Universidade da Beira Interior
Departamento de Informática

P1 - Inteligência Artificial: *Chess Intelligent Agent*

Elaborado por:

Tiago Barreiros - a46118

Orientador:

Professor Doutor Hugo Pedro Proença

Agradecimentos

Quero agradecer a todas as identidades, que direta ou indiretamente, estiveram envolvidas neste trabalho desenvolvido no âmbito da Unidade Curricular (UC), Inteligência Artificial.

Como tal, começo por agradecer ao nosso orientador Prof. Doutor Hugo Pedro Proença, por suscitar e inseriu em nós, alunos, o interesse em saber mais, de nos tornar mais hábeis, proporcionando-nos todas as aprendizagens e conceitos, necessários, para a realização deste projeto. No entanto, é importante realçar que este trabalho é também fruto de uma expressiva investigação pessoal, visto que todos os testes implementados, quantidade e qualidade de resultados dependeria de cada aluno. De igual modo, realçar as aprendizagens obtidas no primeiro ano, como, por exemplo, a escrita de documentos em LaTeX, lecionada na UC - Laboratórios de Programação (LP), regida pelo Prof. Doutor Pedro M. Inácio, a que devo bastante reconhecimento e sou igualmente gratos.

Agradecer também aos meus colegas, e amigos, Simão Augusto Ferreira Andrade(a46852) e João Pedro Santos Paixão(a46936) com os quais pude trocar ideias e testar o meu *client*. Por fim, um especial agradecimento à minha família que está neste momento a investir e a acreditar em mim. Da minha parte um muito obrigado!

Resumo

Este projeto, consistiu em implementar as heurísticas mais completas possíveis para desenvolver a função objetivo para um jogo de Xadrez.

Contamos com o esquema das Algortimos desenvolvidos representado na tabela resumo 1.

Operações sobre:	Algoritmos
Movimento das Peças	`ameacaAtiva(board, piece, play)` - analisa todos os movimentos possíveis de cada peça; - guarda as peças ameaçadas.

Peso das Posições de cada Peça	`avaliarPosicaoBrancas(p, pos, play)`
	- devolve o peso que a posição tem no tabuleiro.
	`avaliarPosicaoPretas(p, pos)`
	- devolve o peso que a posição tem no tabuleiro.

Largura do Movimento	`weightPositions = 2e-1`
Valoração das Peças	Peças Brancas: - Torres = 10 - Cavalos = 7 - Bispos = 6 - Rainha = 100 - Rei = 9999 - Piões = 2 Peças Pretas: - Torres = 20 - Cavalos = 12 - Bispos = 12 - Rainha = 36 - Rei = 9999 - Piões = 4
Movimentos Iniciais das Peças	Táticas de Xadrez: - Abertura Ruy Lopez; - Abertura Ataque Fegatello; - Tática de Defesa Siliciana; - Tática de Defesa Caro-Kann; - Tática de Defesa Nimzoíndia.

Tabela 1: Tabela Resumo dos algoritmos implementados.

Conteúdo

Conteúdo **v**

Lista de Figuras **vii**

Lista de Tabelas **ix**

1 Introdução **1**
1.1 Âmbito e Enquadramento 1
1.2 Motivação 1
1.3 Objetivos 2
1.4 Abordagem 2
1.5 Organização do Documento 3

2 Algoritmos **5**
2.1 Introdução 5
2.2 *In pursuit of the glory heuristic* 6
2.2.1 Peso das Posições de cada Peça 6
2.2.2 Largura do Movimento 9
2.2.3 Mexer com o *return* 9
2.2.4 Mexer com a Valoração 10
2.2.5 Boas Práticas de Xadrez 10
2.3 Táticas de Xadrez 11
2.3.1 Começar com as brancas 12
2.3.2 Começar com as pretas 13
2.3.3 Defesa 1 = Defesa Siciliana + Defesa Caro-Kann 13
2.3.4 Defesa 2 = Defesa Nimzoíndia 14
2.3.5 Funções onde aplico as Táticas 14

3 Conclusões **19**
3.1 Conclusões Principais 19

Lista de Figuras

2.1 Definicão da Defesa 1. 13
2.2 Definicão da Defesa 2. 14

Lista de Tabelas

1 Tabela Resumo dos algoritmos implementados. iv

Lista de Excertos de Código

2.1 avaliarPosicaoBrancas. 6
2.2 avaliarPosicaoPretas. 7
2.3 Alteração final nas posições das Torres na função avaliarPosicaoBrancas. 8
2.4 Alteração final nas posições dos Piões na função avaliarPosicaoBrancas. 9
2.5 Largura no *client.py*. 9
2.6 Alteração feita na largura. 9
2.7 *Return* que ganhava 2x mais rápido ao *client.py* ao jogar com as pretas. 9
2.8 Retornos finais. 9
2.9 *Return* que ganhava 2x mais rápido ao Declaração da valoração de cada uma das peças if play 10
2.10 Parte do código onde dou uma benesse caso ameace o rei. . . . 11
2.11 Exemplo explicativo. 12
2.12 Exemplo explicativo. 12
2.13 Declaração das variáveis globais. 14
2.14 Função Objetivo. 14
2.15 Função que decide o movimento. 16

Palavras-Chave

Heurística, Inteligência Artificial, MiniMax, Python, Xadrez.

