Fair and Efficient Allocations of Chores under Bivalued Preferences*

Thorben Tröbst

Theory Seminar, May 6, 2022

Department of Computer Science, University of California, Irvine

* based on AAAI 2022 paper by Jugal Garg, Aniket Murhekar, and John Qin

Fair division is a classic problem in AGT:

- Given n agents N, and

Fair division is a classic problem in AGT:

- Given n agents N, and
- m goods M,

Fair Division

Fair division is a classic problem in AGT:

- Given n agents N, and
- m goods M,
- allocate goods to agents in an efficient and fair manner.

Fair Division

Fair division is a classic problem in AGT:

- Given n agents N, and
- m goods M,
- allocate goods to agents in an efficient and fair manner.

Note: Multiple goods can go to one agent!

Fair division is a classic problem in AGT:

- Given n agents N, and
- m goods M,
- allocate goods to agents in an efficient and fair manner.

Note: Multiple goods can go to one agent!

Assume: Linear utilities: u_{ij} for all $i \in N, j \in M$.

Fairness

What is fairness?

What is fairness?

- Proportional: $u_i(x_i) \geq \frac{1}{n} u_i(M)$?

Fairness

What is fairness?

- Proportional: $u_i(x_i) \geq \frac{1}{n} u_i(M)$?
- Egalitarian: $u_i(x_i) \geq u_j(x_j)$?

Fairness

What is fairness?

- Proportional: $u_i(x_i) \geq \frac{1}{n} u_i(M)$?
- Egalitarian: $u_i(x_i) \geq u_j(x_j)$?
- Envy-free: $u_i(x_i) \geq u_i(x_j)$?

Fairness

What is fairness?

- Proportional: $u_i(x_i) \geq \frac{1}{n} u_i(M)$?
- Egalitarian: $u_i(x_i) \geq u_j(x_j)$?
- Envy-free: $u_i(x_i) \geq u_i(x_j)$?

Problem: none of these work for indivisible goods!

To get notions of discrete fairness, simply remove one good:

Discrete Fairness

To get notions of discrete fairness, simply remove one good:

- Proportional up to one good: $u_i(x_i) \geq \min_k \frac{1}{n} u_i(M - k)$?

To get notions of discrete fairness, simply remove one good:

- Proportional up to one good: $u_i(x_i) \geq \min_k \frac{1}{n} u_i(M - k)$?
- Egalitarian up to one good (EQ1): $u_i(x_i) \geq \min_k u_j(x_j - k)$?

Discrete Fairness

To get notions of discrete fairness, simply remove one good:

- Proportional up to one good: $u_i(x_i) \geq \min_k \frac{1}{n} u_i(M - k)$?
- Egalitarian up to one good (EQ1): $u_i(x_i) \geq \min_k u_j(x_j - k)$?
- Envy-free up to one good (EF1): $u_i(x_i) \geq \min_k u_i(x_j - k)$?

To get notions of discrete fairness, simply remove one good:

- Proportional up to one good: $u_i(x_i) \geq \min_k \frac{1}{n} u_i(M - k)$?
- Egalitarian up to one good (EQ1): $u_i(x_i) \geq \min_k u_j(x_j - k)$?
- Envy-free up to one good (EF1): $u_i(x_i) \geq \min_k u_i(x_j - k)$?

Note: replacing min with max yields stronger EQX / EFX fairness.

Efficiency

Fairness alone is not that impressive:

Efficiency

Fairness alone is not that impressive:

	Phone	Tablet
Alice	10	1
Bob	1	10

Efficiency

Fairness alone is not that impressive:

	Phone	Tablet
Alice	10	1
Bob	1	10

Allocation Alice – Tablet and Bob – Phone is EFX and EQX but obviously bad!

Efficiency Notions

The classic efficiency notions are:

The classic efficiency notions are:

Definition

An allocation is Pareto-optimal (PO) if no allocation is weakly better for all agents, and strictly better for at least one agent.

