Cost-Sharing Methods for Scheduling Games under Uncertainty

Vasilis Gkatzelis, Drexel University

Joint work with:

Giorgos Christodoulou, **University of Liverpool**

Alkmini Sgouritsa, **Max-Planck Institute**

Load Balancing

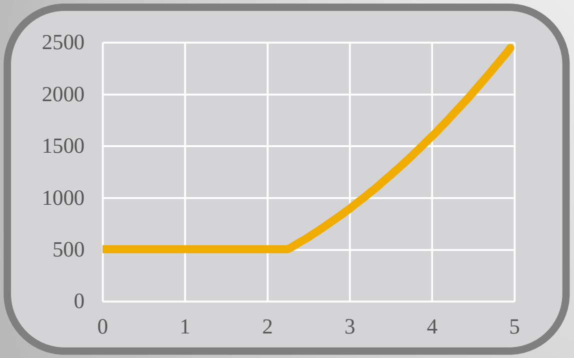
- We have a set M of m machines and set N of n jobs
- Each job i has weight w_i and needs to be processed by some machine
- Each machine j has a non-decreasing cost function $c_j(\ell)$
  - This cost depends on the load $\ell = \sum_i w_i$ of the agents using it
  - For this talk, just assume that $w_i = 1$ for every agent i
  - The cost function satisfies $c_j(0) = 0$ for every machine j
  - Each cost function can be convex, concave, or more complicated

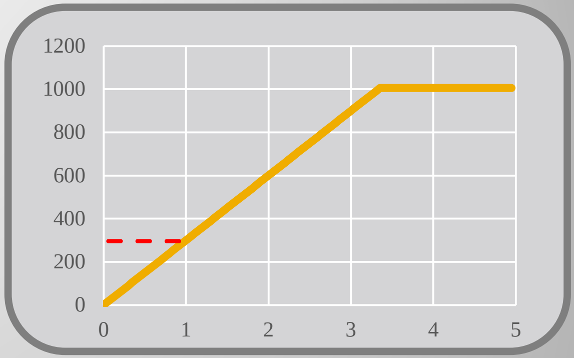
- A schedule $\boldsymbol{s}$ assigns each job to a machine
  - Let $S_j(s)$ be the set of jobs assigned to machine j in schedule $\boldsymbol{s}$
  - Let $\ell_j(s) = \sum_{i \in S_j(\boldsymbol{s})} w_i$ be the total load of the jobs using j in $\boldsymbol{s}$
  - Then the cost on each machine j is $c_j(s) = c_j(\ell_j(s))$

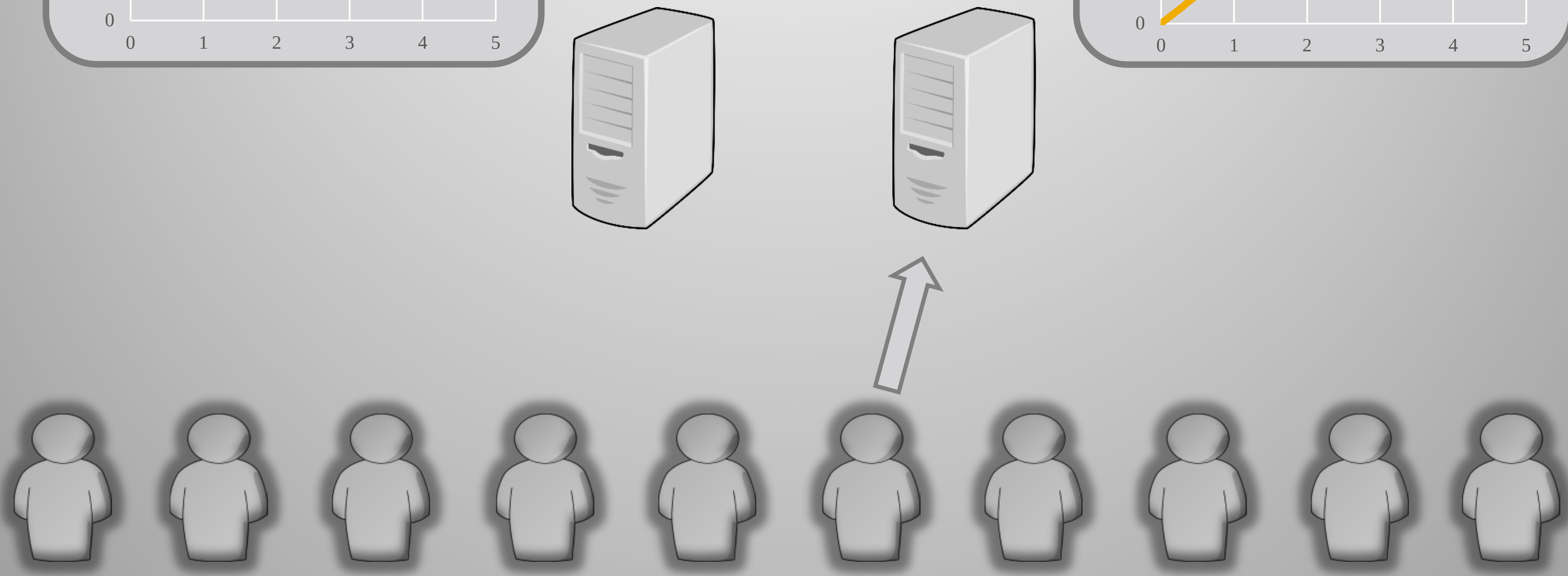
- Our goal is to output a schedule $\boldsymbol{s}$ that minimizes $C(\boldsymbol{s}) = \sum_{j \in M} c_j(\boldsymbol{s})$

