Teoria dos Jogos em Computação - Trabalho Prático

Implementação de um algoritmo Alpha-Beta Pruning para jogar uma versão simplificada de xadrez

Arthur de Senna Rocha - 2013029866

***Resumo*—O objetivo dessa atividade é aplicar os conhecimentos adquiridos durante a disciplina de Teoria dos Jogos em Computação. Assim, será proposto um cenário não-trivial, o qual será descrito de forma informal e modelado formalmente em um jogo conforme os modelos vistos durante a disciplina. A atividade proposta para essa tarefa, portanto, compreende no estudo e modelagem de um algoritmo *alpha-beta pruning* para aplicação do mesmo em uma versão modificada de um jogo de xadrez.**

I. Introdução

Existem diversas formas de representação de jogos, a mais simples dela talvez seja a forma normal. No entanto, representações de jogos na forma normal não incorporam nenhuma noção de sequência, ou tempo, das ações dos jogadores. A forma extensa (ou árvore) é uma representação alternativa que torna explícita a estrutura temporal.

Uma forma de classificação de jogos na forma extensiva é entre jogos de informação perfeita e jogos de informação imperfeita. Informalmente, um jogo de informação perfeita em forma extensa (ou, mais simplesmente, um jogo de informação perfeita) é uma árvore no sentido da teoria dos grafos, na qual cada nó representa a escolha de um dos jogadores, cada borda representa uma ação possível, e as folhas representam resultados finais sobre os quais cada jogador tem uma função de utilidade. De fato, em certos círculos (em particular, em inteligência artificial), eles são conhecidos simplesmente como árvores de caça. Formalmente, nós os definimos da seguinte forma: [1]

- **Jogos de Informação perfeita:** *Um jogo (finito) de informação perfeita (em forma extensa) é uma tupla* $G = (N, A, H, Z, \chi, \rho, \sigma, u)$*, em que:*
  - N *é um conjunto de n jogadores;*
  - A *é um conjunto de ações;*
  - H *é um conjunto de nós de escolha não-terminais;*
  - Z *é um conjunto de nós de escolha terminais, disjuntos de* H*;*
  - $\chi : H \mapsto 2^A$ *é a função de ação, que atribui a cada nó de escolha um conjunto de ações possíveis;*
  - $\rho : H \mapsto N$ *é a função do jogador, que atribui a cada nó não-terminal um jogador* $i \in N$ *que escolhe uma ação nesse nó;*
  - $\sigma : H \times A \mapsto H \cup Z$ *é a função sucessora, que mapeia um nó de escolha e uma ação para um novo nó de escolha ou nó terminal, de modo que para todos os* $h_i, h_j \in H$ *e* $a_i, a_j \in A$ *se* $\sigma(h_i, a_i) = \sigma(h_j, a_j)$ *então* $h_i = h_j$ *e* $a_i = a_j$*;*
  - $u = (u_1, \cdots, u_n)$*, onde* $u_i : Z \mapsto R$ *é uma função de utilidade para o jogador i nos nós terminais Z*

A proposta deste trabalho trata da modelagem de um jogo de informação-perfeita na sua forma extensiva e, em seguida, da aplicação de um algoritmo *alpha-beta pruning* modificado para encontrar a melhor ação de um jogador dado o estado no qual o mesmo se encontra.

II. O Jogo

O jogo modelado trata-se de uma versão simplificada de um jogo de xadrez. Em um jogo de xadrez tradicional, dois jogadores (peças brancas e peças pretas), iniciam um jogo com um total de dezesseis peças cada de seis tipos diferentes, em um tabuleiro com 8x8 casas de cores alternadas. O jogo modelado nesse trabalho é composto de um tabuleiro 4x4, onde cada jogador possuí apenas três peças: um rei, uma torre e um bispo. A **Figura 1** mostra a posição inicial do jogo proposto.

