Simulação do Jogo Tic-Tac-Toe com o uso de Técnicas de Inteligência Artificial

Aluísio de Ávila, Angélica Caetane Pelizza, Bruno Barbosa Prettz, Gabriel Camara Mandeli, Renan Rosauro Rigon, Fábio José Parreira, Sidnei Renato Silveira

Universidade Federal de Santa Maria (UFSM) - Campus Frederico Westphalen - RS
Grupo de Pesquisa IATE/UFSM – Inteligência Artificial e Tecnologia Educacional

aluisiodeavila@hotmail.com, angelicapelizza@hotmail.com,
bruno.prettz@hotmail.com, gabriel.mandeli@hotmail.com,
reerigon@gmail.com, fabiojparreira@gmail.com,
sidneirenato.silveira@gmail.com

Resumo. *Este artigo apresenta uma aplicação desenvolvida em linguagem de programação Java, utilizando técnicas de Inteligência Artificial, que simula um jogador virtual no jogo Tic-Tac-Toe (TTT), mais conhecido como Jogo da Velha. Para a implementação do jogador virtual foi utilizado o algoritmo minimax e a busca em profundidade.*

Palavras-Chave: *Tic-Tac-Toe, Algoritmo Minimax, Busca em Profundidade*

Abstract. *This paper presents an application developed in Java programming language, using Artificial Intelligence techniques, to simulated a virtual player in the Tic-Tac-Toe (TTT) game. The virtual player implementation used the minimax algorithm and depth-first search.*

Keywords: *Tic-Tac-Toe, Minimax Algorithm, Depth-First Search*

1 Introdução

O Tic-Tac-Toe (TTT) é nacionalmente conhecido como o 'Jogo da Velha' e tem por objetivo formar combinações triplas na posição horizontal, diagonal ou vertical, sendo jogador por dois jogadores. O objetivo deste jogo é "ganhar para x", ou seja, quando "x" tem uma das oito possíveis maneiras de criar três possibilidades em uma fila, caso se encaixe (WYDYANTORO; VEMBRINA, 2009). Neste trabalho incluiu-se o conceito de Inteligência Artificial (IA), que é um ramo da ciência da computação que tem por objetivo elaborar dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas, por meio da implementação de um jogador virtual, utilizando técnicas de IA, mais especificamente árvores de decisão e teoria dos jogos.

Neste contexto, o presente trabalho apresenta uma aplicação utilizando-se a linguagem de programação Java, empregando o método MinMax (ou MiniMax), para criar um jogador virtual, com habilidades e inteligência similar a de um indivíduo comum, que irá jogar contra um jogador humano.

2 Fundamentação Teórica

Esta seção apresenta um breve referencial teórico sobre as áreas envolvidas no desenvolvimento da aplicação proposta.

2.1 Tic-Tac-Toe

O *Tic-Tac-Toe*, ou Jogo da Velha, possui um tabuleiro que corresponde a uma matriz de 3 (três) linhas por 3 (três) colunas. Dois jogadores escolhem uma marcação cada, geralmente um círculo (O) ou um xis (X). Os jogadores jogam alternadamente, uma marcação por vez, em uma lacuna que esteja vazia. O objetivo é conseguir três círculos ou três "xis" em linha horizontal, vertical ou diagonal. Ao mesmo tempo, quando possível, impedir o adversário de ganhar na próxima jogada. Quando um jogador conquista o objetivo (preencher as três lacunas predeterminadas) ele vencerá o jogo. Se os dois jogadores jogarem sempre da mesma forma, o jogo terminará sempre em empate (ALVES, 2015).

A lógica do jogo é muito simples, de modo que não é difícil deduzir ou decorar todas as possibilidades para efetuar a melhor jogada. Por esse motivo, é muito comum que o jogo termine em empate (velha). O jogo possui diversas possibilidades de resoluções para vencer o jogo, ou impedir que seu adversário vença. Assim, os jogadores podem escolher a melhor estratégia de jogo já que não há um único caminho para ser seguido.

