Aplicação do Minimax no Jogo dos Dedos

Felipe Miranda de Oliveira
Instituto de Ciência e Tecnologia
Universidade Federal de São Paulo
São José dos Campos, Brasil
felipeoliveirac2@hotmail.com

***Resumo*—A utilização de jogos e brincadeiras na fase de aprendizagem da criança é uma importante ferramenta para aproxima-las dos estudos, e em especial no interesse pela matemática no caso exigir raciocínio lógico. Diante disso, o Jogo dos Dedos, o qual é baseado em operações matemáticas simples foi abstraído em linguagem de programação, sendo desenvolvidas diferentes estratégias baseadas em partidas reais, incluindo jogadas gulosas e defensivas, assim como a construção do algoritmo Minimax. Foram realizadas mil partidas entre cada tipo de estratégia, sendo o Minimax o que mais obteve número de vitórias, além disso, notou-se que a aplicação da poda alfa-beta reduziu em 70% o tempo das partidas.**

Palavras-chave—jogos, Minimax, poda alfa-beta

I. INTRODUÇÃO

É de conhecimento comum que o aprendizado da matemática é um tabu para crianças nos anos iniciais, ao encontro com isso, Fonseca (1995, p. 217) citado por [1] indica que existem vários motivos para se aprender essa matéria, desde falta de embasamento matemático, passando pelo ensino inadequado e chegando até em questões psiconeurológicas.

Pacheco [2] acrescenta que o ensino da matemática causa duas sensações contraditórias, envolvendo quem ensina e que está aprendendo: do ponto de vista de quem está ensinando, a matemática se trata de um ramo de conhecimento importante, já para os alunos é algo que remete a insatisfação pelo fato dos resultados negativos obtidos frequentemente durante sua aprendizagem.

Friedmann (1996) e Martins (2012) citados por Trettel [3] dizem que jogos e brincadeiras são meios pelo qual as crianças desenvolvem de forma positiva habilidades corporais, mentais, à criatividade, socialização e cooperação.

Maria Montessori utilizou materiais manipulativos para ensinar classes comuns, o qual num primeiro momento foi utilizado em classes com crianças excepcionais, pois acreditava que não existia aprendizado sem ação, para Azevedo (1979, p. 27) citado por Fiorentini [4]: "Nada deve ser dado a criança, no campo da matemática, sem primeiro apresentar-se a ela uma situação concreta que a leve a agir, a pensar, a experimentar, a descobrir, e daí, a mergulhar na abstração".

Em vista da importância dos jogos na fase de aprendizagem das crianças, foi realizado nesse trabalho a construção do Jogo dos Dedos, o qual é fundamentado em princípios matemáticos básicos, como adição e divisão. O jogo foi desenvolvido na linguagem Python, onde foram elaboradas diferentes estratégias para aplicar no jogo, sendo aos pares uma colocada em confronto as outras, verificando qual obteria o maior número de vitórias. Buscou-se também avaliar o tempo de processamento da estratégia Minimax com poda e sem poda durante as buscas na árvore do jogo. Para cada partida realizada é criado um arquivo com os resultados, assim como a sequência de jogadas.

II. TRABALHOS RELACIONADOS

Souza [5] aplicou o algoritmo minimax em um jogo digital de cartas chamado Triple Triad, com objetivo de criar uma inteligência artificial capaz de realizar boas jogadas para vencer o adversário, no entanto a poda alfa-beta não foi aplicada, ficando como sugestão de trabalhos futuros, como forma de reduzir o tempo de processamento.

Guedes e Nascimento [6] realizaram uma análise comparativa entre diferentes funções de utilidade aplicadas em um jogo de indígena, conhecido como Jogo da Onça, seus resultados sugeriram que havia jogador com mais probabilidade de vitória do que seu adversário.

González-Rodríguez [7] desenvolveu um jogo para celulares chamado Linja, através da estratégia minimax chegaram à conclusão que é possível aplicar aprendizado de máquina para identificar o vencedor em ambientes multiusuários, assim como a geração da rota com as jogadas ótimas.

Goulart [8] desenvolveu sete diferentes protótipos de inteligência artificial para a resolução do jogo chamado 2048, sendo um com comportamento aleatório e os demais baseados no algoritmo minimax, com e sem aplicação de poda alfa-beta, aplicando variações de recompensa, profundidade e geração de estados para o jogador ‘min’.

Connelly [9] utilizou a linguagem Python para desenvolver o jogo de tabuleiro Othello, onde foram desenvolvidas diferentes tipos estratégias, passando por jogadas que prevalecem em uma direção, jogadas baseadas em um tabuleiro de suporte e o jogador minimax.

III. METODOLOGIA

A. Jogo dos Dedos

Trata-se de um jogo disputado entre duas ou mais pessoas, utilizando simplesmente as mãos dos jogadores para realizar operações como adição e divisão.

