Lecture 2: Computation of CE

ADFOCS 2020

25^{th} August 2020

Ruta Mehta

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

(Recall) Fisher's Model

- Set A of n agents. Set G of m divisible goods.

- Each agent i has
  - budget of B_i euros
  - valuation function $v_i: R_+^m \to R_+$ over bundles of goods.

    Linear: for bundle $x_i = (x_{i1}, \ldots, x_{im})$, $v_i(x_i) = \sum_{j \in G} v_{ij} x_{ij}$

- Supply of every good is one.

(Recall) Competitive Equilibrium

Pirces $p = (p_1, \dots, p_m)$ and allocation $X = (x_1, \dots, x_n)$

- Optimal bundle: Agent i demands

$$x_i \in \underset{x \in R_m^+ :\, p \cdot x \leq B_i}{\operatorname{argmax}} v_i(x)$$

- Market clears: For each good j,

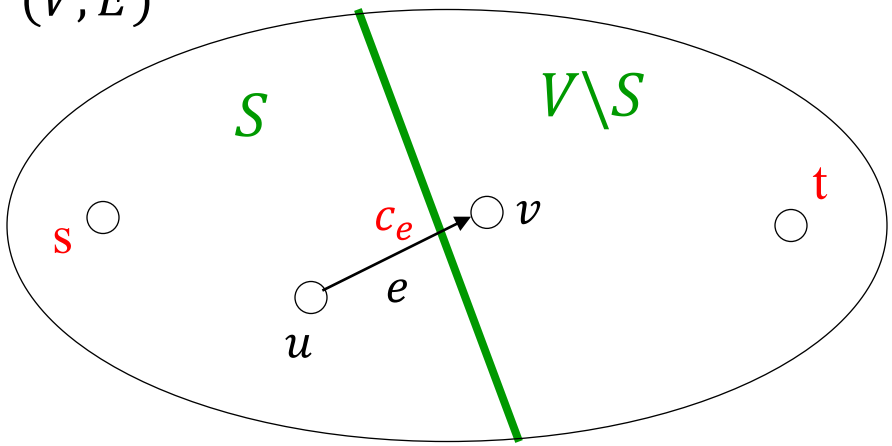
$$\text{demand} = \text{supply}$$

Fairness and efficiency guarantees:
Pareto optimal (PO)
Weighted Envy-free
Weighted Proportional
Maximizes W. NW.

Algorithm: Set up as a "flow problem"

Max Flow (One slide overview)

Directed Graph
(V, E)



Given $s, t \in V$. Capacity c_e for each edge $e \in E$.

Find maximum flow from s to t, $(\boldsymbol{f_e})_{e \in E}$ s.t.

- Capacity constraint

  $$f_e \leq c_e, \ \forall e \in E$$

- Flow conservation: at every vertex $u \neq s, t$

  total in-flow = total out-flow

Theorem: Max-flow = Min-cut
s-t s-t

s-t cut: $S \subset V,\ s \in S,\ t \notin S$

cut-value: $C(S) = \sum_{\substack{(u,v) \in E: \\ u \in S, v \notin S}} c_{(u,v)}$

Min s-t cut: $\min_{\substack{S \subset V: \\ s \in S, t \notin S}} C(S)$

Can be solved in *strongly* polynomial-time

CE Characterization

Pirces $p = (p_1, \dots, p_m)$ and allocation $X = (x_1, \dots, x_n)$

- Optimal bundle: Agent i demands $x_i \in \underset{x:\, p \cdot x \le B_i}{\operatorname{argmax}} v_i(x)$
  - $p \cdot x_i = B_i$
  - $x_{ij} > 0 \Rightarrow \frac{v_{ij}}{p_j} = \max_{k \in G} \frac{v_{ik}}{p_k}$, for all good j

- Market clears: For each good j, demand = supply

$$\sum_i x_{ij} = 1.$$

Competitive Equilibrium → Flow

Pirces $p = (p_1, \ldots, p_m)$ and allocation $F = (f_1, \ldots, f_n)$

$$f_{ij} = x_{ij} p_j \text{ (money spent)}$$

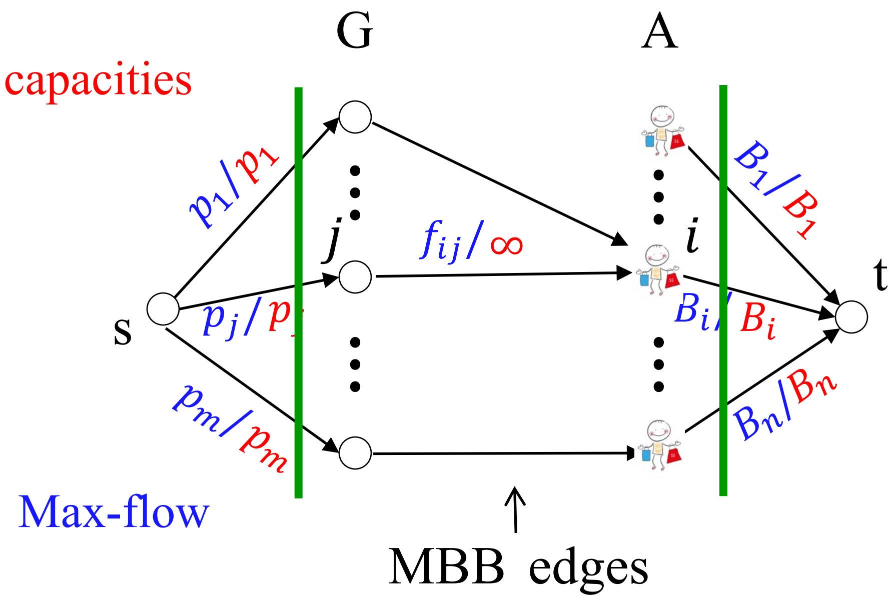
- Optimal bundle: Agent i demands $x_i \in argmax_{x:\, p \cdot x \leq B_i} v_i(x)$
  - $\sum_{j \in G} f_{ij} = B_i$
  - $f_{ij} > 0 \Rightarrow \frac{v_{ij}}{p_j} = \boxed{\max_{k \in G} \frac{v_{ik}}{p_k}}$ for all good j

Maximum bang-per-buck (MBB)

- Market clears: For each good j, demand = supply

$$\sum_{i \in N} f_{ij} = p_j$$

Competitive Equilibrium → Flow



CE: (p, F) s.t.

$$\sum_{i \in N} f_{ij} = p_j \quad \sum_{j \in M} f_{ij} = B_i$$

$f_{ij} > 0$ on MBB edges

Max-flow = min-cut
$= \sum_{j \in G} p_j = \sum_{i \in A} B_i$

Issue: Eq. prices and hence also MBB edges not known!

