Universidade Estadual de Maringá
Centro de Tecnologia
Departamento de Informática

1ª Trabalho de Introdução à Inteligência Artificial

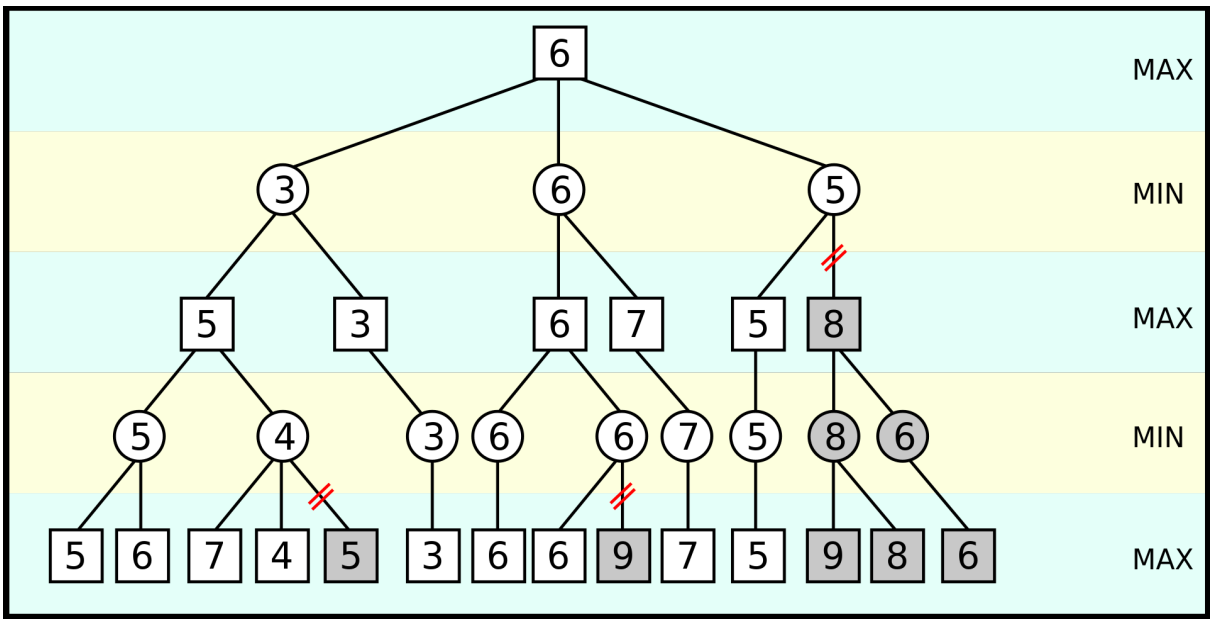
Nome: Vinícius Menossi
RA: 108840
Curso: Ciência da Computação
Professor: Wagner Igarashi

Introdução

Desenvolver atividades que simulem a inteligência, tem sido um dos grandes desafios e avanços contemporâneos, uma das frentes no desenvolvimento de Inteligência Artificial é a implementação de agentes inteligentes que são capazes de jogar jogos que por muito tempo apenas humanos conseguiam, alguns deles são: Damas, Gamão, Go, Xadrez. Na década de noventa a empresa IBM conseguiu produzir um supercomputador software conhecido como Deep Blue, que foi capaz pela primeira vez de vencer um mini match de xadrez contra o campeão mundial de xadrez Garry Kasparov. Desde então as *engines* de xadrez tem cada vez mais evoluído sua performance. Este trabalho busca implementar de forma simplificada e fundamentalista um algoritmo capaz de jogar xadrez, visando o aprendizado dos conceitos de Inteligência Artificial

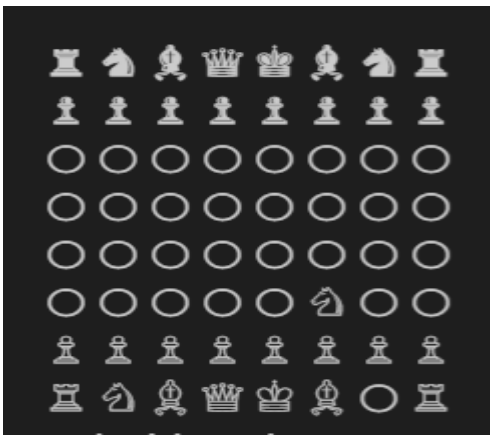
Fundamentação Teórica

Para a implementação do software foi utilizado o algoritmo poda Alfa-Beta, que consiste de um algoritmo de busca que visa diminuir o número de nós que são avaliados pelo algoritmo minimax em sua árvore de busca.
É um algoritmo de busca adversarial, é feita a poda ou seja uma parada na busca de um determinado nó quando ao menos uma possibilidade foi encontrada que prova que o movimento é pior do que um movimento examinado anteriormente. O algoritmo mantém dois valores, alfa e beta, que representam respectivamente a pontuação mínima garantida ao jogador que está maximizando e a pontuação máxima garantida ao jogador que está minimizando. Sempre que a pontuação máxima garantida ao jogador minimizador (beta), torna-se menor do que a pontuação mínima garantida pelo jogador que está maximizando (alfa), o jogador não precisa considerar mais descendentes deste nó, pois eles nunca serão alcançados na busca.



