Developments in Fair Resource Allocation: Fair Division of Mixed Divisible and Indivisible Goods

Haris Aziz **Xinhang Lu** Mashbat Suzuki Toby Walsh

UNSW Sydney

AJCAI 2022 Tutorial (Part 3)
Perth, Australia, 05 December 2022

Motivation

Overview

1. Mixed-Goods Model
2. Envy-freeness for Mixed Goods (EFM)
3. Maximin Share (MMS) Guarantee

Cake Cutting (aka Divisible Goods Allocation)

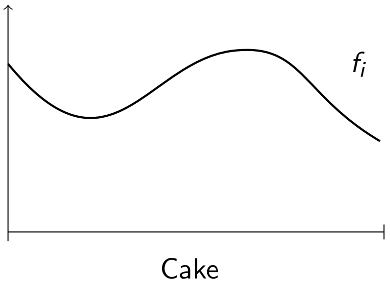
Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \, \mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.

Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

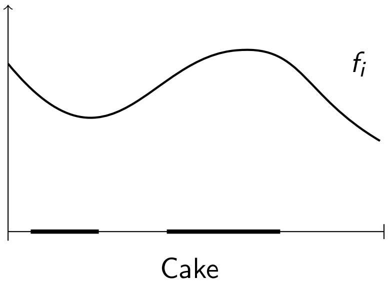
- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \, \mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.



Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

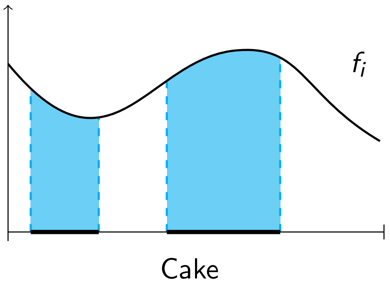
- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \, \mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.



Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

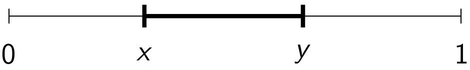
- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \, \mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.



Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \, \mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.
- Robertson-Webb (RW) model:
  - $\textsc{Eval}_i(x, y)$ asks agent i to evaluate the interval $[x, y]$ and returns the value $u_i([x, y])$;
  - $\textsc{Cut}_i(x, \alpha)$ asks agent i to return the leftmost point y such that $u_i([x, y]) = \alpha$, or state that no such point exists.

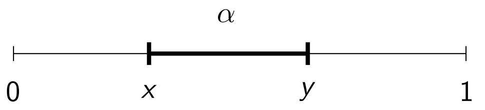


$\textsc{Eval}_i(x, y) = $ **?**

Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \,\mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.
- Robertson-Webb (RW) model:
  - $\textsc{Eval}_i(x, y)$ asks agent i to evaluate the interval $[x, y]$ and returns the value $u_i([x, y])$;
  - $\textsc{Cut}_i(x, \alpha)$ asks agent i to return the leftmost point y such that $u_i([x, y]) = \alpha$, or state that no such point exists.

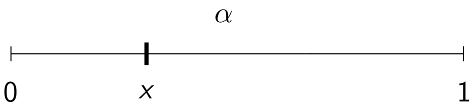


$\textsc{Eval}_i(x, y) = \alpha$

Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \,\mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.
- Robertson-Webb (RW) model:
  - $\textsc{Eval}_i(x, y)$ asks agent i to evaluate the interval $[x, y]$ and returns the value $u_i([x, y])$;
  - $\textsc{Cut}_i(x, \alpha)$ asks agent i to return the leftmost point y such that $u_i([x, y]) = \alpha$, or state that no such point exists.

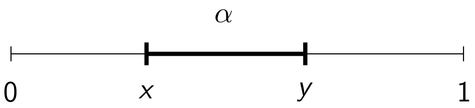


$\textsc{Cut}_i(x, \alpha) = \textbf{?}$

Cake Cutting (aka Divisible Goods Allocation)

Agents $N = \{1, 2, \ldots, n\}$ divide cake $C = [0, 1]$

- Agent i has a density function $f_i \colon [0, 1] \to \mathbb{R}_{\geq 0}$.
- Given a piece of cake $S \subseteq [0, 1]$, agent i has value $u_i(S) = \int_S f_i \, \mathrm{d}x$.
- Allocation: Partition of the cake $(C_1, C_2, \ldots, C_n)$.
- Robertson-Webb (RW) model:
  - $\textsc{Eval}_i(x, y)$ asks agent i to evaluate the interval $[x, y]$ and returns the value $u_i([x, y])$;
  - $\textsc{Cut}_i(x, \alpha)$ asks agent i to return the leftmost point y such that $u_i([x, y]) = \alpha$, or state that no such point exists.