Fix [DPSV'08]: Start with low prices, keep increasing.

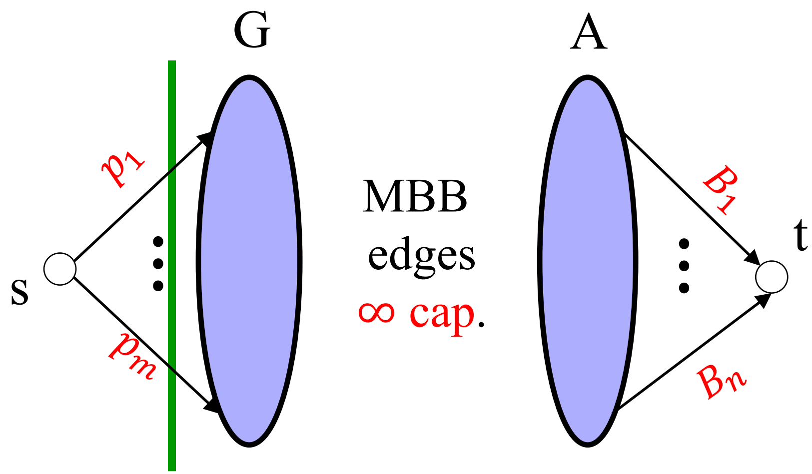
Maintain:

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are fully sold)

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)

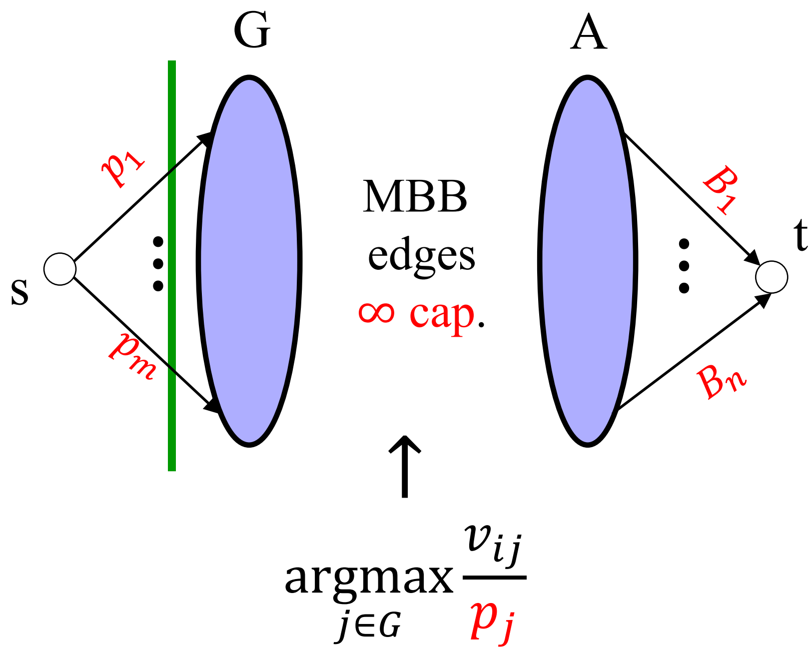


Init: $\forall j \in \mathrm{G},\ p_j < \min_i \frac{B_i}{m}$, and at least one MBB edge to j

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



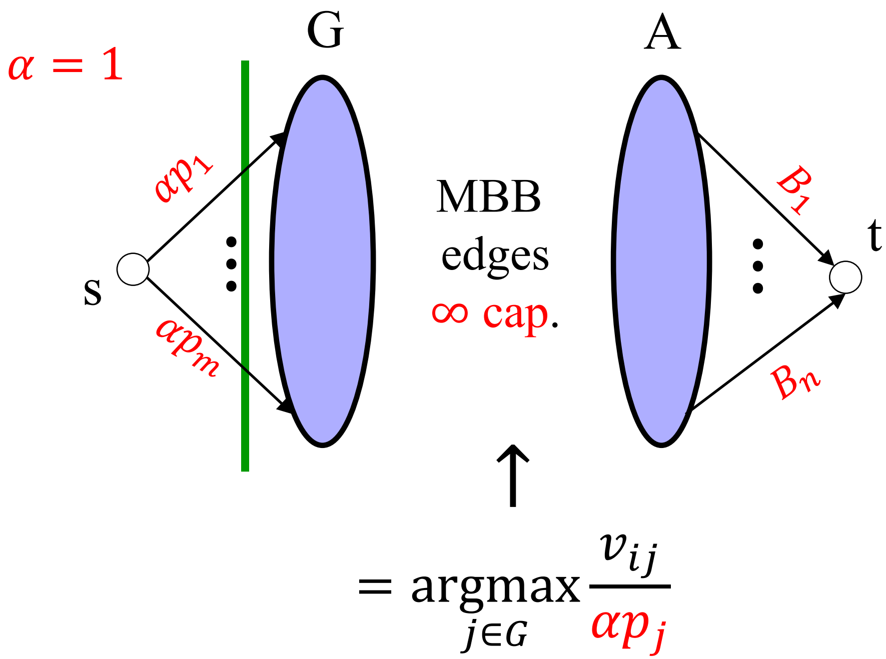
Init: $\forall j \in G,\ p_j < \min_i \frac{B_i}{m}$, and at least one MBB edge to j

Increase p:

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



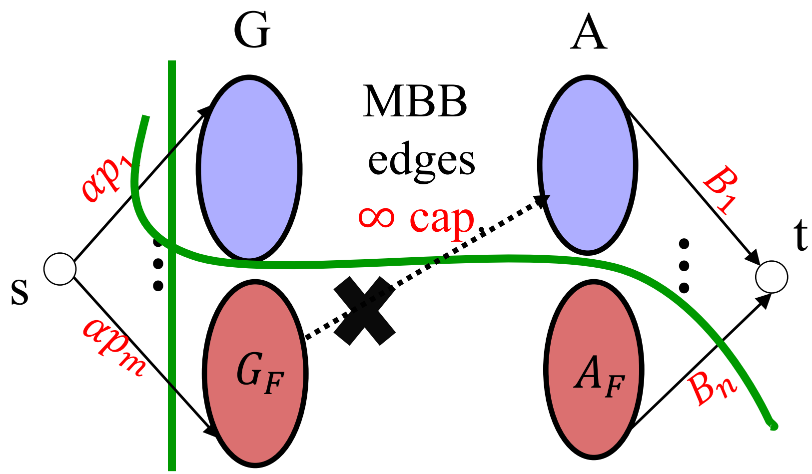
Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$

And at least one MBB edge to j

$\boxed{\textbf{Increase } \boldsymbol{p}: \uparrow \alpha}$

Algorithm (Pictorial)

Invariants
1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$
And at least one MBB edge to j

Increase p: $\uparrow \alpha$

Event 1: New cross-cutting min-cut

Agents in A_F exhaust all their money.

G_F: Goods that have MBB edges only from A_F.

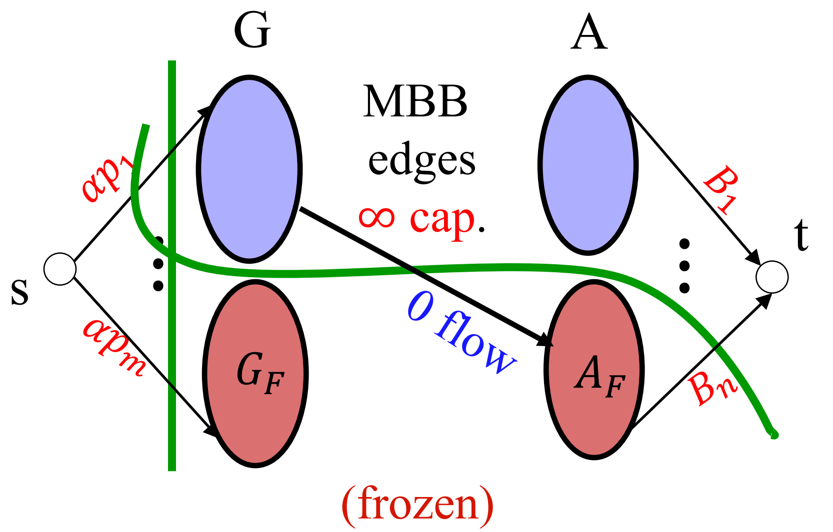
Observation: If α is increased further, then G_F can not be fully sold. And $\{s\}$ will cease to be a min-cut.

A tight-set.

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$

And at least one MBB edge to j

Increase p: $\uparrow \alpha$

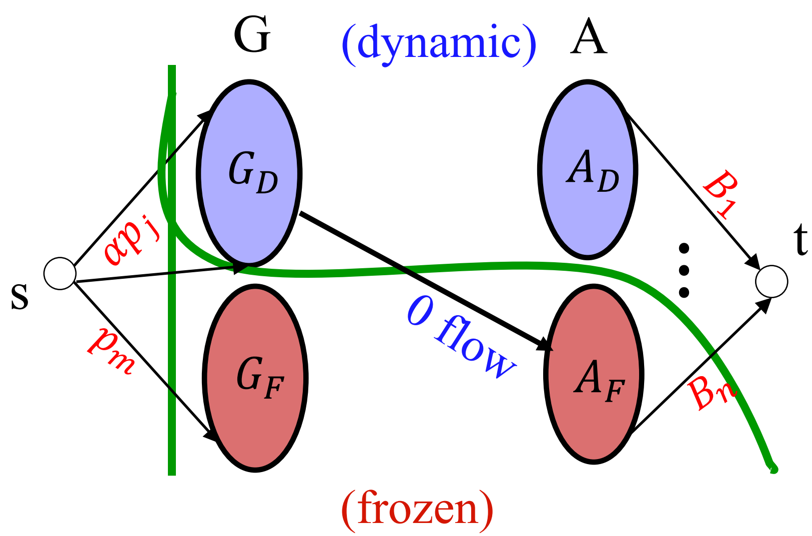
Event 1: A tight subset G_F

Call it *frozen:* (G_F, A_F).

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$

And at least one MBB edge to j

Increase *p*: $\uparrow \alpha$

Event 1: A tight subset G_F

Call it *frozen:* (G_F, A_F).

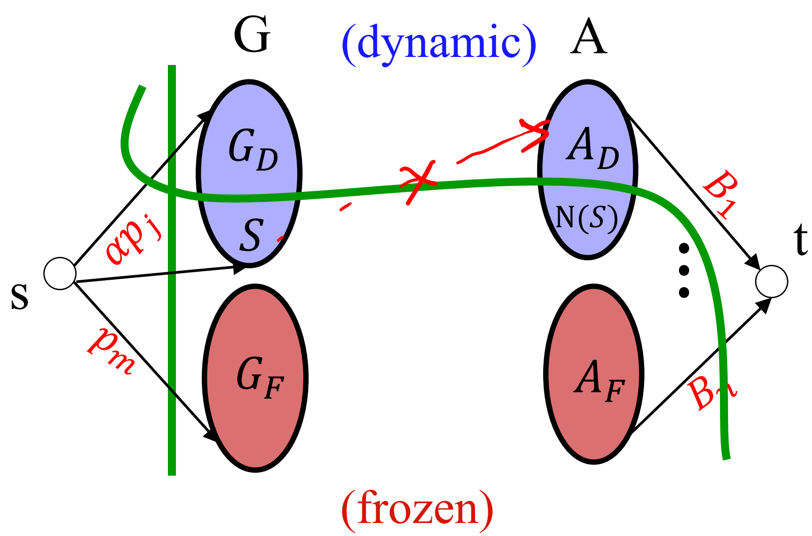
Freeze prices in G_F.

Increase prices in G_D.

