Coordenação do Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

Inteligência artificial: aplicação do algoritmo Minimax em um jogo de xadrez

Eduardo Prates Albuquerque

Msc. Diogo Fernando Trevisan (Orientador)

Msc. Delair Osvaldo Martinelli Júnior (Co-orientador)

Dourados - MS

2016

Coordenação do Curso de Sistemas de Informação
Universidade Estadual de Mato Grosso do Sul

Inteligência artificial: aplicação do algoritmo Minimax em um jogo de xadrez

Eduardo Prates Albuquerque

Outubro de 2016

Banca Examinadora:

Prof. Msc. Diogo Fernando Trevisan (Orientador)

Sistemas de Informação - UEMS

Prof. Msc. Delair Osvaldo Martinelli Júnior (Co-orientador)

Sistemas de Informação - UEMS

Profa. Msc. Jéssica Bassani de Oliveira

Sistemas de Informação - UEMS

Profa. Dra. Mercedes Roco Gonzales Marquez

Sistemas de Informação - UEMS

Inteligência artificial: aplicação do algoritmo Minimax em um jogo de xadrez

Eduardo Prates Albuquerque

Este exemplar corresponde à redação final da monografia da disciplina Projeto Final de Curso devidamente corrigida e defendida por Eduardo Prates Albuquerque e aprovada pela Banca Examinadora, como parte dos requisitos para a obtenção do título de Bacharel em Sistemas de Informação.

Dourados, Outubro de 2016.

Msc. Diogo Fernando Trevisan (Orientador)

Msc. Delair Osvaldo Martinelli Júnior (Co-orientador)

Agradecimentos

Agradeço primeiramente a Deus por todas as oportunidades que tem me dado e por me abençoar em todos os momentos, principalmente naqueles momentos em que não fui merecedor.

A meus pais, Enildo Marinho de Albuquerque e Elaine Seren Prates, pelo amor incondicional que me dão a cada dia. Agradeço também pelo apoio a cada decisão e por me darem toda a estrutura, tanto familiar quanto financeira, que me permitiu chegar a esse momento.

Aos demais familiares, os quais se fizeram presentes a cada êxito e a cada tropeço, me auxiliando nos momentos difíceis e comemorando cada conquista junto comigo.

A minha namorada, por todo o incentivo, carinho e companheirismo.

Meus amigos, pelos diversos gestos de amizade, pelos conselhos, apoio e pela torcida.

À Universidade Estadual de Mato Grosso do Sul pela oportunidade de ingressar em um faculdade e por fornecer a estrutura necessária para a minha graduação.

Ao meu orientador Diogo Fernando Trevisan, por te me ajudado e me guiado no decorrer desse trabalho, se colocando sempre à disposição.

E a todos os professores, pelos ensinamentos transmitidos durante esses anos, que foram fundamentais para a elaboração deste projeto e para a minha formação.

Resumo

Este trabalho tem por objetivo estudar e implementar uma técnica de inteligência artificial dentro de um jogo de xadrez. Nele, serão apresentados alguns conceitos e características sobre inteligência artificial. Também será abordado sobre o jogo de xadrez, suas regras, estrutura, movimentos e conceitos. Em seguida, será apresentada uma técnica de IA (algoritmo Minimax), suas características, as partes que a compõe e sua aplicação prática. Por fim, será apresentado um programa que utiliza o algoritmo Minimax, dentro de um jogo de xadrez, simulando assim a inteligência humana por meio da movimentação automática das peças de um dos jogadores na partida de xadrez.

Palavras-chave: Inteligência Artificial, Xadrez, Minimax.

Abstract

This assignment aims the study and implementation an artificial intelligence technique within a chess game. In it, we show some concepts and characteristics about artificial intelligence. It will also have an approach about the chess game, its rules, structures, movements and concepts. Next, we explain about an AI (Minimax Algorithm), its characterisctics, its composition and its practical appliance. Lastly, we presents a software that uses the Minimax Algorithm within a chess game, simulating the human intelligence through the automatic piece-movement of one of the players of the chess match.

Keyword: Artificial Intelligence, Chess, Minimax.

Lista de Siglas

IA Inteligência Artificial.

NPC *Non Player Character* (Personagem não controlado pelo jogador).

IBM *International Business Machines* (Negócios Internacionais de Máquinas).

EUA *Estados Unidos da América*

Lista de Figuras

Lista de Códigos/Algoritmos

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Simular a inteligência humana e dar à máquinas a capacidade de racionar e tomar decisões a partir desse raciocínio, é algo que vem sendo estudado há vários anos. A inteligência artificial possui diversas técnicas capazes de chegar a uma conclusão, partindo de regras lógicas e um conjunto de informações disponíveis.

Em sua aplicação para jogos eletrônicos, uma técnica de IA é responsável por controlar os NPCs como os eventos aplicados a um personagem, ambiente, etc. Ela está diretamente ligada ao nível de dificuldade do jogo e é um dos componentes cruciais para despertar o interesse do usuário em relação ao mesmo.

1.1 Objetivo

Este trabalho teve por objetivo explorar a área de inteligência artificial, dando ênfase na aplicação de um algoritmo de busca dentro de um jogo de xadrez. Também será apresentado um jogo de xadrez, o qual foi desenvolvido pelo autor, onde um dos jogadores tem o movimento de suas peças controlados pela máquina (sendo gerados e avaliados por meio do algoritmo Minimax), simulando assim a disputa entre um humano (usuário) e máquina.

O jogo foi implementado por meio de linguagem C, possuindo as propriedades básicas um jogo de xadrez. Ele também possui uma técnica de IA capaz de analisar o estado atual do jogo. Em seguida, ele gera uma árvore de jogadas, a fim de avaliar cada movimento possível e retornar uma boa jogada, tendo como objetivo a vitória do jogador

controlado pela máquina.

1.2 Organização do trabalho

O presente trabalho está organizado em um único volume, dividido em 7 capítulos. São eles:

- **Capítulo 2**: retrata alguns conceitos sobre inteligência artificial e um breve histórico sobre essa área.
- **Capítulo 3**: relata sobre o que é o xadrez, sua finalidade, regras, estrutura, peças e qual a relação do jogo com a computação.
- **Capítulo 4**: apresenta detalhes sobre o desenvolvimento do projeto proposto, tendo como o foco a implementação do algoritmo Minimax e as partes que o compõe.
- **Capítulo 5**: expõe os resultados obtidos após o desenvolvimento do jogo e apresenta informações sobre o desempenho do algotirmo Minimax com poda Alfa Beta.
- **Capítulo 6**: apresenta algumas ideias a serem aplicadas em projetos futuros e também oportunidades de melhoria que podem dar ao jogo desenvolvido maior jogabilidade e eficiência na execução da técnica da IA.
- **Capítulo 7**: este ultimo capitulo traz as considerações finais de tudo o que foi aprendido.

Capítulo 2

Conceitos e características da inteligência artificial

Neste capítulo serão apresentados alguns conceitos importantes sobre inteligência artificial e também fatos históricos importantes para o desenvolvimento dessa área.

2.1 Definição

Para Rich e Knight (1994), a inteligência Artificial é o estudo de como fazer os computadores realizarem coisas, que no momento, as pessoas fazem melhor. Já Kurzweil (1990) define IA como a arte de criar maquinas que realizam funções que requerem inteligência quando são executadas por humanos. Ela foi estudada por muitos filósofos, psicólogos e cientistas, onde um outro conceito interessante foi a comparação da mente humana com uma máquina, como descrito por Teixeira (1990):

> "[...] a mente humana funciona como um computador, e por isso o estudo dos programas computacionais é a chave para se compreender alguma coisa acerca de nossas atividades mentais. Podemos construir programas que imitem nossa capacidade de raciocinar, de perceber o mundo e identificar objetos que estão à nossa volta, e até mesmo de falar e de compreender nossa linguagem."Teixeira (1990).

A partir dos conceitos acima, pode-se definir a inteligência artificial como uma área que atua na criação de máquinas e aplicações capazes de simular o raciocínio humano e tomar decisões, visando resolver problemas. É importante ressaltar que a inteligência

artificial não é formada somente para fornecer informações e gerar resultados, sua função também é que esse raciocínio seja semelhante ao humano (como se o computador possuísse um cérebro), conforme descrito também por Teixeira (1990):

> "Assim, por exemplo, não basta simplesmente projetar e criar uma máquina de calcular (como nós a temos e usamos todos os dias, carregando-a no bolso) para dizermos que estamos fazendo IA. É preciso que essa máquina imite nossa atividade mental quando estamos fazendo uma operação aritmética"Teixeira (1990).

