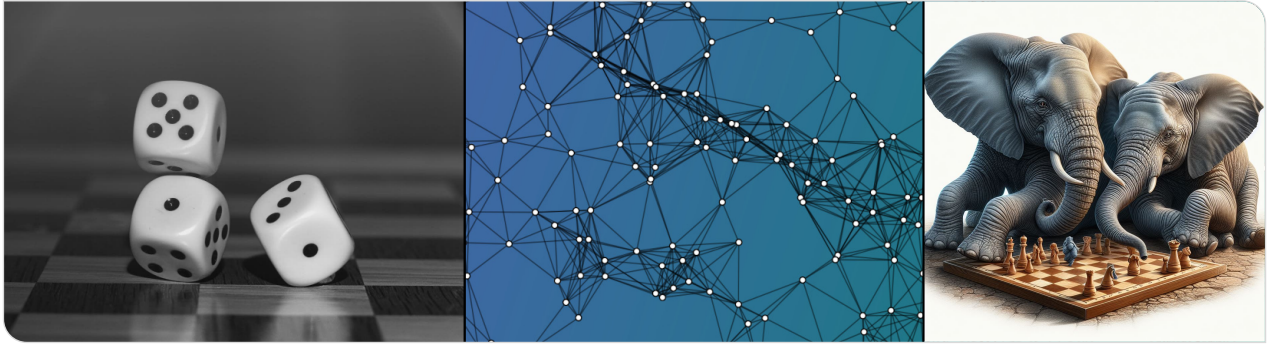
Karlsruhe Institute of Technology

Algorithm Engineering

Probability and Computing – Game Theory & Yao's Principle

Stefan Walzer | WS 2024/2025



Lecture Notes by Worsch

Some of this lecture's content is covered in Thomas Worsch's notes from 2019. 

Content

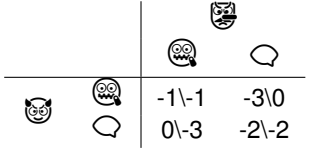
Content

Content

Prisoner's Dilemma

		-1\-1	-3\0
		0\-3	-2\-2

Prisoner's Dilemma



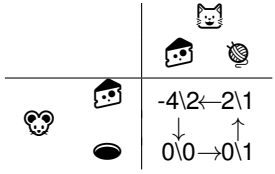
Setting

- *strategies* and available to both *players*
- table shows *payoffs* for players depending on chosen strategies
- here: always better to choose
  ↪ pair (,) is unique *equilibrium*

Definition: Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



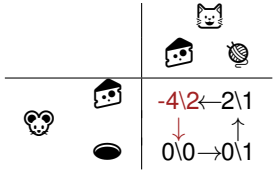
Someone always regrets their decision

		reaction

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



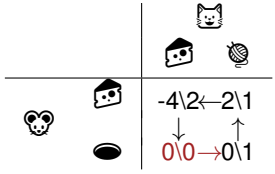
Someone always regrets their decision

		reaction
		should have played

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



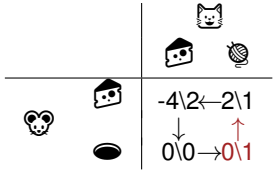
Someone always regrets their decision

		reaction
		should have played
		should have played

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



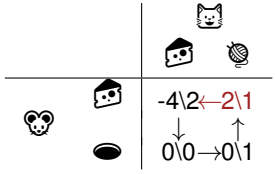
Someone always regrets their decision

		reaction
cheese	cheese	mouse should have played yarn
yarn	cheese	cat should have played yarn
yarn	yarn	mouse should have played cheese

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



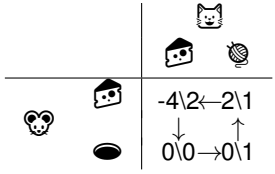
Someone always regrets their decision

		reaction
		should have played
		should have played
		should have played
		should have played

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

A cat and mouse game



Someone always regrets their decision

		reaction
cheese	cheese	mouse should have played yarn
yarn	cheese	cat should have played yarn
yarn	yarn	mouse should have played cheese
cheese	yarn	cat should have played cheese

↪ No combination of *pure* strategies is an *equilibrium*.

Equilibrium

Combination of strategies such that no one can profit by unilaterally switching his or her own strategy.

Nash Equilibria

What a Game *is*

- Finite sets S_1, S_2 of *pure strategies*.
- Utility functions $u_1, u_2 : S_1 \times S_2 \to \mathbb{R}$.

Nash Equilibria

What a Game *is*

- Finite sets S_1, S_2 of *pure strategies*.
- Utility functions $u_1, u_2 : S_1 \times S_2 \to \mathbb{R}$.

How a Game is played

- Players pick a strategy simultaneously
  $\hookrightarrow$ gives pair $(s_1, s_2) \in S_1 \times S_2$.
- player 1 gets payoff $u_1(s_1, s_2)$ and
  player 2 gets payoff $u_2(s_1, s_2)$.

Nash Equilibria

What a Game *is*

- Finite sets S_1, S_2 of *pure strategies*.
- Utility functions $u_1, u_2 : S_1 \times S_2 \to \mathbb{R}$.

How a Game is played

- Players pick a strategy simultaneously
  $\hookrightarrow$ gives pair $(s_1, s_2) \in S_1 \times S_2$.
- player 1 gets payoff $u_1(s_1, s_2)$ and
  player 2 gets payoff $u_2(s_1, s_2)$.

Existence of Mixed-Strategy Nash Equilibria

There exist distributions $\mathcal{S}_1^*$ on S_1 and $\mathcal{S}_2^*$ on S_2, called *mixed strategies* such that $(\mathcal{S}_1^*, \mathcal{S}_2^*)$ is an equilibrium:

player 1 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[u_1(s_1, s_2)] = \max_{s_1 \in S_1} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[u_1(s_1, s_2)]$.

player 2 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[u_2(s_1, s_2)] = \max_{s_2 \in S_2} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[u_2(s_1, s_2)]$.

Nash Equilibria

Karlsruhe Institute of Technology

What a Game *is*

- Finite sets S_1, S_2 of *pure strategies*.
- Utility functions $u_1, u_2 : S_1 \times S_2 \to \mathbb{R}$.

How a Game is played

- Players pick a strategy simultaneously
  $\hookrightarrow$ gives pair $(s_1, s_2) \in S_1 \times S_2$.
- player 1 gets payoff $u_1(s_1, s_2)$ and
  player 2 gets payoff $u_2(s_1, s_2)$.

Existence of Mixed-Strategy Nash Equilibria

There exist distributions $\mathcal{S}_1^*$ on S_1 and $\mathcal{S}_2^*$ on S_2, called *mixed strategies* such that $(\mathcal{S}_1^*, \mathcal{S}_2^*)$ is an equilibrium:

player 1 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[u_1(s_1, s_2)] = \max_{s_1 \in S_1} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[u_1(s_1, s_2)]$.

player 2 cannot increase expected payoff: $\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[u_2(s_1, s_2)] = \max_{s_2 \in S_2} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[u_2(s_1, s_2)]$.

Remark: Theorem holds for $n \geq 3$ players as well.

Nash Equilibrium in Cat & Mouse Game

		-4\2	2\1
		0\0	0\1

Equilibrium

$\mathcal{S}_{\text{mouse}} = \{ : \frac{1}{2}, : \frac{1}{2}\}$

$\mathcal{S}_{\text{cat}} = \{ : \frac{1}{3}, : \frac{2}{3}\}$

Nash Equilibrium in Cat & Mouse Game

		-4\2	2\1
		0\0	0\1

Equilibrium

$\mathcal{S}_{\text{mouse}} = \{ : \frac{1}{2}, : \frac{1}{2}\}$

$\mathcal{S}_{\text{cat}} = \{ : \frac{1}{3}, : \frac{2}{3}\}$

Verification of Equilibrium Property: Calculating Expected Payoffs

for :

- playing gives expected payoff $\frac{1}{3} \cdot (-4) + \frac{2}{3} \cdot 2 = 0$
- playing gives expected payoff $\frac{1}{3} \cdot 0 + \frac{2}{3} \cdot 0 = 0$
- playing $\mathcal{S}_{\text{mouse}}$ is a mix of both
  $\hookrightarrow$ also expected payoff 0.

for :

- playing gives expected payoff $\frac{1}{2} \cdot 2 + \frac{1}{2} \cdot 0 = 1$
- playing gives expected payoff $\frac{1}{2} \cdot 1 + \frac{1}{2} \cdot 1 = 1$
- playing $\mathcal{S}_{\text{cat}}$ is a mix of both
  $\hookrightarrow$ also expected payoff 1.

