Lisbon School of Economics & Management
Universidade de Lisboa

MESTRADO EM

MÉTODOS QUANTITATIVOS PARA A DECISÃO

ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

PROJETO ELABORADO PARA OBTENÇÃO DO GRAU DE MESTRE

ENTRE PEÇAS E LINHAS DE CÓDIGO: O JOGO DE DAMAS EM VBA

ILDA VITÓRIA BÁRBARA

ORIENTAÇÃO:

PROFESSOR DOUTOR FILIPE MANUEL GONÇALVES RODRIGUES

JULHO - 2025

AGRADECIMENTOS

A conclusão deste ciclo representa o culminar de um percurso cheio de altos e baixos, que só foi possível com o apoio de várias pessoas, às quais deixo o meu profundo agradecimento.

Em primeiro lugar, quero agradecer à minha família, pelo amor incondicional, pelo incentivo e por estarem sempre presentes quando mais precisei. Um agradecimento muito especial ao meu irmão, João Lourenço, e à minha melhor amiga, Marta Libânio, que estão sempre à distância de uma chamada e sempre prontos a ajudar-me. E, ainda mais especial, ao meu namorado, por me apoiar todos os dias, por me consolar nos momentos de ansiedade e stress, pela compreensão e por ter sido a minha maior motivação nestes dois últimos anos.

Agradeço também ao meu orientador, Professor Doutor Filipe Rodrigues, pela orientação, paciência, disponibilidade e confiança. Sem o seu acompanhamento incansável e os seus comentários pertinentes, este trabalho não teria sido possível.

Para a realização deste trabalho, contei com o contributo valioso do presidente da Federação Portuguesa de Damas, Sr. Veríssimo Dias, cuja experiência e disponibilidade foram fundamentais para esclarecer aspetos técnicos, como certas regras, e também elementos históricos da modalidade. A sua colaboração foi decisiva para garantir o rigor e a fidelidade do projeto às regras oficias das Damas Portuguesas.

RESUMO

O jogo de damas é um jogo de tabuleiro amplamente conhecido e bastante praticado em Portugal, sobretudo pelo sexo masculino. O presente Trabalho Final de Mestrado consiste no desenvolvimento de uma aplicação que simula o jogo de Damas Portuguesas entre um jogador humano e o computador, garantindo o cumprimento das regras oficiais da Federação Portuguesa de Damas e proporcionando ao utilizador uma experiência interativa e desafiante. Para tomar decisões durante as jogadas do computador, foi implementado o algoritmo Mini-Max com Poda Alfa-Beta e uma função que avalia os estados de jogo, que conjuntamente permitem identificar a jogada mais vantajosa. A aplicação proposta foi desenvolvida no Excel, recorrendo à linguagem de programação *Visual Basic for Applications*.

Foram realizados testes de desempenho, comparando a aplicação proposta com outras aplicações existentes para telemóveis, em diferentes níveis de dificuldade. Nestes testes, a aplicação desenvolvida obteve uma taxa de vitória de 85%, de empate de 6% e de derrota de 9%, evidenciando a sua eficácia.

Este trabalho contribui, assim, para a divulgação e valorização das Damas Portuguesas no meio digital, oferendo um recurso didático e acessível.

PALAVRAS-CHAVE: Jogo de Damas; Mini-Max com Poda Alfa-Beta; VBA; Aplicação Interativa; Jogadas Automáticas.

ABSTRACT

The checkers game is a well-known board game widely played in Portugal, especially by men. This Master's Final Work consists of developing an application that simulates the Portuguese Checkers game between a human player and the computer, ensuring the adherence to the official rules of the Portuguese Checkers Federation and providing the user with an interactive and challenging experience. To make decisions during the computer's moves, the Mini-Max algorithm with Alpha-Beta Pruning and a heuristic that evaluates game states, were implemented allowing the identification of the most advantageous move. The proposed application was developed in Excel, using the Visual Basic for Applications programming language.

Performance tests were conducted, comparing the proposed application with other existing mobile applications at different difficulty levels. In these tests, the application developed obtained a win rate of 85%, a draw rate of 6%, and a loss rate of 9%, demonstrating its effectiveness.

This work, therefore, contributes to the promotion and appreciation of Portuguese Checkers in the digital world, offering a didactic and accessible resource.

KEYWORDS: Checkers Game; Mini-Max Alpha-Beta Pruning; VBA; Interactive Application; Automatic Moves.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

TFM – Trabalho Final de Mestrado

FPD – Federação Portuguesa de Damas

FID – Federação Internacional de Damas

VBA – *Visual Basic for Applications*

1. INTRODUÇÃO

O jogo de damas, praticado há séculos em diferentes culturas pelo mundo inteiro, constitui não só um passatempo, mas também um campo de estudo, devido à sua natureza determinística e à complexidade do seu espaço de estados.

Segundo a Federação Internacional de Damas (FID), é difícil determinar com precisão a origem do jogo. No entanto, estima-se que a sua versão primitiva, denominada *Alquerque*, remonte a cerca de 600 a.C. e era jogada no Médio Oriente e Mediterrâneo, sendo particularmente valorizado no antigo Egito. Esse jogo foi mencionado por figuras da antiguidade, como Homero e Platão, e chegou até a Índia.

Também segundo a FID, a versão moderna surgiu por volta do século XII, provavelmente no sul da França, a partir da junção do *Alquerque* com o tabuleiro de 8x8 casas. Inicialmente, o jogo recebeu nomes como *Fierges* e *Ferses*, evoluindo depois para *Jeu De Dames* ou simplesmente *Damas*. Um dos registos escritos mais antigos sobre o jogo foi publicado em 1547 por Antonio Torquemada. Ao longo da história, diversas figuras históricas admiraram e praticaram o jogo, incluindo Cícero, Napoleão, Edgar Allan Poe, Garibaldi e Andrew Carnegie. A versão conhecida de damas com 64 quadrados surgiu na década de 1930, no Brasil. Era jogada num tabuleiro 8x8, com regras adaptadas da versão internacional, que utiliza um tabuleiro maior, de 10x10.

Este Trabalho Final de Mestrado (TFM) tem como principal objetivo o desenvolvimento de uma aplicação interativa para simular partidas de Damas Portuguesas entre um jogador humano e o computador, assegurando que as regras oficiais da Federação Portuguesa de Damas (FPD) são cumpridas e sendo competitiva para o utilizador. Pretende-se, com este projeto, não apenas oferecer uma ferramenta funcional para jogar damas no ambiente digital, mas também valorizar e divulgar esta modalidade, através de uma plataforma educativa, disponível a qualquer utilizador, que preserve a tradição e cumpra as normas oficias da modalidade em Portugal.

Para alcançar esse objetivo, optou-se por recorrer ao algoritmo Mini-Max com Poda Alfa-Beta, amplamente utilizado em jogos de dois jogadores com informação perfeita, isto é, jogos em que todos os elementos do estado de jogo (posições das peças, jogadas

possíveis ou jogadas anteriores) são totalmente observáveis por ambos os jogadores, sem qualquer informação oculta (Saffidine et al., 2012). A escolha deste algoritmo deve-se à sua capacidade de explorar eficazmente o espaço de estados do jogo e evitar analisar estados desnecessários, mantendo a qualidade das decisões tomadas. A aplicação foi desenvolvida no Excel, com recurso à linguagem de programação *Visual Basic for Applications* (VBA), permitindo criar uma interface gráfica intuitiva.

O restante TFM está organizado da seguinte forma. Na Secção 2, é apresentado um resumo dos principais algoritmos aplicados no jogo de damas, com especial destaque para o algoritmo Mini-Max com Poda Alfa-Beta, selecionado para este trabalho. A Secção 3 é dedicada à descrição detalhada do funcionamento do tabuleiro e das regras das Damas Portuguesas, nomeadamente os movimentos permitidos, as capturas e as condições de término do jogo. A Secção 4, explora o funcionamento do algoritmo implementado para as jogadas do computador, incluindo a função utilizada na avaliação dos estados do jogo. Na Secção 5, é dada uma visão geral da implementação da aplicação, com especial atenção à organização modular do código e à funcionalidade de cada componente. A Secção 6 apresenta a aplicação desenvolvida, destacando as suas funcionalidades e a forma como o utilizador interage com ela. Na Secção 7, são apresentados os resultados obtidos nos testes realizados, que comparam o desempenho da aplicação proposta com diversas aplicações de damas disponíveis para telemóvel. Por fim, a Secção 8 sumariza as conclusões principais deste trabalho, analisando os resultados alcançados, identificando limitações e propondo possíveis melhorias.

2. REVISÃO DE LITERATURA

Esta secção fornece uma breve revisão das abordagens e dos algoritmos usados no jogo de damas, destacando as principais estratégias utilizadas para explorar eficientemente o seu espaço de estados, isto é, o conjunto de todas as possíveis configurações do tabuleiro que podem ocorrer a partir do estado inicial, considerando as posições das peças de ambos os jogadores (Guerra, 2011).

Os algoritmos utilizados no jogo de damas podem ser classificados em Determinísticos e de Procura ou Adaptativos e de Aprendizagem (Sutton & Barto, 2014). A principal diferença entre eles reside na forma como tomam decisões e lidam com a incerteza associada à previsão de movimentos futuros, à avaliação de posições intermediárias e à variabilidade inerente ao comportamento do adversário (Popic et al., 2021 e Russell & Norvig, 2010).

2.1. Algoritmos Determinísticos e de Procura

Os algoritmos Determinísticos e de Procura baseiam-se em regras fixas e na exploração exaustiva do espaço de estados (Russell & Norvig, 2010). Estes algoritmos visam fornecer a melhor solução possível para um determinado estado do jogo, sem aprendizagem a partir de experiências passadas (Caexeta, 2008). Por esta razão, eles não *aprendem* nem se adaptam com o tempo.

Entre os principais métodos nesta categoria, destaca-se o Mini-Max com Poda Alfa-Beta, que explora todas as jogadas possíveis (através de uma árvore de decisão) assumindo que ambos os jogadores jogam de forma ideal (Gabel, 2019). Para auxiliar na tomada de decisão, são usadas funções de avaliação, que atribuem pontuações às jogadas, e o método de transposição de tabelas, que armazena estados (nós) previamente analisados para otimizar o tempo de cálculo (Tomaz, 2013). Um exemplo marcante da aplicação destas estratégias é o projeto *Chinook*, desenvolvido na Universidade de Alberta, em 1989 (Schaeffer et al., 1992). Esse projeto utilizou uma abordagem baseada em pesquisa Mini-Max com Poda Alfa-Beta, conjuntamente com uma função de avaliação, para determinar a qualidade de uma posição no tabuleiro (Schaeffer et al.,

1996). Além disso, o *Chinook* incorporou tabelas de transposição e aprofundamento iterativo (Schaeffer et al., 1992), permitindo uma análise progressiva e refinada das jogadas, e aumentando assim a eficiência da procura.

2.2. Algoritmos Adaptativos e de Aprendizagem

Por outro lado, os algoritmos Adaptativos e de Aprendizagem, utilizam abordagens de inteligência artificial para aprender estratégias de jogo ao longo do tempo, baseando-se em dados ou interações (Mitchell, 1997). Estes algoritmos são capazes de aplicar o conhecimento adquirido a partir de experiências anteriores e melhorar o seu desempenho com a prática (Samuel, 1959).

Entre os algoritmos mais comuns estão as redes neuronais, que aprendem a avaliar estados do tabuleiro com base em dados de jogos anteriores. Muitos utilizadores recorriam a redes neuronais para auxiliar nas aproximações de funções de avaliação, tais como o TD-GAMMON de Tesauro (Caexeta, 2008), o NeuroDraughts de Mark Lynch (Lynch, 1997) e o LS-Draughts de Neto e Julia (Neto, 2007).

Nesta classe de algoritmos destaca-se também a aprendizagem por reforço, usada em problemas como controlo de robôs e jogos de tabuleiro, incluindo o jogo de damas. Nesta técnica, o sistema de decisão aprende interagindo com um ambiente dinâmico e seleciona ações com base nas situações apresentadas, recebendo sinais de recompensa ou penalidade para alcançar os seus objetivos (Tomaz, 2013). Dos vários algoritmos existentes para solucionar o problema de aprendizagem por reforço, realça-se o método das diferenças temporais de Sutton (Caexeta, 2008), que foi aplicado ao sistema NeuroDraughts e ao LS-Draughts de Neto e Julia.

Os algoritmos genéticos são também amplamente aplicados neste contexto, pois empregam conceitos de evolução natural para otimizar funções de avaliação e estratégias de jogo. Geram múltiplas potenciais soluções e refinam-nas ao longo de diversas gerações, utilizando operações como mutação, recombinação e seleção para encontrar melhores soluções (Neto, 2007). Millington & Funge (2009) discutiram a aplicação desses algoritmos em jogos de tabuleiro, incluindo o jogo de damas, como uma abordagem eficaz para evoluir estratégias de jogo.

2.3. Algoritmo Escolhido: Mini-Max com Poda Alfa-Beta

A distinção fundamental entre os dois tipos de algoritmos apresentados nas duas secções anteriores assenta na forma como eles abordam a resolução de problemas e lidam com a incerteza associada à previsão de movimentos futuros, à avaliação de posições intermediárias e à variabilidade inerente ao comportamento do adversário. Enquanto os determinísticos avaliam exaustivamente o espaço de estados do jogo, garantindo soluções otimizadas, os adaptativos aprendem com experiências passadas, melhorando o seu desempenho ao longo do tempo.

O foco deste projeto será o algoritmo Mini-Max com Poda Alfa-Beta, devido à sua eficaz e eficiente exploração do espaço de procura do jogo de damas. Este algoritmo assegura que as jogadas escolhidas são as melhores dentro do espaço de procura analisado, contrariamente aos algoritmos adaptativos que podem gerar soluções sub-ótimas, dependendo da qualidade do treino (Knuth & Moore, 1975). Além disso, a poda reduz significativamente o número de nós avaliados, tornando o processo computacionalmente mais eficiente do que uma procura exaustiva simples (Suancha et al., 2024). Esta abordagem é mais interpretável do que os algoritmos Adaptativos e de Aprendizagem, pois permite entender claramente as decisões tomadas, o que a torna ideal para jogos como o das damas, onde o equilíbrio entre desempenho e complexidade computacional é fundamental.

3. DAMAS PORTUGUESAS

O jogo de damas é disputado entre dois jogadores, que movimentam alternadamente as suas peças num tabuleiro, com o objetivo de capturar ou bloquear todas as peças do adversário. Este jogo não depende de fatores aleatórios como a sorte, mas sim de regras e lógica, sendo por isso, um jogo determinístico. Essa particularidade torna-o num excelente desafio para a aplicação de algoritmos de decisão baseados em avaliação estratégica, permitindo que um jogador enfrente um computador. Nesta secção são descritos o tabuleiro de jogo, a disposição inicial das peças e as regras específicas das Damas Portuguesas.

3.1. Tabuleiro e Disposição das Peças

Segundo a FPD, o jogo é realizado num tabuleiro quadriculado denominado *tabuleiro de Damas*, frequentemente referido de forma simplificada apenas como *tabuleiro*. Este é constituído por 64 quadrados, organizados numa matriz de oito linhas por oito colunas 8x8, dispostos alternadamente entre claros (brancos ou castanhos claros) e escuros (pretos ou castanhos escuros).

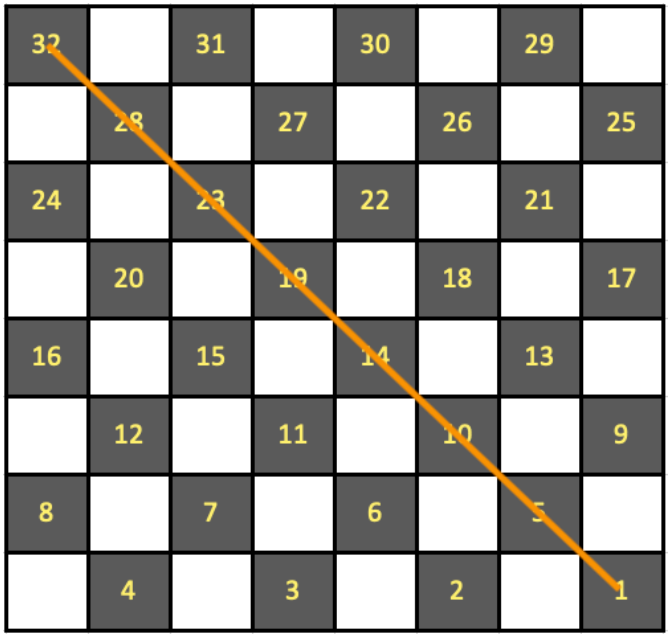

Figura 1: Tabuleiro numerado e Rio (laranja)

Os quadrados escuros, denominados casas, estão numerados de 1 a 32, iniciando-se a contagem na primeira casa inferior direita e prosseguindo da direita para a esquerda, em sentido ascendente. A numeração começa sempre no lado do jogador que possui as peças brancas, o que, conforme indicado na Figura 1, significa que esse jogador se posiciona na parte inferior do tabuleiro. A diagonal que liga as casas 1 e 32, denomina-se *rio*, conforme apresentado na Figura 1 a cor de laranja.