Desta forma, os critérios de avaliação final de uma partida para o jogador humano são os seguintes: vitória (o jogador humano consegue colocar as três marcas em uma das configurações válidas do jogo), empate (o tabuleiro é completamente preenchido e nenhum dos jogadores consegue colocar as três marcas em uma das configurações válidas) e derrota (o oponente artificial consegue colocar as três marcas em uma das configurações válidas) (RADTKE, 2006).

2.2 Método MinMax

Em teoria da decisão, o *minimax* (ou *minmax*) é um método para minimizar a perda máxima possível. Pode ser considerado como a maximização do ganho mínimo (*maximin*). Começa-se com dois jogadores 0-0 da teoria dos jogos, cobrindo ambos os casos em que os jogadores tomam caminhos alternados (por rodadas) ou simultaneamente. Pode-se estender o conceito para jogos mais complexos e para tomada de decisão na presença de incertezas. Nesse caso não existe outro jogador, as consequências das decisões dependem de fatores desconhecidos (DIGIAMPIETRI, 2016).

No TTT, o algoritmo *minimax* ajuda a encontrar a melhor jogada ao caminhar pelas opções válidas a partir do fim do jogo. A cada passo assume-se que o jogador A está tentando maximizar as chances dele ganhar, enquanto na próxima rodada o jogador B está tentando minimizar as chances de isso acontecer. O jogador MAX sempre considera que MIN vai escolher a jogada que o deixa na pior situação (pior caso para MAX) e que ele (o MAX) vai escolher a melhor jogada para si. O algoritmo *minimax* ajuda a encontrar a melhor jogada ao caminhar pelas opções válidas a partir do fim do jogo.

O algoritmo *minimax* pode ser aplicado em jogos, a fim de determinar qual é a melhor jogada, sendo implementado por meio de uma árvore de decisão, contendo todas as jogadas possíveis. O algoritmo considera que existe um adversário que também é

uma entidade inteligente e racional. A minimização do valor de ganho nas jogadas do oponente significa a consideração de que o adversário também irá jogar de forma ótima.

2.3 Método de Busca em Profundidade

Um algoritmo de busca (ou de varredura) é um algoritmo que visita todos os vértices e todos os arcos de um grafo, andando pelos arcos de um vértice a outro. Na teoria dos grafos, busca em profundidade é um algoritmo usado para realizar uma busca ou travessia em uma árvore ou grafo. Intuitivamente, o algoritmo começa em um nó raiz e explora tanto quanto possível cada um dos seus ramos, antes de retroceder. As principais características do método são que ele segue cada caminho até sua maior profundidade antes de seguir para o próximo caminho. Caso a folha não representar um estado objetivo, a busca retrocederá ao primeiro nó anterior que tenha um caminho não explorado, utiliza um método chamado de retrocesso cronológico (volta na árvore de busca) uma vez que um caminho sem saída seja encontrado, e é assim chamado por desfazer escolhas na ordem contrária ao momento em que foram tomadas. É um método de busca exaustiva ou de força bruta (FEOFILOFF, 2016). A Figura 1 representa o funcionamento do método de busca com a seguinte ordem de visita aos nodos à árvore: A, B, D, H, I, E, C, F, G.

