CS177/ECON147: Algorithmic Economics, Spring 2025
http://eml.berkeley.edu/~fechenique/algoecon/.
Instructor: Federico Echenique, UC Berkeley.

Lecture 6, 02/05/2025. Scribed by Joon Kim

1. Envy-Freeness is Stronger than Proportionality

Lemma 1. *If a partition is envy-free, then it is proportional.*

Proof. Let the partition $P_1, ..., P_n$ be envy-free. Then $\forall i, j \in [n], U_i(P_i) \geq U_i(P_j)$ by definition. The inequality below follows:

$$nU_i(P_i) = \sum_{j=1}^{n} U_i(P_i) \geq \sum_{j=1}^{n} U_i(P_j) = U_i(\bigcup_{j=i}^{n} P_j) = U_i(X) = 1.$$

The first equality holds because the internal term in the summation is a constant with respect to j, and the inequality uses the pairwise inequality of every $U_i(P_i) \geq U_i(P_j)$. Finally, considering the first and last terms,

$$nU_i(P_i) \geq 1 \implies U_i(P_i) \geq 1/n.$$

Thus, the partition is proportional. □

However, a division may be proportional but not envy-free. The following instance is a counterexample.

Example 2. *Let $A = \{1, 2, 3\}$, with utility functions defined as:*

$$U_1(x) = \begin{cases} 3 & x \in [0, 1/3], \\ 0 & x > 1/3, \end{cases} U_2(x) = \begin{cases} 3 & x > 2/3, \\ 0 & x < 2/3, \end{cases} U_3(x) = 1.$$

Consider the partition:

$$P_1 = [0, 1/9], P_2 = (1/9, 8/9), P_3 = [8/9, 1].$$

It is easy to see that the partition is proportional but agents 1 and 2 envy agent 3.

2. Algorithm ensuring Proportionality

Is there a runtime-efficient algorithm that guarantees proportionality?

Dubins-Spanier Algorithm

Imagine the cake as a straight line. Move the knife continuously across the cake from left to right. Whenever someone values the piece to the left of the knife as at least

$1/n$, they shout stop and receive the piece to the left. Repeat until the whole cake is given away.

This works intuitively because a person would have shouted before a certain piece was removed if they thought it was worth at least $1/n$.

Lemma 3. *Dubins-Spanier ensures proportionality.*

Proof. For $n = 2$, proportionality is obvious since the agent who calls stop gets utility $1/2$, and the other agent did not deem the piece up until now worth more than $1/2$, ensuring that the leftover piece is worth at least $1/2$ to them.

The general case can be proved by induction. $n = 2$ serves as a base case.

Induction Hypothesis: Suppose that Dubins-Spanier produces a valid proportional partition for $n - 1$ agents.

For n agents: The agent who calls stop gets utility at least $1/n$.

For the remaining agents, the remaining cake is worth at least

$$(1 - 1/n) = (n - 1)/n.$$

With one agent removed, this is now an instance of $n - 1$ agents. By induction hypothesis, each remaining agent gets at least

$$(n - 1)/n * (1/(n - 1)) = 1/n$$

of the cake with respect to their own utility. Then, every agent gets at least $1/n$ of the cake with respect to their utility. Induction is complete. □

Proving the runtime efficiency of Dubins-Spanier is nontrivial. We now introduce the concept of query complexity to tackle its analysis.

3. Query Complexity

In the case of algorithms that have many moving parts, it is easier to abstract away subroutines that are repeated throughout the execution. Such subroutines will be called *oracles*, in the sense that we can consult them to receive some information needed to proceed in the algorithm. Given the complexity of each oracle, the analysis of the entire algorithm will boil down to counting the number of times the oracles are called.

We define two oracles for Dubins-Spanier:

(1) Eval($[a, b]$) returns $u_i([a, b])$.
(2) Cut(a, α)) returns b, such that $u_i([a, b]) = \alpha$.