Today we're going to study

Load balancing & routing games.

These are a good starting point because they are fairly simple & they were one of the 1st forays of CS into economics.

We'll warmup with load balancing games before moving to the mainevent – routing games

---

Warmup: Load Balancing Games

These games are motivated by

data centers / server farms / super computers

... these are all situations where there are a huge number of servers & a huge number of jobs to run and somehow the system must assign the jobs to the servers so that performance is good.

The problem is that there are too many jobs/servers to solve this centrally and so each job "decides" it's own assignment without global information typically.

Hence, the jobs are playing a game.

The phrase "load balancing" comes from the fact that you want to balance the assignment of work (a.k.a. load) across the servers.

Here's a simple model of this:

Here's a simple model of this:

we have m servers
& n jobs to run on the servers

→ job i has processing demand p_i (size)
& can be served at servers in S_i ← realistic!

An assignment A has $(i,j) \in A$ if job i is run on server j, & $j \in S_i$

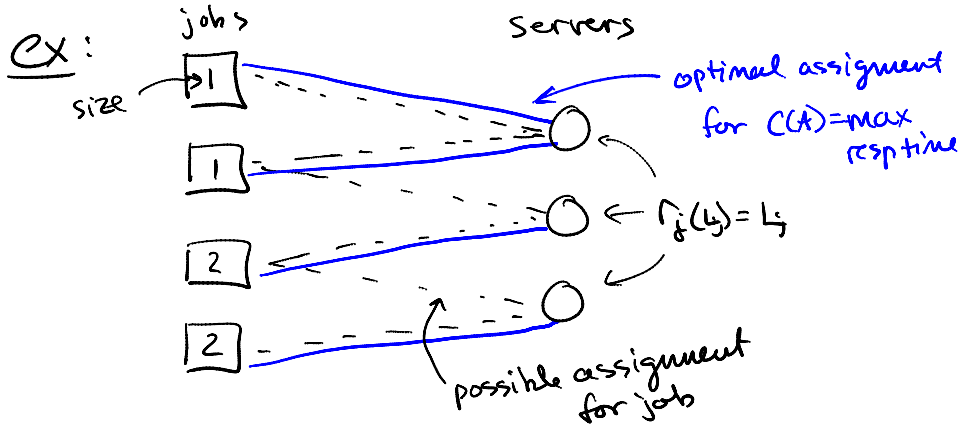
Let $C(A)$ be the "cost" of assigment A
Goal: Find $A^* = \arg\min_A C(A)$

Depending on the setting

- ★ $C(A)$ = max "load" = $\max_j L_j$ where $L_j = \sum_{i:(i,j)\in A} p_i$
- $C(A)$ = average load = $\frac{1}{m}\sum_j L_j$
- ★ $C(A)$ = max "response time" = $\max_j r_j(L_j)$ for inc $r_j(\cdot)$
- $C(A)$ = avg "response time" = $\frac{1}{m}\sum_j r_j(L_j)$

well do this

Ex:



Centralized Approach

Centralized Approach

Given this model, what we want is

$$A^* = \arg\min_A = C(A)$$

⇒ this is NP-hard in general
Can't be found in cases where n & m are large

↓

★ data centers are large, often $n > 1{,}000{,}000$

Realistic Approach

"Greedy" → let each job greedily choose the server that minimizes it's cost

i.e. job i chooses server j^* st $j^* \in S_i$ & $j^* = \arg\min_j r_j(L_j)$

★ So, the jobs are playing a game.

- Players → jobs $= \{1, \ldots, n\}$
- Actions → job i has actions $j \in S_i$
- Payoffs → $r_j(L_j)$ for action $j \in S_i$

deterministic choice of a server → pure strategies

randomized decision → mixed strategies.

we'll focus on pure strategies and leave randomized as a possible thm.

Note: This is a case where we're using game theory as a tool for analyzing an algorithm that doesn't initially sound like a "game".

Q: Is there a dominant strategy for the jobs?

A: No

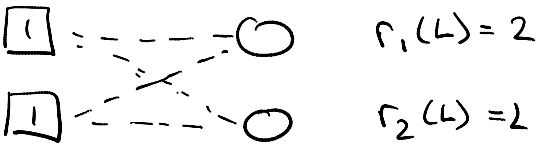
Q: When is an assignment a Nash Equilibrium?

