Variable-Player Learning for Simulation-Based Games

Madelyn Gatchel

Advisor: Bryce Wiedenbeck

Outline

- **Introduction and Motivation**
- Background
- Related Work
- Variable-Player Games
- Model: Approximating robust symmetric Nash equilibria
- Analysis: Equilibrium robustness metrics
- Experiments
- Conclusion & Ongoing Work

Introduction

- **Game theory**: branch of economics that aims to model how people or "agents" interact and make decisions

- **Machine learning**: branch of computer science which uses mathematical techniques to learn functions from data

- **Thesis**: Use machine learning to analyze large, symmetric, variable-player simulation-based games

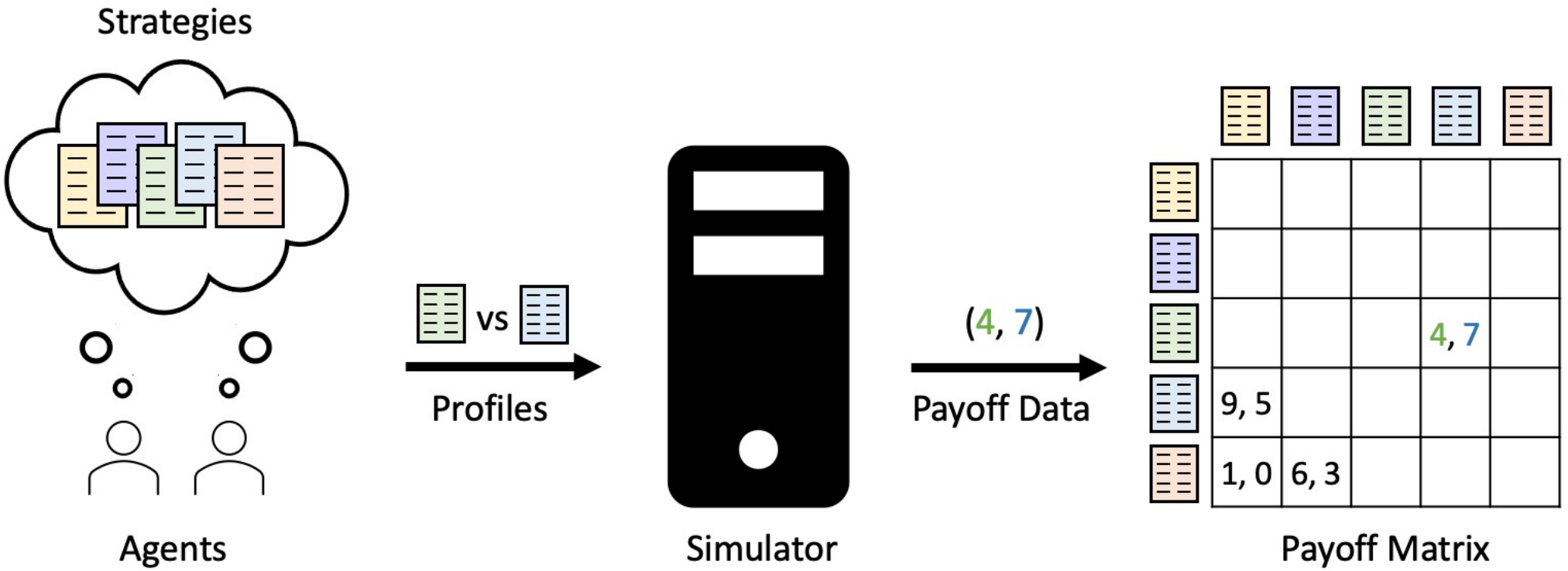
Normal-Form Games

- Type of simultaneous-move game
- Fixed set of players
- For each player
  - Set of strategies
  - Utility function
- Represented using payoff matrix

Rock-Paper-Scissors

		Player 2		
		R	**P**	**S**
Player 1	**R**	0, 0	-1, 1	1, -1
	P	1, -1	0, 0	-1, 1
	S	-1, 1	1, -1	0, 0

Simulation-Based Games



Motivation

- Applications of SBGs
  - Stock market
  - Cybersecurity
  - Credit networks
  - Trading agent competitions
- Likely that the number of players is **large** and **unknown**

https://www.losaltosonline.com/news/sections/business/185-business-columns/62148-how-the-stock-market-has-reacted-to-pandemics

Sokota et al. [11]

Thesis

Thesis Essential Questions

- How do we construct a variable-player game-theoretic model?

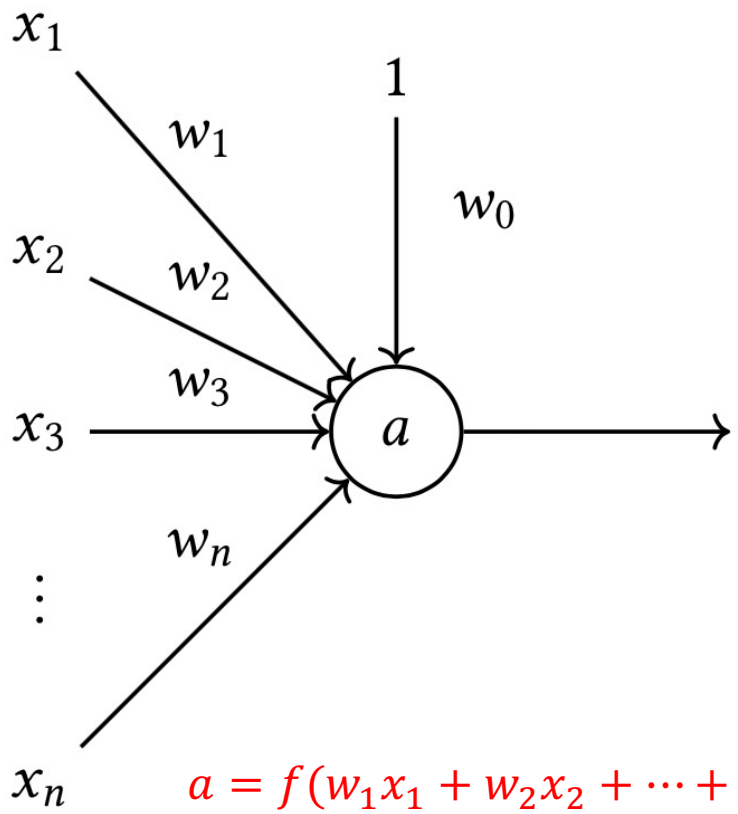
- How do we analyze a variable-player game-theoretic model?

Outline

- Introduction and Motivation
- **Background**
  - **Deep learning**
  - **Game theory**
- Related Work
- Variable-Player Games
- Model: Approximating robust symmetric Nash equilibria
- Analysis: Equilibrium robustness metrics
- Experiments
- Conclusion & Ongoing Work

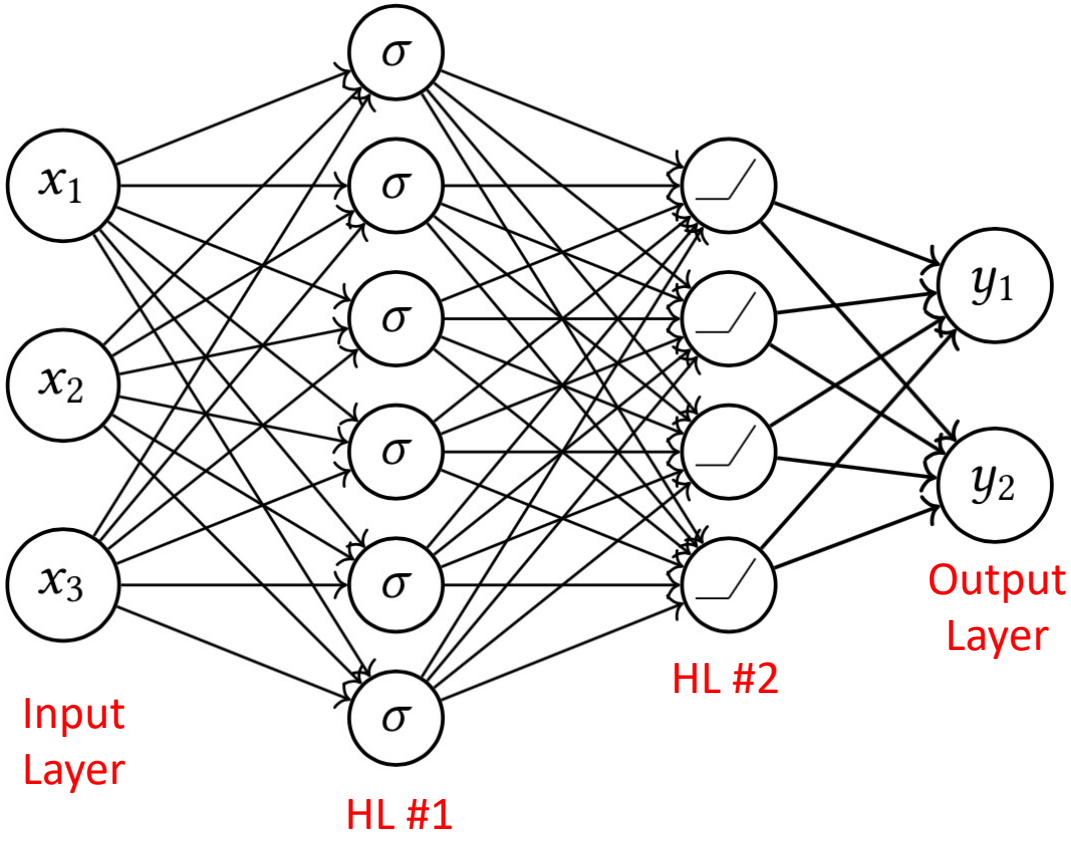
Background: Deep Learning

Artificial Neuron



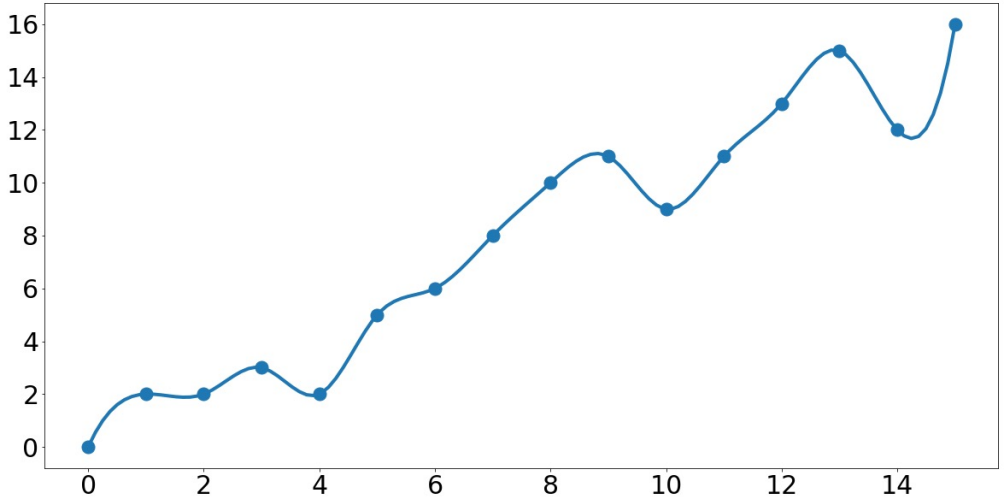
$$a = f(w_1x_1 + w_2x_2 + \cdots + w_nx_n)$$

