UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

KEVIN BRAATHEN DE CARVALHO

Ambiente computacional para desenvolvimento de inteligência artificial aplicada a jogos de *Poker Texas Hold'em*

VIÇOSA
2016

KEVIN BRAATHEN DE CARVALHO

Ambiente computacional para desenvolvimento de inteligência artificial aplicada a jogos de *Poker Texas Hold'em*

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 490 – Monografia e Seminário e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.
Orientador: Prof. Dr.Leonardo Bonato Felix

VIÇOSA
2016

KEVIN BRAATHEN DE CARVALHO

Ambiente computacional para desenvolvimento de inteligência artificial aplicada a jogos de *Poker Texas Hold'em*

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 490 – Monografia e Seminário e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 19 de Dezembro de 2016.

COMISSÃO EXAMINADORA

Prof. Dr. Leonardo Bonato Felix - Orientador
Universidade Federal de Viçosa

Prof. Dr. Alexandre Santos Brandão - Membro
Universidade Federal de Viçosa

Mestre Gerson Pedruzi - Membro
Universidade Federal de Minas Gerais

“O desejo por conhecimento molda o homem”

Kvothe - O Temor do Sábio

Dedico esse trabalho aos meus amigos e familiares pelo apoio incondicional e em especial ao meu amor que nunca deixou de acreditar em mim e sempre me impulsionou. A vida é como uma caminhada e ela é feita das pessoas que andam com você, se hoje sou quem sou, é porque ontem encontrei vocês.

Agradecimentos

Gostaria de agradecer à UFV pela infra estrutura dada para minha formação, ao CNPq pelo investimento dado em minha carreira e aos meus professores por me guiarem nos caminhos do conhecimento.

Resumo

Pesquisas no campo de inteligência artificial têm encontrado no campo de jogos um nicho com muito espaço para desenvolvimento por ter regras fixas, escopo restrito e função dos jogadores bem definida. *Poker* tem atraído o foco de empenhos acadêmicos devido à sua estrutura de decisões de natureza estocástica com dados incompletos. Nesse trabalho é desenvolvido um ambiente de simulações de *poker texas hold'em* com opções de utilizar uma inteligência artificial contra outras personalizadas, com o intuito de facilitar o desenvolvimento dessas inteligências, tendo aplicações, inclusive, em ensino de engenharia. É então explicado o desenvolvimento por trás de uma inteligência artificial criada com lógica nebulosa que foi implementada no programa. Mostra-se o *software* resultante com detalhes de seu funcionamento e como adequar-se ao seu uso. Por fim são feitos testes com diferentes números e tipos de inteligências artificiais foram realizados para poder validar o ambiente computacional e a eficiência de uma inteligência artificial criada utilizando lógica nebulosa tendo tempos de simulação de cem jogos variando da ordem de 100 até 3000 segundos e com taxas de vitória variando até 73%.

Abstract

Researches in the artificial intelligence field have found in games a niche with a lot of room fo development due to the fact of having fixed rules, istrict scope and well defined player roles. *Poker* has been attracting the focus of academic researches due to its stochastic nature of decision making with incomplete data. In this work is developed a poker simulation environment with the options of using a custom artificial intelligence against other custom ones with the intent of making the improvement of those AI easier, having applications even on engineering teaching. Then is explained the development process of a artificial intelligence created using fuzzy logic that was implemented on the final program. It is shown the final *software* with deails about the way it works and how to operate it. Test games with different number and types of artificial intelligences are done to validate the simulation environment and the efficiency of a custom built AI using fuzzy logic having total simulation elapsed time for one hundred games in the order of 100 to 3000 seconds and with victory rates as high as 73%.

Sumário

1 *Introdução*

A maioria das pesquisas em inteligência artificial tem se voltado para a modelagem dos sintomas de sistemas inteligentes que podem ser observados. Em geral, o fator causal da inteligência é sobrepujado com a intenção de ter mais rapidamente os resultados de se ter uma inteligência artificial observável e não o que, de fato, a gera [8].

O ambiente de jogos tais como xadrez, gamão, *Go*, ligue quatro, *Othello* e *Scrable* tem tido jogadores artificiais muito fortes e pesquisas bem sucedidas em seu desenvolvimento [6]. Resultados interessantes, tais como o uso da árvore de procura *Monte Carlo* aplicado a jogos de *Go*, foram obtidos [9]. Recentemente, outros tipos de jogos mais elaborados como *Hex*, *Brigde*, *Poker* e jogos de estratégia em tempo real, RPGs (*Role Playing Games*), têm atraído o foco das pesquisas na área de inteligência computacional. Esse ambiente de jogos tem se provado muito proveitoso para o desenvolvimento de inteligências artificiais, pois as regras são fixas, o escopo do problema é restrito e a interação entre os jogadores é bem definida [7].

Poker é um ambiente com informação incompleta, em que o jogador tem que lidar com análise de risco, análise probabilística contra ações de múltiplos agentes [2], o que torna a tomada de decisões não trivial [5]. Dito isso, fica claro que é um rico ambiente para aplicação e desenvolvimento de inteligência artificial, e tem estimulado diversas pesquisas devido aos muitos fatores que existem em uma partida e à natureza estocástica das decisões que envolvem o jogo [6].