Com o intuito de simplificar a explicação de suas regras, será considerada uma partida entre duas pessoas apenas, sendo elas ‘A’ e ‘B’. O primeiro passo é definir qual jogador iniciará a partida, para esse exemplo considera-se o jogador ‘A’, então ambos jogadores apresentam suas mãos com o dedo indicador esticado, e os demais dedos recolhidos, conforme Fig. 1.

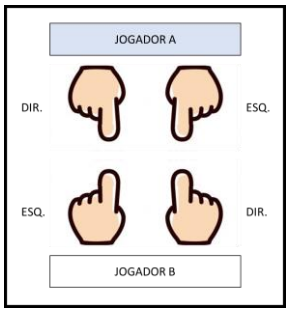

Fig. 1. Representação do início do jogo iniciando com jogador 'A'. Fonte: Elaborada pelo autor.

Uma jogada é realizada quando um jogador toca uma de suas mãos em uma das mãos do adversário, dessa forma, a nova quantidade de dedos esticados na mão tocada do adversário será a soma dos dedos esticados das mãos em contato, suponha que a primeira jogada de 'A' seja tocar sua mão direita na mão esquerda do adversário, portanto o adversário deverá esticar um dedo a mais na sua mão esquerda, totalizando então dois dedos, ilustrado na Fig. 2. O jogo continua dessa forma alternando a vez de cada jogador.

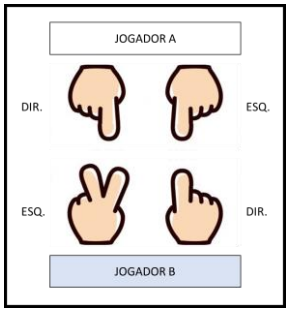

Fig. 2. Representação do turno do jogador 'B' após uma jogada de 'A'. Fonte: Elaborada pelo autor

De forma geral, cada jogada é representada pela fórmula (1), onde 'a' é a quantidade de dedos esticados na mão do jogador que está **recebendo a jogada** e 'b' a quantidade de dedos esticados da mão do jogador que está **realizando a jogada**. Portanto, se por acaso a soma dos dedos de um dado jogador for sete, então o novo valor de sua mão será dois.

$$a \equiv a + b \ (mod\ 5) \qquad (1)$$

Quando um jogador tiver cinco dedos esticados então esta mão estará fora do jogo e não poderá ser mais utilizada pelo jogador e nem ser alvo do adversário. Considera-se a partida encerrada no momento em que um jogador 'perder' suas duas mãos, sendo a vitória designada para seu oponente.

Existe também a jogada de divisão da mão, esta jogada é válida quando o jogador possui apenas uma mão em jogo e a mesma possui um número par de dedos esticados, quando optado por realizar essa jogada então sua outra mão retorna para o jogo, e agora ambas as mãos passam a ter o valor da metade de dedos esticados em sua mão antes da divisão, a Fig. 3 é uma ilustração da jogada de divisão realizada pelo jogador 'B'.

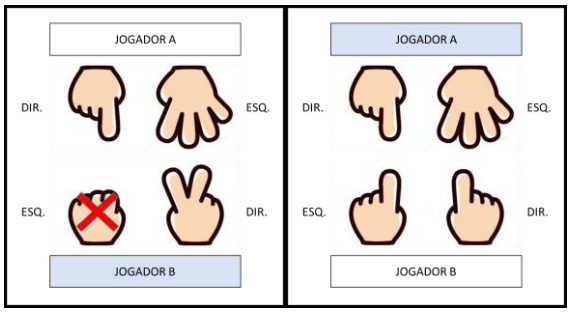

Fig. 3. Representação da jogada de divisão realizada pelo jogador 'B'. Fonte: Elaborada pelo autor.

B. Busca competitiva

A interação dos humanos com jogos permite que seja exercitada a habilidade de planejamento, pois a pessoa deve pensar qual jogada o adversário realizará a partir de cada uma de suas próprias jogadas possíveis, e tomar a melhor decisão como forma de obter vantagem sobre seu adversário. A busca competitiva ocorre justamente nesse tipo de contexto apresentado, ou seja, quando há mais de um agente realizando ações sobre um estado visando seu próprio sucesso, tal ambiente é conhecido como jogo, carregando consigo componentes como, estados, jogadores, ações, resultados, testes de término e utilidade, sendo explicadas a seguir dentro do contexto do Jogo dos Dedos [10].

