Utilizando técnicas de Inteligência Artificial para criar um oponente inteligente de Blackjack

Gustavo Lopes Rodrigues
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica
de Minas Gerais (PUC-MG)
Belo Horizonte, Brasil
Email: gustavolr@gmail.com

Homenique Vieira Martins
Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica
de Minas Gerais (PUC-MG)
Belo Horizonte, Brasil
Email: homenique.t@gmail.com

***Resumo*—A modelagem de oponentes inteligentes para jogar jogos de azar é um tópico que existe a decadas, e possui diferentes estratégias para abordar. Este estudo tem como objetivo encontrar o modelo de inteligência artificial com maior taxa de acerto e performance, para jogar o jogo de cartas 21, também conhecido como Blackjack. No processo para solucionar esse problema, foi utilizado a Godot Engine e sua linguagem GDScript para criar a interface assim como a lógica para modelar sistema de tomada de decisão do oponente. As estratégias testadas neste estudo incluem, Card Counting e Expectimax. Os resultados dos testes neste estudo demonstra que apesar da Expectimax conseguir encontrar resultados de certa forma semelhante ao Counting Card, a estratégia falha em ser viável devido ao seu tempo de execução e por não superar estratégias clássicas para a resolução desse problema.**

***Index Terms*—Inteligência Artificial, Jogos de Azar, Blackjack, Godot Engine**

I. Introdução

No ramo de inteligência artificial, existe um mercado amplo onde têcnicas dessa área são usadas para criarem oponentes inteligentes, que possam derrotar qualquer jogador. Alguns exemplos famosos a serem citados são: a Deep Blue da IBM que conseguiu derrotar o ex-campeão mundial de xadrez Garry Kasparov, ou, o AlphaGo da Google, que derrotou o campeão Fan Hui. Todos os jogos citados anteriormente são jogos determinísticos, ou seja, não são jogos com fatores probabilísticos envolvidos na hora de calcular a jogada. Um dos maiores desafios desta área da computação é como tratar os jogos de azar, onde a sorte é um elemento que precisa ser calculado para a realização das jogadas.

Este estudo tem como objetivo criar uma aplicação prática que possa ser usada para criar um jogador inteligente de Blackjack(no Brasil conhecido como 21). Além disso iremos avaliar diferentes estratégias/políticas e avaliar a utilidade esperada de cada estratégia para a inteligência artificial.

Este artigo está dividido da seguinte maneira: primeiros na secção 2 será explorado um pouco da literatura já existente do assunto. Na secção 3 as regras do Blackjack serão expostas e analisadas no contexto da aplicação que será feita assim como também será apresentado a Godot Engine que é o framework utilizado criar a interface e a lógica do jogo. Na secção 4 será analisado as diferentes estratégias e políticas utilizadas no programa. Na secção 5 acontecerá uma análise dos resultados chegados, a precisão de cada modelo e o tempo de execução. Por fim, terá a conclusão, onde será dado uma opinião final sobre os dados chegados, além de possíveis futuros trabalhos que podem ser feitos a partir deste.

II. Trabalhos Correlatos

Um padrão muito comum encontrado na literatura do assunto que está sendo abordado, é o fato de que os algoritmos de aprendizado de máquina, que usam de técnicas avançadas de inteligência artificial, ainda estão em sua juventude neste campo. Como exemplo, tem o artigo de Raphaela [1], que utiliza de tais técnicas para formar regras para maximizar as vantagens do jogador em relação ao cassino. Entretanto, os algoritmos em geral chegaram a um número pequeno de regras, o que fazia com que certas jogadas não contemplassem as configurações do jogo.Marvin e Fernand [2]. também tentou utilizar uma abordagem moderna utilizando uma arquitetura cognitiva chamada CHREST, entretanto, apesar que tal modelo tenha chegado a bons resultados, foi concluído que tal arquitetura seria melhor utilizada em jogos com um número maior de padrões a serem memorizados e reconhecidos, como o jogo Poker. Por fim temos Robert [3] que além de usar uma abordagem moderna com aprendizado genético, também utilizou de estratégias usando inteligências artificiais baseadas em regras, chegando ao resultado que ainda existe espaço a ser otimizado nas estratégias atuais, enquanto que as abordagens mais tradicionais continuam produzindo melhores resultados de maneira geral.