Figura 1: Posição inicial do jogo

Além do tamanho do tabuleiro e número de peças, algumas regras presentes em um jogo de xadrez tradicional foram desconsideradas para garantir a simplicidade do modelo:

- Primeiramente, em um jogo de xadrez clássico, um jogador não pode realizar um lance que leve o próprio rei a xeque, ou que deixe seu rei em xeque. Essa regra foi desconsiderada e tais lances são absolutamente válidos na versão modificada;

- Como consequência da modificação anterior, outra modificação apresentou-se necessária: pelas regras do xadrez clássico, um jogo termina quando um jogador se encontra em uma posição de xeque e não existem lances válidos possíveis para o mesmo. Como posições que deixe o rei em xeque são válidas na versão modificada, nesse caso o jogo termina quando um jogador capturar o rei adversário;
- Para evitar que o jogo prossiga infinitamente, uma condição de empate foi estabelecida: caso o jogo se encontre em um estado em que ambos os jogadores possuam apenas as peças do rei, o jogo terminará após três lances e, caso nenhum jogador capture o rei adversário neste tempo, o jogo termina em empate;
- Por fim, cada peça possui os mesmos lances disponíveis em um jogo de xadrez clássico: torre se move horizontalmente, bispo diagonalmente e rei pode mover uma casa em qualquer direção e nenhuma peça pode terminar em uma casa onde uma peça da mesma cor está posicionada. No entanto, por questões de simplicidade, uma pequena modificação às regras tradicionais foi feita. Enquanto no jogo de xadrez somente o cavalo pode "pular"casas intermediárias entre sua casa de origem e destino que estejam ocupadas, essa habilidade foi replicada para todas as peças nessa nova versão do jogo (como o rei pode mover apenas uma casa em uma dada direção, essa regra é irrelevante para o mesmo). A **Figura 2** mostra um exemplo dess regra.

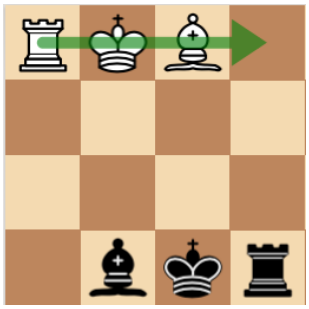

Figura 2: Movimento válido para a torre

Enquanto em um jogo tradicional o movimento mostrado na figura não fosse válido devido à existência de peças entre a posição inicial e final da torre, na versão modificada esse movimento torna-se possível.

A. *Modelagem na forma extensiva*

Como mencionado anteriormente, o jogo proposto será modelado como um jogo soma-zero de informação perfeita na forma extensiva. As regras de jogos na forma extensiva são definidas por posições legais (ou estados legais) e movimentos legais para todas as posições legais. Para cada posição legal, é possível determinar efetivamente todos os movimentos legais. Algumas posições legais são posições iniciais e outras estão terminando.

A melhor maneira de descrever esses termos é usar um gráfico em árvore cujos nós são posições legais e cujas bordas são movimentos legais. O gráfico é direcionado, pois não significa necessariamente que poderemos voltar exatamente de onde viemos na movimentação anterior. Por exemplo, no xadrez clássico, um peão só pode avançar. Este gráfico é chamado de árvore de jogo. Mover para baixo na árvore do jogo representa um dos jogadores que está fazendo uma jogada, e o estado do jogo muda de uma posição legal para outra.

A **Figura 3** a modelagem na forma extensiva do jogo proposto.

