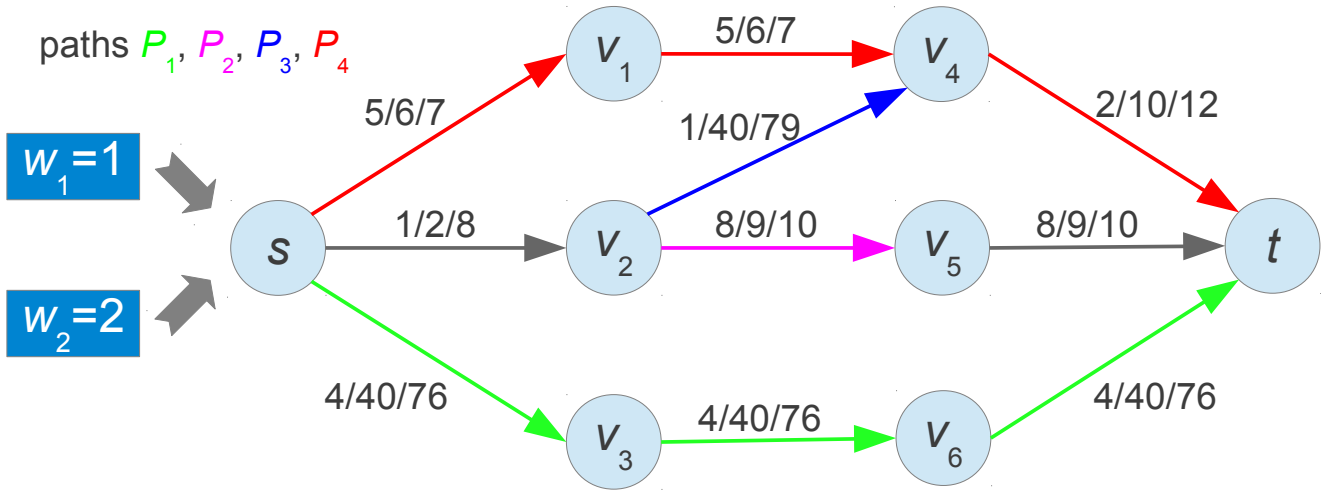
Proof.



Weighted congestion games

A potential does not exist in general

Proof.



- The "cycle" $\gamma = ((P_3, P_2), (P_3, P_4), (P_1, P_4), (P_1, P_2), (P_3, P_2))$ is an "improvement path"! (Each deviator moves to its new best choice.) $\Rightarrow$ No ordinal potential.
- Any other configuration is either one or two best-choice moves away from a configuration in γ $\Rightarrow$ No sink. $\square$

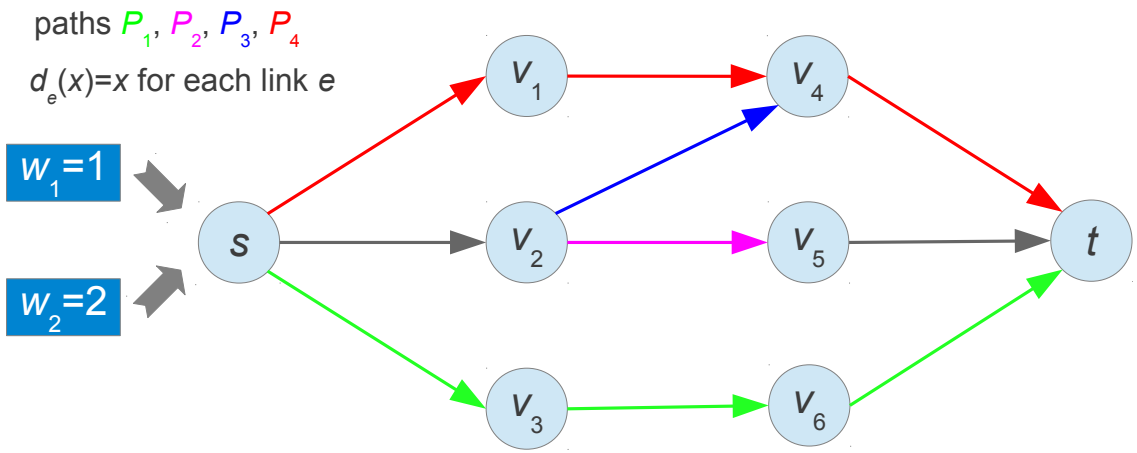
Weighted congestion games

A potential does not exist in general

Theorem

There exist weighted network congestion games with $d_e(x) = x$ which do not admit an exact potential.

Proof. Again (each link has $d_e(x) = x$)



Consider the 4-cycle $\gamma = ((P_1, P_3), (P_2, P_3), (P_2, P_1), (P_1, P_1), (P_1, P_3))$.

Weighted congestion games

A potential does not exist in general

Proof (continued). Let

$$I(\gamma) = \sum_{k=1}^{4} [\lambda_{i_k}(\varpi(k)) - \lambda_{i_k}(\varpi(k-1))]$$

where i_k is the unique deviator when we move from $\varpi(k-1)$ to $\varpi(k)$ in the 4-cycle γ.

Lemma (Monderer and Shapley, 1996)

Γ is an exact potential game $\Leftrightarrow$ $I(\gamma) = 0$ for all closed simple paths γ of length 4.

But our 4-cycle γ has $I(\gamma) = -4$. □

Existence of potentials and PNE

Are there families of weighted network congestion games that admit potentials and possess PNE?