Todos os trabalhos até então mencionados tentam de alguma forma ou outra criar um jogador inteligente de Blackjack para aplicações, Raphaela [1] criou uma aplicação em C, enquanto que tanto Marvin e Fernand [2] e Robert [3] criam aplicações em Java. Nosso objetivo é também criar um jogo, porém que possa ser feito em uma linguagem de alto nível e que tenha uma utilização mais fácil, por isso a linguagem escolhida foi a GDScript da Godot Engine, que possui uma tipagem parecida com Python, mas que já foi construída para criar jogos. Além disso, tentaremos utilizar uma abordagem utilizando Expectimax.

III. Metodologia ou Proposta de Arquiteturas

Antes de compreender a metodologia aplicada a este estudo, é preciso entender a natureza do problema que estudo quer resolver, em outras palavras, é preciso dar uma introdução ao jogo 21 e quais regras do jogo serão usados na simulação feita neste estudo.

A. Entendendo 21 e as regras aplicadas

O 21 [4] (também conhecido como Blackjack) é um jogo de cartas francês onde pode ser jogado por um ou mais jogadores sempre todos contra o dealer, e para ganhar é necessário ter uma pontuação maior que a do dealer e menor ou igual a 21. Essa pontuação é proveniente dos valores das cartas, onde cartas com figuras exceto os "ás" possuem valor igual a 10, o "ás" pode valer 1 ou 11 caso o valor da mão não seja superior a 21, caso contrário o valor do ás se altera para 1 , as demais cartas tem o valor igual ao que está escrito na carta.

No início de toda rodada cada jogador faz a sua aposta, e recebe duas cartas viradas para cima, junto a isso o dealer também recebe duas cartas, porém somente uma fica virada para o cima e ao final do jogo é revelada. Os jogadores podem executar algumas ações [5] sendo elas:

- **hit:** O jogador solicita mais uma carta ao dealer
- **Stand:** O jogador não vai mais solicitar cartas
- **Dobrar:** Somente após receber as duas primeiras cartas o jogador pode dobrar a aposta e em consequência receber uma nova carta.
- **Split:** Após receber a primeira duas cartas e caso elas possuam o mesmo valor, o jogador pode dividir essa mão em duas novas mão
- **Surrender:** O jogador desiste do jogo e perde metade da aposta

Para o propósito do nosso trabalho colocamos apenas as duas funções mais simples do jogo, hit e stand.

Com o jogo e suas regras introduzidos, também seria importante apresentar a tecnologia utilizado neste trabalho, a Godot Engine e sua linguagem GDScript.

B. Godot Engine

A Godot engine [6] é um motor gráfico para jogos *cross-platform* de código aberto, tendo sua primeira versão lançada em 2014 pelos Argentinos Juan Linietsky e Ariel Manzur. O propósito da Godot é criar um ambiente completo para o desenvolvimento de jogos 2D e 3D, com licença aberta, logo, os desenvolvedores tem todos os direitos em cima da sua criação (sem *royalties*). Alem disso, ela está disponível para todas as plataformas (Linux, macOs, Windows).

Uma das grandes vantagens de trabalhar com a Godot Engine, e o motivo pelo qual esta foi escolhida para realizar este estudo, se deve ao fato que a Godot Engine facilita imensamente o desenvolvimento dos jogos, com sua interface fácil de aprender e sua linguagem de programação GDScript, que possui um sistema de tipagem bem parecido com linguagens de alto nível como Python e Javascript. Por fim, a Godot Engine ainda possui suporte a GDNative, permitindo que os jogos sejam programados em C ou C++ e pacotes da comunidade como o Godot-python [7], permite rodar códigos de Python dentro da engine, abrindo as portas para o uso de aprendizado de máquina.