- **Estados**: Configuração do valor de cada mão dos jogadores, e o jogador que realizou o movimento.
- **Jogadores**: Define qual jogador realizará ação sobre o estado.
- **Ações**: Conjunto de possíveis movimentos válidos em um estado, nesse caso representado por no máximo quatro movimentos de ataque, ou seja, que altera o valor da mão do adversário, ou um movimento de defesa, quando o jogador realiza a divisão da mão.
- **Resultados**: Nova configuração do estado após uma ação de ataque ou defesa. Desconsiderando estados simétricos, por exemplo, o estado onde um jogador está com o valor um na mão esquerda e dois na mão direita é o mesmo quando o jogador está com o valor dois na mão esquerda e um na mão direita.
- **Teste de término**: Verificação do tipo verdadeiro ou falso para verificar se um jogo chegou ao final, ou seja, se um dos jogadores 'perdeu' suas duas mãos.
- **Utilidade**: Função que retorna um valor numérico de um estado terminal na vez de um agente, como forma de distinguir as melhores jogadas (que conduzem para uma vitória) das piores jogadas (que conduzem para uma derrota).

C. Algoritmo minimax

Esse algoritmo é responsável por realizar a melhor jogada a partir de um estado atual e da árvore de busca, onde é calculada a função utilidade do estado a partir de novas sucessivas jogadas, considerando que cada jogador esteja jogando de forma ótima, portanto é pretendido maximizar os resultados do jogador a partir da minimização dos resultados do adversário. A função utilidade será aplicada em cada nó folha a partir do estado atual e os melhores resultados propagados recursivamente.

Com objetivo de clarear essa estratégia, na Fig. 4 está indicado uma árvore hipotética em que foi aplicado o algoritmo minimax, o nó A é o estado atual que se deseja maximizar, os nós B e C são os estados sucessores os quais o adversário deseja minimizar, os círculos são os nós folhas após jogada do adversário, o estado B possui o valor minimax 1 pois é o melhor resultado que pode obter a partir das folhas, de forma análoga o estado C possui o valor -2, como o estado atual A pretende maximizar sua pontuação, então sua escolha de jogada é a que leva ao estado B, por possuir maior valor minimax.

Como ambientes competitivos tendem a ter inúmeros estados possíveis, buscar nós folhas é uma tarefa muito custosa, para diminuir esse problema costuma-se utilizar um fator limiar, o qual indica o número máximo de níveis buscado na árvore.

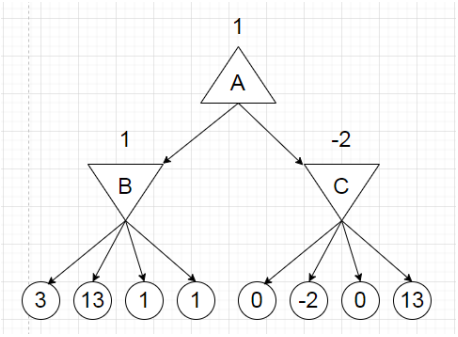

Fig. 4. Exemplo de árvore com aplicação do minimax. Fonte: Elaborada pelo autor.

D. *Algoritmo minimax com poda alfa-beta*

A poda alfa-beta aplicada no algoritmo minimax é uma estratégia para otimizar o processo de busca ao longo da árvore, evitando a exploração de todas as ramificações, já que podem existir estados que não vão influenciar na decisão de escolha do minimax [11]. Importante ressaltar que o valor retornado é o mesmo do algoritmo minimax apresentado anteriormente. A Fig. 5 ilustra um exemplo de poda, observe que o melhor resultado que o jogador 'max' pode obter da subárvore B é 2, ao realizar a busca na subárvore C é encontrada na primeira ramificação o valor 1, portanto, como a subárvore A pretende retornar o maior valor de seus descendentes então independente de qual seja o valor da próxima ramificação da subárvore C não será interessante para A, já que 1 < 2 sendo vantajoso para o jogador 'min'. O mesmo ocorre quando a subárvore C retorna o valor de sua primeira ramificação, nesse caso -2.

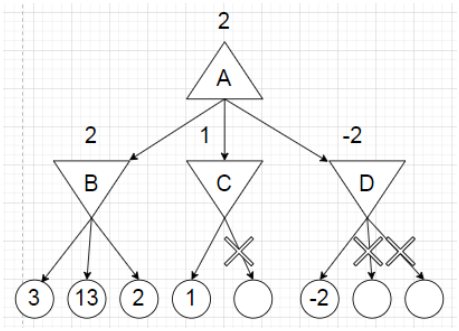

Fig. 5. Exemplo de árvore aplicada poda alfa-beta. Fonte: Elaborada pelo autor.

