Fair Division III: Computational Resources

Teachers: Ariel Procaccia and Alex Psomas (this time)

THE ACTUAL PROBLEM

- We have a big (cloud) computing center.
  - The computing center is comprised of machines.
  - A machine has X amount of CPU, Y amount of RAM, Z amount of hard drive space, runs operating system W, etc etc
- People submit jobs (i.e. a piece of code) that they want to execute.
  - A job requires (at least) some job-specific resources to execute: X amount of CPU, Y amount of RAM, specific operating systems, specific libraries available etc.
- This computing center is internal (e.g. used internally by FB employees), so charging users is not an option.

OUR OBJECTIVE

- We want to allocate the resources in a way that is:
  - *Fair*.
    - For this talk by ``fair'' we mean proportional.
    - Proportional = ``better than $1/n$ of each resource''
  - *Efficient*.
    - In order to improve someone's utility, you have to hurt someone else.
  - *Strategy-proof.*
    - A user should not be able to benefit by lying about the amount of resources she needs.

STRATEGY-PROOF?

- Surprisingly very important!
- Some anecdotal evidence:
  - One of Yahoo!'s Hadoop MapReduce datacenters has different numbers of slots for map and reduce tasks. A user discovered that the map slots were contended, and therefore launched all his jobs as long reduce phases, which would manually do the work that MapReduce does in its map phase.
  - Big search company provided dedicated machines for jobs only if the users could guarantee high utilization. Users would sprinkle their code with infinite loops to artificially inflate utilization levels!

THE MODEL

- There are n users, m resources and a single machine.
- Users want resources in fixed proportions!
  - If you need 1 CPU and 2 GBs RAM to run a task, then getting 1 CPU and 3GBs RAM is the same as getting 1 CPU and 2 GBs RAM (but worse than getting (2,4)).
- $d_i = \left(d_{i,1}, d_{i,2}, \dots, d_{i,m}\right)$ is user i's demand vector.

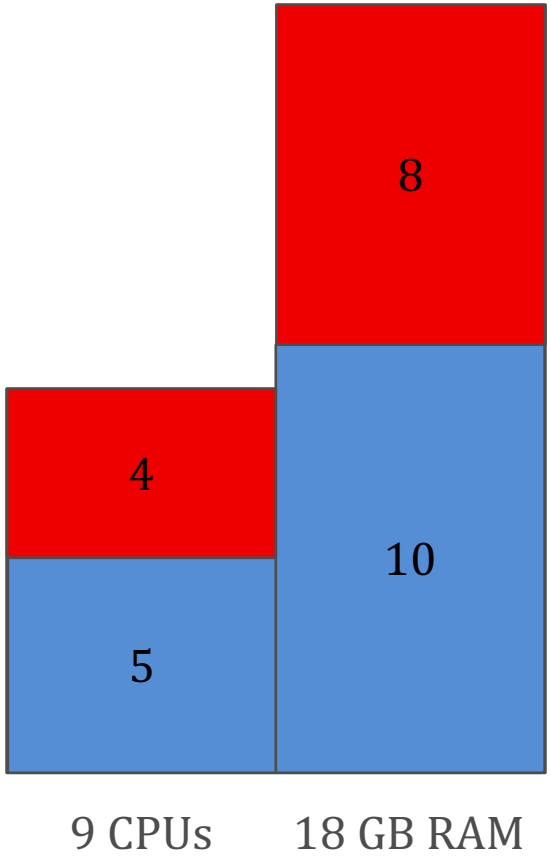
THE MODEL

- The utility of user i for a vector of resources $A_i = \left(A_{i,1}, A_{i,2}, \ldots, A_{i,m}\right)$ is
  - $u_i(A_i) = \max\{\, y \in \mathbb{R} : \forall\, r \in [m], A_{i,r} \geq y \cdot d_{i,r}\}$
  - More intuitively: the utility of an agent is how many times her demand vector ``fits'' in the resource vector she was given.

