On Cutting Cakes and Crossing Curves

Kilian Risse

Lund University

EC 2026

Joint work with Gilbert Maystre and Alexandros Hollender

Cake Cutting

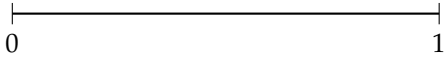
Cake Cutting

Cake Cutting

- n agents

Cake Cutting

- n agents

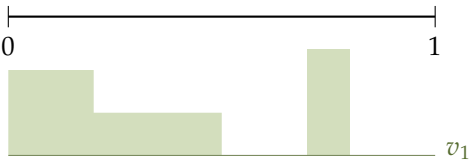


Cake Cutting

- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \left\{[a,b] \colon 0 \leq a \leq b \leq 1\right\} \to \mathbb{R}_{\geq 0}$$

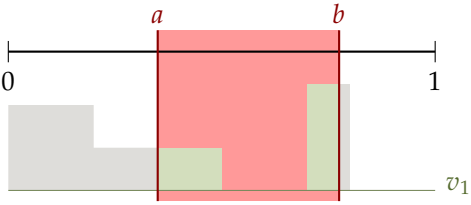
Cake Cutting



- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \big\{[a,b] \colon 0 \le a \le b \le 1\big\} \to \mathbb{R}_{\ge 0}$$

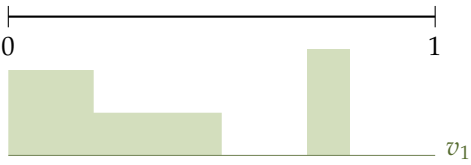
Cake Cutting



- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \left\{[a,b] \colon 0 \le a \le b \le 1\right\} \to \mathbb{R}_{\ge 0}$$

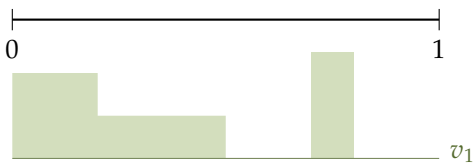
Cake Cutting



- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \big\{[a,b] \colon 0 \le a \le b \le 1\big\} \to \mathbb{R}_{\ge 0}$$

Cake Cutting

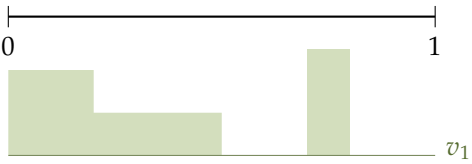


- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \{[a,b] \colon 0 \le a \le b \le 1\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous

Cake Cutting

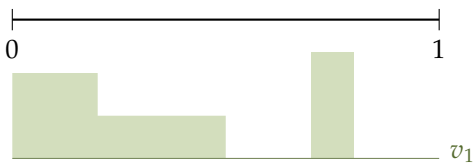


- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \left\{[a,b] \colon 0 \le a \le b \le 1\right\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$

Cake Cutting

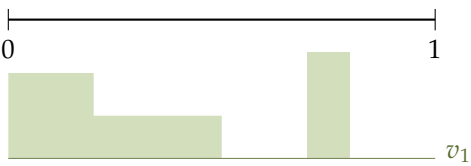


- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \{[a,b] \colon 0 \le a \le b \le 1\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$
  - v_i not assumed to be additive/monotone

Cake Cutting

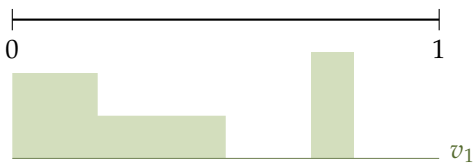


- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \left\{[a,b] \colon 0 \le a \le b \le 1\right\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$
  - v_i not assumed to be additive/monotone
- Focus on connected divisions of cake

Cake Cutting



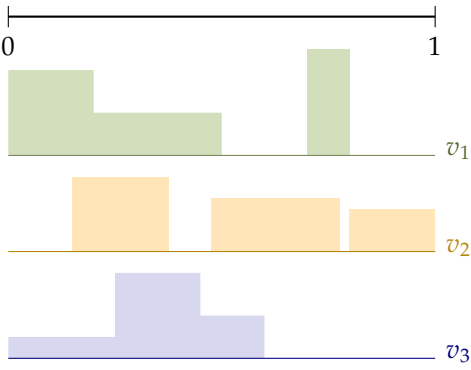
- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \{[a,b] \colon 0 \le a \le b \le 1\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$
  - v_i not assumed to be additive/monotone
- Focus on connected divisions of cake
- Division $A_1, \dots, A_n$ is envy-free if

$$v_i(A_i) \ge v_i(A_j) \qquad \forall i \ne j \in [n]$$

Cake Cutting



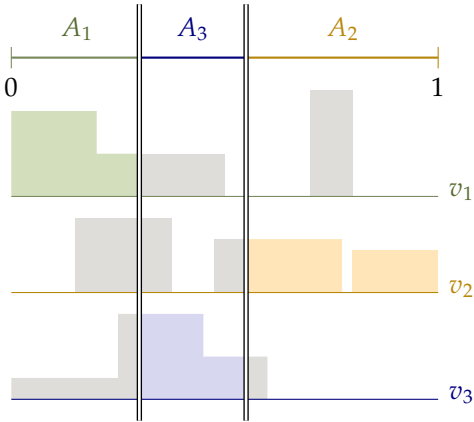
- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \{[a,b] \colon 0 \le a \le b \le 1\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$
  - v_i not assumed to be additive/monotone
- Focus on connected divisions of cake
- Division $A_1, \ldots, A_n$ is envy-free if

$$v_i(A_i) \ge v_i(A_j) \qquad \forall i \ne j \in [n]$$

Cake Cutting



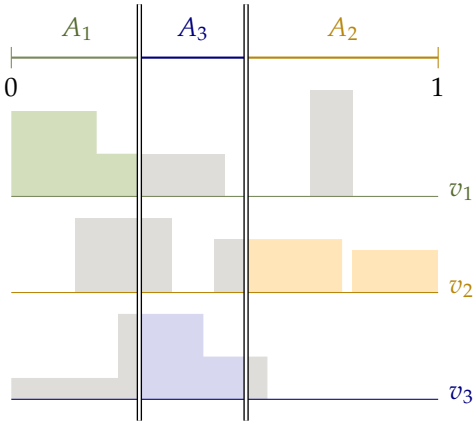
- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \{[a,b] \colon 0 \le a \le b \le 1\} \to \mathbb{R}_{\ge 0}$$

  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$
  - v_i not assumed to be additive/monotone
- Focus on connected divisions of cake
- Division $A_1, \ldots, A_n$ is envy-free if

$$v_i(A_i) \ge v_i(A_j) \qquad \forall i \ne j \in [n]$$

Cake Cutting



- n agents
- Each agent $i \in [n]$ has a valuation function

$$v_i \colon \{[a,b] \colon 0 \le a \le b \le 1\} \to \mathbb{R}_{\ge 0}$$

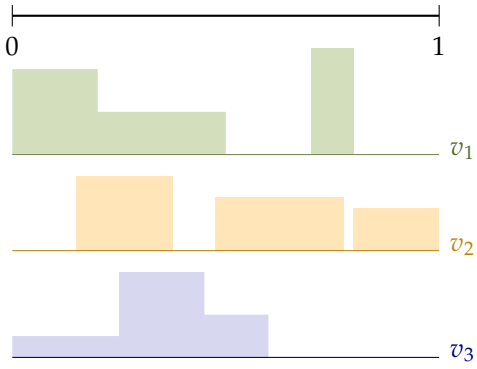
  - v_i is Lipschitz-continuous
  - $v_i(\emptyset) = 0$
  - v_i not assumed to be additive/monotone
- Focus on connected divisions of cake
- Division $A_1, \dots, A_n$ is envy-free if

$$v_i(A_i) \ge v_i(A_j) \qquad \forall i \ne j \in [n]$$

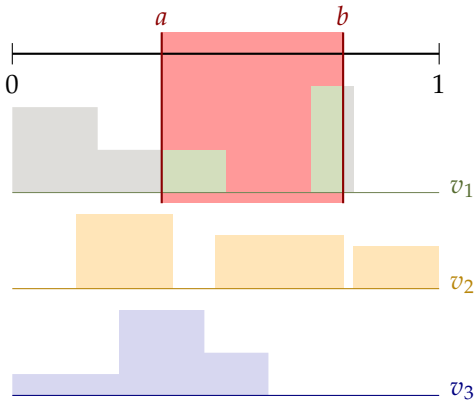
$\Rightarrow$ exists always (by fixed-point theorem) [Str80, Woo80]

Cake Cutting: Model of Computation

Query Model:



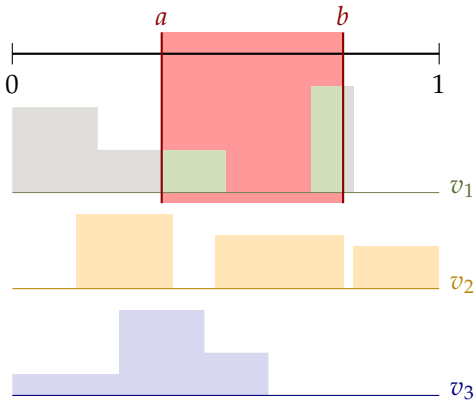
Cake Cutting: Model of Computation



Query Model:

- Queries: "Agent i how do you value interval $[a, b]$?"
  - Answer: $v_i([a, b])$

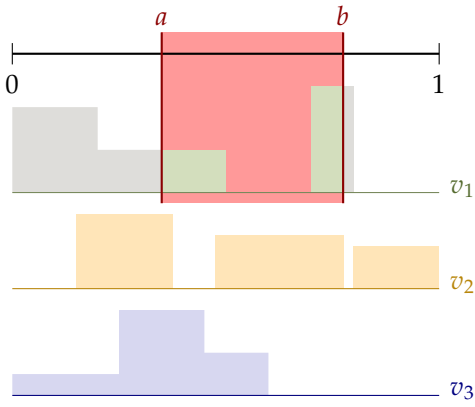
Cake Cutting: Model of Computation



Query Model:

- Queries: "Agent i how do you value interval $[a, b]$?"
  - Answer: $v_i([a, b])$
- Complexity: number of queries to find a solution

Cake Cutting: Model of Computation

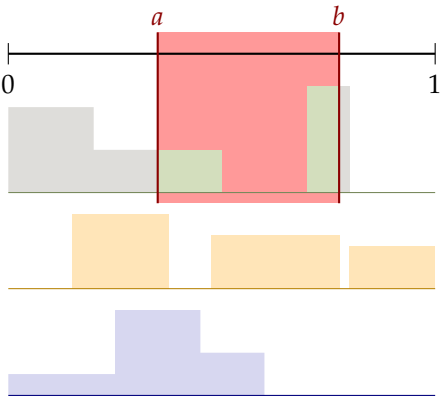


Query Model:

- Queries: "Agent i how do you value interval $[a, b]$?"
  - Answer: $v_i([a, b])$
- Complexity: number of queries to find a solution

$\Rightarrow$ Cannot find an exact envy-free solution! [Str08]

Cake Cutting: Model of Computation



Query Model:

- Queries: "Agent i how do you value interval $[a, b]$?"
  - Answer: $v_i([a, b])$
- Complexity: number of queries to find a solution

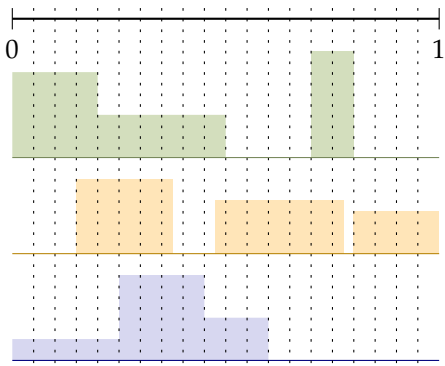
$\Rightarrow$ Cannot find an exact envy-free solution! [Str08]

Relax notion of envy-freeness:

- Division $A_1, \ldots, A_n$ is ε-envy-free if

$$v_i(A_i) \geq v_i(A_j) - \varepsilon \quad \forall i \neq j \in [n]$$

Cake Cutting: Model of Computation



Query Model:

- Queries: "Agent i how do you value interval $[a, b]$?"
  - Answer: $v_i([a, b])$
- Complexity: number of queries to find a solution

$\Rightarrow$ Cannot find an exact envy-free solution! [Str08]

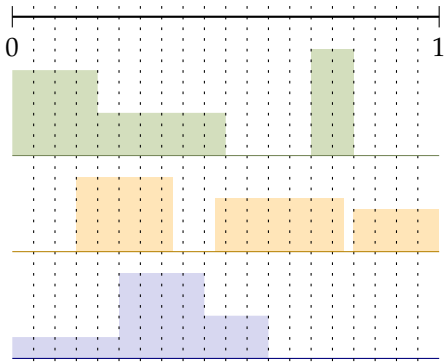
Relax notion of envy-freeness:

- Division $A_1, \ldots, A_n$ is ε-envy-free if

$$v_i(A_i) \geq v_i(A_j) - \varepsilon \quad \forall i \neq j \in [n]$$

  - Brute-force: $\text{poly}\, 1/\varepsilon$ queries

Cake Cutting: Model of Computation



Query Model:

- Queries: "Agent i how do you value interval $[a, b]$?"
  - Answer: $v_i([a, b])$
- Complexity: number of queries to find a solution

$\Rightarrow$ Cannot find an exact envy-free solution! [Str08]

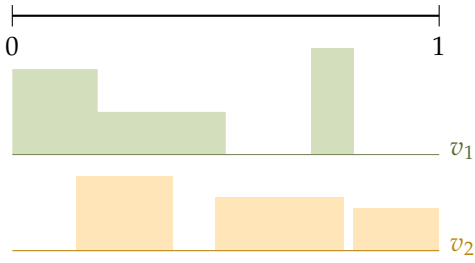
Relax notion of envy-freeness:

- Division $A_1, \ldots, A_n$ is ε-envy-free if

$$v_i(A_i) \geq v_i(A_j) - \varepsilon \quad \forall i \neq j \in [n]$$

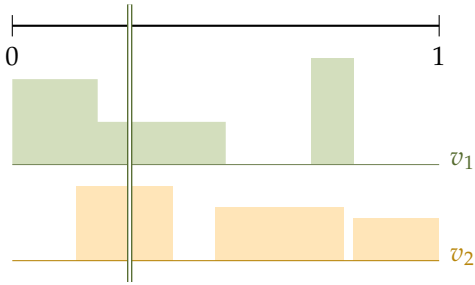
  - Brute-force: $\text{poly}\, 1/\varepsilon$ queries
  - Do $\text{polylog}\, 1/\varepsilon$ queries suffice?

Envy-Free Cake Cutting: Two Agents



Cut-and-Choose procedure:

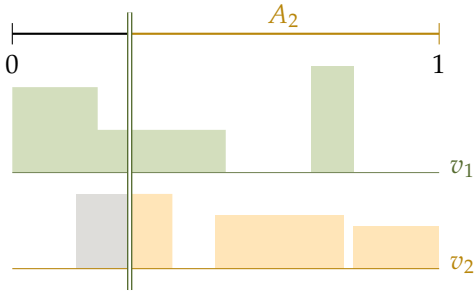
Envy-Free Cake Cutting: Two Agents



Cut-and-Choose procedure:

1. Agent 1 cuts the cake in half (according to v_1)

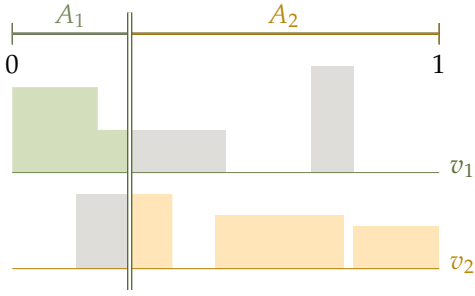
Envy-Free Cake Cutting: Two Agents



Cut-and-Choose procedure:

1. Agent 1 cuts the cake in half (according to v_1)
2. Agent 2 picks its favourite piece

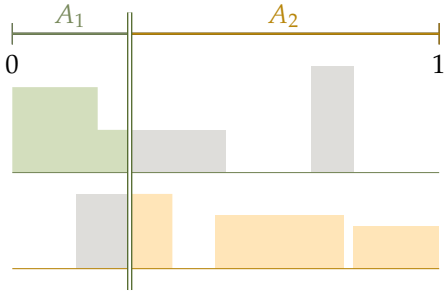
Envy-Free Cake Cutting: Two Agents



Cut-and-Choose procedure:

1. Agent 1 cuts the cake in half (according to v_1)
2. Agent 2 picks its favourite piece
3. Agent 1 gets the remaining piece

Envy-Free Cake Cutting: Two Agents

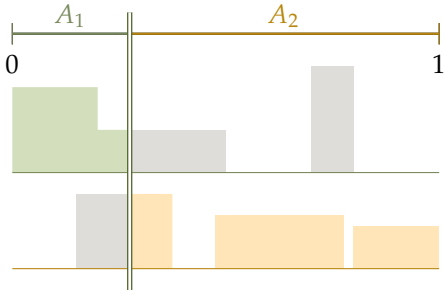


Cut-and-Choose procedure:

1. Agent 1 cuts the cake in half (according to v_1)
2. Agent 2 picks its favourite piece
3. Agent 1 gets the remaining piece

- Efficient algorithm for two agents

Envy-Free Cake Cutting: Two Agents



Cut-and-Choose procedure:

1. Agent 1 cuts the cake in half (according to v_1)
2. Agent 2 picks its favourite piece
3. Agent 1 gets the remaining piece

- Efficient algorithm for two agents
- Works for general valuations

Envy-Free Cake Cutting: Query Complexity

valuations	2 Agents	3 Agents	4 Agents	$n \geq 5$ Agents
monotone	$O(\log 1/\varepsilon)$			
general	$O(\log 1/\varepsilon)$			

Envy-Free Cake Cutting: Query Complexity

valuations	2 Agents	3 Agents	4 Agents	$n \geq 5$ Agents
monotone	$O(\log 1/\varepsilon)$	$O(\log^2 1/\varepsilon)$	$O(\log^3 1/\varepsilon)$	?
general	$O(\log 1/\varepsilon)$			

