Lecture 3: Computation of CE

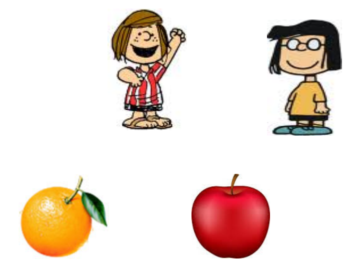
CS 580

Instructor: 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 dollars
  - valuation function $V_i: R_+^m \to R_+$

  Linear: for bundle $x_i = (x_{i1}, \dots, x_{im})$,

  $$V_i(x_i) = \sum_{j \in G} V_{ij} x_{ij}$$

- **Supply of every good is one.**

(Recall) Competitive Equilibrium

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

x_{ij}: Amount of good j agent i gets

- Optimal bundle: Agent i demands

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

$$\sum_j p_j \cdot x_j$$

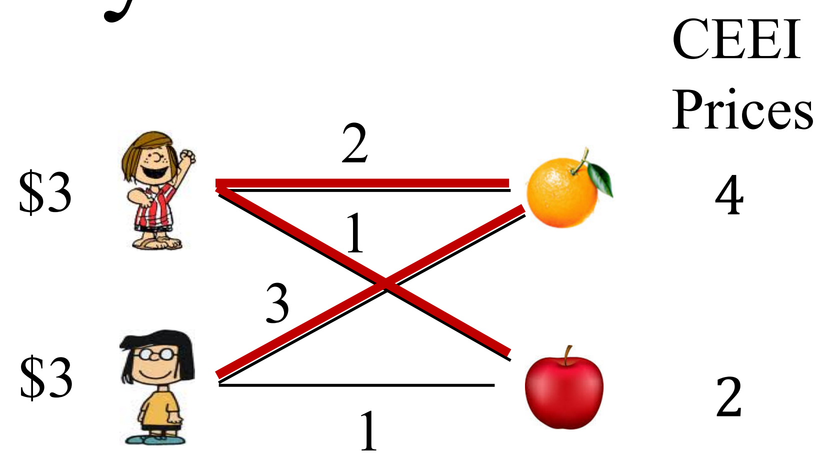
- Market clears: For each good j, demand = supply

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

CEEI Properties: Summary

CEEI ($B_i = 1, \forall i$) allocation is

- Pareto optimal (PO)
- Envy-free
- Proportional



CEEI Allocation:

$$X_1 = \left(\frac{1}{4}, 1\right), X_2 = \left(\frac{3}{4}, 0\right)$$

$$V_1(X_1) = \frac{3}{2},\ V_2(X_2) = \frac{9}{4}$$

$$V_1(X_2) = \frac{3}{2},\ V_2(X_1) = \frac{7}{4}$$

Next…

- Nash welfare maximizing

Max Nash Welfare

$$\max: \prod_{i \in A} V_i(X_{i1}, \dots, X_{im})$$

$$\text{s.t.} \quad \sum_{i \in A} X_{ij} \le 1, \;\; \forall j \in G$$

$$X_{ij} \ge 0, \qquad \forall i, \forall j$$

Feasible allocations

Max Nash Welfare (MNW)

$$\text{max: } \log\left(\prod_{i\in A} V_i(X_{i1}, \dots, X_{im})\right)$$

$$\boxed{\begin{array}{lll} \text{s.t.} & \sum_{i\in A} X_{ij} \le 1, & \forall j \in G \\ & X_{ij} \ge 0, & \forall i, \forall j \end{array}}$$

Feasible allocations

Max Nash Welfare (MNW)

$$\max: \sum_{i \in A} \log V_i(X_{i1}, \ldots, X_{im})$$

$$\text{s.t.} \quad \sum_{i \in A} X_{ij} \le 1, \;\; \forall j \in G$$
$$X_{ij} \ge 0, \qquad \forall i, \forall j$$

Feasible allocations

Eisenberg-Gale Convex Program '59

$$\max: \quad \sum_{i \in A} \log V_i(\bar{X}_i)$$

Dual var.

$$\text{s.t.} \quad \sum_{i \in A} X_{ij} \leq 1, \ \forall j \in G \longrightarrow p_j$$

$$X_{ij} \geq 0, \qquad \forall i, \forall j$$

Theorem. Solutions of EG convex program are exactly the CEEI (p, X).

Proof.

Consequences: CEEI

- **Exists**
- Forms a convex set
- Can be *computed* in polynomial time
- Maximizes Nash Welfare

Theorem. Solutions of EG convex program are exactly the CEEI (p, X).

Proof. $\Rightarrow$ (Using KKT)

Recall: CEEI Characterization

Pirces $p = (p_1, \dots, p_m)$ and allocation $X = (X_1, \dots, X_n)$

- Optimal bundle: For each buyer i
  - $p \cdot X_i = 1$
  - Spend only on the goods that give maximum value/dollar-spent

$$X_{ij} > 0 \Rightarrow \frac{V_{ij}}{p_j} = \max_{k \in M} \frac{V_{ik}}{p_k}, \text{ for all good } j$$

- Market clears: For each good j,

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

Theorem. Solutions of EG convex program are exactly the CEE.

$$\max: \sum_{i\in A} \log(V_i(\bar{X}_i)) \qquad \sum_j V_{ij}X_{ij}$$

Dual var.