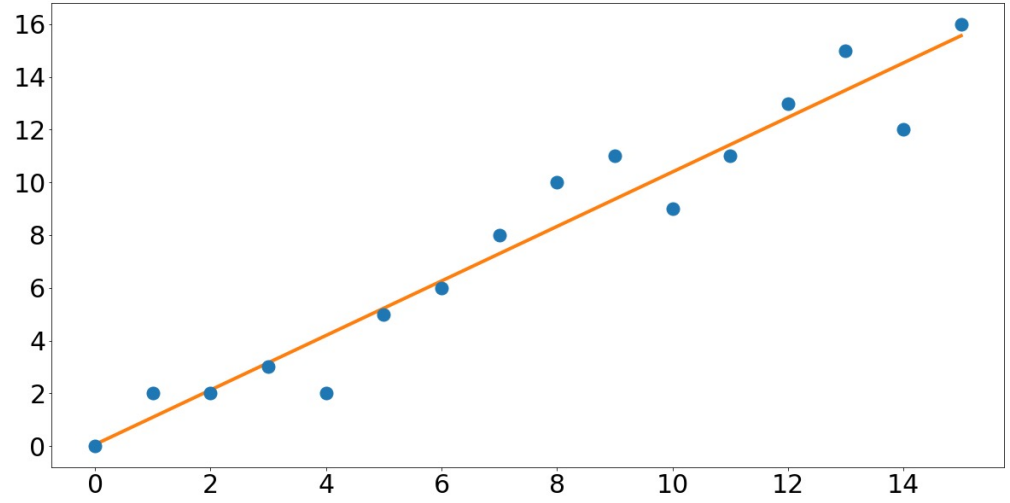
Neural Network



Background: Deep Learning (*cont.*)

- **Training phase:** Optimize neural network *parameters*
  - Objective: minimize error
- **Validation phase:** Optimize neural network *hyperparameters*
  - Objective: model generalizes well to new data
- **Testing phase:** Make predictions on new, unlabeled data





Background: Game Theory

- A **normal-form game** is a tuple $\tau_n = (P, S, u)$ where
  - $P = \{1, \ldots, n\}$
  - $S = S_1 \times \cdots \times S_n$
  - $u : S \mapsto \mathbb{R}^n$

Rock-Paper-Scissors

- $P = \{1, 2\}$
- $S_1 = \{\mathbf{R}, \mathbf{P}, \mathbf{S}\};\ S_2 = \{\mathbf{R}, \mathbf{P}, \mathbf{S}\}$
- $S = S_1 \times S_2$
- $S = \{(\mathbf{R}, \mathbf{R}), (\mathbf{R}, \mathbf{P}), (\mathbf{R}, \mathbf{S}),$
  $(\mathbf{P}, \mathbf{R}), (\mathbf{P}, \mathbf{P}), (\mathbf{P}, \mathbf{S}),$
  $(\mathbf{S}, \mathbf{R}), (\mathbf{S}, \mathbf{P}), (\mathbf{S}, \mathbf{S})\}$
- u is represented by payoff matrix

Types of Strategies

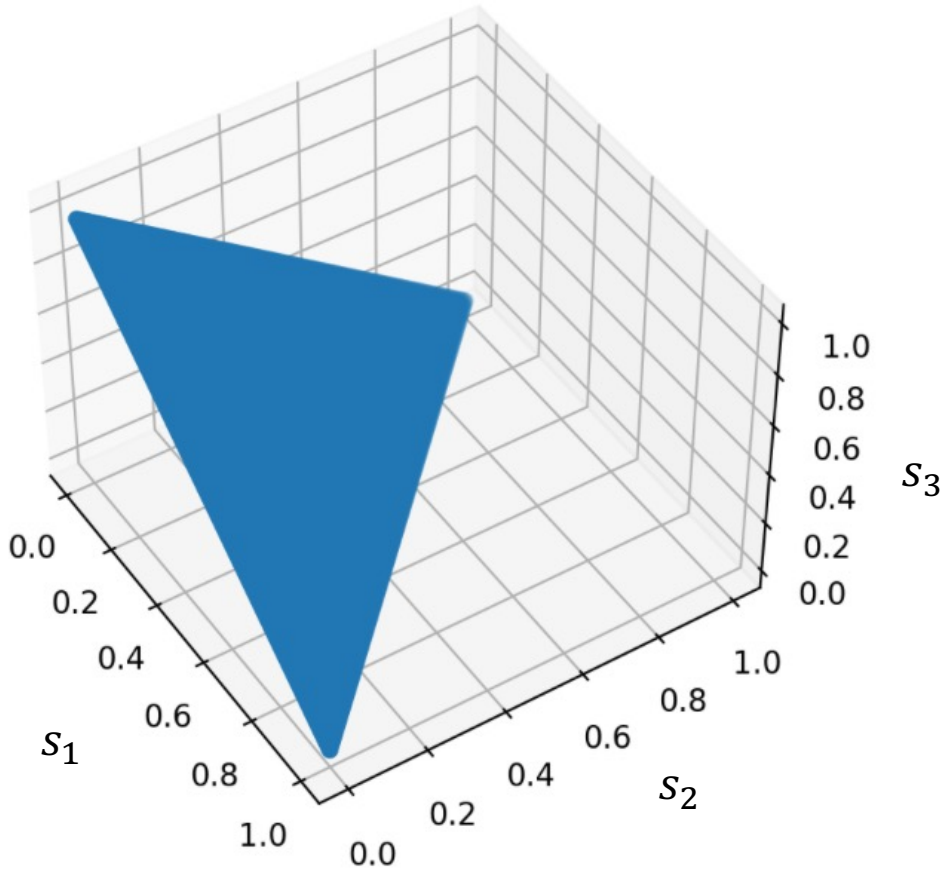
- **Pure strategy:** any $s \in S_i$
- **Pure-strategy profile:** $\vec{s} = (s_1, \dots, s_n)$
- **Mixed strategy:** probability distribution over actions, denoted by σ
- **Mixed-strategy profile:** $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$

Rock-Paper-Scissors

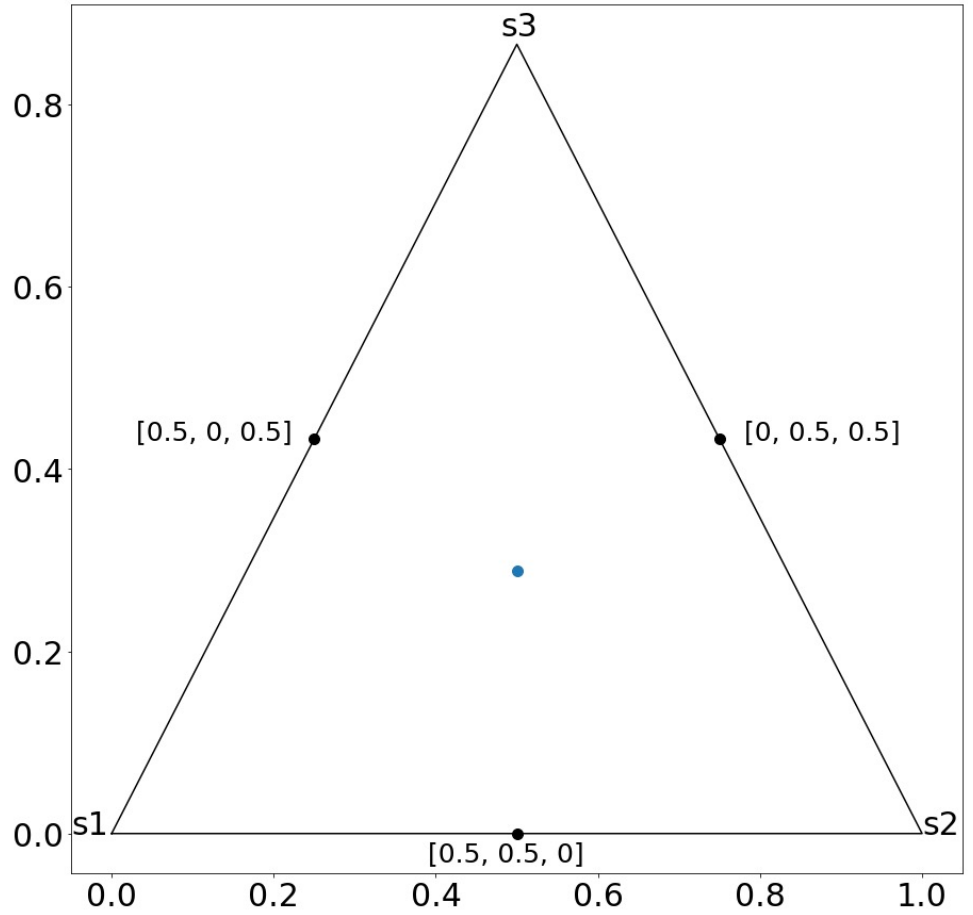
		Player 2		
		1/3 R	1/3 P	1/3 S
Player 1	1/3 R	0, 0	-1, 1	1, -1
	1/3 P	1, -1	0, 0	-1, 1
	1/3 S	-1, 1	1, -1	0, 0

$\vec{\sigma}_1 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $\vec{\sigma}_2 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$

Set of all RPS mixed strategies described by 2-simplex



3D



2D Projection

Expected Utility

- **Expected utility:**

$$u_i(\vec{\sigma}) = \sum_{s \in S} u_i(s) \prod_{j=1}^{n} \vec{\sigma}_j(s_j)$$

$$\begin{aligned} u_1(\vec{\sigma}_2) = &\left(\frac{1}{4}\right)\left(\frac{1}{4}\right) \cdot 0 + \left(\frac{1}{4}\right)\left(\frac{1}{2}\right) \cdot -1 + \left(\frac{1}{4}\right)\left(\frac{1}{4}\right) \cdot 1 \\ &+ \left(\frac{1}{2}\right)\left(\frac{1}{4}\right) \cdot 1 + \left(\frac{1}{2}\right)\left(\frac{1}{2}\right) \cdot 0 + \left(\frac{1}{2}\right)\left(\frac{1}{4}\right) \cdot -1 \\ &+ \left(\frac{1}{4}\right)\left(\frac{1}{4}\right) \cdot -1 + \left(\frac{1}{4}\right)\left(\frac{1}{2}\right) \cdot 1 + \left(\frac{1}{4}\right)\left(\frac{1}{4}\right) \cdot 0 \\ &= 0 \end{aligned}$$

Rock-Paper-Scissors

Player 1 \ Player 2	1/4 R	1/2 P	1/4 S
1/4 R	0, 0	-1, 1	1, -1
1/2 P	1, -1	0, 0	-1, 1
1/4 S	-1, 1	1, -1	0, 0

$$\vec{\sigma}_2 = \left(\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)\right)$$

Deviation Payoffs

- Define $\vec{\sigma}_{-i} = (\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_n)$
- **Deviation payoff:**

$$\text{devPay}_i(s, \vec{\sigma}) = u_i(s, \vec{\sigma}_{-i})$$

- **Deviation payoff function,** $\text{devPay}(\vec{\sigma})$:

$$(\text{devPay}_1(s_1, \vec{\sigma}), \dots, \text{devPay}_n(s_{|s_n|}, \vec{\sigma}))$$

$$\text{devPay}_1(\mathbf{R}, \vec{\sigma}_2) = \left(\frac{1}{4}\right) \cdot 0 + \left(\frac{1}{2}\right) \cdot -1 + \left(\frac{1}{4}\right) \cdot 1 = \mathbf{-0.25}$$

$$\text{devPay}_1(\mathbf{P}, \vec{\sigma}_2) = \mathbf{0}$$

$$\text{devPay}_1(\mathbf{S}, \vec{\sigma}_2) = \mathbf{0.25}$$

Rock-Paper-Scissors

Player 1 \ Player 2	1/4 R	1/2 P	1/4 S
1/4 R	0, 0	-1, 1	1, -1
1/2 P	1, -1	0, 0	-1, 1
1/4 S	-1, 1	1, -1	0, 0

$$\vec{\sigma}_2 = \left(\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)\right)$$

Nash Equilibrium

- **Nash equilibrium:** a set of strategies such that no player can gain by deviating
- **Nash's Theorem:** *Every finite game with two or more players contains at least one Nash equilibrium.*
- In RPS, $\vec{\sigma}_1 = (\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right))$ is a Nash equilibrium

Rock-Paper-Scissors

		Player 2		
		1/3 R	1/3 P	1/3 S
Player 1	1/3 R	0, 0	-1, 1	1, -1
	1/3 P	1, -1	0, 0	-1, 1
	1/3 S	-1, 1	1, -1	0, 0

$$\vec{\sigma}_1 = (\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right))$$