No início do jogo, existem no total 24 peças, todas elas peões, que são colocadas exclusivamente sobre os quadrados escuros. Cada um dos jogadores dispõe de 12 peões: o jogador com os peões brancos (ou castanhos claros) posiciona-os nas casas 1 a 12, inclusive, e o seu adversário, com os peões negros (ou castanhos escuros), posiciona-os nas casas 21 a 32, inclusive, tal como representado na Figura 2.

O jogo inicia-se com o lance (jogada) das peças brancas, atribuídas por sorteio a um dos jogadores, seguindo-se de jogadas alternadas entre as duas cores até ao fim do jogo.

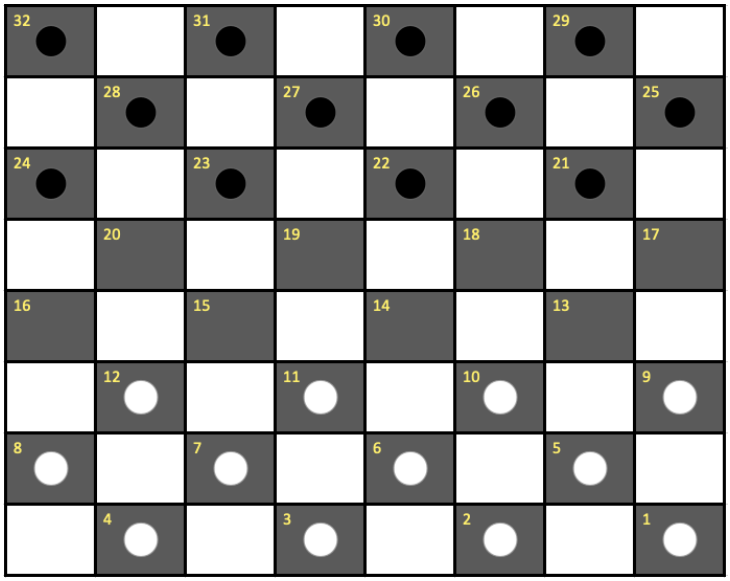

Figura 2: Posição inicial das peças

3.2. Promoção

Sempre que um peão atinge uma das quatro casas da última linha do tabuleiro no lado do adversário, casas 1 a 4 no caso das peças pretas ou casas 29 a 32 no caso das

peças brancas, o jogador adversário coloca sobre esse peão uma peça capturada anteriormente, coroando-o numa dama. Se, no momento da coroação, não houver peça disponível para a representar, o peão assume, ainda assim, o estatuto e as funções de dama, até que seja coroado. Importa referir que a promoção de um peão a dama não é considerada um lance. Além disso, a peça promovida só poderá ser movimentada após a jogada do adversário.

3.3. Movimentação das Peças

Ambos os tipos de peças (peão ou dama) movem-se sobre as casas (quadrados escuros), numa das duas diagonais e têm dois tipos de movimentos: sem captura e com captura. As diferenças principais entre o peão e a dama são o sentido em que se movimentam e o número de casas que avançam. O peão apenas se move para a frente, enquanto a dama se pode mover tanto para a frente como para trás. Em movimentos sem captura, o peão avança uma casa de cada vez e a dama pode avançar o número de casas que quiser.

3.3.1. Movimentos Sem Captura

O peão desloca-se ocupando uma casa vazia imediatamente à sua frente, como ilustrado na Figura 3.

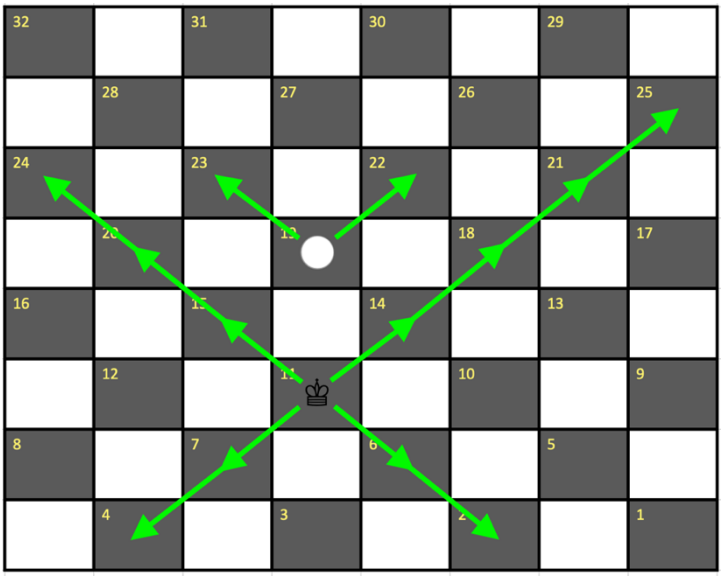

Figura 3: Movimento de peão e dama, sem captura

A dama percorre as casas vagas que quiser, desde que o caminho esteja livre, isto é, sem peças suas ou do adversário pelo meio, como exemplificado na Figura 3.

3.3.2. Movimentos Com Captura

3.3.2.1. Simples

Sempre que um peão, ao iniciar o seu movimento, tenha na casa diagonal à sua frente uma peça adversária (peão ou dama) e a seguir houver uma casa vazia, ele salta por cima dessa peça e captura-a, ocupando essa casa vaga. Isto é exemplificado na Figura 4, o peão branco na casa 14 tem à sua frente um peão preto do adversário, na casa 18, logo pode saltar sobre a peça do adversário e ocupar a casa 21, que se encontra vaga.

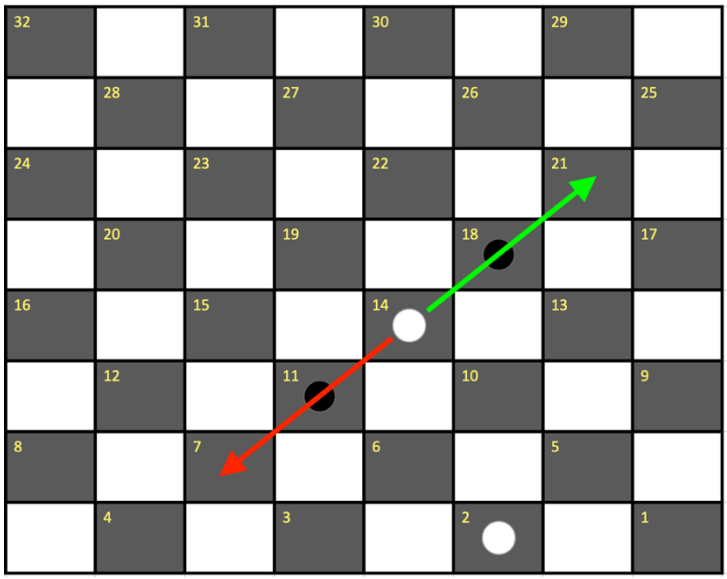

Figura 4: Movimento de peão com captura simples
(Movimento verde – Correto; Movimento vermelho – Errado)

Por outro lado, se a dama tiver numa das suas diagonais uma peça adversária, mesmo que não esteja numa casa imediatamente a seguir à sua, e houver uma ou mais casas vazias imediatamente a seguir à peça adversária, ela pode saltar por cima dessa peça, capturando-a, e pousar em qualquer uma dessas casas vagas (Figura 5). A dama pode capturar tanto para a frente como para trás (retro-captura). Como ilustrado na Figura 5, a dama na casa

19 pode capturar, para a frente, uma das peças do adversário nas casas 23 ou 26, ou, para trás, uma das peças do adversário nas casas 15 ou 10.

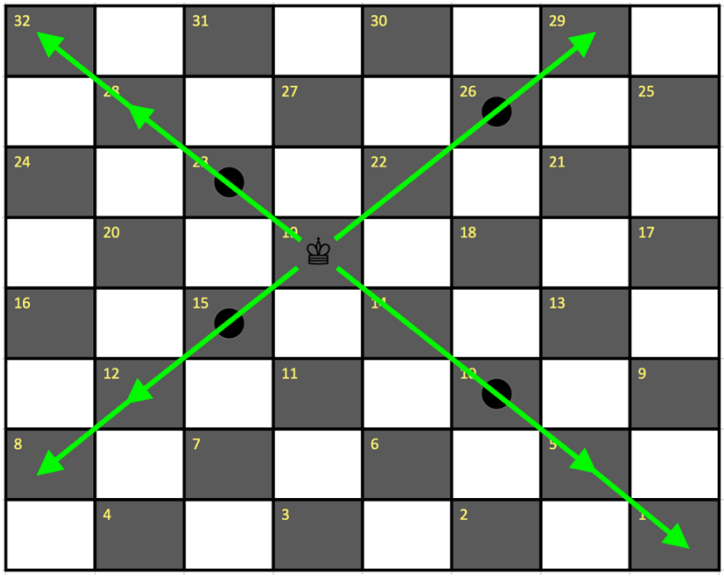

Figura 5: Movimento de dama com captura simples

3.3.2.2. Múltipla

Quando uma peça realiza capturas múltiplas, isto é, captura mais do que uma peça num mesmo lance, todas as peças capturadas são retiradas do tabuleiro após o fim do movimento. Tanto o peão como a dama podem seguir um movimento na mesma diagonal (captura linear) ou mudar para uma diagonal perpendicular (captura ortogonal), desde que escolham o percurso que lhes permita capturar o maior número possível de peças, conforme a Lei da Quantidade (ver Secção 3.4) e a Lei da Qualidade (ver Secção 3.4).

Se, após uma captura simples, um peão pousar numa casa onde possa realizar nova captura, ou seja, se houver outra peça adversária seguida de uma casa vaga, ele continuará a capturar, até que não tenha mais nenhuma peça à sua frente. Tal como se pode ver na Figura 6, o peão branco na casa 2 salta para a 9, de seguida para a 18 e por fim para a 25. Por outro lado, o peão na casa 8 salta para a 15, de seguida para a 22 e por fim para a 29.

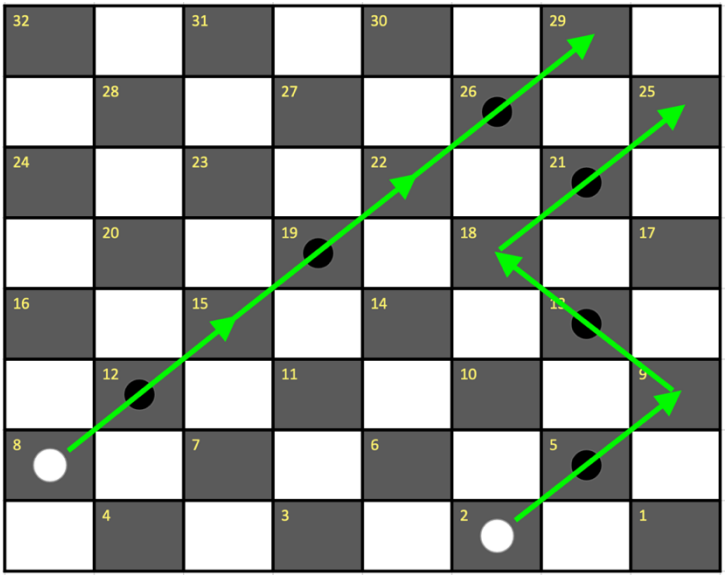

Figura 6: Movimento de peão com captura múltipla

Se, após capturar uma peça na sua diagonal, a dama encontrar outras peças que possa capturar, deve continuar a sua trajetória (Figura 7). A dama não pode saltar por cima de peças da sua própria cor, nem sobre peças adversárias colocadas em casas consecutivas. Também não pode passar duas vezes pela mesma peça durante o mesmo lance. No entanto, pode passar mais do que uma vez por casas vazias e por mais do que uma casa vazia.

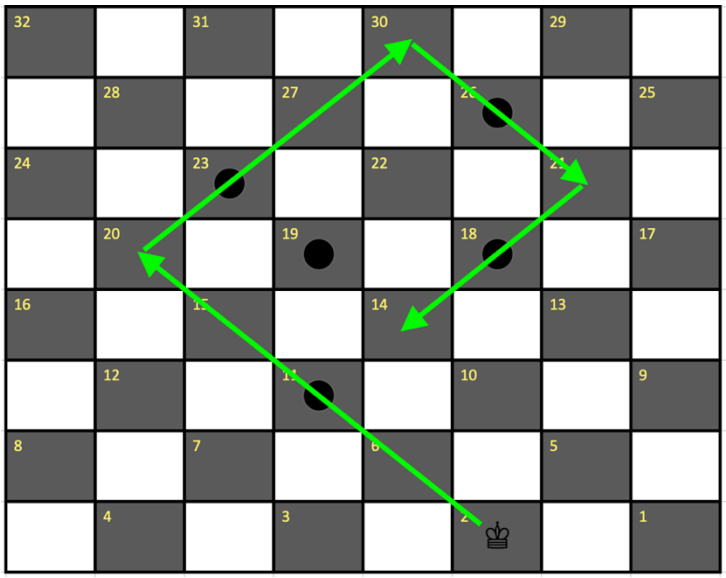

Figura 7: Movimento de dama com captura múltipla

3.4. Leis de Captura

Na captura de peças existem três leis que se aplicam tanto ao movimento de peões como de damas:

a) **Lei de Obrigatoriedade –** A captura é obrigatória, isto é, sempre que houver vários movimentos possíveis e pelo menos um deles possibilitar a captura de peças do adversário, o jogador deverá optar por esse movimento.

b) **Lei da Quantidade –** Caso existam várias possibilidades de captura, é obrigatório escolher a jogada que permita capturar o maior número possível de peças. Por exemplo, na Figura 8, o jogador tem de optar por um dos movimentos marcados a verde, pois em cada um deles captura três peões do adversário, ao contrário do movimento a vermelho que apenas captura um peão.

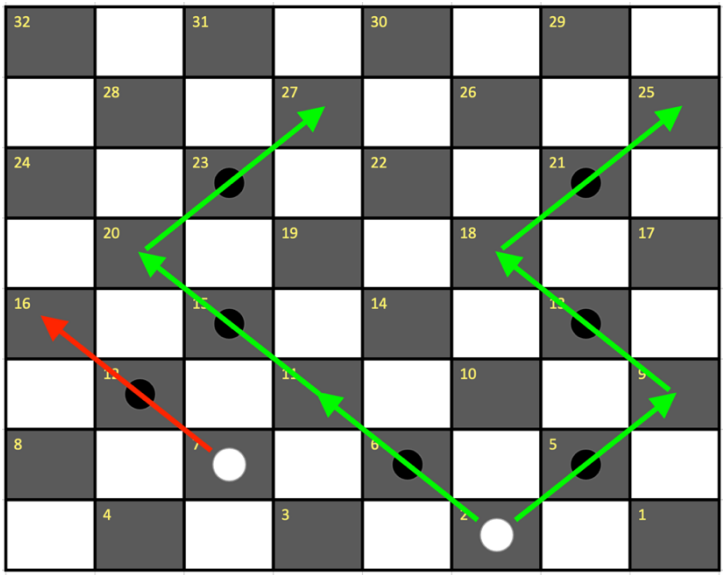

Figura 8: Lei da Quantidade
Movimentos verde – Corretos; Movimento vermelho – Errado

c) **Lei da Qualidade –** Em situações em que existam várias possibilidades de captura com o mesmo número de peças, é obrigatório escolher a hipótese que permita capturar as peças de maior valor. Para este efeito, considera-se que a dama tem um valor superior ao do peão, visto que a dama vale por duas peças. A Figura 9 ilustra esse princípio: o movimento a verde aparenta capturar três peças, mas como inclui uma dama no percurso, a contagem real é de quatro peças. Por outro lado, o movimento a vermelho captura apenas três peças, todas elas peões.