The classic efficiency notions are:

Definition

An allocation is Pareto-optimal (PO) if no allocation is weakly better for all agents, and strictly better for at least one agent.

Definition

An allocation is fractionally Pareto-optimal (fPO) if no fractional allocation is weakly better for all agents, and strictly better for at least one agent.

Known Results

Theorem (Barman, Krishnamurthy, Vaish 2018)

EF1 + PO allocations always exist an can be computed in pseudo-polynomial time.

Known Results

Theorem (Barman, Krishnamurthy, Vaish 2018)

EF1 + PO allocations always exist an can be computed in pseudo-polynomial time.

Theorem (Garg, Murhekar 2021)

EF1 + fPO allocations always exist an can be computed in pseudo-polynomial time.

Known Results

Theorem (Barman, Krishnamurthy, Vaish 2018)

EF1 + PO allocations always exist an can be computed in pseudo-polynomial time.

Theorem (Garg, Murhekar 2021)

EF1 + fPO allocations always exist an can be computed in pseudo-polynomial time.

Theorem (Garg, Murhekar 2021)

EFX + fPO allocations exist under bivalued utilities and can be computed in polynomial time.

Known Results

Theorem (Barman, Krishnamurthy, Vaish 2018)

EF1 + PO allocations always exist an can be computed in pseudo-polynomial time.

Theorem (Garg, Murhekar 2021)

EF1 + fPO allocations always exist an can be computed in pseudo-polynomial time.

Theorem (Garg, Murhekar 2021)

EFX + fPO allocations exist under bivalued utilities and can be computed in polynomial time.

Open problem: Do EFX + PO allocations always exist?

Chores

Sometimes we wish to assign chores instead of goods.

Chores

Sometimes we wish to assign chores instead of goods.

	Dishes	Laundry
Alice	-5	-1
Bob	-1	-2

Chores

Sometimes we wish to assign chores instead of goods.

	Dishes	Laundry
Alice	-5	-1
Bob	-1	-2

Note: Notions of fairness and efficiency extend to chores!

Results

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

Results

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

Theorem

EF + PO allocations of divisible chores exist under bivalued utilities and can be computed in strongly polynomial time.

Results

We want to show:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

We want to show:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

So our instance looks like:

We want to show:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

So our instance looks like:

- n agents N,

We want to show:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

So our instance looks like:

- n agents N,
- m goods M,

We want to show:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

So our instance looks like:

- n agents N,
- m goods M,
- costs c_{ij} where wlog. $c_{ij} \in \{1, k\}$ for some $k \in \mathbb{N}$.

Given earning targets e_i for all $i \in N$, an allocation x and prices p_j for all $j \in G$ form a market equilibrium if

Relation to Market Equilibria

Given earning targets e_i for all $i \in N$, an allocation x and prices p_j for all $j \in G$ form a market equilibrium if

- the market clears, i.e. all chores are allocated,

Relation to Market Equilibria

Given earning targets e_i for all $i \in N$, an allocation x and prices p_j for all $j \in G$ form a market equilibrium if

- the market clears, i.e. all chores are allocated,
- every agent achieves their earning target, i.e. $p(x_i) \geq e_i$,

Relation to Market Equilibria

Given earning targets e_i for all $i \in N$, an allocation x and prices p_j for all $j \in G$ form a market equilibrium if

- the market clears, i.e. all chores are allocated,
- every agent achieves their earning target, i.e. $p(x_i) \geq e_i$,
- agents only receive minimum bang per buck, i.e. chores that minimize c_{ij}/p_j.

Relation to Market Equilibria

Given earning targets e_i for all $i \in N$, an allocation x and prices p_j for all $j \in G$ form a market equilibrium if

- the market clears, i.e. all chores are allocated,
- every agent achieves their earning target, i.e. $p(x_i) \geq e_i$,
- agents only receive minimum bang per buck, i.e. chores that minimize c_{ij}/p_j.