A: If no job can improve cost by changing its assignment

Specifically: A is a nash equilibrium

iff $\forall i, (i,j) \in A \Rightarrow \forall k \neq j, \; r_j(L_j) \leq r_k(L_k + p_i)$

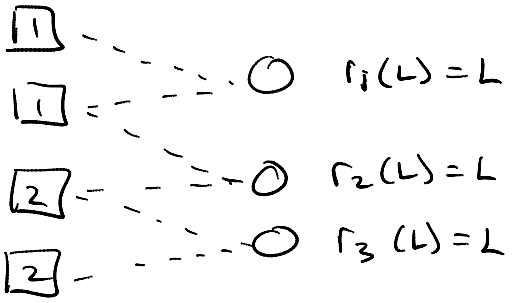
ex:



Q: Which assignments are Nash equilibria?

A: $=$ $\times$ $\rightarrow$

ex:



Q: Is the optimal assignment a Nash Equil.?

A: Yes!

Q: Are there other Nash Equil.?

A: No!

Okay, now we should all be comfortable with the model.

Obviously it's a simplistic view of job assignment in data centers, but it already has enough

features to make analysis interesting and to provide some insight.

Q: What are some critiques of the model?

A: – jobs are assigned in 1 "batch"
– jobs have same p_i at all servers
– why would we arrive at a Nash?

→ To learn about better models take my course next term – 147 !

Q: Why is Nash Equilibrium a relevant concept?

A: Unclear... but maybe
jobs will "expect" others to do best thing and so each job should "expect" others to play the Nash Eq.

... putting aside the critiques though...

What do we want to understand about this model?

we'll focus on these →

1) Does a Nash Equil. always exist?
2) How efficient are the Nash Equil?
(how close to optimal is the resulting cost)
3) How long does it take to find an equilibrium?

These questions will be our focus not just for this model, but for most areas of network economics.

1) Does there always exist an equilibrium?

→ Yes!

In particular, if we start with any assignment and make greedy moves (1 at a time), we'll arrive at an equilibrium

* This is called "Best Response Dynamics"

Q: What's the typical approach for showing that an algorithm like this will terminate?
(... and thus that we'll arrive at an equilibrium)

A: Find "something" that always decreases on each step, thus ensuring that there are no cycles. If there are no cycles then if there are a finite # of assignments the algorithm will eventually terminate

Q: What quantity always improves for a greedy move?

A1: $\max_j r_j(L_j)$ → max response time.

... but this may remain the same, which causes problems.

A2: The sorted vector

$$V = (r_{(1)}(L_{(1)}) \dots r_{(m)}(L_{(m)}))$$

where $r_{(1)}(L_{(1)}) \geq \dots \geq r_{(m)}(L_{(m)})$

we say $V_1 > V_2$ if
V_1 is lexiographically larger than V_2

e.g. $(1,2,3,4,4,5) > (1,2,2,7,8,9)$

To see that a greedy move always decreases the sorted vector, note that if job i goes from $(j) \to (k)$

then 1) $(k) > (j)$

& 2) $r_{(j)}(L_{(j)}) > r_{(k)}(L_{(k)} + p_i)$

so $r_{(1)}(L_{(1)}) \dots r_{(j-1)}(L_{(j-1)})$ don't change

and $r_{(j)}(L_{(j)})$ will change to either

- current $r_{(j)}(L_{(j)} - p_i)$ $\quad < r_{(j)}(L_{(j)})$
- current $r_{(j+1)}(L_{(j+1)})$ $\quad < r_{(j)}(L_{(j)})$
- current $r_{(k)}(L_{(k)} + p_i)$ $\quad < r_{(j)}(L_{(j)})$

$\Rightarrow$ the new vector is lexiographically smaller.

$\downarrow$

So greedy steps will eventually terminate, and when they do, the assignment will be a Nash Equil.

Q: How long will it take to find a Nash Equil this way?

A: It might take exponential time...
...so we haven't made it any more efficient....
:(

Note: In some cases finding an equil. can be done efficiently.

On to question 2:

Q: How "efficient" are nash equilibria?