Envy-Free Cake Cutting: Query Complexity

valuations	2 Agents	3 Agents	4 Agents	$n \geq 5$ Agents
monotone	$O(\log 1/\varepsilon)$	$O(\log^2 1/\varepsilon)$	$O(\log^3 1/\varepsilon)$	?
general	$O(\log 1/\varepsilon)$	?	$\Theta(\text{poly}\, 1/\varepsilon)$	$\Theta(\text{poly}\, 1/\varepsilon)$

Envy-Free Cake Cutting: Query Complexity

valuations	2 Agents	3 Agents	4 Agents	$n \geq 5$ Agents
monotone	$O(\log 1/\varepsilon)$	$O(\log^2 1/\varepsilon)$	$O(\log^3 1/\varepsilon)$	?
general	$O(\log 1/\varepsilon)$	$\Theta(\text{poly}\, 1/\varepsilon)$	$\Theta(\text{poly}\, 1/\varepsilon)$	$\Theta(\text{poly}\, 1/\varepsilon)$

Proof Overview

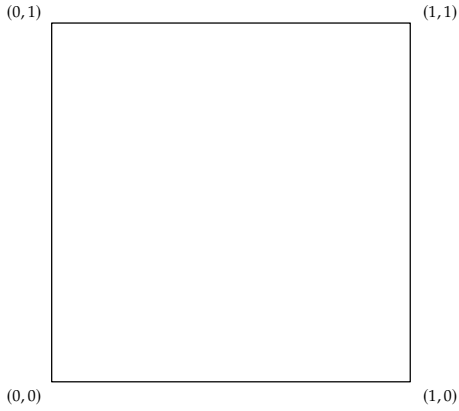
Theorem Cake cutting with 3 agents is at least as hard as the Jordan curve problem.

Proof Overview

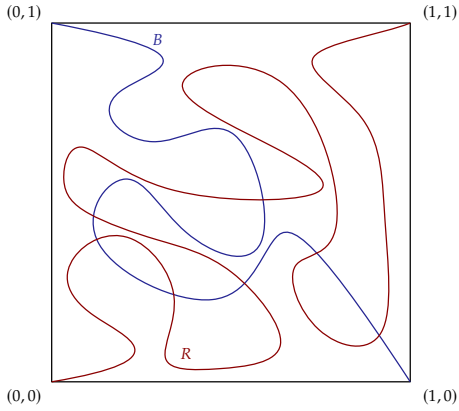
Theorem Cake cutting with 3 agents is at least as hard as the Jordan curve problem.

Theorem The Jordan curve problem requires $\Omega(\operatorname{poly} 1/\varepsilon)$ queries.