Regret

- **Player regret:**

$$\varepsilon_i(\vec{\sigma}) = \underbrace{\max_{s \in S} u_i(s, \vec{\sigma}_{-i})}_{\text{max deviation payoff}} - \underbrace{u_i(\vec{\sigma})}_{\text{exp util}}$$

- **Regret:**

$$\varepsilon(\vec{\sigma}) = \max_{i \in P} \varepsilon_i(\vec{\sigma})$$

- $\varepsilon(\vec{\sigma}_1) = 0, \varepsilon(\vec{\sigma}_2) = 0.25$
- **ε-Nash equilibrium:** $\vec{\sigma}$ such that $\varepsilon(\vec{\sigma}) \leq \varepsilon$ for some ε

Rock-Paper-Scissors

Player 1 \ Player 2	R	P	S
R	0, 0	-1, 1	1, -1
P	1, -1	0, 0	-1, 1
S	-1, 1	1, -1	0, 0

$$\vec{\sigma}_1 = \left(\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)\right)$$

$$\vec{\sigma}_2 = \left(\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right), \left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)\right)$$

Symmetric Games and Equilibria

- A **symmetric game** is a tuple $\tilde{\tau}_n = (n, S, u)$ where
  - n is the number of players
  - $S = \{s_1, \dots, s_k\}$
  - $C = \{\vec{c} \in \mathbb{Z}^k : \vec{c}_j \geq 0, \sum_{j=1}^{k} \vec{c}_j = n\}$
  - $u : C \mapsto \mathbb{R}^k$
- **Symmetric Nash equilibrium:** Nash equilibrium in which all players are playing the same (pure- or mixed-) strategy

Rock-Paper-Scissors

Player 2 \ Player 1	R	P	S
R	0, 0	-1, 1	1, -1
P	1, -1	0, 0	-1, 1
S	-1, 1	1, -1	0, 0

Motivation: Computing Approximate NE

- Computational complexity of computing exact NE
- Representational challenges
- Approximate Nash-finding algorithms
  - Lemke-Howson
  - Govindan-Wilson (global Newton method)
  - Fictitious play
  - Replicator dynamics

Replicator Dynamics

REPLICATORDYNAMICS(Γ_n, $\vec{\sigma}$, ϵ, numIters):

offset $\leftarrow$ [MIN(Γ_n, i) **for** each player i]

repeat

$\vec{\sigma} \leftarrow \vec{\sigma} \cdot$ (DEVPAYS(Γ_n, $\vec{\sigma}$) - offset)

$\vec{\sigma} \leftarrow \vec{\sigma} \,/\, \sum(\vec{\sigma})$

until numIters

if REGRET(Γ, $\vec{\sigma}$) $\leq \epsilon$

return $\vec{\sigma}$

return -1

Learned Deviation Payoff Function

Outline

- Introduction and Motivation
- Background
- **Related Work**
- Variable-Player Games
- Model: Approximating robust symmetric Nash equilibria
- Analysis: Equilibrium robustness metrics
- Experiments
- Conclusion & Ongoing Work

Related Work

- Learning game models from data ([1], [7], [10], [14], [18])
  - Learning Deviation Payoffs in Simulation-Based Games [11]*
- Games with a variable number of players
  - Simulation-based game theory ([2], [9], [15], [16], [17], [19])
  - Other examples ([3], [4], [6], [8], [12], [13])

Outline

- Introduction and Motivation
- Background
- Related Work
- **Variable-Player Games**
- Model: Approximating robust symmetric Nash equilibria
- Analysis: Equilibrium robustness metrics
- Experiments
- Conclusion & Ongoing Work

Variable-Player Symmetric Games

- A **variable-player symmetric game** is a tuple $\tilde{\tau}_{mn} = (P, S, u)$ where
  - $P = \{m, \dots, n\}$
  - $S = \{s_1, \dots, s_k\}$
  - $C_{mn} = \{\vec{c} \in \mathbb{Z}^k : \vec{c}_j \geq 0, m \leq \sum_{j=1}^{k} \vec{c}_j \leq n\}$
  - $u : C_{mn} \mapsto \mathbb{R}^k$

Variable-Player Symmetric Games (*cont.*)

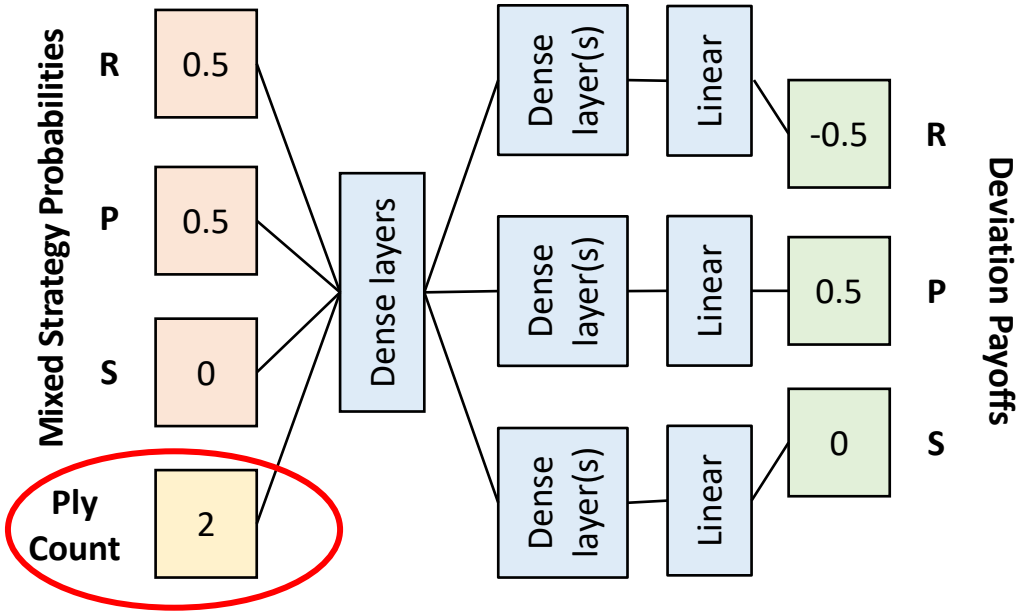
- An **instance** of a variable-player symmetric game is a tuple $\tilde{\tau}_p = (p, S, u)$ where
  - p is the number of players, with $m \leq p \leq n$
  - $S = \{s_1, \ldots, s_k\}$
  - $C_p = \{\vec{c} \in \mathbb{Z}^k : \vec{c}_j \geq 0,\ \sum_{j=1}^{k} \vec{c}_j = p\}$
  - $u : C_p \mapsto \mathbb{R}^k$

Outline

- Introduction and Motivation
- Game Theory Background
- Related Work
- Variable-Player Games
- **Model**
  - **Approximating deviation payoffs**
  - **Approximating symmetric Nash equilibria**
- Analysis: Equilibrium robustness metrics
- Experiments
- Conclusion & Ongoing Work

Model: Approximating deviation payoffs

- **Hypothesis:** the payoffs in a game with x players are similar or related to the payoffs in the same game with $x \pm 1$ players, given a large value of x
- We use a **multi-headed neural network** to learn a mapping from mixed-strategy profiles and number of players to deviation payoffs

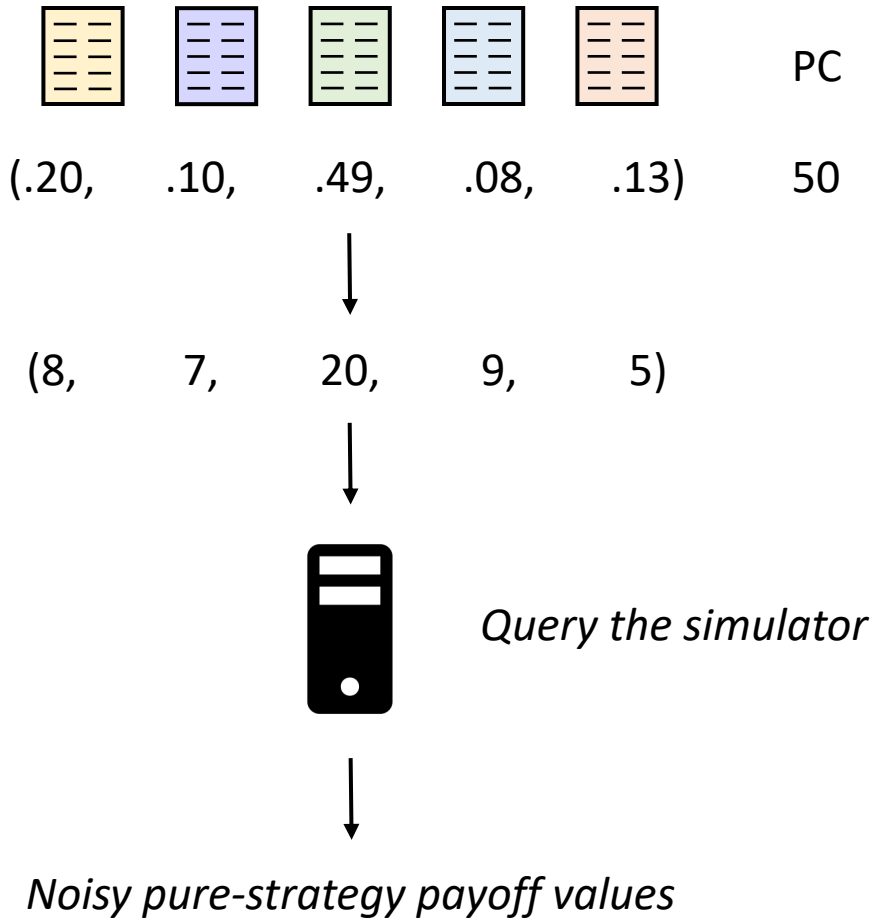


Neural Network Architecture for Variable-Player Learning

Model: Approximating deviation payoffs (*cont.*)

Generating training data:

- Generate random mixed-strategy profile
  - Dirichlet distribution
- Generate random player count
  - Representative instances
  - Uniform random across entire range
- Sample a pure-strategy profile according to mixed-strategy profile for each opponent
- Query simulator for PS payoffs



