max planck institut
informatik

Topics in Computational Social Choice Theory

Lecture 02: Introduction on Fair Cake Division

Nidhi Rathi

Saarland Informatics
Campus

Last Lecture: Discrete Fair Division

What is **fairness as a concept**?

How to **compute a fair allocation**?

Agents with **valuations** over items

Indivisible items

Goal: To **divide** the items among the agents in a **fair** manner

Fair Division

What is **fairness as a concept**?

How to **compute a fair allocation**?

Goal: To **divide** the resource among the agents in a **fair** manner

Fair Division

What is **fairness as a concept**?

How to **compute a fair allocation**?

- **Mathematical study** of fairly allocating resources among agents with distinct preferences, but equal entitlements.

Goal: To **divide** the resource among the agents in a **fair** manner

Fair Division

What is **fairness as a concept**? How to **compute a fair allocation**?

- **Mathematical study** of fairly allocating resources among agents with distinct preferences, but equal entitlements.
- Focus on **provable guarantees**.

Goal: To **divide** the resource among the agents in a **fair** manner

Fair Division

What is **fairness as a concept**? How to **compute a fair allocation**?

- **Mathematical study** of fairly allocating resources among agents with distinct preferences, but equal entitlements.
- Focus on **provable guarantees**.
- **Computational Perspective**: work towards algorithms & hardness results and approximation algorithms

Goal: To **divide** the resource among the agents in a **fair** manner

Divisible Resource

Goal: To **divide** the resource among the agents in a **fair** manner

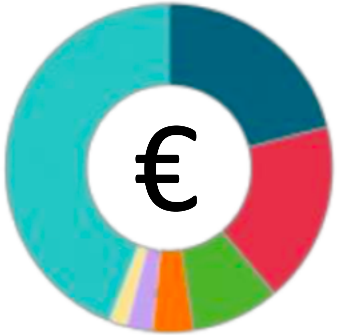
Divisible Resource

Goal: To **divide** the resource among the agents in a **fair** manner

Divisible Resource

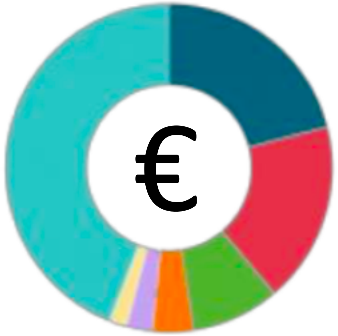
Goal: To **divide** the resource among the agents in a **fair** manner

Divisible Resource



Goal: To **divide** the resource among the agents in a **fair** manner

Divisible Resource



Goal: To **divide** the resource among the agents in a **fair** manner

Cake-Cutting

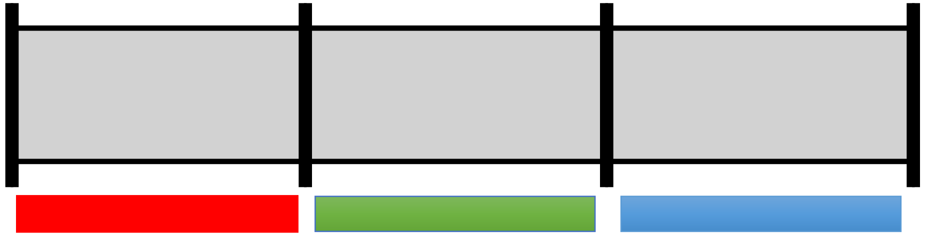
How to ***fairly*** cut the cake?

Cake-Cutting

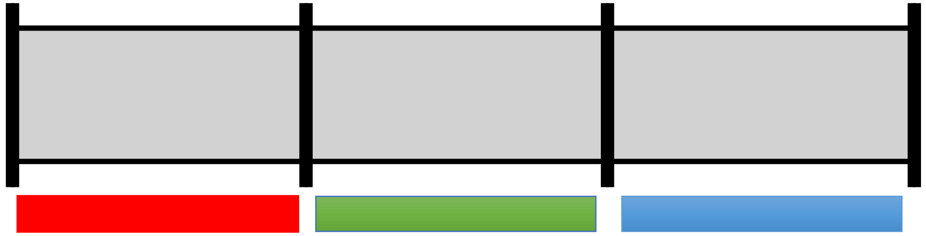
How to ***fairly*** divide a cake
among agents with **differing preferences**?

Why is this problem interesting?

Why is this problem interesting?



Why is this problem interesting?



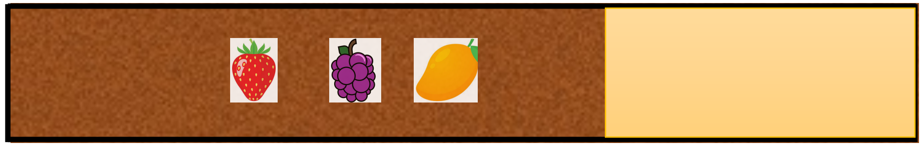
Fair

Why is this problem interesting?

I only like vanilla

I like chocolate and vanilla

I love fruits



Why is this problem interesting?

I only like vanilla

I like chocolate and vanilla

I love fruits

Is this division fair?

Why is this problem interesting?

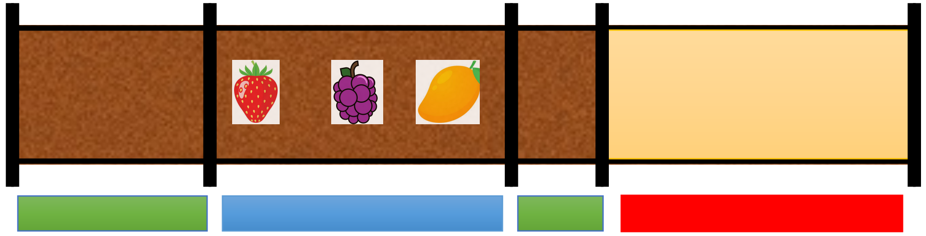
I only like vanilla

I like chocolate and vanilla

I love fruits

Is this division fair? 

Why is this problem interesting?

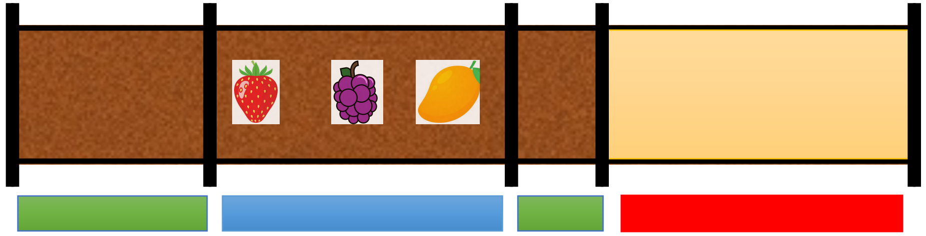


Why is this problem interesting?

I only like vanilla

I like chocolate and vanilla

I love fruits



Is this division fair?

Why is this problem interesting?

Preferences matter!

Cut-and-choose Protocol

Cut-and-choose Protocol

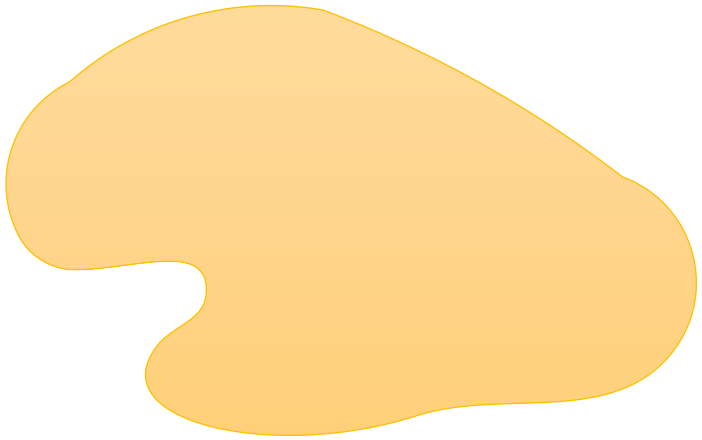
- First known to have appeared in an epic Greek poem of Theogony and then in the Bible.

Cut-and-choose Protocol

- First known to have appeared in an epic Greek poem of Theogony and then in the Bible.

  Two agents: Abraham and Lot

  Resource: A piece of land



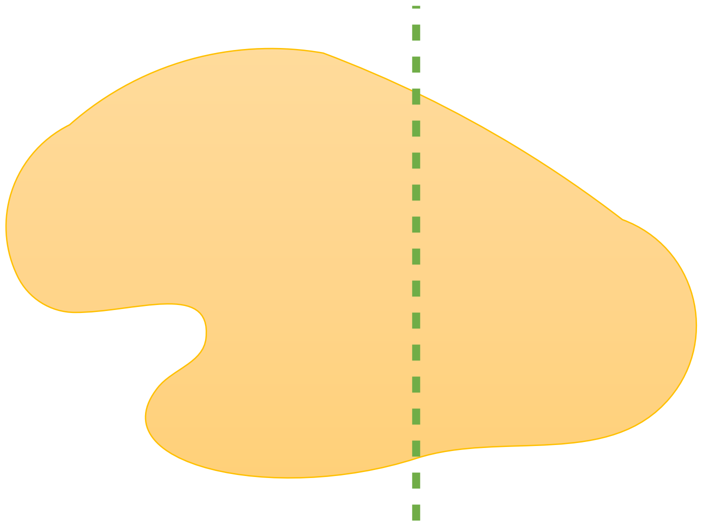
Cut-and-choose Protocol

- First known to have appeared in an epic Greek poem of Theogony and then in the Bible.