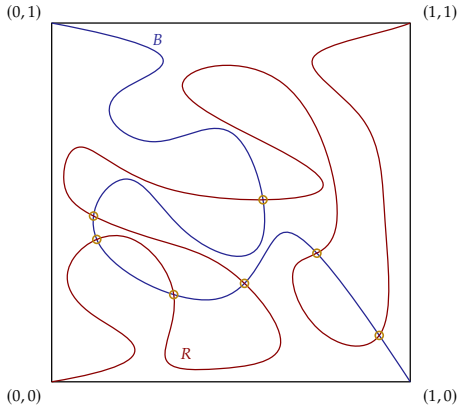
Jordan Curve



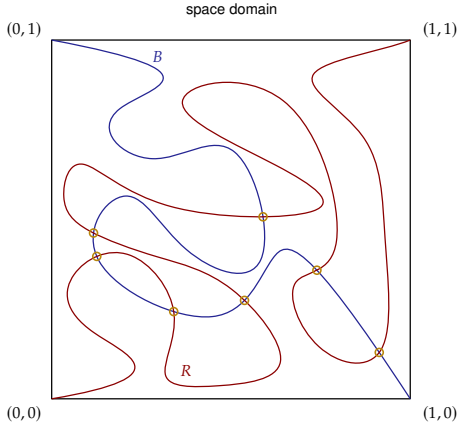
Jordan Curve

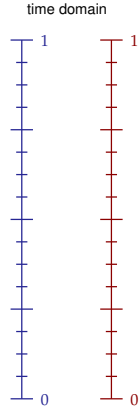


Jordan Curve



Jordan Curve

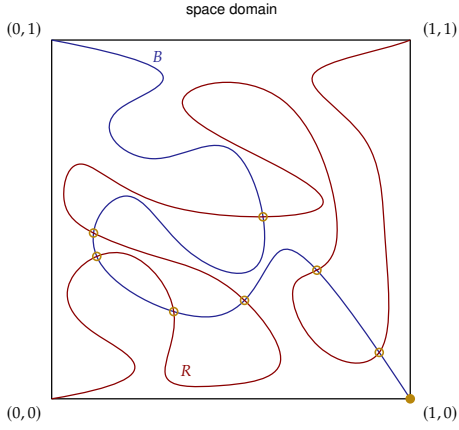


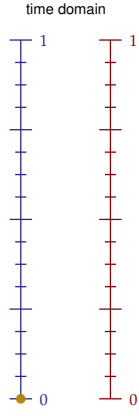


$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Jordan Curve

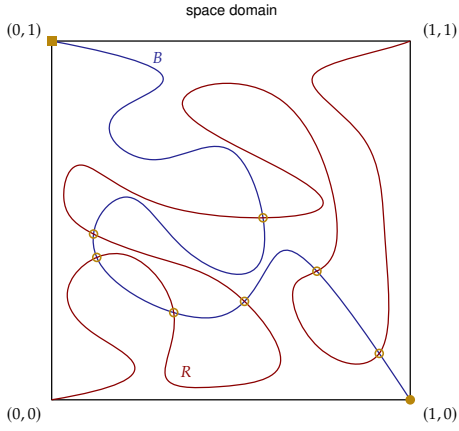


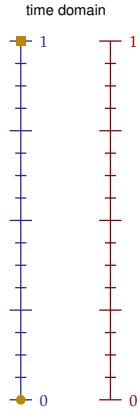


$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Jordan Curve

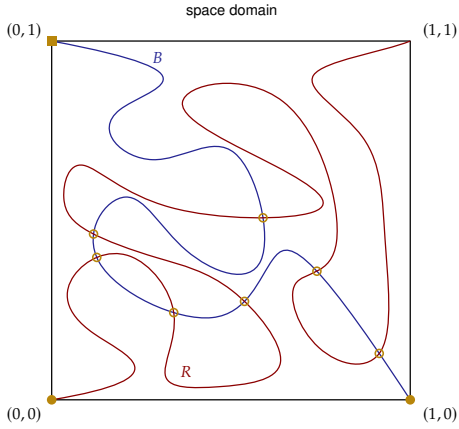


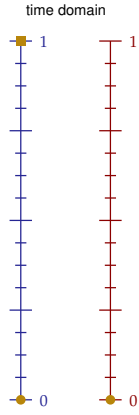


$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Jordan Curve

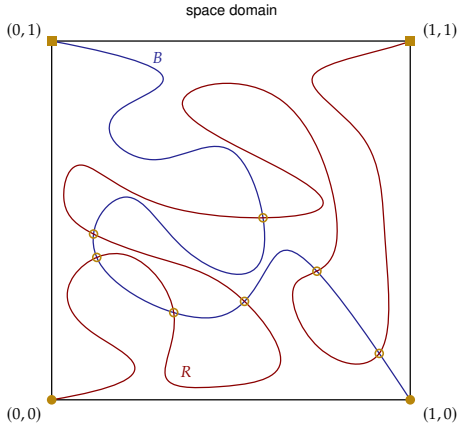


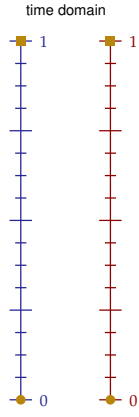


$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Jordan Curve

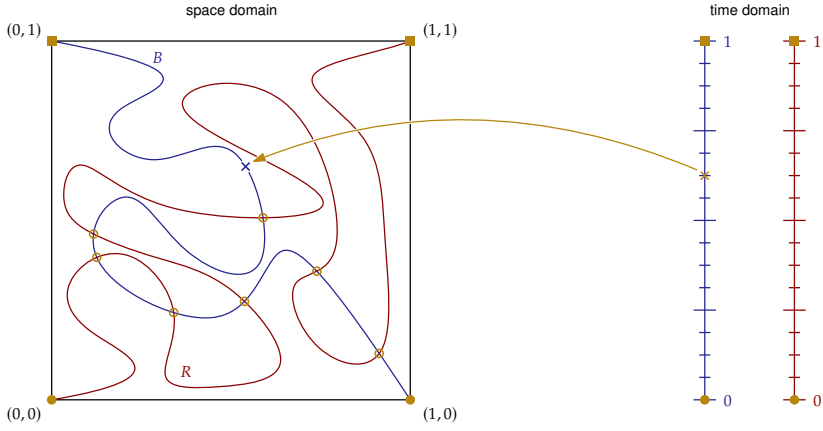




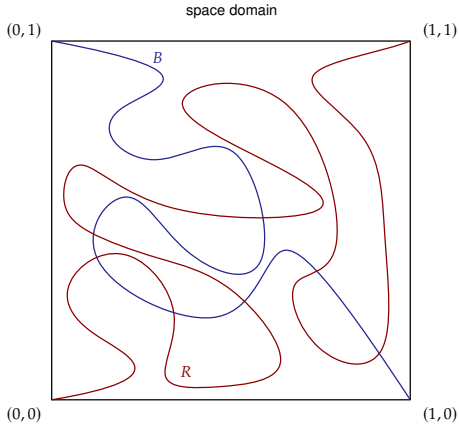
$B, R\colon [0,1] \to [0,1]^2$

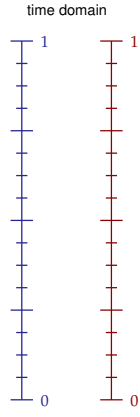
Lipschitz-continuous

Jordan Curve



Jordan Curve

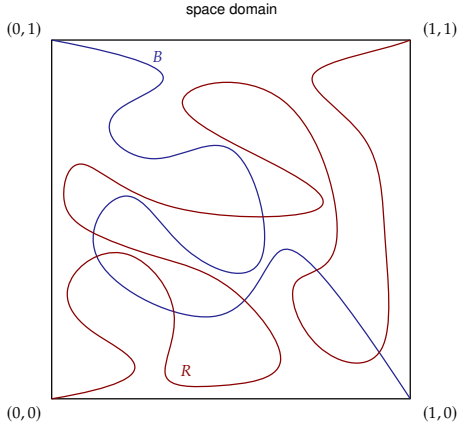


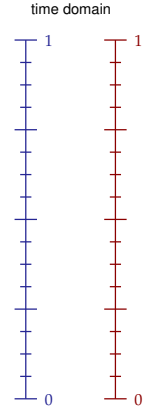


$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Jordan Curve



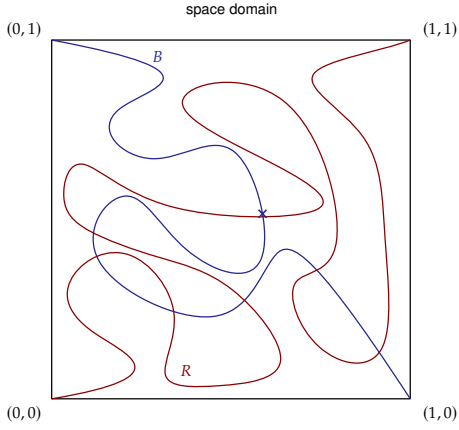


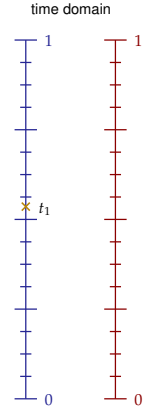
$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Approximate Solution: Timestamps $t_1, t_2 \in [0,1]$ such that $\|B(t_1) - R(t_2)\|_\infty \leq \varepsilon$.

Jordan Curve



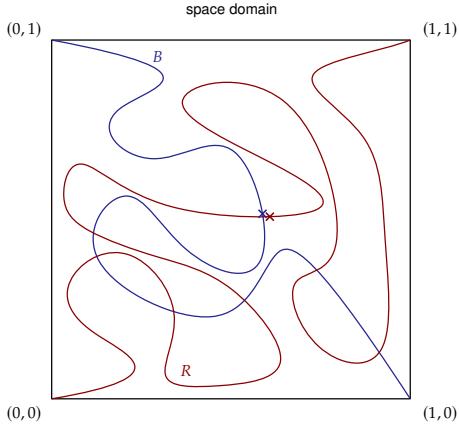


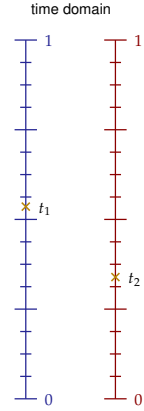
$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Approximate Solution: Timestamps $t_1, t_2 \in [0,1]$ such that $\|B(t_1) - R(t_2)\|_\infty \leq \varepsilon$.

Jordan Curve

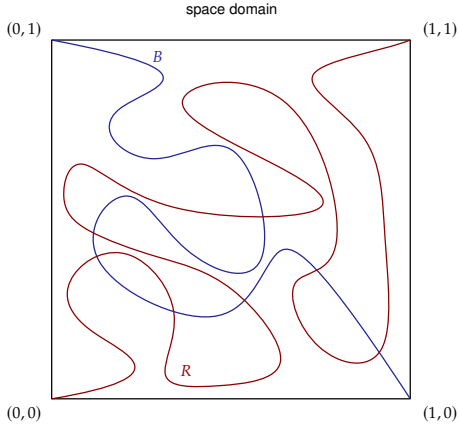


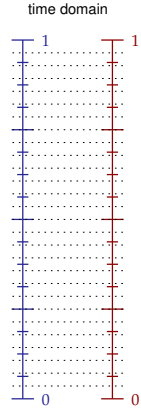


$B, R\colon [0,1] \to [0,1]^2$
Lipschitz-continuous

Approximate Solution: Timestamps $t_1, t_2 \in [0,1]$ such that $\|B(t_1) - R(t_2)\|_\infty \leq \varepsilon$.

Jordan Curve





$B, R\colon [0,1] \to [0,1]^2$

Lipschitz-continuous

Approximate Solution: Timestamps $t_1, t_2 \in [0,1]$ such that $\|B(t_1) - R(t_2)\|_\infty \leq \varepsilon$.

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.

- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.

- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.

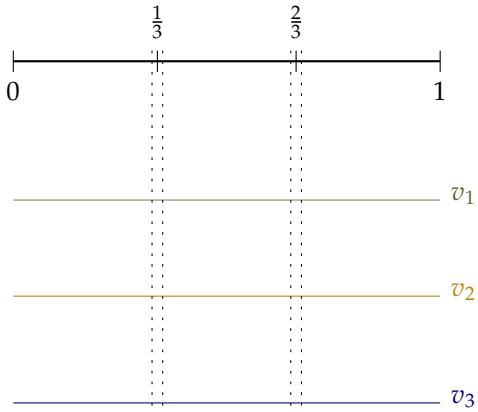
- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



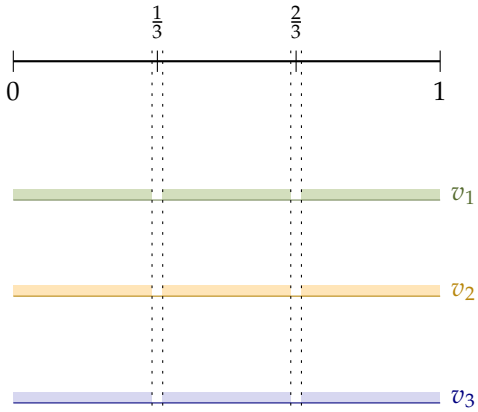
- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



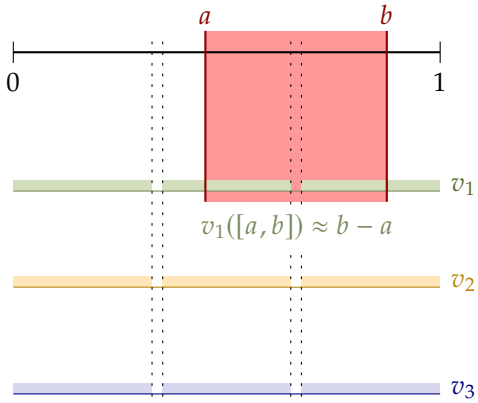
- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



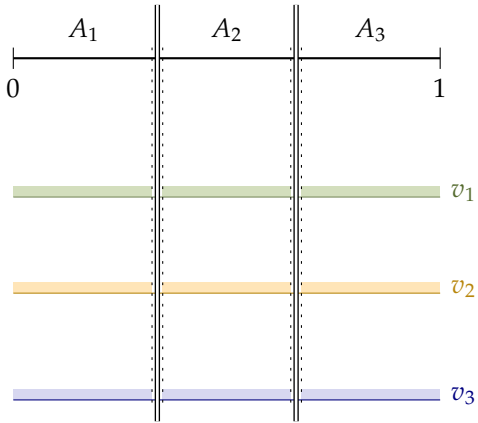
- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



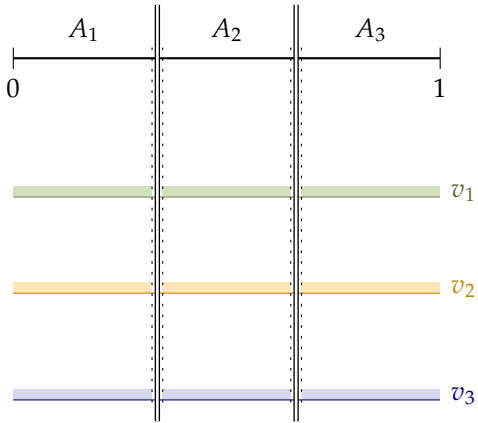
- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



