Lecture 1: The Math and Science of Strategy

Lecture 1: 13 Aug 2025

CS6208 Fall 2025: Computational Game Theory

About me

Assistant Professor @ NUS

- July 2024 – (now)
  - Postdoc @ Columbia (supervised by Christian Kroer, Garud Iyengar)
  - PhD @ CMU (advised by Zico Kolter, Fei Fang)
  - DSO National Labs (Signal Processing Lab, Sensors Exploitation Program, SR division)
  - Undergrad @ NUS (supervised by Bryan Low)

Multiagent systems, Game Theory, ML + Optimization

me @ New York City

Admin

Class is **much larger** than expected (even larger than last year)

- May have to rethink grading
  - More homework, no project presentations

Regarding audit requests

- You are allowed to sit in if space permits
- NUS SoC does **not** have a formal audit system
- **Email me** to be added to Canvas

Regarding course appeals

- I am okay with accepting PhD students (especially junior ones)
- Submit an appeal to SoC grad office directly + email me

Prerequisites: Linear algebra, probability, basic python

Office hours: TBD

Schedule and Grading

My philosophy: help you maximize research output

I will teach ~~the first half of~~ the class

~~Students will lead discussions for the second half~~

- ~~~30 minutes per discussion (pairs or individual)~~

Grading

- Problem sets x2 (or x3): 60%
  - Simple stuff that is covered in class
  - A little bit of coding, plain python (and maybe numpy) needed
  - Probably done in pairs, if class size is still large after this week
- Project proposal/report: 40%

Policy on AI use

AI policy: Follow NUS general guidelines

- https://ctlt.nus.edu.sg/wp-content/uploads/2024/08/Policy-for-Use-of-AI-in-Teaching-and-Learning.pdf
- TLDR: use for homework as you wish, but you are responsible for its outputs (as am I)
  - My **recommendation** is that you try the homework yourself first and then use AI to help check your answers

Plagiarism, cooperate on homework

- The former is strictly not allowed.
- Collaborating on homework is **okay** if acknowledgements are given
  - Teams/Individuals will have to writeup and submit their own work though

Computational Game Theory

Why Game Theory?

Why *Computational* GT?

What are some Applications?

Why Game Theory?

Because you’re not the only smart one in the room

“The formal mathematical study of strategic interactions”

Source: Adobe stock images

The Judgement of Solomon

Two women both claim to be the mother of a baby

King Solomon ordered the baby to be cut into half, with each mother getting one half of the baby

- The first woman accepted the deal
- The second begged Solomon to give the baby to her rival

Solomon judged that the second woman was the mother

- Something about a mother’s love, etc.

Question: Was Solomon wise after all?

- First woman can lie!

Solomon was lucky, not smart. People behave strategically!

Rock, Paper, Scissors

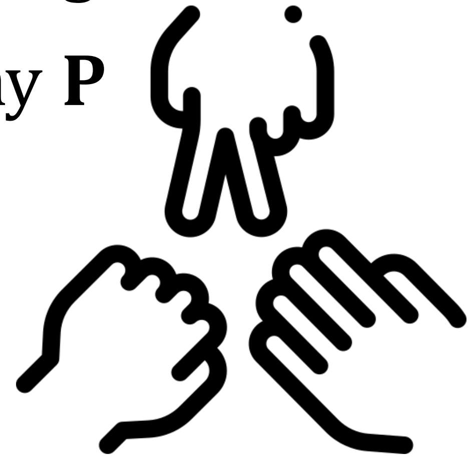
You are playing **R**ock-**P**aper-**S**cissors in a tournament

Based on past matches, **R** is played 50% of the time, **P** and **S** 25%

How should you play to maximize the probability of winning?