EXAMPLE

$d_1 = (1, 4)$ $u_1 = 2$

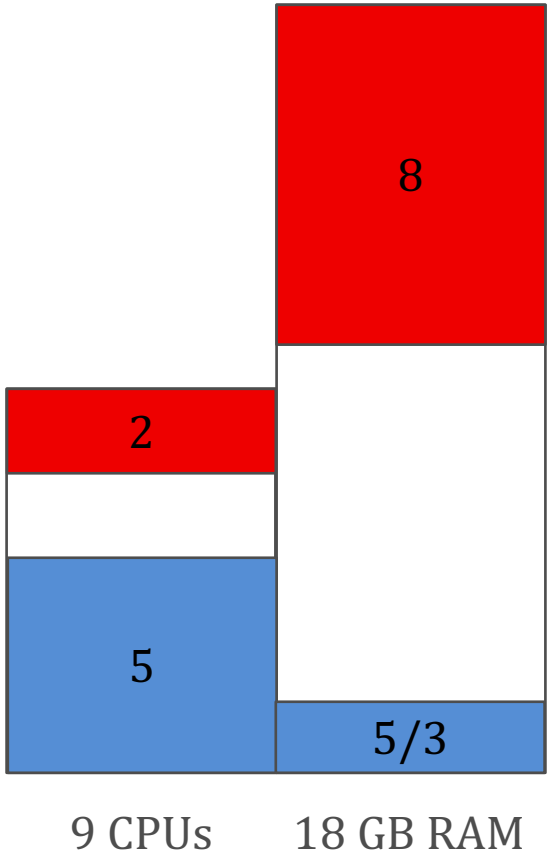
$d_2 = (3,1)$ $u_2 = \frac{5}{3}$



EXAMPLE

$d_1 = (1, 4)$ $u_1 = 2$

$d_2 = (3,1)$ $u_2 = \frac{5}{3}$



POLL

$d_1 = (2, 5)$

$d_2 = (2,1)$

60 CPUs 30 GB RAM

$d_3 = (2,1)$

How much utility does a proportional allocation guarantee?

1. $u_1 = u_2 = u_3 = 10$

3. $u_1 = 6, u_2 = u_3 = 30$

2. $u_1 = 2, u_2 = u_3 = 10$

CANDIDATE #1: ASSET FAIRNESS

- ``1% of CPU is worth the same as 1% of memory''
- Players have the same budget.
- Example:
  - 30 CPUs, 30 GBs RAM
  - 1 CPU is worth as much as 1 GB RAM.
  - $d_1 = (1,3), d_2 = (1,1)$
  - Agent 1 spends $1 + 3 = 4$ per task
  - Agent 2 spends $1 + 1 = 2$ per task
  - Let x be the number of tasks of agent 1, and y the number of tasks of agent 2.
  - $4x = 2y$
  - $x + y \leq 30$ and $3x + y \leq 30$.

CANDIDATE #1: ASSET FAIRNESS

- Solving gives $x = 6, y = 12$
- Agent 1: $(6, 18)$
- Agent 2: $(12, 12)$
- Is there a problem?
- The allocation is unfair.
- Agent 2 does better if we split everything half and half.

CANDIDATE #2: CEEI

- Maximize the product of utilities.
- Example:
  - 100 CPUs, 100 GBs RAM
  - $d_1 = (16,1), d_2 = (1,2)$
  - Maximize $u_1 \cdot u_2$
  - Give everything to agent 1: $u_1 = \frac{100}{16}, u_2 = 0$
    - Product is equal to zero
  - Half half: $u_1 = \frac{50}{16}, u_2 = \frac{50}{2}, u_1 \cdot u_2 \approx 78$
  - Optimal solution gives $\approx (51,3)$ to agent 1 and $\approx (49,97)$ to agent 2.
    - $\frac{100}{31} \approx 3.2$ tasks to agent 1 and $\frac{1500}{31} \approx 48.3$ tasks to 2.
    - $u_1 \cdot u_2 \approx 156$

CANDIDATE #2: CEEI

- Agent 1 reports $d_1' = (16,8)$.
- $d_1' = (16,8)$ is a symmetric and scaled version of $d_2 = (1,2)$.
- Maximizing the product is a scale free solution
  - (1,1) gives you the same stuff as (5,5)
- Agent 1 gets $\frac{2}{3}$ of the CPUs and agent 2 gets $\frac{2}{3}$ of the RAM.
- Agent 1 can now execute $(100 \cdot \frac{2}{3})/16 \approx 4.2$ tasks.
  - Maximizing the product is not strategyproof!

