Co-authors of Mine

Shahar Moses
Esteban Feuerstein
Peter Sanders
Baruch Awerbuch
Benny Pinkas
Alan Roytman
Ilia Gorelik
N. S. Narayanaswamy
Uli Wagner
Leah Epstein
Dotan Emanuel
Erik D. Demaine
Orli Waarts
Rafail Ostrovsky
Elias Koutsoupias
Frank McSherry
Shiri Chechik
Dmitry Pechyony
Dan Feldman
Howard J. Karloff
Ke Chen
Slava Novgorodov
Danny Segev
Dean P. Foster
Rotem Zach
Manor Mendel
Bogdan Cautis
Sandy Irani
Alon Eden
Peiyuan Yan
Andrew V. Goldberg
G. Turpin
Shaddin Dughmi
Alan Siegel
Claire Mathieu
Yishay Mansour
Jeanette P. Schmidt
Steven S. Seiden
Elchanan Mossel
Adi Rosén
Gagan Aggarwal
Angelina Vidali
Meital Levy
Nicole Immorlica
Katrina Ligett
Avrim Blum
Ziv Rosen
Kobbi Nissim
Tova Milo
Miroslaw Kutylowski
Moty Ricklin
Ron Berman
Liam Roditty
Lukasz Jez
Stefano Leonardi
Benny Chor
Sundar Vishwanathan
Frank Thomson Leighton
Vincent Cohen-Addad
Ronen Shabo
Sarel Cohen
Michal Feldman
Mariano Schain
Baruch Schieber
Ilan Reuven Cohen
Renato F. Werneck
Maxwell Young
Rakesh V. Vohra
Haim Kaplan
Manuel Blum
Christos H. Papadimitriou
Svetlana Olonetsky
Lyle A. McGeoch
Noga Alon
Alejandro A. Schäffer
Daniel Dominic Sleator
Richard M. Karp
Harald Räcke
Ariel Levavi
Shakhar Smorodinsky
Serge Abiteboul
Yiftach Ravid
Itai Abraham
Marcin Gomulkiewicz
Anna R. Karlin
Dimitris Achlioptas
Neal E. Young
Jared Saia
J. Ian Munro
Yuval Rabani
Uriel Feige
Jason D. Hartline
Hila Pochter
Edith Cohen
Ran El-Yaniv
Gerhard J. Woeginger
Steven D. Gribble
Piotr Berman
Iddan Golomb
Kira Goldner
Sampath Kannan
James Aspnes
Madhu Sudan
Gábor Tardos
Daniel Delling
Stefan Saroiu
Sagi Shporer
Adi Shamir
Amiram Wingarten
Moni Naor
Marek Klonowski
János Pach
Yossi Azar
Ehud Y. Shapiro
Uri Nadav
Emo Welzl
Piotr Sankowski
Michael E. Saks
Amnon Ta-Shma
Moshe Hershcovitch
Svetlana Obraztsova
David Chaum
Jirí Matousek
Michael Luby
Ilan Shimshoni
Yair Bartal
Tomer Levinboim
Eldad Bar-Eli
Micha Sharir
Serge A. Plotkin

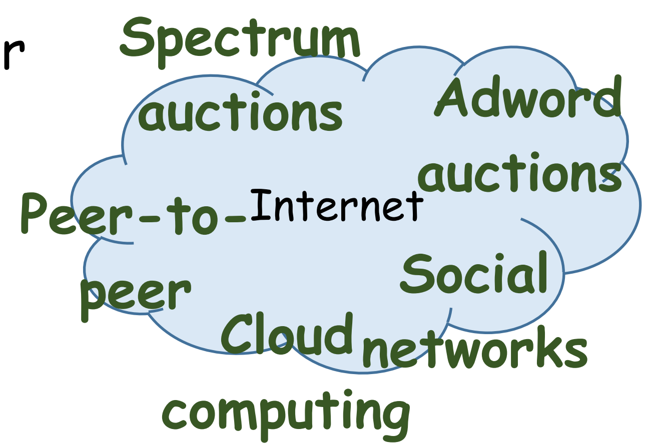
My self image: Not as I look today

Algorithmic mechanism design

70's | late 90's | 2000+

Mechanism Design

Design of protocols for aligning incentives of strategic agents (e.g., pricing, auctions, …)



Algorithmic Mechanism Design

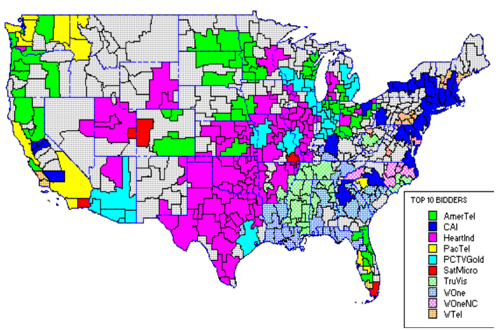
New domains of problems

New orientations toward solutions

New opportunities

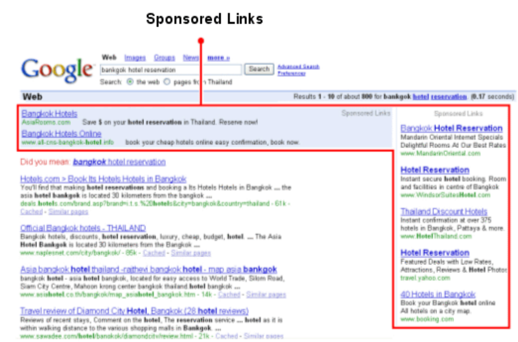
Complex settings

Spectrum auctions



Cloud computing

Adword auctions



First decade of algorithmic mechanism design: truthful mechanisms (complex auctions)

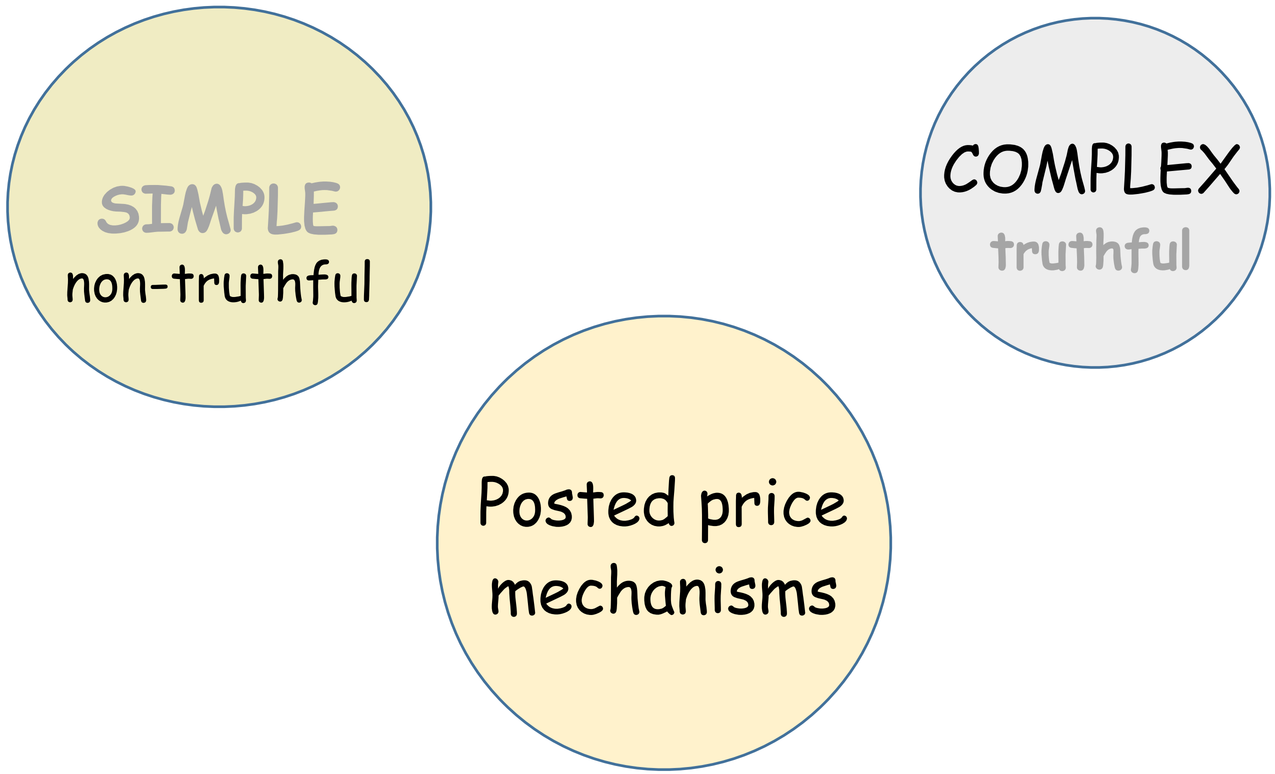
Simple mechanisms

Recently, focus on simple, non-truthful mechanisms

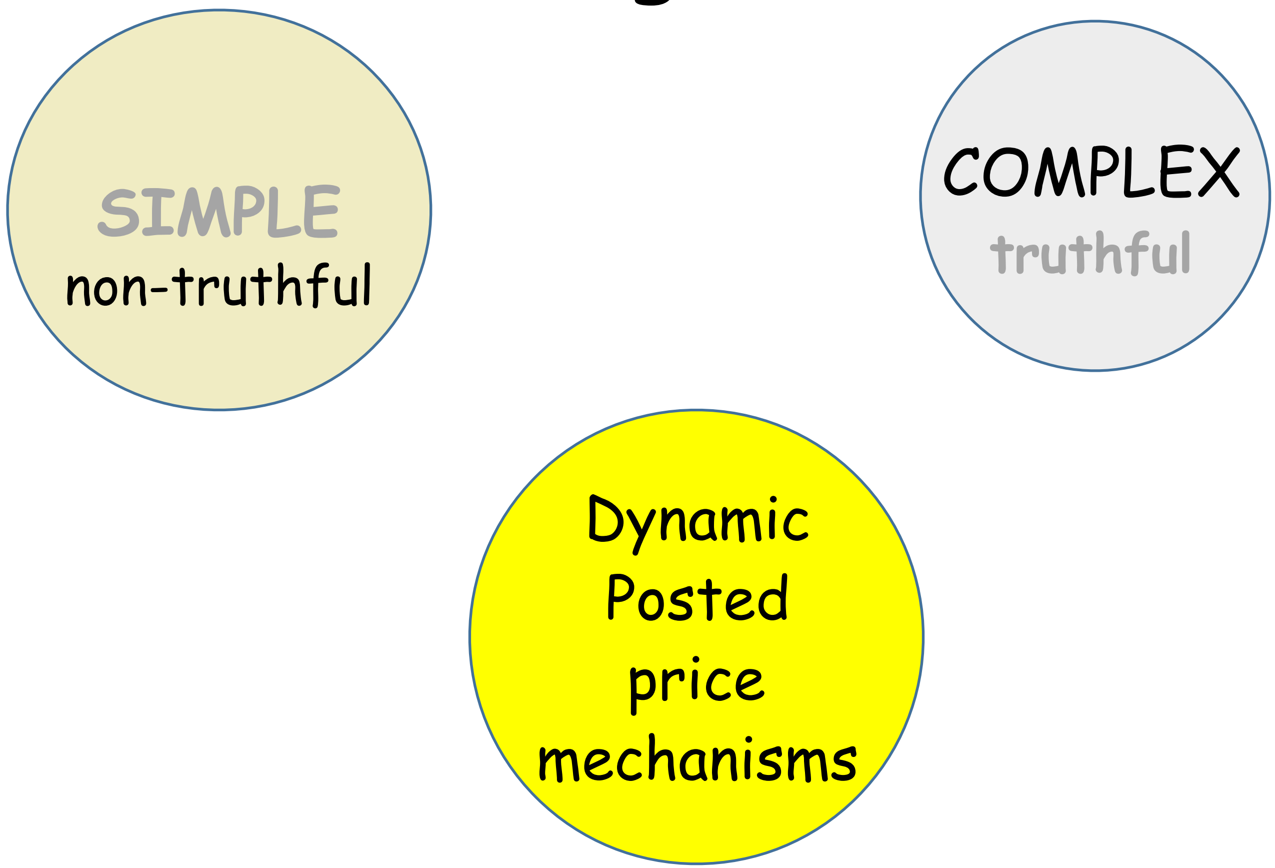
- generalized second price auctions [Edelman Ostrovsky Schwarz 05, Varian 07, Lucier Paes Leme 10, 11, Lucier Paes Leme Tardod 12 ...]
- simultaneous item auctions [Christodoulou Kovacs Schapira 08, Bhawalkar Roughgarden 12, Feldman Fu Gravin Lucier 13, Hassidim Kaplan Mansour Nisan 11, ...]

Evaluated at equilibrium (price of anarchy [Koutsupias Papadimitriou '99])

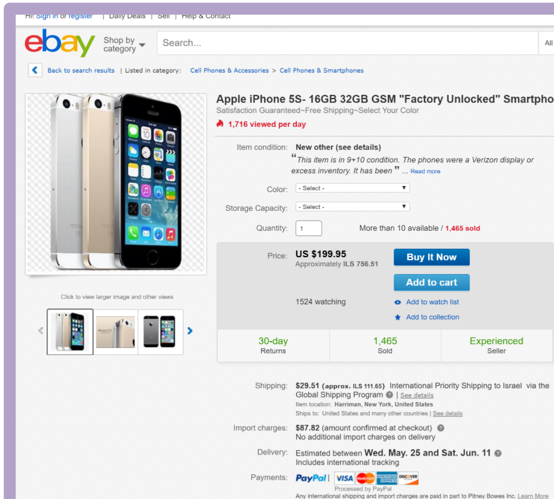
Simple and truthful mechanisms

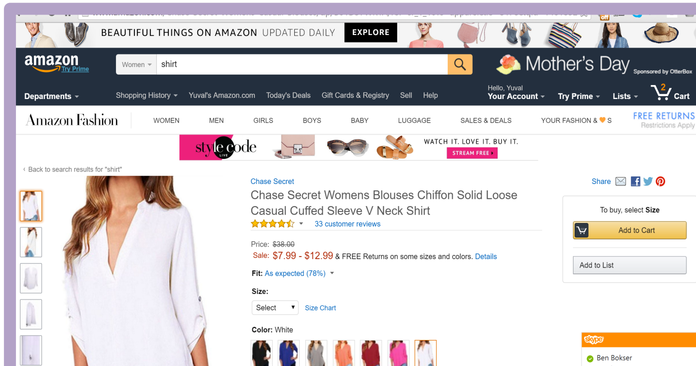


Simple and truthful mechanisms for Online Settings

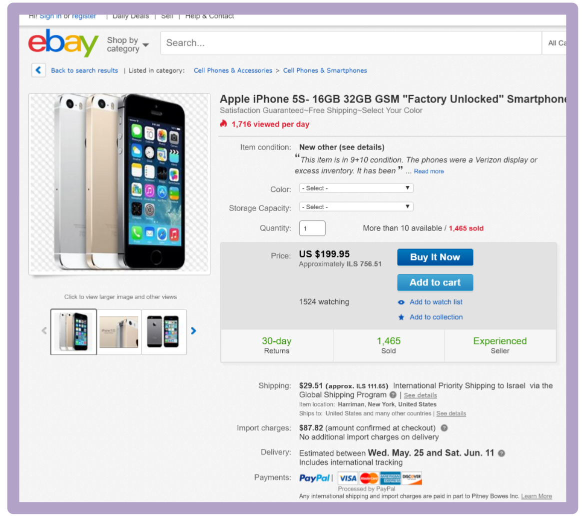


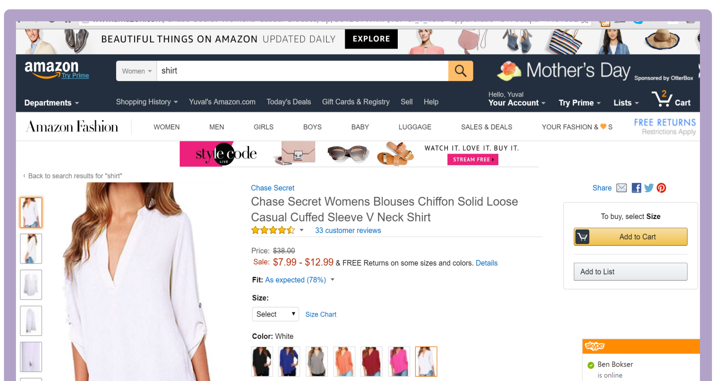
Posted prices



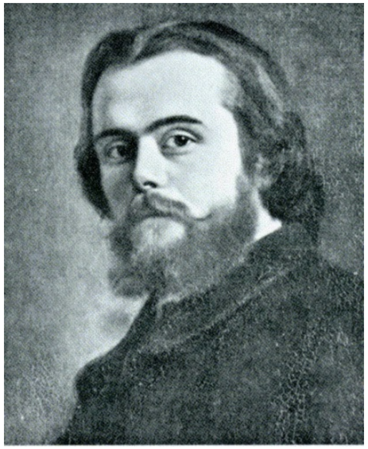


Dynamic Posted prices



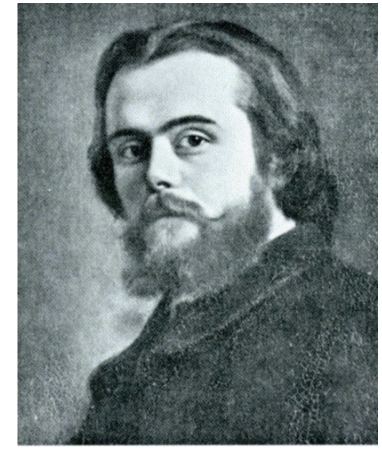


Walrasian equilibrium

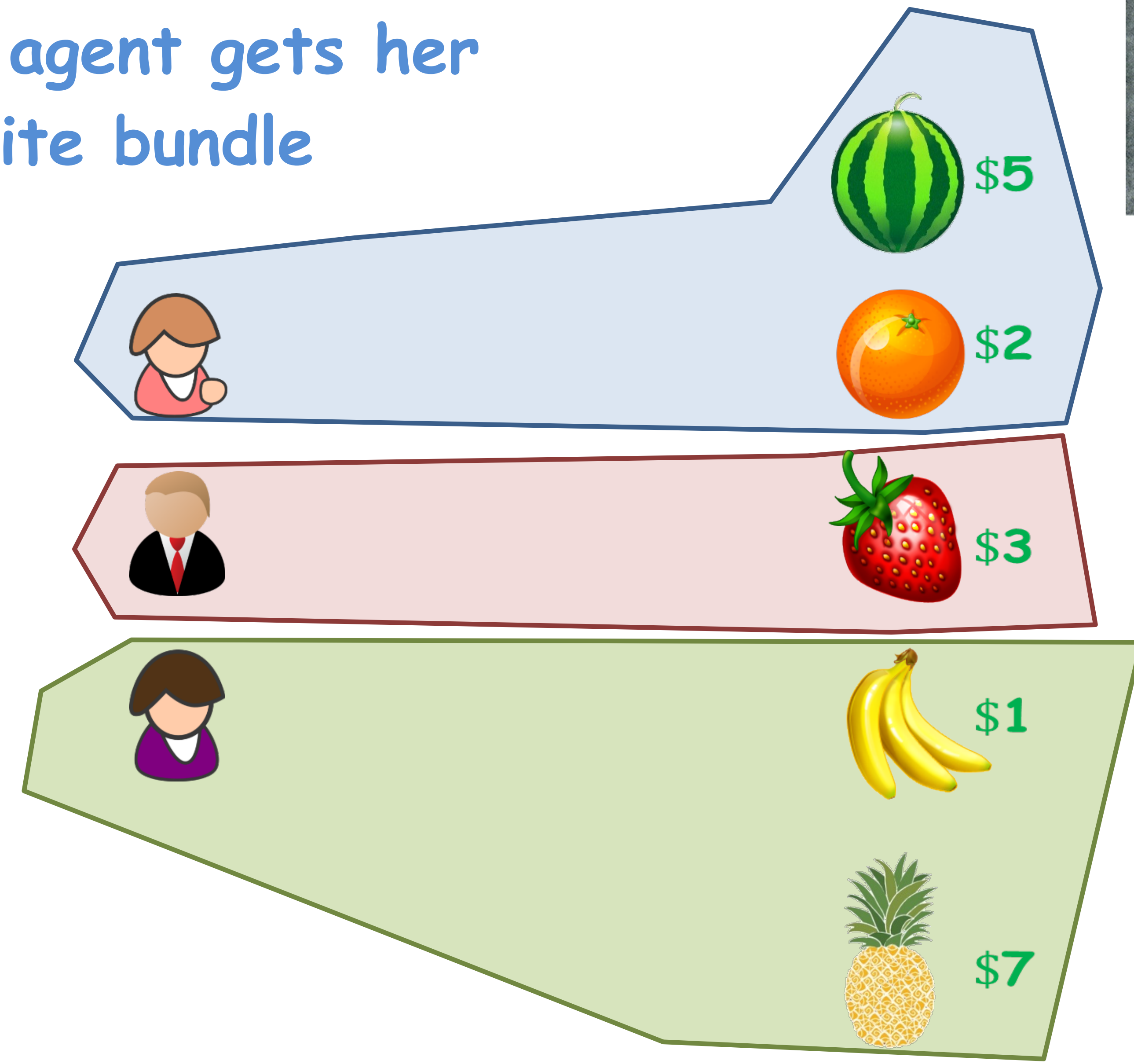


Prices are assigned to items

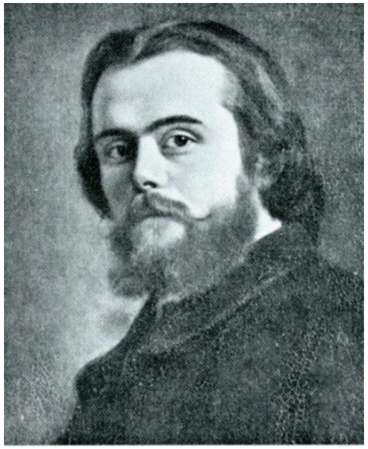
Walrasian equilibrium



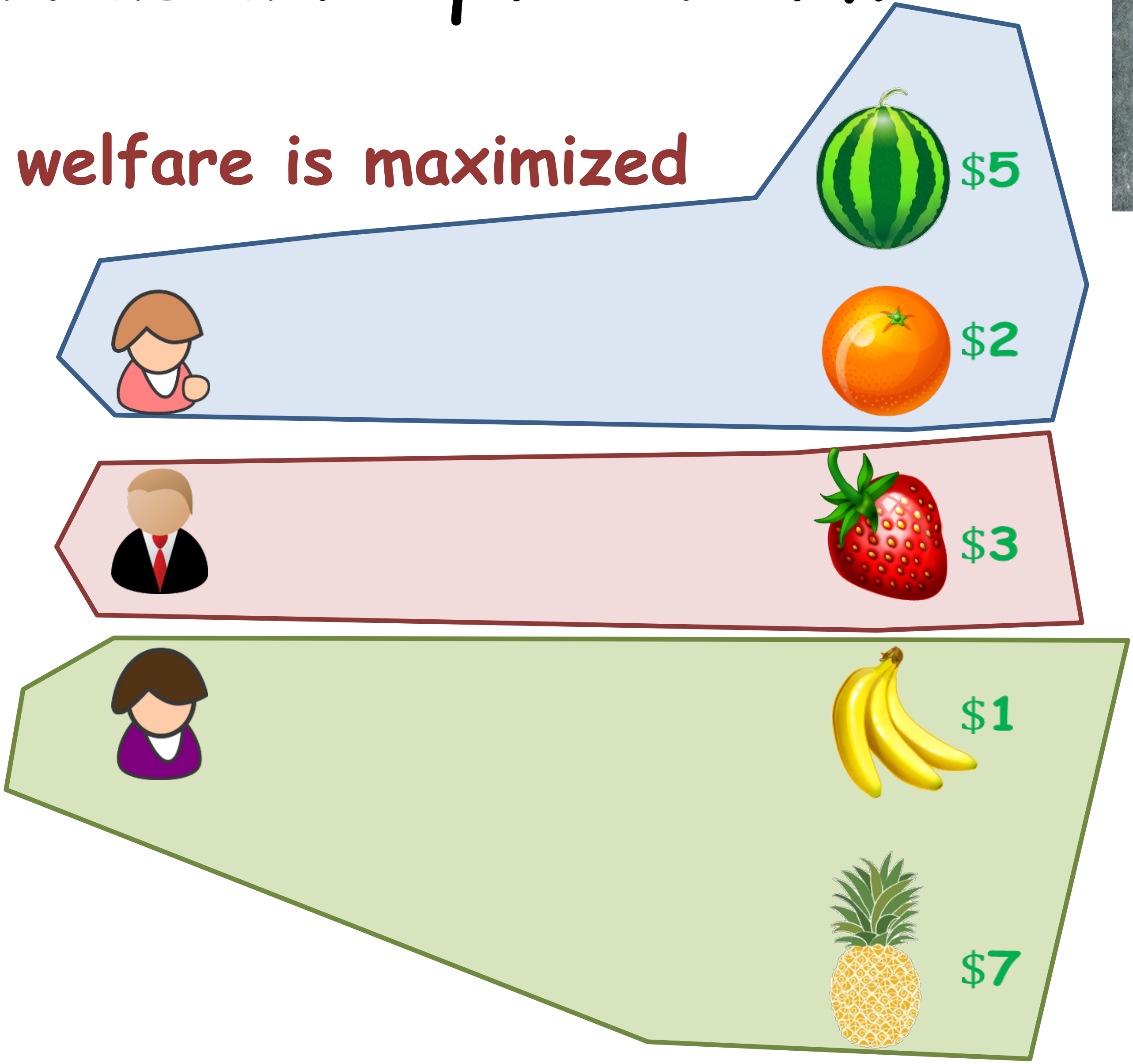
Each agent gets her favorite bundle



Walrasian equilibrium



Social welfare is maximized



Prices

Maximize **profit**: Seller sets prices to maximize her **revenue** subject to demand

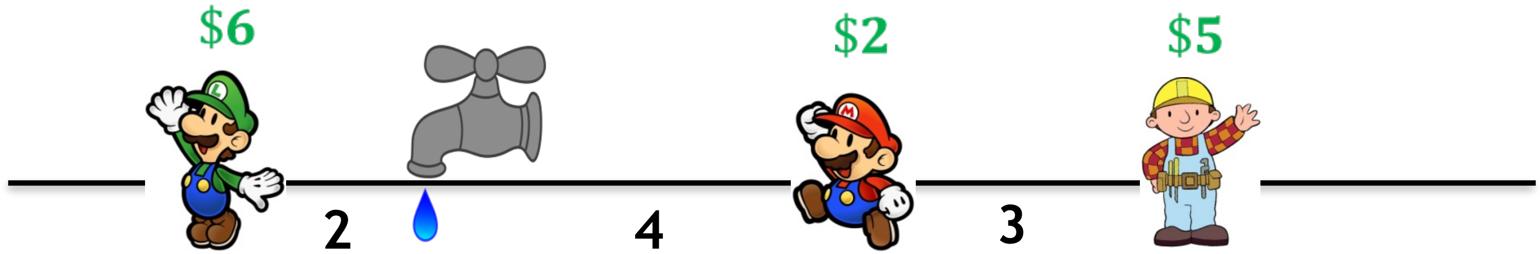
Prices can be used to benefit society

Sin taxes ... ed harmful t ... .).

Carbon t... ewable fuels.

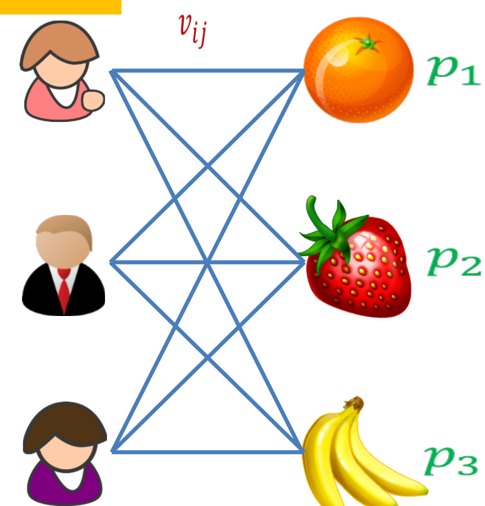
Subsidies ... it society: **t** ... **subsidy**, etc.

Many oth ... vian taxes).



Service calls

Parking



Markets

Queue Control

UBER

Selfish Agents over Time

Setting: Events are generated (or can be manipulated) by selfish agents

Agents only act in their own self interest with no "moral" reasoning

In particular, agents may lie outrageously

Agents will not work against their own self interest – but - they cannot be counted upon to do anything for the greater "good" even if it does

Lots and Lots (and Lots) of Open Problems

Papers

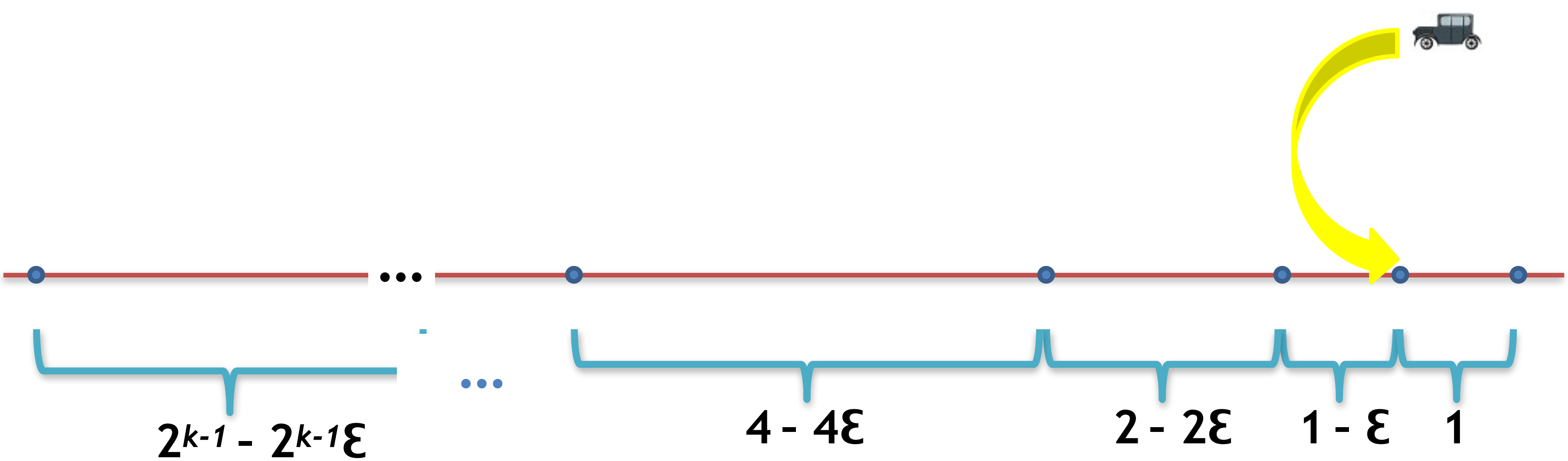
- **Pricing Online Decisions: Beyond Auctions, SODA '15**
  - Cohen, Eden, Fiat, Jez
- **The Invisible Hand of Dynamic Market Pricing, EC '16**
  - Cohen-Addad, Eden, Feldman, Fiat
- **Makespan Minimization via Posted Prices**, EC17
  - Feldman, Fiat, Roytman
- Lottery Pricing Equilibria,
  - Shaddin Dughmi, Alon Eden, Michal Feldman, Amos Fiat, Stefano Leonardi
- Other ongoing work
- **Quarter century ago (???): On-line load balancing with applications to machine scheduling and virtual circuit routing. STOC '93**
  - Aspnes, Azar, Fiat, Plotkin, Waarts

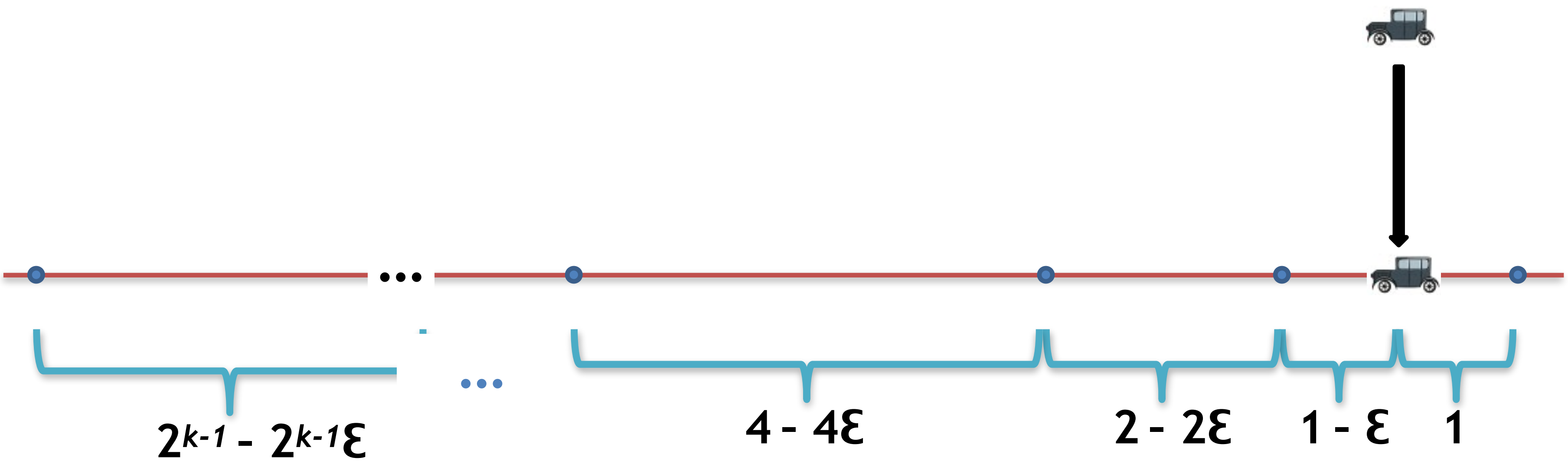
“Killer” Motivation: Parking

Social Cost = Sum of distances from parked cars to destinations

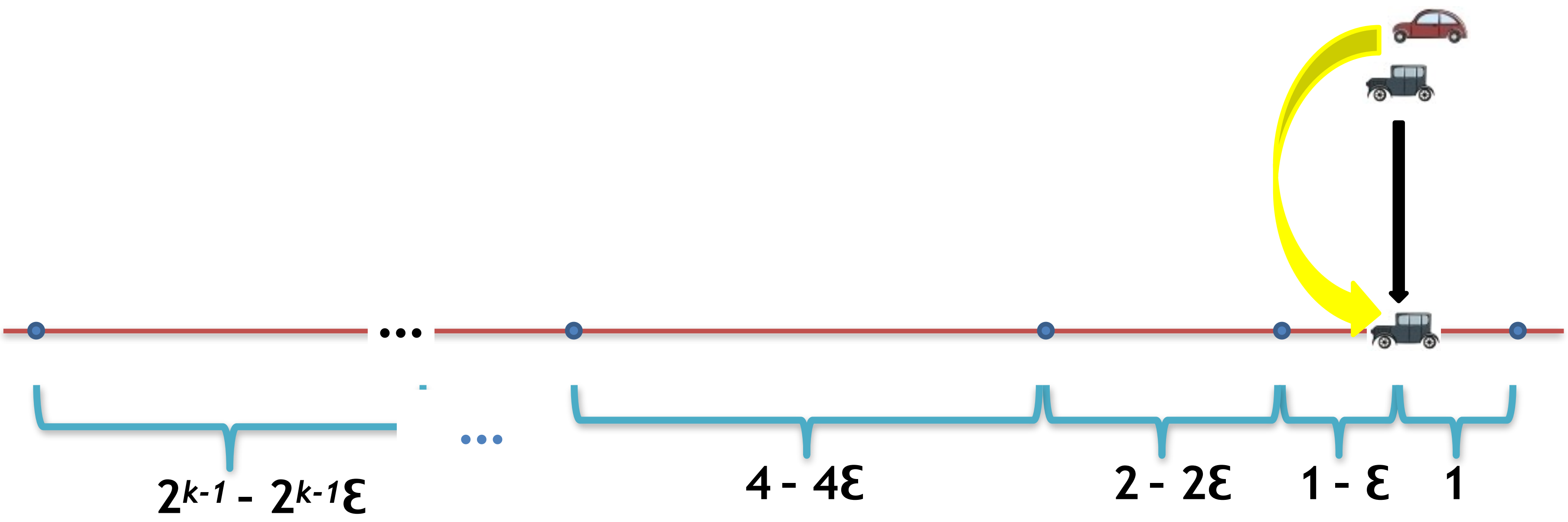
The High Cost of Free Parking

Kalyanasundaram, Pruhs; Khuller, Mitchell, Vazirani

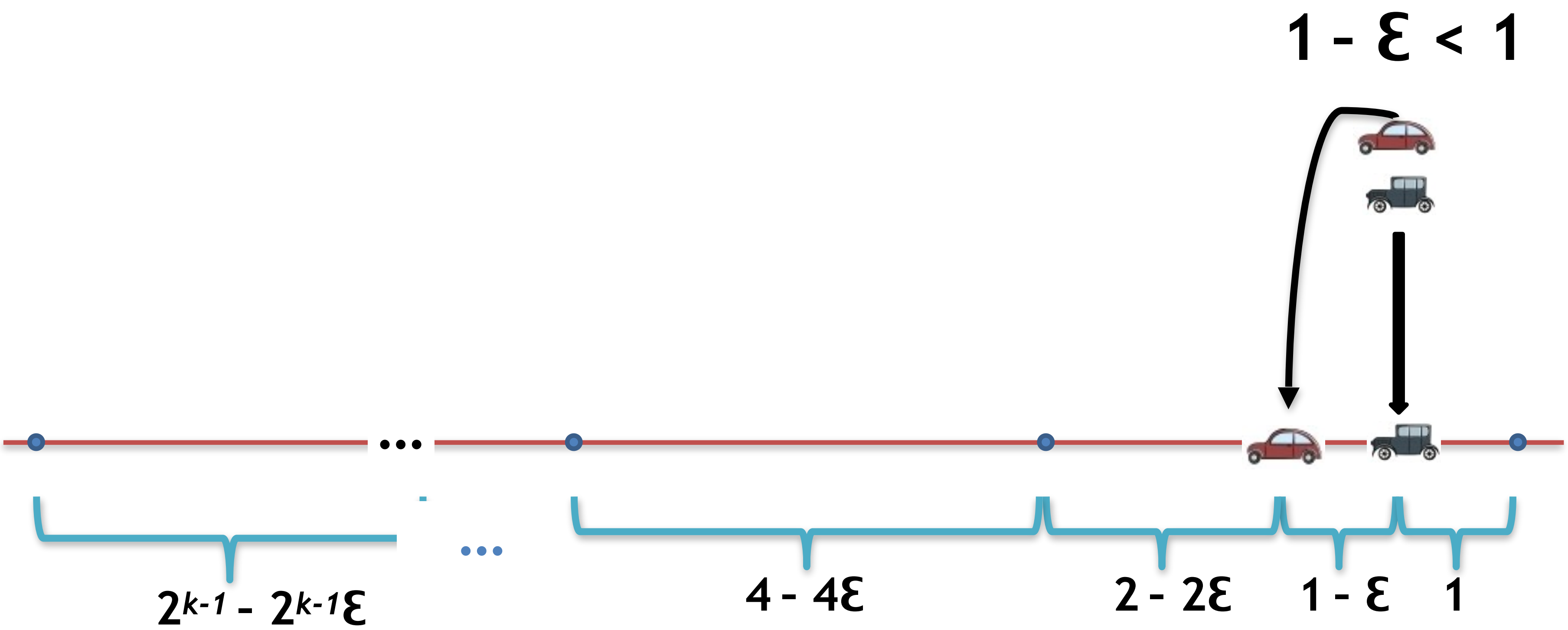




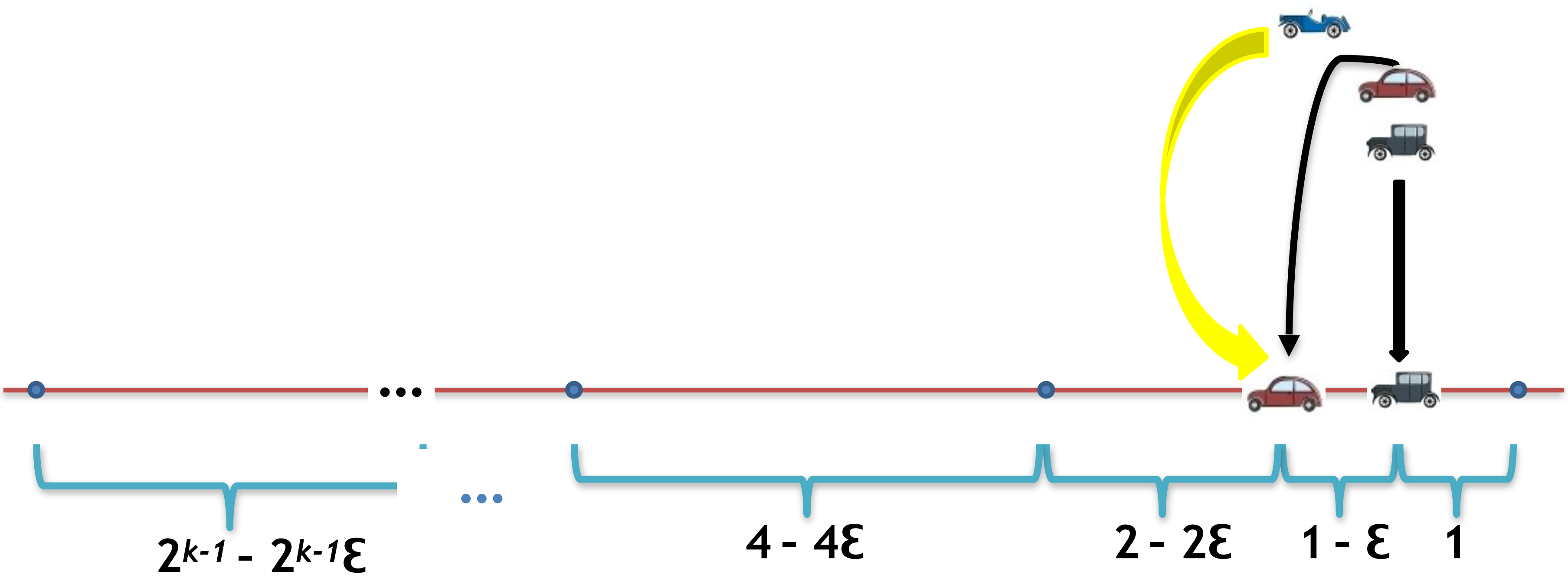
$2^{k-1} - 2^{k-1}\varepsilon$
...
$4 - 4\varepsilon$
$2 - 2\varepsilon$
$1 - \varepsilon$
1



