Introduction to Fair Division

Computational Social Choice

Aditi Sethia
Post-Doctoral Fellow
CSA, IISc

December 10, 2024

Outline

- What is *Social Choice*?
  - Making a $\underbrace{\textit{collective decision}}_{\textit{society}}$ based on the $\underbrace{\textit{individual preferences}}_{\textit{choices}}$.

Outline

- What is *Social Choice*?
  - Making a $\underbrace{\textit{collective decision}}_{\textit{society}}$ based on the $\underbrace{\textit{individual preferences}}_{\textit{choices}}$.
- What properties should a 'good' decision satisfy?

Outline

- What is *Social Choice*?
  - Making a $\underbrace{\textit{collective decision}}_{\textit{society}}$ based on the $\underbrace{\textit{individual preferences}}_{\textit{choices}}$.
- What properties should a 'good' decision satisfy?
  - Maximize individual happiness (Fair)
  - Maximize collective happiness (Welfare)

Outline

- What is *Social Choice*?
  - Making a $\underbrace{\textit{collective decision}}_{\textit{society}}$ based on the $\underbrace{\textit{individual preferences}}_{\textit{choices}}$.
- What properties should a 'good' decision satisfy?
  - Maximize individual happiness (Fair)
  - Maximize collective happiness (Welfare)
- Do such 'good' decisions always exist?

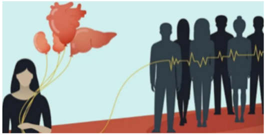
Outline

- What is *Social Choice*?
  - Making a $\underbrace{\textit{collective decision}}_{\textit{society}}$ based on the $\underbrace{\textit{individual preferences}}_{\textit{choices}}$.
- What properties should a ‘good’ decision satisfy?
  - Maximize individual happiness (Fair)
  - Maximize collective happiness (Welfare)
- Do such ‘good’ decisions always exist?
- What *Computation* has to do with it?
  - If such solutions exist, can they be computed efficiently?

Divisible resources

Problem Setting: Allocate Resources!

Indivisible resources



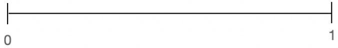
Fair Division of Divisible Items

Cake Division Model

An Allocation $X = (X_1, X_2)$

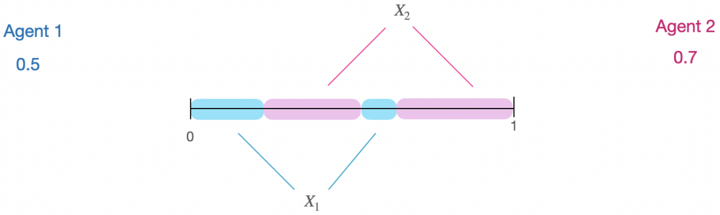
Agent 1

Agent 2



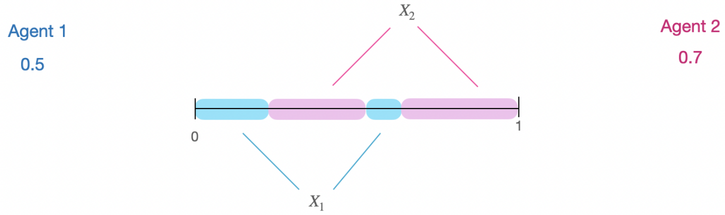
- Valuations functions: $v_i(X) = 1$

Cake Division Model



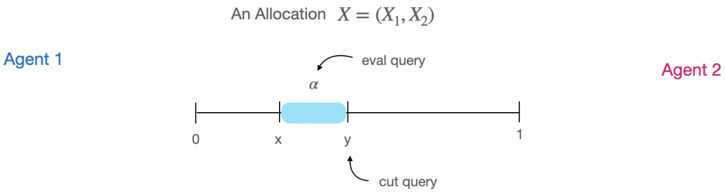
- Valuations functions: $v_i(X) = 1$
- Additivity: $v_1(X_1) = v_1(X_1^1) + v_1(X_1^2)$

Cake Division Model

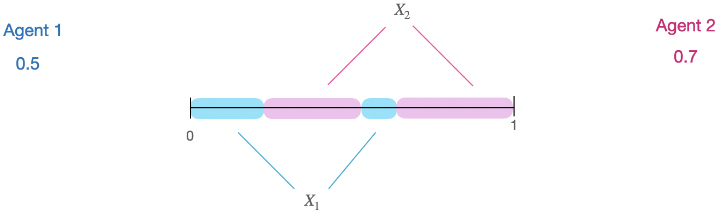


- Valuations functions: $v_i(X) = 1$
- Additivity: $v_1(X_1) = v_1(X_1^1) + v_1(X_1^2)$
- Robertson-Web Model:
  - $eval_i(x, y) = v_i(x, y)$
  - $cut_i(x, \alpha) = y$ such that $v_i(x, y) = \alpha$