E. *Programação Orientada a Objetos*

A Programação Orientada a Objetos (POO) é uma abordagem que permite abstrair objetos do mundo real e representa-los dentro de programas computacionais, com seus atributos (características), métodos (comportamentos) e mensagens (comunicação entre objetos). Quando os objetos compartilham uma certa interface são agrupados em classes, construídos a partir dos componentes citados anteriormente, um exemplo clássico é o carro, existem diversos modelos de carro, no entanto todos compartilham atributos como cor, número de chassi, tamanho, motor e etc., e de comportamentos como acelerar, frear, buzinar, abrir e fechar portas, vidro e etc. [12]

Segundo Isotani [13], a POO possui vantagens como abstração de dados na representação de atributos, possibilidade de combinar interfaces ao se construir classes, diminuição de complexidade pois as classes tornam-se especialistas em uma determinada tarefa, reutilização de código durante o desenvolvimento de um software, tornar o software extensível, uma vez que novas classes podem ser construídas a partir de outras já existente (herança), e por fim, a facilidade de realizar manutenções no código, já que estão separados em módulos de forma natural.

IV. ANÁLISE EXPERIMENTAL

A. *Estratégia na implementação do jogo*

1) *Árvore de busca*

A partir das configurações apresentadas na seção 'Busca competitiva' foi construída a árvore de busca do jogo em seus primeiros níveis, sendo que cada nó representa um estado e as arestas o movimento que um jogador fez para sair do estado atual e alcançar o novo estado. A Fig. 6 indica um trecho da árvore do jogo, onde os nós possuem o valor de cada mão dos jogadores, o sinal entre as mãos indica quem fará a próxima jogada, por exemplo, para (2,1) < (3,1), entende-se que o jogador com a mão (3,1) realizará a jogada sobre o adversário com a mão (2,1), podendo gerar (2,2), (2,4), (3,1) ou (0,1) como novas mãos para o seu adversário, por fim o valor abaixo das mãos é o resultado da função utilidade aplicada ao nó, conforme (2).

$$p = m_{adv} - m_{jog} + b_1 + b_2 \qquad (2)$$

Onde:

- p = pontuação de um estado
- m_{adv} = soma dos dedos das mãos do adversário
- m_{jog} = soma dos dedos das mãos do jogador atual
- b_1 = bônus de 10 pontos para cada mão fora do jogo do adversário.
- b_2 = bônus de 100 pontos quando o jogador atual elimina as duas mãos do adversário (vence o jogo)

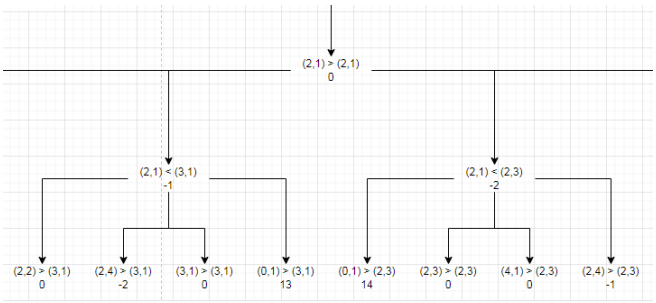

Fig. 6. Esquematização de um trecho da árvore do Jogo dos Dedos. Fonte: Elaborada pelo autor.

Para gerar a árvore completa deste jogo é necessário listar todos os estados distintos, primeiro buscou-se encontrar o número de possíveis jogadas de um jogador através da fórmula (3), sendo composta pela parcela da combinação de seis mãos tomadas duas a duas acrescida de seis movimentos que representam as mãos de mesmo valor, resultando em 21 mãos, por fim aplicou-se a fórmula (4), ao elevar ao quadrado o valor encontrado anteriormente para contar quantas mãos existem considerando as mãos do jogador com as mãos do adversário e então multiplicado por dois para obter os estados iniciando com o jogador e com o adversário.

$$maos = \frac{6!}{2!4!} + 6 = 21 \quad (3)$$

$$estados = 2 \cdot (maos)^2 = 882 \quad (4)$$

2) Construção dos jogadores

Para a construção do jogo foi utilizada a POO para abstrair as regras do jogo e seus jogadores com suas devidas estratégias. Inicialmente construiu-se a classe Jogador, que possui alguns atributos como valor das mãos, cor do jogador e nome do jogador, a seguir foi construída a classe Estado, a qual deve receber em sua criação dois jogadores de cores distintas, o número do turno, uma lista com os próximos estados e seu antecessor, além desses atributos essa classe também realiza algumas ações como, verificação se o estado atual é terminal, ou seja, se o jogador atual não têm mais mãos disponíveis, possui também a ação de retornar o adversário, adicionar sucessores e retornar o resultado de uma partida.

