How to Fairly Allocate Easy and Difficult Chores

Soroush Ebadian¹
University of Toronto

Dominik Peters
University of Toronto

Nisarg Shah
University of Toronto

¹ Email: sorously@cs.toronto.edu
Outline

• Introduction

• Envy-freeness up-to one item + Pareto Optimality
  • Methods for the division of goods
  • Fisher-market-based Algorithms
  • Adapting to chores

• Maximin-Share Fairness
Fair Division of Indivisible Chores
## Fair Division of Indivisible Chores

| Chore   | Man | Woman 1 | Woman 2 | Woman 3 |
|---------|-----|---------|---------|---------|
| Cleaning | -20 | -20     | -20     | -20     |
| Laundry  | -15 | -10     | -10     | -30     |
| Pet Care | -10 | -20     | -10     | -5      |
| Child Care | -5  | -10     | -50     | -10     |
Fair Division of Indivisible Chores

Fair and Efficient Allocations
More Formally

• $n$ agents
• $m$ indivisible items
• Agent $i$ values item $j$ at $v_{i,j}$
  • Chores Instance: $v_{i,j} \in \mathbb{R}_{\leq 0}$
    • Work shifts between staff, house chores between roommates, ...

• Additive utilities: $v_i(S) = \sum_{j \in S} v_{i,j}$
More Formally

- \( n \) agents
- \( m \) indivisible items
- Agent \( i \) values item \( j \) at \( v_{i,j} \)
  - **Chores** Instance: \( v_{i,j} \in \mathbb{R}_{\leq 0} \)
    - Work shifts between staff, house chores between roommates, ...
  - **Goods** Instance: \( v_{i,j} \in \mathbb{R}_{\geq 0} \)
    - Estate (inheritance) division, divorce settlement, ...

- Additive utilities: \( v_i(S) = \sum_{j \in S} v_{i,j} \)
More Formally

• $n$ agents
• $m$ indivisible items
• Agent $i$ values item $j$ at $v_{i,j}$
  • **Chores** Instance: $v_{i,j} \in \mathbb{R}_{\leq 0}$
    • Work shifts between staff, house chores between roommates, ...
  • **Goods** Instance: $v_{i,j} \in \mathbb{R}_{\geq 0}$
    • Estate (inheritance) division, divorces settlement, ...

• Additive utilities:
  $$v_i(S) = \sum_{j \in S} v_{i,j}$$

Goal: Find an allocation $A = (A_1, A_2, \ldots, A_n)$ that is **fair** and **efficient**.
Gold Standard Fairness Notion

• Envy-Freeness (EF):
  • No agent prefers another one’s bundle to their allocated bundle.
  • I.e., for all pairs of agents $i, j$:
    $$v_i(A_i) \geq v_i(A_j)$$
Gold Standard Fairness Notion

• Envy-Freeness (EF):
  • No agent prefers another one’s bundle to their allocated bundle.
  • I.e., for all pairs of agents $i, j$:
    \[ v_i(A_i) \geq v_i(A_j) \]

Envy-free allocations may not exist.
Gold Standard Fairness Notion

• Envy-Freeness (EF):
  • No agent prefers another one’s bundle to their allocated bundle.
  • I.e., for all pairs of agents $i, j$:
    $$v_i(A_i) \geq v_i(A_j)$$

Envy-free allocations may not exist.
Gold Standard Fairness Notion

- Envy-Freeness (EF):
  - No agent prefers another one’s bundle to their allocated bundle.
  - I.e., for all pairs of agents $i, j$:
    \[ v_i(A_i) \geq v_i(A_j) \]

Envy-free allocations may not exist.
Relaxed Fairness Notion

• Envy-Freeness up to one item ($EF1$):
  • No agent prefers another one's bundle to their allocated bundle, after ignoring at most one item.
  • Chores Instance:
    For all pairs of agents $i, j$:
    $\exists c \in A_i : v_i(A_i \setminus \{c\}) \geq v_i(A_j)$
Relaxed Fairness Notion

• Envy-Freeness up to one item (EF1):
  • No agent prefers another one’s bundle to their allocated bundle, after ignoring at most one item.