$$\text{s.t.} \quad \sum_{i\in A} X_{ij} \le 1,\ \forall j \in G \longrightarrow p_j \ge 0$$

$$X_{ij} \ge 0, \qquad \forall i, \forall j$$

Proof. ⇒ (Using KKT)

$$\forall j,\ p_j > 0 \Rightarrow \sum_i X_{ij} = 1$$

Dual condition to X_{ij}:

$$\frac{V_{ij}}{V_i(X_i)} \le p_j \Rightarrow \frac{V_{ij}}{p_j} \le V_i(X_i) \Rightarrow \max_j \frac{V_{ij}}{p_j} \le V_i(X_i)$$

$$\Rightarrow p_j > 0 \Rightarrow$$ market clears

buy only MBB goods

$$\boxed{X_{ij} > 0 \Rightarrow \frac{V_{ij}}{p_j} = V_i(X_i)}$$

$$\sum_j V_{ij}X_{ij} = \left(\sum_j p_j X_{ij}\right) V_i(X_i)$$

$$\Rightarrow \sum_j p_j X_{ij} = 1$$

⇒ optimal bundle

Efficient (Combinatorial) Algorithms

Polynomial time

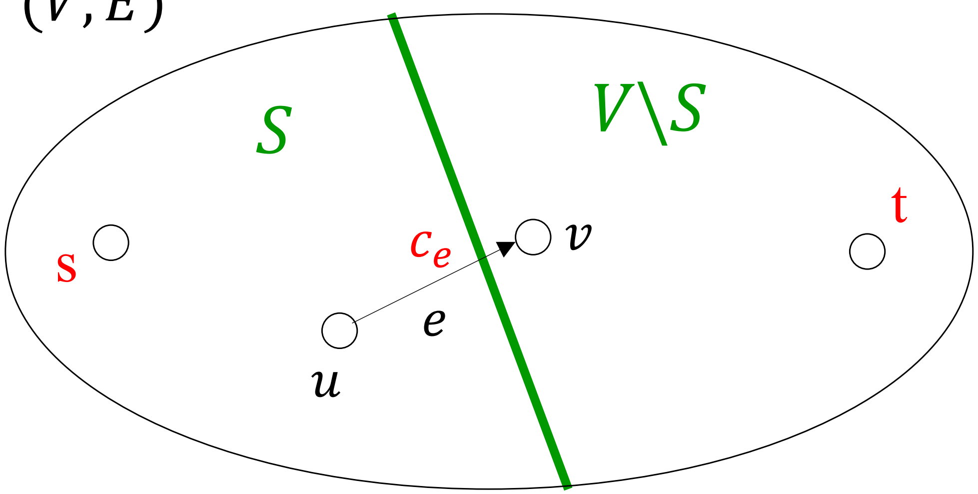
- Flow based [DPSV'08]
  - General exchange model (barter system) [DM'16, DGM'17, CM'18]
- Scaling + Simplex-like path following [GM.SV'13]

Strongly polynomial time

- Scaling + flow [O'10, V'12]
  - Exchange model (barter system) [GV'19]

Max Flow (One slide overview)

Directed Graph
(V, E)



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

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

Find maximum flow from s to t: $(\boldsymbol{f}_{\boldsymbol{e}})_{e\in E}$ 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

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 $x_i \in \operatorname{argmax}_{x:\, p \cdot x \le B_i} V_i(x)$
  - $p \cdot x_i = B_i$
  - $x_{ij} > 0 \Rightarrow \frac{V_{ij}}{p_j} = \max_{k \in G} \frac{V_{ik}}{p_k}$, for all good j

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

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

Competitive Equilibrium → Flow

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

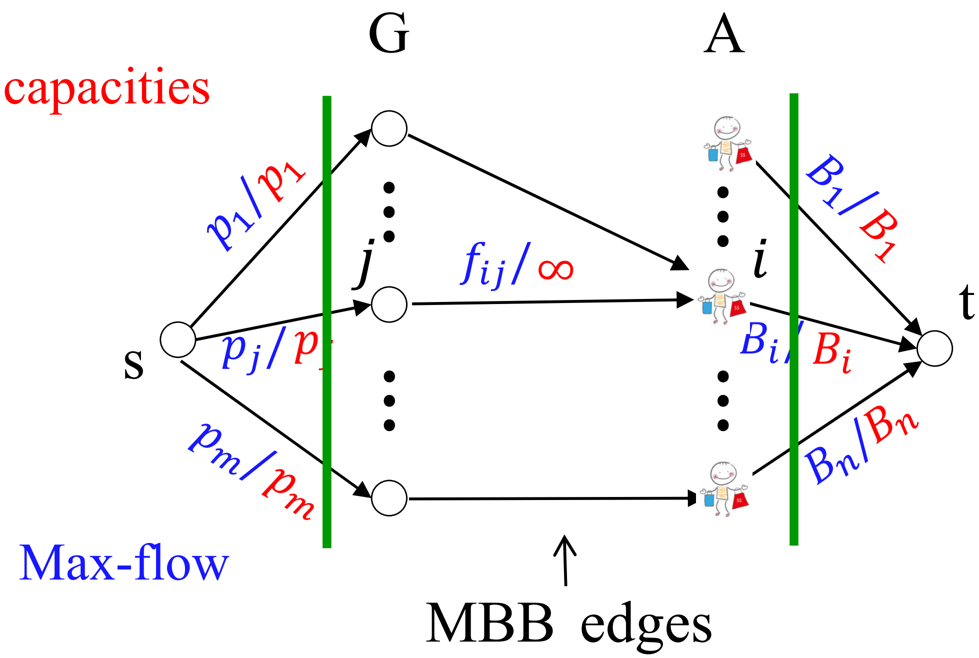
$$f_{ij} = x_{ij} p_j \text{ (money spent by agent i on good j)}$$

- **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.