O ideal da inteligência artificial é simples, mas a execução é bem mais complicada. De modo sucinto, uma técnica de IA coleta (busca) dados sobre uma determinada situação. Em seguida, o computador utiliza seus conhecimentos sobre o tema, gerando soluções para o problema e depois verifica qual dessas soluções é a melhor. Após obter o melhor resultado, ele toma uma decisão e executa uma ou mais ações.

Os algoritmos da busca em jogos eletrônicos tem um papel importante em muitas aplicações no campo da inteligência artificial. O xadrez, por ser um dos primeiros estudos da inteligência artificial, foi o motivo de muitas técnicas serem criadas ou aperfeiçoadas tanto para o desenvolvimento dentro do jogo quanto para a utilização dessas técnicas para outras áreas da IA.

2.2 História da inteligência artificial

O surgimento da inteligência artificial é cercado de dúvidas, onde muitos fatos não podem ser adotados como verídicos por não serem provados. Mas frisa-se que a maioria das referências apresenta que o desenvolvimento dessa área está ligada com a criação dos primeiros computadores e os estudos sobre a mente humana.

Os primeiros passos largos da inteligência artificial estão relacionados à produção dos primeiros computadores, os quais foram idealizados a partir da Segunda Guerra Mundial, onde surgiu a idéia de construir máquinas que possuíssem a capacidade de "pensar", sendo úteis em estratégias de batalha. Após o fim da Segunda Guerra Mundial, essa área passou a ser influenciada por diversas outras áreas como matemática, filosofia, neurociência e psicologia.

Na Figura 2.1, temos uma representação das principais áreas que influenciaram a inteligência artificial, auxiliando o desenvolvimentos dessa área e também algumas subáreas,

que surgiram a partir da IA.

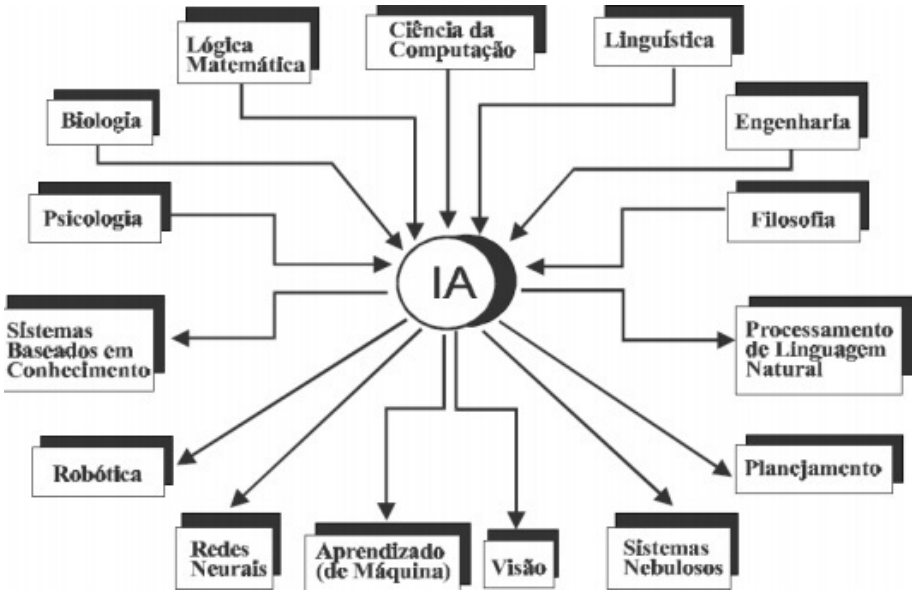


Figura 2.1: Áreas Relacionadas com a Inteligência Artificial.
Fonte: (Monard e Baranauskas, 2000)

Conforme Gomes (2010), em 1950, Alan Turing (cientista considerado como pai da computação devido a suas realizações) propôs um teste que permite avaliar a inteligência de um objeto computacional. Esse teste consiste em colocar uma máquina e um ser humano em duas salas separadas, além de outro humano que atuará como interrogador. Este fará perguntas aos dois sem saber quem é quem e receberá respostas. A máquina é considerada aprovada no teste, caso o interrogador não consiga distinguir o humano da máquina.

O termo inteligência artificial foi utilizado pela primeira vez no ano de 1956 em uma conferência em Dartmouth, EUA, onde pesquisadores apresentaram trabalhos sobre lógica e redes neurais. A partir desse encontro, o tema foi abordado com mais detalhes e entre de 1950 e 1970 se deram as primeiras contribuições da IA na área de jogos.

Segundo Bittencourt (2001), uma das criações mais famosas na implementação de técnicas de inteligência artificial é o Deep Blue, criado pela empresa IBM. Esse supercomputador e software era capaz de jogar xadrez. Seu resultado mais expressivo foi a vitória contra Gary Kasparov (campeão mundial de xadrez daquela época e considerado o melhor jogador de todos os tempos) em partidas de xadrez realizadas nos anos de 1996 e 1997. No primeiro confronto, Kasparov venceu três partidas, empatou duas e perdeu

uma. Já no segundo confronto, após o Deep Blue receber várias atualizações, Kasparov foi derrotado duas vezes, empatou três e venceu uma.

Fujita (2005) propõe que a inteligência artificial se destaca nas seguintes subáreas:

- **Robótica**: destacam-se as criações de robôs utilizados para realizar tarefas rotineiras e de produção. Também se destacam máquinas utilizadas para reconhecimento de planetas. Estima-se que no futuro ocorrerá o sonho de muitos cientistas, a criação de vidas artificiais (robôs programados para agir como humanos em quase todos os sentidos);

- **Jogos**: existem diversos jogos, os quais possuem técnicas de inteligência artificial, de modo que o usuário se depare com adversários e ambientes virtuais capazes de agir sem o controle do mesmo;

- **Controle Autônomo**: sistema capaz de executar tarefas como dirigir um veículo de forma que ganhe experiência diante de experiências anteriores;

- **Medicina**: existem programas que utilizam probabilidade para simular o diagnóstico de um médico especialista em várias áreas da medicina;

- **Sistema de Reconhecimento de Voz**: sistema utilizado para aumentar a segurança de diversos dispositivos e ferramentas, é altamente comercializado atualmente;

- **Redes Neurais**: sistemas baseados em ligações computacionais. Essa subárea é muito utilizada em simulações do cérebro humano.

Capítulo 3

Xadrez

O jogo de xadrez é jogado entre dois adversários que movimentam peças em um tabuleiro. O objetivo do jogo é colocar o rei do oponente sob ataque de um modo que não seja possível escapar deste ataque. Estima-se que 605 milhões de pessoas em todo mundo jogam xadrez, praticando para diversão e torneios.

O origem do jogo ainda não foi confirmada, mas estima-se que o jogo se originou na Ásia, mais especificamente na Índia por volta do século VI d.C. O que se sabe é o jogo surgiu com outras formas e regras. A forma atual do jogo se iniciou no século XV no sudoeste da Europa, sofrendo algumas alterações até chegar ao modo que conhecemos hoje.

Segundo Carolus (2006), o xadrez é um jogo baseado em turnos, ou seja, cada jogador pode realizar suas ações separadamente, enquanto o outro jogador aguarda. O xadrez é um jogo onde toda a informação está sendo vista por ambos os jogadores. Essa característica torna-o um bom jogo para aplicação de técnicas de inteligência artificial. Realça-se que ele é um jogo determinístico, não dependendo de nenhum fator aleatório como a sorte.

3.1 Estrutura e disposição das peças

Segundo Soccol e Zucolotto (2006), o jogo ocorre em um tabuleiro quadriculado, uma matriz 8 X 8 com 64 casas de mesmo tamanho, possuindo cores alternadas em cada casa. Quanto às peças, cada jogador tem a sua disposição 16 peças de uma determinada

cor (geralmente as peças são formadas por um conjunto de peças pretas e outro composto por peças brancas).

Cada jogador possui à sua disposição as seguintes peças: 8 peões, 1 rei, 2 bispos, 2 cavalos e 1 rainha. A Figura 3.1 apresenta a disposição inicial das peças no tabuleiro.