DOMINANT RESOURCE FAIRNESS (DRF)

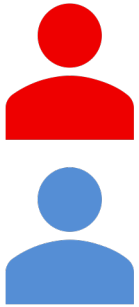
- Maximize utilities subject to equal dominant shares.
  - Dominant share of agent i = amount of dominant resource.
  - Dominant resource = resource for which the agent's task requires the largest fraction of total availability.

$d_1 = (1, 4)$

Machine = (9,18)

Dominant resource

DRF



$d_1 = (1, 4)$

Machine = (9,18)

$d_2 = (3,1)$

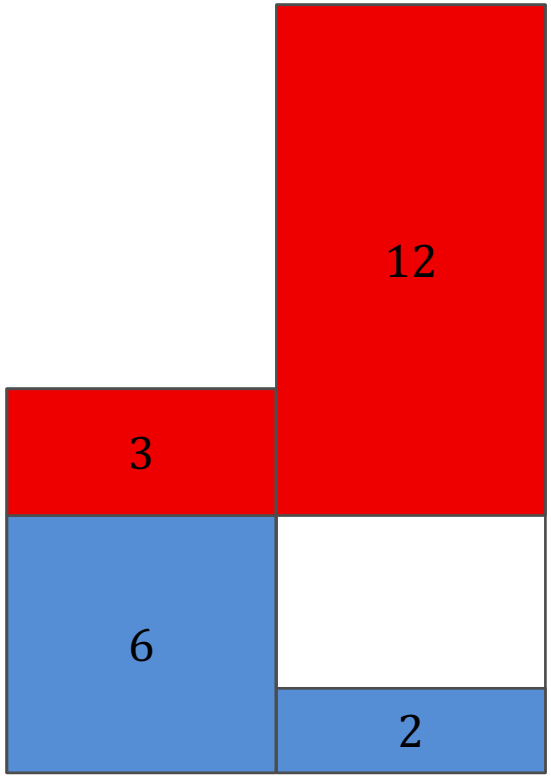
- In order to get utilities u_1 and u_2, we'll have to use up $u_1 + 3u_2$ CPUs and $4u_1 + u_2$ GBs RAM.
- Dominant shares are $\frac{4}{18}u_1$ and $\frac{3}{9}u_2$.
- Constraints:
  - $u_1 + 3u_2 \leq 9$
  - $4u_1 + u_2 \leq 18$
  - $\frac{4}{18}u_1 = \frac{3}{9}u_2$
- Objective: Maximize $u_1 + u_2$ (or just u_1, or just u_2)
- Solution: $u_1 = 3$ and $u_2 = 2$.

DRF

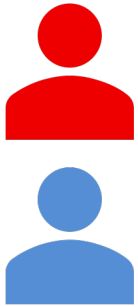
$d_1 = (1, 4)$ $u_1 = 3$

Dominant share $= \frac{4}{18} \cdot 3 = \frac{3}{9} \cdot 2 = \frac{2}{3}$

$d_2 = (3,1)$ $u_2 = 2$



DRF: EASIER THAN IT LOOKS



$d_1 = (1, 4)$

Machine = (9,18)

$d_2 = (3,1)$

- Simplify demands.
  - $D_{i,r} = \frac{d_{i,r}}{m_r}$. So, $D_{1,1} = \frac{1}{9}$, $D_{1,2} = \frac{4}{18}$, …
  - New demand: $d_{i,r} = \frac{D_{i,r}}{\max_{r'} D_{i,r'}}$.
    - So, $d_{1,1} = \frac{1/9}{4/18} = 1/2$, $d_{1,2} = 1$, $d_{2,1} = 1$, $d_{2,2} = 1/6$
- Simplified linear program.
  - Maximize x (dominant share)
  - Subject to: $\forall r, \sum_i x \cdot d_{i,r} \leq 1$

DRF: EASIER THAN IT LOOKS

- $x = \frac{1}{\max_r \sum_i d_{i,r}}$
  - In the example: , $d_1 = (1/2,\ \ 1), d_2 = (1, 1/6)$
  - $x = \frac{1}{\max\{\, d_{1,1} + d_{2,1}\, , d_{1,2} + d_{2,2}\}} = \frac{2}{3}$
- Agent 1 gets 2/3 of resource 2, agent 2 gets 2/3 of resource 1.
- Normalized utilities: $u_1 = u_2 = \frac{2}{3} = x$.

