AlphaSnake: Policy Iteration on a Nondeterministic NP-Hard Markov Decision Process (Student Abstract)

Kevin Du¹, Ian Gemp², Yi Wu³, Yingying Wu⁴,⁵,*
¹ Harvard University, Cambridge, U.S.
² DeepMind, London, U.K.
³ Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
⁴ Center of Mathematical Sciences and Applications, Harvard University, Cambridge, U.S.
⁵ University of Houston, Department of Mathematics, Houston, U.S.
ywu68@uh.edu

Abstract

Reinforcement learning has been used to approach well-known NP-hard combinatorial problems in graph theory. Among these, Hamiltonian cycle problems are exceptionally difficult to analyze, even when restricted to individual instances of structurally complex graphs. In this paper, we use Monte Carlo Tree Search (MCTS), the search algorithm behind many state-of-the-art reinforcement learning algorithms such as AlphaZero, to create autonomous agents that learn to play the game of Snake, a game centered on properties of Hamiltonian cycles on grid graphs. The game of Snake can be formulated as a single-player discounted Markov Decision Process (MDP), where the agent must behave optimally in a stochastic environment. Determining the optimal policy for Snake, defined as the policy that maximizes the probability of winning — or win rate — with higher priority and minimizes the expected number of time steps to win with lower priority, is conjectured to be NP-hard. Compared to prior work in the Snake game, our algorithm is the first to achieve a win rate over 0.5, compared with a uniform random policy, which achieves a win rate < 4.16 × 10⁻¹⁷, computed with a dynamic programming paradigm.

The Snake Environment

The game of Snake is played on an \( n \times n \) board, where the player controls the movement of the snake head, moving one unit in three of the four orthogonal directions every time step. The snake body follows the head’s movements and grows by one unit when the snake eats an apple, increasing the score of the player by one. Once the apple is eaten, another apple is placed in an empty cell uniformly at random. The snake head cannot intersect the snake body, and if the head is surrounded by the body, the game is lost. Starting from a snake size of 2 units, the goal of the game is to maximize the length of the snake; the game is won when the snake body fills the grid, defining a Hamiltonian path on the \( n \times n \) grid graph. Minimizing the win time, or number of time steps required to achieve the maximum snake size, is a secondary objective. Figure 1 shows various example game states.

![Figure 1: Game states from left to right: example (1) starting state (2) intermediate state (3) losing state (4) winning state.](image)

Due to stochasticity in the environment, Snake can be formulated as a nondeterministic MDP, in which finding the optimal policy requires the agent to maximize its expected rewards. The rewards used in this project are +1 for eating an apple, −10 for a losing end state, and +10 for a winning end state. These rewards were chosen arbitrarily, but we found experimentally that this scheme was effective.

Deterministic Algorithms and Complexity

There exists a straightforward strategy that is guaranteed to win Snake with a worst-case win time of \( \Theta(n^4) \) (See...
Table 1: Average score and win rate, as fraction of wins within 1,200 steps.

| Strategy                   | Average score | Win rate |
|----------------------------|---------------|----------|
| AlphaZero algorithm        | 98.227        | 944/1000 |
| Uniform random policy      | 6.277         | 0/1000   |
| Hamiltonian cycle strategy | 30.026        | 0/1000   |
| Naive tree search          | 59.09         | 0/1000   |

The policy network is trained with cross-entropy towards the state-action counts of tree search (See Supp.).

**Experiments**

In our experiments, we ran the above algorithm on a $10 \times 10$ game, using a tree search size of 200 states for 6,000 games. The resulting performance of the algorithm is seen in Figure 2. Table 1 compares this performance with several other algorithms for Snake: a random policy, the Hamiltonian cycle strategy, and a naive tree search (without policy or value predictions) with a search size of 10,000 states.

**Conclusion**

Our work shows that AlphaZero is resilient in performing in a nondeterministic game, developing strategies that are successful irrespective of chance. The game of Snake may be a relevant model for real-world security systems and anti-poaching strategies that often require protecting targets from adversary response (De Nittis and Trovo 2016). Generalizing AlphaZero to stochastic multiagent environments appears to be promising for tackling real-world situations.

**Acknowledgments**

We would like to thank Clifford Taubes, Hui Jiang, Richard Peng, and Roger Fu for helpful discussions.

**References**

Abe, K.; Xu, Z.; Sato, I.; and Sugiyama, M. 2019. Solving np-hard problems on graphs with extended alphago zero. arXiv preprint arXiv:1905.11623.

De Nittis, G.; and Trovo, F. 2016. Machine learning techniques for stackelberg security games: a survey. arXiv preprint arXiv:1609.09341.

Haythorpe, M. 2018. FHCP challenge set: The first set of structurally difficult instances of the hamiltonian cycle problem. Bulletin of the ICA, 83: 98–107.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, 362(6419): 1140–1144.