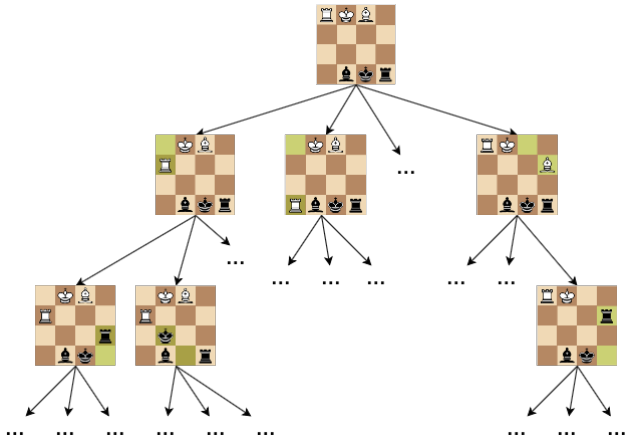

Figura 3: Modelagem simplificada do jogo proposto na forma extensiva

Na figura acima, cada vértice do gráfico representa um possível estado do jogo, enquanto cada aresta representa as possíveis ações a serem tomadas em um dado estado. Os níveis ímpares correspondem aos estados onde o jogador com as peças brancas deve tomar uma ação, enquanto nos níveis pares, o jogador com as peças pretas deve agir.

A árvore de jogo completa é uma árvore de jogo cuja raiz está na posição inicial e todas as folhas estão nas posições finais. Cada árvore de jogo completa tem tantos nós quanto o jogo tem resultados possíveis para cada jogada legal feita. É fácil notar que, para o jogo em questão, a árvore completa é enorme e o número de estados possíveis cresce exponencialmente a cada nível. Por esse motivo, não é uma boa prática criar explicitamente uma árvore de jogo inteira como uma estrutura ao escrever um programa que deve prever a melhor jogada a qualquer momento. No entanto, os nós devem ser criados implicitamente no processo de visita.

III. O Algoritmo

O objetivo dessa seção é a implementação de um algoritmo que encontre a melhor estratégia para o jogo modelado na seção anterior.

O algoritmo *Backpropagation* conta com pesquisa sistemática, ou mais precisamente, com força bruta e uma simples função de avaliação. Em um jogo de soma-zero de dois jogadores, um jogador busca minimizar o *payoff* final do jogo, enquanto o outro busca maximizar o mesmo. Para decidir a ação a ser tomada em um dado estado, o algoritmo em sua forma original realiza uma busca em uma árvore completa

até as folhas. Efetivamente, analisaríamos todos os resultados possíveis e sempre que seríamos capazes de determinar a melhor jogada possível.

No entanto, para jogos não triviais, essa prática é inaplicável. Mesmo a busca a uma certa profundidade, às vezes, leva um tempo inaceitável. O algoritmo *Alpha-Beta Pruning* otimiza esse processo eliminando a busca em caminhos desnecessários e reduzindo consideravelmente o tempo de execução do algoritmo. Mesmo assim, para jogos como o xadrez, a busca na árvore completa é inviável.

Para solucionar esse problema, o algoritmo utilizado aplica a pesquisa a uma profundidade de árvore razoavelmente baixa, auxiliada por heurísticas apropriadas e a uma função de avaliação bem projetada, porém simples. Com essa abordagem, perdemos a certeza de encontrar a melhor jogada possível, mas na maioria dos casos a decisão tomada pelo algoritmo é muito melhor do que qualquer ser humano.

A heurística aplicada é a seguinte:

- À cada peça do jogo é atribuído um valor, sendo valores positivos para as peças brancas, e valores negativos equivalentes para as peças pretas;
- Em cada estado do jogo, o *payoff* é calculado de acordo com soma dos valores das peças ainda presentes no tabuleiro;
- Os valores definidos para cada peça são:
  - **Bisbo**: $\pm$ 3
  - **Torre**: $\pm$ 5
  - **Rei**: $\pm$ 10000

O algoritmo implementado portanto, aplica a heurística descrita ao algoritmo *Alpha-Beta Pruning*. Com isso, é capaz de visualizar apenas um número limitado de lances futuros, mas é capaz de estimar o melhor lance possível dado os *payoffs* dos estados mais distantes visualizados.

