# CSC2556

## Lecture 8

## Fair Division 1: Cake-Cutting

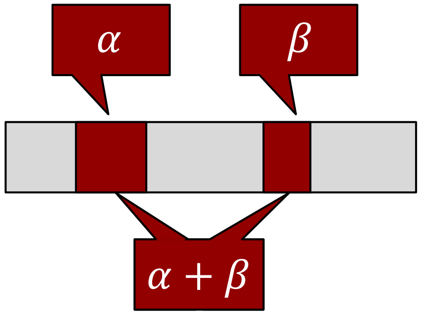
## Cake-Cutting

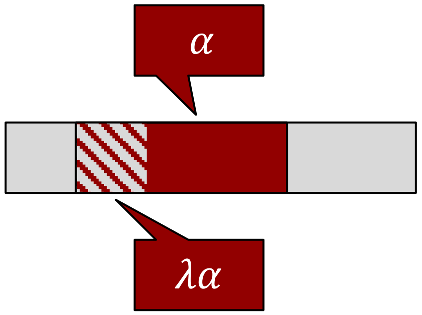
# Cake-Cutting

- A heterogeneous divisible good
  - Heterogeneous = same part may be valued differently by different agents
  - Divisible = can be divided between agents
- Cake $C = [0,1]$
  - Almost without loss of generality
- Agents $N = \{1, \dots, n\}$
- Piece of cake $X \subseteq [0,1]$ = finite union of disjoint intervals
- Allocation $A = (A_1, \dots, A_n)$
  - Partition of the cake where each $A_i$ is a piece of the cake

# Agent Valuations

- Valuation of agent $i$ is given by an integrable value density function $f_i: [0,1] \to \mathbb{R}_+$
  - Her value for a piece of cake $X$ is $V_i(X) = \int_{x \in X} f_i(x) dx$
- Two key properties
  - Additive: For $X \cap Y = \emptyset$, $V_i(X) + V_i(Y) = V_i(X \cup Y)$
  - Divisible: $\forall \lambda \in [0,1]$ and $X$, $\exists Y \subseteq X$ s.t. $V_i(Y) = \lambda V_i(X)$
- WLOG
  - Normalized: $V_i([0,1]) = 1$





# Fairness Goals

- What kind of fairness might we want from an allocation $A$?

- Proportionality (Prop):

$$\forall i \in N: \; V_i(A_i) \geq \frac{1}{n}$$

- Envy-Freeness (EF):

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

- Equitability (EQ):

$$\forall i, j \in N: V_i(A_i) = V_j(A_j)$$

Only makes sense with normalization

## Fairness Goals

- Prop: $\forall i \in N: V_i(A_i) \geq 1/n$
- EF: $\forall i, j \in N: V_i(A_i) \geq V_i(A_j)$

- Question:
  What is the relation between proportionality and EF?

1. Prop $\Rightarrow$ EF
2. EF $\Rightarrow$ Prop
3. Equivalent
4. Incomparable

# Cut-and-Choose

- Algorithm for $n = 2$ agents

- Agent 1 divides the cake into two pieces $X, Y$ s.t.
$$V_1(X) = V_1(Y) = 1/2$$

- Agent 2 chooses the piece she prefers.

- This is EF and therefore proportional.
  - Why?

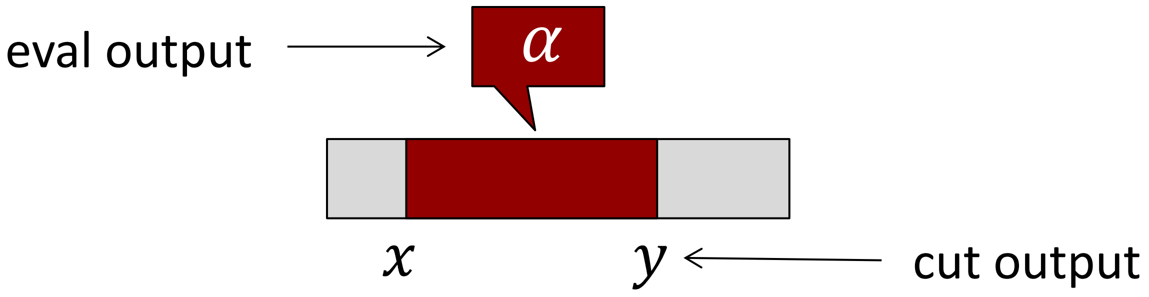
## Measuring Complexity

- Running time does not make sense
  - Typically, we measure the running time as a function of the length of input encoded in binary
  - Our input consists of functions $V_i$, which requires infinitely many bits to encode
  - We want running time just as a function of $n$.

