TH
E GAME

BAGH-BANDI

Bagh-Bandi Game: AI Strategies

Exploring Ancient Games Through Modern AI

- **Authors**: **Team 8**

  Fardin Saad (fsaad)

  Adittya Soukarjya Saha (asaha4)

  Vinay Vobbilichetty (vobbil)

  Uchswas Paul (upaul)

Contents

- Introduction
- What we have done
- Game Play Rules
- BFS
- DFS
- A*
- Monte Carlo Tree Search

Introduction

- Bagh-Bandi is a captivating adversarial board game that originates from Lower Bengal.

- It is played on a 5x5 grid with two opposing sides: one controlling 25 goats and the other 4 tigers

- The objective for the goats is to encircle and immobilize the tigers, while the tigers aim to capture the goats by jumping over them.

- In this project, we have implemented the tiger's movements to be controlled by a human player, whereas the goat's strategies are dictated by sophisticated AI algorithms.

What We have Done

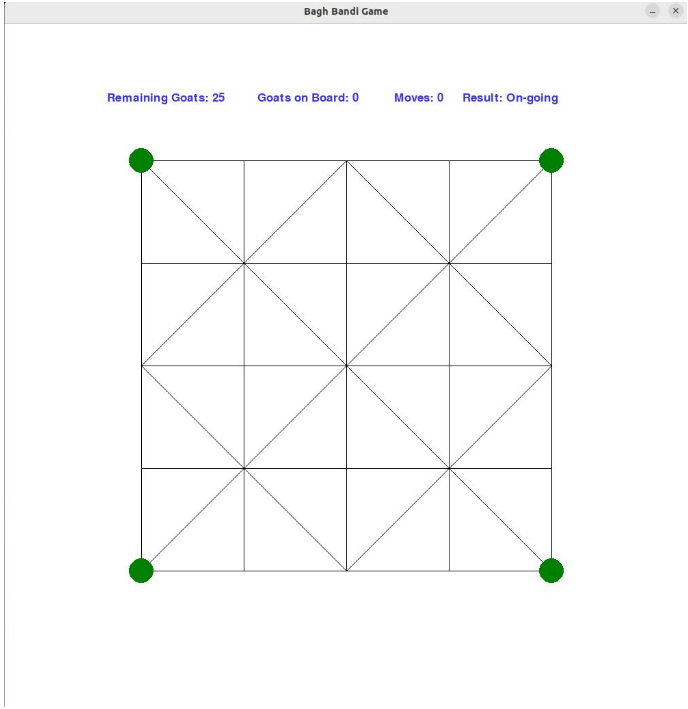
Game Setup

- Graphics rendering and UI
- Event management
- Game logic

Goats movement and AI Algorithms

- Breadth first Search (BFS)
- Depth First Search
- A*
- Monte Carlo Tree Search
- Random Search

Comparative Analysis



Game Playing Rules

Initial Setup

The game begins with four tiger pieces strategically positioned at each corner of the board.

Movement Rules

Both tigers and goats have the ability to move to any adjacent unoccupied cell.

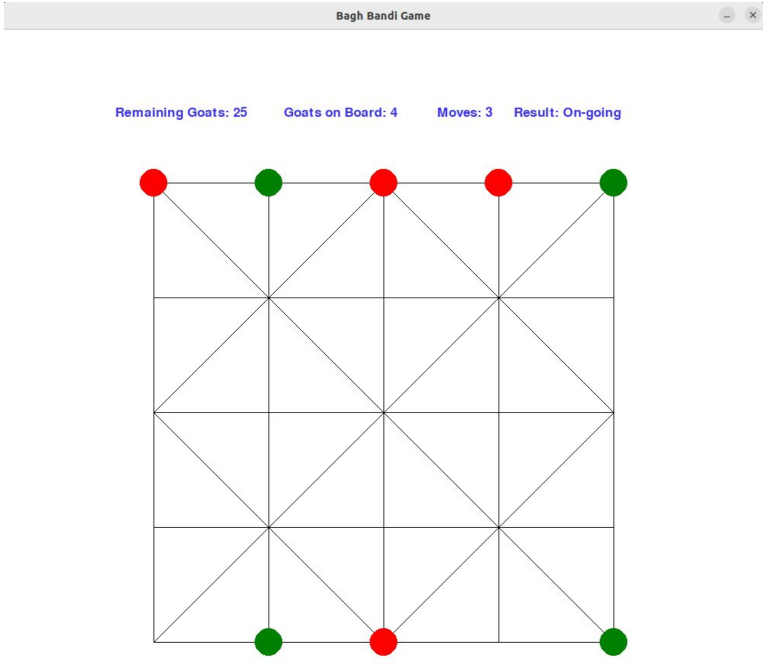
Capture Mechanics

Tigers can capture goats by jumping over them to land on a directly opposite free spot.

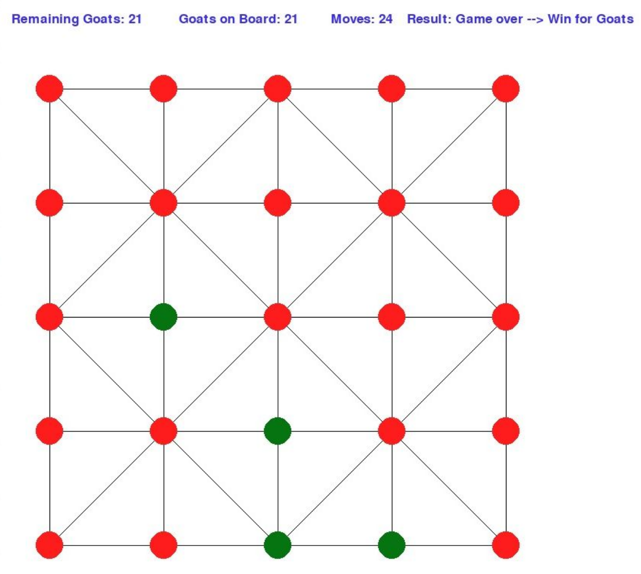
Winning Conditions:

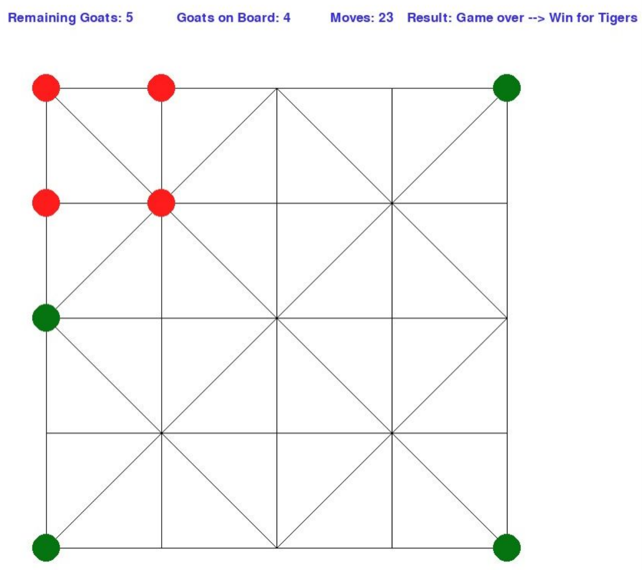
The tigers win if they capture all the goats.

The goats win by immobilizing the tigers, preventing any further movement.