A modelagem das inteligências artificiais aplicadas no *Poker* tem abordado o problema de diversas formas, utilizando redes neurais para auxílio na tomada de decisões e na tentativa de prever a próxima jogada dos oponentes como visto em [1], através do conhecimento probabilístico e da análise da chance de vitória como visto em [12] e [13], ou então com aprendizado supervisionado analisando jogadas de profissionais, como visto em [3]. Uma outra maneira de se tratar uma das grandes variáveis do *Poker*, o tipo de oponente que se está enfrentando, é realizada através de algoritmos de agrupamento, como visto em [4], e com outras abordagens em [10] e [11].

Avaliar o desempenho de uma inteligência artificial pode ser muito útil para melhorar seus resultados, logo, o desenvolvimento de uma plataforma que possibilite não só o treinamento contra alguns tipos de inteligências pré-definidas como contra outras

personalizadas é de grande interesse para a área. Com um *software* capaz de rapidamente entregar resultados do desempenho de uma inteligência artificial em desenvolvimento, a análise de pequenas mudanças na lógica da inteligência ou de novas implementações seria mais rápida e eficaz, melhorando o processo de refinamento da inteligência. Dito isso, pode se criar uma ligação desse *software* com suas aplicações no ensino de engenharia, dado que, com sua disponibilização para os alunos, os mesmos poderiam verificar os resultados de seus pequenos avanços rapidamente e ter um retorno rápido do efeito que aquela mudança faz para o seu código.

Esse trabalho é organizado da seguinte forma: na Seção 2, é trabalhada a metodologia utilizada, subdividindo-se em explicações sobre o software base utilizado e sua comunicação com o Matlab. Na Seção 3, são mostrados os resultados, tais como o *software* final, a inteligência artificial desenvolvida para ser implementada no programa e a discussão sobre os resultados obtidos nas simulações realizadas. Por fim, na Seção 4, é feita a conclusão do trabalho.

1.1 Poker Texas Hold'em

Esta seção será dedicada para a elucidação das regras básicas do *poker* e suas terminologias, pois serão muito utilizadas ao longo desse trabalho.

Poker Texas Hold'em é um jogo de cartas em que se jogam pelo menos duas pessoas, comumente indo até oito ou dez em uma mesma mesa. O tipo de jogo que será tratado aqui é o chamado *Texas Hold'em.* Inicialmente todos os jogadores fazem um *buy in*, que seria o ato de comprar um numero fixo de fichas para jogar; o numero máximo de *buy ins* em uma partida deve ser pré-definido com antecedência. Então, inicia-se propriamente a partida e cada rodada tem o seguinte funcionamento: cada jogador da mesa recebe duas cartas que devem se manter ocultas ao longo do jogo. Um jogador é obrigado a colocar um tributo mínimo em mesa, chamado de *big blind*, e o jogador que vem antes dele é obrigado a colocar metade desse tributo, chamado de *small blind.* Então, começando pelo jogador seguinte ao que pôs o *big blind*, os participantes decidem se querem pagar a aposta mínima para continuar jogando, ação chamada de *call*, ou se preferem sair da rodada, ação chamada de *fold*, e, no caso de ninguém ter aumentado a aposta até chegar na vez de um determinado jogador, ele não é obrigado a apostar, e pode decidir por não fazê-lo, ação chamada de *check.*

Depois dessa primeira rodada de apostas são reveladas três cartas do deck para todos os jogadores, etapa chamada *flop*, e então mais uma rodada de apostas se inicia sem a

obrigação inicial do *big* e *small blind*, que já foram pagos na primeira rodada. Após o término dessa rodada de apostas, é revelada mais uma carta para todos os jogadores, etapa chamada *turn*, em que inicia-se mais uma rodada de apostas, como a que acontece no *flop*, e por fim uma quinta e última carta é revelada para todos, etapa chamada de *river*, e então inicia-se a última rodada de apostas e a última pessoa a dar *call* revela sua mão e com as sete cartas que tem disponível, as cinco da mesa combinadas com as duas de sua mão, faz o jogo mais forte com cinco cartas. Os outros jogadores que ainda não tenham saído da rodada têm a opção de mostrar ou não suas cartas e comparar se o jogo que eles conseguiram fazer tem ou não mais força que os demais jogadores da mesa. Quem tiver o maior jogo adiciona todas as fichas no *Pot*, termo usado para o montante total de fichas apostadas na rodada, e soma ao seu *stack*, termo utilizado para se referir ao número de fichas que o jogador tem disponível para apostar.

1.2 Objetivo Geral

O objetivo deste trabalho é desenvolver um ambiente de simulação de *poker*, o *AIPokerBattle*, em que uma ou mais inteligências artificiais joguem entre si de tal forma a criar condições favoráveis para o aperfeiçoamento de técnicas nessa área.