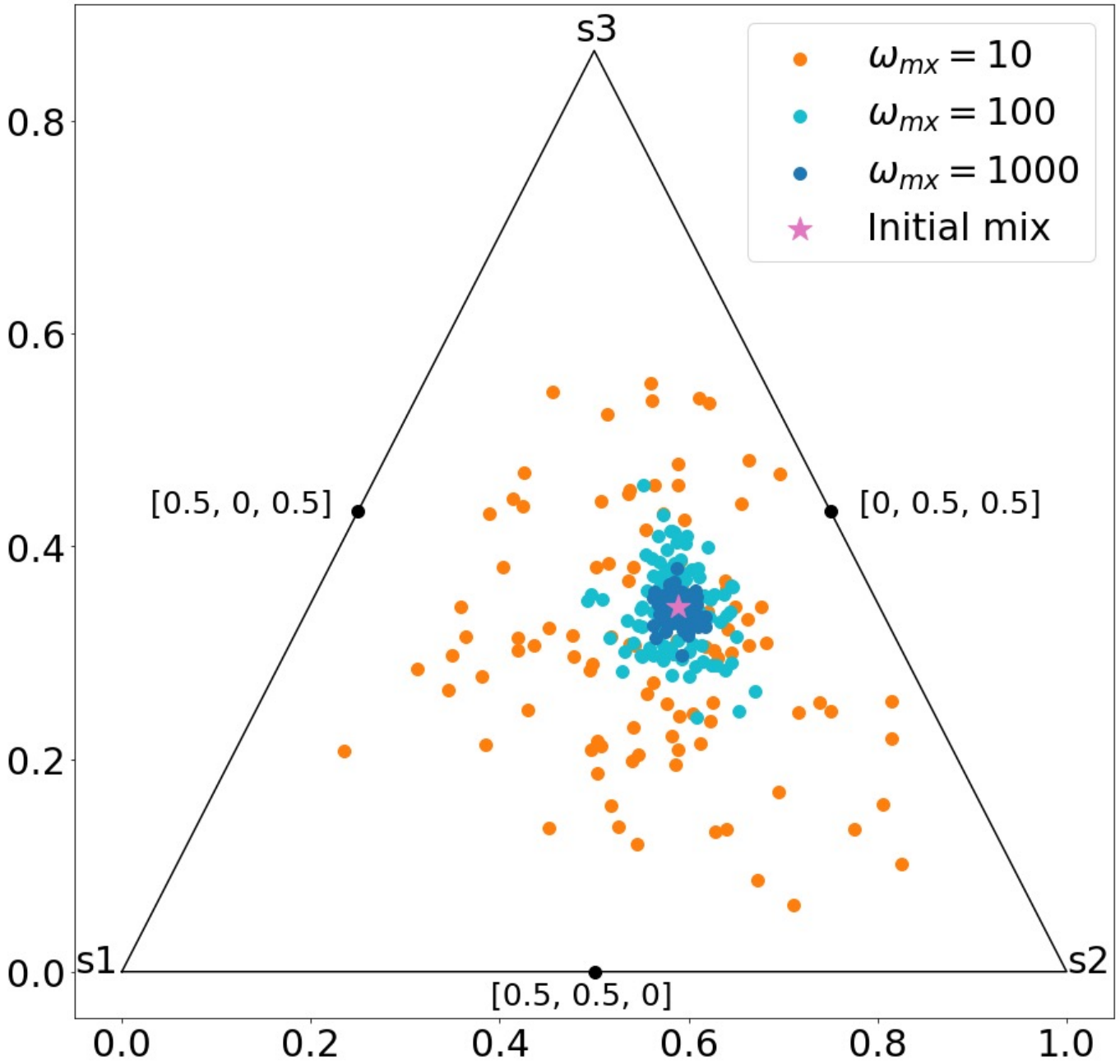
Model: Approximating symmetric NE

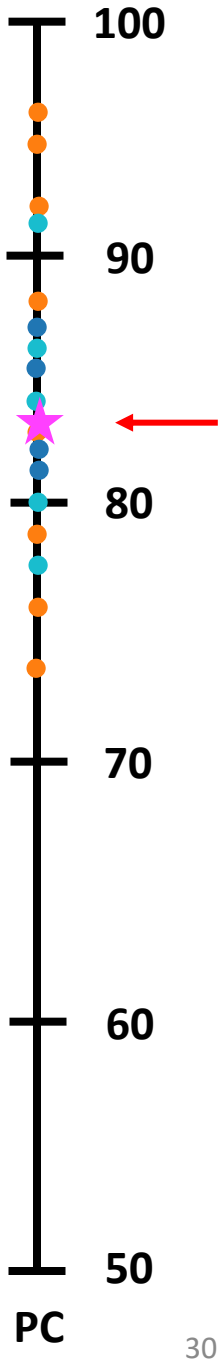
- Want to focus learning on areas of simplex where we think there might be approximate Nash equilibria

Algorithm overview:

- For *i* iterations
  - Run Nash-finding algorithm
  - Sample in neighborhood of candidate Nash and corresponding player counts
  - Retrain network, adding resamples to training data

Neighborhood Sampling





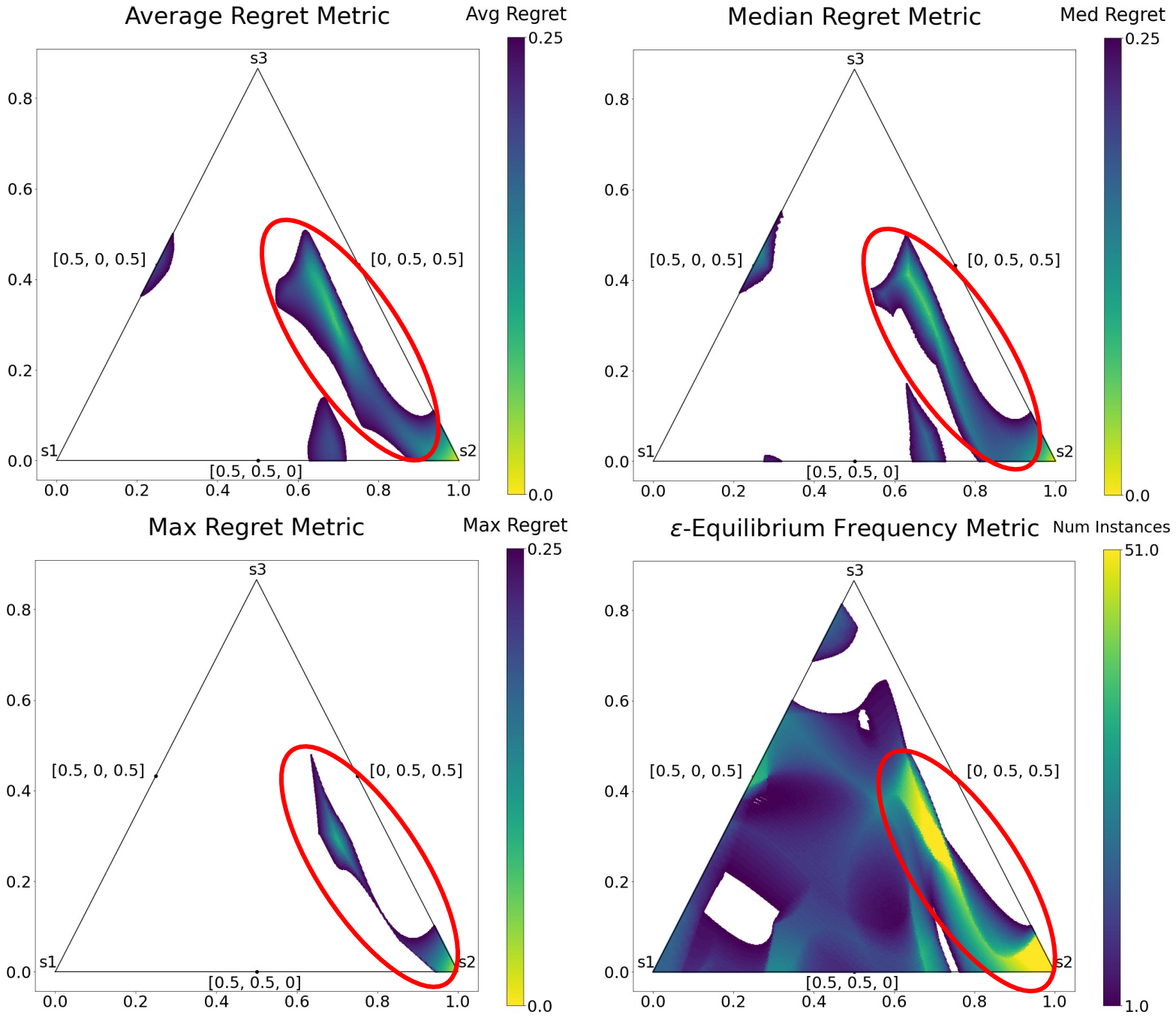
Outline

- Introduction and Motivation
- Game Theory Background
- Related Work
- Variable-Player Games
- Model: Approximating robust symmetric Nash equilibria
- **Analysis**
  - **Equilibrium robustness metrics**
  - **Comparison of robustness metrics**
- Experiments
- Conclusion & Ongoing Work

Analysis: Equilibrium robustness metrics

- Typical game-theoretic analysis: find approximate NE in games with fixed number of players
- Finding approximate NE in game with variable number of players is not as straightforward
- Robustness: measure of how well an equilibrium generalizes across all instances of game
- Several proposed robustness metrics
  - Average regret
  - Median regret
  - Max regret
  - Approximate equilibrium frequency

Analysis: Comparison of robustness metrics



Outline

- Introduction and Motivation
- Game Theory Background
- Related Work
- Variable-Player Games
- Model: Approximating robust symmetric Nash equilibria
- Analysis: Equilibrium robustness metrics
- **Experiments**
  - **Random game generation**
  - **Comparison to existing work**
  - **Approximating symmetric Nash equilibria**
- Conclusion & Ongoing Work

Experiments

- Does our technique outperform Sokota et al.'s technique on variable-player games?

- Is the iterative refinement necessary for the variable-player learning model?

Experiments: Random game generation

- Random additive polynomial sine BAGG-FNAs serve as a proxy for simulator data
- Payoff functions are complex but learnable
- Compact representation -> simplifies ground truth deviation payoff calculation
- Can be easily defined with a variable number of players

Experiment 1: Experimental specification

Random Games

- 250 random symmetric games
- Range: 50 to 100 players
- 5 strategies

Evaluation

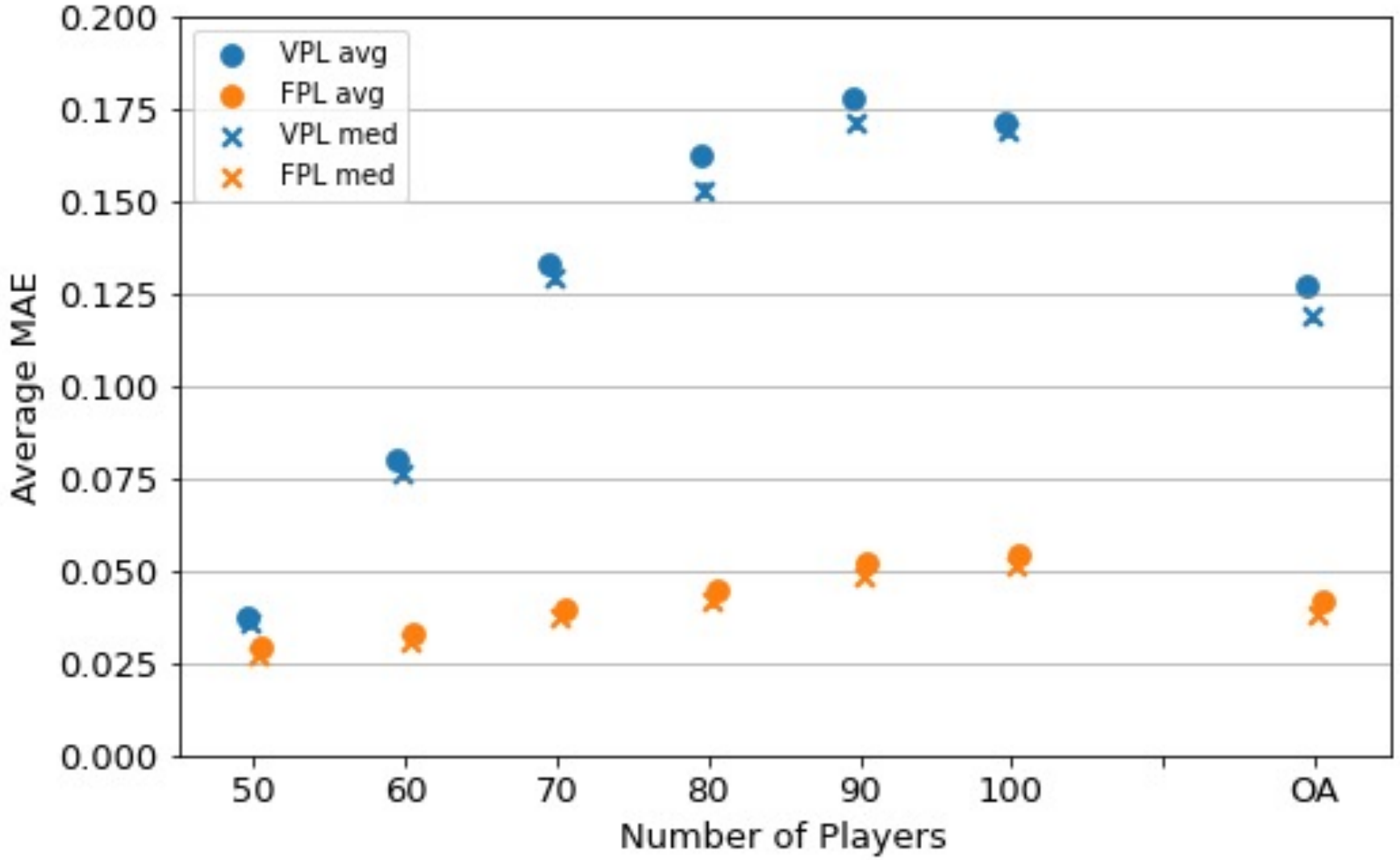
- Average deviation payoff MAE across all strategies