If you believe past trends will continue, should always play **P**

- But opponent could think one step ahead: “I should play **S**”
- But you can think two steps ahead: “I should play **R**”
- But opponent thinks three steps ahead: “I should play **P**”
- …, does this cycle ever end?



Accounting for strategic behavior of other players is important!

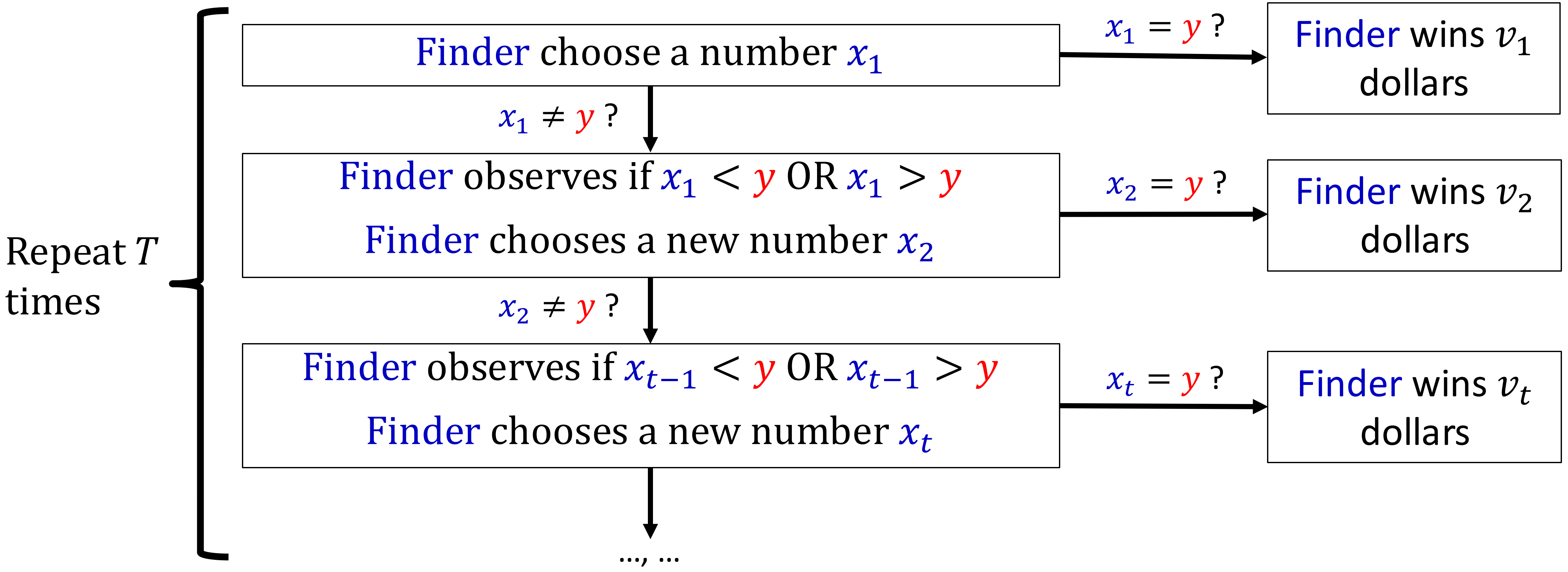
Guess-the-number

Dixit and Nalebuff, The Art of Strategy

Finder

Hider

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.



Repeat T times
Finder choose a number x_1
$x_1 = y$?
Finder wins v_1 dollars
$x_1 \neq y$?
Finder observes if $x_1 < y$ OR $x_1 > y$
Finder chooses a new number x_2
$x_2 = y$?
Finder wins v_2 dollars
$x_2 \neq y$?
Finder observes if $x_{t-1} < y$ OR $x_{t-1} > y$
Finder chooses a new number x_t
$x_t = y$?
Finder wins v_t dollars
…, …

Example: $N = 50, T = 4, v = [80,60,40,20]$

	Finder	Hider
		Chooses $y = 21$ (private!)
$T = 1$	$x_1 = 13$	"too low"
$T = 2$	$x_2 = 24$	"too high"
$T = 3$	$x_3 = 21$	"correct!"

Collect $v_3 = 40$

Note: If no correct guess after T rounds, Finder gets **nothing**

Optimal Finder Strategies?

How should the Finder **maximize its expected total payoff**?

Binary search?

- Maximizes "information gain" after every guess
- Optimal when Hider is chooses y uniformly at random

But Hider is **adversarial**, can mess around with binary search

- Intuition: if first guess is always 25, Hider should just avoid choosing it
  - Hider can choose $y \in \{1, \dots, 50\} \setminus \{25\}$ uniformly at random
  - Expected Finder utility strictly less than against a uniform hider
- Against deterministic Finder, Hider can guarantee that number is **never** guessed
  - Hider: chooses $y = 50$
  - Finder: Guess $x_1 = 25$ → [26, 50], $x_2 = 37$ → [38, 50], $x_3 = 44$→[45,50], $x_4 = 48$ → **FAIL**
- If last guess is randomized:
  - Finder: 25 → [26, 50], 37 → [38, 50], 44→[45,50], Guess [46, 50] u.a.r. → **1/6 prob. of getting 20**

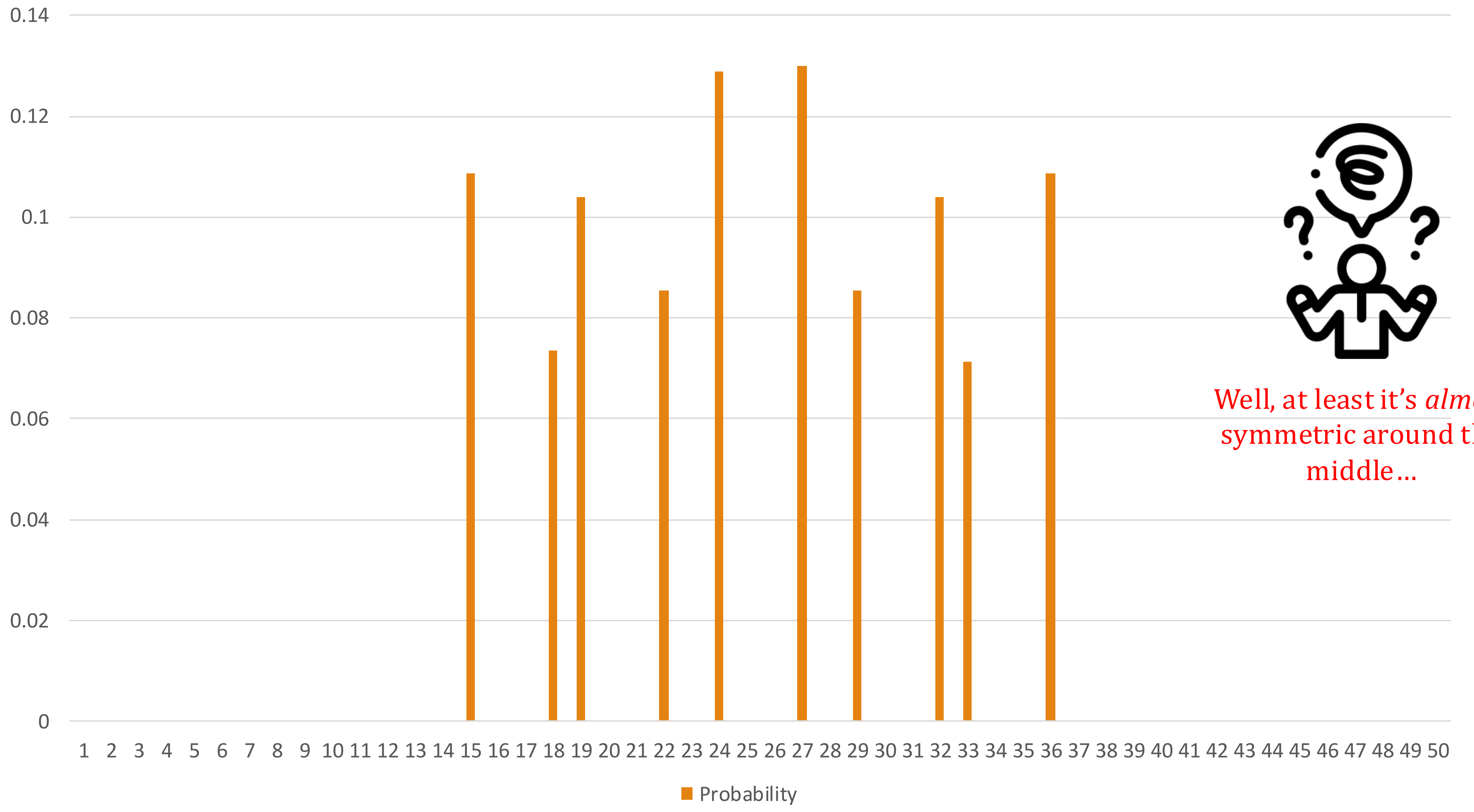
Finder can do a **lot** better

Finder	versus	Hider	Outcome
Binary Search		Uniform	10.4
Binary Search (deterministic)		Anti-deterministic BS	0
Binary Search (randomized a little)		Anti-randomized BS	3.33-3.75
My Optimal Strategy 😎		*Any Strategy*	**10.4**

This is known as the **Nash equilibrium**

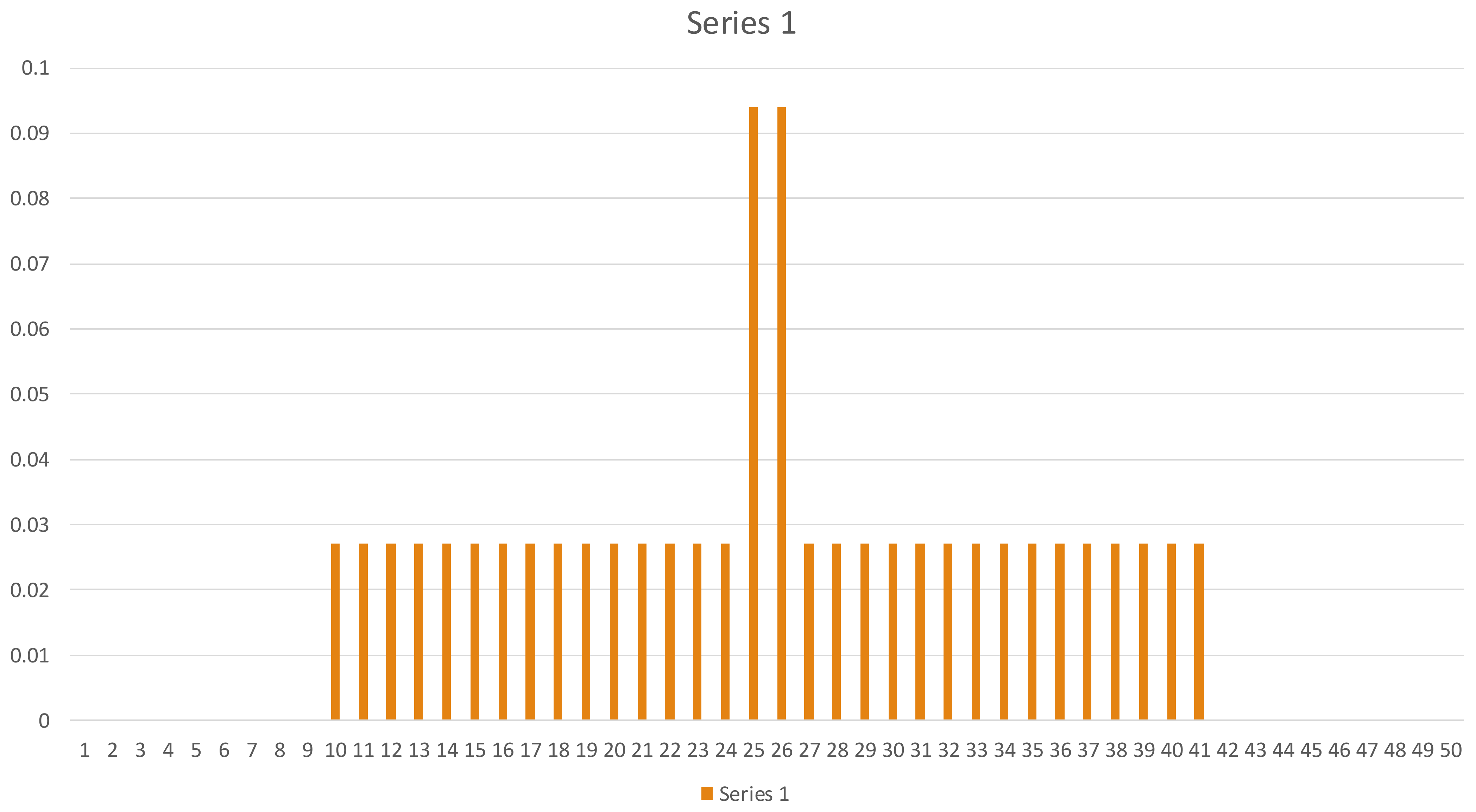
Source code: https://bit.ly/48xvrtg

The Optimal Strategy 😎 for x_1



Well, at least it's *almost* symmetric around the middle…

After adding regularity constraints



In Popular Culture

A Beautiful Mind (2001)

How to play 'Rock, Paper, Scissors, Minus One' | Squid Game: Season 2 | Netflix [ENG SUB]

Squid Game 2 (2024)

Crazy Rich Asians (2018)

"Our brains so hate the idea of losing something that's valuable to us that we abandon all rational thought, and we make some really poor decisions. So, Curtis wasn't playing to win. He was playing not to lose."
-Rachel Chu, *Crazy Rich Asians*

Really…?

Two facets of Game Theory

Mechanism Design (King Solomon)

How can I **design the rules of the game** to achieve my desired social outcome?