- Query models make sense
  - Allow specific types of queries to agents' valuation functions
  - Measure the number of queries that need to be made in order to find an allocation satisfying the given properties

# Robertson-Webb Model

- Two types of queries to an agent's valuation function $V_i$
  - $\text{Eval}_i(x, y)$ returns $V_i([x, y])$
  - $\text{Cut}_i(x, \alpha)$ returns the smallest $y$ such that $V_i([x, y]) = \alpha$
    - If no such $y$ exists, then it returns 1



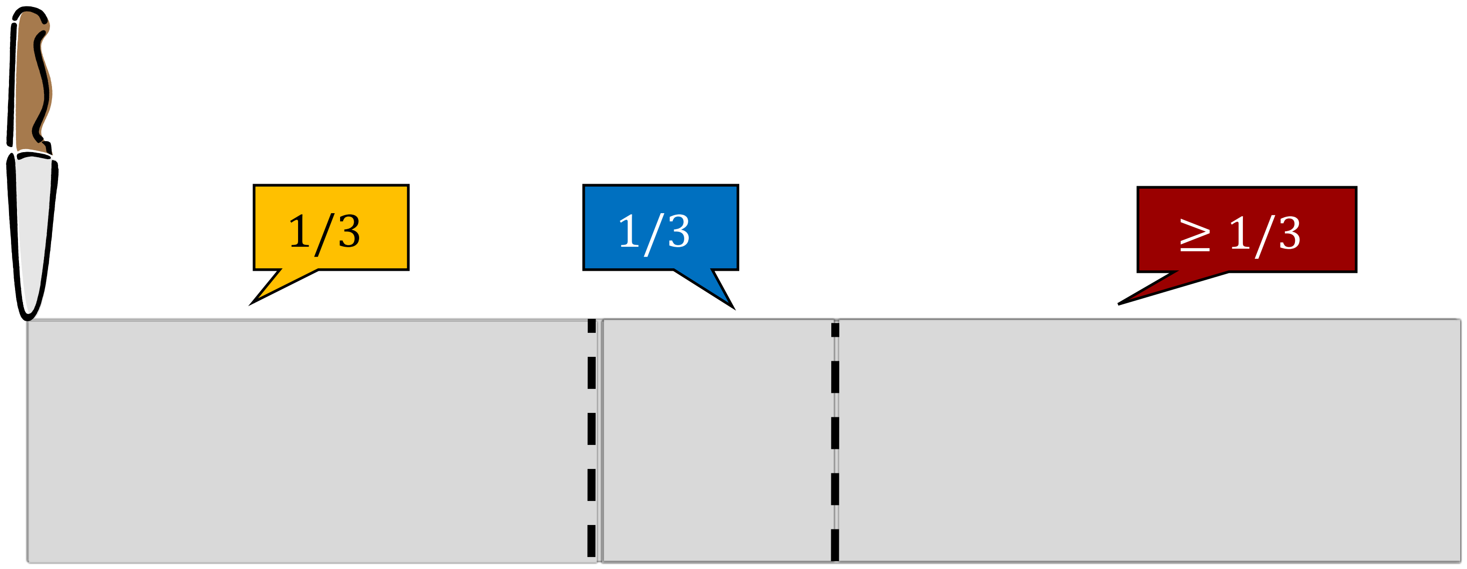
- Question:
  - How many queries are needed to find an EF allocation when $n = 2$?

## DUBINS-SPANIER

- Protocol for finding a proportional allocation for $n$ agents

- Referee starts with a knife at $0$
- Referee continuously moves the knife to the right
- Repeat $n - 1$ times: Whenever the piece to the left of knife is worth $1/n$ to a agent, the agent shouts “stop”, gets the piece, and exits.
- The last agent gets the remaining piece.

