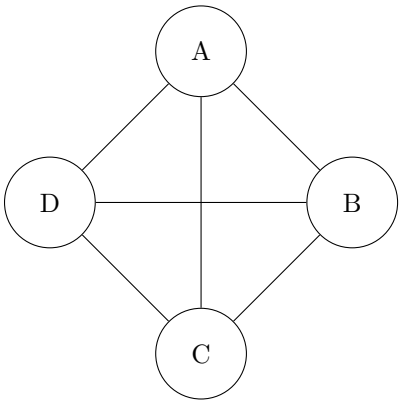
CS 6002: Selected Areas of Mechanism Design **Jan-Apr 2025**

Lecture 7: The Basics of Fair Division

Lecturer: Swaprava Nath *Scribe(s): Dhvanil Gheewala,Akshara*

7.1 Stable Roommate Problem

Unlike stable matching problems, the stable roommate problem has only one pool of agents. Each agent has a preference list over all other agents. The goal is to find a stable matching. It is not always be possible to find a stable matching. There exists a polynomial time algorithm (Irving's algorithm) to find a stable matching that returns a stable matching if one exists and returns that no stable matching exists otherwise.



Preference list of A : B C D

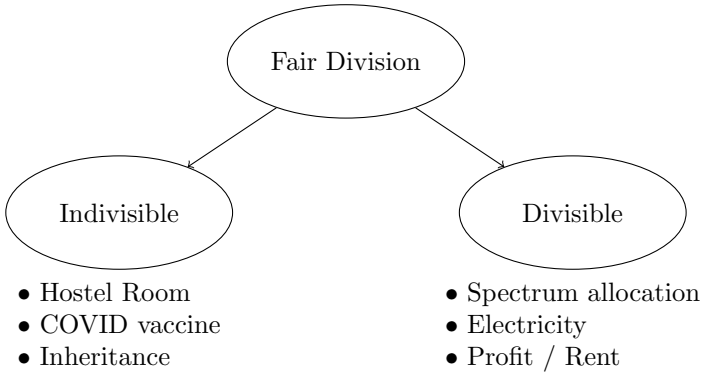
Preference list of B : C A D

Preference list of C : A B D

Preference list of D : B A C

Figure 7.1: Stable Roommate Problem

7.2 Fair Division



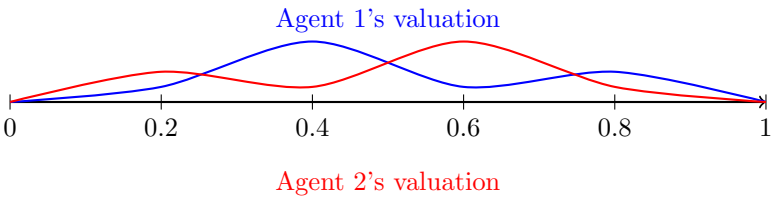


Figure 7.2: Different valuation distributions for two agents over the cake interval $[0, 1]$. The area under each curve represents how each agent values different parts of the cake.

7.2.1 Divisible resource allocation : Cake cutting

- **Heterogenous** : Different pieces of the cake may have different values for an agent
- **Divisible** : The cake can be divided into infinitely many pieces i.e. arbitrary fractional division
- **Non-identical preferences** : Different agents have different preferences over different pieces of the cake
- **Good** : the valuation of each part of cake is greater than or equal to 0 for all agents

$$v_i(A) \geq 0 \quad \forall i \in N, \forall \text{ agents}$$

This is in contrast to chore division where the valuation of each chore is less than 0 for all agents

This can be visualised on a number line from 0 to 1 where each agent has its own distribution of value over the cake.