Acrónimos

EI Engenharia Informática

IA Inteligência Artificial

LP Laboratórios de Programação

MA Matemática Aplicada

UBI Universidade da Beira Interior

UC Unidade Curricular

Capítulo

1

Introdução

1.1 Âmbito e Enquadramento

Encontra-se a analisar um relatório, elaborado no 1º semestre do 3.º ano da Licenciatura em Engenharia Informática (EI) da Universidade da Beira Interior (UBI).

Foi confecionado no contexto da Unidade Curricular (UC) Inteligência Artificial (IA) da UBI, visa familiarizar os alunos da importância dos dados no desenvolvimento de uma heurística.

Abrange três áreas muito importantes do conhecimento, EI, IA e Matemática Aplicada (MA), pois, desenvolvemos todo um programa que, para além de implementar algoritmos, exerce todo um cálculo matemático que na ausência dessas operações lógicas, seria impossível desenvolver este trabalho.

É importante abordar estes conceitos, entender como o analógico pode ser transformado no digital, como podemos tratar os dados e criar algo exato, trivial e na direção do gradiente.

1.2 Motivação

IA é uma das áreas mais interessantes da informática, porém, a meu ver, das mais complicadas. A quantidade de dados a ser analisados, muitas vezes não reflete a qualidade dos mesmos, o cansaço e a forma exaustiva que os procuramos pode tornar-se um pesadelo.

Contudo, continuo a defender que a IA tem um papel fundamental nos dias de hoje. Atualmente, ainda não se conseguiu chegar ao que designamos "IA Forte", mas os agentes da família de IA Fraca desempenham um papel muito importante, pois são muito, muito bons, numa certa especificação.

Além disso, este trabalho foi realizado no âmbito das metas definidas pelo Prof. Doutor Hugo Pedro Proença, como gerente da UC - Inteligência Artificial, como tal considero essa a minha principal motivação.

1.3 Objetivos

Este trabalho pretende analisar, planear e implementar várias heurísticas. O seu principal objetivo, consiste em criar uma função objetivo vencedora.

Além do desenvolvimento destas funções, como objetivos secundários vitais para a nossa vida académica, este trabalho permite uma maior aptidão na pesquisa e elaboração de projetos, analise e recolha de dados, promove o trabalho diário e em aula, de forma a adquirir métodos, experiência e um maior à vontade.

1.4 Abordagem

Comecei por começar a pesquisar acerca do tema, estive meio perdido durante alguns dias, mas passado um tempo comecei a juntar "as peças"e tudo começou a fluir naturalmente.

A organização é o valor mais importante a defender, pois, na sua ausência impossibilitasse a criação de um método de trabalho funcional.

Quando dei por mim, via palestras, competições, uma série de documentários que suscitaram em mim o interesse que precisava. No entanto, nenhuma solução evidente.

Comecei por testar o *client.py* do professor, o meu objetivo era compreender o que se passava e vencer o mesmo. Não foi muito complicado, pois a função objetivo apenas priorizava a quantidade de peças no tabuleiro e o quão distantes elas iam.

Após vencer o *client.py*, o meu objetivo mudou, agora seria tentar vencer com o menor número de rondas em cada lado. Modifiquei o peso do movimento e consegui melhorar, alterei depois a valor de todas as posições no tabuleiro de cada peça e consegui reduzir o número de rondas quando jogava com as peças pretas. Foi ai que cheguei também há conclusão, que aumentar a valoração das peças diminuia as rondas na vitória das peças brancas, mas perdia com as pretas, o que fugia do objetivo.

Fui criando *backups* de funções vitoriosas e fui unindo cada ponto positivo que retinha. Fiz testes a nível do retorno das variáveis, com intuito de melhorar.

Continuei a minha pesquisa e criei um ficheiro repleto de boas práticas de Xadrez, que viria a tentar implementar uma a uma. Comecei a implementar, mas nunca chegava a uma conclusão, pois vencer ao *client.py* já não era um desafio. Falei com alguns colegas e comecei a testar o meu trabalho contra o deles, foquei-me em analisar as jogadas, uma a uma, e tentar vence-los.

Contudo, sempre que modificava a minha função para vencer a um colega, podia perder para o *client.py*, o que não era bom. Com isto, surgiram mais e mais testes, conclusões e confusões.

Crei funções que mapeiam as jogadas iniciais, de forma a aplicar técnicas de ataque e defesa, construi toda a estrutura das peças ameaçadas e ainda implementei uma função que verificava se o rei estava em *check*, mas não fazia diferença, pois a valoração já o fazia.

O meu objetivo era conseguir vencer todos, mas após analizar os dados foquei me em vencer ao *client.py* sempre, e tentar no minimo empatar com os meus colegas. Apesar de ter projetos que ganham aos colegas com que testei, elas perdem com o *client.py* ou empatam, ou seja, tendo com base que a maioria dos trabalhos são baseados no *client.py*, prefiri ganhar-lhe, pelo menos.