Knowing that people can be strategic like the mothers in Solomon's judgement?

Examples: auctions, fair division, truth-telling, market efficiency

Decision Making (RPS)

Given some rules of the game, how can I **play in a way that maximizes my utility**?

Knowing that other players are also thinking the same, e.g., in Rock-Paper-Scissors

This is the focus of this class

Computational Game Theory

Why Game Theory?

You aren't the only smart one in the room

Why *Computational* GT?

What are some Applications?

Why **Computational** Game Theory?

Knowing is not enough; we must apply. Willing is not enough; we must do.

Are games easy to solve?

$$\begin{array}{c|c|cc} & & \multicolumn{2}{c}{\text{Goalkeeper}} \\ & & L & R \\ \hline \text{Shooter} & L & 0 & 1 \\ & R & 1 & 0 \end{array}$$

Penalty shootout, or more commonly known as *Matching Pennies* in Game Theory

By symmetry, optimal is to play L and R **uniformly at random** for both players

The crippled shooter

		Goalkeeper L	Goalkeeper R
Crippled Shooter	L	0	0.5
	R	1	0

50% chance of missing even if goalkeeper dives incorrectly

Shooter wins less if shooting left (e.g., 50% chance to outright miss)

How should the crippled shooter play? Shoot left more, equal, or less frequently?
How should the goalkeeper play? Dive left more, equal, or less frequently?
If both players play optimally, what is the expected utility of shooter?

Reasoning about imperfect information

Simplified Poker

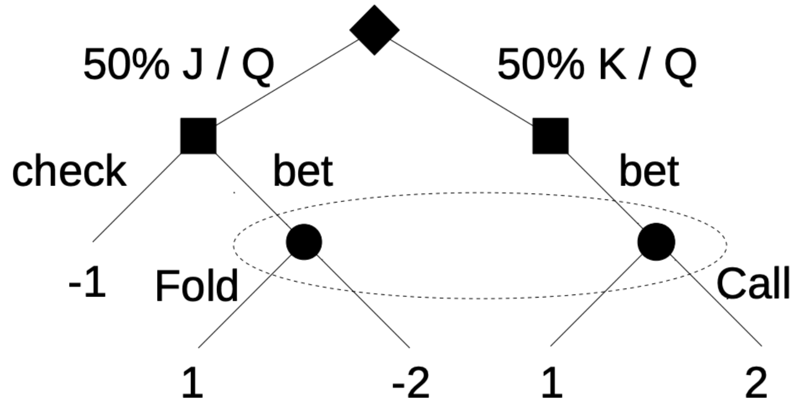
- Opponent needs to guess if we got J or K

Backward induction (minimax search): fold iff we get J

- Expected payoff = 0. Opponent always folds.
- Why? If P1 bets, then it must be holding a K!

If bet 1/3 of the time when holding J

- Expected value = 1/3
- **Bluffing** allows us to earn more!



Reasoning about what to do under imperfect information can be counterintuitive

- Need to be **systematic**

Computational Game Theory

Why Game Theory?

You aren’t the only smart one in the room

Why *Computational* GT?

Game solutions are neither obvious nor easy to compute

What are some Applications?

Application 1: Recreational Games

…also known as what hoodwinked me into studying CS

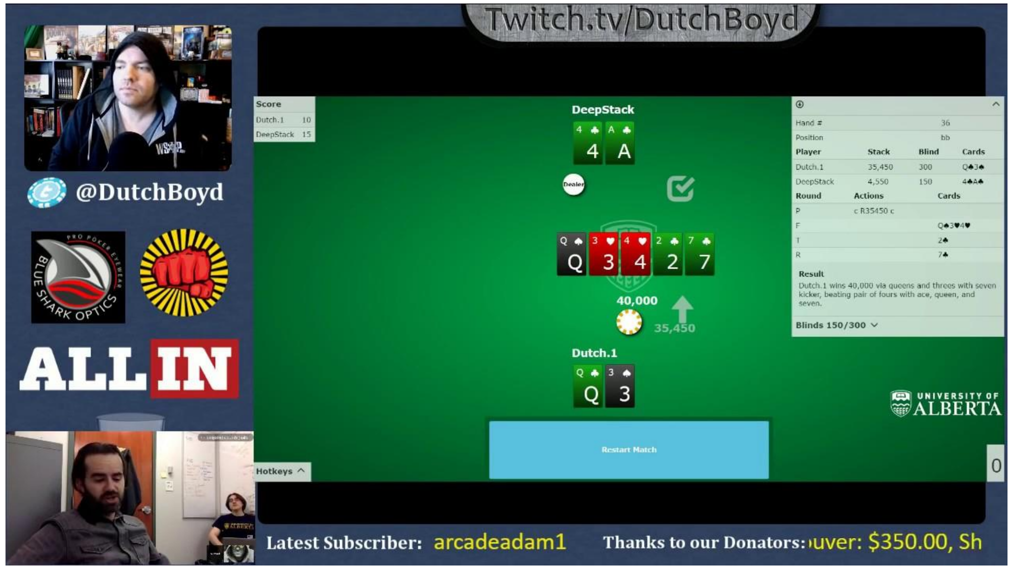
Imperfect vs. Perfect information

Games played at a superhuman level

- Checkers, Chess, Go → Games of perfect information
- "Just" a search problem

Classic benchmark: poker

BRAINS VS. ARTIFICIAL INTELLIGENCE
RIVERS CASINO
Carnegie Mellon University School of Computer Science



Score	
Dutch.1	10
DeepStack	15

Hand #		36	
Position		bb	
Player	**Stack**	**Blind**	**Cards**
Dutch.1	35,450	300	Q♣3♠
DeepStack	4,550	150	4♣A♣
Round	**Actions**		**Cards**
P	c R35450 c		
F			Q♠3♥4♥
T			2♣
R			7♣

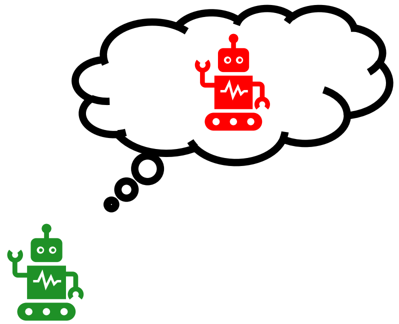
Result
Dutch.1 wins 40,000 via queens and threes with seven kicker, beating pair of fours with ace, queen, and seven.

Blinds 150/300

Latest Subscriber: arcadeadam1 Thanks to our Donators: uver: $350.00, Sh

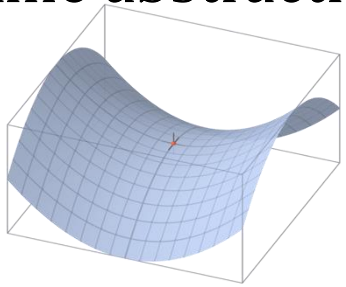
A History of Poker bots

Opponent Modeling

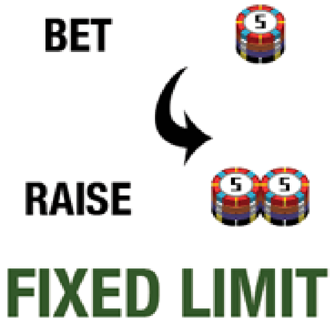


*Optimal strategy may not be deterministic!