Content

Two Player Zero Sum Games

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$
- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

		0	-1	1
		1	0	-1
		-1	1	0

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

		0	-1	1
		1	0	-1
		-1	1	0

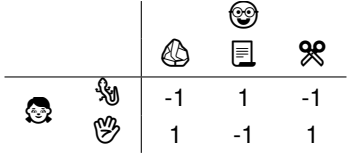
Unique equilibrium of 

$\mathcal{S}_1 = \mathcal{S}_2 = \{$  $: \frac{1}{3},$  $: \frac{1}{3},$  $: \frac{1}{3}\}$

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$

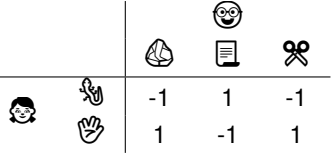


		-1	1	-1
		1	-1	1

Two Player Zero Sum Games and their Matrix Formulation

- Finite sets of pure strategies
  - S_1 for player 1
  - S_2 for player 2
- utility function $u : S_1 \times S_2 \to \mathbb{R}$
  - player 1 gets $u(s_1, s_2)$
  - player 2 gets $-u(s_1, s_2)$

- Implicit sets of pure strategies
  - $S_1 = [n]$ for the *row player*
  - $S_2 = [m]$ for the *column players*
- matrix $M \in \mathbb{R}^{n \times m}$
  - row player gets M_{s_1, s_2}
  - column player gets $-M_{s_1, s_2}$



		-1	1	-1
		1	-1	1

Equilibria of

Work it out yourself!

Nash Equilibria for Two-Player Zero-Sum Games

Nash's Theorem (1950), Special Case

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition

When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Nash Equilibria for Two-Player Zero-Sum Games

Nash's Theorem (1950), Special Case

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition

When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n \times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Intuition

No first-mover disadvantage if

- first player choses mixed strategy
- second player answers with pure strategy

Nash Equilibria for Two-Player Zero-Sum Games

Nash's Theorem (1950), Special Case

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition

When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n \times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Intuition

No first-mover disadvantage if

- first player choses mixed strategy
- second player answers with pure strategy

Proof of Corollary ("$\geq$")

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] \geq \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}] \overset{\text{Nash}}{=} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] \geq \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Nash Equilibria for Two-Player Zero-Sum Games

Nash's Theorem (1950), Special Case

For any $M \in \mathbb{R}^{n \times m}$ there exist distributions $\mathcal{S}_1^*$ on $[n]$ and $\mathcal{S}_2^*$ on $[m]$ such that

$$\mathbb{E}_{s_1 \sim \mathcal{S}_1^*, s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2^*}[M_{s_1,s_2}] = \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1^*}[M_{s_1,s_2}].$$

Intuition

When the players play according to $\mathcal{S}_1^*$ and $\mathcal{S}_2^*$, then no player can benefit by deviating from his strategy.

Corollary: Loomis (1946) Von Neumann (1928)

For any $M \in \mathbb{R}^{n \times m}$ we have

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Intuition

No first-mover disadvantage if

- first player choses mixed strategy
- second player answers with pure strategy

Proof of Corollary ("$\leq$")

$$\max_{\mathcal{S}_1} \min_{s_2 \in [m]} \mathbb{E}_{s_1 \sim \mathcal{S}_1}[M_{s_1,s_2}] = \max_{\mathcal{S}_1} \min_{\mathcal{S}_2} \mathbb{E}_{s_1 \sim \mathcal{S}_1, s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}] \leq \min_{\mathcal{S}_2} \max_{\mathcal{S}_1} \mathbb{E}_{s_1 \sim \mathcal{S}_1, s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}] = \min_{\mathcal{S}_2} \max_{s_1 \in [n]} \mathbb{E}_{s_2 \sim \mathcal{S}_2}[M_{s_1,s_2}]$$

Content

Algorithm Design as a 2-Player Zero-Sum Game

Setting

- P: a computational problem
- **Inputs**: *finite* set of inputs
- **Algos**: *finite* set of deterministic algorithms
- $C(A, I) \in \mathbb{R}$ cost of $A \in$ **Algos** on $I \in$ **Inputs**.

Algorithm Design as a 2-Player Zero-Sum Game

Setting

- P: a computational problem
- **Inputs**: *finite* set of inputs
- **Algos**: *finite* set of deterministic algorithms
- $C(A, I) \in \mathbb{R}$ cost of $A \in$ **Algos** on $I \in$ **Inputs**.

Example: Sorting

- $P =$ "sort n numbers comparison-based"[a]
- **Inputs** $= S_n$ //permutations of $[n]$
- **Algos** $=$ e.g. suitable set of decision trees
- $C(A, I) =$ # of comparisons of A for input I

[a] n finite, though possibly $n \to \infty$ later.

Algorithm Design as a 2-Player Zero-Sum Game

Setting

- P: a computational problem
- **Inputs**: *finite* set of inputs
- **Algos**: *finite* set of deterministic algorithms
- $C(A, I) \in \mathbb{R}$ cost of $A \in$ **Algos** on $I \in$ **Inputs**.

Example: Sorting

- $P =$ "sort n numbers comparison-based"[a]
- **Inputs** $= S_n$ //permutations of $[n]$
- **Algos** $=$ e.g. suitable set of decision trees
- $C(A, I) =$ # of comparisons of A for input I

[a] n finite, though possibly $n \to \infty$ later.

A Two-Player Zero-Sum Game

- Designer chooses (randomised) algorithm, i.e. a distribution on **Algos**.
  $\hookrightarrow$ Goal: Minimise (expected) cost.
- Adversary chooses (randomised) input, i.e. a distribution on **Inputs**.
  $\hookrightarrow$ Goal: Maximise (expected) cost.

Algorithm Design as a 2-Player Zero-Sum Game

Karlsruhe Institute of Technology

Setting

- P: a computational problem
- **Inputs**: *finite* set of inputs
- **Algos**: *finite* set of deterministic algorithms
- $C(A, I) \in \mathbb{R}$ cost of $A \in$ **Algos** on $I \in$ **Inputs**.

A Two-Player Zero-Sum Game

- Designer chooses (randomised) algorithm, i.e. a distribution on **Algos**.
  $\hookrightarrow$ Goal: Minimise (expected) cost.
- Adversary chooses (randomised) input, i.e. a distribution on **Inputs**.
  $\hookrightarrow$ Goal: Maximise (expected) cost.

Example: Sorting

- $P =$ "sort n numbers comparison-based"[a]
- **Inputs** $= S_n$ //permutations of $[n]$
- **Algos** $=$ e.g. suitable set of decision trees
- $C(A, I) =$ # of comparisons of A for input I

[a] n finite, though possibly $n \to \infty$ later.

Sorting (x, y, z)

		Adversary $(1,2,3)$	$(3,1,2)$	$(2,3,1)$	...
Algorithm Designer	$x < y$ then $y < z$ then* $z < x$	2	3	3	
	$y < z$ then $z < x$ then* $x < y$	3	2	3	
	...				

* Only if needed.