$\textsc{Cut}_i(x, \alpha) = y$

Fairness

Envy-freeness (EF)

For any pair of agents i, j,

$$u_i(C_i) \geq u_i(C_j).$$

Theorem (Alon [1987] and Aziz and Mackenzie [2016])

An envy-free allocation

- *always exists;*
- *can be found via a discrete and bounded protocol.*

Indivisible Goods Allocation

Agents $N = \{1, 2, \ldots, n\}$ divide indivisible goods $M = \{1, 2, \ldots, m\}$

- Agent i has $u_i(g) \geq 0$ for each good g.
- Additive utility: $u_i(M') = \sum_{g \in M'} u_i(g)$ for each subset of goods M'.
- Allocation: Partition of the goods $\mathcal{M} = (M_1, M_2, \ldots, M_n)$.

Envy-freeness up to one good (EF1)

For any agents i, j, there exists $g \in M_j$ such that

$$u_i(M_i) \geq u_i(M_j \setminus \{g\}).$$

Theorem (Lipton et al. [2004])

An EF1 allocation always exists and can be found in polynomial time.

Mixed-Goods Model

- Agents $N = \{1, 2, \ldots, n\}$
- m indivisible goods and a cake
- Each agent has
  utility function for the indivisible goods;
  density function for the cake.
- Allocation $\mathcal{A} = (A_1, A_2, \ldots, A_n)$, where $A_i = M_i \cup C_i$
  Indivisible goods: $(M_1, M_2, \ldots, M_n)$
  Cake: $(C_1, C_2, \ldots, C_n)$
- Utility $u_i(A_i) = u_i(M_i) + u_i(C_i)$

Candidate Fairness Notions

- Envy-freeness (EF): No agent envies another.

$$\forall i, j \in N, u_i(A_i) \geq u_i(A_j)$$

- Envy-freeness up to one (indivisible) good (EF1): Any envy that an agent has towards another agent can be eliminated by removing *some* good from the latter agent's bundle.

$$\forall i, j \in N, \exists g \in A_j \text{ such that } u_i(A_i) \geq u_i(A_j \setminus \{g\})$$

- EF for divisible goods + EF1 for indivisible goods.

Alice and Bob divide three indivisible goods and two dollars

Alice	5	4	3		
Bob	5	4	3		

Candidate Fairness Notions

- Envy-freeness (EF): No agent envies another.

$$\forall i, j \in N, u_i(A_i) \geq u_i(A_j)$$

- Envy-freeness up to one (indivisible) good (EF1): Any envy that an agent has towards another agent can be eliminated by removing *some* good from the latter agent's bundle.

$$\forall i, j \in N, \exists g \in A_j \text{ such that } u_i(A_i) \geq u_i(A_j \setminus \{g\})$$

- EF for divisible goods + EF1 for indivisible goods.

Alice and Bob divide three indivisible goods and two dollars

Alice	5	4	3
Bob	5	4	3

Candidate Fairness Notions

- Envy-freeness (EF): No agent envies another.

$$\forall i, j \in N, u_i(A_i) \geq u_i(A_j)$$

- Envy-freeness up to one (indivisible) good (EF1): Any envy that an agent has towards another agent can be eliminated by removing *some* good from the latter agent's bundle.

$$\forall i, j \in N, \exists g \in A_j \text{ such that } u_i(A_i) \geq u_i(A_j \setminus \{g\})$$

- EF for divisible goods + EF1 for indivisible goods.

Alice and Bob divide three indivisible goods and two dollars

Alice	5	4	3
Bob	5	4	3

Candidate Fairness Notions

- Envy-freeness (EF): No agent envies another.

$$\forall i, j \in N, u_i(A_i) \geq u_i(A_j)$$

- Envy-freeness up to one (indivisible) good (EF1): Any envy that an agent has towards another agent can be eliminated by removing *some* good from the latter agent's bundle.

$$\forall i, j \in N, \exists g \in A_j \text{ such that } u_i(A_i) \geq u_i(A_j \setminus \{g\})$$

- EF for divisible goods + EF1 for indivisible goods.

Alice and Bob divide three indivisible goods and two dollars

Alice	5	4	3
Bob	5	4	3

Alice	5	4	3
Bob	5	4	3

Envy-freeness for Mixed Goods (EFM)

Definition (EFM [Bei, Li, Liu, Liu, and Lu, 2021])

For any pair of agents i, j,

- if agent j's bundle consists of *only* indivisible goods, there exists $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \setminus \{g\})$;
- otherwise, $u_i(A_i) \geq u_i(A_j)$.

With only divisible goods: EFM reduces to EF.