- Let us consider linear resource delays $d_e(x) = a_e \cdot x + b_e$ (where x is the load on edge e)

Theorem

For any weighted (network) congestion game with linear (affine) resource delays, there exists a **b***-potential (weighted potential), and thus a PNE exists.*

Existence of potentials and PNE

Proof. Let ϖ be an arbitrary configuration. Define

$$\Phi(\varpi) = C(\varpi) + W(\varpi)$$

where

$$C(\varpi) = \sum_{e \in E} d_e(\theta_e(\varpi)) \cdot \theta_e(\varpi) = \sum_{e \in E} \left(a_e \theta_e^2(\varpi) + b_e \theta_e(\varpi)\right)$$

and

$$W(\varpi) = \sum_{i=1}^{n} \sum_{e \in \varpi_i} d_e(w_i) \cdot w_i = \sum_{i=1}^{n} \sum_{e \in \varpi_i} \left(a_e w_i^2 + b_e w_i\right) .$$

Existence of potentials and PNE

Proof (continued). Then $\Phi(\varpi)$ is a **b**-potential for $b_i = \frac{1}{2w_i}$:
Show that for player i, π_i a path different from ϖ_i, and $\varpi' = (\pi_i, \varpi_{-i})$,

a) $C(\varpi') - C(\varpi) =$
$2w_i \left(\sum_{e \in \pi_i \setminus \varpi_i} d_e(\theta_e(\varpi')) - \sum_{e \in \varpi_i \setminus \pi_i} d_e(\theta_e(\varpi)) \right) - t$

b) $W(\varpi') - W(\varpi) = t$

$$\Rightarrow \Phi(\varpi') - \Phi(\varpi) = 2w_i \left(\lambda_i(\varpi') - \lambda_i(\varpi) \right) \quad .$$

□

Existence of potentials and PNE

- Now let us consider the case of exponential delays: $d_e(\theta_e(\varpi)) = \exp(\theta_e(\varpi))$.

Theorem

Any weighted (network) congestion game with resource delays exponential to their loads admits a **b***-potential (and thus a PNE):*

$$\Phi(\varpi) = \sum_{e \in E} \exp(\theta_e(\varpi)) \ .$$

Existence of potentials and PNE

Proof. Show:

a) $\lambda_i(\varpi') - \lambda_i(\varpi)$

$$= \exp(w_i)\left(\sum_{e\in\pi_i\setminus\varpi_i}\exp(\theta_e(\varpi_{-i})) - \sum_{e\in\varpi_i\setminus\pi_i}\exp(\theta_e(\varpi_{-i}))\right)$$

$$= \exp(w_i)\cdot t$$

b) $\Phi(\varpi') - \Phi(\varpi) = (\exp(w_i) - 1)\cdot t$

$\Rightarrow \Phi(\varpi')$ is a **b**-potential with $b_i = \frac{\exp(w_i)}{\exp(w_i)-1}$. □

Existence of potentials and PNE

- (PNE-)consistency: which types of delays guarantee existence of PNE
- FIP-consistency: congestion games possessing the FIP (finite improvement property)

Theorem (Harks and Klim, 2012)

For at least 3 players:

(a) *For weighted congestion, consistency $\equiv$ FIP-consistency.*

(b) *A set of continuous delay functions is consistent iff (b1) it is affine: $d_e(x) = a_e x + b_e$, or (b2) it is exponential: $d_e(x) = a_e e^{\xi \cdot x} + b_e$ (where ξ same for all resources).*

I.e., for $|N| \geq 3$ players, linear-affine and exponential delays (as generalized by Harks and Klim for $d_e(x) = a_e \exp(\xi x) + b_e$) are the only possible cases!

Explains why many other "practical" delays (e.g., M/M/1) are unstable!

Inefficiency of equilibria: the price of anarchy

- The outcome of rational behavior by selfish players can be inferior to a centrally designed outcome (recall the Prisoner's Dilemma and Braess's paradox).
- By how much?

The most popular measure of the inefficiency of equilibria is the price of anarchy:

Definition

The price of anarchy of a game is defined as the ratio between the worst objective function value of an equilibrium of the game and that of an optimal outcome.

The price of anarchy

- Note that the price of anarchy of a game is defined with respect to a choice of objective function and a choice of equilibrium concept.
- We are interested in identifying games in which the price of anarchy is close to 1; in these games, all equilibria are good approximations of an optimal outcome.
- We view selfish behavior as benign in such games.

We will study the price of anarchy in the context of load balancing games.

Selfish load balancing

- A set of weighted tasks shall be assigned to a set of machines with possibly different speeds such that the load is distributed evenly among the machines.
- In computer science, this problem is traditionally treated as an optimization problem.
- One of the classical objectives is to minimize the makespan, i.e., the maximum load over all machines.
- A natural game-theoretic variant of the problem is to assume that the tasks are managed by selfish agents: each task has an agent that aims at placing the task on the machine with smallest load.
- We will study the Nash equilibria and the price of anarchy of this game.

Load balancing

Load balancing is a fundamental problem to networks and distributed systems. The most fundamental load balancing problem is makespan scheduling on uniformly related machines:

- There are m machines with speeds $s_1, \ldots, s_m$ and
- n tasks with weights $w_1, \ldots, w_n$.
- Let $[n] = \{1, \ldots, n\}$ denote the set of tasks and $[m] = \{1, \ldots, m\}$ the set of machines.
- One seeks for an assignment $A : [n] \to [m]$ of the tasks to the machines that the assignment is as balanced as possible.

Load balancing

- The load of machine $j \in [m]$ under assignment A is defined as

$$\ell_j = \sum_{i:j=A(i)} \frac{w_i}{s_j} \ .$$

- The makespan is defined to be the maximum load over all machines.
- The objective is to minimize the makespan.
- If all machines have the same speed, then the problem is known as makespan scheduling on identical machines, in which case we shall assume $s_1 = s_2 = \cdots = s_m = 1$.