IV. Resultados

A implementação do jogo e do algoritmo de aprendizado foi realizada em Python. O produto final consta em um jogo simples, que permite que um jogador humano jogue contra o computador. O jogador humano joga com as peças brancas, e tem acesso a uma sugestão de lance que corresponde ao melhor lance encontrado pelo algoritmo. Esse jogador pode seguir a sugestão do algoritmo ou não. O computador, que joga com as peças pretas, irá sempre seguir a melhor solução encontrada pelo algoritmo.

As **Figuras 4** e **5** a seguir mostram o resultado de um jogo quando ambos os jogadores seguem os lances sugeridos pelo algoritmo, o qual consegue analisar até 6 lances futuros.

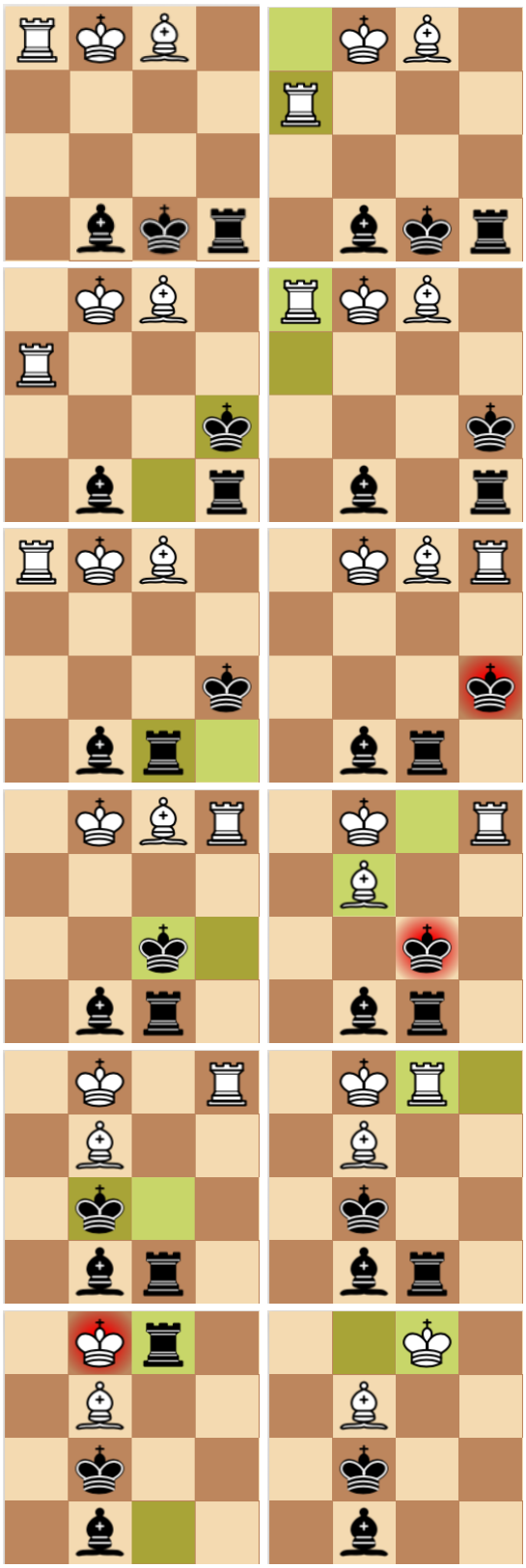

Figura 4: Jogo com ambos os jogadores aplicando o algoritmo implementado - lances: 0 à 11

