Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

# O Gambito do Explanador: Construindo uma Engine de Xadrez Interpretável por Humanos

Felipe Zan Coelho
Orientador: Lucas Nascimento Ferreira
Proposta de pesquisa tecnológica para a disciplina Projeto Orientado à Computação 2

# 1 Introdução

O xadrez é um jogo de tabuleiro, de natureza determinística e informação perfeita. A forma moderna do xadrez surgiu na Europa no século XV, evoluída a partir do jogo indiano Chaturanga, do século VI. Desde então o xadrez se espalhou mundialmente, com regras padronizadas e uma rica história de competições.

No século XX, o xadrez tornou-se também um campo de testes para Inteligência Artificial. As primeiras tentativas de programas de xadrez datam da década de 1940. Entre os primeiros algoritmos desenvolvidos destaca-se Turochamp, projetado por Alan Turing e David Champernowne. Com o aumento do poder computacional, programas começaram a participar de torneios exclusivos para computadores a partir de 1970. Em 1981, o programa Cray Blitz derrotou um mestre com rating FIDE de 2262, atingiu rating de 22258 e tornou-se o primeiro software a atingir nível de Mestre. Um marco histórico ocorreu em 1997, quando o computador IBM Deep Blue venceu um match contra o então campeão mundial Garry Kasparov [3]. A partir de então, as engines de xadrez superaram em muito o desempenho humano, tornando-se praticamente imbatíveis até pelos melhores grandes mestres. Hoje, programas como Stockfish e Leela Chess Zero, dominam o jogo, com ratings estimados por volta 3600 Elo [4], e representam pilares fundamentais de diferentes maneiras de construção de engines [9].

Com o domínio discrepante das engines, jogadores humanos passaram a utilizá-las não somente como oponentes, mas como tutores e fonte de informação sobre os melhores movimentos. Surge então um novo desafio: a explicabilidade [1], já que a necessidade de entender a lógica subjacente às avaliações fornecidas surge naturalmente durante o treinamento em xadrez, em qualquer nível de proficiência humana. Infelizmente tais engines são ainda projetadas de modo a produzir análises, em sua maioria mediadas pelo protocolo UCI, opacas e que não fornecem uma explicação facilmente interpretável das variantes encontradas, tampouco comunica o intento geral de um movimento para o seu utilizador.

Para além disso, há diferenças fundamentais nos estilos de jogos entre humanos e computador que devem ser levadas em consideração. Tradicionalmente humanos se destacam na capacidade de avaliar posições complexas com um número extremamente reduzido de cálculos explícitos. Isto diferencia os humanos primariamente em relação a quantidade de posições efetivamente analisadas em vista que Engines modernas atingem sua força combinando busca em profundidade, costumeiramente superior a 20 meias-jogadas, e heurísticas de avaliação (primariamente manuais, porém incluindo redes neurais no caso de Leela/AlphaZero ou da avaliação de movimentos quietos em engines tradicionais).

Ademais, humanos otimizam a sua probabilidade de vitória considerando características peculiares ao seu jogo, como sensibilidade ao tempo disponível, capacidade limitada de cálculo de variantes, aversão à risco, dificuldade esperada para o adversário, proporção de sequências favoráveis e tempo computacional requerido para validação de um movimento. Tais aspectos, fundamentais à humanos, são estrangeiros às engines que simplesmente otimizam a probabilidade de vitória assumindo jogo perfeito por parte do oponente. Em função disto, é comum que mesmo grande-mestres participantes dos principais torneios, informalmente denomidados por "Super Grande-Mestres", não compreendam os movimentos recomendados pelos motores de xadrez, optando então por movimentos subótimos porém com lógica examinável. Tal fato cul-

minou recentemente na denominação informal de "jogada de computador", isto é, movimentos aparentemente inócuos que ocorrem em situações intrincadas, e geralmente distantes das peças tensionadas. Esta situação é solucionável. Categorizar padrões de movimentação e a relação deles com a árvore de busca é perfeitamente exequível e contribuiria com a produção de análises mais úteis pra jogadores humanos, podendo assim ser de extrema utilidade à comunidade global de xadrez.