Scheduling Games

Scheduling Games

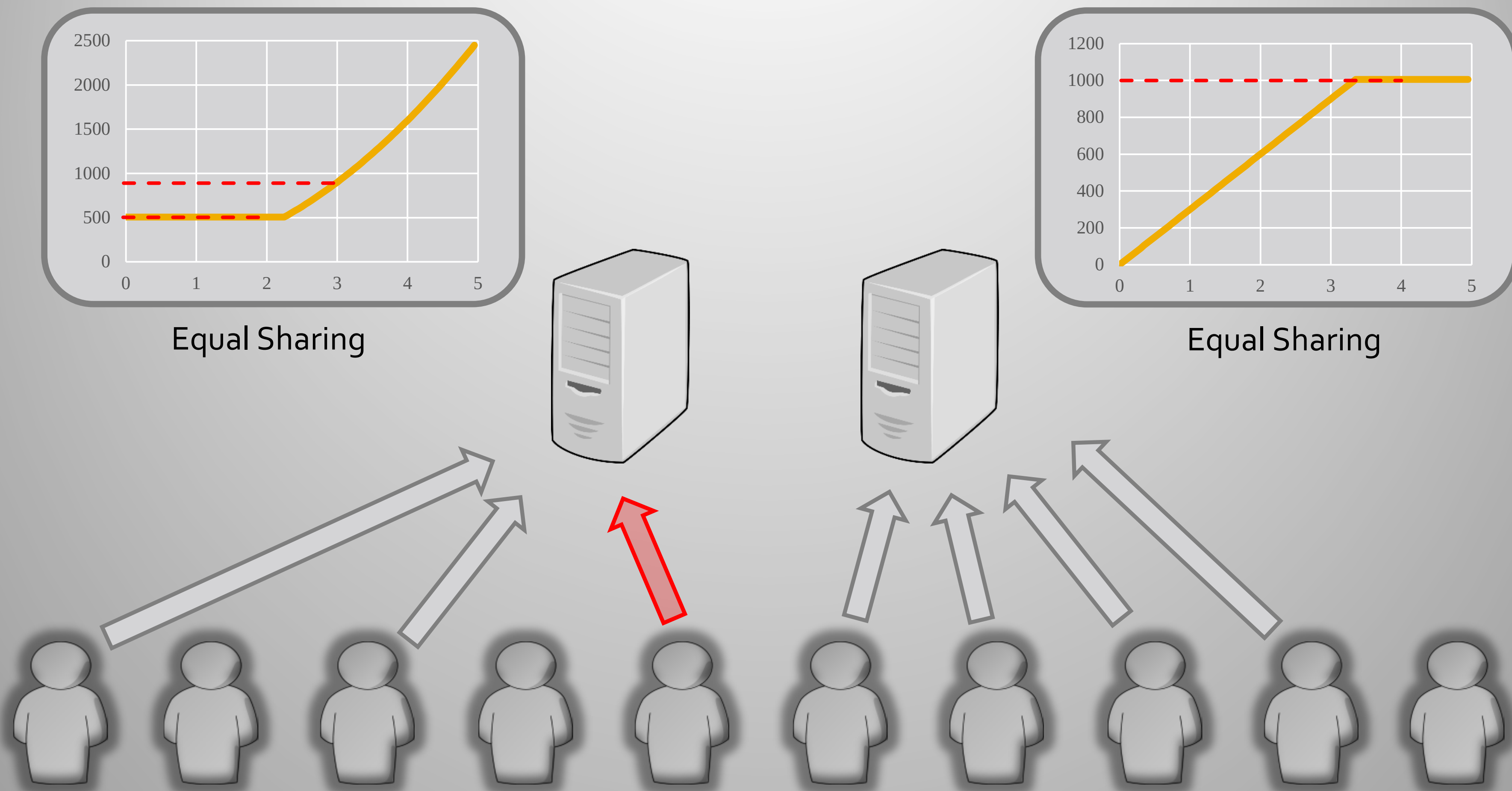




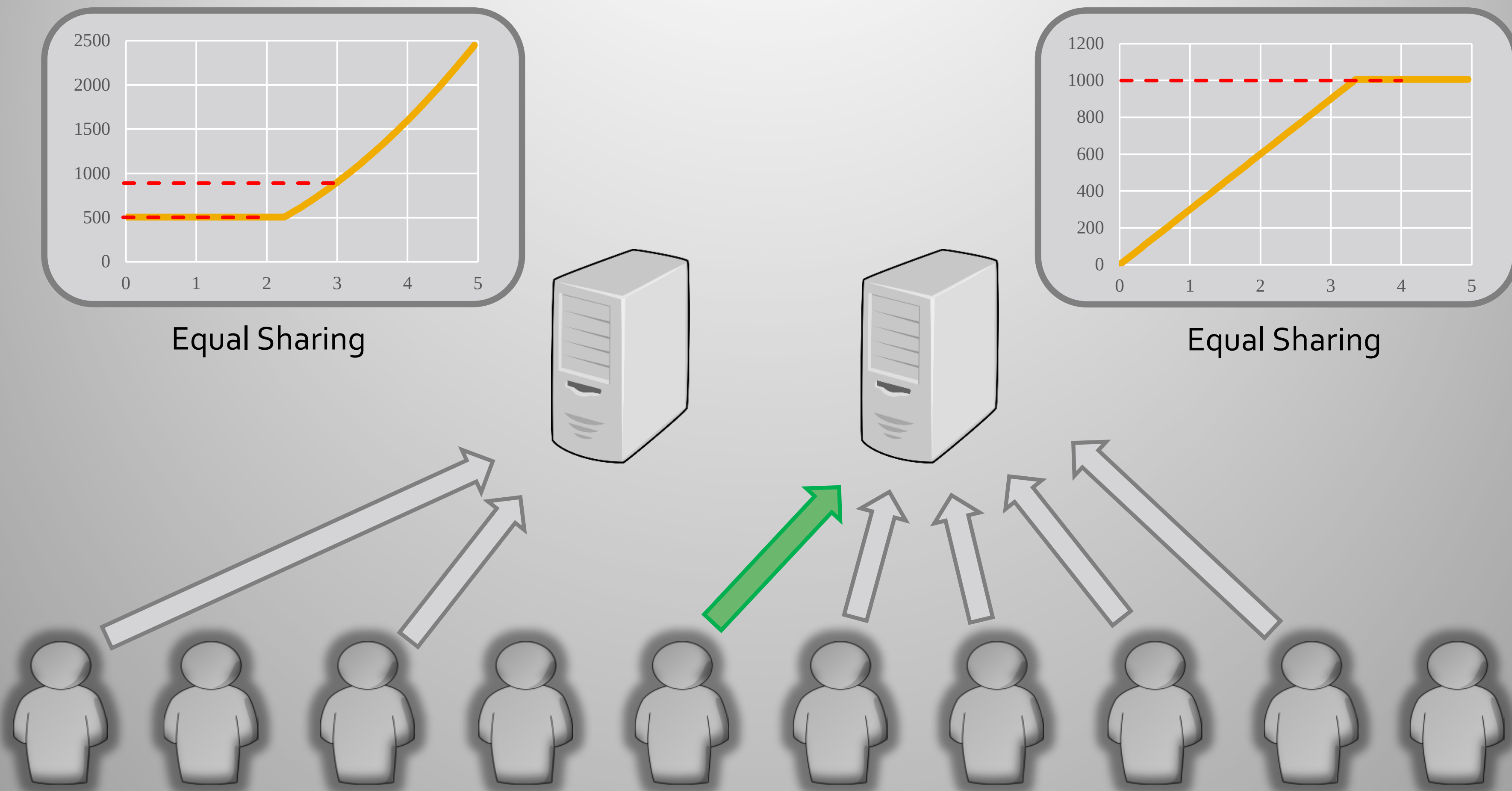


Scheduling Games

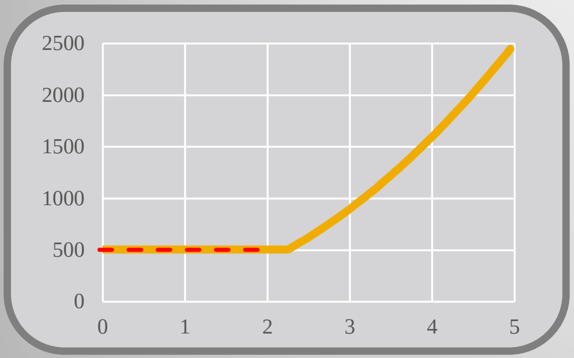
Scheduling Games



Scheduling Games