A piece of cake is a finite union of disjoint subintervals of $[0, 1]$

$$S_i = I_{i_1} \cup I_{i_2} \ldots I_{i_{k_i}} \subseteq [0, 1]$$

Valuation function

Agent preferences valuation function : A valuation function v_i assigns a non-negative real number to any piece

Properties

1. **Additivity**:
$$v_i(A \cup B) = v_i(A) + v_i(B) \quad \forall A, B \subseteq [0, 1] \text{ such that } A \cap B = \emptyset$$
2. **Divisibility**:
$$\forall X \subseteq [0, 1] \text{ and } \forall \lambda \in [0, 1], \exists Y \subseteq X \text{ such that } v_i(Y) = \lambda v_i(X)$$
This property rules out atomic valuation functions.
Note : The cake is a good : Larger pieces are always better than smaller pieces and thus trimming is possible

3. **Normalisation**: The total valuation of the cake is 1 for all the agents

$$v_i([0,1]) = 1 \forall i \in N$$

Allocation : A partition of the cake into disjoint pieces $A_1, A_2 \dots A_N$ such that A_i is allocated to agent i.

$$A_1 \cup A_2 \cdots \cup A_N = [0,1]$$

Note : Disposal of any piece does not put anyone better off

7.2.2 Desirable Fairness Ideas

- Proportionality : An allocation is called proportional if each agent gets at least 1/N of the cake

$$v_i(A_i) \geq \frac{1}{N} \quad \forall i \in N$$

  This is worth noting that this is a personal notion and is not dependent on the allocation of other agents.

- Envy-freeness : An allocation is called envy-free if no agent prefers the piece of another agent over his own piece

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

  It should be noted that this notion of fairness is dependent on the allocation of other agents.

Theorem 7.1. *Envy freeness implies proportionality*

Proof. According to the definition of Envy-Freeness (EF),

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

Summing over all allocations A_j of agents j, we get:

$$\sum_{j=1}^{N} v_i(A_i) \geq \sum_{j=1}^{N} v_i(A_j) \quad \forall \text{ agents } i \in N$$

Simplifying the left-hand side, we obtain:

$$N \cdot v_i(A_i) \geq \sum_{j=1}^{N} v_i(A_j) \quad \forall \text{ agents } i \in N$$

By normalization: (i.e., $\sum_{j=1}^{N} v_i(A_j) = 1$)

$$N \cdot v_i(A_i) \geq 1 \quad \forall \text{ agents } i \in N$$

Finally, dividing both sides by N, we obtain:

$$v_i(A_i) \geq \frac{1}{N} \quad \forall \text{ agents } i \in N$$

Thus, the allocation is proportional. □

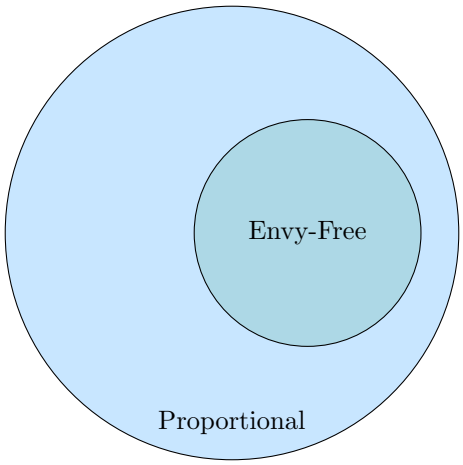


Figure 7.3: Envy-Free implies Proportional

Example 1. *The following example illustrates an allocation which is Proportional but not Envy free:*

	A	B	C
1	$\frac{1}{3}$	$\frac{1}{3}+\epsilon$	$\frac{1}{3}-\epsilon$
2	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
3	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

In the above example, agent 1 is allocated A, 2 is allocated B and 3 is allocated C. The allocation is proportional but not envy free as agent 1 prefers the piece of agent 2 over his own piece. □

7.2.3 Robertson - Webb's Query Model (1998)

Oracle Based Model : The agents can ask different queries to the oracle to get information about the cake. The oracle can answer the following queries:

- Eval Query : The agent can ask the oracle to evaluate the value of a piece of cake

$$Eval_i(x, y) = v_i([x, y]) \quad \text{x} \leq \text{y}$$

- Cut Query : The agent can ask the oracle to cut the cake with specified value

$$Cut_i(x, \alpha) = z \quad \text{such that } v_i([x, z]) = \alpha$$

  This query can return NULL if such a cut is not possible. **Note :** We have made a smoothness assumption.

While counting the number of operations required to do a given allocation, we shall not count the number of cuts made but we shall count the number of queries made to the oracle.