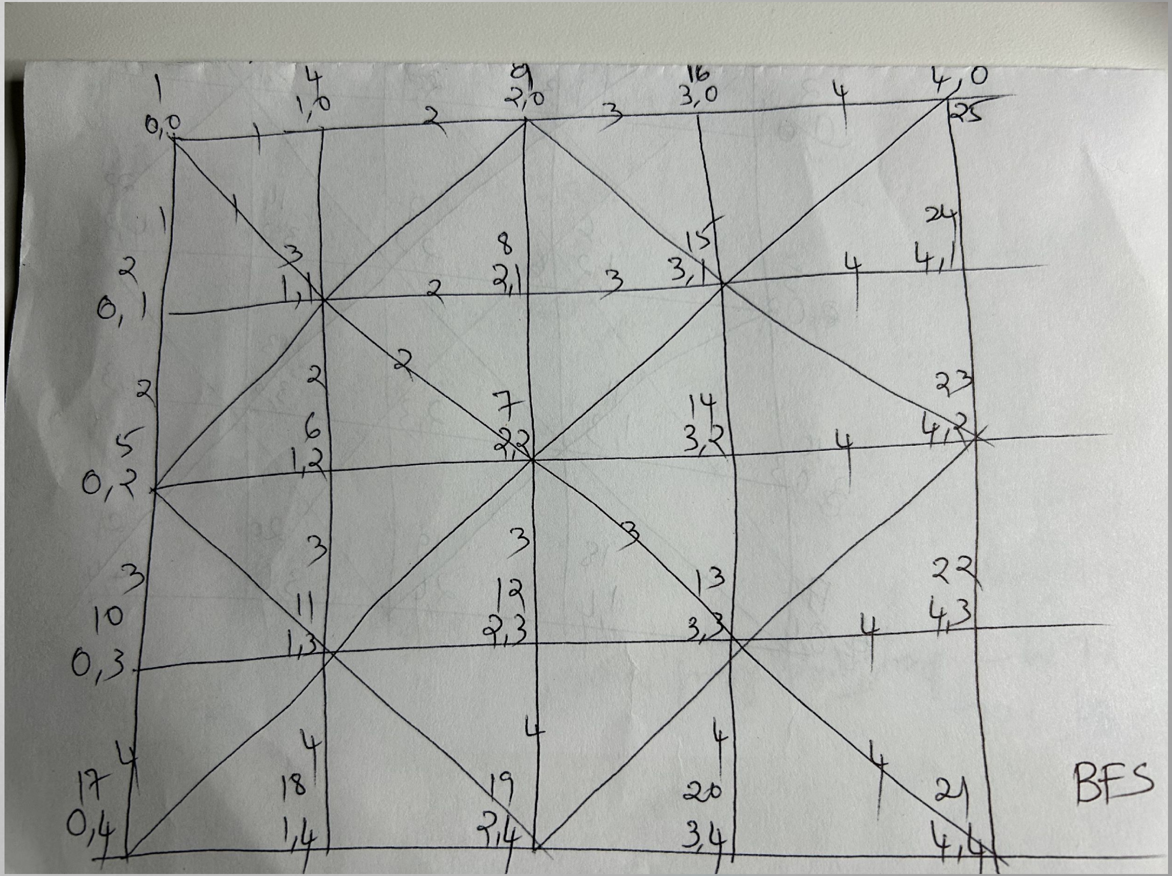


Winning Conditions

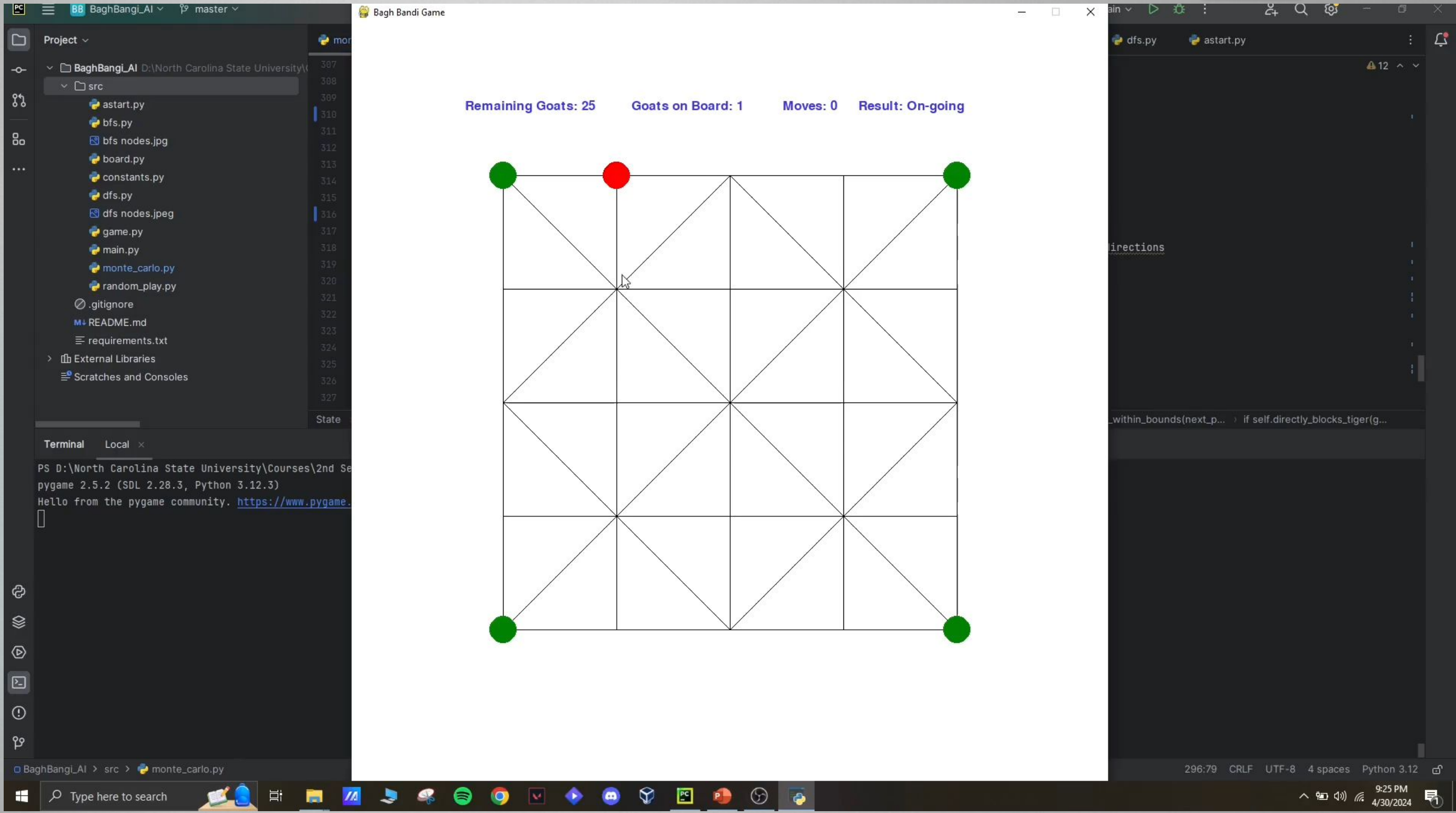




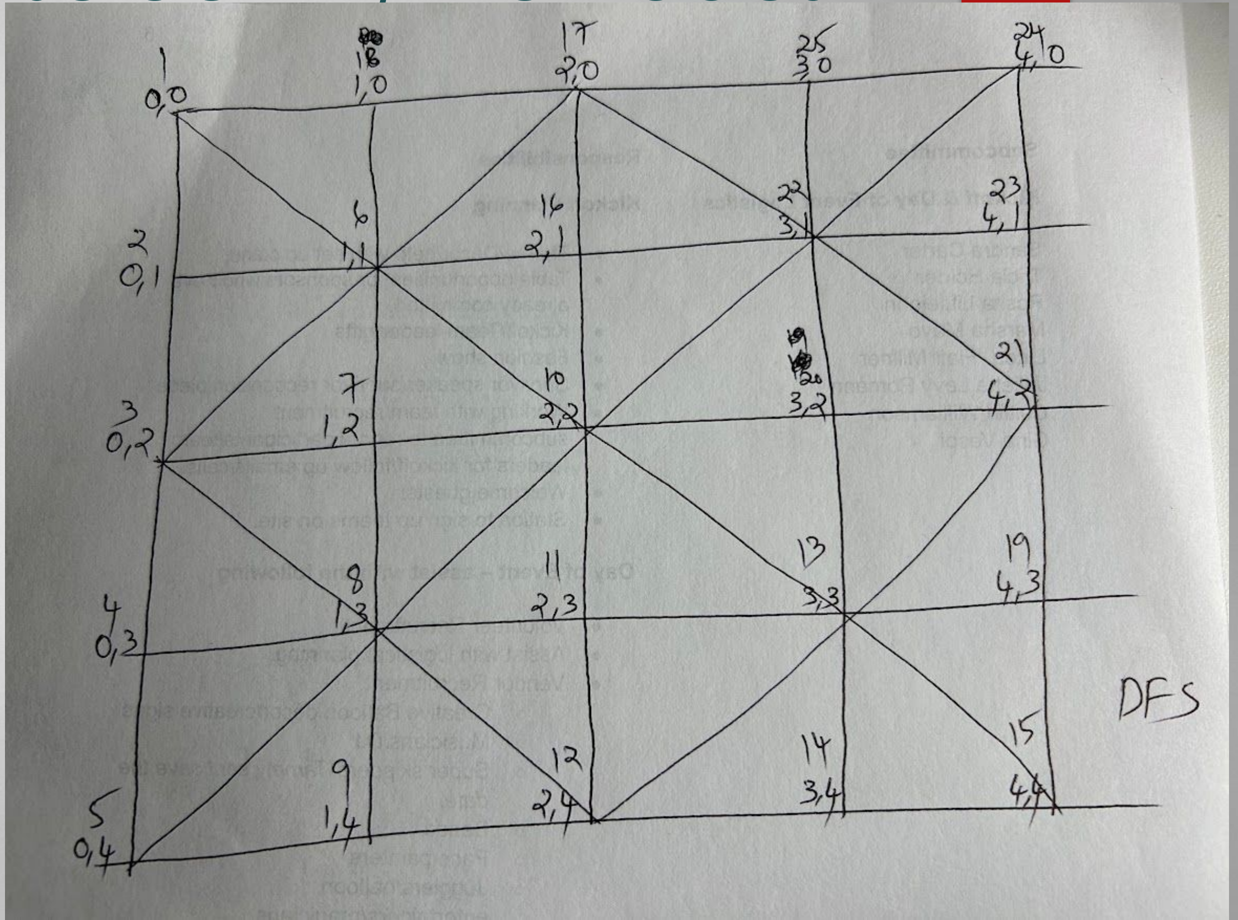
BFS – order of selecting the nodes



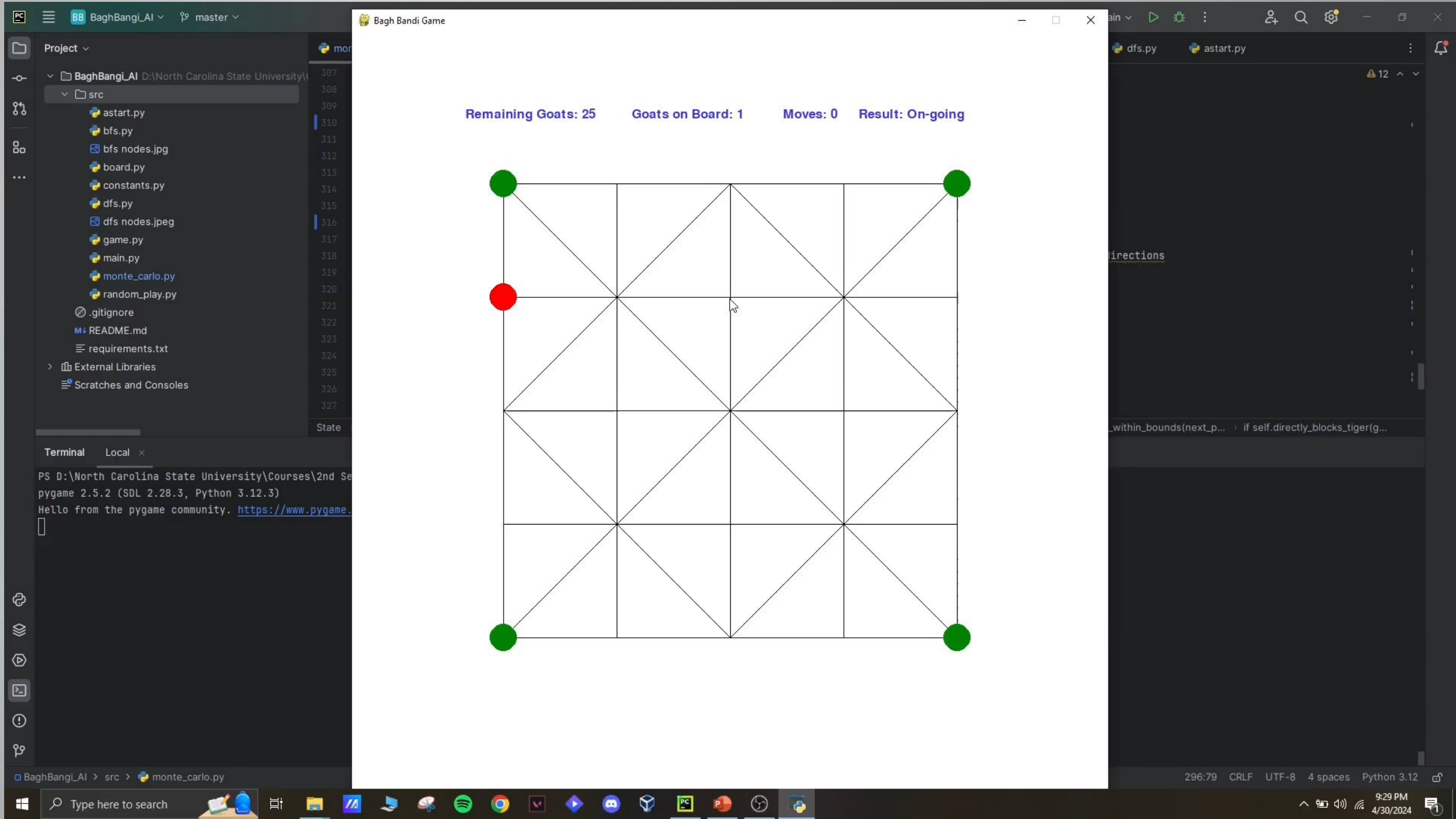
BFS – Game play



DFS – order of selecting the nodes



DFS – Game play



A* Algorithm

Background on the A* Algorithm

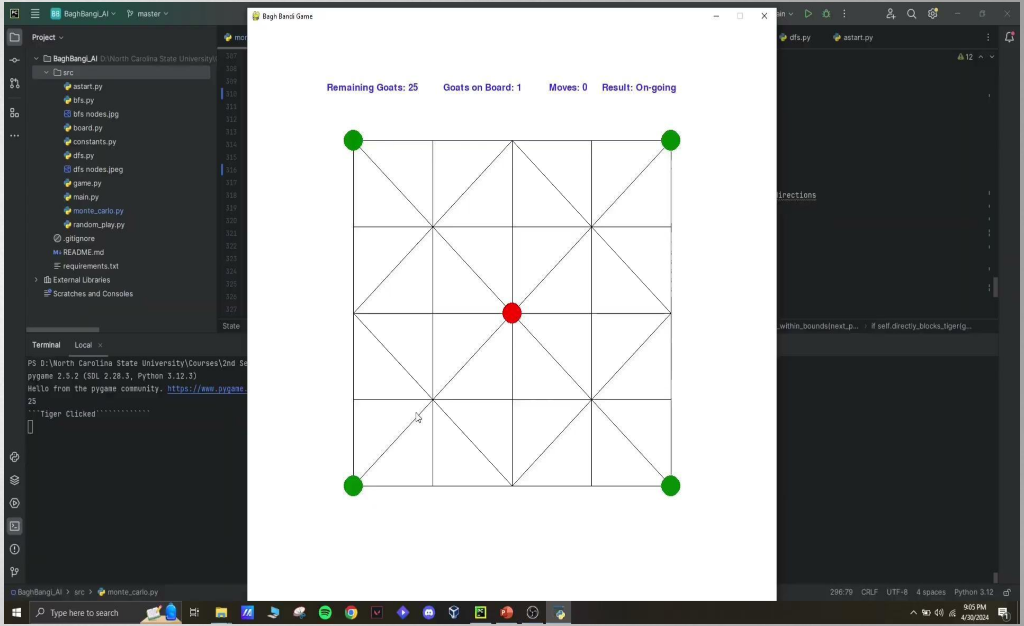
- A* algorithm is an extension of Dijkstra's algorithm with a heuristic component, which helps it to search more efficiently by guiding the search towards the goal node.
- At each step, A* evaluates nodes to expand based on their total cost f(n), prioritizing nodes with lower total cost.

