University of Puerto Rico
Department of Electrical and Computer Engineering
ICOM5015 Artificial Intelligence

Adversarial Search

Group C

Manuel Alejandro Umaña Rodriguez -Undergraduate Computer Engineering
Dahyna Martínez Pérez -Undergraduate Computer Engineering
Jan Luis Pérez de Jesús -Undergraduate Computer Engineering
Christopher Hans Mayens Matías -Undergraduate Computer Engineering

For: Profesor José Fernando Vega Riveros
Date: April 23, 2025

Agenda

01 Purpose of experiment

02 Hypothesis

03 Concepts

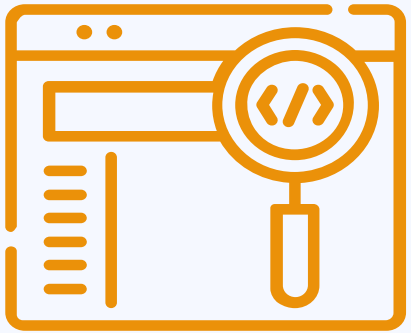
04 Experiments set up

05 Information

06 Conclusion

07 Credits & References

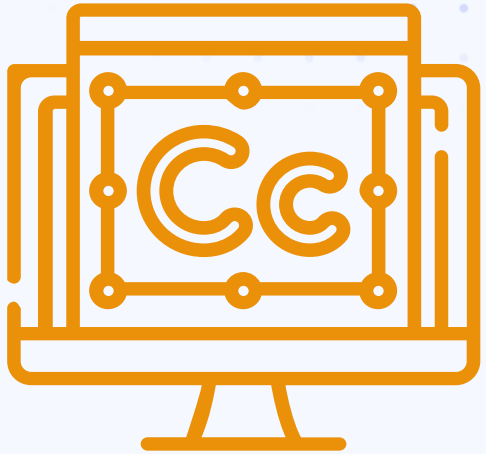
01
Purpose of experiment



- Implement a functional AI agent capable of playing the game of Dominoes under modified Puerto Rican rules.

- Study the effectiveness of the Monte Carlo Simulation as an adversarial search strategy.

- Evaluate AI performance across different player configurations such as solo play, team-based, and full AI simulations.



02 Key questions and hypothesis

Can an AI agent using Monte Carlo simulation consistently perform effectively in different Domino game configurations?

- Does the amount of simulations affect the performance of the agent?
- How can efficiency be measured for an AI agent in a Domino game?

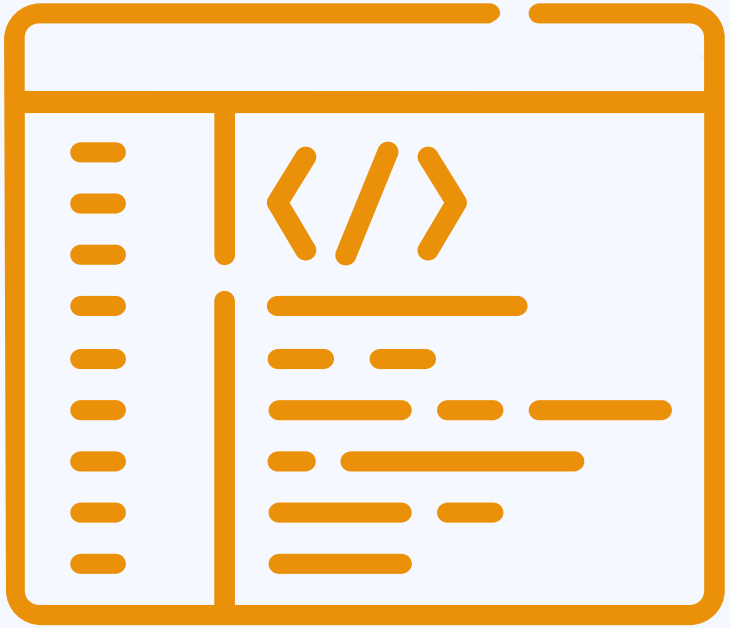
Hypothesis: If the AI agent is allowed to perform a sufficient amount of Monte Carlo simulations per turn, then it will be able to outperform human players in all the game configurations.

03 Concepts

Experimental concepts

Platform

Python: Programming language of high level. Emphasizes code readability with the use of indentation.