Este projeto orientado à computação propõe o desenvolvimento de uma engine de xadrez forte com foco em explicabilidade, ou seja, capaz de articular o raciocínio subjacente envolvido nas escolhas de tais movimentos de maneira compreensível e útil, melhor amparando assim o corrente processo de tutoria que apresenta-se meramente improvisado sobre ferramentas desenvolvidas para outro propósito[2]. A seguir, detalhamos os fundamentos teóricos relevantes, os protocolos de comunicação envolvidos, as métricas de desempenho, metodologias de avaliação e resultados esperados.

# 2 Referencial Teórico

Esta seção estabelece os fundamentos conceituais que sustentam o desenvolvimento de uma engine de Xadrez, abrangendo os princípios clássicos de programação enxadrística. É importante destacar que abordagens neurais não serão utilizadas no módulo de busca de variantes, em função da baixa explicabilidade de redes neurais, quando removidas do campo nascente de interpretabilidade mecanicista e, em função disto não será nesta seção discutido a abordagem *Tabula Rasa* que, em 2017, superou momentaneamente os métodos baseados em pesquisa que serão discutidos abaixo[7].

## 2.1 Fundamentos de Engines de Xadrez

### 2.1.1 Modelo Inicial do Jogo

O xadrez pode ser modelado em teoria dos jogos como um jogo de dois jogadores, determinístico, de soma zero e informação perfeita. Formalmente, define-se um conjunto de estados de jogo $S$ (configurações do tabuleiro), um estado inicial $S_0$ (posição inicial das peças) e dois jogadores $\{MAX, MIN\}$ representando, por convenção, as Brancas (MAX) e pretas (MIN). Em cada estado $s \in S$, define-se:

- uma função $player(s)$ que indica o jogador que tem a vez de jogar em $s$;
- uma função $actions(s)$ que fornece o conjunto de ações (movimentos) válidas em $s$;
- uma função de transição $result(s, a)$ que retorna o novo estado após o jogador em turno executar a ação $a$ em $s$;
- um predicado $terminal\_test(s)$ que indica se $s$ é um estado terminal (fim de jogo, por xeque-mate, empate etc.);
- uma função de utilidade $utility(s, p)$ que atribui um valor numérico de resultado ao jogador $p$ no estado terminal $s$ (por exemplo, +1, 0 ou -1).

Em um jogo *soma zero*, as utilidades dos dois jogadores somam zero para qualquer resultado. Podemos definir, por exemplo, $utility(s, White) = -utility(s, Black)$ para todo estado terminal $s$. No caso do xadrez, podemos estabelecer: $utility(s, White) = 1$ (vitória das brancas), $utility(s, White) = -1$ (derrota das brancas/vitória das pretas) e $utility(s, White) = 0$ em caso de empate. Essa definição satisfaz a condição de soma zero. Ademais, apresenta informação perfeita pois em cada estado, têm conhecimento completo da configuração das peças e dos movimentos possíveis. E é um jogo determinístico, pois não há elementos aleatórios durante o jogo após a escolha das cores iniciais. Essas propriedades garantem que, teoricamente, o xadrez é um jogo *estritamente determinado* segundo a teoria dos jogos: existe um valor minimax do jogo com jogo perfeito de ambas as partes.

### 2.1.2 Busca em Árvore e Algoritmo Minimax

Considerando o modelo acima, o melhor movimento pode ser tratado como uma busca em árvore de jogo adversarial. O algoritmo clássico para jogos de soma zero é o MiniMax, que explora recursivamente os estados possíveis para encontrar a estratégia ótima [5]. Formalmente, podemos definir a função minimax de forma recursiva por indução sobre a árvore de jogo:

$$\text{Minimax}(s) = \begin{cases} utility(s, MAX), & \text{se } terminal\_test(s), \\ \max_{a \in actions(s)} \text{Minimax}(result(s, a)), & \text{se } player(s) = MAX, \\ \min_{a \in actions(s)} \text{Minimax}(result(s, a)), & \text{se } player(s) = MIN\ . \end{cases}$$