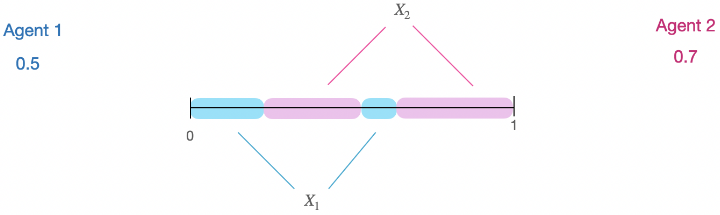
Cake Division Model



- Valuations functions: $v_i(X) = 1$
- Additivity: $v_1(X_1) = v_1(X_1^1) + v_1(X_1^2)$
- Robertson-Web Model:
  - $eval_i(x, y) = v_i(x, y)$
  - $cut_i(x, \alpha) = y$ such that $v_i(x, y) = \alpha$



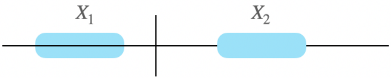
- ([Steinhaus, 1948](#)) Proportionality: $v_i(X_i) \geq \frac{1}{n}$



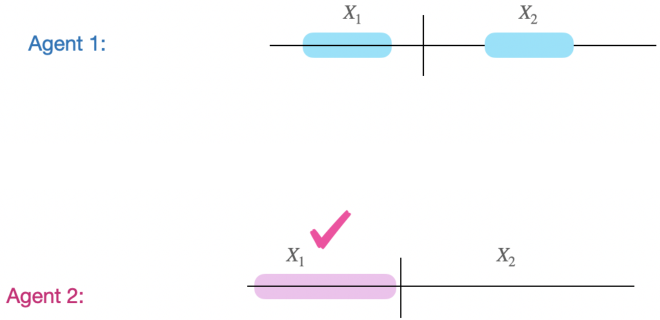
- (Steinhaus, 1948) Proportionality: $v_i(X_i) \geq \frac{1}{n}$
- (Gamow and Stern, 1958; Foley, 1967) Envy-Freeness: $v_i(X_i) \geq v_i(X_j)$

Envy-Free Cake Division for 2 Agents

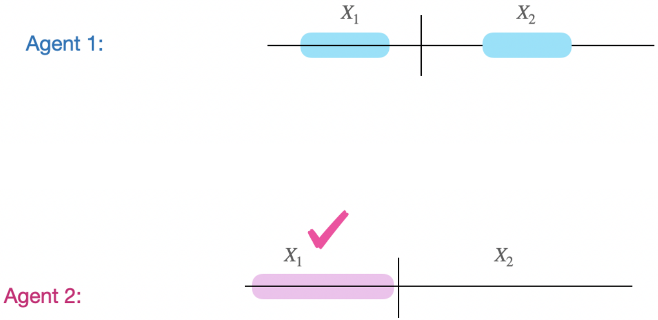
Agent 1:



Envy-Free Cake Division for 2 Agents



Envy-Free Cake Division for 2 Agents



- Envy-Free
- Also Proportional: $v_1(X_1) = \frac{1}{2}$ and $v_2(X_2) \geq \frac{1}{2}$

Proportional Cake Division for n Agents

Dubins-Spanier Procedure

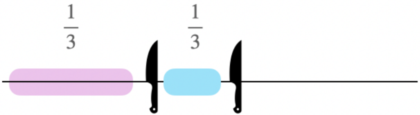
$$v_i(X_i) \geq \frac{1}{n}$$

- A knife moves on the interval $[0, 1]$
- An agent i shouts when the knife reaches a point y such that $v_i([0, y]) = \frac{1}{n}$
- The agent leaves with the piece $[0, y]$
- The process repeats with the remaining agents

Dubins-Spanier Procedure

$$v_i(X_i) \geq \frac{1}{n}$$

- A knife moves on the interval $[0, 1]$
- An agent i shouts when the knife reaches a point y such that $v_i([0, y]) = \frac{1}{n}$
- The agent leaves with the piece $[0, y]$
- The process repeats with the remaining agents