1.3 Objetivos Específicos

Esse trabalho tem como objetivos específicos os seguintes:

- Atualizar o *software open source* base.

- Documentar os dados importados e exportados durante o funcionamento do programa e as variáveis de entrada e saída das funções que chamam as inteligências artificiais personalizadas.

- Desenvolver uma inteligência artificial utilizando lógica nebulosa para a implementar no programa final.

- Realizar testes dessa IA contra as diferentes IAs disponíveis, avaliando seu desempenho e tempo de simulação.

2 Metodologia

Nessa seção será discutido e explicado o processo por trás do desenvolvimento do *AIPokerBattle*, inicialmente falando sobre o programa utilizado como base para a modificação, as limitações do mesmo, as mudanças feitas sobre ele e o caminho feito para efetivar a comunicação com o *Matlab*.

2.1 Software OO Poker

Para desenvolver um ambiente flexível de simulações de jogos de *poker* entre IAs, foi utilizado como base um programa *open source* escrito em *C++*, que foi utilizado como simulador em outras pesquisas, como visto em [1].

Dentre as funcionalidades desse programa, estão a possibilidade de jogar manualmente contra uma ou mais IAs, ou colocar uma ou mais IAs para jogarem, sendo possível definir parâmetros importantes como *stack* inicial e valores do *big blind* e *small blind* e quantos *buy ins* seriam permitidos. O *OO Poker* também vem munido de diferentes IAs para uso, como a *IA Raise*, que responde com um *raise* todas as ações dos oponentes; com um raciocínio análogo, há a possibilidade de selecionar a *IA Call*, *IA Check/Fold* e por fim a *IA Random*, que responde a todas as ações dos oponentes com uma ação aleatória, e a *IA Smart*, que toma decisões de maneira mais inteligente, sem ter um padrão de ação simplório pré-definido. As funcionalidades de cada uma das IAs do *OO Poker* se encontram na Tabela 1.

Tabela 1: IAs do *OO Poker*

AI Call	Sempre dá *Call* nas apostas dos oponentes
AI Raise	Sempre dá *Raise* nas apostas dos oponentes
AI Check/Fold	Sempre dá *Check*, se não puder da *Fold*

AI Random	Faz ações aleatórias
AI Smart	Utiliza de uma lógica mais sofisticada na sua tomada de decisões

2.2.1 Mudanças no *OO Poker*

Para se testar diferentes IAs personalizadas no *OO Poker*, seria necessário fazer algumas modificações no código do programa para identificar a IA desenvolvida a ser testada, e desenvolvê-la obrigatóriamente utilizando a linguagem C++, além do fator limitante de simular apenas uma partida por vez.

Iniciamente foi definido que seria mais interessante que fosse possível selecionar o número de partidas desejadas para o jogo. Modificou-se então o código original para poder se jogar mais de uma partida em sequência.

Dado que o ambiente de simulação do *AIPokerBattle* foi proposto para IAs apenas, removeu-se a possibilidade de jogar manualmente e separou-se o desenvolvimento em dois programas, um para colocar IAs desenvolvidas no *Matlab* para jogar entre si, variando de dois a oito jogadores, e outro para fazer o *benchmarking* de uma IA desenvolvida fora do *OO Poker*, podendo jogar contra um a sete oponentes iguais escolhidos dentre as IAs pré-programadas do *software* base.

2.3 Comunicação com *Matlab*

Fazer o *Matlab* executar o *OO Poker* em uma tentativa de fusão não seria um caminho bom, devido às diferenças em declaração de variáveis e sintaxe de expressões. Então, para comunicar os dois, foi necessária uma série de implementações.

Inicialmente, foi estudado como uma IA personalizada seria chamada durante uma simulação no *OO Poker*. Feito isso, modificou-se o código original para que o *software* exportasse as informações do jogo e esperasse a ação realizada pela inteligência artificial

personalizada que está no *Matlab*. Essa comunicação está esquematizada nos fluxogramas representados na Figura 1.