Partindo de um estado $s$, se este é terminal, a utilidade é retornada. Caso contrário, o jogador da vez escolhe a jogada que maximiza sua vantagem assumindo que o oponente, na sequência, minimizará essa vantagem. O Minimax percorre a árvore de decisões até os estados terminais, propagando os valores de utilidade durante a volta. No xadrez, a árvore completa de possíveis jogos é enorme, sendo estimada em $10^{120}$ posições diferentes por Claude Shannon [6] e $(4.822 \pm 0.028) \times 10^{44}$ por John Tromp [10]. Isto inviabiliza uma busca exaustiva até estados terminais em todas as posições. Logo, na prática, o Minimax puro é combinado com a otimização de poda $\alpha$–$\beta$ e funções de avaliação heurísticas.

### 2.1.3 Otimização de Poda Alpha Beta

A otimização poda alpha-beta do Minimax reduz drasticamente o número de nós avaliados e é o algoritmo clássico para busca em xadrez. A ideia é evitar explorar ramos da árvore de busca que não podem mais afetar a decisão final, dado que já se encontrou uma opção melhor em outra parte. O algoritmo essencialmente mantém dois valores dinâmicos durante a busca em profundidade: $\alpha$, que é o valor mínimo garantido para o jogador MAX ao longo do caminho corrente (limite inferior do valor maximizador), e $\beta$, o valor máximo garantido para o jogador MIN (limite superior do valor minimizador). Inicialmente, $\alpha = -\infty$ e $\beta = +\infty$. Conforme a busca avança pelos nós:

- Se em algum nó ocorrer que $\alpha \geq \beta$, significa que o jogador maximizador já tem uma opção

com valor $\geq \beta$ em algum ramo explorado anteriormente. Nesse caso, o ramo atual pode ser podado dado que jogadores racionais não permitirão chegar a esse nó se existe uma alternativa melhor já conhecida acima na árvore.

A poda $\alpha$–$\beta$ não altera o resultado final do Minimax (garante a mesma jogada ótima), porém aumenta a eficiência. No pior caso, se a ordenação dos movimentos for desfavorável, a poda não elimina nenhum nó e a complexidade permanece $O(b^d)$. No melhor caso, com ordenação ótima de movimentos, a complexidade melhora para $O(b^{d/2})$, efetivamente dobrando a profundidade alcançável na mesma quantidade de tempo comparado ao Minimax. Em termos de fator de ramificação efetivo, a poda $\alpha$–$\beta$ ideal reduz o fator de $b$ para aproximadamente $\sqrt{b}$. No xadrez, onde $b$ médio é cerca de 35, a busca com poda $\alpha$–$\beta$ bem ordenada pode reduzir o fator efetivo para algo em torno de 6, permitindo análises muito mais profundas pela engine.

### 2.1.4 Ordenação Heurística

Para se beneficiar da poda $\alpha$–$\beta$, é necessário a aplicação de heurísticas para ordenação de movimentos a serem analisados. Gera-se primeiro os lances mais promissores, de modo que podas ocorram mais cedo nos lances menos promissores. Várias técnicas são utilizadas para ordenação. Entre elas:

- MVV-LVA (*Most Valuable Victim, Least Valuable Attacker*): prioriza capturas em que se captura a peça de maior valor possível com a peça de menor valor.
- Movimentos de xeque ou que capturam peças sem resposta costumam ser analisados primeiro.
- Movimento do hash: se a posição atual foi vista anteriormente na busca, em uma *tabela de transposição*, com um melhor lance conhecido, esse lance é tentado primeiro.
- Killer move: lances que causaram um corte $\beta$ em um nó irmão de mesma profundidade são lembrados e tentados antes nos próximos nós na mesma profundidade.
- Entre outras.

Por meio dessas heurísticas, aproxima-se a *ordenacão ótima* dos movimentos, atingindo na prática acelerações significativas da busca.