Envy-Free Cake Division for 3 Agents

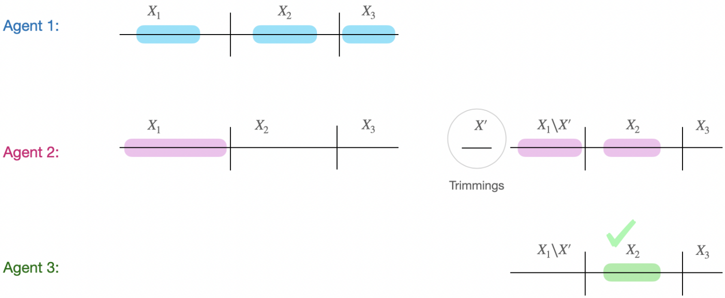
Selfridge-Conway Algorithm (1960's)

- Agents 1, 2, 3 and a Cake $C = [0, 1]$
- Valuations v_1, v_2, v_3

Envy-Free Cake Division for 3 Agents

Selfridge-Conway Algorithm (1960's)

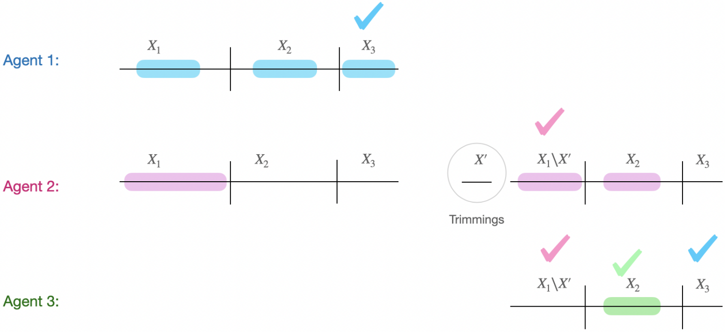
- Agents 1, 2, 3 and a Cake $C = [0, 1]$
- Valuations v_1, v_2, v_3



Envy-Free Cake Division for 3 Agents

Selfridge-Conway Algorithm (1960's)

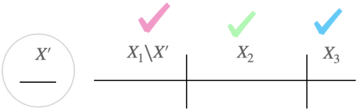
- Agents 1, 2, 3 and a Cake C
- Valuations v_1, v_2, v_3



Envy-Free Cake Division for 3 Agents

Selfridge-Conway Algorithm (1960's)

The Trimmed Piece



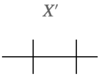
Trimmings

Agent 1:

Agent 2:

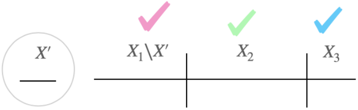
Agent 3:

1,



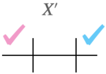
Selfridge-Conway Algorithm (1960's)

The Trimmed Piece

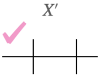


Trimmings

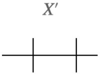
Agent 1: 3.



Agent 2: 2.

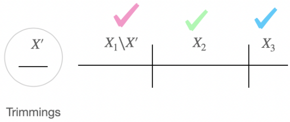


Agent 3: 1,

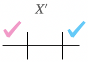


Gets the last left piece

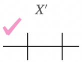
Selfridge-Conway Algorithm (1960's)
The Trimmed Piece



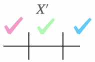
Agent 1: 3.

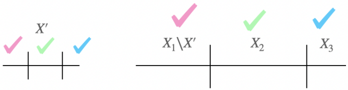


Agent 2: 2.



Agent 3: 1,





Envy-Free Cake Division: The Story

- (Selfridge-Conway, 1960) EF cake division exists for $n = 3$ agents.

Envy-Free Cake Division: The Story

- (Selfridge-Conway, 1960) EF cake division exists for $n = 3$ agents.
- (Brams and Taylor, 1995) An EF cake division exists for n agents and there is a finite but *unbounded* procedure to compute such a division.

Envy-Free Cake Division: The Story

- (Selfridge-Conway, 1960) EF cake division exists for $n = 3$ agents.
- (Brams and Taylor, 1995) An EF cake division exists for n agents and there is a finite but *unbounded* procedure to compute such a division.

Q. Does there exist a bounded procedure to compute an EF cake division for n agents?

Envy-Free Cake Division: The Story

- (Selfridge-Conway, 1960) EF cake division exists for $n = 3$ agents.
- (Brams and Taylor, 1995) An EF cake division exists for n agents and there is a finite but *unbounded* procedure to compute such a division.