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity of a Problem

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)]$$

designer moves first

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity of a Problem

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \qquad \text{designer moves first}$$

$$\overset{\text{Loomis}}{=} \max_{\mathcal{I} \text{ dist. on } \textbf{Inputs}} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)] \qquad \text{adversary moves first}$$

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity of a Problem

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \qquad \text{designer moves first}$$

$$\overset{\text{Loomis}}{=} \max_{\mathcal{I} \text{ dist. on } \textbf{Inputs}} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)] \qquad \text{adversary moves first}$$

Upper Bounds on $\mathcal{C}$

Let $\mathcal{A}_0$ be a distribution on **Algos**

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \overset{\text{(old news)}}{\geq} \mathcal{C}$$

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity of a Problem

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \qquad \text{designer moves first}$$

$$\overset{\text{Loomis}}{=} \max_{\mathcal{I} \text{ dist. on } \textbf{Inputs}} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)] \qquad \text{adversary moves first}$$

Yao's Principle: (Upper and) Lower Bounds on $\mathcal{C}$

Let $\mathcal{A}_0$ be a distribution on **Algos** and $\mathcal{I}_0$ a distribution on **Inputs**. Then

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \overset{\text{(old news)}}{\geq} \mathcal{C} \overset{\text{"Yao's Principle"}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Tightness: Loomis implies that "$=$" is possible.

Randomised Complexity and Yao's Principle

Definition: Randomised Complexity of a Problem

$$\mathcal{C} := \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \qquad \text{designer moves first}$$

$$\overset{\text{Loomis}}{=} \max_{\mathcal{I} \text{ dist. on } \textbf{Inputs}} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}}[C(A, I)] \qquad \text{adversary moves first}$$

Yao's Principle: (Upper and) Lower Bounds on $\mathcal{C}$

Let $\mathcal{A}_0$ be a distribution on **Algos** and $\mathcal{I}_0$ a distribution on **Inputs**. Then

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \overset{\text{(old news)}}{\geq} \mathcal{C} \overset{\text{"Yao's Principle"}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Tightness: Loomis implies that "$=$" is possible.

$\hookrightarrow$ Can attain (tight) lower bounds on $\mathcal{C}$ by thinking about deterministic algorithm only!

Content

Computational Problem: $\overline{\wedge}$-Tree-Evaluation

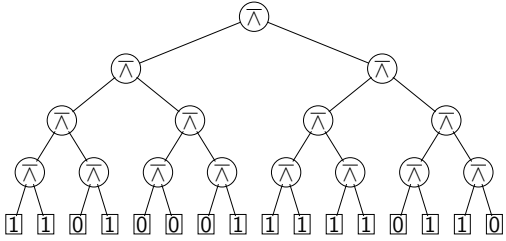
Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$



Computational Problem: $\overline{\wedge}$-Tree-Evaluation

Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

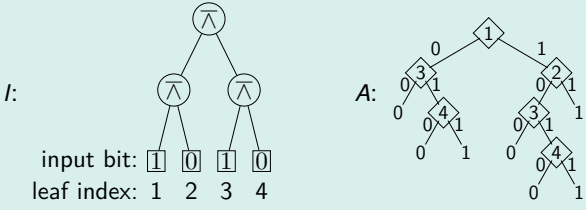
Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

Example and possible formalisation of **Algos** (that we won't use)

Each $A \in$ **Algos** corresponds to a *decision tree*. In the example:

Each leaf queried at most once per path
$\Rightarrow$ depth $\leq n \Rightarrow |\textbf{Algos}| < \infty$



Computational Problem: $\overline{\wedge}$-Tree-Evaluation

Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

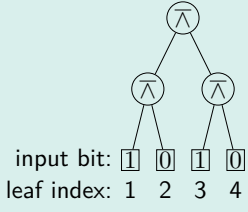
$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

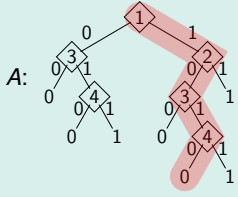
Example and possible formalisation of **Algos** (that we won't use)

Each $A \in$ **Algos** corresponds to a *decision tree*. In the example:

- $C(A, (1,0,1,0)) = 4$

Each leaf queried at most once per path
$\Rightarrow$ depth $\leq n \Rightarrow |\textbf{Algos}| < \infty$





Computational Problem: $\overline{\wedge}$-Tree-Evaluation

Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

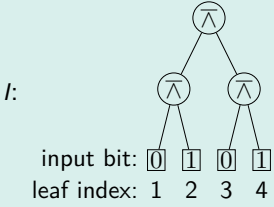
$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

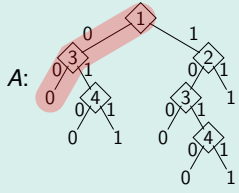
Example and possible formalisation of **Algos** (that we won't use)

Each $A \in$ **Algos** corresponds to a *decision tree*. In the example:

- $C(A, (1,0,1,0)) = 4$
- $C(A, (0,1,0,1)) = 2$

Each leaf queried at most once per path
$\Rightarrow$ depth $\leq n \Rightarrow |\textbf{Algos}| < \infty$





Computational Problem: $\overline{\wedge}$-Tree-Evaluation

Problem: Evaluate $\overline{\wedge}$-Tree of depth d

- **Inputs** $= \{0,1\}^n$ for $n = 2^d$. Specify bits at leafs.
- **Algos** $=$ Algorithms computing value at root.
- $C(A, I) = \#$ bits of I that A examines
  $\hookrightarrow$ query complexity of A on I

Goal

Bound randomised query complexity

$$\mathcal{C} = \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)].$$

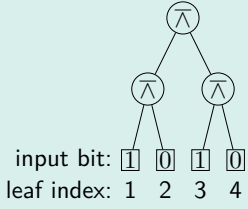
Example and possible formalisation of **Algos** (that we won't use)

Each $A \in$ **Algos** corresponds to a *decision tree*. In the example:

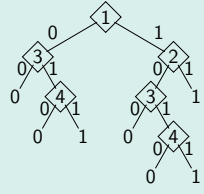
- $C(A, (1,0,1,0)) = 4$
- $C(A, (0,1,0,1)) = 2$

Each leaf queried at most once per path
$\Rightarrow$ depth $\leq n \Rightarrow |\textbf{Algos}| < \infty$

I:

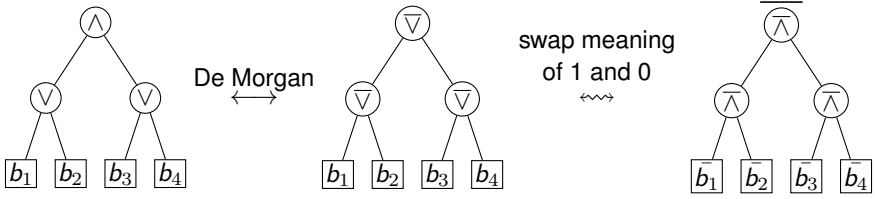


A:



What we already know

$\wedge$-$\vee$-trees are $\overline{\vee}$-trees are $\overline{\wedge}$-trees



What we already know

$\wedge$-$\vee$-trees are $\overline{\vee}$-trees are $\overline{\wedge}$-trees

Deterministic Query Complexity is n (Übungsblatt 2, Aufgabe 3)

For all $A \in$ **Algos** there exists $I \in$ **Inputs** such that $C(A, I) = n$.

What we already know

$\wedge$-$\vee$-trees are $\overline{\vee}$-trees are $\overline{\wedge}$-trees

Deterministic Query Complexity is n (Übungsblatt 2, Aufgabe 3)

For all $A \in$ **Algos** there exists $I \in$ **Inputs** such that $C(A, I) = n$.

Randomised Query Complexity is $\mathcal{O}(n^{\log_4(3)}) \approx \mathcal{O}(n^{0.792})$ (Lecture "The Power of Randomness")

Let $\mathcal{A}$ be the randomised algorithm that evaluates one of the two depth $d-1$ subtrees at random (recursively) and, if that yields 1, also evaluates the other subtree (recursively).

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \mathcal{O}(3^{d/2}) = \mathcal{O}(n^{\log_4(3)}).$$

What we already know

$\wedge$-$\vee$-trees are $\overline{\vee}$-trees are $\overline{\wedge}$-trees

Deterministic Query Complexity is n (Übungsblatt 2, Aufgabe 3)

For all $A \in$ **Algos** there exists $I \in$ **Inputs** such that $C(A, I) = n$.