How the heuristic values are set

- Positive heuristic when no adjacent nodes contain tigers [Irrespective of goats being present at the adjacent nodes]
- Negative heuristic when there is an adjacent node containing tiger and exactly opposite adjacent node contains empty space.

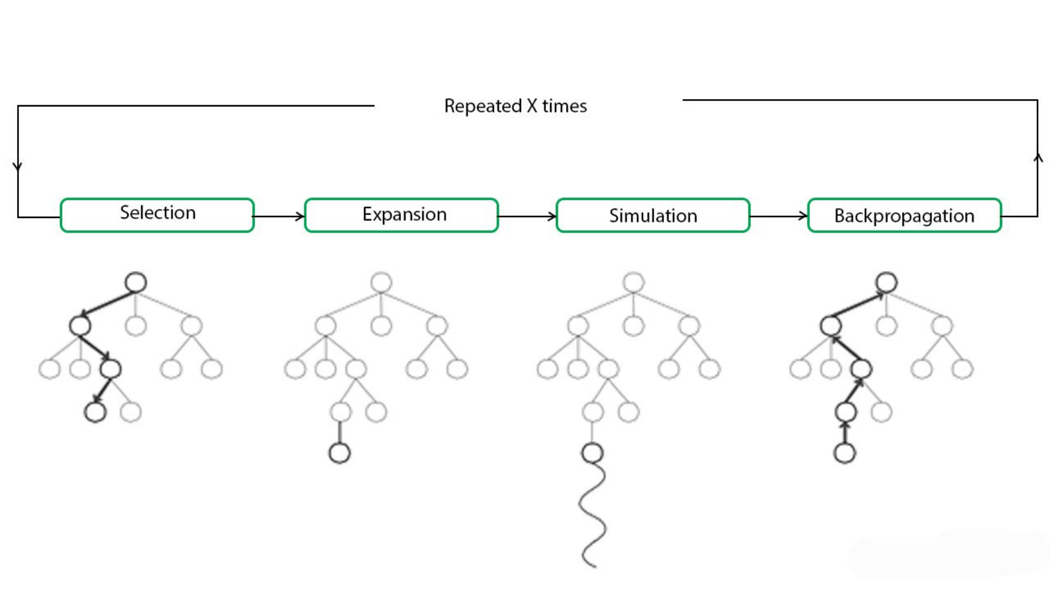
Demo of A* Algorithm

- Application in the Bagh Bandi game
  - In the A* algorithm we considered paths that minimize the likelihood of encountering tigers.
  - The A* algorithm would explore all the possible empty locations that the goats can be placed, corresponding to the moves of the goat while considering the positions of both the player’s pieces and the tigers on the board.
  - Based on heuristics stated, at each position based on the estimated risk of encountering a tiger, it would evaluate each potential move and then choose the move that gives out a better heuristic value.

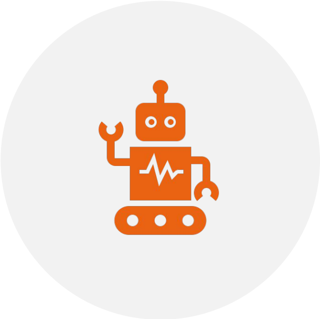


Monte-Carlo Tree Search

- It is a probabilistic and heuristic driven search algorithm that combines the classic tree search implementations alongside machine learning principles of Reinforcement Learning (RL).
- Four key steps of algorithm.
  i. Selection:
  ii. Expansion:
  iii. Simulation
  iv. Back-propagation