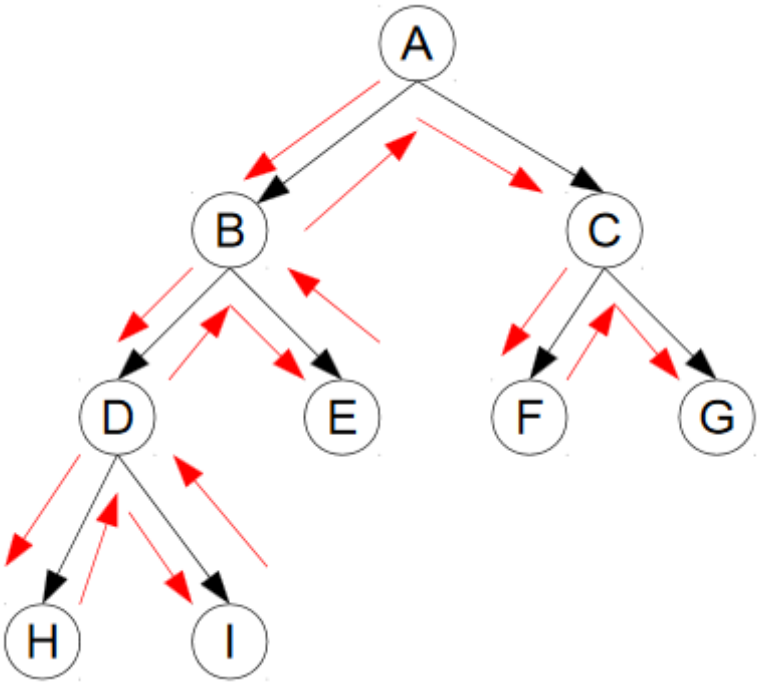

Figura 1: Busca em profundidade

3 Estado da arte

Dentre os diversos estudos existentes focados no desenvolvimento de estratégias para vencer um jogo, apresentam-se alguns estudos de casos que envolvem a aprendizagem de máquina a partir de jogos de tabuleiro, como o TTT, que apresentam as mesmas habilidades humanas (WIDYANTORO; VEMBRINA, 2009), visto que um jogador no TTT é tido como um tomador de decisões e adota estratégias que conduz às decisões

para atingir seus objetivos de vencer um jogo. Stange (2011) apresenta um sistema adaptativo em aprendizagem de máquina, onde se extraem do jogo os problemas reais, permitindo a fácil associação entre a utilização das técnicas adaptativas e as técnicas de aprendizado de máquina. Utiliza-se um autômato adaptativo para representar os conjuntos de regras do jogo, que são aprendidas a partir das partidas disputadas. A finalidade desse objeto é a de associar as saídas com as transições determinando um sucesso ou insucesso nas partidas em que são mapeadas as saídas geradas pelo autômato.

Sant´Ana (2014) explica que o uso de aprendizado por reforço para programar a Inteligência Artificial (IA) de jogos tem como destaque a criação nos anos 90 de um jogador virtual de gamão em nível profissional, por Gary Tesauro da IBM, com o objetivo de utilizar o aprendizado por reforço e fazer duas versões do programa jogarem entre si. No fim das partidas atribuía-se uma pontuação positiva para o vencedor e uma negativa para o perdedor. Depois de 200.000 partidas, o programa conseguia jogar como os melhores jogadores do mundo, sem precisar da experiência de jogadores profissionais ou descrever todos os mais de 1 trilhão de estados possíveis em uma partida.

Nesse contexto insere-se o TTT, que é um jogo mais simples. Analisando algumas partidas de TTT, perceberam-se algumas de suas propriedades, tais como: um jogador pode vencer, perder ou empatar; jogos entre dois jogadores que conhecem as melhores jogadas sempre terminam empatados; muitos dos estados são equivalentes - ao se girar o tabuleiro em 90^o ou espelhá-lo, obtêm-se até oito estados equivalentes. Exemplos de estados equivalentes: sem a equivalência, há nove quadrados que podem ser preenchidos de três modos diferentes (X, O ou vazio), considerando apenas os válidos e a equivalência entre estados, são necessários 765 possíveis estados para cobrir o jogo todo.