Em suma, considero que o meu maior problema foi a forma como analisei os dados e de não ter definido boas valorações, antes de tentar vencer aos meus colegas. A falta de boas valorações, prejudicou me na criação da função `ameacaAtiva()`, pois nenhum valor que retornava ia em conta com o valor das minhas peças.

O relatório foi o último ponto a ser concluído.

1.5 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – refere o projeto que nos foi proposto, o seu âmbito e Enquadramento, a Motivação para a sua realização, os objetivos e a forma como foram abordados;

2. O segundo capítulo – **Algoritmos** – descrição do funcionamento dos algoritmos;

3. O quarto capítulo – **Conclusões e Trabalho Futuro** – conclusão do projeto desenvolvido incluindo considerações sobre o desempenho e trabalho futuro por realizar.

Capítulo

2

Algoritmos

2.1 Introdução

Neste capítulo o objetivo é descrever a minha abordagem de uma forma mais detalhada. O capítulo divide-se do seguinte modo:

- a secção 2.2, **In pursuit of the glory heuristic**, explica todo o processo que me levou a terminar este projeto;

  Contém as suas características em cinco subsecções, subsecção 2.2.1 - **Peso das Posições de cada Peça**, subsecção 2.2.2 - **Largura do Movimento**, subsecção 2.2.3 **Mexer com o return**, subsecção 2.2.4 - **Mexer com a Valoração** e subsecção 2.2.5 - **Boas Práticas de Xadrez**.

- a secção 2.3, **Táticas de Xadrez**, explica as táticas de abertura e de defesa aplicadas;

  Contém as suas características cinco subsecções, subsecção 2.3.1 - **Começar com as brancas**, subsecção 2.3.2 - **Começar com as pretas**, subsecção 2.3.3 **Defesa 1 = Defesa Siciliana + Defesa Caro-Kann**, subsecção 2.3.4 - **Defesa 2 = Defesa Nimzoíndia** e subsecção 2.3.5 - **Funções onde aplico as Táticas**.

2.2 *In pursuit of the glory heuristic*

2.2.1 Peso das Posições de cada Peça

A primeira coisa que adicionei foi um peso a todas posição no tabuleiro a cada peça, isto foi suficiente para vencer ao *client.py*.

```
    pts_peoes_pos = [0, 0, 0, 0, 0, 0, 0, 0,
                     5, 10, 10, -20, -20, 10, 10, 5,
                     5, -5, -10, 0, 0, -10, -5, 5,
                     0, 0, 0, 20, 20, 0, 0, 0,
                     5, 5, 10, 25, 25, 10, 5, 5,
                     10, 10, 20, 30, 30, 20, 10, 10,
                     50, 50, 50, 50, 50, 50, 50, 50,
                     0, 0, 0, 0, 0, 0, 0, 0]

    pts_torres_pos = [0, 0, 0, 5, 5, 0, 0, 0,
                      -5, 0, 0, 0, 0, 0, 0, -5,
                      -5, 0, 0, 0, 0, 0, 0, -5,
                      -5, 0, 0, 0, 0, 0, 0, -5,
                      -5, 0, 0, 0, 0, 0, 0, -5,
                      -5, 0, 0, 0, 0, 0, 0, -5,
                      5, 10, 10, 10, 10, 10, 10, 5,
                      0, 0, 0, 0, 0, 0, 0, 0]

    pts_cavalos_pos = [-50, -40, -30, -30, -30, -30, -40, -50,
                       -40, -20, 0, 5, 5, 0, -20, -40,
                       -30, 5, 10, 15, 15, 10, 5, -30,
                       -30, 0, 15, 20, 20, 15, 0, -30,
                       -30, 5, 15, 20, 20, 15, 5, -30,
                       -30, 0, 10, 15, 15, 10, 0, -30,
                       -40, -20, 0, 0, 0, 0, -20, -40,
                       -50, -40, -30, -30, -30, -30, -40, -50]

    pts_bispos_pos = [-20, -10, -10, -10, -10, -10, -10, -20,
                      -10, 5, 0, 0, 0, 0, 5, -10,
                      -10, 10, 10, 10, 10, 10, 10, -10,
                      -10, 0, 10, 10, 10, 10, 0, -10,
                      -10, 5, 5, 10, 10, 5, 5, -10,
                      -10, 0, 5, 10, 10, 5, 0, -10,
                      -10, 0, 0, 0, 0, 0, 0, -10,
                      -20, -10, -10, -10, -10, -10, -10, -20]

    pts_rainha_pos = [-20, -10, -10, -5, -5, -10, -10, -20,
                      -10, 0, 0, 0, 0, 0, 0, -10,
                      -10, 5, 5, 5, 5, 5, 0, -10,
                      0, 0, 5, 5, 5, 5, 0, -5,
                      -5, 0, 5, 5, 5, 5, 0, -5,
                      -10, 0, 5, 5, 5, 5, 0, -10,
```

```
                  -10, 0, 0, 0, 0, 0, 0, -10,
                  -20, -10, -10, -5, -5, -10, -10, -20]

pts_rei_pos = [20, 30, 10, 0, 0, 10, 30, 20,
               20, 20, 0, 0, 0, 0, 20, 20,
               -10, -20, -20, -20, -20, -20, -20, -10,
               -20, -30, -30, -40, -40, -30, -30, -20,
               -30, -40, -40, -50, -50, -40, -40, -30,
               -30, -40, -40, -50, -50, -40, -40, -30,
               -30, -40, -40, -50, -50, -40, -40, -30,
               -30, -40, -40, -50, -50, -40, -40, -30]
```