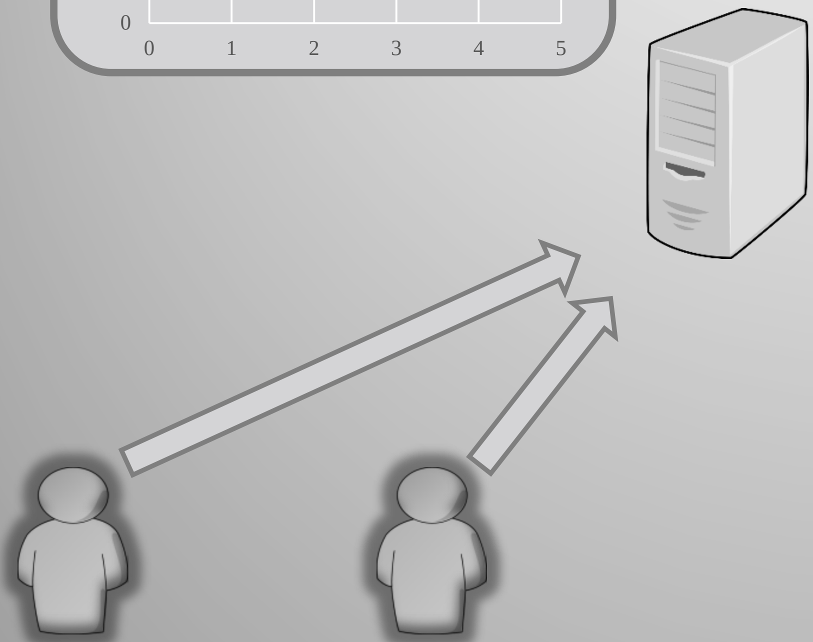


Oblivious Cost-Sharing

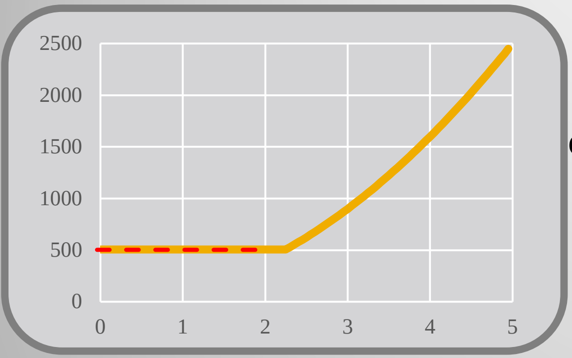


Oblivious Cost-Sharing

Each machine's policy sees only agents using it and is independent of other machines in the system