Após a criação dessas classes principais, foram abstraídas seis tipos de estratégias para a condução do jogo, as quais são herdadas de Jogador, sendo elas:

- Aleatório: a partir de cada estado, o jogador lista as possíveis jogadas válidas e escolhe de forma aleatória seu próximo movimento.
- Aleatório Ataque: quando uma das possíveis jogadas resultar na eliminação de alguma mão do adversário então a mesma será selecionada, caso contrário, é escolhido de forma aleatória o seu próximo movimento.
- Aleatório Defesa: esse jogador possui a estratégia de realizar o processo de divisão da mão, como forma de recuperar sua outra mão para o jogo, caso contrário uma jogada é escolhida de forma aleatória.
- Ataque e Defesa: uma combinação dos três jogadores anteriores, a ordem de preferência é realizar a jogada de ataque para eliminar a mão do adversário, caso contrário se possível é realizada a jogada de divisão da mão, se não, uma jogada é escolhida de forma aleatória.
- Humano: não se trata de uma estratégia, mas sim de um modo de jogo, habilitando a possibilidade de um humano jogar o jogo, o controle é simplesmente informar o número da mão que será realizada a jogada e o número da mão do adversário que receberá a jogada, ou selecionar a opção de divisão da mão.
- Minimax sem poda: aplicação do algoritmo minimax, o qual realiza a busca em todas as ramificações da árvore de busca a partir de um estado atual até o nível máximo de busca informado pelo usuário.
- Minimax com poda: similar ao jogador anterior, diferindo apenas na aplicação da poda alfa-beta para otimizar o espaço de busca.

Ao final o foco foi realizar a construção do código para estratificar os resultados das partidas, com o tempo de jogo, número de vitórias do jogador, número de vitórias do adversário, número de empates, partida finalizada com mínimo e máximo de jogadas, e por fim, a mediana de jogadas das partidas vencidas por cada jogador, conforme indicado na Fig. 7. Além disso, também são gerados arquivos com a representação visual de cada partida, um trecho de uma partida é exibido na Fig. 8 com uma jogada de divisão realizada pelo jogador Aleatório.

```
----------------------RESUMO-------------------------
|                                                   |
| Tempo de jogo: 0:00:00.717601                     |
| Total de jogos: 1000                              |
| Jogadas permitidas: 200                           |
|                                                   |
| Vitorias AleatorioAtaque [RED] : 536 (53.6%)      |
| Vitorias AtaqueDefesa [BLUE] : 464 (46.4%)        |
| Empates : 0 (0.0%)                                |
|                                                   |
| AleatorioAtaque [RED]                             |
| -Vitoria mais rapida: 5 jogadas (partida 5)       |
| -Vitoria mais longa: 45 jogadas (partida 296)     |
| -Mediana das vitorias: 5 jogadas                  |
|                                                   |
| AtaqueDefesa [BLUE]                               |
| -Vitoria mais rapida: 6 jogadas (partida 15)      |
| -Vitoria mais longa: 40 jogadas (partida 382)     |
| -Mediana das vitorias: 10 jogadas                 |
-----------------------------------------------------
```

Fig. 7. Exemplo de arquivo de resumo de uma partida. Fonte: Elaborada pelo autor.

```
-----------------------------------------------
          Rodada:7 - Jogador: [RED]
-----------------------------------------------

      Aleatorio [RED]:              - ||||

  AleatorioDefesa [BLUE]:     ||||  -  |||

-----------------------------------------------
          Rodada:8 - Jogador: [BLUE]
-----------------------------------------------

  AleatorioDefesa [BLUE]:     ||||  -  |||

      Aleatorio [RED]:          ||   -  ||
```

Fig. 8. Trecho de uma partida entre Aleatório e AleatorioDefesa, onde o jogador Aleatorio realizou uma jogada de divisão. Fonte: Elaborada pelo autor.

3) Funcionamento do jogo

Ao iniciar o jogo é solicitado ao usuário que sejam escolhidos dois jogadores, ao primeiro jogador será atribuído ao atributo cor o valor 'BLUE' enquanto que para o segundo será 'RED'. Caso algum dos jogadores seja da categoria

minimax, então é solicitado ao usuário que informe o número máximo de busca na árvore. A seguir o usuário deve definir o número de partidas e o número máximo de turnos de cada partida. A partir dessas entradas as partidas são executadas e computados os resultados para a geração dos arquivos mencionados anteriormente.

B. *Configuração do algoritmo e do ambiente computacional*

O algoritmo foi construído na linguagem Python e executado em um computador desktop com sistema operacional Windows 7 Ultimate, processador Intel Core i3, memória RAM de 4GB com disco rígido de 500GB.

C. *Critérios de análise*

A análise foi realizada diretamente a partir dos arquivos de resumo gerados em cada partida, sendo avaliado os números de vitórias, número de jogadas nas partidas vencidas e tempo de jogo. Cada confronto foi configurado em mil partidas, sendo metade iniciado pelo jogador selecionado e a outra metade iniciado pelo adversário.