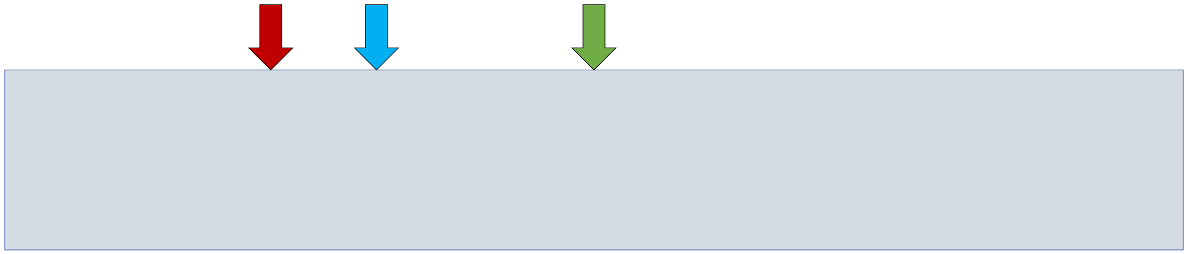
# Dubins-Spanier



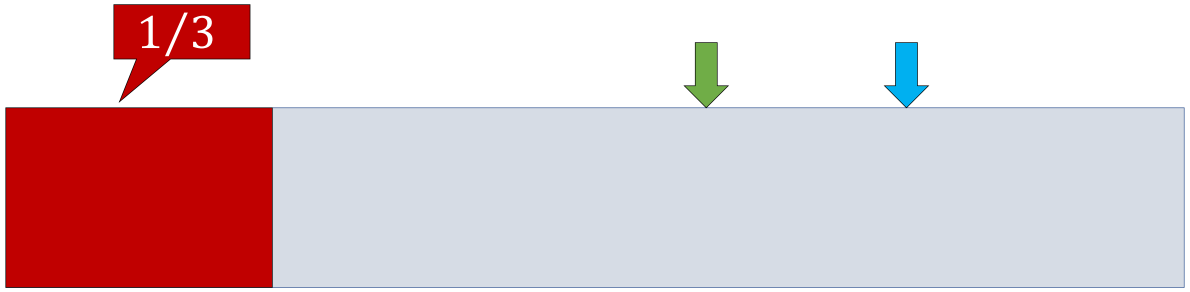
## Dubins-Spanier

- Moving a knife continuously is not really needed.
- At each stage, we can ask each remaining agent a cut query to mark his $1/n$ point in the remaining cake.
- Move the knife to the leftmost mark.

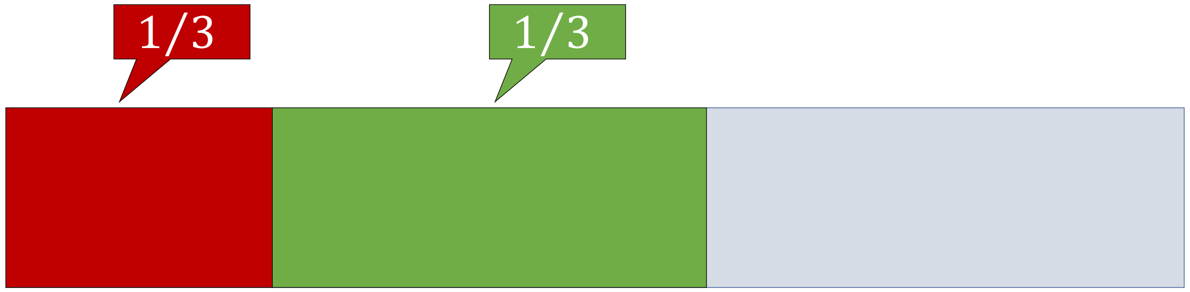
# Dubins-Spanier



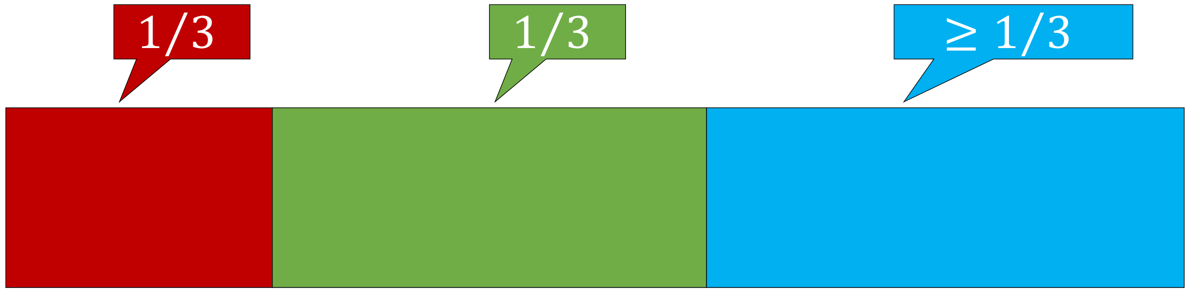
# DUBINS-SPANIER



## DUBINS-SPANIER



# Dubins-Spanier



# Dubins-Spanier

- Question: What is the complexity of the Dubins-Spanier protocol in the Robertson-Webb model?