Strategic AI Implementation

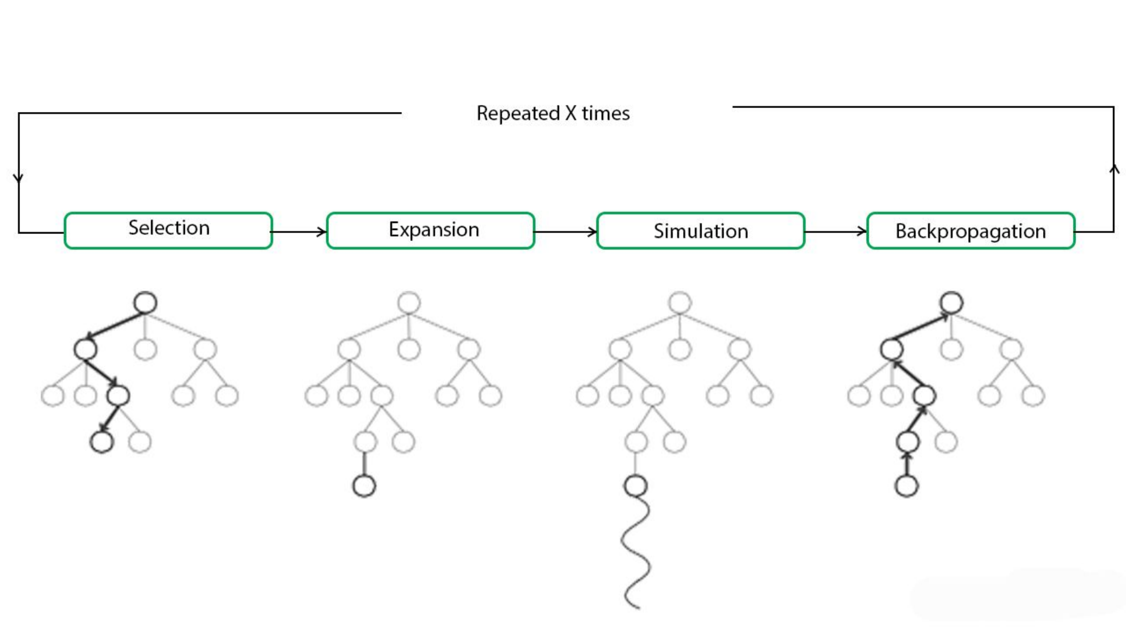


UTILITY EVALUATION: HOW THE AI EVALUATES THE UTILITY OF MOVES AND PREDICTS OUTCOMES.

DECISION SPACE NAVIGATION: TECHNIQUES USED BY AI TO NAVIGATE THROUGH THE DECISION SPACE.

Monte-Carlo Tree Search

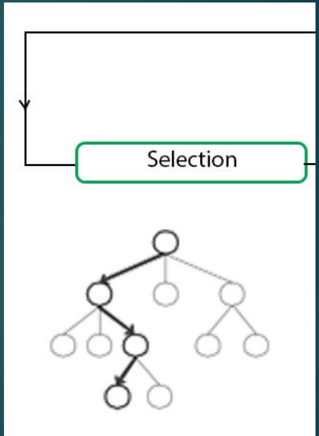
- It is a probabilistic and heuristic driven search algorithm that combines the classic tree search implementations alongside machine learning principles of Reinforcement Learning (RL).
- Four key steps of algorithm.
  i. Selection:
  ii. Expansion:
  iii. Simulation
  iv. Back-propagation



MCTS: Selection

- Selection: Starts from the root node and progress towards the most promising leaf node based on a policy known as the **Upper Confidence Bound 1 (UCB1)**.
  - The core decision-making aspect of the Selection phase in the MCTS class is governed by the UCB1 formula, which is designed to balance exploitation and exploration, ensuring that the search is both thorough and directed.

$$UCB1 = \frac{W_i}{N_i} + C\sqrt{\frac{\ln N}{N_i}}$$



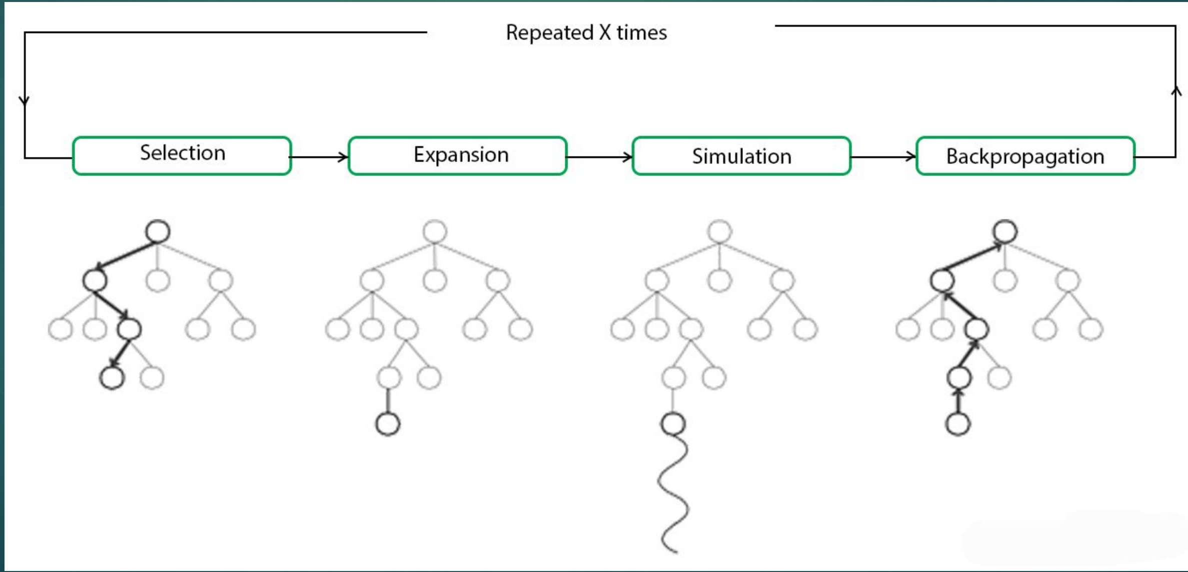
- W_i = no. of wins after i-th move
- N_i = no. of simulations after i-th move
- N = total number of simulations
- C = exploration constant $\sqrt{1.5}$ (balance between exploration and exploitation)
- Exploitation = $\frac{W_i}{N_i}$, focusing on nodes that have historically performed well.
- Exploration = $C\sqrt{\frac{\ln N}{N_i}}$, encouraging the algorithm to explore less-visited nodes to ensure that no potential move is overlooked.

Expansion, Simulation, & Backpropagation

ii. Expansion: A new child node is added to the tree to that node which was optimally reached during the selection process.

iii. Simulation: A simulation is performed by choosing moves or strategies until a result or predefined state is achieved.