Opt. Bundle $\begin{cases} \sum_{j \in M} f_{ij} = B_i \\ f_{ij} > 0 \text{ on MBB edges} \end{cases}$

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

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)

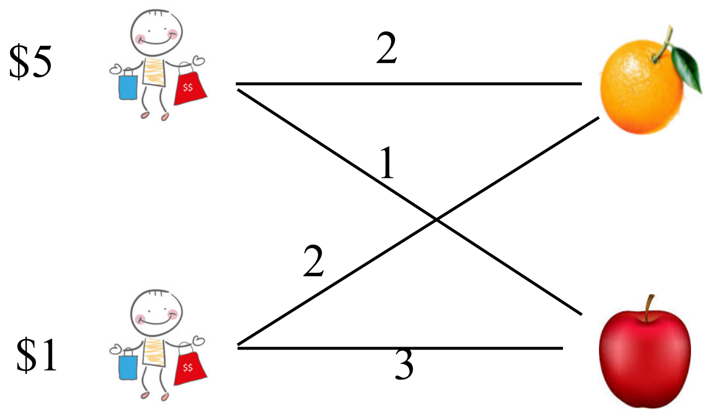
Demand ≥ Supply

Example

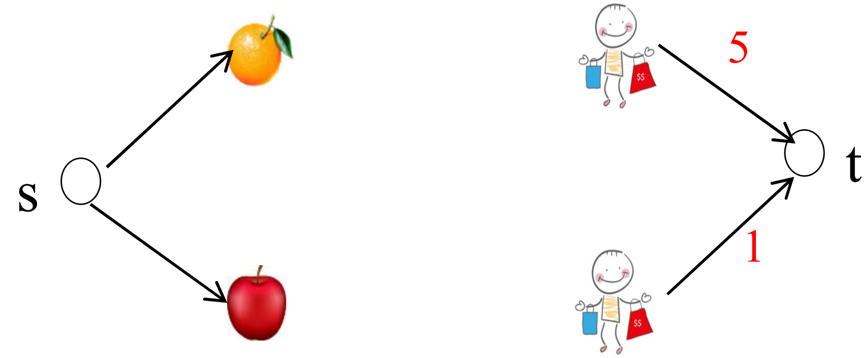
Invariants

1. Flow only on MBB edges
2. Min-cut = {s}
   (Demand ≥ Supply)

Input



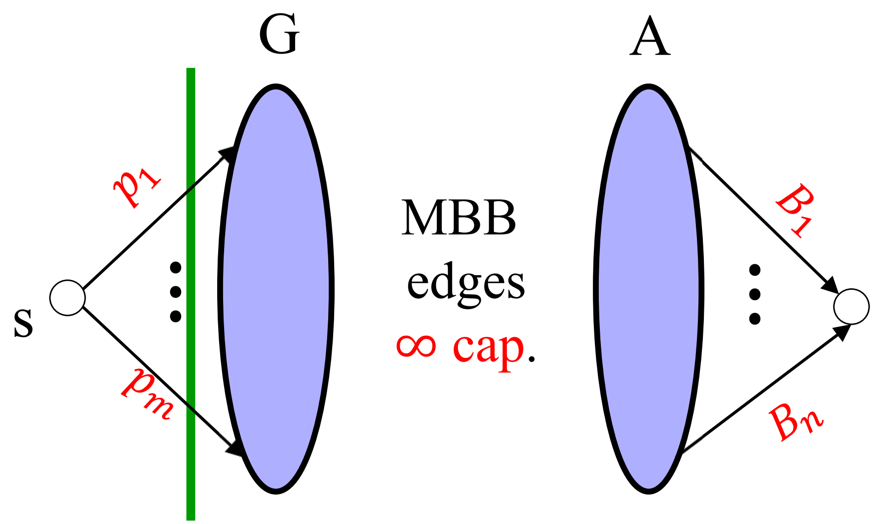
Init.



Algorithm (Pictorial)

Invariants

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

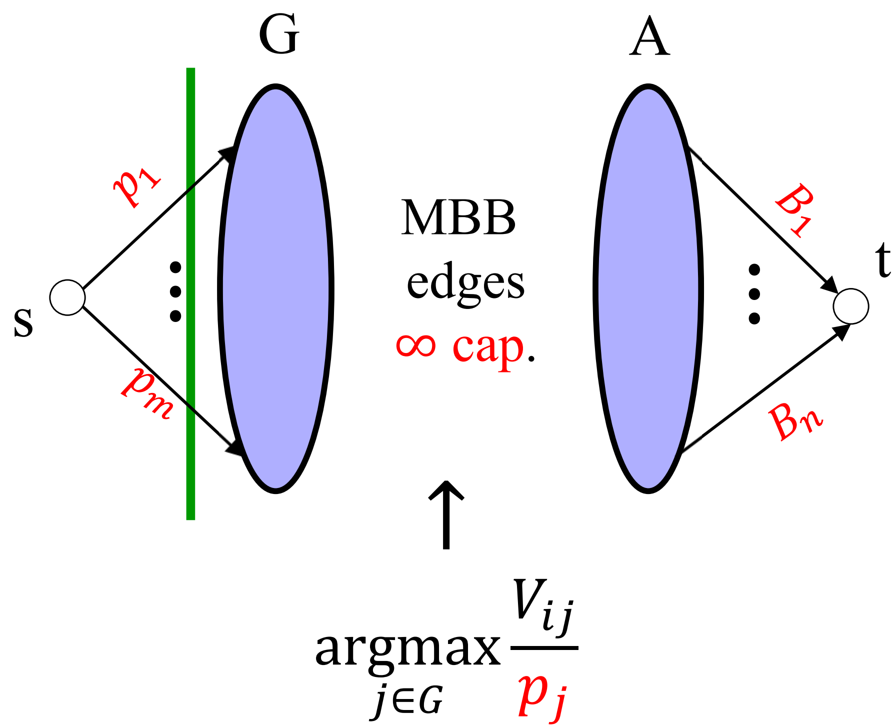


Init: $\forall j \in \mathrm{G},\ p_j < \min_i \frac{B_i}{m}$, and at least one MBB edge to j

