



UNIVERSIDADE DE BRASÍLIA

ESTRUTURA DE DADOS - TURMA B - TRABALHO II

Árvores de Jogos

GameTrees

CIC - Departamento de Ciência da Computação

Autores:

Tiago L. P. de Pádua - 12/1042457

Ronaldo S. Ferreira Jr. - 09/48721

Alex Leite - 05/97694

Professor:

Eduardo A. P. Alchieri

Fevereiro de 2013

1 Introdução

Através do estudo de estrutura de dados, este trabalho tem o objetivo a confecção de um projeto, onde deverá ser feito um *software* que utilize uma árvore genérica, chamada de *gametree*, para implementar a inteligência artificial de um jogo de tabuleiro utilizando o algoritmo conhecido como *Minimax*.

Neste projeto foi utilizado o jogo de tabuleiro Mancala [1,2], onde o objetivo geral do jogo é mover o maior número de peças para a "vala" do jogador, que é a cavidade mais a direita do tabuleiro:

- No início do jogo, cada "vala"(casa) é preenchida com 4 sementes. (Dependendo do estilo, podem ser colocadas 3, 5 ou 6 sementes);
- Cada jogador controla uma fileira de 6 casas. As sementes movem-se no sentido anti-horário;
- Na vez de cada jogador, escolhe-se uma casa, com sementes e, movem-se todas as sementes no sentido anti-horário;
- As sementes devem percorrer todas as casas, até atingir a casa maior do jogador. Neste caso, a semente é capturada;
- Caso a última semente da casa menor caia na casa maior. O jogador ganha uma vez extra;
- Caso a última semente caia numa casa vazia e, a casa do oponente contiver sementes, todas as sementes da casa que estava vazia e da casa do oponente são capturadas. (Vão para a casa maior do jogador);
- O jogo acaba quando não existem mais nenhuma semente na casa do dado jogador.

Demais regras do jogo podem ser obtidas em [1,2].

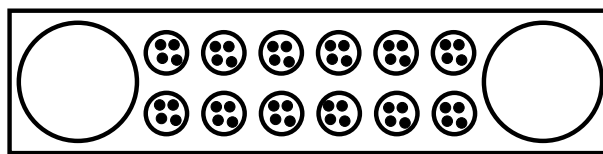


Figura 1: Tabuleiro Mancala.

O algoritmo *Minimax* [3] é utilizado para indicar qual a melhor jogada a ser realizada a partir da disposição atual do tabuleiro utilizando uma árvore de jogadas possíveis. A idéia central do algoritmo é que os jogadores sempre escolherão a jogada que lhe traga maior benefício em detrimento do oponente.

A árvore gerada funciona como um preditor de movimentos e é de grande valia para implementação de processos de inteligência artificial, principalmente em jogos, porém pode ser adaptada para outras aplicações que necessite de predição de resultados.

2 Implementação

A implementação do código foi realizada utilizando a linguagem de programação Java (javac 1.7.0_13) através da plataforma de desenvolvimento Netbeans 7.2.1.

As seguintes classes foram criadas:

- Arvore.java
- ListaLigada.java
- MancalaConsole.java
- No.java
- Elemento.java
- Mancala.java
- MancalaException.java

2.1 MancalaConsole.java

Classe que possui a função *main* do programa, ou seja, a primeira função a ser chamada quando o programa é executado, sua função principal é fornecer uma interface via linha de comando pela qual o usuário irá interagir com o sistema;

2.2 MancalaException.java

Utilizada para indicar que ocorreu um erro dentro da execução do programa.

2.3 Arvore.java

Classe que implementa as funções básicas de uma árvore, possui somente o campo *raiz*, do tipo *No* e, uma função que retorna um booleano, caso a árvore esteja vazia.

2.4 No.java

Representa um nó de uma árvore, possui um valor (utilizado pela função *Minimax*), um booleano *alpha*, que indica de qual jogador é a "vez" de jogar, um objeto, do tipo *No*, chamado de "pai", que é uma referência para o nó "pai".

O nó "pai" é necessário, uma vez que o algoritmo *Minimax* é executado no sentido das folhas para a raiz da árvore. Além disso, possui também uma lista ligada, que representa os "filhos" deste nó e, um número inteiro que representa qual casa foi movida na mancala para que ela chegasse no estado atual.