Learning Models

- Fixed-Player Learning (FPL)
  - Train 6 NNs with player counts: 50, 60, 70, 80, 90, 100
- Variable-Player Learning (VPL)
  - Train 1 NN which learns across range of player counts
- Hyperparameters optimized separately

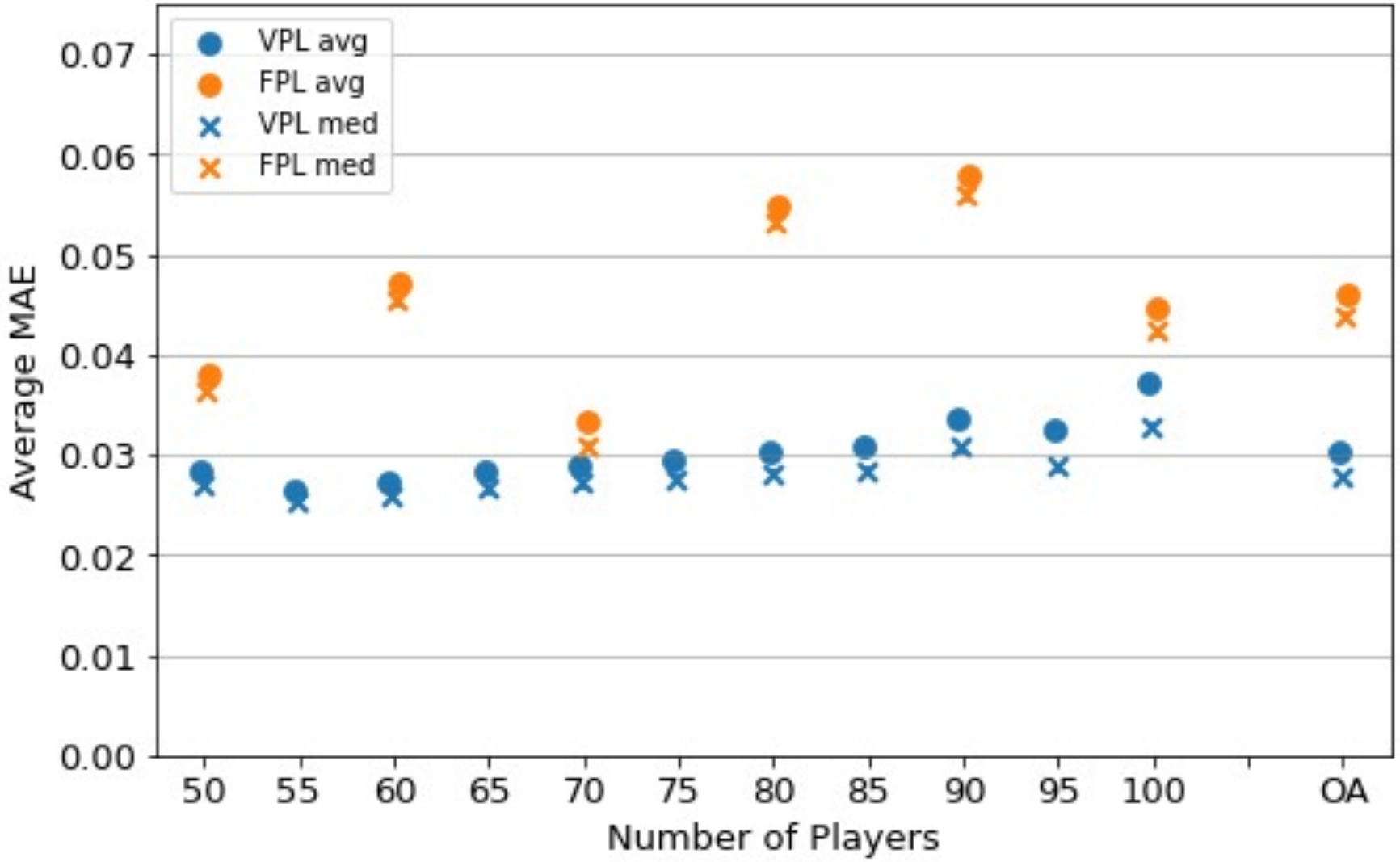
Experiment 1: Comparison to Existing work

PC: Representative instances



Experiment 1: Comparison to Existing work

PC: Uniform across entire range



Experiment 2: Experimental specification

Random Games

- 500 random symmetric games
- Range: 50 to 100 players
- 5 strategies

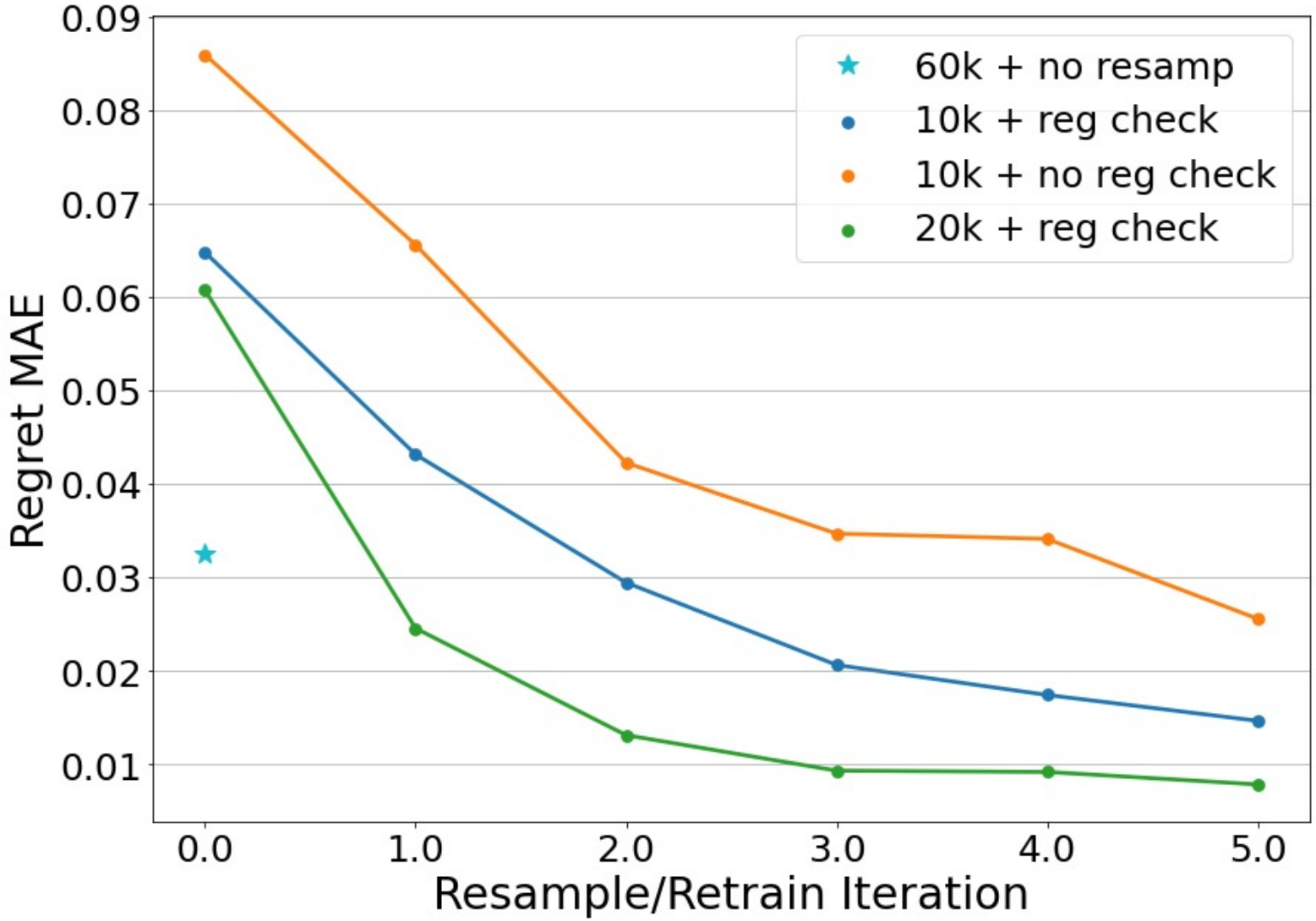
Evaluation

- Regret MAE

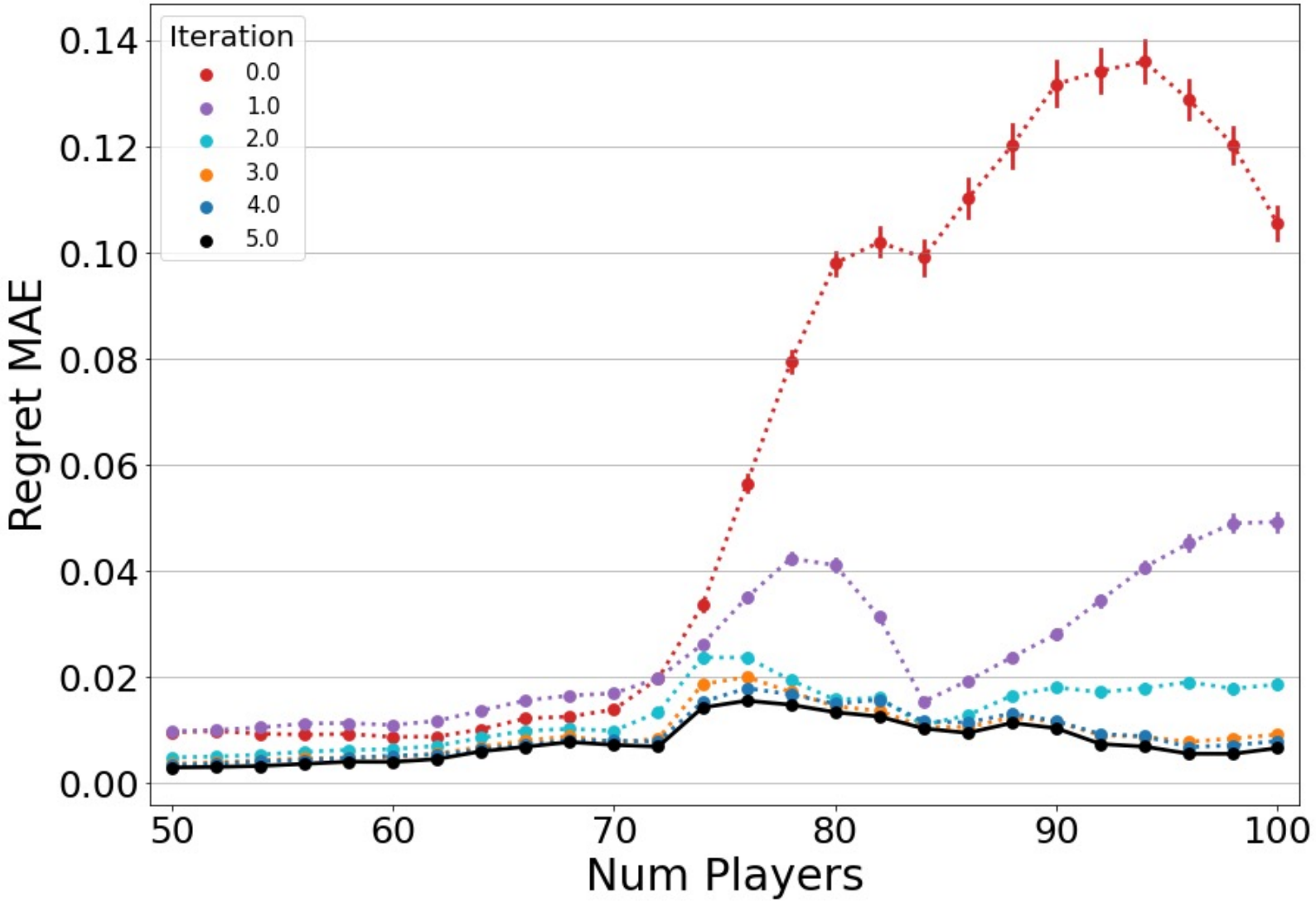
Learning Models

- Model 1
  - No resampling
- Model 2 (baseline)
  - Resampling and intermediate regret check
- Model 3
  - Resampling and no intermediate regret check
- Model 4
  - Resampling and intermediate regret check

Experiment 2: Approximating Symmetric NE



Experiment 2: Approximating Symmetric NE



Outline

- Introduction and Motivation
- Background
- Related Work
- Variable-Player Games
- Model: Approximating robust symmetric Nash equilibria
- Analysis: Equilibrium robustness metrics
- Experiments
- **Conclusion & Ongoing Work**

Conclusion

- Motivation: many real-world interactions of interest have a large, uncertain number of players
- Current game-theoretic analysis: limited to fixed-player games
- Related work: number of players is hyperparameter and/or small range analyzed
- **Main contribution: new type of analysis which accommodates uncertainty in the number of players**

Ongoing Work

- Scalability experiments
  - Size of range
  - Magnitude of player counts
  - Number of strategies
- Evaluate performance on a wider range of games
- Extend this technique to analyze variable-player role-symmetric games
- Extend this technique to vary parameters of simulation environment
- Theoretical guarantees?