Excerto de Código 2.1: avaliarPosicaoBrancas.

```
pts_peoes_pos = [0, 0, 0, 0, 0, 0, 0, 0,
                 50, 50, 50, 50, 50, 50, 50, 50,
                 10, 10, 20, 30, 30, 20, 10, 10,
                 5, 5, 10, 25, 25, 10, 5, 5,
                 0, 0, 0, 20, 20, 0, 0, 0,
                 5, -5, -10, 0, 0, -10, -5, 5,
                 5, 10, 10, -20, -20, 10, 10, 5,
                 0, 0, 0, 0, 0, 0, 0, 0]

pts_torres_pos = [0, 0, 0, 0, 0, 0, 0, 0,
                  5, 10, 10, 10, 10, 10, 10, 5,
                  -5, 0, 0, 0, 0, 0, 0, -5,
                  -5, 0, 0, 0, 0, 0, 0, -5,
                  -5, 0, 0, 0, 0, 0, 0, -5,
                  -5, 0, 0, 0, 0, 0, 0, -5,
                  -5, 0, 0, 0, 0, 0, 0, -5,
                  0, 0, 0, 5, 5, 0, 0, 0]

pts_cavalos_pos = [-50, -40, -30, -30, -30, -30, -40, -50,
                   -40, -20, 0, 0, 0, 0, -20, -40,
                   -30, 0, 10, 15, 15, 10, 0, -30,
                   -30, 5, 15, 20, 20, 15, 5, -30,
                   -30, 0, 15, 20, 20, 15, 0, -30,
                   -30, 5, 10, 15, 15, 10, 5, -30,
                   -40, -20, 0, 5, 5, 0, -20, -40,
                   -50, -40, -30, -30, -30, -30, -40, -50]

pts_bispos_pos = [-20, -10, -10, -10, -10, -10, -10, -20,
                  -10, 0, 0, 0, 0, 0, 0, -10,
                  -10, 0, 5, 10, 10, 5, 0, -10,
                  -10, 5, 5, 10, 10, 5, 5, -10,
                  -10, 0, 10, 10, 10, 10, 0, -10,
                  -10, 10, 10, 10, 10, 10, 10, -10,
                  -10, 5, 0, 0, 0, 0, 5, -10,
                  -20, -10, -10, -10, -10, -10, -10, -20]
```

```
    pts_rainha_pos = [-20, -10, -10, -5, -5, -10, -10, -20,
                      -10, 0, 0, 0, 0, 0, 0, -10,
                      -10, 0, 5, 5, 5, 5, 0, -10,
                      -5, 0, 5, 5, 5, 5, 0, -5,
                      0, 0, 5, 5, 5, 5, 0, -5,
                      -10, 5, 5, 5, 5, 5, 0, -10,
                      -10, 0, 0, 0, 0, 0, 0, -10,
                      -20, -10, -10, -5, -5, -10, -10, -20]

    pts_rei_pos = [-30, -40, -40, -50, -50, -40, -40, -30,
                   -30, -40, -40, -50, -50, -40, -40, -30,
                   -30, -40, -40, -50, -50, -40, -40, -30,
                   -30, -40, -40, -50, -50, -40, -40, -30,
                   -20, -30, -30, -40, -40, -30, -30, -20,
                   -10, -20, -20, -20, -20, -20, -20, -10,
                   20, 20, 0, 0, 0, 0, 20, 20,
                   20, 30, 10, 0, 0, 10, 30, 20]
```

Excerto de Código 2.2: avaliarPosicaoPretas.

Após vários testes, concluí que quando jogava com as peças brancas e, por regra, quem joga com as peças brancas tem tendência a ser mais ofensivo, que as minhas torres não saiam dos cantos ou ficavam presas pelo rei, que se encostava. Então, depois de inúmeros testes, esta foi a modificação que fiz, tendo em conta que se jogasse com as pretas a forma como avaliava o adversário também conta.

```
# caso seja o jogador 0 (brancas)
    if play == 0:
        pts_torres_pos = [0, 5, 0, 5, 5, 0, 5, 0,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          5, 10, 10, 10, 10, 10, 10, 5,
                          0, 0, 0, 0, 0, 0, 0, 0]
    else:
        pts_torres_pos = [0, 0, 0, 10, 10, 0, 0, 0,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          -5, 0, 0, 0, 0, 0, 0, -5,
                          5, 10, 10, 10, 10, 10, 10, 5,
                          0, 0, 0, 0, 0, 0, 0, 0]
```

Excerto de Código 2.3: Alteração final nas posições das Torres na função avaliarPosicaoBrancas.