PROPERTIES OF DRF

- *Efficient*
  - In order to improve someone's utility, you have to hurt someone else.
  - Proof: maximizes utilities.

PROPERTIES OF DRF

- *"Fair"*
  - Lemma: DRF is proportional, i.e. utility of agent i is at least her utility for $1/n$ of the resources.
  - Proof:
  - $n \geq \max_r \sum_i d_{i,r}$: the RHS is maximized when every term of the sum is 1.
  - $u_i = x = \frac{1}{\max_r \sum_i d_{i,r}} \geq \frac{1}{n}$
  - Lemma: DRF is envy-free.

PROPERTIES OF DRF

- *Truthful.*
  - *Proof:*
  - Say report was d_i' instead of d_i.
  - $x = \frac{1}{\max_r \sum_i d_{i,r}}, x' = \frac{1}{\max_r \sum_i d'_{i,r}}$.
  - If $x' \leq x$, then $\forall r, x' \cdot d'_{i,r} \leq x \cdot 1 = x \cdot d_{i,r_i^*}$, where r_i^* is the dominant resource.
  - So, no improvement in i's utility.

TRUTHFULNESS PROOF CONTINUED

- Otherwise $x' > x$. Let $A_{i,r}$ be the amount of resource r allocated to player i when reporting d_i, and $A'_{i,r}$ when reporting d'_i.
- Let r be a resource that was fully allocated in A.

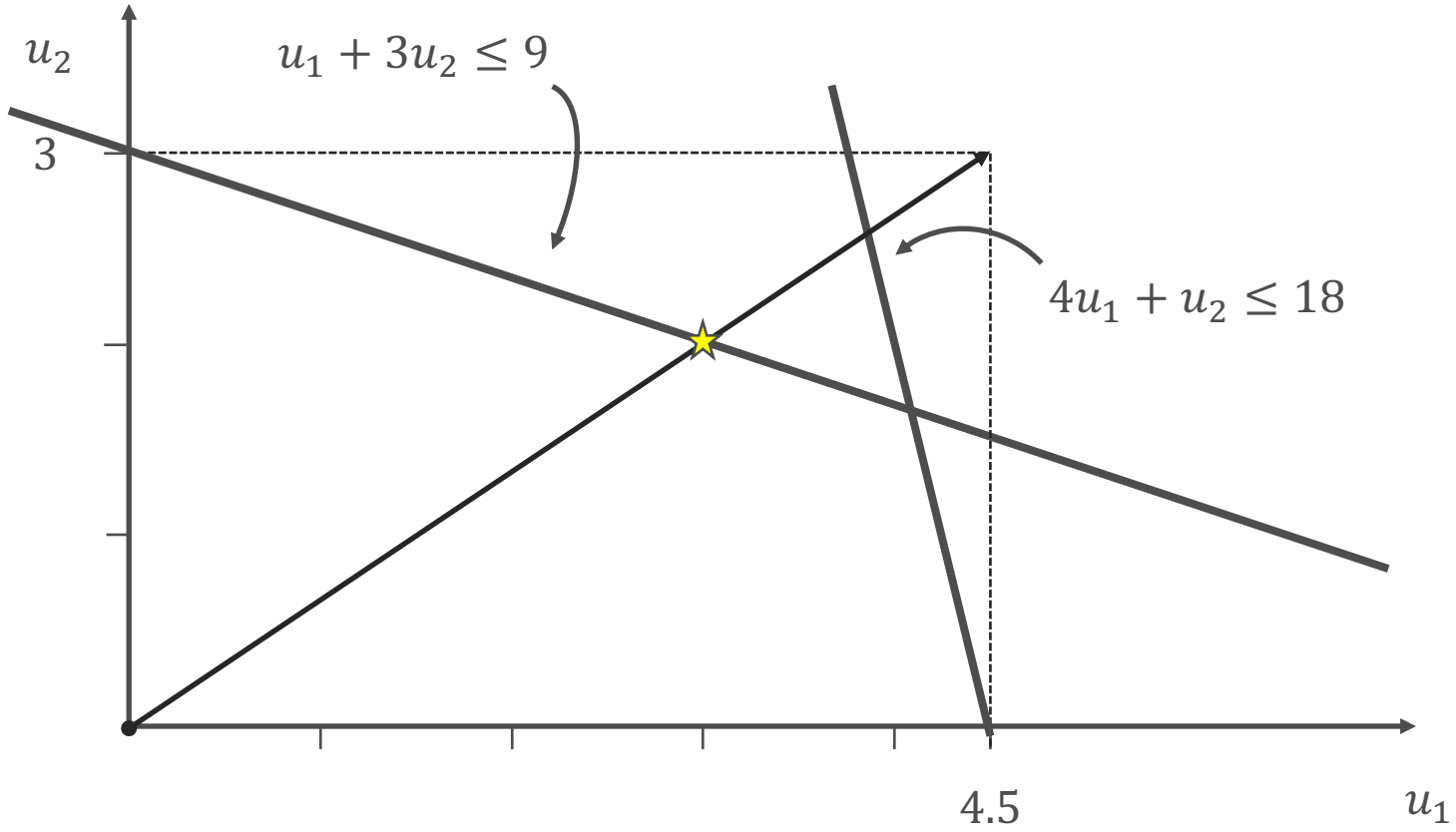
$$\begin{aligned} A_{i,r} &= 1 - \sum_{j \neq i} A_{j,r} \\ &= 1 - \sum_{j \neq i} x \cdot d_{j,r} \\ &> 1 - \sum_{j \neq \mathrm{i}} x' \cdot d_{j,r} \\ &= 1 - \sum_{j \neq i} A'_{j,r} \\ &\geq A'_{i,r} \end{aligned}$$