Equilibrium Finding [Game abstraction]



Limit Poker solved



Superhuman Performance: Libratus, Deepstack [Search, Function Approximation]

~2000

~2010

2015

Bowling et. al. (2015)

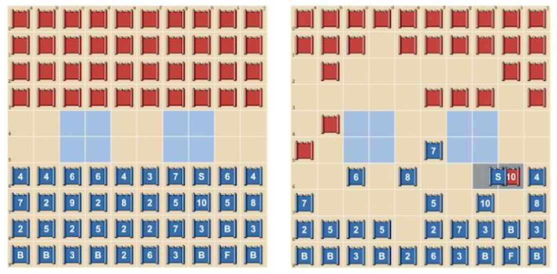
2017

Brown and Sandholm (2017), Moravcik et. al. (2017)

Other milestones!

Multiplayer Poker

Stratego



Diplomacy

~2019

Brown and Sandholm

2022

Perolet et. al.

late 2022

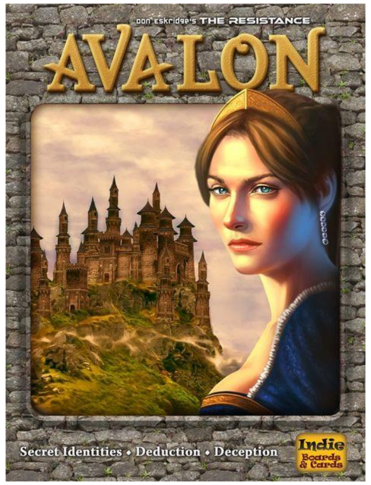
Brown et. al.

Complex, "continuous time" games

Multiplayer games

Hidden identity games

Team games


Don Eskridge's THE RESISTANCE
AVALON
Secret Identities • Deduction • Deception
Indie Boards & Cards

There is non-trivial theory involved. NOT as simple as just "training a neural network"

Application 2: Security

Or, how to break things. Two sides of the same coin. Your choice.

US Coast Guard

LAX Airport Security

Anti-poaching

Would like to patrol/guard everywhere but cannot practically do so. Can randomization help?

See Pita et. al., https://doi.org/10.1609/aimag.v30i1.2173, https://rdcu.be/dQMes
https://www.airport-technology.com/features/featuregame-theory-airport-security-teamcore-stackelberg/?cf-view

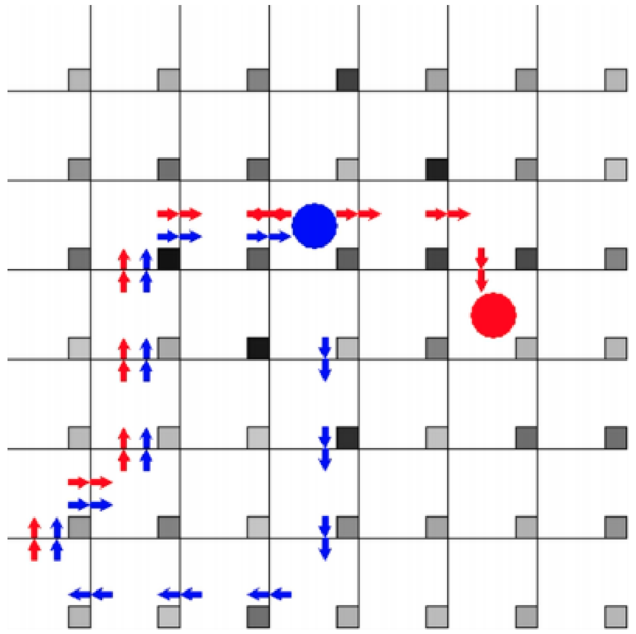
Modeling Wildlife Protection

Poachers:

- Place snares and traps
- Leaving footprints, trails

Patrollers:

- Want to catch poachers



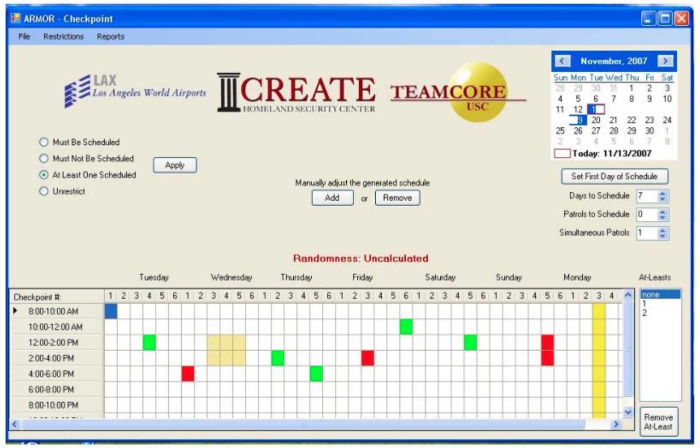
··· Patroller's route

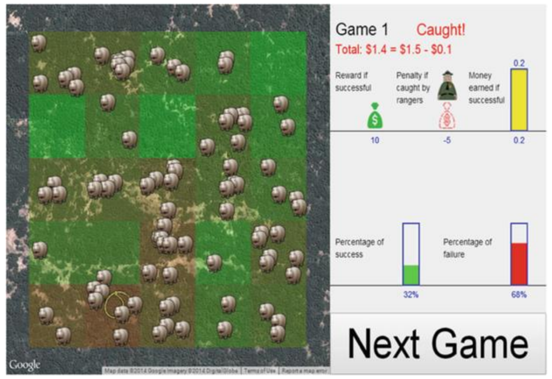
··· Poacher's route

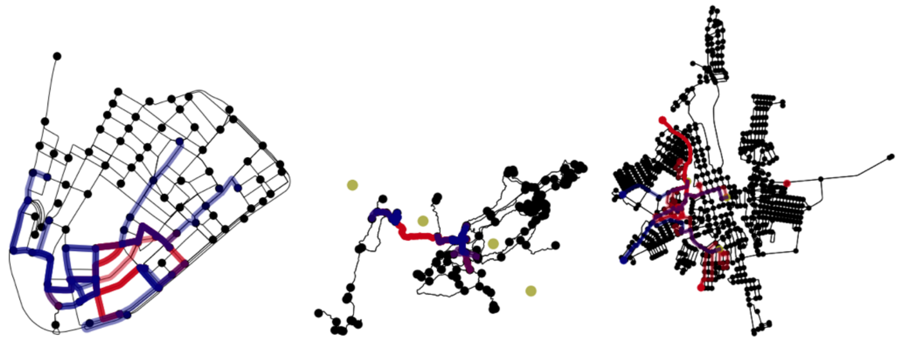
■ Animal density (darker=higher)

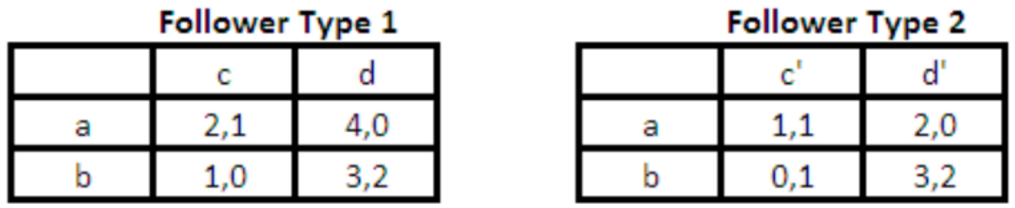
See Wang et. al., Deep Reinforcement Learning for Green Security Games with Real-Time Information

Game solving via optimization






(a) L. Manhattan (b) Minnewaska SP (c) City of Bakhmut



Follower Type 1

	c	d
a	2,1	4,0
b	1,0	3,2

Follower Type 2

	c'	d'
a	1,1	2,0
b	0,1	3,2

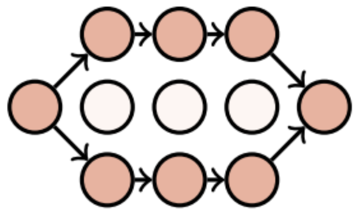
Figure 3: Security Agent vs Followers 1 and 2

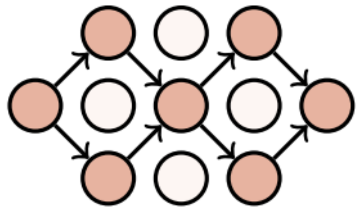
$$
\begin{aligned}
\max_{x,q,a} \quad & \sum_{i\in X}\sum_{l\in L}\sum_{j\in Q} p^l R^l_{ij} x_i q^l_j \\
\text{s.t.} \quad & \textstyle\sum_{i\in X} x_i = 1 \\
& \textstyle\sum_{j\in Q} q^l_j = 1 \\
& 0 \le (a^l - \textstyle\sum_{i\in X} C^l_{ij} x_i) \le (1-q^l_j)M \\
& x_i \in [0\dots 1] \\
& q^l_j \in \{0,1\} \\
& a \in \Re
\end{aligned}
$$

Extensions to sequential settings are also possible

Layered Graph Security Games

https://arxiv.org/abs/2405.03070





Chooses a path P1

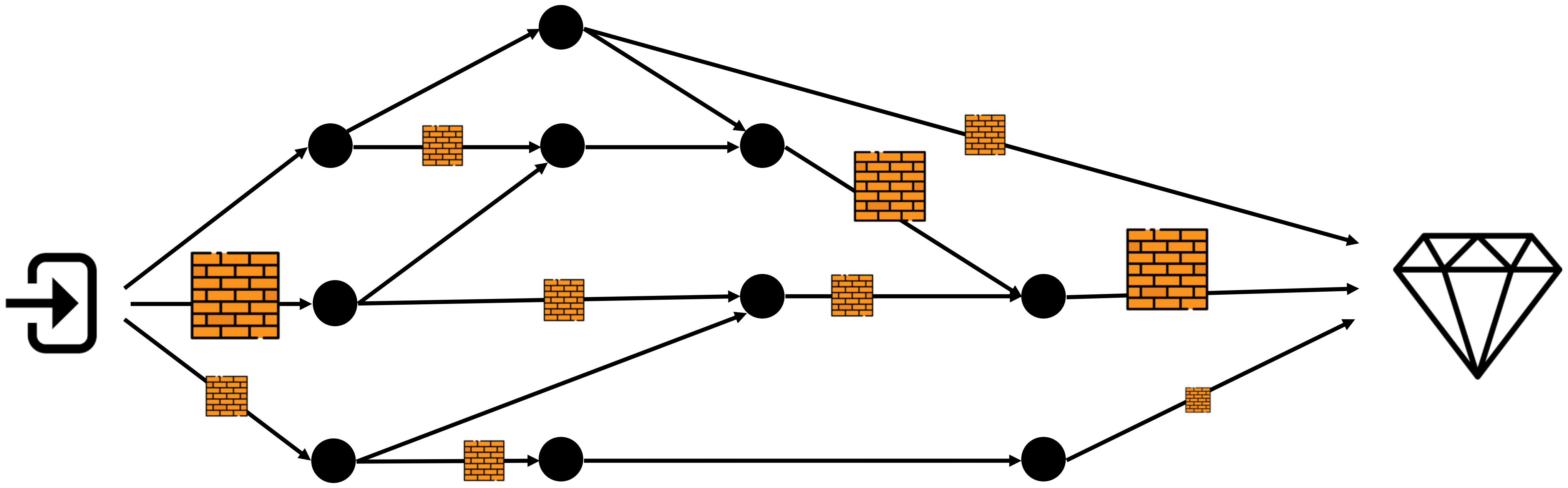
Chooses a path P2

If P1 shares edge with P2, P1 wins, else
P2 gets some payoff (dependent on P2)

- Very general formulation
  - Some types of pursuit evasion (Red avoids getting caught)
  - Patrolling games, Anti-terrorism, Logistical interdiction (Blue prevents Red from performing or reaching goal)
  - Persistent threats (Red gets more payoff the longer it survives)

Example: Network Protection Game

*Abstracted and super simplified



● Nodes: Machines, servers, routers

➡ Edges: Potential vulnerabilities

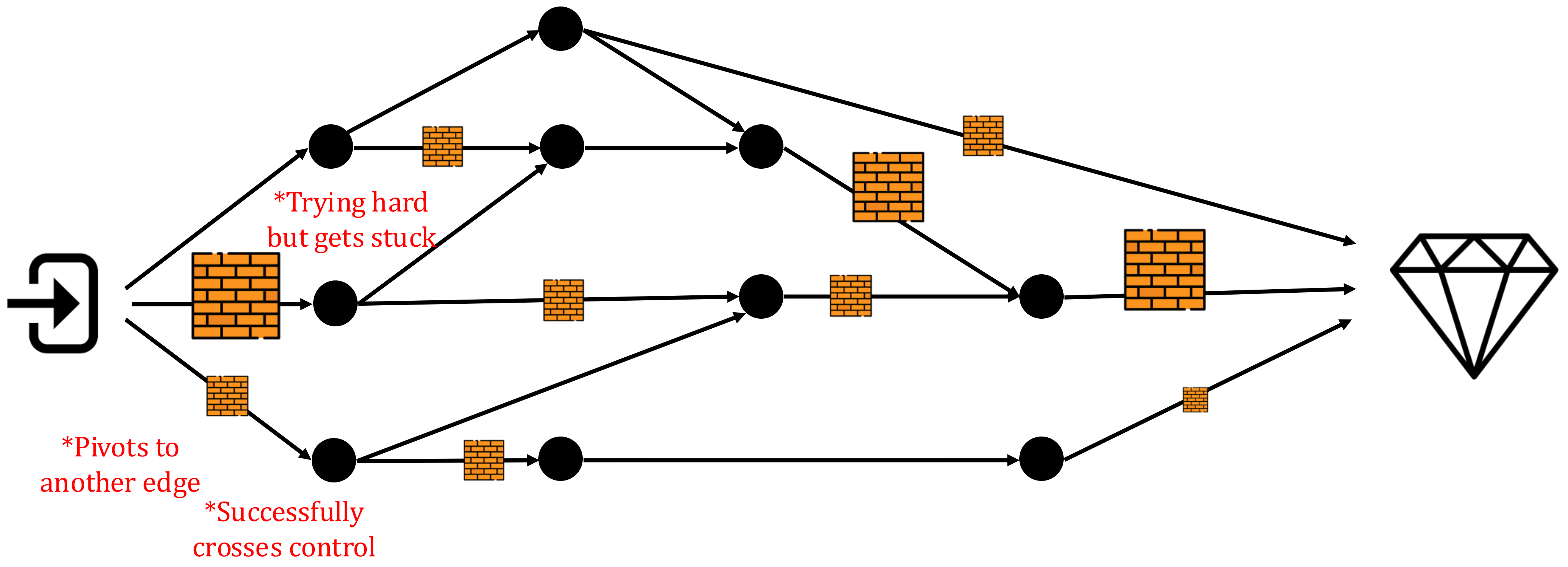
▨ Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Reduces to a longest shortest path problem