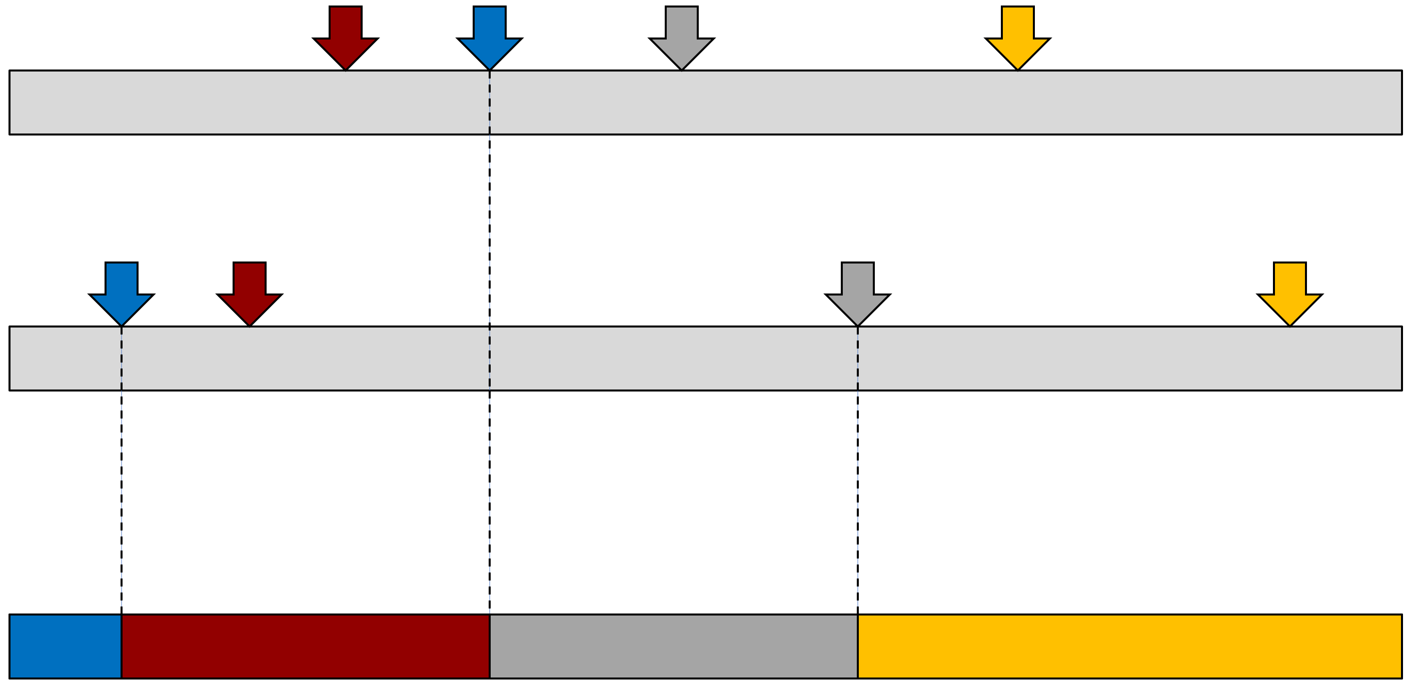
1. $\Theta(n)$
2. $\Theta(n \log n)$
3. $\Theta(n^2)$
4. $\Theta(n^2 \log n)$

# Even-Paz

- Input: Interval $[x, y]$, number of agents $n$
  - Assume $n = 2^k$ for some $k$

- If $n = 1$, give $[x, y]$ to the single agent.
- Otherwise, let each agent $i$ mark $z_i$ s.t.
$$V_i([x, z_i]) = \frac{1}{2} V_i([x, y])$$
- Let $z^*$ be the $n/2$-th mark from the left.
- Recurse on $[x, z^*]$ with the left $n/2$ agents and on $[z^*, y]$ with the right $n/2$ agents.