Before we can answer this we have to define how we'll measure "efficiency"...

how we'll measure "efficiency"...

Let $A^* = \underset{A}{\text{argmax}}\ C(A)$

and let A^{ne} be a nash equilibrium assignment.

We want to understand

$$\frac{C(A^{ne})}{C(A^*)}$$

but there are multiple nash equil., so we have to be more careful.

There are 2 measures people use:

1) Price of Anarchy (PoA) of a game G

$$PoA(G) = \max_{A^{ne}} \frac{C(A^{ne})}{C(A^*)}$$

This PoA of a class of games $\mathcal{H}$ is

$$PoA(\mathcal{H}) = \max_{G \in \mathcal{H}} \max_{A^{ne}} \frac{C(A^{ne})}{C(A^*)}$$

2) Price of Stability (PoS) of a game G.

$$PoS(G) = \min_{A^{ne}} \frac{C(A^{ne})}{C(A^*)}$$

and similarly

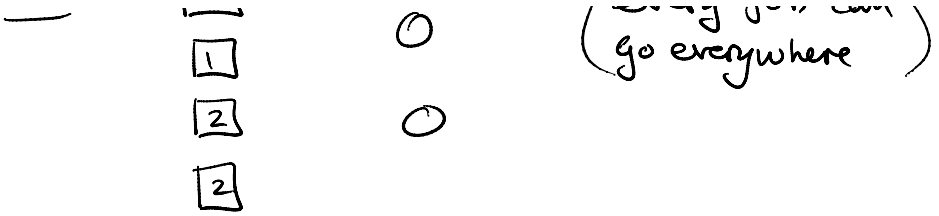
$$PoS(\mathcal{H}) = \max_{G \in \mathcal{H}} \min_{A^{ne}} \frac{C(A^{ne})}{C(A^*)}$$

Q: Any guesses about the PoS or PoA of our load balancing games?

Let's consider only $r_j(L_j) = L_j$ for now...

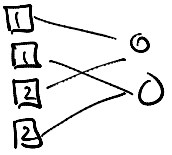
ex: 

O (every job can go everywhere)



Q: What is the optimal assignment?

A:

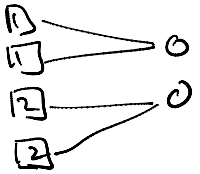


Q: Is this a Nash?

A: Yes! $\Rightarrow$ PoS = 1

Q: Is this the only Nash?

A: No!



is too.

Q: What's the PoA?

A: The two Assignments we've seen are the only Nash equil.

So the 2nd is the worst

$\Rightarrow$ PoA = $\frac{4}{3}$

Q: In this example the PoS was 1, does this hold more generally?

A: Yes! (for all LB games with $C(A) = \max r_i(A_i)$)

Consider A^*. No selfish move can increase $\max r_i(A_i)$, so

$\exists A^{ne}$ st $C(A^{ne}) \leq C(A^*)$, and

$C(A^{ne}) < C(A^*)$ is impossible,

so $C(A^{ne}) = C(A^*)$.

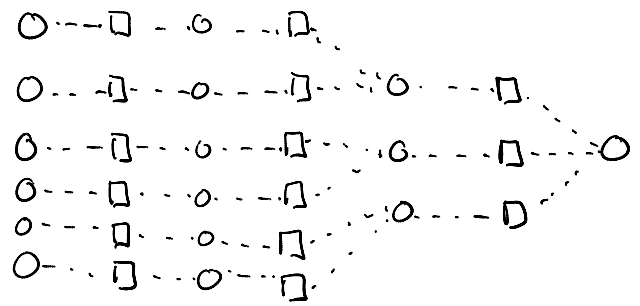
Q: What about the PoA? Can it be worse than 4/3 in other load balancing games?

A: Yes!

One way → make $r_j(\cdot)$ steep
e.g. set $r_j(L_j) = L_j^{\alpha}$ for $\alpha \uparrow \infty$

Even with $r_j(L_j) = L_j$ we can make the PoA bad though:

Here's a way to get a PoA of 3:



All jobs are of size 1 & $r_j(L_j) = L_j$

2 Nash equil: assign left → $C(A) = 1$
assign right → $C(A) = 3$

Clearly this can be generalized easily to get PoA of any integer value.