### 2.1.5 Tablebases

As tablebases são bases de dados pré-computadas que armazenam, para posições com poucos materiais (finais com 7 peças ou menos), o resultado teórico do jogo com *jogo perfeito* de ambos os lados. Cada posição na tablebase codifica se é uma posição ganha, perdida ou empate, e normalmente quantos lances faltam para o mate, em caso de vitória, seguindo a defesa perfeita do lado adversário. Por exemplo, uma tablebase pode indicar que certa posição de rei e dama contra rei e cavalo é vitória em 23 lances com jogo perfeito. Elas são geradas por retro-análise, partindo de posições de xeque-mate triviais e recuando todos os lances possíveis. Isso garante exatidão absoluta naqueles finais. Atualmente, já foram resolvidas todas as posições com até 7

peças, incluindo ambos os reis, no tabuleiro. A razão de seu uso é para que as engines joguem perfeitamente nos finais de poucos materiais e também para melhorar a avaliação em meio-jogo, através da detecção, durante busca, de posições que conduzem a posições presentes nas tablebases. Em termos formais, podemos considerar a tablebase $T$ como uma função $T(p)$ que, dado um estado $p$ pertencente ao subconjunto de posições de final armazenadas, retorna o valor exato do jogo (por exemplo, $T(p) = \{\text{“Mate em } n''|\text{Vitória}, \text{Empate}, \text{Derrota}\}$). Quando a busca de uma engine atinge um estado $p$ presente na tablebase, ela não precisa continuar expandindo aquele ramo e ao invés disso, recupera diretamente o valor de $T(p)$ e propaga essa informação para cima na árvore. Isso efetivamente torna solucionada o estágio final do jogo.

### 2.1.6 Centipawns

Engines de xadrez habitualmente expressam a avaliação de uma posição em centésimos de peão, ou *centipawns*, que equivale a 0,01 na unidade de um peão. Sendo assim, uma avaliação de +100 centipawns equivale a +1.00 peão de vantagem. Avaliações positivas indicam vantagem das Brancas e negativas indicam vantagem das pretas. Essa métrica contínua permite quantificar pequenas diferenças posicionais. As interfaces gráficas geralmente exibem o valor já em peões, mas internamente os protocolos de comunicação com a engine usam inteiros em centipawns, pois definem a menor unidade de pontuação padronizada como 1 centipawn.

O uso de centipawns tem suas limitações. Trata-se de uma escala relativa, dependente do motor e que usualmente depende da inserção de limites práticos aos valores. Valores muito altos, como +9000, são utilizados para representar uma vitória forçada, substituindo a noção de ”mate em X”lances. Outro conceito derivado é a perda em centipawn (*centipawn loss*), que quantifica o *quanto* um lance foi inferior ao melhor lance possível segundo a engine. Se a melhor continuação numa posição daria, digamos, +0.80 e o jogador escolhe um lance que leva a +0.50, considera-se que houve uma perda de 30 centipawns naquele lance. Essa métrica é frequentemente utilizada no cálculo de precisão de jogos para um determinado jogador, pois uma perda média de centipawn baixa indica que a maioria dos lances do jogador foram próximos das escolhas ótimas do motor.

# 3 Protocolos de Comunicação Engine-Interface (UCI)

Para integrar a engine de xadrez com interfaces gráficas, é necessário um protocolo de comunicação padrão entre os dois. Historicamente, dois protocolos principais se destacam: o WinBoard (CECP) e o UCI (Universal Chess Interface). O foco aqui discutido será o UCI, pois esta é atualmente praticamente unanimidade entre engines fortes.

O protocolo UCI foi lançado em 2000 e desenvolvido por Rudolf Huber e Stefan Meyer-Kahlen, como uma alternativa ao protocolo WinBoard, então dominante [12]. Hoje, virtualmente todas as principais engines e GUIs suportam UCI, tornando-o um padrão aberto de fato para motores de xadrez. Concretamente o protocolo é baseado em texto, com a engine e a interface trocando comandos via linha de comando. Uma característica importante do UCI é ser essencialmente sem estado, isto é, a cada vez que a GUI solicita à engine que pense em uma posição, ela envia todo o contexto necessário, e a engine retorna apenas os resultados para

aquele instante.