Load balancing games

- We identify agents and tasks, i.e., task $i \in [n]$ is managed by agent i.
- Each agent can place its task on one of the machines.
- In other words, the set of pure strategies for an agent is $[m]$.
- A combination of pure strategies, one for each task, yields an assignment $A : [n] \to [m]$.
- We assume that the cost of agent i under the assignment A corresponds to the load on machine $A(i)$, i.e., its cost is $\ell_{A(i)}$.
- The social cost of an assignment is denoted $cost(A)$ and is defined to be the makespan, i.e.,

$$cost(A) = \max_{j \in [m]} (\ell_j) \ .$$

Load balancing games

- Agents may use mixed strategies, i.e., probability distributions on the set of pure strategies.
- Let p_i^j denote the probability that agent i assigns its task to machine j, i.e.,
$$p_i^j = \Pr\{A(i) = j\} \ .$$
- A strategy profile $P = (p_i^j), i \in [n], j \in [m]$ specifies the probabilities for all agents and all machines.
- For $i \in [n], j \in [m]$, let x_i^j be a random variable that takes the value 1 if $A(i) = j$ and 0, otherwise.
- The expected load of machine j under the strategy profile P is thus
$$E[\ell_j] = E\left[\sum_{i \in [n]} \frac{w_i x_i^j}{s_j}\right] = \sum_{i \in [n]} \frac{w_i E[x_i^j]}{s_j} = \sum_{i \in [n]} \frac{w_i p_i^j}{s_j} \ .$$

Load balancing games

- The social cost of a strategy profile P is defined as the expected makespan, i.e.,

$$cost(P) = E[cost(A)] = E\left[\max_{j\in[m]}(\ell_j)\right] \ .$$

- We assume that every agent aims at minimizing its expected cost.
- From the point of view of agent i, the expected cost on machine j, denoted by c_i^j, is

$$c_i^j = E[\ell_j \mid A(i) = j] \ .$$

- For any profile P,

$$c_i^j = \frac{w_i + \sum_{k\neq i} w_k p_k^j}{s_j} = E[\ell_j] + (1 - p_i^j) \cdot \frac{w_i}{s_j} \ .$$

Nash equilibria

Definition

- In general, a strategy profile of a game is a Nash equilibrium if there is no incentive for any agent to unilaterally change its strategy.
- For the load balancing game, such a profile is characterized by the property that every agent assigns positive probabilities only to those machines that minimize its expected cost. This is formalized as follows.

Definition

A strategy profile P is a Nash equilibrium if and only if, for all $i \in [n]$, for all $j \in [m]$,

$$p_i^j > 0 \implies c_i^j \leq c_i^k \quad \forall k \in [m] \ .$$

Nash equilibria

Existence

- The existence of a Nash equilibrium (in mixed strategies) is guaranteed by Nash's theorem.
- Is there a pure Nash equilibrium (PNE)?

Definition

An assignment A is a pure Nash equilibrium if and only if, for all $i \in [n]$,

$$c_i^{A(i)} \leq c_i^k \quad \forall k \in [m] \ .$$

- In words, an assignment is a pure Nash equilibrium if and only if no agent can improve its cost by unilaterally moving its task to another machine.
- Load balancing games always admit pure Nash equilibria.

Existence of PNE

Why do load balancing games admit PNE?

- Observe that they are a special case of weighted network congestion games with linear resource delays, so they admit a weighted potential, so they have a PNE.
- We will give an alternative proof of existence of a PNE, and in particular of an optimal PNE, i.e., a PNE that minimizes the makespan.

Proof of PNE existence

Theorem

Every instance of the load balancing game admits at least one pure Nash equilibrium.

Proof.

- An assignment A induces a sorted load vector $(\lambda_1, \ldots, \lambda_m)$, where λ_j denotes the load on the machine that has the jth highest load.
- If an assignment is not a Nash equilibrium, then there exists an agent i that can perform an improvement step, i.e., it can decrease its cost by moving its task to another machine.
- If we show that the sorted load vector an improvement step is lexicographically smaller than the one preceding it, then a pure Nash equilibrium is reached after a finite number of improvement steps.

Proof of PNE existence

Theorem

Every instance of the load balancing game admits at least one pure Nash equilibrium.

Proof (continued).

- Given any sorted load vector $(\lambda_1, \ldots, \lambda_m)$, assume i performs an improvement step and moves from machine j to k where the indices are w.r.t. to the positions of the machines in the vector.
- Clearly, $k > j$.
- The improvement step decreases the load on machine j and it increases the load on machine k.
- The increased load on machine k is smaller than λ_j as, otherwise, agent i would not decrease its cost.

Proof of PNE existence

Theorem

Every instance of the load balancing game admits at least one pure Nash equilibrium.

Proof (continued).

- Hence, the number of machines with load at least λ_j is decreasing.
- The loads on all other machines are left unchanged.
- Consequently, the improvement step yields a sorted load vector lexicographically smaller than $(\lambda_1, \ldots, \lambda_m)$.

□

Optimal PNE

- Thus improvement steps naturally lead to a pure Nash equilibrium.
- This convergence result implies that there exists even a pure Nash equilibrium that minimizes the makespan.
- Given any optimal assignment, such an equilibrium can be found by performing improvement steps until a Nash equilibrium is reached because improvement steps do not increase the makespan.

Price of stability and price of anarchy

In the context of load balancing games with social cost equal to the makespan:

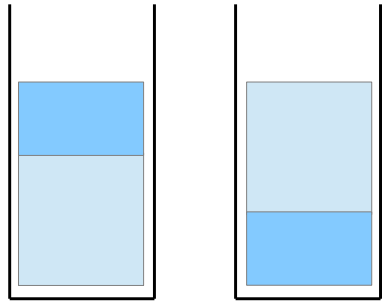
- The price of stability, i.e., the ratio between the social cost in a best Nash equilibrium and the optimal social cost, is 1.
- We are interested in the ratio between the social cost of the worst Nash equilibrium and the optimal social cost, the so-called price of anarchy.