But: this took advantage of S_i not being equal.

Q: What if $S_i = s$ $\forall i$? How large can the PoA be now?
(keep $r_j(L_j) = L_j$)

Thm: If $r_j(L_j) = L_j$ & $S_i = s$ $\forall i$, then
$PoA \leq 2 - \frac{2}{m+1}$

So, in this case, all nash equilibria are nearly optimal.

nearly optimal.

It turns out that PoAs are surprisingly low in many cases... which is why game theory based algorithms have become popular.

Pf: Consider the worst nash A^{ne}
Let server j^* have the highest load
and job i^* have the smallest size.
among jobs at j^*.

WLOG: Assume ≥ 2 jobs at j^*
(else PoA = 1)

Claim 1: $p_{i^*} \leq \frac{1}{2} C(A^{ne})$

Claim 2: $\forall j \neq j^*, L_j \geq L_{j^*} - p_{i^*} \geq \frac{1}{2} C(A^{ne})$
(by def of nash equil.)

Now, let's use these to bound $C(A^*)$

$$C(A^*) \geq \frac{\sum p_i}{m} = \frac{\sum L_i}{m} \geq \frac{C(A^{ne}) + \frac{1}{2} C(A^{ne})(m-1)}{m}$$

- $C(A^*) \geq$: max $\geq$ avg
- $\frac{\sum p_i}{m} = \frac{\sum L_i}{m}$: sum of job sizes in the same
- $\geq$: above claims
- $C(A^{ne})$: server j^*
- $\frac{1}{2} C(A^{ne})(m-1)$: other $j \neq j^*$

$$\Rightarrow C(A^{ne}) \leq \frac{m C(A^*)}{1 + \frac{1}{2}(m-1)}$$

$$= \frac{2m}{m+1} C(A^*)$$

$$= C(A^*)\left(2 - \frac{2}{m+1}\right)$$

□

Q: Is this tight?

A: A possible HW question...

Okay, that's enough on load balancing games.

We've seen that

- Nash equilibria always exist
- PoS = 1
- PoA can be large, but in some cases ($S_i = S$ & $r_j(L_j) = L_j$) it is very small.

Now on to the main event: routing games

Topic 1: Routing Games aka Congestion Games

Now let's move on to the main event: routing games.

Here the idea is that sources must choose their own routes through a network, and they do this to minimize their own latency.

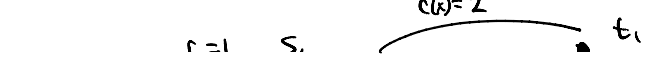
→ How much inefficiency does this create?
(compared to if they all decided together how to route)

Non-CS example: Maybe an even better example of this is driving to work

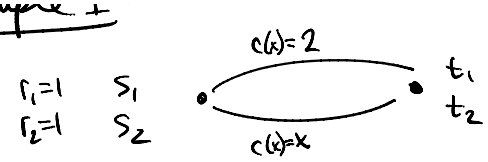
- every day people choose a route to drive to work without knowledge of what others will pick, but how long it takes depends on the choices of others

---

Example 1:



Example 1



Nash → Both on bottom : Avg delay = 2
or 1 on top, 1 on bottom : Avg delay = 3/2

OPT → 1 on top 1 on Bottom : Avg delay = 3/2

Defining the model

Graph = (V, E, C)
↑ each edge has a cost/latency function $c_e(x)$ that depends on the traffic x through the link. $c_e(\cdot)$ is non-neg, increasing, continuous.

Players: Each player has (s_i, t_i, r_i)
↑ source ↑ destination ↑ amt of traffic

Actions: send r_i on a particular path from $s_i \to t_i$.
(pure strategy again)

Payoff: Total latency of traffic:
latency of sending x on path p is

$$C_p(x) = \sum_{e \in P} c_e(f_e)$$

↑ flow on edge e (depends on other players)

Global Goal:

min avg latency

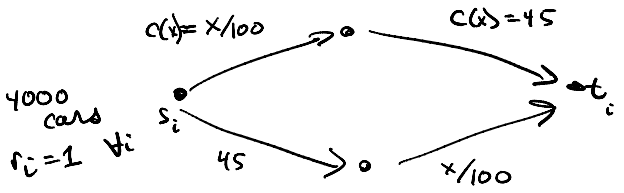
$$\Rightarrow \text{Cost} = \sum_p C_p(f_p) f_p$$

↑ cost of path p ↑ flow on path p

$$= \sum_{e \in E} c_e'(f_e) f_e$$

(like before we could choose many other goals here.)