Com a base do estudo edificada, é possível agora apresentar as metodologias.

C. Metodologias utilizadas

Para esse trabalho o objeto de discussão é a utilização do algoritmo de Expectimax e para a base de comparação será utilizado o algoritmo de Contar Cartas.

A escolha do expectiminimax se dá, pois em um cenário de um jogo de soma zero como o blackjack, para um jogador vencer é necessário que o outro jogador perca , e onde o algoritmo de expectiminimax se encaixa perfeitamente pois além de buscar maximizar os ganhos e minimizar as perdas,dado o algoritmo que foi baseado ,o minimax, ele adiciona a árvore de avaliação de cenários, um peso probabilístico, o que encaixa perfeitamente no cenário do blackjack onde existe uma probabilidade de uma carta vim.

Em contrapartida temos o algoritmo de contar carta, que neste artigo iremos usar o estratégia Hi-Lo, onde é atribuído um valor mais um as cartas de dois a seis, do sete ao novo é atribuído o valor zero e as cartas restantes o valor menos um, tendo isso em mente o jogador vai somar todas as cartas que são dadas na mesma e se a contagem tiver positiva ele tem uma maior chance e em consequência ele pode apostar valores maiores caso contrário ele evita fazer novas aposta naquele jogo. O uso da estratégia Hi-Lo neste artigo serve como uma métrica de comparação pois, como está estratégia já é bem estabelecida e ter resultados consistentes, ela é um ótimo fator de comparação. [8] [9]

IV. Avaliação dos Resultados

Com as metodologias explicadas, é possível analisar o desempenho de cada uma das estratégias em partidas controladas de 21 dentro da aplicação que foi criada. Antes de ir direto para os resultado, seria interessante como os testes foram realizados e configurados para cada uma das situações.

A. Analisando a interface de testes

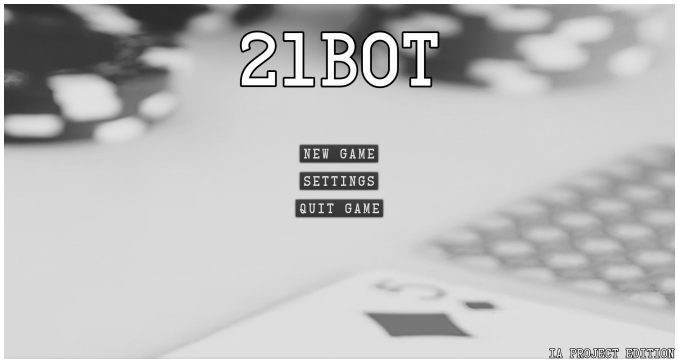

Figura 1. Tela do menu principal

Na figura 1 temos a tela do menu toda vez que o jogador rodar pela primeira vez a aplicação. O menu principal possui duas opções que são relevantes: *New game* e *Settings*

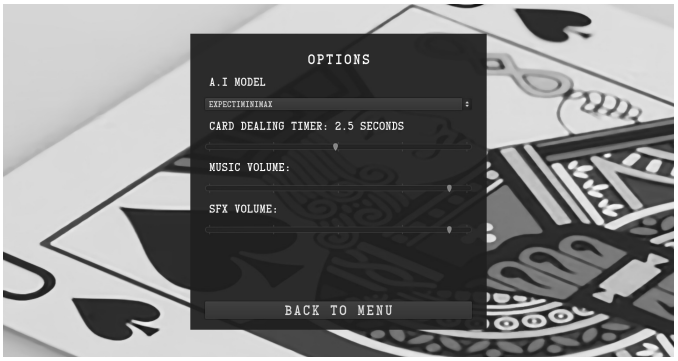

Figura 2. Tela do menu de configurações

Na tela de mostrada na figura 2 é possível escolher qual será a estratégia empregada pelo inimigo, que no caso as opções são: *Expectimax* e *Counting Cards*.