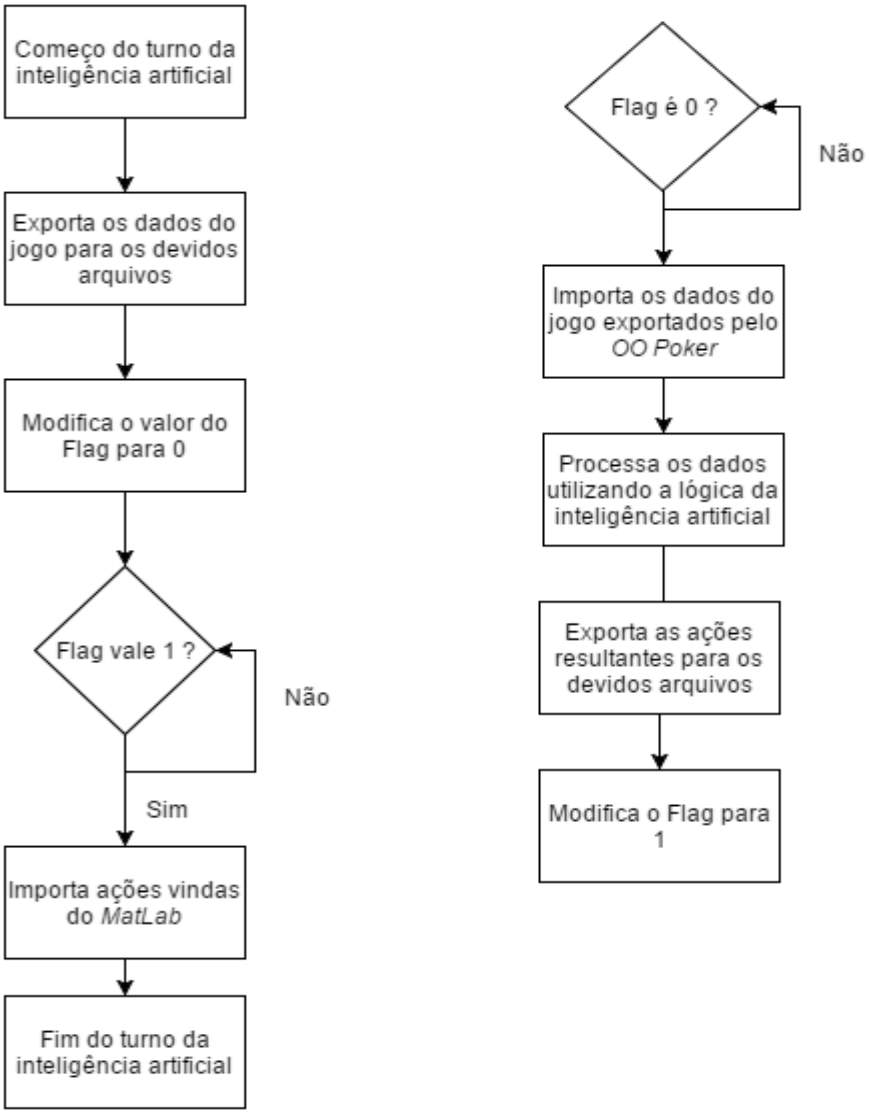

Figura 1: Fluxograma de funcionamento da comunicação, em que o esquerdo é o do *OO Poker* e o direito é o do *Matlab*.

A adição de IAs para as simulações feitos no *OO Poker* é feita através de edição do código fonte do mesmo, pois isso não seria interessante para o *AIPokerBattle*, então a próxima mudança feita foi, através de arquivos texto, que os dados iniciais do jogo fossem feitos através do *prompt* do *Matlab*, dessa forma poderia ser selecionado o valor dos *blinds* e *stack*, juntamente com quais e quantas IAs jogariam sem ter que modificar o código fonte.

Por fim, como o *OO Poker* não foi feito para se jogar várias partidas repetidas, e a intenção é que o *AIPokerBattle* funcionasse todo pelo *Matlab*, foi feito um último arquivo que, ao final da partida, seria exportado para um arquivo de texto o nome da IA vencedora, que a rotina no *Matlab* importasse para poder contabilizar as vitórias e oferecer a taxa de vitória no fim das partidas.

3 Resultados e Discussões

3.1 Inteligência Artificial Fuzzy

Devido à facilidade inicial, flexibilidade no desenvolvimento, e uma *toolbox* excepcionalmente boa no *Matlab*, escolheu-se o método de *fuzzy logic mandani* para o desenvolvimento dessa IA personalizada, com o intuito de testá-la contra as IAs pré-feitas do *OO Poker* e de entregar uma outra opção para os usuários do *AIPokerBattle.* É importante ressaltar que ela foi desenvolvida inicialmente para casos de *Heads Up*, ou seja, com apenas dois jogadores na mesa.

Como descrito anteriormente, *Poker* é um jogo com muitas possíveis variáveis de entrada para se analisar, então no desenvolvimento dessa IA, primeiramente, definiu-se quais parâmetros de entrada seriam relevantes. Através da consulta com um jogador experiente de *poker*, reduziu-se a IA em dois casos, um para análise pré-*flop* e outro para analisar as demais situações em jogo. Para o pré *flop*, leva-se em conta o *stack* do jogador, contado em número de *big blinds*, a força do par de cartas iniciais, o fato de estar ou não em posição de jogo, o que implica em ser ou não o último a apostar na rodada e por fim a quarta variável é o *PotOdds*, que é uma relação do quanto o jogador tem que apostar em relação ao valor do *Pot*.

Já a análise para as demais situações do jogo leva em conta a força do jogo que o jogador tem combinando as cartas da mão com as cartas da mesa, o *stack* contado em número de *big blinds*, se o jogador está ou não em boa posição de jogo, o *PotOdds* e duas variáveis extras, a periculosidade da mesa, que indica se dentre as cartas da mesa já tem alto grau de possibilidade de ser formado um jogo forte, e, por fim, os *outs* do jogador, que representam a chance de melhora dos atual jogo para o jogador.