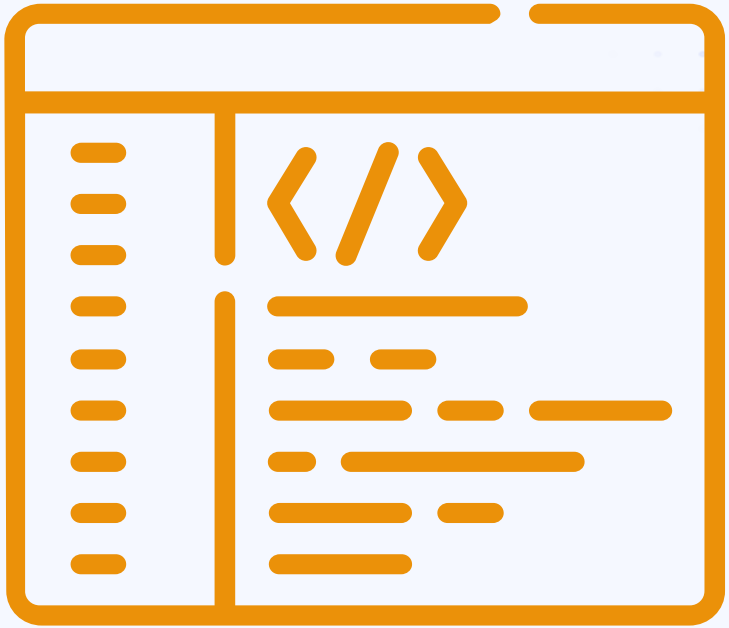


Subjects

- Adversarial Search
- Monte Carlo Simulation
- Multi-Player Configurations
- Human Win Ratio
- AI win Ratio

Experimental concepts

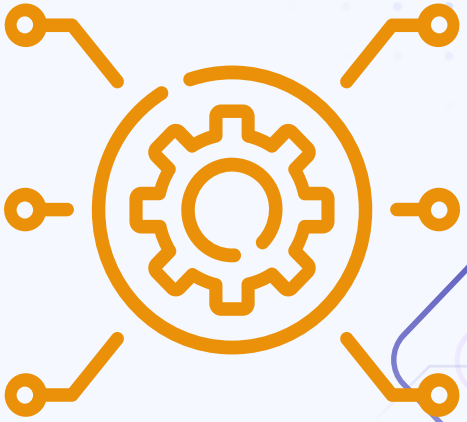
Dominoes Game (Puerto Rican Rules)- A digital implementation of the classic Dominoes game with adjusted rules for localization. It featured multiple play modes, supporting both human and AI players, with team-based and free-for-all configurations. The AI uses Monte Carlo Simulation to make strategic decisions in real-time gameplay.



Measure

Performance - Amount of wins obtained by the AI divided by Number of games played multiplied by 100.

04
Experiments set up.

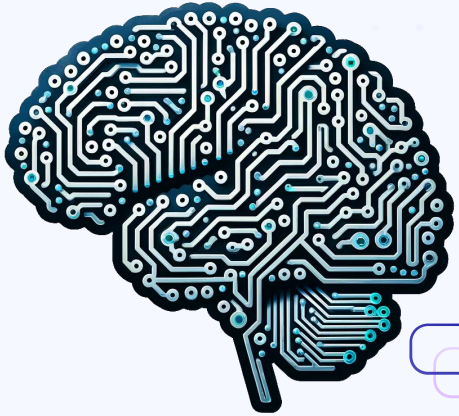


Tools and Resources Utilized

The following tools and resources were used:

- **Domino Game**: An article by the website Medium which had a base Dominoes game was used to gather insight.

- **Modified Domino Rules**:
  - First, the tiles are shuffled and each player is given 7 tiles.
  - The first play is determined by the player with the highest double tile.
  - The players match tiles to the open ends that have the same number on them, but if they can't play they can either draw an extra tile at random or pass their turn.
  - The game ends when a player runs out of tiles. If the game gets stuck the player with the least amount of points wins. Points refer to the total sum of the numbers in tiles in the player's hand.



Tools and Resources Utilized

The following tools and resources were used:

- **Monte Carlo Simulation:** Is a statistical technique for calculating the likelihood of various outcomes in a process that involves randomness or uncertainty.

- It operates by conducting numerous trials with randomly generated inputs based on predetermined probability distributions. The range and probability of the potential outcomes are then determined by analyzing the trail results.