Randomised Query Complexity is $\mathcal{O}(n^{\log_4(3)}) \approx \mathcal{O}(n^{0.792})$ (Lecture "The Power of Randomness")

Let $\mathcal{A}$ be the randomised algorithm that evaluates one of the two depth $d-1$ subtrees at random (recursively) and, if that yields 1, also evaluates the other subtree (recursively).

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] = \mathcal{O}(3^{d/2}) = \mathcal{O}(n^{\log_4(3)}).$$

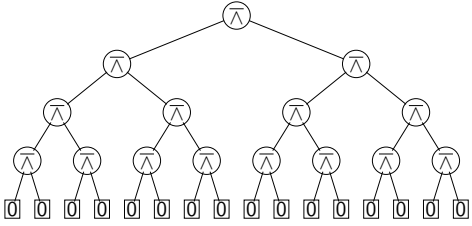
Goal: Show lower bound of $\Omega(\varphi^d) \approx \Omega(n^{0.694})$ using Yao's Principle (φ is the golden ratio).

Remark: actual complexity is $\Theta(n^{\log_4(3)})$, but that's more difficult.

Warm Up: A simple lower bound

Observation

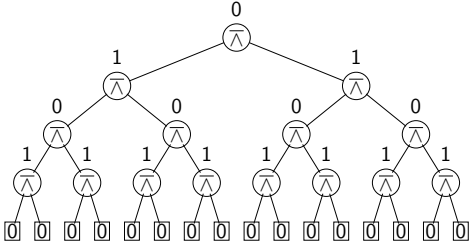
For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.



Warm Up: A simple lower bound

Observation

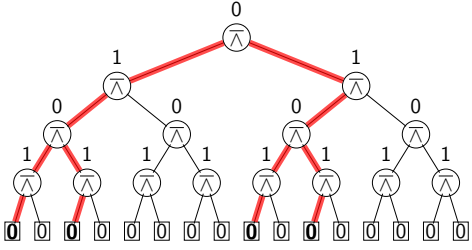
For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.



Warm Up: A simple lower bound

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.



Proof

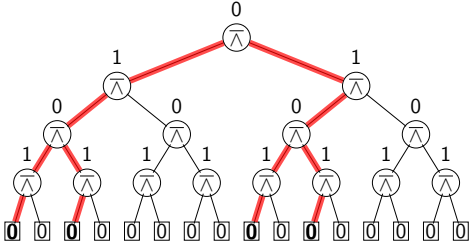
- in the end A knows that the root is 0.
- knowing a 0 requires knowing that both children are 1.
- Knowing a 1 requires knowing of one child that it is 0.

$\hookrightarrow$ A knows of $\geq 2^{d/2}$ leafs that they are 0 and must have checked them.

Warm Up: A simple lower bound

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.



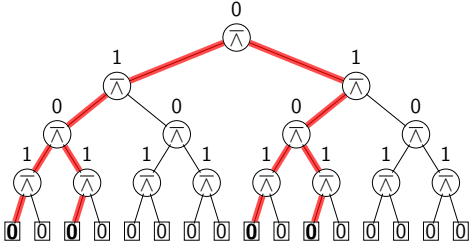
Corollary: Randomised Complexity is $\Omega(\sqrt{n})$

$$
\begin{aligned}
C &= \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \\
&\geq \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, (0, \ldots, 0))] \\
&= \min_{A \in \textbf{Algos}} [C(A, (0, \ldots, 0))] \\
&\geq 2^{d/2} = 2^{\log_2(n)/2} = n^{1/2}.
\end{aligned}
$$

Warm Up: A simple lower bound

Observation

For any even $d \in \mathbb{N}$ and $A \in$ **Algos** we have $C(A, (0, \ldots, 0)) \geq 2^{d/2}$.

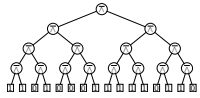


Corollary: Randomised Complexity is $\Omega(\sqrt{n})$

$$
\begin{aligned}
\mathcal{C} &= \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, I)] \\
&\geq \min_{\mathcal{A} \text{ dist. on } \textbf{Algos}} \mathbb{E}_{A \sim \mathcal{A}}[C(A, (0, \ldots, 0))] \\
&= \min_{A \in \textbf{Algos}} [C(A, (0, \ldots, 0))] \\
&\geq 2^{d/2} = 2^{\log_2(n)/2} = n^{1/2}.
\end{aligned}
$$

Note Yao's spirit: Lower bound on *randomised complexity* from result on *deterministic algorithms*.

A stronger lower bound

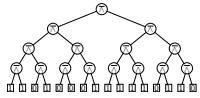


Karlsruhe Institute of Technology

Independent Bernoulli Inputs

Let $\mathcal{I}_p = \mathrm{Ber}(p)^n$ be the distribution where leafs are assigned independently values with distribution $\mathrm{Ber}(p)$.

A stronger lower bound



Karlsruhe Institute of Technology

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

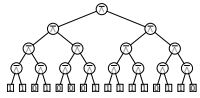
Independent Bernoulli Inputs

Let $\mathcal{I}_p = \text{Ber}(p)^n$ be the distribution where leafs are assigned independently values with distribution $\text{Ber}(p)$.

Deterministic Algorithm

Algorithm simpleEval(T)**:**
- **if** $T = \text{leaf}(b)$ **then**
  - **return** b
- **else**
  - $(T_\ell, T_r) \leftarrow T$
  - **if** simpleEval(T_ℓ) $= 0$ **then**
    - **return** 1
  - **else**
    - **return** $\neg$simpleEval(T_r)

A stronger lower bound



Karlsruhe Institute of Technology

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Independent Bernoulli Inputs

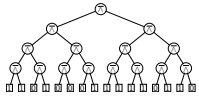
Let $\mathcal{I}_p = \text{Ber}(p)^n$ be the distribution where leafs are assigned independently values with distribution $\text{Ber}(p)$.

Deterministic Algorithm

Algorithm simpleEval(T)**:**

- **if** $T = \text{leaf}(b)$ **then**
  - **return** b
- **else**
  - $(T_\ell, T_r) \leftarrow T$
  - **if** simpleEval(T_ℓ) $= 0$ **then**
    - **return** 1
  - **else**
    - **return** $\neg$simpleEval(T_r)

A stronger lower bound



Karlsruhe Institute of Technology

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A,I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1+p_0)^d = \varphi^d.$$

Corollary: $\mathcal{C} = \Omega(\varphi^d) \approx \Omega(n^{0.694})$

$$\mathcal{C} \overset{\text{Yao}}{\geq} \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A,I)] \overset{\text{Tarsi}}{=} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$$

$$\overset{\text{Lemma}}{=} \varphi^d = \varphi^{\log_2 n} = n^{\log_2 \varphi} \approx n^{0.694}.$$

Independent Bernoulli Inputs

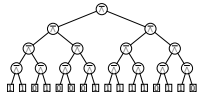
Let $\mathcal{I}_p = \text{Ber}(p)^n$ be the distribution where leafs are assigned independently values with distribution $\text{Ber}(p)$.

Deterministic Algorithm

Algorithm simpleEval(T)**:**

- **if** $T = \text{leaf}(b)$ **then**
  - **return** b
- **else**
  - $(T_\ell, T_r) \leftarrow T$
  - **if** simpleEval(T_ℓ) $= 0$ **then**
    - **return** 1
  - **else**
    - **return** $\neg$simpleEval(T_r)

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

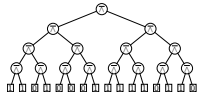
$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Deterministic Algorithm

Algorithm simpleEval(T)**:**

- **if** $T = \text{leaf}(b)$ **then**
  - **return** b
- **else**
  - $(T_\ell, T_r) \leftarrow T$
  - **if** simpleEval(T_ℓ) $= 0$ **then**
    - **return** 1
  - **else**
    - **return** $\neg$simpleEval(T_r)

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Proof.

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim \text{Ber}(p_0)$ then $a \overline{\wedge} b \sim \text{Ber}(1 - p_0^2) = \text{Ber}(p_0)$.

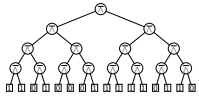
□

Deterministic Algorithm

Algorithm simpleEval(T)**:**

```
if T = leaf(b) then
    return b
else
    (T_ℓ, T_r) ← T
    if simpleEval(T_ℓ) = 0 then
        return 1
    else
        return ¬simpleEval(T_r)
```

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Proof.