D. *Resultados e discussão*

Apesar de ser conhecida a quantidade de estados distintos, tornou-se impraticável a construção da árvore completa do jogo, agravado ainda mais pela existência de estados repetidos, gerando uma sequência de movimentos iguais, assim como representado na Fig. 9.

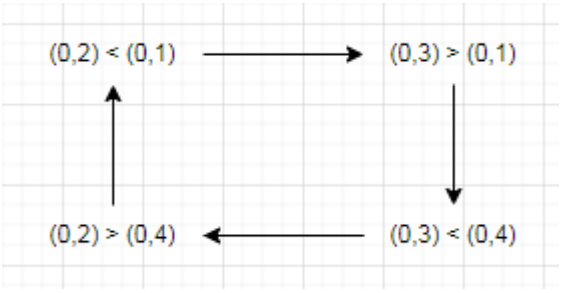

Fig. 9. Exemplo de looping no Jogo dos Dedos. Fonte: Elaborada pelo autor.

As primeiras partidas foram realizadas entre os quatro tipos de jogadores aleatórios, pela Fig. 10 nota-se que os jogadores que obtiveram maior número de vitórias foram o AleatorioAtaque e AtaqueDefesa, os quais possuem estratégia gulosa, estando próximo as jogadas realizadas nas partidas presenciais com familiares. Destaca-se também o baixo número de vitórias do jogador Aleatorio e a demora para vencer o jogador AleatorioDefesa, conforme Fig. 11.

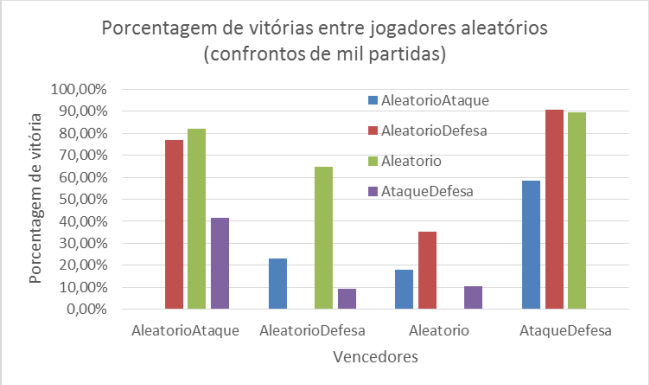

Fig. 10. Gráfico com resultado das partidas entre jogadores aleatórios em porcentagem. Fonte: Elaborada pelo autor.

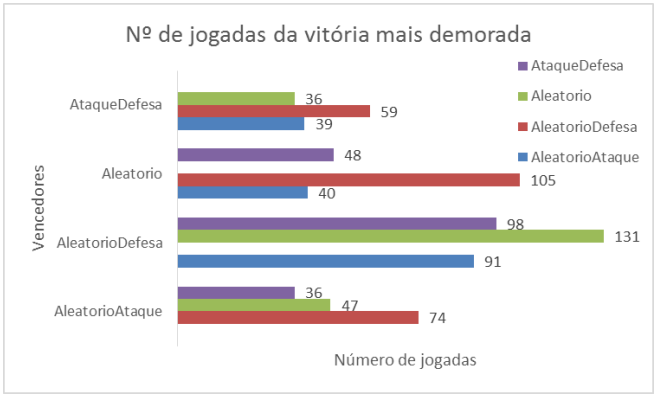

Fig. 11. Gráfico com o número de jogadas das partidas vencidas mais longas. Fonte: Elaborada pelo autor.

A seguir foi colocado em prova o Minimax contra os demais jogadores aleatórios, realizando a alteração do parâmetro de profundidade da busca, onde o melhor desempenho obtido foi através da utilização do nível dois, ou seja, o algoritmo Minimax tomava a decisão do ponto de vista da próxima jogada do adversário, comportamento também encontrado em partidas presenciais, no entanto ao aumentar o parâmetro de profundidade houve uma queda abrupta na quantidade vitórias, principalmente com relação às estratégias gulosas, como pode ser visto na Fig. 12.

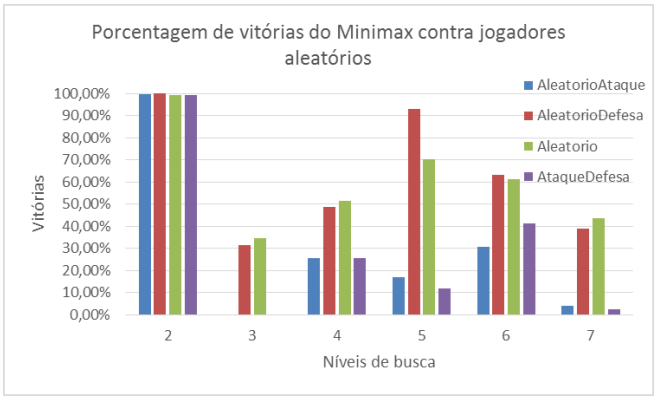

Fig. 12. Gráfico de vitórias do Minimax com diferentes níveis de busca contra jogadores aleatórios. Fonte: Elaborada pelo autor.