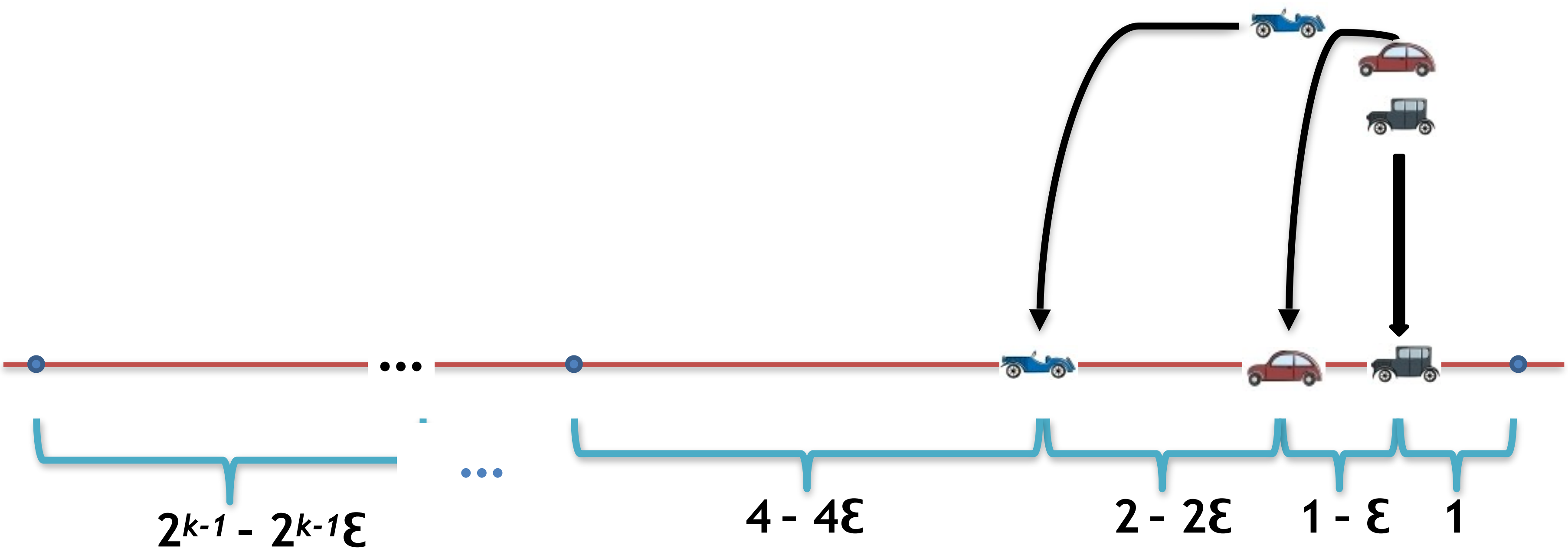
•••
•••
$2^{k-1} - 2^{k-1}\mathcal{E}$
4 - 4Ɛ
2 - 2Ɛ
1 - Ɛ
1

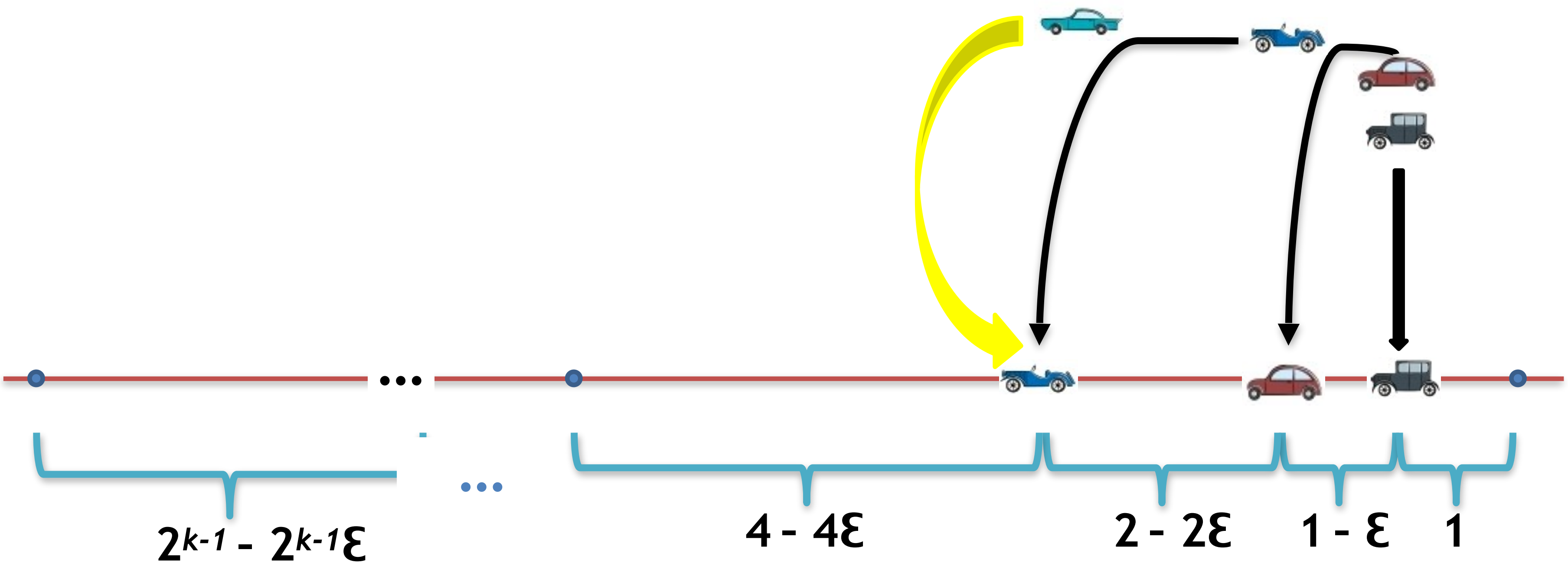


1 - ε < 1
2^k-1 - 2^k-1ε
...
4 - 4ε
2 - 2ε
1 - ε
1


$2^{k-1} - 2^{k-1}\varepsilon$
…
$4 - 4\varepsilon$
$2 - 2\varepsilon$
$1 - \varepsilon$
1

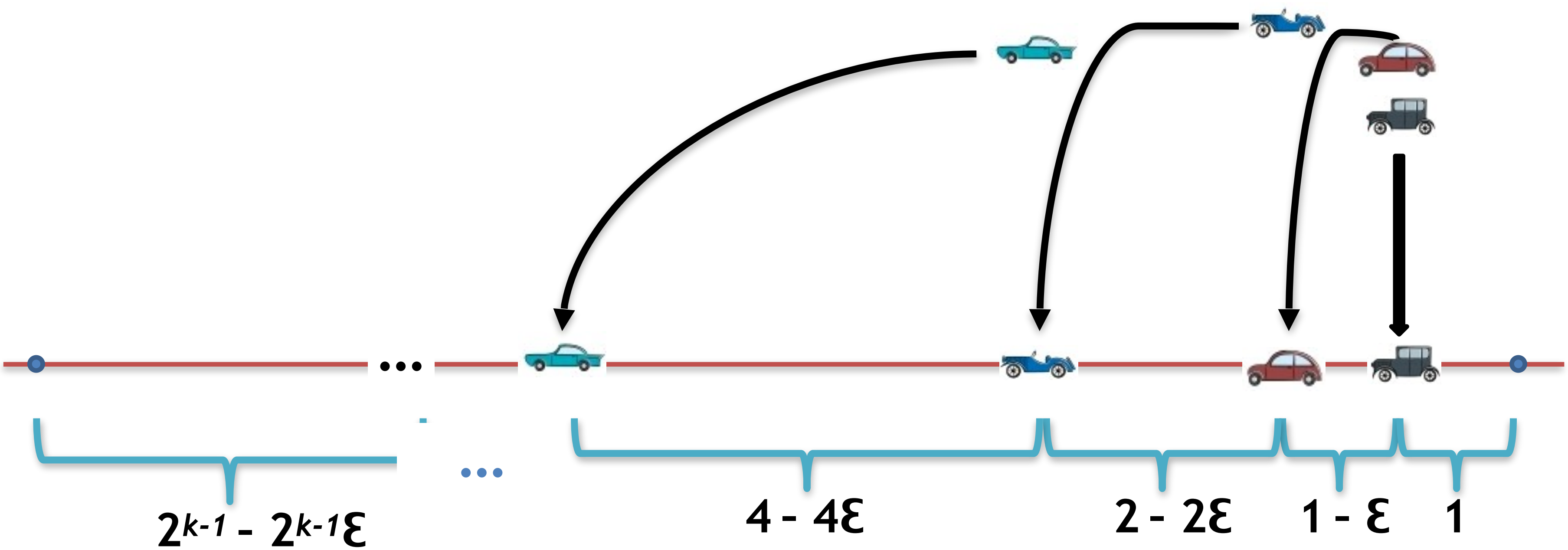
$2 - 2\varepsilon < 2 - \varepsilon$

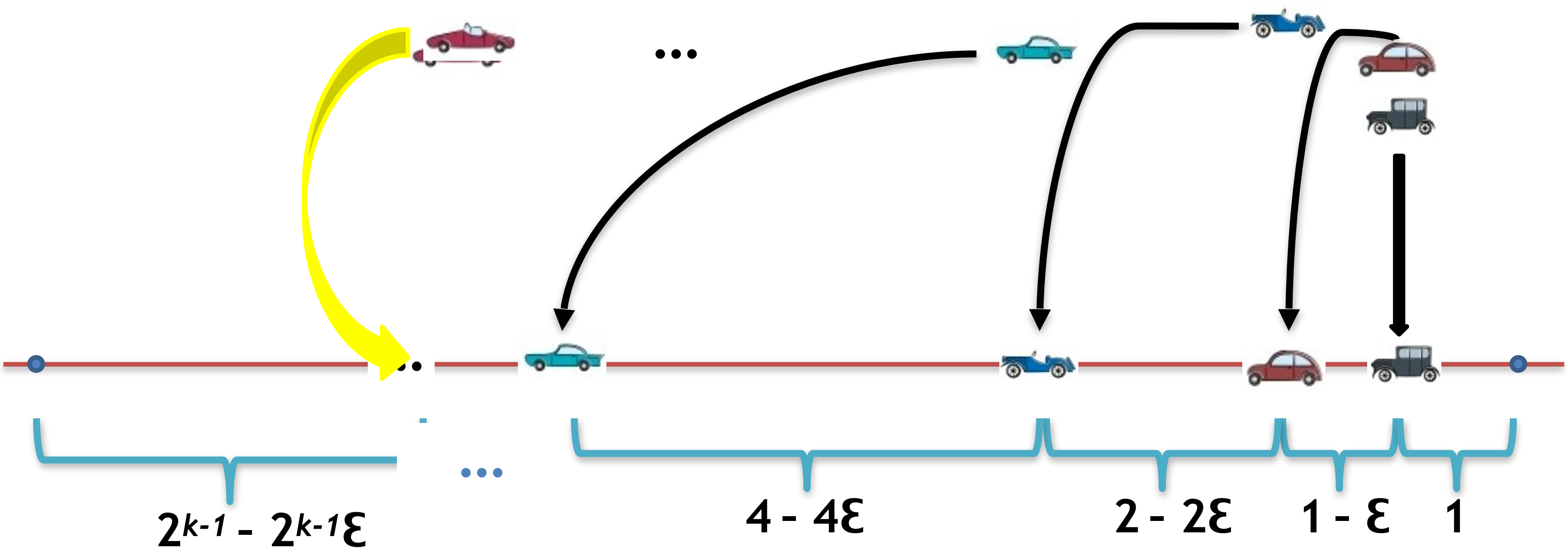


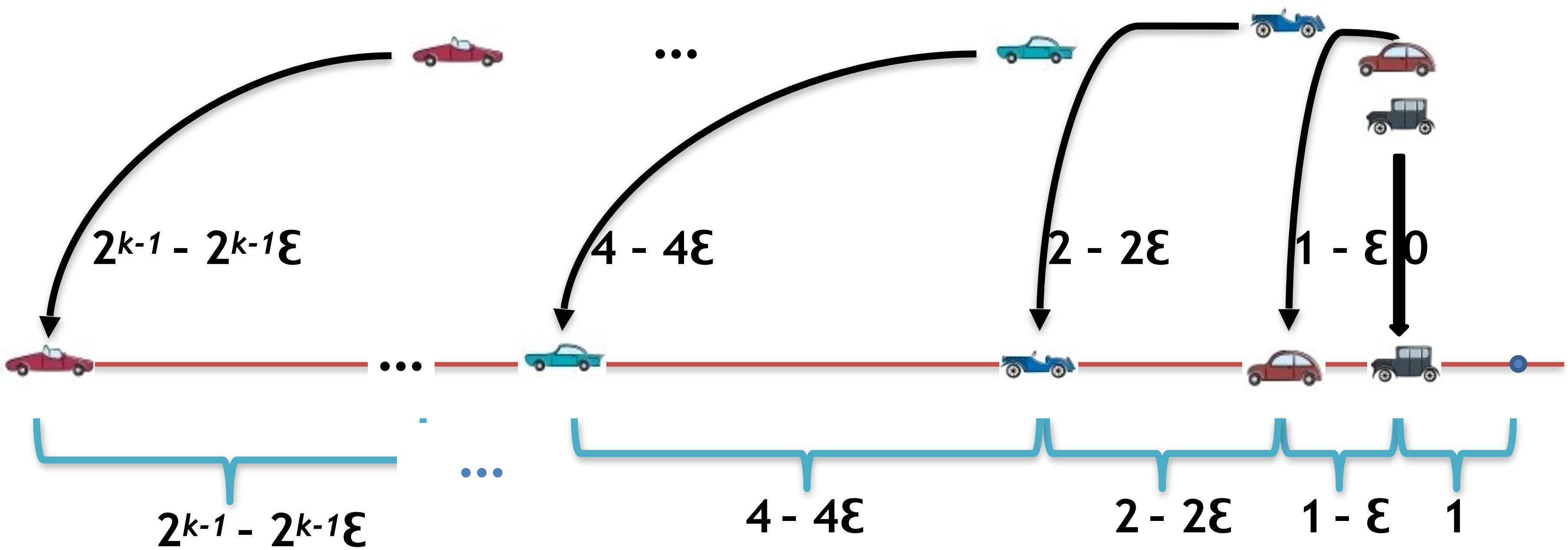


•••
•••
$2^{k-1} - 2^{k-1}\varepsilon$
$4 - 4\varepsilon$
$2 - 2\varepsilon$
$1 - \varepsilon$
1

$$4 - 3\epsilon > 4 - 4\epsilon$$




...
...
$2^{k-1} - 2^{k-1}\varepsilon$
$4 - 4\varepsilon$
$2 - 2\varepsilon$
$1 - \varepsilon$
1



Cost of free parking $\Omega(2^k)$
Optimal cost 1

First take: online algorithms

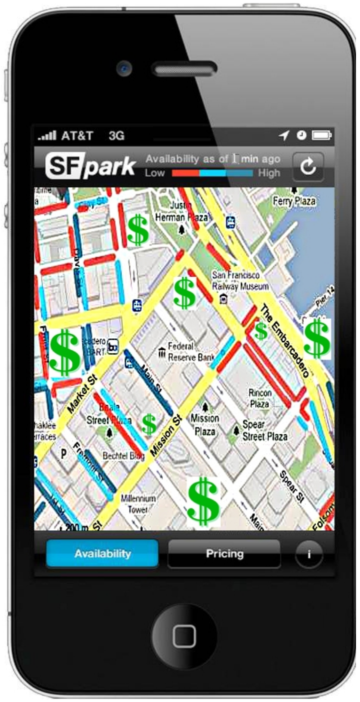
- Lewi and Gupta (2012) devised a very simple randomized $O(\log n)$ competitive algorithm for this problem.
- Ask drivers where they want to go.
- Direct them to a parking spot.
- **Not truthful.**
- Try telling drivers that they **"Can't park here, go there"**.

Dynamic Pricing

- Dynamic pricing sets a surcharge on possible decisions “by future events”.
- Market forces do the rest.

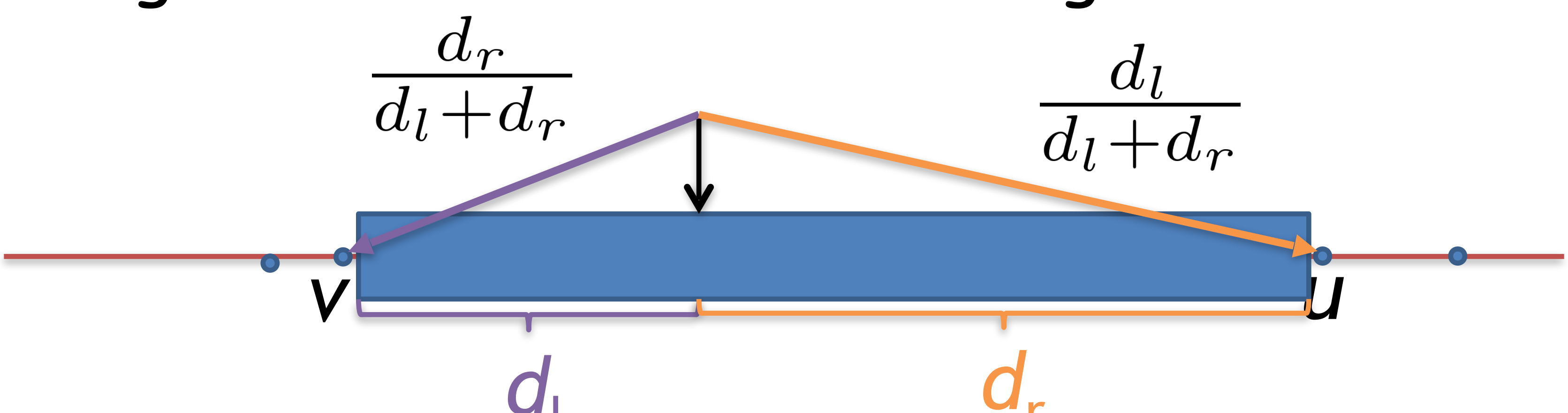
Dynamic pricing in practice

- Add surcharges to parking slots and adapt to changing circumstances.
- **SFpark, LA Express Park:**
  - Demand responsive pricing.
  - At least one parking slot per block.
  - Reduces circling. "circle less live more".
  - Does not minimize sum of distances (social cost)



The complete dynamic pricing algorithm

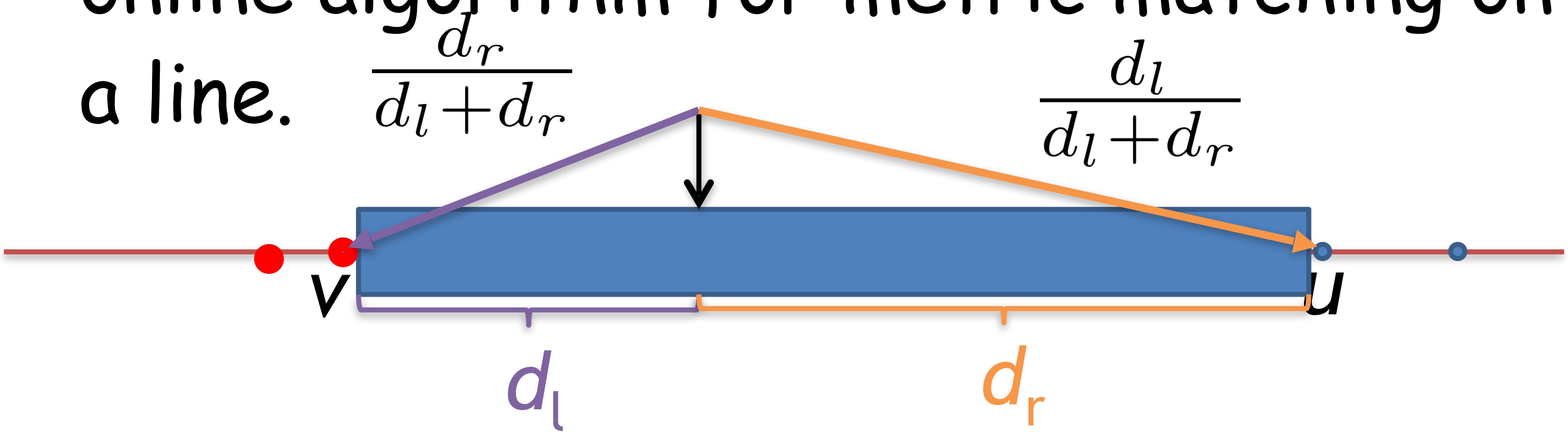
- Lewi and Gupta (12') gave a randomized online algorithm for metric matching on a line.



- Competitive ratio $O(\log d_{\max})$.
- $O(\log n)$ using doubling.
- Clearly Not Truthful

Lewi and Gupta are not truthful

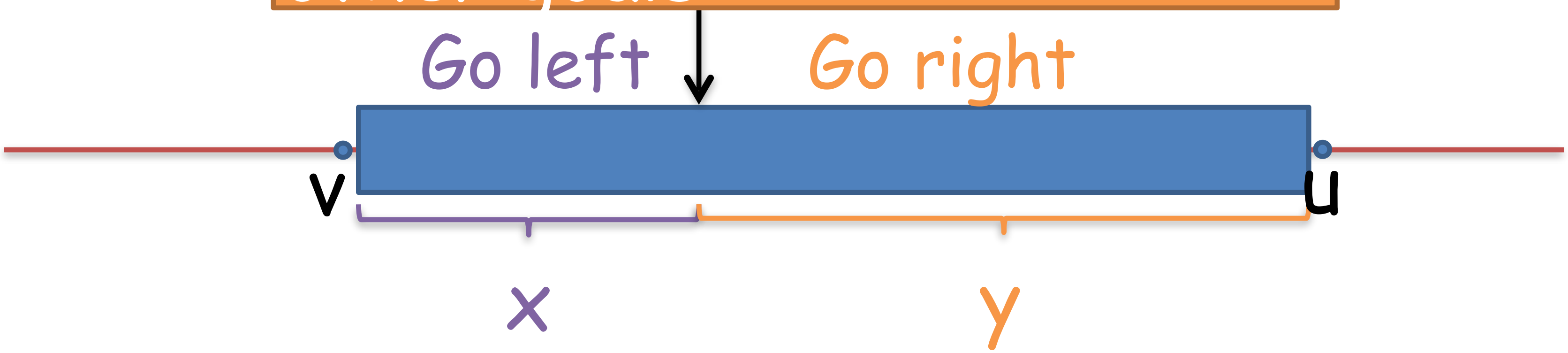
- Lewi and Gupta (12') gave a randomized online algorithm for metric matching on a line.



Much better to say your destination is at or even beyond the leftmost point

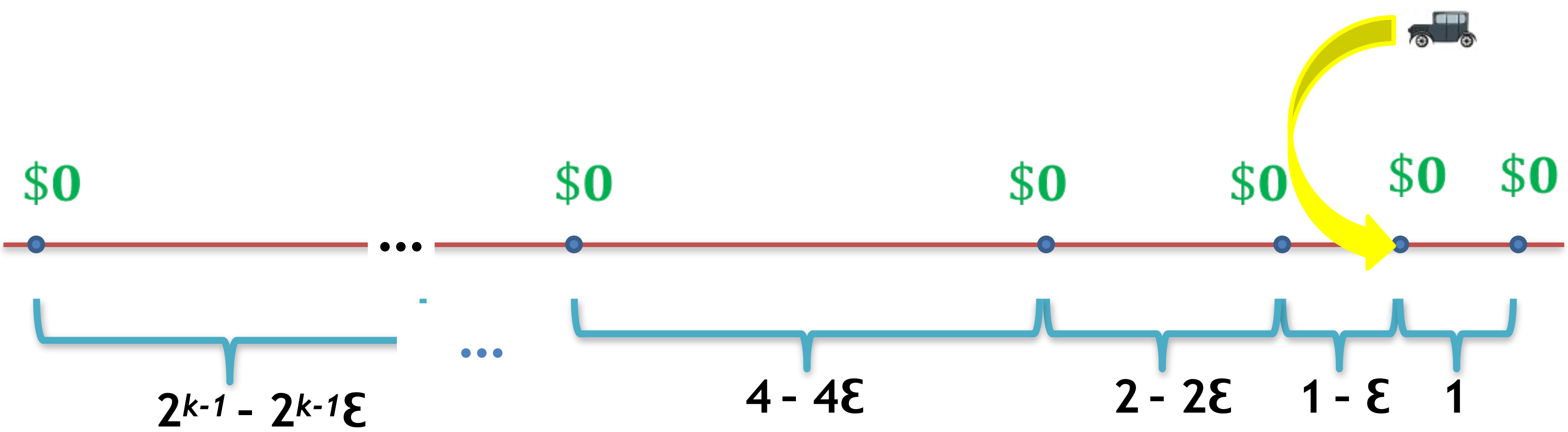
Emulating Lewi-Gupta via Prices

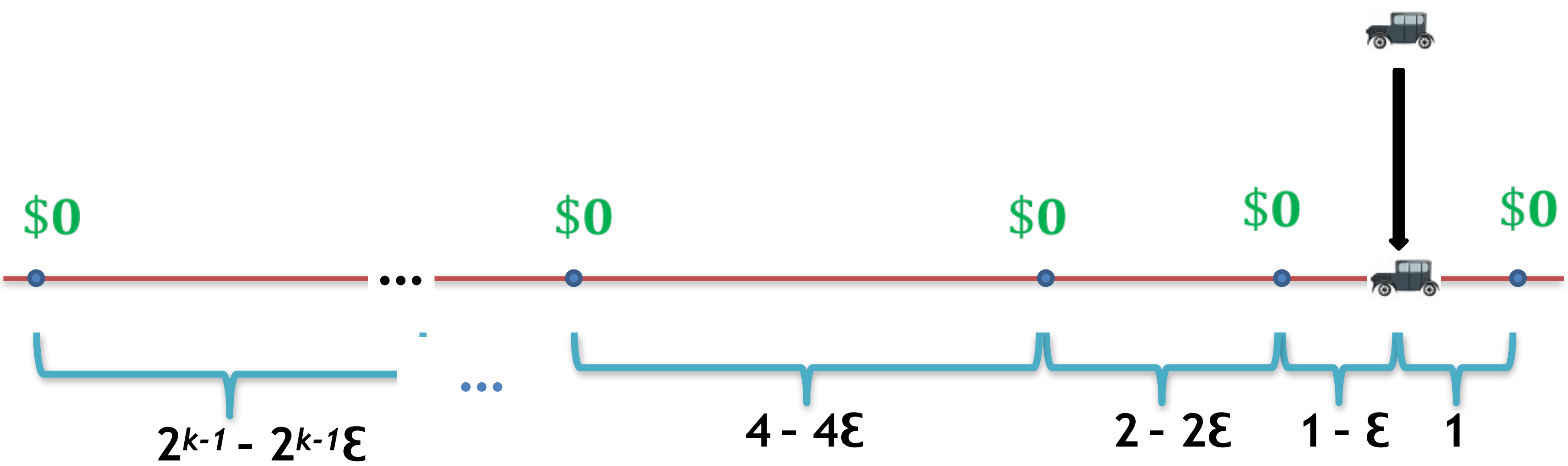
Random "break even" point. Can use LP to determine prices consistent with other goals



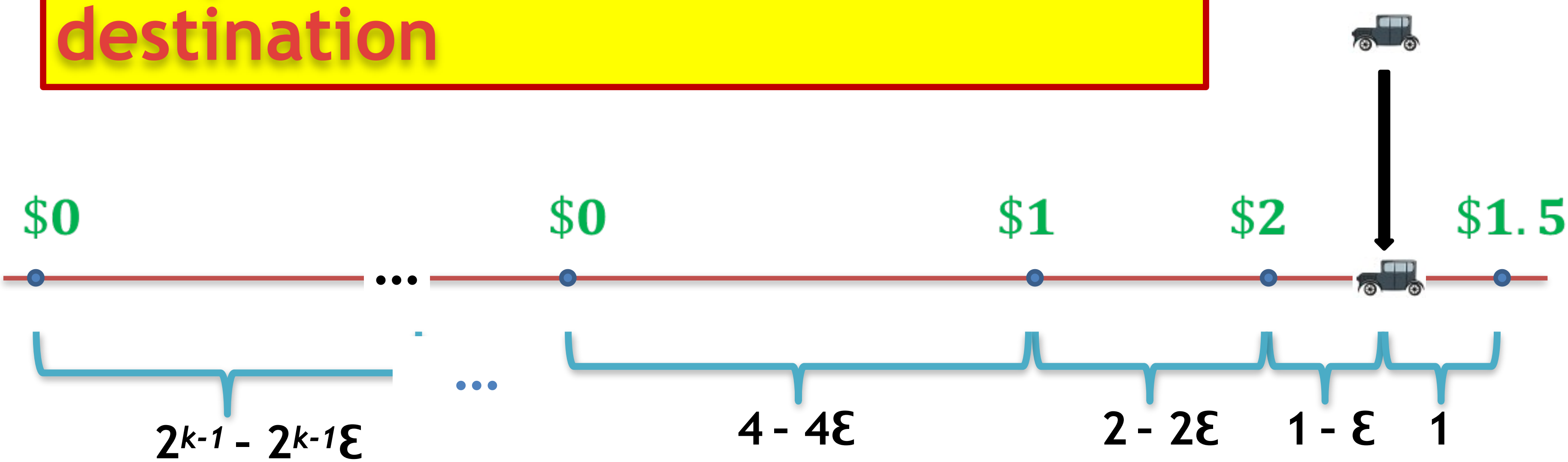
$$P(v) + x = P(u) + y$$

Make believe run of dynamic pricing

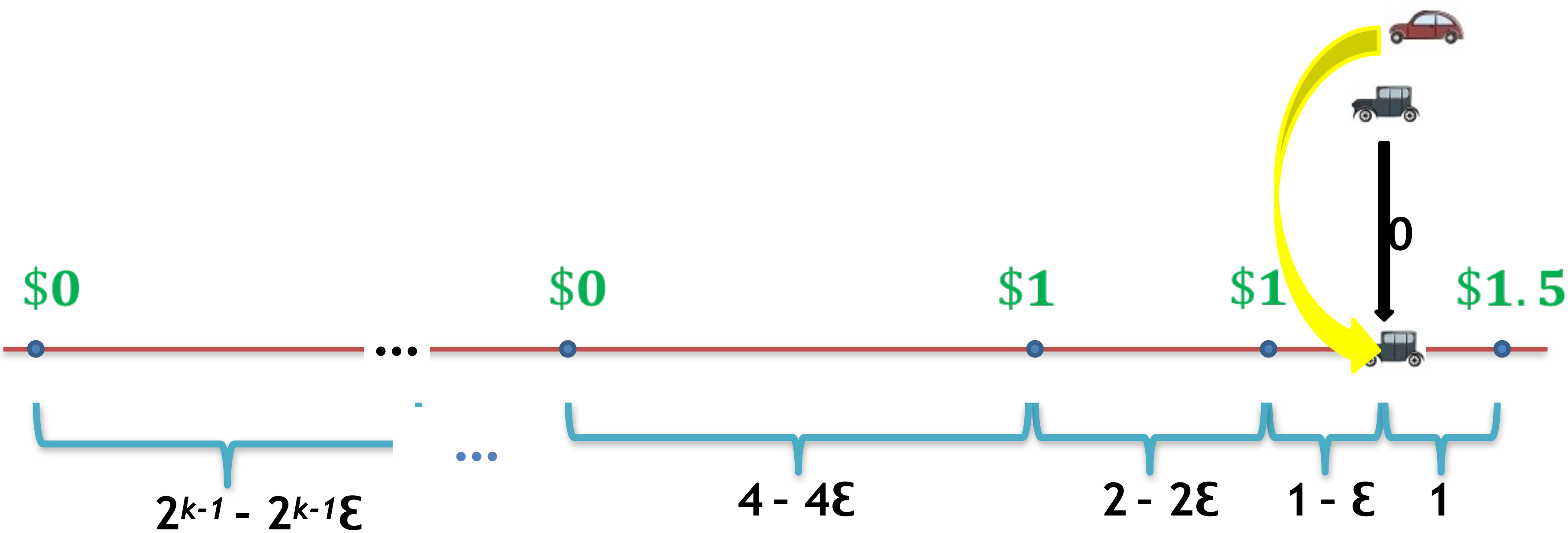


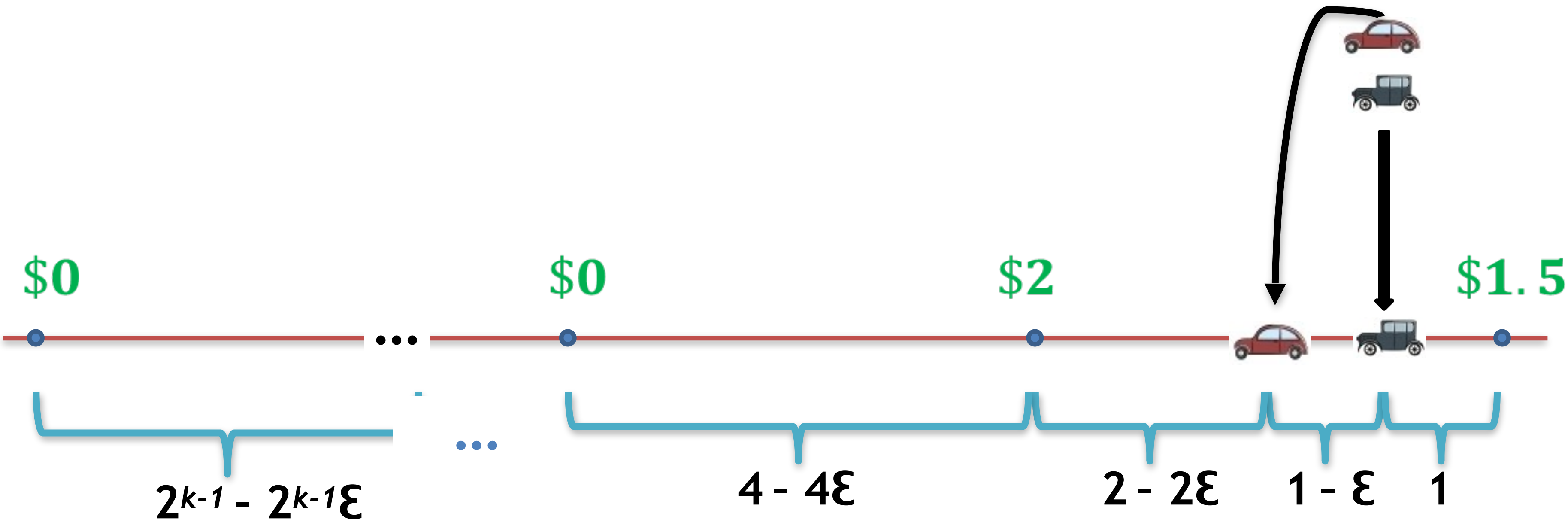

$0
$0
$0
$0
$0
…
…
$2^{k-1} - 2^{k-1}\varepsilon$
$4 - 4\varepsilon$
$2 - 2\varepsilon$
$1 - \varepsilon$
1

Surcharges Change over time, Independent of the next destination



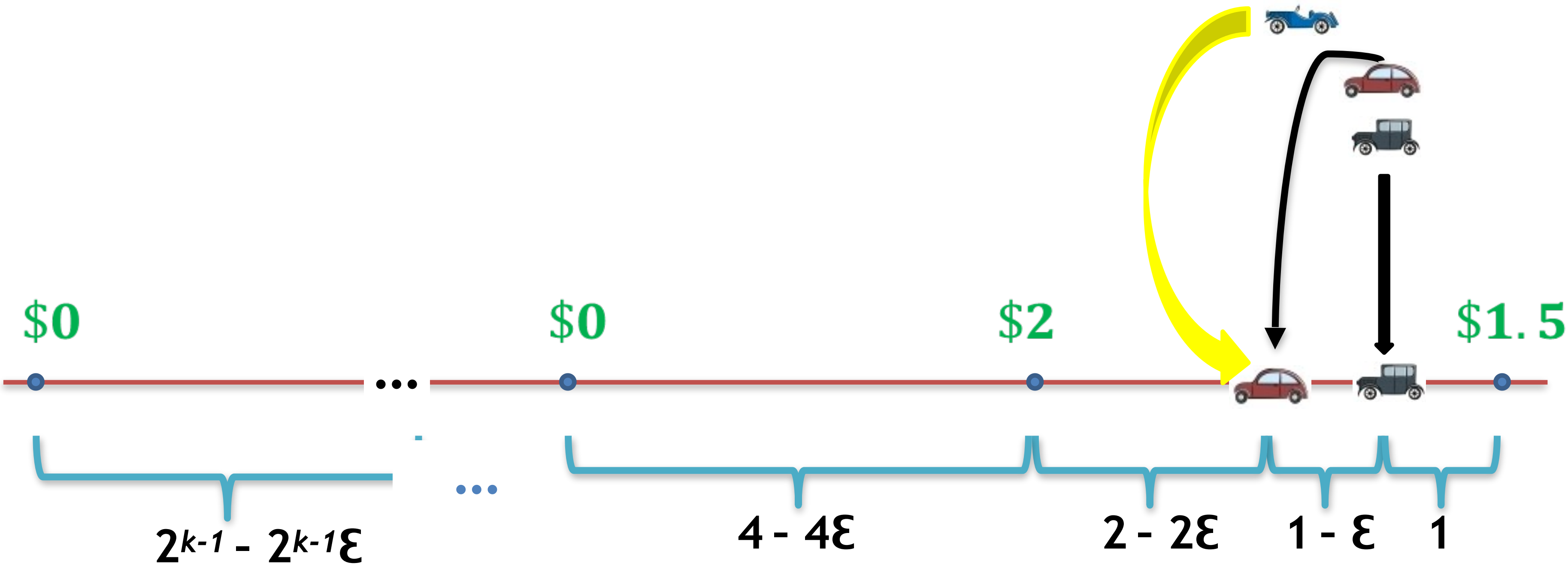
$1 + 1 - \varepsilon < 1.5 + 1$



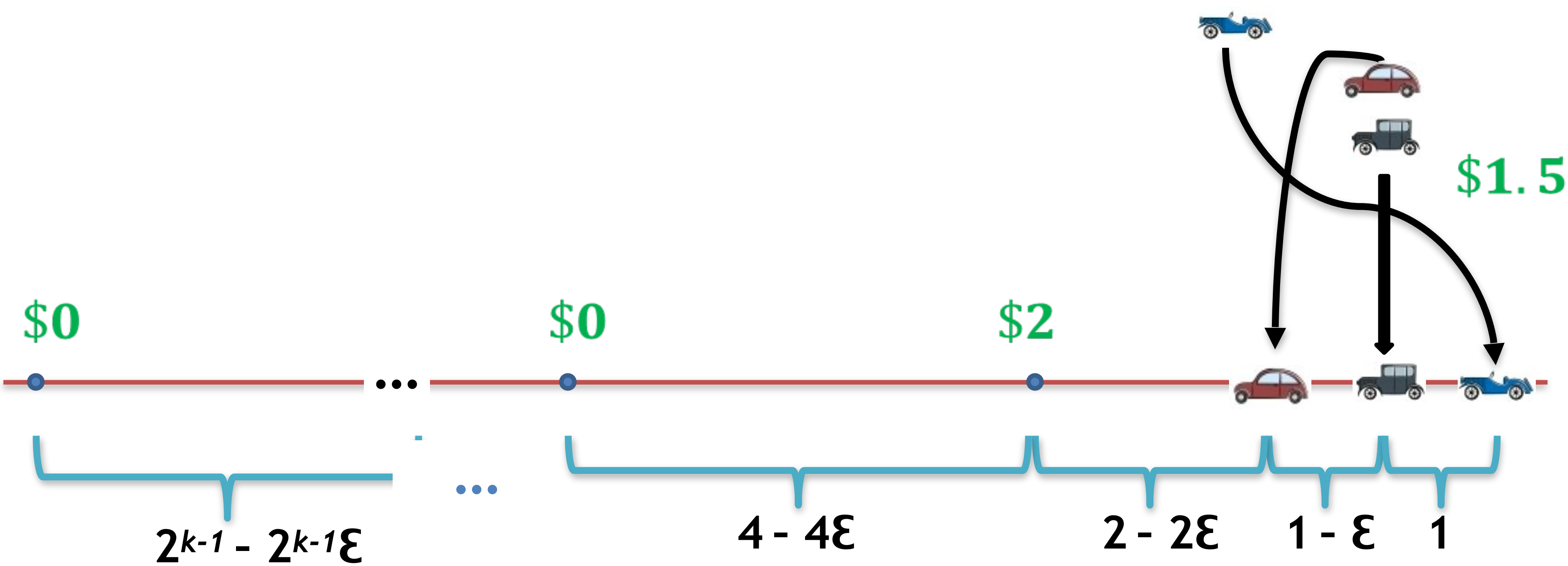


$0
$0
$2
$1.5
$2^{k-1} - 2^{k-1}\varepsilon$
$4 - 4\varepsilon$
$2 - 2\varepsilon$
$1 - \varepsilon$
1

$$2 + 2 - 2\varepsilon > 1.5 + 2 - \varepsilon$$



$$2 + 2 - 2\varepsilon > 1.5 + 2 - \varepsilon$$



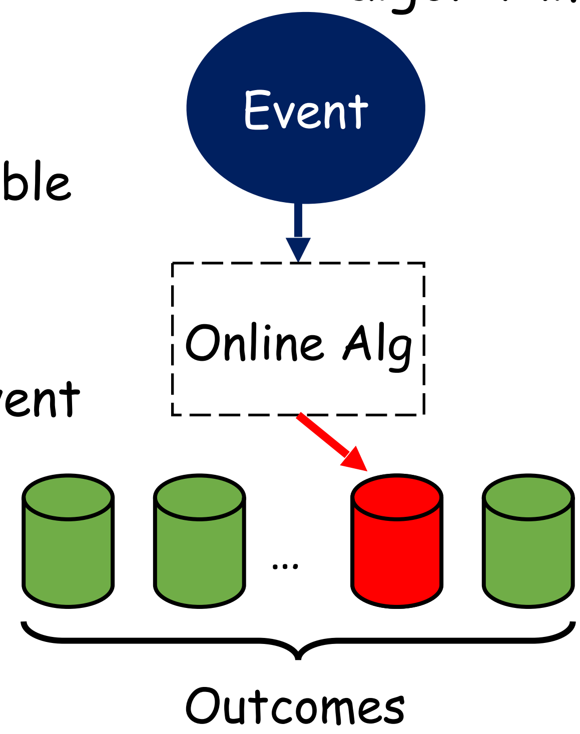
Cost of free parking $\Omega(2^k)$
Cost with "dynamic parking": ≈ 3
Cost = weight of metric matching (social cost)