• Chores Instance:
  For all pairs of agents $i, j$:
  $$\exists c \in A_i : v_i(A_i \setminus \{c\}) \geq v_i(A_j)$$

• Goods Instance:
  For all pairs of agents $i, j$:
  $$\exists g \in A_j : v_i(A_i) \geq v_i(A_j \setminus \{g\})$$
Relaxed Fairness Notion

• Envy-Freeness up to one item (EF1):
  • No agent prefers another one’s bundle to their allocated bundle, after ignoring at most one item.

• Chores Instance:
  For all pairs of agents $i, j$:
  $$\exists c \in A_i : v_i(A_i \setminus \{c\}) \geq v_i(A_j)$$

• Goods Instance:
  For all pairs of agents $i, j$:
  $$\exists g \in A_j : v_i(A_i) \geq v_i(A_j \setminus \{g\})$$

EF1 allocations *always* exist.
Efficiency Notion

• Pareto Optimality (PO):
  • Allocation $A$ is Pareto optimal, if there is no allocation $B$ such that $\forall i : v_i(B) \geq v_i(A)$, and $\exists j : v_j(B) > v_j(A)$. 
  
$A$ is not Pareto Optimal as $B$ Pareto dominates it.
Fair and Efficient Allocations

Does a fair (EF1) and efficient (PO) allocation always exist?
Does a *fair* (EF1) and *efficient* (PO) allocation always exist?

**Goods**

- EF1 + PO allocations always exist. (Caragiannis et al., 2016)

- Can be found in pseudo-polynomial time. (Barman et al., 2018)

- Poly-time when utility levels are poly-sized / constantly many agents. (Garg et al., 2021)
Fair and Efficient Allocations

Does a fair (EF1) and efficient (PO) allocation always exist?

**Goods**
- EF1 + PO allocations always exist. (Caragiannis et al., 2016)
- Can be found in pseudo-polynomial time. (Barman et al., 2018)
- Poly-time when utility levels are poly-sized / constantly many agents. (Garg et al., 2021)

Open Problem 1.
Can EF1+PO allocations be found in poly time?
Fair and Efficient Allocations

Does a *fair (EF1)* and *efficient (PO)* allocation always exist?

**Chores**
- Still open for additive valuations.

**Our key contribution:**

**Theorem 1.** For *Bivalued* chores, EF1 + PO allocations *always exist*, and can be found in *poly time*.

**Bivalued** utilities:
\[
\forall i, j: v_{i,j} \in \{a, b\}, \quad a \leq b \leq 0,
\]

**Goods**
- EF1 + PO allocations *always exist*. (Caragiannis et al., 2016)

- Can be found in *pseudo-polynomial time*. (Barman et al., 2018)

- Poly-time when utility levels are poly-sized / constantly many agents. (Garg et al., 2021)
Results and Techniques for Goods

• Solution 1: Maximizing Nash Welfare (MNW) i.e., $\max \prod_{A} v_i(A_i)$ or $\max \sum_{A} \log(v_i(A_i))$.
  
  • Goods: MNW yields EF1 + PO (Caragiannis et al., 2016)
Results and Techniques for Goods

• Solution 1: Maximizing Nash Welfare (MNW)  
  i.e., \( \max \prod_{A_i} v_i(A_i) \) or \( \max \sum_{A_i} \log(v_i(A_i)) \).
  
  • Goods: MNW yields EF1 + PO (Caragiannis et al., 2016)

• For integral utilities:  
  Maximizing Harmonic Welfare yields EF1 + PO  
  i.e., \( \max \sum_{A_i} H(v_i(A_i)) \) (Montanari et al., 2022)
  
  • \( H(i) = 1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{i} \)
Results and Techniques for Goods

• **Solution 1:** Maximizing Nash Welfare (MNW)
  i.e., \( \max_A \prod_i v_i(A_i) \).