Pelo seu design simples e antigo, o UCI delega várias responsabilidades à interface, tais como avaliação da probabilidade de vitória, livros de aberturas e manejo das tablebases. A popularidade pode ser explicada por comunicação seguindo UCI ser simples de parsear, e pela natureza sem estado do protocolo facilitar análises a partir de posições arbitrárias, com a GUI enviando somente a FEN da posição. Contudo, apesar de sua difusão, o UCI foi concebido com foco em simplicidade e universalidade, não em explicações detalhadas. Por conta disso possui limitações notáveis, oferecendo apenas mecanismos básicos do ponto de vista de explicabilidade. Os recursos que o presente projeto planeja implementar vão além do escopo do UCI e em razão disto *provavelmente* não será utilizado.

# 4 Arquitetura do Sistema

O desenvolvimento do sistema proposto seguirá uma arquitetura modular composta por:

## 4.1 Módulo de Busca

Algoritmo de busca responsável por analisar sequências de variantes encontrando àquelas que maximizam os fatores de interesse estabelecidos.

## 4.2 Sistema de Explicação

Componente responsável por traduzir as avaliações internas recebidas do Módulo de Busca em explicações estruturadas em texto e gráficos.

## 4.3 Interface CLI

Componente responsável por disponibilizar em interface CLI, as avaliações de variantes do módulo de busca conjuntamente as avaliações textuais fornecidas pelo Sistema de Explicação.

## 4.4 Interface Gráfica

Componente responsável por disponibilizar em interface gráfica, acessível via plataforma WEB, o tabuleiro de Xadrez para jogo em tempo real, e as avaliações textuais e gráficas produzidas pelo Módulo de Busca e pelo Sistema de Explicação.

# 5 Metodologia e Métricas de Avaliação

A metodologia deste trabalho envolve a implementação das seguintes métricas:

## 5.1 Cálculo do Rating Elo

Métrica para quantificar a força de jogo de uma engine ou de um humano. Calcula níveis relativos de habilidade a partir dos resultados de partidas. Cada jogador tem um rating $R$. A diferença de ratings entre dois jogadores fornece a expectativa de resultado. A expectativa de score do jogador A contra B é dada pela fórmula [11]:

$$E_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}} \ ,$$

onde $E_A$ varia de 0 a 1. Analogamente, $E_B = \frac{1}{1+10^{(R_A - R_B)/400}}$, de modo que $E_A + E_B = 1$. Como exemplo, se $R_A - R_B = 0$, então $E_A = 0,5$ (50% para cada lado). Já se $R_A - R_B = 400$, então $E_A \approx 0,91$, indicando que A venceria ∼91% dos pontos contra B em média.

Após um jogo, os ratings são ajustados com base no resultado real comparado ao esperado. Seja $S_A$ o score real obtido pelo jogador A (1 para vitória, 0 para derrota, 0.5 para empate). A fórmula de atualização do rating de A é:

$$R'_A = R_A + K \cdot (S_A - E_A) \ ,$$

onde $K$ é um fator de desenvolvimento (constante que varia dependendo da federação/regra, e valores específicos em listas de motores). Esse ajuste basicamente aumenta $R_A$ se ele superou a expectativa ($S_A > E_A$) ou diminui se foi abaixo do esperado. O $K$ controla o tamanho das mudanças. Estimaremos o ELO via torneios de bots. Há organizações que produzem listas de rating para os motores. Essas listas (CCRL, CECS, CEGT) rodam milhares de partidas entre diferentes engines para calcular ratings relativos. Se temos $n$ engines jogando entre si, podemos usar os resultados para resolver um sistema de equações para os ratings $R_1, R_2, ..., R_n$ que melhor se ajustam: para cada par $i, j$, a fração de pontos que $i$ marcou contra $j$ deve ser próxima de $E(R_i, R_j)$. Neste projeto, iremos estimar o rating da engine desenvolvida jogando partidas contra o Stockfish 17.1, que é o motor mais forte atualmente, e, possivelmente, outros motores fortes.