References

[1] E. Areyan Viqueira, C. Cousins, and A. Greenwald, “Improved algorithms for learning equilibria in simulation-based games,” in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ‘20. p. 79-87.

[2] E. Brinkman and M. P. Wellman, “Empirical mechanism design for optimizing clearing interval in frequent call markets,” in *Proceedings of the 2017 ACM Conference on Economics and Computation*. EC ‘17. p. 205-221.

[3] S. Fatima, “Sequential versus simultaneous auctions: A case study,” in *Proceedings of the 8th International Conference on Electronic Commerce: The New e-Commerce: Innovations for Conquering Current Barriers, Obstacles, and Limitations to Conducting Successful Business on the Internet.* ICEC ‘06. p. 82-91.

[4] N. Hanaki and J. Rouchier, “If you are so rich, why aren’t you smart?” in *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*. WSC ‘13. IEEE Press, 2013, p. 1731-1741.

[5] A. X. Jiang, K. Leyton-Brown, and N. Bhat, “Action-graph games,” *Games and Economic Behavior,* vol. 71, pp. 141-173, 01 2011.

[6] J. Honorio and L. Ortiz, “Learning the structure and parameters of large-population graphical games from behavioral data,” vol. 16, no. 1, p. 1157-1210, Jan. 2015.

[7] Z. Li and M.P. Wellman, “Structure learning for approximate solution of many-player games.” in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020.* AAAI Press, 2020. p. 2119-2127.

[8] P. E. Petruzzi, J. Pitt, and D. Busquets, “Electronic social capital for self-organizing multi-agent systems,” vol. 12, no. 3, Sept. 2017.

[9] M. Shearer, G. Rauterberg, and M. P. Wellman, “An agent-based model of financial benchmark manipulation,” *ICML-19 Workshop on AI in Finance*.

[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484-489, 2016.

References

[11] S. Sokota, C. Ho, and B. Wiedenbeck, “Learning deviation payoffs in simulation-based games,” *AAAI*, vol. 33, no. 1, pp. 2173-2180, 2019.

[12] D. R. Thompson, O. Lev, K. Leyton-Brown, and J. Rosenschein, “Empirical analysis of plurality election equilibria,” in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems*. AAAMAS ‘13. 2013, p. 391-398.

[13] B. Tuffin and P. Maillé, “How many parallel TCP sessions to open: A pricing perspective,” in *Performability Has its Price*, B. Stiller, P. Reichl, and B. Tuffin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 2-12.

[14] Y. Vorobeychik, M. Wellman, and S. Singh, “Learning payoff functions in infinite games,” *Machine Learning*, vol. 67, pp. 145-168, 05 2007.

[15] E. Wah, D. Hurd, and M. P. Wellman, “Strategic market choice: Frequent call markets vs. continuous double auctions for fast and slow traders,” *EAI Endorsed Trans. Serious Games,* vol. 3, no. 10, p. e1, 2016.”

[16] E. Wah, M. Wright, and M. P. Wellman, “Welfare effects of market making in continuous double auctions,” *J. Artif. Intell. Res.,* vol. 59, pp. 613-650, 2017.

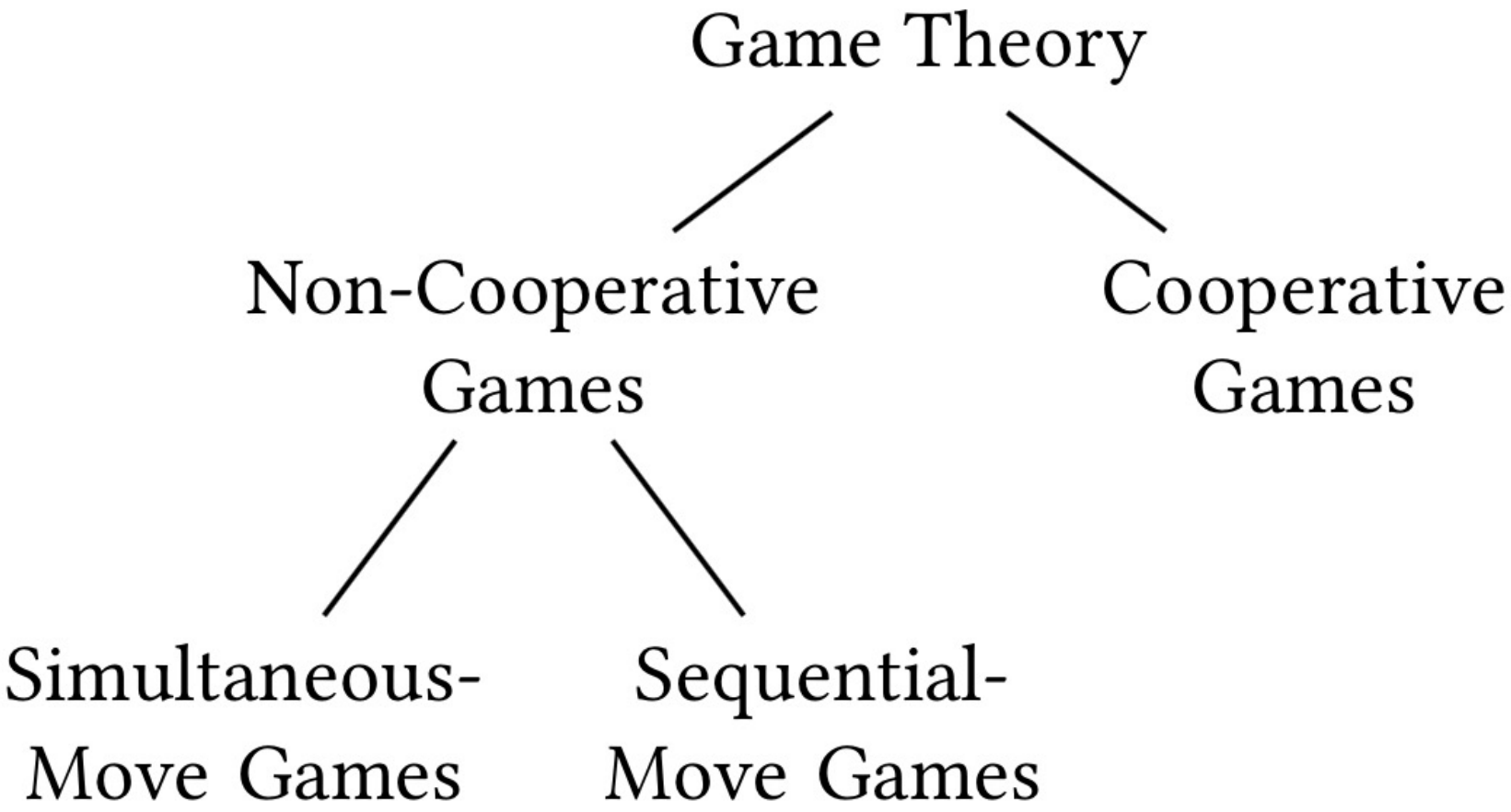
[17] M. P. Wellman, T. H. Kim, and Q. Duong, “Analyzing incentives for protocol compliance in complex domains: A case study of introduction-based routing,” *CoRR*, 2013.

[18] B. Wiedenbeck, F. Yang, and M. P. Wellman, “A regression approach for modeling games with many symmetric players,” in *Proceedings of the Thirty-Seccond AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 1266-1273.

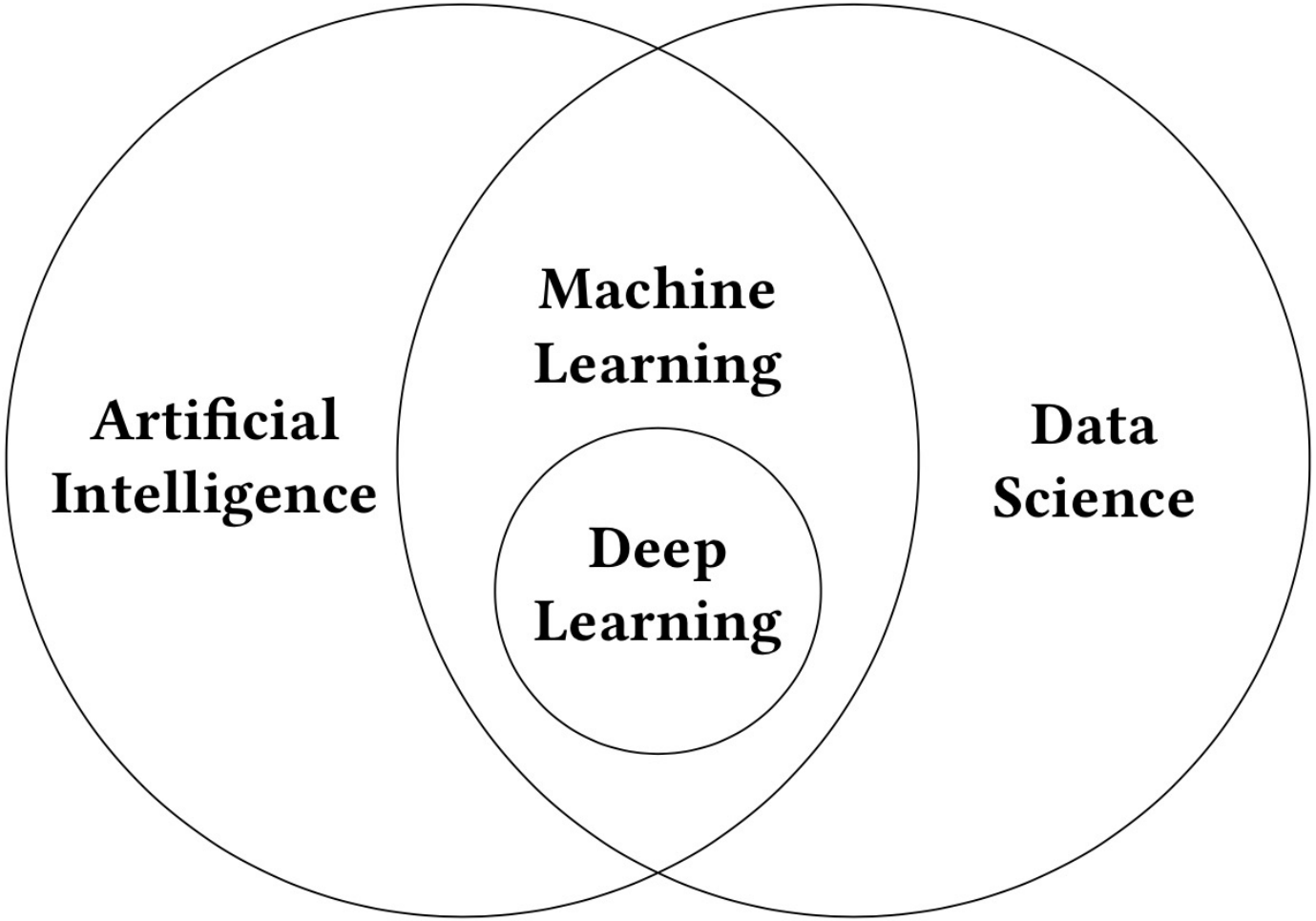
[19] M. Wright and M. P. Wellman, “Evaluating the stability of non-adaptive trading in continuous double auctions,” in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. AAMAS ‘18. 2018, p. 614-622.