2.5 ListaLigada.java

Implementa uma lista encadeada que possui uma referência para o primeiro elemento da lista, além das funções utilitárias de inserção na lista e verificação de lista vazia.

2.6 Elemento.java

Classe utilizada como unidade básica da lista ligada.

2.7 Mancala.java

Representa o tabuleiro do jogo, contém a "inteligência" necessária para mover as peças de uma casa e redistribuí-las, corretamente, conforme as regras do jogo e, após a movimentação das peças, indica de quem é a "vez de jogar".

Indica, também, se o jogo terminou.

Nesta classe está implementado o algoritmo *Minimax*, ressaltamos que a questão mais importante do algoritmo é a função de avaliação da Mancala. Em nosso projeto, definimos que a avaliação da Mancala é realizada pela diferença entre a quantidade de peças dos jogadores em suas "valas", assim, o computador irá escolher jogadas que maximizem o número de peças em sua "vala" e minimize do oponente.

2.8 Execução do programa

Ao ser executado, o programa, inicialmente, solicita que o usuário informe qual o número de níveis que devem ser utilizados na criação da árvore de jogadas. Em nossos testes, valores acima de 8 tornaram o sistema instável devido à utilização de toda a memória do computador.

É então instanciado um novo objeto, do tipo Mancala, que representa o tabuleiro de jogadas, iniciando-se, então, um *loop*, que é executado enquanto o jogo não terminar (esta informação é obtida diretamente do objeto Mancala).

O próximo passo do programa é identificar de quem é a "vez de jogar", caso seja do jogador humano, é solicitado que este informe a casa a ser movida, caso seja a vez do computador, são seguidos os seguintes passos:

1. Cria-se uma árvore de jogadas a partir do estado atual da Mancala com o número de níveis informados pelo usuário;
2. Aplica-se o algoritmo *Minimax* à árvore gerada;
3. Identifica-se na árvore resultante qual deve ser o movimento adotado pelo computador;
4. O computador move a casa indicada pelo algoritmo;
5. São impressos em tela o tempo gasto de processamento e o número de jogadas analisadas;

2.9 Testes

Foram criadas as classes de teste "ArvoreTeste.java" e "MinimaxTeste.java", nestes testes foi utilizado como referência o exemplo da aplicação do algoritmo *Minimax*, a árvore da Figura 2 foi comparada a árvore resultante e, identificamos que estas eram idênticas.

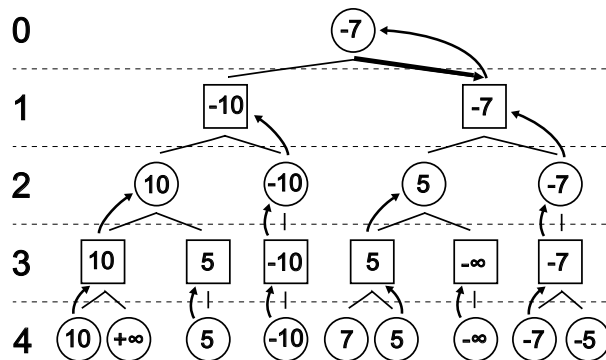


Figura 2: Árvore *Minimax* [3].

3 Conclusão

Pode-se concluir que a utilização de árvores de jogadas em conjunto com o algoritmo *Minimax* é uma boa alternativa para se implementar inteligência artificial em jogos de tabuleiro, no entanto, o fator limitante é o número de jogadas disponíveis para os jogadores, uma vez que em jogos mais complexos como o Xadrez, a árvore de jogadas se tornará muito grande em um pequeno número de níveis, devendo assim ser utilizado um algoritmo de "poda" para economizar a memória do sistema, já a memória é o fator mais crítico na utilização das *gametrees*. Alternativamente, quando adaptadas, estas árvores podem ser utilizadas em outras aplicações de predição e inteligência artificial.

Referências

- [1] Diversos. Mancala. <http://pt.wikipedia.org/wiki/Mancala>, 2013. [acessado em 19 de Fevereiro de 2013].
- [2] Diversos. Mancala. <http://en.wikipedia.org/wiki/Kalah>, 2013. [acessado em 19 de Fevereiro de 2013].
- [3] Diversos. Minimax. <http://en.wikipedia.org/wiki/Minimax>, 2013. [acessado em 19 de Fevereiro de 2013].
- [4] Roberto Tamassia Michael T. Goodrich. *Data Structures and Algorithms in Java*. 2005.