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$
And at least one MBB edge to j

Increase $\boldsymbol{p}$: $\uparrow \alpha$

Event 1: A tight subset $S \subseteq G_D$

$N(S)$: Neighbors of S

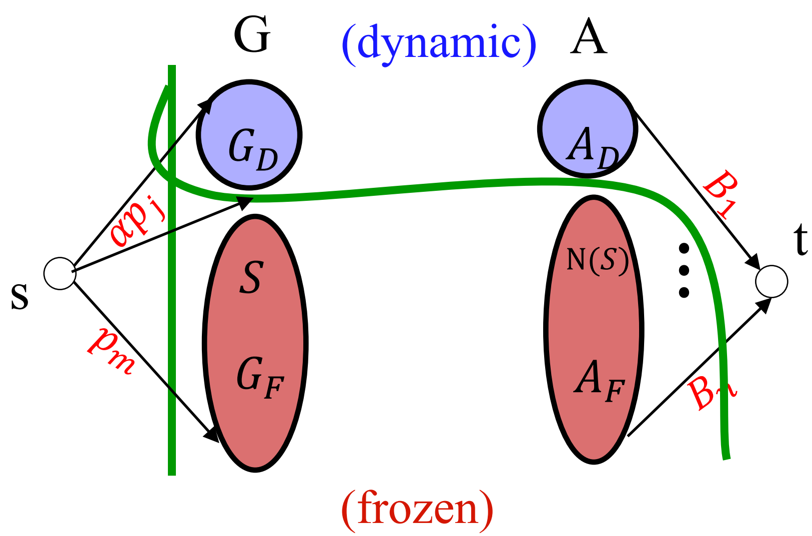
Move $(S, N(S))$ from dynamic to frozen.

Observation: If α is increased further, then **S** can not be fully sold. And $\{s\}$ will cease to be a min-cut.

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$

And at least one MBB edge to j

Increase p: $\uparrow \alpha$

Event 1: A tight subset $S \subseteq G_D$

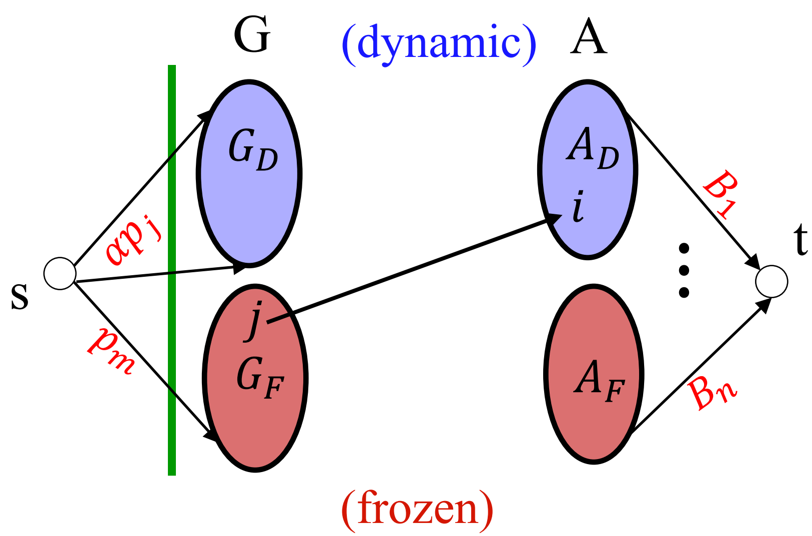
Move $(S, N(S))$ to frozen part

Freeze prices in G_F, and increase in G_D.

Algorithm (Pictorial)

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$
And at least one MBB edge to j

Increase p: $\uparrow \alpha$

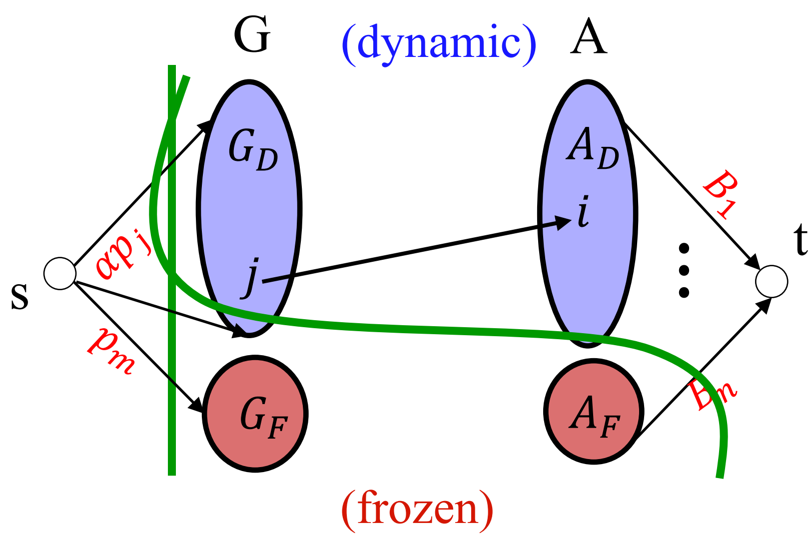
Event 1: A tight subset $S \subseteq G_D$
Move $(S, \mathrm{N}(S))$ from active to frozen
Freeze prices in G_F, and
increase in G_D.

OR

Event 2: New MBB edge
Must be between $i \in A_D$ & $j \in G_F$.
Recompute active and frozen.

Algorithm (Pictorial)

Invariants
1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$
And at least one MBB edge to j

Increase p: $\uparrow \alpha$

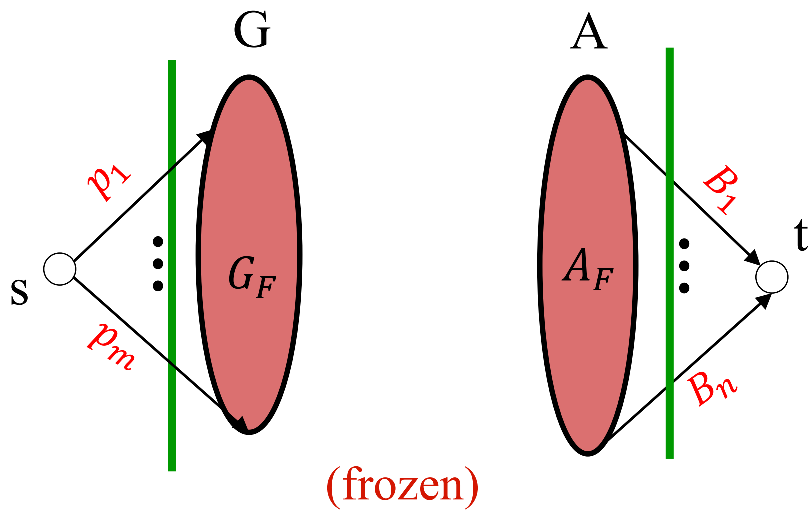
Event 1: A tight subset $S \subseteq G_D$
Move $(S, \mathrm{N}(S))$ from active to frozen
Freeze prices in G_F, and increase in G_D.