Method for comparing

Criteria Used for Comparison:

- **Performance:** If the AI agent successfully won a game against any number of human player it added to its performance with a best out of 3 games being the base for the experimentation of the effectiveness of the agent.

- **Efficiency:** Evaluates if the agent could successfully beat human players at the game. If it could it was deemed as a good.

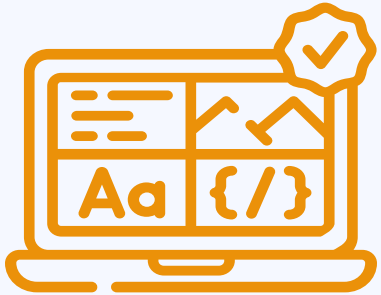
Method for graph creation

How the Data for the graph was collected and used.

- **Data was collected by playing the game:** The information was collected by playing 3 games of dominos with different configurations for both free for all and in different teams.

- The collected data was compared to different instances of game win ratios of the AI agent to evaluate the overall effectiveness of the agent.

05



Information.

Code Implementation:

```python
def monte_carlo_ai_move(self, simulations=30):  1 usage  JanPerez35
    """
    Evaluate possible moves via Monte Carlo playouts and return best one.
    Currently, it runs 30 simulations per move.

    Args:
        simulations (int): Number of random playouts per move.

    Returns:
        Optional[Tuple[int, int]]: Best tile to play or None to pass.
    """
    hand = self.game.players[1]
    valid = self.game.get_valid_moves(hand)
    if not valid:
        return None

    scores = {}
    for m in valid:
        wins = 0
        for _ in range(simulations):
            sim = copy.deepcopy(self.game)
            try:
                sim.play_tile( player: 1, m)
            except ValueError:
                continue
            if self.simulate_random_playout(sim) == 1:
                wins += 1
        scores[m] = wins / simulations
    return max(scores, key=scores.get)
```

Figure 1. Monte Carlo Algorithm in Python

Information about the games:

Game mode no teams	Performance out of 3 games for AI	Game mode in teams (2v2)	Performance out of 3 games for AI.
1 player VS 1AI	33.33%	N/A	N/A
2 AI (spectate)	66.67.%	N/A	N/A
3 players VS 1 AI	66.67%	3 players VS 1 AI	66.67%
2 players VS 2 AI	66.67%	2 players VS 2 AI	33.33%
1 player VS 3 AI	100.00%	1 player VS 3 AI	66.67%
4 AI (spectate)	100.00%	4 AI (spectate)	100.00%

Table 1. AI Performance Results in team-mode utilizing 2 and 4 Player games

Information for games:

Simulations for Monte Carlo (1 AI vs 1 Player Gamemode)	Performance out of 3 games for AI
30	33.33%
100	33.33%
500	66.67%
1000	66.67%
5000	66.67%
10000	100%