(super easy for simple constraints, easily scale to around tens of millions of nodes)

Pivoting: Intruder is not stupid!

https://lingchunkai.github.io/pdf/AICS-2025.pdf



● Nodes: Machines, servers, routers

➡ Edges: Potential vulnerabilities

▨ Walls: Defenses controls to be beefed up, subject to budget/performance constraint

Given intruder's action **changes with time as they explore the network,** how should defenses be placed?

*Interesting connection with the Gittins index, ongoing work

Application 3: Robust Optimization

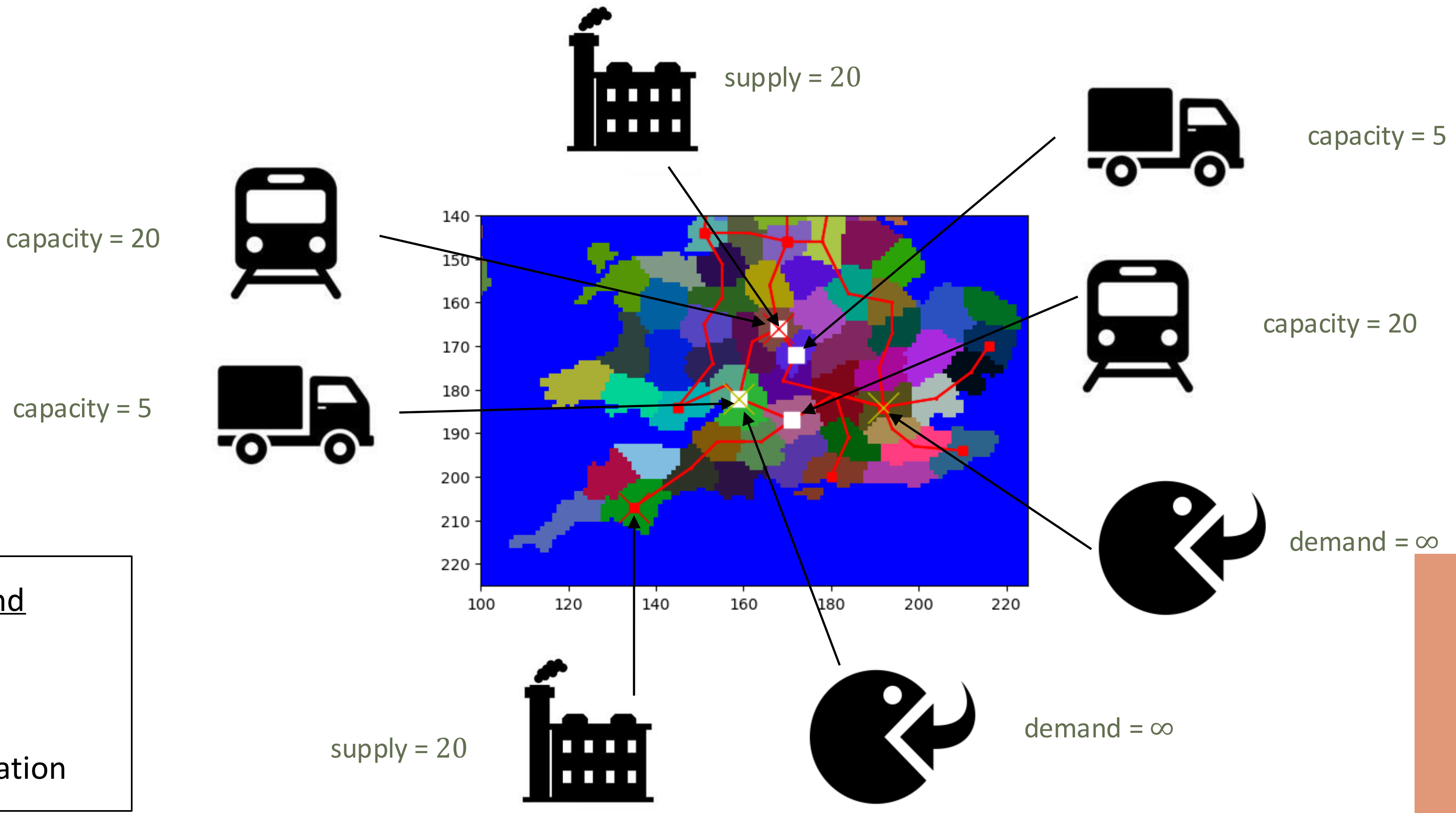
I guarantee this isn’t as boring as it sounds

Application 3: Contested logistics

Logistics player **chooses paths** for each vehicle

Red player chooses 2 ‘edges’ to **interdict**

Goal: Maximize expected demand met after **10 timesteps**



Legend
--: Rails
X: Supply
X: Demand
□: Start location

How do we solve this?

Really big game

- Number of joint paths over vehicles is **doubly exponential**

A Nash strategy of support size 4 for each player was found

- **Guaranteed** expected utility of 9.259
- Blue, red strategy = [4/9, 1/9, 5/21, 13/63], [7/18, 1/9, 7/18, 1/9]
- Naïve strategies perform much worse

Red Strategy (minimizer)

Blue Strategy (maximizer)

13.33 >	13.33 >	6.67 >	0
10	10	10	3.33
5	5	11.67	20
5 <	5 <	11.67 <	20

Note: even though they look the same, these 2 blue strategies do **not** perform the same against red strategies **not shown** in table

Application 4: Nontransitive Rankings and Preferences

Evaluation of "Agents"

How do we rate chess players?

- n players, play many rounds against each other, want to assign score based on the probability of beating other players (usually empirical)

ELO score

$$\hat{p}_{ij} := \frac{10^{r_i/400}}{10^{r_i/400} + 10^{r_j/400}} = \sigma(\alpha r_i - \alpha r_j) \quad \sigma(x) = \frac{1}{1+e^{-x}} \quad \textbf{and} \quad \alpha = \frac{\log(10)}{400}$$

Problems? Non-transitivity!

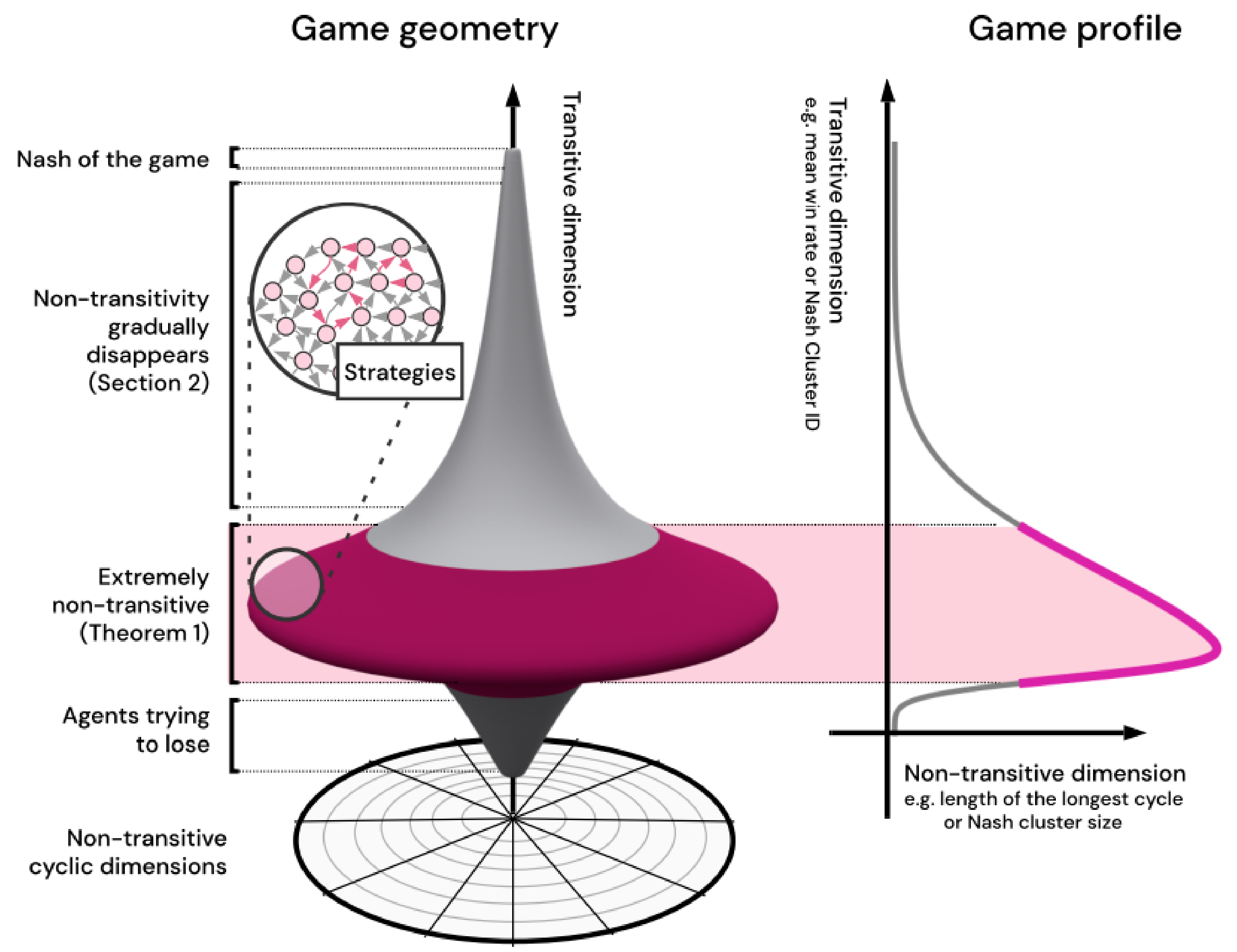
- A beats B, B beats C does *not* imply A beats C!

Nash Averaging

- Calculate (maximum entropy) *equilibrium* of a *zero-sum* game
- Nice properties, e.g., adding the same agent does not affect it's "worth"

See Balduzzi et. al.: Re-evaluating Evaluation

Aside: Games of Skill Hypothesis


Game geometry
Game profile
Nash of the game
Non-transitivity gradually disappears (Section 2)
Extremely non-transitive (Theorem 1)
Agents trying to lose
Non-transitive cyclic dimensions
Strategies
Transitive dimension
Transitive dimension
e.g. mean win rate or Nash Cluster ID
Non-transitive dimension
e.g. length of the longest cycle or Nash cluster size

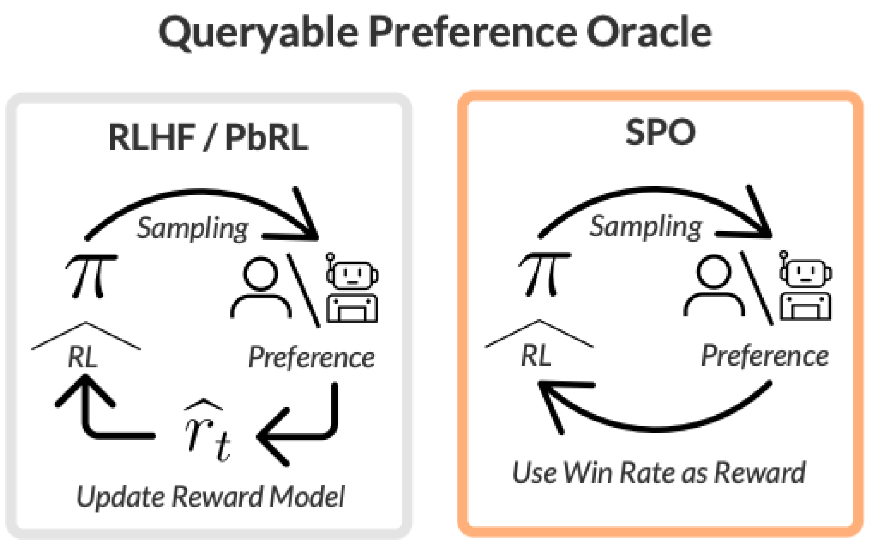
See Czarnecki et. al.: Real World Games Look Like Spinning Tops

Preference Based Optimization/RL

Problem with RLHF: non-transitive preferences

- In RLHF, need to estimate “reward” from preferences
- Can use preference matrix as win-loss matrix
- Use local regret minimizers (will learn later in this class) to optimize losses

	a	b	c	d
a	0	$+1$	$+1$	-1
b	-1	0	$+1$	-1
c	-1	-1	0	$+1$
d	$+1$	$+1$	-1	0



