Mihai Lache

Report

In your report you should answer the following questions:

Briefly describe how you implemented MCTS. Which selection strategy do you use?

In order to implement MCTS I set up the Treenode class that implemented selection,expansion, rollout, and backpropogation. In select i checked for a terminal state, then backpropogate immediately if the battle ended. If not all actions have child nodes, I call expand to grow 1 new child. Otherwise I used the UCB-1 formula in order to pick which child needs to be explored next. In expand, I chose one untried action, then rollout from that new node. This basically picks legal actions until the battle ends and returns the final damage fraction. Finally the backpropogate goes up throughout all ancestors.

How does your agent perform in the various scenarios? How does it compare to the Sampling Bot? Do not include results as screenshots of e.g. console output; use a table or list with actual text.

I ran both agents for 20 games at 50 iterations each on the three core scenarios

- In the giant test both the sampling and my bot won every game — (100%)
- In the offerings test the sampling won 80%, while my bot won 85%
- In the low HP test tthe sampling won a 60% win rate but my bot won 70%.

Which parameter values (number of iterations, value for the constant "c" if you are using UCB-1, ε if you use the ε-greedy strategy)

I ran everything with 50 iterations per turn and set c to 0.5. If I set it any higher then the bot got too random, and if I set it too low it got stuck repeating the same lines.

Did you encounter any particular challenges during the assignment?

One of the main challenges was getting consistent results in the lowhp scenario. The agent would sometimes repeat bad moves or get stuck on certain actions. Another problem was tuning how the expansion should be behaving, especially if a node had many children. Picking the right balance helped avoid wasting rollouts on weak actions too early. This really mattered for the lowhp scenario, where surviving an extra turn could be more important than dealing damage immediately.

Restrospective

Mihai Lache

I handled the entire implementation of the MCTS agent, including the selection logic using UCB-1, the expansion and rollout functions, and making sure results were being tracked and backpropagated correctly. I also tuned the c parameter and iteration count based on scenario performance. I learned was how sensitive MCTS can be to small changes. Overall, I got a better understanding of how a clean algorithm like MCTS still needs a lot of tuning to work well in a real game, and I learned a lot.

I used ChatGPT during debugging to help understand why certain parts weren't working as expected.