  Two agents: Abraham and Lot

  Resource: A piece of land

1. Abraham **cuts** the land into two pieces:
   the left & the right part



Cut-and-choose Protocol

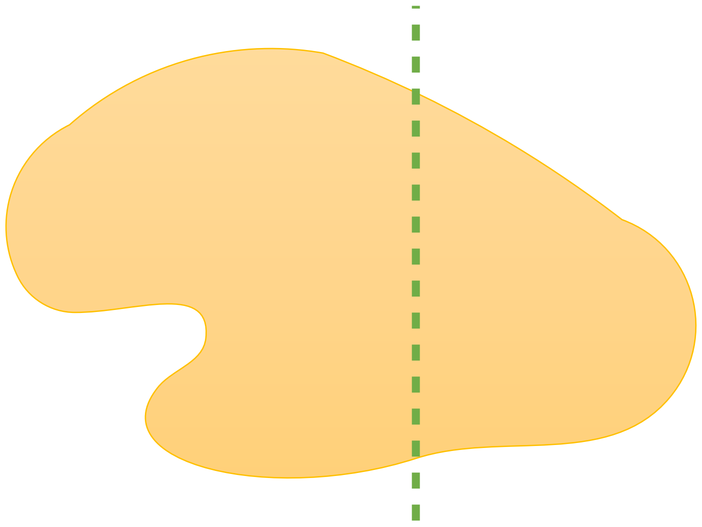
- First known to have appeared in an epic Greek poem of Theogony and then in the Bible.

  Two agents: Abraham and Lot

  Resource: A piece of land

1. Abraham **cuts** the land into two pieces:
   the left & the right part

2. Lot **chooses** between the two.



Cut-and-choose Protocol

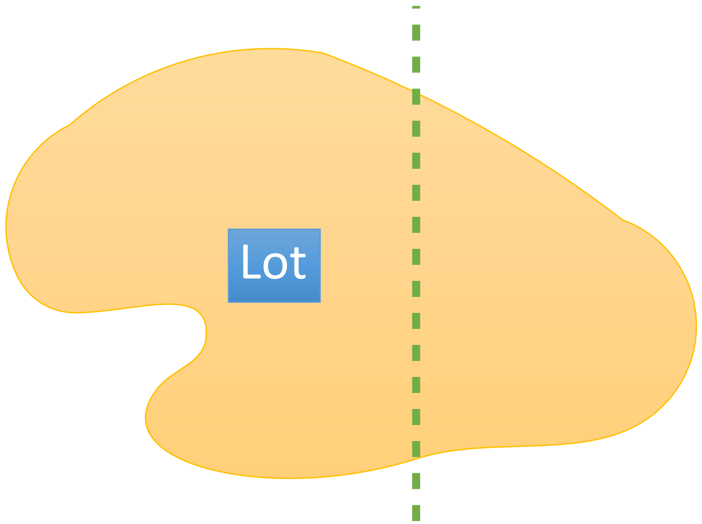
- First known to have appeared in an epic Greek poem of Theogony and then in the Bible.

  Two agents: Abraham and Lot

  Resource: A piece of land

1. Abraham **cuts** the land into two pieces:
   the left & the right part

2. Lot **chooses** between the two.



Cut-and-choose Protocol

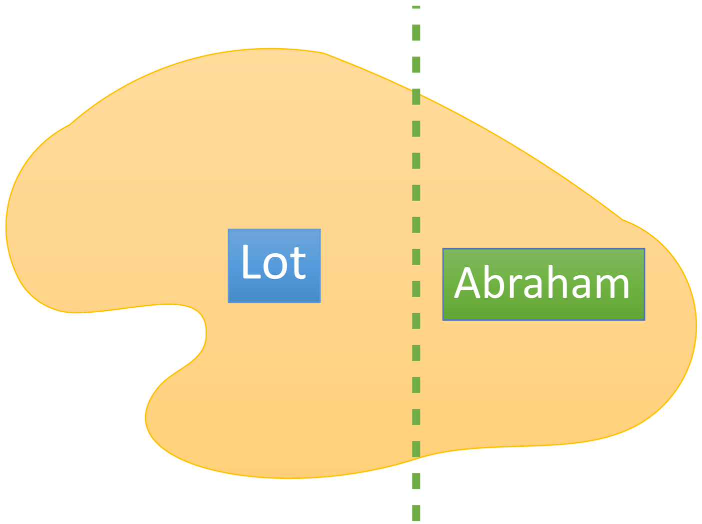
- First known to have appeared in an epic Greek poem of Theogony and then in the Bible.

  Two agents: Abraham and Lot

  Resource: A piece of land

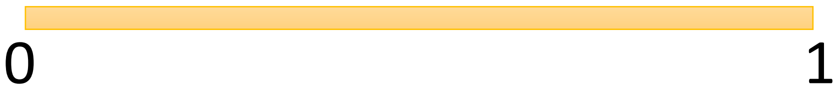
1. Abraham **cuts** the land into two pieces:
   the left & the right part

2. Lot **chooses** between the two.



Cut-and-choose Protocol

1. Abraham **cuts** the land into two pieces: the left & the right part
2. Lot **chooses** between the two.



Cut-and-choose Protocol

1. Abraham **cuts** the land into two pieces: the left & the right part
2. Lot **chooses** between the two.

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

We have:

$$v_1(A_1) = v_1(A_2) = 1/2$$

$$v_2(A_2) \geq v_2(A_1) \text{ and } v_2(A_2) \geq 1/2$$

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

We have:

$$v_1(A_1) = v_1(A_2) = 1/2$$

$v_2(A_2) \geq v_2(A_1)$ and $v_2(A_2) \geq 1/2$

$$v_1(A_1) \geq 1/2$$
$$v_2(A_2) \geq 1/2$$

Proportionality

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

We have:

$$v_1(A_1) = v_1(A_2) = 1/2$$

$v_2(A_2) \geq v_2(A_1)$ and $v_2(A_2) \geq 1/2$

$$v_1(A_1) \geq 1/2$$
$$v_2(A_2) \geq 1/2$$

Proportionality

$$v_1(A_1) \geq v_1(A_2)$$
$$v_2(A_2) \geq v_2(A_1)$$

Envy-freeness

The Model

The Model

- The resource: **Cake [0,1]** (heterogeneous and divisible)

The Model

- The resource: **Cake [0,1]** (heterogeneous and divisible)
- Set of **agents**: {1,2, …, n}

The Model

- The resource: **Cake [0,1]** (heterogeneous and divisible)
- Set of **agents**: {1,2, …, n}
- **Piece** of a cake: finite union of subintervals of [0,1]

Preferences of Agents

Preferences of Agents

- (**Cardinal**) preferences are expressed via valuation function

$$v_i : 2^{[0,1]} \rightarrow \mathbb{R}^+ \cup \{0\}$$

Preferences of Agents

- (**Cardinal**) preferences are expressed via valuation function

$$v_i : 2^{[0,1]} \to \mathbb{R}^+ \cup \{0\}$$

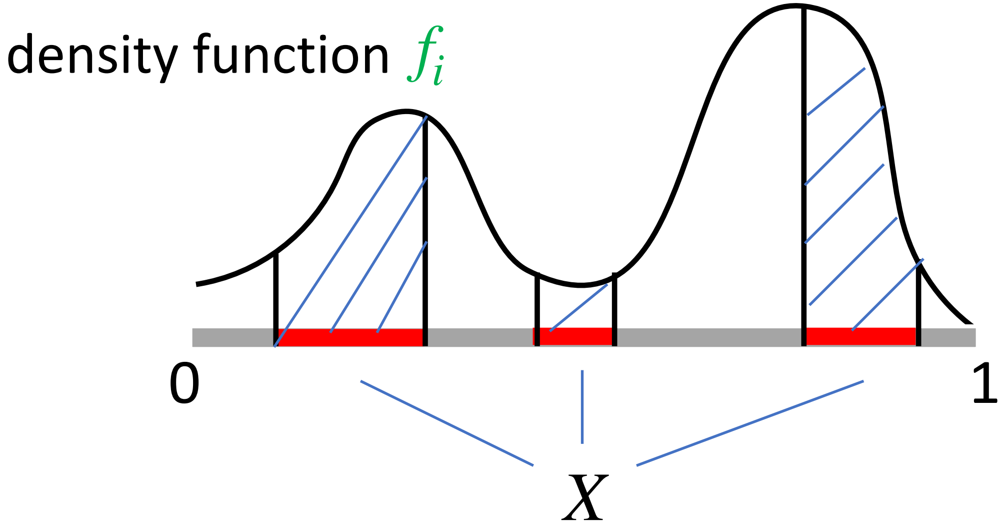
that assigns a non-negative value $v_i(X)$ to any piece $X \subseteq [0,1]$ of the cake

Preferences of Agents

- (**Cardinal**) preferences are expressed via valuation function

$$v_i : 2^{[0,1]} \rightarrow \mathbb{R}^+ \cup \{0\}$$

that assigns a non-negative value $v_i(X)$ to any piece $X \subseteq [0,1]$ of the cake



$$v_i(X) = \int_{x \in X} f_i(x)\, dx$$

Preferences of Agents

- Valuation function v_i : Agent i values piece X at $v_i(X) \geq 0$ (non-negative)

Preferences of Agents

- Valuation function v_i : Agent i values piece X at $v_i(X) \geq 0$ (non-negative)

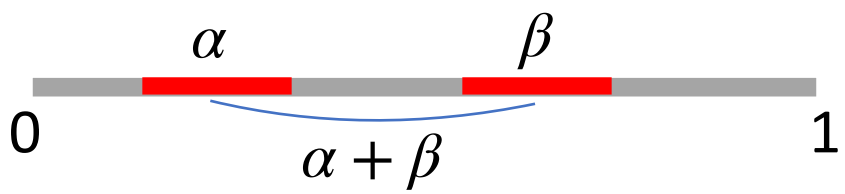
  Normalized: $v_i\,(\text{cake}) = 1$

Preferences of Agents

- Valuation function v_i : Agent i values piece X at $v_i(X) \geq 0$ (non-negative)

  Normalized: v_i (cake) $= 1$

  Additive:
  For disjoint $X, Y \subset [0,1]$, we have $v_i(X \cup Y) = v_i(X) + v_i(Y)$



