CS 4200 Final Project Report: Connect Four

Corey Lang and Jenna Mauch

Introduction:

Using artificial intelligence, or AI, to make game-playing agents is extremely popular. Game-playing agents can make decisions in complex game environments based on their algorithm, with some agents even having the capability to defeat or outperform human players. As AI continues to advance, researchers and developers are looking for the most effective algorithms in order to perfect AI. For the purpose of our project, we attempted to replicate this research on a small-level scale with the game Connect Four.

Connect Four is a two player game utilizing a 6x7 grid. Each player must drop their team-colored disc into a column with the goal of forming a line of four with their own discs. The line may be vertical, horizontal, or diagonal. Each player must compete against each other strategically, as each piece placed can be used either defensively or offensively in an attempt to be the first to win. The game is simple and allows for game-playing agents to be easily analyzed, making Connect Four an ideal candidate for our project.

The purpose of our project is to determine the highest performing algorithm to use for a game-playing agent in Connect Four, with the primary objective of maximizing the win-rate. The definition of highest performing in this case is defined by the win-rate; this is important to note, as the definition of high performance can depend on the context, i.e. if the algorithm is meant to play against a child and is meant to easily be played against.

In order to achieve our objective, we chose three algorithms for our game-playing agents: Breadth-First Search (BFS), Greedy Search, and Minimax. Originally we had planned to do two uninformed search algorithms and two informed search algorithms. A* Search and Uniform Cost Search were part of this plan, but could not be implemented due to the nature of the game. Minimax was added to our plan, as it seemed to be popular with related projects. Each of these algorithms provides a unique approach to the same problem, allowing for broader results and interesting comparison.

After implementation of our three algorithms, we evaluated their performance in a series of Connect Four games. By comparing their win-rates, we aimed to determine the best algorithm of the three for this particular game. Adding to another layer of analysis, we also increased the size of the board to an 8x10 grid. Through our process, we aimed to gain valuable experience in assessing the strengths and weaknesses of each algorithm and why they performed the way they did. Because of our approach, we were successful in addressing the main problem; finding the highest-performing algorithm to use for Connect Four.

Related Works:

The application of artificial intelligence to game-playing agents has been a subject of interest for many years, with a wealth of literature exploring various approaches and algorithms. In this section, we present a brief overview of the most relevant research on AI algorithms applied to the Connect Four game, as well as more general game-playing strategies that can be adapted to Connect Four.

1. *Allis, V. (1988). A Knowledge-based Approach to Connect-Four. The Game is Solved: White Wins. Master's thesis, Vrije Universiteit.*

In this seminal work, Allis presents a knowledge-based approach to solving Connect Four, utilizing a combination of domain-specific knowledge and search techniques to prove that the first player (white) can always win the game if played optimally. This work provides valuable insights into the strategies that can be employed to improve the performance of game-playing agents in Connect Four.

2. *Donkers, J. H. M. (2001). Nosce Hostem - Searching with opponent models. Ph.D. thesis, Universiteit Maastricht.*

Donkers' thesis investigates the concept of opponent modeling in game-playing agents, with a specific focus on Connect Four. The work explores the impact of incorporating knowledge about the opponent's playing style into the decision-making process of a game-playing agent. This research is relevant to our project as it demonstrates the potential benefits of accounting for an opponent's strategy when designing an AI agent for Connect Four.

3. *Kocsis, L., & Szepesvári, C. (2006). Bandit based Monte-Carlo planning. In ECML (pp. 282-293).*

While not specifically focused on Connect Four, this paper introduces the Monte Carlo Tree Search (MCTS) algorithm, which has been applied to various games, including board games like Go and Chess. MCTS is a valuable addition to the repertoire of algorithms that can be employed in game-playing agents and may provide an alternative approach for our Connect Four agents.