KALAI-SMORODINSKY BARGAINING SOLUTION

$d_1 = (1, 4)$

Machine = (9,18)

$d_2 = (3,1)$



BEYOND DRF

- Zero demands, weights.
  - Easy-ish to fix
- Indivisible tasks! [Parkes, Procaccia, Shah 2012].
  - If you need 1 CPU and 1 GB RAM per task, getting 1.5 of each is the same as getting 1 of each.
  - Under indivisibilities you can't have efficiency, truthfulness and proportionality at the same time.

INDIVISIBLE TASKS: A LOWER BOUND

- Proportionality, efficiency and truthfulness are incompatible.
  - Proof:
  - 2 agents, 1 resource. $d_1 = d_2 = \frac{1}{2} + \epsilon$.
  - Efficiency → someone gets a task. Wlog it's agent 1.
  - $d'_2 = \frac{1}{2}$
  - Proportionality →agent 2 must get half the resource.
  - Efficiency → 2 gets the other half as well. (useless for 1)
  - Agent 2 gained by lying.

BEYOND DRF

- Zero demands, weights.
  - Easy-ish to fix
- Indivisible tasks! [Parkes, Procaccia, Shah 2012].
  - If you need 1 CPU and 1 GB RAM per task, getting 1.5 of each is the same as getting 1 of each.
  - Under indivisibilities you can't have efficiency, truthfulness and proportionality at the same time.
- Multiple machines!
  - Assumption: under-reporting gives zero utility. (very well motivated)
  - There exists a randomized mechanism (extension of the Kalai-Smorodinsky interpretation) that satisfies all the properties (ex-ante). [Friedman, Ghodsi, P 2014].
- Huge assumption: one shot game!
  - If we have a solution for time t and a new job arrives at time t+1, how do we get to a new solution?

BACK TO THE DRAWING BOARD: DYNAMIC FAIR DIVISION

- Agents arrive over time.
- When an agent arrives we give her resources.
- No reallocation allowed [Kash, Procaccia, Shah 2013].
- When an agent arrives we can take resources from a fixed number of agents [Friedman, P, Vardi 2015,2017].
  - Optimize the ``fairness ratio'':

$$\min_t \frac{\min_i u_i^t}{\text{``ideal allocation at } t\text{''}}$$

REFERENCES

- Dominant Resource Fairness: Fair Allocation of Multiple Resource Types. Ghodsi, Zaharia, Hindman, Konwinski, Shenker, Stoica 2011
- Beyond Dominant Resource Fairness: Extensions, Limitations, and Indivisibilities. Parkes, Procaccia, Shah 2012
- No agent left behind: Dynamic fair division of multiple resources. Kash, Procaccia, Shah 2013
- Strategyproof Allocation of Discrete Jobs on Multiple Machines. Friedman, Ghodsi, Psomas 2014
- Controlled Dynamic Fair Division. Friedman, Psomas, Vardi 2017