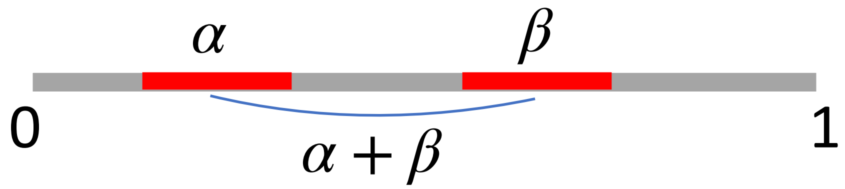
Preferences of Agents

- Valuation function v_i : Agent i values piece X at $v_i(X) \geq 0$ (non-negative)

Normalized: v_i (cake) $= 1$

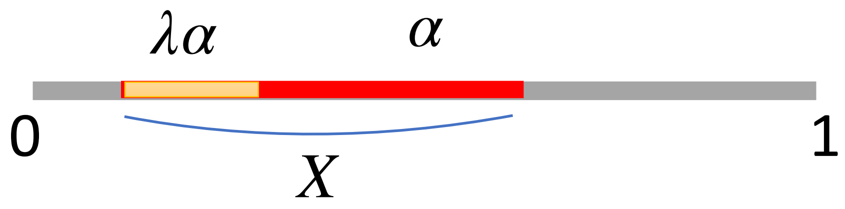
<u>Additive:</u>

For disjoint $X, Y \subset [0,1]$, we have $v_i(X \cup Y) = v_i(X) + v_i(Y)$



<u>Divisible:</u>

For any $X \subseteq [0,1]$ and $\lambda \in [0,1]$, there exists a $Y \subseteq X$ s.t. $v_i(Y) = \lambda v_i(X)$



Preferences of Agents

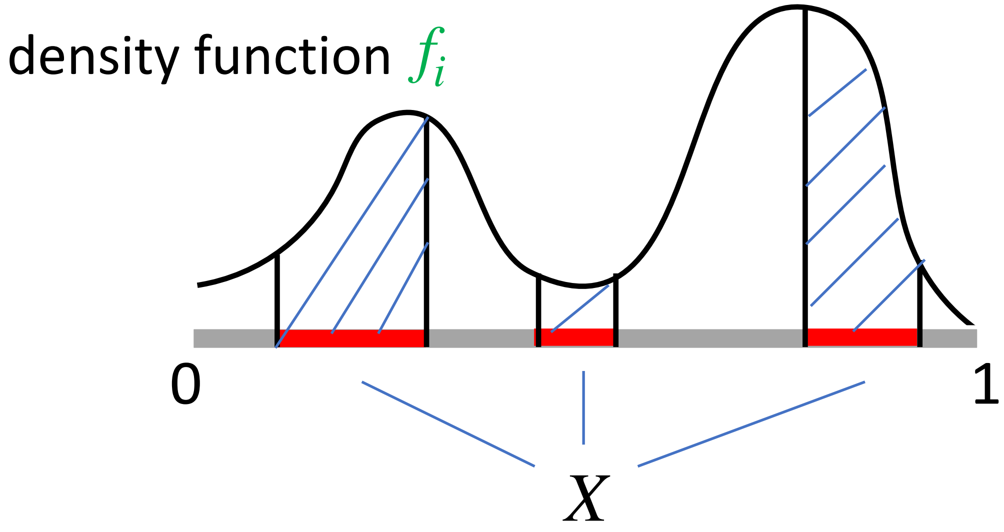
- Valuation function v_i : Agent i values piece X at $v_i(X) \geq 0$ (non-negative)

Normalized
Additive
Divisible

Preferences of Agents

- Valuation function v_i : Agent i values piece X at $v_i(X) \geq 0$ (non-negative)

Normalized
Additive
Divisible



$$v_i(X) = \int_{x \in X} f_i(x)\ dx$$

v_i is a probability distribution over $[0,1]$

Robertson-Webb Query Model

Robertson-Webb Query Model

Two types of queries to access the valuations:

Robertson-Webb Query Model

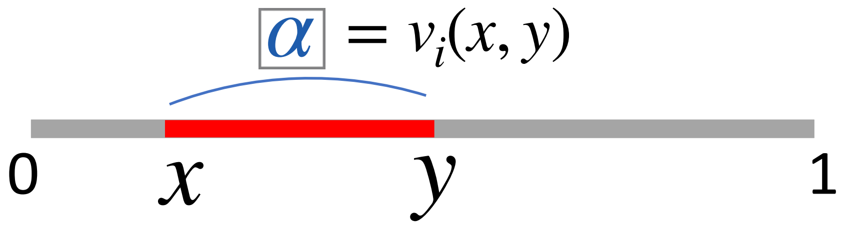
Two types of queries to access the valuations:

(1) $\text{eval}_i\ ([x, y]) = v_i\ ([x, y])$

Robertson-Webb Query Model

Two types of queries to access the valuations:

(1) $\text{eval}_i\ ([x, y]) = v_i\ ([x, y])$



Robertson-Webb Query Model

Two types of queries to access the valuations:

(1) $\text{eval}_i\ ([x, y]) = v_i\ ([x, y])$

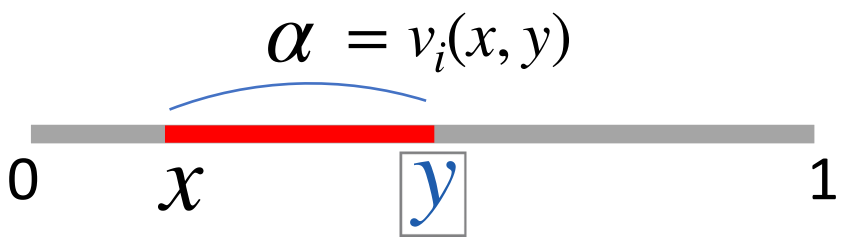
(2) $\text{cut}_i\ (x, \alpha) = y$ such that $v_i\ ([x, y]) = \alpha$

Robertson-Webb Query Model

Two types of queries to access the valuations:

(1) $\text{eval}_i\ ([x, y]) = v_i\ ([x, y])$

(2) $\text{cut}_i\ (x, \alpha) = y$ such that $v_i\ ([x, y]) = \alpha$

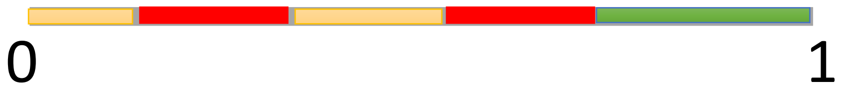


Fairness Notions

Fairness Notions

Allocation:

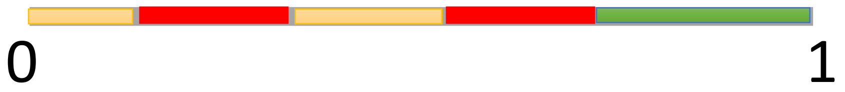
A partition $A = (A_1, A_2, \ldots, A_n)$ of the cake $[0,1]$ where piece A_i belongs to agent i



Fairness Notions

Allocation:

A partition $A = (A_1, A_2, \ldots, A_n)$ of the cake $[0,1]$ where piece A_i belongs to agent i



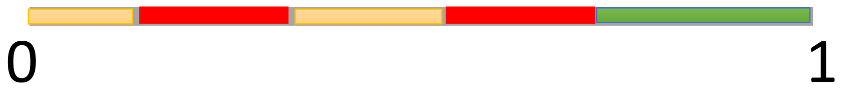
- **Proportionality:** for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$

  [Steinhaus, 1948]

Fairness Notions

Allocation:

A partition $A = (A_1, A_2, \ldots, A_n)$ of the cake $[0,1]$ where piece A_i belongs to agent i



- **Proportionality:** for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$

  [Steinhaus, 1948]

- **Envy-freeness:** for every pair $i, j \in [n]$ of agents, we have $v_i(A_i) \geq v_i(A_j)$

  [Foley 1967]

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

We have:

$$v_1(A_1) = v_1(A_2) = 1/2$$

$v_2(A_2) \geq v_2(A_1)$ and $v_2(A_2) \geq 1/2$

The cut-and-choose outcome is **EF** and **Prop**

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

We have:

$$v_1(A_1) = v_1(A_2) = 1/2$$

$$v_2(A_2) \geq v_2(A_1) \text{ and } v_2(A_2) \geq 1/2$$

The cut-and-choose outcome is **EF** and **Prop**

EF and **Prop** are *equivalent for two agents*

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

The cut-and-choose outcome is **EF** and **Prop**

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

The cut-and-choose outcome is **EF** and **Prop**

Can cut-and-choose be implemented in RW model?

Cut-and-choose Protocol

1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

The cut-and-choose outcome is **EF** and **Prop**

Can cut-and-choose be implemented in RW model? Yes!

$$\text{cut}_1(0,1/2) = x$$

$$\text{eval}_2(0,x)$$

Cut-and-choose Protocol

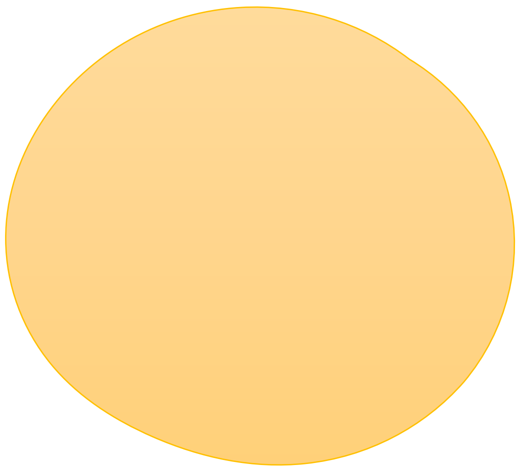
1. Abraham (agent 1) cuts the cake $[0,1]$ into two pieces of equal value to him.
2. Lot (agent 2) selects of the two pieces $[0,x]$ or $[x,1]$ the one of higher value to him.