Online Algorithms

Events arrive on the fly and are processed by an online algorithm

Decision is irrevocable

Based on:
history + current event

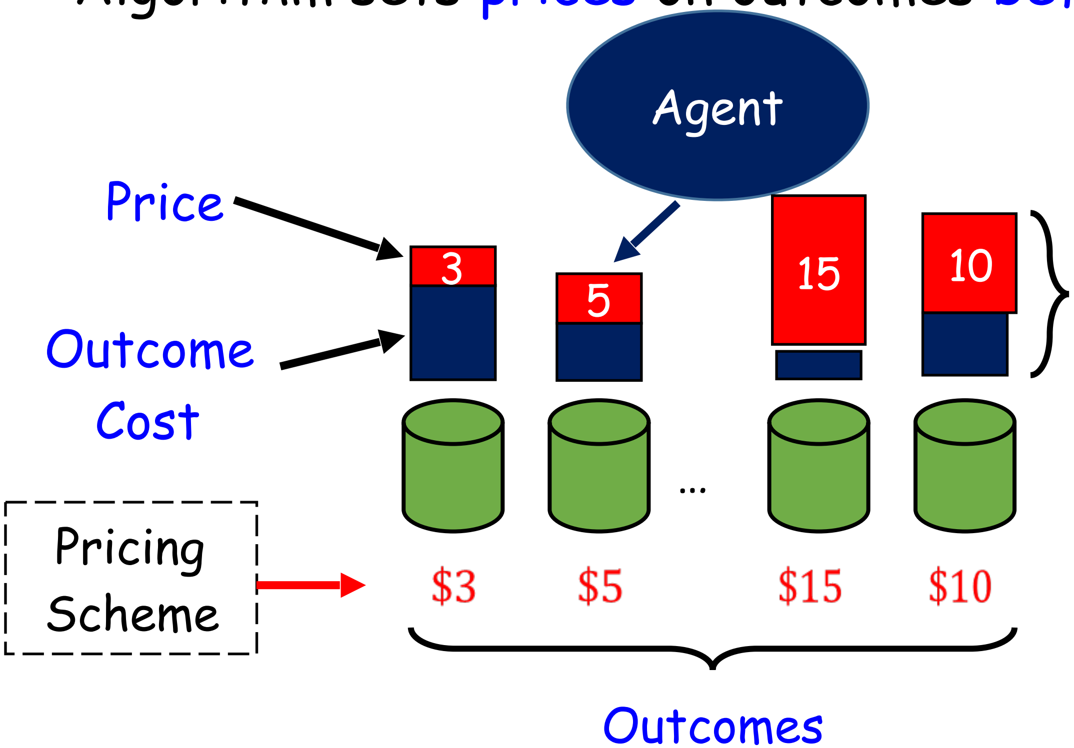


Minimize Competitive Ratio c

$$\frac{ALG}{OPT} \leq c$$

Dynamic Pricing

Algorithm sets prices on outcomes before seeing the event

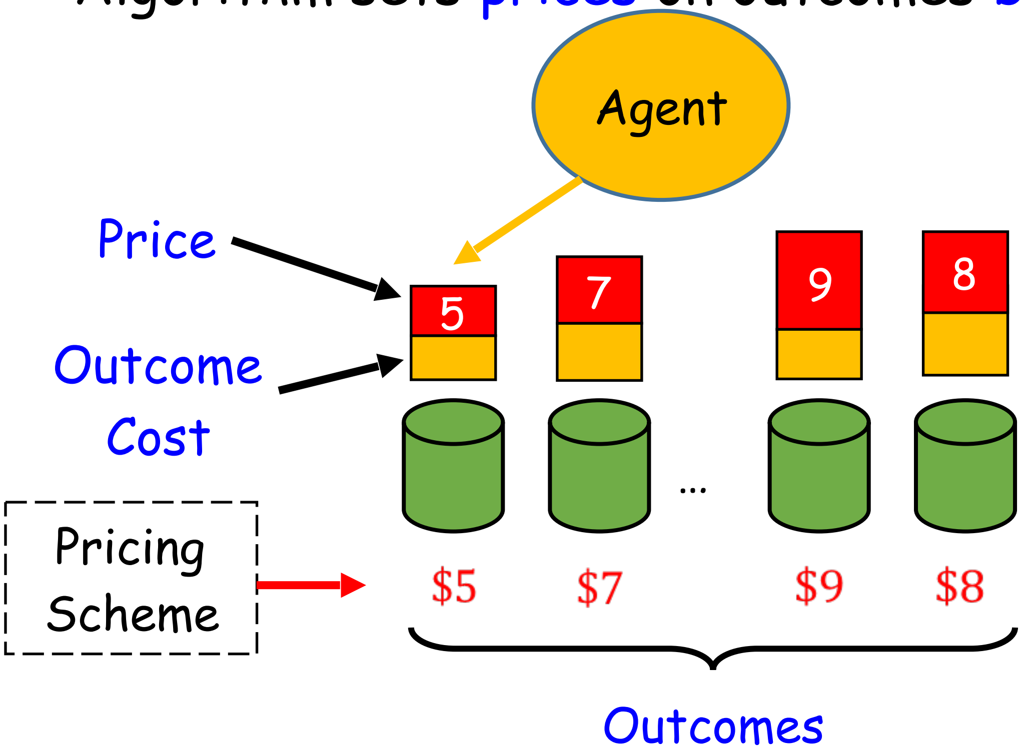


Agent Cost:
(Outcome Cost) + Price

Agents minimize cost

Dynamic Pricing

Algorithm sets prices on outcomes before seeing the event



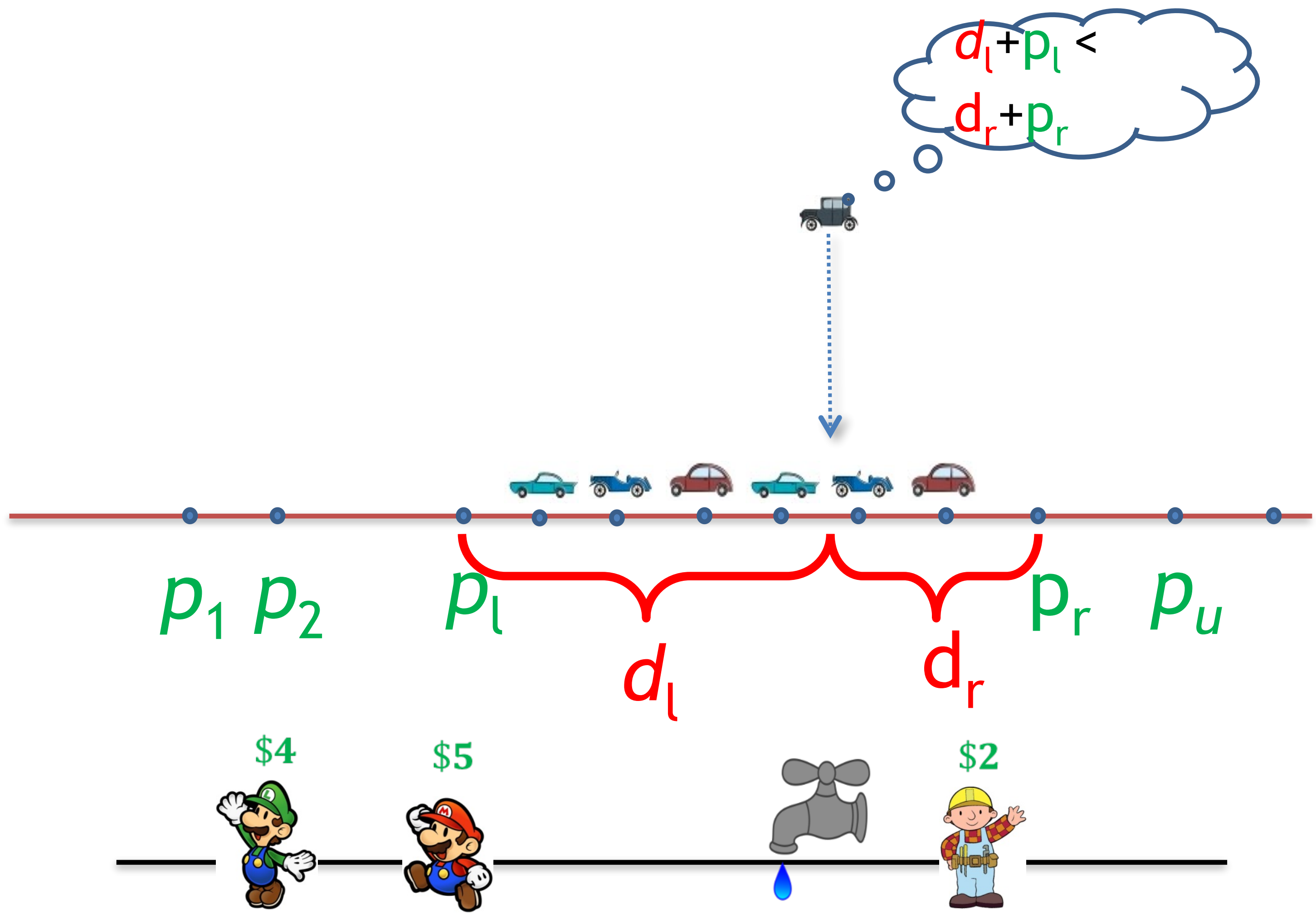
Dynamic Pricing:
Prices change over time

Static Pricing:
Prices set once and for all

Truthful and Simple

Dynamic prices are set before next event arrives

Parking



k server

k-server problem

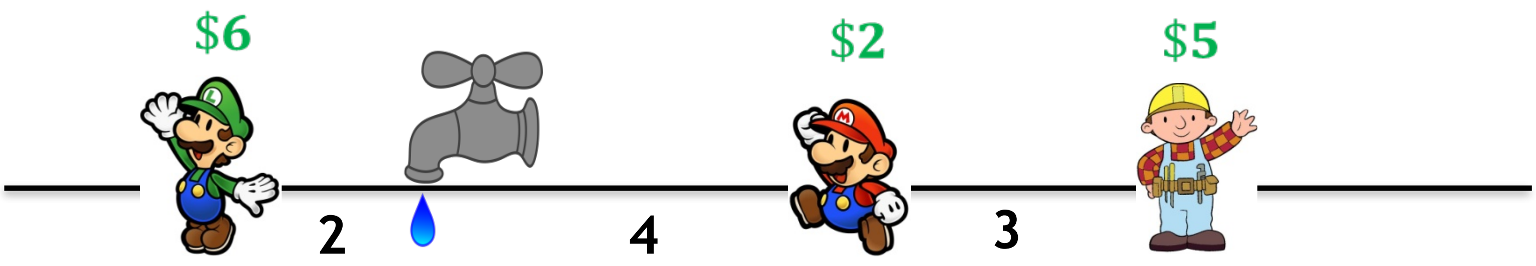
- k servers are located in some metric space.
- A request sequence arrives on line.
- Each request **must** be served by moving a server to its location.
- **Goal:** to minimize the total movement of the servers.
- Note that requests are served without knowing future events.

Selfish k-server problem

- Agents arrive online.
- The position where the agent requires service is private information.
- Each server has an associated surcharge, which the agent needs to pay in order to use the server.
- Every agent seeks to selfishly minimize her disutility – distance traversed by server **plus** surcharge.
- Once agent i moves a server, the system observes the new server configuration and updates surcharges.

Selfish k-server problem

$$6 + 2 > 2 + 4$$



Selfish k-server problem

Selfish k-server problem

Double cover algorithm

- A k-competitive algorithm for line/tree metrics.
- Move adjacent servers at same “speed” until one of them reaches the request.

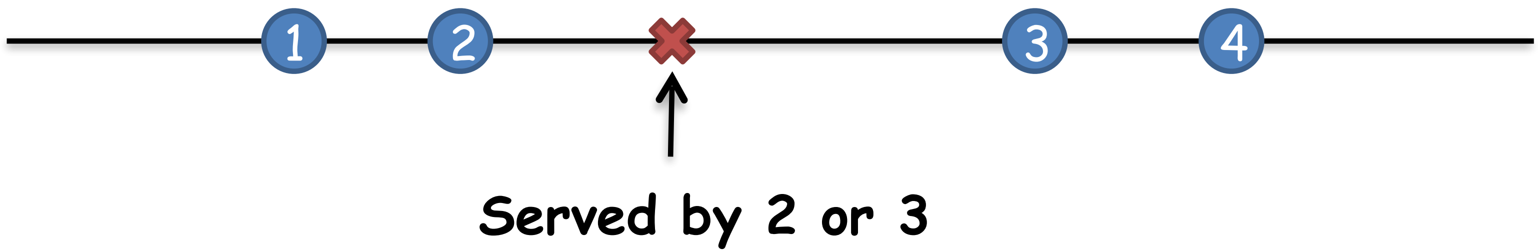
Double cover algorithm

- A k-competitive algorithm for line/tree metrics.
- Move adjacent servers at same “speed” until one of them reaches the request.

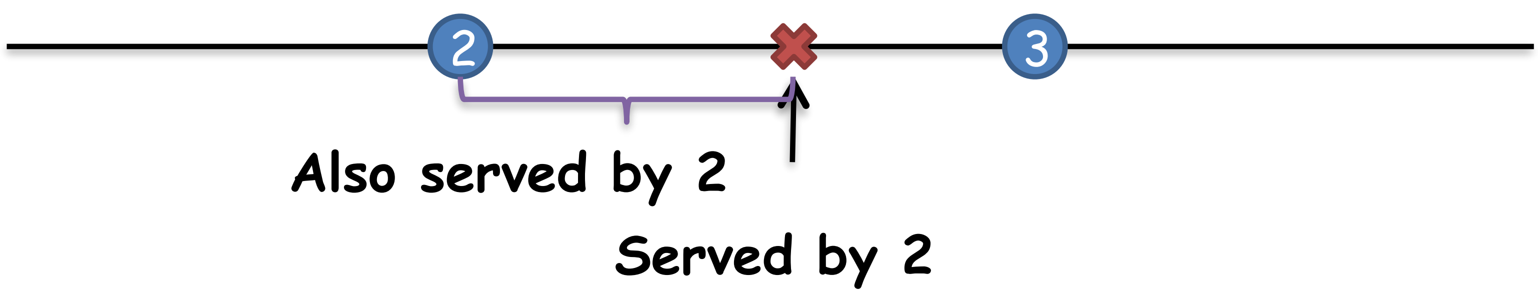
DC is an optimal k-competitive algorithm

Desired Properties for pricing

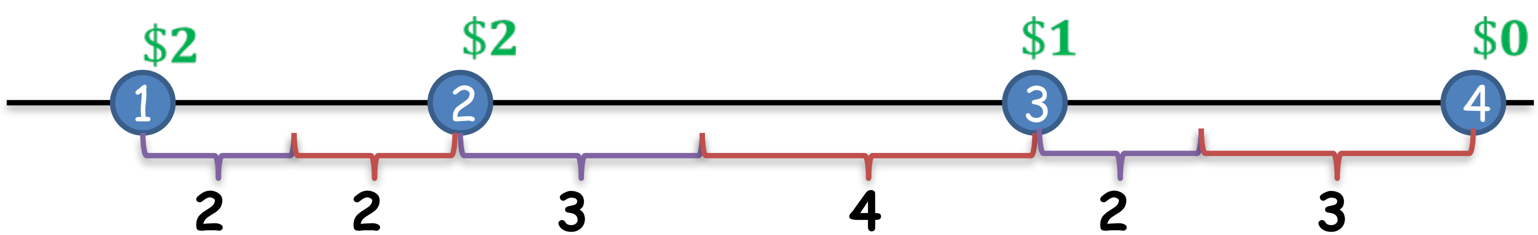
- **Locality**



- **Monotonicity**



Theorem: every local and monotone algorithm can be priced.

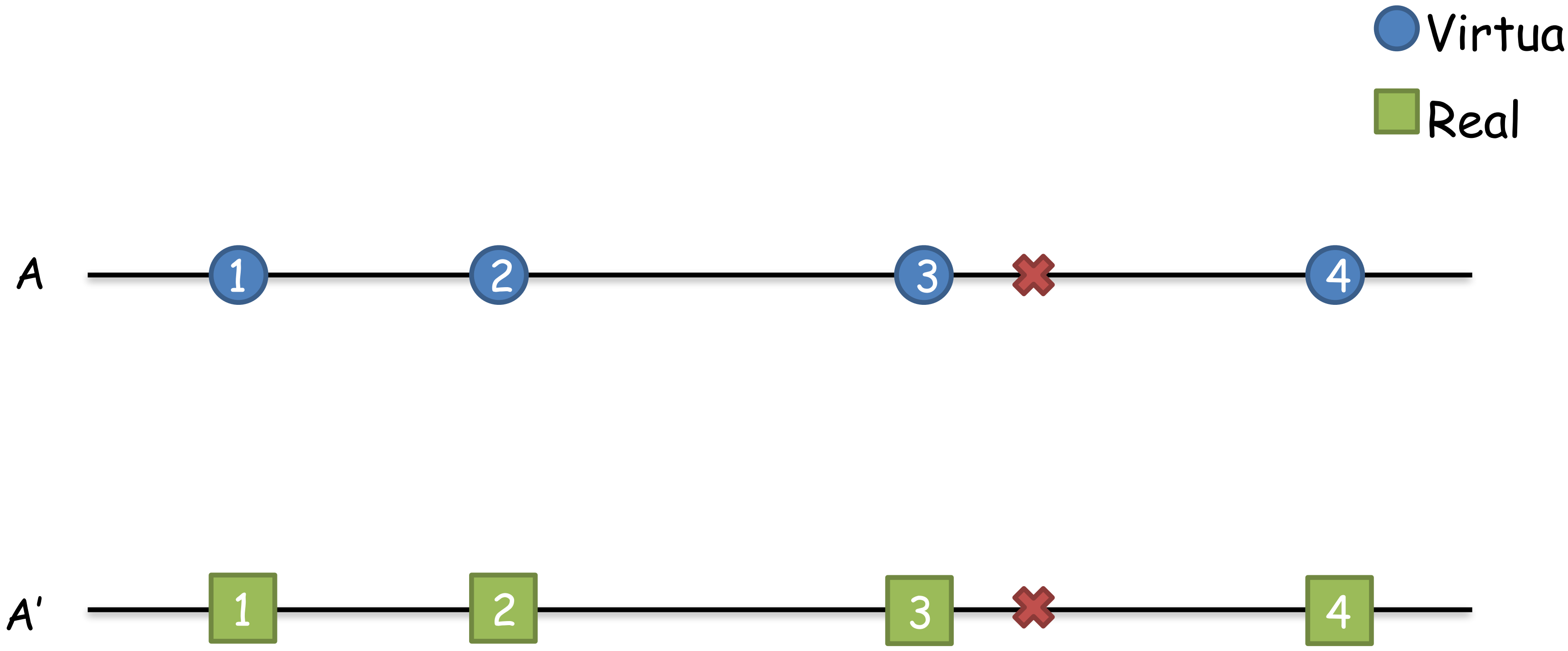


At the (single) point of transition, we want agents to be indifferent between the left and right server.

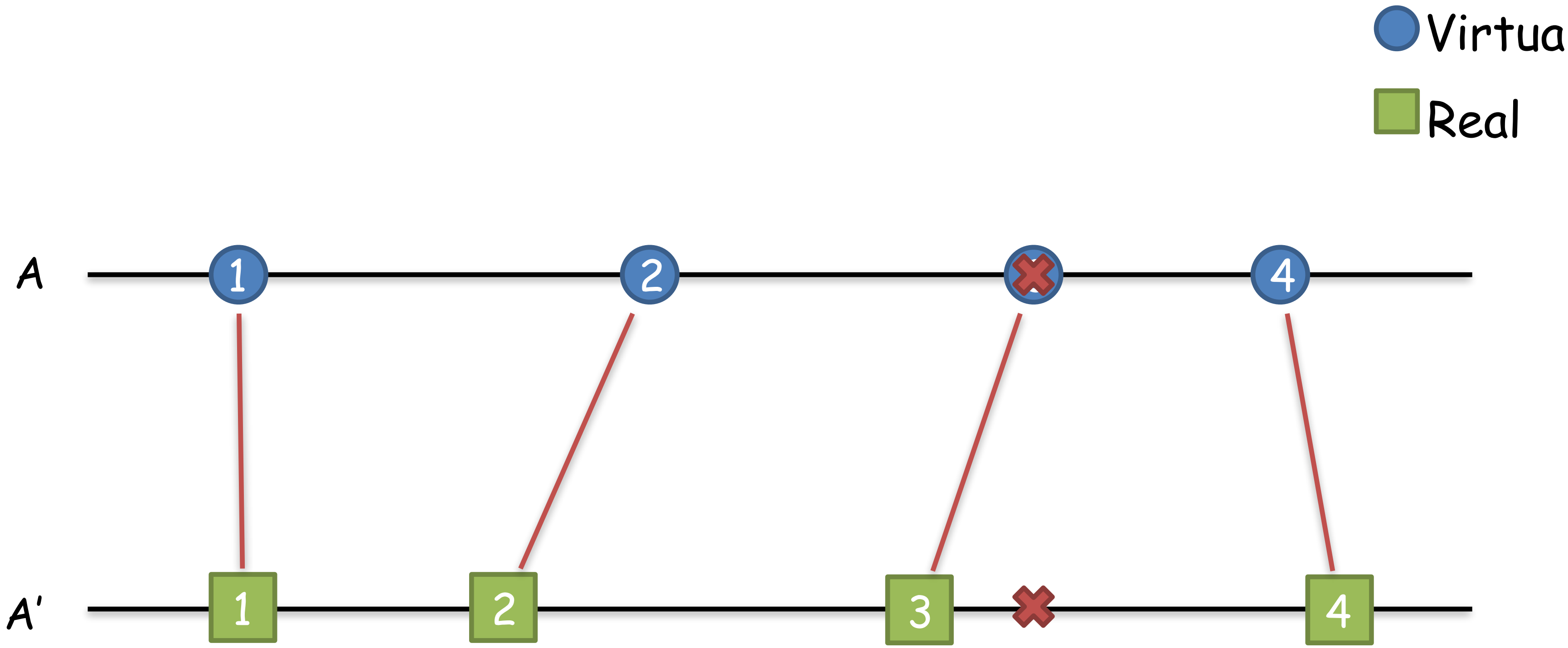
Producing a local algorithm

- **Input:** an online k-server algorithm A.
- **Output:** a local algorithm with no greater cost A'.
- Given request r_i :
  1. Simulate $A(r_i)$. Let **s** be the server that served the request.
  2. Find a local min cost matching between A and A' servers.
  3. Serve the request with the server matched to **s**.