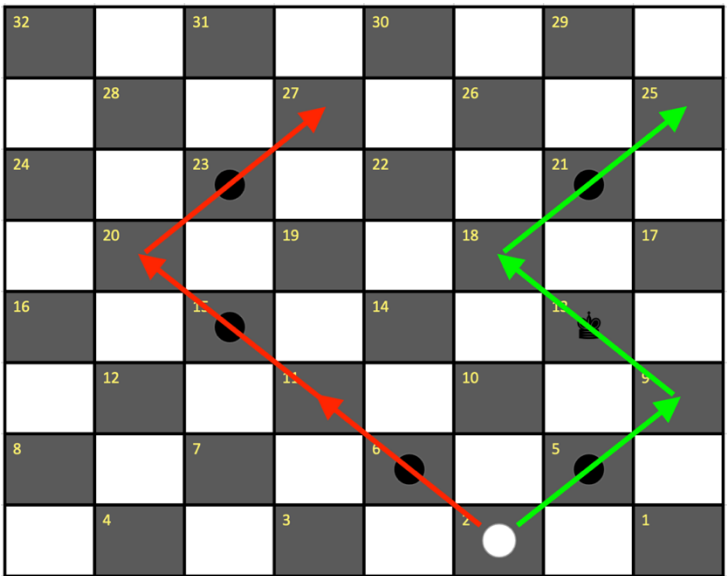

Figura 9: Lei da Qualidade
Movimento verde – Correto; Movimento vermelho – Errado

3.5. Limitação de Lances

Existem duas regras que visam evitar a duração excessiva do jogo, estabelecendo limites ao número de lances possíveis na tentativa de alcançar a vitória. Assim, o jogo deve ser considerado empatado sempre que se verifique uma das seguintes condições:

Regra dos 20 lances: Quando ambos os jogadores realizarem, no total, 20 lances de damas, isto é, 10 lances de damas por jogador, de forma consecutiva, sem que ocorra qualquer captura de peças ou movimentação de peões, a partida é considerada empatada. A contagem reinicia-se sempre que seja capturada uma peça ou um peão seja movimentado.

Regra dos 12 lances ou Forçada: Quando nenhum dos jogadores possui peões e restam somente quatro damas no tabuleiro, tendo o jogador dominante três damas (com, pelo menos, uma delas posicionada no rio), e o adversário apenas uma - posição técnica denominada *Forçada* - podem ser realizados 12 lances. A contagem desses lances inicia-se imediatamente após o movimento em que uma das três damas do jogador dominante ocupa o rio, conforme ilustrado na Figura 10. As seguintes situações podem ocorrer:

a) Se, no decorrer dos 12 lances, o jogador com uma dama capturar damas do jogador dominante, deixando de haver a configuração de três contra um, esta regra deixa de se aplicar. Neste caso, passa a vigorar exclusivamente a regra dos 20 lances.

b) Se, ao fim dos 12 lances, o jogador dominante não tiver capturado a dama do adversário, a partida será considerada empatada.

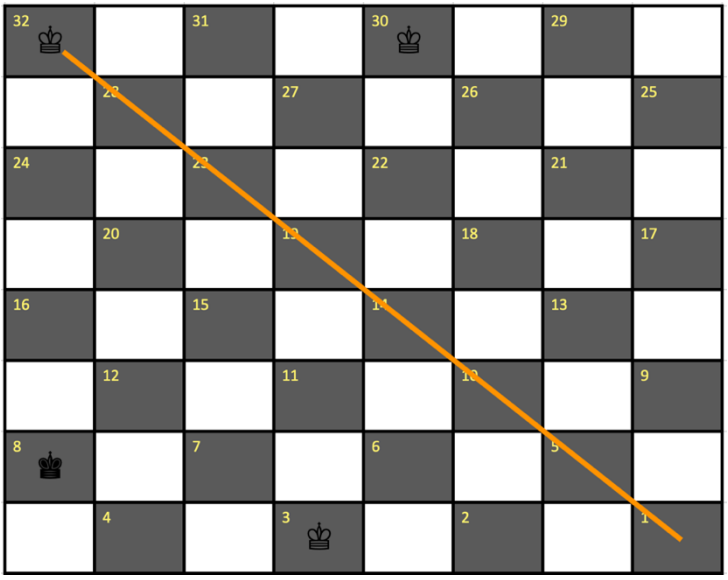

Figura 10: Regra dos 12 lances – Forçada, Rio (laranja)

3.6. Término do Jogo

O jogo de damas pode acabar com a vitória de um jogador e a derrota do outro ou com um empate entre ambos. A partida termina com vitória para um dos jogadores e derrota para o outro quando se verificar uma das seguintes situações:

a) Um jogador captura todas as peças do adversário;

b) Um jogador impede o adversário de mover qualquer peça, por estarem todas bloqueadas;

c) Um jogador declara o abandono da partida.

Por outro lado, a partida é considerada empatada quando se atinge o número de lances estabelecido pelas regras de limitação de lances, apresentadas na Secção 3.5.

4. ALGORITMOS USADOS

Nesta secção serão descritos os algoritmos utilizados para o desenvolvimento do jogo de damas, mais precisamente aqueles que são aplicados durante a jogada efetuada pelo computador. Primeiramente, será detalhado o algoritmo Mini-Max. De seguida, será introduzida uma versão otimizada do mesmo, denominada algoritmo Mini-Max com Poda Alfa-Beta, sendo esta a que foi implementada neste projeto. Por fim, será apresentada e explicada a função usada internamente neste algoritmo para avaliar estados do jogo.

4.1. Algoritmo Mini-Max

O algoritmo Mini-Max é uma técnica clássica utilizada em jogos de dois jogadores, como o xadrez ou o jogo de damas (Guerra, 2011), cujo objetivo é encontrar a melhor jogada assumindo que o adversário também joga de forma ótima (Youru et al., 2015). O seu funcionamento baseia-se numa abordagem recursiva, em que os jogadores alternam entre MAX e MIN (Nasa et al., 2018). O jogador MAX (*Computador*) tenta maximizar os ganhos da sua jogada, e o jogador MIN (*Humano*) tenta minimizar os ganhos do adversário.

Cada vez que o computador é chamado a jogar, é construída uma árvore de decisão com profundidade *p* predefinida, baseada no estado atual do tabuleiro. A raiz representa o estado atual do jogo, enquanto os restantes nós representam estados de jogo que dela possam resultar. A Figura 11 ilustra esse processo para uma árvore com profundidade 2. Na profundidade 0 está o estado atual do tabuleiro, a partir desse estado inicial, são identificados todos os possíveis movimentos que o computador pode fazer nesse turno. O jogador MAX (*Computador*), origina os estados de tabuleiro A, B e C, localizados na profundidade 1. Em seguida, o jogado MIN (*Humano*) responde a cada um desses estados com os seus movimentos possíveis, dando origem aos estados terminais, que se encontram na profundidade 2, que serão então avaliados pela função de avaliação.

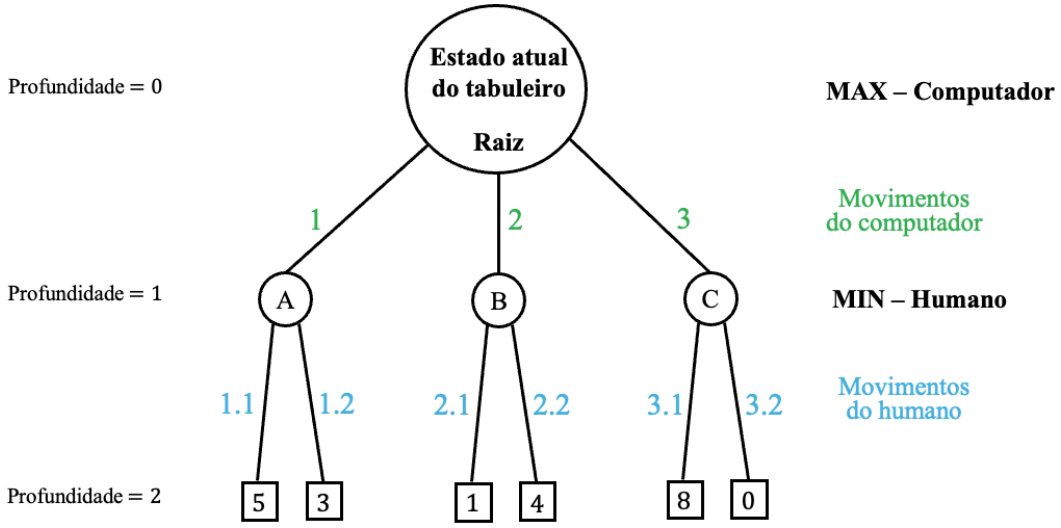

Figura 11: Exemplo de uma árvore com profundidade 2

Cada nível da árvore representa um jogador e é inicializado com a pior pontuação possível para o jogador que representa: $-\infty$ para o jogador MAX e $+\infty$ para o jogador MIN, assegurando que qualquer valor de avaliação encontrado será, respetivamente, um aumento ou uma redução face ao valor inicial. A análise inicia-se pelas posições terminais (folhas da árvore), as quais correspondem a possíveis estados de jogo após *p* jogadas. Cada um desses estados é avaliado por uma função de avaliação, resultando num valor para cada estado. Esses valores de avaliação são depois propagados, ao longo dos ramos da árvore até à raiz, aplicando alternadamente operações de maximização (quando é a vez do jogador MAX) e minimização (quando é a vez do jogador MIN). Concluído esse processo, o valor referente ao nó raiz permitirá então identificar a jogada mais vantajosa para o jogador que inicia a jogada, o computador (Guerra, 2011).

Para melhor perceber o funcionamento do algoritmo Mini-Max considere-se o exemplo apresentado na Figura 12. Nesse exemplo, é apresentada uma árvore de decisão com profundidade 3, ou seja, cada jogada do computador será avaliada tendo em conta tudo o que pode acontecer nas 2 jogadas seguintes. A raiz da árvore e os nós na profundidade 2 irão corresponder a estados do tabuleiro anteriores à jogada do jogador MAX (*Computador*). Os nós na profundidade 1 irão corresponder a estados do tabuleiro anteriores à jogada do jogador MIN (*Humano*).

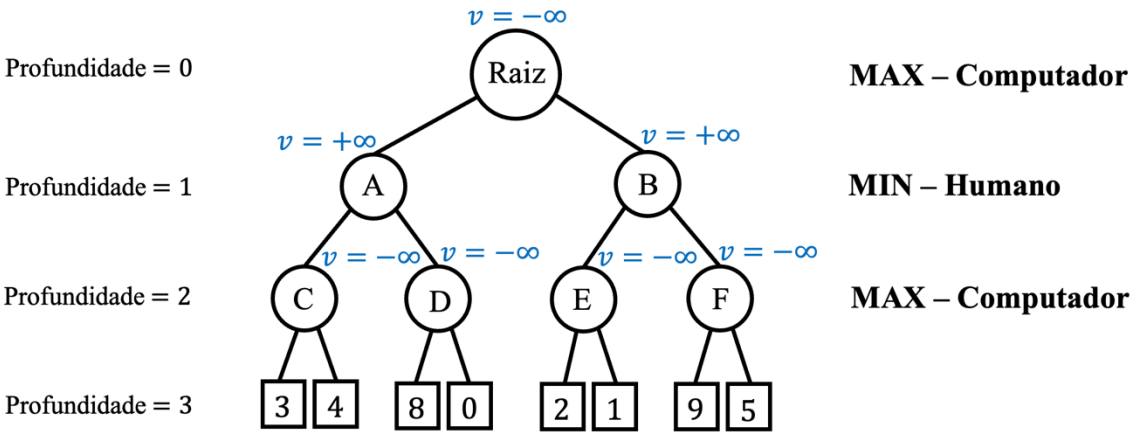

Figura 12: Árvore de decisão inicial

Na Figura 12, observa-se uma árvore composta por quatro níveis: a raiz (profundidade 0), 6 nós intermédios (2 com profundidade 1 – A e B e 4 com profundidade 2 – C, D, E e F) e os nós terminais ou folhas da árvore (profundidade 3), com valores de avaliação 3, 4, 8, 0, 2, 1, 9, 5.

Nos níveis referentes ao jogador MAX (*Computador*), cada nó escolhe o maior valor entre o seu valor atual e o dos seus filhos, procurando maximizar a pontuação. Nos níveis correspondentes ao jogador MIN (*Humano*), cada nó escolhe o menor valor entre o seu valor atual e o dos seus filhos, de forma a minimizar a vantagem do adversário.

O algoritmo inicia-se pela raiz, que representa o jogador MAX (*Computador*). De seguida, analisa os seus filhos (MIN), nós A e B. Fixando inicialmente o nó A, o algoritmo desce na árvore até alcançar os nós folha, ou seja, os que têm valores de avaliação atribuídos. Assim sendo, a execução prossegue da seguinte forma: Raiz → A → C: $\max(-\infty, 3) = 3$, $\max(3, 4) = 4$; C retorna 4 → A: $\min(+\infty, 4) = 4$ → D: $\max(-\infty, 8) = 8, \max(8, 0) = 8$; D retorna 8 → A: $\min(4, 8) = 4$; A retorna 4 → Raiz: $\max(-\infty, 4) = 4$ → B → E: $\max(-\infty, 2) = 2$, $\max(2, 1) = 2$; E retorna 2 → B: $\min(+\infty, 2) = 2$ → F: $\max(-\infty, 9) = 9, \max(9, 5) = 9$; F retorna 9 → B: $\min(2, 9) = 2$; B retorna 2 → Raiz: $\max(4, 2) = 4$; Raiz retorna 4, como ilustrado na Figura 13. Assim, o computador escolherá a jogada correspondente ao nó A, pois é aquela que maximiza o seu valor de avaliação.

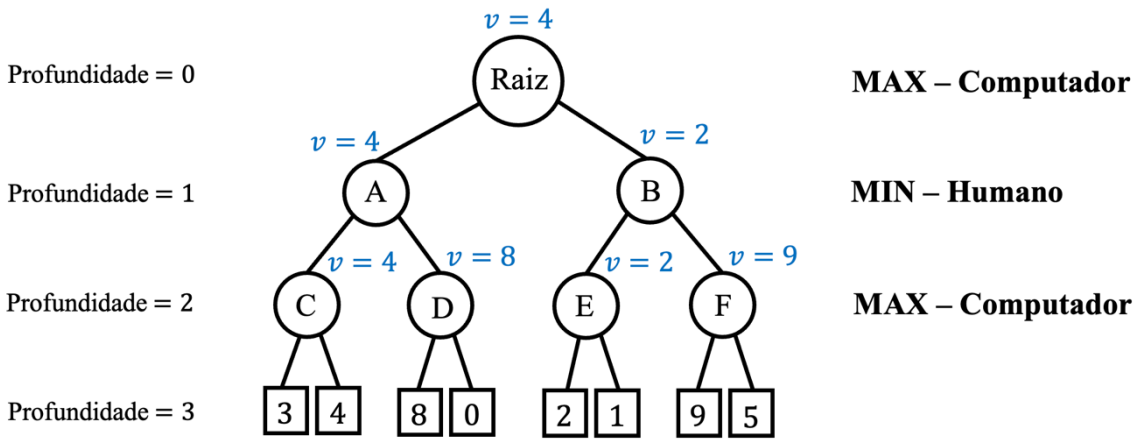

Figura 13: Resultado do Mini-Max simples

Importa referir que a árvore de jogo apresentada neste exemplo é intencionalmente simplificada**,** servindo apenas para ilustrar de forma clara o funcionamento do algoritmo. Na prática, especialmente em jogos como o de damas, a árvore de decisão é significativamente mais extensa, com milhares de possíveis estados a considerar em cada jogada. Além disso, os valores de avaliação atribuídos às folhas neste exemplo foram definidos arbitrariamente, com o objetivo de facilitar a compreensão. No entanto, em situações reais de jogo, esses valores são geralmente calculados com base em funções de avaliação específicas**,** definidas de acordo com a lógica estratégica do jogo (Hoang, 2022), conforme veremos na Secção 4.3.

4.2. Algoritmo Mini-Max com Poda Alfa-Beta

O algoritmo Mini-Max com Poda Alfa-Beta é uma versão otimizada do algoritmo Mini-Max tradicional, também ele recursivo. Esta versão otimizada visa reduzir o número de nós avaliados na árvore de decisão, eliminado ramos que não influenciarão a decisão final, ou seja, permite obter o mesmo resultado que seria obtido com o algoritmo Mini-Max (Hoang, 2022), mas de forma mais eficiente, poupando tempo e recursos de processamento (Nasa et al., 2018). Para tal, são introduzidos dois parâmetros de controlo: alfa (α) e beta (β), que representam, respetivamente (Jofanda & Yasin, 2021):

- **Alfa –** a melhor pontuação (mais alta) que o jogador MAX (*Computador*) pode garantir até ao momento;
- **Beta –** a melhor pontuação (mais baixa) que o jogador MIN (*Humano*) pode garantir até ao momento.