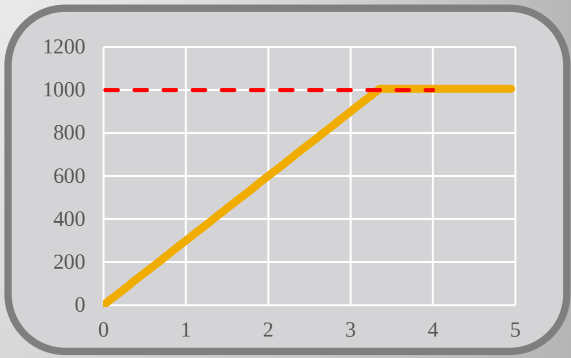


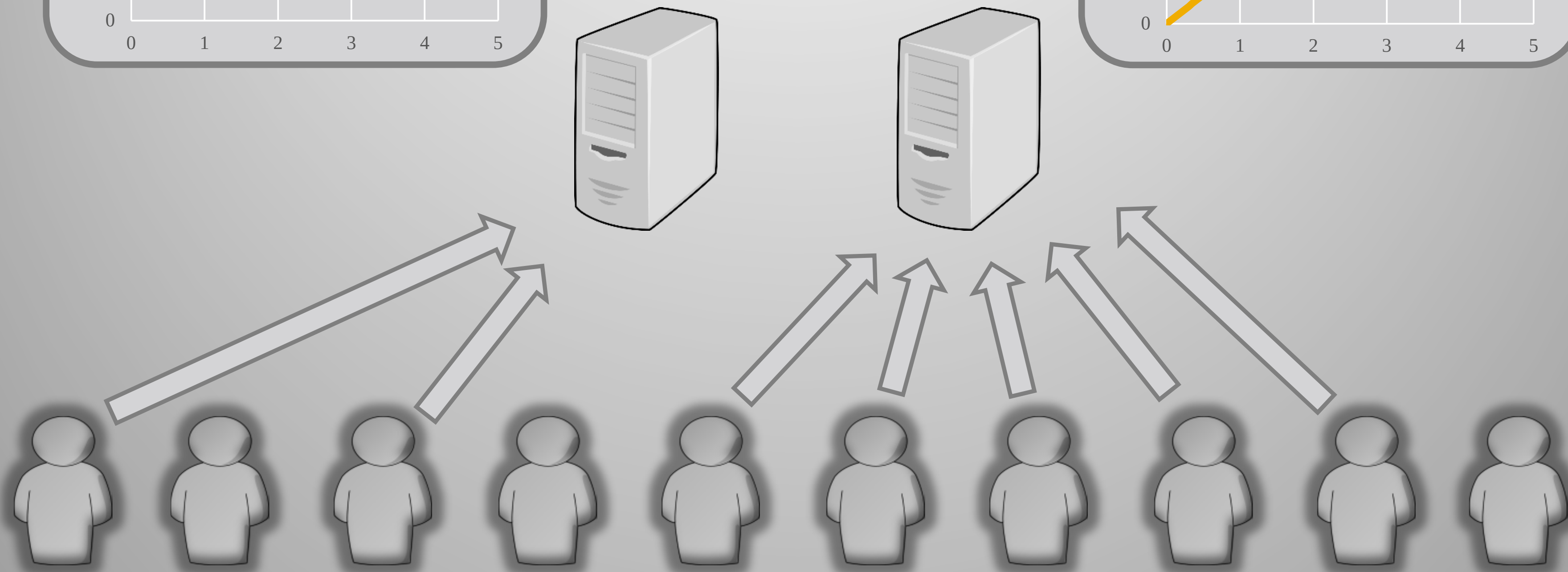
Omnipotent Cost-Sharing



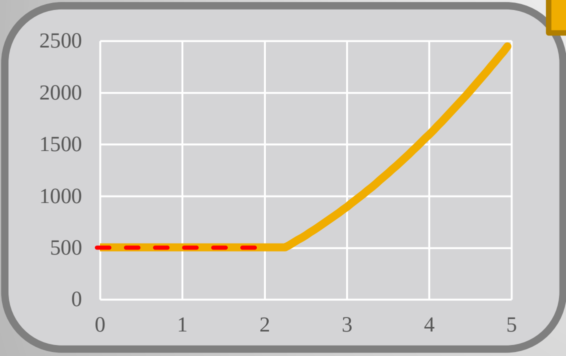
Omnipotent Cost-Sharing

Each machine's policy sees and can depend on all the agents and all the machines in the system



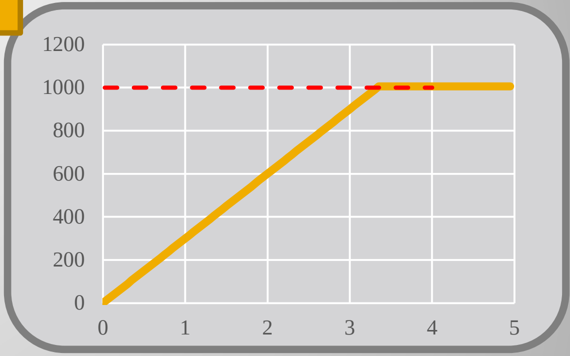


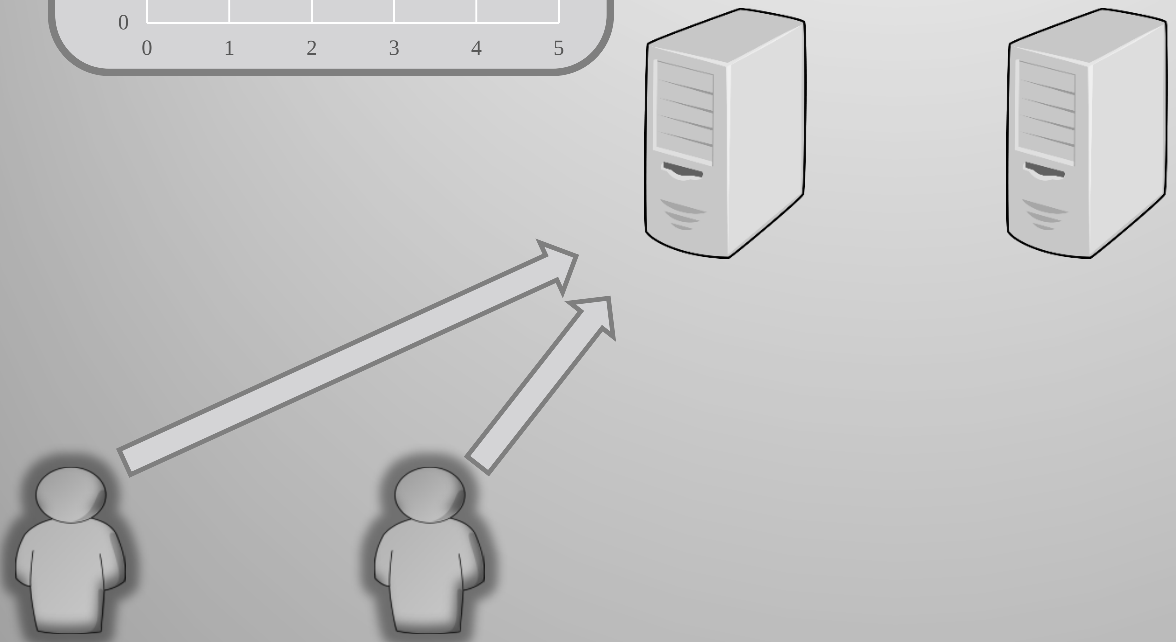
Resource-Aware Cost-Sharing



Resource-aware Cost-Sharing

Each machine's policy sees only agents using it but can depend on other machines in the system





Scheduling Games Model

- Set M of m machines and set N of n agents
- Agent i needs **one machine** to process a job with weight w_i
- Machine j has a cost function $c_j(\ell)$
  - This cost depends on the load $\ell = \sum_i w_i$ of the agents using it
  - The cost function satisfies $c_j(0) = 0$ for every machine j