Thank you!

Hierarchy of Game Classes



Machine Learning Venn Diagram



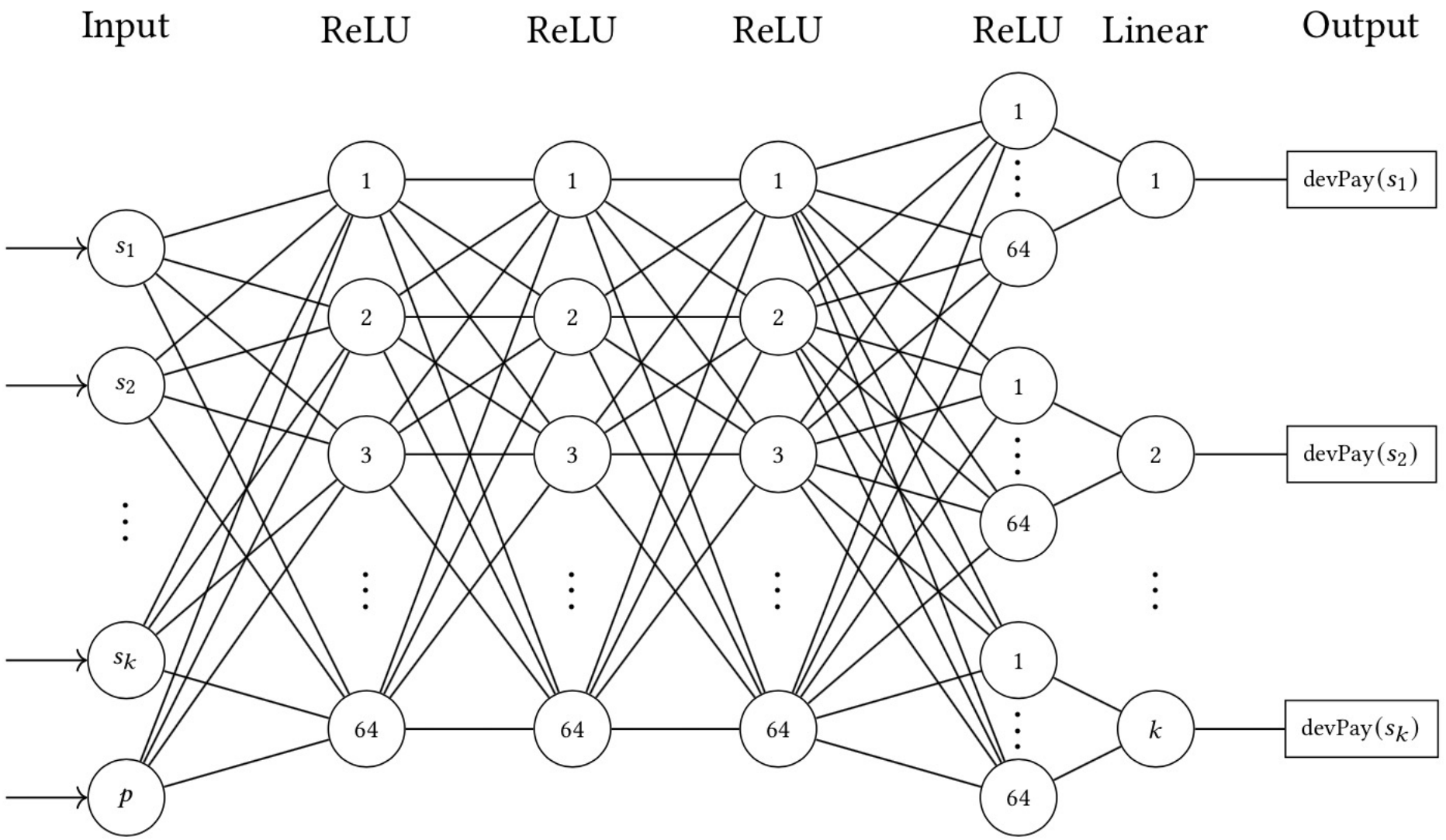
Common Activation Functions

Function Name	Function Definition
Linear	$f(x) = x$
Sigmoid (σ)	$f(x) = \frac{1}{1+e^{-x}}$
Rectified Linear Unit (ReLU)	$f(x) = \max(0, x)$
Hyperbolic Tangent (tanh)	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$

Common Hyperparameters

- Number of hidden layers
- Number of nodes per layer
- Type(s) of layers
- Choice of activation function
- Optimizer
- Learning rate
- Batch size
- Number of epochs
- Weight initialization
- Learning rate decay
- Batch normalization
- Regularization

Variable-Player Learning (VPL) Architecture

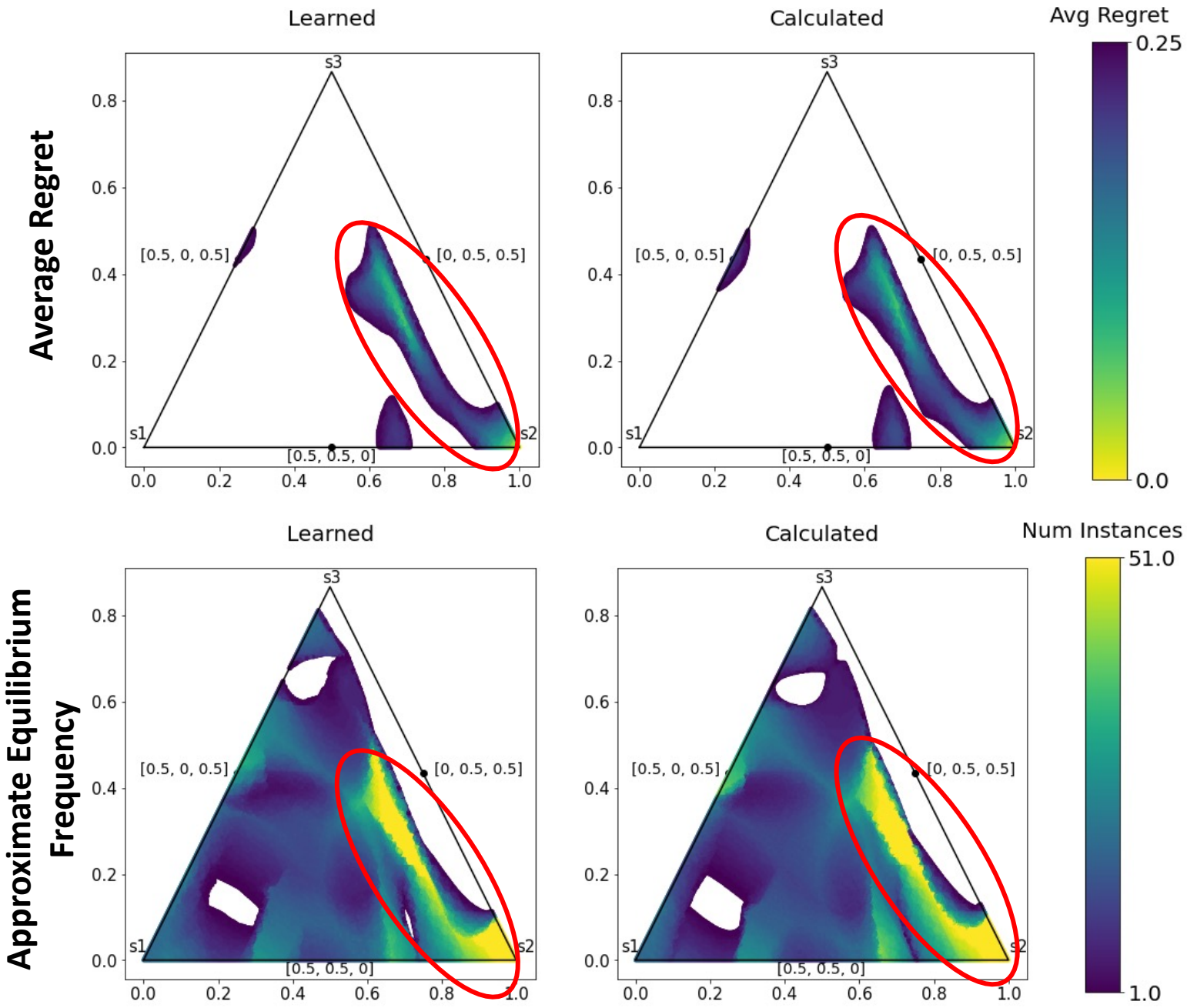


Model: Approximating robust symmetric NE

AppxRobustNashEquilibria(numInitQueries, numResampQueries, numIters, $\vec{\alpha}$, $\vec{\omega}$, m, n):

$[\vec{\sigma}] \leftarrow \text{Dir}(\vec{\alpha}, \text{numInitQueries})$
$[p] \leftarrow \text{Uniform}([\vec{\sigma}], m, n)$
$[\vec{s}] \leftarrow \text{sampleOppProfiles}([\vec{\sigma}], [p])$
$[\vec{u}] \leftarrow \text{samplePSPayoffs}([\vec{s}])$
$\text{data} \leftarrow ([\vec{\sigma}], [p], [\vec{u}])$
regressor.fit(data)
repeat
    $([\vec{\sigma}^*], [p^*]) \leftarrow \text{findNash}(\text{regressor})$
    $([\vec{\sigma}], [p]) \leftarrow \text{sampleNbhd}([\vec{\sigma}^*], [p^*], \vec{\omega}, \text{numResampQueries})$
    $[\vec{s}] \leftarrow \text{sampleOppProfiles}([\vec{\sigma}], [p])$
    $[\vec{u}] \leftarrow \text{samplePSPayoffs}([\vec{s}])$
    $\text{data} \leftarrow \text{data} + ([\vec{\sigma}], [\vec{s}], [\vec{u}])$
    regressor.fit(data)
until numIters
$([\vec{\sigma}^*], [p^*]) \leftarrow \text{findNash}(\text{regressor})$
return $\text{findRobustNash}([\vec{\sigma}^*], [p^*])$