Example 2:



Q: What is the equilibrium?

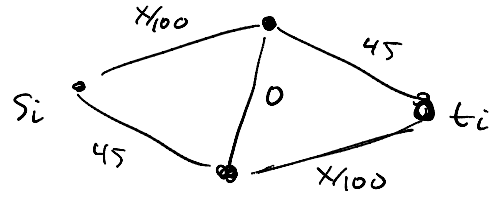
A: 2000 take top
2000 take the bottom

$$\text{cost} = \frac{2000}{100} + 45 = 65$$

Q: Is this the optimal?

A: Yes.

Suppose we now add a freeway...



Q: What is the equilibrium now?

A:

$$\text{Cost} = \frac{4000}{100} + \frac{4000}{100} = 80$$

$\Rightarrow$ The cost went up!

$(PoS = 80/65 > 1!)$

... can make $PoS \geq 4/3$ with this example.

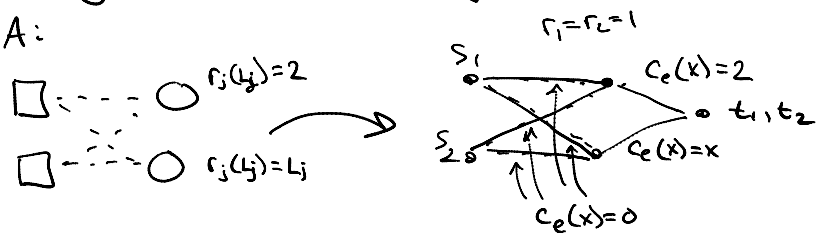
This is called Braess' Paradox

It is surprising that added capacity can hurt... but this happens frequently in real life (see ppt)

Example 8:

Q: How can we turn a load balancing game into a routing game.

A:



As with LB games, we'll discuss two things

1) Existence of Nash Equil
2) Efficiency of Nash Equil

---

1)

We'll start by discussing the efficiency of the Nash Equilibrium for routing games...

We saw that in load balancing game, the PoA is unbounded in general, but is small (<2) when $r_j(L_j) = L_j$ & $S_i = S$.

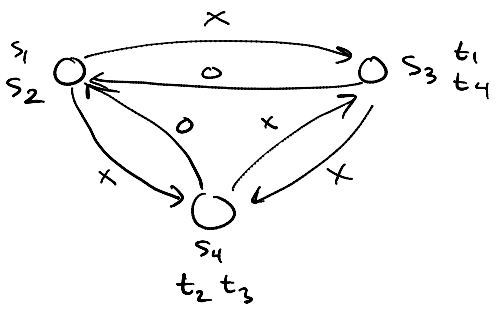
Again it is easy to see that its bad in general

in general

But, what if $c_e(x) = ax+b$?

Q: Can you give me an example w/ PoA > 2?

A:



Q: What is the opt way to route traffic?

A: the 1 hop paths cost = 4

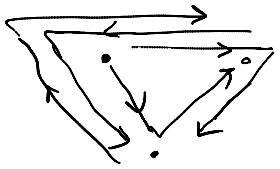
Q: Is this an equilibrium?

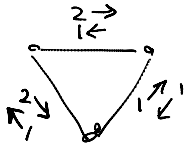
A: Yes

Q: Are there other equilibria?

A: Yes!

the 2 hop paths





cost = 10

$PoA = \frac{10}{4}$

★ This turns out to be almost the worst case!

Thm If all players have $r_i = r$ and $c_e(x) = ax+b$ then $PoA \leq 2.5$

Thm If $c_e(x) \neq ax+b$ then

$$PoA \leq \phi^2 = \left(\frac{1+\sqrt{5}}{2}\right)^2 \approx 2.618$$

↑ golden ratio.