Algorithm (Pictorial)

Invariants

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



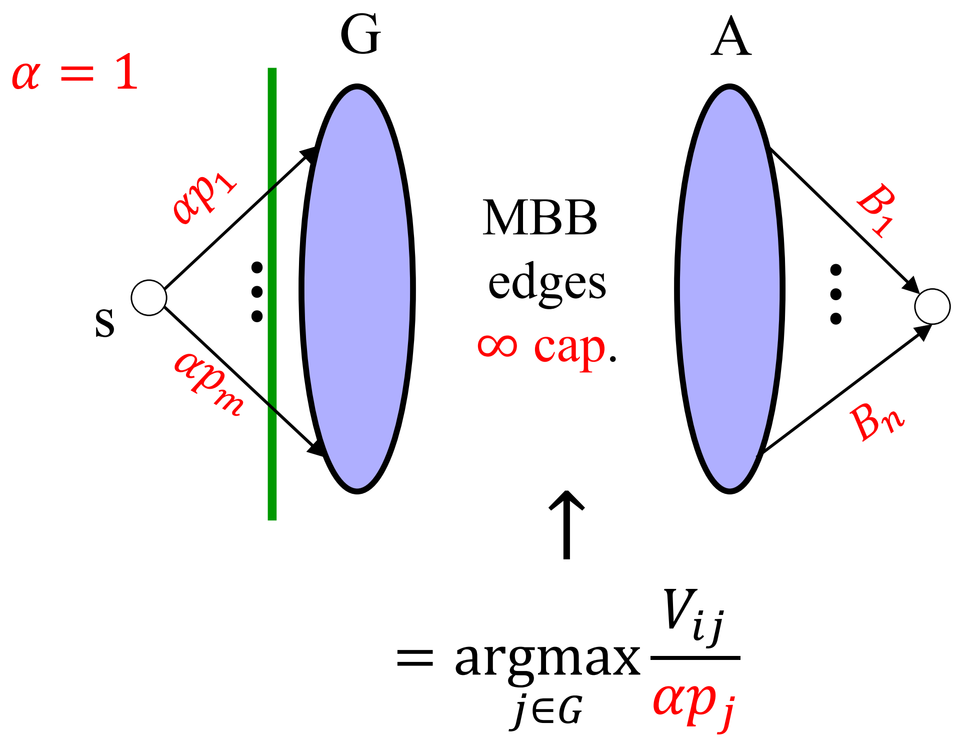
Init: $\forall j \in G,\ p_j < \min_i \frac{B_i}{m}$, and at least one MBB edge to j

Increase p:

Algorithm (Pictorial)

Invariants

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



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

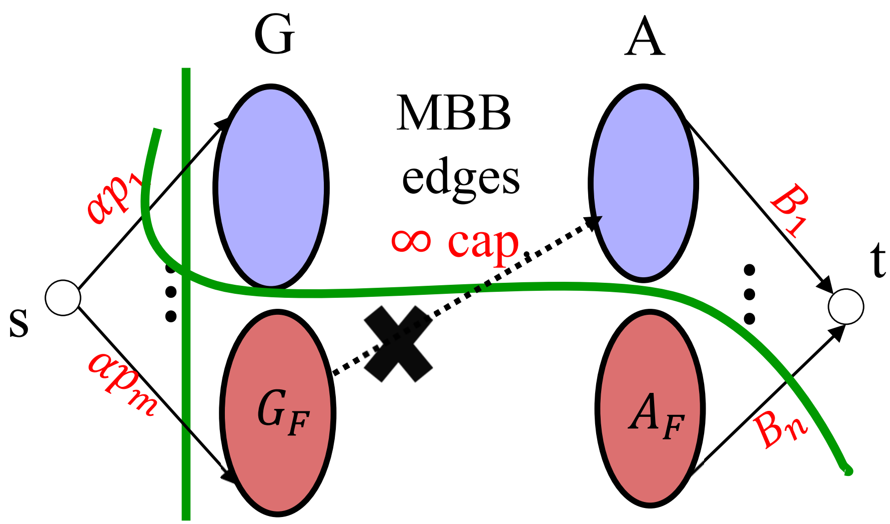
And at least one MBB edge to j

Increase 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: **Supply = Demand for G_F!** So, if prices of G_F are increased, then these will be under-demanded (supply > demand for G_F). And $\{s\}$ will cease to be a min-cut.

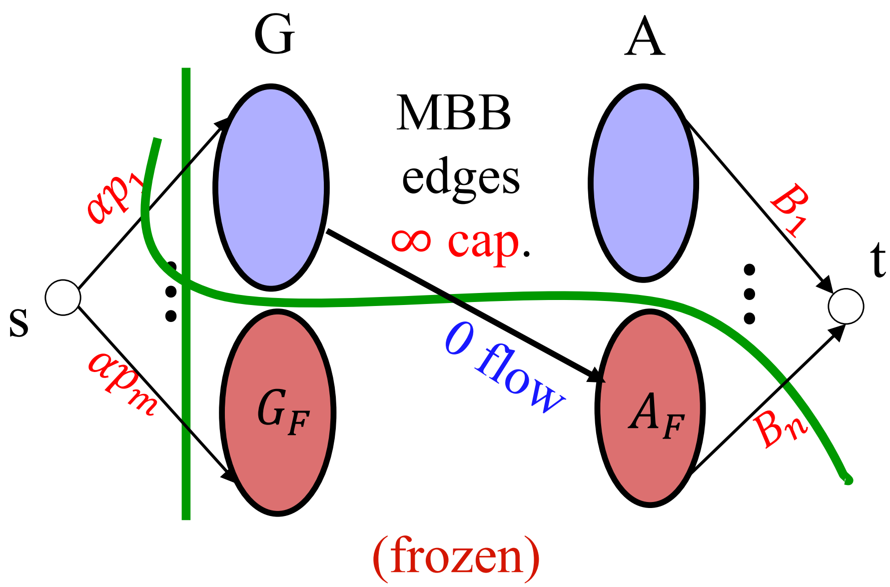
A tight-set.