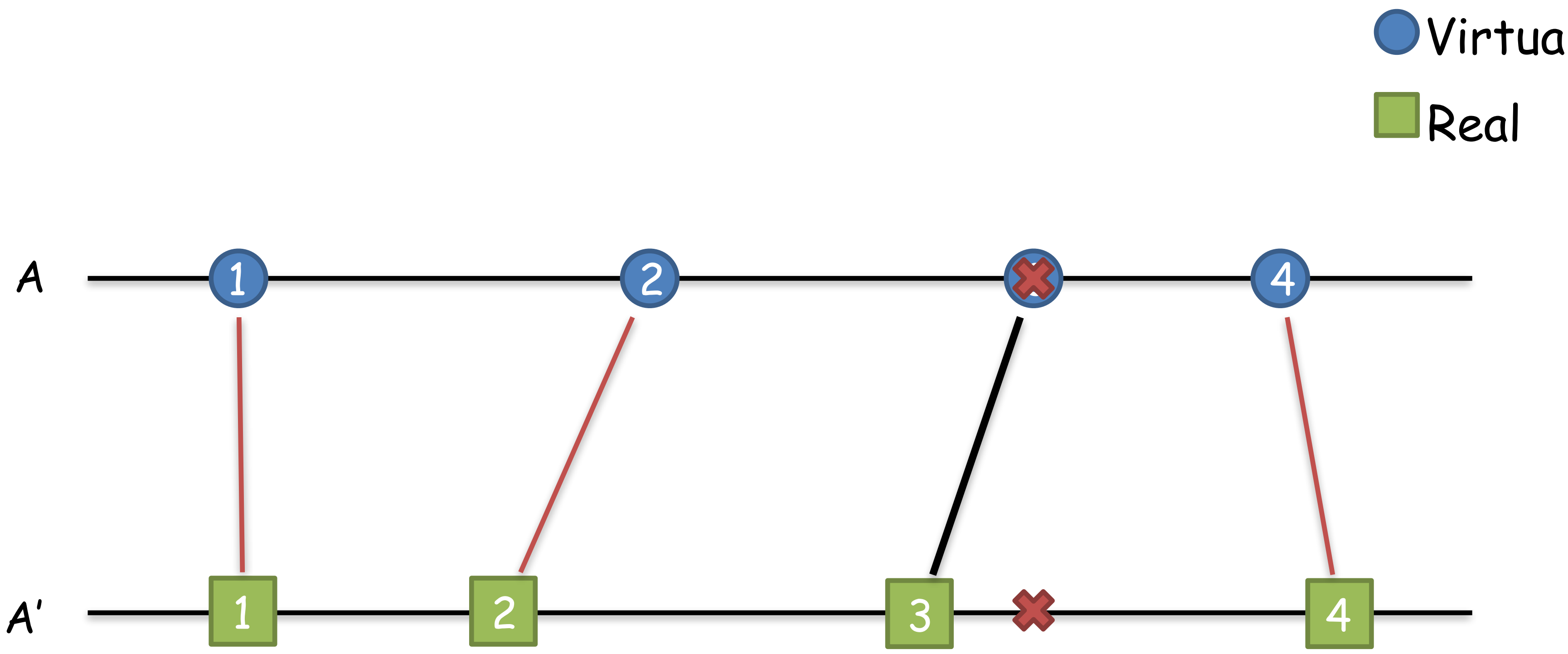
Producing a local algorithm



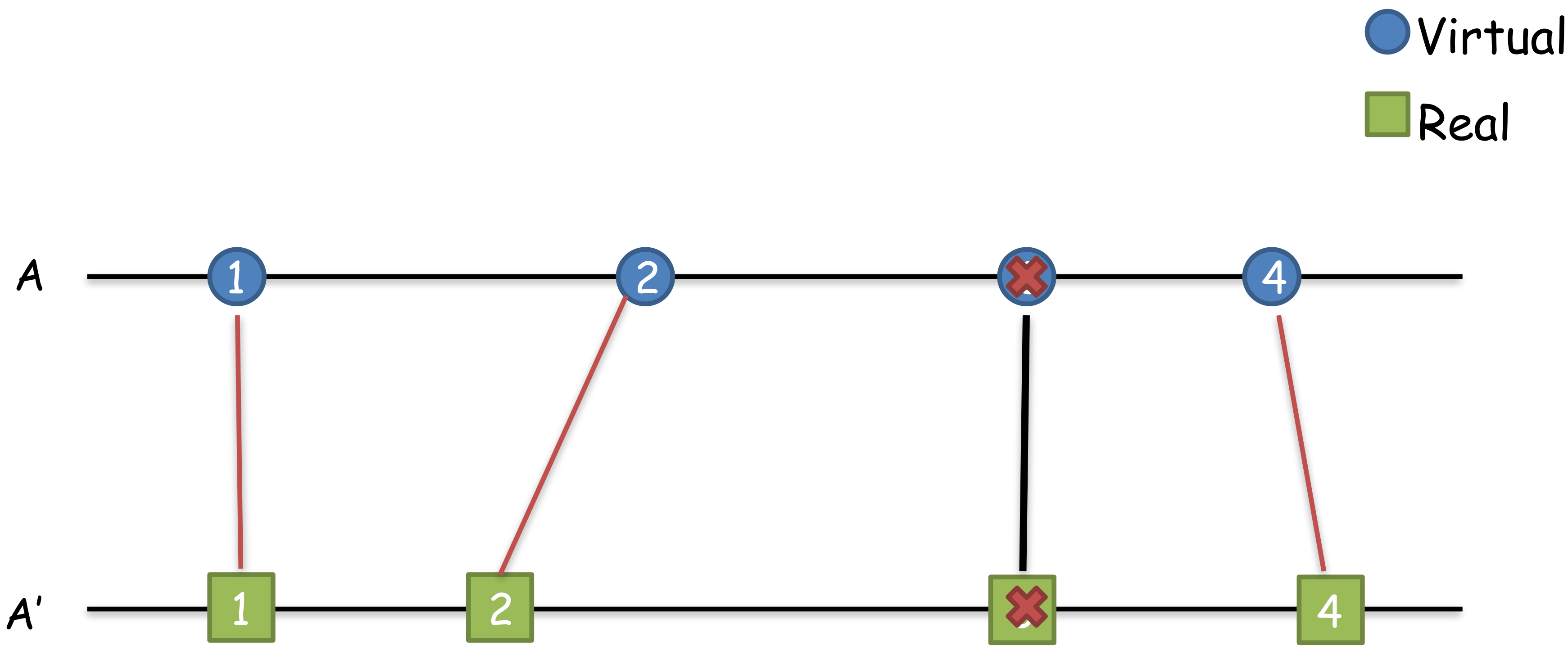
Producing a local algorithm



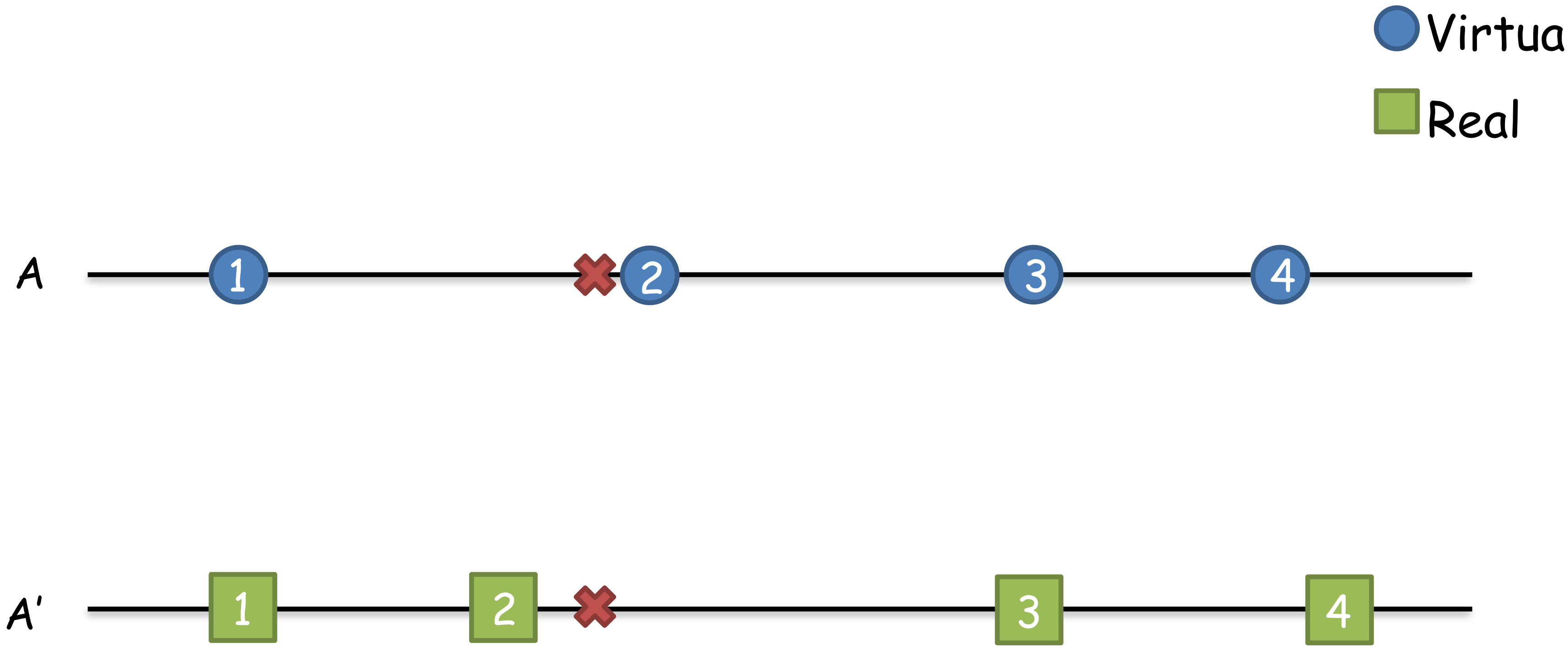
Producing a local algorithm



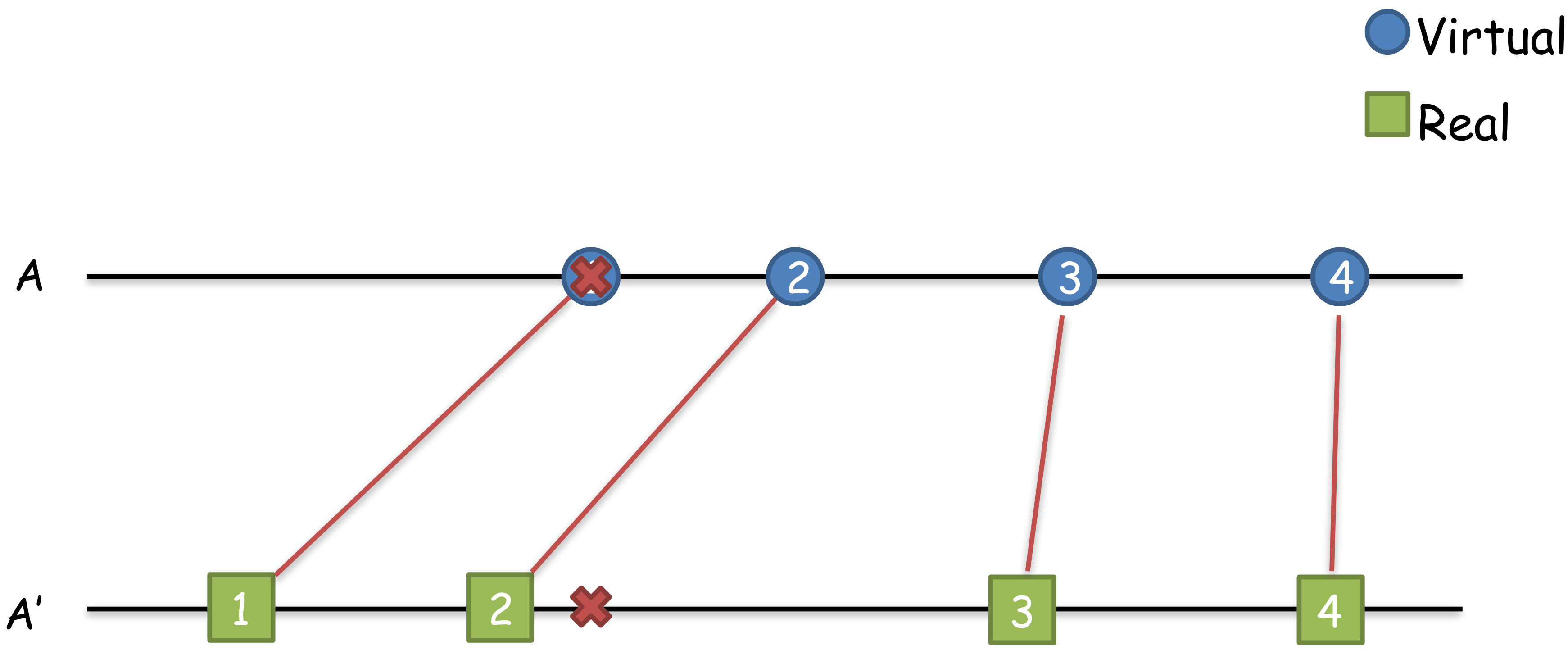
Producing a local algorithm



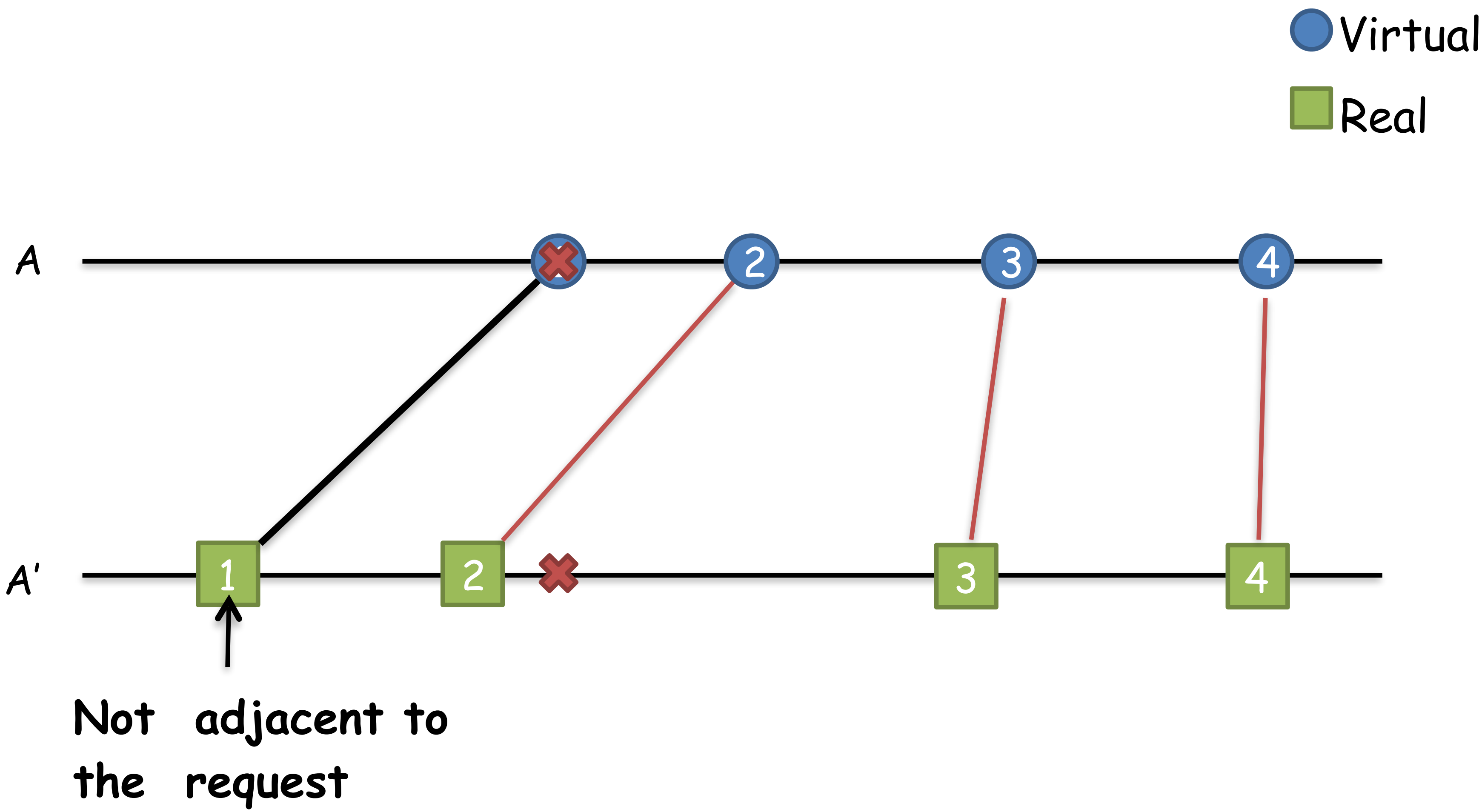
Producing a local algorithm



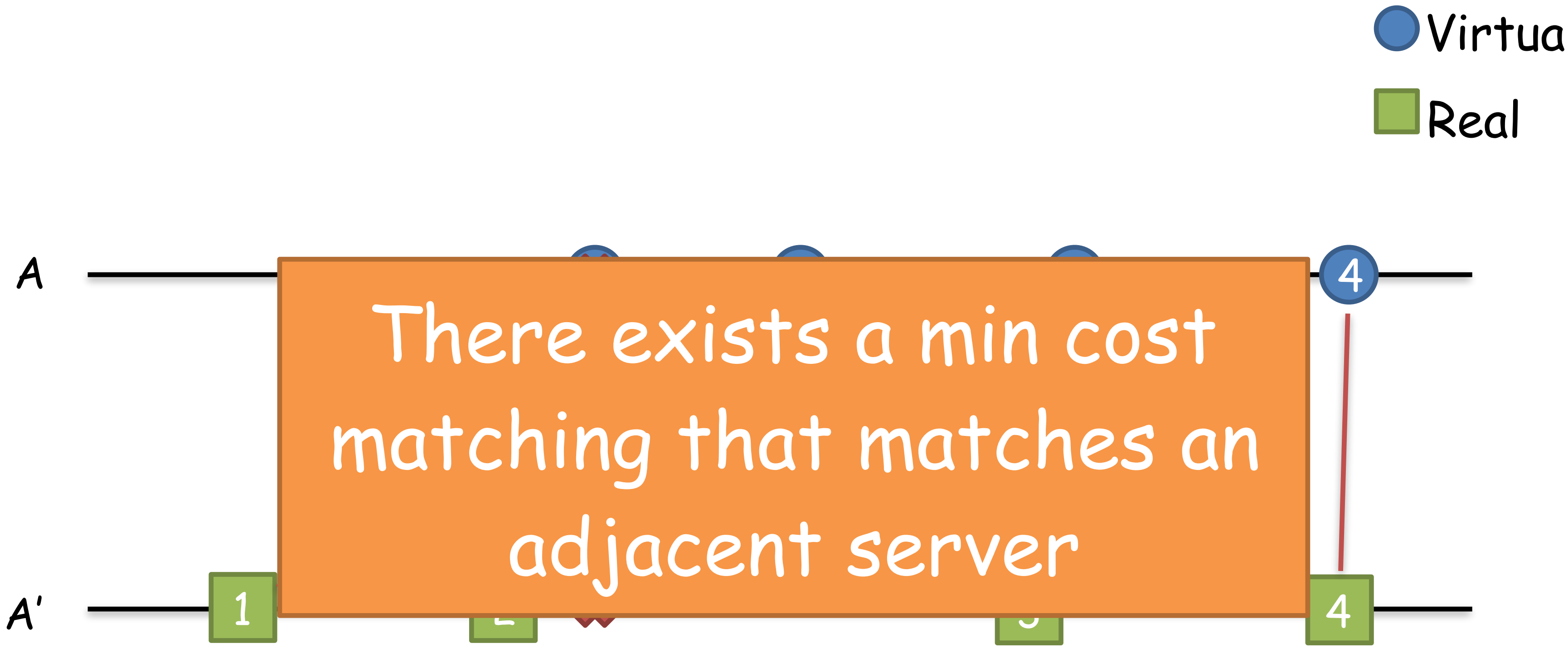
Producing a local algorithm



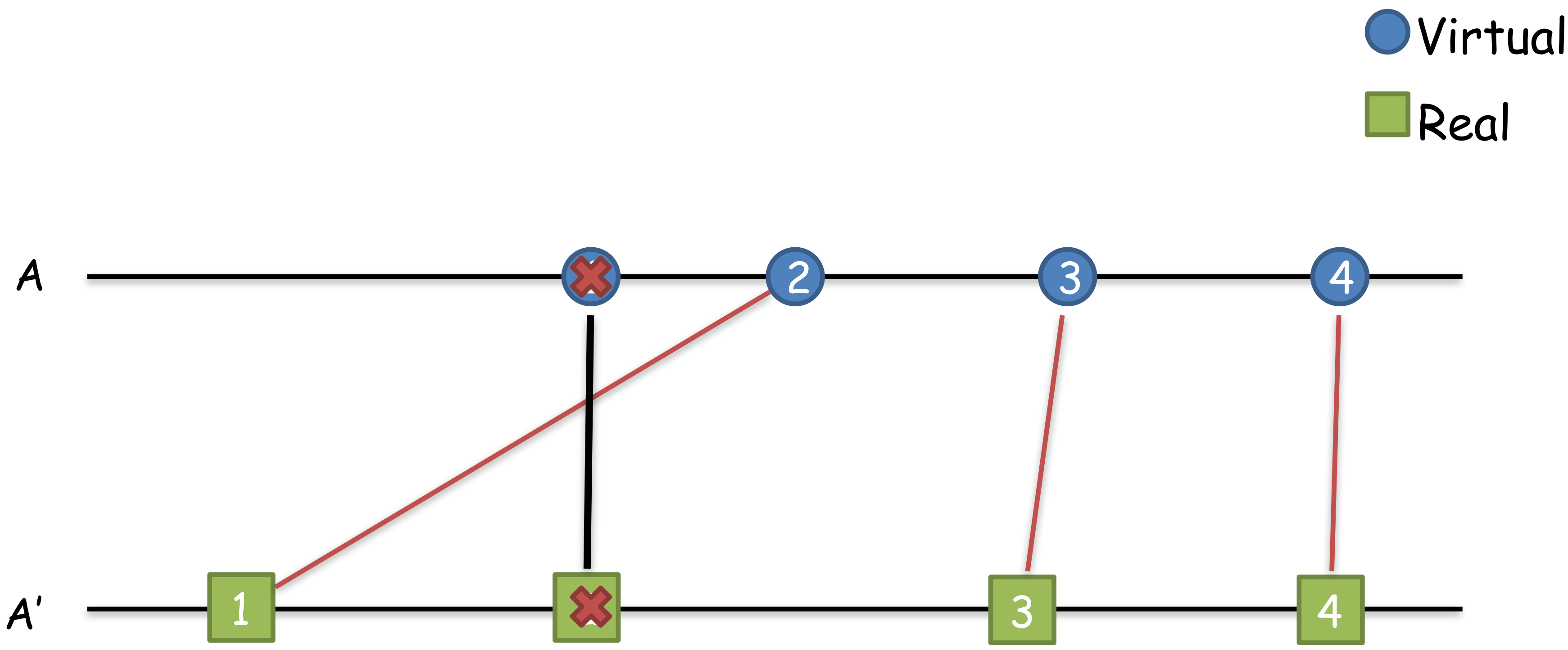
Producing a local algorithm



Producing a local algorithm



Producing a local algorithm



Metrical Task System (MTS)

- A set of states $S = \{1, \ldots, m\}.$
- A metric - transition cost state s and t is represented by $d_{s,t}.$
- An online sequence of tasks $w^1, w^2, w^3 \ldots$
- Task $w^i = (w^i_1, w^i_2, \ldots, w^i_m)$ is a vector where w^i_j is the cost of processing task i in state j.

MTS - cont.

- Given task i, an online algorithm must choose a state of the system on which to process the task.
- Let S_{i-1} and S_i be the previous and current states of the system, the cost associated with task i is $d_{S_{i-1},S_i} + w^i_{S_i}$.
- **Goal:** minimize the sum of transition costs plus task processing costs.

MTS – State of the Art

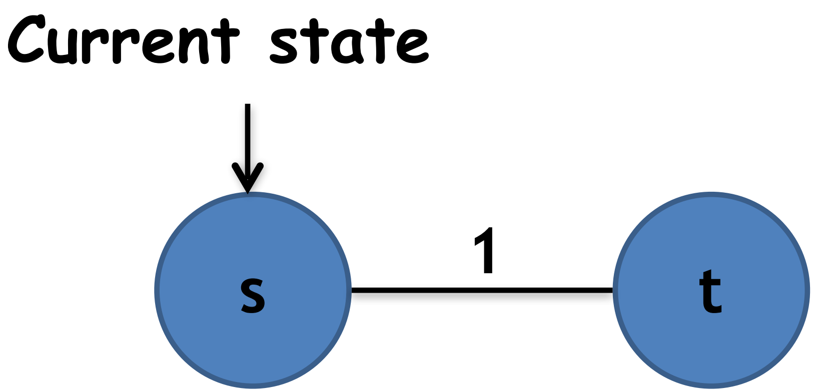
- A deterministic lower bound of $2m-1$ was shown in the seminal paper of Borodin, Linial and Saks [1987].
- They also showed a matching upper bound using a work function algorithm.
- A simple traversal algorithm was shown to be $8(m-1)$ competitive.
- Randomized algorithms were devised to get a sublinear approximation.

Selfish MTS

- An online sequence of selfish agents.
- Agents are associated with tasks they want to perform. The task is private information.
- Each state has an associated surcharge, which the agent needs to pay in order to process the task in that state.
- Every agent seeks to selfishly minimize her disutility – sum of transition cost, processing cost and surcharge.
- Once the player i processes her task, the system observes S_i and $w^i_{S_i}$.

Selfish MTS without pricing

- Can be arbitrarily bad…



- Every task is of the form $w_s = \frac{1}{2}$ and $w_t = 0$.

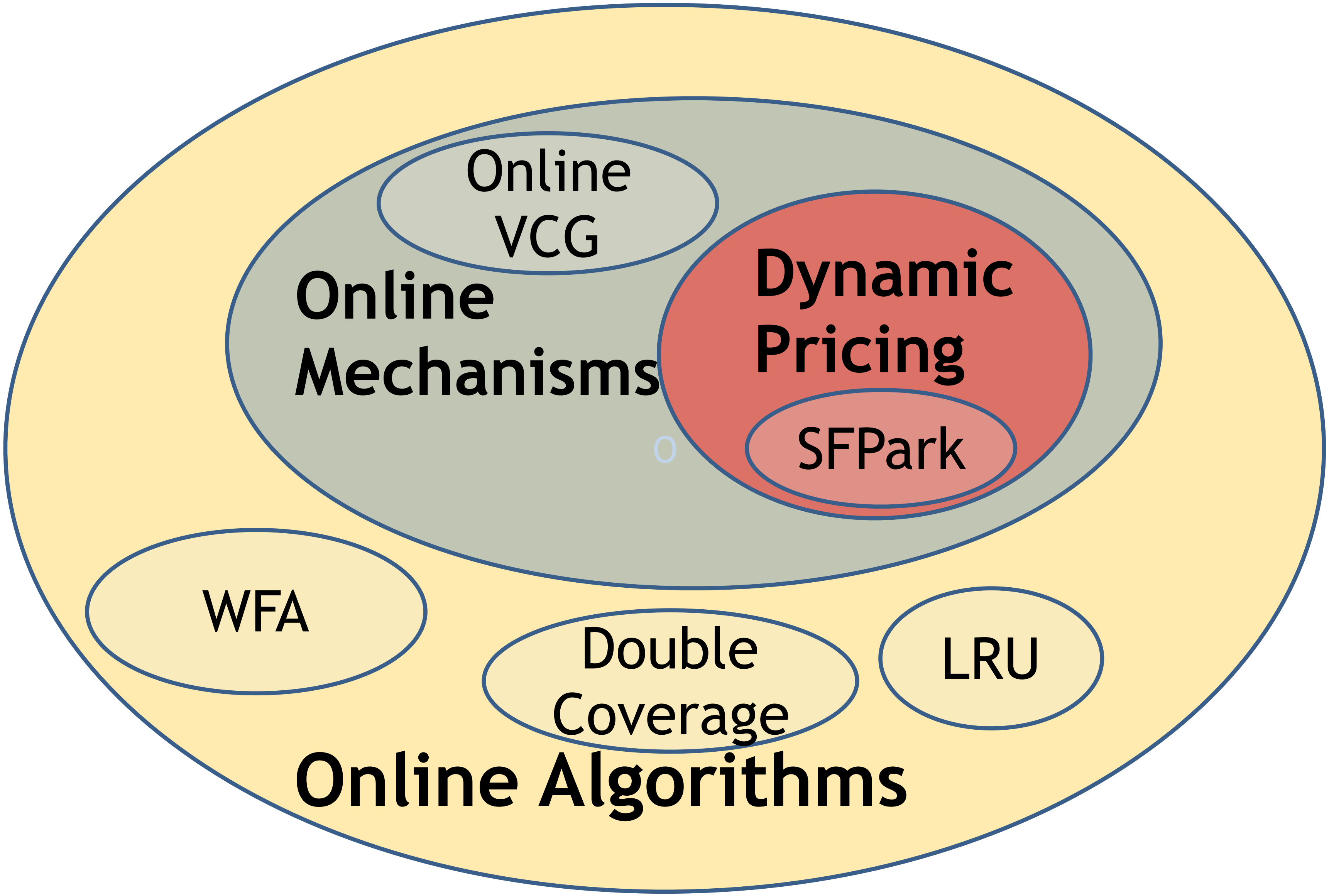
- No agent ever switches to state t.

[CEFJ15] on pricing online decisions

	Without pricing	Best Online	Dynamic Pricing
Metrical Task System	$\Omega(t)$	Optimal 2k-1-competitive (BLS)	(16k-1)-competitive
k-server on a * line	$\Omega(t)$ $\Omega(2^k)$	Optimal k-competitive (CL)	k-competitive $O(\log d_{\max})$
Metric matching on a line	$\Omega(d_{\max})$	Randomized $O(\log k)$	$O(\log k)^*$

t — number of agents in sequence

The world as we know it

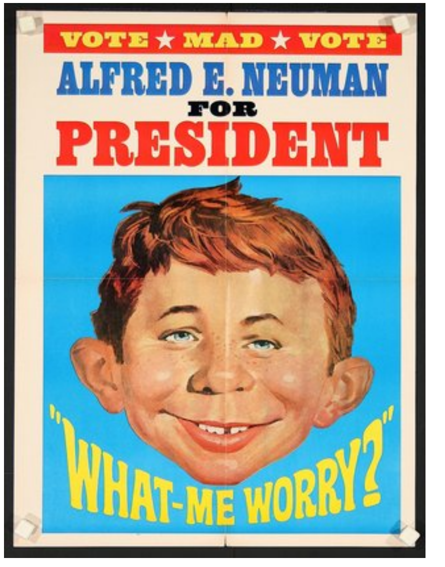


Open Problems

- Dynamic pricing for k-server on other metric spaces?
  - Remark: Result on line can be extended to trees
- Metric matching via dynamic pricing on other metric spaces?
- Limiting the rate of change of the prices over time?
- Knowledge requirements (?)

States of knowledge

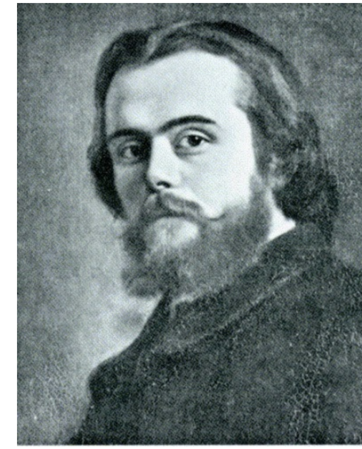
- 2m-1 optimal MTS online algorithm based on work function, in particular needs to know cost in states not used
- 8m-1 dynamic pricing for MTS only requires observed costs in states actually used.



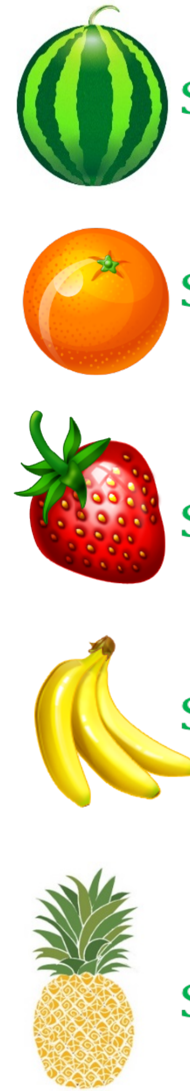
States of knowledge

Yet another dimension to the problem

Walrasian equilibrium



Prices are assigned to items



$5

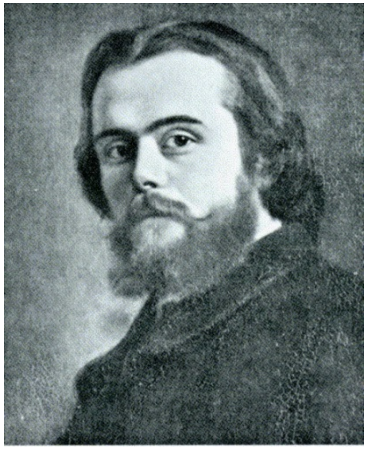
$2

$3

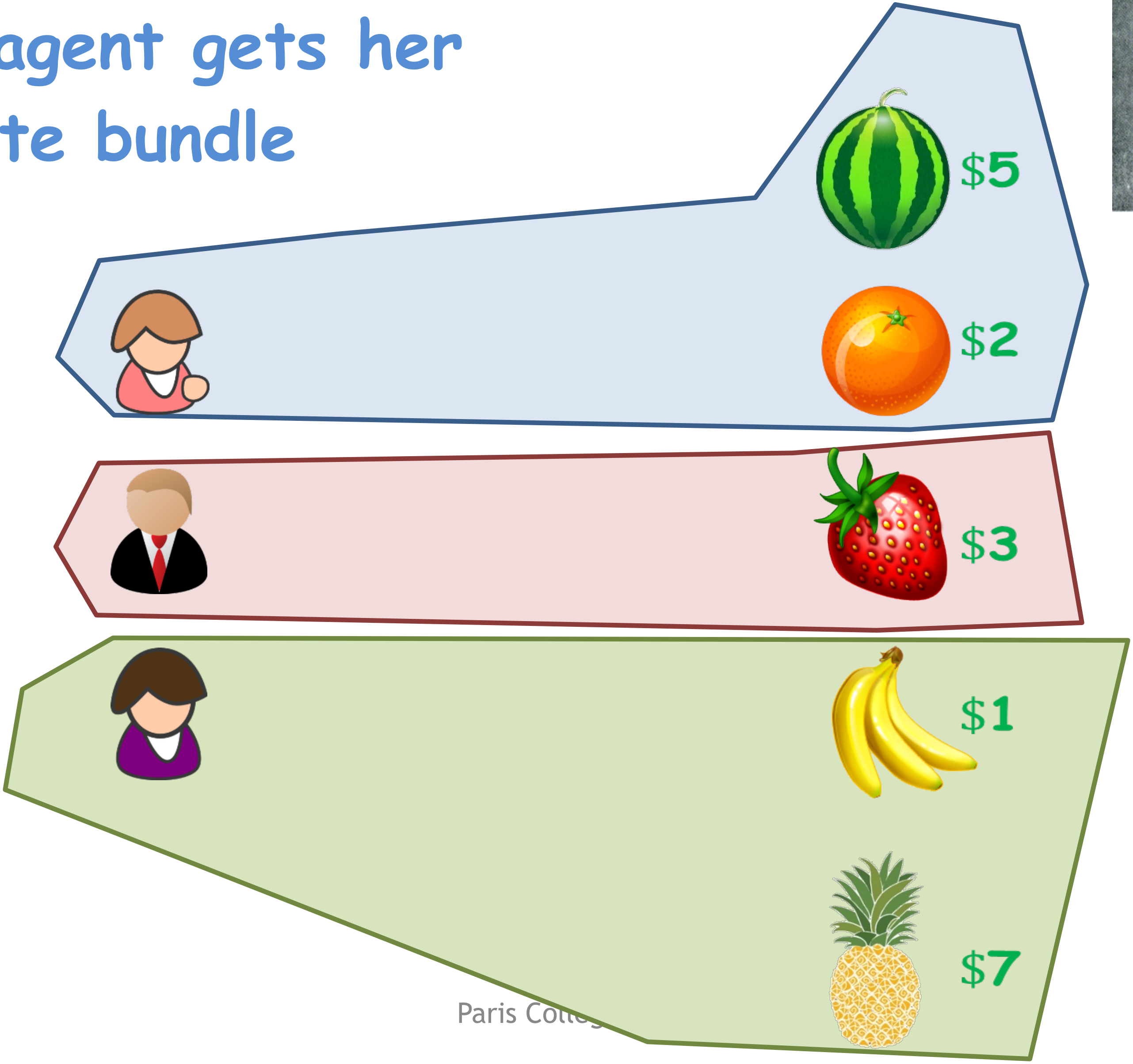
$1

$7

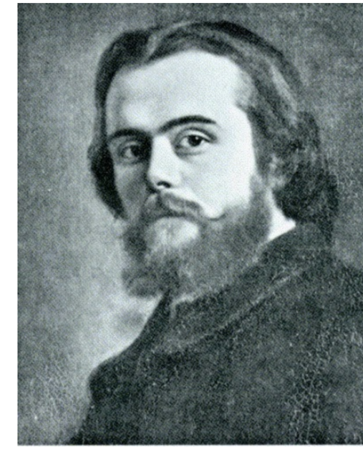
Walrasian equilibrium



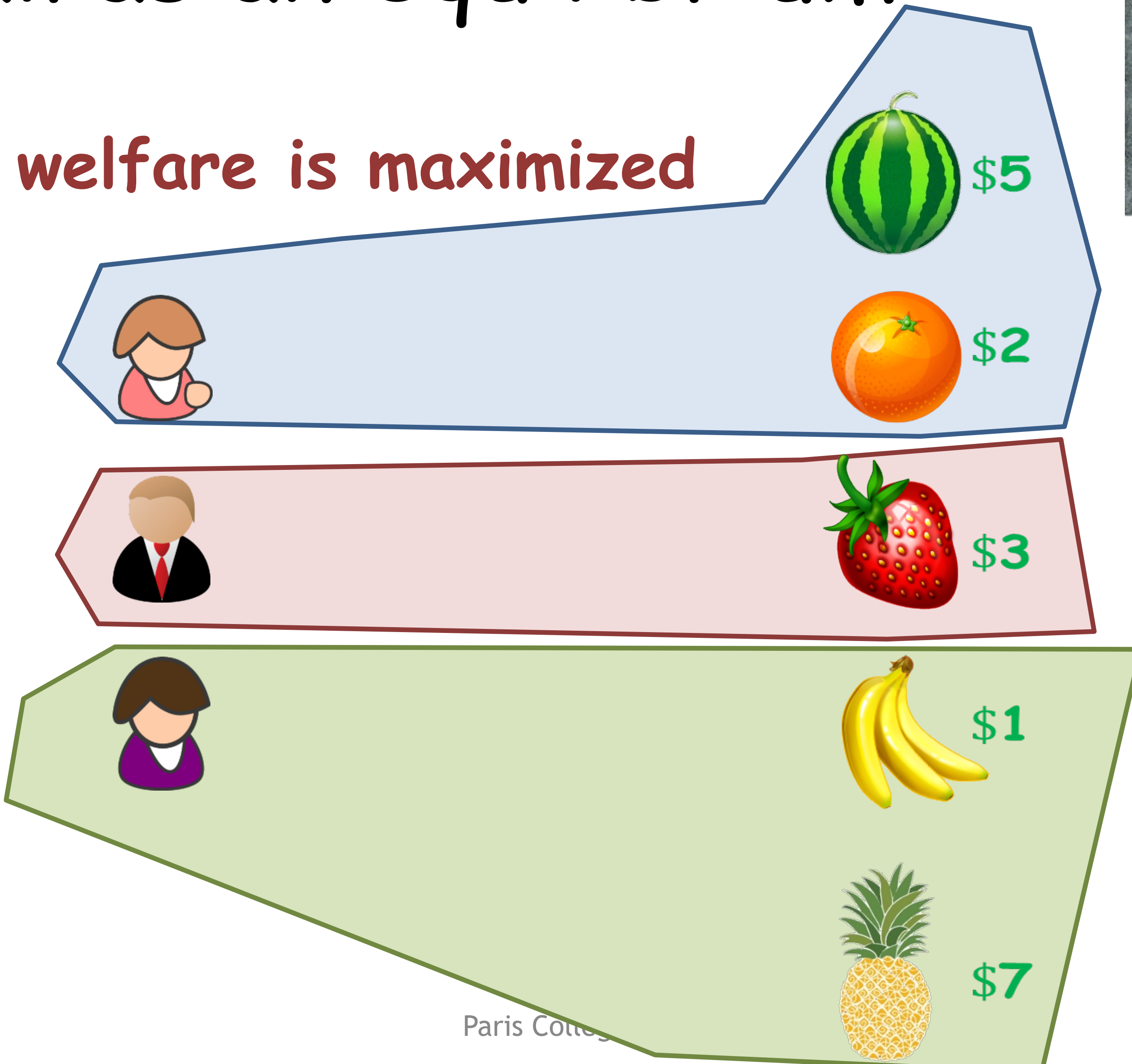
Each agent gets her favorite bundle



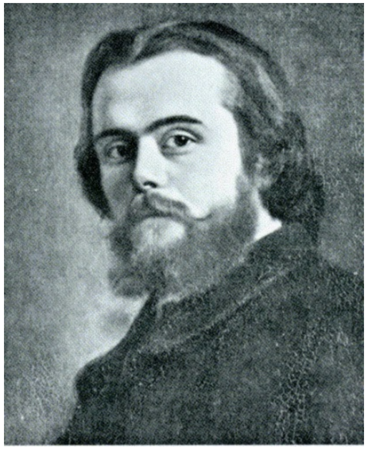
Walrasian equilibrium



Social welfare is maximized

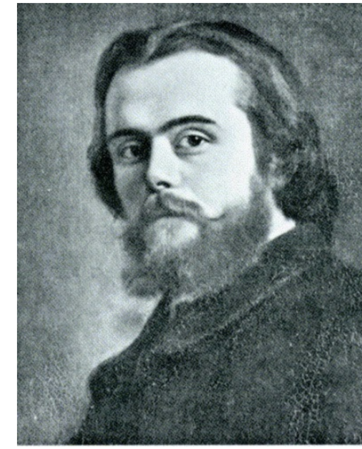


Walrasian prices

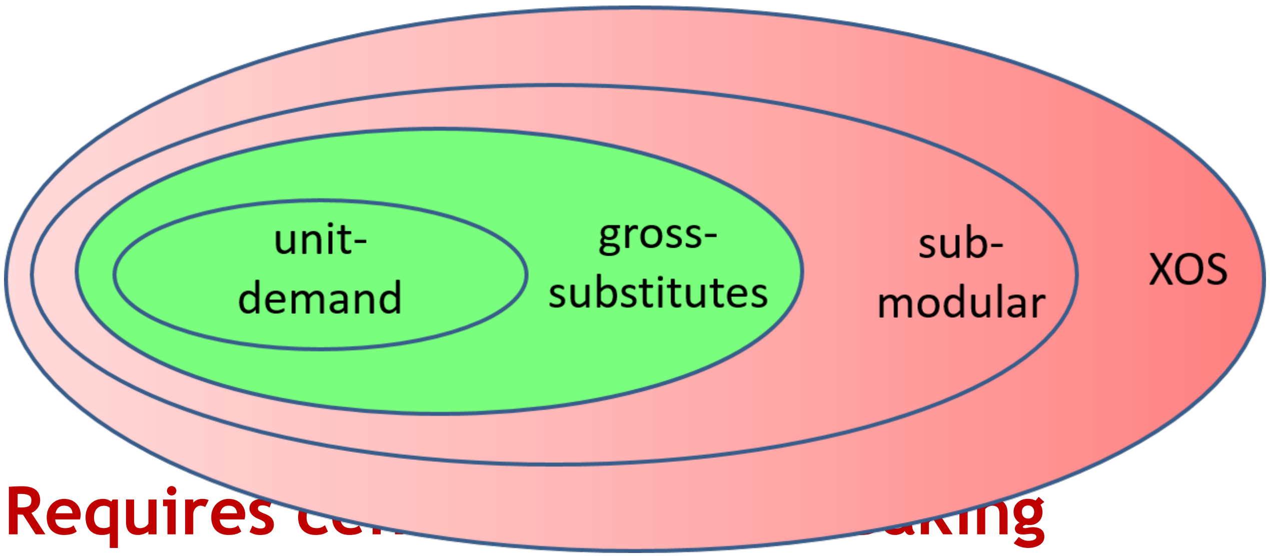


- Prices are assigned to items such that:
  - Every agent "gets" a utility-maximizing allocation.
  - **Welfare** is **maximized**.
- But…
  - Do not exist in general (guaranteed to exist for gross-substitutes).
  - Require centralized tie-breaking be done on behalf of agents.

Walrasian equilibrium



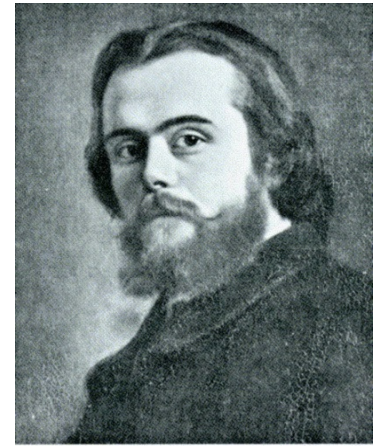
Guaranteed to exist for **Gross Substitutes**
[Kelso and Crawford 1982]

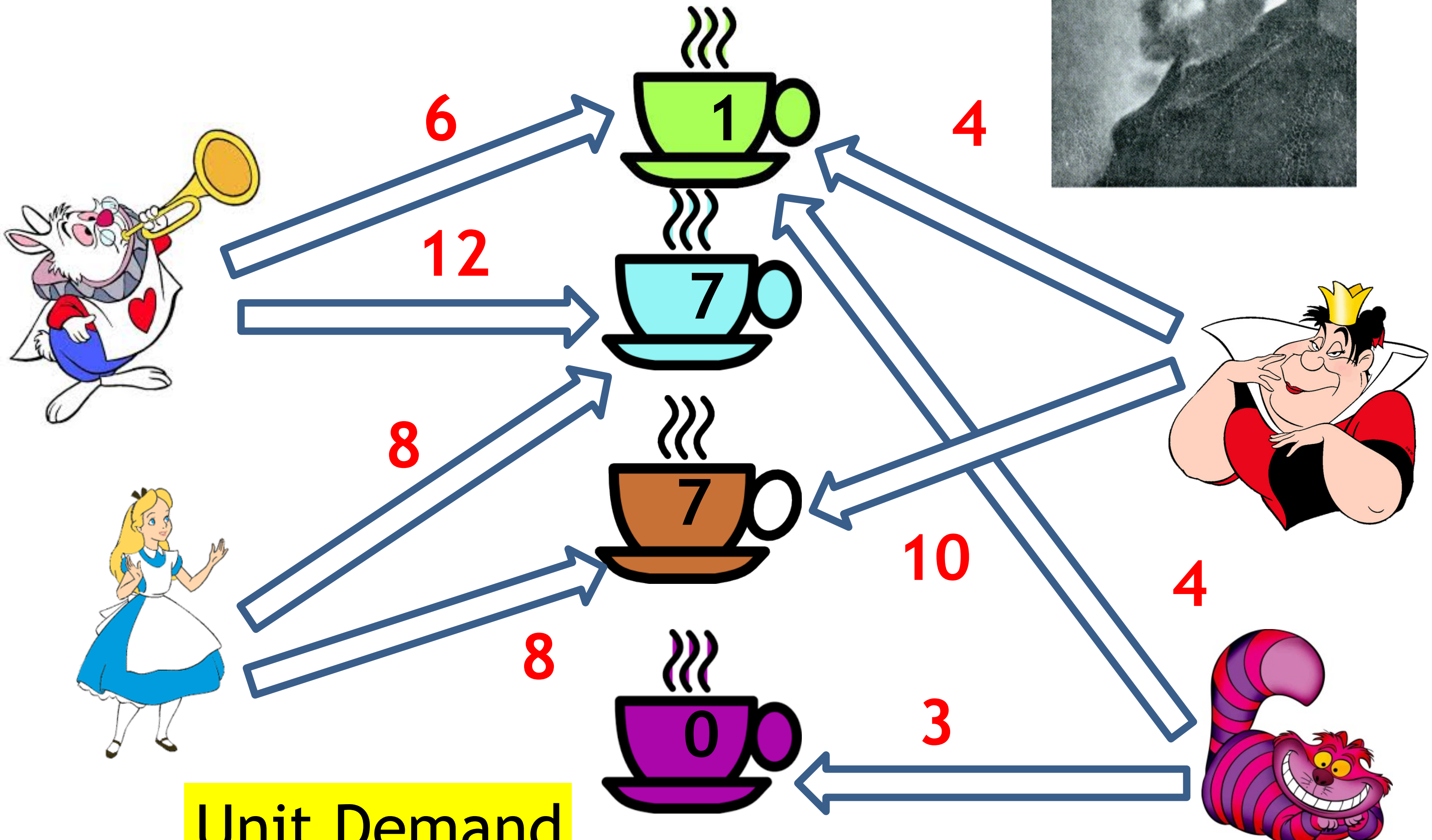


Requires ce ... aking

Question: Can the market be coordinated in a **transparent** way (without forcing decisions)?