Note: If (x, p) is a market equilibrium, then x is fPO!

Market Equilibrium Example

	Dishes	Laundry	Cleaning
Alice	5	1	3
Bob	1	2	4
Charlie	3	1	5

Allocate: Alice – Cleaning, Bob – Dishes, Charlie – Laundry

$$p(\text{Laundry}) = 1, p(\text{Dishes}) = 1, p(\text{Cleaning}) = 3$$

Market Equilibrium Example

	Dishes	Laundry	Cleaning
Alice	5	1	3
Bob	1	2	4
Charlie	3	1	5

Allocate: Alice – Cleaning, Bob – Dishes, Charlie – Laundry

$$p(\text{Laundry}) = 1, p(\text{Dishes}) = 1, p(\text{Cleaning}) = 3$$

Note: If all e_i are the same, then this is very fair. This is possible for divisible goods but no algorithm exists.

Market equilibria have a fairness notion too:

Definition

(x, p) is price envy-free up to one chore (pEF1) if
$\min_k p(x_i - k) \leq p(x_j)$ for all i, j.

Market equilibria have a fairness notion too:

Definition

(x, p) is price envy-free up to one chore (pEF1) if $\min_k p(x_i - k) \leq p(x_j)$ for all i, j.

Lemma

If (x, p) is pEF1 then x is EF1.

General Strategy

We can now give the general strategy:

We can now give the general strategy:

- Compute some initial market equilibrium (x, p)

General Strategy

We can now give the general strategy:

- Compute some initial market equilibrium (x, p)
- If (x, p) is not pEF1, identify the big earner $b = \arg\max_i \min_j p(x_i - k)$ and the least earner $l = \arg\min_i p(x_i)$.

General Strategy

We can now give the general strategy:

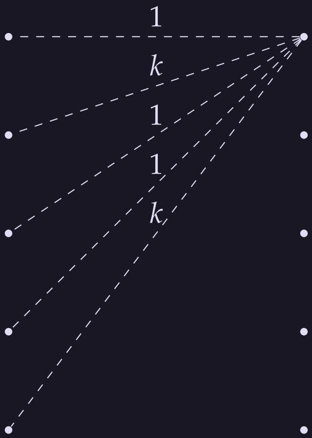
- Compute some initial market equilibrium (x, p)
- If (x, p) is not pEF1, identify the big earner $b = \arg\max_i \min_j p(x_i - k)$ and the least earner $l = \arg\min_i p(x_i)$.
- Try to funnel chores from the big earner to the least earner.

General Strategy

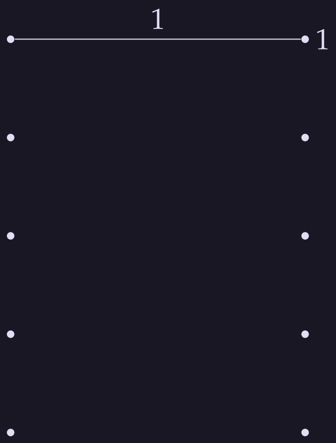
We can now give the general strategy:

- Compute some initial market equilibrium (x, p)
- If (x, p) is not pEF1, identify the big earner $b = \arg\max_i \min_j p(x_i - k)$ and the least earner $l = \arg\min_i p(x_i)$.
- Try to funnel chores from the big earner to the least earner.
- If this is not possible, raise prices.