Figura 3.1: Posição inicial das peças na partida
Fonte: http://www.tabuleirodexadrez.com.br/tabuleiro-e-pecas-do-xadrez.html

3.2 Movimentação e ataque das peças

Segundo o site Tabuleiro de Xadrez, cada peça possui uma particularidade ao se deslocar e atacar no tabuleiro. São elas:

- **Peão** (*pawn*): se desloca somente para frente, uma casa por jogada (frisando que no primeiro movimento de cada peão, o mesmo pode se deslocar duas casas a frente), onde seu ataque só pode ser realizado caso a peça adversário esteja uma casa a frente e uma casa a direita ou uma casa a frente e a esquerda. Diferente das demais peças, o peão é a única peça, a qual ataca de modo diferente ao qual se desloca;

- **Rei** (*king*): ele pode se deslocar e atacar em qualquer sentido no tabuleiro, deslocando-se somente uma única casa em cada movimento, desde que este não o coloque sob ataque. Realça-se que sua captura resulta no fim do jogo;
- **Rainha** (*queen*): também chamada de dama, esta peça é considerada a peça mais poderosa do jogo, pois pode se mover/atacar em posição horizontal, vertical ou diagonal. A quantidade de casas ficará a critério do jogador;
- **Torre** (*rook*): esta peça se move na horizontal ou vertical, onde a quantidade fica a critério do jogador;
- **Bispo** (*bishop*): cada bispo move-se em diagonal, nas casas da mesma cor em que a peça se encontrava no início da partida. Do mesmo modo que a torre e rainha, ela pode se mover quantas casas o jogador desejar;
- **Cavalo** (*knigth*): esta peça move-se em "L", ou seja, duas peças horizontalmente e uma verticalmente ou vice-versa.

Conforme a Figura 3.2, podemos visualizar a maneira como cada peça se movimenta e ataca dentro da partida.

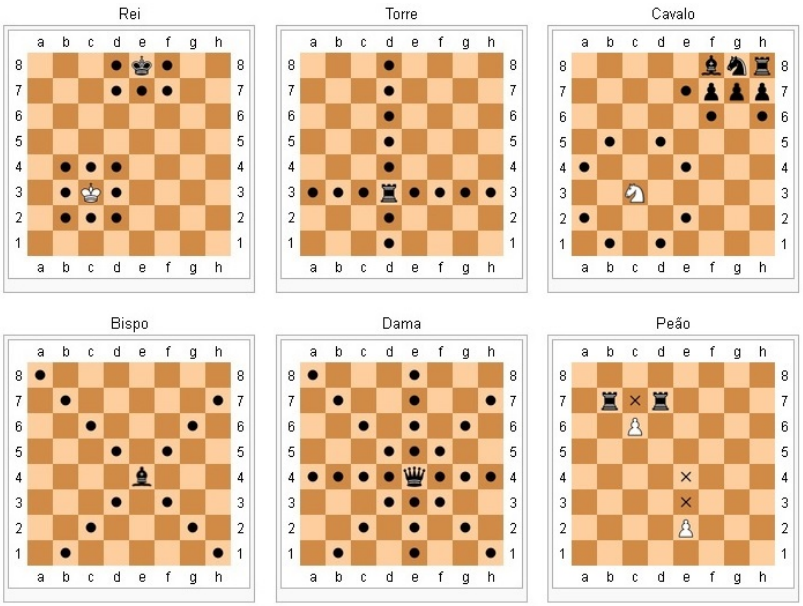


Figura 3.2: Movimentação das peças no xadrez
Fonte: http://www.douradosnews.com.br/cultura-lazer/jogar-xadrez-pode-ajudar-a-melhorar-o-raciocinio

É importante frisar que nenhuma peça, exceto o cavalo, pode pular outra peça em seu caminho. Realça-se também que as peças brancas sempre devem realizar o primeiro movimento da partida.

3.3 Promoção

Apesar de ser a única peça que não pode recuar de posição, o peão é a única peça que pode ser promovida. Se um peão conseguir avançar até a última fileira do outro lado do tabuleiro, ele é promovido e pode escolher qual peça deseja se tornar: rainha, torre, bispo ou cavalo. Normalmente, a peça escolhida é a rainha (por ser a peça mais poderosa). Assim, é possível que um time tenha até 9 rainhas disponíveis dentro de uma mesma partida.

3.4 Captura de peças

Dentro do jogo, é possível capturar (tomar) peças do adversário. Se uma peça inimiga estiver no raio de ataque de uma determinada peça, aquela pode ser capturada. Quando uma peça é capturada, esta é retirada do jogo e a peça que a atacou assume sua posição.

3.5 Término do jogo

Conforme dito anteriormente, o jogo de xadrez pode acabar de três maneiras: derrota, empate ou vitória.

Uma vitória ou derrota pode ocorrer de em duas formas: quando um jogador abandonar a partida ou quando ocorrer um xeque mate.

Para entendermos o que é o xeque mate, precisamos primeiro entender o que é o xeque. O xeque é um estado do jogo, o qual indica que há uma ameaça de captura ao Rei. Já o xeque mate, se refere ao estado em que não há nenhum movimento capaz de fazer com que o rei saia da posição de xeque, resultando no fim do jogo. É importante frisar que se um rei está em xeque, o movimento das peças do mesmo time que esse rei só

é válido se este tirá-lo do estado de xeque.

O empate pode ocorrer em diversas situações:

- Quando ambos os jogadores decidem encerrar a partida;
- Por insuficiência de peças: quando não há peças suficientes para garantir o xeque mate;
- Por 'afogamento' do rei: é uma situação em que o rei está encurralado em uma posição, a qual não consegue sair, pois as opções são inválidas ou resultam em xeque mate para o adversário. É importante frisar que essa situação é diferente do xeque mate, pois o rei não está em xeque;
- Por regra dos 50 movimentos: estado do jogo em que o empate pode ser solicitado, devido ao fato dos 50 últimos movimentos terem sido realizados sem que nenhuma peça fosse capturada e que houvesse movimento de qualquer peão;
- Por regra das três posições: situação em que um jogador pode solicitar empate quando uma posição for repetida três vezes durante a partida com o mesmo jogador (importante frisar que possibilidade de execução de movimentos especiais é considerada para validar esse tipo de empate).

3.6 Relação com a computação

Estima-se que cientistas da computação começaram a trabalhar na construção de máquinas que jogam xadrez desde 1890 e que os primeiros programas de computador enxadristas foram os primeiros trabalhos a serem desenvolvidos relacionados a IA (entre os anos de 1950 a 1970).

Segundo Aguiar (2007), o xadrez é um jogo finito (o jogo sempre termina) e de informações completas (não sofre infuência de fatores como sorte e não possui informações ocultas) e é um jogo de soma zero, ou seja, quando um jogador está em vantagem, o oponente está em desvantagem. Assim, podemos notar que o xadrez possui propriedades que fazem dele um jogo que exige intelecto. Logo, é uma boa opção para se trabalhar com IA.

A computação e a inteligência aritifical se relacionam, pois esta trata problemas para os quais ainda não há solução pronta, em forma de um algoritmo. Assim, ela necessita criar espaços de busca capazes de avaliar o cenário em um determinado contexto, analisar as possibilidades e encontrar uma solução.

Por sua vez, a computação possui propriedades que possibilitam que essa análise seja feita de modo mais rápido e eficiente (uma vez que máquinas podem efetuar cálculos mais rapidamente e não sofrem influência de fatores humanos como cansaço), garantindo assim que a solução (caso exista) seja encontrada.

A aplicação da inteligência artificial é o maior desafio no desenvolvimento do jogo computacional de xadrez, pois esse jogo possui um grande espaço de busca. Segundo Shannon (1950), o estudo denominado Grandeza de Shannon propõe que existem cerca de 10^{120} posições possíveis em um jogo de xadrez no meio da partida, variando de acordo com o estado atual.

Assim, no caso do xadrez, é impossível (em um tempo razoável) uma máquina examinar todo o espaço de busca. Mas mesmo assim, a utilização de técnicas de busca é capaz de retornar bons resultados. A técnica mais utilizada para esse jogo de tabuleiro é o algoritmo Minimax (que será melhor descrito no subitem 4.5).

3.7 Outras técnicas

Para aplicar uma técnica de inteligência artificial sobre um problema, é necessário obter informações sobre esse problema e possuir um conjunto de regras bem definido. Algoritmos de busca tem por finalidade analisar um cenário com diversas possibilidades de solução e encontrar uma solução eficiente para o problema. Entratanto, se o algoritmo não possuir informações relevantes sobre o contexto, ele percorrerá o cenário, mas não o analisará corretamente (não retornando um bom resultado e não contribuindo para uma boa tomada de decisão).

Em jogos como o xadrez, são necessários algoritmos capazes de analisar um cenário composto por dois agentes (jogadores), de modo que eles atuem em turnos diferentes. Além do algoritmo Minimax, há outras técnicas as quais podem ser utilizadas na solução do problema do jogo de xadrez. São elas:

- Algoritmos de aprendizado: são técnicas capazes de estabelecer padrões e alterar sua forma de análise, ou seja, se adaptar a novas situações visando ser mais eficiente na solução do problema. Aplicando o conceito ao xadrez, consiste basicamente num aprendizado indutivo a partir de exemplos ou experiências anteriores (jogadas) e classificação dos mesmos em categorias, onde uma das técnicas mais utilizadas nesse contexto é o Q-Learning juntamente com funções heurísticas (como o Minimax).
- Algoritmo Negamax: é uma variação do algoritmo Minimax que, segundo Silva e Silva (2014), utiliza variação de sinal de modo que tanto os nós max quanto os nós min tentem maximizar o peso, descartando a necessidade da minimização feita pelos nós min.