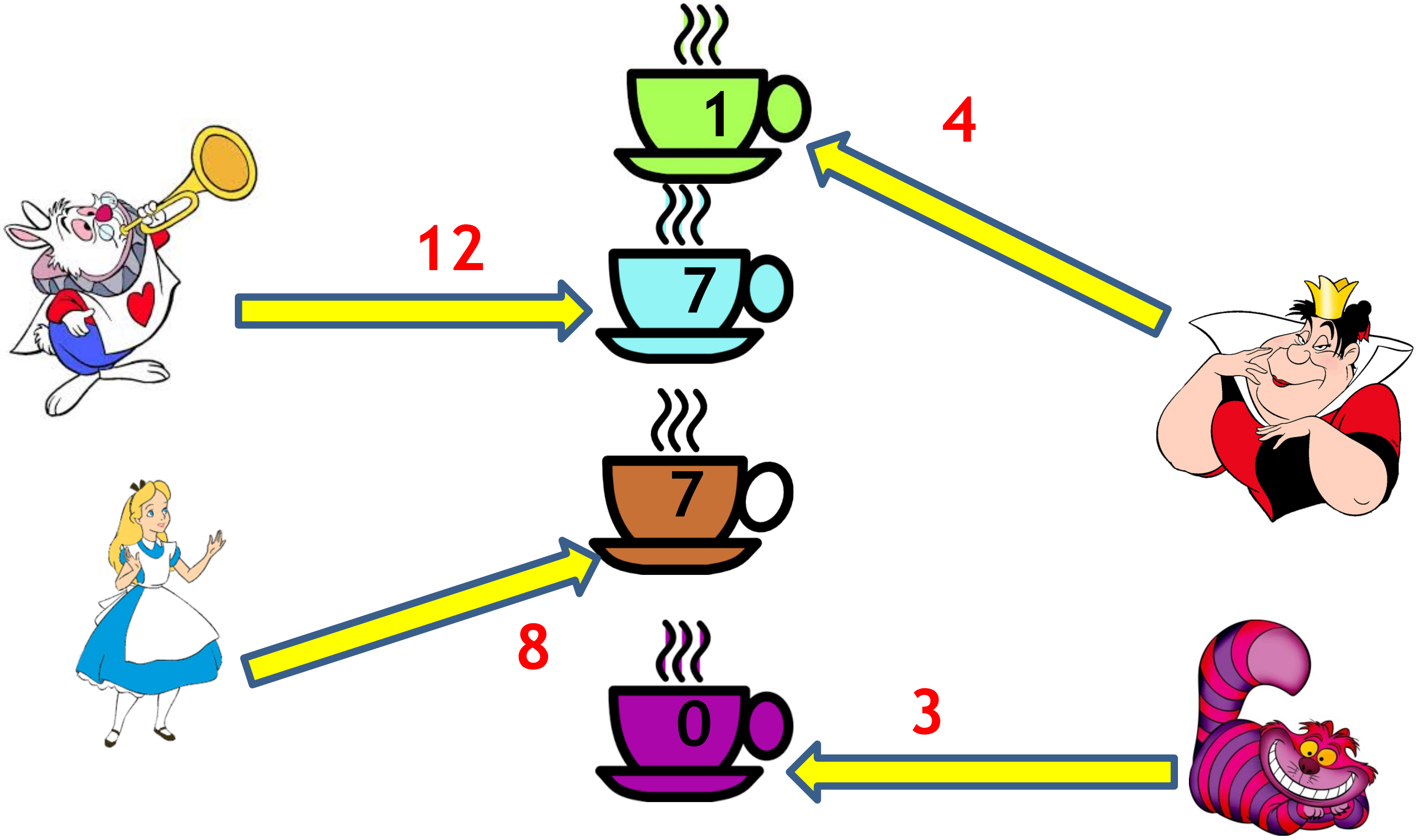
Walrasian Pricing



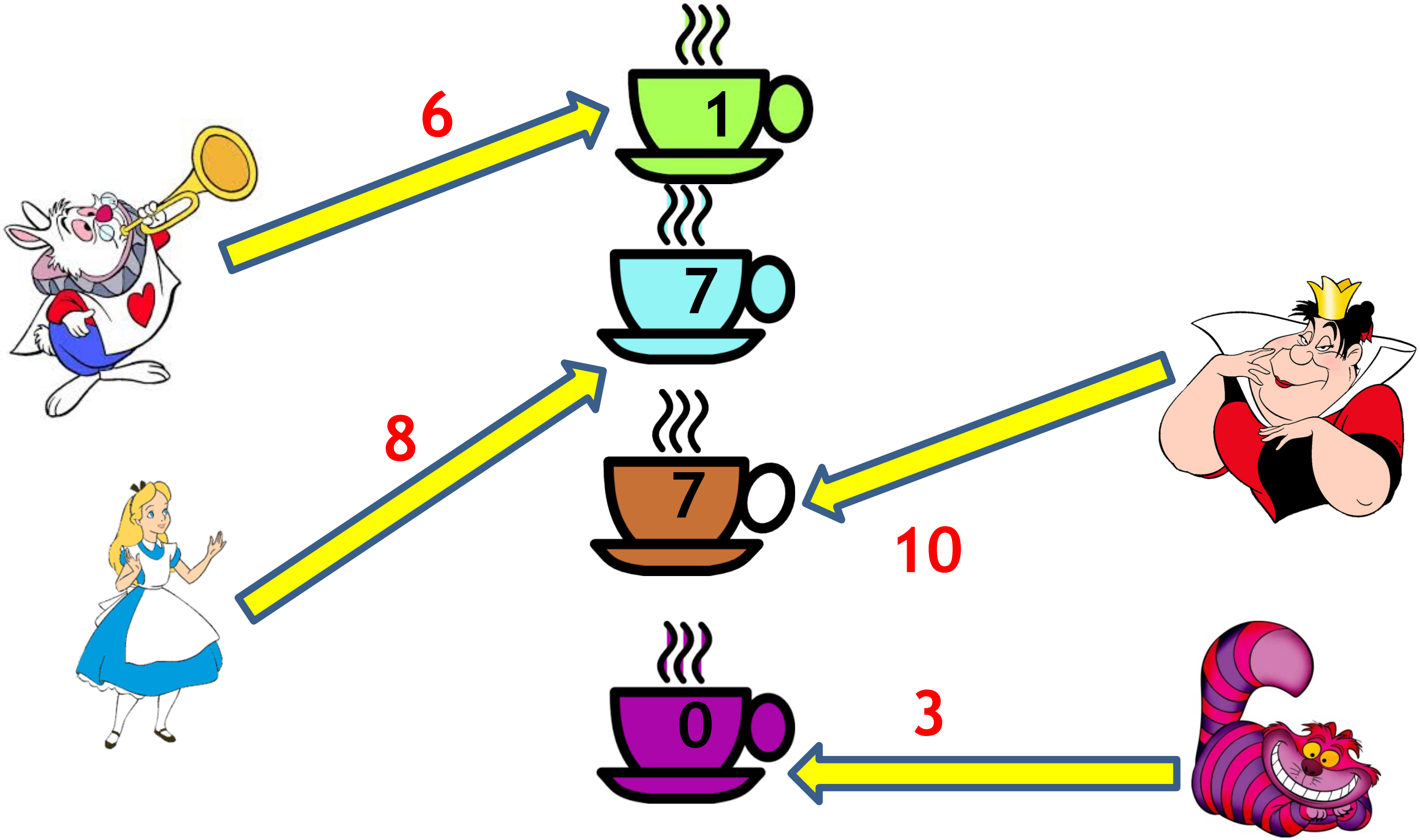


Unit Demand

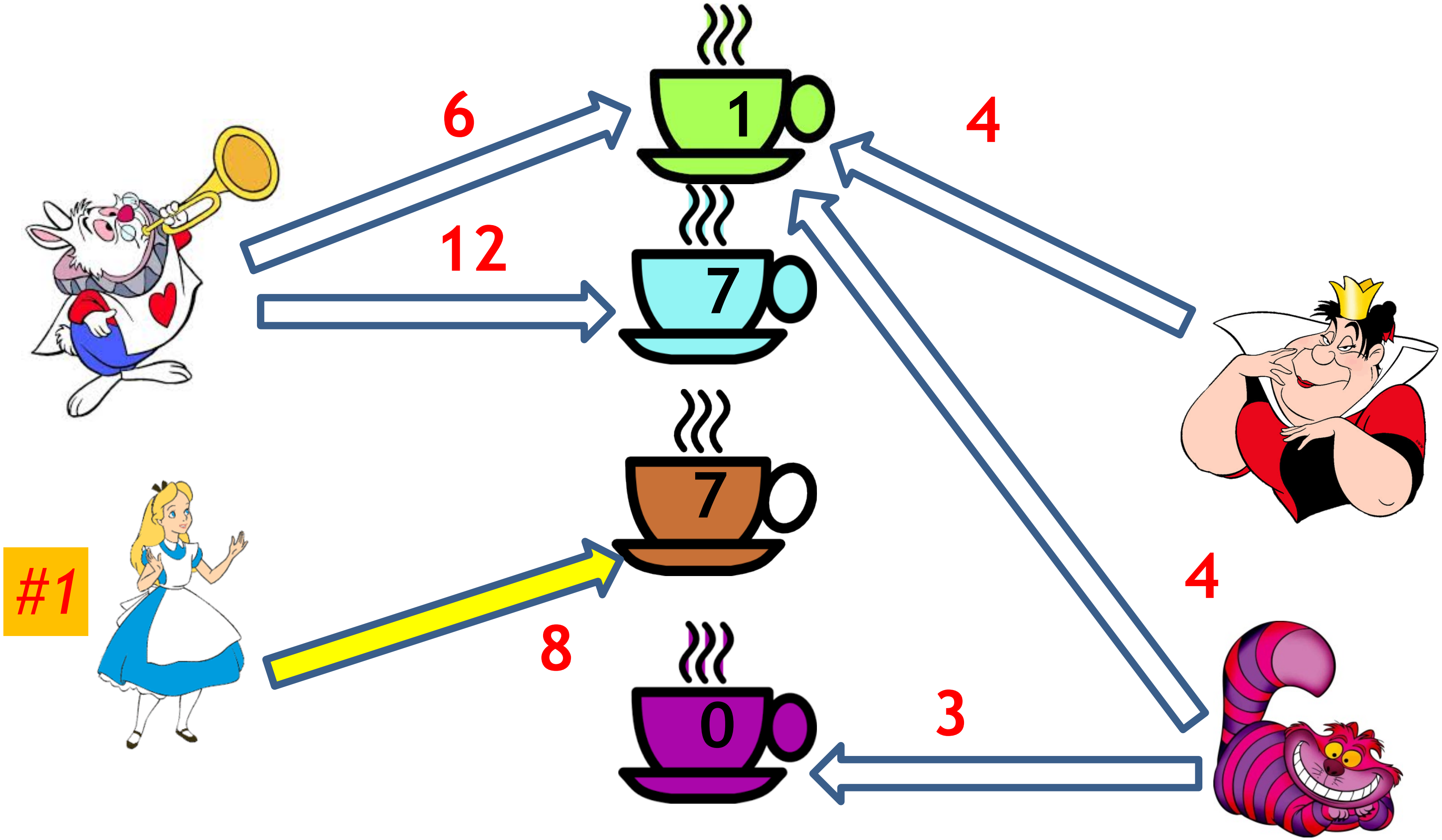
Market Clears, Welfare Maximized



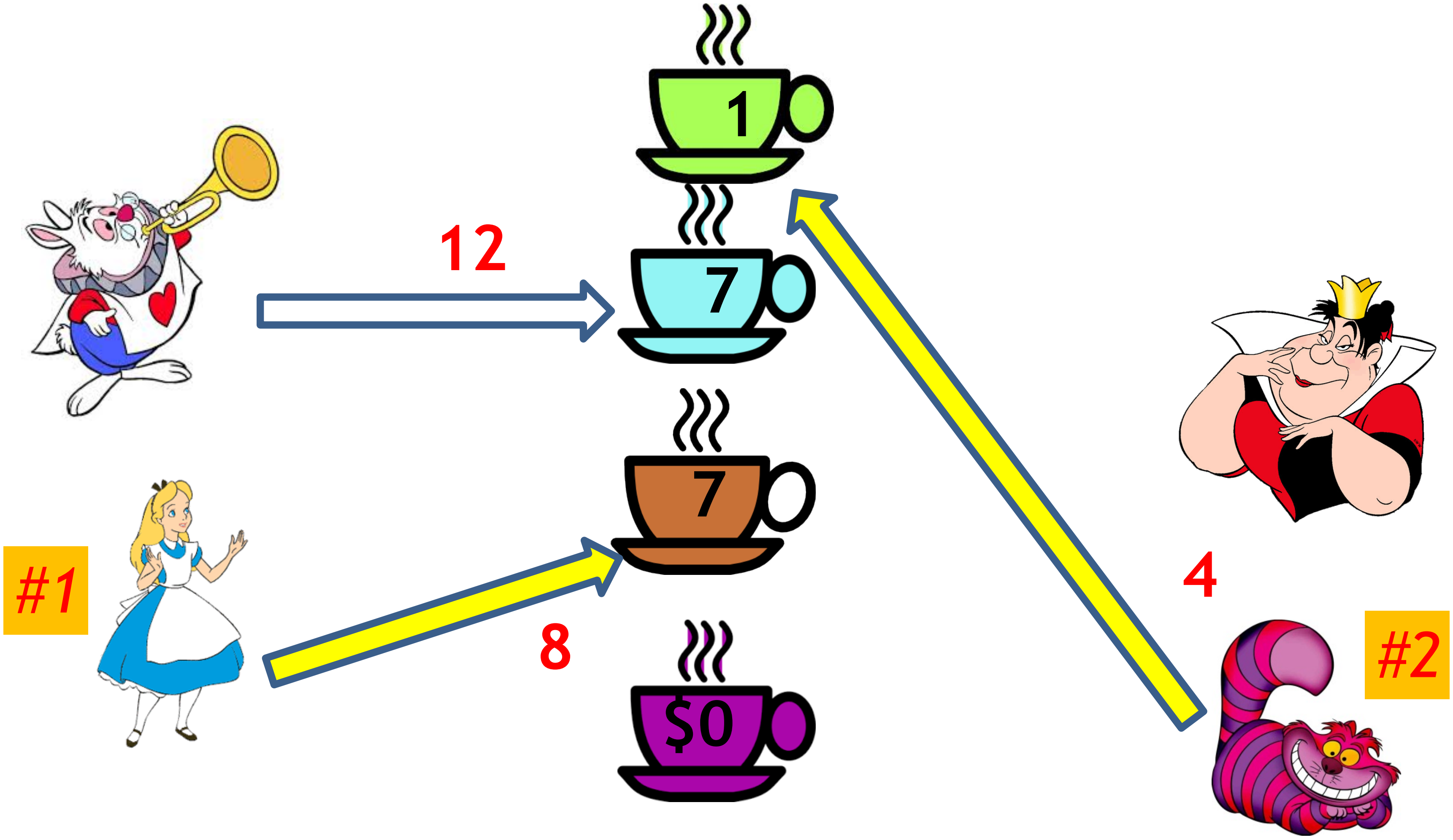
A different allocation



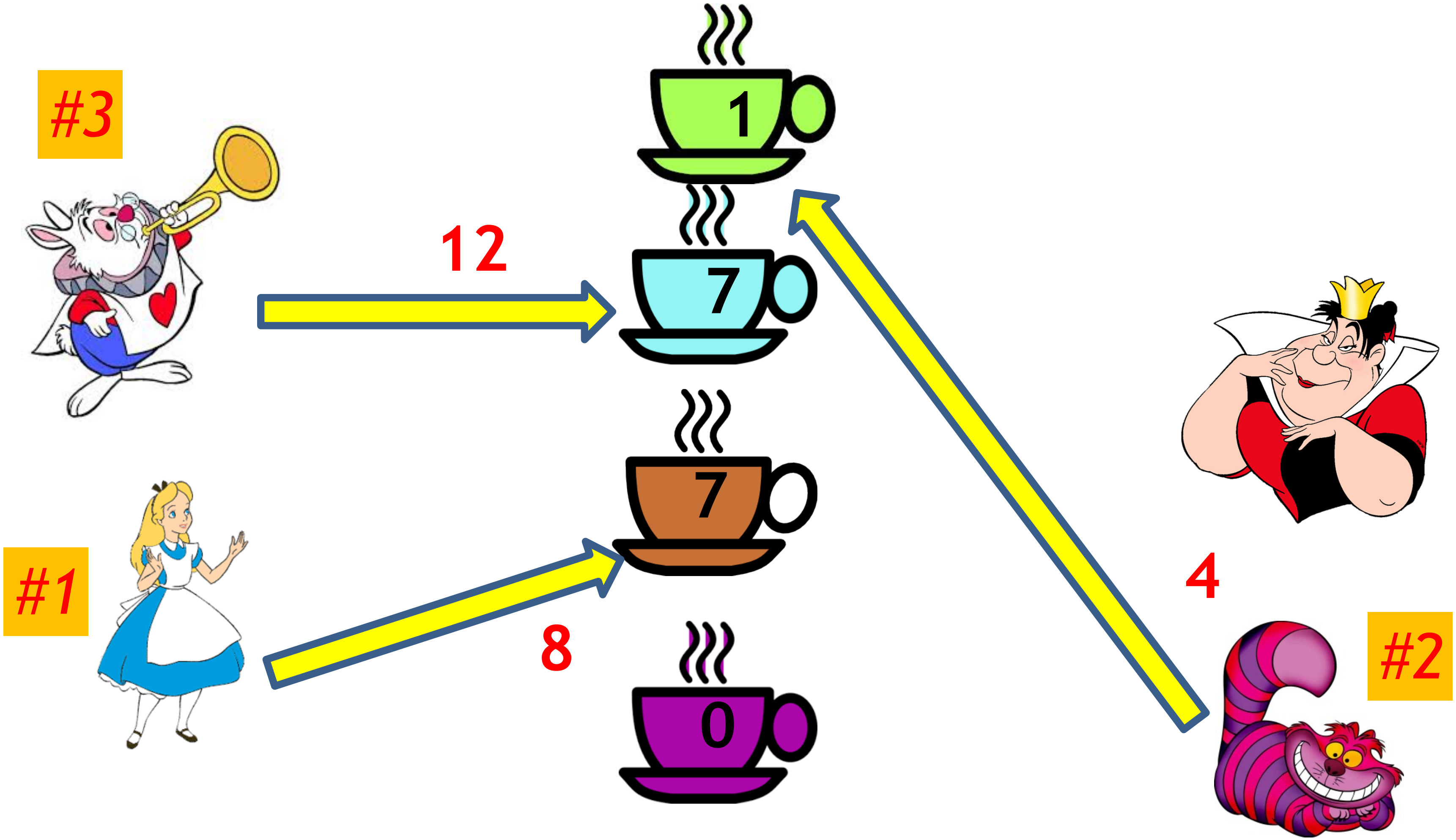
Wrong Arrival Order for Walras



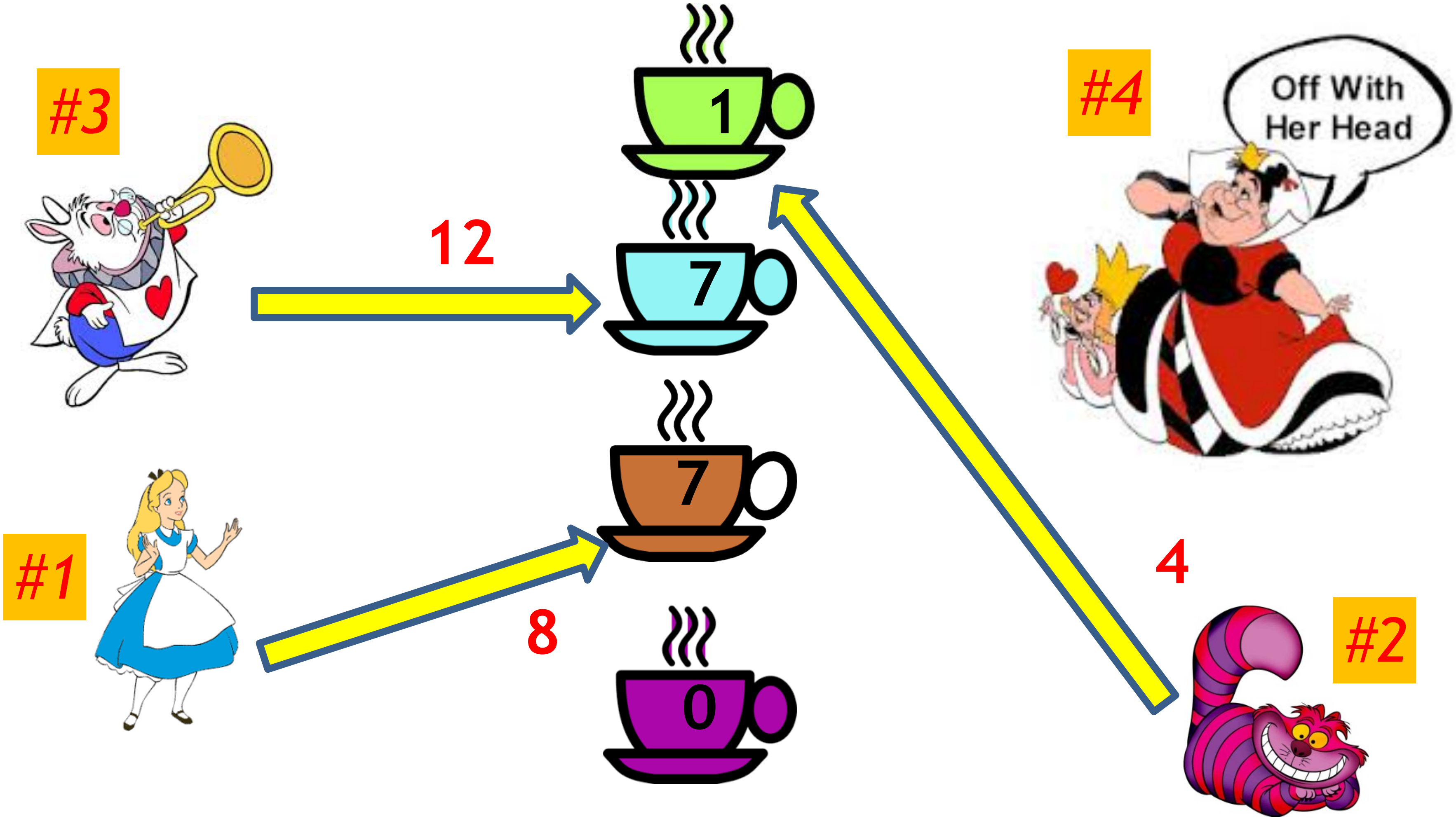
Wrong Arrival Order for Walras



Wrong Arrival Order for Walras

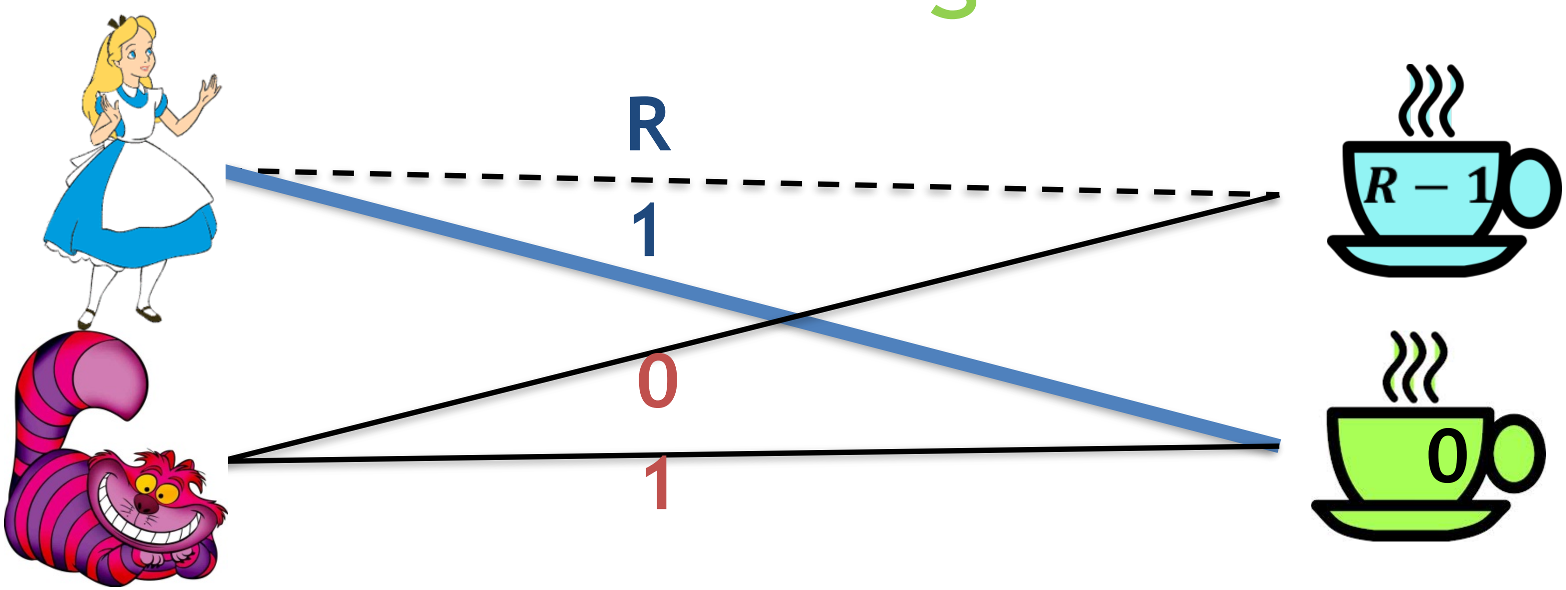


Wrong Arrival Order for Walras



Walrasian prices might be very bad

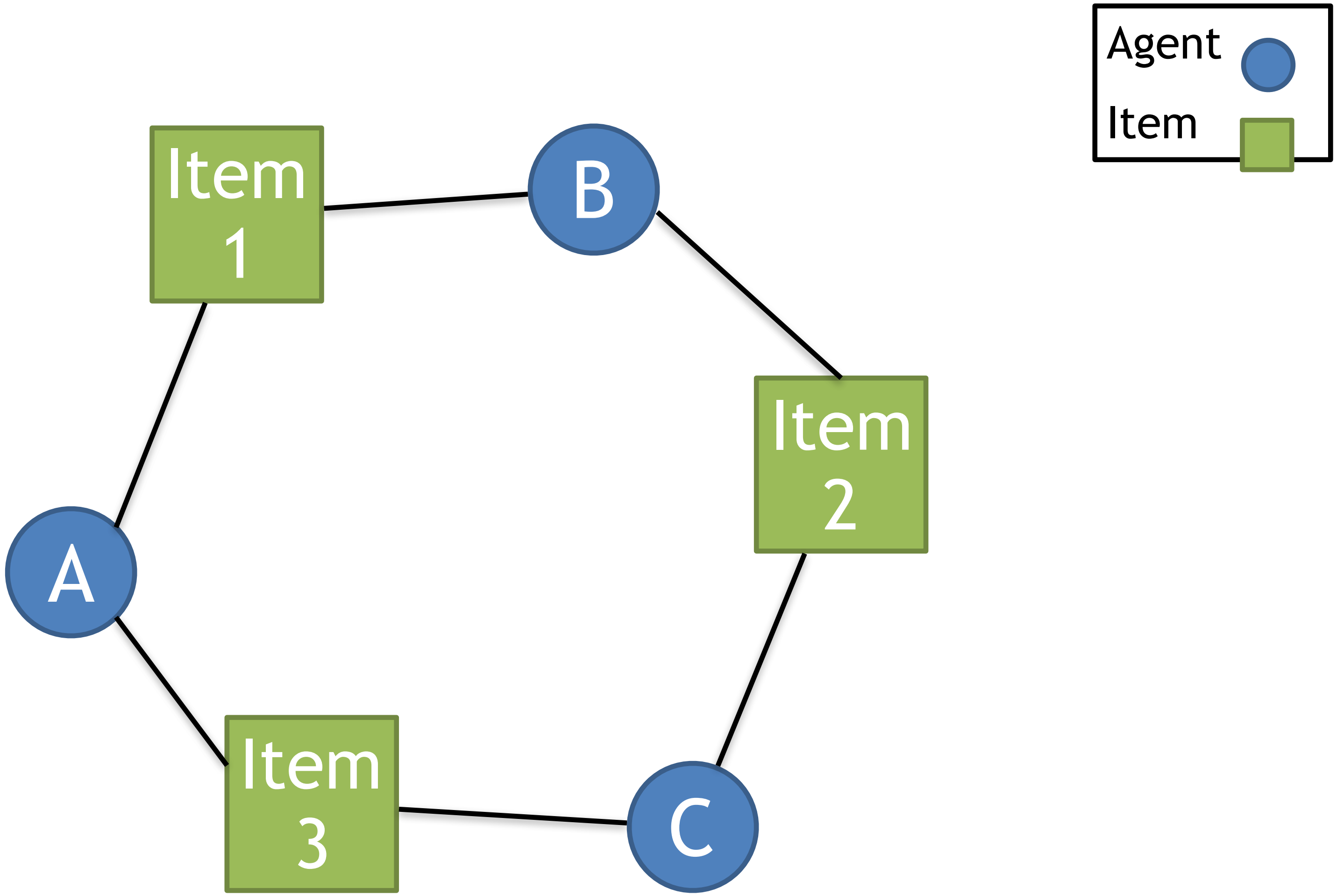
Alice takes green



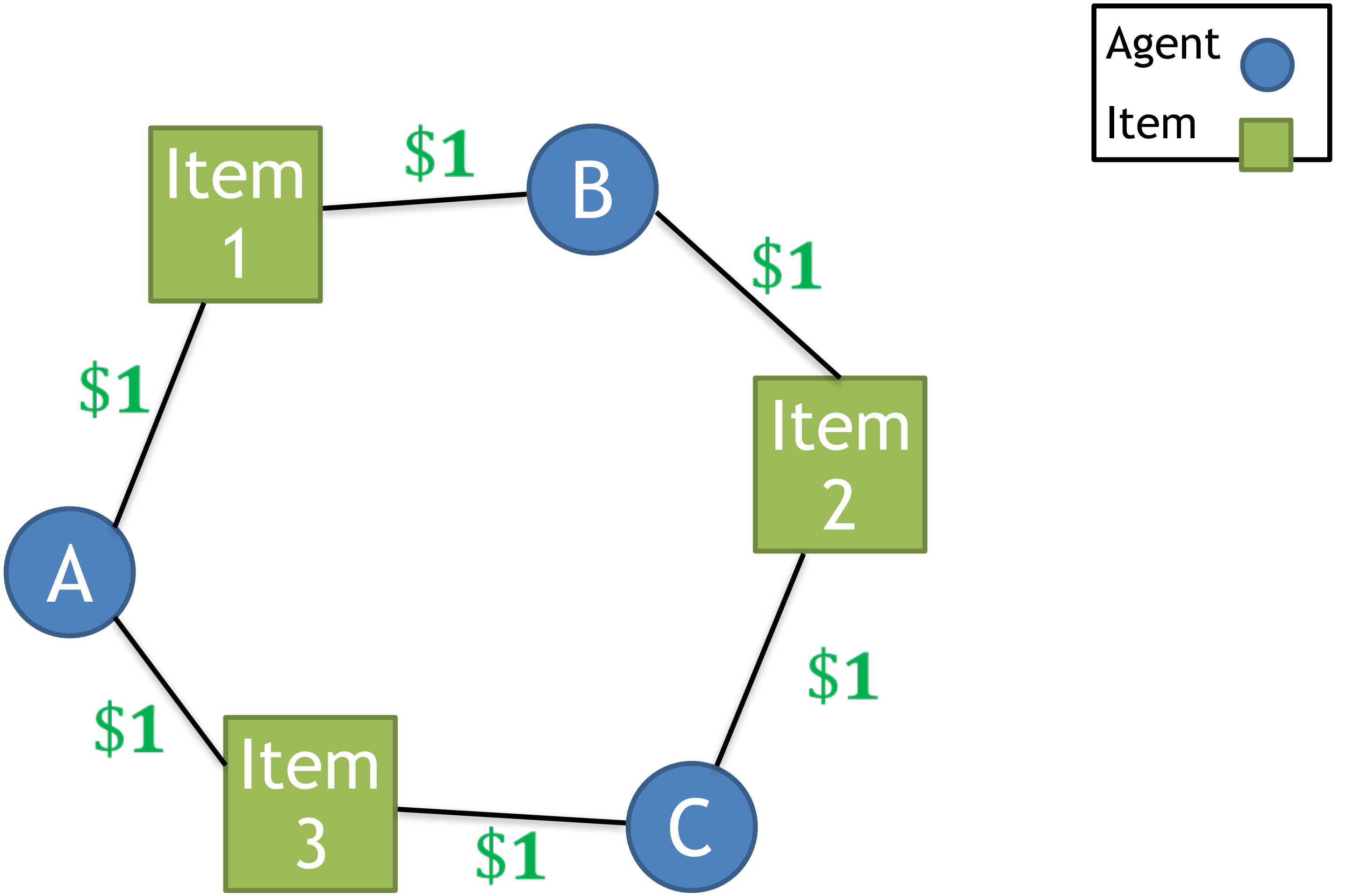
Bob takes nothing

Social welfare of 1 (instead of R+1)

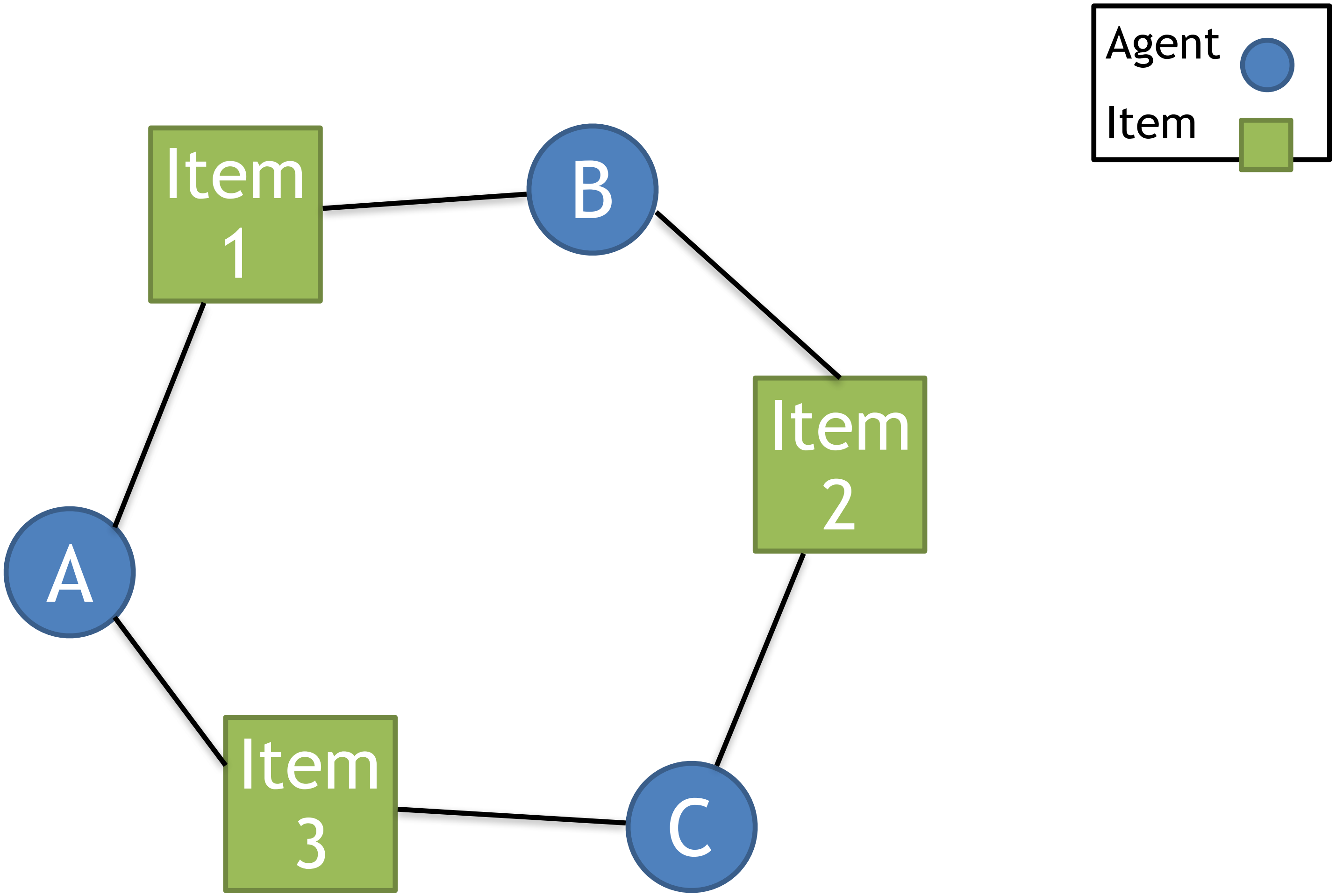
Static prices cannot coordinate the market



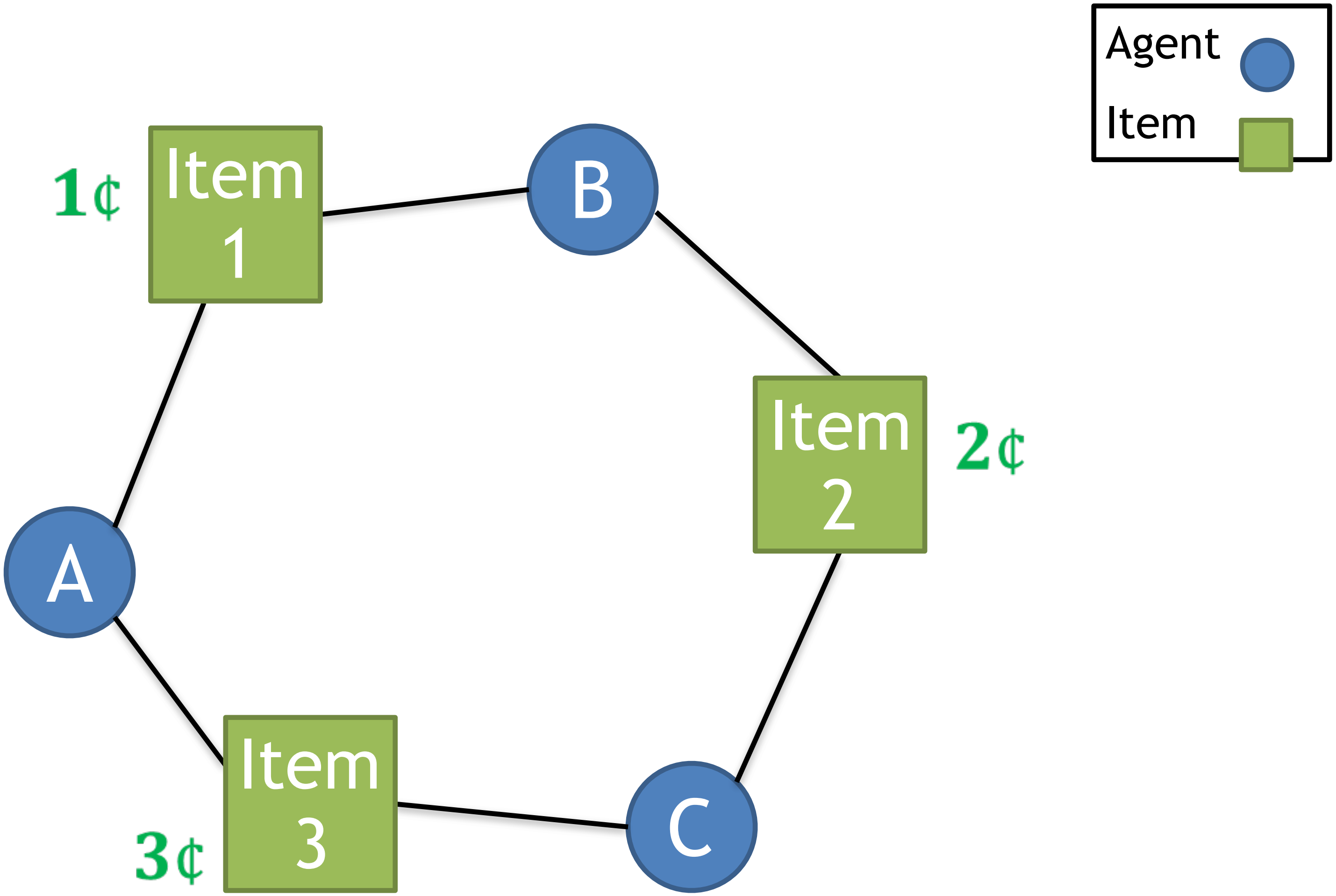
Static prices cannot coordinate the market



Static prices cannot coordinate the market

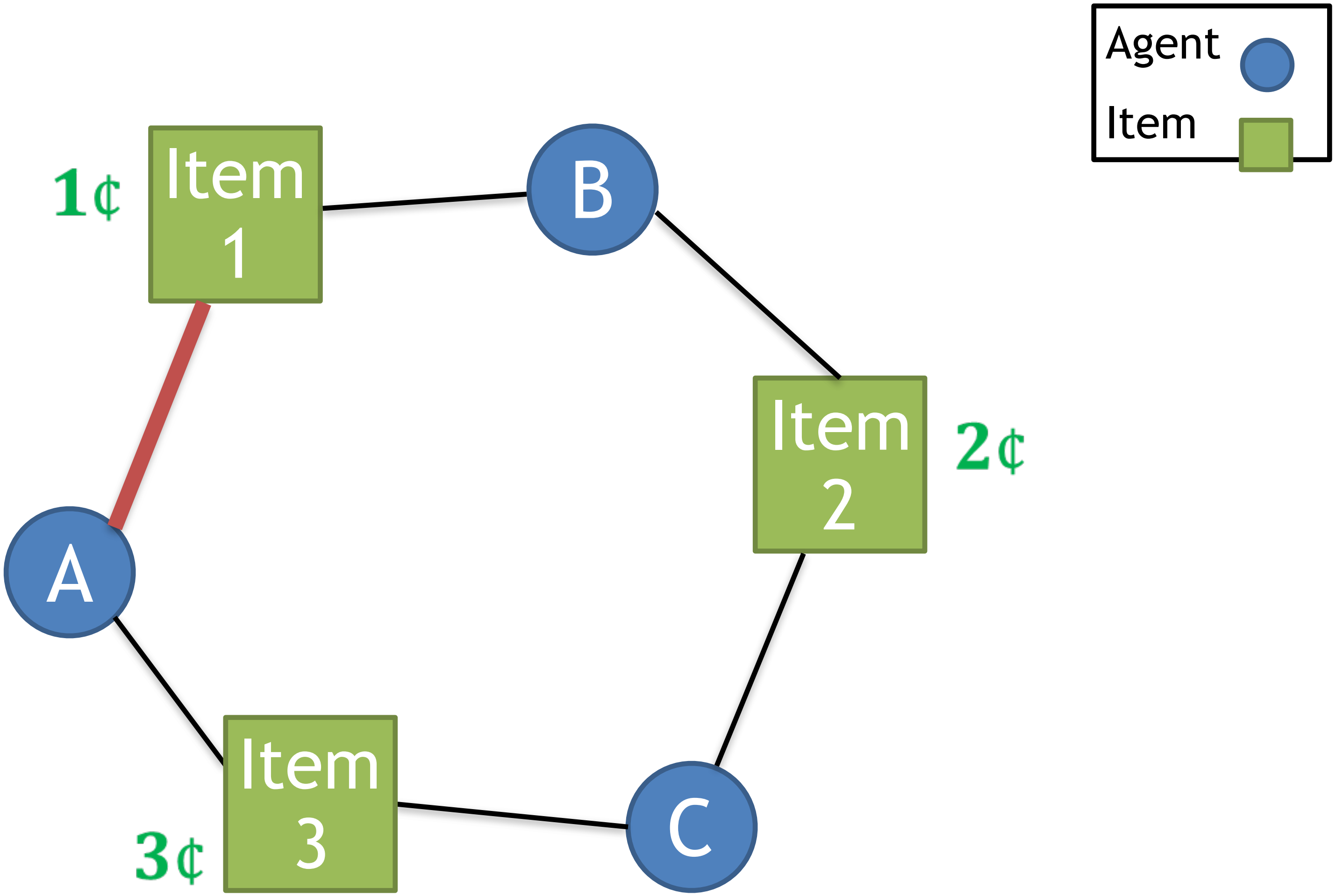


Static prices cannot coordinate the market



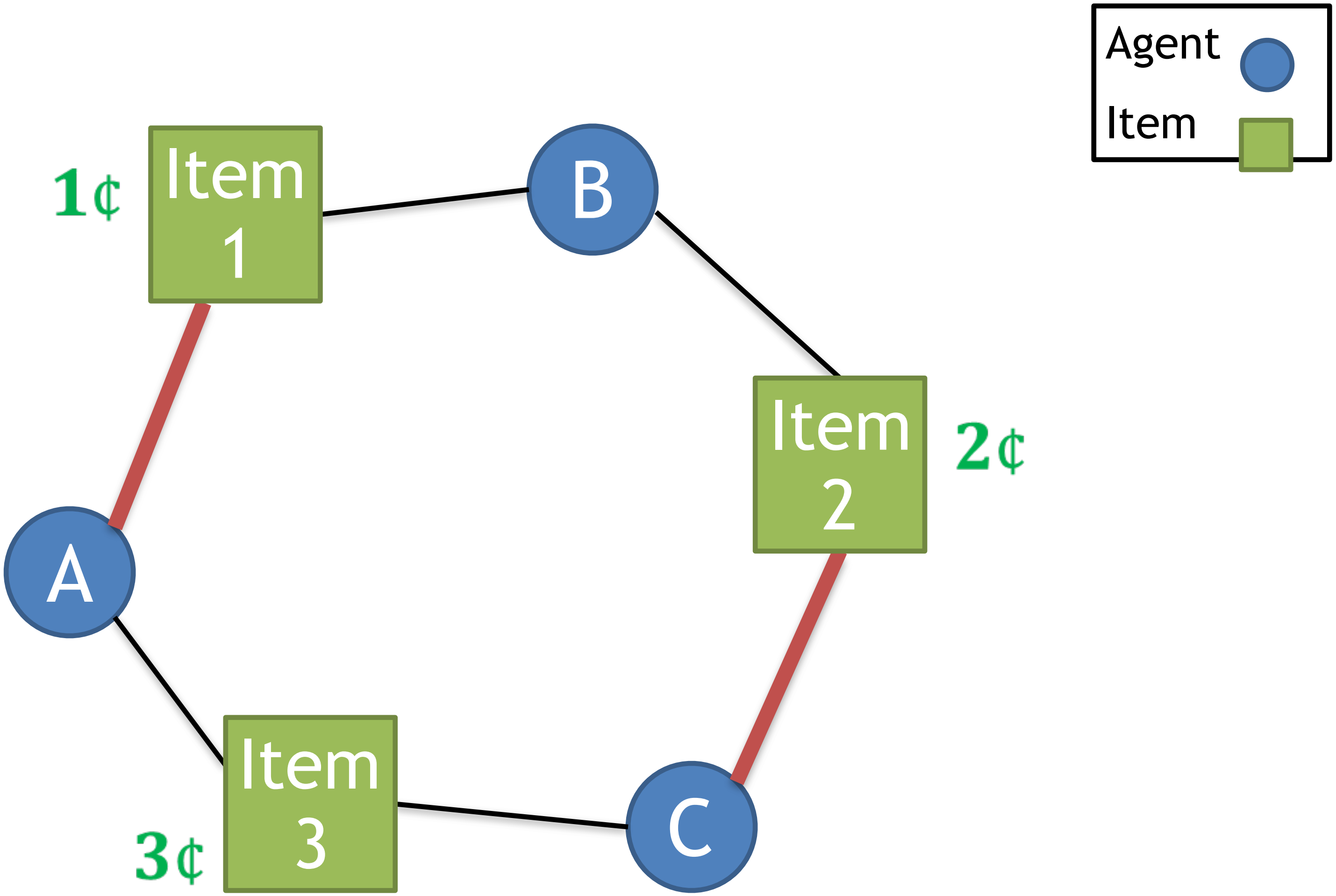
$$\text{WLOG price}(1) \leq \text{price}(2) \leq \text{price}(3)$$