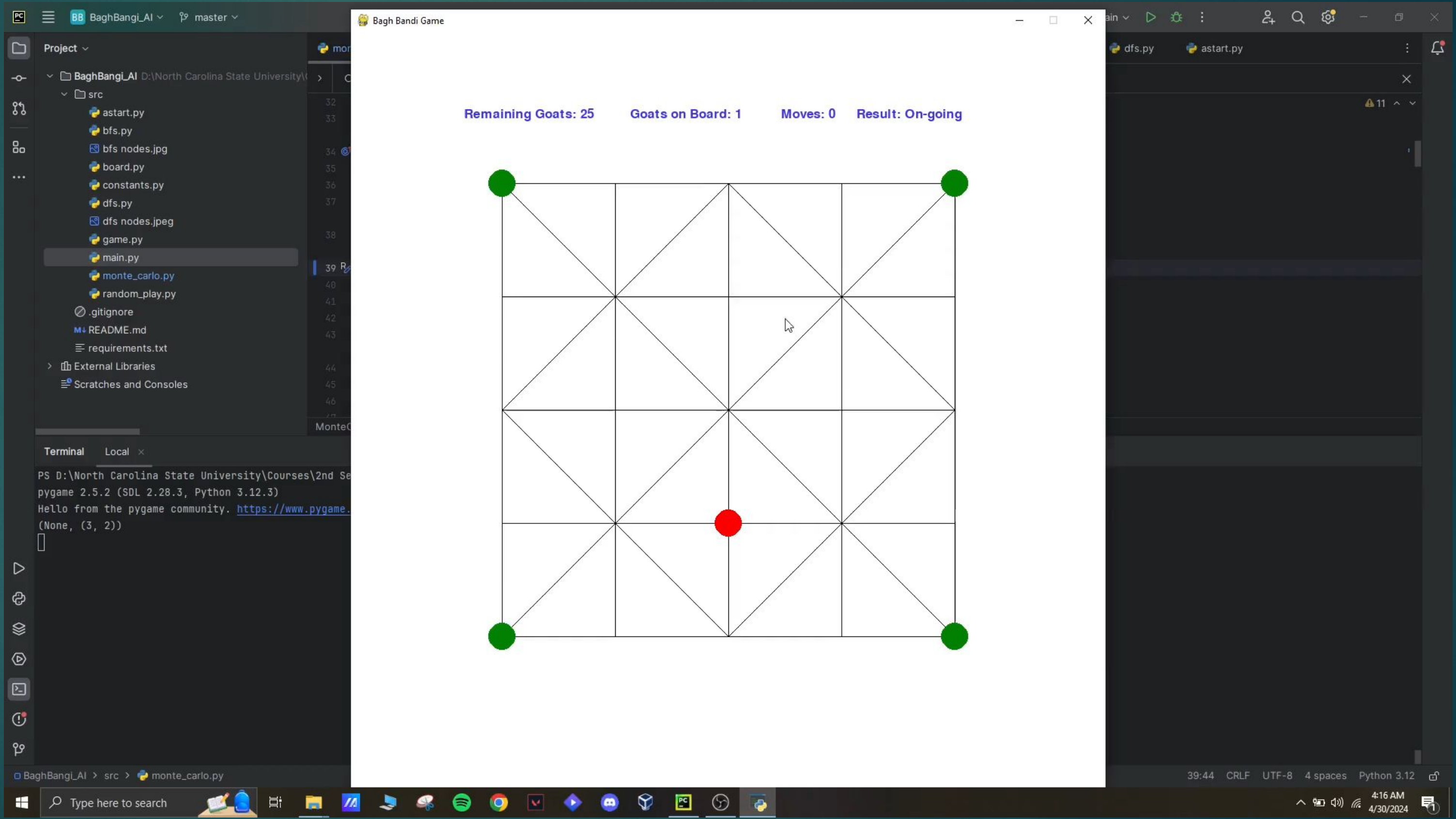
iv. Back-propagation: After the simulation, the results are propagated back up the tree. Each node visited during the Selection phase is updated with the outcome of the rollout, influencing future traversals and selections.