Should freeze prices in G_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: $\uparrow \alpha$

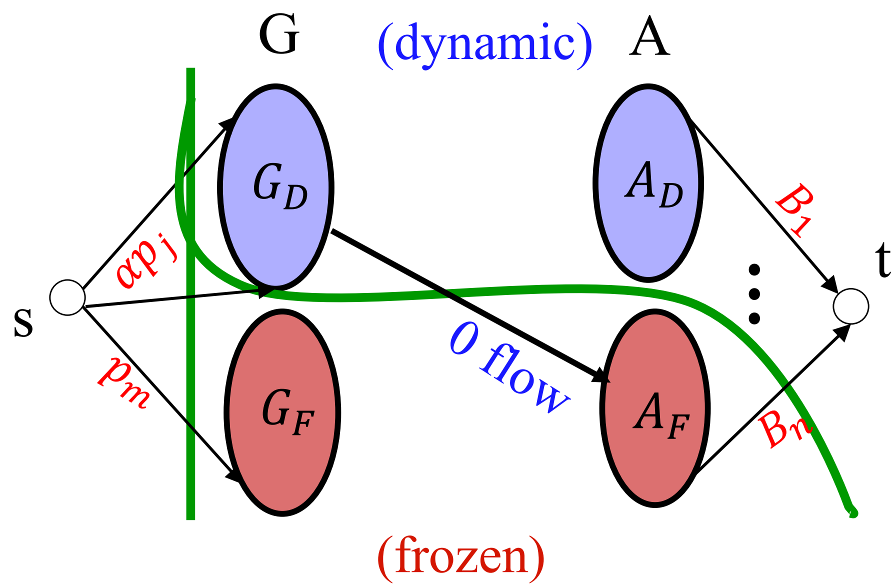
Event 1: A tight subset G_F

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

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).

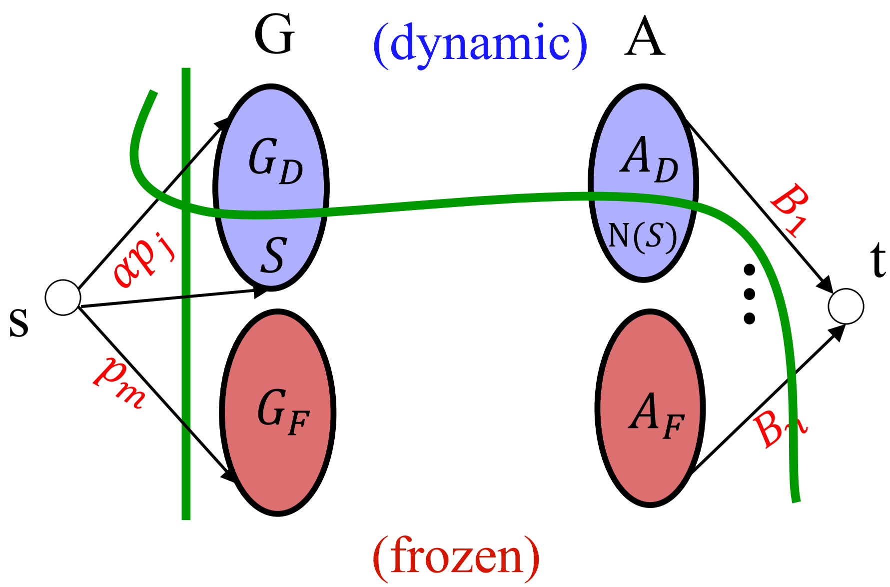
Freeze prices in G_F.

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 *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.

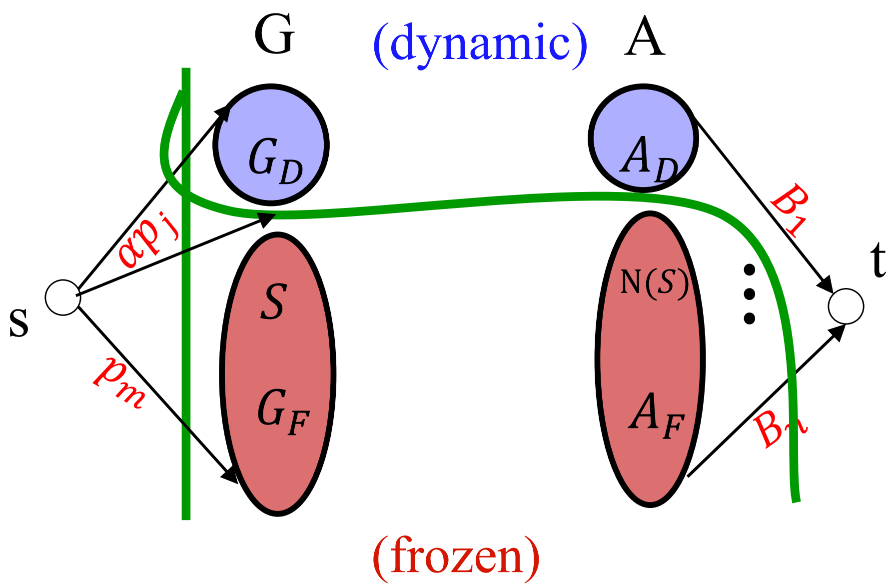
Observation: Again, supply=demand for goods in S. If prices of S is increased further, then **S** can not be fully sold. And $\{s\}$ will cease to be a min-cut.