As saídas possíveis desse sistema *fuzzy* são *check/fold* , *call*, *raise, raise* muito, *all in,* e as regras tentam fazer o jogo ser bastante abrangente, analisando mais se o jogo em mãos e o quanto deve-se pagar para participar da rodada valem a pena. Uma amostra das regras utilizadas nas tomadas de decisão do sistema *fuzzy* se encontram na Figura 2 .

1. If (ForcaJogo is ruim) and (Outs is not Ruim) and (Posicao is Nao) then (Saidas is Fold/check) (1)
2. If (ForcaJogo is ruim) then (Saidas is Fold/check) (1)
3. If (ForcaJogo is Medio) and (Outs is Ruim) and (Stack is Baixo) then (Saidas is Fold/check) (1)
4. If (ForcaJogo is Medio) and (Outs is Ruim) and (Stack is Medio) and (PotOdds is not ruim) then (Saidas is Call/check) (1)
5. If (ForcaJogo is Medio) and (Outs is Ruim) and (Stack is Medio) and (PotOdds is ruim) then (Saidas is Fold/check) (1)
6. If (ForcaJogo is Medio) and (Outs is Ruim) and (Posicao is Sim) and (Stack is Alto) and (PotOdds is not ruim) and (Perigo is not perigosa) then (Saidas is Call/check) (1)
7. If (ForcaJogo is Medio) and (Outs is Ruim) and (Posicao is Sim) and (Stack is Alto) and (PotOdds is not ruim) and (Perigo is perigosa) then (Saidas is Fold/check) (1)
8. If (ForcaJogo is Medio) and (Outs is Ruim) and (Posicao is Sim) and (Stack is Alto) and (PotOdds is ruim) and (Perigo is not perigosa) then (Saidas is Fold/check) (1)
9. If (ForcaJogo is Medio) and (Outs is Ruim) and (Posicao is Nao) and (Stack is Alto) and (PotOdds is not ruim) and (Perigo is not perigosa) then (Saidas is Call/check) (1)
10. If (ForcaJogo is Medio) and (Outs is Ruim) and (Posicao is Nao) and (Stack is Alto) and (PotOdds is ruim) and (Perigo is not perigosa) then (Saidas is Fold/check) (1)

Figura 2: Exemplo de regras da *AI Fuzzy*.

3.2 AIPokerBattle

O *AIPokerBattle* totalmente desenvolvido funciona em duas rotinas diferentes, uma voltada para testar a IA em desenvolvimento contra outras IAs pré-feitas do *OO Poker* e outra voltada para a simulação de uma partida entre várias IAs externas. Nas próximas seções, será descrito o funcionamento geral do *AIPokerBattle* e as diferenças de cada modo e como deve ser feito para que uma IA externa seja feita dentro dos padrões de importação de dados e exportação de ações.

3.2.1 Interface e saída do modo *Benchmarking*

Nessa seção será demonstrado o funcionamento desse módulo do *AIPokerBattle*. Ele está na forma de uma rotina feita para ser executada no *Matlab* versão 2016. No *prompt* será pedido para o usuário escolher as devidas configurações, seguindo como demonstrado no fluxograma da Figura 3 e exemplificado na Figura 4.

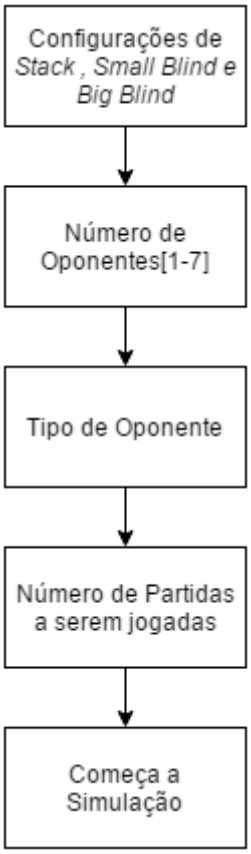

Figura 3: Fluxograma das configurações iniciais do modo *Benchmarking.*

```
Command Window
  Bem Vindo ao AIPokerBattle modo Benchmarking
  Decida entre uma das seguintes configurações de Stack, Small e Big Blind
  1: 1000, 5, 10
  2: 1000, 10, 20
  3: 1000, 50, 100
  4: 1000, 100, 200
  5: 100000, 5, 10
  6: 100000, 10, 20
  7: 100000, 50, 100
  8: 100000, 100, 200
  Gostaria de qual?[1-8] 4
  Gostaria de jogar contra quantas AIS ?[1-7] 5
  Selecione a AI que irá preencher os demais lugares da mesa contra a sua AI
  Aperte 1 para AI Smart
  Aperte 2 para AI Random
  Aperte 3 para AI Check/Fold
  Aperte 4 para AI Call
  Aperte 5 para AI Raise
   Deseja jogar contra qual?[1-5] 4
fx Quantas partidas gostaria de jogar seguidas? 10
```

Figura 4: Menu inicial do modo *Benchmarking* com as possíveis seleções de parâmetros para a simulação.

Nesse caso exemplificado, o usuário escolheu a quarta configuração de *stack*, *small* e *big blind*, jogar contra cinco IAs personalizadas, jogar contra a *AI Call* por dez partidas seguidas.