Capítulo 4

Implementação do jogo de xadrez

Conforme proposto, um programa foi criado, cuja sua finalidade é representar um jogo de xadrez entre usuário (peças brancas) e máquina (peças pretas). O mesmo foi desenvolvido em linguagem C, voltado para a plataforma Windows, podendo ser compilado por meio do GCC (versão 4.7.2), tendo sua inferface apresentada no prompt de comando do Windows.

Esse jogo procura atender as propriedades básicas de um jogo de xadrez, como tabuleiro, peças, movimentação das mesmas, validação dos movimentos, propriedades de ataque e outras regras básicas. Também pode-se encontrar nele, um algoritmo capaz de avaliar o cenário atual do jogo e a disposição das peças dos dois jogadores, de modo que gere uma boa jogada para responder ao movimento das peças brancas. Em outras palavras, a máquina será o adversário do usuário e controlará o movimento das peças pretas.

Sales (2008) define o jogo de xadrez como um problema de busca com os componentes a seguir:

- **Estado inicial**: posições atuais das peças no tabuleiro e o jogador que fará o próximo movimento.
- **Operadores**: jogadas legais para em um determinado estado da partida.
- **Teste de término**: verifica se o estado corresponde ao fim do jogo.
- **Função de utilidade**: também chamada de função objetivo, ela retorna o valor numérico para os estados finais que representa o fim do jogo, no caso xadrez: 1 (vitória), 0 (empate), -1 (derrota).

4.1 Estrutura dos arquivos

Essa seção lista e explica, de modo sucinto, cada arquivo que compõe o código. São os arquivos:

- **conio.v3.2.4.c e conio.v3.2.4.h**: biblioteca que permite trabalhar com as propriedades do prompt de comando, como identificação da posição atual do cursor do mouse, pressionamento de teclas, cor de fundo, cor da letra, título, entre outras propriedades.
- **console.v1.5.4.c e console.v1.5.4.h**: permite trabalhar com detalhes relacionados à janela do prompt e identificação de eventos do mouse e do teclado.
- **make.bat**: responsável por efetuar a compilação do código e identificação de erros e warnings de modo automático.
- **func.c e func.h**: biblioteca responsável pela construção do tabuleiro, identificação, validação e realização dos movimentos das peças. Elas também são responsáveis por tomar decisões diante da ocorrência de um evento.
- **xeque.c e xeque.h**: responsáveis pela verificação das ocorrêcias de xeque e xeque mate.
- **arvore.c e arvore.h**: bibliotecas que geram possibilidades de jogada, criam a árvore de jogadas, analisam e avaliam os nós gerados e retornam um movimento considerado bom a ser realizado pela máquina.

Outras considerações importantes podem ser encontradas no arquivo "README.txt", que acompanha o código fonte do programa.

4.2 Representação do tabuleiro

Na aplicação proposta, o posicionamento das peças dentro do tabuleiro foi feito através de uma matriz bidimensional com dimensões 8x8. Cada elemento dessa matriz possui um determinado valor e esse valor definirá se há uma peça naquela determinada

posição. As vantagens dessa metodologia são a fácil implementação e facilidade em avaliar o valor das peças, sua desvantagem é a necessidade de um loop $O(n)^2$ para percorrer todas as posições dessa matriz. Abaixo está disposta, a maneira como as peças foram definidas:

- Peças cujo último dígito é 1, representam as peças pretas.
- Peças cujo último dígito é 0, (excedendo campos cuja o valor é zero), representam as peças brancas.
- Peças cujo valor é zero, representam posições não ocupadas no tabuleiro.

No que se refere ao tipo de peça, a idenficação ocorre do seguinte modo:

- 10 e 11 representam os peões brancos e os peões pretos, respectivamente.
- 20 e 21 representam os bispos brancos e bispos pretos, respectivamente.
- 30 e 31 representam os cavalos brancos e os cavalos pretos, respectivamente.
- 40 e 41 representam as torres brancas e as torres pretas, respectivamente.
- 50 e 51 representam a rainha branca e a rainha preta, respectivamente.
- 60 e 61 representam o rei branco e o rei preto, respectivamente.

Logo, para acessar uma determinada peça, basta informar a posição x e a posição y em que ela se encontra dentro da matriz.

4.3 Controle de eventos

Após a construção do tabuleiro, o programa fica aguardando que algum evento ocorra. Ele possui o controle de jogadas, o qual identifica qual jogador realizará o próximo movimento. Quando é o turno da máquina, o programa chama as funções que fazem parte do algoritmo Minimax e o movimento ocorre automaticamente (por meio da execução de funções que serão citadas posteriomente).

Já quando é o turno do usuário (peças brancas), o programa aguarda que o usuário informe qual peça ele deseja movimentar e qual a posição do tabuleiro ele deseja que essa peça assuma.

Todos os eventos válidos do jogo proposto se dão através do clique do botão esquerdo do mouse, desde que no momento do clique, o cursor esteja dentro do intervalo estipulado. Os únicos eventos válidos do teclado (dentro do jogo), são o pressionamento da tecla "ESC", caso o usuário queira encerrar o jogo antes de seu término e o pressionamento de qualquer tecla após o encerramento da partida.

4.4 Função de avaliação estática

Essa função é uma metodologia de avaliação que quantifica e qualifica um determinado estado do jogo. Ela consiste na passagem de informações base (regras e valores) que servem como conhecimento para solução do problema. A partir dela, a função retorna o status atual do jogo naquele determinado momento. O Código 4.1 apresenta um trecho do programa desenvolvido, o qual retrata parte da avaliação estática:

Código 4.1: Avaliação Estática

```
float avalia (int Mat_Tabuleiro[ ][8])
{
 int linha, coluna;
 float aval_ppt=0, aval_pbr=0;

   /* Varre o tabuleiro, verificando se há peças*/
  for (linha=0;linha<8;linha++)
  {
     for (coluna=0;coluna<8;coluna++)
     {
        /* verifica se há algum peão no tabuleiro*/
        if (M[linha][coluna] / 10 == 1)
        {
           /*  Verifica se  peão é branco*/
           if (M[linha][coluna] % 10 == 0)
           {
              aval_pbr +=1;

             /* Verifica se está no meio do Tabuleiro*/
```

```
            if (linha == 3 || linha == 4)
            {
                aval_pbr +=0.2;
             }
          }
        }
... /* Continuação da função*/
     }
  }

 /* Retorna valor da soma das peças  pretas - peças brancas*/
 return aval_ppt-aval_pbr
 }
```

Fonte: (Criação do Autor)

Na aplicação desenvolvida, foi utilizada a avaliação estática proposta por Turing que, segundo Rich (1988), é uma das mais simples e mais utilizadas no jogo de xadrez. Essa metodologia consiste em atribuir valores para cada tipo de peça e em seguida contar a quantidade de peças não capturadas no jogo, multiplicar cada tipo de peça pelo seu respectivo valor e somar os valores das peças pretas, os valores das peças brancas, e por fim calcular a vantagem de uma sobre a outra (subtração).

Assim, a avaliação é feita com base na ponderação de valores, onde cada peça possui um peso, de acordo com determinados critérios. São eles:

- O valor do peão será definido com valor 1 e o mesmo valor será dado para sua posição no tabuleiro. Se ele estiver na terceira ou quarta linha do tabuleiro, seu peso será acrescido de 0,2.
- Cada cavalo e bispo recebe o valor 3 para qualquer posição do tabuleiro.
- O valor da torre será definido com o valor 5 para qualquer posição do tabuleiro.
- O valor da dama será definido com o valor 9 para qualquer posição do tabuleiro.

Apesar de ser um boa metodologia, há casos em que a função estática pode apresentar um resultado enganoso (Conforme a Figura abaixo):

Figura 4.1: Avaliação estática com resultado enganoso
Fonte: (Revista Conteúdo, Capivari, v.1, n.4, ago./dez. 2010)

De acordo com a Figura 4.1, é possível notar que a avaliação estática retorna um falso resultado, pois dá a entender que as peças brancas estão em vantagem, mas não visualizando a provável perda da rainha branca (posição B4) na próxima jogada .

Assim, somente a avaliação estática não é capaz de resolver o problema de avaliação de jogadas no xadrez, pois essa metodologia de avaliação consegue avaliar somente aquele determinado momento do jogo, não se atentando as jogadas futuras que podem ocorrer após a realização desse movimento, podendo apresentar resultados enganosos.

Nesse contexto, surgiu a necessidade de avaliar o estado atual do jogo e pensar também nos próximos movimentos do oponente e de si mesmo. Logo, algoritmos foram criados para suprir essa demanda, onde se destaca o algoritmo Minimax (que será melhor descrito a seguir).