Hence it needs to be moved to the *frozen 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 $S \subseteq G_D$

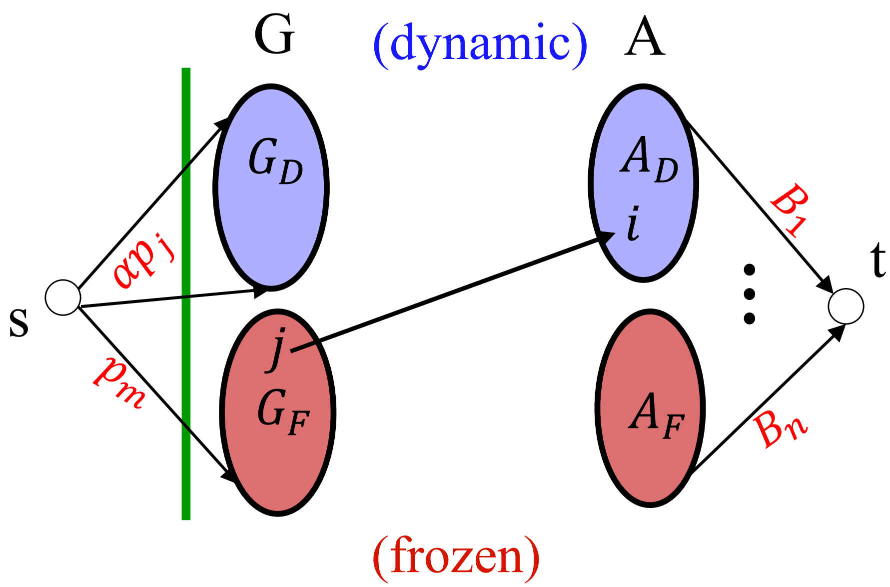
Move $(S, \mathrm{N}(S))$ to frozen part

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

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

Move $(S, \mathrm{N}(S))$ from dynamic to frozen

Freeze prices in G_F, and increase in G_D.

OR

Event 2: New MBB edge

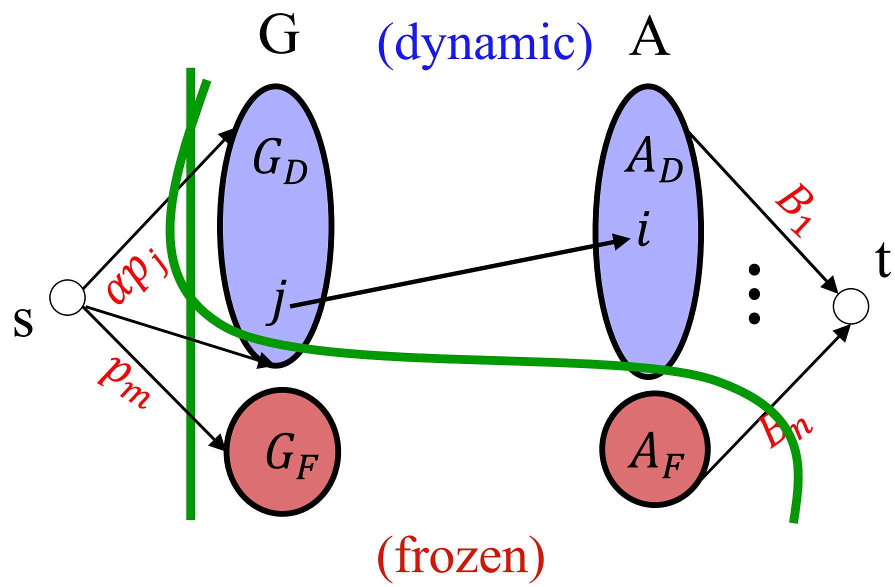
Must be between $i \in A_D$ & $j \in G_F$.

Recompute dynamic 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 $\boldsymbol{p}$: $\uparrow \alpha$

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

Move $(S, \mathrm{N}(S))$ from dynamic 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 $j \in G_F$.

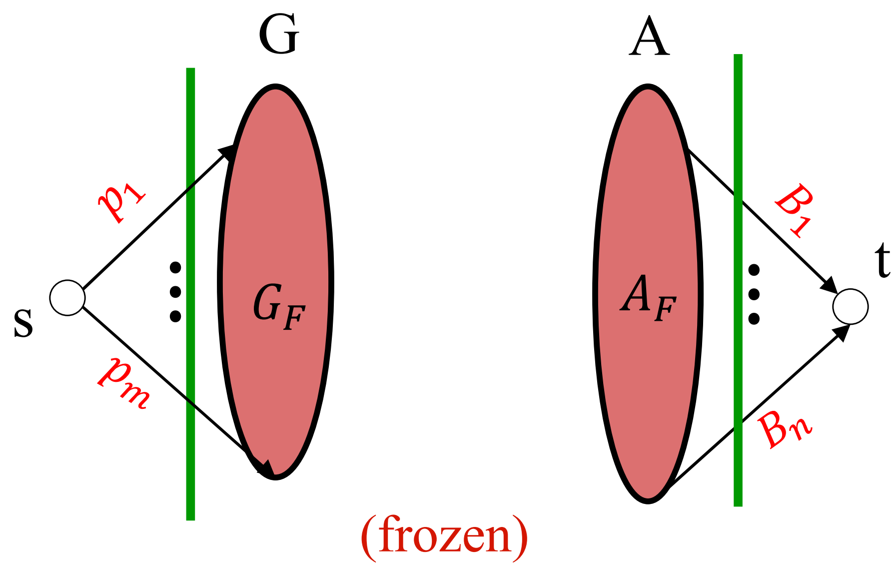
Recompute dynamic and frozen:

Move the component containing good j from frozen to dynamic.

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 $S \subseteq G_D$

Move $(S, \mathrm{N}(S))$ from dynamic 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 dynamic and frozen.

Stop: all goods are frozen.

