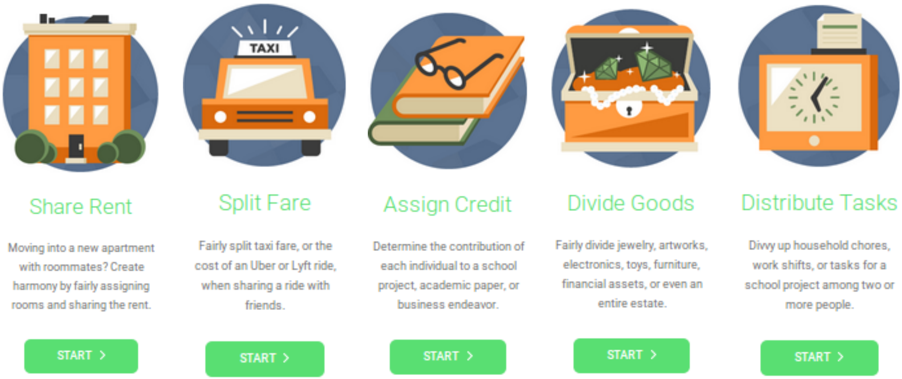
Week 8: Fair Allocation of Indivisible Goods

Instructor: Warut Suksompong

National University of Singapore

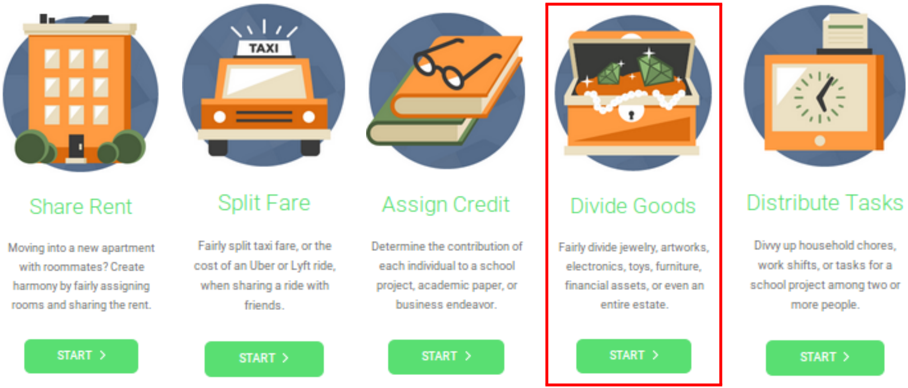
CS4261/5461
Semester 1, 2025

Fair Division



www.spliddit.org

Fair Division



www.spliddit.org

Setting

- Players $N = \{1, 2, \ldots, n\}$
- Indivisible goods $G = \{g_1, g_2, \ldots, g_m\}$
- Player i has value $v_i(g)$ for good g.
- Unless noted otherwise, assume that valuations are additive: $v_i(G') = \sum_{g \in G'} v_i(g)$ for all $G' \subseteq G$
  - Advantage: Ease of elicitation (each agent has m values instead of 2^m)
  - Disadvantage: Does not allow for complements/substitutes
- An allocation is a partition of the goods, $A = (A_1, A_2, \ldots, A_n)$, where bundle A_i is allocated to player i.

Fairness Notions

- When is an allocation fair?
- Proportionality: $v_i(A_i) \geq \frac{1}{n} \cdot v_i(G)$ for all $i \in N$
- Envy-freeness: $v_i(A_i) \geq v_i(A_j)$ for all $i, j \in N$
- **Questions:**
  - Which of the two notions is stronger for $n = 2$?
  - What about for $n \geq 3$?