With only indivisible goods: EFM reduces to EF1.

Envy-freeness for Mixed Goods (EFM)

Definition (EFM [Bei, Li, Liu, Liu, and Lu, 2021])

For any pair of agents i, j,

- if agent j's bundle consists of *only* indivisible goods, there exists $g \in A_j$ such that $u_i(A_i) \geq u_i(A_j \setminus \{g\})$;
- otherwise, $u_i(A_i) \geq u_i(A_j)$.

With only divisible goods: EFM reduces to EF.

With only indivisible goods: EFM reduces to EF1.

EFM Existence

Theorem (Bei, Li, Liu, Liu, and Lu [2021])

EFM allocations always exist for any number of agents and can be found in polynomial time.

Proof Sketch.

- Start with an EF1 allocation of indivisible goods.
- Iteratively (and carefully) add some cake.
- Maintain EFM throughout the process.

□

EFM Existence

Theorem (Bei, Li, Liu, Liu, and Lu [2021])

EFM allocations always exist for any number of agents and can be found in polynomial time.

Proof Sketch.

- Start with an EF1 allocation of indivisible goods.
- Iteratively (and carefully) add some cake.
- Maintain EFM throughout the process. □

Envy Graph

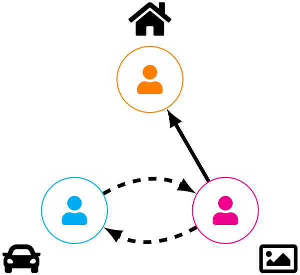
Definition

A directed graph of agents with

Envy edge: $i \longrightarrow j$ if $u_i(A_i) < u_i(A_j)$;

Equality edge: $i \dashrightarrow j$ if $u_i(A_i) = u_i(A_j)$.

	5	4	1	5
	3	4	4	5
	4	3	3	5

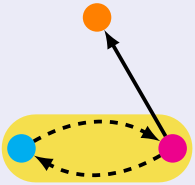


Addable Set

Definition

A subset of agents $S \subseteq N$ such that

- no envy edge in S;
- no edge from $N \setminus S$ to S.



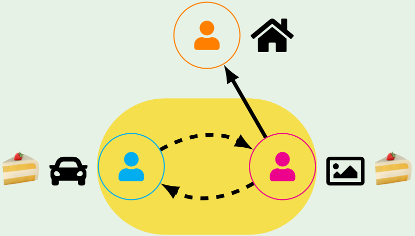
Intuition

Add some cake to an addable set (in a "perfect" manner).

Cake-Adding Phase

Add some cake to the maximal addable set S

5	4	1	5
3	4	4	5
4	3	3	5



Perfect allocation [Alon, 1987]

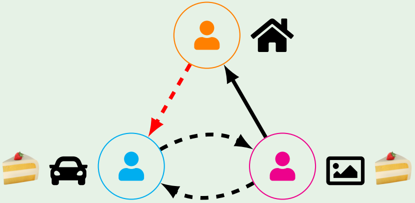
Every agent in N values all $|S|$ pieces equally.

Given an EFM allocation, after a cake-adding phase, the resulting allocation is still EFM.

Cake-Adding Phase

Add some cake to the maximal addable set S

	5	4	1	5
	3	4	4	5
	4	3	3	5



Perfect allocation [Alon, 1987]

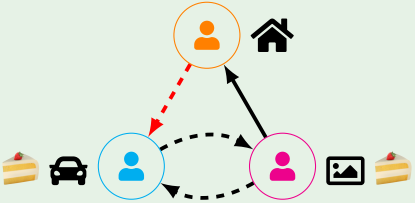
Every agent in N values all $|S|$ pieces equally.

Given an EFM allocation, after a cake-adding phase, the resulting allocation is still EFM.

Cake-Adding Phase

Add some cake to the maximal addable set S

	5	4	1	5
	3	4	4	5
	4	3	3	5



Perfect allocation [Alon, 1987]

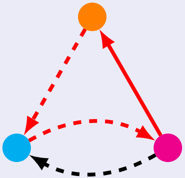
Every agent in N values all $|S|$ pieces equally.

Given an EFM allocation, after a cake-adding phase, the resulting allocation is still EFM.