Diante disso, foi feito com que duas versões de agentes (jogadores virtuais) para o TTT jogassem entre si de maneira aleatória. A pontuação varia de acordo com o resultado da partida, uma vitória é mais importante que um empate ou uma derrota e por sua vez um empate é melhor que uma derrota. Também foi levado em consideração o número de jogadas necessárias para se vencer a partida, vencer na terceira rodada é mais importante do que vencer apenas na quinta. A pontuação dada a cada estado após a partida é uma função do Resultado (R), de quantas rodadas para o fim o estado ocorreu (i) e quantas rodadas o jogador teve durante o jogo (d). Após testar algumas combinações de cada variável, a função que apresentou resultados mais satisfatórios é apresentada na Figura 2, onde (R) assume valor 7 para vitória, 2 para empate e -2 para derrota; (i) varia de 0 e 4 e d entre 3 e 5, utilizando-se um espectro de 100.000 jogos (SANT´ANA, 2014).

$$score = R \times 2^{(5-i)} \times 3^{(5-d)}$$

Figura 2: Função de Resultado

Depois da fase de treinamento e com os testes feitos, o agente tem capacidade de determinar qual melhor jogada a seguir para pelo menos não perder. Confirmando o funcionamento do programa, duas versões foram colocadas uma contra a outra, uma jogando aleatoriamente e outra com os conhecimentos adquiridos. Em 50 mil jogos, o

agente inteligente venceu aproximadamente 91% dos jogos e empatou 9%, sem perder nenhuma vez.

4 Aplicação Desenvolvida

O *Tic-Tac-Toe* aqui apresentado é uma implementação do Jogo da Velha desenvolvida na linguagem de programação Java, utilizando o algoritmo *minimax*. A partir do estado inicial MAX tem nove movimentos possíveis. O jogo se alterna entre a colocação de um X por MAX e de um 0 por MIN até que se alcance o estado final, onde um jogador obtém três símbolos em uma coluna, linha ou diagonal; ou até que todos os quadrados estejam preenchidos. A Figura 3 apresenta o diagrama de funcionamento do TTT.

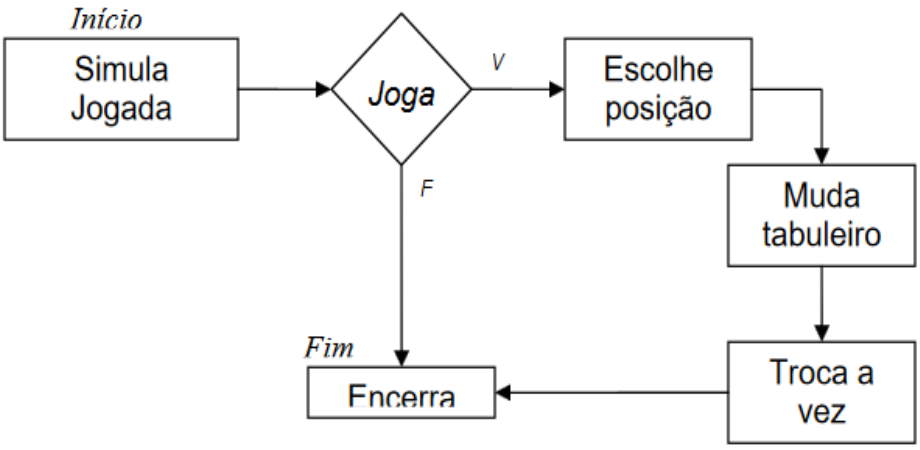

Figura 3: Diagrama de Funcionamento do TTT (RADTKE, 2006)

A implementação do algoritmo é composta por: 1) Velha.java - Classe principal da Aplicação; 2) Minimax.java - Classe responsável por aplicar o algoritmo MiniMax; 3) Tabuleiro.java - Classe responsável pela manipulação do tabuleiro do jogo. A Figura 4 apresenta a interface inicial do TTT.

Figura 4: Interface Inicial

Na Figura 5 tem-se a atuação do jogador que decidiu jogar na Coluna 1, Linha 3.

Figura 5: Primeira jogada

A primeira jogada do jogador virtual foi marcar com um X na Coluna 2, Linha 2. Posteriormente foram sendo escolhidas as opções restantes até todas as opções serem preenchidas. De acordo com a ordem das casas preenchidas, não se obteve, neste caso, um vencedor, como mostra a Figura 6.