Após a confirmação dos parâmetros iniciais, a simulação inicia, sendo carregada uma janela do *prompt* de comando do *windows*. Ao fim dela, é emitido no *prompt* do *Matlab* o tempo que demorou a simulação a partir da definição incial, e a taxa de vitória da IA do usuário, e o número total de partidas jogadas, como mostrado na Figura 5.

Figura 5: Dados emitidos no final da simulação do modo *Benchmarking.*

3.2.2 Interface e saída do modo *AIBattle*

Similarmente ao módulo anterior, este está em forma de uma rotina no *Matlab*, para acessá-lo basta executar o arquivo *AIPokerBattleAIMode.m* e será pedido que o usuário insira os parâmetros inicias, seguindo como mostrado no fluxograma da Figura 6 e exemplificado na Figura 7.

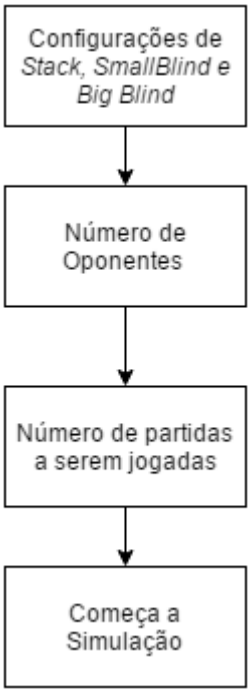

Figura 6: Fluxograma das configurações iniciais do modo *AIBattle.*

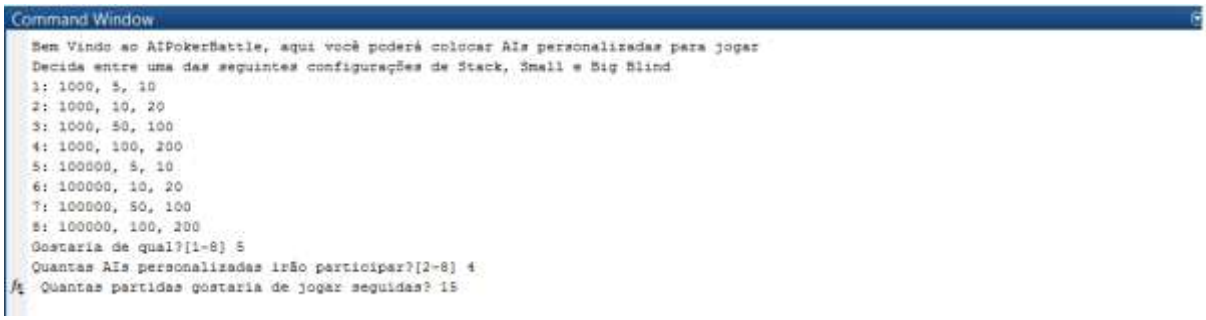

Figura 7: Menu inicial do modo *AIBattle* com as possíveis seleções de parâmetros para a simulação.

Nesse exemplo, o usuário selecionou a quinta opção de valores para *stack, small* e *big blind,* decidindo por quatro IAs personalizadas jogarem por quinze partidas seguidas. O decorrer da simulação é similar ao do módulo *benchmarking* com o importante fator que, em geral, haverá simulações mais demoradas, pois para realizar as ações de cada IA personalizada é necessário trabalhar com leitura e inscrição de dados em arquivos, o que aumenta consideravelmente o esforço computacional.

No término da simulação, é mostrada a porcentagem de vitória de cada jogador participante, o tempo total de simulação e o número total de partidas jogadas, como visto na Figura 8.

Figura 8: Dados emitidos no final da simulação do modo *AIBattle.*

3.2.3 Parâmetros exportados

Nessa seção serão detalhados todos os parâmetros exportados da rotina de simulação do *OO Poker* e em quais variáveis eles estão armazenados no *AIPokerBattle*.

Para dar condições de se criar uma IA para ser utilizada, é necessário que ela tenha acesso à informações vitais do jogo, para isso foram salvas em variáveis distintas as seguintes informações: Índice e Naipe das cartas da mão do jogador, Índice e Naipe das cartas da mesa, qual etapa do jogo está (Pré-*Flop, Flop, Turn* ou *River), Stack* atual do jogador, quantas fichas o jogador já apostou nessa rodada, valor necessário para dar *call* na sua vez, quantas fichas estão atualmente no *Pot*, valor do *big blind*, número de jogadores ainda dentro do jogo, número de jogadores que ainda estão na rodada, posição em relação ao *dealer*, posição do *dealer* e índice do jogador. Todas essas variáveis são bastante intuitivas, exceto as últimas três. O seguinte exemplo servirá para mostrar como elas se relacionam.