Example of a load balancing game

Suppose that there are

- two identical machines both of which have speed 1 and
- four tasks, two small tasks of weight 1 and two large tasks of weight 2.

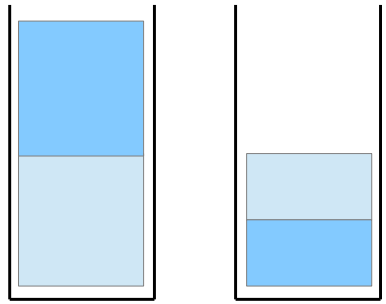
An optimal assignment would map a small and a large task to each of the machines so that the load on both machines is 3:



Example of a load balancing game

Pure equilibria

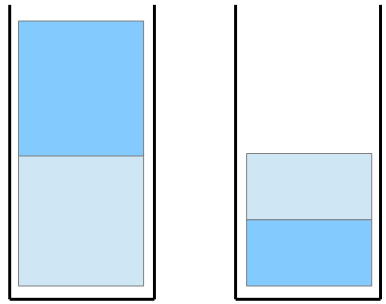
Now consider an assignment A that maps the two large tasks to the first machine and the two small tasks to the second machine:



- The first machine has a load of 4 and the second machine has a load of 2.
- A small task cannot improve its cost by moving from the second to the first machine.
- A large task cannot improve its cost by moving from the first to the second machine either as its cost would remain 4 if it does.

Example of a load balancing game

Pure equilibria



- Thus this assignment A constitutes a pure Nash equilibrium with $cost(A) = 4$.
- All assignments that yield a larger makespan than 4 cannot be a Nash equilibrium as, in this case, one of the machines has a load of at least 5 and the other has a load of at most 1 so that moving any task from the former to the latter would decrease the cost of this task.
- Thus, for this instance of the load balancing game, the social cost of the worst pure Nash equilibrium is 4.

Example of a load balancing game

Mixed equilibria

Clearly, the worst mixed equilibrium cannot be better than the worst pure equilibrium: the set of mixed equilibria is a superset of the set of pure equilibria.

But can it really be worse?

Example of a load balancing game

Mixed equilibria

Clearly, the worst mixed equilibrium cannot be better than the worst pure equilibrium: the set of mixed equilibria is a superset of the set of pure equilibria.

But can it really be worse?

- Suppose that each task is assigned to each of the machines with probability $1/2$.
- This corresponds to the strategy profile $P = (p_i^j)$ with $p_i^j = 1/2$ for $1 \leq i \leq 4, \quad 1 \leq j \leq 2$.
- The expected load on machine j is

$$E[\ell_j] = \sum_{i=1}^{4} w_i p_i^j = 2 \cdot 2 \cdot \frac{1}{2} + 2 \cdot 1 \cdot \frac{1}{2} = 3 \ .$$

Example of a load balancing game

Mixed equilibria

Notice that the expected cost of a task on a machine is larger than the expected load of the machine, unless the task is assigned with probability 1 to this machine:

- If we assume that task 1 is a large task, then

$$c_1^1 = E[\ell_1] + (1 - p_1^1)w_1 = 3 + \frac{1}{2} \cdot 2 = 4 \ ,$$

- and if task 3 is a small task, then

$$c_3^1 = E[\ell_1] + (1 - p_3^1)w_3 = 3 + \frac{1}{2} \cdot 1 = 3.5 \ .$$

Example of a load balancing game

Mixed equilibria

For symmetry reasons, the expected cost of each task under profile P is the same on both machines so that P is a Nash equilibrium.

- The social cost of this NE, $cost(P)$, is defined to be the expected makespan, $E[cost(A)]$, of the random assignment A induced by P.
- The makespan, $cost(A)$, is a random variable. This variable can possibly take one of the four values 3, 4, 5, or 6.
- There are $2^4 = 16$ different assignments of four tasks to two machines. The number of assignments that yield a makespan of 3 is 4, 4 is 6, 5 is 4, and 6 is 2.
- Consequently, the social cost of the mixed Nash equilibrium is

$$cost(P) = E[cost(A)] = \frac{1}{16}(3 \cdot 4 + 4 \cdot 6 + 5 \cdot 4 + 6 \cdot 2) = 4.25 \ .$$

Thus mixed equilibria can, in fact, be worse than the worst pure equilibrium.

The Price of Anarchy: Definition

We are interested in the ratio between the social cost (makespan) of a worst-case Nash equilibrium, and the social cost of an optimal assignment.

Definition (Price of anarchy)

- For $m \in \mathbb{N}$, let $\mathcal{G}(m)$ denote the set of all instances of load balancing games with m machines.
- For $G \in \mathcal{G}(m)$, let $Nash(G)$ denote the set of all strategy profiles being a Nash equilibrium for G, and let $opt(G)$ denote the minimum social cost over all assignments.

Then the price of anarchy is defined by

$$PoA(m) = \max_{G \in \mathcal{G}(m)} \max_{P \in Nash(G)} \frac{cost(P)}{opt(G)} .$$

The price of anarchy in load balancing

We will study the PoA in load balancing games in four different variants, based on two criteria:

1. games with identical and uniformly related machines;
2. pure Nash equilibria and mixed Nash equilibria.

- When considering the PoA for games with identical machines then we restrict the set $G(m)$ to instances in which the m machines have all the same speed.
- When considering the PoA with respect to pure equilibria, we take the maximum only among pure equilibrium assignments rather than among possibly mixed equilibrium strategy profiles.

PoA for PNE on identical machines

Theorem

Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m identical machines. Let $A : [n] \to [m]$ denote any Nash equilibrium assignment. Then,

$$cost(A) \leq \left(2 - \frac{2}{m+1}\right) opt(G) \ .$$

Proof.