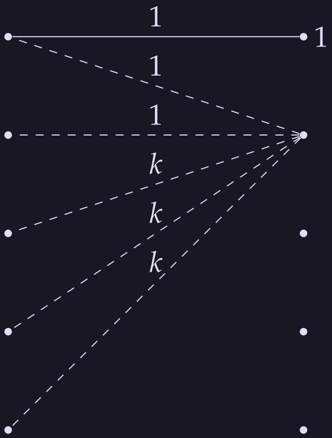
Phase 1: Initial Equilibrium



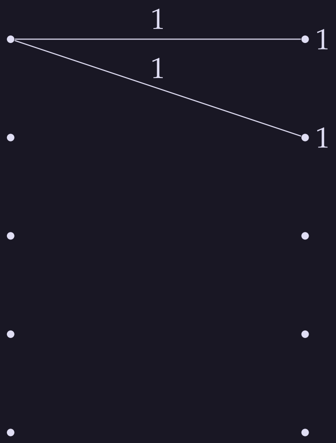
Phase 1: Initial Equilibrium



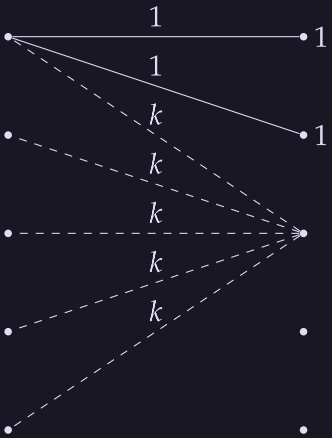
Phase 1: Initial Equilibrium



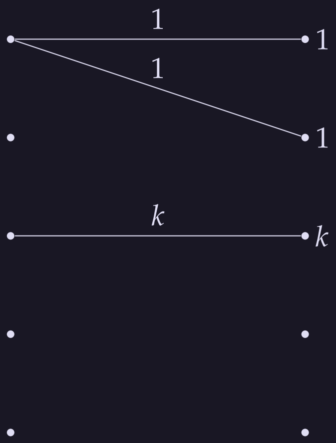
Phase 1: Initial Equilibrium



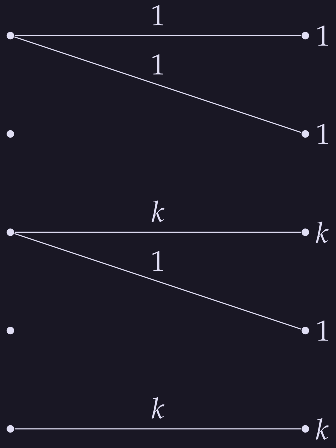
Phase 1: Initial Equilibrium



Phase 1: Initial Equilibrium

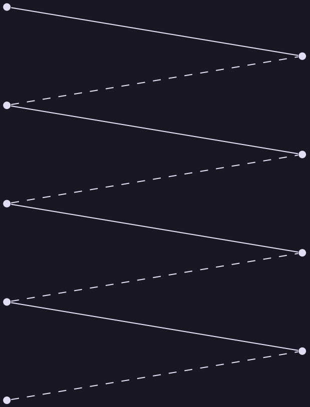


Phase 1: Initial Equilibrium

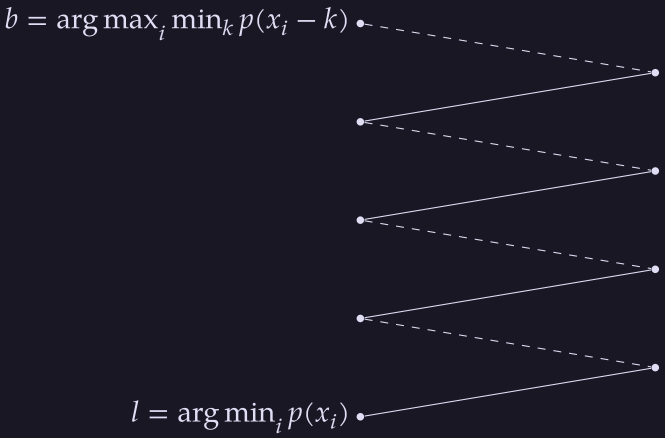


Phase 1: Initial Equilibrium II

$b = \arg\max_i \min_k p(x_i - k)$



$l = \arg\min_i p(x_i)$


$b = \arg\max_i \min_k p(x_i - k)$
$l = \arg\min_i p(x_i)$

Run the following algorithm:

Run the following algorithm:

1. Let b be the biggest earner.

Phase 1: Initial Equilibrium III

Run the following algorithm:

1. Let b be the biggest earner.
2. Is there some l in the component of b with $\min_k p(x_b - k) > p(x_l)$?

Run the following algorithm:

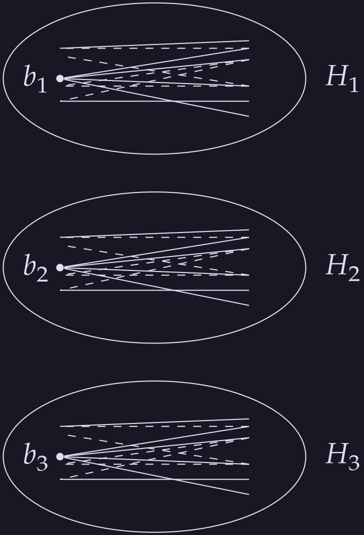
1. Let b be the biggest earner.
2. Is there some l in the component of b with $\min_k p(x_b - k) > p(x_l)$?
3. If yes, funnel chores from b to l and go back to 1.

Phase 1: Initial Equilibrium III

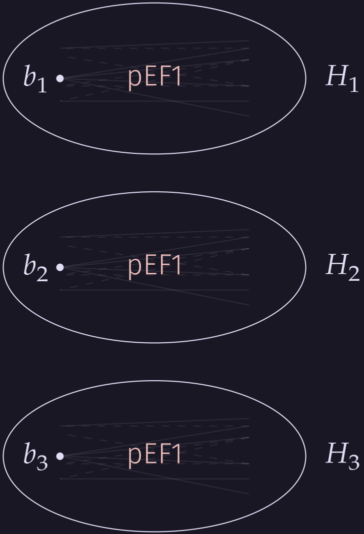
Run the following algorithm:

1. Let b be the biggest earner.
2. Is there some l in the component of b with $\min_k p(x_b - k) > p(x_l)$?
3. If yes, funnel chores from b to l and go back to 1.
4. If no, remove the component of b from the graph and go back to 1.