The cut-and-choose outcome is **EF** and **Prop**

For two agents, an EF/Prop cake division can be computed using two queries

Fairness Notions

- Proportionality: for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$
- Envy-freeness: for every pair $i, j \in [n]$ of agents, we have $v_i(A_i) \geq v_i(A_j)$

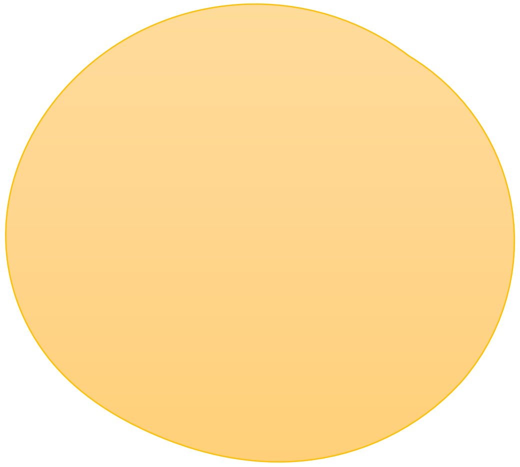


All allocations

Fairness Notions

- Proportionality: for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$
- Envy-freeness: for every pair $i, j \in [n]$ of agents, we have $v_i(A_i) \geq v_i(A_j)$

EF $\Longrightarrow$ Prop for *any* number of agents



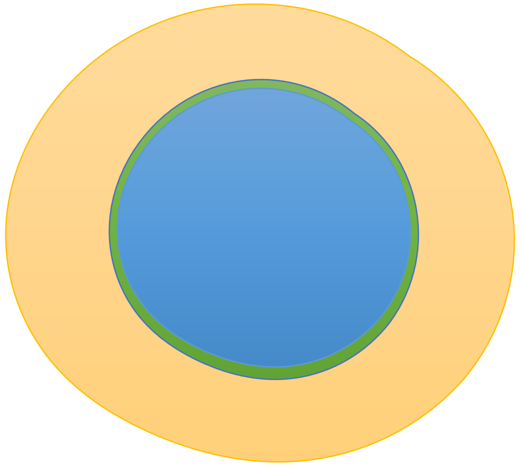
All allocations

Fairness Notions

- Proportionality: for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$
- Envy-freeness: for every pair $i, j \in [n]$ of agents, we have $v_i(A_i) \geq v_i(A_j)$

EF $\Longrightarrow$ Prop for *any* number of agents

Prop $\Longrightarrow$ EF for *two* agents



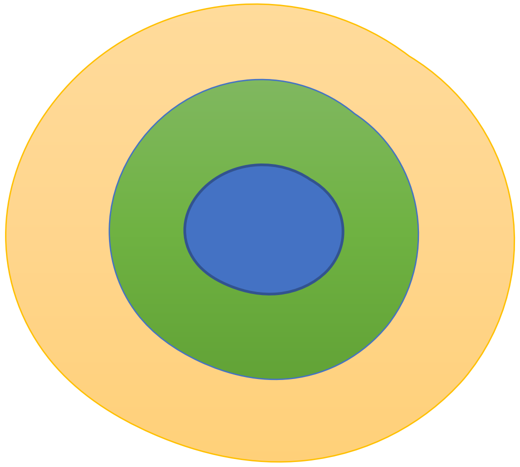
All allocations

Fairness Notions

- Proportionality: for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$
- Envy-freeness: for every pair $i, j \in [n]$ of agents, we have $v_i(A_i) \geq v_i(A_j)$

EF $\Longrightarrow$ Prop for *any* number of agents

Prop $\Longrightarrow$ EF for *two* agents *(but no more)*



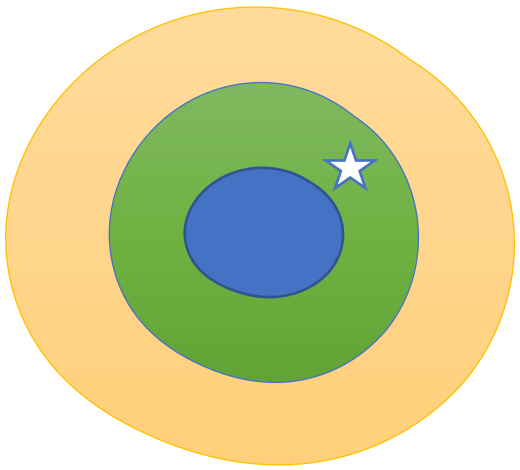
All allocations

Fairness Notions

- Proportionality: for each agent $i \in [n]$, we have $v_i(A_i) \geq 1/n$
- Envy-freeness: for every pair $i, j \in [n]$ of agents, we have $v_i(A_i) \geq v_i(A_j)$

EF $\Longrightarrow$ Prop for *any* number of agents

Prop $\Longrightarrow$ EF for *two* agents *(but no more)*



All allocations

A proportional cake division always exists and can be computed efficiently

A proportional cake division always exists and can be computed efficiently

(i) Moving-knife Protocol - Dubins and Spanier [1961]
(ii) Even-Paz Protocol [1984]

Reference: Handbook of Computational Social Choice, see Chapter 13 by Ariel Procaccia.

Moving-Knife Protocol (Dubins-Spanier)

An efficient proportional cake division protocol for any number of agents

Moving-Knife Protocol (Dubins-Spanier)

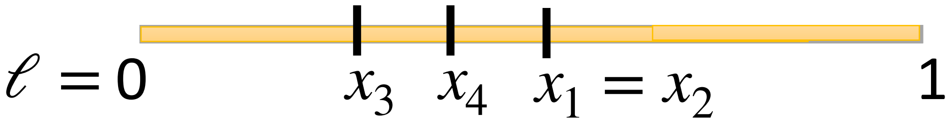
1. Initialize $\ell = 0$ and $W = [n]$

$\ell = 0$ 1

$W = \{1,2,3,4\}$

Moving-Knife Protocol (Dubins-Spanier)

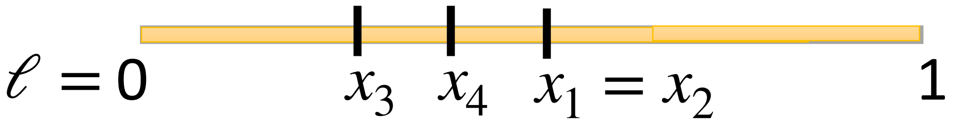
1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
    - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$



$W = \{1,2,3,4\}$

Moving-Knife Protocol (Dubins-Spanier)

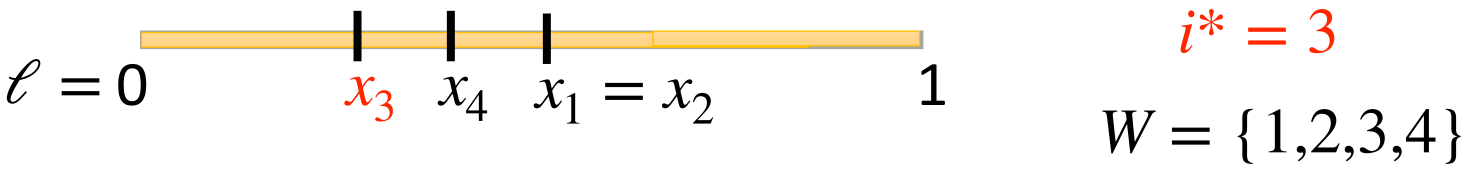
1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$



$W = \{1,2,3,4\}$

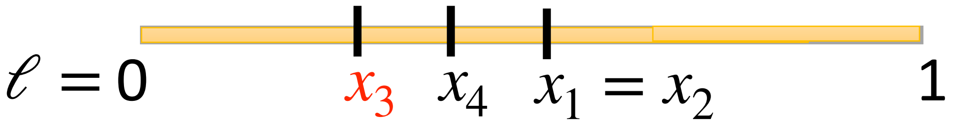
Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$



Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \underset{i \in W}{\operatorname{argmin}} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$

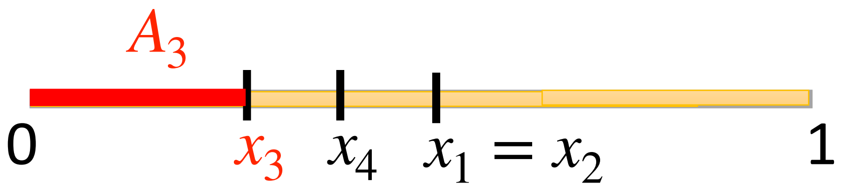


$i^* = 3$

$W = \{1,2,3,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$



$i^* = 3$

$W = \{1,2,3,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \backslash \{i^*\}$

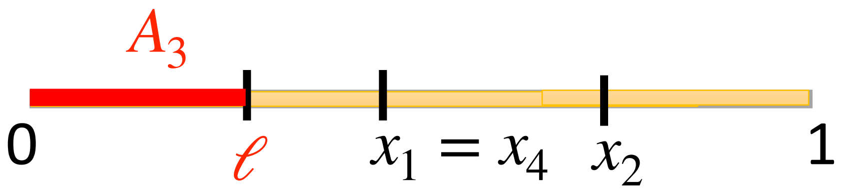
$v_3(A_3) = 1/4$

$W = \{1,2,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$

$v_3(A_3) = 1/4$

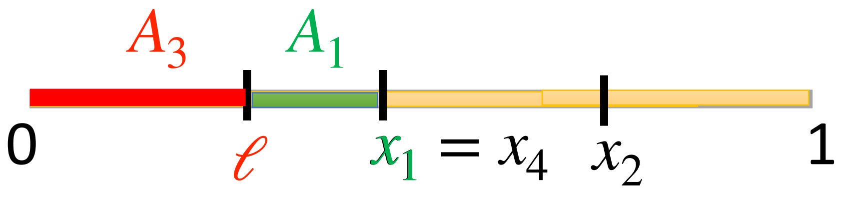


$W = \{1,2,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname*{argmin}_{i \in W} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \backslash \{i^*\}$