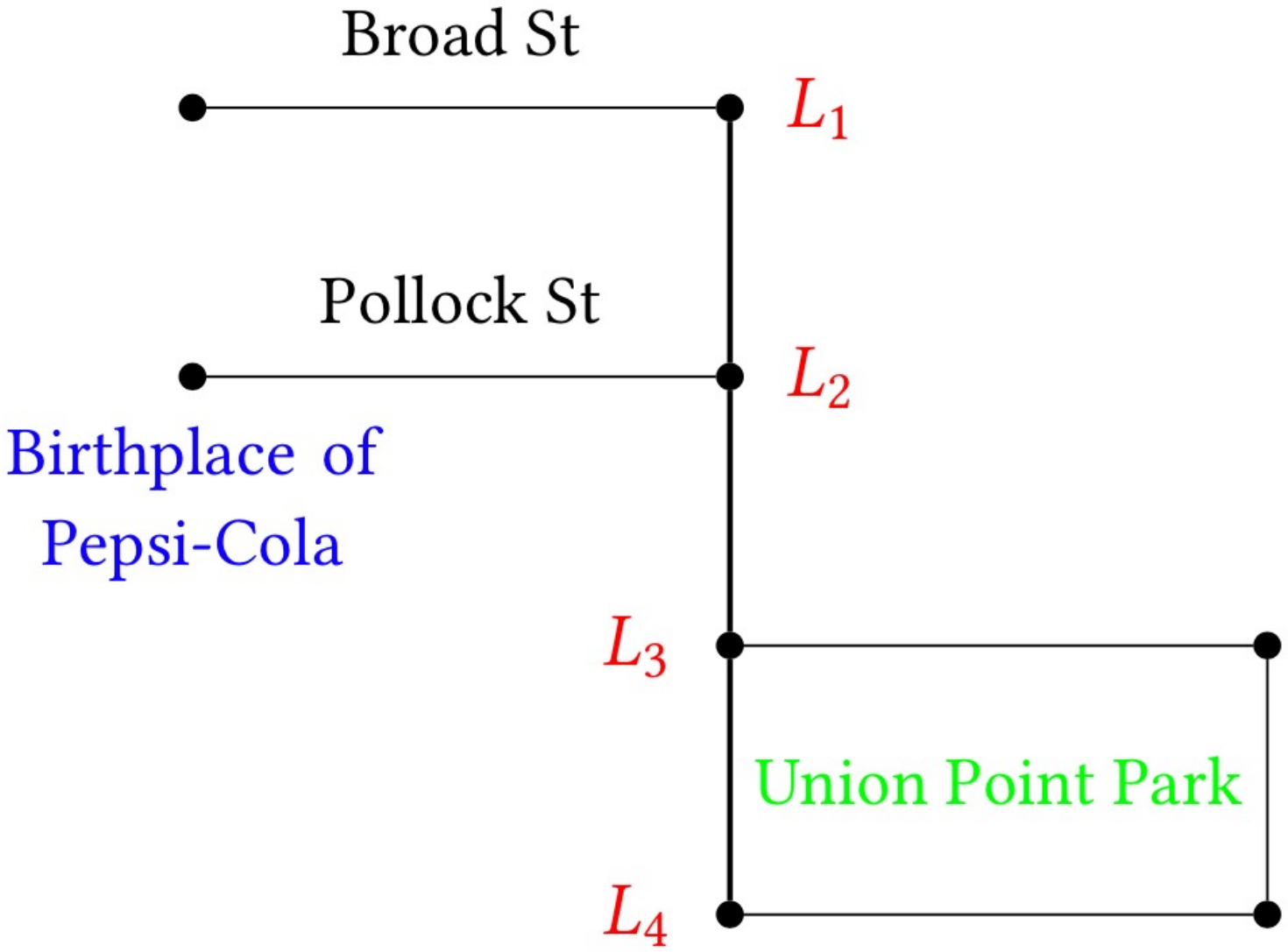
Analysis: Comparison of robustness metrics



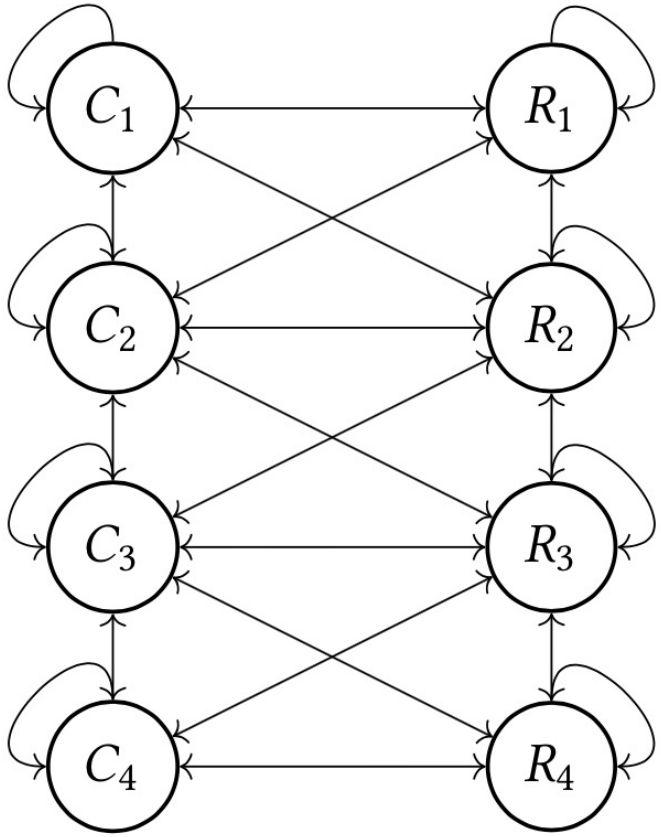
Experiments: Random game generation

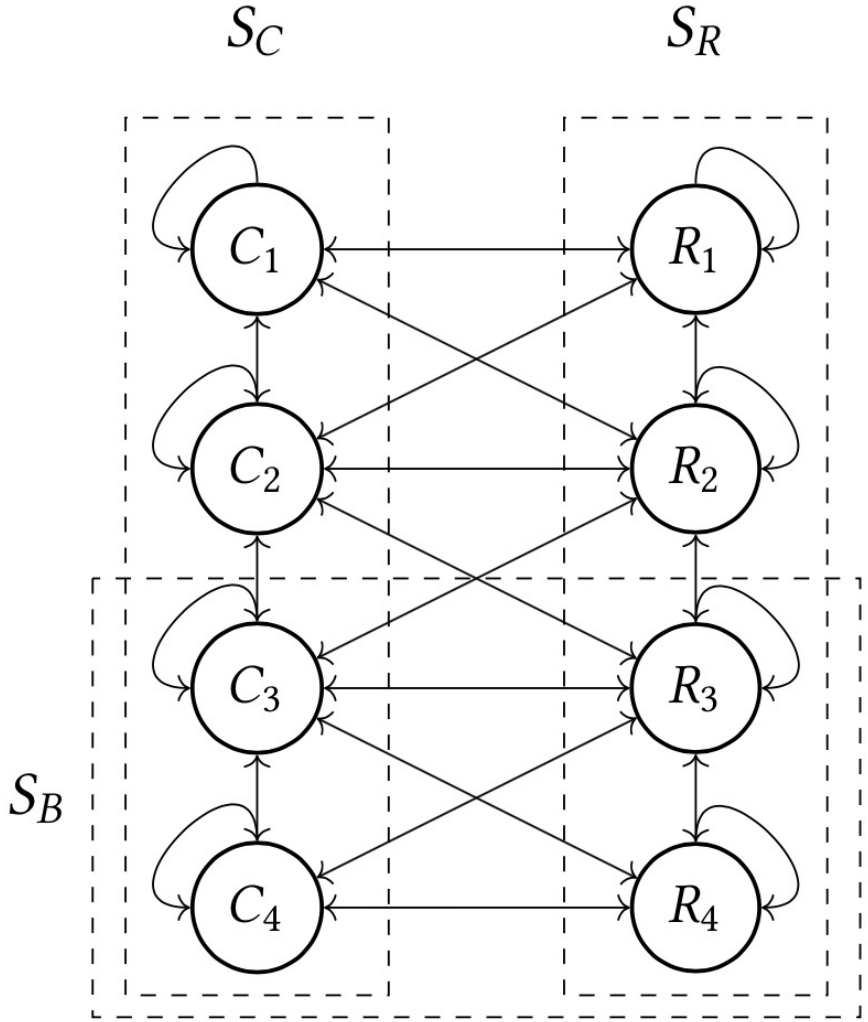
- Action-graph games ([5])
  - Actions are represented as vertices
  - Edges encode dependence among actions
  - Players have access to a subset of the actions
- AGGs can be directed or undirected
- To compute payoffs—only necessary to look at configuration of neighborhood
- Most useful when game exhibits player symmetries

AGG Example


Broad St
L_1
Pollock St
L_2
Birthplace of
Pepsi-Cola
L_3
Union Point Park
L_4

AGG Example (cont.)



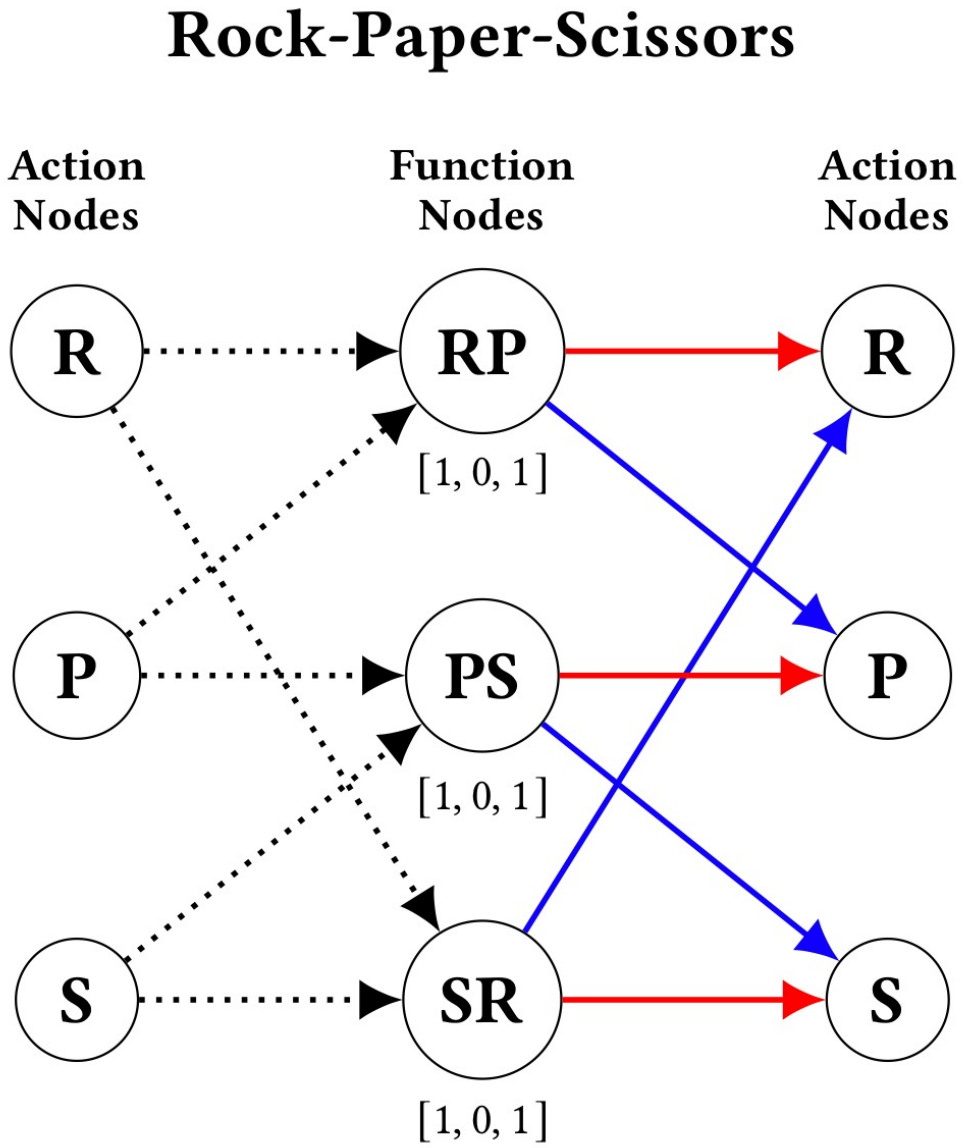


Experiments: Random game generation

- BAGG-FNAs
  - Nodes of graph can be partitioned into two independent sets (action nodes and function nodes)
  - Function nodes
    - Input: total number of players playing actions in neighborhood of given function node
    - Output: payoff contributed by given function node, according to function table
  - Payoff for playing action a: weighted sum of function outputs from neighboring function nodes

RPS as BAGG-FNA

	R	P	S
R	0, 0	-1, 1	1, -1
P	1, -1	0, 0	-1, 1
S	-1, 1	1, -1	0, 0



Experiments: Random game generation ([11])

- Random additive polynomial sine BAGG-FNAs
  - Subgraph containing edges from action nodes to function nodes: Erdös-Rényi random graph
  - Subgraph containing edges from function nodes to action nodes: complete bipartite graph
  - Action weights generated according to normal distribution
  - Function nodes: sum of long-period sine function and low-degree polynomial
- Add noise to BAGG-FNA training data payoffs

Additional Hyperparameters

- Number of initial queries
- Number of initial training epochs
- Number of replicator dynamics mixtures
- Max regret (if intermediate regret check)
- ω_{mx}, the mixture resample factor
- Number of resample queries
- Number of retrain epochs
- Number of replicator dynamics iterations
- Number of resample/retrain iterations
- ω_{p}, the player count resample factor