## Even-Paz



# EVEN-PAZ

- Theorem: EVEN-PAZ returns a Prop allocation.
- Proof:
  - Inductive proof. We want to prove that if agent $i$ is allocated piece $A_i$ when $[x, y]$ is divided between $n$ agents, $V_i(A_i) \geq (1/n)V_i([x, y])$
    - Then Prop follows because initially $V_i([x, y]) = V_i([0,1]) = 1$
  - Base case: $n = 1$ is trivial.
  - Suppose it holds for $n = 2^{k-1}$. We prove for $n = 2^k$.
  - Take the $2^{k-1}$ left agents.
    - Every left agent $i$ has $V_i([x, z^*]) \geq (1/2)\, V_i([x, y])$
    - If it gets $A_i$, by induction, $V_i(A_i) \geq \frac{1}{2^{k-1}}\, V_i([x, z^*]) \geq \frac{1}{2^k}\, V_i([x, y])$

# Even-Paz

- Question: What is the complexity of the Even-Paz protocol in the Robertson-Webb model?

1. $\Theta(n)$
2. $\Theta(n \log n)$
3. $\Theta(n^2)$
4. $\Theta(n^2 \log n)$

# Complexity of Proportionality

- Theorem [Edmonds and Pruhs, 2006]: Any proportional protocol needs $\Omega(n \log n)$ operations in the Robertson-Webb model.

- Thus, the EVEN-PAZ protocol is (asymptotically) provably optimal!

## Envy-Freeness?

- "I suppose you are also going to give such cute algorithms for finding envy-free allocations?"

- Bad luck. For $n$-agent EF cake-cutting:
  - [Brams and Taylor, 1995] gave an unbounded EF protocol.
  - [Procaccia 2009] proved $\Omega(n^2)$ lower bound for EF.
  - In 2016, the long-standing major open question of "bounded EF protocol" was resolved!
  - [Aziz and Mackenzie, 2016]: $O\left(n^{n^{n^{n^{n^{n}}}}}\right)$ protocol!
    - Not a typo!

# Other Desiderata

- There are two more properties that we often desire from an allocation.

- Pareto optimality (PO)
  - Notion of efficiency
  - Informally, it says that there should be no “obviously better” allocation

- Strategyproofness (SP)
  - No agent should be able to gain by misreporting her valuation

# Strategyproofness (SP)

- Deterministic mechanisms
  - Strategyproof: No agent should be able to increase her *utility* by misreporting her valuation, irrespective of what other agents report.

- Randomized mechanisms
  - Strategyproof-in-expectation: Replace *utility* with *expected utility* in the above definition.
  - For simplicity, we'll just call this strategyproofness too.

# Strategyproofness (SP)

- Deterministic
  - Bad news!
  - Theorem [Menon & Larson '17]: No deterministic SP mechanism is (even approximately) proportional.

- Randomized
  - Good news!
  - Theorem [Chen et al. '13, Mossel & Tamuz '10]: There is a randomized SP mechanism that always returns an envy-free allocation.

# Perfect Partition

- Theorem [Lyapunov '40]:
  - There always exists a "perfect partition" $(B_1, \ldots, B_n)$ of the cake such that $V_i(B_j) = {}^1/_n$ for every $i, j \in [n]$
  - Every agent values every piece at exactly $1/n$

- Theorem [Alon '87]:
  - There exists a perfect partition that only cuts the cake at $poly(n)$ points
  - In contrast, Lyapunov's proof is non-constructive and might need an unbounded number of cuts

- Unfortunately, computing a perfect partition needs an unbounded number of RW queries

## Perfect Partition

- If you're given an algorithm for finding a perfect partition…
  - Can you use it to design a randomized protocol that *always* returns an EF allocation and is SP-in-expectation?

  - Yes! Compute a perfect partition and assign the $n$ bundles to the $n$ agents uniformly at random

  - Why is this *always* EF?
    - Every agent values every bundle at $^1/_n$

  - Why is this SP-in-expectation?
    - Because an agent is assigned a random bundle, her expected utility is $^1/_n$, irrespective of what she reports

## Pareto Optimality (PO)

- Definition
  - We say that an allocation $A = (A_1, \dots, A_n)$ is PO if there is no alternative allocation $B = (B_1, \dots, B_n)$ such that

1. Every agent is at least as happy: $V_i(B_i) \geq V_i(A_i), \forall i \in N$
2. Some agent is strictly happier: $V_i(B_i) > V_i(A_i), \exists i \in N$