Além disso, é também configurar outras opções do jogo, como o tempo para as cartas descerem e o volume do jogo.

Por fim vamos a atração principal, o jogo em si:

Figura 3. Tela principal do jogo

Como é possível ver na figura 3, o jogador (na parte inferior) da tela, recebe duas cartas, enquanto que o inimigo(Inteligência Artificial) recebe duas cartas sendo uma delas viradas para cima e a outra para baixo. Em outras palavras o inimigo sempre será o dono da mesa(dealer).

Acima das carta, o jogador tem duas opções:

- *Hit*: O jogador receberá uma carta, se o jogador ficar com um placar superior a 21 pontos, o jogo irá terminar e a tela de Fim de jogo (Game Over) irá aparecer na tela. Caso contrário o jogo irá somar o resultado das cartas ao placar e exibir na tela
- *Stand*: O jogador irá passar seu turno para o oponente e não poderá comprar mais cartas.

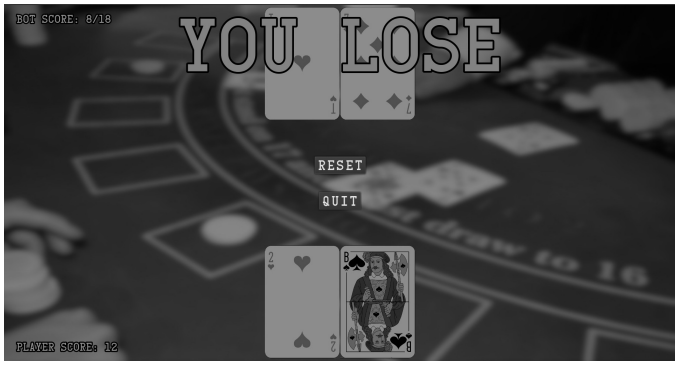

Figura 4. Tela quando o jogador perde

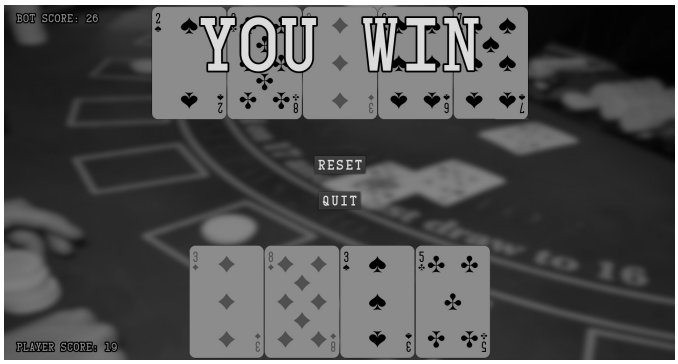

Figura 5. Tela quando o jogador ganha

B. Testes

A maneira em como foi realizado os testes, foi a partir do teste manual de cinquenta(50), cem(100) e cento e cinquenta(150) testes isolados, feito no total trezentos testes(300) com o intuito de analisar a precisão das diferentes estratégias e analisando a porcentagem de vitória de cada uma das políticas. Por fim, foi também analisado o tempo de execução das funções.

Tabela I
PRECISÃO DAS POLÍTICAS

Estratégias	**Porcentagem de vitória por número de testes**		
	50	100	150
Contar cartas	41.0%	41.4%	41.9%
Expectiminimax	38.4%	38.6%	38.9%

Tabela II
TEMPO DE EXECUÇÃO

Estratégias	Tempo médio de execução (em segundos)
Contar cartas	0.2
Expectiminimax	0.8

A partir da análise das tabelas I e II o Expectimax obteve resultado levemente piores que o algoritmo de contar cartas, o que pode indicar que a heurística escolhida [10] [11] [12] ou o que o fator de possibilidade não foram os

melhores , e com mais teste e ajuste poderia a ter resultados melhores que o contar cartas. Porém, por causa da sua complexidade e o seu tempo de execução sendo consideravelmente maior,o algoritmo do Expectimax neste cenários explorados não demonstrou-se uma solução viável para substituir uma técnica bem estabelecida como o contar cartas, mas em situações futuras com uma melhor heurística talvez em determinados cenários e também com um número mais substancial de testes, pode ser que o algoritmo em questão possa demonstra-se viável para uso.