Fairness Notions

- For $n = 2$, envy-freeness and proportionality are equivalent.
- For $n \geq 3$, envy-freeness is stronger than proportionality.
  - Envy-freeness $\Rightarrow$ proportionality: If $v_i(A_i) \geq v_i(A_j)$ for all $i, j \in N$, then

    $$n \cdot v_i(A_i) \geq v_i(A_1) + \cdots + v_i(A_n) = v_i(G),$$

    so $v_i(A_i) \geq \frac{1}{n} \cdot v_i(G)$.
  - Proportionality $\not\Rightarrow$ envy-freeness: Assume $m = n$. Player 1 has value 1 for every good, while other players have value 0 for all goods. $A_1 = \{g_1\}$, $A_2 = \{g_2, \ldots, g_m\}$, and $A_3, \ldots, A_n$ are empty.

Fair Division

Envy-freeness/proportionality cannot always be satisfied!

Maximum Utilitarian Welfare & EF1

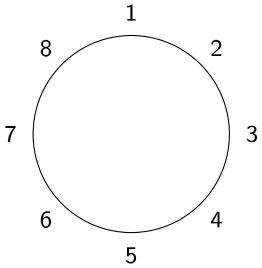
- Maximize the utilitarian welfare, i.e., the sum of the players' utilities.
- This simply means we give each good to a player who has the highest value for it (breaking ties arbitrarily).

Player	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8
1	**10**	**10**	**10**	**10**	**10**	**10**	**10**	3
2	9	9	9	9	9	9	9	**10**

- Envy-freeness up to one good (EF1): Player i may envy player j, but the envy can be eliminated by removing a good from j's bundle.
  - Formally, for any $i, j \in N$, if $A_j \neq \emptyset$, then there exists $g \in A_j$ such that $v_i(A_i) \geq v_i(A_j \setminus \{g\})$.
- Maximizing the utilitarian welfare may fail EF1.

Round-Robin Algorithm

- EF1 can be satisfied by the round-robin algorithm: Let the players take turns choosing their favorite good from the remaining goods, in the order $1, 2, \ldots, n, 1, 2, \ldots, n, 1, 2, \ldots$ until the goods run out.



Round-Robin Algorithm

- EF1 can be satisfied by the round-robin algorithm: Let the players take turns choosing their favorite good from the remaining goods, in the order $1, 2, \ldots, n, 1, 2, \ldots, n, 1, 2, \ldots$ until the goods run out.

- **Proof:**
  - If i is ahead of j in the round-robin ordering, then in every "round", i does not envy j.
  - If i is behind j in the ordering, we consider the first round to start with i's first pick. Then i does not envy j up to j's first good.

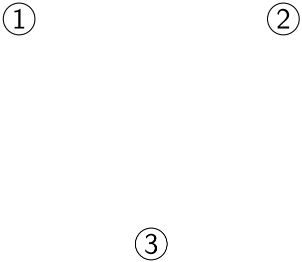
- **Bonus:** The resulting allocation is always balanced.

Envy-Cycle Elimination Algorithm

- General monotonic valuations: $v_i(S) \leq v_i(T)$ for any $S \subseteq T \subseteq G$
- We can still get an EF1 allocation using the envy-cycle elimination algorithm.
  1. Allocate one good at a time in an arbitrary order.
  2. Maintain an envy graph with the players as its vertices, and a directed edge $i \rightarrow j$ if i envies j with respect to the current (partial) allocation.
  3. At each step, the next good is allocated to a player with no incoming edge. Any cycle that arises is eliminated by giving j's bundle to i for any edge $i \rightarrow j$ in the cycle.

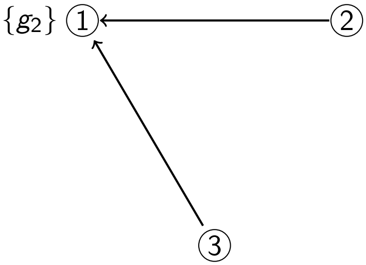
Envy-Cycle Elimination Algorithm

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9



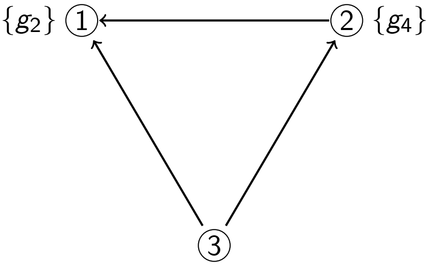
Envy-Cycle Elimination Algorithm

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9



Envy-Cycle Elimination Algorithm

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9



Envy-Cycle Elimination Algorithm

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9

$$\begin{tikzcd}
\{g_2\}\ (1) \arrow[dr] & & (2)\ \{g_4\} \arrow[ll] \\
& (3)\ \{g_1\} \arrow[ur] &
\end{tikzcd}$$