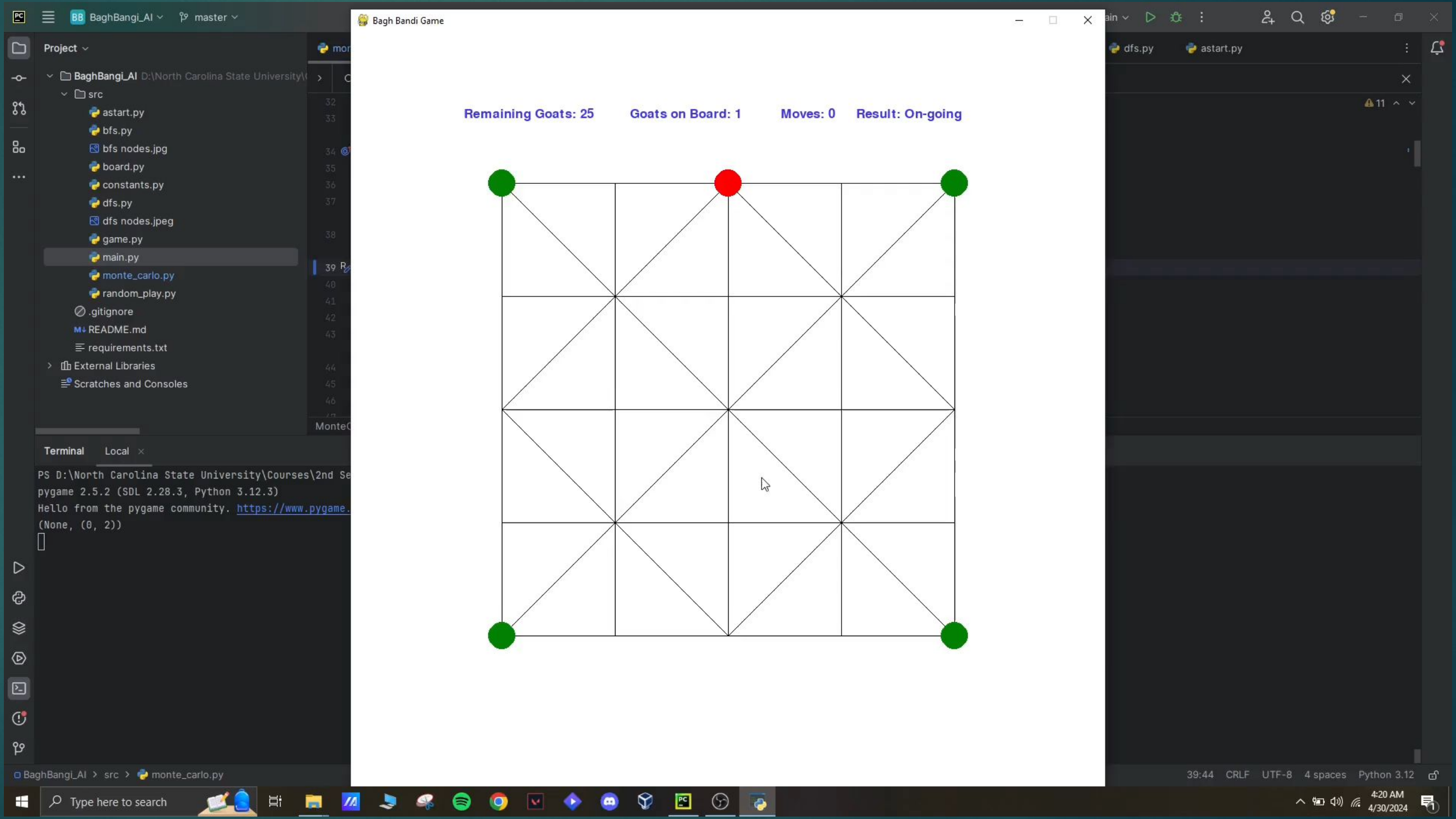
Our Implementation

- In our implementation we strategically implemented the goat's move by assigning rewards and penalty (scores) for different goat positions.
- Goat's strategic moves:
  - **Protective moves:** Placing a goat adjacent to an endangered goat.
  - **Escape moves:** Moving an endangered goat that is in threat of being captured by the tiger.
  - Each move is evaluated based on its strategic impact on the game, focusing on preserving goat pieces and restricting tiger movement.
- Evaluating the game state:
  - **Goat Captures:** A negative score is applied for each goat in immediate danger of being captured, encouraging moves that lead to safer positions.
  - **Protective Neighbors:** A positive score is awarded for goats that are positioned next to other goats, creating protective clusters that are harder for tigers to breach.
  - **Escape Potential:** Points are given for moves that allow endangered goats to escape to safer locations.
  - **Terminal State:** A high score is applied if the goats win and very low score is applied if the tigers win.

Game Played using MCTS (1)



Game Played using MCTS (2)



Comparative Analysis of AI Algorithms

- Total games played for each algorithm = 20

Algorithms	No. of Wins (N_W)	Avg. no. of Moves (M)	Good Moves (G_M)	Bad Moves (B_M)	Performance (P) $P = \frac{G_M}{M}$	Winning Rate (W) $W = \frac{N_W}{20}$
Random Play	1	73	22	51	30.14%	5%
BFS	7	39	22	17	56.41%	35%
DFS	5	43	20	23	46.51%	25%
A*	13	33	24	8	72.73%	65%
Monte-Carlo	**17**	31	28	3	**90.33%**	**85%**

Conclusion

The analysis highlights the strength of heuristic-based and probabilistic strategies in complex games like Bagh Bandi.

Monte-Carlo Tree Search, leveraging Reinforcement Learning exploration-exploitation and deep heuristic evaluations, outperformed the rest of the AI search algorithms

This demonstrates that advanced AI techniques can significantly improve performance in games needing tactical depth and foresight.

This study also confirms the theoretical effectiveness of these algorithms and their practical impact on AI gaming strategies.

References

[1] Bagh-chal (2024) wikipedia. available at. https://en.wikipedia.org/wiki/Bagh-chal. [Accessed 08-03-2024].

[2] I. Ahmed, D. Guan, M. N. U. Laskar, and T. C. Chung. Minimax self playing game alquerque. In Future Information Technology: FutureTech 2013, pages 197–203. Springer, 2014.

[3] D. F. Beal. A generalised quiescence search algorithm. Artificial Intelligence, 43(1):85–98, 1990.

[4] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. IEEE Transactions on Computational Intelligence and AI in games, 4(1):1–43, 2012.

[5] Y. Farhang, M. R. Meybodi, and A. Hatamlou. Improving the efficiency of forward checking algorithm for solving constraint satisfaction prob-

lems. In 2008 Eighth International Conference on Intelligent Systems Design and Applications, volume 1, pages 240–245. IEEE, 2008.

[6] P. C. Iloh. A comprehensive and comparative study of dfs, bfs, and a* search algorithms in a solving the maze transversal problem. International Journal of Social Sciences and Scientific Studies, 2(2):482–490, 2022.

[7] R. Pramanik and S. Bhattacharya. Play and indigenous games of children: A cultural heritage of western odisha, india. Antropolog´ıaExperimental, (19), 2019.

[8] R. Rahim, R. Dijaya, M. Multazam, A. D. GS, and D. Sudrajat. Puzzle game solving with breadth first search algorithm. In Journal of Physics: Conference Series, volume 1402, page 066040. IOP Publishing, 2019.

[9] P. Van Beek. Backtracking search algorithms. In Foundations of artificial intelligence, volume 2, pages 85–134. Elsevier, 2006.

THANK YOU!!