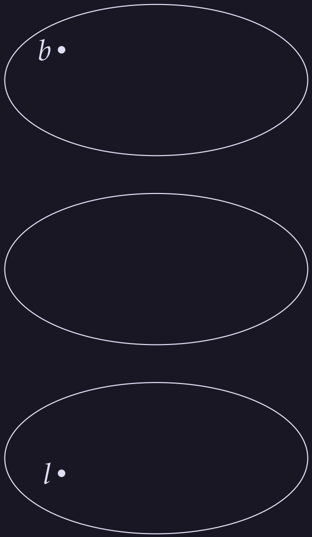
Phase 1: Initial Equilibrium IV



Phase 1: Initial Equilibrium IV

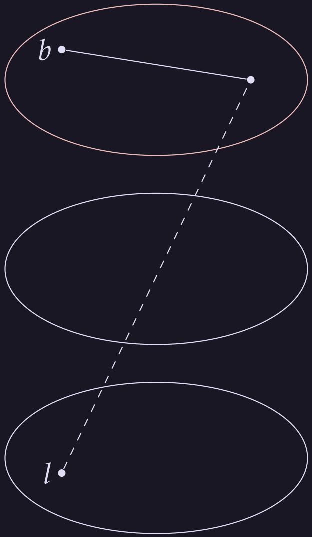


Phase 2: Raise Prices



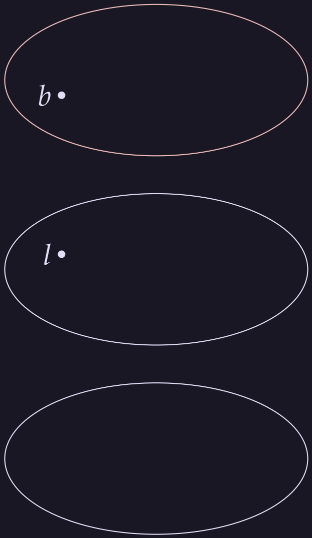
Phase 2: Raise Prices

Phase 2: Raise Prices

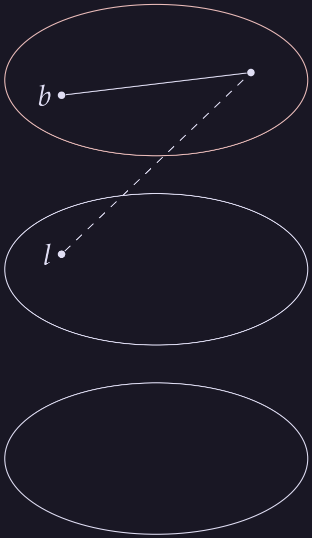


Phase 2: Raise Prices

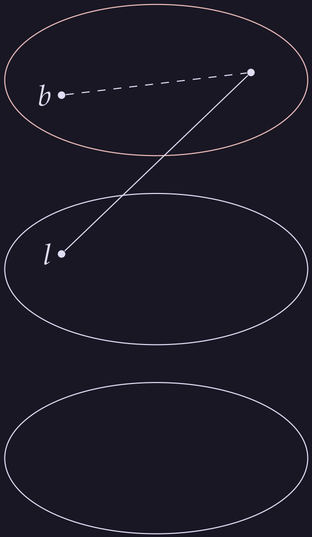
Phase 2: Raise Prices



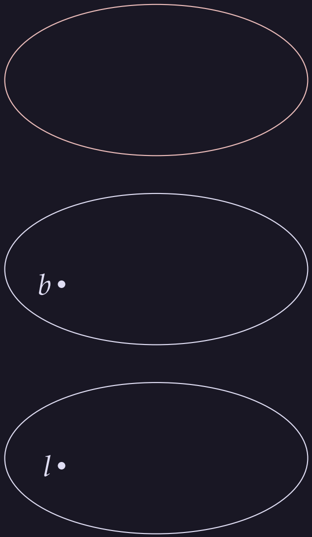
Phase 2: Raise Prices



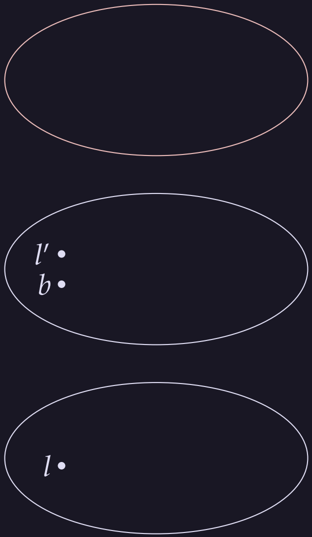
Phase 2: Raise Prices



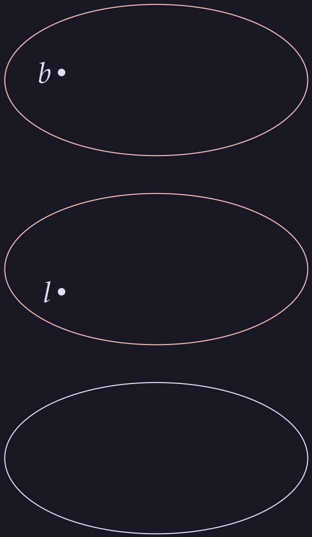
Phase 2: Raise Prices



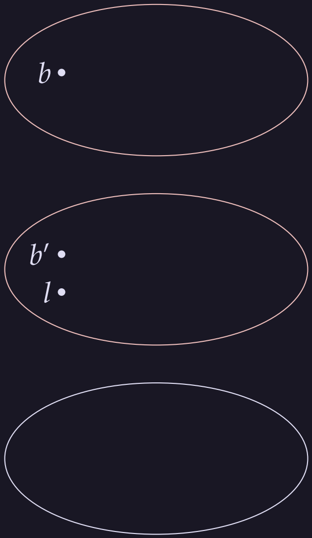
Phase 2: Raise Prices



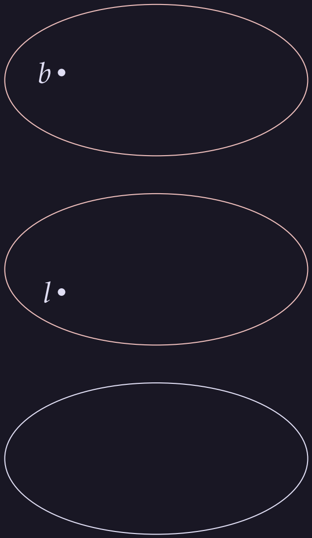
Phase 3: Final Trades



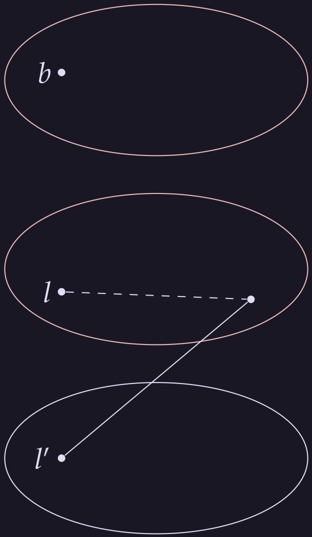
Phase 3: Final Trades



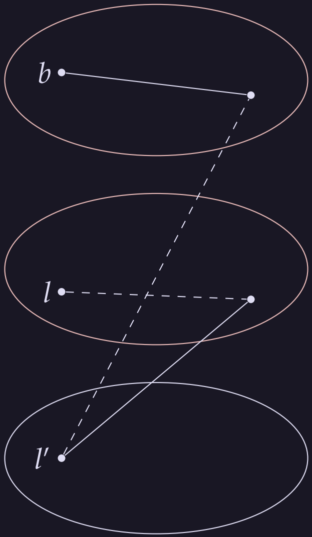
Phase 3: Final Trades



Phase 3: Final Trades



Phase 3: Final Trades



The overall algorithm is:

The overall algorithm is:

- Create simple initial market equilibrium.

Summary

The overall algorithm is:

- Create simple initial market equilibrium.
- Create pEF1 groups $H_1, \ldots, H_r$.

Summary

The overall algorithm is:

- Create simple initial market equilibrium.
- Create pEF1 groups $H_1, \ldots, H_r$.
- Transfer chores from b to l by successively raising prices in groups $H_1, \ldots, H_r$.

Summary

The overall algorithm is:

- Create simple initial market equilibrium.
- Create pEF1 groups $H_1, \dots, H_r$.
- Transfer chores from b to l by successively raising prices in groups $H_1, \dots, H_r$.
- If l ends up in a raised group, transition to phase 3 and trade along alternating paths.

We showed:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

We showed:

Theorem

EF1 + fPO allocations of chores exist under bivalued utilities and can be computed in strongly polynomial time.

Can use similar techniques for:

Theorem

EF + PO allocations of divisible chores exist under bivalued utilities and can be computed in strongly polynomial time.

Open Problems

Main open problems:

Open Problems

Main open problems:

- Do EF1 + PO allocations always exist?

Main open problems:

- Do EF1 + PO allocations always exist?
- If so: can we compute them? If not: is decidability hard?

Open Problems

Main open problems:

- Do EF1 + PO allocations always exist?
- If so: can we compute them? If not: is decidability hard?
- EF + PO allocations of divisible chores are known to always exist. Is there a polynomial time algorithm?

THANK YOUR FOR LISTENING!