4.5 Algoritmo Minimax

Segundo Fortes (2009), o algoritmo minimax é uma função recursiva de busca (semelhante ao método de busca em profundidade), criado por Von Neumman em 1928, que examina várias sequências de movimentos, a fim de encontrar uma que leve à vitória. O principio básico dessa técnica é retornar uma jogada, de modo que ela reduza as possibilidades de seu adversário realizar uma boa jogada.

Conforme Russel e Norvig (2004, p.160), o algoritmo minimax toma uma decisão, partindo do estado atual da partida, analisando jogadas futuras e as avaliando. É usado um gerador de movimentos que cria uma árvore de jogadas possíveis (onde a posição atual do jogo é a raiz da árvore e as jogadas futuras são os nós filhos dessa raiz). Em seguida, a árvore é percorrida e avaliada (utilizando a função de avaliação estática). Por fim, é retornado o valor que corresponde ao melhor caminho da árvore, o qual resultará no próximo movimento.

Este é um método que auxilia na tomada de decisão e é representado por dois agentes: MAX, retrata as possiveís jogadas de um jogador controlado pela máquina e MIN, que representa os possíveis movimentos do jogador controlado pelo usuário. Esse algoritmo toma como verdade que o oponente (MIN) sempre fará a melhor jogada possível. Assim, a função de MAX é escolher a melhor jogada, supondo que MIN executará sempre sua melhor jogada.

O algoritmo é baseado em cinco passos (vale realçar que o algoritmo em si, se dá a partir do 2º passo): (i) gerar toda a árvore de jogadas até os estados terminais ou até que o nível estipulado da árvore seja atingido; (ii) aplicar a função de avaliação estática aos nós folha, obtendo um peso para cada um deles; (iii) comparar o valor dos estados finais para determinar qual nó subirá um nível na árvore de busca; (iv) continuar passando os valores das folhas em direção ao topo, um nível por vez; (v) quando o valor da avaliação atingir a raiz da árvore, teremos um bom movimento a ser realizado pelas peças pretas. Frisa-se que esse processo será realizado após cada jogada do usuário (peças brancas) até que o jogo termine.

Quaglio (2013), se a profundidade máxima da árvore é m e b são movimentos válidos em cada ponto, a complexidade de tempo do algoritmo é O (b^m). A complexidade de espaço é O (bm) para um algoritmo que gera todos os sucessores de uma vez ou O(m)

para um algoritmo que gera um sucessor de cada vez.

4.5.1 Gerador de movimentos

Também chamada de árvore de jogadas, o gerador de movimentos é uma maneira de apresentar os movimentos possíveis dos dois jogadores, a partir do estado atual do jogo, para que possam ser avaliados.

Conforme a Figura 4.2, podemos notar a representação de uma árvore de jogadas em uma partida de xadrez, retratando algumas possíveis jogadas (tanto pelo jogador MIN quanto pelo jogador MAX) diante do estado atual.

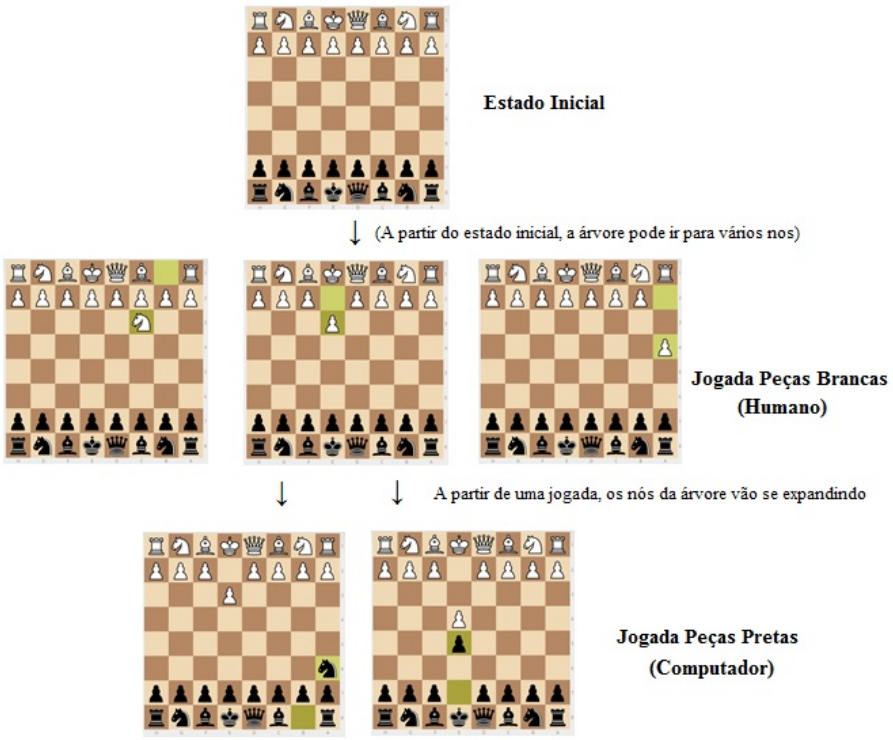


Figura 4.2: Exemplo de uma árvore com minimax
Fonte: Criação do autor

O modelo de representação do gerador de jogadas se dá por meio da estrutura de árvores. A árvore de jogadas é constituída por quatro fatores:

- **Nó raiz**: estado atual da partida.
- **Nós não folha**: possíveis movimentos a realizados no decorrer do jogo.
- **Níveis da árvore**: por se tratar de um jogo realizado entre dois jogado-

res que jogam alternadamente, cada nível da árvore representa as jogadas possíveis de um determinado jogador.

- **Nós folha**: representa os estados finais (vitória, empate ou derrota) ou os nós gerado até o nível limite da árvore (uma vez que a árvore de jogadas possui uma profundidade limitada).

Como já dito anteriormente, o maior desafio dessa etapa está ligado à quantidade de nós possíveis dentro do espaço de busca. Conforme também já descrito, é impossível mapear (em tempo hábil) todas os movimentos possíveis do xadrez, inserindo-os dentro de uma árvore de jogadas e os analisando em seguida. Assim, a árvore de jogadas do programa proposto criará uma árvore de busca com **X níveis e Y nós** (sendo X e Y um número fixo definido pelo desenvolvedor, levando em conta todos os fatores como memória disponível, nível de análise desejada, tempo de resposta, entre outros). Mas é importante realçar que quanto maior for a profundidade da árvore e a quantidade de nós por nível, mais eficiente será a avaliação e maiores serão as chances do computador realizar uma ótima jogada.

Aplicando os conceitos acima ao programa a ser desenvolvido, devem ser feitas algumas considerações sobre a árvore de jogadas presente no mesmo. São elas :

- Devido a limitações de memória, a árvore de jogadas possui 4 níveis de altura.
- O nível 0 é a raiz, ou seja, o estado atual da partida naquele determinado momento.
- Os níveis ímpares apresentam movimentos possíveis para as peças pretas.
- O nível par apresenta os movimentos possíveis para as peças brancas.
- Cada nó possui, no máximo, 20 nós filhos e assim sucessivamente.

Em vários estados do jogo, a quantidade de movimentos possíveis é maior do que 20 (quantidade de nós filhos para cada nó pai). Ex.: Se uma rainha estiver no meio do tabuleiro, ela pode realizar diversos movimentos diferentes, mas como a árvore de jogadas trabalha com tamanho limitado de nós, não há como inserir na árvore de jogadas todos os movimentos possíveis dessa rainha e das demais peças.

O jogo desenvolvido possui funções que realizam um filtro (antes de inserir o movi-

mento na árvore de jogadas), onde o mesmo seleciona boas jogadas e em seguida as insere na árvore de jogadas. Em contraposição, esse mesmo filtro (através da análise) opta por não inserir na árvore outras jogadas possíveis por não considerá-las boas opções. Assim, o jogo procura inserir as seguintes jogadas na árvore:

- Todas as possibilidades de uma peça preta atacar uma peça branca.
- Uma possibilidade de defesa para cada peça preta sob o raio de ataque de uma peça branca.
- Uma possibilidade de movimento para cada tipo de peça preta, de modo que essa peça não seja capturada na próxima jogada.
- Uma possibilidade de movimento para cada tipo de peça preta, de modo que essa peça possa capturar uma peça branca na próxima jogada.

Foi definido um tamanho limitado de 20 nós filhos para cada pai, pois permite a análise de uma boa quantidade de cenários (estado atual no primeiro nível da árvore + até 20 estados possíveis no segundo nível + até 400 estados possíveis no terceiro nível + até 8000 estados possíveis no quarto nível) e garante uma resposta rápida.