4. *Arambakam, M. (2020). A multi-agent implementation of the game Connect-4 using MCTS, Minimax and Exptimax algorithms. Retrieved from https://github.com/nikolajob/Connect-Four-AI-Bot*

This repository implements and compares four distinct algorithms for solving the Connect Four game: One-Step Look Ahead Bot (onestep), MiniMax Bot (minimax), ExpectiMax Bot (expectimax), and Monte Carlo Tree Search (montecarlo). The project aims to determine the relative strengths and weaknesses of each algorithm in the context of Connect Four, providing valuable insights and code examples that can inform the development of our own game-playing agents.

5. *Galli, K. (2018). How does a Board Game AI Work? (Connect 4, Othello, Chess, Checkers) - Minimax Algorithm Explained. Retrieved from https://www.youtube.com/watch?v=y7AKtWGOPAE*

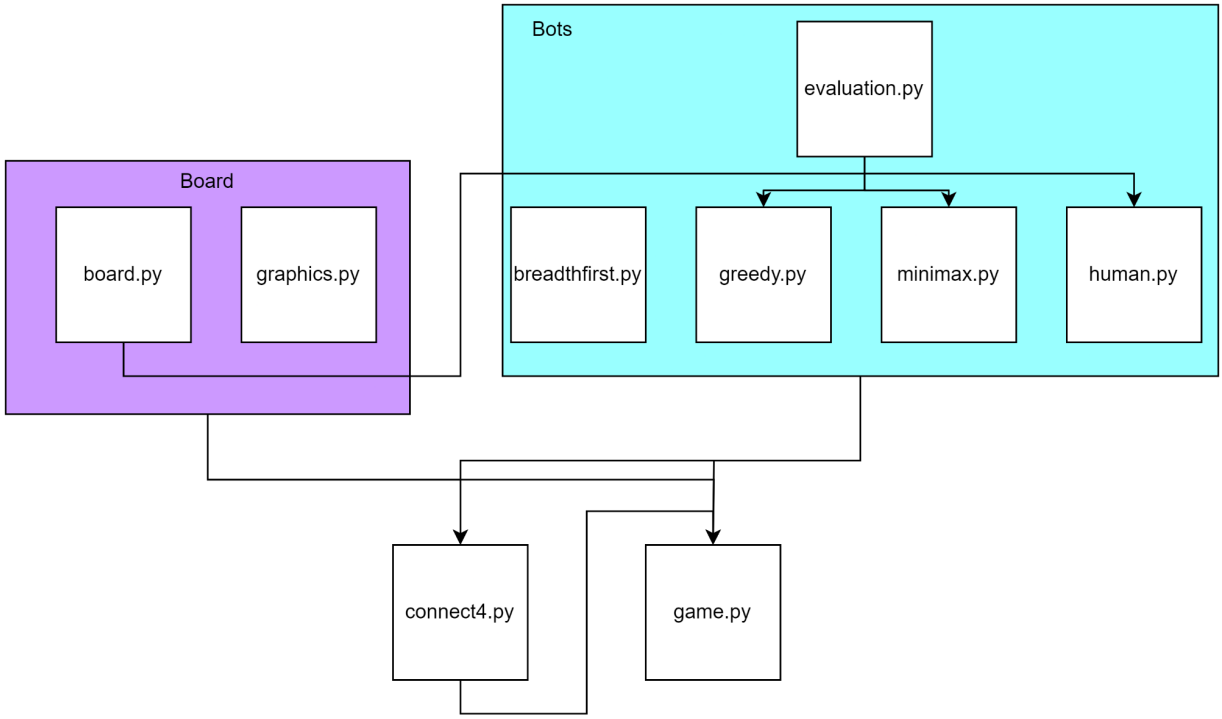
In this video, Keith Galli builds up the intuition for how an expert-level board game AI works. The video begins with a simple approach, making random moves, and then progresses to building board scoring heuristics and finally, to the MiniMax algorithm. The video offers a clear and concise explanation of the algorithm, making it an excellent resource for those interested in understanding how the MiniMax algorithm works and how it can be applied to the Connect Four game.

Approach:

To solve the problem, we chose search algorithms as our strategy. We did this because search algorithms are relatively simple, as well as being able to determine the next action based on the current environment. Connect Four is a relatively simple game, so using a simpler algorithm would be a better fit. Using Greedy Search and Breadth-First Search ensures the comparison of uninformed vs informed search algorithms, and the inclusion of the Minimax algorithm ensures that other popular methods are considered in the comparison.