Na ilustração da poda alfa-beta os níveis máximo e mínimo representam a vez do jogador e do adversário, respectivamente e as subárvores acinzentadas não precisam ser exploradas.

Para a implementação foi utilizada a linguagem Python, e a biblioteca Python Chess, os algoritmos para simular agentes inteligentes foram: Poda alfa-beta, e dois algoritmos aleatórios. Um dos algoritmos de escolha aleatória foi chamado de Cheques-Capturas-Ataques(CCA), este tenta prioritariamente fazer jogadas na dita ordem, caso não haja cheques nem capturas nem ataques é feito um lance aleatório entre os demais, essa estratégia foi escolhida por ser conhecida como uma ordem de prioridade de jogadas a serem consideradas na hora da análise da posição por enxadristas. Já o outro algoritmo chamado de Minimizador de Movimentos consiste em olhar para os movimentos que mais restringem os números de movimentos do adversário, esta estratégia foi escolhida, pois em alguns dos fundamentos do xadrez está a tentativa de manter a simplicidade para que haja poucos cálculos, outro fator é que existem posições no xadrez chamadas de zugzwang que consistem em uma posição em que o jogador fica sem movimentos “bons” para serem jogados, ou seja, há uma sufocação nas escolhas de movimentos. Assim o algoritmo tenta se aproximar heurísticamente dessas possibilidades.
Para a utilização do algoritmo de poda Alfa-Beta é necessário a utilização de uma heurística, assim foi feita uma função para avaliação da posição, sendo ela: a avaliação material, ou seja, a quantidade de peças no tabuleiro, a avaliação posicional que foi divida em duas fases a de inicio-meio de jogo e a avaliação de final de jogo, foram criadas tabelas onde dado uma devida peça ela é avaliada de acordo com sua posição no tabuleiro
Foi feita a implementação de uma ordenação dos movimentos possíveis antes da realização da busca, assim potencialmente minimizando custos computacionais, visto que se as possíveis posições mais relevantes forem avaliadas antes há uma grande chance de que haja mais cortes na poda Alfa-Beta, essa ordenação foi feita de maneira que primeiro são verificados os lances que produzem cheques, depois os lances que produzem capturas e os outros demais lances.
Uma melhoria do algoritmo de poda Alfa-Beta é a função de busca *Quiescence*, esta função tem o propósito de evitar um fenômeno conhecido como problema do horizonte, onde em algumas jogadas por não considerar profundidades seguintes são feitos lances criticamente ruins, assim a função *Quiescence* busca continuar a busca mesmo após a profundidade limite ter sido atingida, mas apenas para movimentos considerados criticamente relevantes, neste caso os movimentos considerados foram cheques e capturas, a busca Quiescence apenas se encerra quando não encontra mais esses lances relevantes.
Uma última melhoria para o algoritmo poda Alfa-Beta foi a implementação de uma tabela de transposição, consistindo de uma tabela Hash com a chave única sendo essa a notação FEN da posição, a tabela armazena o valor de avaliação da posição, assim evitando cálculos desnecessários quando uma posição acaba sendo atingida repetidamente.
A interface consiste do usuário digitar uma opção através do terminal se deseja jogar contra a máquina ou simular uma partida entre os algoritmos, caso selecione jogar contra a máquina basta digitar a casa da peça que seja mover para a casa que deseja que ela seja movida,
por exemplo ‘g1f3’:



Assim, se for um movimento legal, é realizado e mostrado na tela, neste caso o calado salta de g1 para f3.

Experimentos:

	Alfa-Beta prof1 brancas	Alfa-Beta prof2 brancas	Alfa-Beta prof3 brancas	Alfa-Beta prof4 brancas	CCA brancas	MinMov brancas
Alfa-Beta prof1 pretas	½ - ½	1 - 0	1 - 0	1 - 0	0 - 1	0 - 1
Alfa-Beta prof2 pretas	0 - 1	1 - 0	1 - 0	1 - 0	0 - 1	0 - 1
Alfa-Beta prof3 pretas	0 - 1	0 - 1	1 - 0	1 - 0	0 - 1	0 - 1
Alfa-Beta prof4 pretas	0 - 1	0 - 1	0 - 1	1 - 0	0 - 1	0 - 1
CCA pretas	1 - 0	1 - 0	1 - 0	1 - 0	½ - ½	½ - ½
MinMov	1 - 0	1 - 0	1 - 0	1 - 0	½ - ½	½ - ½

Para jogadores Iniciantes a IA performou bem, porém pela profundidade pequena não conseguiu superar jogadores de nível intermediário.
As possíveis melhorias a serem feitas, são principalmente no quesito da performance de nós por segundo, e nas heurísticas, podendo deixar elas mais sofisticadas como por exemplo avaliar estrutura de peões, mobilidade das peças etc.

Bibliografia:

Chessprogramming wiki
https://python-chess.readthedocs.io/en/latest/core.html
https://towardsdatascience.com/implementing-a-chess-engine-from-scratch-be38cbdae91
https://www.freecodecamp.org/news/simple-chess-ai-step-by-step-1d55a9266977/
https://medium.com/@SereneBiologist/the-anatomy-of-a-chess-ai-2087d0d565