4.5.2 Avaliação de jogadas

Essa função tem por finalidade examinar os nós da árvore de jogadas e estimar quão bem o computador está indo ou qual a probabilidade de vencer o jogo (a partir desta posição). O Código 4.2 apresenta o pseucódigo do algoritmo Minimax:

Código 4.2: Algoritmo Minimax

```
função Minimax (nó, profundidade, jogador)
Inicio
  Se profundidade = 0 ou nó é um nó terminal então
    Retorne Valor Heurístico do nó
  Fim se
  Se jogador então
    melhorvalor := -  ∞
    Para cada nó filho faça
      v := Minimax (filho, profundidade - 1, FALSO)
```

```
      melhorvalor := max (melhorvalor, v)
    Fim para
    Retorne melhorvalor
  Senao
    melhorvalor := + ∞
    Para cada nó filho faça
      v := Minimax (filho, profundidade - 1, VERDADEIRO)
      melhorvalor := min (melhorvalor, v)
    Fim para
    Retorne melhorvalor
  Fim se
Fim
```

Fonte: (http://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/)

Já na Figura 4.3, está representada uma árvore de jogadas construída e avaliada de acordo com o algoritmo Minimax.

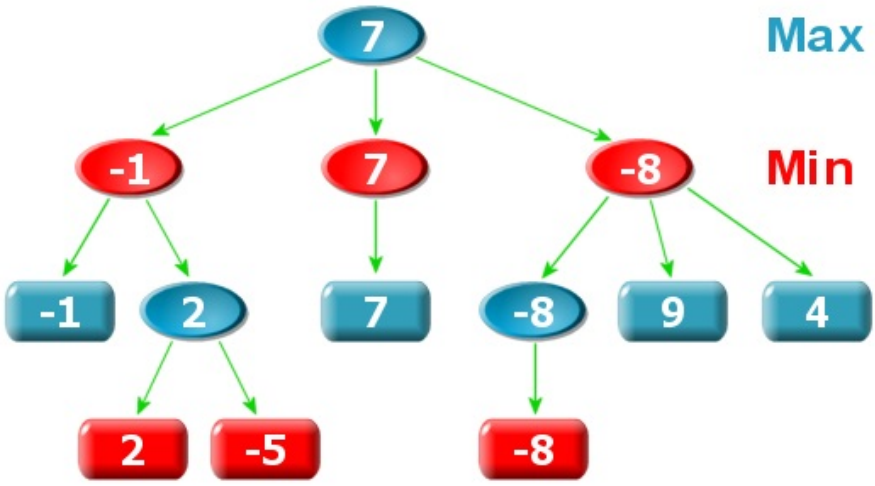


Figura 4.3: Exemplo de uma árvore de busca com avaliação Minimax
Fonte: (http://www.hamedahmadi.com/gametree/)

Conforme o Código 4.2 e a Figura 4.3, pode-se notar que a execução do algoritmo ocorre da seguinte maneira: quando a árvore atinge um nó folha ou o tamanho máximo do nível da árvore, a função de avaliação estática é aplicada nesse nó; quando o jogador

Min é analisado, é verificado qual o nó filho possui o menor valor (esse valor sobe para o nó pai); já quando o jogador Max é analisado, é retornado o valor da avaliação do filho com maior valor (esse valor também sobe para seu respectivo nó pai). Assim, quando o resultado da busca minimax chegar à raiz da árvore, será possível identificar qual é o melhor movimento a ser realizado pela máquina.

4.5.3 Algoritmo de poda Alfa Beta

Apesar de ser um bom algoritmo, o Minimax possui limitações e quando trabalha com um grande número de estados (como é o caso do xadrez), pode demorar um longo tempo para avaliar as ramificações da árvore, uma vez que é baseado no processo de busca em profundidade e analisa todos os nós da árvore de jogadas.

O algoritmo de poda Alfa Beta surgiu para suprir essa necessidade e ganhar tempo durante a busca. Segundo Justus (2006), ele funciona como uma otimização do algoritmo minimax e permite que nós que não apresentarão as melhores soluções não sejam analisados, podando assim a árvore de jogadas.

Conforme o Código 4.3, pode-se notar que o algoritmo de poda não muda o modelo de busca do algoritmo Minimax, ele apenas insere mais um critério de pesquisa, de modo que seja possível identificar se os próximos nós da árvore realmente precisam ser anlisados.

Código 4.3: Algoritmo Minimax com poda Alfa Beta

```
Função alfabeta (nó, profundidade, α, β, jogador)
Início
  Se profundidade = 0 ou  nó é um nó terminal então
    Retorne valor heurístico do nó
  Fim se
  Se jogador então
    melhorvalor= - ∞
    Para cada nó filho faça
      valor = alfabeta(filho, profundidade -1, α, β, FALSO))
      melhorvalor = max (melhorvalor, valor)
      α = max (α, melhorvalor)
      Se β ≤ α então
        Interrompa
```

```
      Fim se
    Fim Para
    Retorne valor
  Senão
    valor = + ∞
    Para cada nó filho faça
      valor  = alfabeta(filho, profundidade -1, α, β, VERDADEIRO))
      melhorvalor = min (melhorvalor, valor)
      β = min (β, melhorvalor)
      Se β ≤ α então
        Interrompa
      Fim Se
    Fim Para
    Retorne valor
  Fim se
Fim
```

Fonte: (http://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/)

Segundo Russel e Norvig (2004, p. 162; Rich, 1988, p. 140), o algoritmo é implementado da seguinte forma:

- Ele possui dois parâmetros: Alfa (α), é o valor mais alto do jogador Max e Beta (β), é o valor mais baixo do jogador Min. No algoritmo, essas variáveis são inicializadas com dois valores extremos, um negativo para Alfa e um positivo para Beta.
- A busca atualiza os valores de Alfa e Beta durante a análise, e poda os nós que possuem um valor pior que o valor atual de α (para o jogador Max) ou β (para o jogador Min)

Ainda segundo Russel e Norvig (2004), o algoritmo Alfa Beta (teoricamente) precisa de O($b^{m/2}$) nós para avaliar a árvore e escolher o melhor movimento, enquanto o algoritmo Minimax precisa de O(b^m). Mas na prática, isso não pode ser ocorrer, pois esse tempo depende da ordem em que os sucessores são examinado.

A Figura 4.4 apresenta um exemplo de árvore avaliada pelo algoritmo Minimax com poda Alfa Beta, seguindo a ordem de busca citada acima.

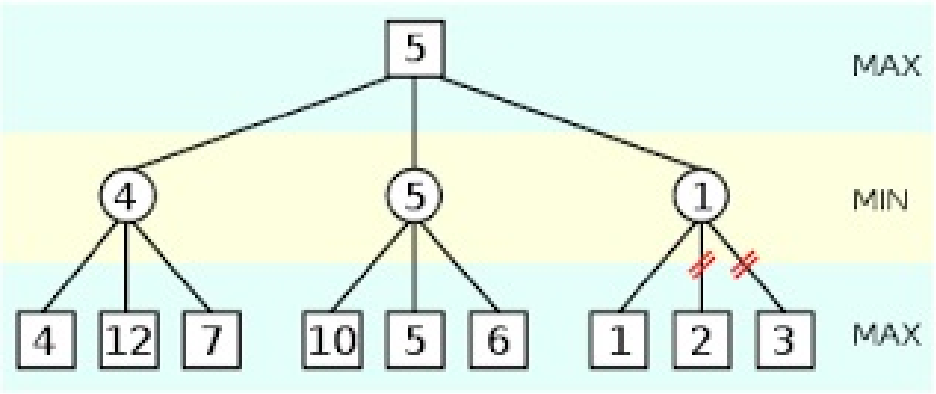


Figura 4.4: Exemplo de árvore de busca com algoritmo Minimax e poda Alfa Beta

Fonte: (http://www.wikiwand.com/es/Poda_alfa_beta)

Capítulo 5

Testes

Todos os testes foram executados em um mesmo computador, de modo que seja possível obter informações mais coesas e realizar comparações. O computador utilizado foi um notebook Acer, com processador Intel Core i5, sistema operacional Windows 8 (64 bits) e 6 GB de memória RAM.

O programa proposto é capaz de realizar uma partida de xadrez entre um usuário e máquina. A Figura 5.1 apresenta a interface do jogo desenvolvido.