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim \text{Ber}(p_0)$ then $a \overline{\wedge} b \sim \text{Ber}(1 - p_0^2) = \text{Ber}(p_0)$.
- For $I \sim \mathcal{I}_{p_0}$ the probability that an *internal* tree node evaluates to 1 is p_0.

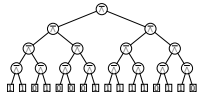
□

Deterministic Algorithm

Algorithm simpleEval(T)**:**

```
if T = leaf(b) then
    return b
else
    (T_ℓ, T_r) ← T
    if simpleEval(T_ℓ) = 0 then
        return 1
    else
        return ¬simpleEval(T_r)
```

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Proof.

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim \text{Ber}(p_0)$ then $a \overline{\wedge} b \sim \text{Ber}(1 - p_0^2) = \text{Ber}(p_0)$.
- For $I \sim \mathcal{I}_{p_0}$ the probability that an *internal* tree node evaluates to 1 is p_0.
- Let $c_d := \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$ for trees of depth d. Then

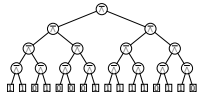
□

Deterministic Algorithm

Algorithm simpleEval(T)**:**

```
if T = leaf(b) then
    return b
else
    (T_ℓ, T_r) ← T
    if simpleEval(T_ℓ) = 0 then
        return 1
    else
        return ¬simpleEval(T_r)
```

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1 + p_0)^d = \varphi^d.$$

Proof.

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim \text{Ber}(p_0)$ then $a \overline{\wedge} b \sim \text{Ber}(1 - p_0^2) = \text{Ber}(p_0)$.
- For $I \sim \mathcal{I}_{p_0}$ the probability that an *internal* tree node evaluates to 1 is p_0.
- Let $c_d := \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$ for trees of depth d. Then
  - $c_0 = 1$ // tree of depth 0 is just the leaf

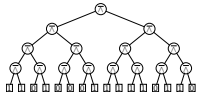
□

Deterministic Algorithm

Algorithm simpleEval(T)**:**

```
if T = leaf(b) then
    return b
else
    (T_ℓ, T_r) ← T
    if simpleEval(T_ℓ) = 0 then
        return 1
    else
        return ¬simpleEval(T_r)
```

Proof of Lemma: Cost of simpleEval on $\mathcal{I}_{p_0}$



Karlsruhe Institute of Technology

Lemma

Let $\varphi = \frac{\sqrt{5}+1}{2}$ be the golden ratio and $p_0 = \varphi - 1$. Then

$$\mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)] = (1+p_0)^d = \varphi^d.$$

Proof.

- $p_0 = \frac{\sqrt{5}-1}{2}$ is the solution to $p = 1 - p^2$.
- If $a, b \sim \text{Ber}(p_0)$ then $a \overline{\wedge} b \sim \text{Ber}(1-p_0^2) = \text{Ber}(p_0)$.
- For $I \sim \mathcal{I}_{p_0}$ the probability that an *internal* tree node evaluates to 1 is p_0.
- Let $c_d := \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)]$ for trees of depth d. Then
  - $c_0 = 1$ // tree of depth 0 is just the leaf
  - $c_d = c_{d-1} + p_0 \cdot c_{d-1} = (1+p_0)c_{d-1} \overset{\text{Ind.}}{=} (1+p_0)(1+p_0)^{d-1} = (1+p_0)^d$
    // Always one recursive call, with probability p a second one.

□

Deterministic Algorithm

Algorithm simpleEval(T)**:**

```
if T = leaf(b) then
    return b
else
    (T_ℓ, T_r) ← T
    if simpleEval(T_ℓ) = 0 then
        return 1
    else
        return ¬simpleEval(T_r)
```

Content

Tarsi's Theorem

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

Tarsi's Theorem

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

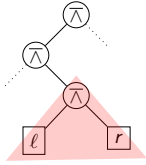
Proof idea:

- Take optimal Algorithm A.
- Transform A into simpleEval step by step.
- Show: Expected query complexity never increases.

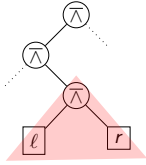
Lemma: Evaluating Superleafs like simpleEval

Definition: Superleafs

A *superleaf* consists of two sibling leafs and their parent.



Lemma: Evaluating Superleafs like simpleEval



Definition: Superleafs

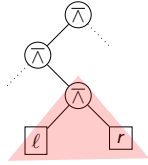
A *superleaf* consists of two sibling leafs and their parent.

Lemma

For any $p \in [0, 1]$ and any $A \in$ **Algos** there exists $A' \in$ **Algos** such that

- $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$
- A' behaves on any superleaf $T = (\ell, r)$ like simpleEval:
  - i never visits r before ℓ
  - ii never visits r if $\ell = 0$
  - iii immediately visits r after visiting ℓ if $\ell = 1$

Lemma: Evaluating Superleafs like simpleEval



Definition: Superleafs

A *superleaf* consists of two sibling leafs and their parent.

Lemma

For any $p \in [0, 1]$ and any $A \in$ **Algos** there exists $A' \in$ **Algos** such that

- $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$
- A' behaves on any superleaf $T = (\ell, r)$ like simpleEval:
  - i never visits r before ℓ
  - ii never visits r if $\ell = 0$
  - iii immediately visits r after visiting ℓ if $\ell = 1$

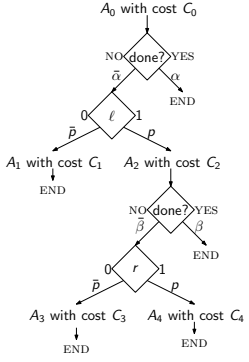
Proof Idea

- We fix every superleaf one by one. Let T be superleaf that needs fixing.
- Property i: Switch roles of ℓ and r if needed. Does not change the expected cost.
- Property ii: r does not contribute to result. Not visiting r *reduces* expected cost.
- Property iii: More difficult. See next slide.

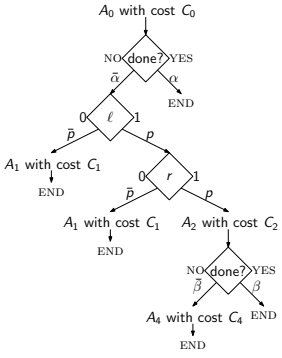
Modified Algorithm B

Original Algorithm A

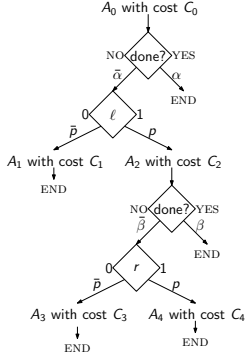
Modified Algorithm D



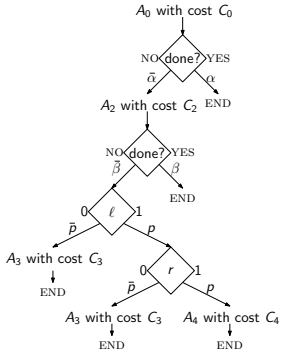
Modified Algorithm B

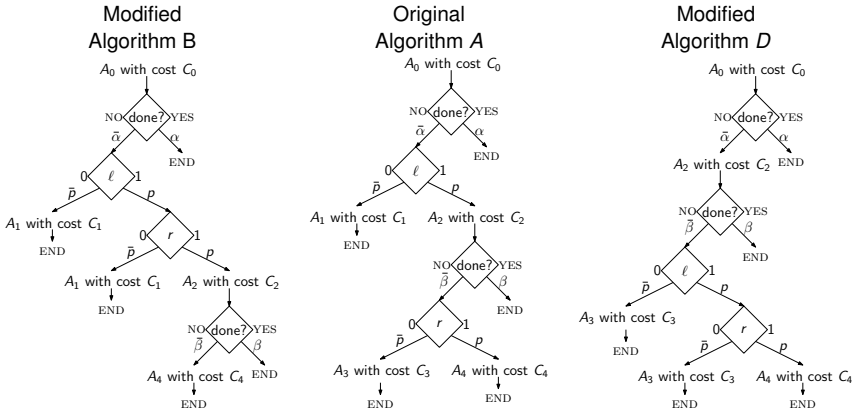


Original Algorithm A



Modified Algorithm D





$$C_A := \mathbb{E}[C(A,I)] = \mathbb{E}[C_0 + \bar{\alpha}\cdot(1 + \bar{p}C_1 + p\cdot(C_2 + \bar{\beta}(1 + \bar{p}C_3 + pC_4)))]$$
$$C_B := \mathbb{E}[C(B,I)] = \mathbb{E}[C_0 + \bar{\alpha}\cdot(1 + \bar{p}C_1 + p\cdot(1 + \bar{p}C_1 + p(C_2 + \bar{\beta}C_4)))]$$
$$C_D := \mathbb{E}[C(D,I)] = \mathbb{E}[C_0 + \bar{\alpha}\cdot(C_2 + \bar{\beta}(1 + \bar{p}C_3 + p(1 + \bar{p}C_3 + pC_4)))]$$