Envy Cycle

Definition

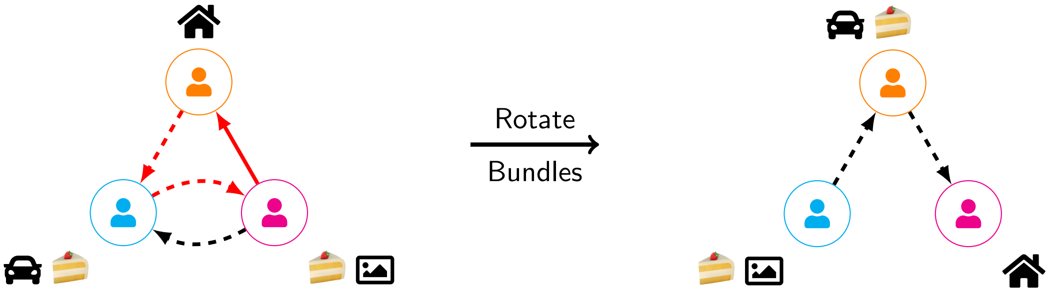
A cycle in the envy graph with at least one *envy* edge.



Intuition

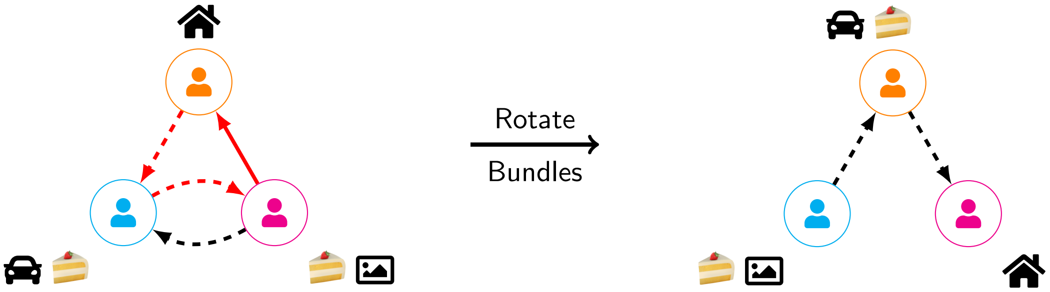
Eliminate an envy cycle by rotating bundles.

Envy-Cycle-Elimination Phase



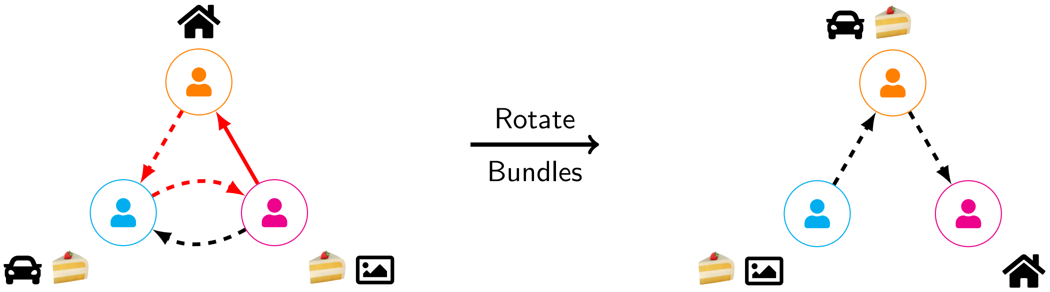
Given an EFM allocation, after an envy-cycle-elimination phase, the allocation is still EFM.

Envy-Cycle-Elimination Phase



Given an EFM allocation, after an envy-cycle-elimination phase, the allocation is still EFM.

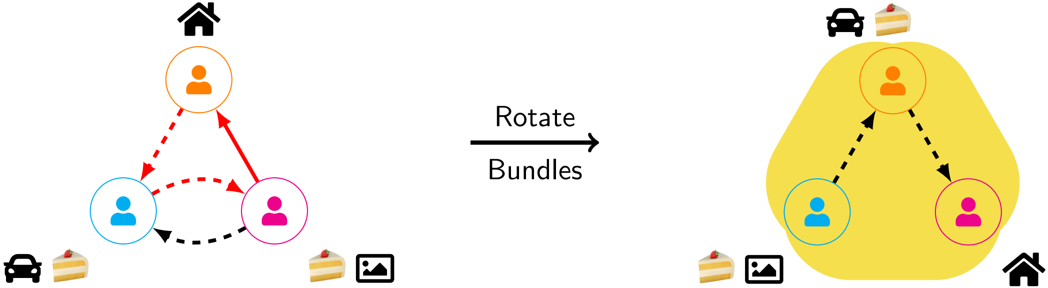
Envy-Cycle-Elimination Phase



Given an EFM allocation, after an envy-cycle-elimination phase, the allocation is still EFM.

What can we do now?

