AI Snakes

Deon Lillo, Mike Moschitto, Cagan Sevencan, Kyle Thompson, Chase VanderZwan, Adley Wong
Cal Poly,
San Luis Obispo, United States
{dlillo, mmoschit, csevenca, rkthomps, vanderzw, awong189}@calpoly.edu

Abstract—**Snakes, along with other retro games, has been used as a testbed for artificial intelligence (AI) techniques. There are existing agents that play Snakes using minimax, alpha-beta pruning, classical reinforcement learning (RL), and deep reinforcement learning (DRL). However, we have not identified a Snakes agent that uses monte carlo tree search (MCTS). We will create such an agent, and evaluate it against the winner of the CoG AI Snakes competition in 2021.**

I. Introduction

Games provide a simple environment to implement and compare techniques in AI. Many of the major breakthroughs in AI involve agents' performance in games. For example, Mnih et al. [1] is a seminal work in DRL that integrates deep learning and RL to create an agent that surpasses human level performance on three games in the Atari Arcade Learning Environment. In fact, Mnih et al. [1] work was a motivating factor for creating the CoG AI Snakes competition [2]. The founders of the competition noted the resurgence of retro games in AI and created an educational testbed that students and researchers could use to compare agents that play Snakes.

Though agents have been implemented for Snakes using a variety of AI techniques, we have not found one that uses MCTS [3]. We were surprised that we could not find any agents that used MCTS particularly because it plays an integral role in AlphaGo [4], and AlphaGo-Zero [5] which show superhuman performance in Go with and without the supervision of expert games respectively. Therefore, we propose to implement MCTS for Snakes, and compare the results to Serpentine [6], the winner of the 2021 CoG AI Snakes competition [2]. This project has strong ties to topics in CSC-580. To put our project in terms used in class, our goal is to make a rational agent that plays Snakes using MCTS. Our group plans to achieve a deep understanding of MCTS, and, if time permits, a deep understanding of using deep learning to improve MCTS as is done in [5].

Snakes has been shown to be amenable to many different AI techniques. For example, in Wang et al. [7] proposed a DRL agent for a three versus three version of snakes where agents are rewarded for both collaboration and competition. Buttner et al. [8] proposed a RL-based whose policy takes the form of a lookup table, and is refined by a genetic algorithm through self-play. Furthermore, Almalki et al. [9] implemented a DRL-based agent to play a variation of snakes that has poisoned apples. Also, Wei et al. [10] proposed an agent that plays snake without knowledge of the internal game state. Instead, their agent models the game state, and monitors its reward by observing the screen's pixels.

There are three distinct versions of the Snakes game used in the four approaches mentioned above. We will focus on the version specified in the AI CoG Snakes competition [2]. This version entails two snakes, and a single randomly placed apple. A snake's length grows by one unit whenever it eats the apple. A snake can beat another snake if it is longer than the other snake after 3 minutes, or if it survives longer than the other snake. The rules are further defined as follows:

- Snakes can move up, down, left, or right.
- A snake loses if it leaves the board, hits its own body, hits the body of another snake, or takes more than one second to make a decision.
- If the two snakes collide head-to-head, the longest snake wins.
- There always exists one apple in a random location on the board. If the apple is uneaten after ten seconds, it changes locations.

We will use MCTS to create an agent that plays this version of snakes. We will take advantage of the game engine supplied by the 2021 AI CoG Snakes competition. We already have a running version of the game, and have ported the logic of the game to python in case we would like to leverage popular deep learning libraries like TensorFlow [11].

II. Related Works

As discussed prior, competitors in the CoG Snakes competition made use of many techniques to play and win the competitions. The techniques that were used were minimax plus alpha beta pruning, classical reinforcement learning, and deep reinforcement learning.

In Buttner et al. [8], they developed a traditional reinforcement learning algorithm which used self play to learn strategies. Reinforcement learning requires the usage of reward functions to help each iteration of the algorithm improve itself. Their training regime were 2500 learning games and 100 test games where each rule set would play against the previous best. Any rule set that won 55% of the time replaces the old rule set. This would be done for 5 iterations. However, Bettner et al. draws a conclusion that this form of training has various flaws, such as being able to back itself into a losing position, drawing, and sometimes exiting the arena.

Wang et al. [7] used a deep reinforcement learning algorithm to play in teams of three. It introduced a rule enhanced multi-agent reinforcement learning algorithm which used territory matrix which helped state features and mask illegal actions. Additionally, they improved the reward function to reward individual team relationships and friends foes recognition.