$$(C_B - C_A) + p\cdot(C_D - C_A) = \ldots = 0$$
$$\Rightarrow C_B - C_A \leq 0 \vee C_D - C_A \leq 0$$
$\Rightarrow$ B or D (or both) are at least as good as A and both visit superleaf (ℓ, r) as desired.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

The expected cost for evaluating a superleaf is $1 + p$.

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

The expected cost for evaluating a superleaf is $1 + p$.
Hence

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] = (1 + p)\mathbb{E}[L']$$
$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)] = (1 + p)\mathbb{E}[L]$$

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

The expected cost for evaluating a superleaf is $1 + p$.
Hence

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] = (1 + p)\mathbb{E}[L']$$
$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)] = (1 + p)\mathbb{E}[L]$$

Finally we obtain:

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(\text{simpleEval}, I)] = (1 + p)\mathbb{E}[L] \leq (1 + p)\mathbb{E}[L']$$
$$= \mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Theorem (Tarsi 1984)

For any $p \in [0,1]$ simpleEval is optimal for input distribution $\mathcal{I}_p$, i.e.

$$\min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(A, I)] = \mathbb{E}_{I \sim \mathcal{I}_{p_0}}[C(\text{simpleEval}, I)].$$

We use induction on d. For $d = 0$ simpleEval is clearly optimal. Let now $d \geq 1$.
Let $A \in$ **Algos** be an algorithm minimising $\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)]$.
By Lemma: There exists $A' \in$ **Algos** that behaves like simpleEval on superleafs such that

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Let L' be the number of superleafs visited by A' and L the number of superleafs visited by simpleEval.
Superleafs evaluate to 1 with probability $1 - p^2$ independently and are in a complete binary tree of depth $d - 1$.

Apply induction for $d' = d - 1$ and $p' = 1 - p^2$.

$$\mathbb{E}_{I \sim \mathcal{I}_p}[L] \overset{\text{Ind.}}{\leq} \mathbb{E}_{I \sim \mathcal{I}_p}[L'].$$

The expected cost for evaluating a superleaf is $1 + p$.
Hence

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] = (1 + p)\mathbb{E}[L']$$
$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)] = (1 + p)\mathbb{E}[L]$$

Finally we obtain:

$$\mathbb{E}_{I \sim \mathcal{I}_p}[C(\text{simpleEval}, I)] = (1 + p)\mathbb{E}[L] \leq (1 + p)\mathbb{E}[L']$$
$$= \mathbb{E}_{I \sim \mathcal{I}_p}[C(A', I)] \leq \mathbb{E}_{I \sim \mathcal{I}_p}[C(A, I)].$$

Hence, simpleEval is optimal for $\mathcal{I}_p$. □

Content

Ski Rental – A Prototypical Online Problem

Setting: You are on a ski trip

Trip lasts for unknown number of days $I \in \mathbb{N}$ (“as long as there is snow”).

Every day, if no skis bought yet:

- RENT skis for one day for cost 1 *or*
- BUY skis for cost $B \in \mathbb{N}$.

Ski Rental – A Prototypical Online Problem

Setting: You are on a ski trip

Trip lasts for unknown number of days $I \in \mathbb{N}$ ("as long as there is snow").

Every day, if no skis bought yet:

- RENT skis for one day for cost 1 *or*
- BUY skis for cost $B \in \mathbb{N}$.

Framing using Online Algorithms

- **Inputs** $= \mathbb{N}$: number of days (not known in advance)
- **Algos** $= \mathbb{N}$: specify day for choosing BUY
- cost for $A \in$ **Algos** on $I \in$ **Inputs**:

$$C(A, I) = \begin{cases} I & \text{if } I < A \\ A - 1 + B & \text{otherwise.} \end{cases}$$

- cost of optimum *offline* solution

$$\mathrm{OPT}(I) = \begin{cases} I & \text{if } I < B \\ B & \text{otherwise.} \end{cases}$$

Ski Rental – A Prototypical Online Problem

Setting: You are on a ski trip

Trip lasts for unknown number of days $I \in \mathbb{N}$ ("as long as there is snow").

Every day, if no skis bought yet:

- RENT skis for one day for cost 1 *or*
- BUY skis for cost $B \in \mathbb{N}$.

Goal: Minimise Competitive Ratio

The *competitive ratio* of $A \in$ **Algos** is

$$C_A = \sup_{I \in \textbf{Inputs}} \frac{C(A, I)}{\mathrm{OPT}(I)}.$$

Framing using Online Algorithms

- **Inputs** $= \mathbb{N}$: number of days (not known in advance)
- **Algos** $= \mathbb{N}$: specify day for choosing BUY
- cost for $A \in$ **Algos** on $I \in$ **Inputs**:

$$C(A, I) = \begin{cases} I & \text{if } I < A \\ A - 1 + B & \text{otherwise.} \end{cases}$$

- cost of optimum *offline* solution

$$\mathrm{OPT}(I) = \begin{cases} I & \text{if } I < B \\ B & \text{otherwise.} \end{cases}$$

Ski Rental – A Prototypical Online Problem

Setting: You are on a ski trip

Trip lasts for unknown number of days $I \in \mathbb{N}$ ("as long as there is snow").

Every day, if no skis bought yet:

- RENT skis for one day for cost 1 *or*
- BUY skis for cost $B \in \mathbb{N}$.

Goal: Minimise Competitive Ratio

The *competitive ratio* of distribution $\mathcal{A}$ on **Algos** is

$$C_{\mathcal{A}} = \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A \sim \mathcal{A}}[C(A, I)]}{\mathrm{OPT}(I)}.$$

Framing using Online Algorithms

- **Inputs** $= \mathbb{N}$: number of days (not known in advance)
- **Algos** $= \mathbb{N}$: specify day for choosing BUY
- cost for $A \in$ **Algos** on $I \in$ **Inputs**:

$$C(A, I) = \begin{cases} I & \text{if } I < A \\ A - 1 + B & \text{otherwise.} \end{cases}$$

- cost of optimum *offline* solution

$$\mathrm{OPT}(I) = \begin{cases} I & \text{if } I < B \\ B & \text{otherwise.} \end{cases}$$

Break-Even is the best deterministic algorithm

Observation

The algorithm breakEven $:= B$ has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof

Break-Even is the best deterministic algorithm

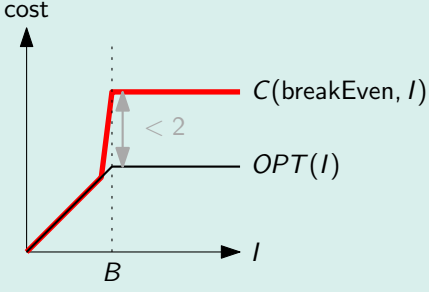
Observation

The algorithm breakEven := B has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for breakEven is attained for input $I = B$.

$$C_{\text{breakEven}} = \sup_{I \in \mathbb{N}} \frac{C(\text{breakEven}, I)}{\text{OPT}(I)} = \frac{C(\text{breakEven}, B)}{\text{OPT}(B)}$$
$$= \frac{B - 1 + B}{B} = \frac{2B-1}{B}.$$