I can't resist making sure you all know about the golden ratio?

$$\phi = \frac{1}{1+\frac{1}{1+\frac{1}{1+\cdots}}} = \sqrt{1+\sqrt{1+\sqrt{1+\cdots}}}$$

$$= \frac{1+\sqrt{5}}{2} = \lim \frac{F(n+1)}{F(n)}$$

← $n+1^{st}$ Fibonacci #

See ppt

★ These are telling us (amazingly) that the network structure doesn't matter much for the efficiency of equilibria!

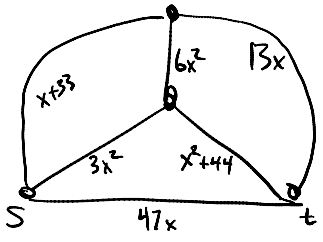
---

2) Existence of equilibria

Q: Do you think that there always exists an equilibria in these routing games?

A: No!

Here's an example w/ no equilibria



$r_1 = 1$
$r_2 = 2$ both have same s & t

To see that there is no equilibria:

suppose 2 takes ⌒
then 1 takes ⌒
then 2 takes ⌒
then 1 takes —
then 2 takes ⌒

I'll leave the details to you to check

... I'll leave the details to you to check...

Q: So when does an equilibrium exist?

A: 1) if all flows have the same r_i
2) if all edges have $C_e(x) = a_e x + b_e$

To prove this we will again use a variant/potential function.

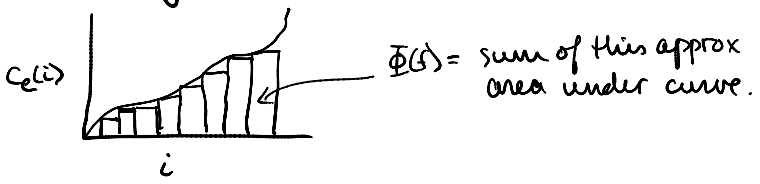
★ Let's start w/ case of $r_i = 1 \ \forall i$ (WLOG)

In particular, let's choose the variant func

$$\Phi(f) = \sum_{e \in E} \sum_{i=1}^{f_e} C_e(i)$$

($\Phi(f)$: total flow through graph)

This may seem strange, but notice that



To see why this is a good variant func, let's consider the change to $\Phi(f)$ when one flow in f switches its route from $P \to P'$. Call the resulting flow f'

network flow: $f \to f'$
flow i: $P \to P'$

$$\Phi(f') - \Phi(f) = \sum_{e \in P'/P} C_e(f_e + 1) - \sum_{e \in P/P'} C_e(f_e)$$

$$= C_{P'}(f') - C_P(f)$$

Change in Φ = change in flow payoff

Q: what does this property give us?

... (cost ⇒ pot func

A: Greedy move ↓ cost ⇒ ↓ pot fnc

So, Any local minima of Φ is a Nash Equilibrium since no player can make a move to ↓ Φ

⇒ no player can improve payoff by deviating.

This is such an important property, it's given a name:

def A game is called a potential game if ∃ a potential function Φ st

$\forall a, a' : \Phi(a') - \Phi(a) = u_i(a') - u_i(a)$

$u_i(x)$ is utility of player i (who changed move between a & a') for action profile x

In any potential game

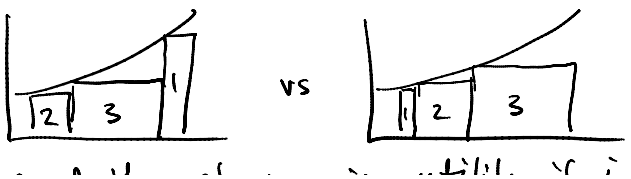
1) ∃ an equilibrium (the minima of Φ)
2) Best response dynamics will converge to an equilibrium.

Okay, back to routing games...

the fact that they are a potential game (when $r_i = 1$) means that an equilibrium exists & that Best response dynamics will converge.

Q: What if r_i can be different? Why doesn't Φ work anymore?

A: The order in the sum would now matter...



... and the change in utility if i switches route always corresponds to having i last in the sum of Φ.

✯ Note that this is not a problem if $c_e(x)$ is linear
$\Rightarrow \Phi$ is still a potential function in this case.