O índice do jogador é fixo na partida, a variável do índice do *dealer* mostra qual o índice do jogador que é o *dealer* na atual rodada, e a variável posição diz quão depois do *dealer* você está. Então, em uma situação com sete jogadores, e a IA em questão tem o índice zero e o *dealer* tem o índice um, certamente sua posição deve ser seis, pois os índices dos jogadores vão de zero à seis, totalizando sete jogadores o que significa que a IA em questão é o sexto jogador após o *dealer*, o que implica em ser o quarto jogador a apostar na rodada, pois depois do *dealer* vem o jogador que aposta o *small blind* e em seguida o jogador que aposta por último, o que começou com o *big blind*.

Na rotina tanto do modo *AIBattle*, quanto do modo *Benchmarking*, existe uma função chamada *InteligenciaArtificial.m*, que é o único lugar em que o usuário deve alterar o código do programa para poder inserir sua inteligência artificial. É importante ressaltar que essa função tem como entradas todos os dados do jogo descritos acima e tem como saída duas variáveis, uma que representa a ação que a IA irá tomar, podendo ser zero no caso de *check/fold*, um no caso de *call* e dois no caso de *raise*, e a segunda variável é o valor de aposta no caso de a ação ter sido *raise*, e deve ser preenchido com um valor qualquer nos outros dois casos. Dessa forma, a comunicação entre os dados do jogo e as ações da IA personalizada será feita sem problemas. A única diferença na implementação de uma IA personalizada nos dois modos é que no *Benchmarking* só existe e só se deve alterar uma função de inteligência artificial, já no modo *AIBattle*, foram implementadas outras sete funções que operam do mesmo modo descrito acima, porém com nomes de arquivo diferentes, indo de *InteligenciaArtificial2.m* até *InteligenciaArtificial8.m*, que podem ter lógicas diferentes para cada caso, desde que respeitando os padrões de entrrada e saída.

3.3 Resultados Experimentais

A fim de verificar o funcionamento do *AIPokerBattle*, comparar o desempenho da *AI Fuzzy* contra as IAs pré-feitas e ter uma noção do tempo necessário para simular várias partidas, foi feita uma série de simulações adotando a quarta opção de *stack, small* e *big blind*, tendo valores de 1000, 100 e 200, respectivamente, e foram jogadas três grupos de cinquenta partidas. As simulações foram feitas em um computador com 16 Gb de memória RAM e um processador i7-4810MQ com manipulação de arquivos feita em dísco rígido. Os resultados de média de taxa de vitória e de média tempo de execução estão mostrados em Tabela 2 e Tabela 3

Tabela 2: Porcentagem de vitórias da *AI fuzzy*

Numero / Oponente	*Raise*	*Call*	*Random*	*Smart*
7	9,33	24,66	25,33	21
6	14,66	28,33	21	23
5	16,66	32,66	22	25
4	17	38,33	24,66	36,66
3	15,33	38,33	26	42,66
2	22	46,66	32	51,33
1	27,33	48	52,66	73,33

Tabela 3: Tempo, em segundos, de simulação

Numero / Oponente	*Raise*	*Call*	*Random*	*Smart*
7	121,33	512,33	230,333	883,66
6	116,33	410	195,333	808
5	118,66	363	228,333	649,66
4	122,33	278,66	223,66	619
3	108	239,66	219,66	552,66
2	101	180,66	151,66	368
1	105,66	137,66	126,33	256

Analisando essas tabelas pode-se observar alguns pontos importantes. O primeiro e mais claro deles é que o desempenho da *AI Fuzzy* cai conforme se tem mais jogadores, afinal, seus parâmetros são voltados para situações *Heads Up*. Ainda sobre desempenho, pode-se concluir que a a *AI Fuzzy* irá funcionar melhor contra oponentes que, de fato, tenham uma razão por trás de seus atos, já que contra a *AI Smart* se teve resultados bastante promissores. A baixa taxa de vitórias contra a *AI Raise* em especial se dá ao fato de um dos parâmetros de entrada mais relevantes ser o *PotOdds,* e contra um jogador que sempre aumenta a aposta esse parâmetro sempre ficará com um indidicativo ruim, e portanto acontecem mais *folds.*

Deve ser ressaltado que os resultados obtidos em [1] apontam 78% de taxa de vitória no *Head's Up* contra a *AI Smart*, utilizando apenas uma sequência única de testes.

Os resultados obtidos nas simulações com mais oponentes estão com um valor aceitável, afinal, se cada jogador ganhasse o mesmo número de partidas, teríamos uma taxa de vitória de 12,5% contra 7 oponentes, por exemplo, e o resultado mostrou-se melhor que isso, exceto contra a *AI Raise*.

Em relação ao tempo de simulação, é de se esperar que quanto mais jogadores maior o tempo total, e também que ao aumentar-se a taxa de vitória, aumente-se o tempo de simulação

também, tendo em vista que para simular as jogadas das IAs do *OO Poker* não é necessário a troca de arquivos com o *Matlab,* tornando o correr do jogo substancialmente mais rápido, pois eveita troca de arquivos que podem tornar a simulação mais lenta. Porém pode ser observado que a relação de maior taxa de vitória com maior tempo de simulação, até mesmo para o mesmo número de oponentes, não é linear, afinal se for destacado os casos contra seis oponentes *AI Random* e *AI Raise* , podemos observar uma discrepância maior no tempo do que na taxa de vitória.