Figura 6: Final do Jogo – Empate

Para decidir onde jogar, o jogador virtual executa o algoritmo *minimax* e escolheu uma posição que lhe favorecia, ao mesmo tempo em que prejudicava o seu adversário. Além disso utilizou-se a busca em profundidade, onde o algoritmo percorre as nove profundidades disponíveis e escolhe a melhor opção. A Figura 7 mostra um exemplo onde o computador foi o vencedor da partida.

Figura 7: Computador como ganhador

A validação da aplicação foi realizada por meio de 25 testes, como mostram os resultados da Tabela 1. As 1ª, 3ª e 5ª rodadas foram iniciadas pelo jogador (pessoa) e a 2ª e 4ª rodadas foram iniciadas pelo jogador virtual (computador). Baseando-se nos testes efetuados, concluiu-se que em 90% das vezes que o computador começou a partida, ele ganhou e, em somente 10%, o jogador (pessoa) conseguiu empatar com o computador. Em nenhuma das 25 partidas disputadas o jogador (pessoa) conseguiu ganhar do jogador virtual (computador).

Tabela 1: Tabela de Resultados de Iterações com o Jogo

Jogador	1ª Rodada	2ª Rodada	3ª Rodada	4ª Rodada	5ª Rodada
Aluisio	Empate	Derrota	Empate	Derrota	Empate
Angélica	Empate	Derrota	Derrota	Derrota	Derrota
Bruno	Empate	Derrota	Empate	Derrota	Derrota
Gabriel	Empate	Derrota	Empate	Derrota	Derrota
Renan	Empate	Empate	Empate	Derrota	Derrota

5 Considerações Finais

No decorrer dos testes da aplicação desenvolvida para o jogo TTT, considerando-se 25 partidas demonstradas na Tabela 1, o jogador virtual obteve 56% de vitórias. Aém disso, obteve uma porcentagem de 90% de vitórias quando iniciou a rodada (2ª e 4ª rodadas). O algoritmo busca 9 (nove) possibilidades disponíveis para cada jogada, optando sempre pela melhor opção. Manteve-se o foco principal nos testes finais, a fim de verificar a credibilidade do algoritmo no maior número de possibilidades. Os resultados demonstram que o jogador virtual efetivamente simula o raciocínio de um jogador humano no Jogo da Velha.

Os resultados apresentados por Sant´ana (2014), em 50.000 jogos, indicaram que o jogador virtual implementado venceu cerca de 91% das vezes e empatou 9% das vezes, sem perder nenhuma vez, resultados semelhantes ao trabalho aqui apresentado.

Referências

ALVES, W. (2015) *Jogos de Tabuleiro.* Disponível em: <http://welmajogosdetabuleiro.blogspot.com.br/2015_03_01_archive.html>. Acesso em setembro de 2016.

DIGIAMPIETRI, L. A. (2016) *Ensinando Técnicas de Inteligência Artificial utilizando o Jogo da Velha.* Disponível em: <http://www.uspleste.usp.br/digiampietri/jogos/palestras/TecnicasDeIA_JogoDaVelhaII.pdf>. Acesso em setembro de 2016.

FEOFILOFF, P. (2016). *Busca em Profundidade (DFS).* Disponível em: <http://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dfs1.html>. Acesso em setembro de 2016.

RADTKE, P. V. W. (2006) *Projeto Jogo da Velha.* Disponível em: <http://www.ppgia.pucpr.br/~radtke/jogos/velha/projeto-jogo_da_velha.pdf>. Acesso em setembro de 2016.

SANT´ANA, P. (2014). *Inteligência Artificial para o Jogo da Velha.* Disponível em: <https://infosimples.com/artigos/inteligencia-artificial-jogo-da-velha>. Acesso sem setembro de2016.

STANGE, L. R. (2011) *Adaptatividade em Aprendizagem de Máquina:* Conceitos e Estudo de Caso. In: Escola Politecnica da Universidade de São Paulo. São Paulo.

WIDYANTORO, D. H.; VEMBRINA, Y. G. (2009). *Learning To Play Tic-Tac-Toe.* In: International Conference on Eletrical Engineering an Informatics, 2009.