See Swamy et. al.: A Minimaximalist Approach to Reinforcement Learning from Human Feedback

Application 5: Solving Cute Puzzles

Saving the best for the last

A common finance interview problem

Deck of 52 cards, 26 **B**lack + 26 **R**ed. Start with 100 dollars

Cards are drawn in sequence without replacement over t rounds

- Before draw t, place a bet $x_t \geq 0$ and guess if the card is B or R
- Cannot bet more than what you have
- If guess is correct, get x_t, if not, lose x_t

Q1: How to maximize **expected** money at the end?

- Go all in every round (prove it!)

Q2: Two players are playing separate games, having more money at the end wins. How to maximize **probability** of winning?

- Just need to beat opponent, anything more doesn't help
- Similar to what are known as *General Blotto* games

Help save the prisoners!

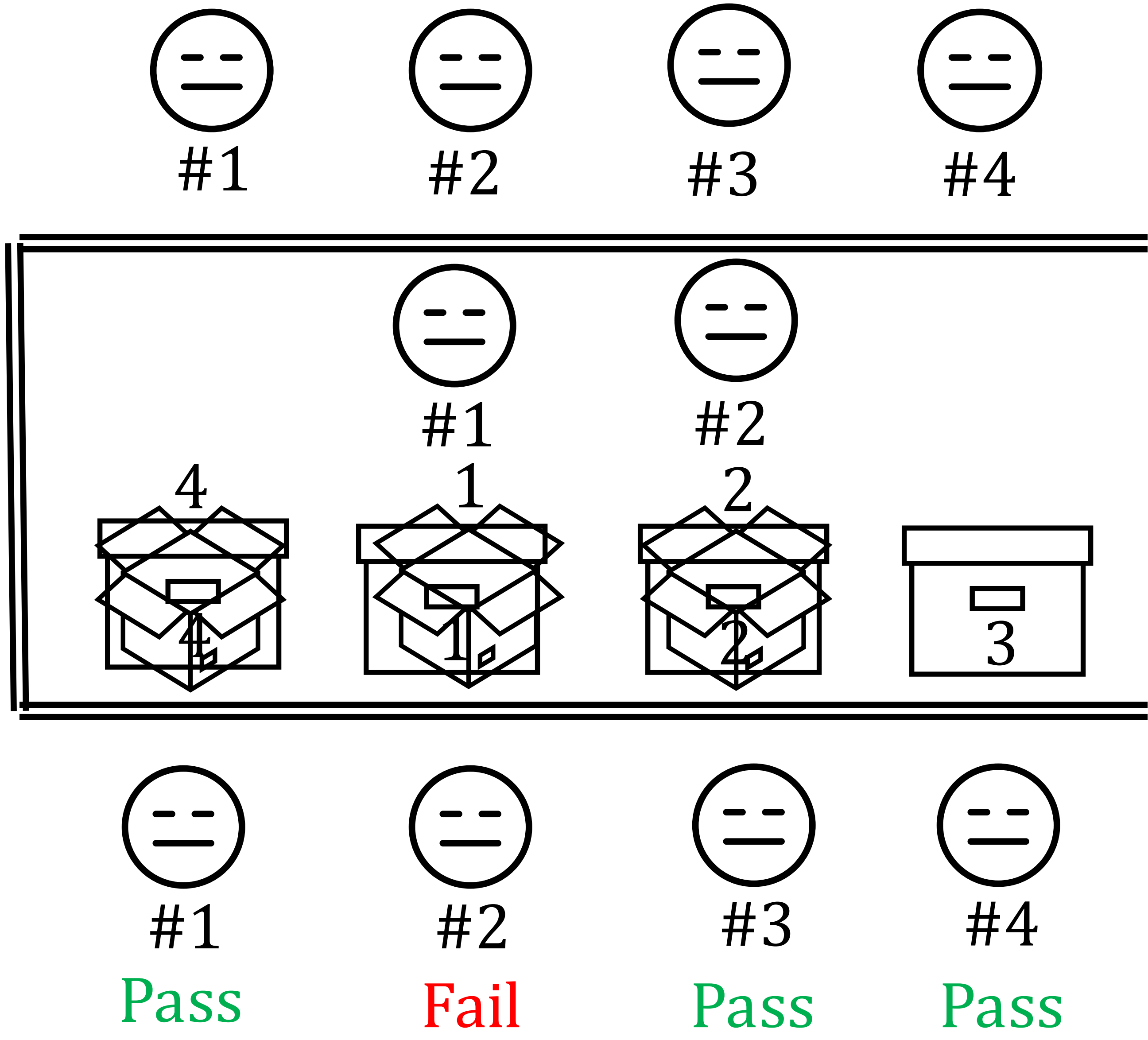
n=4 prisoners, each given a unique number in [1, n]

Room with n closed boxes, each contains a unique num in [1, n]

Each prisoner takes turns entering the room and chooses n/2=2 boxes to open

- If chosen boxes contains his number, he *passes*, if not, he *fails*
- No message left in the room for other prisoners

https://en.wikipedia.org/wiki/100_prisoners_problem



Help save the prisoners (contd.)

#1	#2	#3	#4
Pass	Pass	Pass	Pass

If all prisoners pass, they are freed

#1	#2	#3	#4
Pass	Fail	Pass	Pass

If just one prisoner fails, all executed

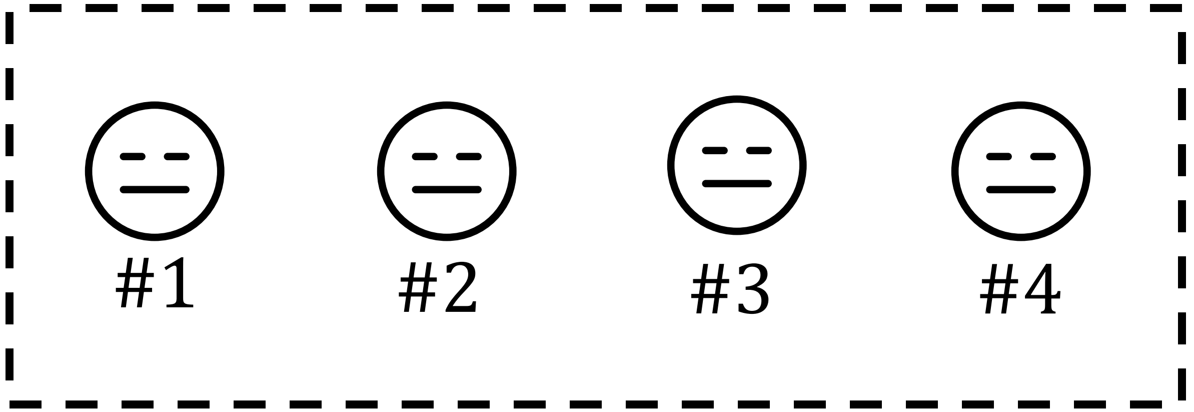
Without coordination, chance of getting free = $1/2^n$=1/16

With coordination, chance of being freed $= 0.5$ when $n = 4$
More generally, there exists a strategy averaging $\sim$30% for all n

Method is randomized, requires knowledge of permutation chains

What if we don't have a mathematician?

Can be formulated as a *team* game!



v.s.

Team of n players

Adversary placing numbers

Can apply generic team game solvers to get optimal solution!

Using GT to help in in theory research

Very common situation in Theory research: need to come out with bounds on the performance of an algorithm

Can think of an adversary choosing a (distribution over) problem instances, while we are choosing a (randomized) algorithm

Sometimes you can restrict the set of problem instances into a smaller space (with finite size), solve it as a game, which will then show the "restricted worst case" instance

Solve large enough games with this, helps with spotting "patterns" in the worst problem instances and eventually proving bounds

- I've done this before in my own research ☺

See Choo and Ling: A short note about the learning-augmented secretary problem

Summary

Why Game Theory?

You aren't the only smart one in the room

Why *Computational* GT?

Game solutions are neither obvious nor easy to compute

What are some Applications?

Recreational games, but also real-world uses and deployments

Open to collaborating on any interesting problems

What I’m interested in

… because everyone has their own agenda

I’m looking for students!

“So far the field has struggled to take techniques like this *out of the laboratory and game environments and into the real world…”*