- Strategy $s_i \in M$ from each job i leads to profile $\boldsymbol{s}$
- Let $S_j(s)$ be the set of jobs using machine j in profile $\boldsymbol{s}$
- Let $\ell_j(\boldsymbol{s}) = \sum_{i \in S_j(\boldsymbol{s})} w_i$ be the total load of the jobs using j in $\boldsymbol{s}$

- Cost-sharing method $\xi_{ij}(s)$ defines cost of i in profile $\boldsymbol{s}$
  - **Budget-balanced** if for every $\boldsymbol{s}$ and every j: $\sum_{i \in S_j(\boldsymbol{s})} \xi_{ij}(\boldsymbol{s}) = c_j(\boldsymbol{s})$
  - **Stable** if a pure Nash equilibrium exists for all sets M and N

Price of Anarchy

- We measure the efficiency of a schedule $\boldsymbol{s}$ using $C(\boldsymbol{s}) = \sum_{j \in M} c_j(\boldsymbol{s})$
- Given a class of games $\mathcal{G}$, for each game G ∈ $\mathcal{G}$:
  - Let $F(G)$ be the set of all possible schedules
  - Let $E(G) \subseteq F(G)$ be the set of pure Nash equilibria
- **Price of anarchy (PoA)** of a class $\mathcal{G}$ is: $\sup\limits_{G \in \mathcal{G}} \frac{\max_{\boldsymbol{s} \in \boldsymbol{E}(\boldsymbol{G})} C(\boldsymbol{s})}{\min_{\boldsymbol{s}^* \in F(G)} C(\boldsymbol{s}^*)}$
- We may **overcharge** so that $\sum_{i \in S_j(\boldsymbol{s})} \xi_{ij}(\boldsymbol{s}) = \hat{C}(\boldsymbol{s}) > C(\boldsymbol{s})$
- With overcharging, the PoA of a class $\mathcal{G}$ becomes: $\sup\limits_{G \in \mathcal{G}} \frac{\max_{\boldsymbol{s} \in \boldsymbol{E}(\widehat{\boldsymbol{G}})} \hat{C}(\boldsymbol{s})}{\min_{\boldsymbol{s}^* \in F(G)} C(\boldsymbol{s}^*)}$

Price of Anarchy

- We measure the efficiency of a schedule $\boldsymbol{s}$ using $C(\boldsymbol{s}) = \sum_{j \in M} c_j(\boldsymbol{s})$
- Given a class of games $\mathcal{G}$, for each game G ∈ $\mathcal{G}$:
  - Let $F(G)$ be the set of all possible schedules
  - Let $E(G) \subseteq F(G)$ be the set of pure Nash equilibri

Price of Stability (PoS)
if we change max to min

- **Price of anarchy (PoA)** of a class $\mathcal{G}$ is: $\sup_{G \in \mathcal{G}} \frac{\max_{\boldsymbol{s} \in \boldsymbol{E}(\boldsymbol{G})} C(\boldsymbol{s})}{\min_{\boldsymbol{s}^* \in F(G)} C(\boldsymbol{s}^*)}$
- We may **overcharge** so that $\sum_{i \in S_j(\boldsymbol{s})} \xi_{ij}(\boldsymbol{s}) = \hat{C}(\boldsymbol{s}) > C(\boldsymbol{s})$
- With overcharging, the PoA of a class $\mathcal{G}$ becomes: $\sup_{G \in \mathcal{G}} \frac{\max_{\boldsymbol{s} \in \boldsymbol{E}(\widehat{\boldsymbol{G}})} \hat{C}(\boldsymbol{s})}{\min_{\boldsymbol{s}^* \in F(G)} C(\boldsymbol{s}^*)}$

Related work

Many papers on **cost-sharing** and **coordination mechanisms**

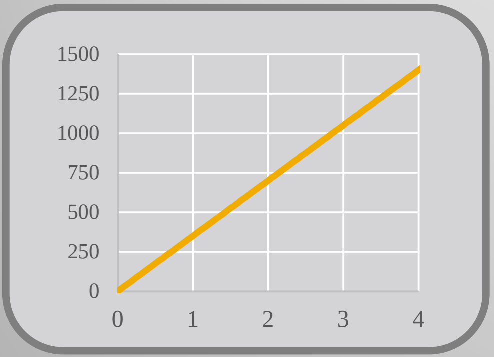
- **Chen, Roughgarden, and Valiant 2010**
  - Network design games (constant cost functions)
  - Agents can choose multiple machines
  - Characterization of stable cost-sharing protocols

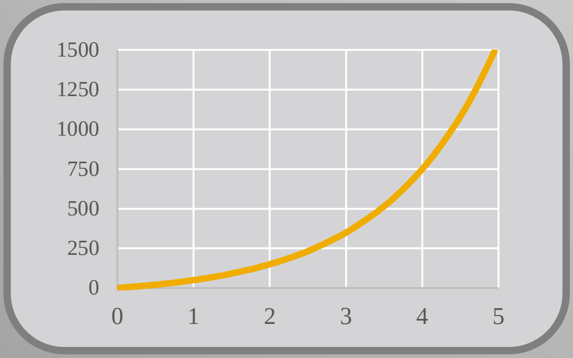
- **von Falkenhausen and Harks 2013**
  - Studied general cost functions
  - Also considered extension to matroids