$v_3(A_3) = 1/4$

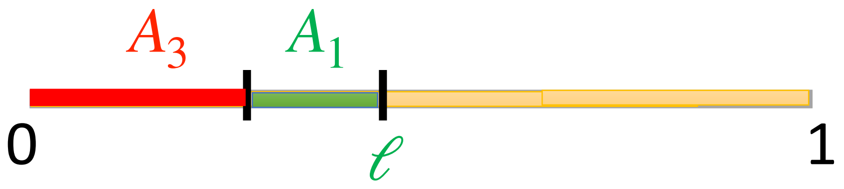


$i^* = 1$

$W = \{1,2,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$

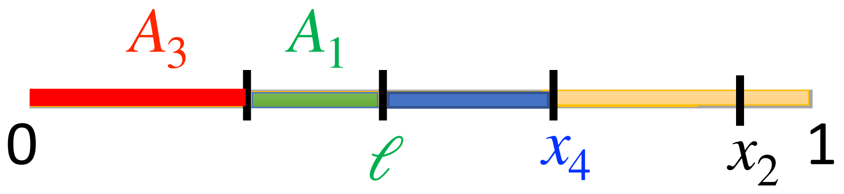


$v_3(A_3) = 1/4$
$v_1(A_1) = 1/4$

$W = \{2,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname*{argmin}_{i \in W} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \backslash \{i^*\}$



$v_3(A_3) = 1/4$

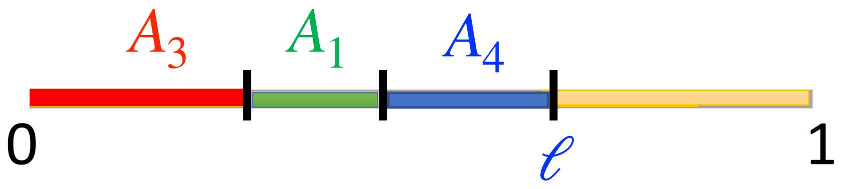
$v_1(A_1) = 1/4$

$i^* = 4$

$W = \{2,4\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \underset{i \in W}{\text{argmin}} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$



$v_3(A_3) = 1/4$

$v_1(A_1) = 1/4$

$v_4(A_4) = 1/4$

$W = \{2\}$

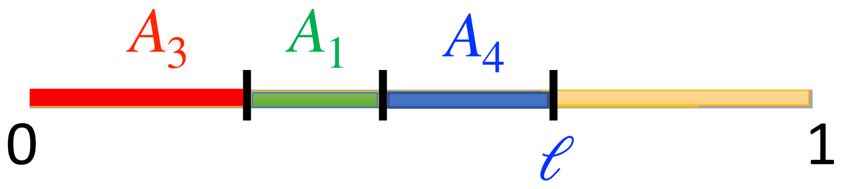
Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \underset{i \in W}{\operatorname{argmin}} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \backslash \{i^*\}$
3. Give the remaining piece to the agent left in W

$v_3(A_3) = 1/4$

$v_1(A_1) = 1/4$

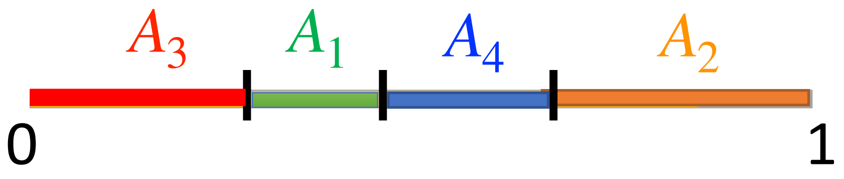
$v_4(A_4) = 1/4$



$W = \{2\}$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \backslash \{i^*\}$
3. Give the remaining piece to the agent left in W



$W = \varnothing$

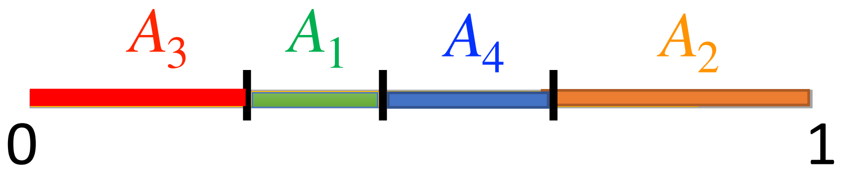
$v_3(A_3) = 1/4$

$v_1(A_1) = 1/4$

$v_4(A_4) = 1/4$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \backslash \{i^*\}$
3. Give the remaining piece to the agent left in W



$W = \varnothing$

$v_3(A_3) = 1/4$

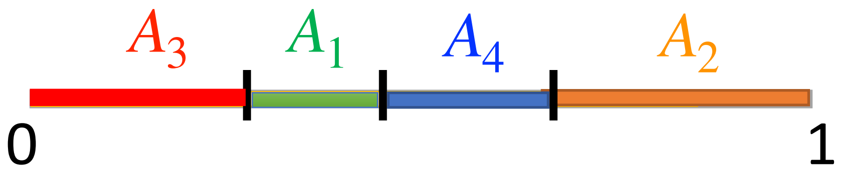
$v_1(A_1) = 1/4$

$v_4(A_4) = 1/4$

$v_2(A_2) \geq 1/4$

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \underset{i \in W}{\operatorname{argmin}} \; x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$
3. Give the remaining piece to the agent left in W



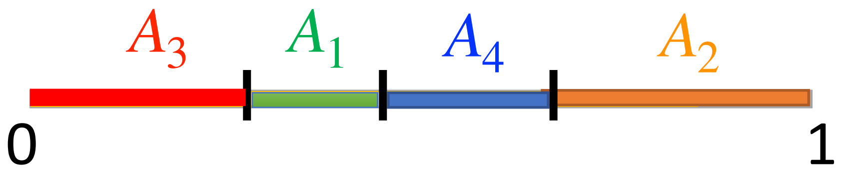
In general,

$v_i(A_i) = 1/n$ for all agents

$v_i(A_i) \geq 1/n$ for the last agent

Moving-Knife Protocol (Dubins-Spanier)

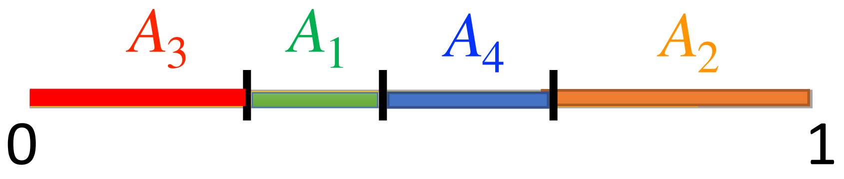
1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - Each agent $i \in W$ marks $x_i \in [\ell, 1]$ such that $v_i([\ell, x_i]) = 1/n$
   - Set $i^* = \operatorname{argmin}_{i \in W} x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$
3. Give the remaining piece to the agent left in W



Prop

Moving-Knife Protocol (Dubins-Spanier)

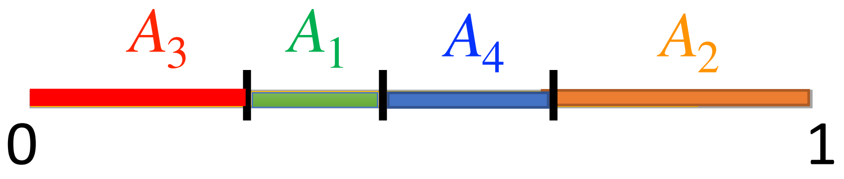
1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - $\text{cut}_i([\ell, 1],\ 1/n)$ to each agent $i \in W$
   - Set $i^* = \text{argmin}_{i \in W}\ x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$
3. Give the remaining piece to the agent left in W



Prop

Moving-Knife Protocol (Dubins-Spanier)

1. Initialize $\ell = 0$ and $W = [n]$
2. While $|W| > 1$,
   - $\text{cut}_i([\ell, 1],\ 1/n)$ to each agent $i \in W$
   - Set $i^* = \underset{i \in W}{\text{argmin}}\ x_i$
   - Set $A_{i^*} = [\ell, x_{i^*}]$
   - Update $\ell = x_i$ and $W = W \setminus \{i^*\}$
3. Give the remaining piece to the agent left in W