Another use for the potential function:

We know that the minimizer of Φ, a^{ne} is a NE ... so if we can characterize the welfare of a^{ne}, we can bound the PoS.

Idea: If Φ is well "aligned" with the global cost function, then the PoS is small.

Thm: If $\forall a$, $\Phi(a) \le C(a) \le \alpha \Phi(a)$, then $PoS \le \alpha$.

Pf: Let $a^{ne} = \operatorname{argmin}_a \Phi(a)$

Then
$$C(a^{ne}) \le \alpha \Phi(a^{ne}) \le \alpha \Phi(a^*) \le \alpha C(a^*)$$
(a^*: optimizer of $C(\cdot)$)

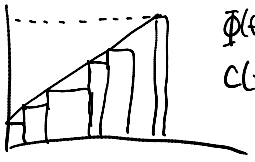
$\Rightarrow PoS \le \alpha$.

□

Q: How can we apply this to routing games?

A: Consider $c(x) = ax + b$. ($b \ge 0$ since $c(0) \ge 0$ $\forall x$)

Then



$\Phi(f)$ = area under solid line
$C(f)$ = area under dotted line $(f_e c_e(f_e))$

$$\frac{1}{2} C(f) \le \Phi(f) \le C(f)$$

$\frac{1}{2}C(f) \leq \Phi(f) \leq C(f)$

Okay, we've covered the basics of routing games

We've seen

1) When $\exists$ an equilibrium
2) How bad equilibria can be

The conclusion is that in simple cases equilibria will exist and be fairly efficient, but in general equilibria can be very inefficient.

So, we want to do something to make the network more efficient.

(e.g. reduce traffic in our road network example)

Q: What can we do?

A1: Add capacity... but Brayes Paradox points out that this needs to be done carefully.

A2: Add tolls, but how?

Consider the case of $r_i = 1$ $\forall i$.

Key idea of tolls →

"cost" of a flow should be not only latency, but also should include the added cost to others being slowed

Cost to others being shared with.

★ Principle of "marginal cost pricing"

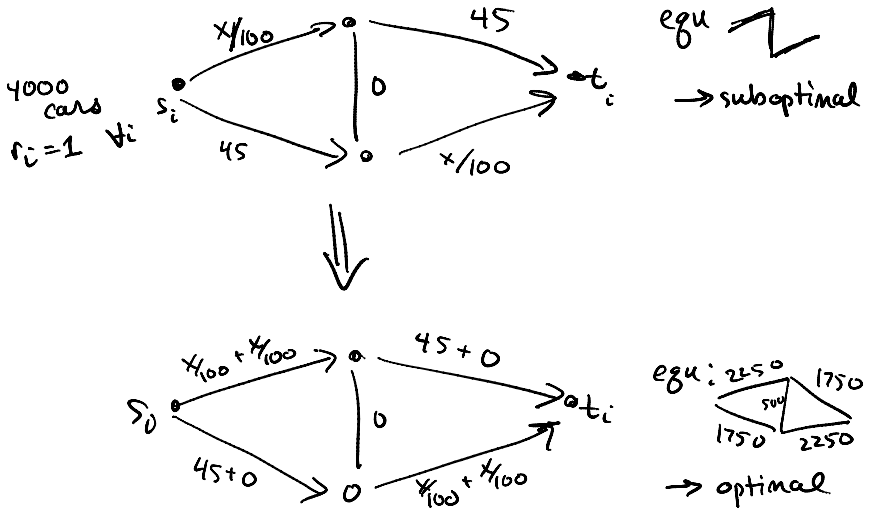
$c_e(x) \longrightarrow c_e^*(x) = c_e(x) + (x-1)\big(c_e(x) - c_e(x-1)\big)$

($c_e(x)$: your latency; $(x-1)\big(c_e(x) - c_e(x-1)\big)$: added latency to others being shared with.)

Thm: Marginal cost pricing guarantees that the PoS = 1.

pf: on HW!

ex: Braess' Paradox.



That's it for routing games (for us).

As always, there's much more we could have covered...

Important things to take away:

- game theory as algorithm design / analysis tool
- price of anarchy / price of stability
- potential functions / games
- conflict between local choices and global goal.
- principle of marginal cost pricing to align local choices with global goal.

Thanks!