7.2.4 Cake Cutting Algorithms

7.2.4.1 PROP for n Agents : Dubins-Spanier Algorithm (1961)

Steps:

1. **A continuously moving knife:** A knife moves from left to right over the cake, starting at position 0. Every agent is given a buzzer.

2. **Claiming the share:** Each participant continuously evaluates the portion of the cake from the starting point up to the knife's position. When a participant values the piece as at least $\frac{1}{n}$ of the total cake, they hit the buzzer.

3. **Remove the Claimed Piece & Repeat:** The claimed piece is removed, and the process is repeated with the remaining participants and leftover cake.

4. **Continue Until All Participants Receive a Piece:** This guarantees that each of the n participants receives at least $\frac{1}{n}$ of the total cake in their own valuation.

Complexity : The algorithm requires $O(N^2)$ queries

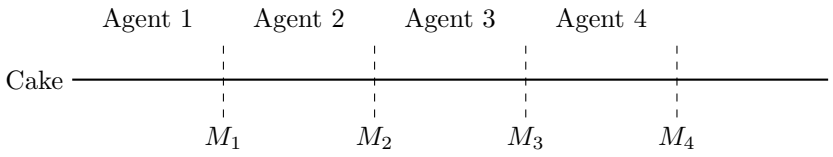
Theorem 7.2. *The Dubins-Spanier algorithm returns a proportional allocation.*

Proof. All agents $\{1, 2, \ldots, N-1\}$ get exactly $\frac{1}{N}$ of the cake as they shout as soon as the left piece evaluates to $\frac{1}{N}$. The last agent has not shouted during the first $N-1$ pieces,

$$\begin{aligned} \implies \quad v_N(a_i) &\leq \frac{1}{N} \quad \forall i \in N-1 \\ \sum_{i=1}^{N-1} v_N(a_i) &\leq N-1 \cdot \frac{1}{N} = 1 \\ v_N(a_N) &\geq 1 - \sum_{i=1}^{N-1} v_N(a_i) \\ &\geq 1 - (N-1) \cdot \frac{1}{N} = \frac{1}{N} \end{aligned}$$

Hence, agent N will also receive a piece worth greater than equal to $\frac{1}{N}$ of the cake. □

7.2.4.2 Recursive Cake Cutting Algorithm (Even-Paz 1984)



Algorithm : (Assume 2^n agents for simplicity)

Base Case : $n = 2$ requires 2 queries

1. Given piece $[x, y]$ each agent marks z_i such that
$$v_i([x, z_i]) = \frac{1}{2} \cdot v_i([x, y])$$
2. Let z^* be the the $\frac{n}{2}^{th}$ from the left
3. Recurse on $[x, z^*]$ with the left $\frac{n}{2}^{th}$ agents and $[z^*, y]$ with remaining agents

Theorem 7.3. *The Even-Paz cutting algorithm returns a proportional allocation*

Proof.
- At stage zero , each agent values the cake 1
- At each subsequent stage, the agents who share a piece $[x, y]$ values it atleast $\frac{1}{2}v_i([x, y])$
- Hence, if ast stage k each agent has value atleast $\frac{1}{2^k}$ of the cake, then at stage $k + 1$ each agents has value atleast $\frac{1}{2^{k+1}}$ of the cake
- The binary tree of division has depth $\log N$ and hence the algorithm is PROP with $O(N \log N)$ queries

□

The time complexity of the Even-Paz Divide-and-Conquer Algorithm is $O(n \log n)$. This is because:

- At each stage, the participant calls a cut-query. So, there are total n cut-queries.
- There will be $\log n$ number of rounds (As every time the piece is getting halved).

Thus, the total time complexity is $O(n \log n)$.

Claim 7.4. *If the number of agents is n, which is not a power of 2, we need to ask every agent to make a cut at $\frac{n-1}{n}$. We have verified it for $n = 3$ in the class. It will be interesting to verify or refute this statement.*

Questions to think on-

- What if the valuation is not normalized?
- Can we improve the complexity of the cake-cutting algorithms?
- How can we work if **Free-Disposability** (Throwing away some cake) is allowed? How can we ensure efficiency?