Static prices cannot coordinate the market



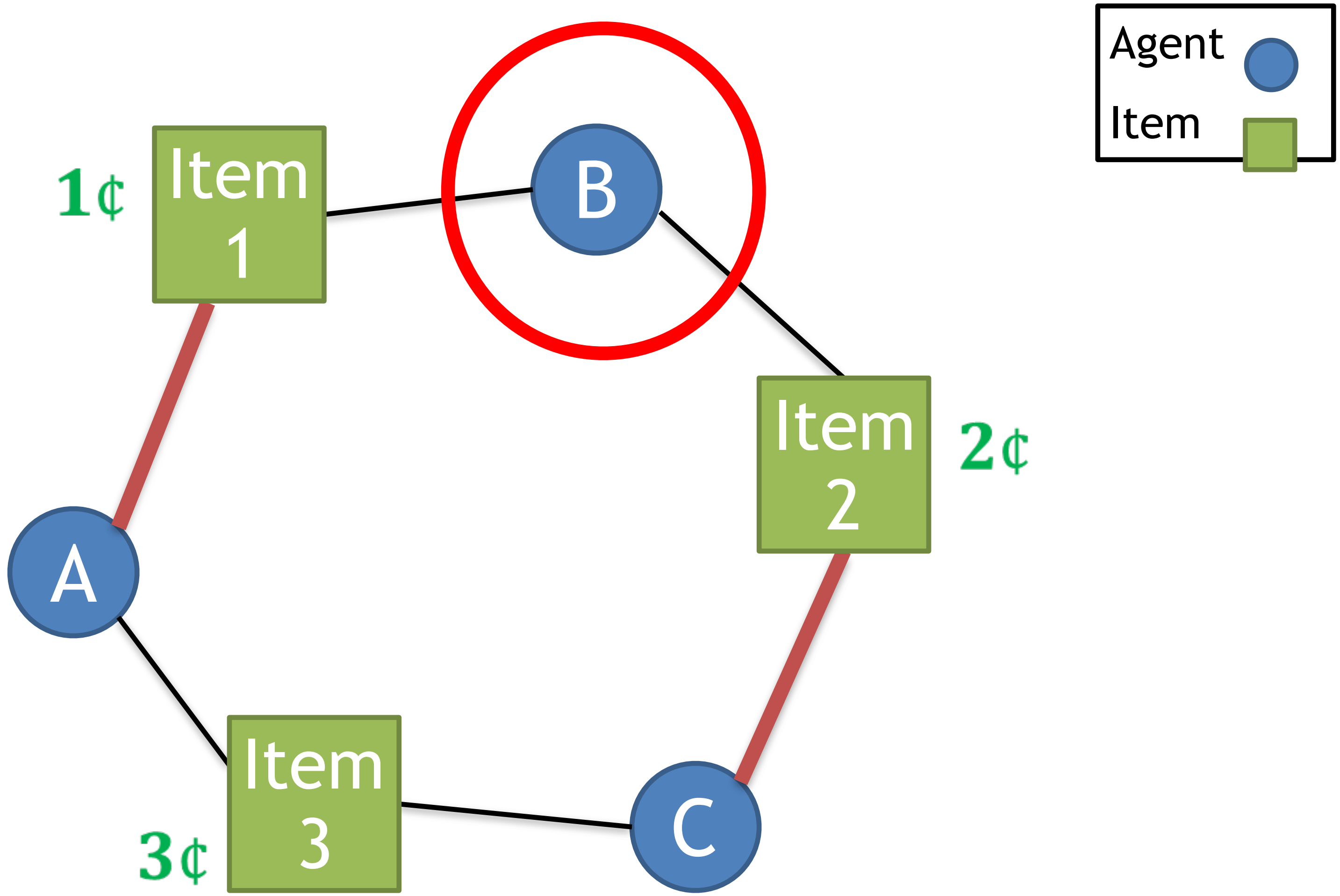
$$\text{WLOG price}(1) \leq \text{price}(2) \leq \text{price}(3)$$

Static prices cannot coordinate the market



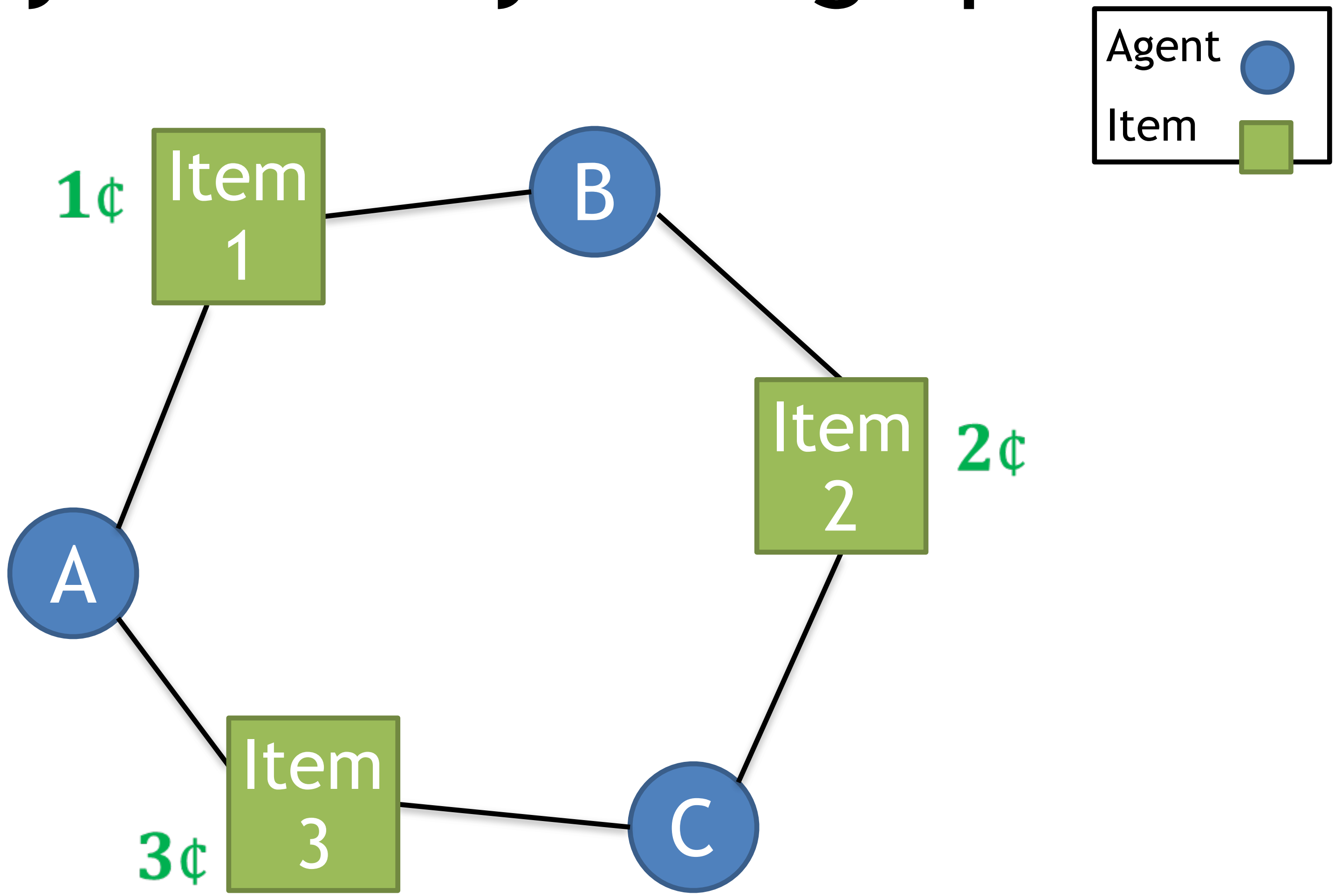
$$\text{WLOG price}(1) \leq \text{price}(2) \leq \text{price}(3)$$

Static prices cannot coordinate the market

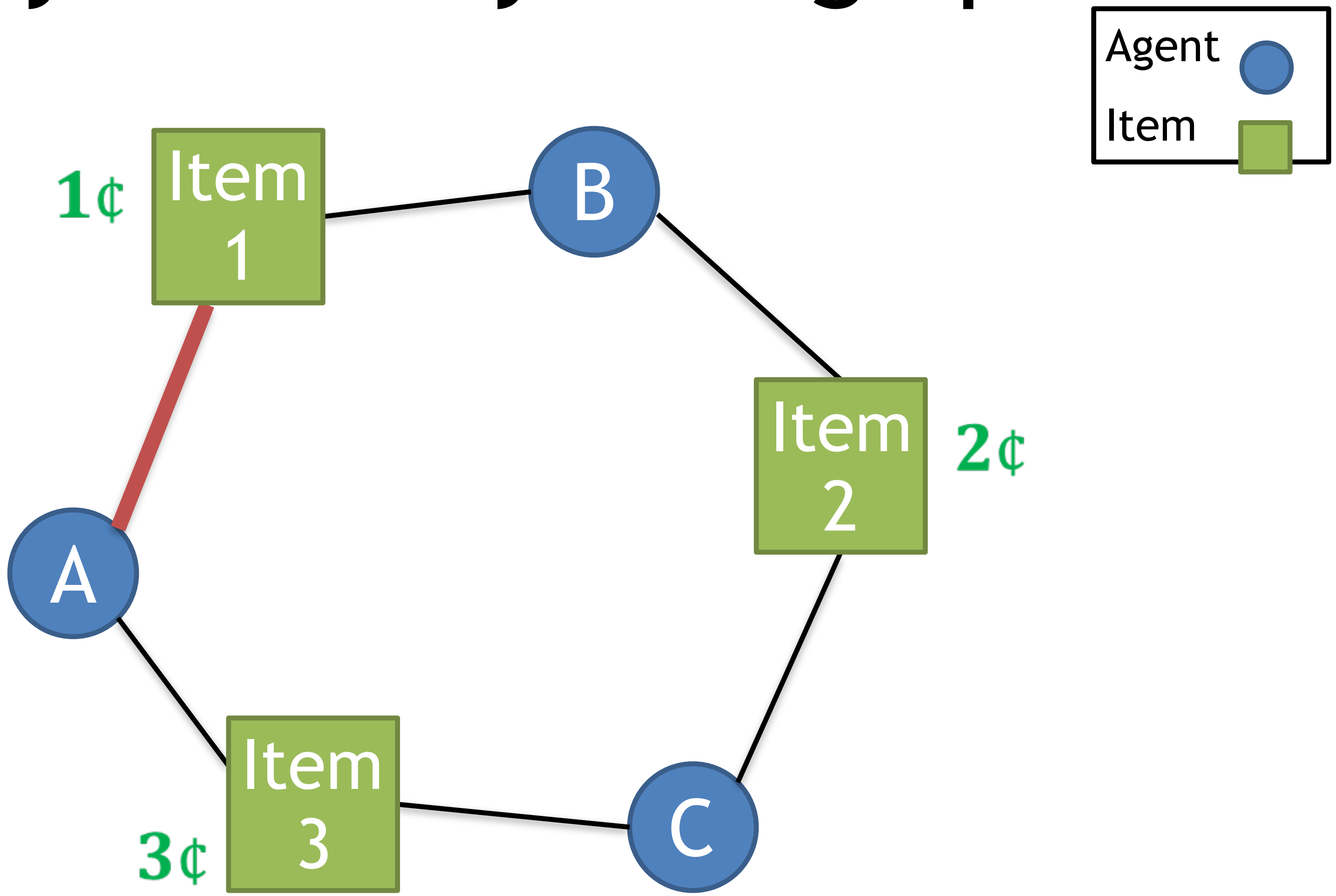


No static prices gurantee more than $\frac{2}{3}\mathrm{OPT}$

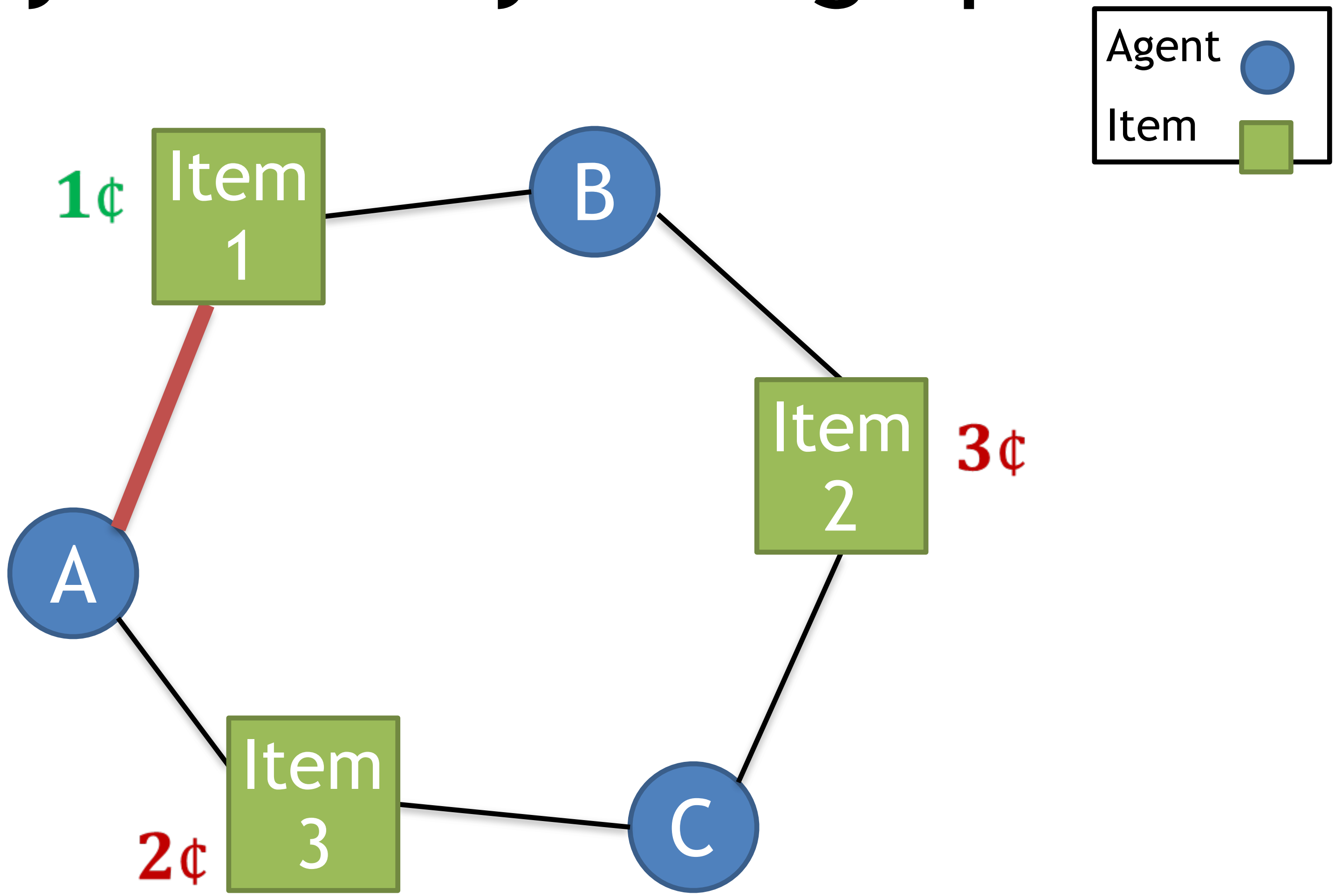
Idea: dynamically change prices



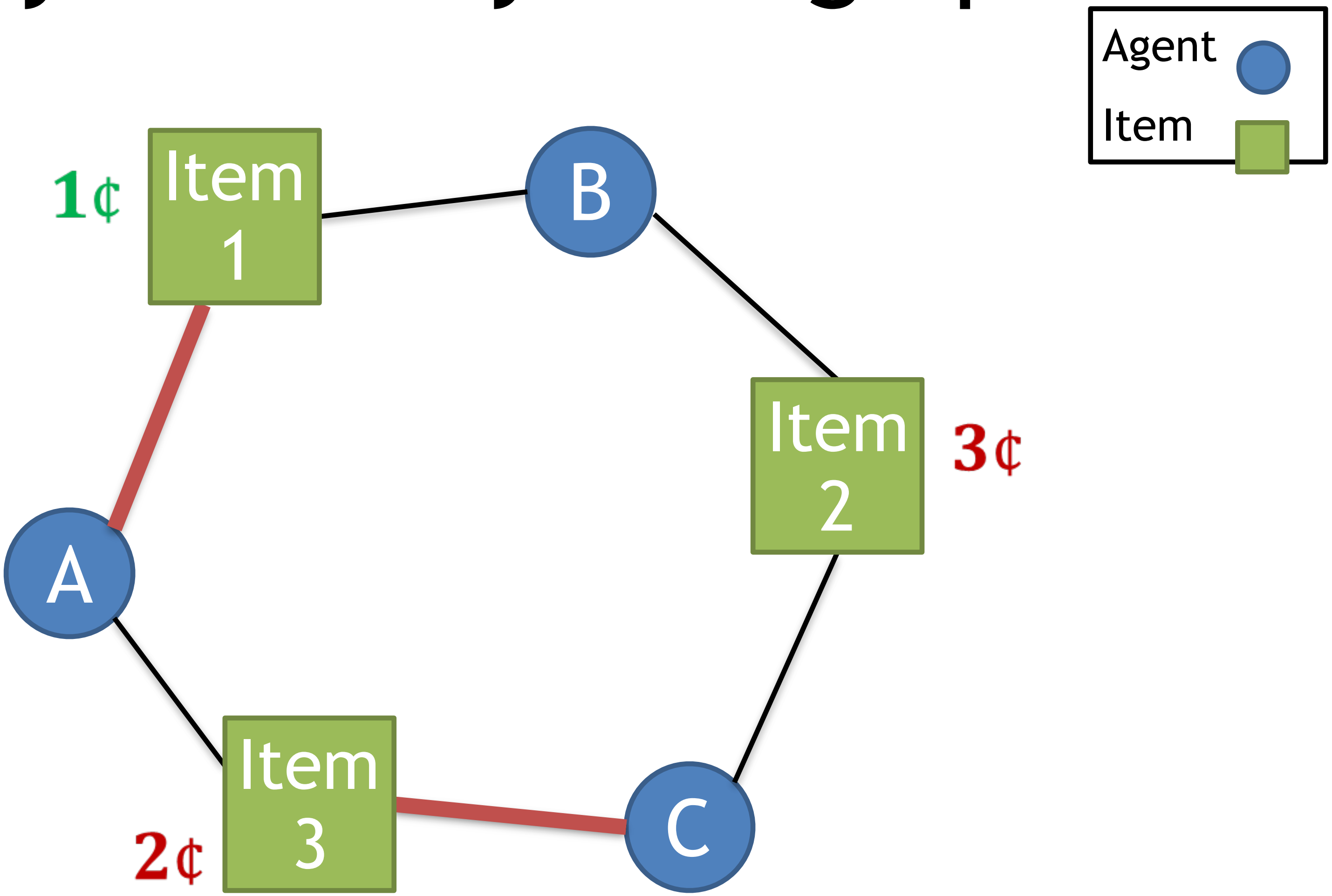
Idea: dynamically change prices



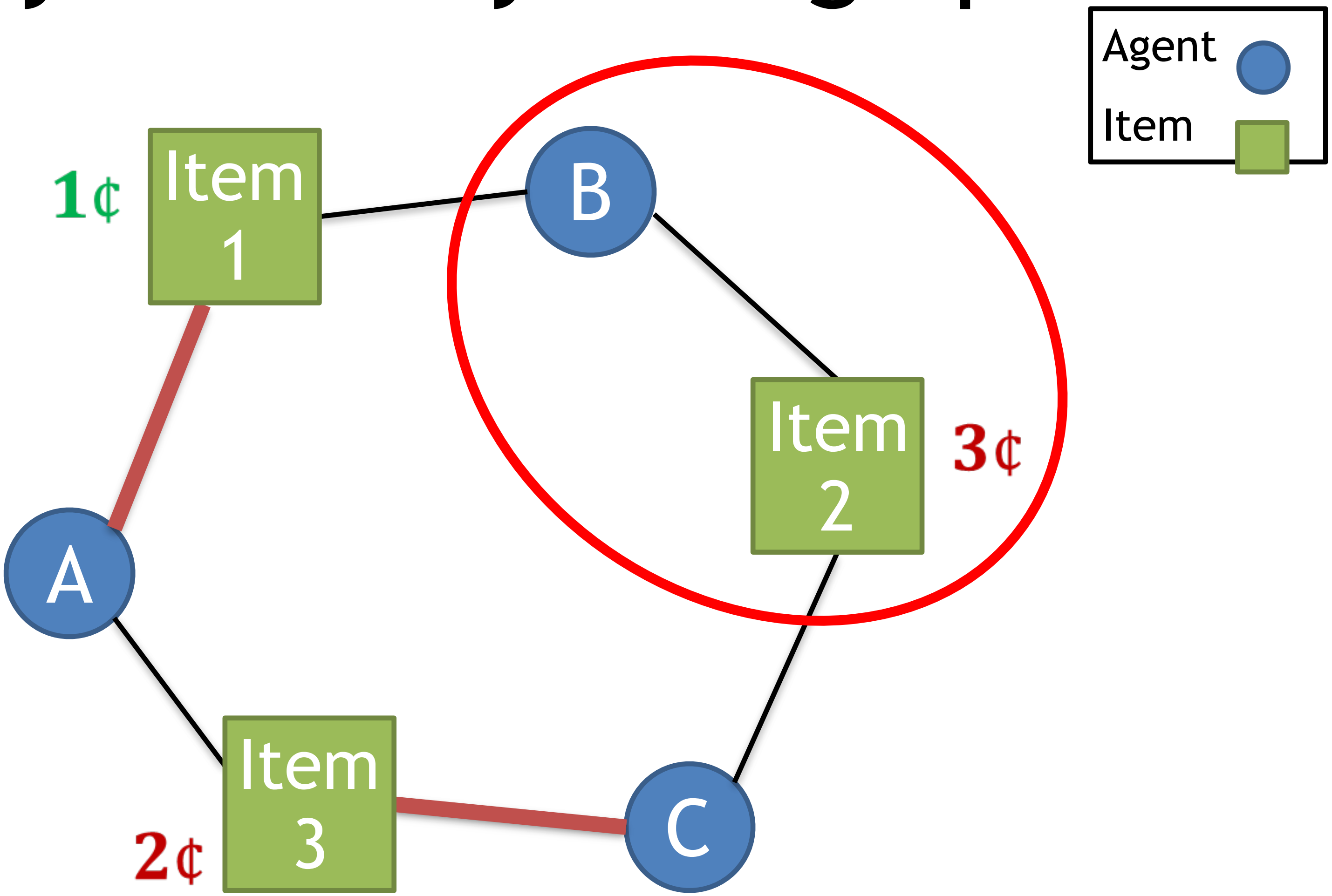
Idea: dynamically change prices



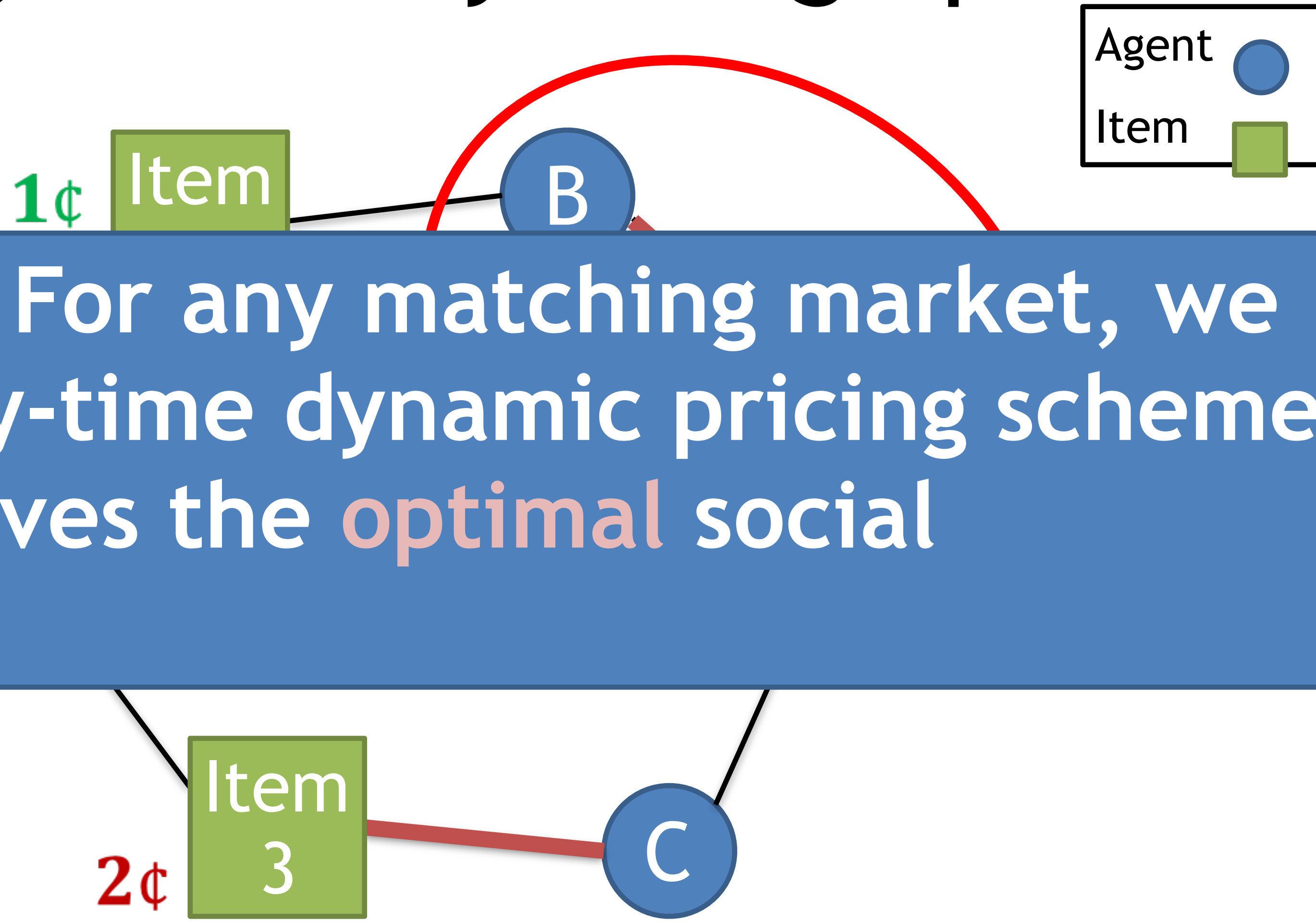
Idea: dynamically change prices



Idea: dynamically change prices



Idea: dynamically change prices

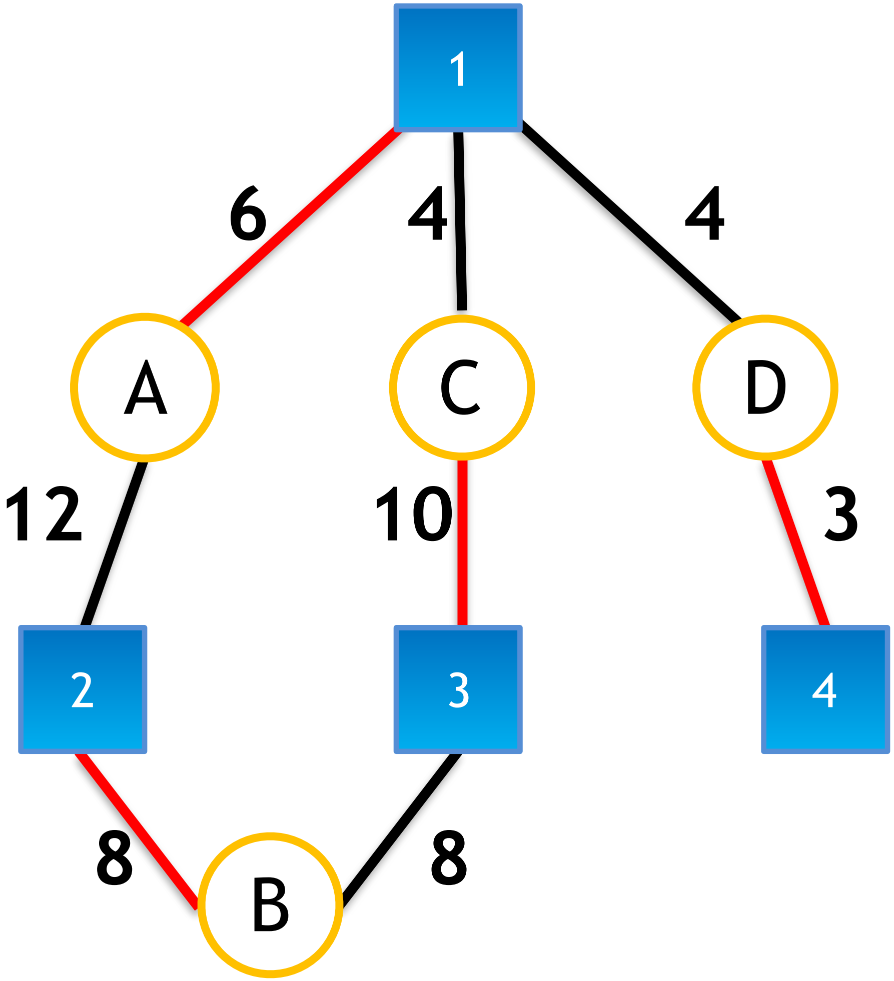


Theorem. For any matching market, we give a poly-time dynamic pricing scheme that achieves the optimal social welfare.

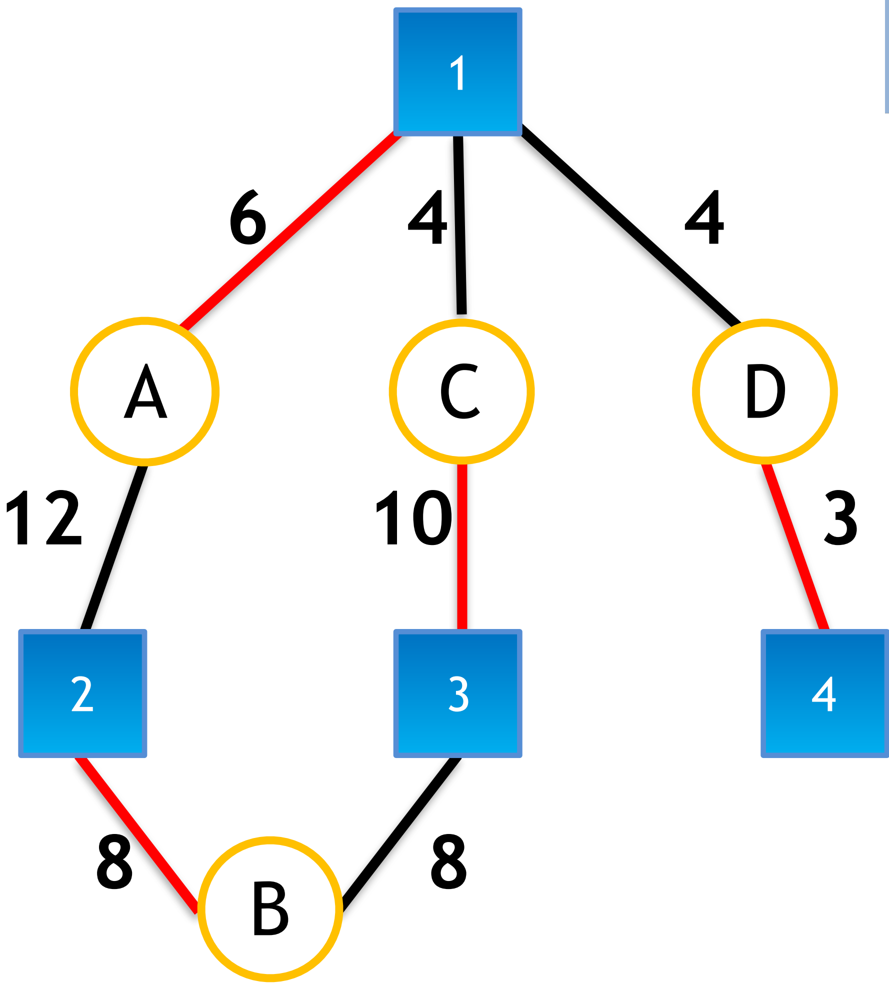
Algorithm

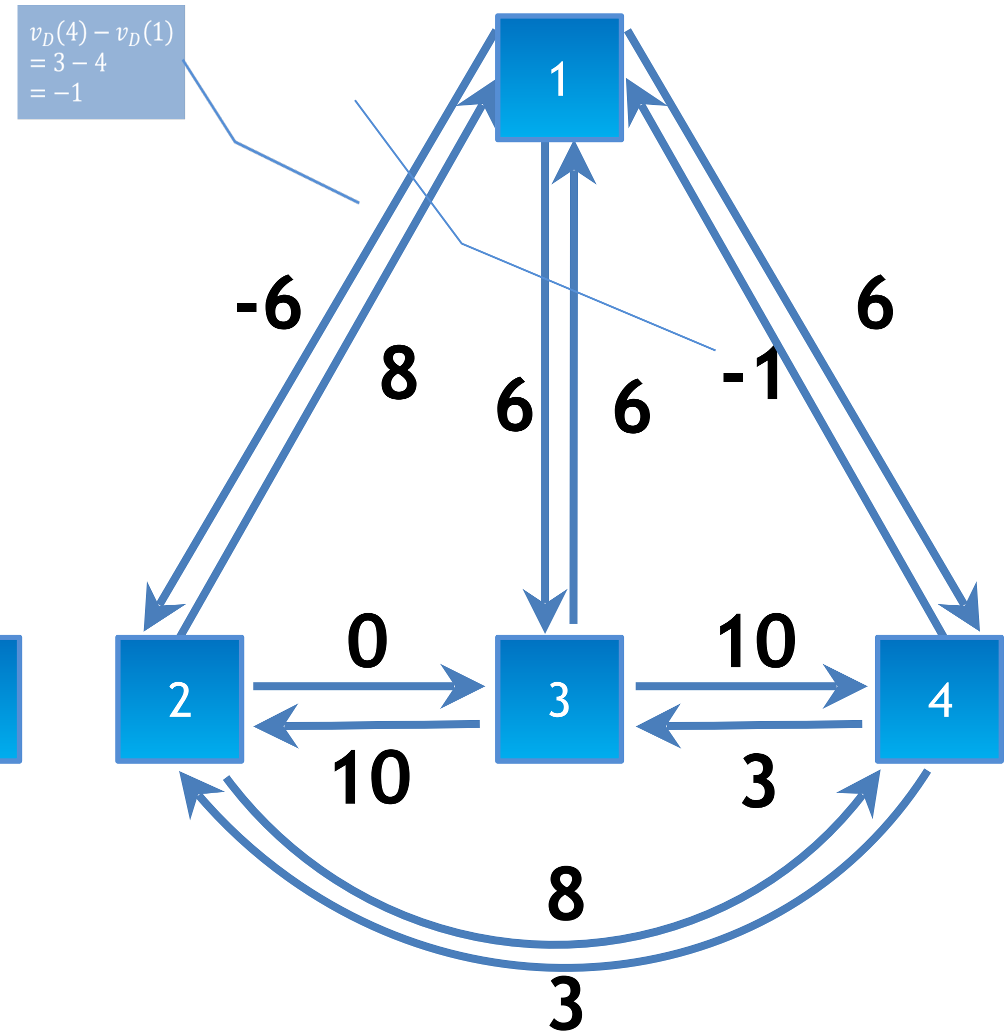
- Combinatorial algorithm
- Invariant: every buyer picks an item allocated to her in some optimal allocation in the residual market

1. Fix an arbitrary max weight matching



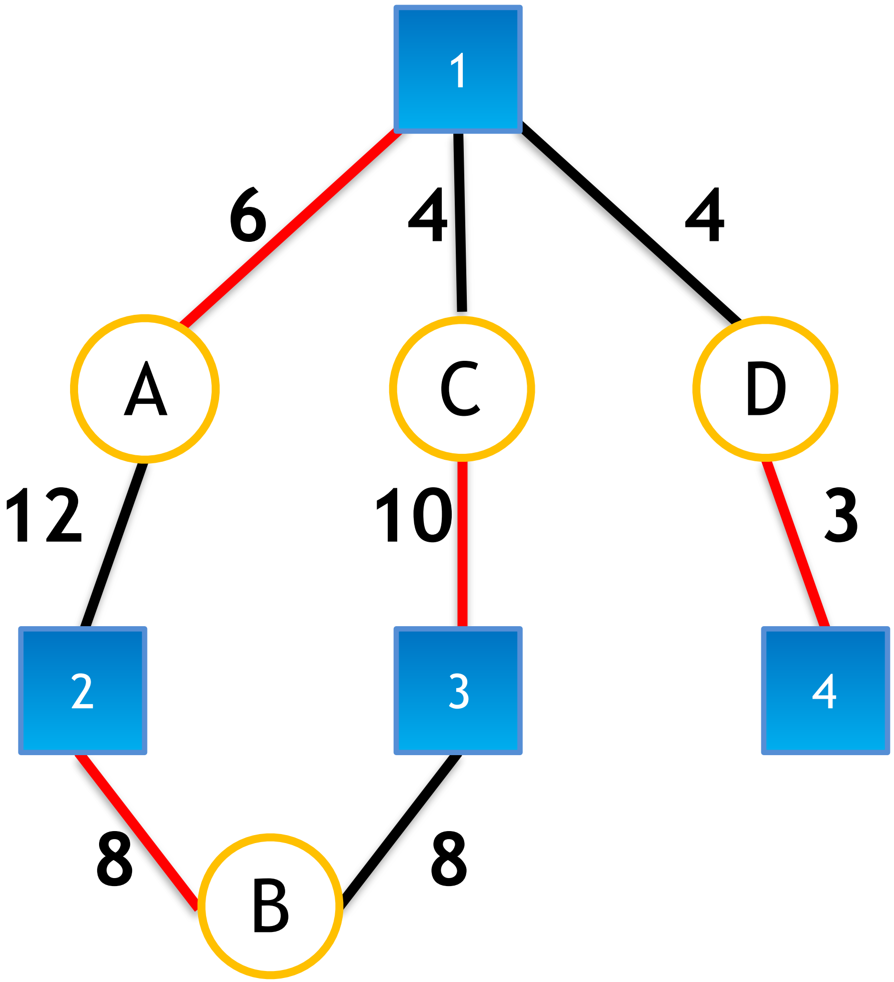
2. Build a **weighted directed** graph of items

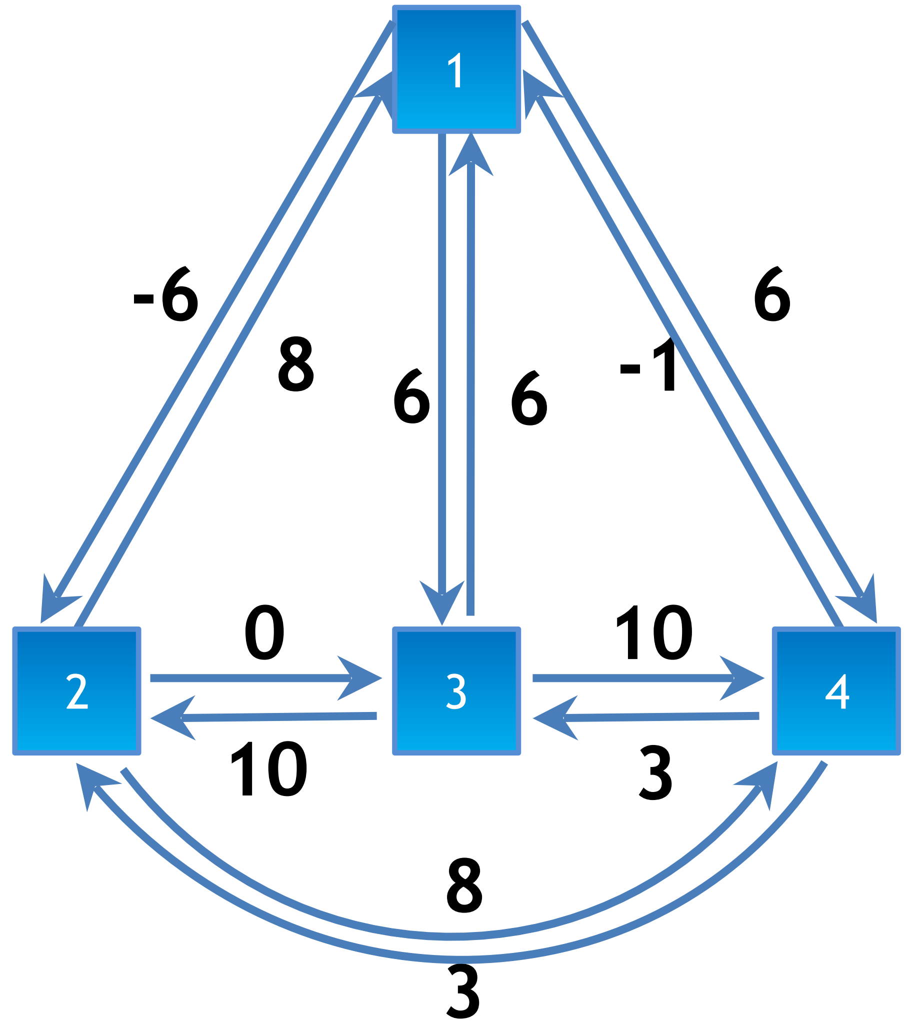




Weight of edge $\langle i, j \rangle$: $\boldsymbol{v_X(i) - v_X(j)}$, where $\boldsymbol{i}$ is matched to buyer $\boldsymbol{X}$