- Let j^* be a machine with the highest load under assignment A.
- Let i^* be a task of smallest weight assigned to this machine.
- W.l.o.g., there are at least two tasks assigned to machine j^* as, otherwise, $cost(A) = opt(G)$ so that the theorem follows trivially.
- Thus $w_{i^*} \leq \frac{1}{2} cost(A)$.

PoA for PNE on identical machines

Proof (continued).

- Suppose there is a machine j with load less than $\ell_{j^*} - w_{i^*}$.
- Then moving the task i^* from j^* to j would decrease the cost for this task.
- Hence, as A is a Nash equilibrium, it holds

$$\ell_j \geq \ell_{j^*} - w_{i^*} \geq cost(A) - \frac{1}{2} cost(A) = \frac{1}{2} cost(A) \ .$$

PoA for PNE on identical machines

Proof (continued). Now observe that the cost of an optimal assignment cannot be smaller than the average load over all machines so that

$$\begin{aligned} opt(G) &\geq \frac{\sum_{i\in[n]} w_i}{m} = \frac{\sum_{j\in[m]} \ell_j}{m} \\ &\geq \frac{cost(A) + \frac{1}{2}cost(A)(m-1)}{m} \\ &= \frac{(m+1)cost(A)}{2m} \ , \end{aligned}$$

implying that

$$cost(A) \leq \frac{2m}{m+1} opt(G) = \left(2 - \frac{2}{m+1}\right) opt(G) \ .$$

□

Convergence time of best responses

How may agents find or compute a Nash equilibrium efficiently?

- The existence proof for pure equilibria implicitly shows that every sequence of improvement steps by the agents leads to a Nash equilibrium.
- Do the players converge to an equilibrium in reasonable time?
- How good is the finally reached equilibrium?

We will show that, in the case of identical machines, there is a short sequence of improvement steps that leads from any given initial assignment to a pure Nash equilibrium.

Convergence time of best responses

- An agent is said to be satisfied if it cannot reduce its cost by unilaterally moving its task to another machine.
- The max-weight best response policy:
  - activates the agents one after the other, always activating an agent with maximum weight among the unsatisfied agents;
  - an activated agent plays a best response; i.e., the agent moves its task to the machine with minimum load.

Convergence time of best responses

Theorem

Let $A : [n] \to [m]$ denote any assignment of n tasks to m identical machines. Starting from A, the max-weight best response policy reaches a pure Nash equilibrium after each agent was activated at most once.

Proof. We will show that, once an agent $i \in [n]$ was activated and played its best response, it never gets unsatisfied again.

- An agent is satisfied if and only if its task is placed on a machine on which the load due to the other tasks is minimal.
- A best response never decreases the minimum load among the machines.

Convergence time of best responses

Proof (continued). As a consequence, a satisfied agent can get unsatisfied only for one reason: the load on the machine holding its task increases because another agent moves its task to the same machine.

- Suppose that agent k is activated after agent i, and it moves its task to the machine holding task i.
- Let j^* be the machine on which i is placed and to which k is moved.
- For $j \in [m]$, let ℓ_j denote the load on machine j at the time immediately after the best response of agent k.

Convergence time of best responses

Proof (continued).

- Since the assignment of k to j^* is a best response and as $w_k \leq w_i$ because of the max-weight policy, it follows

$$\ell_{j^*} \leq \ell_j + w_k \leq \ell_j + w_i \quad \forall j \in [m] \ .$$

- Hence, after the best response of k, agent i remains satisfied on machine j^*.

□

PoA for PNE on uniformly related machines

Theorem

Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m machines of speed $s_1, \ldots, s_m$. Let $A : [n] \to [m]$ denote any Nash equilibrium assignment. Then, it holds that

$$cost(A) = O\left(\frac{\log m}{\log \log m}\right) \cdot opt(G) \ .$$

Proof.

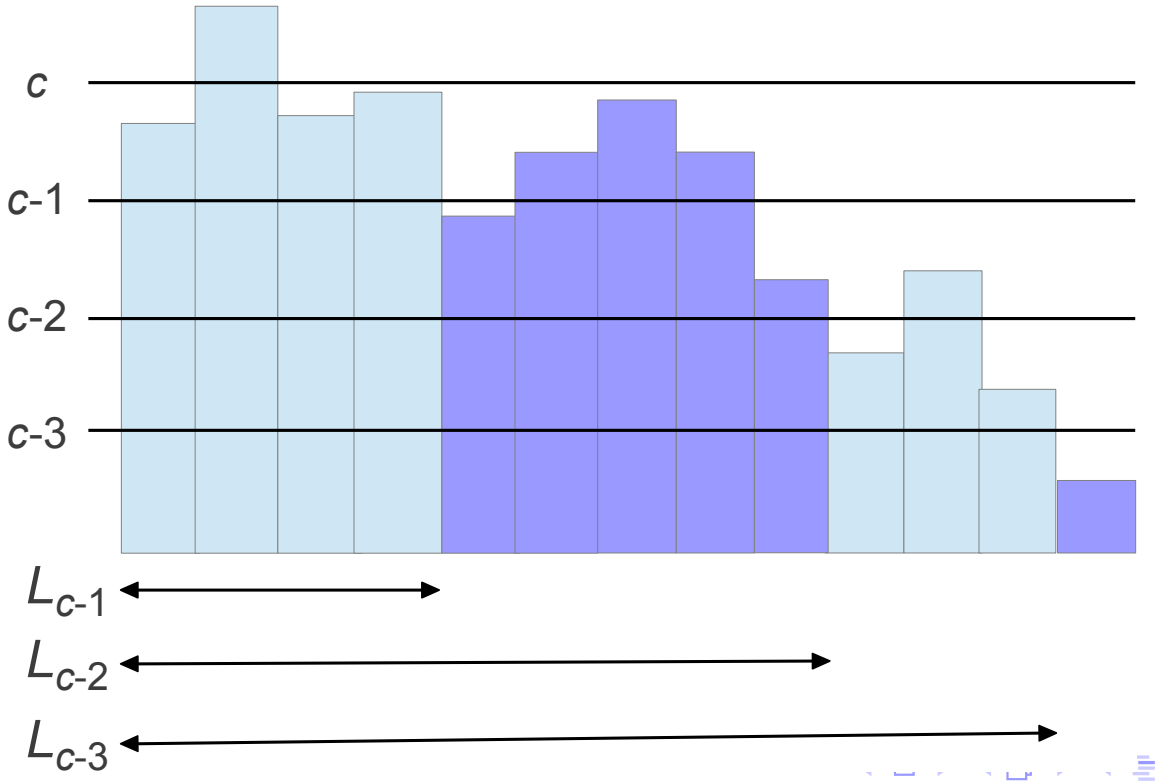
- Let $c = \lfloor cost(A)/opt(G) \rfloor$.
- We will show that $c \leq \Gamma^{-1}(m)$, where Γ^{-1} is the inverse of the gamma function:

$$\begin{aligned} \Gamma(k) &= (k-1)! \\ \Gamma^{-1}(k) &= \Theta\left(\frac{\log k}{\log \log k}\right) \ . \end{aligned}$$

PoA for PNE on uniformly related machines

Proof (continued).

- Assume $s_1 \geq \cdots \geq s_m$ and let $L = [1, \ldots, m]$ denote the list of machines in non-increasing order of speed.
- For $k \in \{0, \ldots, c-1\}$, let L_k denote the maximum length prefix of L such that the load of each server in L_k is at least $k \cdot opt(G)$:



PoA for PNE on uniformly related machines

Proof (continued). Assume L_{c-1} is empty. Then:

- The load of machine 1 is less than $(c-1)\cdot opt(G)$ in the equilibrium assignment A.
- Let i be a task placed on a machine j with load at least $c\cdot opt(G)$.
- Moving i to machine 1 reduces the cost of i to strictly less than

$$(c-1)\cdot opt(G)+\frac{w_i}{s_1}\leq (c-1)\cdot opt(G)+opt(G)=c\cdot opt(G)\ ,$$

which contradicts the assumption that A is a Nash equilibrium.

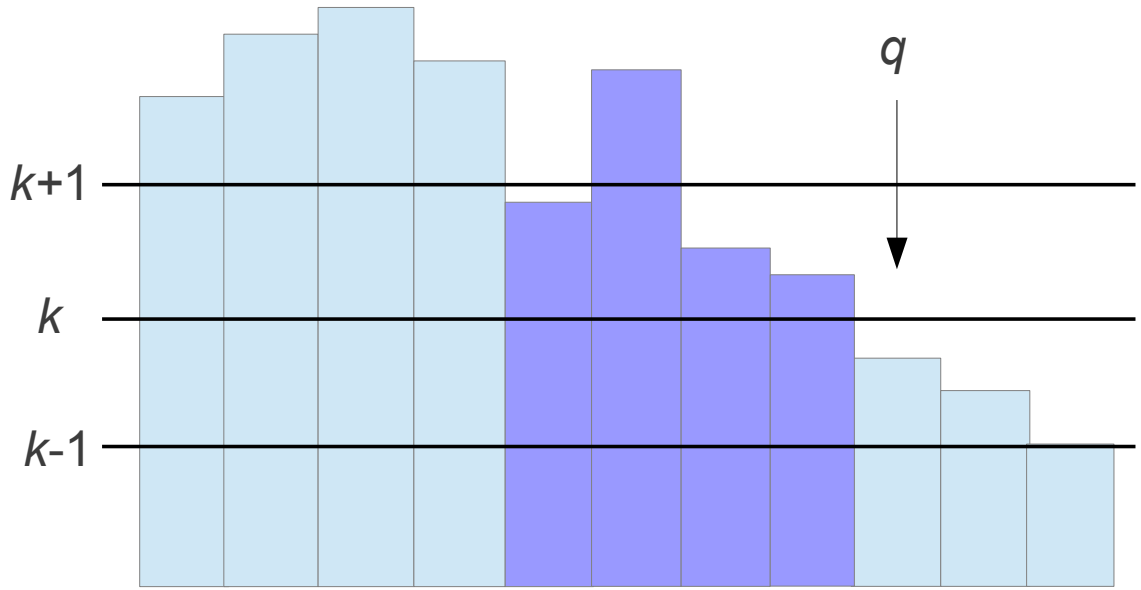
Thus, we have shown that

$$|L_{c-1}|\geq 1\ .$$

PoA for PNE on uniformly related machines

Proof (continued). Let A^* be an optimal assignment. We will show that, if i is a task with $A(i) \in L_{k+1}$ then $A^*(i) \in L_k$.

- If $L \setminus L_k = \emptyset$ the claim follows trivially.
- Let q be the smallest index in $L \setminus L_k$. Then $\ell_q < k \cdot opt(G)$.



PoA for PNE on uniformly related machines

Proof (continued).

- $A(i) \in L_{k+1}$ implies $\ell_{A(i)} \geq (k+1) \cdot opt(G)$.
- If $w_i \leq s_q \cdot opt(G)$, then moving task i to q would reduce its cost to

$$\ell_q + \frac{w_i}{s_q} < k \cdot opt(G) + opt(G) \leq \ell_{A(i)} \ ,$$

  contradicting the equilibrium.
- Hence $A(i) \in L_{k+1}$ implies $w_i > s_q \cdot opt(G)$.

PoA for PNE on uniformly related machines

Proof (continued).