  Q. Does there exist a bounded procedure to compute an EF cake division for n agents?

- (Lindner and Rothe 2015) “Despite intense efforts over decades, up to this date no one has succeeded in finding a finite bounded cake-cutting protocol that guarantees envy-freeness for any number of players"

Envy-Free Cake Division: The Story

- (Selfridge-Conway, 1960) EF cake division exists for $n = 3$ agents.
- (Brams and Taylor, 1995) An EF cake division exists for n agents and there is a finite but *unbounded* procedure to compute such a division.

> Q. Does there exist a bounded procedure to compute an EF cake division for n agents?

- (Lindner and Rothe 2015) "Despite intense efforts over decades, up to this date no one has succeeded in finding a finite bounded cake-cutting protocol that guarantees envy-freeness for any number of players"
- (Aziz and Mackenzie 2016) There is a bounded finite protocol that guarantees EF cake division for any number of agents with at most $n^{n^{n^{n^{n^{n}}}}}$ queries.

Envy-Free Cake Division: The Story

- (Selfridge-Conway, 1960) EF cake division exists for $n = 3$ agents.
- (Brams and Taylor, 1995) An EF cake division exists for n agents and there is a finite but *unbounded* procedure to compute such a division.

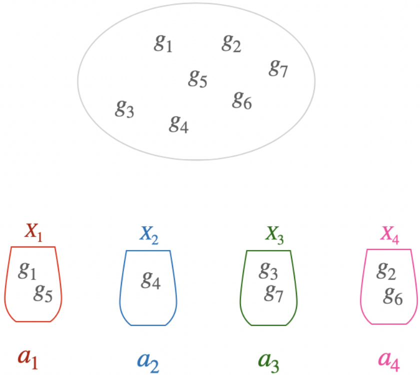
Q. Does there exist a bounded procedure to compute an EF cake division for n agents?

- (Lindner and Rothe 2015) "Despite intense efforts over decades, up to this date no one has succeeded in finding a finite bounded cake-cutting protocol that guarantees envy-freeness for any number of players"
- (Aziz and Mackenzie 2016) There is a bounded finite protocol that guarantees EF cake division for any number of agents with at most $n^{n^{n^{n^{n^{n}}}}}$ queries.
- (Proccacia, 2009) Any EF protocol requires at least $\Omega(n^2)$ queries.

Fair Division of Indivisible Items

Dividing the Indivisible!

- Set of Items $\{g_1, g_2, \ldots g_m\}$
- Set of Agents $\{a_1, a_2, \ldots a_n\}$



Dividing the Indivisible!

	g_1	g_2	g_3	g_4	g_5
a_1	5	10	2	3	10
a_2	10	5	2	4	12

Dividing the Indivisible!

	g_1	g_2	g_3	g_4	g_5
a_1	5	10	2	3	10
a_2	10	5	2	4	12

$$v_1(X_1) = v_1(\{g_1, g_2\}) = v_1(g_1) + v_1(g_2) = 5 + 10 = 15$$

Dividing the Indivisible!

EF does not exist for indivisible items!

Deciding if there is an EF allocation is NP-Hard even for binary valuations!

Dividing the Indivisible

- (Budish 2011) EF upto *one* item (EF1): $v_i(X_i) \geq v_i(X_j \setminus \{g\})$

Dividing the Indivisible

- (Budish 2011) EF upto *one* item (EF1): $v_i(X_i) \geq v_i(X_j \setminus \{g\})$
- (Caragiannis et al. 2016) EF upto *any* item (EFX): $v_i(X_i) \geq v_i(X_j \setminus \{g\}) \; \forall g \in X_j$

Dividing the Indivisible

- (Budish 2011) EF upto *one* item (EF1): $v_i(X_i) \geq v_i(X_j \setminus \{g\})$
- (Caragiannis et al. 2016) EF upto *any* item (EFX): $v_i(X_i) \geq v_i(X_j \setminus \{g\}) \; \forall g \in X_j$

	g_1	g_2	g_3
a_1	$\boxed{1}$	1	2
a_2	1	$\boxed{1}$	$\boxed{2}$

- (Budish 2011) EF upto *one* item (EF1): $v_i(X_i) \geq v_i(X_j \setminus \{g\})$
- (Caragiannis et al. 2016) EF upto *any* item (EFX): $v_i(X_i) \geq v_i(X_j \setminus \{g\})\ \forall g \in X_j$

	g_1	g_2	g_3
a_1	$\boxed{1}$	1	2
a_2	1	$\boxed{1}$	$\boxed{2}$

EF1 but not EFX

- (Budish 2011) EF upto *one* item (EF1): $v_i(X_i) \geq v_i(X_j \setminus \{g\})$
- (Caragiannis et al. 2016) EF upto *any* item (EFX): $v_i(X_i) \geq v_i(X_j \setminus \{g\}) \; \forall g \in X_j$