E também alterei o valor das posições dos piões, de forma a valorizar que eles tentem chegar ao final do tabuleiro, pois em alguns testes, tive casos em que piões transformavam-se em rainhas.

```
    pts_peoes_pos = [0, 0, 0, 0, 0, 0, 0, 0,
                     5, 10, 10, -20, -20, 10, 10, 5,
                     5, -5, -10, 0, 0, -10, -5, 5,
                     0, 0, 0, 20, 20, 0, 0, 0,
                     5, 5, 10, 25, 25, 10, 5, 5,
                     10, 10, 20, 30, 30, 20, 10, 10,
                     50, 50, 50, 50, 50, 50, 50, 50,
                     60, 60, 60, 60, 60, 60, 60, 60]
```

Excerto de Código 2.4: Alteração final nas posições dos Piões na função avaliarPosicaoBrancas.

2.2.2 Largura do Movimento

Como já vencia ao *client.py*, tentei ao máximo reduzir o número de rondas que levavam à vitória, essa solução estava numa simples alteração na largura do movimento.

```
weight_positions = 1e-1
```

Excerto de Código 2.5: Largura no *client.py*.

```
weight_positions = 2e-1
```

Excerto de Código 2.6: Alteração feita na largura.

2.2.3 Mexer com o *return*

Fiz testes com os valores que a função objetivo retornava, enquanto tentava reduzir o número de rondas para ganhar ao *client.py*. Quando apenas valorizava o valor das peças pretas que estavam em jogo, ganhava ao cliente 2x mais rápido, isto jogando com as pretas.

```
return score_b - score_w + weight_positions * score_b_positions
```

Excerto de Código 2.7: *Return* que ganhava 2x mais rápido ao *client.py* ao jogar com as pretas.

Isto, obviamente, não se veio a refletir em testes seguintes e no final os meus retornos ficaram os seguintes.

```
Retornar o score para as pe as brancas
    if play == 0:
        return score_w + score_w_positions * weight_positions - score_b
            - score_b_positions * weight_positions
```

```
    # Retornar o score para as pe as pretas
    else:
        return score_b + score_b_positions * weight_positions - score_w
            - score_w_positions * weight_positions
```

Excerto de Código 2.8: Retornos finais.

2.2.4 Mexer com a Valoração

Alterei a valoração das peças centenas e centenas de vezes, sinto que meu problema foi não mexer numa fase inicial nelas. A valoração melhorava quando jogava contra colegas, mas fazia-me perder com o *client*.

Após 381 testes de valoração, eis a combinação que entendi submeter.

```
    else:
        pts = [20, 12, 12, 36, 9999, 12, 12, 20, 4, 4, 4, 4, 4, 4, 4, 4]
```

Excerto de Código 2.9: *Return* que ganhava 2x mais rápido ao Declaração da valoração de cada uma das peças if play

Aquando as brancas, esta leva-me a vencer ao *client.py* e a empatar com os meus dois colegas. Se for as pretas, ganho à mesma ao *client.py*, empato contra um, mas, infelizmente, perco contra outro.

2.2.5 Boas Práticas de Xadrez

Comecei a implementar e testar boas práticas de Xadrez, como por exemplo:

- Evitar trocar Cavalos ou Bispos por 3 Piões;
- manter os bispos na mesma linha ou coluna;
- cavalo e bispo > torre e piao;
- bispo > Cavalo > 3 pioes;
- bispo + Cavalo = torre + 1.5;
- pontuação de cada peça em cada posição;
- quanta maior o score de peças em campo, melhor;
- verificar rei em check;
- peças ameaçadas;

- mover os peões para o centro do tabuleiro, permite uma melhor movimentação das peças mais fortes;
- se for as brancas devo atacar primeiro e controlar o jogo;
- se for as pretas, devo aguardar um pouco mais;
- nunca mova a mesma peça duas vezes;
- priorizar AINDA MAIS o rei.

Muitas destas práticas não me trouxeram melhores resultados, por isso mesmo na minha função `ameacaAtiva(board, piece, play)`, eu apenas adiciono 20 caso a peça esteja a ameaçar o rei.

```
        for p in ataque:
            if p == 'E' or p == 'e':
                score += 20
            #elif p == 'D' or p == 'd':
            #    score += 70
            # bispo comer uma torre vale mais do que a torre comer um
                bispo, mas menos que a torre comer uma rainha
            # ou rei
            #elif p == 'A' or p == 'a' or p == 'H' or p == 'h':
            #    score += 40
            #elif p == 'B' or p == 'b' or p == 'F' or p == 'f' or p == '
                G' or p == 'g' or p == 'C' or p == 'c':
            #    score += 10
```

Excerto de Código 2.10: Parte do código onde dou uma benesse caso ameace o rei.

2.3 Táticas de Xadrez

Testei várias táticas para mapear as jogadas iniciais, caso jogasse com as pretas ou com as brancas. Tirei conclusões e decidi fundir duas técnicas de abertura, e formar uma para começar com as brancas e três táticas de defesa, duas delas juntas também.