In order to get results, we put our game-playing agents in match-ups against each other. Each agent played against each other for 5 matches, with wins being recorded. From this, we'd be able to evaluate the win-rate of each algorithm. This format was done for both the 6x7 board and the 8x10 board.

The software was developed in Python and uses the library Pygame. The game itself uses Corey's pre-existing code of Connect Four, which was developed by following KeithGalli's video walkthrough of programming the game. The code for Minimax algorithm is also from KeithGalli's repository using his video walkthrough of programming the AI [1]. The software also uses a GUI menu from a similar multi-agent repository of a Connect Four game [2]. The GUI allows us to easily select options for the purpose of our testing. The flowchart below shows the structure of our code, with the purple rectangle representing the Board directory and the blue rectangle representing the Bots directory. Each square represents a file. Each arrow pointing to a square represents that file being imported to another file. The flowchart clearly shows game.py as the centerpiece of the project, with every file being imported into it.



connect4.py	Defines rules for the game, including turns and defining game over.
game.py	File to run the game with - also puts the game together, defines who's playing against who, and creates the menu GUI.
board.py	Defines the board. Provides game-playing agents with information on the board, such as valid locations or piece positions.
graphics.py	Creates the graphics of the board.
breadthfirst.py	Algorithm for Breadth-First Search.
greedy.py	Algorithm for Greedy Search.
minimax.py	Algorithm for Minimax Algorithm.
evaluation.py	Provides score evaluation functions for the Greedy and Minimax algorithms.
human.py	Provides code for a human player to be able to play the game.

Evaluation and Results:

The Breadth-First Search bot explores the game tree at one level, considering all possible moves at the given depth before moving on to the next. This approach offers a comprehensive exploration of the search space, which can provide valuable insights into the optimal moves for the particular game state.

The Greedy bot, on the other hand, selects the move that appears to be the most beneficial in the short term, without considering the long-term consequences of its actions. This can sometimes result in rapid progress towards a solution, but may also lead to suboptimal decisions in certain scenarios.

Lastly, the Minimax bot employs a more strategic approach, evaluating the possible moves by simulating the game several moves ahead and selecting the move that minimizes the opponent's maximum gain. By considering both its own and its opponent's best moves, the Minimax bot aims to make the most informed decision possible.

To determine a solution from our experiment, we wanted to see which bot would win when competing against another bot as well as the time each bot takes in total to make a decision about which move to take. The unit for the timer metric is second to the nearest hundredths decimal place. We had each bot compete against another in a best of 5 games and keep note of the total time each bot took to decide on a move.

Connect4-AI tournament best of 5

#	Player One Bot	Player Two Bot	Board Size	Winner
1	BFS	Greedy	6 x 7	1-4 Greedy
2	BFS	Minimax	6 x 7	0-5 Minimax
3	Greedy	Minimax	6 x 7	0-5 Minimax
4	Greedy	Minimax	8 x 10	0-5 Minimax
5	BFS	Greedy	8 x 10	4-1 BFS
6	BFS	Minimax	8 x 10	0-5 Minimax

From this table of our experimental results, we can see that the Minimax algorithm was able to win every game against the other bots. The reason for this is because the Minimax bot utilizes a decision process unlike the other two bots. The Minimax bot is able to see all potential game states at the depth of 8 moves into the future of the game. This allows the bot to pick the

best move that will result in the highest advantage further along the game's progress. However at a downside, this bot would take the longest time to decide on a move. The larger the size of the board, the longer the Minimax bot would take to pick its best move. During one instance of the game with a board size of 6 x 7, the minimax bot could beat the Greedy bot with a combined time of 4.63 seconds which is a considerable amount when compared to the Greedy bot's 0.03 seconds. The bot with the second highest win rate would depend on the size of the board. We found this to be an interesting find. With a standard board size of 6 x 7, our Greedy bot would beat out the BFS 4 out of 5 times. When the board size is increased to 8 x 10, the inverse results were noted. The BFS bot would beat the Greedy bot 4 out of 5 times. Both of these bots would have a decision process of a nearly identical time. This is because neither bots consider future moves, only making a decision from the current game state. The reason for the BFS bot to win in larger board instances is because the Greedy Bot is more concerned with connecting four game pieces than stopping its opponent. This allows the BFS bot to find a place on the board where the Greedy bot is not trying to connect pieces and win before the Greedy bot can connect four pieces together.

