Estruturas de dados - Trabalho 2
Inteligência Artificial Para O Jogo Mancala Com Game Trees

Alberto Tavares Duarte Neto, 18/0011707

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)

albertotdneto@hotmail.com

1. Introdução

O problema a ser resolvido é implementar uma inteligência artificial para o jogo mancala. Uma forma de fazer isso é através de árvores de decisão. Como cada jogada tem apenas 6 possibilidades (um número baixo quando comparado com outros jogos, como o xadrez ou o go), esse método é viável de executar.

Para a implementação da IA do jogo, será primeiro necessário implementar o jogo, que funcionará com um simples vetor como tabuleiro e algumas funções que o modificam.

Para a IA, será utilizado o algoritmo Minimax para decidir a melhor jogada baseado no valor de retorno da função de avaliação.

Nas regras do jogo foi usada a mecânica de captura: caso a última pedra caia em uma casa vazia, a casa do lado oposto (inimigo) assim como a que acabou de cair, são capturadas e vão para a minha kahala.

Foi decidido não utilizar a regra que permite jogar outra vez caso a última pedra caia na kahala.

2. Implementação

As estruturas de dados utilizadas foram o vetor e a árvore 6-ária.

O jogo em si funciona assim que a função iniciarJogo é chamada. Ela, por sua vez, decide a vez de quem joga e chama as funções corretas. A função input é para jogada do Jogador e a função jogarIA é para a jogada da IA.

Para a avaliação dos estados do jogo para o algorítmo Minimax, temos as funções avaliarFinal e avaliarTabuleiro.

2.1. Vetor

O vetor para o tabuleiro foi utilizado principalmente por sua simplicidade e economia de memória, tendo em vista que a árvore pode crescer muito rápido e, para cada nó, precisaremos de um vetor associado.

Como é um vetor de int (4 bytes) com tamanho 14, cada vetor tabuleiro terá 56 bytes de tamanho.

2.2. Árvore 6-ária

Esta árvore, utilizada para representar cada estado de jogo e suas possíveis jogadas, é implementada da seguinte forma:

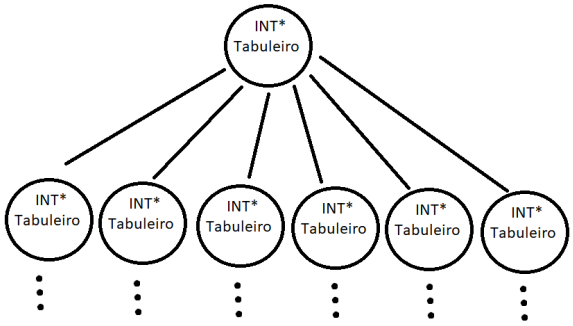

Figure 1. Diagrama da pilha de char

Ou seja, cada nó da árvore possui um ponteiro para um vetor de int (tabuleiro) e, no máximo, 6 filhos.

Isto significa que cada nível da árvore terá, no máximo, 6^{nivel} nós. Como cada nó tem um vetor associado de 56 bytes, no nível 9 (último nível de uma árvore de altura 10), teremos 564 350 976 bytes apenas de informação do vetor.

Foram utilizadas as funções:

criarNo, que recebe um ponteiro de int e retorna um ponteiro de nó com seu ponteiro de int com o valor que recebeu.

freeArvore, que libera da memória a árvore dada, assim como os vetores de cada nó.

2.3. Função iniciarJogo

Esta é a função que contêm o loop principal do jogo. Nele, caso a vez seja do jogador, é chamada a função de input, que recebe a jogada do teclado e retorna. Caso seja a vez da IA, é chamada a função jogarIA, que chama a função decidir e, após criar a árvore e aplicar o algorítmo Minimax, retorna a melhor jogada interpretada por este.

Após receber a jogada, ela é passada para a função jogar, que a aplica de fato caso seja uma jogada válida.

Ao final, é chamada a função fimJogo que verifica se algum dos lados não possui mais movimentos válidos. Caso seja verdade, o jogo é terminado, as pedras restantes vão para sua kahala e o vencedor é anunciado.

2.4. Função jogar

Recebe o tabuleiro, a posição da jogada, e a vez de quem está jogando.

O funcionamento desta é simples: dado uma posição de uma casa, caso esta seja válida, as n pedras da dela serão distribuidas para as n casas seguintes, 1 pedra pra cada. Note que a pedra não é entregue à Kahala do oponente, ela é apenas pulada.

Caso a pedra caia numa casa do jogador que fez o movimento e esta casa possua 0 pedras, a casa (do oponente) do lado oposto é capturada (suas pedras, assim como a 1 pedra que caiu na casa vazia, vão para a kahala de quem fez o movimento). A função jogar verifica as condições de captura e chama a função capturar para aplicar a jogada.

2.5. Função jogarIA

Sendo a função que retorna a jogada da IA, recebe o tabuleiro e, a partir dele, cria o nó raiz da árvore de decisão. O algoritmo Minimax é efetuado com a função decidir e esta.

Após criar a raiz da árvore de decisão, são criados os filhos desta a partir das 6 jogadas possíveis do estado (caso uma jogada não seja possível por não ter pedras na casa, o filho é nulo), e pra cada jogada possível é chamada a função decidir.

Isso tudo gerará até 6 números, 1 pra cada possível jogada, sendo que quanto maior o número, maior são as chances da IA vencer. Assim, a jogada com maior chance é retornada.

2.6. Função decidir

É semelhante a função jogarIA. Seu funcionamento é da seguinte forma:

Dada uma árvore, são criados 6 nós, cada um representando uma possível jogada (representação feita pelo vetor do nó), e cada um desses nós vira um filho da árvore. Após isso, é chamada a própria função decidir em cada um desses filhos.

A função só para quando o tabuleiro da árvore é um fim de jogo ou quando a altura da árvore chega em um determinado limite. Nestes casos, é chamada uma das funções de avaliação e, caso a árvore represente uma jogada da IA, é retornado o movimento que dá maior vantagem à IA. Caso represente uma jogada do jogador, é retornado o que dá maior vantagem ao jogador. Assim, é implementado o algoritmo Minimax.

Foi decidido separar a função de decisão da IA em duas funções pois decidir não retorna o índice da jogada, informação essencial apenas para o primeiro movimento pois é ele que precisamos retornar.

3. Conclusão

O jogo e sua IA foram implementadas com sucesso. Foi adquirida ao longo do trabalho uma noção da utilidade das árvores na resolução de problemas, assim como suas desvantagens. Como o crescimento do número de nós da árvore é exponencial, deve-se ter muito cuidado no gerenciamento de memória, tanto para evitar a falta dela como o vazamento.

Uma dificuldade encontrada foi a reutilização de parte da árvore para as próximas decisões, tal que isto não foi implementado na versão final do trabalho.