Trabalho sobre Métodos de busca (2016-2)

Universidade Federal de Santa Catarina - Ciência da Computação

Inteligência Artificial - INE5430-06208 (20162)

Ana Cristina Medaglia Dyonisio

Giovanni Rotta

A implementação realizada para a primeira parte do trabalho sobre Métodos de busca, Inteligência artificial para o jogo gomoku utilizando algoritmo minMax com podas, consiste em 4 classes, responsáveis por representar um estado do jogo, ou seja, uma 'foto' do tabuleiro, estruturar e criar os estados do jogo de uma forma adequada para realizar o algoritmo de minMax, verificar possíveis jogadas por peça e cálculo da pontuação por heurística.

A linguagem escolhida foi javascript, pela facilidade de se programar a parte gráfica assim como a parte lógica, e por ser a linguagem mais usada pelos integrantes do grupo, a intenção é que o resultado final seja acessado por uma página web.

Ciclo do Jogo

Para entender melhor o funcionamento da implementação é preciso antes ter uma visão geral do ciclo do programa. O primeiro passo, é decidir quem começa jogando, essa opção é feita pelo jogador humano, caso ele opte por começar, o jogo segue na lógica comum, caso o computador comece, terá um exceção ao ciclo natural apenas nesta jogada, que, sem nenhum cálculo, a peça é colocada no centro do tabuleiro.

O ciclo consiste em esperar a jogada do humano, após a jogada do humano, o computador começará a realizar seus cálculos a fim de optar pela melhor jogada, e para isso, precisa mapear todas as jogadas possíveis, levando em conta que o estado atual do tabuleiro é o que se encontra após a jogada do humano. É chamada então a função gerarNovosEstados, explicaremos mais detalhadamente posteriormente, essa função vai levantar todas as possibilidades coerentes de jogadas que pode ser realizadas a partir do estado atual, para cada estado gerado é possível chamar novamente a função, e então será gerado um novo nível de estados. Estes estados são conectados ao estado pai (estado a partir do qual foram gerados), por uma estrutura de dados de Grafo, essa estrutura foi construída, pelo membro do grupo, na disciplina de Grafos e esta implementada na classe Grafo.

Antes de inserido na estrutura de grafo, cada estado passa por uma verificação de pontos, a verificação que assim como o cálculo de heurística serão apresentados

em detalhes posteriormente, analisará as possíveis jogadas da peça adicionada ao novo estado e passará pela função de cálculo de heurística para gerar uma pontuação, quanto maior a pontuação melhor, o cálculo de heurística só analisa os benefícios daquela peça em relação a proximidade de realizar sequências, não avalia a 'defesa', para esta avaliação é realizada a mesma verificação e cálculo da heurística, porém para a peça do adversário, deste modo obtemos a pontuação que o adversário teria, quanto maior esta pontuação maior a importância para defender essa posição, podemos então somar os 2 valores para obter o valor final do estado. O inverso pode ser utilizado de forma análoga para o cálculo da heurística adversária.

Os pontos de cada estado herdam os pontos dos estados pai, e somam o valor da peça adicionada, deste modo não precisamos refazer o cálculo de todas as peças para cada estado.

O benefício de se utilizar a estrutura de grafos é que podemos manter os estados que pertencem ao ramo escolhido, gerando novos estados a partir dos últimos estados gerados, não foi definido o número de níveis de estados gerados ainda, mas no exemplo de gerar 3 níveis a cada jogada do humano, sobrará 1 nível de estado após a próxima jogada do humano, e mais 3 serão geradas, quanto mais jogadas forem realizadas mais níveis teremos.

Outra otimização não realizada é aumentar o nível de estados a medida que forem diminuíndo os espaços em brancos, pois terá menos estados para representar.

Classe GeradorEstados

A classe GeradorEstados possuí 3 parâmetros, tabuleiro, numeroCorte, verificador.

Tabuleiro é na verdade o estado inicial do jogo, e será o nodo raíz do grafo, uma instância da classe Tabuleiro, que representa um estado do jogo. NumeroCorte será utilizado na função de gerar novos estados, responsável por limitar as opções de jogadas possíveis. Verificador é uma instância do verificador, e será utilizada para verificar os pontos de cada estado.

A classe é construída criando o nodo raíz, e possuí 3 funções, gerarNovosEstados, criarEstado e calculoCentroDeMassa.