Break-Even is the best deterministic algorithm

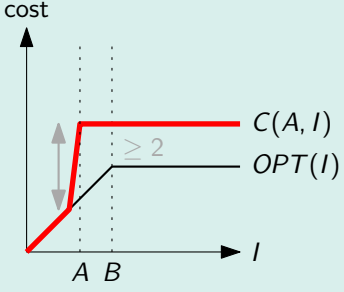
Observation

The algorithm breakEven := B has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for $A \in$ **Algos** with $A < B$ is attained for input $I = A$.

$$C_A = \sup_{I \in \mathbb{N}} \frac{C(A,I)}{\mathrm{OPT}(I)} = \frac{C(A,A)}{\mathrm{OPT}(A)} = \frac{A-1+B}{A} = 1 + \frac{B-1}{A} \geq 1+1 = 2.$$

Break-Even is the best deterministic algorithm

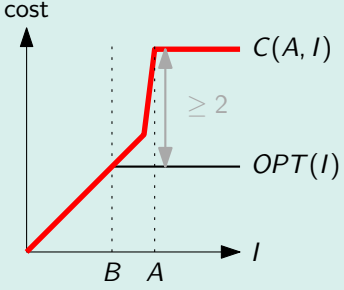
Observation

The algorithm breakEven := B has competitive ratio $\frac{2B-1}{B} \approx 2$.
All other $A \in$ **Algos** have competitive ratio ≥ 2.

Recall

B is the cost to BUY.

Proof



The worst ratio for $A \in$ **Algos** with $A > B$ is attained for input $I = A$.

$$C_A = \sup_{I \in \mathbb{N}} \frac{C(A,I)}{\mathrm{OPT}(I)} = \frac{C(A,A)}{\mathrm{OPT}(A)} = \frac{A-1+B}{B} = 1 + \frac{A-1}{B} \geq 1+1 = 2.$$

A randomised algorithm can beat break-even

Observation (assuming wlog that B is a multiple of 3)

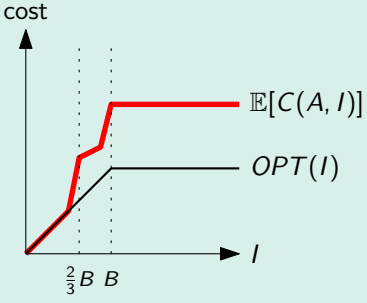
The randomised algorithm $\mathcal{A} = \mathcal{U}(\{\frac{2}{3}B, B\})$ has competitive ratio $\approx 1 + \frac{5}{6}$.

A randomised algorithm can beat break-even

Observation (assuming wlog that B is a multiple of 3)

The randomised algorithm $\mathcal{A} = \mathcal{U}(\{\frac{2}{3}B, B\})$ has competitive ratio $\approx 1 + \frac{5}{6}$.

Proof



The competitive ratio of $\mathcal{A}$ "spikes" for inputs $\frac{2}{3}B$ and B. It is decreasing in between and constant after B.

$$\mathbb{E}_{A\sim\mathcal{A}}[C(A, \tfrac{2}{3}B)] = \underbrace{\tfrac{2}{3}B - 1}_{\text{rent}} + \underbrace{\tfrac{1}{2}(1 + B)}_{\text{rent or buy}} < \tfrac{7}{6}B, \qquad \mathrm{OPT}(\tfrac{2}{3}B) = \tfrac{2}{3}B,$$

$$\mathbb{E}_{A\sim\mathcal{A}}[C(A, B)] = \underbrace{B}_{\text{buy}} + \underbrace{\tfrac{2}{3}B - 1}_{\text{rent}} + \underbrace{\tfrac{1}{2}(\tfrac{1}{3}B)}_{\text{maybe rent}} < \tfrac{11}{6}B, \qquad \mathrm{OPT}(B) = B.$$

Hence $C_{\mathcal{A}} = \sup_{I\in\mathbb{N}} \dfrac{\mathbb{E}_{A\sim\mathcal{A}}[C(A, I)]}{\mathrm{OPT}(I)} \leq \max\left\{\dfrac{7/6}{2/3}, \dfrac{11/6}{1}\right\} = \max\left\{\dfrac{7}{4}, \dfrac{11}{6}\right\} = \dfrac{11}{6}.$

What's next?

Goal: Lower bound

No randomised algorithm has competitive ratio better than $\frac{e-1}{e} \approx 1.582$.

Yao's Principle for Online Algorithms

Theorem (see Online Optimization Lecture, Corollary 3.8, Prof. Yann Disser, Darmstadt, 2023)

For any distribution $\mathcal{A}_0$ on **Algos**

$$C_{\mathcal{A}_0} \stackrel{\text{def}}{=} \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)]}{\mathrm{OPT}(I)} \geq \frac{\mathbb{E}_{I \sim \mathcal{I}_0, A \sim \mathcal{A}_0}[C(A, I)]}{\mathbb{E}_{I \sim \mathcal{I}_0}[\mathrm{OPT}(I)]}$$

Yao's Principle for Online Algorithms

Theorem (see Online Optimization Lecture, Corollary 3.8, Prof. Yann Disser, Darmstadt, 2023)

For any distribution $\mathcal{A}_0$ on **Algos** and any distribution $\mathcal{I}_0$ on **Inputs** we have

$$C_{\mathcal{A}_0} \stackrel{\text{def}}{=} \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A\sim\mathcal{A}_0}[C(A,I)]}{\mathrm{OPT}(I)} \geq \frac{\mathbb{E}_{I\sim\mathcal{I}_0, A\sim\mathcal{A}_0}[C(A,I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)]} \geq \frac{\inf_{A\in\textbf{Algos}} \mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)]}.$$

Yao's Principle for Online Algorithms

Theorem (see Online Optimization Lecture, Corollary 3.8, Prof. Yann Disser, Darmstadt, 2023)

For any distribution $\mathcal{A}_0$ on **Algos** and any distribution $\mathcal{I}_0$ on **Inputs** we have

$$C_{\mathcal{A}_0} \stackrel{\text{def}}{=} \sup_{I \in \textbf{Inputs}} \frac{\mathbb{E}_{A \sim \mathcal{A}_0}[C(A,I)]}{\mathrm{OPT}(I)} \geq \frac{\mathbb{E}_{I \sim \mathcal{I}_0, A \sim \mathcal{A}_0}[C(A,I)]}{\mathbb{E}_{I \sim \mathcal{I}_0}[\mathrm{OPT}(I)]} \geq \frac{\inf_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A,I)]}{\mathbb{E}_{I \sim \mathcal{I}_0}[\mathrm{OPT}(I)]}.$$

Remark

- Yao's principle exists for other settings as well.
- Proof of "$\geq$" relatively easy to prove.
- Tightness typically follows from duality of optimisation problems or fixed point theorems.
  (though I'm not sure how it works here)

A hard distribution for Ski-Rental: Intuition

$$\mathcal{I}_0 := \mathsf{Geom}_1(\tfrac{1}{B}).$$

A hard distribution for Ski-Rental: Intuition

$$\mathcal{I}_0 := \mathsf{Geom}_1(\tfrac{1}{B}).$$

Why $\mathcal{I}_0$?

- distribution is **memoryless**, i.e. $\Pr_{I\sim\mathcal{I}_0}[I = i + t \mid I > i] = \Pr_{I\sim\mathcal{I}_0}[I = t]$.
  Assume no skis bought on day i: Minimising expected *future* cost is the same problem as on day 1.
  $\hookrightarrow$ wlog: either buy right away or not at all.

A hard distribution for Ski-Rental: Intuition

$$\mathcal{I}_0 := \mathsf{Geom}_1(\tfrac{1}{B}).$$

Why $\mathcal{I}_0$?