Com relação ao número de jogadas realizadas nas partidas vencidas pelo Minimax, notou-se que exceto nas partidas contra o jogador AleatorioDefesa, os demais adversários foram derrotados antes de se atingir a vigésima jogada (Fig. 13).

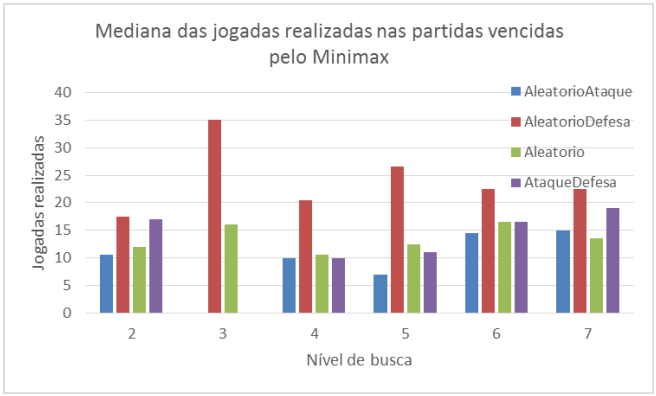

Fig. 13. Gráfico da mediana das jogadas nas partidas vencidas pelo Minimax. Fonte: Elaborada pelo autor.

Por fim, foi explorado o desempenho do algoritmo Minimax utilizando a poda alfa-beta, em comparação ao mesmo algoritmo sem a poda, tal comparação ocorreu ao realizar partidas de cada Minimax contra o adversário AleatorioDefesa, por ser um jogador com estratégia defensiva, e como consequência as partidas são mais demoradas, os resultados indicaram que a poda alfa-beta reduziu em aproximadamente 70% o tempo de jogo, conforme Fig. 14.

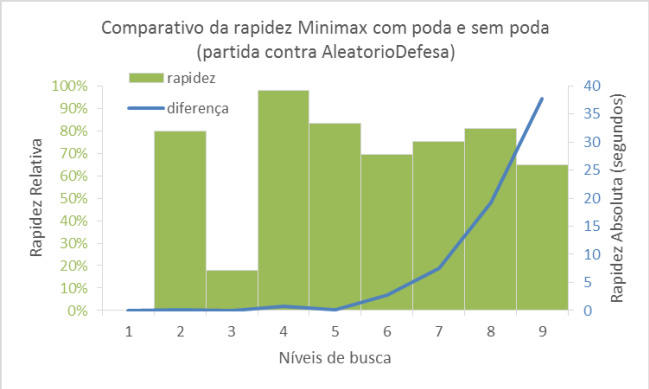

Fig. 14. Gráfico de comparação do processamento Minimax com poda e sem poda. Fonte: Elaborada pelo autor.

V. CONCLUSÃO

O Jogo dos Dedos mostra-se como uma iniciativa de exercitar operações matemáticas de forma lúdica, podendo ser um gatilho para quebrar bloqueios mentais no impedimento do aprendizado da matemática.

A construção desse projeto possibilitou a prática da programação orientada a objeto, tornando o código mais legível e intuitivo para acrescentar novas regras ou realizar modificações.

Observou-se que as estratégias dos jogadores AleatorioAtaque, AtaqueDefesa e Minimax com nível de busca 2 se aproximaram as estratégias utilizadas nas partidas presenciais, sendo que Minimax obteve os melhores resultados, vencendo praticamente todas as partidas. Além disso, confirmou-se que os jogos que envolviam o jogador AleatorioDefesa tinham as partidas mais demoradas, assim como esperado. A pior estratégia identificada foi do jogador Aleatorio, o qual obteve o menor número de vitórias acumuladas. Partidas realizadas entre o jogador AleatorioDefesa e Minimax, com poda e sem poda, evidenciaram que o tempo do jogo foi reduzido a cerca de 70% devido a utilização do método da poda alfa-beta

Ao aumentar o nível de profundidade da busca do jogador Minimax, notou-se que o número de vitórias foi reduzido significativamente em comparação à busca de profundidade 2. A busca ou desenvolvimento de uma heurística que melhore o desempenho desse jogador é deixado como proposta para trabalhos futuros, já o código desenvolvido pode ser encontrado no seguinte link, https://github.com/fmoliveira13/JogoDedos.

