Missouri University of Science & Technology — Department of Computer Science
Fall 2024 — **CS 5408: Game Theory for Computing**

Project 2

Instructor: *Sid Nadendla* — **Due:** *December 10, 2024*

Project Description

The goal of this project is to learn how modern game-theoretic techniques and tools can be designed/implemented to solve a large game, or any real-world problem. The project can be either pursued by a team of at most two students. Each team gets to pick one of the two following options for Project 2.

Option 1: Research Project

This option is primarily designed for those who are interested in pursuing research in game theory. Interested students can pick a research problem inspired from any real-world application, and demonstrate how this real-world problem can be solved using game-theoretic techniques. However, students should first discuss their plan if they wish to pick this option. Examples of such real-world problems can be found in the references below.

- **Example 1:** This is a journal article on the use of online learning algorithms in bidding auctions in construction management, which was pursued as a course project by students in my game theory class about four years ago, and published their outcomes in the following journal article.

  Ref. R. Asssaad, M. O. Ahmed, I. H. El-Adaway, A. Elsayegh, and V. S. S. Nadendla, "Comparing the impact of learning in bidding decision-making processes using algorithmic game theory," *Journal of Management in Engineering*, vol. 37, no. 1, pp. 04020099, 2021.

- **Example 2:** A very interesting research problem in the domain of security is the field of strategic deception, where the defender employs deceptive strategies to confuse the attacker.

  Ref. J. Pawlick, E. Colbert, and Q. Zhu, "A Game-theoretic Taxonomy and Survey of Defensive Deception for Cybersecurity and Privacy," *ACM Computing Surveys*, vol. 52, no. 4, Article 82, July 2020.

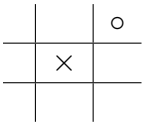
Option 2: Solving Latent Tic-Tac-Toe using MC-CFR

This option is primarily designed to implement *Monte-Carlo Counterfactual Regret Minimization* (MC-CFR) algorithm on a game called *Latent Tic-Tac-Toe* (LTTT) in Python.

Latent Tic-Tac-Toe

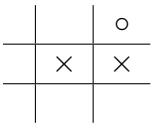
LTTT is a modified version of the classical Tic-Tac-Toe, where the game's state is not revealed until after the opponent's next move, and lost if deemed invalid at the time of revelation.

For example, consider a 3×3 grid, in the state



Assume that the cross-player places the next cross at location $(2, 3)$. However, this is only revealed to the nought-player after the nought-player chooses its decision. Of course, if the move is an illegal move, i.e. if the nought-player also decides to place a nought at $(2, 3)$, the move is voided and the game state is revealed to the nought-player.

At this time, both the players know the game state to be



However, the cross-player may choose their next choice thinking that the nought-player could have played a valid move.

Monte-Carlo Counterfactual Regret Minimization

Large games demand online learning of the game tree, which makes it practically impossible to use traditional equilibrium-finding algorithms such as SPNE. In such a case, online learning approaches based on counterfactual regret minimization (CFR) provides a way forward to solve large games.

Ref. M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret Minimization in Games with Incomplete Information," Advances in Neural Information Processing Systems, vol. 20, 2007. URL: http://www.johanson.ca/publications/poker/2007-nips-cfr/2007-nips-cfr.pdf

Unforunately, vanilla CFR developed by Zinkevich *et al.* relies on *chance sampling*, which requires a full game tree traversal. On the contrary, two Monte-Carlo sampling approaches, namely *outcome sampling* and *external sampling* have been proposed as an alternative to chance sampling to avoid full-tree traversal. This project focuses on the former approach, namely MC-CFR_OutcomeSampling, whose details can be found in the following reference.

Ref. M. Lanctot, K. Waugh, M. Zinkevich, and M. Bowling. "Monte Carlo sampling for regret minimization in extensive games," *Advances in Neural Information Processing Systems*, vol. 22, 2009. URL: https://mlanctot.info/files/papers/nips09mccfr_techreport.pdf

A modified version of this algorithm was later developed using deep neural network models to solve games with super-human performance, which also got the Nobel Prize in 2024 in

Biology for its impact in computational protein folding.

Ref. D. Silver *et al.*, "Mastering the Game of Go without Human Knowledge," *Nature*, vol. 550, no. 7676, pp. 354-359, 2017. URL: https://augmentingcognition.com/assets/Silver2017a.pdf

Ref. J. Jumper *et al.*, "Highly Accurate Protein Structure Prediction with AlphaFold." *Nature*, vol. 596, no. 7873, pp. 583-589, 2021. URL: https://computingbiology.github.io/docs/alphafold2-jumper2021.pdf

Instructions

You are expected to strictly follow the timeline provided in Table 1. No extensions will be provided for title and final-report submission deadlines.

Milestones	**Deadlines**
Project 2 statement released	Nov 19, 2024
Deliberation week	Nov 19 - 22, 2024
Submit your project title and plan	Nov 22, 2024
Progress Update and Feedback	Dec 3, 2024
Final Report Submission	Dec 10, 2024

Table 1: Milestones and Timeline for Project 2

Expected Deliverables

In your project implementation,

- **Game Model:** Implement a BOARD class where you maintain the true state of the board, as well as the two board states that are revealed to cross and nought players respectively. Also, the BOARD class should have a function that checks for illegal moves and game-ending moves (winning/loosing/draw moves).

  Note: A very good source on classic tic-tac-toe games is available in Chapter 4 in a book titled Deep Learning and the Game of Go, which is available as a digital copy through our S&T library.

- **MC-CFR:** In games with large number of possible sequence of moves, the tree is typically unknown a priori. Then, the typical practice is to estimate the best move, given the current game state. Implement two algorithms: (i) MINIMAX, and (ii) MCCFR_OUTCOMESAMPLING to solve LTTT, and compare their average performance of the three algorithms. Note that MINIMAX algorithm is the one that finds subgame perfect Nash equilibrium (SPNE) in zero-sum games.

Note: labml.ai provides an detailed annotated implementation (includes both theoretical justification and Pythoncode) for Monte-Carlo counterfactual regret minimization with chance sampling for Kuhn poker.

- **Final Report:** Your report should clearly explain the need for regret minimization algorithms and their connection to equilibrium notions, and why regular counterfactual regret minimization is insufficient to solve large games. Also, clearly describe the MC-CFR_OutcomeSampling algorithm in detail, and demonstrate how the algorithm solves LTTT using an illustrative example. Finally, analyze the average performance of your MC-CFR_OutcomeSampling program at the cross player when it plays against a nought player using Minimax algorithm. While students are allowed to use Generative AI (GenAI) tools during project development, they should clearly disclose the submitted queries/tokens, the responses given by the GenAI tool, and how they developed their solution using the GenAI's response.

Note: A mandatory requirement is that the first page of the report should clearly describe the contributions of each team member.

Rubrics

Grades will be assigned based on the students' weekly update as well as the written report as shown below:

- Modeling the Game: 3 pts
- Solution Concept: 3 pts
- Implementation Style: 2 pts
- Final Report: 2 pts