In last year's competition, an algorithm named Serpentine [6] was able to beat every other competitor in the competition. It used a unique strategy to defeat all other snakes by actively seeking and circling the apple. This prevented the other snakes from being able to grow larger while also making them more likely to cause a head on collision or hitting the body of their snake. Serpentine [6] made use of minimax, which is a depth first search tree search algorithm, along with alpha-beta pruning to reduce the search space by not searching moves that have a worse outcome than another search space.

Additionally, this is just one type of AI snakes game. There are other AI snake games with a different set of rules and ways to play. One other competition that we found is snakes [12]. This competition has a different way of competing. Rather through a programmed bot as a Java class, the snake bots interact with a web API to perform actions and view its environment. Additionally, there can be more than two snakes competing and a more comprehensive list of bots to play against.

III. METHODS

We propose to use MCTS [3] to play the snake game. We note that each tile in the 14×14 board can have just one of 6 states:

- Head of snake 1.
- Head of snake 2.
- Body of snake 1.
- Body of snake 2.
- Apple.
- Empty.

As the selection policy of our search we will use the Upper Confidence Bounds applied to trees (UCT) [3]. This policy is proven to have a good balance between exploration and exploitation.

Because the game enforces a one second time constraint on how long an agent can take to move, we can set the maximum number of moves allowed in a game to 180 as games are a maximum of 180 seconds long. Therefore, our search tree will have a depth of no more than 180. There is some complication in our search since actions are not necessarily deterministic. The two snakes move simultaneously, and without knowledge of which move the other snake will take. However, this complication does not prohibit MCTS. We can simply choose the action that is best in its worst case as is done in the minimax algorithm. Therefore, our tree will have a branching factor of $4^2 = 16$, where each edge in the tree is labeled by an action from both snakes. Both snakes will simultaneously use (UCT) in the tree search.

If time permits, we can adapt the method for AlphaZero [5] to refine the selection policy in our tree search. That is, given a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, and a policy function, $p : \mathcal{S} \longrightarrow [0,1]^{|A|}$, we can use p to strengthen our MCTS. Then, suppose that our MCTS yields the refined probabilities $\pi \in [0,1]^{|A|}$ at state $s \in \mathcal{S}$. We can then use the training example (s, π) to refine p via stochastic gradient descent.

IV. EVALUATION

Because we are following the CoG AI Snakes competition format, it will be simple for us to evaluate our MCTS agent against Serpentine [6], the winning agent in 2021. If we have time to extend our MTCS with deep learning, we can evaluate all three agents against each-other. Our evaluation between agents will include win-rate and ending length. We include ending length because it clues whether an agent wins by eating apples, or getting in the way of the other agent.

V. CONCLUSIONS

We propose an agent for playing the snakes game as described in the CoG Snakes Competition [2]. Unlike any agents we have seen, ours will use MCTS to inform its actions. If time permits, we will extend our MCTS by following the reinforcement learning technique used in [5]. Finally, we will evaluate our agents against the winner of the 2021 CoG snakes competition, Serpentine [6].

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] J. A. Brown, L. J. P. de Araujo, and A. Grichshenko, "Snakes ai competition 2020 and 2021 report," *arXiv preprint arXiv:2108.05136*, 2021.

[3] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[4] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[6] B. Grooten, I. Schilstra, W. Hert, and D. Genuchten, "Serpentine," 2020, https://serpentine.ai/wp-content/uploads/2020/10/Serpentine-AISnakes-paper.pdf.

[7] J. Wang, D. Xue, J. Zhao, W. Zhou, and H. Li, "Mastering the Game of 3v3 Snakes with Rule-Enhanced Multi-Agent Reinforcement Learning," Aug. 2022, pp. 229–236, iSSN: 2325-4289.

[8] J. Büttner and S. Von Mammen, "Training a Reinforcement Learning Agent based on XCS in a Competitive Snake Environment," Aug. 2021, pp. 1–5, iSSN: 2325-4289.

[9] A. J. Almalki and P. Wocjan, "Exploration of Reinforcement Learning to Play Snake Game," in *2019 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2019, pp. 377–381.

[10] Z. Wei, D. Wang, M. Zhang, A.-H. Tan, C. Miao, and Y. Zhou, "Autonomous Agents in Snake Game via Deep Reinforcement Learning," in *2018 IEEE International Conference on Agents (ICA)*, Jul. 2018, pp. 20–25.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[12] "Building an AI-powered Battlesnake with reinforcement learning on Amazon SageMaker | AWS Machine Learning Blog," Mar. 2020, section: Amazon SageMaker. [Online]. Available: https://aws.amazon.com/blogs/machine-learning/building-an-ai-powered-battlesnake-with-reinforcement-learning-on-amazon-sagemaker/