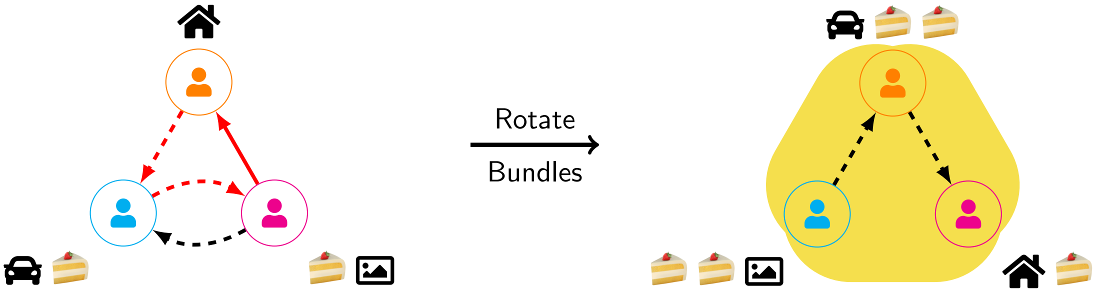
Envy-Cycle-Elimination Phase



Given an EFM allocation, after an envy-cycle-elimination phase, the allocation is still EFM.

What can we do now?

Envy-Cycle-Elimination Phase



Given an EFM allocation, after an envy-cycle-elimination phase, the allocation is still EFM.

What can we do now?

Connection Between Addable Set and Envy Cycle

Key Lemma [Bei, Li, Liu, Liu, and **Lu**, 2021]

At any time, there exists either an addable set or an envy cycle.

- Always make progress.
- The partial allocation is always EFM.
- The process always terminates.

Caveat

- A polynomial-time algorithm if we have a perfect allocation orcale for cake cutting.
- The perfect allocation oracle cannot be implemented in a bounded time in the Robertson-Webb model.

Open Question

A bounded (or even finite) EFM protocol in the Robertson-Webb model?

Caveat

- A polynomial-time algorithm if we have a perfect allocation orcale for cake cutting.
- The perfect allocation oracle cannot be implemented in a bounded time in the Robertson-Webb model.

Open Question

A bounded (or even finite) EFM protocol in the Robertson-Webb model?

More Open Questions

- EFM with economic efficiency considerations (like Pareto Optimality).
    - Preliminary results in Bei, Li, Liu, Liu, and **Lu** [2021]
- EFM with both goods and *chores* (items that yield non-positive utilities).
    - Recent progress by Bhaskar, Sricharan, and Vaish [2021]
- Fair division in the presence of strategic agents.
- . . .

Maximin Share (MMS) Guarantee

Definition (MMS [Budish, 2011])

- Define the maximin share (MMS) of agent i as

$$\text{MMS}_i = \max_{(P_1, P_2, \dots, P_n)} \min_{j \in [n]} u_i(P_j).$$

- Allocation $(A_1, \dots, A_n)$ is said to satisfy the maximin share (MMS) guarantee if for every agent $i \in N$,

$$u_i(A_i) \geq \text{MMS}_i.$$

						MMS
	0.5	0.5	0	0	0.5	0.5
	0.9	0.2	0.3	0.6	1	1
	1	0.2	0.1	0.7	1	1

Maximin Share (MMS) Guarantee

Definition (MMS [Budish, 2011])

- Define the maximin share (MMS) of agent i as

$$\text{MMS}_i = \max_{(P_1, P_2, \ldots, P_n)} \min_{j \in [n]} u_i(P_j).$$

- Allocation $(A_1, \ldots, A_n)$ is said to satisfy the α-approximate MMS guarantee (α-MMS), for some $\alpha \in [0, 1]$, if $\forall i \in N$,

$$u_i(A_i) \geq \alpha \cdot \text{MMS}_i.$$

						MMS
	0.5	0.5	0	0	0.5	0.5
	0.9	0.2	0.3	0.6	1	1
	1	0.2	0.1	0.7	1	1

MMS with Indivisible Goods

- With indivisible goods, MMS guarantee cannot always be satisfied, but a constant multiplicative approximation can [Kurokawa, Procaccia, and Wang, 2018].
- Better approximation ratio, simpler algorithms, tighter negative example, etc. [Amanatidis et al., 2017; Garg, McGlaughlin, and Taki, 2019; Barman and Krishnamurthy, 2020; Ghodsi et al., 2021; Garg and Taki, 2021; Feige, Sapir, and Tauber, 2021] . . .

Research Questions

1. Is the worst-case MMS approximation guarantee with mixed goods the same as that with only indivisible goods?
2. Given any problem instance, would adding some divisible goods to it always (weakly) increase the MMS approximation ratio of this instance?
3. How to design algorithms that finds allocations with good MMS approximation guarantee?