OR

Event 2: New MBB edge
Has to be from $i \in A_D$ to $j \in G_F$.
Recompute active and frozen:
Move the component containing good j from frozen to active.

Algorithm (Pictorial)

Invariants
1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)



Init: $\forall j \in M,\ p_j < \min_i \frac{B_i}{n}$
And at least one MBB edge to j

Increase p: $\uparrow \alpha$

Event 1: A tight subset $S \subseteq G_D$
Move $(S, \mathrm{N}(S))$ from active to frozen.
Freeze prices in G_F, and
increase in G_D.

OR

Event 2: New MBB edge
Must be from $i \in A_D$ to $j \in G_F$.
Recompute active and frozen.

Stop: all goods are frozen.

Observations: Prices only increase.
Each increase can be lower bounded.
Both the events can be computed efficiently.

⇓

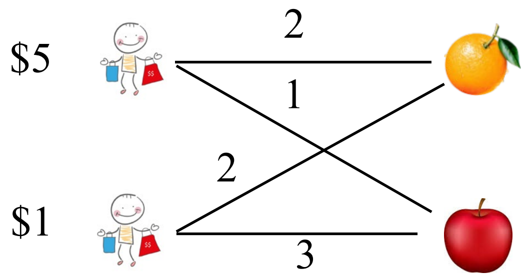
Converges to CE in finite time.

Example

Invariants

1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)

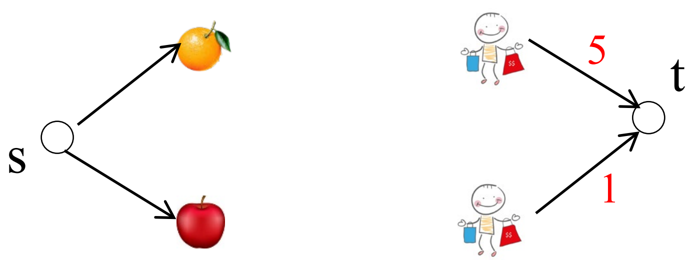
Input



Init.

Event 1

Event 2



Formal Description

- Init: $p \leftarrow$ "low-values" s.t. $\{s\}$ is a min-cut. $(G_D, A_D) \leftarrow (G, A),\ (G_F, A_F) \leftarrow (\emptyset, \emptyset)$
- While$(G_D \neq \emptyset)$
  - $\alpha \leftarrow 1,\ p_j \leftarrow \alpha p_j\ \ \forall j \in G_D$. Increase α until

    Event 1: Set $S \subseteq G_D$ becomes tight.

    $\mathrm{N}(S) \leftarrow$ agents w/ MBB edges to S (neighbors).

    Move $(\mathrm{S}, \mathrm{N}(\mathrm{S}))$ from (G_D, A_D) to (G_F, A_F).

    Event 2: New MBB edge appears between $i \in A_D$ and $j \in G_F$

    Add $(j \rightarrow i)$ edge to graph.

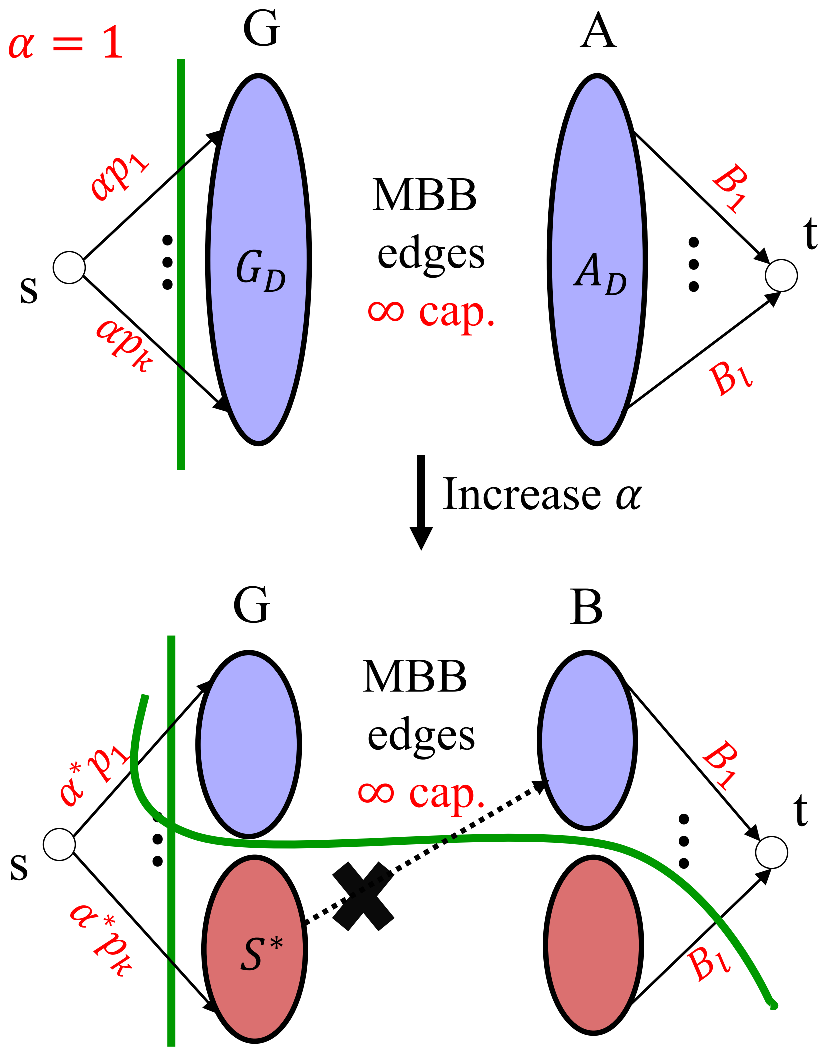
    Move component of j from (G_F, A_F) to (G_D, A_D).
- Output (p, F)