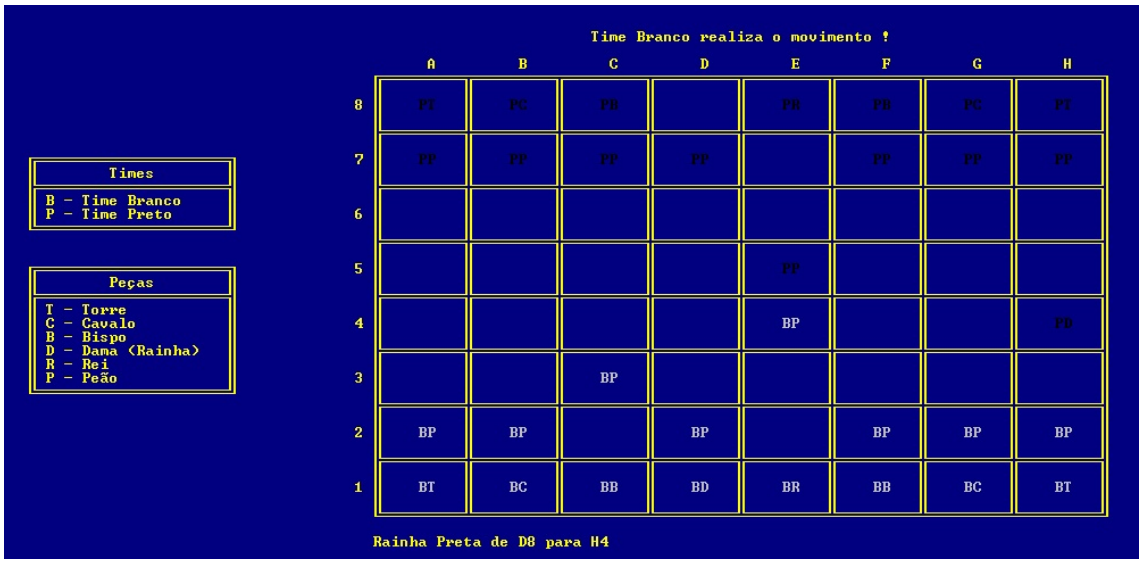


Figura 5.1: Algoritmo criando a analisando a árvore de jogadas
Fonte: (Criação do Autor)

Conforme a Figura 5.1, pode-se visualizar:

- O tabuleiro;

- A disposição das peças em um determinado estado do jogo;
- Tabelas de legenda para identificação das peças;
- Mensagem de informações sobre quem realiza a jogada;
- Em caso de jogada das peças pretas, é informado ao usuário para que aguarde a máquina realizar seu movimento (dado através geração da árvore e avaliação por meio do algoritmo Minimax com poda Alfa Beta).

Há outras informações dadas ao usuário pelo programa (mas não estão visiveis na Figura 5.1). São elas:

- Xeque, caso exista;
- Estado de fim do jogo (caso ocorra): xeque mate, empate por "rei afogado"ou empate por 50 movimentos;
- Após o movimento de cada peça preta, é informado ao usuário qual foi a peça movimentada pela máquina, sua posição atual e sua posição anterior.

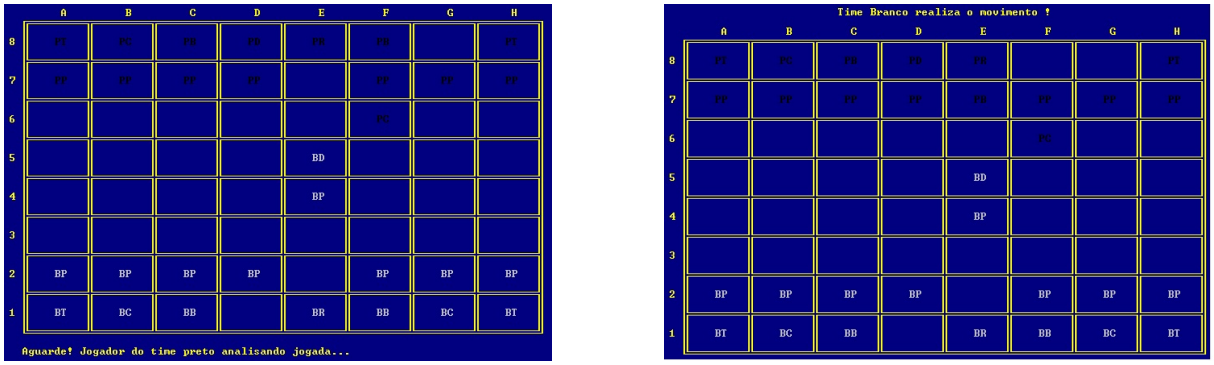
Conforme dito anteriormente, o projeto desenvolvido atuará com o algoritmo Minimax com poda Alfa Beta analisando uma árvore de 4 níveis (onde o nível 0 é o estado atual) onde cada nó pai possui, no máximo, 20 nós filhos. Mas para fins de avaliação de desempenho, a Tabela 5.1 apresenta o desempenho do algoritmo analisando diferentes limites de profundidade da árvore de jogadas.

Tabela 5.1: Tempo de execução algoritmo Minimax com poda Alfa Beta

Nível árvore	Pior Tempo	Tempo Médio	Melhor Tempo
3	0,000002 seg.	0,000008 seg.	0,0000005 seg.
4	0,2 seg.	0,06 seg.	0,04 seg.
5	5 seg.	2 seg.	0,3 seg.
6	50 seg.	20 seg.	8 seg.

Assim, pode-se observar que no algoritmo Minimax (com ou sem a poda Alfa Beta), o tempo de resposta está diretamente ligado ao número de nós filhos de cada pai e também a quantidade de níveis da árvore de jogadas, onde quanto maior a quantidade de nós a analisados, maior o tempo de processamento e mais lenta é a resposta. Em contraposição, quanto maior forem esses parâmetros, melhor será a jogada escolhida pela máquina.

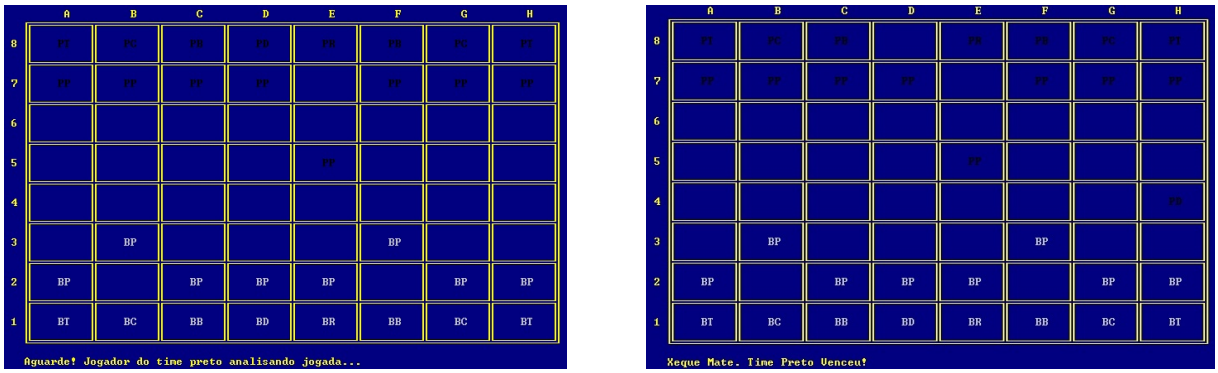
Nas figuras abaixo podemos notar alguns resultados encontrados durante os testes na ferramenta desenvolvida. Em caso de ataque adversário ao rei, a prioridade do algoritmo é a defesa desse rei. Na Figura 5.2 a, é possível identificar um movimento realizado pelo usuário de modo que coloque o Rei preto em Xeque (posição E5 no tabuleiro). É importante frisar que existem três formas de defesa: captura da ameaça, sacrifício de uma peça amiga e fuga. No estado específico, a metodologia de defesa adotada foi o sacrifício de uma peça preta de menor peso (posição E7 no tabuleiro), conforme a Figura 5.2 b.



a) Jogada do Usuário. b) Jogada da Máquina.

Figura 5.2: Defesa ao xeque realizado pelo adversário

Caso o rei preto não esteja em xeque, é analisada a possibilidade de realizar um xeque mate ao rei adversário. Caso essa possibilidade exista, o algoritmo a considera prioridade, pois essa possível jogada estará no segundo nível da árvore de jogadas contendo o maior valor possível obtido na avaliação estática. Na Figura 5.3 a, é possível notar uma jogada realizada pelo usuário. Já na Figura 5.3 b, pode-se notar a jogada gerada pela máquina, onde o estado atual do jogo é analisado, é identificada uma possibilidade de ataque que resulte em xeque mate (posição H4 no tabuleiro).



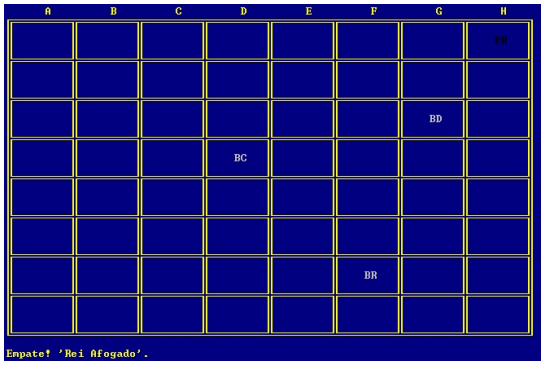
a) Jogada do Usuário. b) Jogada da Máquina.