• **Goods:** MNW yields EF1 + PO (Caragiannis et al., 2016)

• **Chores:**
  1. Maximizing \( \prod_i |v_i(A_i)| \) or \( \prod_i (v_i(A) - v_i(A_i)) \)
     No, favors higher disutilities. No, counter example.
  2. Maximizing \( \prod_i |v_i(A_i)| \) subject to PO?
     No, fails EF1. (Example with bivalued utilities, n=4, m=8)
  3. Minimizing \( \prod_i |v_i(A_i)| \)
     No, favors having an idle agent with no tasks.
Results and Techniques for Goods

• Solution 1: Maximizing Nash Welfare (MNW)
  • Extension to Chores: not obvious.
Results and Techniques for Goods

• Solution 1: Maximizing Nash Welfare (MNW)
  • Extension to Chores: not obvious.

• Solution 2: Fisher market adaptation (Barman et al. (2018))
  • Finds an EF1 + PO allocation in pseudo-poly time
  • Idea: a local search terminates due to invariants and potential functions
Results and Techniques for Goods

- **Solution 1:** Maximizing Nash Welfare (MNW)
  - Extension to Chores: not obvious.

- **Solution 2:** Fisher market adaptation (Barman et al. (2018))
  - Finds an EF1 + PO allocation in pseudo-poly time
  - Idea: a *local search* terminates due to *invariants* and potential functions

- Extension to Chores:
  - Non-trivial, *invariants* cease to hold
  - Our result:
    - With a more intricate analysis $\rightarrow$ EF1 + PO for *Bivalued* Utilities
Fisher Markets in Fair Division
Fisher Markets

Setup:
• $n$ agents, $m$ items
• Item prices: $p_j \in \mathbb{R}_{\geq 0}$

Def. Bang per Buck: $\frac{v_{i,j}}{p_j}$

Maximum Bang per Buck: $MBB_i = \max_j \frac{v_{i,j}}{p_j}$

| Item  | Price | BB: | Value |
|-------|-------|-----|-------|
| $\text{Clothing}$ | $2$ | -3 | -1.5 |
| $\text{Food}$ | $10$ | -20 | -2   |
| $\text{Cleaning}$ | $10$ | -15 | -1.5 |

| Item  | Price | BB: | Value |
|-------|-------|-----|-------|
| $\text{Clothing}$ | $2$ | -3 | -1.5 |
| $\text{Food}$ | $10$ | -20 | -2   |
| $\text{Cleaning}$ | $10$ | -15 | -1.5 |

$2$ $10$ $10$
Fisher Markets

Setup:
- \( n \) agents, \( m \) items
- Item prices: \( p_j \in \mathbb{R}_{\geq 0} \)

**Def.** Bang per Buck: \( \frac{v_{i,j}}{p_j} \)

Maximum Bang per Buck: \( \text{MBB}_i = \max_j \frac{v_{i,j}}{p_j} \)

Equilibrium:
- All items are allocated
- Agents are only allocated MBB items

First Welfare Theorem:

Pareto Optimal (PO)
Fisher Markets

• An allocation is price envy-free up to one item (pEF1) if for all pairs of agents $i, j$:
  \[ \exists c \in A_j : p(A_j \setminus \{c\}) \leq p(A_i) \]

\[ \text{pEF1 + equilibrium} \rightarrow \text{EF1 + PO} \]
Fisher Markets

• An allocation is price envy-free up to one item \((pEF1)\) if for all pairs of agents \(i, j\):
  \[
  \exists c \in A_j : \ p (A_j \setminus \{c\}) \leq p(A_i)
  \]

\(\text{pEF1} \ + \ \text{equilibrium} \quad \rightarrow \quad \text{EF1} \ + \ \text{PO}\)

• Algorithmic Framework:
  • Start with an allocation and prices in equilibrium
  • Make local changes reducing envy \((\text{while remaining in equilibrium})\)
  • Reach pEF1 \((+ \ \text{equilibrium})\)
Fisher Market Algorithm Ideas

• MBB Graph
  • Edge $i \leftarrow j$

• Local changes:
  Suppose $i \leftarrow j$ exists and $p(A_i) < p(A_j) - p_c$ (violation of pEF1), then transferring $c$ to $i_1$ (1) remains in the equilibrium (2) reduces envy “overall”
Fisher Market Algorithm for Goods