Envy-Cycle Elimination Algorithm

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9

$\{g_1\}$ 1

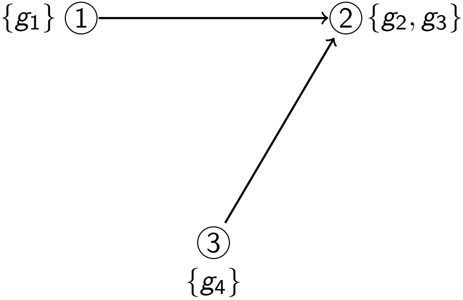
2 $\{g_2\}$

3

$\{g_4\}$

Envy-Cycle Elimination Algorithm

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9



Envy-Cycle Elimination Algorithm

- **Claim 1:** At each step, the procedure of eliminating cycles must end.
  - Each time we eliminate a cycle, the utilitarian welfare increases.
  - Another way to see this is that the number of envy edges decreases.
- **Claim 2:** When the procedure of eliminating cycles ends, there is an unenvied player (i.e., a source in the envy graph).
  - Proof by contradiction. If a is envied by b, b is envied by c, $\dots$, then we will get a cycle in the envy graph.
- **Claim 3:** At each step, the partial allocation is EF1.
  - We allocate a good to an unenvied player, so any envy towards that player is by at most one good (i.e., the newly allocated good).

Maximum Nash Welfare

- The Nash welfare of an allocation is the product of the players' utilities: $\prod_{i=1}^{n} v_i(M_i)$.

Theorem

An allocation that maximizes the Nash welfare, called a maximum Nash welfare (MNW) allocation, is EF1.

- If $\text{MNW} = 0$, maximize the number of players with positive utility, then maximize Nash welfare among these players.
- **Proof sketch:**
  - Suppose for contradiction that player i envies player j even after removing any good from j's bundle.
  - Consider a good g in j's bundle that minimizes the ratio $v_j(g)/v_i(g)$.
  - Moving g to i's bundle increases the Nash welfare.
- **Bonus:** The resulting allocation is always Pareto optimal: we cannot make some player better off without making another player worse off.

Maximum Nash Welfare

Algorithm Overview

We assume that the value a participant derives from a bundle of goods is the sum of points the participant assigns to individual goods in the bundle. Our algorithm then finds the division of goods into bundles that maximizes the product of values derived by participants. The optimization problem is formulated as a mixed integer linear program. This division is guaranteed to be envy free up to one good and efficient, and provably satisfies other approximate fairness guarantees.

www.spliddit.org/apps/goods

EFX

- Envy-freeness up to one good (EF1):
  - For any $i, j \in N$, if $A_j \neq \emptyset$, there exists $g \in A_j$ such that $v_i(A_i) \geq v_i(A_j \setminus \{g\})$.
  - "If some good g is removed from your bundle, I no longer envy you."
- Envy-freeness up to any good (EFX):
  - For any $i, j \in N$ and any $g \in A_j$, we have $v_i(A_i) \geq v_i(A_j \setminus \{g\})$.
  - "If any good g is removed from your bundle, I no longer envy you."
- Envy-freeness $\Rightarrow$ EFX $\Rightarrow$ EF1

EF1 vs EFX

- **Example:**

Player	g_1	g_2	g_3
1	2	2	3
2	4	4	1

- The allocation with $A_1 = \{g_1\}$ and $A_2 = \{g_2, g_3\}$ is EF1 but not EFX.
- The allocation with $A_1 = \{g_3\}$ and $A_2 = \{g_1, g_2\}$ is EF1 and EFX.