Figura 5.3: Xeque Mate realizado pelo algoritmo Minimax com poda Alfa Beta

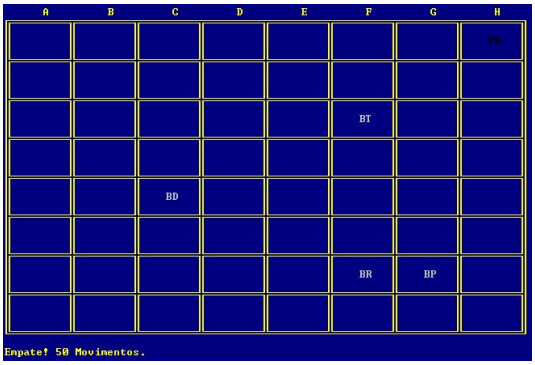
As demais possibilidades de jogada não resultam em término imediato da partida. Assim, o algoritmo percorre toda a árvore de jogadas, tendo como objetivo encontrar uma boa jogada, visando minimizar a possibilidade do adversário realizar uma boa jogada. As possibilidade de jogadas são:

- Defender uma peça preta sob ataque (de modo que o rei não esteja em xeque).
- Atacar uma peça adversário, de modo que essa peça não seja capturada na próxima jogada do adversário.
- Atacar uma peça adversário de maior expressão (peso).
- Movimentação visando um ataque na próxima jogada da máquina.
- Caso nenhuma das 4 opções acima seja possível, o algoritmo também propões opção de movimentação visando ataque em mais de duas jogadas futuras ou movimentar peça de modo que seja capturada pelo adversário (sempre visando a menor perda possível).

Durante a realização dos testes também foram testados os resultados de empate por 'rei afogado' e '50 movimentos', conforme as Figuras



a) Empate por 'Rei afogado'.



b) Empate dos 50 movimentos.

Figura 5.4: Resultados de empate na ferramenta desenvolvida.

Capítulo 6

Projetos futuros

Durante o desenvolvimento deste projeto, foram identificados alguns aspectos que podem ser melhor explorados e poderão ser objeto de trabalhos futuros, visando melhorar significativamente o jogo proposto no que diz respeito a jogabilidade e eficiência.

6.1 Busca de jogadas

A árvore de jogadas pode apresentar melhores resultados da seguinte maneira:

- Aumentando o número de nós filhos para cada nó pai da árvore de jogadas, de modo que mais jogadas possíveis sejam inseridas na árvore de jogadas, sendo analisadas e consequentemente aumentando a possibilidade do algoritmo Minimax retornar uma boa jogada.
- Aumentando o nível da árvore de jogadas, o que melhorará a análise de cada movimento possível inserido nessa árvore, resultando no retorno de melhores jogadas.

A melhoria dos dois aspectos citados acima também proporcionará o aumento da efetividade do algoritmo de poda Alfa Beta, o qual terá mais possibilidades de jogadas a analisar. Reforça-se o cuidado para que essas oportunidades de melhoria não tornem o jogo lento, uma vez que a árvore de jogadas será maior, o tempo de resposta também será maior e consumo de memória aumentará consideravelmente.

6.2 Inclusão de movimentos especiais e tipos de empate

Com o intuito de seguir as regras básicas do xadrez, propõe-se a inserção de movimentos especiais: “roque” e captura “em passant” e também a inserção dos empates por “Insuficiência Material”, “Repetição de lances” e “Xeque Perpétuo”

6.3 Interface

Será necessário criar uma interface para o jogo, permitindo que o usuário tenha melhor visão do tabuleiros e das peças, além de proporcionar maior entretenimento ao usuário.

Capítulo 7

Conclusões finais

Este trabalho propôs o desenvolvimento de uma ferramenta voltada ao jogo de xadrez, onde o principal objetivo era a realização de uma partida entre humano x máquina.

Embora o software desenvolvido tenha realizado bons resultados, ele ainda está muito distante dos softwares profissionais.

A principal dificuldade do projeto foi gerar uma quantidade limitada de jogadas possiveis, de modo que as jogadas geradas fossem boas, que a análise fosse eficiente que o tempo de resposta do algoritmo fosse rápido.

O algoritmo Minimax é um bom algoritmo de busca, pois consegue analisar todas as possibilidades da árvore de jogo, entretanto seu tempo de execução pode ser muito alto. Apesar da existência de mecanismos de poda, como o algoritmo de poda Alfa Beta, é impossível (em um tempo razoável) uma máquina examinar todo o espaço de busca de um jogo de xadrez durante o decorrer do jogo.

Referências Bibliográficas

RICH, E.; KNIGHT, K. **Inteligência Artificial**. Tradução 2. Ed. São Paulo: editora Makro Books, 1994.

TEIXEIRA, J. F. **O que é Inteligência Artificial**. São Paulo: editora Brasiliense, 1990.

FUJITA, E. **Algoritmos de IA para jogos**. Paraná: Universidade Estadual de Londrina, 2005.

STEFANIN, E. L. V.; FILHO, V. A. V. **Teorema Minimax:** uma aplicação em jogos de xadrez. Revista Conteúdo, Capivari, v. 1, n. 4., ago./dez. 2010.

SHANNON, C. E. **A Chess-Playing Machine**. Scientific American, v. 182, nº. 2, February.1950.

GOMES, D. S. **Inteligência Artificial:** conceitos e aplicações. Revista Olhar Científico, v. 1, n. 2, ago./dez. 2010.

RUSSEL, S.; NORVIG, P. **Inteligência Artificial**. Tradução 2ª Ed. Rio de Janeiro: editora Campus, 2004.

QUAGLIO, V. G. **Técnicas de inteligência artificial aplicada do jogo Othelo:** um estudo comparativo. Paraná: Universidade Estadual de Londrina, 2013.

KURZWEIL, R. **The Age of Spiritual Machines**. Massachusetts: The MIT Press, 1990.

ROSA, E. **Jogar xadrez pode ajudar a melhorar o raciocínio**. Disponível em http://www.douradosnews.com.br/cultura-lazer/jogar-xadrez-pode-ajudar-a-melhorar-o-raciocinio. Acesso em 12 de Abril de 2016.

JUSTUS, C. G. **Jogo de dama embarcado multinível**. Curitiba: Centro Universitário Positivo, 2006.

SOCCOL, C. M.; ZUCOLOTTO, G. **Implementação do GNU Chess em uma plataforma multiprocessada com arquitetura de comunicação baseada em rede intrachp**. Rio Grande do Sul: Universidade Católica do Rio Grande do Sul, 2006.

CAROLUS, J. W. T. **Alpha-Beta with sibling prediction prunning in chess**. Amsterdam: University of Amsterdam, 2006.

SALES, D. O. **Projeto e implementação de jogos eletrônicos**. Minas Gerais: Universidade Estadual de Lavras, 2008.

AGUIAR, F. M. **Ferramentas e métodos para apoiar o ensino de xadrez na fronteira entre o fundamento e a perícia**. Curitiba: Universidade Federal do Paraná, 2007.

BITTENCOURT, G. **Inteligência Artificial**: ferramentas e teorias. 2. Ed. Florianópolis: Universidade Federal de Santa Catarina, 2001.

FORTES, R. P. **Web API para disponibilização de jogo educativos em ambientes virtuais**. Santa Catarina: Universidade do Vale do Itajaí, 2009.

RICH, E. **Inteligência Artificial**. São Paulo: editora Mcgrau Hill, 1988.

Poda alfa-beta. Disponível em: http://www.wikiwand.com/es/Poda_alfa_beta. Acesso em 01 de junho de 2016.

Tabuleiro de Xadrez. Disponível em: http://www.tabuleirodexadrez.com.br. Acesso em 12 de abril de 2016.

MONARD, M. C.; BARANAUSKAS,J. A. **Aplicações de Inteligência Artificial:** uma visão geral. São Paiulo: Universidade de São Paulo, 2000.

AHMADI, H. **An Introduction to Game Tree Algorithms** Disponível em www.hamedahmadi.com/gametree. Acesso em 08 de maio de 2016.

SILVA, F. O. B.; SILVA, M. P. O. **Projeto e Desenvolvimento de um Tabuleiro Autônomo de Xadrez**. Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2014.

GeeksforGeeks. **Minimax Algorithm in Game Theory (Tic-Tac-Toe AI - Finding**

optimal move). Disponível em http://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move. Acesso em 20 de julho de 2016.

GeeksforGeeks. **Minimax Algorithm in Game Theory (Alpha-Beta Pruning)**. Disponível em http://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-4-alpha-beta-pruning/. Acesso em 20 de julho de 2016.