SegundoTrabalho de IA

Relatório

Nuno Filipe Araújo Gonçalves nº mecanográfico: 201402720

Introdução

Dado o grande leque de jogos que envolvam apenas 2 jogadores e que se cingem em jogadas por turnos, segundo um conjunto de regras, este trabalho visa descrever alguns métodos automáticos sobre como uma máquina poderá, dada uma determinada instância desse jogo, decidir a jogada que a põe mais perto da vitória.

Para efeitos deste trabalho iremos considerar o jogo 4 em linha e os algoritmos minmax, alfabeta e árvore de pesquisa de Monte Carlo.

Descrição dos algoritmos

MinMax

O algoritmo, dada uma instância do tabuleiro de jogo, gera uma árvore de pesquisa que contém todos os sucessivos movimentos possíveis a partir desse tabuleiro, e, para altura da árvore, calcula qual o melhor resultado para o jogador em questão. Para isso o algoritmo serve-se de uma heurística para calcular o valor da instância de tabuleiro nas folhas da árvore, e vai retornando recursivamente esse valor para os nós pais, escolhendo alternadamente entre o melhor e o pior valor dos filhos, seguindo a lógica de que o oponente escolherá, tbm ele, sempre a melhor opção possível. De notar que a recursividade do algoritmos obriga a que este se comporte como um dfs quando percorre a árvore de pesquisa (pesquisa em profundidade).

AlfaBeta

O algoritmo segue a mesma ideia que o algoritmo minimax, estando a grande diferença na maneira como percorre a árvore de pesquisa. Ao contrário do minmax, o algoritmo guarda os

melhores e piores valores heurísticos já encontrados num determinado nível da árvore, e só continua a percorrer para níveis inferiores se estiver certo de que esse caminho poderá melhorar o valor de heurística já encontrado.

Árvores de pesquisa de Monte Carlo

O algoritmo tenta apresentar uma alternativa ao problema de limitação espacial no contexto da geração das árvores de pesquisas dos outros dois algoritmos. O cerne do algoritmo consiste em gerar, no início, apenas uma parte da árvore e ir simulando, um número determinado de vezes, os filhos de uma das folhas da árvore. O algoritmo serve de uma fórmula matemática para a escolha equilibrada das folhas a simular e, no fim de todas as simulações, retorna o nó que foi visitado mais vezes. Apesar do resultado deste algoritmo ser sempre um valor aproximado daquele que na realidade é a jogada a executar, a incrível redução do tamanho espacial utilizado permite a solução para jogos que tenham um espaço amostral muito maior.

Quatro em linha: descrição

O jogo do quatro em linha consiste em alternadamente depositar peças brancas e pretas num tabuleiro de 6 linhas por 7 colunas, com o intuito de ser o primeiro a juntar 4 peças da mesma cor seguidas num segmento horizontal, vertical ou diagonal. A cada jogador é atribuída uma cor e as peças só podem ser jogadas na última linha do tabuleiro ou em cima de outras peças já jogadas no tabuleiro.

Algoritmos aplicados ao quatro em linha

Neste projeto utiliza se uma classe board para guardar instâncias do tabuleiro, bem como todos os métodos essenciais para a modificação e testagem dessa instância consoante os objetivos dos algoritmos e regras do jogo quatro em linha.

Para a construção das árvores de pesquisa utilizadas nos algoritmos minmax e alfabeta usa-se uma classe chamada Node, implementada na classe Algorithm, que guarda uma instância do nó da árvore, bem como os seus descendentes e outros valores utilizados na iteração dos algoritmos.

Para a construção de árvores de Monte Carlo, usa-se uma nova classe chamada MC Node, também ela implementada na classe Algorithm, que se encarrega de guardar instâncias dos nós dessas mesmas árvores. Como a interação deste algoritmos necessita que se guardem e atualizem valores diferentes que nas outras árvores, a criação de uma nova classe ajuda na leitura e estrutura do código construído.

MinMax: Constrói a árvore de pesquisa a partir da raiz (instância de tabuleiro da jogada anterior) através da chamada da função construtTree. Depois chama uma função auxiliar que, recursivamente, percorre e atualiza os valores heurísticos dos nós da árvore e retorna o melhor valor encontrado. Tendo esse valor sido calculado e retornado, o algoritmo vê qual dos filhos imediatos herdou esse valor heurístico e retorna a instância do tabuleiro que pertence a esse no filho que corresponde à melhor jogada .

AlfaBeta: Muito semelhante ao minmax, apenas se diferenciam na função auxiliar utilizada para o cálculo da melhor heurística. A função utilizada no alfabeta vai guardando e atualizando os melhores e piores valores de heurística encontrados, para escolher que ramos deve expandir na sua pesquisa, cortando assim caminhos que não influenciam no valor final da heurística.

Tanto o minmax como o alfabeta tem como fator limitante o tamanho da árvore de nós de pesquisa, sendo que no meu ambiente de teste e criação dos algoritmos apenas tinha capacidade para testar árvores com altura máxima 7. Assim sendo ambas as implementações dos algoritmos utilizam um valor, dado na altura da sua chamada, que limita o tamanho das árvores de pesquisa criadas.

Monte Carlo: Este algoritmo divide-se em 3 grandes partes: Primeiro, o algoritmo percorre a árvore de pesquisa selecionando o nó com o melhor valor da função de escolha. Quando chega a uma folha verifica se esta já foi visitada. Se esse for o caso então escolhe esse nó para a iteração da próxima parte. Se não for esse o caso, gera os nós filho dessa folha e escolhe o primeiro para a iteração da próxima parte.

Segundo, simula um caminho até um nó terminal, a partir do nó escolhido anteriormente, e retorna o valor heurístico desse no terminal.

Terceiro, o algoritmo faz backtracking a partir do nó folha escolhido na primeira parte, até á raiz da árvore, indo atualizando o número de vezes que os nós desse caminho foram visitados, e o seu valor total heurístico (que é a soma de todos os valores heurísticos de caminhos simulados que passaram por esse nó)