Por fim, vale comparar a disparidade entre a simulação mais rápida, com apenas 101 segundos, e a mais demorada, com 883 segundos, sendo um fator relevante a permanência da *AI Fuzzy* no jogo, tendo em vista que ,como antes mencionado,ao ser eliminada a comunicação entre e o *OO Poker* e o *Matlab* não é mais realizada, então não se perde tempo com manipulação de arquivos.

Com intuito de ter uma referência em relação ao tempo de simulação, foram feitos vários testes no modo *AIBattle*, colocando desde *AI Fuzzy* para jogarem até oito, para verificar como o tempo escala com maior número de IAs personalizadas, e os resultados podem se encontrar na Tabela 4.

Tabela 4: Tempo de simulação entre IAs personalizadas

8	3122,33
7	2673
6	2220,33
5	1612,66
4	1150,66
3	704,33
2	307

Fica claro que, ao aumentar o número de IAs personalizadas jogando, o tempo de simulação aumenta consideravelmente. Vale ressaltar que esse tempo deve variar para IAs

diferentes, devido ao tempo demorado para a realização da troca de informações entre a rotina no *Matlab* e o *OO Poker*, sendo o melhor dos casos demorando 307 segundos e o pior demorando 3122 segundos.

4 Conclusões

Nesse trabalho foram feitas as devidas mudanças no *OO Poker* para que viabilizasse adaptações que melhorassem o desenvolvimento de inteligências artificiais utilizando o *software* final, possibilitando a comunicação entre ele e o *Matlab*, dando escolhas de configurações para o jogo e também simulando várias partidas subsequentes.

Para que essa relação fosse possível, foram documentadas as variáveis em que os dados do jogo são armazenadas e como utilizá-las para adequar-se aos moldes de funcionamento do *AIPokerBattle*.

Desenvolveu-se uma inteligência artificial utilizando lógica nebulosa e foi implementada no programa final, onde a tomada de decisão é pautada na viabilidade de apostar dada a força do jogo, levando em conta especialmente o *PotOdds* da aposta.

Os resultados dos testes da *AI Fuzzy* contra números e oponentes variados foram positivos, sendo, em sua maioria, acima do desempenho dos oponentes e comparável a trabalhos já publicados no caso do *Head's Up* contra a melhor inteligência disponível no *OO Poker*. Os tempos de simulação variaram de 213 a 1755 no modo *benchmarking* e de 615 até 6325 no modo *AIBattle*.

Tais resultados mostram a viabilidade do uso do *AIPokerBattle* para fins de teste e treino de Ias, assim como seu potencial para a utilização no ensino de engenharia. Para trabalhos futuros, poderia se refinar o código para melhorar o tempo de simulação, exportar dados mais específicos das partidas, como o número de vezes em que o jogador *foldou* tendo uma mão melhor que dos oponentes, dando um *feedback* melhor do desempenho da IA.

Referências Bibliográficas

[1] Ziółko, Bartosz, Daniel Bochniak, and Grzegorz Jankowski. "Neural network application for automatic decisions in poker." *Journal of Applied Computer Science* 20.1 (2012): 119-127.

[2] Billings, Darse, et al. "The challenge of poker." *Artificial Intelligence* 134.1 (2002): 201-240.

[3] Teófilo, Luís Filipe, and Luís Paulo Reis. "Building a No Limit Texas Hold'em Poker Agent Based on Game Logs Using Supervised Learning." International Conference on Autonomous and Intelligent Systems. Springer Berlin Heidelberg, 2011.

[4] Teófilo, Luís Filipe, and Luís Paulo Reis. "Identifying Player\'s Strategies in No Limit Texas Hold\'em Poker through the Analysis of Individual Moves." *arXiv preprint arXiv:1301.5943* (2013).

[5] Southey, Finnegan, et al. "Bayes' bluff: Opponent modelling in poker." *arXiv preprint arXiv:1207.1411* (2012).

[6] Rubin, Jonathan, and Ian Watson. "Computer poker: A review." *Artificial Intelligence* 175.5 (2011): 958-987.

[7] Schaeffer, Jonathan. "A gamut of games." *AI Magazine* 22.3 (2001): 29.

[8] Fogel, David Bruce. "Evolving artificial intelligence." (1992).

[9] Gelly, Sylvain, and David Silver. "Monte-Carlo tree search and rapid action value estimation in computer Go." *Artificial Intelligence* 175.11 (2011): 1856-1875.

[10] Billings, Darse, et al. "Opponent modeling in poker." *AAAI/IAAI*. 1998.

[11] Davidson, Aaron. "Opponent modeling in poker: Learning and acting in a hostile and uncertain environment." (2002).

[12] Billings, Darse, et al. "Using probabilistic knowledge and simulation to play poker." *AAAI/IAAI*. 1999.

[13] Teófilo, Luís Filipe. "Estimating the Probability of Winning for Texas Hold'em Poker Agents." *Proceedings 6th Doctoral Symposium on Informatics Engineering*. 2011.