- distribution is **memoryless**, i.e. $\Pr_{I\sim\mathcal{I}_0}[I = i + t \mid I > i] = \Pr_{I\sim\mathcal{I}_0}[I = t]$.
  Assume no skis bought on day i: Minimising expected *future* cost is the same problem as on day 1.
  $\hookrightarrow$ wlog: either buy right away or not at all.
- **expectation** tuned such that

$$\mathbb{E}_{I\sim\mathcal{I}_0}[C(\text{never buy}, I)] = \mathbb{E}_{I\sim\mathcal{I}_0}[C(\text{immediately buy}, I)] = B.$$

  $\hookrightarrow$ all strategies equally good

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \text{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "Pr[❄]"*. Then*

i. $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] = B$ *for all* $A \in \mathbb{N}$.

ii. $\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] = B(1 - (1 - \frac{1}{B})^B)$.

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \text{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "$\Pr[\text{❄}]$"*. Then*

i $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] = B$ *for all* $A \in \mathbb{N}$.

ii $\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] = B(1 - (1 - \frac{1}{B})^B)$.

Proof of (i)

$$\mathbb{E}_{I\sim\mathcal{I}_0}[C(A, I)] = \underbrace{\Pr[I \geq A] \cdot B}_{\text{buy}} + \sum_{i=1}^{A-1} \underbrace{\Pr[I \geq i] \cdot 1}_{\text{rent}}$$

$$= q^{A-1} \cdot B + \sum_{i=1}^{A-1} q^{i-1} = q^{A-1} \cdot B + \sum_{i=0}^{A-2} q^i = q^{A-1} \cdot B + \frac{1 - q^{A-1}}{1 - q}$$

$$= q^{A-1} \cdot B + (1 - q^{A-1})B = B.$$

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \text{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "$\Pr[$❄$]$". *Then*

i $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)] = B$ *for all* $A \in \mathbb{N}$.

ii $\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] = B(1-(1-\frac{1}{B})^B)$.

Tail sum formula:

For random variable X with values in $\mathbb{N}$:

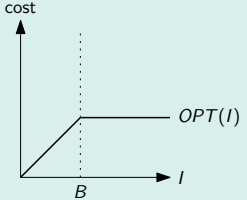
$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Proof of (ii)

$$\mathbb{E}_{I\sim\mathcal{I}_0}[\text{OPT}(I)] \overset{\text{TSF}}{=} \sum_{j\geq 1} \Pr[\text{OPT}(I) \geq j] = \sum_{j=1}^{B} \Pr[\text{OPT}(I) \geq j]$$

$$= \sum_{j=1}^{B} \Pr[I \geq j] = \sum_{j=1}^{B} q^{j-1} = \sum_{j=0}^{B-1} q^j$$

$$= \frac{1-q^B}{1-q} = B(1-(1-\tfrac{1}{B})^B).$$



Note: $\text{OPT}(I) = I$ for $I \in [B]$.

A hard distribution for Ski-Rental: Analysis

Lemma

Let $\mathcal{I}_0 := \mathrm{Geom}(\frac{1}{B})$ *and* $q := 1 - \frac{1}{B} =$ "$\Pr[$ ❄ $]$"*. Then*

i. $\mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)] = B$ *for all* $A \in \mathbb{N}$.

ii. $\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)] = B(1-(1-\frac{1}{B})^B)$.

Tail sum formula:

For random variable X with values in $\mathbb{N}$:

$$\mathbb{E}[X] = \sum_{j\geq 1} \Pr[X \geq j].$$

Lower bound for Ski-Rental

By Yao's theorem any randomised algorithm $\mathcal{A}$ for ski-rental has competitive ratio at least

$$C_{\mathcal{A}} \overset{\text{Yao}}{\geq} \frac{\inf_{A\in\textbf{Algos}} \mathbb{E}_{I\sim\mathcal{I}_0}[C(A,I)]}{\mathbb{E}_{I\sim\mathcal{I}_0}[\mathrm{OPT}(I)]} = \frac{B}{B(1-(1-\frac{1}{B})^B)} = \frac{1}{1-(1-\frac{1}{B})^B}.$$

For large B the lower bound converges to $\lim_{B\to\infty} \frac{1}{1-(1-\frac{1}{B})^B} = \frac{1}{1-1/e} = \frac{e}{e-1} \approx 1.582$.

Upper bound for Ski-Rental

Karlsruhe Institute of Technology

Remark: The lower bound is tight (Karlin et al. 1994)

There exists a distribution $\mathcal{A}$ on $[B]$ such that $c_{\mathcal{A}} \leq \frac{e}{e-1}$.

Upper bound for Ski-Rental

Karlsruhe Institute of Technology

Remark: The lower bound is tight (Karlin et al. 1994)

There exists a distribution $\mathcal{A}$ on $[B]$ such that $c_{\mathcal{A}} \leq \frac{e}{e-1}$.

Applications

Very basic online question:

Should I pay a small possibly recurring cost or a large one time cost?

Occurs in:

- Cache management.
- Networking.
- Scheduling.
- ...

Conclusion

Algorithm Design as a Two-Player Game

- “we” choose algorithm to minimise cost
- “adversary” chooses input to maximise cost
- Nash/Loomis: It does not matter who moves first
  if mixed strategy is allowed for first player.

Yao’s Principle

Lower bound on worst-case expected cost of *any randomised algorithm* $\mathcal{A}_0$ by analying *any deterministic algorithm* on specific input distribution $\mathcal{I}_0$.

$$\max_{I \in \textbf{Inputs}} \mathbb{E}_{A \sim \mathcal{A}_0}[C(A, I)] \geq \mathcal{C} \geq \min_{A \in \textbf{Algos}} \mathbb{E}_{I \sim \mathcal{I}_0}[C(A, I)].$$

Can narrow down randomised complexity $\mathcal{C}$ of underlying problem from both sides.

Anhang: Mögliche Prüfungsfragen I

Spieltheorie:

- Was ist ein Zwei-Spieler-Spiel im Sinne der Spieltheorie?
- Was ist ein Nash-Equilibrium?
- Gibt es immer ein Nash-Equilibrium?
- Was ist ein Nullsummenspiel?
- Was besagt der Satz von Nash (für Zwei-Spieler Nullsummenspiele)?
- Was besagt der Satz von Loomis?
- Beweise den Satz von Loomis! (anspruchsvolle Aufgabe)

Yaos Prinzip:

- Worin besteht die Verbindung zwischen Spieltheorie und dem Entwurf von Algorithmen?
- Wie ist die randomisierte Komplexität (bzgl. einer Kostenfunktion C) normalerweise definiert? Welche andere Sichtweise ergibt sich darauf durch den Satz von Loomis?
- Formuliere Yaos Prinzip! Wofür ist es nützlich?

Anhang: Mögliche Prüfungsfragen II

Anwendung auf $\overline{\wedge}$-Bäume:

- Welches Ziel haben wir uns bei der Auswertung von $\overline{\wedge}$-Bäumen gesetzt? (Anfragekomplexität minimieren)
- Welche Worst-Case Kosten lassen sich mit einem deterministischen Algorithmus erreichen?
- Können randomisierte Algorithmen das besser? Wie?
- Man kann sich recht leicht überlegen, dass die randomisierte Komplexität $\Omega(\sqrt{n})$ beträgt. Wie ging das?
- Wir haben auch eine schärfere Analyse gesehen. Welche Komponenten hatte diese? Insbesondere: Wie kommt dabei Yao Prinzip zur Anwendung?
- Was besagt der Satz von Tarsi?

Ski-Rental-Problem:

- Formuliere das Ski-Rental Problem.
- Wie nennt man diese Art von Problem? (*Online* Problem)
- Spielt das nur im Wintersport eine Rolle? (nur Stichworte)
- Wie ist der kompetitive Faktor definiert?

Anhang: Mögliche Prüfungsfragen III

- Was ist der beste deterministische Algorithmus? Wie kann man das einsehen?
- Gibt es einen randomisierten Algorithmus der Break-Even schlagen kann? (nur die Idee)
- Formuliere Yaos Prinzip für Online Algorithmen.
- Welche Eingabeverteilung haben wir für die untere Schranke für Ski-Rental zugrunde gelegt? Was ist die Intuition?
- Welche Kosten ergeben sich für Online und Offline Algorithmen für diese Eingabeverteilung? Was lässt sich entsprechend über den kompetitiven Faktor sagen?