-Gary Marcus on Alphastar, 2019

Game Theory in the real-world

Traditional → More modern

1. Algorithms & Structure

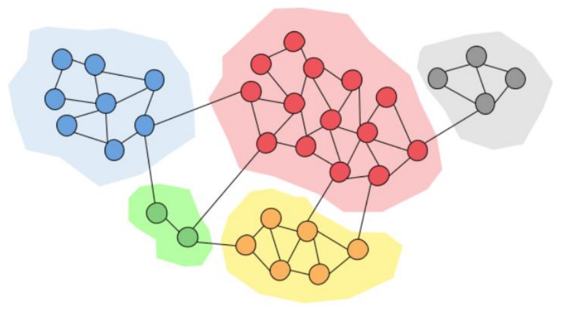
2. Interpretability & sensemaking

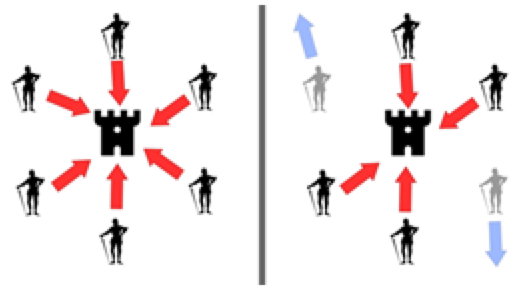
3. Challenging behavioral assumptions

Direction 1: Algorithms & Structure

Real world problems are tough (large) but have lots of structure!

Ridesharing


Social media analysis


Distributed systems

Logistics

Cybersecurity

Politics

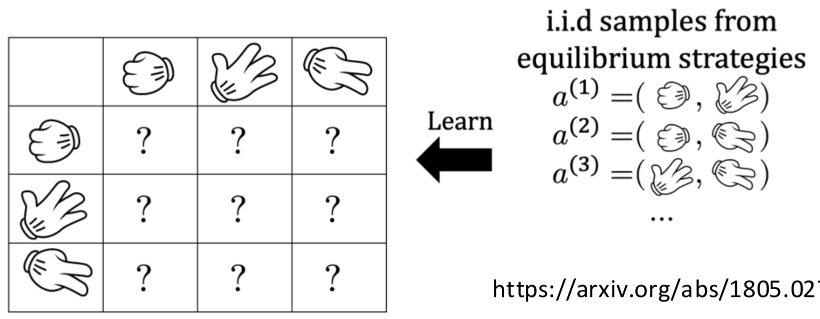
- Multiplayer, team games
- General-sum
- Imperfect information
- Jointly continuous & discrete
- Huge, combinatorial structure

Direction 2: Sensemaking in Games

Making sense of player strategies can be difficult!

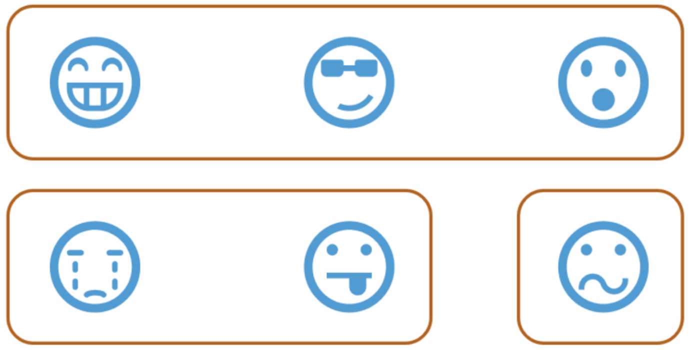
Inverse Game Theory/Computational Rationalization

- Explain why players are behaving the way they do



https://arxiv.org/abs/1805.02777

https://arxiv.org/abs/1903.04101



https://arxiv.org/abs/2312.09058

IGT via differentiable optimization

Coalition Structure Learning

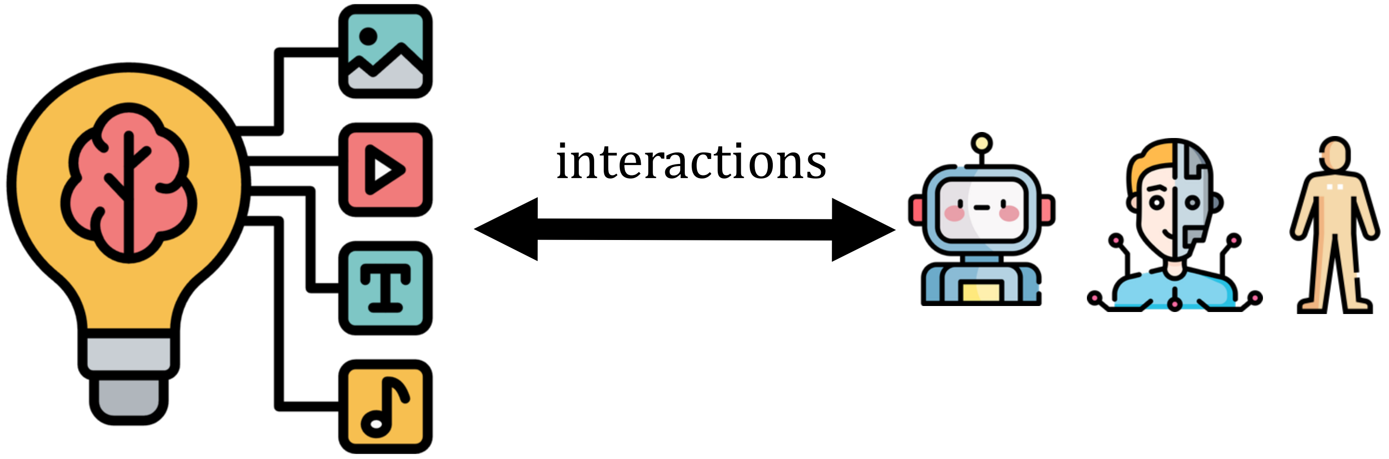
Learning strategies that are interpretable/operationalizable

- Restriction to strategies that are compact e.g., product distributions, sparse
- Regularization to "human-like" strategies

Direction 3: Challenging Assumptions

Tackle new *problems* by allowing interactions in human modalities

- Example: language models used in bargaining, negotiations
- How to incorporate strategy into such high-modality action spaces?



Handling player bounded rationality

- Fusion with opponent modeling and exploitation
- What about games without common prior?
- Are equilibrium even the right concept?

Shameless Plug: CS6101

If you are interested in doing lab rotation with me…

- Speak to me after the lecture
- OR email me

#46: Topics in Game Theory, Strategy and Multiagent Systems

Do so **before** registering!

Summary

Why Game Theory?

You aren't the only smart one in the room

Why *Computational* GT?

Game solutions are neither obvious nor easy to compute

What are some Applications?

Recreational games, but also real-world uses and deployments

Open to collaborating on any interesting problems

About **this** class

What’s in it for you?

What this class is about

How to **model** problems as games

Multi-agent decision making as **equilibrium finding**

Common **computational** approaches towards game solving
- Basic analysis of these algorithms

Lectures will **focus on exposure** rather than depth
- Topics are chosen based on what I think is important for research
- Need **input from you**: what would you like to learn? Do the survey!

Who should take this class

Targeted at PhD students, students interested in research

- I am **looking for PhD students**

People who work on multiagent-related areas

People interested in RL, robotics

- We will focus on multiagency, but not so much RL

People who work in theory or optimization

- Online learning, control theory, integer programming, convex optimization

People who work on general AI/ML and want to explore more

- e.g., strategic classification, human-computer interaction, collaborative AI