V. Conclusões

Neste trabalho abordamos o uso de diferentes estratégias para criar um oponente inteligente de 21. O resultado encontrado demonstrou que a estratégia utilizada não se demonstrou ideal para o uso neste caso.

O fato do algoritmo não se demonstrar ideal para o Blackjack, pode ser apenas resultado de que poucas funcionalidades do jogo foram implementadas, incluindo uma que é essencial para o contesto dos cassinos, as apostas. Por fim, uma quantidade baixa de testes foram realizados, o que pode dificultar garantir se a baixa porcentagem que os testes conseguiram foi por causa da ineficiência do modelo ou pela baixa quantidade de testes em si realizados.

Uma proposta de trabalho futuro seria criar um sistema automatizado de testes, permitindo com que mais uma quantidade maior de testes possa ser extraída e analisada. Além disso também seria interessante adicionar as funcionalidades excluída destes testes, assim como fazer a adição de novas estratégias, principalmente as mais modernas que incorporam técnicas de aprendizado de máquina.

Referências

[1] K. R. Cardoso, "Um estudo sobre a definicao de estrategias para o jogo de blackjack usando tecnicas de aprendizagem de maquina e sistemas fuzzy," *Programa de Pós-Graduação em Ciência da Computação*, 2015.

[2] M. Schiller and F. Gobet, "A comparison between cognitive and ai models of blackjack strategy learning," 09 2012, pp. 143–155.

[3] G. P. de Oliveira, "Metodos de inteligência artificial aplicados em jogos baseados em turnos," *Repositório Institucional - Universidade Federal de Uberlândia*.

[4] Wikipedia contributors, "Blackjack — Wikipedia, the free encyclopedia," 2022, [Online; Acessado em 5 de Junho de 2022]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Blackjack&oldid=1091524833

[5] M. S. Isai Scheinberg, "Blackjack rules," 2001, [Online; Acessado em 5 de Junho de 2022]. [Online]. Available: https://www.pokerstarscasino.com/how-to-play/blackjack/rules/

[6] A. M. Juan Linietski, "Godot engine," Janeiro 2014. [Online]. Available: https://github.com/godotengine/godot

[7] E. Leblond, "Godot python, python support for godot," internet, Julho 2017. [Online]. Available: https://github.com/touilleMan/godot-python

[8] Desconhecido, "Hi-lo system," [Online; Acessado em 5 de Junho de 2022]. [Online]. Available: https://www.casinoguardian.co.uk/blackjack/hi-lo-blackjack-system/

[9] M. Stevens, "Why does everybody recommend the hi-lo card counting system?" 2021, [Online; Acessado em 5 de Junho de 2022]. [Online]. Available: https://www.gamblingsites.org/blog/why-does-everybody-recommend-the-hi-lo-card-counting-system/#:~:text=Works%20on%20Multi%2DDeck%20Games&text=The%20Hi%2DLo%20is%20one%20such%20strategy%20that%20allows%20you,are%20left%20in%20the%20shoe

[10] Desconhecido, "Introduction to the high-low card counting strategy," [Online; Acessado em 6 de Junho de 2022]. [Online]. Available: https://wizardofodds.com/games/blackjack/card-counting/high-low/

[11] ——, "Blackjack odds house edge explained," [Online; Acessado em 6 de Junho de 2022]. [Online]. Available: https://www.onlinegambling.com/blackjack/odds/#:~:text=The%20odds%20of%20winning%20at,of%20a%20loss%20at%2049.10%25.

[12] ——, "Win/loss/push data," [Online; Acessado em 6 de Junho de 2022]. [Online]. Available: https://www.blackjackincolor.com/truecount5.htm