- Now assume $A^*(i) = j$ and $j \in L \setminus L_k$. Then the load on j under A^* would be at least

$$\frac{w_i}{s_j} > \frac{s_q \cdot opt(G)}{s_j} \geq opt(G) \ ,$$

  contradicting the optimality of A^*.
- Hence, if i is a task with $A(i) \in L_{k+1}$ then $A^*(i) \in L_k$.

PoA for PNE on uniformly related machines

Proof (continued). We will now show that $|L_k| \geq (k+1) \cdot |L_{k+1}|$ for all k such that $0 \leq k \leq c-2$.

- By definition of L_{k+1}, the sum of the weights that A assigns to a machine $j \in L_{k+1}$ is at least $(k+1) \cdot opt(G) \cdot s_j$.
- An optimal assignment has to assign all this weight to the machines in L_k such that the load on each is at most $opt(G)$. Therefore:

$$\begin{aligned} \sum_{j \in L_{k+1}} (k+1) \cdot opt(G) \cdot s_j &\leq \sum_{j \in L_k} opt(G) \cdot s_j \\ \sum_{j \in L_{k+1}} k \cdot s_j &\leq \sum_{j \in L_k \setminus L_{k+1}} s_j \ . \end{aligned}$$

PoA for PNE on uniformly related machines

Proof (continued).

- Let $s^* = s_{|L_{k+1}|}$ be the speed of the slowest machine in L_{k+1}. Then

$$\sum_{j \in L_{k+1}} k \cdot s^* \leq \sum_{j \in L_k \setminus L_{k+1}} s^* \ ,$$

which implies

$$|L_{k+1}| \cdot k \leq |L_k \setminus L_{k+1}| = |L_k| - |L_{k+1}|$$

and our claim that

$$|L_k| \geq (k+1) \cdot |L_{k+1}|$$

follows.

PoA for PNE on uniformly related machines

Proof (continued). We have shown the following recurrence:

$$\begin{aligned} |L_k| &\geq (k+1) \cdot |L_{k+1}| \quad 0 \leq k \leq c-2 \\ |L_{c-1}| &\geq 1 \ . \end{aligned}$$

Solving the recurrence yields $|L_0| \geq (c-1)! = \Gamma(c)$.
Observe that $|L_0| = m$, so $m \geq \Gamma(c)$ and

$$c \leq \Gamma^{-1}(m)$$

which proves the theorem. □

Algorithms for computing pure equilibria

- The proof of existence of PNE reveals that, starting from any initial assignment, a PNE is reached after a finite number of improvement steps.
- We have seen that there exists a sequence of improvement steps of length $O(n)$ in case of identical machines and this sequence can be computed efficiently.
- For the case of uniformly related machines, it is not known whether there always exists a short sequence of improvement steps and whether such a sequence can be efficiently computed.
- However, the well-known LPT (largest processing time) scheduling algorithm allows us to efficiently compute a Nash equilibrium.

Algorithms for computing pure equilibria

The LPT algorithm:

- inserts the tasks in a nonincreasing order of weights;
- each task is assigned to a machine that minimizes the cost of the task at its insertion time.

Theorem

The LPT algorithm computes a PNE for load balancing games on uniformly related machines.

Algorithms for computing pure equilibria

Proof.

- Let the tasks be numbered from 1 to n in the order of their insertion.
- Let time $t \in \{0, \ldots, n\}$ denote the point of time after the first t tasks have been inserted.
- We will show by induction that the partial assignment $A : [t] \to [m]$ computed by LPT at time t is a Nash equilibrium.

Algorithms for computing pure equilibria

Proof (continued).

- By induction assumption tasks $1, \ldots, t-1$ are satisfied at time $t-1$: none of these tasks can improve its cost by a unilateral deviation.
- When task t is inserted, it might be mapped to a machine $j^* \in [m]$ that holds already some other tasks. We only have to show that these tasks do not get unsatisfied because of the increased load on j^*.
- Let $i < t$ be one of the tasks mapped to machine j^*. For $j \in [m]$, let ℓ_j denote the load on machine j at time t. Since the assignment of task t to machine j^* minimizes the cost of agent t and as $w_t \leq w_i$,

$$\frac{\ell_{j^*}}{s_{j^*}} \leq \frac{\ell_j + w_t}{s_j} \leq \frac{\ell_j + w_i}{s_j} \quad \forall j \in [m] \ .$$

Hence, at time t, agent i is satisfied on machine j^*. □

PoA for (mixed) NE on identical machines

Theorem

Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m identical machines. Let $P = (p_i^j)_{i\in[n],j\in[m]}$ denote any Nash equilibrium strategy profile. Then, it holds that

$$cost(P) = O\left(\frac{\log m}{\log\log m}\right) \cdot opt(G) \ .$$

Proof. Recall that

$$cost(P) = E[\max_{j\in[m]}(\ell_j)] \ .$$

We will start with proving an upper bound on the maximum expected load instead of the expected maximum load.

PoA for (mixed) NE on identical machines

Proof (continued). We will show that, for every $j \in [m]$,

$$E[\ell_j] \leq \left(2 - \frac{2}{m+1}\right) \cdot opt(G) \ .$$

The proof for this claim follows the course of the analysis for the PoA for pure equilibria:

- Instead of considering a smallest weight task i^* placed on a maximum load machine j^*, we define i^* to be the smallest weight task with positive probability on a machine j^* maximizing the expected load.
- Also in all other occurrences we consider the expected load instead of the load.

We conclude that the maximum expected load is less than $2opt(G)$.

PoA for (mixed) NE on identical machines

Proof (continued).

- Now we will show that the expected maximum load deviates at most by a factor of $O(\frac{\log m}{\log \log m})$ from the maximum expected load.
- We use a weighted Chernoff bound to show that it is unlikely that there is a machine that deviates by a large factor from its expectation.