	g_1	g_2	g_3
a_1	$\boxed{1}$	1	2
a_2	1	$\boxed{1}$	$\boxed{2}$

EF1 but not EFX

EF $\Rightarrow$ EFX $\Rightarrow$ EF1

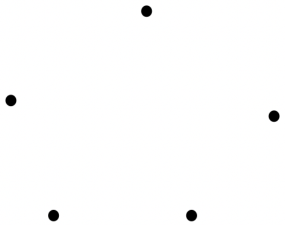
Results At a Glance

- EF1 always exists (even for monotone valuations)
- EFX always exists for 2 agents (Plaut and Roughgarden 2020), 3 agents (Chaudhury et al. 2020), 2 types of agents (Mahara 2023)

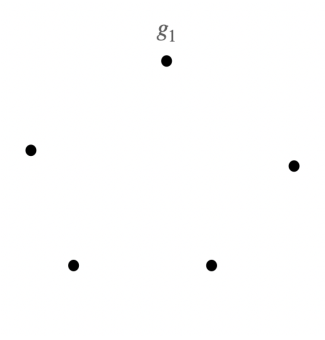
Do EFX Allocations always exists?

EF1 Allocations

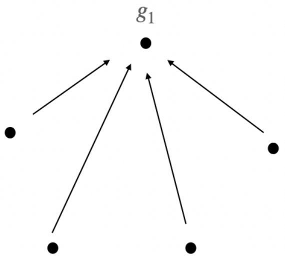
Envy Cycle Elimination (Lipton et al. 2004)



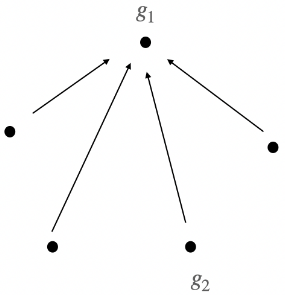
Envy Cycle Elimination



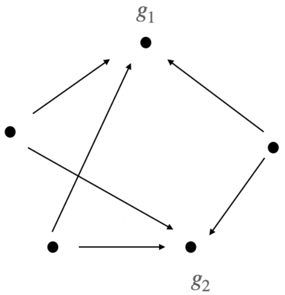
Envy Cycle Elimination



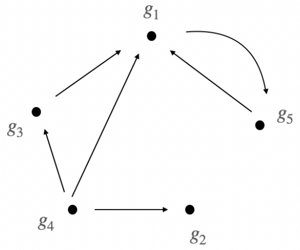
Envy Cycle Elimination



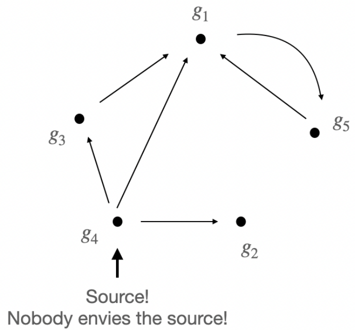
Envy Cycle Elimination



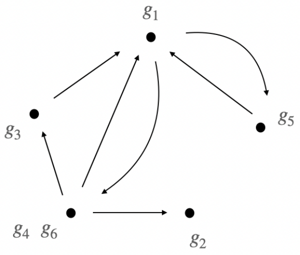
Envy Cycle Elimination



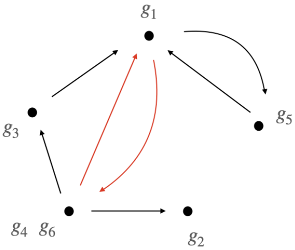
Envy Cycle Elimination



Envy Cycle Elimination

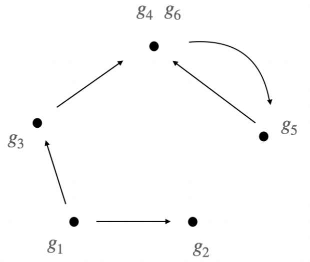


Envy Cycle Elimination



No Source!
But there is an Envy Cycle!

Envy Cycle Elimination



EF1 Allocations

Envy Cycle Elimination
While there is a good g to be allocated:

- Construct Envy Graph of the partial allocation A
- Find a source in the Envy Graph and allocate g to the source
- If there is no source, then eliminate the envy cycles by rotating the bundles on the cycle

The process terminated in polynomial time
The final allocation is EF1

Thank you!