Tal como no algoritmo Mini-Max, os nós da árvore referentes a cada jogador começam com os piores valores possíveis: $-\infty$ para o jogador MAX e $+\infty$ para o jogador MIN. Adicionalmente, no nó raiz da árvore, α é iniciado com o valor $-\infty$ e β com $+\infty$, sendo eles propagados para os nós filhos durante a análise da árvore. Esta configuração inicial permite que os valores sejam progressivamente comparados e ajustados ao longo da exploração da árvore (Nasa et al., 2018). A atualização é feita da seguinte forma: nos nós do jogador MAX (*Computador*), o valor de avaliação é comparado com o atual α, e ambos são atualizados sempre que se encontra um valor superior ao atual; nos nós do jogador MIN (*Humano*), o valor de avaliação é comparado com o atual β, e ambos são atualizados quando se encontra um valor inferior ao atual (Nasa et al., 2018).

Durante o processo de exploração da árvore, se o algoritmo deteta que não é possível obter um resultado mais favorável para o computador do que aquele que já foi encontrado, interrompe a análise desse ramo, o que se designa por poda (Suancha et al., 2024). Existem dois tipos de poda, consoante o tipo de nó que está a ser explorado:

- **Poda Alfa:** ocorre quando o algoritmo está num nó MAX (jogador *Computador*), se $\beta \leq \alpha$. Neste caso, o jogador MIN já tem garantida uma opção mais vantajosa, pelo que o jogador MAX não precisa de continuar a explorar esse ramo, pois não conseguirá obter um resultado melhor.
- **Poda Beta:** ocorre quando o algoritmo está num nó MIN (jogador *Humano*), se $\beta \leq \alpha$. Neste cenário, o jogador MAX já possui uma jogada com valor superior, pelo que o jogador MIN não poderá melhorar a decisão, e a exploração desse ramo é interrompida.

A Figura 14 ilustra os dois tipos de poda. A árvore parcial à esquerda representa um caso de poda Alfa. O nó B é controlado pelo jogador MAX e é filho do nó A, que é controlado pelo jogador MIN. Durante a exploração do nó B, o jogador MAX encontra o valor $\alpha = 6$ como melhor valor até ao momento. No entanto, o jogador MIN, que visa

minimizar o valor de avaliação do nó, já possui uma alternativa com valor $\beta = 2$. Assim, no nó B, como $\alpha = 6$ e $\beta = 2$, verifica-se a condição de poda $\beta \leq \alpha$ e ocorre poda Alfa. Esta poda ocorre porque, sendo B um nó controlado pelo jogador MAX, qualquer filho adicional explorado só poderia aumentar o valor de α (acima de 6). No entanto, o seu pai, nó A, controlado pelo jogador MIN, nunca aceitaria um valor do nó B, pois este resultaria num valor igual ou superior a 6 e o nó A já tem garantido o valor 2, que é melhor (menor). Portanto, a continuação da exploração dos filhos do nó B é desnecessária, e os ramos seguintes são descartados (podados).

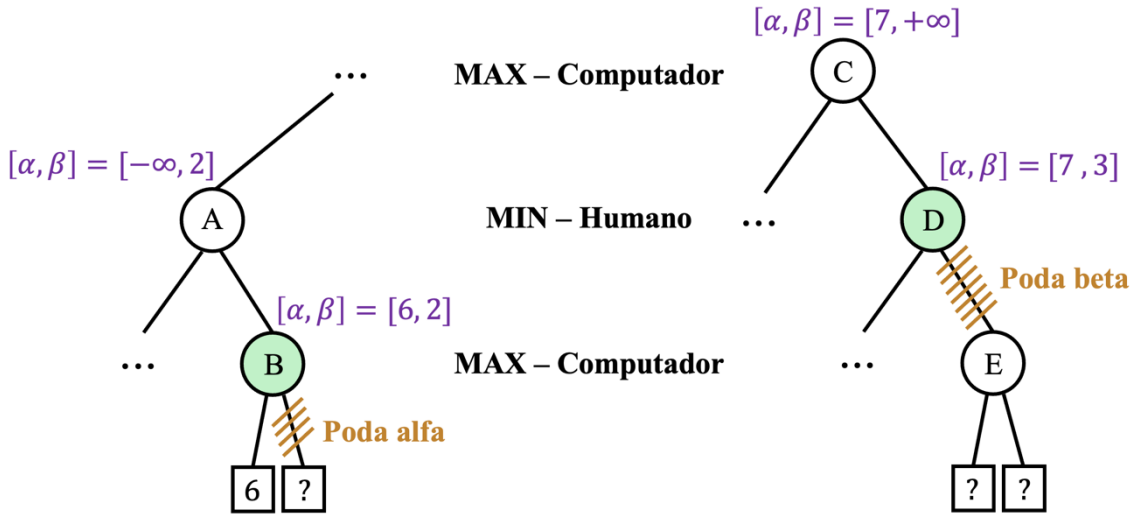

Figura 14: Exemplo de poda Alfa e poda Beta

A árvore parcial da Figura 14 à direita representa um caso de poda Beta. O nó D é controlado pelo jogador MIN e é filho do nó C, controlado pelo jogador MAX. Durante a exploração do nó D, o jogador MIN encontra o valor $\beta = 3$. No entanto, no nó C, o jogador MAX, que visa maximizar o valor de avaliação do nó, já possui o valor $\alpha = 7$. Logo, no nó D, temos $\alpha = 7$ e $\beta = 3$, pelo que se verifica a condição de poda, ocorrendo uma poda Beta. Esta poda ocorre porque, sendo o nó D controlado pelo jogador MIN, qualquer valor retornado pelos seus filhos apenas poderia diminuir o valor de β. Contudo, como o jogador Max, no nó C, já possui o valor 7, nunca optará pelo valor que o nó D lhe oferece, pois será sempre igual ou inferior a 3 e, consequentemente, inferior a 7. Dessa forma, a exploração dos restantes filhos do nó D é interrompida (podada).

O uso de poda traduz-se numa redução significativa do número de iterações, permitindo que o algoritmo analise árvores com maior profundidade, aumentando assim a qualidade da decisão (Hoang, 2022).

Para melhor ilustrar o funcionamento do algoritmo Mini-Max com Poda Alfa-Beta consideremos o exemplo da árvore de decisão apresentada na Figura 12. Neste exemplo os valores de avaliação nos nós folha serão apenas revelados quando, e se, necessário.

Inicialmente, os valores de avaliação (v) de cada jogador são definidos como descrito anteriormente. No nó raiz, os parâmetros de controlo α e β são inicializados com os valores $-\infty$ e $+\infty$, respetivamente. Estes valores (α, β) são propagados ao longo da árvore para os primeiros nós intermédios analisados que não possuem valor de avaliação por não serem folhas. Neste exemplo, são propagados para o nó A e posteriormente, para o nó C, conforme ilustrado na Figura 15.

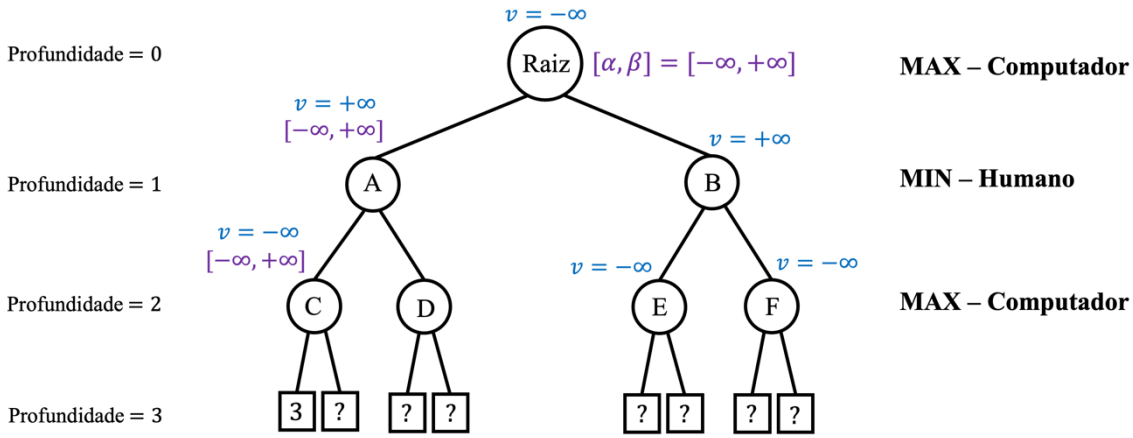

Figura 15: Inicialização do Mini-Max com Poda Alfa-Beta

O nó C tem valor de avaliação 3 e é referente ao jogador MAX (*Computador*). Assim, o seu objetivo é maximizar o valor da sua avaliação, pelo que quer esse valor quer α são atualizados para 3, pois $3 > -\infty$ (Figura 16).

Para determinar se deve continuar a explorar o nó C, o algoritmo verifica a validade da condição $\beta \leq \alpha$. Neste caso, a condição não é satisfeita, pois β mantém o valor $+\infty$ e $\alpha = 3$. Assim sendo, o nó C prossegue para avaliar o seu filho da direita, que possui um valor de avaliação 4. Como 4 é superior a 3, o valor de α e o valor de avaliação do nó C

são atualizados. O valor final atribuído ao nó C é, portanto, 4, que é então devolvido ao seu nó pai A. Este, sendo um nó do jogador MIN, atualiza não só o seu valor atual como também o valor de β para 4, assegurando que não se obterá um valor superior a 4 com o estado de tabuleiro representado pelo nó A, conforme ilustrado na Figura 16.

Neste momento, o valor de β no nó A já não é o inicial ($+\infty$), foi atualizado para 4, e α continua a ser $-\infty$. O nó A, sendo pai do nó D, transmite-lhe estes valores (Figura 16). De seguida, verifica se vale a pena analisar esse nó, recorrendo à condição de poda $\beta \leq \alpha$. Como esta condição não se verifica, procede à análise do nó D para determinar se existe algum valor inferior a 4 que justifique uma mudança de decisão.

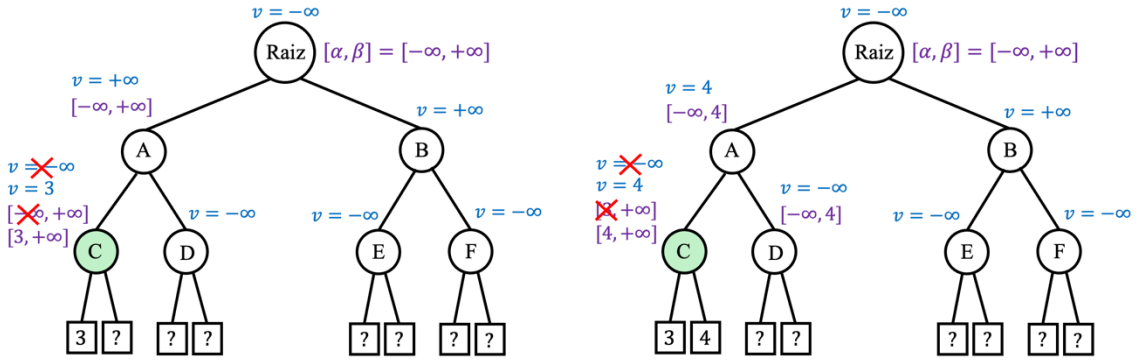

Figura 16: Análise do nó C

O nó D é controlado pelo jogador MAX, e inicia a sua exploração pelo primeiro filho (à esquerda), que tem valor de avaliação 8, e que por isso vai atualizar o valor de α e o valor atual do nó D (Figura 17). Posteriormente, o nó D verifica se vale a pena explorar o seu outro filho (à direita). Neste caso, como $\alpha = 8$ e $\beta = 4$, a condição $\beta \leq \alpha$ verifica-se, logo o nó D não avalia o seu segundo filho, e ocorre poda Alfa. Esta situação está representada na Figura 17, com os traços castanhos sobre o ramo podado.

O valor final do nó D permanece 8 e é devolvido ao nó A. No entanto, como o nó A é um nó MIN e já tinha anteriormente recebido o valor 4 do nó C, mantém o valor 4, tal como é visível na árvore da direita, na Figura 17. Este valor é então devolvido à raiz da árvore.

Sendo a raiz um nó MAX, o valor de α e o valor de avaliação do nó são atualizados para o máximo entre $-\infty$ e 4, resultando em 4. Com esta atualização, o jogador MAX assegura que não obterá um valor inferior a 4. O nó B, sendo filho da raiz, herda estes valores, como se observa também na Figura 17.

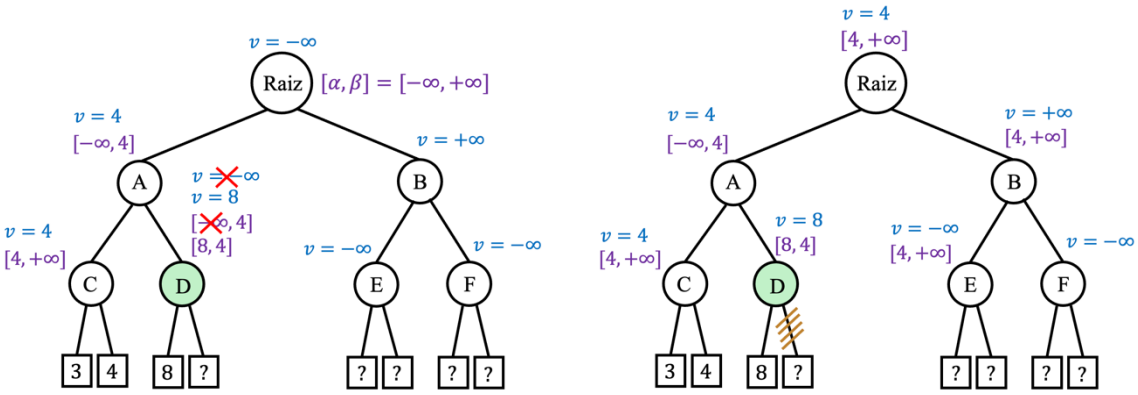

Figura 17: Análise do nó D

Como o nó B é controlado pelo jogador MIN, verifica-se novamente se vale a pena analisar o seu primeiro filho, o nó E (à esquerda), recorrendo, mais uma vez, à condição de poda. Como esta condição não se verifica, prossegue-se para a avaliação desse nó e o nó B passa-lhe os valores herdados da raiz, tal como ilustrado na Figura 17.

A análise no nó E começa com o nó terminal, que possui valor 2. Como se trata de um nó MAX, o valor atual da sua avaliação é atualizado para 2, mas como 2 é inferior ao valor herdado $\alpha = 4$, o valor de alfa não é alterado. Como a condição de poda não se verifica após a atualização de valores, o nó E avalia o seu segundo filho, cujo valor é 1. Porém o valor de α e do nó E não são atualizados, pois 1 é inferior a 4 e a 2, respetivamente.

Após concluir a análise do nó E, o valor 2 é devolvido ao nó B. Como B quer minimizar, o β e o seu valor do nó são atualizados para 2, pois 2 é inferior ao valor $+\infty$ que tinha inicialmente (Figura 18).

Neste momento, no nó B, $\beta = 2$ e $\alpha = 4$, pelo que a condição $\beta \leq \alpha$ se verifica. Isto indica que o jogador MAX (na raiz) já dispõe de uma opção melhor (com valor 4, proveniente do nó A), e, por conseguinte, não terá vantagem em escolher qualquer

caminho associado ao nó B, independentemente do valor que o nó F possa ainda devolver. Note-se que qualquer valor devolvido por F será, igual ou superior ao valor atual de β, mas ainda assim inferior ao valor já garantido pelo jogador MAX. Assim sendo, a avaliação do nó B é interrompida, e ocorre poda Beta sobre esse ramo, Figura 18. O valor final atribuído ao nó B permanece 2, sendo ele devolvido à raiz. Dado que, os valores de B são piores que os que a raiz já tem, nenhum dos valores é atualizado.

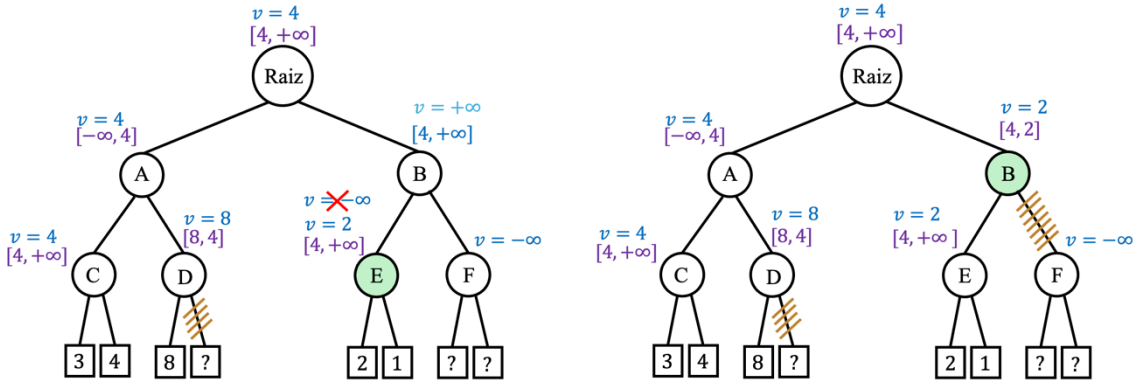

Figura 18: Análise dos nós E e B

O valor final atribuído à raiz após a execução do algoritmo Mini-Max com Poda Alfa-Beta será então 4, como ilustrado na Figura 18. Este será o valor correspondente à jogada do computador que origina o estado de jogo representado pelo nó A. Recorde-se que, cada vez que o computador é chamado a jogar, é criada uma árvore de decisão cujo valor da melhor jogada é indicado pelo valor da sua raiz.