Prop

A total of $\mathcal{O}(n^2)$ queries

Query Complexity of Proportionality

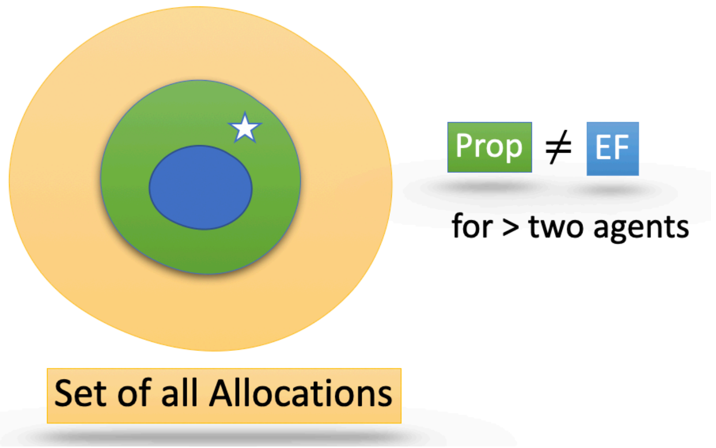
Query Complexity of Proportionality

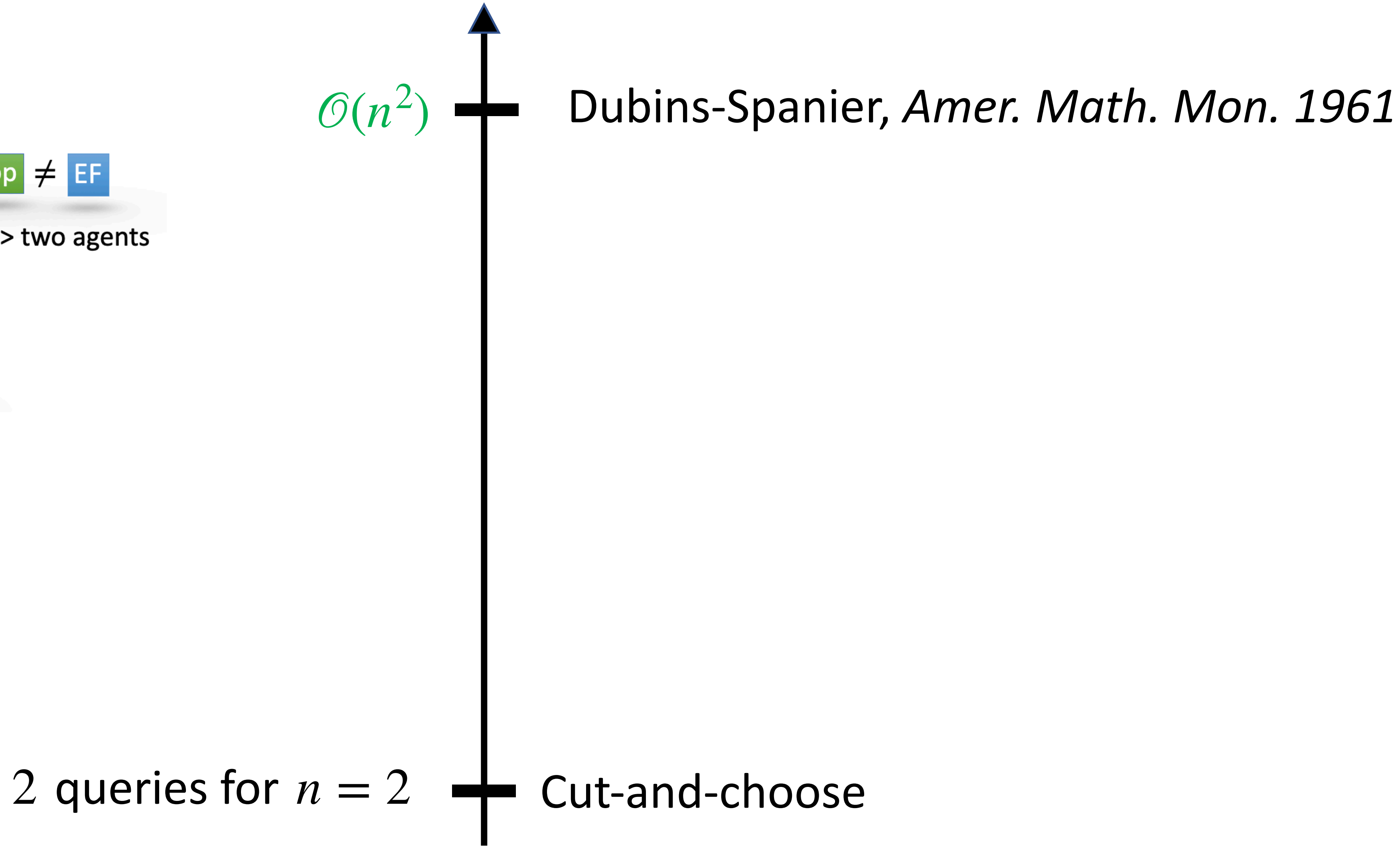
Set of all Allocations

Prop ≠ EF

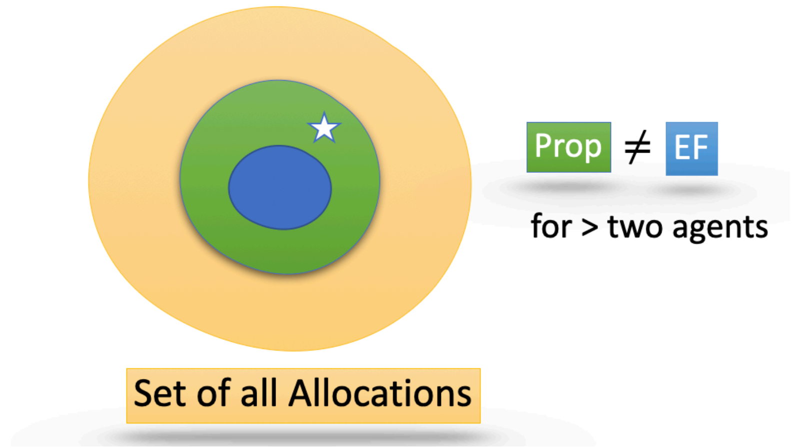
for > two agents

Query Complexity of Proportionality





Query Complexity of Proportionality

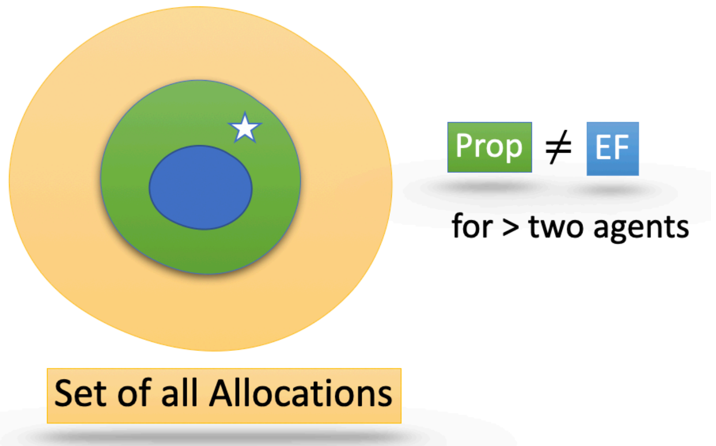


$\mathcal{O}(n^2)$ — Dubins-Spanier, *Amer. Math. Mon. 1961*

$\mathcal{O}(n \log n)$ — Even-Paz [1984] (via recursion)

2 queries for $n = 2$ — Cut-and-choose

Query Complexity of Proportionality



$\mathcal{O}(n^2)$ — Dubins-Spanier, *Amer. Math. Mon. 1961*

$\mathcal{O}(n \log n)$ — Even-Paz [1984] (via recursion)

$\Omega(n \log n)$ — Edmonds & Pruhs, *TALG 2011]*

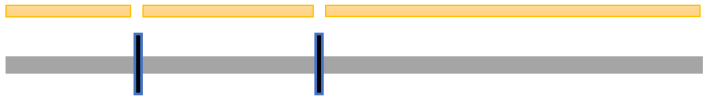
2 queries for $n = 2$ — Cut-and-choose

Envy-free Protocol

Envy-free Protocol for 3 agents

Envy-free Protocol for 3 agents

make three
equal pieces



Envy-free Protocol for 3 agents

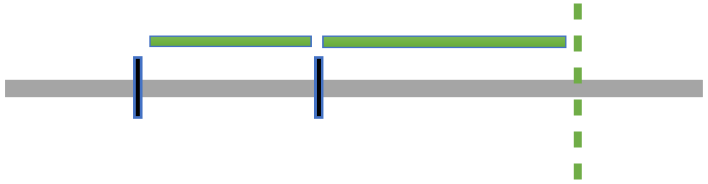
make three
equal pieces

make my
top two
pieces equal

Envy-free Protocol for 3 agents

make three
equal pieces

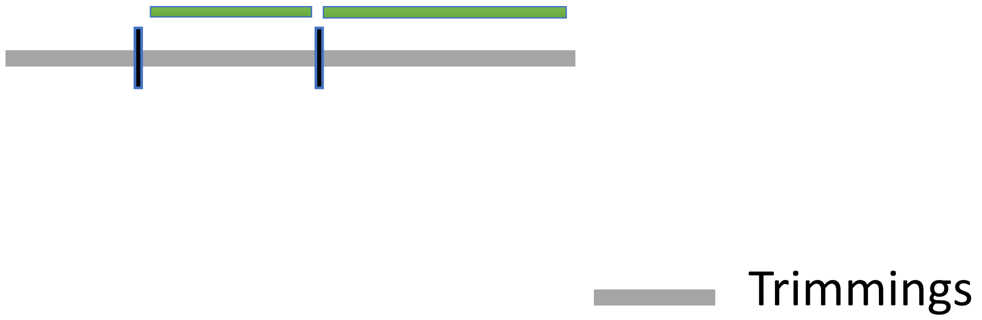
make my
top two
pieces equal



Envy-free Protocol for 3 agents

make three
equal pieces

make my
top two
pieces equal



Envy-free Protocol for 3 agents

make three
equal pieces

make my
top two
pieces equal

I pick first

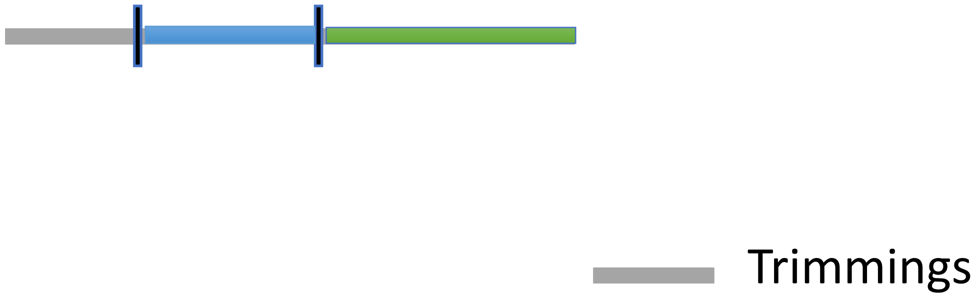
Envy-free Protocol for 3 agents

make three
equal pieces

make my
top two
pieces equal

I pick second
(one of the trimmed pieces)

I pick first



Envy-free Protocol for 3 agents

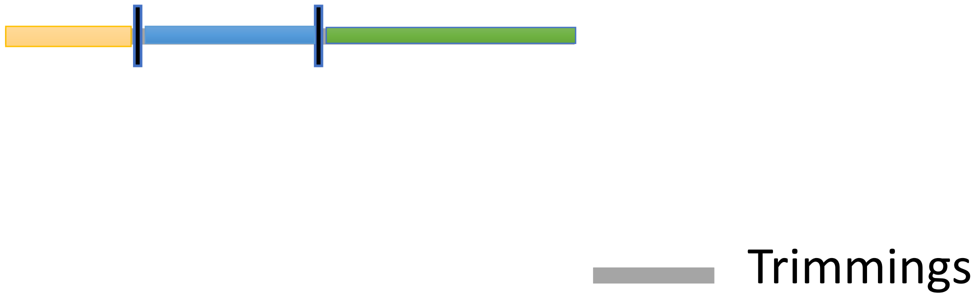
make three equal pieces

I pick last

make my top two pieces equal

I pick second *(one of the trimmed pieces)*

I pick first



Envy-free Protocol for 3 agents

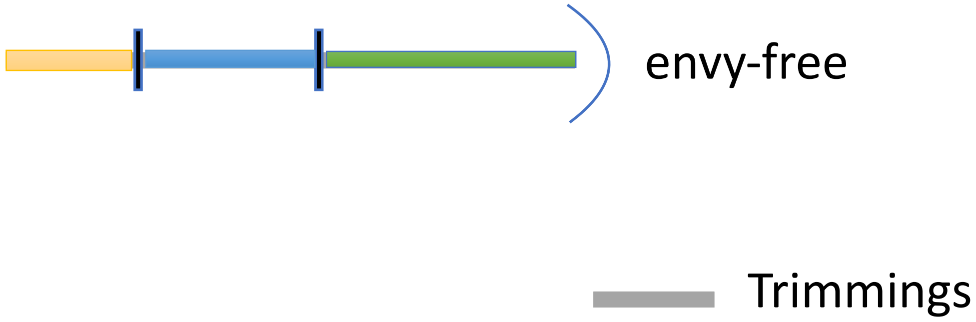
make three equal pieces

I pick last
(untrimmed piece)

make my top two pieces equal

I pick second
(one of the trimmed pieces)