- The output of round-robin, envy-cycle elimination, and maximum Nash welfare may not satisfy EFX.
- **Question:** Does there always exist an EFX allocation?

EFX

- An EFX allocation always exists when $n = 2$.
  - "Cut-and-choose"
  - The first player divides the goods into two bundles that are as equal as possible in her view.
  - The second player chooses the bundle that he prefers.
  - The first player will be EFX, and the second player will be envy-free.
- Existence is guaranteed for $n = 3$ (much more complicated proof)
- For $n \geq 4$, this question is open!

Maximin Share

- A relaxation of proportionality is the maximin share (MMS).
- A player's MMS can be computed by performing the following thought experiment: The player divides the goods into n bundles so as to maximize the value of the minimum-value bundle.

Player	g_1	g_2	g_3	g_4
1	11	10	5	1
2	1	6	1	2
3	8	7	4	9

- Player 1's partition: $\{g_1\}, \{g_2\}, \{g_3, g_4\} \Rightarrow \text{MMS}_1 = 6$
- Player 2's partition: $\{g_1, g_3\}, \{g_2\}, \{g_4\} \Rightarrow \text{MMS}_2 = 2$
- Player 3's partition: $\{g_1\}, \{g_2, g_3\}, \{g_4\} \Rightarrow \text{MMS}_3 = 8$

Maximin Share

- Maximin share is a relaxation of proportionality: $\mathrm{MMS}_i \leq \frac{v_i(G)}{n}$
- An allocation that gives every player at least his/her maximin share always exists when there are two players ...
- ... but not when there are at least three players!
- However, for any number of players, we can always give every player more than $\frac{3}{4}$ of his/her maximin share.

Maximin Share

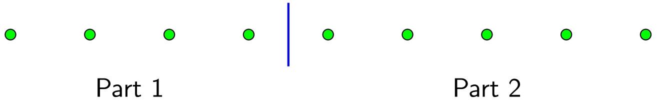
- An allocation that gives every player at least his/her maximin share always exists when there are two players.
- **Proof:** Cut-and-choose strikes again.
  - Alice divides the goods into two parts that are of as equal value as possible in her view.
  - Either part yields value at least Alice's MMS to her.
  - Bob chooses a part that he prefers.
  - Bob is envy-free (and therefore proportional & gets at least his MMS)

Query Complexity

- How many times do we need to query the players?
  - With each query, the algorithm can find out a certain player's value for a certain set of goods.
- Especially relevant when valuations are not additive.
- The envy-cycle elimination algorithm can be implemented using $O(nm)$ queries, even with monotonic valuations.
  - It suffices to query the value of each agent for the n bundles in each partial allocation in order to construct the envy graph.
  - Since there are m partial allocations, the number of queries is $O(nm)$.

Query Complexity

- For two agents with monotonic valuations, $O(\log m)$ queries suffice!
  - Arrange the goods on a line, and implement cut-and-choose.
  - The first agent can find an EF1 cut using binary search.



- The same technique cannot be used for EFX, as there may not exist an EFX partition on a line.
  - **Example:** $m = 3$ and the values are $1, 2, 1$.

Query Complexity

- Any deterministic EF1 algorithm needs $\Omega(\log m)$ queries.
  - Holds even for identical additive valuations where each player has value 1 for two goods and 0 for the rest.
  - In any EF1 allocation, the two valuable goods must be separated.

- Proof by adversary argument.
  - Let $G' = G$ initially.
  - Suppose the algorithm queries $v(H)$.
  - If $|G' \cap H| \geq \frac{|G'|}{2}$, the adversary answers 2 and replaces G' by $G' \cap H$.
  - Otherwise, the adversary answers 0 and replaces G' by $G' \setminus H$.
  - Since $|G'| = m$ at the beginning and decreases by a factor of at most 2 after each query, $\Omega(\log m)$ queries are needed.

- Any deterministic EFX algorithm needs a number of queries exponential in m.