Uso a profundidade a 1 no inicio, pois não é necessário pesquisar mais fundo, o minha função sabe o que fazer, caso o adversário não faça um movimento que é esperado, a variável global "`decisao`"fica com o valor zero e a profundidade retorna à normal.

2.3.1 Começar com as brancas

Juntei duas técnicas de começo de jogo, a técnica "Abertura Ruy Lopez"e a "Abertura Ataque Fegatello".

1. A "Abertura Ruy Lopez"serve para libertar o meu bispo e o meu cavalo;
2. avança com o peão do rei duas casas, `board[28] == 'm';`

```
    if ronda == 1 and board[28] == 'm' and play == 0:
        return 1000
```

Excerto de Código 2.11: Exemplo explicativo.

   Retorna um valor alto para obrigar a executar o movimento.

3. com isto ganhamos controlo do centro do tabuleiro;
4. se o adversário imitar o movimento, `board[36] == 'M';`

```
    elif ronda == 2 and board[36] == 'M' and play == 0:
        decisao = 1
```

Excerto de Código 2.12: Exemplo explicativo.

   `decisao = 1` pois a profundidade vai continuar a ser um, visto que o adversário jogou como esperávamos. Caso não tivesse jogado assim, esta lista seria quebrava aqui e a `decisao` seria zero, logo a profundidade retomava ao normal e quebrava a tática inicial aqui.

   E assim sucessivamente,

5. devolver o cavalo do rei à casa `board[21] == 'g'` para forçar o peão do adversário;
6. se ele reagir movendo o cavalo da rainha à casa `board[42] == 'B';`
7. levar o meu bispo do lado do rei para a posição `board[26] == 'f';`
8. para pressionar a lateral do rei adversário;
9. se ele trouxer o cavalo "G"para `board[45] == 'G';`
10. eu capturo o piao da rainha `board[45] == 'G'` com o meu cavalo;
11. se o adversário mover o piao da rainha para `board[35] == 'L';`
12. vou capturar o pião `board[35] == 'm';`
13. então o adversário vai comer o meu pião `board[35] == 'G';`

14. como com o meu cavalo `board[35] == 'G'`.

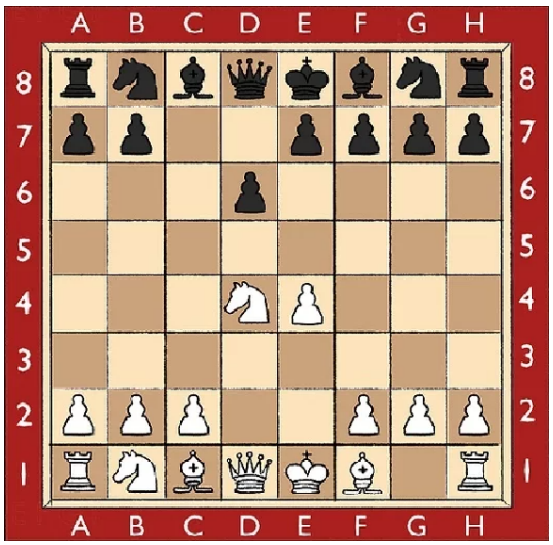
O objetivo é trazer as pelas mais fortes que estão na fila de trás, para o centro do tabuleiro. Isto, permite as peças fazer mais movimentos, bem como aplicar a técnica "ROQUE"que é uma técnica de defesa do rei. Esta encosta o rei à torre.

2.3.2 Começar com as pretas

O objetivo é jogar mais defensivo e conter as peças no começo. O esquema é idêntico ao anterior. Usei três táticas de defesa

2.3.3 Defesa 1 = Defesa Siciliana + Defesa Caro-Kann

Definições representadas na figura 2.1



Use a defesa Siciliana para retomar a ofensiva no começo do jogo. Se o jogador das brancas abrir o jogo com o peão do rei para a casa e4, mova um peão para c5 para controlar a casa d4. Seu adversário, então, responderá desenvolvendo o cavalo para f3. Mova o peão da sua rainha uma casa à frente até d6 para controlar a casa e5. Se o jogador das brancas avançar o peão dele para d4, capture-o com o peão da c5. Mesmo que seu adversário acabe capturando seu peão com o cavalo, você ainda terá um bom controle do tabuleiro.[6]

- Mesmo perdendo um peão com essa jogada, você ainda poderá mover sua rainha e seu bispo com mais facilidade se precisar utilizá-los.

Tente a Defesa Caro-Kann para montar uma estrutura reforçada de peões no fim do jogo. Como na Defesa Siciliana, se seu oponente abrir movendo o peão para e4, posicione um de seus peões em c6 para pressionar a casa d5. Se ele prosseguir movendo um peão para d4, reaja avançando um peão para d5. O jogador das brancas provavelmente moverá o cavalo para c3, fortalecendo a própria defesa, e você poderá responder capturando o peão em e4, retomando o controle do centro.[9]

- A muralha diagonal de peões ao lado da rainha ajudará a proteger suas peças no fim do jogo, além de liberar espaço para seu bispo atacar o lado do rei.