Theorem (Bei, Liu, Lu, and Wang [2021])

Given any mixed goods problem instance, an α-MMS allocation always exists, where

$$\alpha = \min\left\{1, \frac{1}{2} + \min_{i \in N}\left\{\frac{\text{agent } i\text{'s value for the divisible goods}}{2 \cdot (n-1) \cdot \text{agent } i\text{'s maximin share}}\right\}\right\}$$

4. Algorithms with *better* MMS approximation guarantee ?

Research Questions

1. Is the worst-case MMS approximation guarantee with mixed goods the same as that with only indivisible goods? ✔
2. Given any problem instance, would adding some divisible goods to it always (weakly) increase the MMS approximation ratio of this instance?
3. How to design algorithms that finds allocations with good MMS approximation guarantee?

Theorem (Bei, Liu, Lu, and Wang [2021])

Given any mixed goods problem instance, an α-MMS allocation always exists, where

$$\alpha = \min\left\{1, \frac{1}{2} + \min_{i \in N}\left\{\frac{\text{agent } i\text{'s value for the divisible goods}}{2 \cdot (n-1) \cdot \text{agent } i\text{'s maximin share}}\right\}\right\}$$

4. Algorithms with *better* MMS approximation guarantee ?

Research Questions

1. Is the worst-case MMS approximation guarantee with mixed goods the same as that with only indivisible goods? ✔
2. Given any problem instance, would adding some divisible goods to it always (weakly) increase the MMS approximation ratio of this instance? ✘
3. How to design algorithms that finds allocations with good MMS approximation guarantee?

Theorem (Bei, Liu, Lu, and Wang [2021])

Given any mixed goods problem instance, an α-MMS allocation always exists, where

$$\alpha = \min\left\{1, \frac{1}{2} + \min_{i \in N}\left\{\frac{\textit{agent } i\textit{'s value for the divisible goods}}{2 \cdot (n-1) \cdot \textit{agent } i\textit{'s maximin share}}\right\}\right\}$$

4. Algorithms with *better* MMS approximation guarantee ?

Research Questions

1. Is the worst-case MMS approximation guarantee with mixed goods the same as that with only indivisible goods? ✔
2. Given any problem instance, would adding some divisible goods to it always (weakly) increase the MMS approximation ratio of this instance? ✘
3. How to design algorithms that finds allocations with good MMS approximation guarantee?

Theorem (Bei, Liu, Lu, and Wang [2021])

Given any mixed goods problem instance, an α-MMS allocation always exists, where

$$\alpha = \min\left\{1, \frac{1}{2} + \min_{i \in N}\left\{\frac{\text{agent } i\text{'s value for the divisible goods}}{2 \cdot (n-1) \cdot \text{agent } i\text{'s maximin share}}\right\}\right\}$$

4. Algorithms with *better* MMS approximation guarantee ?

Research Questions

1. Is the worst-case MMS approximation guarantee with mixed goods the same as that with only indivisible goods? ✔
2. Given any problem instance, would adding some divisible goods to it always (weakly) increase the MMS approximation ratio of this instance? ✘
3. How to design algorithms that finds allocations with good MMS approximation guarantee?

Theorem (Bei, Liu, Lu, and Wang [2021])

Given any mixed goods problem instance, an α-MMS allocation always exists, where

$$\alpha = \min\left\{1, \frac{1}{2} + \min_{i \in N}\left\{\frac{\textit{agent } i\textit{'s value for the divisible goods}}{2 \cdot (n-1) \cdot \textit{agent } i\textit{'s maximin share}}\right\}\right\}$$

4. Algorithms with *better* MMS approximation guarantee ?

Algorithms for Computing Approximate MMS Allocations

High-level Idea

- Assign some agent i a bundle with value at least $\alpha \times \text{MMS}_i$;
- Reduce the problem to a smaller size.

Example ($\alpha = 0.75$)

						MMS	α-MMS
	0.5	0.5	0	0	0.5	0.5	0.375
	0.9	0.2	0.3	0.6	1	1	0.75
	1	0.2	0.1	0.7	1	1	0.75

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						MMS	α-MMS	$(1-\alpha) \times$ MMS
	0.5	0.5	0	0	0.5	0.5	0.375	0.125
	0.9	0.2	0.3	0.6	1	1	0.75	0.25
	1	0.2	0.1	0.7	1	1	0.75	0.25

Lemma (Bei, Liu, Lu, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times \text{MMS}$
	0.5	0.5	0	0	0.5	0.5	0.375	0.125
	0.9	0.2	0.3	0.6	1		0.75	0.25
	1	0.2	0.1	0.7	1		0.75	0.25