Importa destacar que, como esperado, o valor obtido é exatamente o mesmo que foi obtido com o algoritmo Mini-Max, uma vez que a poda não afeta a qualidade de decisão, apenas elimina ramos desnecessários, otimizando o processo de avaliação.

4.3. Função de Avaliação de Estados de Jogo

Quer o algoritmo Mini-Max quer o algoritmo Mini-Max com Poda Alfa-Beta, implementado neste projeto, utilizam uma função de avaliação para atribuir um valor numérico a cada estado de jogo analisado, orientando o algoritmo para a seleção da jogada mais vantajosa para o computador. Esta função é chamada sempre que o algoritmo atinge a profundidade máxima da árvore. Nessa altura, a função calcula uma pontuação ponderada para cada jogador com base em fatores estratégicos, e, em seguida, determina a diferença entre a pontuação do *Computador* e a do *Humano*. Esta abordagem permite ao algoritmo priorizar decisões que conduzam à vitória e evitar desfechos desfavoráveis (Nasa et al., 2018).

Dado um estado de jogo a pontuação de um jogador, seja ele o *Computador* ou o *Humano*, baseia-se na função de avaliação proposta por Gregor & Boeck-Chenevier (2017) a qual destaca a importância de elementos estratégicos e atribui pesos específicos a cada fator, garantindo decisões otimizadas e consistentes para o jogo de damas. Gregor & Boeck-Chenevier (2017), definiram fatores estratégicos e respetivos pesos unitários, os quais foram testados com diferentes configurações. Os valores apresentados a seguir correspondem aos que proporcionaram o melhor desempenho (maior número de vitórias) para o jogador (*Computador)* nos testes efetuados. Assim, no cálculo da pontuação de um jogador (*Computador* ou *Humano*), os fatores estratégicos e os respetivos pesos considerados para um dado estado de jogo são os seguintes:

- Contagem de peças do jogador, com pesos de 5 para peões e 7,75 para damas;
- Presença de peões do jogador na sua linha inicial, para prevenir promoções adversárias, ponderada com peso 4;
- Centralização: número de peças do jogador na caixa do meio, isto é, nas casas 14, 15, 18 e 19, e controlo do centro, com as casas 13, 16, 17 e 20, com pesos de 2,5 e 0,5, respetivamente;
- Análise de vulnerabilidade imediata e proteção das peças, com pesos de -3 para peças vulneráveis e 3 para peças protegidas.

Consideram-se como vulneráveis todas as peças que podem ser capturadas de forma direta na jogada seguinte do adversário. Consideram-se como protegidas todas as peças

que estejam protegidas de capturas diretas quer por estarem nas bordas do tabuleiro quer por estarem apoiadas por peças aliadas (do próprio jogador) nas diagonais posteriores. Esta avaliação considera assim a estrutura defensiva do jogador, refletindo a robustez da posição e a dificuldade que o adversário terá em realizar capturas eficazes.

A combinação de todos os fatores anteriores permite gerar uma pontuação ponderada para cada um dos jogadores (*Computador* e *Humano*). A avaliação final do estado de jogo será então representada pela diferença entre a pontuação do computador e a pontuação do humano. Diferenças positivas indicam vantagem para o computador e diferenças negativas indicam vantagem para o humano. Para ilustrar o processo de cálculo da pontuação de um jogador, considere-se o tabuleiro de jogo representado na Figura 19, o qual será analisado do ponto de vista do jogador *Humano* (peças brancas).

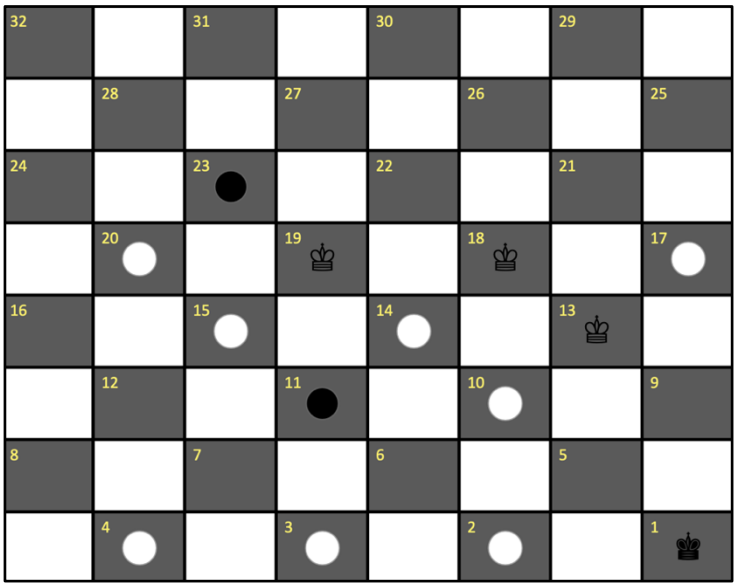

Figura 19: Possível estado do tabuleiro

Neste cenário, o jogador *Humano* possui oito peões e três damas, totalizando 11 peças. Três dos peões, encontram-se na sua linha de origem, o que, segundo a função de avaliação, contribui positivamente para impedir promoções adversárias.

Relativamente ao posicionamento estratégico, quatro peças ocupam casas da caixa do meio (casas 14, 15, 18 e 19), zonas centrais taticamente vantajosas, e outras três situam-se nas linhas centrais (casas 13, 17 e 20), contribuindo para o controlo do espaço.

No que respeita à segurança das peças, a função de avaliação identificou uma peça vulnerável (na casa 20), ou seja, passível de ser capturada diretamente na jogada seguinte do adversário, e seis peças protegidas (nas casas 2, 3, 4, 17, 18 e 19), posicionadas de forma a impedir capturas imediatas. As peças 2, 3, 4 e 17 estão protegidas por se encontrarem no limite do tabuleiro, enquanto, a peça na casa 19 está defendida pelas peças localizadas nas casas 14 e 15, que ocupam as diagonais atrás. De forma semelhante, a peça na casa 18 conta com a proteção das peças nas casas 13 e 14. Por outro lado, as peças nas casas 14 e 20 não se encontram protegidas, pois apenas uma das suas diagonais posteriores contém uma peça aliada. Note-se que a peça na casa 20 tem uma diagonal desocupada, enquanto a peça na casa 14 tem as duas diagonais ocupadas, mas uma delas contém uma peça do adversário.

Com base nos fatores identificados anteriormente e nos seus respetivos pesos, é possível calcular a pontuação de avaliação para o jogador *Humano* para o estado de jogo representado na Figura 19. Aplicando os pesos a cada critério estratégico, isto é, multiplicando o número de peças em cada fator pelo peso respetivo, obtém-se:

- 40 pontos pelos 8 peões (8×5);
- $23{,}25$ pontos pelas 3 damas $(3 \times 7{,}75)$;
- 12 pontos pelos peões na linha inicial (3×4);
- 10 pontos pelas peças na caixa do meio $(4 \times 2{,}5)$;
- $1{,}5$ pontos pelas peças nas linhas centrais $(3 \times 0{,}5)$;
- -3 pontos por 1 peça vulnerável $(1 \times (-3))$;
- 18 pontos pelas 6 peças protegidas (6×3).

Somando todos os contributos, obtém-se uma pontuação total de $101{,}75$, que reflete o valor estratégico desse estado de jogo do ponto de vista do jogador *Humano*.

Repetindo a análise para as peças pretas obtém-se o valor estratégico do estado de jogo do ponto de vista do jogador *Computador*, o que resultaria no valor $17{,}75$. Assim, a avaliação final do estado de jogo representado na Figura 19 seria $17{,}75 - 101{,}75 = -84$, o que indica vantagem para o jogador *Humano*.

5. IMPLEMENTAÇÃO DA APLICAÇÃO

A aplicação desenvolvida neste TFM tem como finalidade simular o jogo de damas entre um jogador *Humano* (utilizador) e o *Computador*, e foi desenvolvida usando a linguagem VBA do Excel. A interface gráfica da aplicação é baseada na própria folha do Excel. As células da folha representam as casas do tabuleiro, sendo utilizadas apenas as escuras no decorrer do jogo, conforme estipulado pelas regras. Essas células são manipuladas visualmente e logicamente por macros, que permitem a interação do utilizador com o jogo, a movimentação das peças e o controlo das jogadas do computador.

Para garantir clareza e uma melhor organização do código, o projeto foi dividido em vários módulos, com diferentes funções e métodos, consoante o seu propósito, tal como apresentado na Figura 20. A seguir, são descritos os principais componentes da aplicação.

Classe Peça

Atributos:
- Tipo – "Peão" ou "Dama"
- Jogador – "Humano" ou "Computador"
- Cor – "Branco" ou "Preto"
- Linha – linha atual na matriz (1 a 8)
- Coluna – coluna atual na matriz (1 a 8)
- Valor – 1 para Peão; 2 para Dama
- Símbolo – círculo para Peão; coroa para Dama (caracteres Unicode)

Métodos:
- Inicializar – inicializa a peça
- GetTipo; ...; GetSimbolo – acede aos atributos
- SetTipo; SetLinha; SetColuna – altera os atributos
- Mover – move para nova linha e nova coluna
- Promover – altera o tipo, valor e símbolo
- Clonar – copia a peça com todos os atributos

Classe Tabuleiro

Atributos:
- Tabuleiro (8 × 8) – armazena objetos da classe Peça

Métodos:
- Inicializar – insere as peças no tabuleiro inicial
- GetPeca – acede à peça, numa dada posição
- SetPeca – altera ou remove a peça da posição

Módulo Principal

Variáveis Globais:
- jogadorHumano; jogadorComputador – cor das peças atribuídas a cada jogador
- game – objeto do tipo tabuleiro, armazena o estado de jogo
- turnoAtual – indica o jogador atual ("Humano" ou "Computador")
- profundidadeUsada – define a profundidade do algoritmo
- contadorLancesSemEvento – conta jogadas sem captura ou promoção (usado para empates)
- contadorEmpateForcado – contador auxiliar para verificar empates forçados
- modoEmpateForcadoAtivo – ativa/desativa a verificação de empate forçado

Métodos:
- IniciarTabuleiro
- InserirPecas
- AlternarTurno

Módulo Jogada do Computador

Métodos:
- JogadaComputador
- GerarMovimentos
- SimularMovimento
- MinimaxAlfaBeta
- AvaliarTabuleiro
- ExecutarMovimento

Módulo Auxiliar

Métodos:
- CopiarTabuleiro

Módulo Aspeto Visual do Tabuleiro

Métodos:
- ExibirTabuleiro
- LimparDestaques

Módulo Movimentos

Métodos:
- TodosMovimentos
- GetDirecoesPeao
- MovimentosCapturaRecursiva
- MovimentosCapturaDamaRecursiva
- CopiarArray
- CollectionParaArray

Módulo Regras e Verificações de Término

Métodos:
- TemPecas
- TemJogadasPossiveis
- AtualizarContadores
- AtualizarContadoresSimulados
- HouveEventoResetaLances
- VerificarCondicaoEmpateForcado
- JogoTerminado
- ResultadoFinal

Figura 20: Estrutura modular do projeto

5.1. Classe Peça

Para representar cada peça do jogo de damas de forma estruturada, foi criada uma classe denominada "Peça". Esta classe contém todos os atributos e métodos necessários para gerir uma peça durante o jogo, como o tipo (Peão ou Dama), o jogador a que pertence (*Humano* ou *Computador*), a cor (Branco ou Preto), a posição no tabuleiro (linha e coluna), o valor associado (usado internamente para identificar o tipo de peça: 1 para peão e 2 para dama) e o símbolo que representa visualmente a peça na folha de Excel (círculo para peão, coroa para dama). Estes atributos são definidos no momento da inicialização da peça, pelo método "Inicializar*"*, que configura a peça segundo as entradas fornecidas e inicializa os atributos corretamente.

A utilização de uma classe específica para representar cada peça permite organizar toda a informação e lógica a ela associada, facilitando a sua manipulação durante o jogo. Com esta estrutura, cada peça é tratada como um objeto independente, o que torna mais simples a gestão das posições no tabuleiro, a verificação de movimentos válidos, a promoção de peões a damas e a comunicação com a interface.

Os atributos da classe foram definidos como privados para garantir que as propriedades das peças não sejam alteradas diretamente de forma indesejada. Por essa razão, a classe possui propriedades de acesso (*getters*), que permitem obter os valores dos atributos da peça de forma controlada, e de modificação (*setters*), que possibilitam alterar esses valores quando necessário, de forma segura. Além destas, existem outras funcionalidades importantes, como "Mover", que permite alterar a posição da peça no tabuleiro, "Promover", que converte um peão numa dama quando a peça atinge a primeira linha do adversário (modificando o tipo da peça para "Dama", o valor para 2 e o símbolo apresentado no tabuleiro do Excel) e, ainda, "Clonar" que cria uma nova instância da peça com os mesmos atributos (útil no algoritmo de decisão do computador).

5.2. Classe Tabuleiro

A classe "Tabuleiro" é responsável pela representação do estado do tabuleiro de jogo e pela organização das peças no mesmo. Para isso, foi definida uma matriz 8x8, correspondente às 64 casas do tabuleiro. Cada célula dessa matriz pode conter uma instância da classe peça ou estar vazia (definida como *Nothing*), caso não exista qualquer peça na posição correspondente. A matriz é declarada como privada, respeitando o princípio do encapsulamento da programação orientada a objetos, segundo o qual, apenas a própria classe pode aceder diretamente à estrutura interna.

Para permitir a interação com o conteúdo da matriz, são necessários dois métodos públicos: "GetPeca(linha, coluna)", que devolve a peça existente na posição indicada da matriz e "SetPeca(linha, coluna, peça)", que permite substituir ou remover uma peça numa dada posição do tabuleiro. Estes métodos garantem que a integridade da estrutura interna do tabuleiro seja preservada, permitindo ao mesmo tempo que outras partes do sistema acedam ou modifiquem o estado do jogo de forma segura.

O método "Inicializar (jHumano, jComputador)" posiciona as peças iniciais nas casas escuras do tabuleiro, conforme a cor atribuída a cada jogador (determinada aleatoriamente).

Neste projeto, optou-se por posicionar o jogador *Computador* sempre na parte superior do tabuleiro e o jogador *Humano* na parte inferior. Para cada peça posicionada no tabuleiro, é criada uma instância da classe peça, sendo esta devidamente inicializada com: o tipo ("Peão"), o jogador a que pertence, a cor atribuída e a respetiva posição no tabuleiro (linha e coluna).

5.3. Módulo Principal

A lógica principal do jogo é gerida através de procedimentos centrais definidos num módulo específico e é suportada por variáveis globais. A utilização destas variáveis, permite partilhar informação entre os diferentes módulos, facilitando a coordenação entre componentes, como o tabuleiro, a jogada do computador e a interface do utilizador.

As variáveis utilizadas são: “jogadorHumano” e “jogadorComputador” – guardam as cores das peças dos jogadores; “game” – representa o estado atual do tabuleiro; “turnoAtual” – indica o jogador a jogar; “profundidadeUsada” – define a profundidade do algoritmo utilizado nas jogadas do computador e ainda três variáveis que são usadas para controlar os diferentes casos de empate, de acordo com as regras do jogo.

Os procedimentos criados são: “IniciarTabuleiro” – configura o ambiente inicial do jogo na folha Excel; “InserirPecas” – define as peças de cada jogador; e “AlterarTurno” – garante a alternância de jogadas entre o jogador *Humano* e o *Computador*, e é chamado após cada jogada válida realizada por qualquer um dos jogadores.

5.4. Módulo Auxiliar

Para garantir a modularidade e facilitar a reutilização de código ao longo do desenvolvimento do jogo, foi implementada uma função auxiliar essencial para a manipulação do estado do tabuleiro.