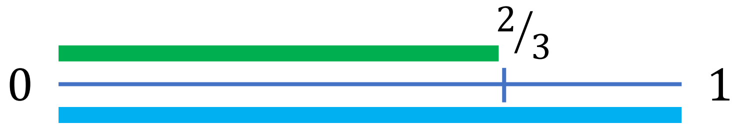
- Q: Is it PO to give the entire cake to agent 1?
  - A: Not necessarily. But yes, if agent 1 values every part of the cake positively.
  - But a "sequential dictatorship" is always Pareto optimal
    - Let agent 1 take whatever she values positively
    - From the rest, let agent 2 take whatever she values positively
    - And so on…

# PO + EF

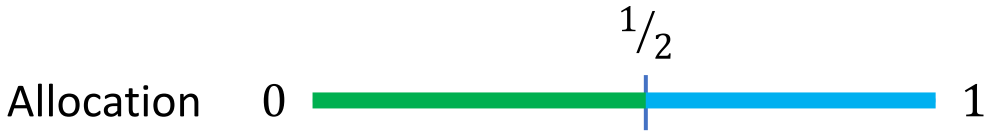
- Theorem [Weller '85]:
  - There always exists an allocation of the cake that is both envy-free and Pareto optimal.

- One way to achieve PO+EF:
  - Nash-optimal allocation: $\text{argmax}_A \prod_{i \in N} V_i(A_i)$
  - Obviously, this is PO. The fact that it is EF is somewhat non-trivial.
  - Named after John Nash
    - Nash social welfare = product of utilities
    - Different from utilitarian social welfare = sum of utilities

# Nash-Optimal Allocation



- Example:
  - Green agent has value 1 distributed over $[0, {}^2/_3]$
  - Blue agent has value 1 distributed over $[0,1]$
  - Without loss of generality (why?) suppose:
    - Green agent gets $x$ fraction of $[0, {}^2/_3]$
    - Blue agent gets the remaining $1 - x$ fraction of $[0, {}^2/_3]$ AND all of $[{}^2/_3, 1]$.
  - Green's utility = $x$, blue's utility = $(1 - \mathrm{x}) \cdot \frac{2}{3} + \frac{1}{3} = \frac{3-2x}{3}$
  - Maximize: $x \cdot \frac{3-2x}{3} \Rightarrow x = {}^3/_4$ (${}^3/_4$ fraction of ${}^2/_3$ is ${}^1/_2$).



Green has utility $\frac{3}{4}$

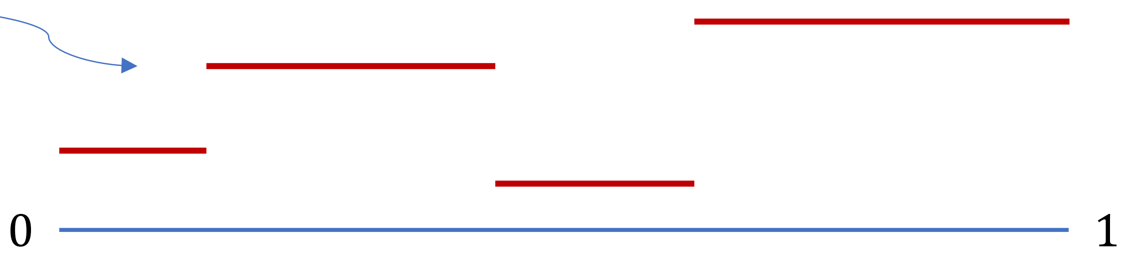
Blue has utility $\frac{1}{2}$

## Problem with Nash Solution

- Computing any Pareto optimal allocation already requires an unbounded number of queries

- Theorem [Aziz & Ye ‘14]:
  - For *piecewise constant* valuations, the Nash-optimal solution can be computed in polynomial time.

The density function of a piecewise constant valuation looks like this



# Homogeneous Divisible Goods

- Suppose there are $m$ homogeneous divisible goods
  - Each good can be divided fractionally between the agents
- Let $x_{i,g}$ = fraction of good $g$ that agent $i$ gets
  - Homogeneous = agent doesn't care which "part"
    - E.g., CPU or RAM
- Special case of cake-cutting
  - Line up the goods on $[0,1]$ → piecewise uniform valuations

# Homogeneous Divisible Goods

- Nash-optimal solution:

  Maximize $\sum_i \log U_i$

  $$U_i = \Sigma_g \, x_{i,g} * v_{i,g} \qquad \forall i$$
  $$\Sigma_i \, x_{i,g} = 1 \qquad \forall g$$
  $$x_{i,g} \in [0,1] \qquad \forall i, g$$

- This is known as the Gale-Eisenberg convex program
  - Can be solved *exactly* in strongly polynomial time