Efficiently Computing Event 2

Event 2: New MBB edge appears between $i \in A_D$ and $j \in G_F$

Exercise ☺

Efficiently Computing Event 1



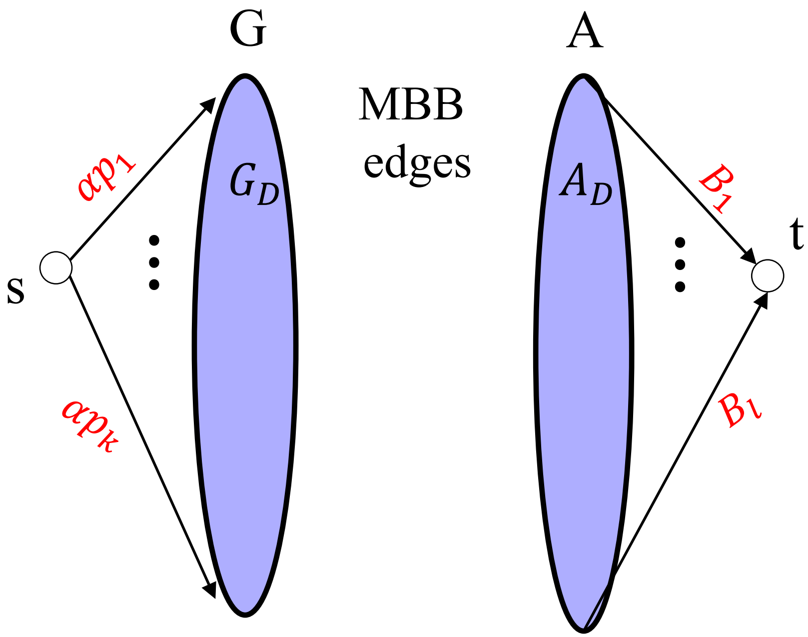
Event 1: Set $S^* \subseteq G_D$ becomes tight.

- $\alpha^* = \frac{\sum_{i \in N(S^*)} B_i}{\sum_{j \in S^*} p_j}$

  $= \min_{S \subseteq G_D} \boxed{\frac{\sum_{i \in N(S)} B_i}{\sum_{j \in S} p_j}}$ $\alpha(S)$

- Find $S^* = \underset{S \subseteq G_D}{\operatorname{argmin}}\, \alpha(S)$

Efficiently Computing Event 1



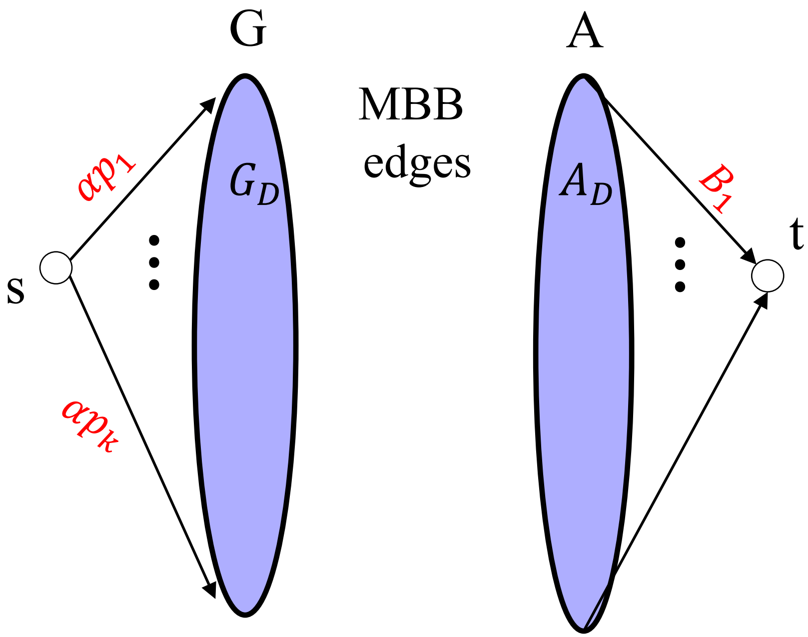
Event 1: Set $S^* \subseteq G_D$ becomes tight.

- $\alpha^* = \dfrac{\sum_{i \in N(S^*)} B_i}{\sum_{j \in S^*} p_j}$

  $= \min_{S \subseteq G_D} \boxed{\dfrac{\sum_{i \in N(S)} B_i}{\sum_{j \in S} p_j}}$ $\alpha(S)$

- Find $S^* = \underset{S \subseteq G_D}{\operatorname{argmin}}\ \alpha(S)$

Efficiently Computing Event 1



Event 1: Set $S^* \subseteq G_D$ becomes tight.

- $\alpha(S) = \frac{\sum_{i \in N(S)} B_i}{\sum_{j \in S} p_j}$

Find $S^* = \underset{S \subseteq G_D}{\operatorname{argmin}}\, \alpha(S)$

Claim. Can be done in O(n) min-cut computations

Efficient Flow-based Algorithms

- Polynomial running-time
  - Compute *balanced-flow:* minimizing l_2 norm of agents' surplus [DPSV'08]
- Strongly polynomial: Flow + scaling [Orlin'10]

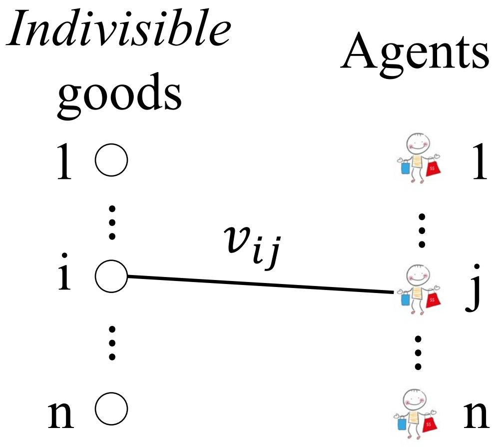
Exchange model (barter):

- Polynomial time [DM'16, DGM'17, CM'18]
- Strongly polynomial for exchange
  - Flow + scaling + approximate LP [GV'19]

Hylland-Zeckhauser

(an extension)

Motivation: Matching



Goal: Design a method to match goods to agents so that

- The outcome is **Pareto-optimal** and **envy-free**
- **Strategy-proof**: Agents have no incentive to lie about their $v_{ij}s$.