- Approximate Jordan curve solution satisfies

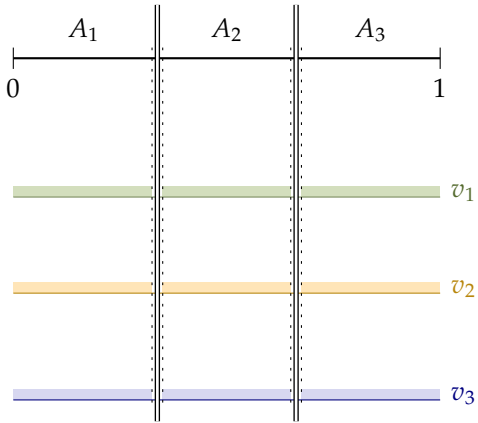
$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

- for cuts (t_1, t_2) close to $(1/3, 2/3)$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



- Approximate Jordan curve solution satisfies

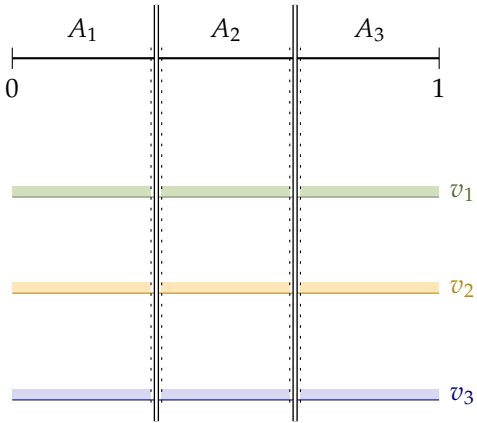
$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

- for cuts (t_1, t_2) close to $(1/3, 2/3)$
  - $A_1 = [0, t_1]$ has value $v(A_1) = B_x(t_1) + B_y(t_1)$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



- Approximate Jordan curve solution satisfies

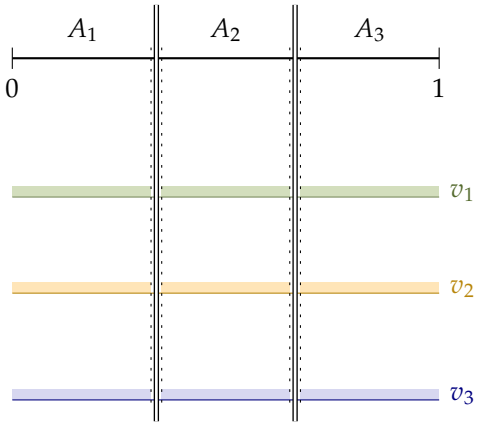
$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

- for cuts (t_1, t_2) close to $(1/3, 2/3)$
  - $A_1 = [0, t_1]$ has value $v(A_1) = B_x(t_1) + B_y(t_1)$
  - $A_2 = [t_1, t_2]$ has value $v(A_2) = R_x(t_2) + B_y(t_1)$

From Jordan Curve to Cake Cutting

Theorem Cake cutting with 3-agent is at least as hard as the Jordan curve problem.



- Approximate Jordan curve solution satisfies

$$B_x(t_1) \approx R_x(t_2) \quad \text{and} \quad B_y(t_1) \approx R_y(t_2)$$

$$\Leftrightarrow \; B_x(t_1)+B_y(t_1) \approx R_x(t_2)+B_y(t_1) \approx R_x(t_2)+R_y(t_2)$$

- for cuts (t_1, t_2) close to $(1/3, 2/3)$
  - $A_1 = [0, t_1]$ has value $v(A_1) = B_x(t_1) + B_y(t_1)$
  - $A_2 = [t_1, t_2]$ has value $v(A_2) = R_x(t_2) + B_y(t_1)$
  - $A_3 = [t_2, 1]$ has value $v(A_3) = R_x(t_2) + R_y(t_2)$

Proof Overview

Theorem Cake cutting with 3 agents is at least as hard as the Jordan curve problem.

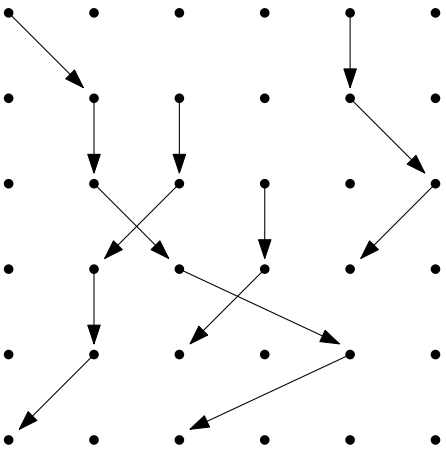
Theorem The Jordan curve problem requires $\Omega(\operatorname{poly} 1/\varepsilon)$ queries.

Proof Overview

Theorem Cake cutting with 3 agents is at least as hard as the Jordan curve problem.

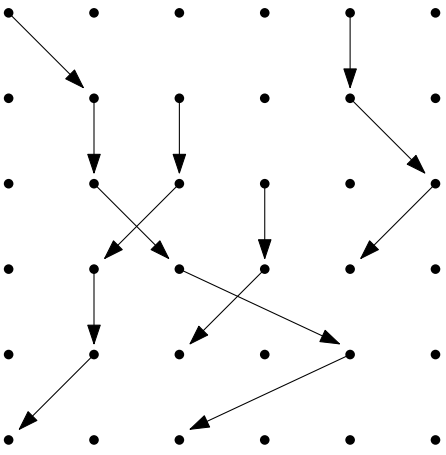
Theorem The Jordan curve problem is at least as hard as the UEoPL problem

Unique End of Potential Line



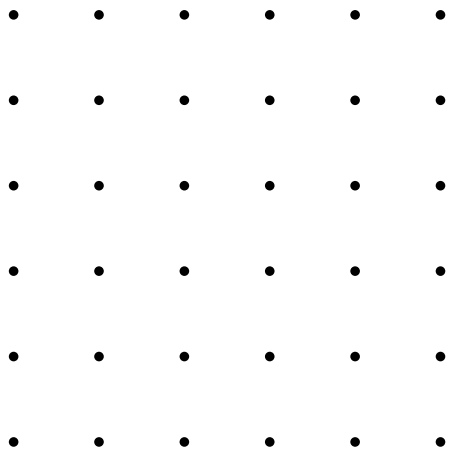
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1

Unique End of Potential Line



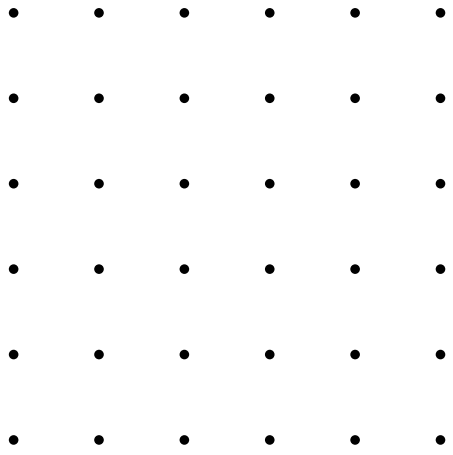
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:

Unique End of Potential Line



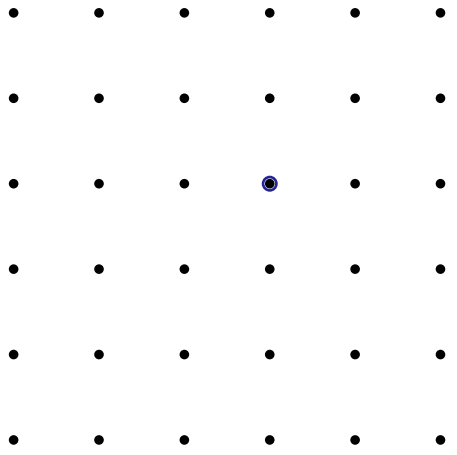
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:

Unique End of Potential Line



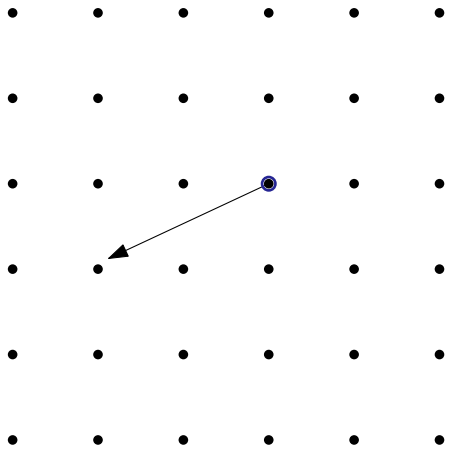
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"

Unique End of Potential Line



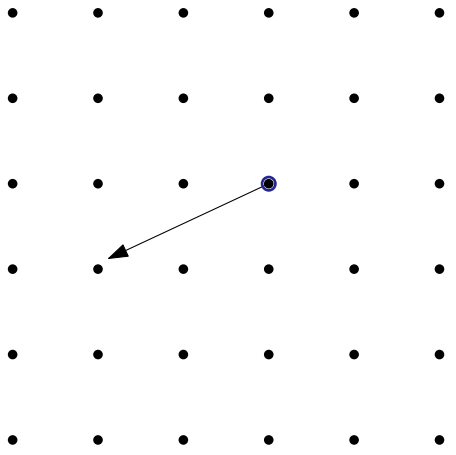
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"