A função “CopiarTabuleiro” tem como objetivo criar uma réplica independente de um objeto do tipo “Tabuleiro”, garantindo a integridade dos dados originais e possibilitando simulações e avaliações sem alterar o estado atual do jogo. Esta função percorre todas as posições da matriz 8x8 do tabuleiro original e, para cada célula, verifica a presença de uma peça. Se existir, invoca o método “Clonar” da classe Peça, criando uma instância com as mesmas propriedades da peça original. Essa peça clonada é então posicionada no novo tabuleiro através do método “SetPeca”. Caso a célula esteja vazia, é atribuído o valor *Nothing*.

Esta abordagem garante que alterações posteriores ao novo tabuleiro copiado não afetam o tabuleiro original, assegurando o isolamento entre instâncias. Essa característica é especialmente útil em cenários onde o computador executa jogadas de teste, evitando interferências nos dados reais do jogo.

5.5. Módulo Aspeto Visual do Tabuleiro no Excel

Para proporcionar uma interface gráfica funcional e intuitiva para o utilizador, foi desenvolvido um módulo que gere visualmente o tabuleiro de jogo na folha de cálculo do Excel. Esta funcionalidade é essencial para representar o estado do jogo de damas em tempo real, tanto ao nível da disposição das peças como da interação com o utilizador.

Neste módulo, foi criado o método "ExibirTabuleiro" para atualizar graficamente o tabuleiro após cada jogada. Ele percorre a matriz interna do jogo (gerida pelo objeto "game") e insere nas células as representações simbólicas das peças, utilizando o método "GetSimbolo()" de cada peça. Antes de cada atualização, o conteúdo da área do tabuleiro é limpo, garantindo que não persistem elementos gráficos obsoletos.

O módulo inclui também o método "LimparDestaques" que visa eliminar qualquer destaque gráfico aplicado em jogadas anteriores (por exemplo, sugestões de jogada). Este método percorre todas as casas pretas do tabuleiro e restabelece a sua cor original (cinzento escuro). Esta funcionalidade é fundamental para garantir que o utilizador tem sempre uma visão clara e atualizada do estado do jogo.

5.6. Módulo Movimentos

Este módulo reúne a lógica responsável pela determinação de todos os movimentos válidos no jogo, quer de peões quer de damas, respeitando as regras específicas de movimentação e captura de cada peça.

A função principal, "TodosMovimentos", percorre todo o tabuleiro e verifica quais as possibilidades de movimento, para cada peça do jogador atual. Usa a função auxiliar "GetDirecoesPeao", que devolve as direções de avanço dos peões. Estes podem realizar movimentos simples (diagonais para frente) ou capturas, sendo estas últimas verificadas também de forma recursiva pela função "MovimentosCapturaRecursiva", que permite identificar múltiplas capturas obrigatórias na mesma jogada.

Por outro lado, as damas podem realizar movimentos diagonais, tanto para a frente como para trás, e podem percorrer múltiplas casas, desde que o caminho esteja livre. A função "MovimentosCapturaDamaRecursiva" permite encontrar todas as possíveis

capturas múltiplas de uma dama, respeitando o requisito de não capturar a mesma peça mais do que uma vez na mesma jogada.

Cada movimento válido é armazenado num *array* com as seguintes informações:

- Posição inicial da peça (linha, coluna);
- Posição final após o movimento (linha, coluna);
- Valor total das peças capturadas (0 se movimento simples);
- Array com as posições das peças capturadas (linha, coluna);
- Caminho percorrido até ao destino final (*array* com todas as casas atravessadas).

No final do processo, a função filtra e retorna apenas os movimentos que resultam no maior número de capturas, garantindo que o jogo obedeça à obrigatoriedade de captura máxima. Para lidar com coleções e *arrays*, foram desenvolvidas funções auxiliares como "CopiarArray" e "CollectionParaArray", que ajudam a manipular listas de movimentos e capturas de forma eficiente.

Esta abordagem modular e recursiva assegura um cálculo preciso de todos os movimentos possíveis, respeitando as regras e permitindo ao motor de jogo tomar decisões estratégicas com base nas melhores jogadas disponíveis.

5.7. Módulo Jogada do Computador

Esta secção descreve a implementação da jogada do computador, que recorre ao algoritmo Mini-Max com Poda Alfa-Beta, para escolher o movimento mais vantajoso e cuja teoria foi detalhada na Secção 4.2.

O processo inicia-se com o método "JogadaComputador", que cria uma cópia do estado atual do tabuleiro usando o método "CopiarTabuleiro", garantindo assim que os movimentos testados não afetam o estado real da partida. Em seguida, a função "GerarMovimentos" gera todos os movimentos válidos disponíveis para o computador naquele turno, recorrendo à função "TodosMovimentos" do Módulo Movimentos.

Para cada movimento possível, a função "SimularMovimento" cria uma simulação do tabuleiro resultante, incluindo o eventual avanço para a promoção de peões e a

remoção das peças capturadas. Estes estados de tabuleiro simulados são então preparados para avaliação e comparação.

Cada tabuleiro resultante é avaliado com a função "MinimaxAlfaBeta", responsável pela execução recursiva do algoritmo Mini-Max com Poda Alfa-Beta até à profundidade previamente definida. A estrutura da função "MinimaxAlfaBeta" pode ser consultada no Apêndice, onde são apresentados os passos detalhados da sua execução.

Quando é atingido um estado terminal (vitória, derrota ou empate) ou a profundidade máxima, isto é, quando se atinge as folhas da árvore de decisão, o algoritmo invoca a função "AvaliarTabuleiro", que retorna o valor de avaliação do estado analisado. A função de avaliação aplica uma abordagem quantitativa com base em múltiplos critérios estratégicos, conforme descrito na Secção 4.2.1, calculando uma pontuação ponderada para ambos os jogadores. A diferença entre a pontuação do computador e a do humano constitui o valor atribuído ao estado.

Após simular e avaliar todos os movimentos possíveis, o computador seleciona o movimento com a pontuação mais elevada como a jogada ótima e executa-o utilizando o método "ExecutarMovimento". Este método atualiza o estado real do tabuleiro, o estado interno do jogo e a interface visual no Excel, promovendo peões e removendo as peças capturadas, quando aplicável.

Finalmente, a função "ResultadoFinal" é chamada para verificar se a jogada resultou no fim do jogo. Se o retorno da função for "Continuar", os contadores relacionados com o empate são atualizados com o método "AtualizarContadores" e o controlo do turno é transferido para o jogador *Humano* através do método "AlternarTurno".

Este processo assegura que o computador realiza jogadas estrategicamente fundamentadas, considerando não apenas o movimento imediato, mas também as suas consequências.

5.8. Módulo Regras e Verificações de Término

Este módulo é responsável por gerir toda a lógica de verificação do estado final do jogo, garantindo que as condições de vitória, derrota ou empate sejam corretamente detetadas e tratadas.

A função "TemPecas" determina se ainda existem peças em jogo de um jogador. Já a função "TemJogadasPossiveis" utiliza a função "TodosMovimentos" para averiguar se o jogador ainda tem movimentos válidos, sendo fundamental para evitar situações em que o jogador está bloqueado, mesmo tendo peças no tabuleiro.

Para garantir o correto funcionamento das regras de empate do jogo, especialmente em contextos distintos (execução real vs. simulações do algoritmo Mini-Max), foram desenvolvidos dois procedimentos específicos para a atualização de contadores. O método "AtualizarContadores", é utilizado durante o jogo real, a cada jogada realizada por um jogador (*Humano* ou *Computador*). Ele atua diretamente sobre variáveis globais e atualiza dois contadores fundamentais: o contador de lances sem evento (lances em que não há captura nem movimento de peões) usado na regra do empate por 20 lances, e o contador de lances de empate forçado (relevante para a configuração especial de "3 damas vs 1 dama") que ativa a regra de empate por 12 lances, se aplicável.

Em contrapartida, o método "AtualizarContadoresSimulados", é uma versão paralela para simulações, utilizada exclusivamente no interior do algoritmo Mini-Max com Poda Alfa-Beta. Diferente da versão real, este procedimento não altera variáveis globais, pois opera com parâmetros passados por valor. Isso assegura que os contadores sejam atualizados corretamente conforme as regras, durante cada nó da árvore de decisão e não afetem o estado real do jogo, mantendo a integridade da execução fora do algoritmo de decisão.

As funções "HouveEventoResetaLances" e "VerificarCondicaoEmpateForcado" analisam o tabuleiro para determinar se é necessário reiniciar os contadores (voltar a zero) ou se o modo de empate forçado está ativo, com base no posicionamento das damas e nos movimentos realizados.

A função principal “JogoTerminado” integra todas essas verificações e devolve uma mensagem interna ao programa (“Vitória”, “Derrota”, “Empate” ou “Continuar”), que indica o estado final:

- Vitória ou derrota (por falta de peças ou de movimentos);
- Empate (por 20 lances sem evento ou pela regra forçada de 12 lances);
- Continuação do jogo (caso nenhuma condição de fim seja satisfeita).

Finalmente, a função “ResultadoFinal” apresenta ao utilizador uma mensagem adequada ao desfecho do jogo, com base no resultado da função “JogoTerminado”. As mensagens geradas são apresentadas na secção seguinte, com o respetivo registo visual.

6. APLICAÇÃO DESENVOLVIDA

Nesta secção, são apresentadas as funcionalidades implementadas na aplicação desenvolvida, bem como a forma de interação com o utilizador. Serão descritos os principais elementos da interface, a configuração inicial do tabuleiro e os mecanismos que permitem a execução de uma partida de damas de forma interativa.

Sempre que o ficheiro Excel da aplicação é aberto, o ambiente inicial do jogo é configurado automaticamente. É criada uma página com um tabuleiro 8x8, atribuindo a cada célula as dimensões, cores e estilos apropriados. A disposição alternada das casas (escuras e claras) visa replicar o padrão característico do jogo de damas. Adicionalmente, são inseridas as identificações "Computador" e "Jogador" nas extremidades superior e inferior da área de jogo para identificar o lado de cada jogador. A zona de jogo é destacada com uma borda preta espessa, tal como se vê na Figura 21.

Além disso, é apresentado o botão "Ver Regras das Damas Portuguesas", que permanece visível durante toda a execução do jogo, permitindo ao utilizador consultar as regras a qualquer momento, de forma rápida. Ao premir este botão, o utilizador é redirecionado para um ficheiro PDF guardado numa *drive*, onde estão descritas detalhadamente as regras do jogo. Adicionalmente, é exibido ao utilizador o botão "Iniciar Jogo", para permitir que a partida comece de forma controlada.

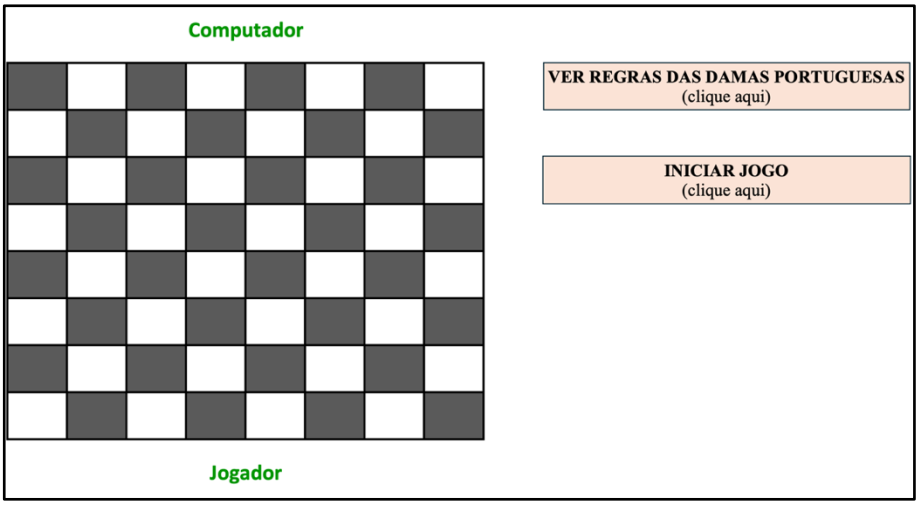

Figura 21: Abertura do jogo

O jogo apenas se inicia após o utilizador carregar no botão "Iniciar Jogo", e, nessa altura, é preparado visualmente o tabuleiro na folha Excel. Este botão é responsável por: sortear a cor atribuída a cada jogador; definir o jogador inicial (jogador que começa com as peças brancas), destacando a verde a célula correspondente na interface do Excel; informar o utilizador sobre a distribuição das cores através de uma caixa de diálogo (Figura 22); colocar as peças no tabuleiro e, caso o computador seja o primeiro a jogar, executar automaticamente a sua jogada. Para facilitar a interação do utilizador com o tabuleiro, o botão de início insere ainda caixas de texto sobre as casas válidas para movimentação (casas escuras), numerando-as de 1 a 32. As caixas são configuradas para garantir que o utilizador não interaja diretamente com as mesmas, prevenindo alterações acidentais ao clicar no rato durante o jogo. A ordem da numeração depende da perspetiva do jogador *Computador*, ou seja, se este jogar com as peças brancas, a numeração progride de cima para baixo; se jogar com as peças pretas, a ordem é invertida. No exemplo da Figura 23, o *Computador* inicia o jogo com as peças pretas, pelo que a numeração se inicia da parte inferior para a parte superior do tabuleiro.

Figura 22: Mensagem de início do jogo

Após ser clicado, o botão de início é ocultado, tal como se pode ver na Figura 23, impedindo reinicializações acidentais e garantindo que o jogo decorre de forma contínua a partir desse momento. Este procedimento garante que todas as condições estão corretamente definidas antes de começar a interação com o jogador *Humano*.

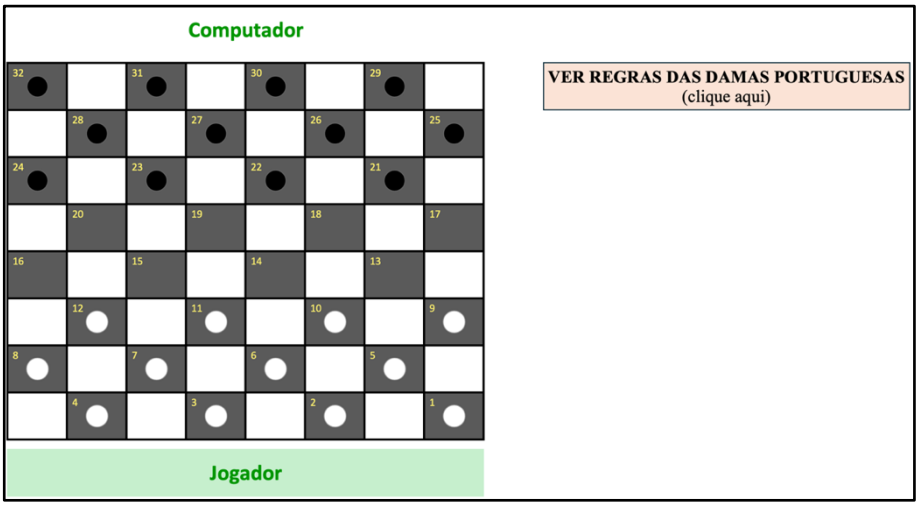

Figura 23: Jogo iniciado

Para garantir a alternância de jogadas entre o jogador *Humano* e o *Computador*, a interface do jogo é atualizada após cada jogada válida realizada por qualquer um dos jogadores, modificando visualmente a folha de cálculo do Excel. O jogador que tem a vez de jogar aparece destacado com um tom verde claro. Por exemplo, na Figura 23 aparece destacado a verde o "Jogador", ou seja, é a vez do jogador *Humano* jogar.

Na vez do *Computador* jogar, é chamado automaticamente o método que executa a jogada com base no algoritmo Mini-Max com Poda Alfa-Beta, detalhado anteriormente.

Quando o jogador *Humano* seleciona uma célula no tabuleiro, o sistema distingue automaticamente dois momentos principais: seleção da peça a mover e seleção do destino da mesma. Após o jogador selecionar uma célula no tabuleiro, o sistema verifica se a célula contém uma peça pertencente ao jogador *Humano* e se essa peça tem movimentos disponíveis (respeitando a obrigatoriedade de capturas, caso existam). Caso a seleção seja inválida (por exemplo, se se selecionar uma peça do adversário, uma célula vazia ou uma célula fora do tabuleiro), são apresentadas mensagens informativas (Figura 24).