The results obtained from the Connect4 tournament involving AI bots provided valuable insights into the performance of each bot, enabling us to assess their relative strengths and weaknesses. By evaluating these results, we were able to determine which AI bot was better at solving the Connect4 problem. The following discussion outlines how the tournament results reflect our ability to solve this problem.

1. Objective Performance Metrics: The metrics used to assess the AI bots' performance, such as win rate, average game length, win rate by board size, and cumulative win rate, allowed us to quantitatively compare the bots. These metrics provided a clear understanding of each bot's effectiveness in winning games, efficiency in making moves, and adaptability to different board sizes. By examining these metrics, we could identify the superior bot in terms of overall performance and specific aspects of gameplay.
2. Consistency of Performance: The tournament format ensured that each AI bot played against every other bot on both board sizes. This comprehensive approach allowed us to evaluate the bots' performance consistency across different opponents and playing conditions. A bot that consistently performed well against various opponents and across board sizes demonstrated its superior problem-solving ability in the Connect4 game.
3. Analysis of Game Strategies: By reviewing the AI bots' moves and strategies throughout the tournament, we could gain insights into their decision-making processes and tactical approaches. Analyzing the effectiveness of these strategies in various game scenarios helped us identify which bot was better at adapting and responding to the challenges posed by different opponents and board configurations.

Conclusion:

In this report, we have presented the design, implementation, and evaluation of a Connect Four AI tournament involving three AI bots. Through the tournament format and the use of performance metrics, consistency evaluation, and strategic analysis, we successfully determine the relative performance of each AI bot in the Connect Four game. As a result, we were able to identify the superior bot in terms of its problem-solving ability and overall effectiveness.

The Key Lessons Learned were:

1. Importance of diverse evaluation: The comprehensive evaluation methodology, which included various performance metrics, highlighted the importance of considering multiple aspects of AI bot performance to determine their effectiveness in solving the Connect Four problem.
2. Adaptability to board size: Analyzing the AI bots' performance across different board sizes provided valuable insights into their ability to adapt to varying game conditions and challenges.
3. Impact of strategic decision-making: A close examination of the AI bots' game strategies and decision-making processes revealed the significance of effective tactics and adaptability in achieving success in the Connect Four game.
4. Our Superior bot: After comparing the performance and consistency of our bots we can say with confidence that the MiniMax bot was the best from our bot pool. This bot was able to win every game although it did take the most time during its turn. This bot was superior because of its ability to plan out moves based on potential future moves in the game state. The bot was able to determine how the game will progress five moves in the future and this allowed the bot to make the most logical move. For this reason, the MiniMax bot was superior to the BFS bot and Greedy bot.

Future Work / Next Steps:

1. Enhancing AI algorithms: The results from the tournament provide a solid foundation for improving the AI bots' algorithms and strategies. Future work could focus on refining the AI models based on the lessons learned, such as improving adaptability to different board sizes and optimizing strategic decision-making.
2. Expanding the tournament: To further validate the conclusions drawn from this study, future work could involve organizing larger-scale tournaments with more AI bots, varying levels of difficulty, and a greater number of games.
3. Incorporating player feedback: Integrating human player feedback into the evaluation process could provide additional insights into the AI bots' performance from a user perspective, ultimately contributing to the development of more engaging and challenging AI opponents.
4. Exploring new AI techniques: As AI research continues to evolve, future work could investigate the potential of emerging techniques, such as reinforcement learning and neural networks, in enhancing the AI bots' performance in Connect Four gameplay

In conclusion, the results of the Connect Four AI tournament enabled us to effectively determine the superior bot by utilizing a combination of performance metrics, consistency evaluation, and strategic analysis. This comprehensive and accurate assessment of the AI bots' abilities provided a solid foundation for our findings. Building upon these insights and lessons learned, future work can continue to refine and improve AI bots' performance in Connect Four, ultimately contributing to the development of more sophisticated and engaging AI opponents for players to challenge.