• MBB Graph
  • Edge $i \leftarrow j$

• Algorithm Sketch for Goods (Barman et al. 2018)
  1. Start with welfare maximizing allocation
  2. Least Spender: $ls = \arg\min p(A_i)$
  3. While there is $ls \leftarrow i_2 \leftarrow i_3 \leftarrow \ldots \leftarrow i_{\ell}$
     where $ls$ price envies $i_{\ell}$:
        Take the shortest path, make a local transfer, go to 2.
  4. If not pEF1:
     Raise prices of items allocated to $ls$ and agents reaching $ls$, go to 2.
Fisher Market Algorithm for Goods

• MBB Graph
  • Edge \( i \leftarrow j \)

• Algorithm Sketch for Goods (Barman et al. 2018)
  1. Start with welfare maximizing allocation
  2. Least Spender: \( ls = \arg\min p(A_i) \)
  3. While there is \( ls \leftarrow i_2 \leftarrow i_3 \leftarrow ... \leftarrow i_\ell \) where \( ls \) price envies \( i_\ell \):
     Take the shortest path, make a local transfer, go to 2.
  4. If not pEF1:
     Raise prices of items allocated to \( ls \) and agents reaching \( ls \), go to 2.

Key Invariants:
\[
\downarrow \max_{i} \min_{c \in A_i} p(A_i) - p(c) \\
\uparrow \min_{i} p(A_i)
\]
Attempt 1: Algorithm for Chores

- MBB Graph
  - Edge $i \leftarrow j$

- Sketch of Adaptation for Chores
  1. Start with welfare maximizing allocation
  2. Least Spender: $ls = \arg\min_i p(A_i)$
  3. While there is $ls \leftarrow i_2 \leftarrow i_3 \leftarrow \ldots \leftarrow i_\ell$
     where $ls$ price envies $i_\ell$:
     Take the shortest path, make a local transfer, go to 2.
  4. If not pEF1:
     Raise Reduce prices of items allocated to $ls$ and agents reaching $ls$, go to 2.
Algorithm for Bivalued Chores

[Phase 1: Init]
1. Start with welfare maximizing allocation

[Phase 2a]
2.

[Phase 2b: Reallocate chores]
3. Least Spender: \( l_s = \arg\min_i p(A_i) \)
4. While there is \( l_s \leftarrow i_2 \leftarrow i_3 \leftarrow \ldots \leftarrow i_\ell \)
   where \( l_s \) price envies \( i_\ell \) after removing \( c_{\ell-1} \):
   Take the shortest path, make a local transfer, go to 3.

[Phase 3: Price Reduction]
5. If not pEF1:
   Reduce prices of items allocated to \( l_s \) and agents reaching \( l_s (H_k) \), go to 2.
Algorithm for Bivalued Chores

[Phase 1: Init]
1. Start with welfare maximizing allocation, \( k = 0 \)

[Phase 2a]
2. Eliminate price envy between \( H_k \)'s, \( k = k + 1 \)

[Phase 2b: Reallocate chores]
3. Least Spender: \( l_s = \arg \min_i p(A_i) \)
4. While there is \( l_s \leftarrow i_2 \leftarrow i_3 \leftarrow \ldots \leftarrow i_\ell \) where \( l_s \) price envies \( i_\ell \) after removing \( c_{\ell-1} \):
   Take the shortest path, make a local transfer, go to 3.

[Phase 3: Price Reduction]
5. If not pEF1:
   Reduce prices of items allocated to \( l_s \) and agents reaching \( l_s \), go to 2.
Algorithm for Bivalued Chores

[Phase 1: Init]
1. Start with welfare maximizing allocation, $k = 0$

[Phase 2a]
2. Eliminate price envy between $H_k$’s, $k = k + 1$

[Phase 2b: Reallocate chores]
3. Least Spender: $ls = \arg\min_i p(A_i)$
4. While there is $ls \leftarrow i_2 \leftarrow i_3 \leftarrow ... \leftarrow i_\ell$
   where $ls$ price envies $i_\ell$ after removing $c_{\ell - 1}$:
   Take the shortest path, make a local transfer, go to 3.