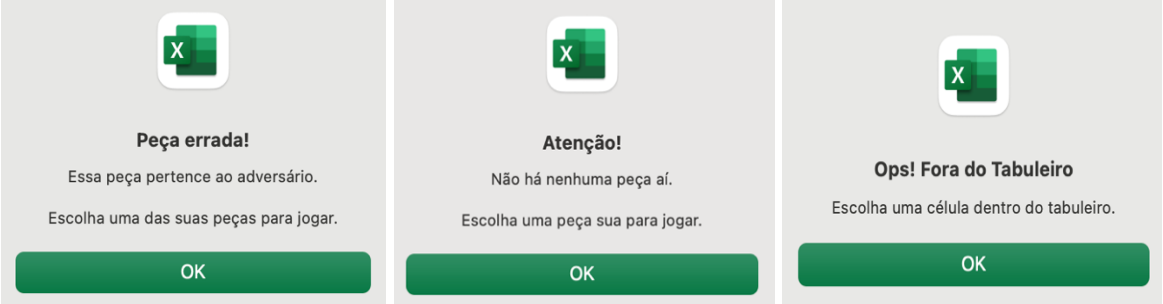

Figura 24: Três tipos de mensagem apresentadas quando não se seleciona uma das peças corretas

Por outro lado, se o jogador selecionar uma peça sua que não obedeça às regras de movimento ou captura aparece outra mensagem de erro (Figura 25).

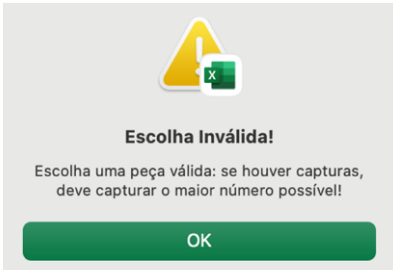

Figura 25: Mensagem de erro de seleção de peça inválida de acordo com as regras

Após duas tentativas incorretas de seleção de peças, o sistema oferece ao jogador a possibilidade de destacar automaticamente as peças que podem efetuar a jogada naquele momento (Figura 26).

Figura 26: Mensagem de ajuda – destacar peças com movimentos válidos

Se o jogador clicar em "sim", as peças com movimentos válidos são destacadas a azul na interface (Figura 27).

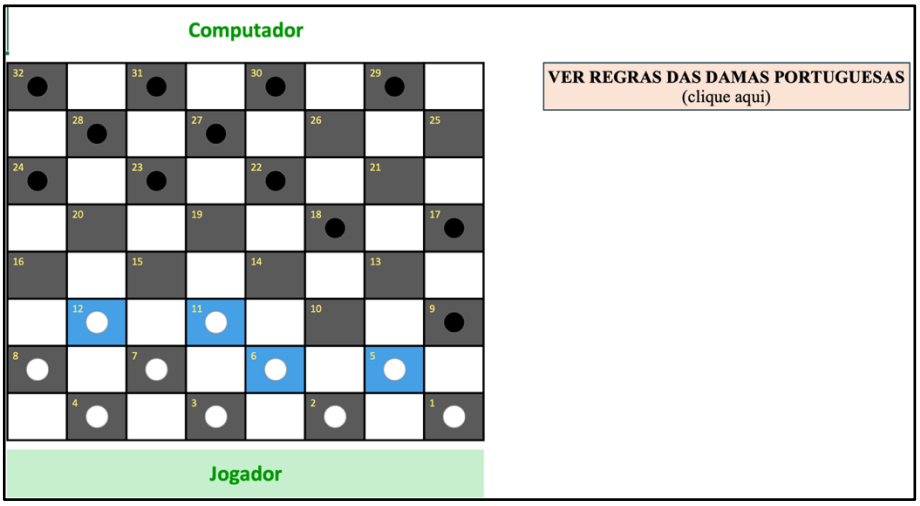

Figura 27: Destaque a azul das peças com movimentos válidos

Se após o destaque o jogador selecionar uma peça válida, o sistema identifica movimentos possíveis para essa peça, distinguindo entre casas de destino finais e casas intermediárias (onde ocorre a captura de peças adversárias). Os caminhos de captura são destacados a laranja, enquanto os destinos finais são destacados a verde (Figura 28). Este realce facilita a escolha de um destino válido por parte do jogador.

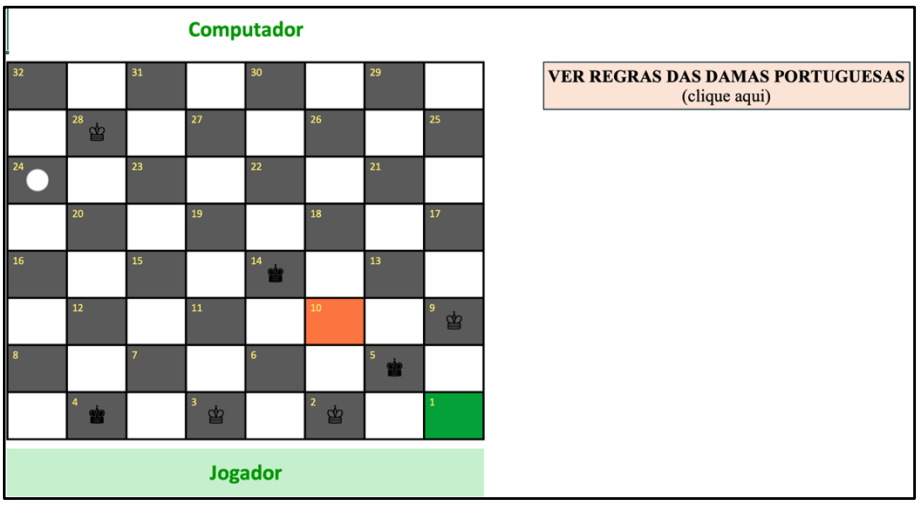

Figura 28: Destaque de caminhos de captura (a laranja) e de destinos finais (a verde) para a dama branca na casa 28

Após a seleção de uma peça, o jogador deve indicar o seu destino final, que é então validado. Se o destino não for permitido, aparece uma mensagem de erro a avisar que tem de clicar na casa destacada a verde (Figura 29).

Figura 29: Mensagem de erro de destino

Se o destino for permitido (destinos destacados a verde), o sistema executa o movimento no tabuleiro: a peça é movida para a nova posição, promovida a dama, se necessário, e, caso haja peças capturadas, elas são removidas do tabuleiro. A interface gráfica do tabuleiro é atualizada em tempo real para refletir o novo estado. Na Figura 30, à esquerda, a dama branca na casa 28 consegue capturar as damas pretas nas casas 14 e 5 e tem como destino final a casa 1 (a verde), logo no tabuleiro à direita as peças nas casas 14 e 5 vão desaparecer e a dama branca na casa 28 aparecerá na casa 1.

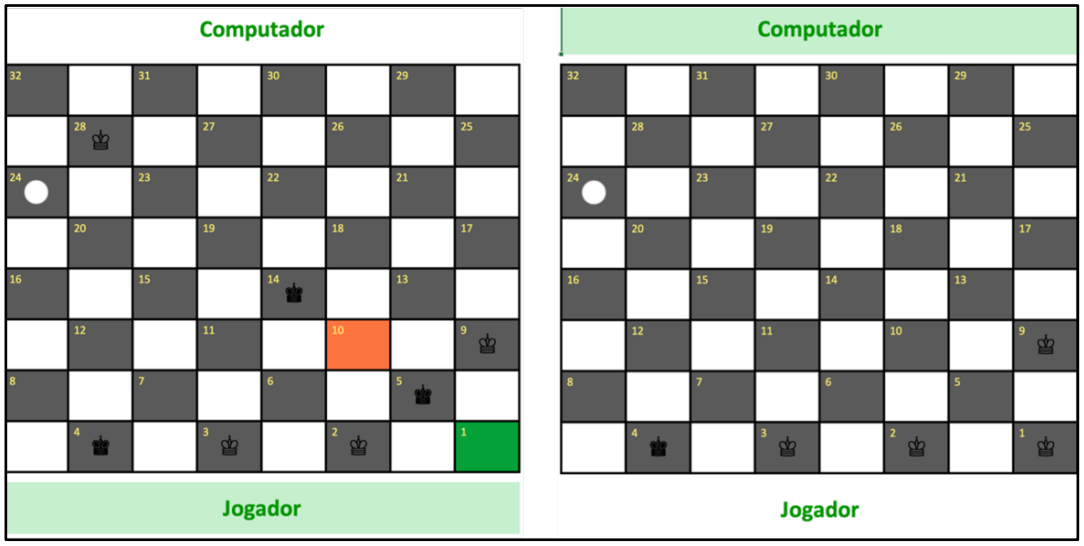

Figura 30: À esquerda – antes de selecionar o destino; à direita – depois de selecionar o destino

Por fim, o sistema determina se a partida acabou (vitória, derrota ou empate) ou se deverá prosseguir com o turno do adversário. Caso o jogo continue, o turno é alterado, garantindo a continuidade da partida. Na Figura 30, no tabuleiro da esquerda, é visível que é o jogador *Humano* a efetuar a jogada, pois este encontra-se destacado a verde. Depois de escolher o destino, passou a ser o *Computador* a jogar, dado que no tabuleiro da direita este é que aparece destacado, logo percebemos que internamente o código reconheceu que o jogo ainda não acabou.

Caso o jogo tenha terminado é ativada a lógica de verificação do estado final do jogo e é apresentada ao utilizador uma mensagem apropriada com base no resultado obtido (ver Figura 31).

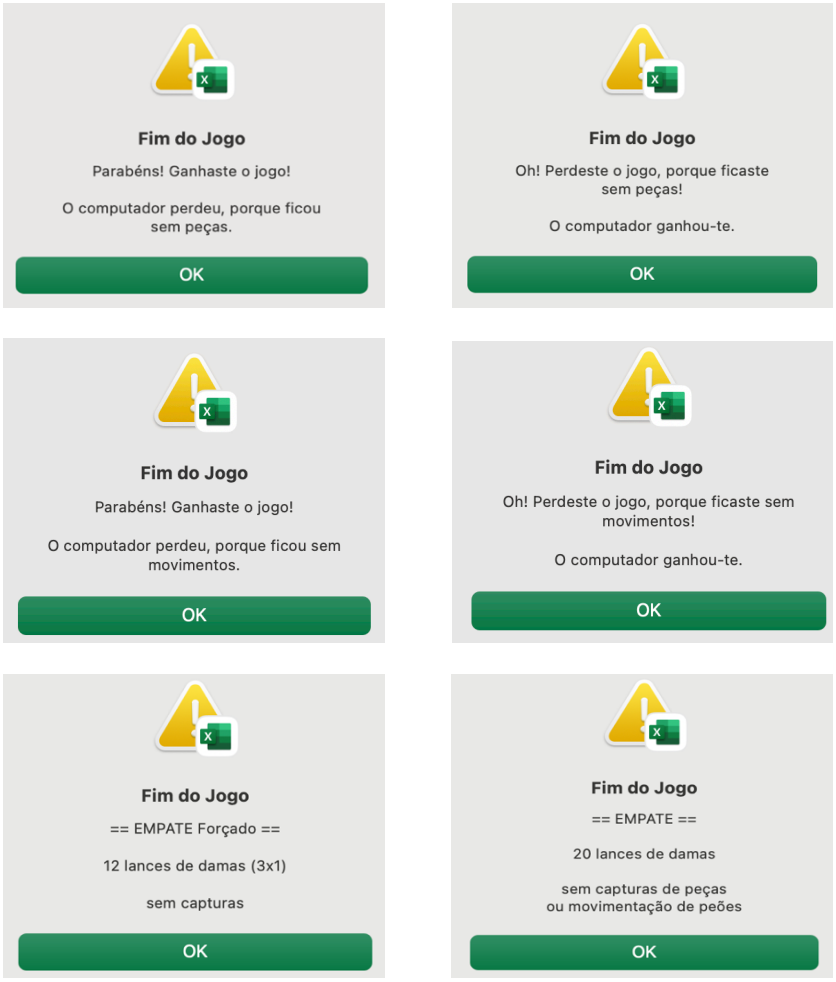

Figura 31: Seis tipos de mensagem de fim de jogo

7. RESULTADOS

Esta secção é dedicada à avaliação do desempenho da aplicação, em particular do algoritmo responsável pelas jogadas do *Computador*. Todos os testes foram executados num MacBook Air, com chip Apple M3, 8 GB de memória RAM e armazenamento Macintosh HD. Para tal, foram realizados confrontos entre a aplicação desenvolvida e outras aplicações de jogo de damas existentes para telemóvel, permitindo assim comparar a eficácia da aplicação proposta. As aplicações utilizadas foram:

- *Jogo de damas!* (Marcarelli, 2025) – possui três níveis de dificuldade: fácil, médio e difícil;
- *Damas* (Chatenet, 2025) – possui quatro níveis de dificuldade: 0, 1, 2 e 3;
- *Damas V+, fun board game* (Limited, 2025) – possui seis níveis de dificuldade: principiante, novato, amador, intermédio, difícil e perito;
- *Damas Clássicas (Portuguesas)* (Target Works, 2024) – possui sete níveis de dificuldade: 1, 2, 3, 4, 5, 6 e 7.

Os respetivos ícones de cada aplicação encontram-se apresentados na Figura 32.

Figura 32: Ícones das aplicações

Teoricamente, quanto maior for a profundidade definida no algoritmo Mini-Max com Poda Alfa-Beta, melhores tendem a ser os resultados obtidos, uma vez que o algoritmo consegue analisar um maior número de possibilidades de jogadas futuras. No entanto, esse aumento de profundidade também implica um maior tempo de execução, devido ao crescimento exponencial do número de estados a serem explorados. Assim, para determinar a profundidade a fixar no algoritmo, foi realizado um teste para medir o tempo que o *Computador* leva a efetuar a sua primeira jogada. Os resultados obtidos são apresentados na Tabela 1, onde a primeira coluna indica a profundidade usada pelo

algoritmo e a segunda coluna mostra o tempo de execução que o *Computador* demora a efetuar a primeira jogada.

Profundidade	Tempo da 1ª Jogada do Computador (min:seg)
1	00:00
2	00:00
3	00:01
4	00:03
5	00:14
6	00:30
7	01:35

Tabela 1: Tempo da 1ª jogada do *Computador* por profundidade

Nos confrontos com as aplicações, a profundidade do algoritmo Mini-Max com Poda Alfa-Beta, foi fixado com a profundidade 6, exceto na aplicação *Damas V+, fun board game*, conforme será explicado mais adiante. A escolha da profundidade 6 foi baseada no equilíbrio entre o tempo de execução e a teoria que quanto maior a profundidade melhor são os resultados. Conforme apresentado na Tabela 1, o tempo da primeira jogada do *Computador* aumenta conforme a profundidade, e a profundidade 6 oferece um tempo de resposta que se considerou adequado (30 segundos).

Durante os jogos, atuei como intermediária entre o *Computador* e as aplicações, uma vez que não seria possível que a aplicação proposta e as já existentes jogassem entre si diretamente. Assim, em cada jogo, o jogador *Computador* iniciou sempre a partida jogando com as peças brancas na aplicação proposta. A jogada do *Computador* foi depois replicada na aplicação do telemóvel. Após a aplicação produzir uma jogada de resposta, essa jogada for replicada novamente na aplicação do Excel. Este ciclo foi repetido para todas as jogadas até o jogo terminar. Significa isto que era a aplicação do telemóvel que gerava as jogadas do jogador *Humano* na aplicação proposta.

Nas tabelas abaixo encontram-se os resultados obtidos ao jogar contra cada uma das aplicações. Em cada tabela, a primeira coluna indica o nível de dificuldade utilizado na aplicação, a segunda coluna mostra o número total de jogadas efetuadas pelo jogador

Computador na aplicação proposta, a terceira coluna apresenta o tempo total, em minutos, que o jogador *Computador* demorou a fazer todas as suas jogadas, a quarta coluna representa o tempo médio, em segundos, das jogadas do *Computador* e a última coluna indica o resultado obtido no jogo.

Jogo de damas!				
Nível	Jogadas	Tempo Total (min)	Tempo Médio (seg)	Resultado
Fácil	23	9	23	Vitória
	25	10	24	Vitória
	21	6	16	Vitória
Médio	27	10	23	Vitória
	28	10	22	Vitória
	24	7	18	Vitória
Difícil	25	5	11	Vitória
	26	3	7	Derrota
	32	14	25	Vitória

Tabela 2: Resultados do confronto com a aplicação *Jogo de damas!* (usando profundidade 6)

Damas				
Nível	Jogadas	Tempo Total (min)	Tempo Médio (seg)	Resultado
0	19	4	13	Vitória
	23	4	10	Vitória
	21	5	13	Vitória
1	22	7	20	Vitória
	29	8	16	Vitória
	25	3	8	Vitória
2	25	6	15	Vitória
	41	69	100	Vitória
	27	10	23	Vitória
3	42	18	25	Empate
	28	23	49	Vitória
	30	15	31	Vitória

Tabela 3: Resultados do confronto com a aplicação *Damas* (usando profundidade 6)

Damas Clássicas (Portuguesas)				
Nível	Jogadas	Tempo Total (min)	Tempo Médio (seg)	Resultado
1	22	5	14	Vitória
	25	3	7	Vitória
	19	6	18	Vitória
3	29	12	24	Vitória
	35	17	30	Vitória
	23	8	21	Vitória
5	42	54	77	Vitória
	42	55	78	Empate
	29	5	10	Vitória
7	31	27	51	Vitória
	27	4	9	Derrota
	29	4	8	Derrota

Tabela 4: Resultados do confronto com a aplicação *Damas Clássicas (Portuguesas)* (usando profundidade 6)

A Tabela 5 sumariza as taxas de vitória, derrota e empate por aplicação e no total de jogos realizados.