I pick first



Envy-free Protocol for 3 agents

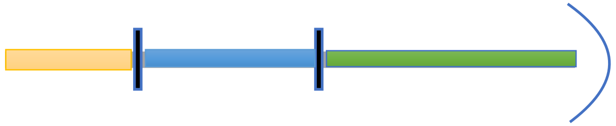
make three equal pieces

I pick last
(untrimmed piece)

make my top two pieces equal

I pick second
(one of the trimmed pieces)

I pick first hence EF



envy-free

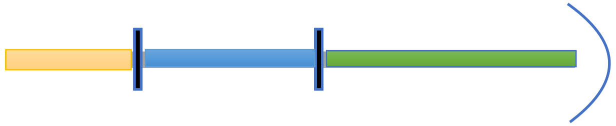
Trimmings

Envy-free Protocol for 3 agents

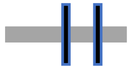
Trimmings (T)

Envy-free Protocol for 3 agents

equi-divide T
& pick last



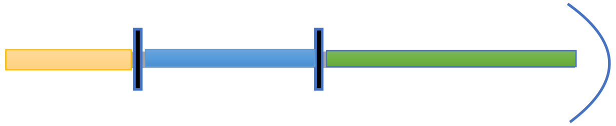
envy-free



Trimmings (T)

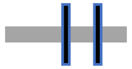
Envy-free Protocol for 3 agents

I pick second



envy-free

I pick first

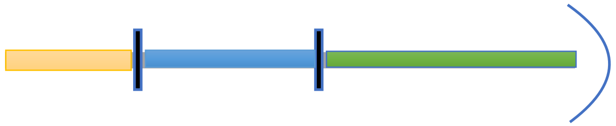


Trimmings (T)

equi-divide T
& pick last

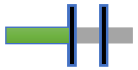
Envy-free Protocol for 3 agents

I pick second



envy-free

I pick first,
hence EF



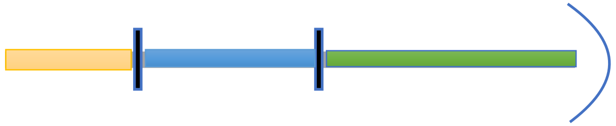
Trimmings (T)

equi-divide T
& pick last

Envy-free Protocol for 3 agents

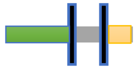
I pick second

Advantage from first round, hence EF



envy-free

I pick first,
hence EF



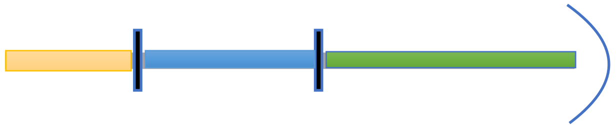
Trimmings (T)

equi-divide T
& pick last

Envy-free Protocol for 3 agents

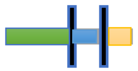
I pick second

Advantage from first round, hence EF



envy-free

I pick first, hence EF



Trimmings (T)

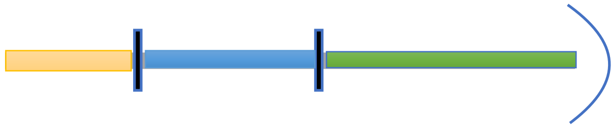
equi-divide T & pick last

I equi-divided hence EF

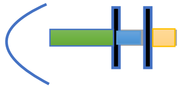
Envy-free Protocol for 3 agents

I pick second

Advantage from first round, hence EF



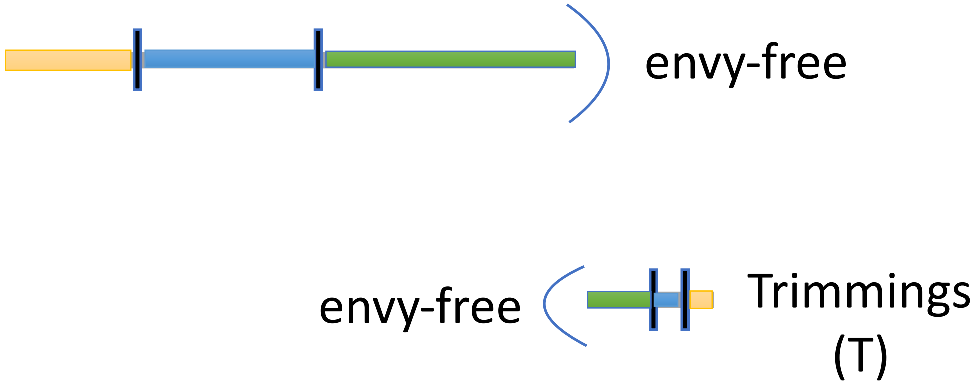
I pick first, hence EF



equi-divide T & pick last

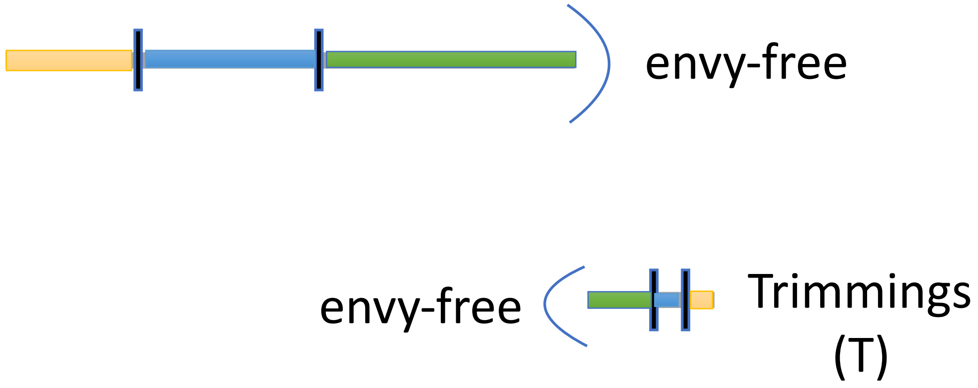
I equi-divided hence EF

Envy-free Protocol for 3 agents



Hence, we find an envy-free cake division

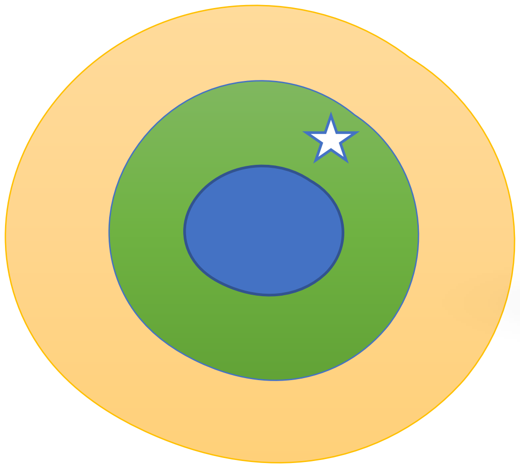
Envy-free Protocol for 3 agents



Selfridge-Conway protocol finds an EF cake division among *three* agents using $\mathcal{O}(1)$ queries

Existence of Envy-free Cake Divisions

Existence of Envy-free Cake Divisions



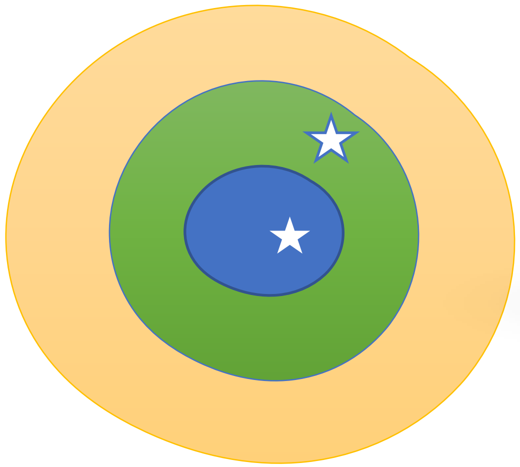
All allocations

EF $\Rightarrow$ Prop, EF $\not\Leftarrow$ Prop

Stromquist [1980], Su [1999]

Envy-free cake division exist for any number of agents (Lecture 04)

Existence of Envy-free Cake Divisions



All allocations

EF $\Longrightarrow$ Prop

EF $\not\Longleftarrow$ Prop

Stromquist [1980], Su [1999]