Unique End of Potential Line



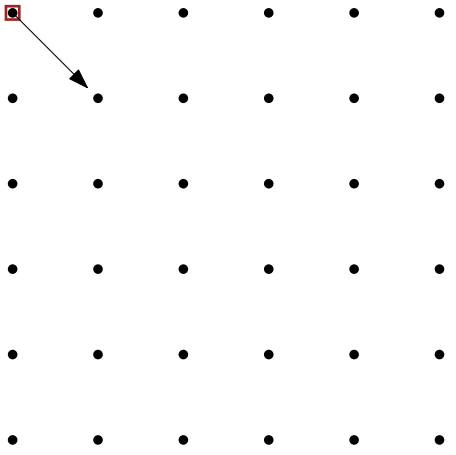
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"

Unique End of Potential Line



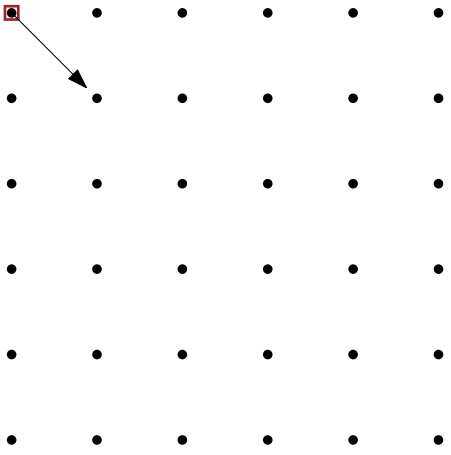
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour

Unique End of Potential Line



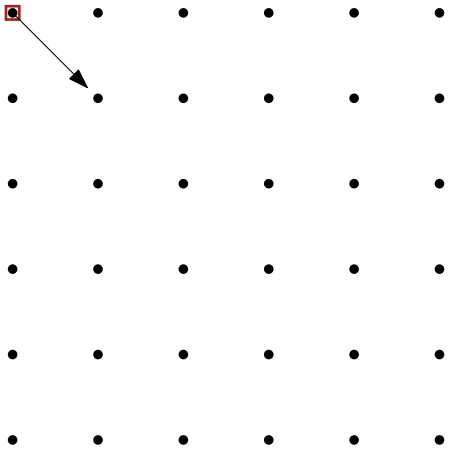
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour

Unique End of Potential Line



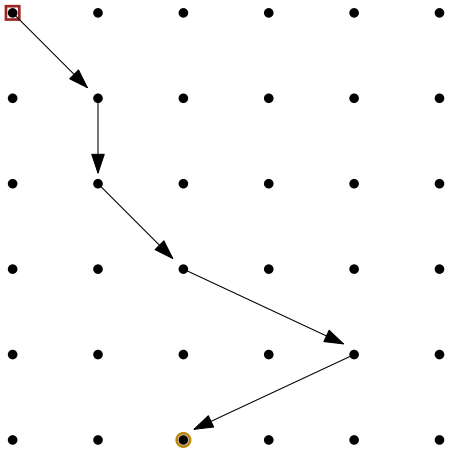
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:

Unique End of Potential Line



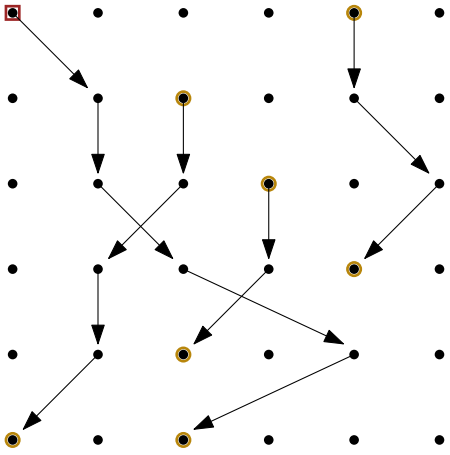
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree

Unique End of Potential Line



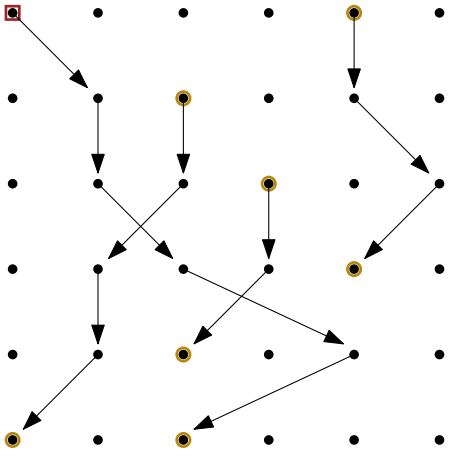
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree

Unique End of Potential Line



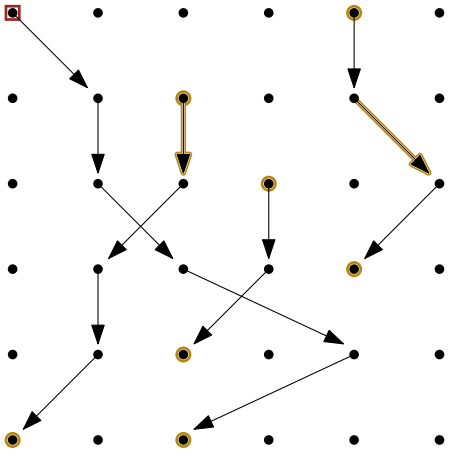
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree

Unique End of Potential Line



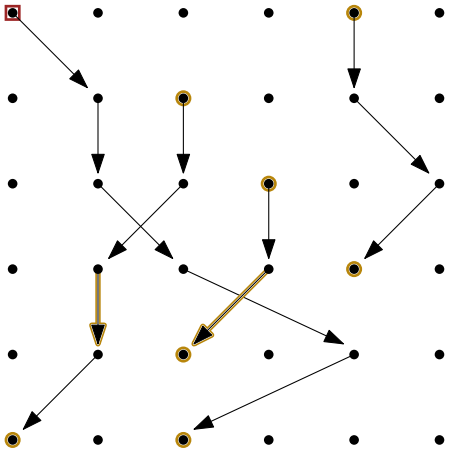
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree
  - two "parallel" edges

Unique End of Potential Line



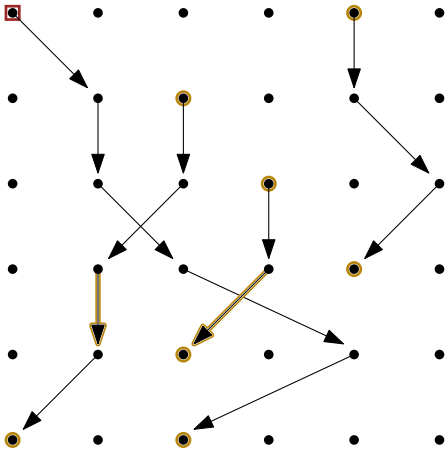
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree
  - two "parallel" edges

Unique End of Potential Line



- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree
  - two "parallel" edges

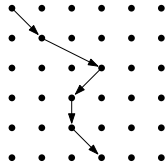
Unique End of Potential Line



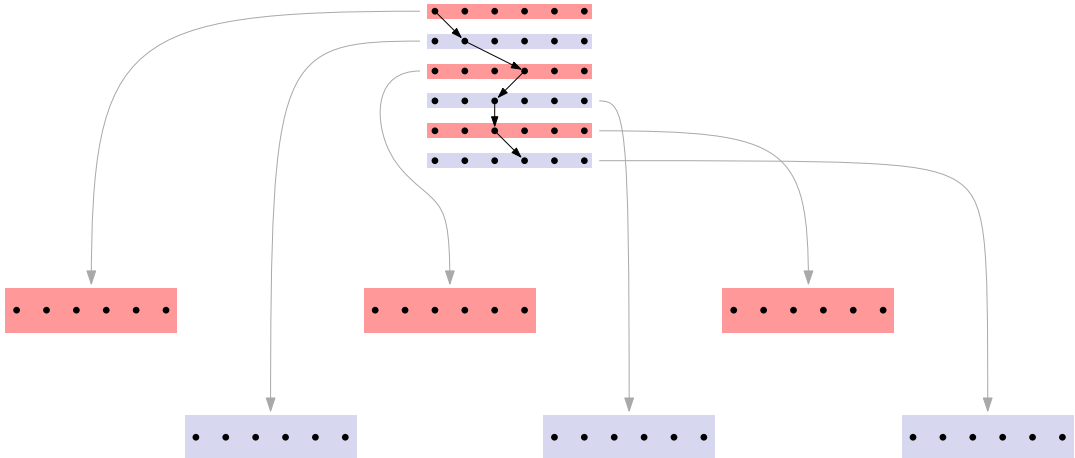
- $[n] \times [n]$ grid, directed downward edges
- Each node has in-, out-degree ≤ 1
- Query access to graph:
  - "Who is your in-/out-neighbour?"
- Start node with out-neighbour
- Solutions:
  - node with in-degree $\neq$ out-degree
  - two "parallel" edges

Theorem It requires $\Omega(n^{\varepsilon})$ queries to find a solution.