- A fair bit of misinformation out on the internet
- If you are working in multiagent/game theory + X, **I'm interested in learning more!**

Topics Covered

Equilibrium concepts: Nash, Stackelberg, Correlated

Sequential decision making
- Handling imperfect information, Extensive form games
- Focus on zero-sum games and Nash equilibrium

Computation
- Mathematical programming (LPs, MILPs)
- Online learning + Self-play
- Subgame solving
- Double-oracle methods

Stuff I want you to know well by the end of the class

Related work
- Markov/Stochastic games, Multiagent RL
- If there is a topic that you want me to cover, let me know and I'll try to accommodate

What this class is **not** (about)

Writing lots of code, running experiments, tuning hyperparameters

General ML, Deep Learning, GenAI, AGI

A substitute for mechanism design
- take Warut's class (CS4261, CS5461) for this

A substitute for an optimization class

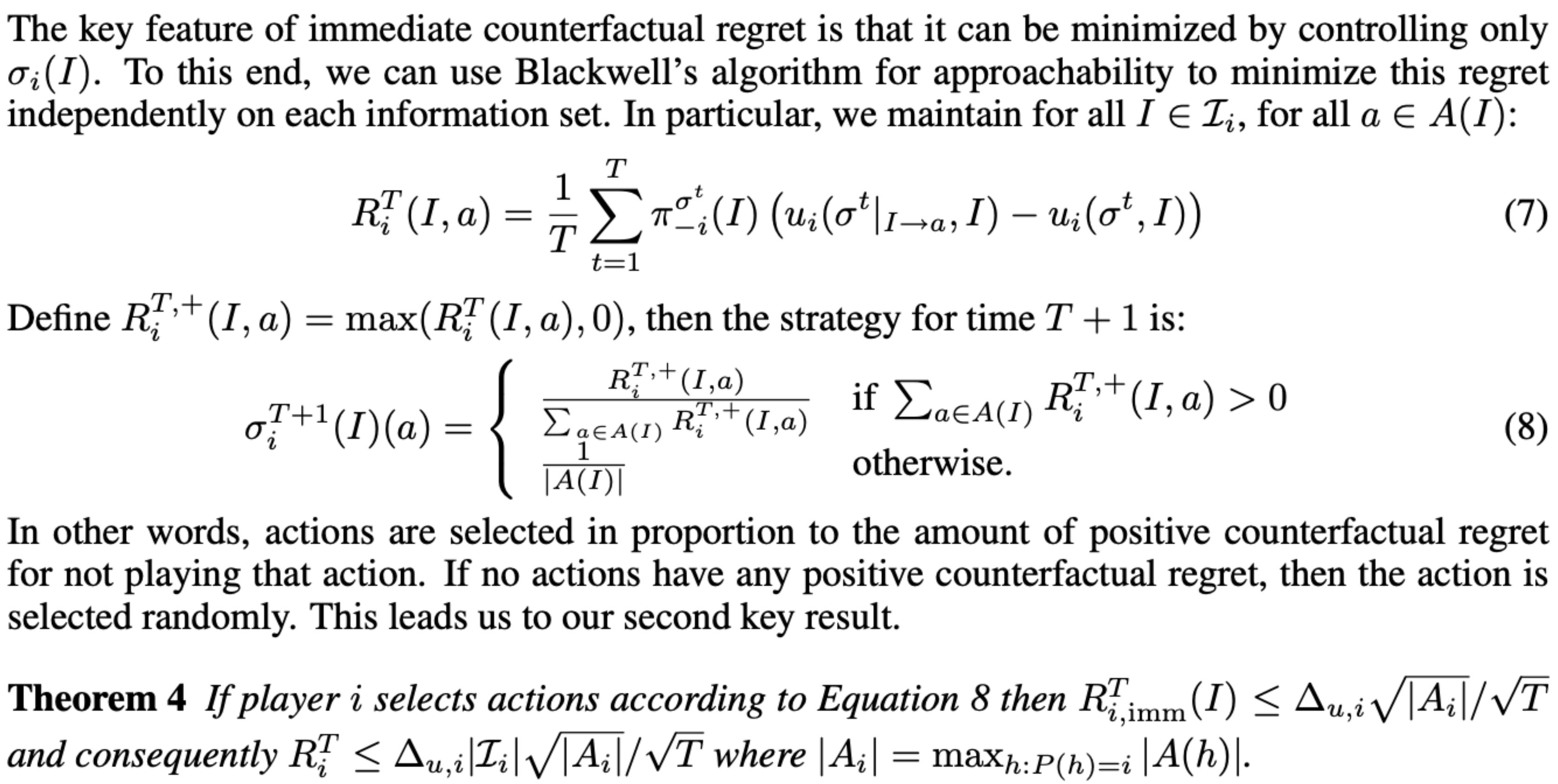
Building a superhuman bot (don't have enough resources/time)
- Contact me outside class to discuss if you are interested

If you are here **primarily** for any of the above reasons, I strongly encourage you to **take another class**

Example of what we will study

$$\min_{x\in\mathcal{X}}\max_{y\in\mathcal{Y}} x^T A y$$

We will be looking a lot at these sorts of saddle-point problems



The key feature of immediate counterfactual regret is that it can be minimized by controlling only $\sigma_i(I)$. To this end, we can use Blackwell's algorithm for approachability to minimize this regret independently on each information set. In particular, we maintain for all $I \in \mathcal{I}_i$, for all $a \in A(I)$:

$$R_i^T(I,a) = \frac{1}{T}\sum_{t=1}^{T} \pi_{-i}^{\sigma^t}(I)\left(u_i(\sigma^t|_{I\to a}, I) - u_i(\sigma^t, I)\right) \qquad (7)$$

Define $R_i^{T,+}(I,a) = \max(R_i^T(I,a), 0)$, then the strategy for time $T+1$ is:

$$\sigma_i^{T+1}(I)(a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{a\in A(I)} R_i^{T,+}(I,a)} & \text{if } \sum_{a\in A(I)} R_i^{T,+}(I,a) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise.} \end{cases} \qquad (8)$$

In other words, actions are selected in proportion to the amount of positive counterfactual regret for not playing that action. If no actions have any positive counterfactual regret, then the action is selected randomly. This leads us to our second key result.

Theorem 4 *If player i selects actions according to Equation 8 then $R_{i,\mathrm{imm}}^T(I) \le \Delta_{u,i}\sqrt{|A_i|}/\sqrt{T}$ and consequently $R_i^T \le \Delta_{u,i}|\mathcal{I}_i|\sqrt{|A_i|}/\sqrt{T}$ where $|A_i| = \max_{h:P(h)=i}|A(h)|$.*

Regret Minimization in Games with Incomplete Information (2007, Zinkevich et. al.)

Counterfactual Regret Minimization: One of the fundamental algorithms used for game solving

Things to do

Drop the class if you are no longer interested

Fill in the demographic survey

Contact me after this or by email if you’d like to learn more about my CS6101 lab rotation

Next week: equilibrium concepts

For students looking to audit the class, please contact me!