REFERÊNCIAS

[1] ZATTI, F.; AGRANIONIH, N. T.; ENRICONE, J. R. B. Aprendizagem matemática: **desvendando dificuldades de cálculo dos alunos**. Perspectiva, v. 34, n. 128, p. 115-132, 2010. Disponível em: https://uricer.edu.br/site/pdfs/perspectiva/128_142.pdf. Acesso em: 28 jul. 2022.

[2] PACHECO, M. B.; ANDREIS, G. S. L.. Causas das dificuldades de aprendizagem em Matemática: percepção de professores e estudantes do 3º ano do Ensino Médio. Revista Principia - Divulgação Científica e Tecnológica do IFPB, João Pessoa, n. 38, p. 105-119, fev. 2018. ISSN 2447-9187. Disponível em: https://periodicos.ifpb.edu.br/index.php/principia/article/view/1612. Acesso em: 28 Jul. 2022.

[3] TRETTEL, U. R.; BATISTA, E. C. A importância da brincadeira no processo de ensino e aprendizagem na educação infantil. p. 18 – 21. 2016. Disponível em: https://www.researchgate.net/publication/331008449_A_IMPORTANCIA_DA_BRINCADEIRA_NO_PROCESSO_DE_ENSINO_E_APRENDIZAGEM_NA_EDUCACAO_INFANTIL. Acesso em: 28 jul. 2022.

[4] FIORENTINI, D. et al. Uma reflexão sobre o uso de materiais concretos e jogos no Ensino da Matemática. Boletim da SBEM-SP, v. 4, n. 7, p. 5-10, 1990. Disponível em: https://www.academia.edu/9027307/Metodologia_do?bulkDownload=thisPaper-topRelated-sameAuthor-citingThis-citedByThis-secondOrderCitations&from=cover_page. Acesso em: 28 jul. 2022.

[5] SOUZA, M. C. **A utilidade do algoritmo minimax em jogos digitais**. 2014, 63 f. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Faculdade do Espírito Santo, Multivix Cachoeiro de Itapemirim, Cachoeiro de Itapemirim. Disponível em: https://multivix.edu.br/wp-content/uploads/2018/08/a-utilidade-do-algoritmo-minimax-em-jogos-digitais.pdf. Acesso em: 15 jul. 2022.

[6] GUEDES, E. B.; NASCIMENTO, F. G. L. Uma Análise Comparativa de Funções de Utilidade para o Algoritmo Minimax Aplicado ao Jogo da Onça. *In*: ENCONTRO NACIONAL DE INTELIGÊNCIA ARTIFICIAL E COMPUTACIONAL (ENIAC), 16., 2019, Salvador. **Anais** […]. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 49-59. DOI: https://doi.org/10.5753/eniac.2019.9271. Disponível em: http://www.scielo.org.co/pdf/rfing/v31n59/2357-5328-rfing-31-59-e202.pdf. Acesso em: 15 jul. 2022.

[7] GONZÁLEZ-RODRÍGUEZ, C. D.; GONZALEZ-SANABRIA, J. S.; RINCÓN-DÍAZ, H. J.; SUÁREZ-BARÓN, M. J. Linja: A Mobile Application Based on Minimax Strategy and Game Theory. Revista Facultad de Ingeniería, vol. 31 (59), e14136, 2022. https://doi.org/10.19053/01211129.v31.n59.2022.14136. Acesso em: 27 jul. 2022.

[8] GOULART, D. S. **Um estudo sobre a aplicação de inteligência artificial em jogos**. 2014, 77 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade Federal do Pampa, Alegrete. Disponível em: https://repositorio.unipampa.edu.br/jspui/handle/riu/1589. Acesso em: 27 jul. 2022.

[9] CONNELLY, D. Othello. Disponível em: http://dhconnelly.com/paip-python/docs/paip/othello.html. Acesso em: 27 jul. 2022.

[10] NORVIG, Peter. **Inteligência Artificial**. [s.l]: Grupo GEN, 2013. ISBN 9788595156104. p. 30 – 50. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595156104/. Acesso em: 20 jul. 2022.

[11] FARIA, F. A. Busca competitiva (jogos adversariais). Inteligência Artificial. Universidade Federal de São Paulo, 2017. Disponível em: https://www.ic.unicamp.br/~ffaria/ia1s2017/class06/class06-Busca_competitiva.pdf. Acesso em: 25 jul. 2022.

[12] HENRIQUE, J. Programação orientada a objetos e programação estruturada. Alura, 2019. Disponível em: https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos. Acesso em: 25 jul. 2022.

[13] ISOTANI, S. Aula 1 Introdução à Orientação a Objetos (OO) e UML. Análise e Projeto Orientados a Objetos. Departamento de Sistemas de Computação, Universidade de São Paulo. Disponível em: https://edisciplinas.usp.br/pluginfile.php/1947303/mod_resource/content/1/Aula01-IntroducaoOO.pdf. Acesso em: 25 jul. 2022.