Table 2. AI performance utilizing different Monte Carlo Simulations

Graph for number of wins VS simulations

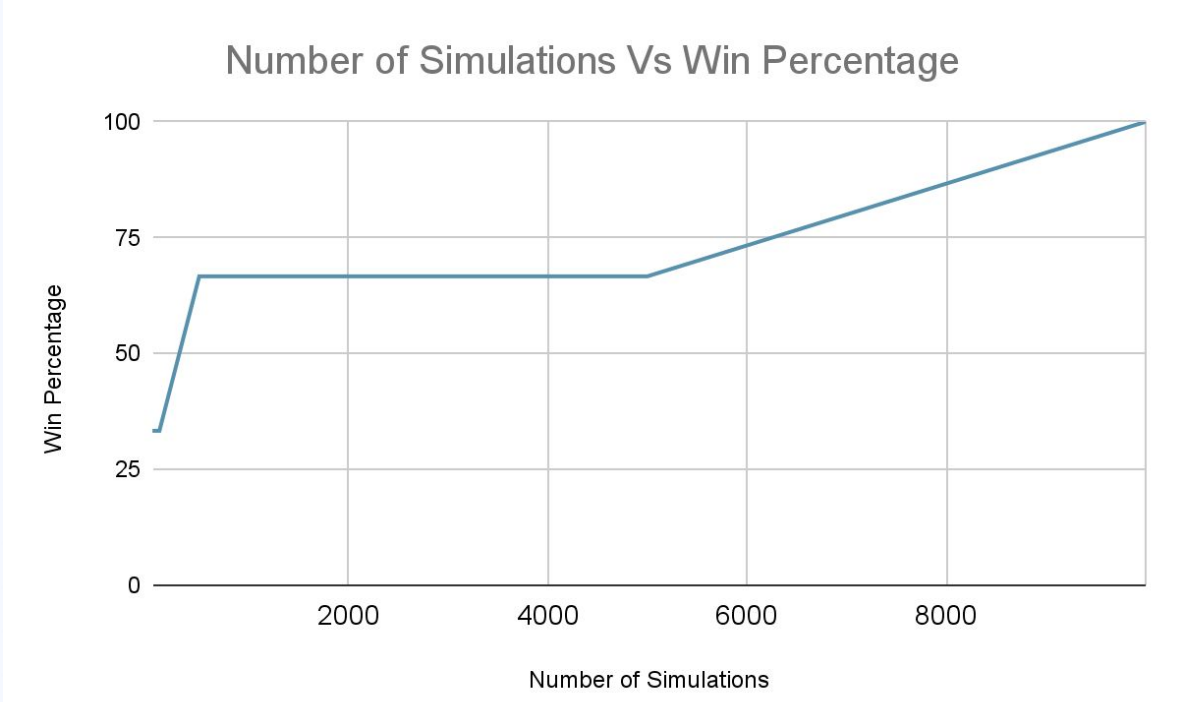


Figure 2. Number of Simulations vs Win Performance

Information about the games:

Game mode	Performance out of 3 games for AI
2 players VS 2 AI in teams	33.33%
4 player free for all	33.33%
4 player in teams	100.00%

Table 3. Tournament simulation with 3 Players and 1 AI

06

Conclusion and Lessons Learned.

Conclusions and Lessons Learned

01 **Monte Carlo and Number of Simulations**

Results obtained showed that the AI struggled in scenarios where human uncertainty was prevalent. Performance significantly improved with the number of simulations per turn.

02 **Structured Environments**

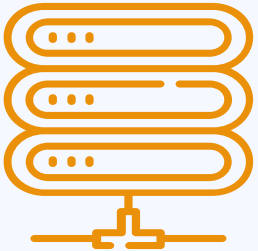
The performance of the AI agent increased when certain conditions were met such as having the first turn or playing with the lowest amount of humans as they introduced unpredictability.

Conclusions and Lessons Learned

03 — **Computation Cost and Performance**

Increasing the level of simulations improved the performance of the Monte Carlo AI significantly, but it came with a big computational cost. Optimizing for different levels of complexity and performance would be the best way to obtain an efficient solution.

07
Credits and References.



References

[1] S. Sharma, “What is Monte Carlo Simulation,” geeksforgeeks.org, Aug. 26 ,2024. [Online]. Available:https://www.geeksforgeeks.org/what-is-monte-carlo-simulation/ . [Accessed Apr.19 , 2025].

[2] Domino Rules , “What is Monte Carlo Simulation,” dominorules.com, 2016. [Online]. Available: The Basics . [Accessed Apr. 19, 2025].

[3] A. Golovin, “Dominoes game with simple AI in Python,” medium.com, Jun 7, 2022. [Online]. Available: Dominoes game with simple AI in Python | by Aleksandr Golovin | Medium .[Accessed Apr. 19, 2025].

[4] M. Gupta, “Stochastic Games in Artificial Intelligence,” geeksforgeeks.org, Sep 11, 2024, [Online]. Available: Stochastic Games in Artificial Intelligence | GeeksforGeeks .[Accessed Apr. 19, 2025].

References

[5] L. Kaelbling, M. Littman, A. Cassandra, “Planning and acting in partially observable stochastic domains” sciencedirect.com, Jan. 17, 1998, [Online]. Available: Planning and acting in partially observable stochastic domains - ScienceDirect .[Accessed Apr. 19, 2025].

[6] G.Piliouras, R.Sim, S.Skoulakis, “Beyond Time-Average Convergence: Near-Optimal Uncoupled Online Learning via Clairvoyant Multiplicative Weights Update,” arxiv.org , Jun 7, 2022, [Online]. Available:[2111.14737] Beyond Time-Average Convergence: Near-Optimal Uncoupled Online Learning via Clairvoyant Multiplicative Weights Update.[Accessed Apr. 20, 2025].

[7] D. Silver et. all, “ Mastering the game of Go with deep neural networks and tree search,” nature.com, Jan 27, 2016. [Online]. Available: Mastering the game of Go with deep neural networks and tree search | Nature .[Accessed Apr. 20, 2025].