Buyer D must take item 4





Buyer D must take item 4

required: $v_D(4) - p_4 > v_D(1) - p_1$

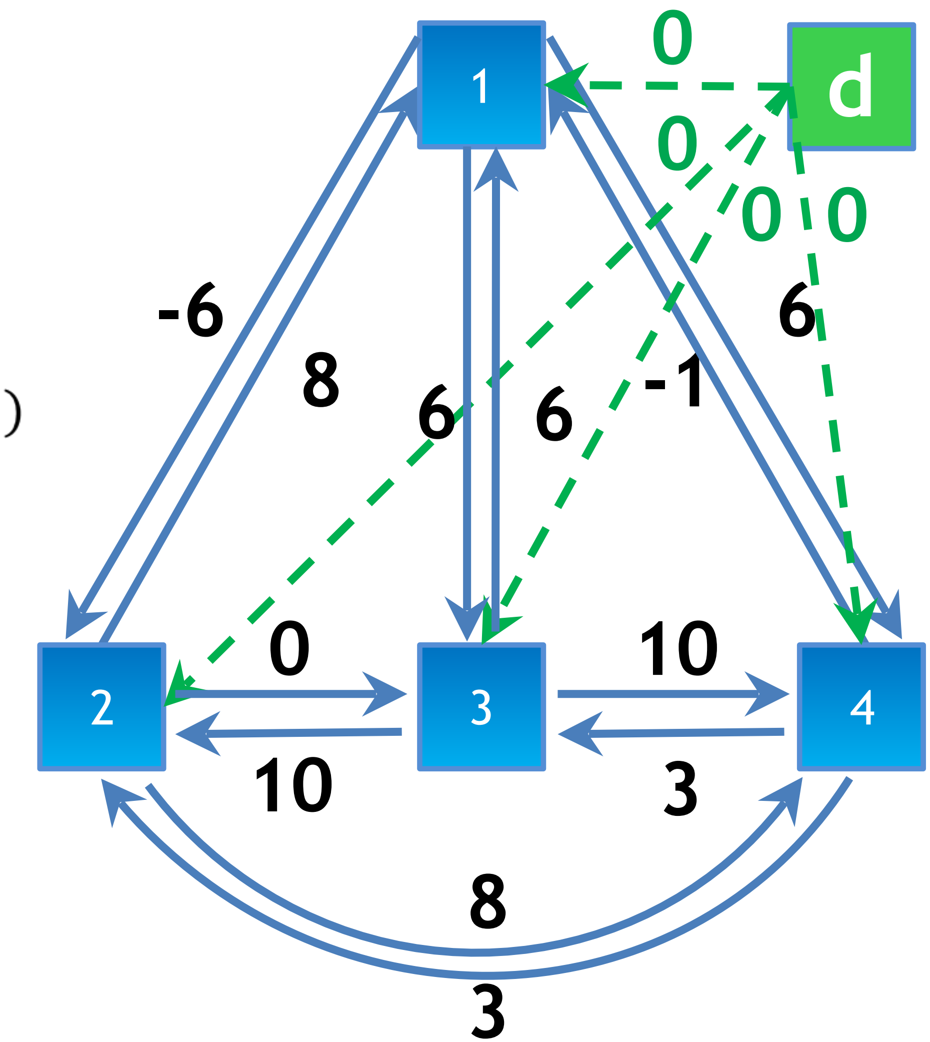
Claim: $p_j = -\mathbf{ShortestPath}(d,j)$ works

$$SP(d,4) + \left(v_D(4) - v_D(1)\right) \geq SP(d,1)$$

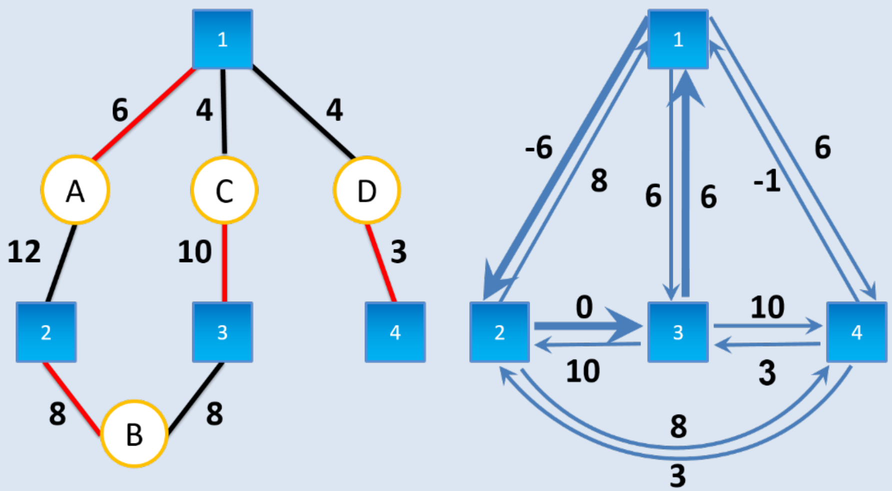
$$v_D(4) - p_4 \geq v_D(1) - p_1$$

Problem: D weakly prefer 4 to 1
Solution: decrease all weights by ϵ

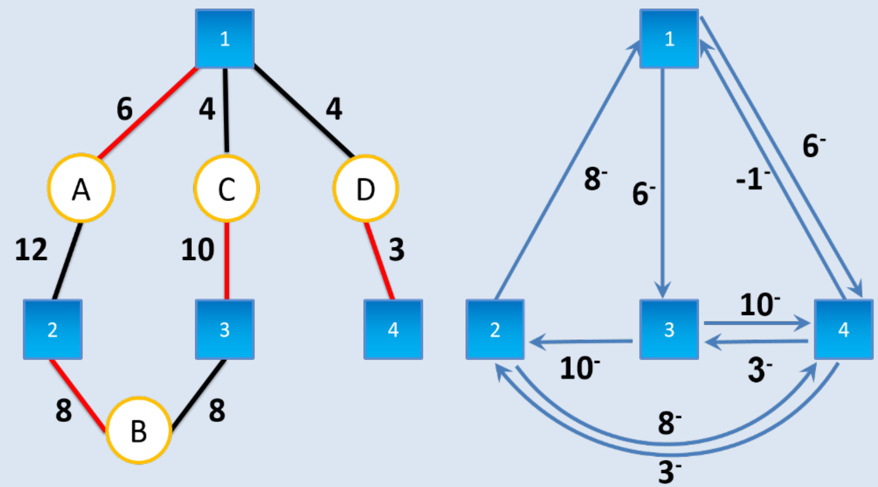
Problem: might introduce negative cycle
Solution: remove all 0-cycles a priori



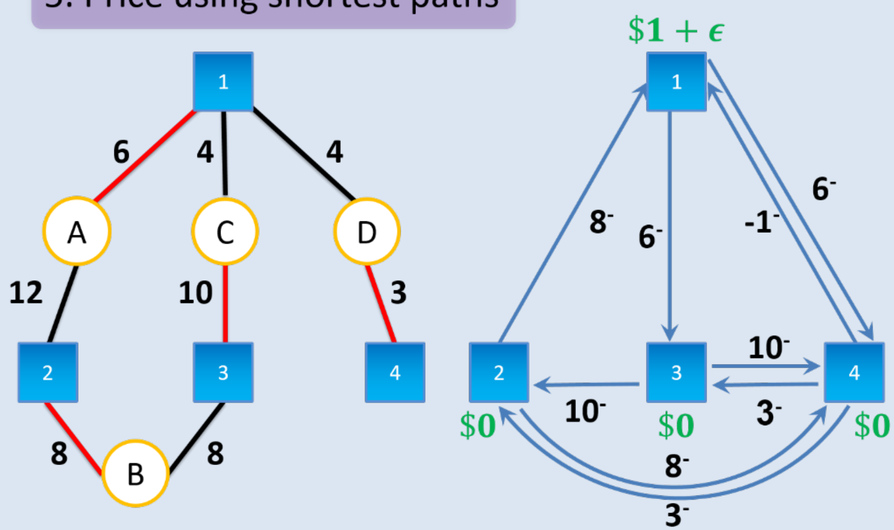
3. Delete 0 cycles



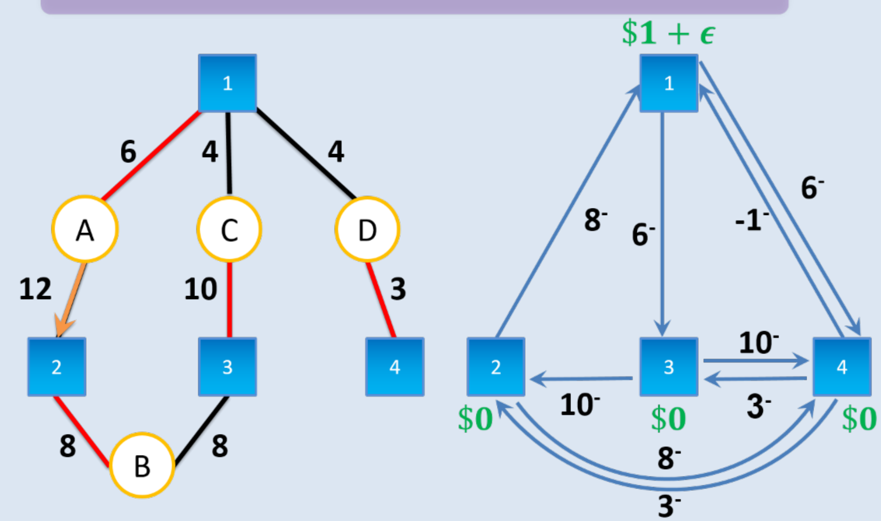
4. Decrease all weights by epsilon.



5. Price using shortest paths



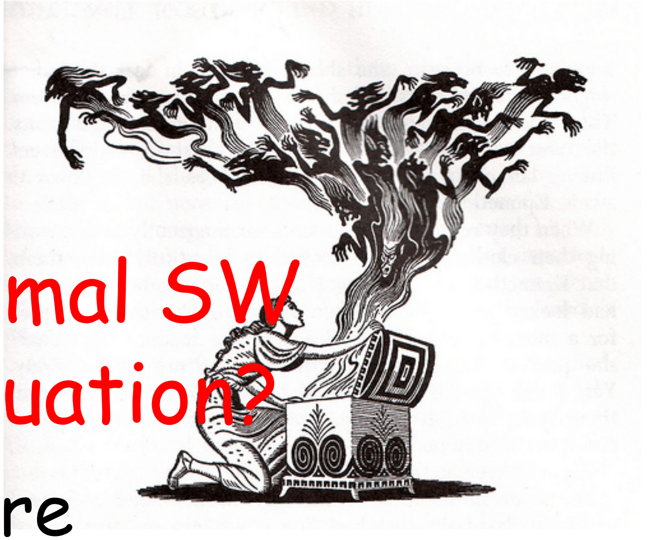
6. An arbitrary agent arrives and picks an item



[C-AEFF16] on Walrasian Pricing Matching Markets

<table>
<tr><th></th><th>Social Welfare / Opt Social Welfare</th></tr>
<tr><td>Walrasian Prices, Centralized tie Breaking</td><td>1</td></tr>
<tr><td>Walrasian Prices, Arbitrary (Selfish) tie Breaking</td><td><table><tr><th></th><th>Social Welfare / Opt Social Welfare</th></tr><tr><td>Walrasian Prices, Centralized tie Breaking</td><td>1</td></tr><tr><td>Walrasian Prices, Arbitrary (Selfish) tie Breaking</td><td>∞</td></tr><tr><td>Dynamic Pricing</td><td>1</td></tr></table></td></tr>
<tr><td>Dynamic Pricing</td><td>1</td></tr>
</table>

Open Problems



- Can dynamic prices give optimal SW for any gross-substitute valuation?
  - Remark: There are cases where Walrasian prices exist (not gross substitute) but no dynamic pricing gives optimal social welfare
- What about static prices and unweighted edges: is 2/3 of the optimal SW always achievable?

Makespan minimization, dynamic surcharge to join queue (before cart size known) Servers have known speed

$5 to join queue

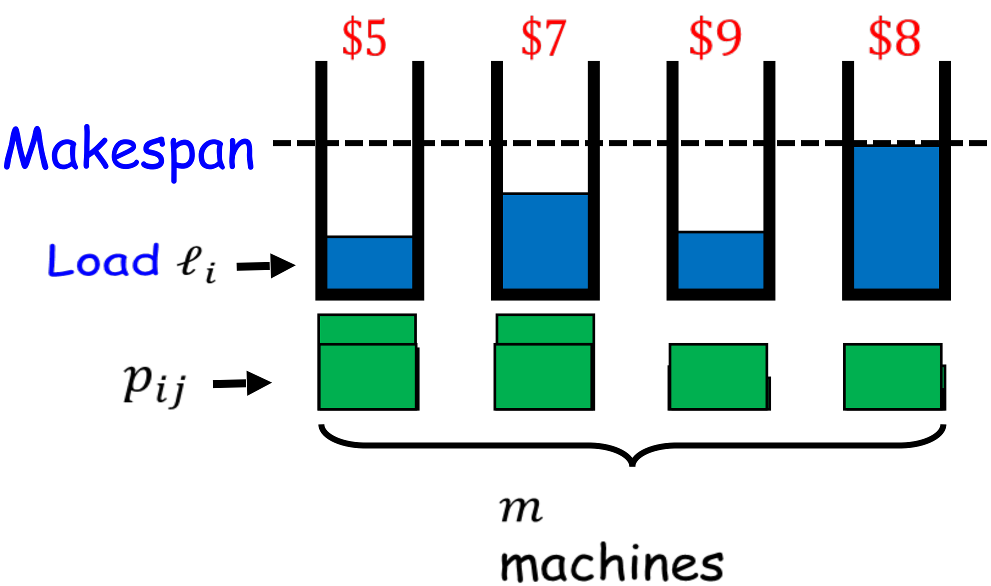
1 item/sec

$3 to join queue

3 items/sec

Makespan Minimization

- **Outcomes**: m machines, each machine i has load ℓ_i
- **Agents**: n jobs arrive online, job j has processing time p_{ij} on i
- **Objective**: Minimize $\max_i \ell_i$



Unrelated Machines: arbitrary p_{ij}

Prices over machines

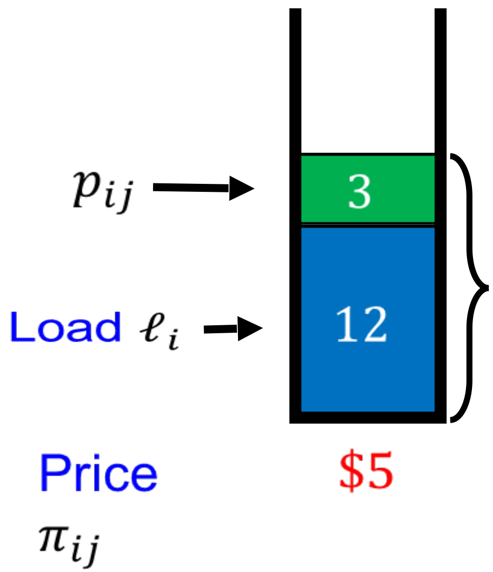
Related Machines:

Speeds $s_1 \leq \cdots \leq s_m$, $p_{ij} = \frac{p_j}{s_i}$

Identical Machines:

$p_{ij} = p_j$

Agent Costs



Outcome

$\ell_i + p_{ij} = 15$

Agent j's cost c_{ij}:

$\ell_i + p_{ij} + \pi_{ij} = 20$

Agent j chooses machine

$i^* \in \mathrm{argmin}_i c_{ij}$

Feldman Fiat Roytman Results

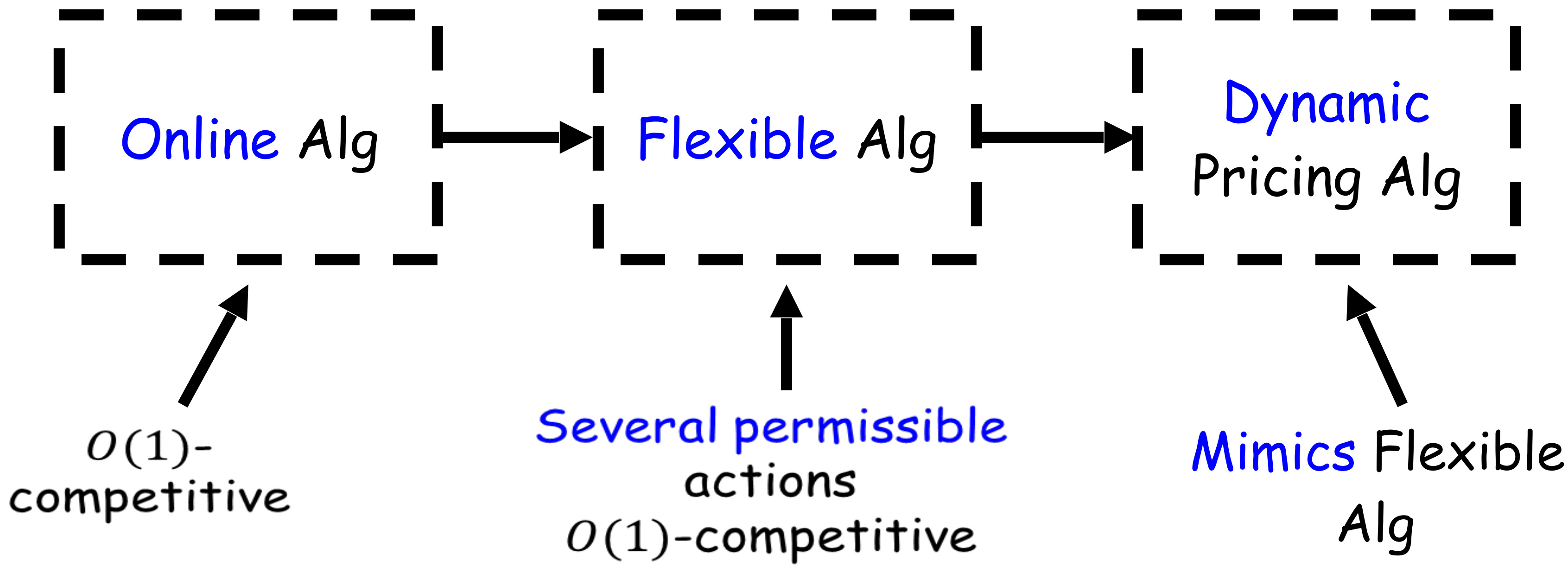
- $O(1)$-competitive dynamic pricing for related machines
- $\Omega(m)$ lower bound for dynamic pricing on unrelated machines

Machine Model	Dynamic Pricing	Static Pricing	Best Online	Greedy
Identical	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$ [1]	$\Theta(1)$ [2]
Related	$\Theta(1)$	$\Theta(\log m)$	$\Theta(1)$ [3]	$\Theta(\log m)$ [3]
Unrelated	$\Theta(m)$	$\Theta(m)$	$\Theta(\log m)$ [3,4]	$\Theta(m)$ [3]

[1]: [Albers 1997]
[2]: [Graham 1966]
[3]: [Aspnes, Azar, Fiat, Plotkin, Waarts 1993]
[4]: [Azar, Naor, Rom 1992]

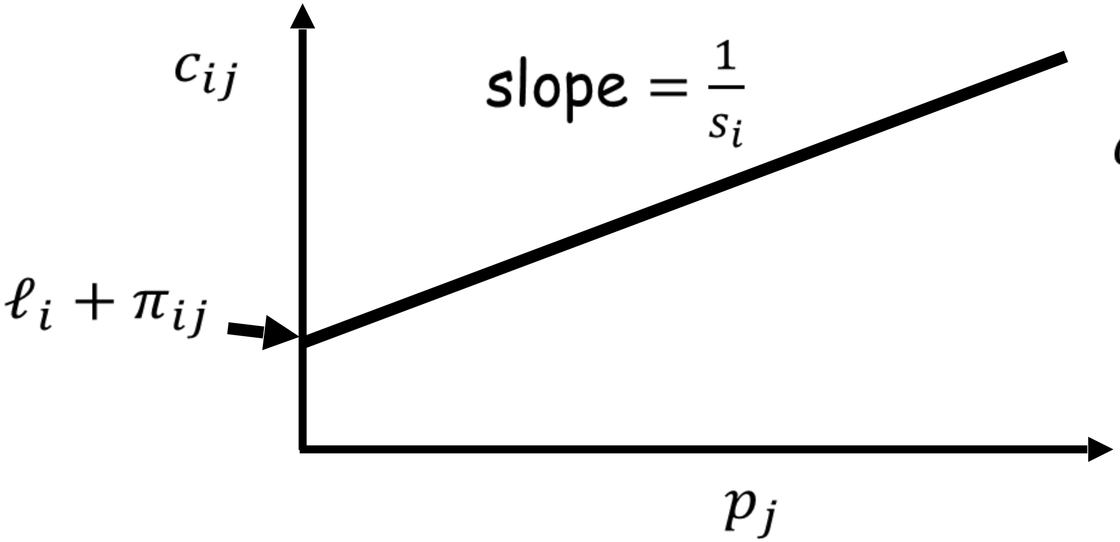
Greedy $\equiv$ Static Pricing

Main Idea: Related Machines



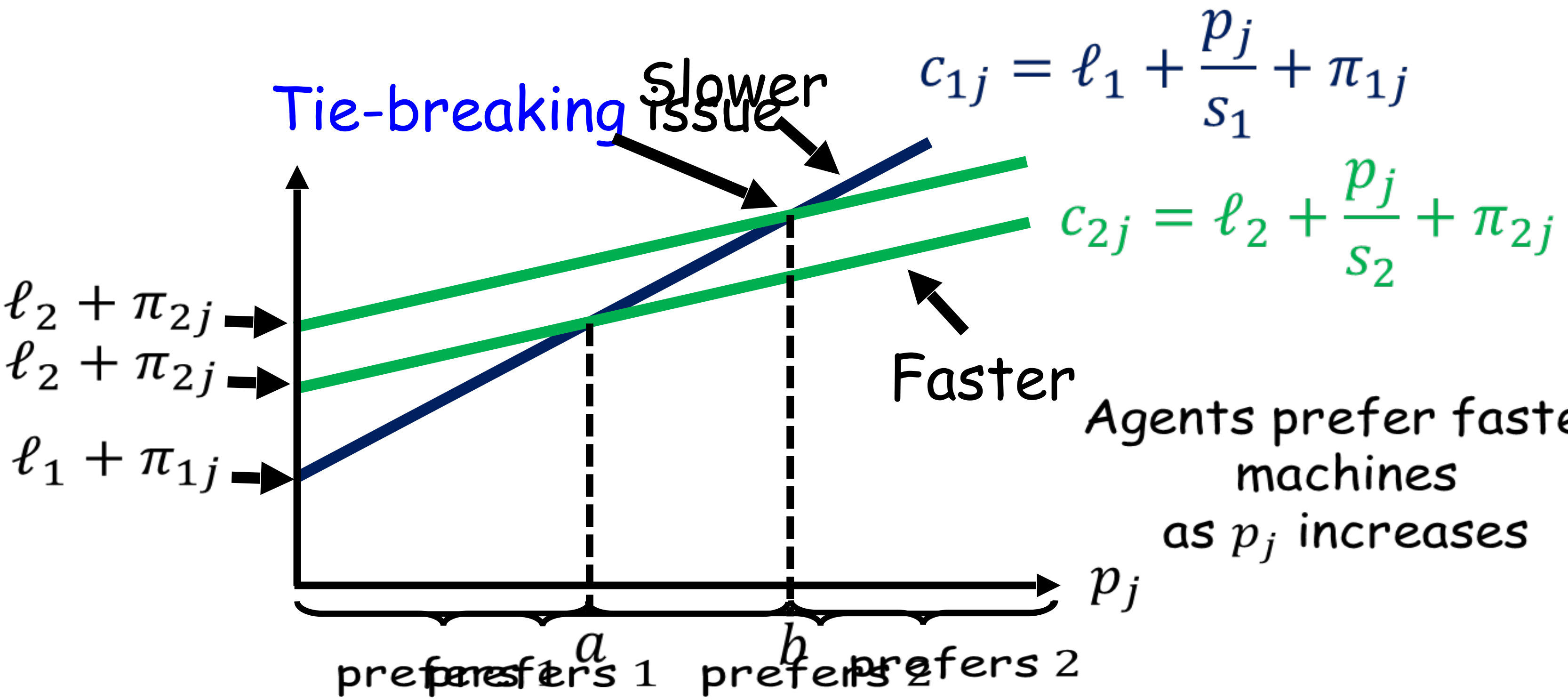
Powers and Limits of Dynamic Pricing

- Agent costs are linear in p_j



$$c_{ij} = \ell_i + \frac{p_j}{s_i} + \pi_{ij}$$

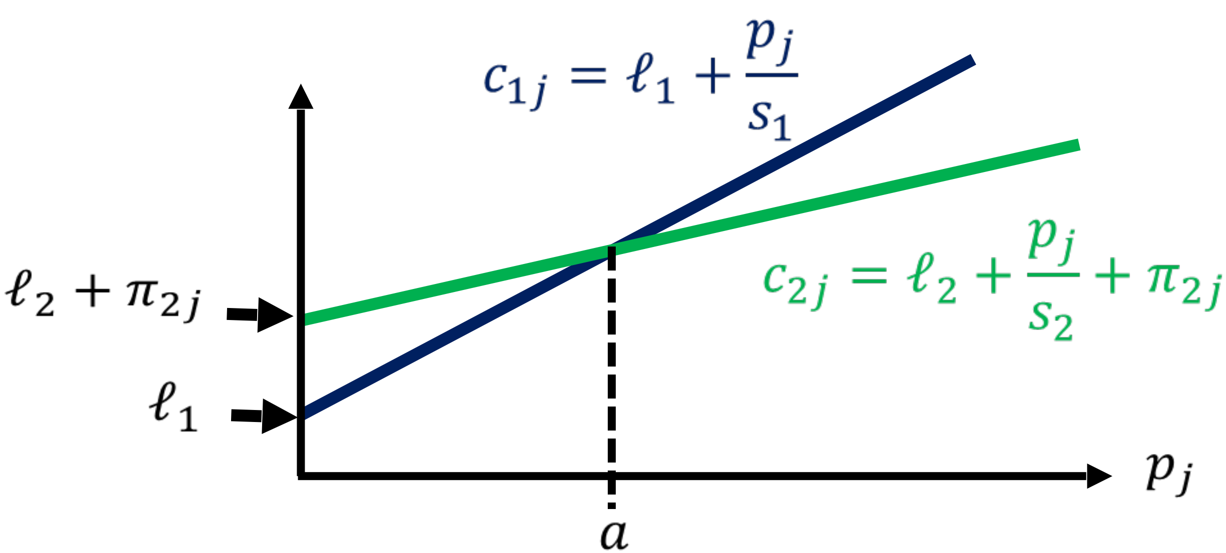
Powers and Limits of Dynamic Pricing



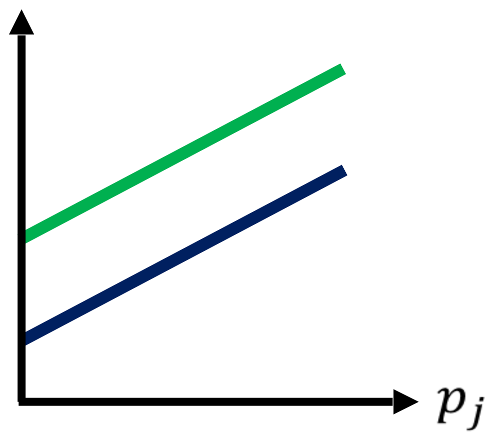
Agents prefer faster machines as p_j increases

Expressing Preferences: Two Machines

- Suppose Online Alg assigns j to machine $\begin{cases} 1 & \text{if } p_j \le a \\ 2 & \text{if } p_j > a \end{cases}$
- Set π_{2j} so that $c_{1j} = c_{2j}$ when $p_j = a$

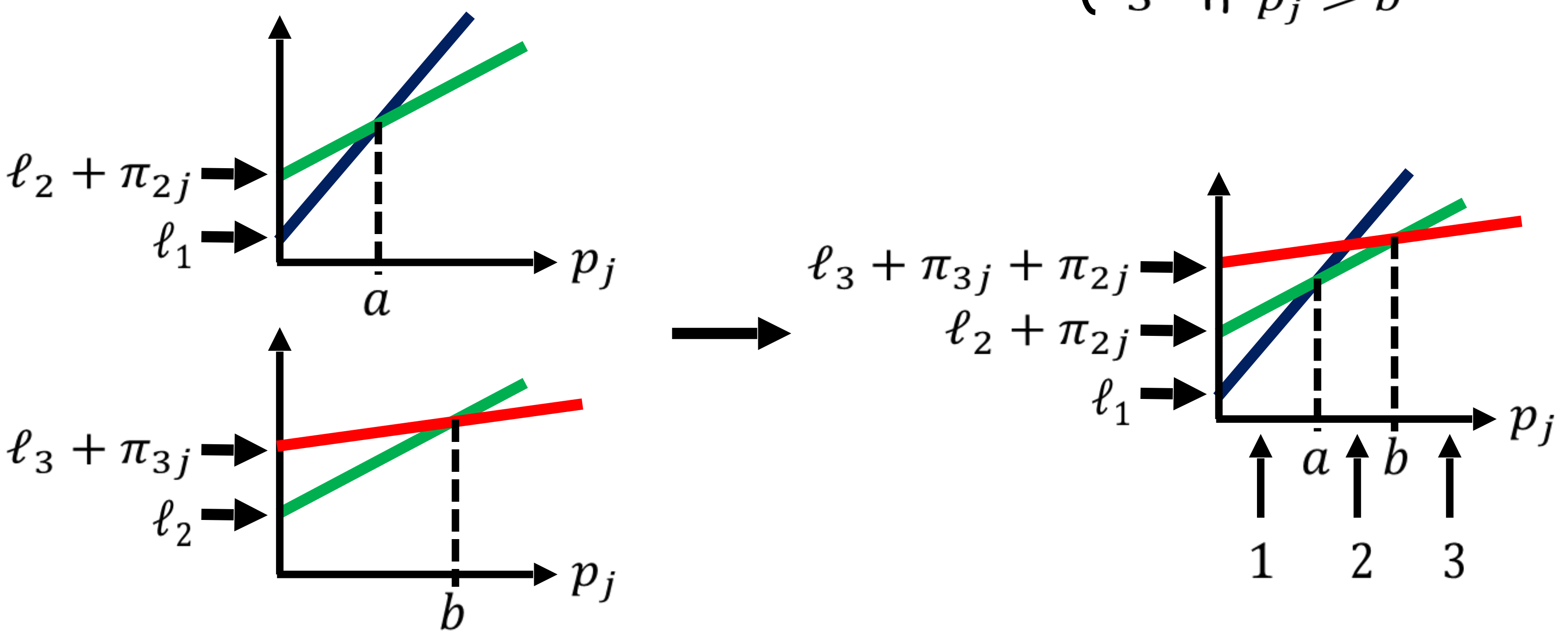


Issue: $s_1 = s_2$



Expressing Preferences: Many Machines

- Suppose Online Alg assigns j to machine $\begin{cases} 1 & \text{if } p_j \leq a \\ 2 & \text{if } a < p_j \leq b \\ 3 & \text{if } p_j > b \end{cases}$



Slow-Fit

- Online algorithm Slow-Fit is 8-competitive
[Aspnes, Azar, Fiat, Plotkin, Waarts 1993]

- Assume we know $\Lambda = OPT$ (doubling removes this assumption)

Slow-Fit:

While jobs j arrive:

Let $S = \left\{ i : \ell_i + \frac{p_j}{s_i} \leq 2 \cdot \Lambda \right\}$

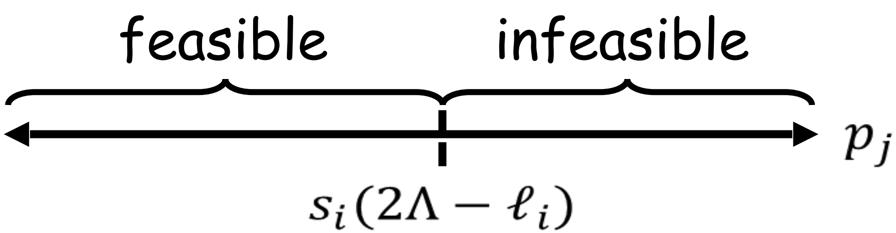
Assign j to machine $i^* = \min\{i : i \in S\}$

$S \neq \emptyset$ if $\Lambda \geq OPT$

Towards a Dynamic Pricing Scheme

- Slow-Fit assigns to lowest indexed machine in $S = \{i : \ell_i + \frac{p_j}{s_i} \leq 2\Lambda\}$
- Job j is feasible on machine i if $i \in S$:

$$\ell_i + \frac{p_j}{s_i} \leq 2\Lambda \Leftrightarrow p_j \leq s_i(2\Lambda - \ell_i)$$



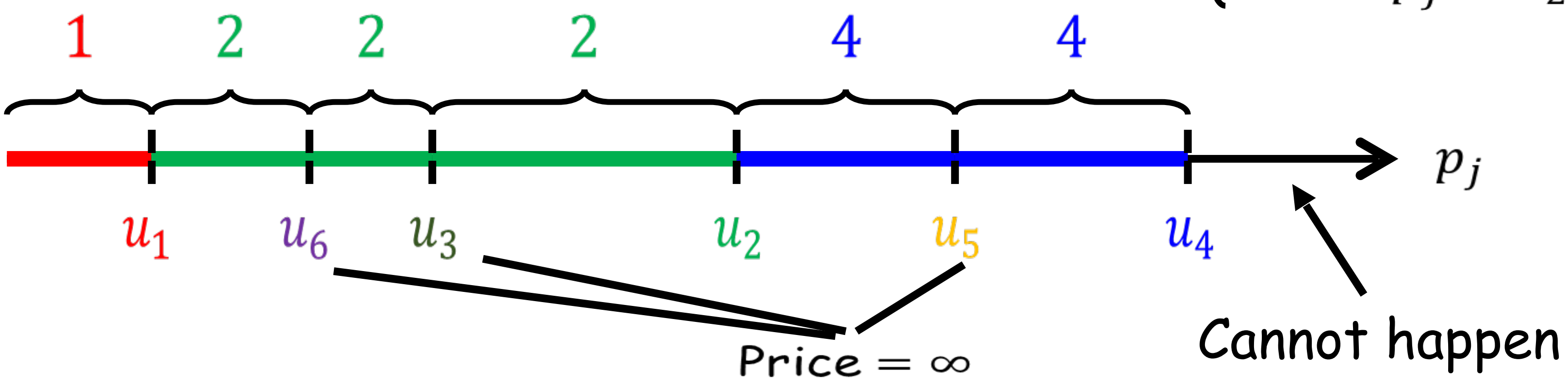
$\Lambda \geq OPT \Longrightarrow$

Always $\exists$ a feasible machine

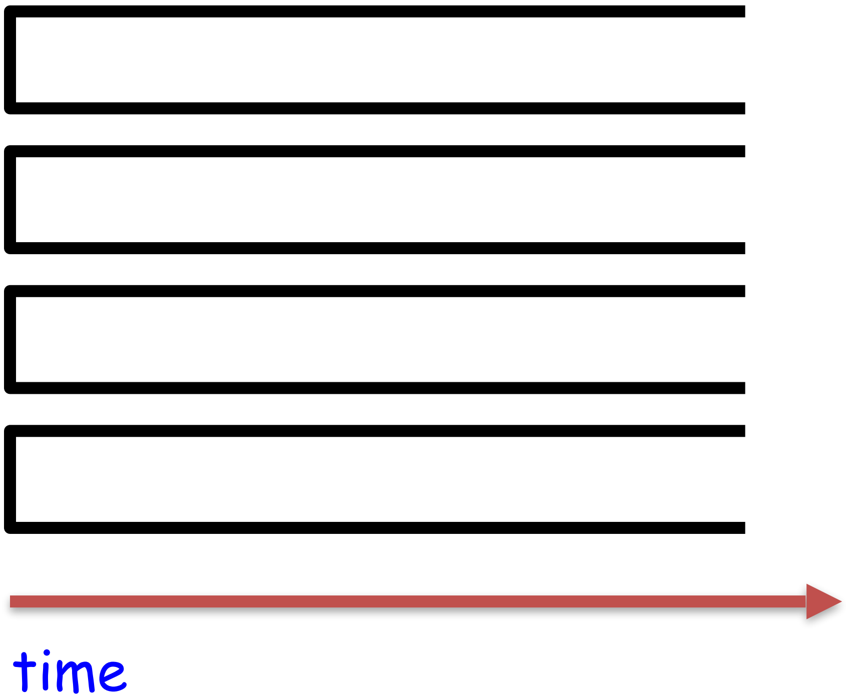
Towards a Dynamic Pricing Scheme

- Let $u_i = s_i(2\Lambda - \ell_i)$
- Sort the u_i in ascending order

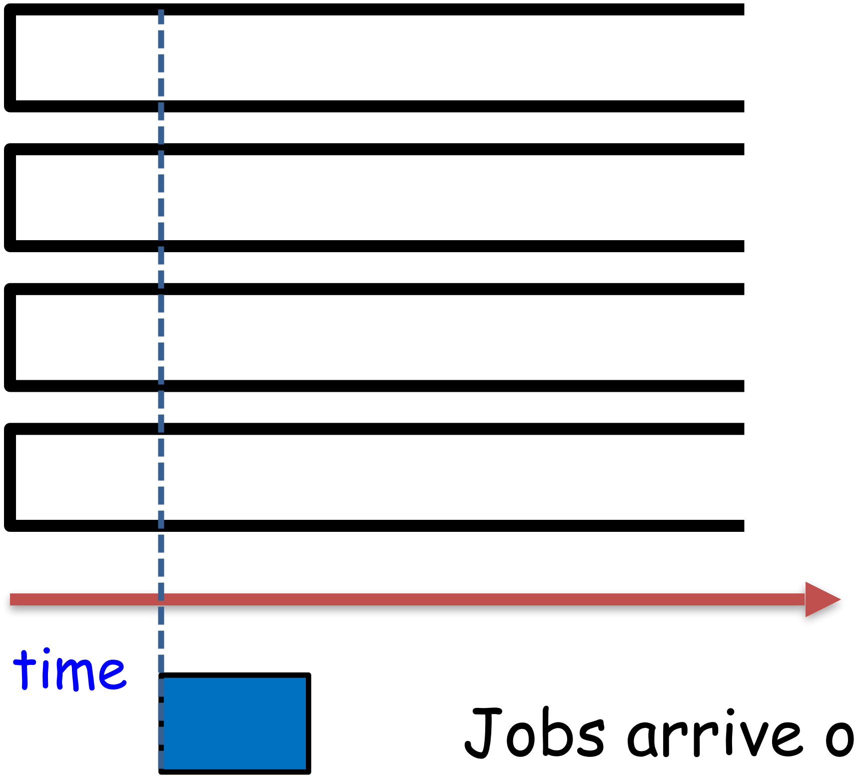
$$\text{Assignment} \begin{cases} 1 & \text{if } p_j \le u_1 \\ 2 & \text{if } u_1 < p_j \le u_2 \\ 4 & \text{if } p_j > u_2 \end{cases}$$