Hylland-Zeckhauzer'79: Compute CEEI where every agent wants total amount of at most one unit.

But the outcome is a fractional allocation!

Think of it as probabilities/time-shares/… []

HZ Equilibrium

Given:

- Agents $A = \{1, \dots, n\}$, indivisible goods $G = \{1, \dots, n\}$
- v_{ij}: value of agent i for good j.
  - If i gets j w/ prob. x_{ij}, then the expected value is: $\sum_{j \in G} v_{ij} x_{ij}$

Want: prices $p = (p_1, \dots, p_n)$, allocation $X = (x_1, \dots, x_n)$

- Each good j is allocated: $\sum_{i \in A} x_{ij} = 1$
- Each agent i gets an optimal bundle subject to
  - $1 budget, and **unit allocation**.

$$x_i \in \underset{x \in R_+^m}{\operatorname{argmax}} \left\{ \sum_j v_{ij} x_j \,\middle|\, \boldsymbol{\sum_j x_j = 1}, \sum_j p_j x_j \leq 1 \right\}$$

HZ Equilibrium

Hyllander-Zeckhauzer'79

- Exists. Pareto optimal, Strategy proof in large markets.

Vazirani-Yannakakis'20

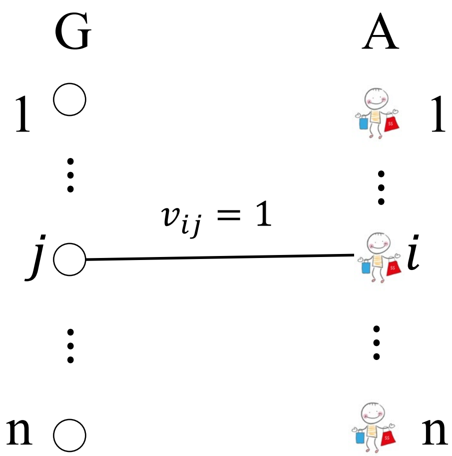
- Irrational equilibrium prices ⇒ not in PPAD
- In FIXP
- Algorithm for bi-valued preferences:

$$v_{ij} \in \{a_i, b_i\} \text{ where } a_i, b_i \geq 0$$

VY'20 Algorithm

$(v_{ij} \in \{0,1\})$

Want: (p, X)
All goods are sold.
Each agent i gets
$$x_i \in \underset{x: \sum_j x_j = 1, \sum_j p_j x_j \le 1}{\operatorname{argmax}} \sum_{j \in G} v_{ij} x_j$$



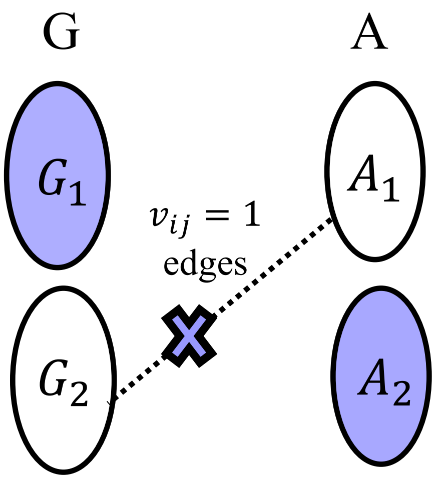
At equilibrium, an agent's utility is at most 1.

Perfect matching ⇒ An equilibrium is,

- Allocation on the matching edges
- Zero prices

VY'20 Algorithm

$(v_{ij} \in \{0,1\})$



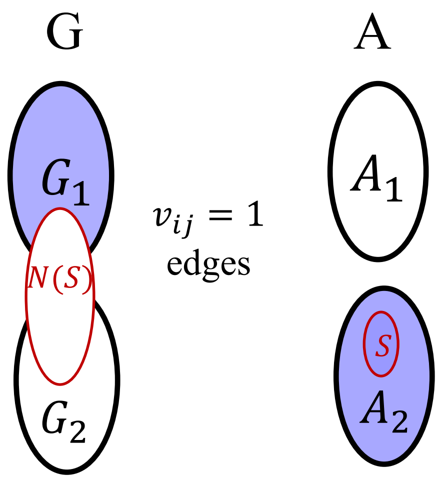
Want: (p, X)
Each good j is sold (1 unit)
Each agent i gets
$$x_i \in \underset{x:\sum_j x_j=1, \sum_j p_j x_j \le 1}{\operatorname{argmax}} \sum_{j\in G} v_{ij} x_j$$

No perfect matching

- Min vertex cover: $(G_1 \cup A_2)$
  - No $A_1 - G_2$ edge

VY'20 Algorithm

$(v_{ij} \in \{0,1\})$



Want: (p, X)
Each good j is sold (1 unit)
Each agent i gets
$x_i \in \underset{x: \sum_j x_j = 1, \sum_j p_j x_j \le 1}{\text{argmax}} \sum_{j \in G} v_{ij} x_j$

No perfect matching

- Min vertex cover: $(G_1 \cup A_2)$
  - No $A_1 - G_2$ edge
  - For each $S \subseteq A_2$, $|N(S) \cap G_2| \geq |S|$
    - Else get smaller VC by replacing S with $N(S) \cap G_2$

⇓

Subgraph (G_2, A_2) satisfies hall's condition for A_2.