## 5.2 Acurácia

Podemos definir *acurácia* como o quão próximas as jogadas da nossa engine estão das jogadas que o Stockfish escolheria. Considere um conjunto de posições de teste $\mathcal{P}$ Denotemos por $a_X(p)$ o movimento que a engine em desenvolvimento $X$ escolheria na posição $p$, e por $a_{SF}(p)$ o movimento escolhido pelo Stockfish 17.1 na mesma posição $p$. Podemos então usar a proporção de coincidência de lances:

$$\text{Acurácia}_1(X) \ = \ \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \mathbf{1}\{ \, a_X(p) = a_{SF}(p) \, \} \ ,$$

## 5.3 Avaliação Humana através de Questionários

Além das métricas objetivas de performance, precisamos avaliar a explicabilidade do sistema junto a usuários humanos. Isso requer estudos de *experiência do usuário* que será feito com coleta de feedback qualitativo e quantitativo através de questionários. Precisamos também seguir diretrizes e rigor metodológico de avaliação que envolve interação humana, tais quais. [8].:

- Garantia de anonimato
- Perguntas não tendenciosas

- Escolha de escala apropriada
- Definição de objetivos claros
- Agrupamento consistente
- Teste com questionário piloto para refino

Tal etapa nos fornecerá uma medida de usabilidade da explicabilidade implementada, complementando as medidas puramente técnicas.

# 6 Cronograma

| **Etapa** | **Descrição** | **Semanas** |
|---|---|---|
| Definição de Critérios | Revisão sobre critérios utilizados por jogadores humanos para avaliar posições. | 1 e 2 |
| Desenvolvimento da Base de Avaliação | Implementação de métodos de avaliação que capturem os critérios definidos na etapa anterior. | 3 e 4 |
| Construção da Engine | Desenvolvimento da engine utilizando C++ e Python. | 5 à 9 |
| Construção das Interfaces | Desenvolvimento da interface gráfica e da CLI para interação com o sistema. | 8 e 9 |
| Coleta de Métricas | Coleta de métricas para comprovação dos resultados, antecedendo o relatório final. | 10 e 11 |
| Relatório Final | Construção do relatório e pitch final. | 11 e 12 |

Tabela 1: Planejamento das etapas do projeto

# 7 Resultados Esperados

Os resultados brutos serão interpretados à luz das hipóteses da monografia. Objetivamos que, ao final, o nosso sistema,além de forte, seja capaz de articular claramente as razões estratégicas e táticas por trás de suas recomendações, oferecendo perspectivas sobre posições complexas e reconhecendo trade-offs entre diferentes abordagens de jogo. Concretamente intencionamos verificar que a engine alcança um rating Elo competitivo ao de Grande-Mestres humanos, que usualmente assumem posição de tutores. Além disso esperamos que o feedback dos usuários aponte utilidade nas explicações. Com isso conseguiremos garantir avaliação empírica sólida do trabalho desenvolvido.

# 8 Referências Bibliográficas

# Referências

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard

Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[2] Yngvi Björnsson. Chess and explainable ai. *ICGA Journal*, 46(2):67–75, 2024.

[3] Murray Campbell, A. Joseph Hoane Jr., and Feng-Hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

[4] Computer Chess Rating Lists. CCRL 40/15 rating list – maio 2025. Disponível em: `https://computerchess.org.uk/ccrl/4040/`. Acesso em: 6 jun. 2025.

[5] Donald E. Knuth and Ronald W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[6] Claude E. Shannon. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine*, 41(314):256–275, 1950.

[7] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[8] SoundRocket. Best practices for questionnaire design, 2025. Disponível em: `https://soundrocket.com/best-practices-for-questionnaire-design/`. Acesso em: 16 mai. 2025.

[9] Dustin Tanksley and Daniel B. Hier. Findings via reverse engineering: Replication of alphago zero by crowdsourced leela zero. *European Scientific Journal*, 18(4):61–79, 2022.

[10] John Tromp. Chess position ranking, 2022. Disponível em: `https://github.com/tromp/ChessPositionRanking`. Acesso em: 16 mai. 2025.

[11] Wikipedia contributors. Elo rating system, 2025. Disponível em: `https://en.wikipedia.org/wiki/Elo_rating_system`. Acesso em: 16 mai. 2025.

[12] Wikipedia contributors. Universal chess interface, 2025. Disponível em: `https://en.wikipedia.org/wiki/Universal_Chess_Interface`. Acesso em: 16 mai. 2025.