Questionário do Projeto Connect4

Antonio Fuziy

Fabrício Barth

Agentes Autônomos e Reinforcement Learning

1. Que algoritmo deve ser utilizado para desenvolver um agente jogador de *Connect4 PopOut* vencedor? Deve-se utilizar uma implementação de Min-Max com poda alpha-beta? Se sim, qual a profundidade que deverá ser utilizada para evitar processamentos superiores a 10 segundos por jogada? Qual a função de utilidade que deve ser utilizada?

   Acredito que para esse projeto seja possível utilizar o Min-Max normal, porém o Min-Max alpha-beta apresenta-se mais eficiente, de forma que ele identifica o caminho da árvore que não terá um bom resultado no final, dessa forma ele realiza a poda para ignorar esse caminho e escolher outro melhor para tomar a melhor decisão. Por fim, a poda alpha-beta apresenta mais benefícios para esse projeto, visto que ela reduz o número de nós gerados pela árvore, ao utilizar essa otimização, o processamento do algoritmo torna-se menor, sendo assim eficiente para o limite de tempo de 10 segundos estabelecido no projeto.

   Para o caso do agente FuziyPlayer a profundidade 5 foi testada e obteve um processamento mais razoável para o tempo de 10 segundos de término do jogo. A função utilidade foi explicada mais a fundo na questão 2, porém a ideia principal é fazer o agente atacar ou defender com jogadas ofensivas ou defensivas baseadas no estado atual do jogo, tentando realizar bloqueios para permanecer vivo no jogo ou realizar uma jogada que acabe com o jogo.

2. O seu jogador faz uso de alguma base de conhecimento? Se sim, como ela é utilizada durante o processo de tomada de decisão?

   Sim, o jogador implementado utiliza uma inteligência de verificação que observa diagonal, vertical e horizontal, observando se podem vir a existir 1, 2, 3 ou 4 moedas no próximo estado do jogo, e assim o jogador utiliza uma função que estabelece pesos sobre as possibilidades de movimentações e baseado nela ele toma a decisão do próximo movimento. Além disso, o algoritmo do jogador observa situações de possível fim de jogo tanto para si mesmo quanto para o oponente, dessa forma em casos em que existe uma possibilidade de vitória do oponente, o agente bloqueia essa possibilidade como forma de jogada defensiva, ao passo em que os casos onde existe a forma de finalizar o jogo com vitória, o agente olha pra essa possibilidade e fecha o jogo, porém isso muda de acordo com a ordem que se implementa essas duas formas de jogo (defensivo ou ofensivo), caso a chamada do método ofensivo apareça primeiro, o agente utilizará de jogadas focadas na vitória nos casos de possibilidade de fim de jogo e no caso da chamada do método defensivo como preferência, o jogador fica mais focado em jogadas de defesa nos casos de fim de jogo.

3. Foi utilizada alguma função de utilidade não definida manualmente, por exemplo, alguma função de utilidade gerada a partir de um processo de aprendizagem de máquina supervisionado? Se sim, como é que foi o treinamento desta função de utilidade? Como foi feita a integração desta função de utilidade com o restante do código?

   Não utilizou-se aprendizado de máquina para a implementação do projeto.

4. Qual a sua expectativa com relação ao desempenho do seu agente? Você acredita que ele irá desempenhar bem na competição? Por que? Você executou testes contra outros jogadores? Quais foram os resultados?

Acredito que o meu agente pode conseguir um bom desempenho na competição, uma vez que o algoritmo utilizado na implementação tem foco em atacar e defender-se do oponente, de forma que em certas situações do jogo, o agente bloqueia uma possibilidade de vitória do usuário em várias posições, desde diagonal, vertical e horizontal, tanto nas posições do meio e do final da sequência de 4 moedas. Além disso, foi testado o algoritmo contra o manual player, random player e barth player e em todos os testes até agora resultaram em vitória do meu agente. Vale ressaltar, que o teste com o RandomPlayer foi realizado 200 vezes, sendo 100 o RandomPlayer começando e 100 o FuziyPlayer começando e os resultados obtidos foram razoáveis.

5. Quais foram as principais referências utilizadas para a implementação do seu jogador?
   Abaixo estão as principais referências para a implementação do agente:

   Referência sobre como implementar um agente de connect4 e exemplos de connect4 comum:

   - http://fbarth.net.br/Connect4-Python/
   - https://github.com/fbarth/Connect4-Python/tree/master/src

   Referência utilizada sobre teoria de MinMax:

   - http://fbarth.net.br/agents/slides/04_busca_competitiva/buscaCompetitiva.pdf

   Referência para regras e quais as obrigações de implementações:

   - http://fbarth.net.br/agents/code/games/

   Referência de implementação:

- https://medium.com/analytics-vidhya/artificial-intelligence-at-play-connect-four-minimax-algorithm-explained-3b5fc32e4a4f

6. Existem diferenças significativas entre um jogador de *Connect4* e um jogador de *Connect4 PopOut* em termos de árvore de busca e função de avaliação? É possível utilizar o jogador implementado para o *Connect4 PopOut* em competições de *Connect4* sem muitas modificações?

   Acredito que não existe tanta diferença na implementação, visto que a única diferença é a possibilidade de remover uma moeda da última linha, dessa forma a diferença mais a fundo acredito que seria em verificar a última linha do jogo e prever o resultado da remoção da moeda na função de avaliação e assim gerando sucessores diferentes do Connect4 comum e mudando os pesos sobre a escolha dos movimentos, porém os movimentos de remoção seriam muito menos recorrentes que os movimentos de inserção de moedas. Portanto, com as considerações anteriores pode-se dizer que um jogador de Connect4 PopOut não precisa de tantas alterações no algoritmo para se adaptar a um jogo de Connect4 comum.