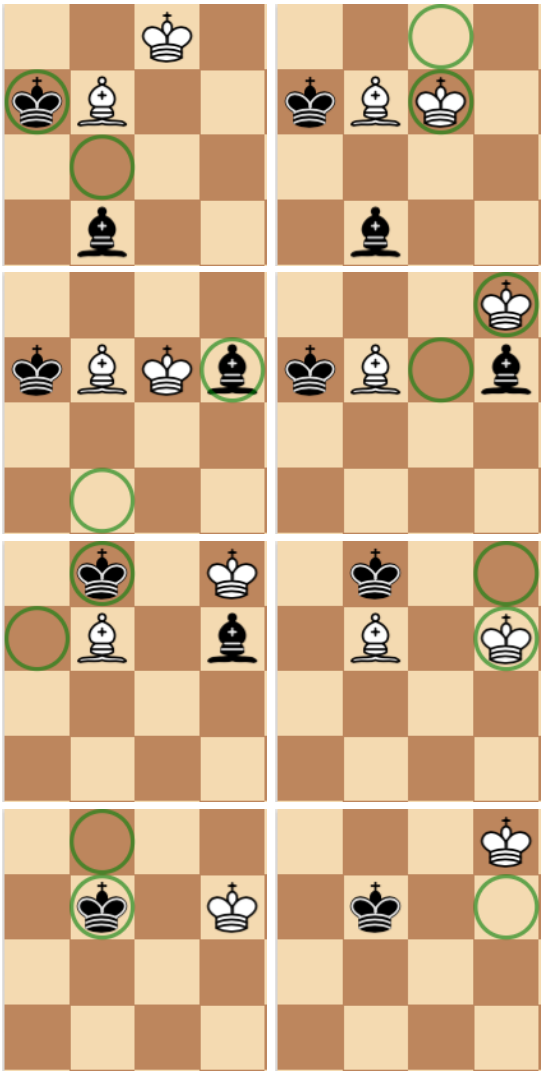

Figura 5: Jogo com ambos os jogadores aplicando o algoritmo implementado - lances 12 à 19. Resultado final: Empate!

V. Conclusões

Mesmo as peças brancas tendo a iniciativa devido ao fato de realizarem o lance inicial, quando ambos os jogadores jogam a "melhor" solução, o resultado é sempre empate. Devido à simplicidade do jogo, um jogador humano experiente consegue empatar com o algoritmo implementado, mas dificilmente sairá vitorioso, mesmo quando o algoritmo consegue visualizar apenas 6 passos futuros. Em um jogo mais complexo, como o xadrez clássico, e com algoritmos mais potentes e capazes de visualizar dezenas de passos futuros, um jogador humano dificilmente conseguirá alcançar resultados próximos de um computador.

Um ponto interessante a se observar, é que o algoritmo *Alpha-Beta Pruning* obteve um tempo de execução médio de cerca de 15 segundos para cada avaliação. Esse valor foi reduzido consideravelmente a medida que o jogo foi se aproximando do fim, devido à redução do número de peças no tabuleiro e, consequentemente, de possíveis estados futuros. Por outro lado, o algoritmo *Backpropagation* tradicional, obteve resultados muito piores, demorando vários minutos para encontrar o mesmo resultado que o *Alpha-Beta Pruning* obteve em segundos. Isso mostra a grande otimização e vantagem computacional que o *Alpha-Beta Pruning* oferece em relação ao algoritmo *Backpropagation* original.

Por fim, podemos concluir que diversas situações do mundo real podem ser modeladas no formato de um jogo na forma extensiva. O algoritmo *Backpropagation* é capaz de encontrar a melhor solução para qualquer jogo modelado dessa forma em tempo exponencial. O algoritmo *Alpha-Beta Pruning* é composto de uma versão modificada do *Backpropagation* e consegue obter o mesmo resultado ótimo em um tempo médio muito menor. Apesar de seus ótimos resultados, mesmo o algoritmo *Alpha-Beta Pruning* pode ter dificuldade em encontrar o resultado ótimo em tempo hábil de um jogo suficientemente complexo, como o xadrez. Nesses casos, aplica-se uma heurística ao algoritmo de busca. Com uma heurística suficientemente bem elaborada, o algoritmo é capaz de visualizar apenas um número limitado de lances futuros, mas consegue estimar o melhor lance possível dado os *payoffs* dos estados mais distantes visualizados.

Referências

[1] K. L. B. Yoav Shoham. *Multi Agent Systems*. 2010.