Maximum flow time minimization

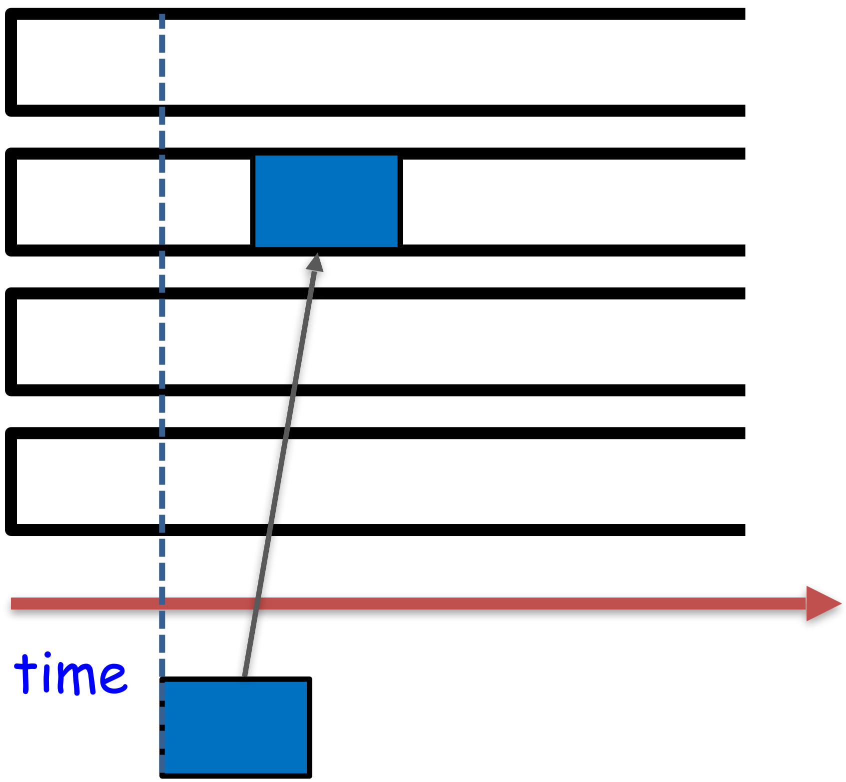


Maximum flow time minimization

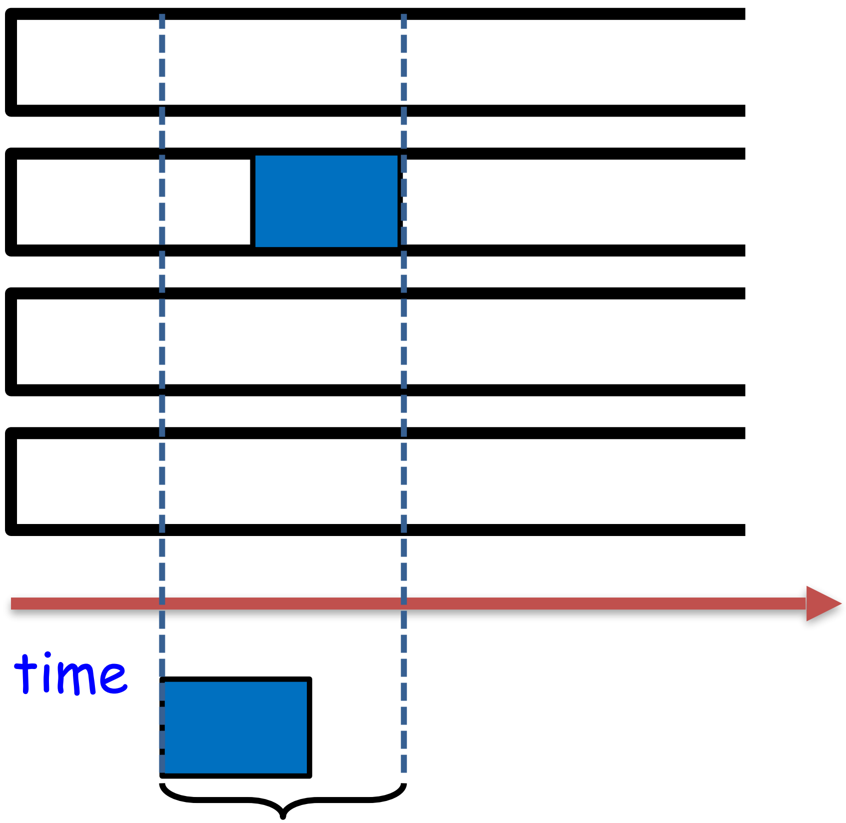


Jobs arrive over time

Maximum flow time minimization

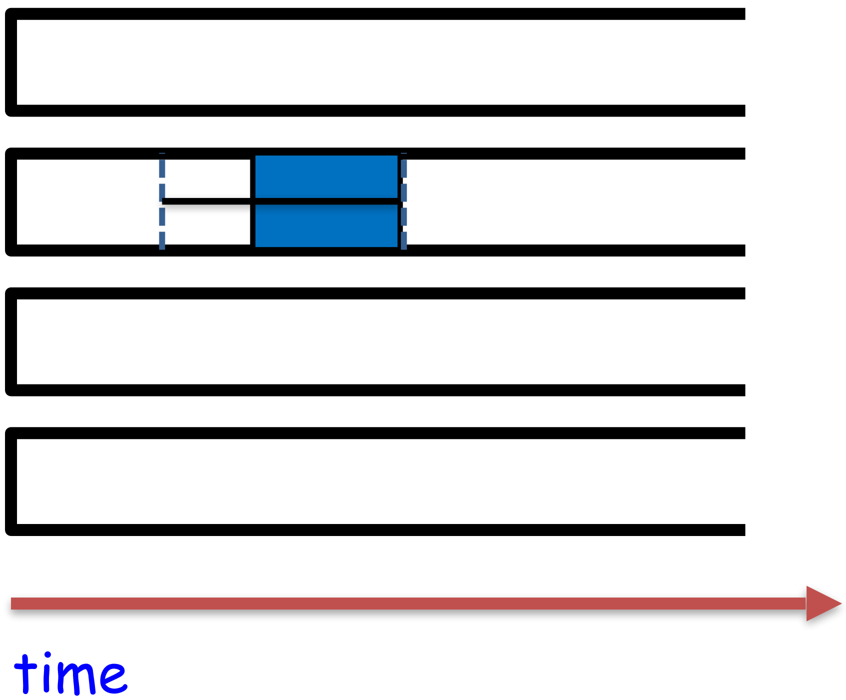


Maximum flow time minimization

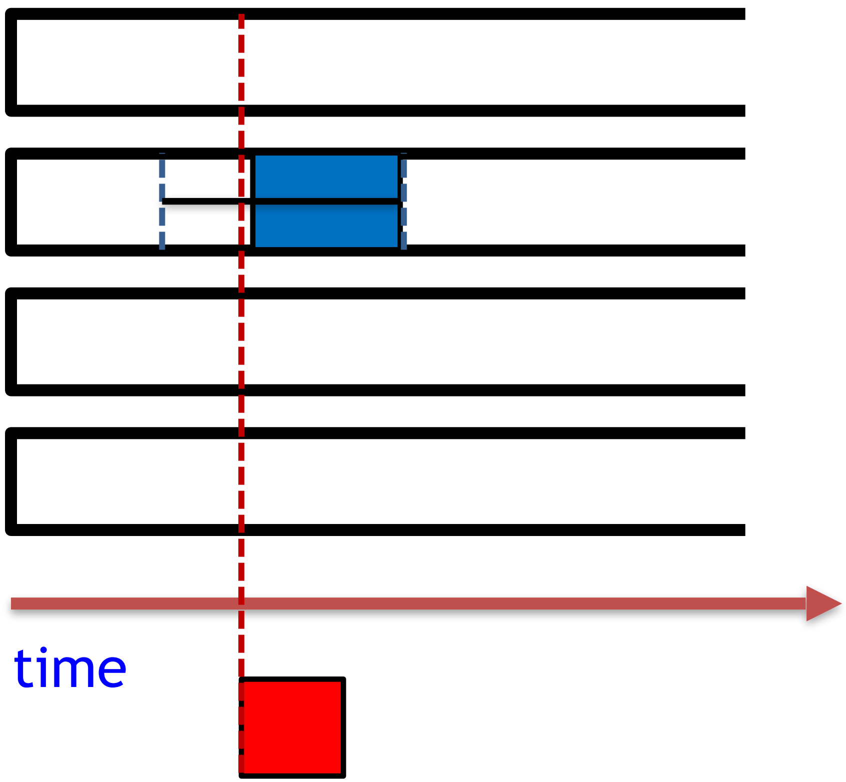


Flow time =
Δ between release and completion time

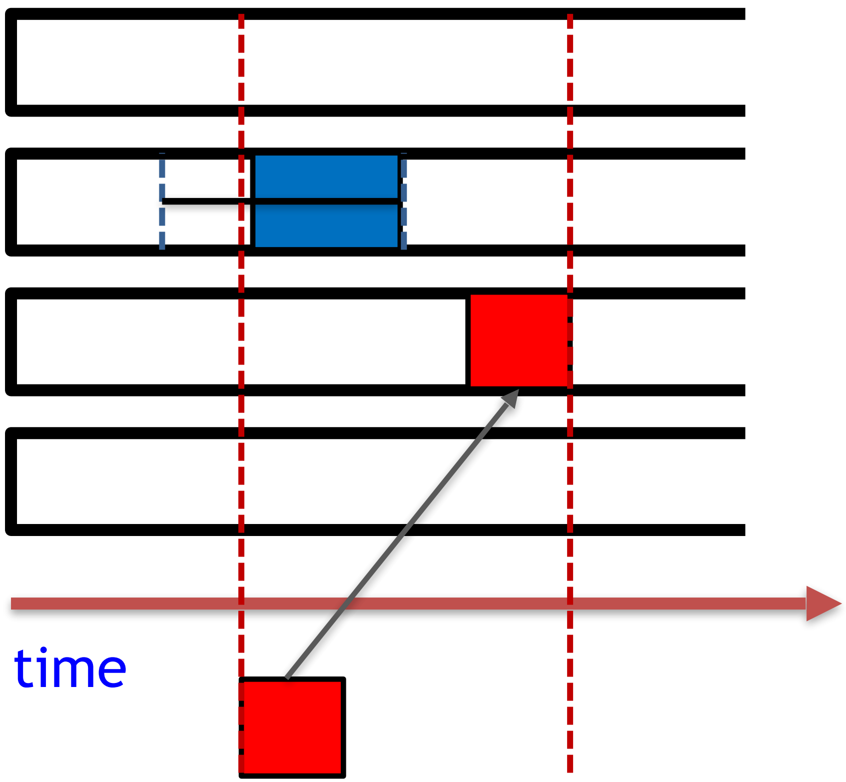
Maximum flow time minimization



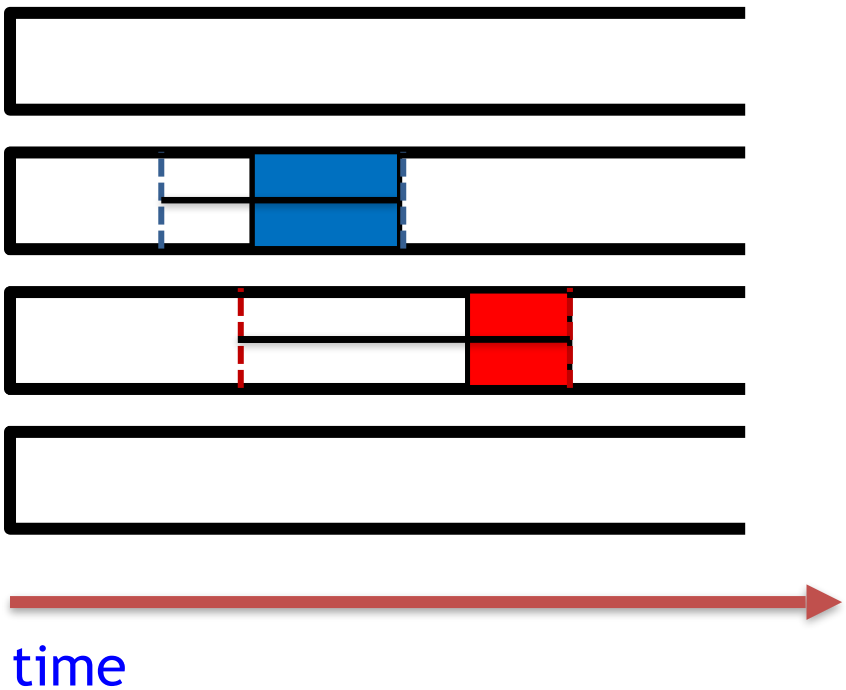
Maximum flow time minimization



Maximum flow time minimization

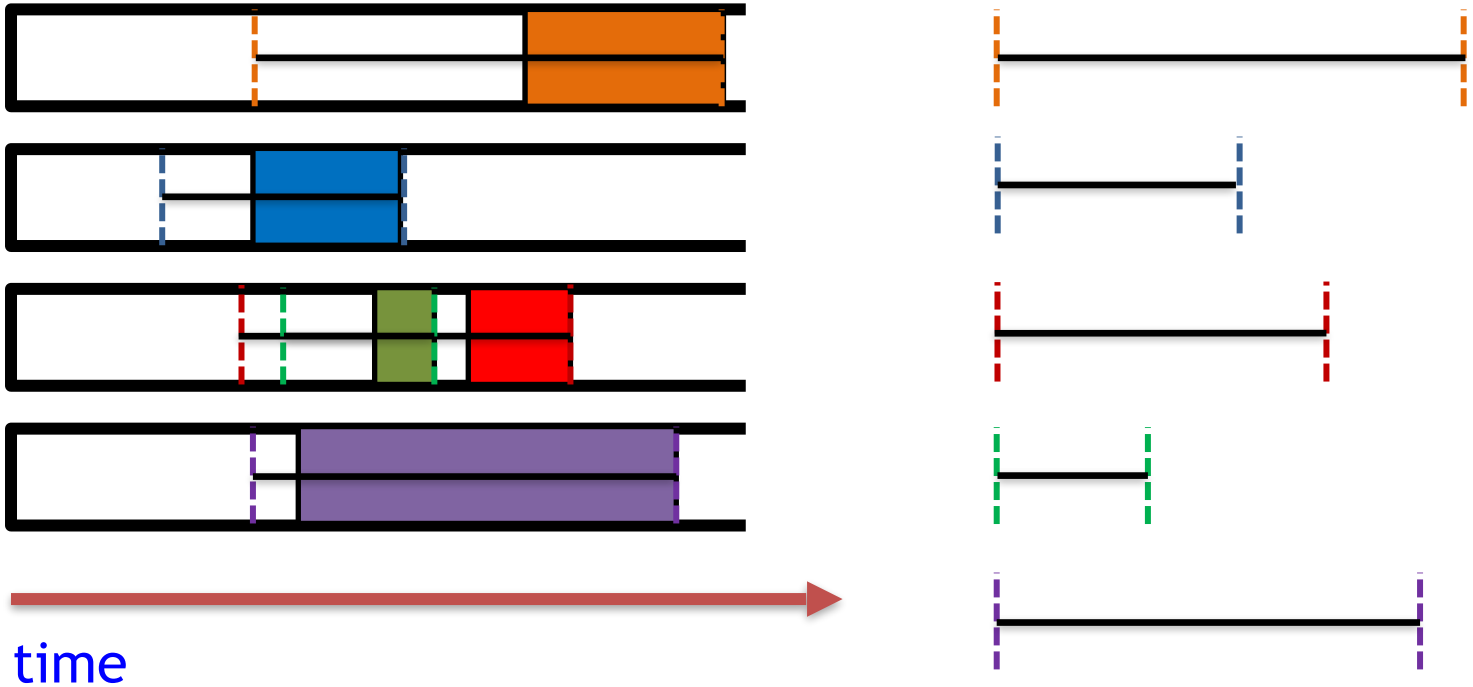


Maximum flow time minimization

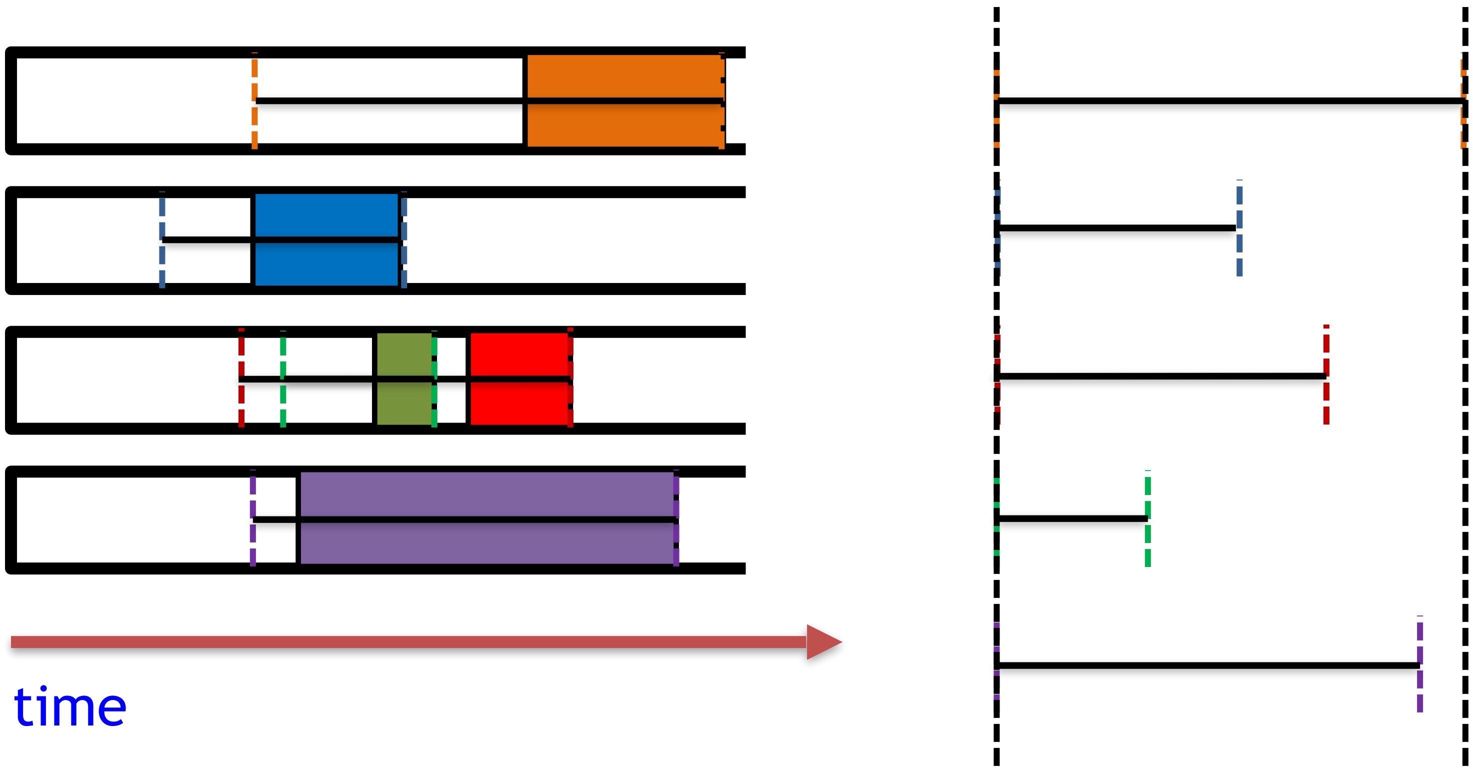


Maximum flow time minimization

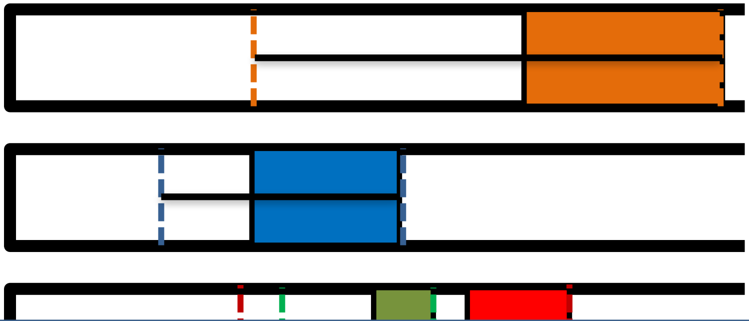
Maximum flow time minimization



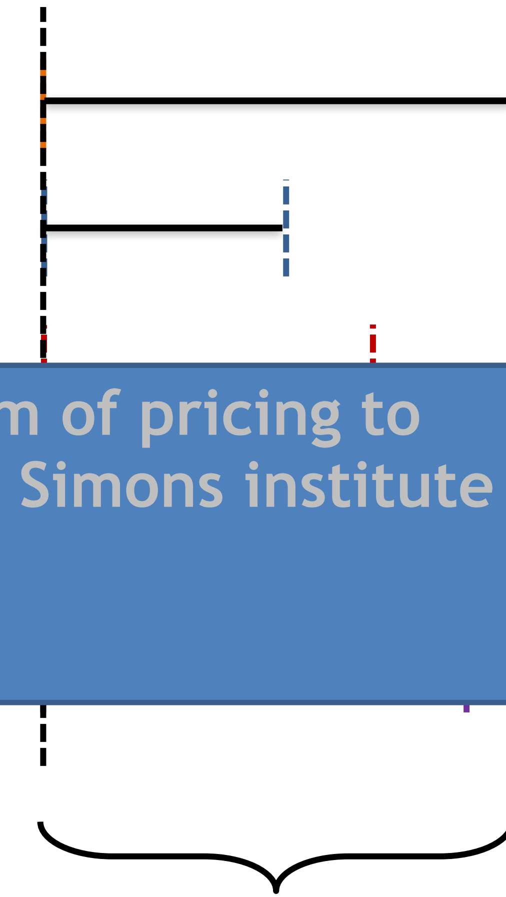
Maximum flow time minimization



Maximum flow time minimization

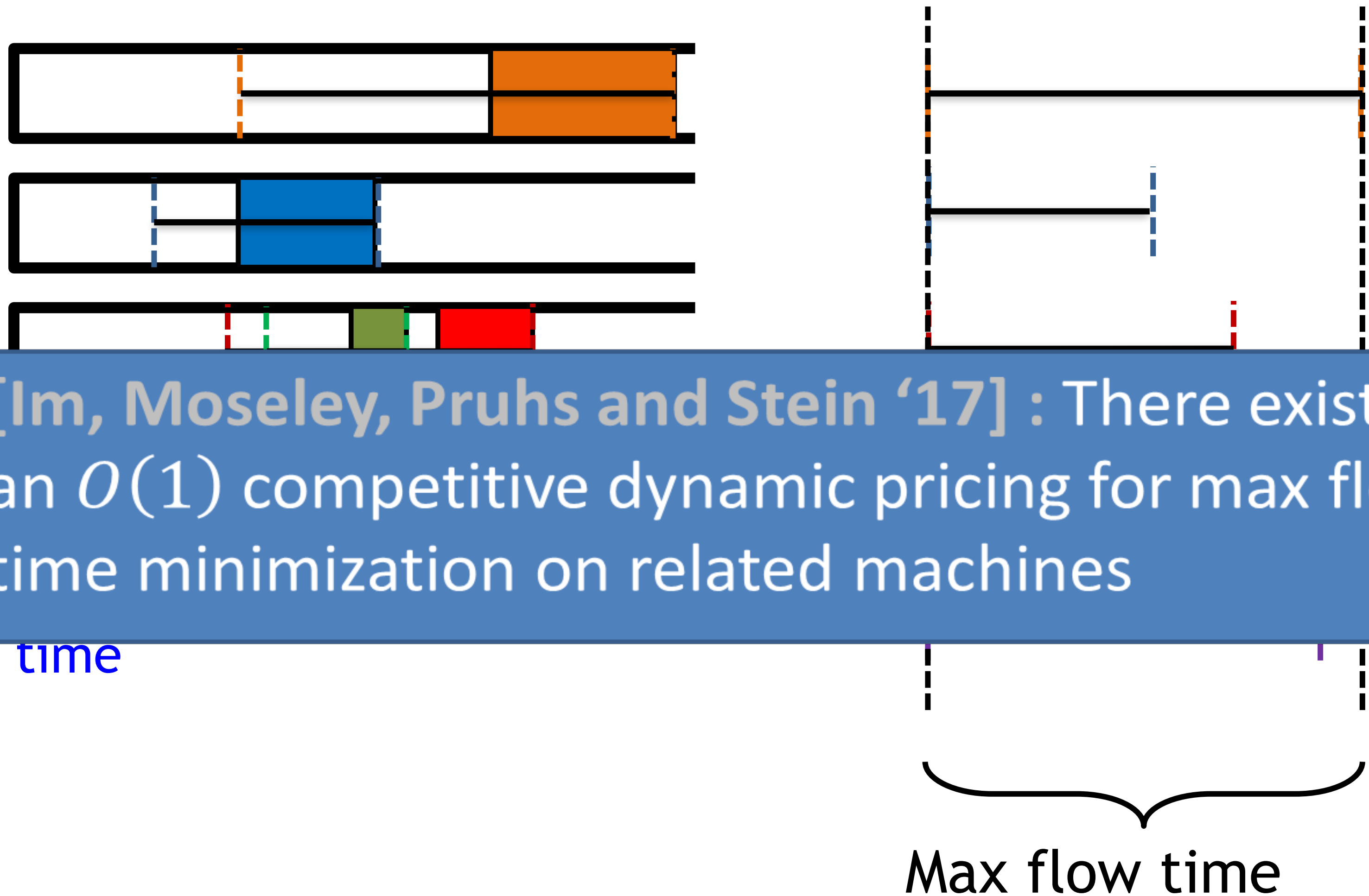


Last November I posed the problem of pricing to approximate max flow time at the Simons institute at UC Berkeley



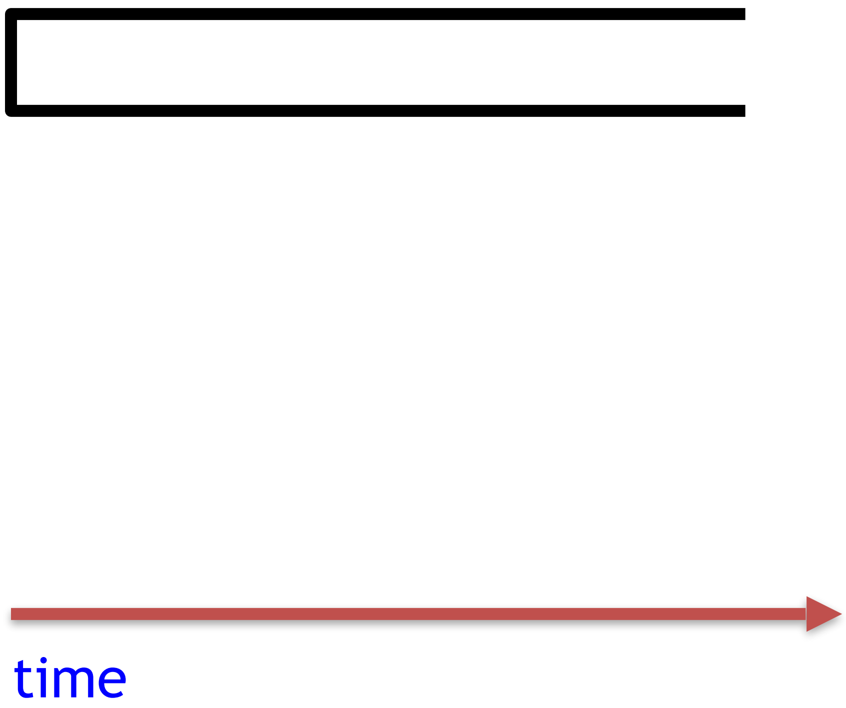
Max flow time

Maximum flow time minimization

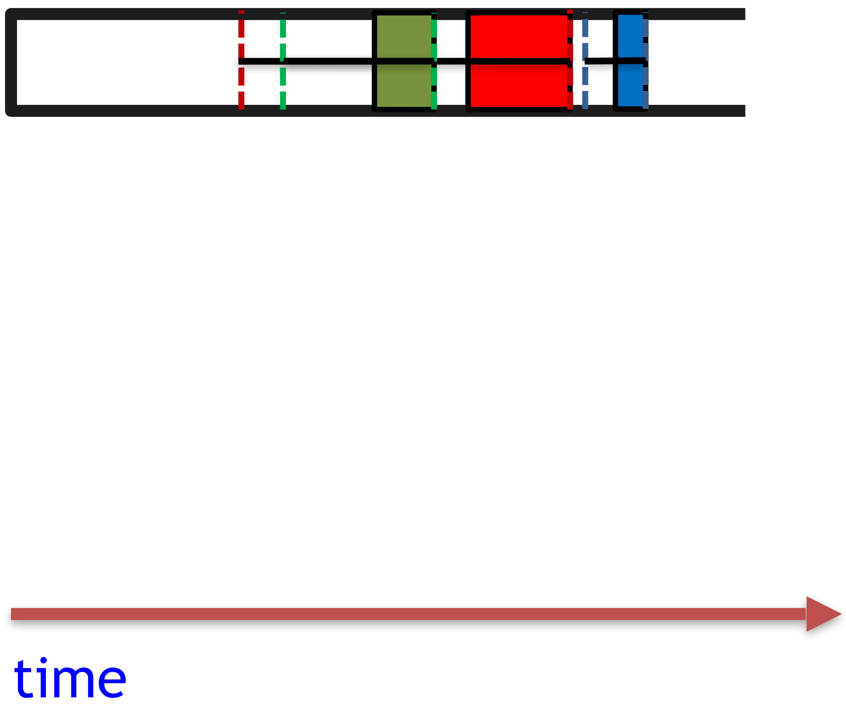


[Im, Moseley, Pruhs and Stein '17] : There exists an $O(1)$ competitive dynamic pricing for max flow time minimization on related machines

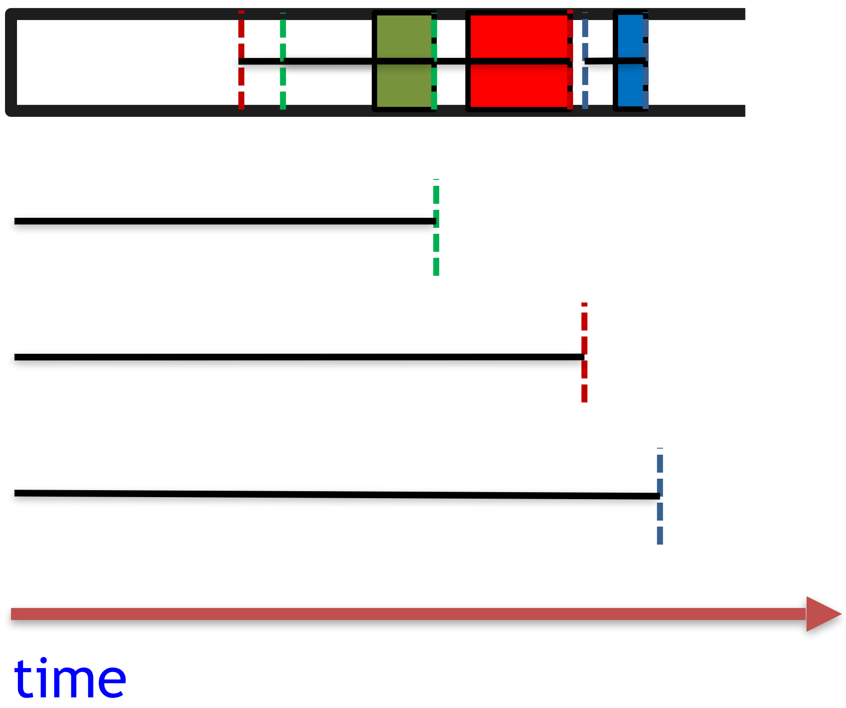
Sum of completion times



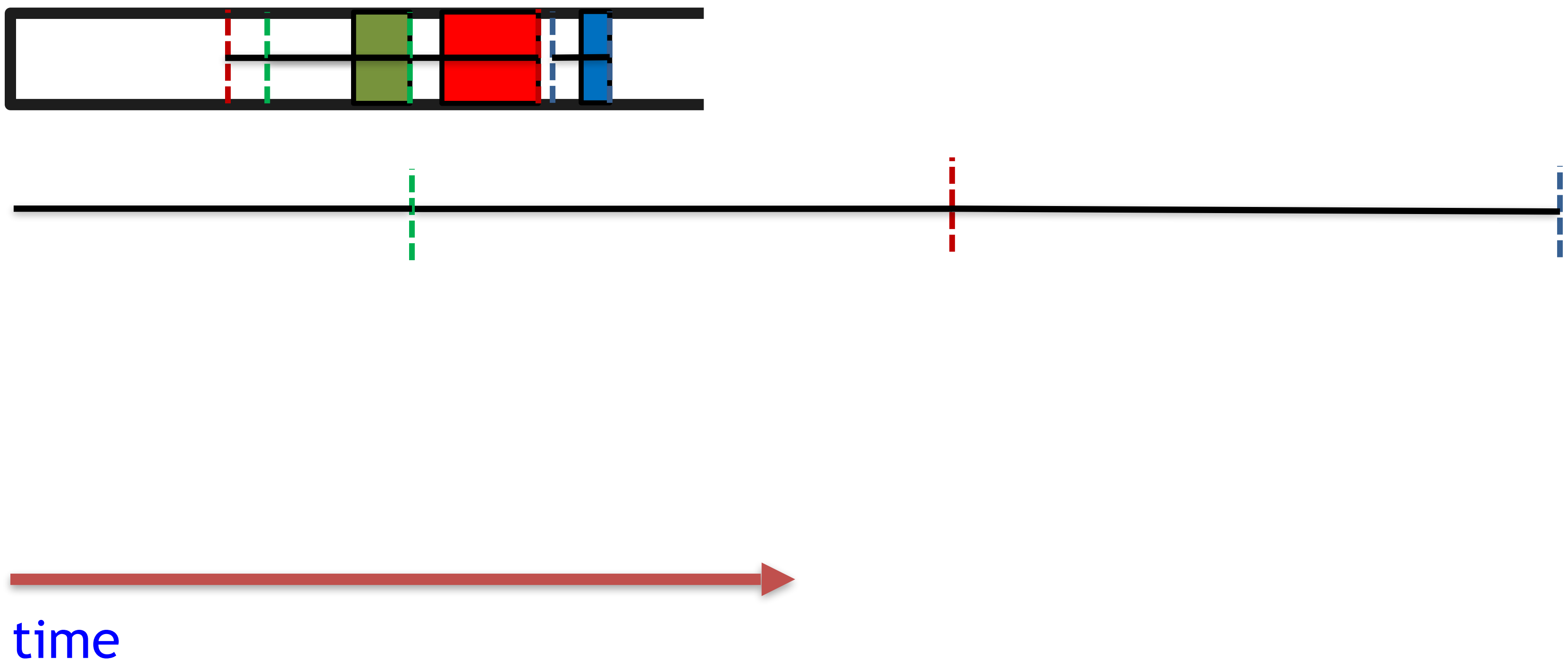
Sum of completion times



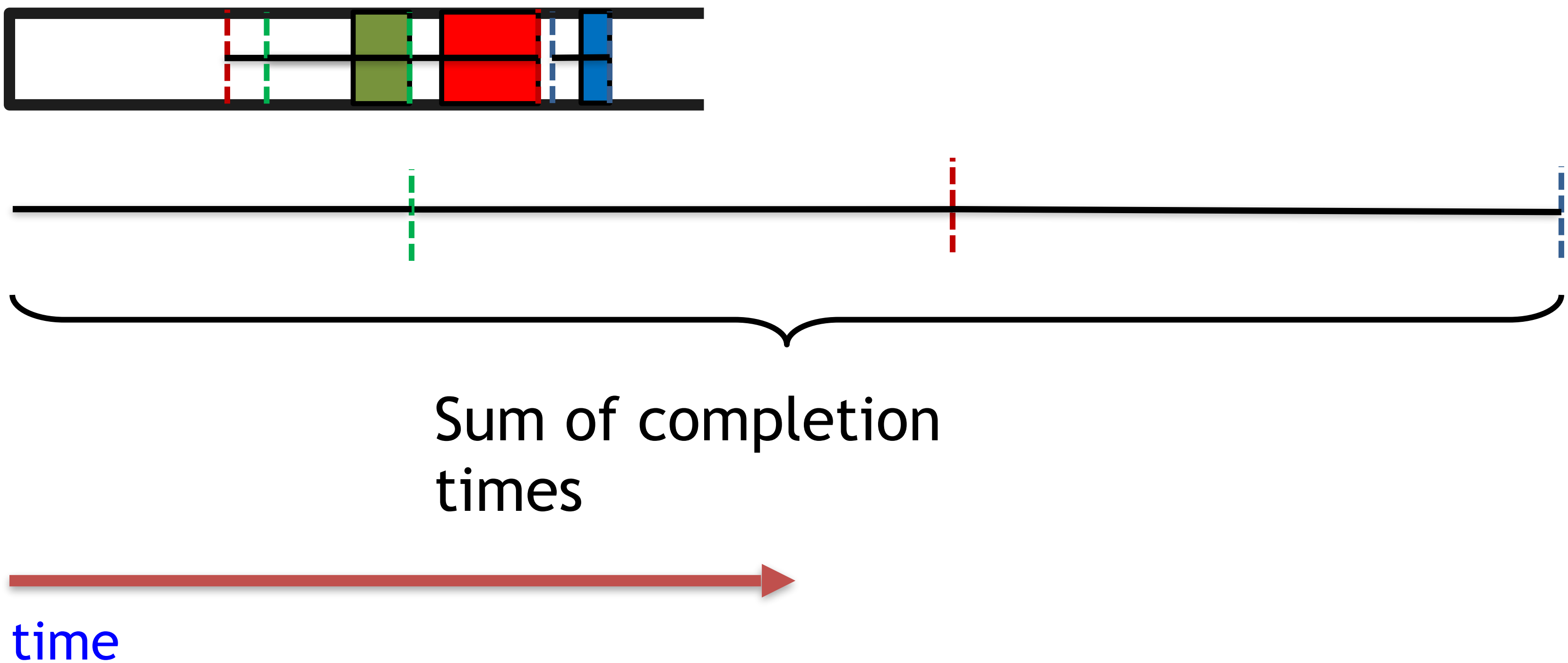
Sum of completion times



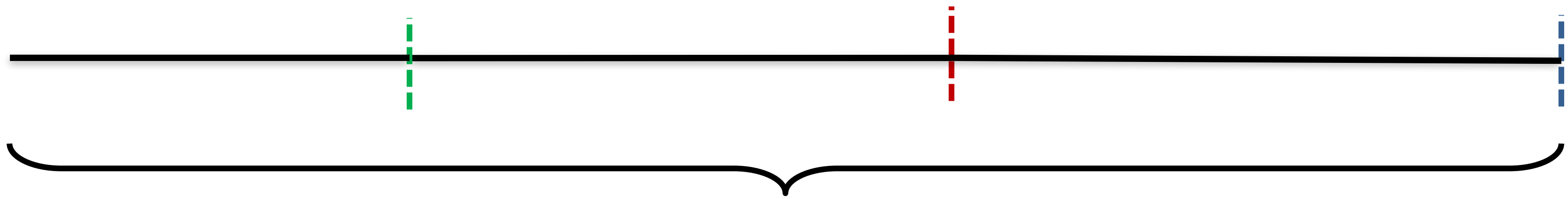
Sum of completion times



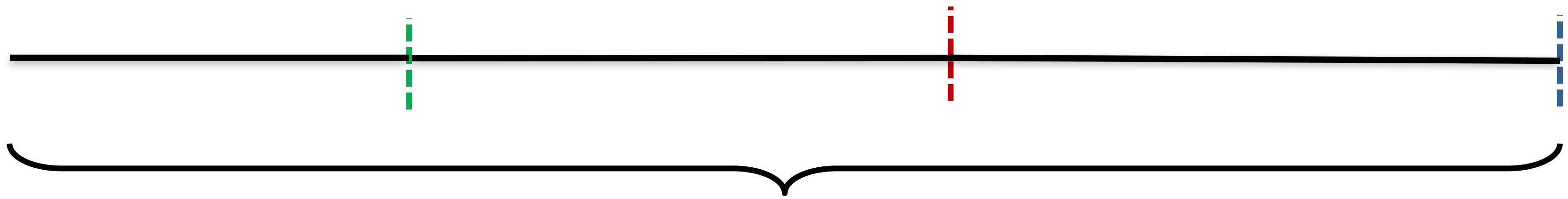
Sum of completion times



Sum of completion times



Sum of completion times



Give a menu of options

- Non-migrative
- Non-preemptive
- Prompt (immediate dispatch)

Sum of completion times – some results [Eden, Feldman, Fiat and Taub]

Sum of completion times – some results [Eden, Feldman, Fiat and Taub]

Let V_k be the total volume of jobs of size $(2^{k-1}, 2^k]$

Sum of completion times – some results [Eden, Feldman, Fiat and Taub]

Let V_k be the total volume of jobs of size $(2^{k-1}, 2^k]$

Lower bound of $\Omega(\log(\max_k V_k))$ for prompt online algorithms even for a single machine with release times are all 0

Sum of completion times – some results [Eden, Feldman, Fiat and Taub]

Let V_k be the total volume of jobs of size $(2^{k-1}, 2^k]$

Lower bound of $\Omega(\log(\max_k V_k))$ for prompt online algorithms even for a single machine with release times are all 0

Competitive ratio of $O(\log(\max_k V_k))$ with multiple machines and release times

Open Problem: What about Sum of Flow times?