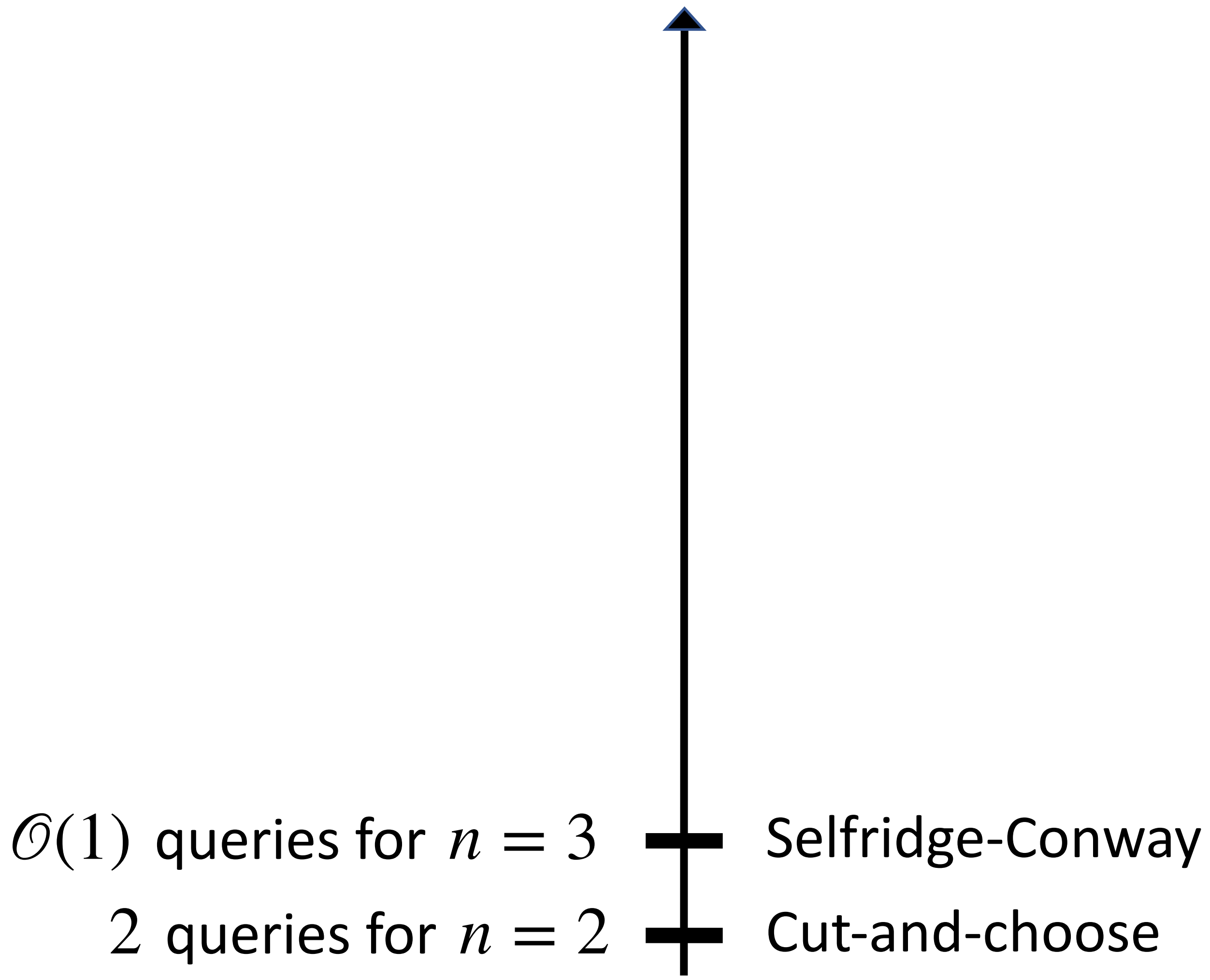
Envy-free cake division exist for any number of agents (Lecture 04)

Query Complexity of Envy-freeness

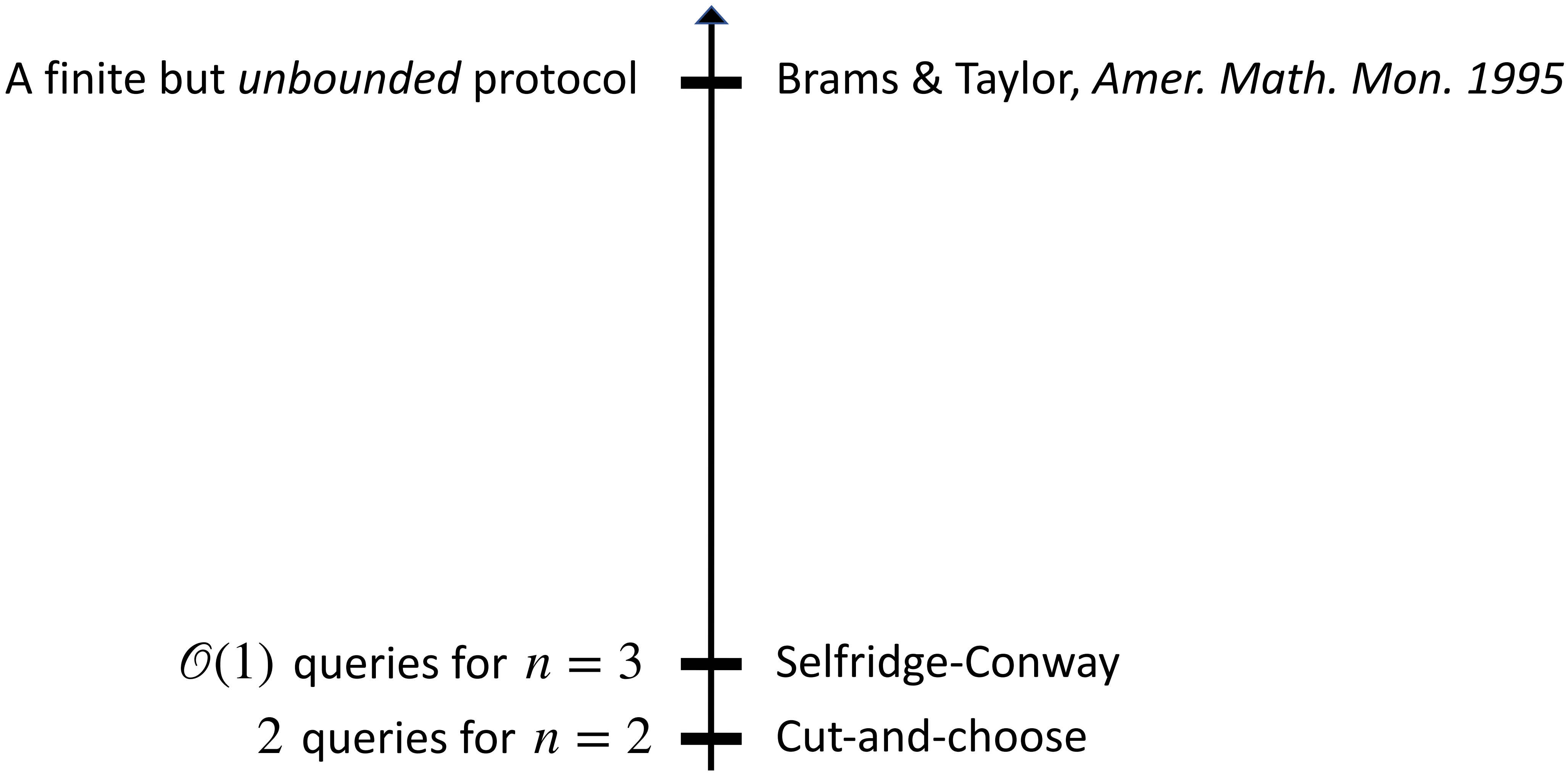
Query Complexity of Envy-freeness

2 queries for $n = 2$ Cut-and-choose

Query Complexity of Envy-freeness



Query Complexity of Envy-freeness



Query Complexity of Envy-freeness

A finite but *unbounded* protocol — Brams & Taylor, *Amer. Math. Mon. 1995*

$\mathcal{O}(n^{n^{n^{n^{n^{n}}}}})$ — Aziz & Mackenzie, *FOCS 2016*

$\mathcal{O}(1)$ queries for $n = 3$ — Selfridge-Conway

2 queries for $n = 2$ — Cut-and-choose

Query Complexity of Envy-freeness

A finite but *unbounded* protocol — Brams & Taylor, *Amer. Math. Mon. 1995*

$\mathcal{O}(n^{n^{n^{n^{n^{n}}}}})$ — Aziz & Mackenzie, *FOCS 2016*

$\Omega(n^2)$ — Procaccia, *IJCAI 2009*

$\mathcal{O}(1)$ queries for $n = 3$ — Selfridge-Conway

2 queries for $n = 2$ — Cut-and-choose

Query Complexity of Envy-freeness

A finite but *unbounded* protocol — Brams & Taylor, *Amer. Math. Mon. 1995*

$\mathcal{O}(n^{n^{n^{n^{n}}}})$ — Aziz & Mackenzie, *FOCS 2016*

Open

$\Omega(n^2)$ — Procaccia, *IJCAI 2009*

$\mathcal{O}(1)$ queries for $n = 3$ — Selfridge-Conway

2 queries for $n = 2$ — Cut-and-choose

Source: Lecture slides of Rohit Vaish, IIT Delhi

Query Complexity of Envy-freeness

What happens when every agent wishes to have a ***contiguous*** piece of the cake?

Query Complexity of Envy-freeness

What happens when every agent wishes to have a ***contiguous*** piece of the cake?

Stromquist [1980], Su [1999]

Envy-free cake division exists for any number of agents

(4th Lecture)

Query Complexity of Envy-freeness

What happens when every agent wishes to have a ***contiguous*** piece of the cake?

Stromquist [1980], Su [1999] **connected pieces**

Envy-free cake division exists for any number of agents (4th Lecture)

Query Complexity of Envy-freeness

Stromquist [1980], Su [1999] **connected pieces**

Envy-free cake division exists for any number of agents

Query Complexity of Envy-freeness

Stromquist [1980], Su [1999] **connected pieces**

Envy-free cake division exists for any number of agents

Stromquist, *J. of Combinatorics 2008* even for *three* agents!

No finite-query protocol exists for connected EF cake division

Query Complexity of Envy-freeness

Stromquist [1980], Su [1999] **connected pieces**

Envy-free cake division exists for any number of agents

(30 April)

Stromquist, *J. of Combinatorics 2008* even for *three* agents!

No finite-query protocol exists for connected EF cake division

[ABK**R**] *WINE 2019* (Fair and Efficient Cake Division with Connected Pieces)

An efficient algorithm: **1/2-EF + 1/3-NSW** allocation for *connected* EF cake division

(28 May)

Don't forget!

Send us your preferred list of the student papers by April 30th.