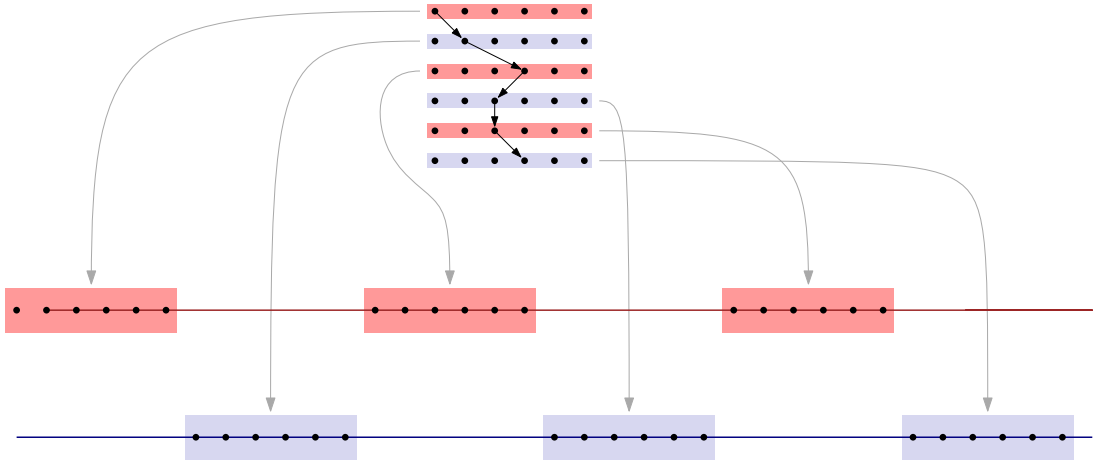
From Unique End of Potential Line to Jordan Curve



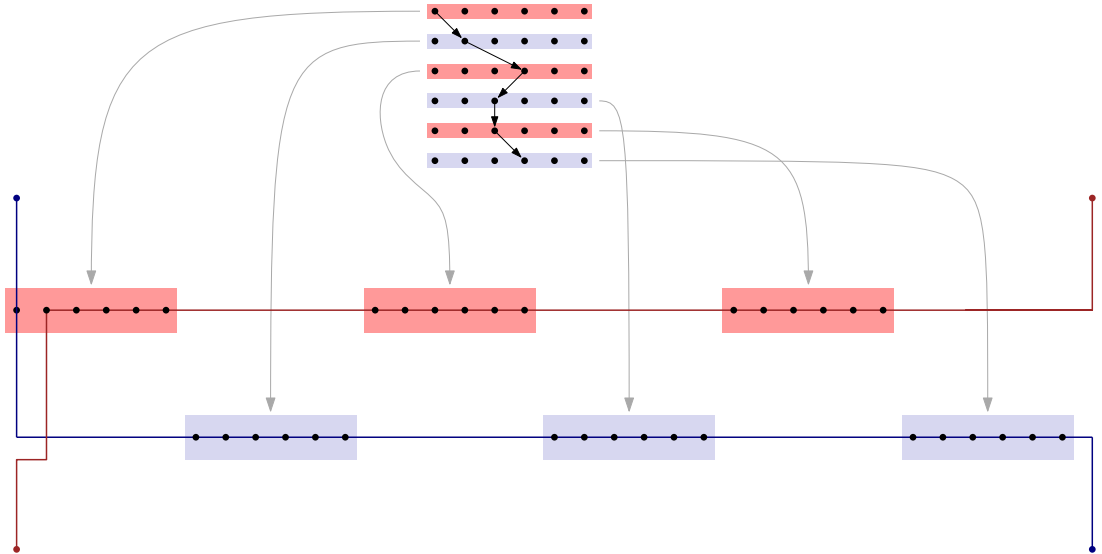
From Unique End of Potential Line to Jordan Curve



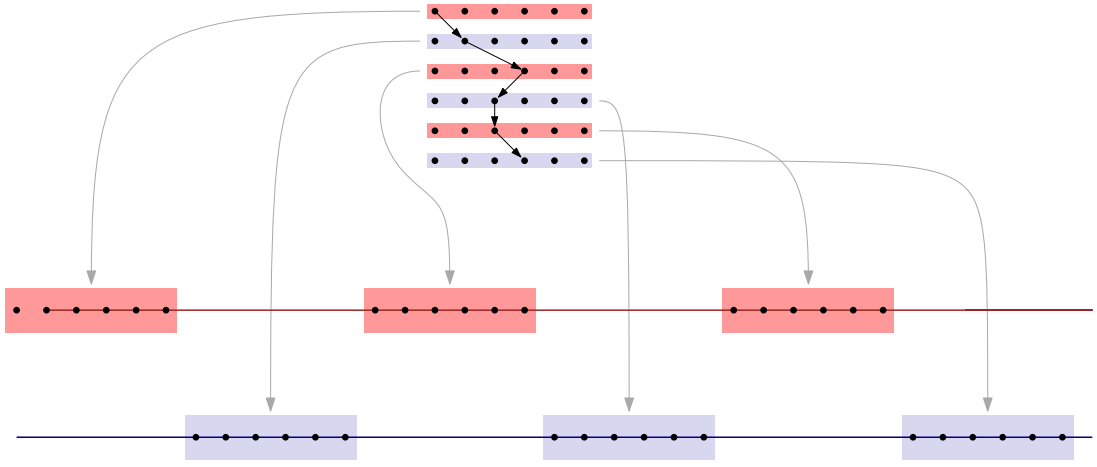
From Unique End of Potential Line to Jordan Curve



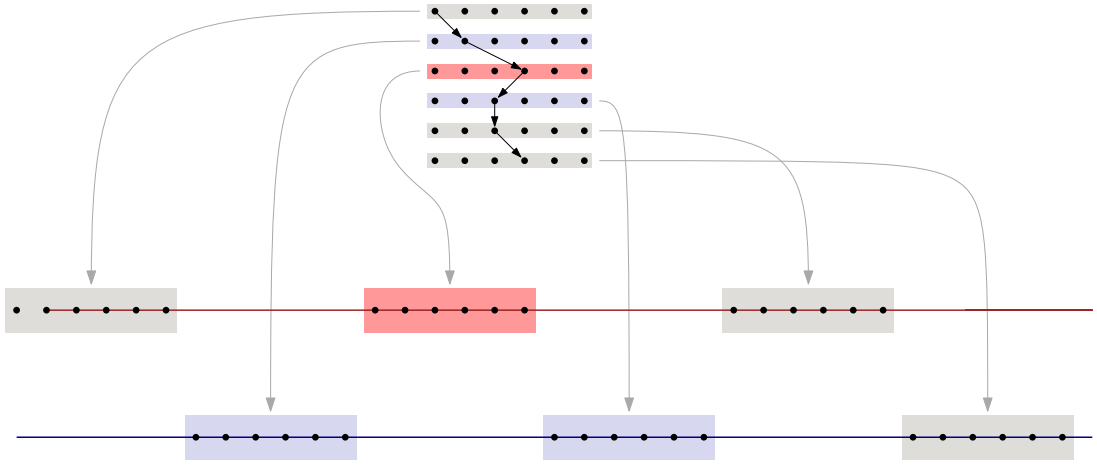
From Unique End of Potential Line to Jordan Curve



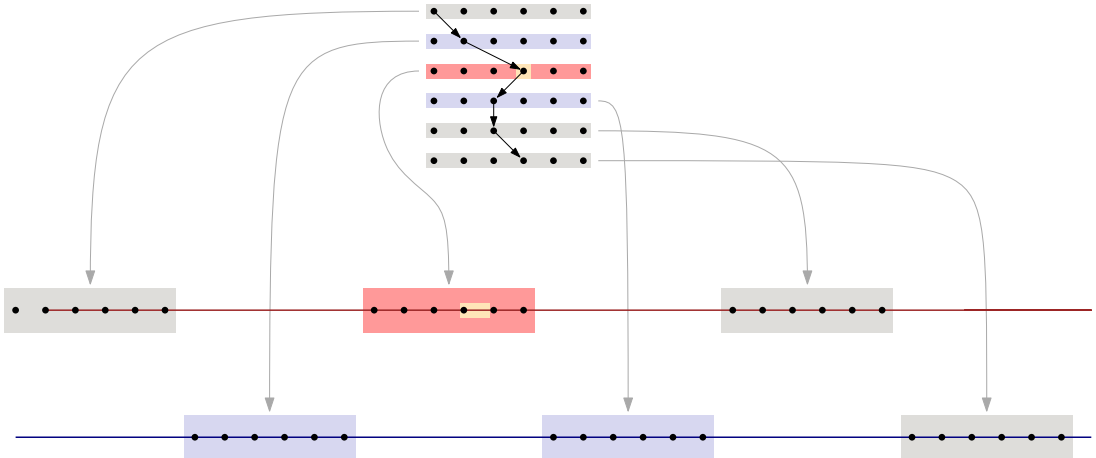
From Unique End of Potential Line to Jordan Curve