References:

[1] K. Galli. “Connect4-Python: Connect 4 programmed in python using pygame.” Github. https://github.com/KeithGalli/Connect4-Python.

[2] M. Arambakam. “connect-4: A multi-agent implementation of the game Connect-4 using MCTS, Minimax and Exptimax algorithms.” Github. https://github.com/mukeshmk/connect-4.

Appendix:

Corey Lang:

Key Contributions:

1. Development of the BFS bot: I was responsible for designing and implementing the Breadth-First Search (BFS) algorithm for one of the AI bots. This involved understanding the BFS approach and adapting it to suit the Connect Four game environment.
2. Research and presentation of the Minimax bot: I conducted extensive research on the Minimax algorithm and its application to the Connect Four game. I presented the findings and proposed the implementation of the Minimax bot as a viable AI opponent in our project.
3. Presenting the PEAS for this AI environment: I analyzed and described the Performance measure, Environment, Actuators, and Sensors (PEAS) components of the AI environment for our Connect4 bots, providing a clear understanding of the agents' roles and interactions within the game.
4. Integration of the wait function for better GUI visualization: I played a significant role in incorporating the wait function into the AI bots' decision-making process. This function improved the graphical user interface (GUI) visualization by ensuring that the move timer remained unaffected by the bots' computational time and allowed the pygames GUI to not show moves the bots made.
5. Report sections: I contributed to the writing of the Related Works, Evaluation and Results, and Conclusion sections of the report. My work involved researching relevant literature, analyzing the results of the tournament, and summarizing the project's outcomes and future directions.

Personal Lessons Learned:

1. Importance of diverse algorithms: Working on this project highlighted the value of understanding and implementing various AI algorithms to address different challenges and optimize performance in a game environment.
2. Effective communication and presentation: Presenting the research findings on the Minimax bot and the PEAS components reinforced the importance of conveying complex information in a clear and concise manner to ensure understanding among team members.
3. Balancing theory and practice: Conducting research and implementing the algorithms in the Connect Four game environment taught me the importance of balancing theoretical knowledge with practical application to develop effective AI solutions.
4. Time management and organization: Contributing to various sections of the report and coordinating with team members highlighted the need for effective time management and organizational skills to ensure a successful project outcome.

Jenna Mauch:

Key Contributions:

1. Development of the Greedy Search algorithm: I was responsible for implementing the Greedy Search code to be used in the game. I was involved in understanding how the heuristic worked.
2. Creation of the demonstration video: I edited together the demonstration video used in the presentation slides. The video clearly shows which algorithms are being put against each other, and allowed us to explain the process or algorithms as it went along during the presentation.
3. Presentation of the Greedy Search algorithm: Because I worked on the Greedy algorithm, I presented it.
4. Report sections: I contributed to the writing of the Introduction, Approach, and References. Along with the written work, I also contributed to creating the flowchart and the table included in the Approach section.
5. Cleaning up the repository: I organized the repository and made sure the deliverables were included. Much of our source code was in the main directory so I manually moved everything to the Source Code directory.

Personal Lessons Learned:

1. Pygame: This project allowed me to learn the basics of coding a game in Python. It also made me more confident in my programming abilities with Python. I have never done a project involved with programming a game before so this was a helpful experience for me.
2. Research: Going into this project, the thought of others making game-playing agents for Connect Four hadn't crossed my mind. However, finding multiple repositories including the one we used the GUI for surprised me how others have thought of and done the same. From this, I realized how easy it can be to start learning something new as long

as the right research is put into place. From this lesson, I hope to be able to learn projects on my own with others' works as guidance.

3. Adaptability: As roadblocks came up in our project, there were times we had to change the project from how the original proposal was written up. From this, I learned that it's best to keep an open-mind to alternatives, and to deviate from plans if things do not work out as planned.