Observations: Prices only increase.

Each increase can be lower bounded.

Both the events can be computed efficiently.

⇓

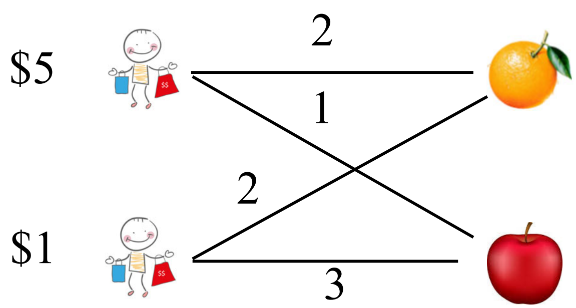
Converges to CE in finite time.

Example

Invariants

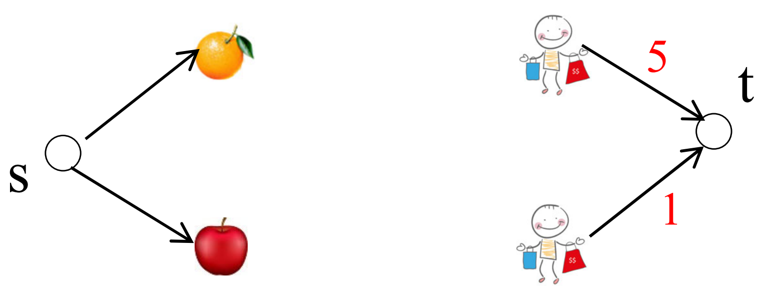
1. Flow only on MBB edges
2. Min-cut = $\{s\}$ (goods are sold)

Input

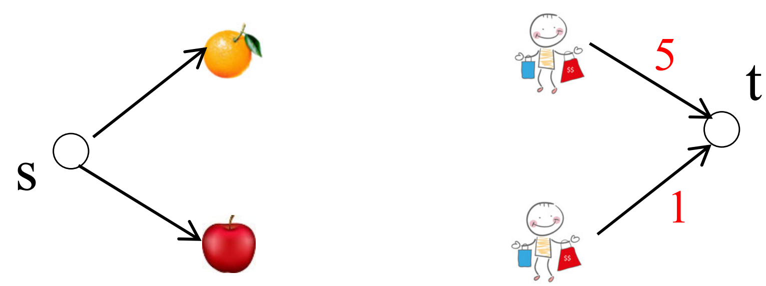


Init.

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.

    N$(S) \leftarrow$ agents w/ MBB edges to S (neighbors of S).

    Move $(\mathrm{S}, \mathrm{N(S)})$ 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.

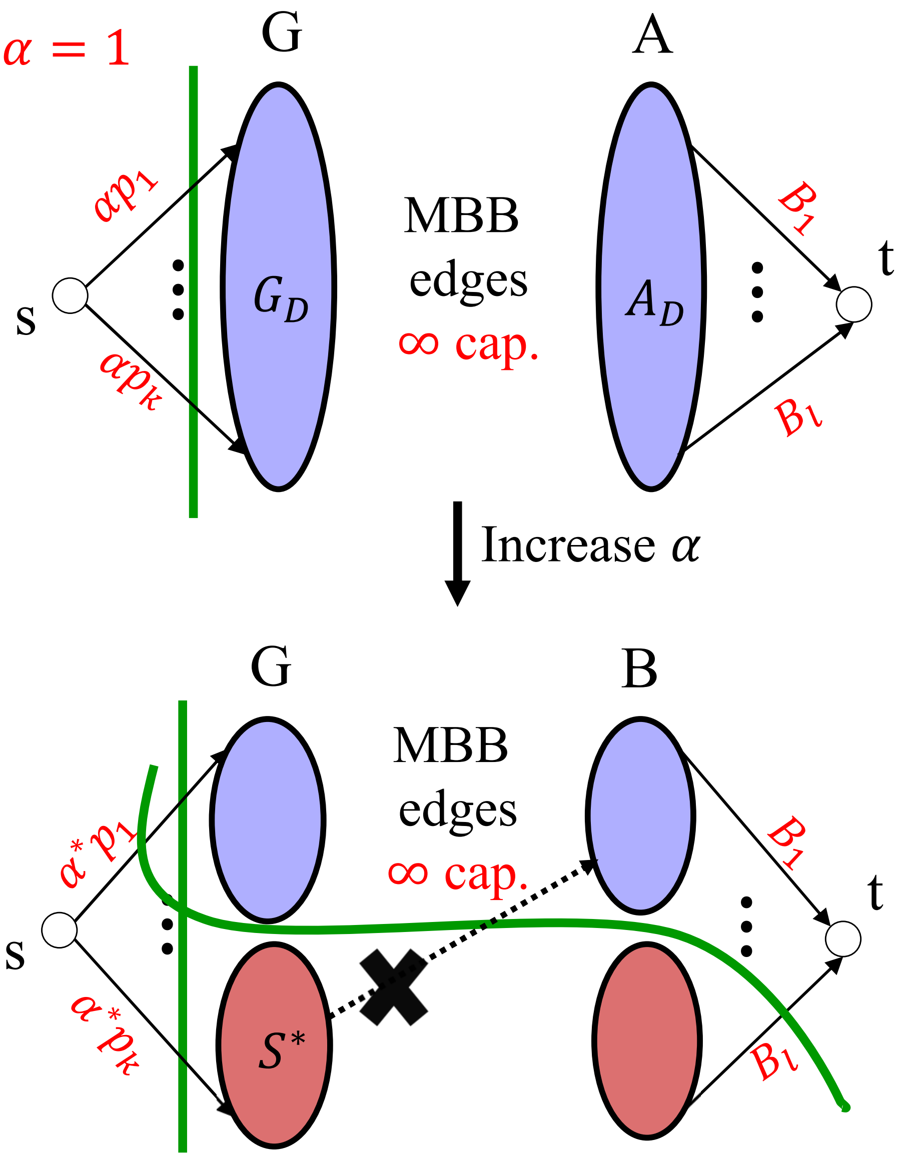
    Move component of j from (G_F, A_F) to (G_D, A_D).
- Output (p, F)