Lemma (Bei, Liu, Lu, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times$ MMS
Agent 1	0.5	0.5	0	0		0.5	0.375	0.125
Agent 2	0.9	0.2	0.3	0.6	1		0.75	0.25
Agent 3	1	0.2	0.1	0.7	1		0.75	0.25

Lemma (Bei, Liu, **Lu**, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times$ MMS
	0.5	0.5	0	0		0.5	0.375	0.125
	0.9	0.2	0.3	0.6	1		0.75	0.25
	1	0.2	0.1	0.7	1		0.75	0.25

Lemma (Bei, Liu, **Lu**, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times \text{MMS}$
	0.5	0.5	0	0		0.5	0.375	0.125
	0.9	0.2	0.3	0.6	1		0.75	0.25
	1	0.2	0.1	0.7	1		0.75	0.25

Lemma (Bei, Liu, Lu, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times$ MMS
	0.5	0.5	0	0		0.5	0.375	0.125
	0.9	0.2	0.3	0.6		0.75	0.75	0.25
	1	0.2	0.1	0.7	1		0.75	0.25

Lemma (Bei, Liu, Lu, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times \text{MMS}$
	0.5	0.5	0	0		0.5	0.375	0.125
	0.9	0.2	0.3	0.6		0.75	0.75	0.25
	1	0.2	0.1	0.7	1		0.75	0.25

Lemma (Bei, Liu, **Lu**, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \mathrm{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \mathrm{MMS}_j$.

						Utility	α-MMS	$(1-\alpha) \times \mathrm{MMS}$
	0.5	0.5	0	0		0.5	0.375	0.125
	0.9	0.2	0.3	0.6		0.75	0.75	0.25
	1	0.2	0.1	0.7		1.45	0.75	0.25

Lemma (Bei, Liu, **Lu**, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for *Homogeneous* Cake $\widehat{C}$

The Algorithm

- Phase 1: Allocate big indivisible goods.
- Phase 2: Allocate small indivisible goods and cake $\widehat{C}$:
  1. $u_{i^*}(A_{i^*}) \geq \alpha \cdot \text{MMS}_{i^*}$;
  2. For each agent j remaining in N, $u_j(A_{i^*}) \leq \text{MMS}_j$.

					Utility	α-MMS	$(1-\alpha) \times$ MMS
	0.5	0.5	0	0	0.5	0.375	0.125
	0.9	0.2	0.3	0.6	0.75	0.75	0.25
	1	0.2	0.1	0.7	1.45	0.75	0.25

Lemma (Bei, Liu, **Lu**, and Wang [2021])

Cake $\widehat{C}$ is enough to be allocated during the algorithm's run.

Algorithm for Heterogeneous Cake C

- Replace cake C with a homogeneous cake $\widehat{C}$ such that

$$u_i(\widehat{C}) = u_i(C).$$

- Allocate the indivisible goods and homogeneous cake $\widehat{C}$ using the previous algorithm. In other words, for each agent i, we have

$$u_i(M_i \cup \widehat{C}_i) = u_i(M_i) + u_i(\widehat{C}_i) \geq \alpha \cdot \text{MMS}_i.$$

- Use an algorithm of Cseh and Fleiner [2020] to allocate cake C in the sense that

$$u_i(C_i) \geq u_i(\widehat{C}_i).$$

Algorithm for Heterogeneous Cake C

- Replace cake C with a homogeneous cake $\widehat{C}$ such that

$$u_i(\widehat{C}) = u_i(C).$$

- Allocate the indivisible goods and homogeneous cake $\widehat{C}$ using the previous algorithm. In other words, for each agent i, we have

$$u_i(M_i \cup \widehat{C}_i) = u_i(M_i) + u_i(\widehat{C}_i) \geq \alpha \cdot \mathrm{MMS}_i.$$

- Use an algorithm of Cseh and Fleiner [2020] to allocate cake C in the sense that

$$u_i(C_i) \geq u_i(\widehat{C}_i).$$

Algorithm for Heterogeneous Cake C

- Replace cake C with a homogeneous cake $\widehat{C}$ such that

$$u_i(\widehat{C}) = u_i(C).$$

- Allocate the indivisible goods and homogeneous cake $\widehat{C}$ using the previous algorithm. In other words, for each agent i, we have

$$u_i(M_i \cup \widehat{C}_i) = u_i(M_i) + u_i(\widehat{C}_i) \geq \alpha \cdot \mathrm{MMS}_i.$$