Lemma (Weighted Chernoff bound)

Let $X_1, \ldots, X_N$ be independent random variables with values in the interval $[0, z]$ for some $z > 0$, and let $X = \sum_{i=1}^{N} X_i$. Then for any t it holds that $\Pr\{X \geq t\} \leq (e \cdot E[X]/t)^{t/z}$.

PoA for (mixed) NE on identical machines

Proof (continued). Fix $j \in [m]$ and let w be the largest weight of any task. Applying the Chernoff bound shows that, for any t,

$$\Pr\{\ell_j \geq t\} \leq \min\left\{1, \left(\frac{e \cdot E[\ell_I]}{t}\right)^{t/w}\right\} \leq \left(\frac{2e \cdot opt(G)}{t}\right)^{t/opt(G)},$$

since $E[\ell_j] \leq 2opt(G)$ and $w \leq opt(G)$.
Now, if we let $\tau = 2opt(G)\frac{\ln m}{\ln \ln m}$, we get for any $x \geq 0$:

$$\Pr\{\ell_j \geq \tau + x\} \leq m^{-1} \cdot e^{-x/opt(G)}.$$

PoA for (mixed) NE on identical machines

Proof (continued). Recall that for every nonnegative random variable X, $E[X] = \int_0^\infty \Pr\{X \geq t\} dt$. So:

$$cost(P) = E[\max_{j\in[m]} \ell_j] = \int_0^\infty \Pr\left\{\max_{j\in[m]} \ell_j \geq t\right\} dt \ .$$

Substituting t by $\tau + x$ yields

$$\begin{aligned} cost(P) &= \tau + \int_0^\infty \Pr\left\{\max_{j\in[m]} \ell_j \geq \tau + x\right\} dx \\ &\leq \tau + \int_0^\infty \sum_{j\in[m]} \Pr\left\{\ell_j \geq \tau + x\right\} dx \ . \end{aligned}$$

PoA for (mixed) NE on identical machines

Proof (continued). Finally we apply the tail bound derived above and obtain

$$cost(P) \leq \tau + \int_0^\infty e^{-x/opt(G)} dx = \tau + opt(G)$$

which yields the theorem as $\tau = 2opt(G)\frac{\ln m}{\ln \ln m}$. □

PoA for (mixed) NE on uniformly related machines

We come to the most general case of mixed equilibria on uniformly related machines.

Theorem

Consider an instance G of the load balancing game with n tasks of weight $w_1, \ldots, w_n$ and m machines of speed $s_1, \ldots, s_m$. Let P be any Nash equilibrium strategy profile. Then, it holds that

$$cost(P) = O\left(\frac{\log m}{\log\log\log m}\right) \cdot opt(G) \ .$$

I.e., the PoA for this case is only slightly larger than the one for mixed equilibria on identical machines or pure equilibria on uniformly related machines.

PoA for (mixed) NE on uniformly related machines

Sketch of proof. The analysis combines the methods from the cases of mixed equilibria on identical machines and pure equilibria on uniformly related machines.

- First, we show that the maximum expected makespan is bounded by

$$O\left(\frac{\log m}{\log\log m}\right) \cdot opt(G)$$

  using the same kind of arguments as in the analysis of the PoA for PNE on uniformly related machines.
- Then, as in the case of mixed equilibria on identical machines, we use a Chernoff bound to show that the expected maximum load is not much larger than the maximum expected load.
- This results in an upper bound on the price of anarchy of

$$O\left(\frac{\log m}{\log\log\log m}\right) .$$

Summary and discussion

	Identical	Uniformly related
Pure	$2 - \frac{2}{m+1}$	$\Theta\left(\frac{\log m}{\log \log m}\right)$
Mixed	$\Theta\left(\frac{\log m}{\log \log m}\right)$	$\Theta\left(\frac{\log m}{\log \log \log m}\right)$

- Note that all upper bounds on the PoA are tight (examples of matching PoA exist).
- In the case of PNE on identical machines, the PoA is bounded from above by a small constant term.
- In all other cases, the PoA is bounded from above by a slowly growing, sublogarithmic function in the number of machines.

Summary and discussion

Both the PoA for pure equilibria on uniformly related machines and the PoA for mixed equilibria on identical machines are of order $\log m / \log \log m$.

- In the first case, equilibrium assignments correspond to local optima with respect to moves of single tasks. That is, tasks are placed in a suboptimal but nevertheless coordinated fashion.
- In the second case, the increase in cost is due to collisions between uncoordinated random decisions.
- If one combines these two effects, then one loses only another very small factor of order $\log \log m / \log \log \log m$, which results in a PoA of order $\log m / \log \log \log m$ for mixed equilibria on uniformly related machines.

Summary and discussion

How do agents reach a Nash equilibrium?

- Any sequence of improvement steps reaches a PNE after a finite number of steps.
- In case of identical machines the max-weight best-response policy reaches an equilibrium in only $O(n)$.
- In case of uniformly related machines, it is open whether there exists a short sequence of improvement steps that lead from any given assignment to a pure Nash equilibrium.
- Are improvement steps the only reasonable approach for the agents to reach a Nash equilibrium in a distributed way? There might also be other, possibly more strategic or more coordinated behavioral rules that quickly converge to a Nash equilibrium or to an *approximate* Nash equilibrium.

Further reading

- David Easley and Jon Kleinberg: Networks, Crowds, and Markets: Reasoning About a Highly Connected World, **Chapter 8**. Cambridge University Press, 2010.
- Noam Nisan, Tim Roughgarden, Éva Tardos, and Vizay Vazirani (eds): Algorithmic Game Theory, **Chapter 20**. Cambridge University Press, 2007.
- Dimitris Fotakis, Spyros Kontogiannis, and Paul Spirakis: Selfish unsplittable flows. Theoretical Computer Science **348**(2), 2005.