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

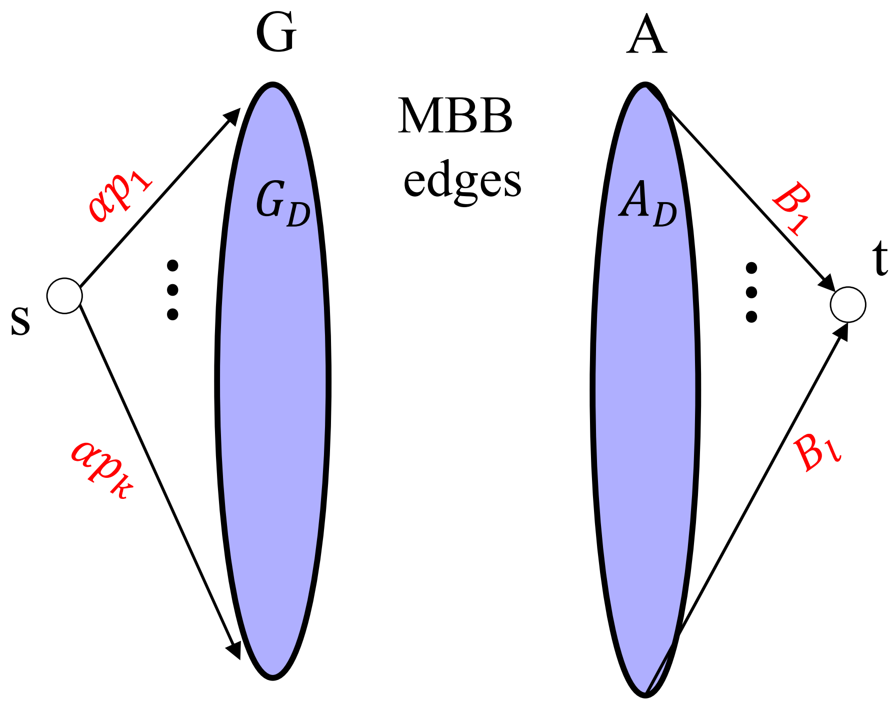


Event 1: Set $S^* \subseteq G_D$ becomes tight.

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

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

Efficiently Computing Event 1



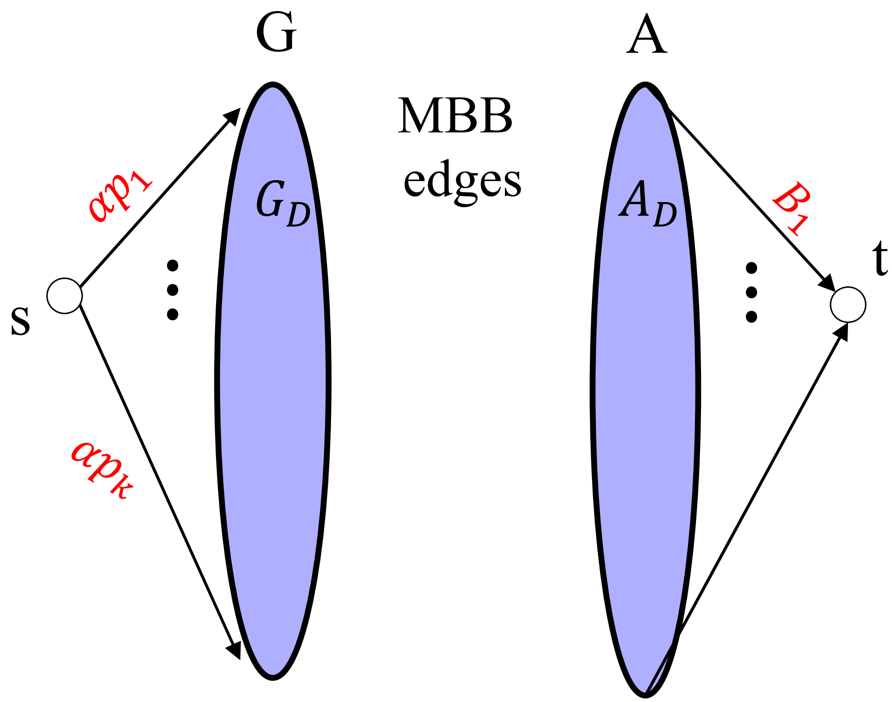
Event 1: Set $S^* \subseteq G_D$ becomes tight.

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

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

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

Efficiently Computing Event 1



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

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

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

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

$(G', A') \leftarrow (G_D, A_D)$
Repeat{
  $\alpha \leftarrow \alpha(G')$. Set $c_{(s,j)} \leftarrow \alpha p_j, \forall j \in G'$
  $(s \cup \{S\} \cup N(S)) \leftarrow$ min-cut in (G', A')
  $(G', A') \leftarrow (S, N(S))$
}Until($\{s\}$ not a min-cut)
Return α

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]

Application to Display Ads: Pacing Eq.

- Google Display Ads
  - Each advertiser has
    - Budget B_i. Value v_{ij} for keyword j
  - Pacing Eq.: $(\lambda_1, \ldots, \lambda_n) \in [0,1]^n$ s.t.
    - First price auction with bids $\lambda_i v_{ij}$
    - For each agent i, if $\lambda_i < 1$ then total payment $= B_i$, else $\leq B_i$
- Equivalent to Fisher market with quasi-linear utilities!

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