- Both of these papers are restricted to **budget-balanced** protocols and the **omnipotent** and **oblivious** models

Convex Cost Functions

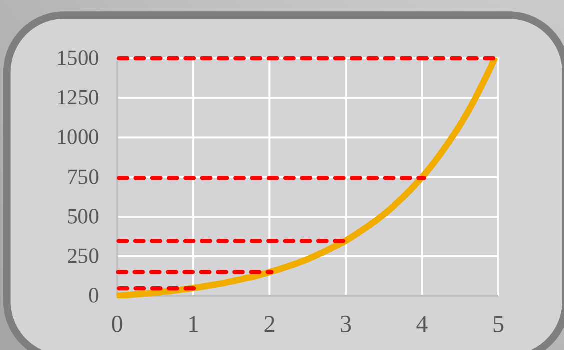
- Assume all the cost functions are convex
- How inefficient can the outcome be if we use equal sharing?
- E.g., **2 machines** $c_1(\ell) = 50 \cdot 2^{\ell-1}$ and $c_2(\ell) = 350\,\ell$ and **5 agents**

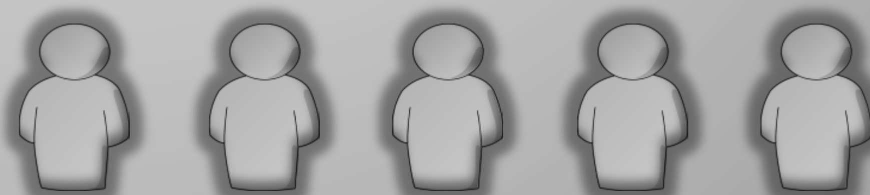




Convex Cost Functions

- Given global ordering π over the universe of agents
- **Incremental cost-sharing protocol** [Moulin '99]
  - Order the agents using a machine based on π
  - Charge each agent for a cost equal to its marginal contribution
  - $\xi_{ij}(\boldsymbol{s}) = c_j\left(\ell_j^{<i}(\boldsymbol{s}) + \boldsymbol{w_i}\right) - c_j\left(\ell_j^{<i}(\boldsymbol{s})\right)$
- This protocol is **stable**, **budget-balanced**, and **oblivious**, and it achieves a **PoA of 1** for unweighted agents and convex functions



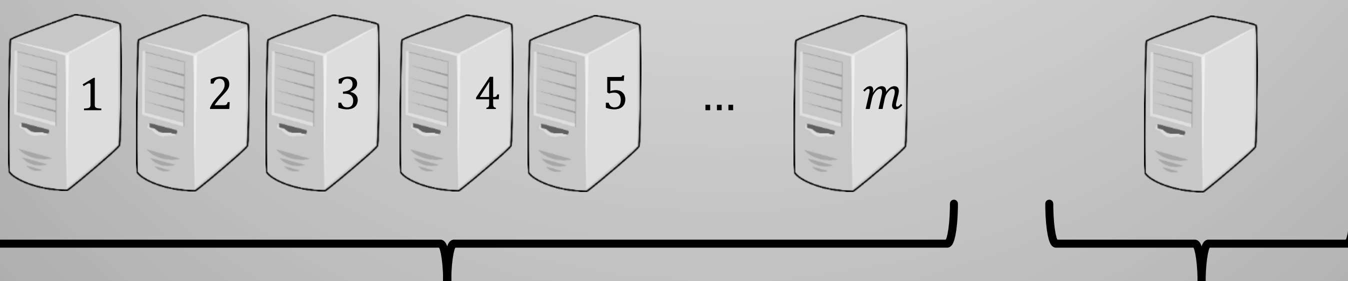


General Cost Functions

Theorem: Every **stable, budget-balanced, resource-aware** mechanism has a **PoA of $\boldsymbol{\Omega(m)}$** for general cost functions

- What if we allow the use of overcharging?

Theorem: Every **stable**, (**non-budget-balanced**,) **resource-aware** mechanism has a **PoA of $\boldsymbol{\Omega(\sqrt{m})}$** for general cost functions



m machines with $c_j(1) = 1$, $c_j(\ell) = \infty$ for $\ell > 1$

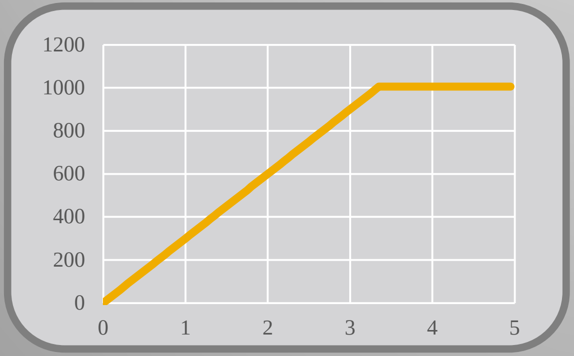
$c_j(\ell) = \sqrt{m}$ for $\ell \leq m$
$c_j(\ell) = \infty$ for $\ell > m$