- Use an algorithm of Cseh and Fleiner [2020] to allocate cake C in the sense that

$$u_i(C_i) \geq u_i(\widehat{C}_i).$$

Wrap-Up

1. Mixed-Goods Model
2. Envy-freeness for Mixed Goods (EFM)
3. Maximin Share (MMS) Guarantee

Resources

- Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D. Procaccia, eds. [2016]. *Handbook of Computational Social Choice*. Cambridge University Press
- Ayumi Igarashi and Warut Suksompong [2019]. *Fair Division of Indivisible Items: Asymptotics and Graph-Theoretic Approaches*. Tutorial presented at IJCAI-19. URL: https://www.comp.nus.edu.sg/~warut/ijcai19-tutorial.html
- Rupert Freeman and Nisarg Shah [2020]. *Recent Advances in Fair Resource Allocation*. Tutorial presented at EC-19, AAAI-20, and AAMAS-20. URL: https://www.cs.toronto.edu/~nisarg/papers/Fair-Division-Tutorial.pdf
- Warut Suksompong [2021]. "Constraints in Fair Division". In: *ACM SIGecom Exchanges* 19.2, pp. 46–61. URL: https://www.sigecom.org/exchanges/volume_19/2/SUKSOMPONG.pdf
- Ayumi Igarashi and Warut Suksompong [2022]. *Constraints in Fair Division*. Tutorial presented at IJCAI-22. URL: https://www.comp.nus.edu.sg/~warut/ijcai22-tutorial.html
- Georgios Amanatidis, Haris Aziz, Georgios Birmpas, Aris Filos-Ratsikas, Bo Li, Hervé Moulin, Alexandros A. Voudouris, and Xiaowei Wu [2022]. *Fair Division of Indivisible Goods: A Survey*. arXiv preprint. URL: https://arxiv.org/abs/2208.08782v1

References I

Alon, Noga (1987). "Splitting Necklaces". In: *Advances in Mathematics* 63.3, pp. 247–253.

Amanatidis, Georgios, Evangelos Markakis, Afshin Nikzad, and Amin Saberi (2017). "Approximation Algorithms for Computing Maximin Share Allocations". In: *TALG* 13.4, 52.

Aziz, Haris and Simon Mackenzie (2016). "A Discrete and Bounded Envy-Free Cake Cutting Protocol for Any Number of Agents". In: *Proc. FOCS*, pp. 416–427.

Barman, Siddharth and Sanath Kumar Krishnamurthy (2020). "Approximation Algorithms for Maximin Fair Division". In: *ACM Transactions on Economics and Computation* 8.1, 5.

Bei, Xiaohui, Zihao Li, Jinyan Liu, Shengxin Liu, and **Xinhang Lu** (2021). "Fair Division of Mixed Divisible and Indivisible Goods". In: *Artificial Intelligence* 293, 103436.

Bei, Xiaohui, Shengxin Liu, **Xinhang Lu**, and Hongao Wang (2021). "Maximin Fairness with Mixed Divisible and Indivisible Goods". In: *Autonomous Agents and Multi-Agent Systems* 35.2, 34.

Bhaskar, Umang, A. R. Sricharan, and Rohit Vaish (2021). "On Approximate Envy-Freeness for Indivisible Chores and Mixed Resources". In: *Proc. APPROX*, 1:1–1:23.

Budish, Eric (2011). "The Combinatorial Assignment Problem: Approximate Competitive Equilibrium from Equal Incomes". In: *Journal of Political Economy* 119.6, pp. 1061–1103.

References II

- Cseh, Ágnes and Tamás Fleiner (2020). "The Complexity of Cake Cutting with Unequal Shares". In: *ACM Transactions on Algorithms* 16.3, 29.
- Feige, Uriel, Ariel Sapir, and Laliv Tauber (2021). "A Tight Negative Example for MMS Fair Allocations". In: *Proc. WINE*, pp. 355–372.
- Garg, Jugal, Peter McGlaughlin, and Setareh Taki (2019). "Approximating Maximin Share Allocations". In: *Proc. SOSA*, 20:1–20:11.
- Garg, Jugal and Setareh Taki (2021). "An Improved Approximation Algorithm for Maximin Shares". In: *Artificial Intelligence* 300, 103547.
- Ghodsi, Mohammad, MohammadTaghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Hadi Yami (2021). "Fair Allocation of Indivisible Goods: Improvement". In: *Mathematics of Operations Research* 46.3, pp. 1038–1053.
- Kurokawa, David, Ariel D. Procaccia, and Junxing Wang (2018). "Fair Enough: Guaranteeing Approximate Maximin Shares". In: *Journal of the ACM* 65.2, 8.
- Lipton, Richard J., Evangelos Markakis, Elchanan Mossel, and Amin Saberi (2004). "On Approximately Fair Allocations of Indivisible Goods". In: *Proc. EC*, pp. 125–131.

Thank You!