Aplicação	Nº de Jogos	Nº de Vitórias	Nº de Derrotas	Nº de Empates	Taxa de Vitória	Taxa de Derrota	Taxa de Empate
Jogo de damas!	9	8	1	0	89%	11%	0%
Damas	12	11	0	1	92%	0%	8%
Damas Clássicas	12	9	2	1	75%	17%	8%
Total	33	28	3	2	85%	9%	6%

Tabela 5: Análise dos resultados (usando profundidade 6)

Através dos resultados obtidos na Tabela 5, é possível verificar que a taxa de vitória em cada aplicação, bem como no total de jogos realizados, é sempre muito superior às taxas de derrota e de empate. Além disso, com o auxílio das Tabelas 2, 3 e 4, conclui-se que as derrotas e empates ocorrem apenas em níveis mais elevados das aplicações, indicando que o algoritmo desenvolvido na aplicação proposta no TFM é eficaz.

Por outro lado, é evidente que o tempo médio por jogada do *Computador* na aplicação *Jogo de damas!* é sempre inferior a meio minuto, e nas duas restantes aplicações, *Damas* e *Damas Clássicas (Portuguesas)*, pode ultrapassar 1 minuto. É importante realçar que o tempo médio de resposta do *Computador* pode variar a cada jogada, uma vez que existem jogadas obrigatórias que limitam as opções disponíveis, fazendo com que o *Computador* só possa escolher entre essas alternativas para realizar a jogada mais vantajosa. Nestes casos, o *Computador* executará a sua jogada mais rapidamente.

O jogo realizado contra a aplicação *Damas V+, fun board game*, foi efetuado com uma profundidade fixa de 4. Esta abordagem foi adotada para observar o comportamento do algoritmo com uma profundidade mais baixa, no nível mais elevado da aplicação (perito). Como é visível na Figura 33, o *Computador*, representado na Figura por "Ilda Barbara", fez as suas jogadas num total de aproximadamente 14 minutos, enquanto a aplicação, representada por "Computador", demorou cerca de 2 horas e meia. Dessa forma, conclui-se que o algoritmo implementado para as jogadas do *Computador* na aplicação proposta é significativamente mais eficiente do que a da aplicação *Damas V+, fun board game*.

Figura 33: Resultados da aplicação *Damas V+, fun board game*

Tendo em conta que, mesmo com uma profundidade média, o tempo de resposta desta aplicação foi significativamente superior ao da aplicação proposta, considerou-se desnecessária a realização de mais testes. O tempo de execução excessivo observado nesta

aplicação, permite concluir com clareza que o algoritmo implementado na aplicação proposta é mais eficiente.

Posteriormente, o algoritmo Mini-Max com Poda Alfa-Beta foi executado com a profundidade 4, para testar o funcionamento da aplicação proposta com diferentes profundidades. Desta forma, foram efetuados confrontos entre a aplicação proposta e cada uma das aplicações existentes, utilizando os níveis de dificuldade mais elevados de cada uma delas.

Nas tabelas abaixo encontram-se os resultados obtidos ao jogar contra cada uma das aplicações. Em cada tabela, a primeira coluna indica o número total de jogadas efetuadas pelo jogador *Computador* na aplicação proposta, a segunda coluna apresenta o tempo total, em minutos, que o jogador *Computador* demorou a fazer todas as suas jogadas, a terceira coluna representa o tempo médio, em segundos, das jogadas do *Computador* e a última coluna indica o resultado obtido no jogo.

Jogo de damas!			
Jogadas	Tempo Total (min)	Tempo Médio (seg)	Resultado
44	2	2	Empate
24	1	2	Derrota
31	1	2	Derrota

Tabela 6: Resultados do confronto com a aplicação *Jogo de damas!* (usando profundidade 4)

Damas			
Jogadas	Tempo Total (min)	Tempo Médio (seg)	Resultado
39	2	3	Empate
29	1	2	Derrota
40	2	3	Empate

Tabela 7: Resultados do confronto com a aplicação *Damas* (usando profundidade 4)

Damas Clássicas (Portuguesas)			
Jogadas	Tempo Total (min)	Tempo Médio (seg)	Resultado
40	2	3	Derrota
27	1	2	Derrota
39	2	3	Derrota

Tabela 8: Resultados do confronto com a aplicação *Damas Clássicas (Portuguesas)* (usando profundidade 4)

A Tabela 9 sintetiza as taxas de vitória, derrota e empate por aplicação e no total de jogos realizados.

Aplicação	Nº de Jogos	Nº de Vitórias	Nº de Derrotas	Nº de Empates	Taxa de Vitória	Taxa de Derrota	Taxa de Empate
Jogo de damas!	3	0	2	1	0%	67%	33%
Damas	3	0	1	2	0%	33%	67%
Damas Clássicas	3	0	3	0	0%	100%	0%
Total	9	0	6	3	0%	67%	33%

Tabela 9: Análise dos resultados (usando profundidade 4)

Recorrendo à Tabela 9 verifica-se que não houve qualquer vitória, no total dos 9 jogos realizados. Além disso, verifica-se que a taxa de derrota é bastante elevada, 67%, e a taxa de empate é de 33%. Em contrapartida, os tempos totais de jogo para o jogador *Computador* diminuíram consideravelmente ao se reduzir a profundidade de 6 para 4, nunca ultrapassaram os 2 minutos.

Através destes resultados, podemos concluir que, embora profundidades mais baixas reduzam o tempo total de jogo do *Computador*, elas podem comprometer seriamente a eficácia das jogadas, uma vez que não houve qualquer vitória com esta profundidade.

8. CONCLUSÃO

No presente TFM foi desenvolvida uma aplicação funcional para a simulação do jogo de Damas Portuguesas entre um jogador humano e o computador, respeitando as regras oficiais da modalidade e proporcionando uma experiência de jogo interativa e desafiante ao utilizador. Para tal, foi implementado o algoritmo Mini-Max com Poda Alfa-Beta, cuja eficácia na análise de jogos determinísticos entre dois jogadores com informação perfeita ficou devidamente demonstrada neste trabalho. A combinação deste algoritmo com uma função de avaliação cuidadosamente definida permitiu ao computador tomar decisões estratégicas com base em critérios objetivos, equilibrando eficiência computacional com qualidade de jogo.

A aplicação foi desenvolvida inteiramente no ambiente do Excel, com recurso à linguagem VBA, o que evidencia que é possível criar aplicações interativas em plataformas que não são especificamente criadas para desenvolvimento de jogos. A estrutura modular da aplicação, os algoritmos implementados e a atenção aos detalhes gráficos e funcionais da interface contribuíram para um produto final robusto e intuitivo.

Os testes realizados, através da comparação com outras aplicações de damas disponíveis, demostraram que quando se fixa a profundidade 6 no algoritmo da aplicação proposta, esta apresenta um desempenho mais competitivo, alcançando uma taxa de vitória de 85%, de empate de 6% e de derrota de apenas 9%, mesmo contra níveis de dificuldade elevados. Por outro lado, quando a profundidade é fixada em 4, a aplicação não tem um desempenho eficaz. Isto sugere que, quanto mais elevada for a profundidade fixada, melhores serão os resultados, apesar do tempo total de jogo do *Computador* aumentar. Estes resultados empíricos reforçam a validade da abordagem adotada e a eficácia dos algoritmos e estratégias implementadas.

Contudo, é importante reconhecer algumas limitações do trabalho desenvolvido. O tempo de resposta do computador pode aumentar significativamente com o uso de profundidades superiores, devido ao crescimento exponencial do número de estados a explorar. Além disso, o ambiente do Excel impõe restrições ao nível gráfico e de desenvolvimento, que poderiam ser colmatadas com o uso de plataformas mais adequadas ao desenvolvimento de jogos.

Como perspetivas de desenvolvimento futuro, destacam-se algumas possibilidades de melhoria, nomeadamente a otimização do tempo de resposta em profundidades superiores, a expansão para outros ambientes de desenvolvimento mais flexíveis, como por exemplo o Python, ou até a criação de uma versão online acessível a qualquer utilizador. Além disso, a integração de funcionalidades pedagógicas para ensino das regras poderia alargar o alcance e a utilidade da aplicação.

Em suma, o trabalho alcançou os objetivos inicialmente propostos e contribui com uma solução acessível e didática para a prática e estudo das Damas Portuguesas, valorizando esta modalidade tradicional num contexto digital.

REFERÊNCIAS BIBLIOGRÁFICAS

Caexeta, G. S. (2008). VisionDraughts – Um Sistema de Aprendizagem de Jogos de Damas Baseado em Redes Neurais, Diferenças Temporais, Algoritmos Eficientes de Busca em Árvores e Informações Perfeitas Contidas em Bases de Dados [Tese de Mestrado]. Universidade Federal de Uberlândia. https://repositorio.ufu.br/handle/123456789/12460 [Acedido em 08 02 2025]

Chatenet, H. (2025). Damas. https://apps.apple.com/pt/app/damas/id404242044 [Acedido em 01 07 2025]

Gabel, F. (2019). Some Studies in Machine Learning Using the Game of Checkers [Relatório Final de Curso]. https://hci.iwr.uni-heidelberg.de/system/files/private/downloads/636026949/report_frank_gabel.pdf [Acedido em 13 02 2025]

Gregor, K., & Boeck-Chenevier, M. (2017). Optimal Heuristic For Checkers [Relatório Final de Curso]. https://github.com/kevingregor/Checkers/blob/master/Final%20Project%20Report.pdf [Acedido em 14 05 2025]

Guerra, J. (2011). Classical Checkers [Resumo de Tese de Mestrado]. Instituto Superior Técnico. https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://fenix.tecnico.ulisboa.pt/downloadFile/395143159113/resumo.pdf&ved=2ahUKEwie48CggOKOAxVrNfsDHUI2GjsQFnoECBcQAQ&usg=AOvVaw2pUyJqBdOeTLN9Sq7v_jB6 Acedido em 29 04 2025]

Hoang, N. V. (2022). Board Game AI and Minimax Algorithm [Tese de Bacharelato]. University of Applied Sciences.

Jofanda, A. N. W., & Yasin, M. (2021). Design of Checkers Game Using Alpha-Beta Pruning Algorithm. *INTENSIF: Jurnal Ilmiah Penelitian Dan Penerapan Teknologi Sistem Informasi*, 5(2), 279–295. https://doi.org/10.29407/intensif.v5i2.15863

Knuth, D. E., & Moore, R. W. (1975). An Analysis of Alpha-Beta Priming. *Artificial Intelligence*, 6(4), 293–326. https://doi.org/10.1016/0004-3702(75)90019-3

Limited, Z. (2025). Damas V+, Fun Board Game. https://apps.apple.com/pt/app/damas-v-fun-board-game/id657066794 [Acedido em 01 07 2025]

Lynch, M. (1997). An Application of Temporal Difference Learning to Draughts [Relatório Final de Curso]. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7cb3c45694056ca0104f86011e6b23493febeaaa [Acedido em 19 02 2025]

Marcarelli, S. (2025). Jogo de Damas! https://apps.apple.com/pt/app/jogo-de-damas/id1076622926 [Acedido em 01 07 2025]

Millington, I., & Funge, J. D. (2009). Artificial Intelligence for Games (Second Edition). Morgan Kaufmann/Elsevier.

Mitchell, T. M. (1997). Machine Learning (Nachdr.). McGraw-Hill.

Nasa, R., Didwania, R., Maji, S., & Kumar, V. (2018). Alpha-Beta Pruning in Mini-Max Algorithm – An Optimized Approach for a Connect-4 Game. *International Research Journal of Engineering and Technology (IRJET)*, 05(04), 1637–1641.

Neto, H. de C. (2007). LS-DRAUGHTS – Um Sistema de Aprendizagem de Jogos de Damas baseado em Algoritmos Genéticos, Redes Neurais e Diferenças Temporais [Tese de Mestrado]. Universidade Federal de Uberlândia. https://repositorio.ufu.br/bitstream/123456789/12575/1/HCNetoDISSPRT.pdf [Acedido em 08 02 2025]

Popic, J., Boskovic, B., & Brest, J. (2021). Deep Learning and the Game of Checkers. *MENDEL Soft Computing Journal*, 27(2), 1–6. https://doi.org/10.13164/mendel.2021.2.001

Russell, S. J., & Norvig, P. (com Davis, E., & Edwards, D.). (2010). Artificial Intelligence: A Modern Approach (Third Edition). Prentice Hall.

Saffidine, A., Finnsson, H., & Buro, M. (2012). Alpha-Beta Pruning for Games with Simultaneous Moves. *Proceedings of the AAAI Conference on Artificial Intelligence*, 26(1), 556–562. https://doi.org/10.1609/aaai.v26i1.8148

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3), 210–229. https://doi.org/10.1147/rd.33.0210

Schaeffer, J., Culberson, J., Treloar, N., Knight, B., Lu, P., & Szafron, D. (1992). A World Championship Caliber Checkers Program. *Artificial Intelligence*, 53(2–3), 273–289. https://doi.org/10.1016/0004-3702(92)90074-8

Schaeffer, J., Lake, R., Lu, P., & Bryant, M. (1996). Chinook: The World Man-Machine Checkers Champion. *AI Magazine*, 17(1). https://doi.org/10.1609/aimag.v17i1.1208

Suancha, C. C., Suarez, M. J., & Besoain, F. A. (2024). Implementation of Alpha-Beta Pruning and Transposition Tables on Checkers Game. *IEEE Access*, 12, 46636–46645. https://doi.org/10.1109/ACCESS.2024.3381958

Sutton, R. S., & Barto, A. G. (2014). Reinforcement Learning: An Introduction (Second edition (in progress)). The MIT Press.

Target Works, U. (2024). Damas Clássicas (Portuguesas). https://apps.apple.com/pt/app/damas-cl%C3%A1ssicas-portuguesas/id1116717315 [Acedido em 01 07 2025]

Tomaz, L. (2013). D-MA-Draughts: Um sistema Multiagente Jogador de Damas Automático que Atua em um Ambiente de Alto Desempenho [Tese de Mestrado]. Universidade Federal de Uberlândia. https://repositorio.ufu.br/handle/123456789/12544 [Acedido em 08 02 2025]

Youru, L., Xueping, L., Yajie, W., & Yuting, X. (2015). A Systematic Research About the Computer Game of Checkers Based on the Decision-Making of AI. *The 27th Chinese Control and Decision Conference (2015 CCDC)*, 6437–6442. https://doi.org/10.1109/CCDC.2015.7161978

APÊNDICE

```
Função MinimaxAlfaBeta(tabuleiro, profundidade, alfa, beta, jogadorAtual):
    Se profundidade = 0  ou  Jogo terminado:
        Retornar avaliação heurística para o estado de jogo atual
    Se jogadorAtual "Computador":   'MAX – quer maximizar
        melhorValor = − ∞
        Para cada movimento do "Computador":
            tabuleiroNovo = Simular Movimento (tabuleiro, movimento)
            pontuacao = MinimaxAlfaBeta(tabuleiroNovo, profundidade − 1, alfa, beta, "Humano")
            melhorValor = max(melhorValor, pontuacao)   'atualiza o melhor valor (max possível)
            alfa = max (alfa,melhorValor)   'atualiza o valor de alfa
            Se beta ≤ alfa:   'verifica condição de poda
                Interromper ciclo   'poda verificada
    Caso contrário:   'jogadorAtual = "Humano"   MIN – quer minimizar
        melhorValor= + ∞
        Para cada movimento "Humano":
            tabuleiroNovo = Simular Movimento (tabuleiro, movimento)
            pontuacao = MinimaxAlfaBeta(tabuleiroNovo, profundidade − 1, alfa, beta,"Computador")
            melhorValor = min(melhorValor, pontuacao)   ' atualiza o melhor valor (min possível)
            beta = min (beta, melhorValor)   'atualiza o valor de beta
            Se beta ≤ alfa:   'verifica condição de poda
                Interromper ciclo   'poda verificada
    Retornar melhorValor

'Chamada inicial
MinimaxAlfaBeta(tabuleiro, profundidade_definida,  − ∞, +∞, "Humano")
```

Apêndice 1: Estrutura da função "MinimaxAlfaBeta"