[Phase 3: Price Reduction]
5. If not pEF1:
   Reduce prices of items allocated to $ls$ and agents reaching $ls$, go to 2.

Key Idea:
We can make $H_k$’s disjoint.
Each agent experiences price reduction at most once.
⇒ At most $n$ Phase 3’s

Proof by Induction.
So far

• **EF1 + PO allocations** always exist for **bivalued chores**

• **Major open problems:**
  
  • Complexity of **EF1 + PO allocations** for goods?
  
  • Does **EF1 + PO allocations** always exist for chores?

• **Chores division** seems *harder* than **Goods division**
Maximin Share Fairness
Another Fairness Notion

• Maximin Share (MMS) Allocation (Budish, 2011)
  
  • For all agents $i$,

  \[ v_i(A_i) \geq MMS_i \quad \text{(MMS value)} \]

• MMS value of agent $i$:

\[
MMS_i = \max_{P=(P_1,P_2,\ldots,P_n)} \min_{P_j \in P} v_i(P_j)
\]
Another Fairness Notion

- **Maximin Share (MMS) Allocation** (Budish, 2011)
  - For all agents $i$,
    \[ v_i(A_i) \geq \text{MMS}_i \quad \text{(MMS value)} \]
  - **MMS value** of agent $i$:
    \[
    \text{MMS}_i = \max_{P=(P_1,P_2,\ldots,P_n)} \min_{P_j \in P} v_i(P_j)
    \]
  - Finding MMS values is **NP-hard**.
MMS Allocations

• **MMS allocations may not exist in general.** (Procaccia et al. (2014), Kurokowa et al. (2016))
  
  • Approach 1: Approximation results for general instances (Huang and Lu (2021), Garg and Taki (2020), ...)
  
  • Approach 2: Existential results for **restricted** instances
    
    • Binary: $v_{i,j} \in \{0, 1\}$ or $v_{i,j} \in \{0, -1\}$
    
    • Ternary: $v_{i,j} \in \{0, 1, 2\}$ (Amanatidis et al. (2017))
    
    • Lexicographical (Hosseini et al. (2018))
    
    • **Two other classes of utilities** (This work)
Factored Valuations

- Factored valuations:
  \[ v_{i,j} \in \{0, p_1, p_2, \ldots, p_k\} \mid p_\ell = p_{\ell-1} \cdot q, \text{ for some } q \in \mathbb{N}. \]

Lemma. For factored valuations, **MMS value** and a corresponding partition can be found in **poly-time**.
Personalized Factored Bivalued

- Personalized Bivalued: $v_{i,j} \in \{a_i, b_i\}$
- Factored: $\frac{a_i}{b_i} \in \mathbb{N}$

**Theorem 2 (a).** For personalized *factored* bivalued chores or goods:
- MMS allocation always exist
- MMS + PO allocation can be found in poly time

- Feige (2022): MMS exists for bivalued utilities (non-personalized).
Weakly Lexicographic Preferences

• Agents rank items by undesirability allowing ties
  Undesirability levels: \( \{a \sim b \sim d\} > \{e \sim f\} > \{g \sim h \sim k\} \)

• Ties within a level: \( c \sim c' \rightarrow |v_{i,c}| = |v_{i,c'}| \)
  E.g. \( v_{i,a} = v_{i,b} \)

• Lexicographic preference between levels: \( |v_{i,c}| > \sum_{c', < c} |v_{i,c'}| \)
  E.g. \( |v_i(a)| > |v_i(\{e, f, g, h, k\})| \)

**Theorem 2 (b).** For weakly lexicographic chores or goods:
• MMS allocation always exist
• MMS + PO allocation can be found in poly time
Conclusion and Future Work

- EF1 + PO exists for bivalued chores
  - Chores seem harder than Goods
- MMS exists for two subclasses of factored utilities
  - Weakly lexicographic, Personalized factored bivalued

Open questions
- EFX + PO for bivalued?
  - EFX: no envy after removing any chore
- EF1 + PO for trivalued or weakly lexicographic instances?
- MMS for factored valuations?

Acknowledgements
Our EF1 + PO result was recently independently obtained by a AAAI paper, using a similar technique (Garg et al. (2022))
Thank you!