Figura 2.1: Definicão da Defesa 1.

2.3.4 Defesa 2 = Defesa Nimzoíndia

Definição representada na figura 2.1

Figura 2.2: Definicão da Defesa 2.

2.3.5 Funções onde aplico as Táticas

Primeiro declaro as variáveis globais.

```
 Vari vel usada para obrigar a que a primeira jogada seja sempre a
     mesma e manter a profundidade a 1
## de forma e evitar pesquisas desnecess rias (visto que vou come ar
    sempre com a mesma jogada)
decisao = 1

# Vari vel usada para definir qual a T tica de Defesa a usar, tendo em
     conta o primeiro movimento do advers rio
defesa = 0

# Vari vel usada para evitar que ap s entrar uma vez em determinado "
    if", n o volte a entrar
aux = 0

# Declarar variavel globar 'ronda'
ronda = 0
```

Excerto de Código 2.13: Declaração das variáveis globais.

Função objetivo.

```
def f_obj(board, play):
    # D  peso  s  jogadas, define o qu o longe a pe a se vai mover
```

```
    weight_positions = 2e-1

    # a, h = torres; b,g = cavalos; c,f =bispo; d= rainha; e = rei;
        restantes = pe es
    w = 'abcdedghijklmnop'
    b = 'ABCDEFGHIJKLMNOP'

    # Declara  o da valora  o de cada uma das pe as
    if play == 0:
        pts = [10, 7, 6, 100, 9999, 6, 7, 10, 2, 2, 2, 2, 2, 2, 2, 2]
    else:
        pts = [20, 12, 12, 36, 9999, 12, 12, 20, 4, 4, 4, 4, 4, 4, 4, 4]

    score_w = 0
    score_w_positions = 0

    global aux

    # Abertura Ruy Lopez + abertura Ataque Fegatello
    ## As pe as brancas (letras minusculas), fazem sempre as mesmas
        jogadas iniciais (mediante o advers rio)
    # defesa a 1 = Defesa Siciliana + Defesa Caro-Kann
    # defesa a 2 = Defesa Nimzo ndia
    ## As pe as pretas (letras maiusculas), fazem determinada jogada
        caso o advers rio comece de acordo com algum if
    if ronda == 1 and board[28] == 'm' and play == 0:
        return 1000
    elif ronda == 1 and board[34] == 'K' and play == 1 and defesa == 1:
        return 1000
    elif ronda == 1 and board[45] == 'G' and play == 1 and defesa == 2:
        return 1000
    elif ronda == 2 and board[21] == 'g' and play == 0:
        return 1000
    elif ronda == 2 and board[43] == 'L' and play == 1 and defesa == 1:
        return 1000
    elif ronda == 2 and board[44] == 'M' and play == 1 and defesa == 2:
        return 1000
    elif ronda == 2 and board[35] == 'L' and play == 1 and defesa == 4:
        return 1000
    elif ronda == 3 and board[26] == 'f' and play == 0:
        return 1000
    elif ronda == 3 and board[27] == 'K' and play == 1 and defesa == 1:
        return 1000
    elif ronda == 3 and board[44] == 'M' and play == 1 and defesa == 2:
        return 1000
    elif ronda == 3 and board[28] == 'L' and play == 1 and defesa == 4:
        return 1000
    elif ronda >= 4 and board[25] == 'F' and play == 1 and defesa == 2
        and aux == 1:
```

```
        aux += 1
        return 1000
    elif ronda >= 5 and board[18] == 'F' and play == 1 and defesa == 2
        and aux == 3:
        aux += 1
        return 1000
    elif ronda == 4 and board[38] == 'g' and play == 0:
        return 1000
    elif ronda == 5 and board[35] == 'm' and play == 0:
        return 1000
    elif ronda == 6 and board[53] == 'g' and play == 0:
        return 1000

    # Heur stica para as pe as brancas
    ## vai percorrer todas as posi  es do tabuleiro
    for i in range(64):
        if board[i] in w:
            # vai somar o valor de cada pe a
            score_w += pts[w.index(board[i])]
            # vai somar o valor da posi  o de cada pe a
            score_w_positions += avaliar_posicao_brancas(board[i], i,
                play)
            # somar em score_w o valor da ameaca_ativa
            score_w += ameaca_ativa(board, board[i], 0)


    score_b = 0
    score_b_positions = 0

    # Heur stica para as pe as brancas
    for i in range(64):
        if board[i] in b:
            score_b += pts[b.index(board[i])]
            score_b_positions += avaliar_posicao_pretas(board[i], i)
            score_b += ameaca_ativa(board, board[i], 1)

    # Retornar o score para as pe as brancas
    if play == 0:
        return score_w + score_w_positions * weight_positions - score_b
            - score_b_positions * weight_positions
    # Retornar o score para as pe as pretas
    else:
        return score_b + score_b_positions * weight_positions - score_w
            - score_w_positions * weight_positions
```

Excerto de Código 2.14: Função Objetivo.