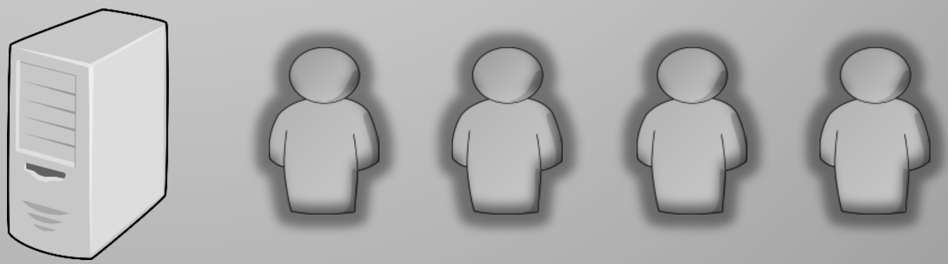
Concave Cost Functions

Theorem: Any **stable**, **oblivious**, and **budget-balanced** cost-sharing policy has **PoA** $\geq \boldsymbol{n}$ for strictly concave valuations

- **Observation**: optimal solution assigns all jobs to one machine, but which machine this is depends on the total load of the jobs

- The lower bound above is for unweighted, but our mechanism works for general weights as well





Concave Cost Functions

- $\boldsymbol{\phi}(\ell) = \min_{j \in M} c_j(\ell)$: smallest cost over all machines at load ℓ
- $\boldsymbol{X}_{\mathbf{min}}(\ell) = \arg\min_{j \in M} c_j(\ell)$: set of machines with cost $\phi(\ell)$ at load ℓ
- $\boldsymbol{h_j}(\boldsymbol{s}) = \arg\min_{i' \in S_j(s)} \{\pi(i')\}$: highest priority agent on machine j

$$\xi_{ij}(s) = \begin{cases} c_j\left(\ell_j(s)\right) & if\ j \notin X_{min}\left(\ell_j(s)\right)\ and\ \ i = h_j \\ 0 & if\ j \notin X_{min}\left(\ell_j(s)\right)\ and\ \ i \neq h_j \\ \phi(w_i) & if\ j \in X_{min}\left(\ell_j(s)\right)\ and\ \ i = h_j \\ w_i \dfrac{c_j\left(\ell_j(s)\right) - \phi(w_{h_j})}{\ell_j(s) - w_{h_j}} & if\ j \in X_{min}\left(\ell_j(s)\right)\ and\ \ i \neq h_j \end{cases}$$

Theorem: This cost-sharing mechanism is **stable**, **budget-balanced**, **resource-aware**, and it achieves **PoA of 1** for concave cost functions

Convex & Concave Functions

- No **budget-balanced** mechanism can guarantee a PoS better than $O(\log m)$ **even if it is omnipotent [vFH 13]**
- If $\boldsymbol{j} \in \boldsymbol{M_{convex}}$ and $\ell \geq \boldsymbol{n_j^{max}}$, instead of $c_j(\ell)$, we use cost functions $\hat{\boldsymbol{c}}(\ell) = \mathbf{max}\left\{\max_{j' \in M_{concave}} \boldsymbol{c_{j'}}(\mathbf{1}), \boldsymbol{c_j}(\ell)\right\}$
- **VC mechanism**: Using the over-charged cost functions above
  - For convex machines use incremental cost-sharing protocol
  - For concave machines use our concave cost-sharing protocol

Theorem: This mechanism is **stable**, **resource-aware** and achieves a **PoA of 2** for instances with convex and concave functions

Two-Machine Instances

- Let $\alpha(\boldsymbol{s})$ be highest priority agent using the first machine
- Let $\beta(\boldsymbol{s})$ be lowest priority agent using second machine

- **Increasing-Decreasing mechanism**: For any profile $\boldsymbol{s}$,
  - Charge agent $\alpha(\boldsymbol{s})$ for the whole cost of the first machine
  - Charge agent $\beta(\boldsymbol{s})$ for the whole cost of the second machine.

Theorem: This mechanism is **stable**, **budget-balanced**, **resource-aware**, and it achieves a **PoA of 2** for arbitrary cost functions

Note that this mechanism is stable, but not a Shapley value variant

Theorem: Any **stable**, (**non-budget-balanced**), **resource-aware** mechanism has **PoA** $> \mathbf{1.36}$, even for instances with just two machines with convex and concave cost functions

Conclusion

- **Resource-aware** cost-sharing
  - Well motivated middle-ground between omnipotent and oblivious
  - Non-trivial use of extra information may enable improvements
- Power of **over-charging**
  - Leads to improvements despite the additional costs
  - For omnipotent protocols, over-charging can yield PoA of 1
  - Budget-balanced omnipotent protocols have PoA $\Omega(\log n)$ [vFH 13]
- **Open problems**
  - Weighted upper bounds in our setting
  - Applying the resource-aware model in other settings

Thank you!