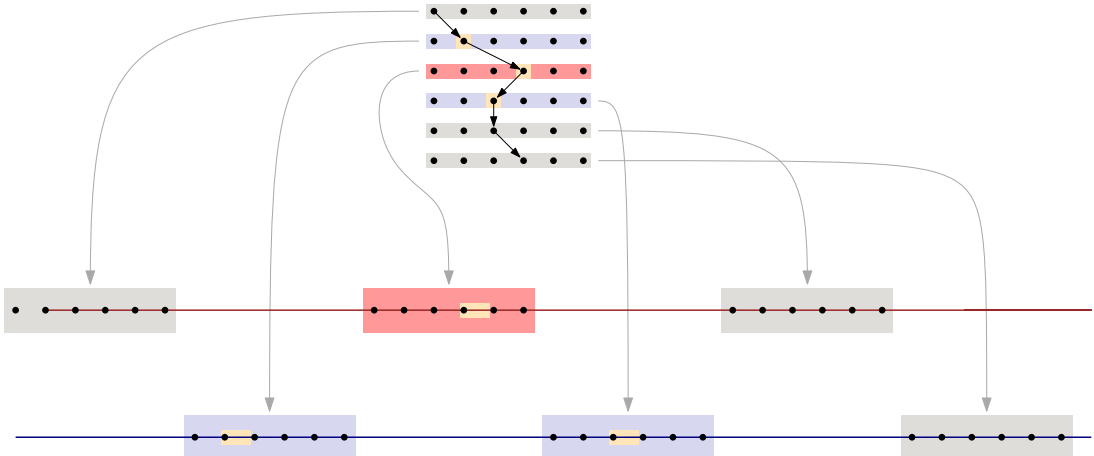
From Unique End of Potential Line to Jordan Curve



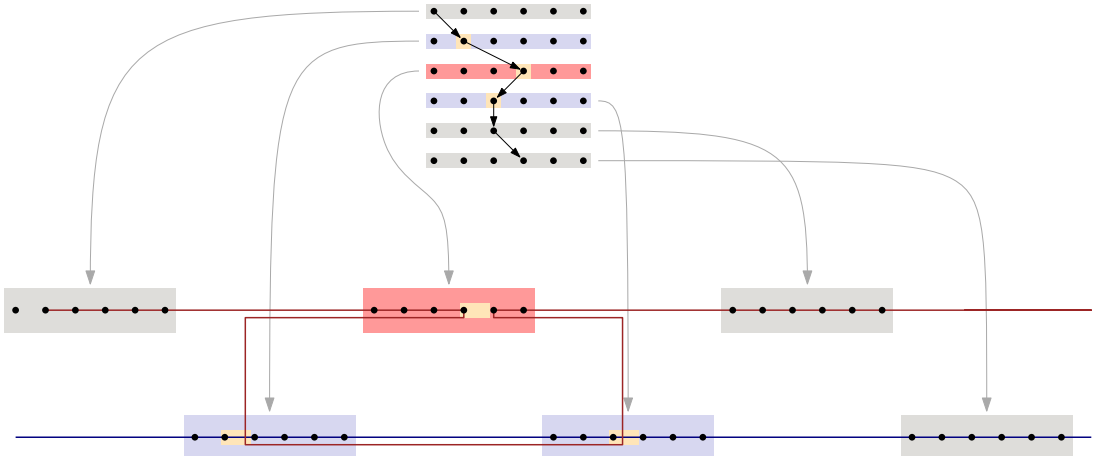
From Unique End of Potential Line to Jordan Curve



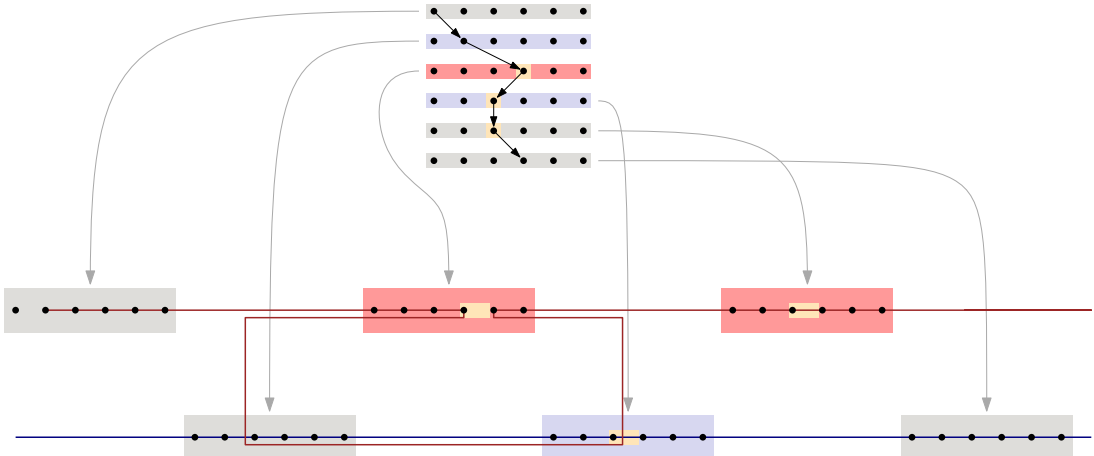
From Unique End of Potential Line to Jordan Curve



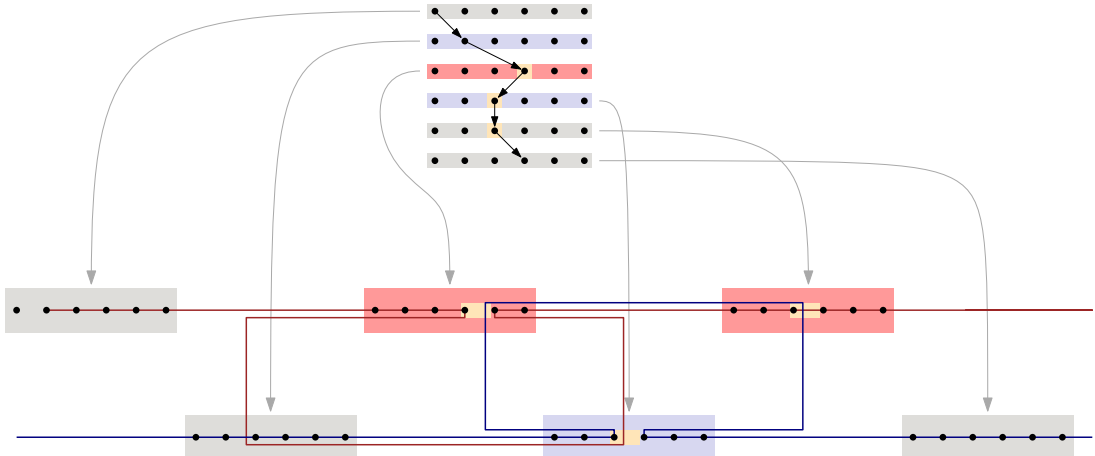
From Unique End of Potential Line to Jordan Curve



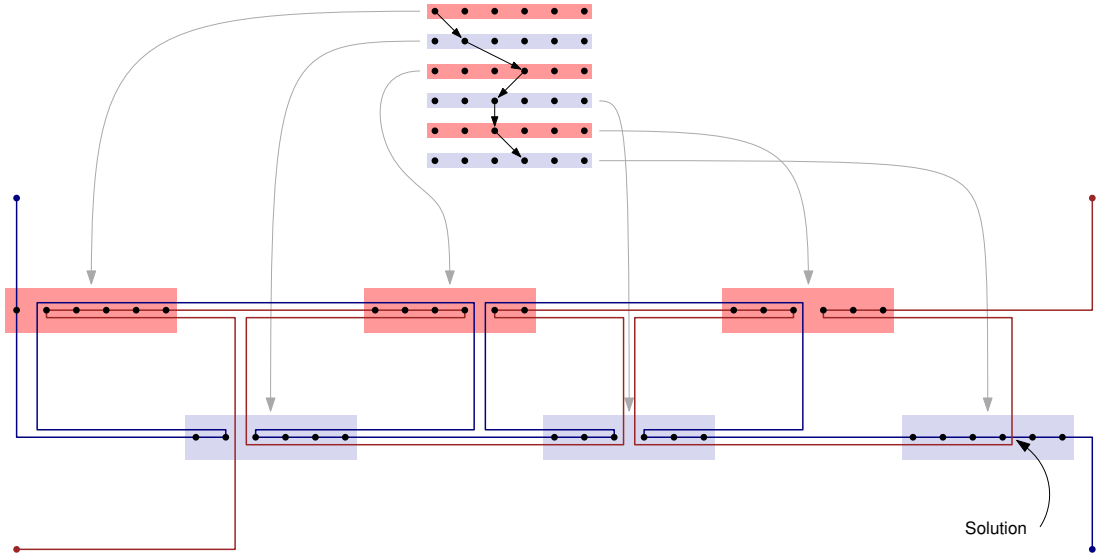
From Unique End of Potential Line to Jordan Curve



From Unique End of Potential Line to Jordan Curve



From Unique End of Potential Line to Jordan Curve



Summary

Theorem Cake cutting with 3 agents and general valuations requires $\Omega(\text{poly}\, 1/\varepsilon)$ queries.

- By reduction: cake cutting ← Jordan curve ← unique end of potential line

Open problems

- Communication complexity of 3 agent cake cutting with general valuations?
- Is there an efficient PLS algorithm for Jordan curve?
- Is 3 agent cake cutting PPAD-complete?
- Query complexity of 5 agent cake cutting with monotone valuations?