Uso quando decido o movimento.

```
def decide_move(board, play, nick):
```

```
global depth_analysis_original
global ronda
global decisao
global defesa
global aux

# incrementar a cada jogada
ronda += 1

# Abertura Ruy Lopez + Abertura Ataque Fegatello
## O movimento inicial das pe as brancas     sempre o mesmo
### existe a continuidade desse movimento caso o advers rio
    responda mediante algum if
# defesa a 1 = Defesa Siciliana + Defesa Caro-Kann
# defesa a 2 = Defesa Nimzo ndia
# se for a primeira jogada do jogo e for as brancas, coloco a
    profundidade a 1 porque a jogada inicial
##     sempre a mesma
if ronda == 1 and play == 0:
    decisao = 1
#se o jogador for as brancas e na sua primeira jogada (depois de mim
    ) for colocar o pi o na posi   o 28 do tabuleiro
##entao a profundidade continua a 1 e a tatica de defesa     a 1
##e assim sucessivamente
elif ronda == 1 and board[28] == 'm' and play == 1:
    decisao = 1
    defesa = 1
elif ronda == 1 and board[27] == 'l' and play == 1:
    decisao = 1
    defesa = 2
elif ronda == 2 and board[36] == 'M' and play == 0:
    decisao = 1
elif ronda == 2 and board[31] == 'g' and play == 1 and defesa == 1:
    decisao = 1
elif ronda == 2 and board[27] == 'l' and play == 1 and defesa == 1:
    decisao = 1
    defesa = 4
elif ronda == 2 and board[26] == 'k' and play == 1 and defesa == 2:
    decisao = 1
elif ronda == 3 and board[42] == 'B' and play == 0:
    decisao = 1
elif ronda == 3 and board[27] == 'l' and play == 1 and defesa == 1:
    decisao = 1
elif ronda == 3 and play == 1 and defesa == 2:
    decisao = 1
elif ronda == 3 and board[18] == 'b' and play == 1 and defesa == 4:
    decisao = 1
elif ronda >= 4 and board[18] == 'k' and play == 1 and defesa == 2
```

```
        and aux == 0:
            decisao = 1
            aux += 1
        elif ronda >= 5 and board[16] == 'i' and board[25] == 'F' and play
            == 1 and defesa == 2 and aux == 2:
            decisao = 1
            aux += 1
        elif ronda == 4 and board[45] == 'G' and play == 0:
            decisao = 1
        elif ronda == 5 and board[35] == 'L' and play == 0:
            decisao = 1
        elif ronda == 6 and board[35] == 'G' and play == 0:
            decisao = 1
        else:
            decisao = 0

        if decisao == 1:
            depth_analysis = 1
        else:
            depth_analysis = depth_analysis_original


        states = expand_tree([board, random.random(), 0, f_obj(board, play),
             []], 0, depth_analysis,
                             play)  # [board, hash, depth, g(), f_obj(), [
                                 SUNS]]

        # show_tree(states, play, nick, 0)
        print('Total nodes in the tree: %d' % count_nodes(states))

        choice, value = minimax_alpha_beta(states, depth_analysis, play,
            True, -math.inf, math.inf)

        # print('Choose f()=%f' % value)
        # print('State_%s_' % choice[0])

        next_move = get_next_move(states, choice)

        # print('Next_%s_' % next_move[0])
        # input('Trash')

        return next_move[0]
```

Excerto de Código 2.15: Função que decide o movimento.

Capítulo

3

Conclusões

3.1 Conclusões Principais

Com todas as alterações que efetuei, a minha escolha deve-se sobretudo à analise dos dados que obtive. Analisei pelo menos 2030 JOGOS onde 40% foram VITÓRIAS, 28,5% EMPATES e 31,5% derrotas.

Todos os dados analisados ajudaram me a concluir que a sua quantidade e diversidade é importante, mas também a qualidade e o caminho percorrido, pois este último pode-nos distanciar do objetivo. Na minha opinião IA é uma ciência, mas uma ciência com vida, como tal cada decisão tem uma consequência, não tem de ser obrigatóriamente negativa, no entanto, também não se pode apostar que seja positiva, no fundo, cremos estimar que seja a melhor para a ocasião.

Com isto, pretendo passar a minha ideia acerca do trabalho, há centenas de soluções para um problema, mas essas soluções apenas se resolvem com pesquisa, trabalho, investigação, procura e , de facto, IA é isso.

Aprendi bastante com este trabalho, foi desafiador, não no sentido de "ah, há muitas soluções na *internet*", mas sim no sentido em que é pegar no problema, perceber o problema e tentar por nós mesmos, situação a situação, encontrar uma solução para o mesmo, tentar moldá-lo a nós mesmos, aprimorar. Muitas vezes é nestes pequenos trabalhos que aprendemos muita coisa, ser engenheiro é isto, nem sempre temos de fazer uma obra de arte, ou esperar que nos "batam as palma"s, mas sim resolver o problema. Não foi a melhor solução, não vai ganhar os jogos todos, nem precisa de ganhar nenhum de facto, mas deu luta.