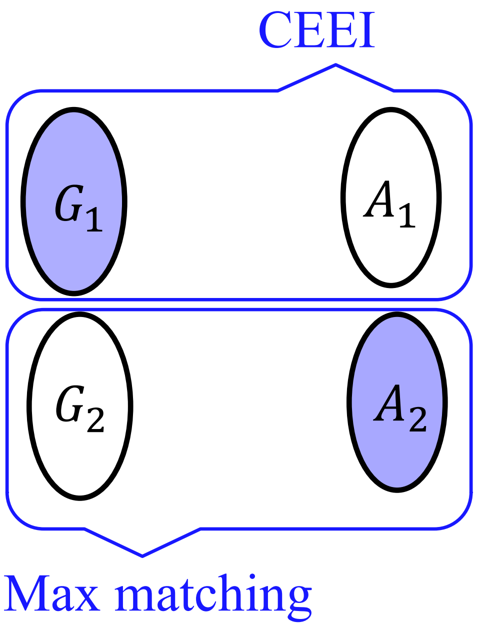
⇐

Max matching in (G_2, A_2) matches all of A_2.

VY'20 Algorithm

$(v_{ij} \in \{0,1\})$

Want: (p, X)
Each good j is sold (1 unit)
Each agent i gets
$$x_i \in \underset{x:\sum_j x_j = 1, \sum_j p_j x_j \le 1}{\text{argmax}} \sum_{j \in G} v_{ij} x_j$$



No perfect matching

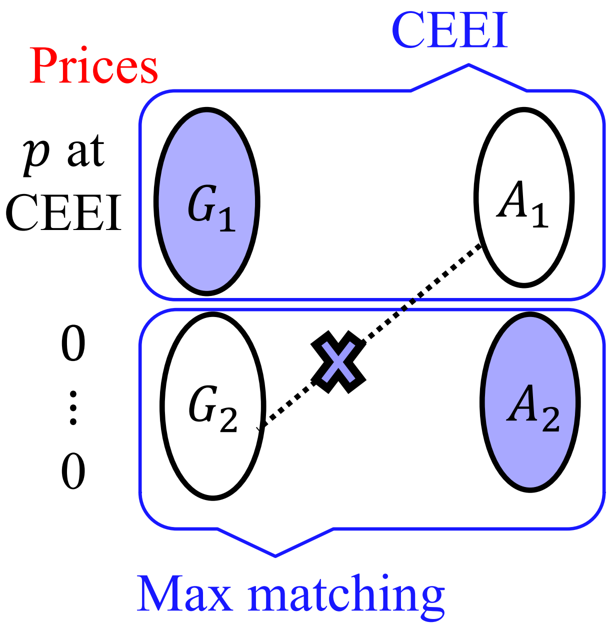
- Min vertex cover: $(G_1 \cup A_2)$
  - No $A_1 - G_2$ edge
  - For each $S \subseteq A_2$, $|N(S) \cap G_2| \ge |S|$
    - Max matching in (G_2, A_2) matches all of A_2.

VY'20 Algorithm

$(v_{ij} \in \{0,1\})$

Want: (p, X)
Each good j is sold (1 unit)
Each agent i gets

$$x_i \in \underset{x:\sum_j x_j = 1, \sum_j p_j x_j \le 1}{\text{argmax}} \sum_{j \in G} v_{ij} x_j$$



No perfect matching

- Min vertex cover: $(G_1 \cup A_2)$
- Eq. Prices: CEEI prices for G_1, and 0 prices for G_2
- Eq. Allocation
  - $i \in A_2$ gets her matched good
  - $i \in A_1$ gets CEEI allocation + unmatched goods from G_2

Running-time:
Strongly polynomial

VY’20 Algorithm

bi-values: $v_{ij} \in \{a_i, b_i\}, 0 \leq a_i < b_i$

Reduces to $v_{ij} \in \{0,1\}$

Exercise.

Open Questions

HZ Equilibrium

Computation for the general case.

Is it hard? OR is it (approximation) polynomial-time?

- Efficient algorithm when #goods or #agents is a constant [DK'08, AKT'17]
  - Cell-decomposition and enumeration

What about chores?

- CEEI exists but may form a non-convex set [BMSY'17]
- Efficient Computation?
  - **Open: Fisher as well as for CEEI**
  - For constantly many agents (or chores) [BS'19, GM'20]
  - *Fast* path-following algorithm [CGMM.'20]
- Hardness result for an exchange model [CGMM.'20]

References.

[AKT17] Alaei, Saeed, Pooya Jalaly Khalilabadi, and Eva Tardos. "Computing equilibrium in matching markets." *Proceedings of the 2017 ACM Conference on Economics and Computation*. 2017.

[BMSY17] Anna Bogomolnaia, Herv´e Moulin, Fedor Sandomirskiy, and Elena Yanovskaia. Competitive division of a mixed manna. Econometrica, 85(6):1847–1871, 2017.

[BMSY19] Anna Bogomolnaia, Herv´e Moulin, Fedor Sandomirskiy, and Elena Yanovskaia. Dividing bads under additive utilities. Social Choice and Welfare, 52(3):395–417, 2019.

[BS19] Brânzei, Simina, and Fedor Sandomirskiy. "Algorithms for Competitive Division of Chores." *arXiv preprint arXiv:1907.01766* (2019).

[GM20] Garg, Jugal, and Peter McGlaughlin. "Computing Competitive Equilibria with Mixed Manna." *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. 2020.

[CGMM20] Chaudhury, B. R., Garg, J., McGlaughlin, P., & Mehta, R. (2020). Competitive Allocation of a Mixed Manna. *arXiv preprint arXiv:2008.02753*.

[CGMM20] Chaudhury, B. R., Garg, J., McGlaughlin, P., & Mehta, R. (2020). Dividing Bads is Harder than Dividing Goods: On the Complexity of Fair and Efficient Division of Chores. *arXiv preprint arXiv:2008.00285*.

[DK08] Devanur, Nikhil R., and Ravi Kannan. "Market equilibria in polynomial time for fixed number of goods or agents." *2008 49th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2008.

[DPSV08] Devanur, Nikhil R., et al. "Market equilibrium via a primal--dual algorithm for a convex program." *Journal of the ACM (JACM)* 55.5 (2008): 1-18.

[HZ79] Aanund Hylland and Richard Zeckhauser. The efficient allocation of individuals to positions. Journal of Political economy, 87(2):293–314, 1979.

[VY20] Vazirani, Vijay V., and Mihalis Yannakakis. "Computational Complexity of the Hylland-Zeckhauser Scheme for One-Sided Matching Markets." *arXiv preprint arXiv:2004.01348* (2020).

THANK YOU