O calculoCentroDeMassa é como o nome diz, uma função que cálcula qual ponto do plano cartesiano formado pelo tabuleiro se encontra o centro de massa em relação a todas as peças do jogo, esse ponto será de onde começaremos nosso algoritmo de geração de estados, e supostamente se encontra onde a maior concentração de peças esta alocada e supostamente desta forma poderá gerar os estados com maior pontuação primeiro.

A função criarEstado é muito simples, consiste em adicionar a peça correspondente ao estado, verificar a pontuação da jogada através do verificador e retornar o novo estado.

A função gerarNovosEstados utiliza o valor do ponto de centro de massa para, a partir dele, gerar todas as combinações de jogadas possíveis. As jogadas são geradas de forma ondular, ou seja, primeiro são geradas todas os estados correspondentes as casas vizinhas do centro de massa (somente os espaços vazios irão gerar estados), depois acontece o mesmo com as casas vizinhas deste, e por ai vai, essas casas são captadas separadamente por direção, topo, baixo, esquerda e direita, e vão até o final do tabuleiro ou até o numeroCorte ser alcançado. Este número é alcançado quando em algumas das direções forem encontradas somente casas vazias n vezes, sendo n o número de corte, ou seja, caso a direção cima encontre 2 linhas vazias, e o número de corte for 2, então o algoritmo deixa de criar estados correspondentes as casas acima desta linhas. Esse algoritmo foi criado para evitar que sejam gerados estados irrelevantes as jogadas encontradas.

Classe Grafo

A classe grafo foi criada durante a disciplina de grafos no curso de computação da UFSC, e simula a estrutura de um grafo, podendo adicionar, remover e conectar vertices, além de uma série de outras funções relacionadas a grafos.

Classe Tabuleiro

Esta classe possuí como parâmetro 2 variáveis, linhas e colunas. Linhas representa o número de linhas do tabuleiro. Colunas, o número de colunas, e conexões. Esta classe possuí alguns atributos interessantes como número de peças brancas e pretas, um array com a posição das peças brancas e pretas o valor acumulado e o tabuleiro em si, um array 2d, com tamanho definido pelas variáveis do parâmetro.

O tabuleiro na verdade representa um estado do jogo, ou seja, uma 'foto' do jogo em um determinado momento.

Possuí 2 funções, adicionar peça ao tabuleiro, ou seja, adicionar um valor correspondente a cor no tabuleiro, e criar um novo tabuleiro, onde todas as casas do array 2d, estarão vazias.

Classe Verificador

Esta classe tem 2 própositos, verificar em um estado, suas características e fazer um cálculo de heurística baseado nelas.

A classe contém um parâmetro, multiplicadorDefensivo, que serve justamente como um índice para multiplicar a pontuação resultante para se defender do oponente, ou seja, impedir que ele complete a sequência de 5, quanto maior este multiplicador, maior importância será dada a jogadas defensiva.

A função verificarJogada é responsável por chamar as funções de verificação de características e com o resultado em mãos chamar a função de heurística para pontuar, este processo é feito 2 vezes, uma com a peça do jogador da vez, para verificar a importância ofensiva, e outra com a peça do oponente, para verificar a importância defensiva daquela casa, o valor obtido pela 2ª vez é multiplicado pelo multiplicador defensivo, as duas pontuações são somadas e representam o valor do estado.

Existem 4 funções de verificação: verificarLinha, verificarColunas, verificarDiagonalNoroeste e verificarDiagonalNordeste. Estas funções avaliam, a partir da peça colocada no estado, as possibilidades de jogadas que ela tem. Dois laços, um

para cada direção percorre a linha, ou coluna ou diagonais, até encontrar uma peça adversária ou então atingir o limite de 4 casas, após completar o laço alguns dados podem ser coletados, como por exemplo, o valor de peças em sequência, **e é nesta hora que o jogo verifica se determinado estado possuí uma sequência de 5 ou de 4 peças,** é também analisada as semisequencias, nome dado ao número de peças que não estão ligadas diretamente a peça analisada mas que podem formar sequência dentro daquele espaço, o número de casas vazias também é contado. Neste momento é verificado também se é possível completar 5 peças em sequência com aquela casa, em caso negativo, todas as características acima são zeradas, e o cálculo de heurística irá retornar um valor zero, ou seja, não vale a pena utilizar aquele estado em caso ofensivo.

Por fim, existe a função de heurística, ela calcula para cada verificador (linha, coluna, diagonais) do estado, sua pontuação, ela possuí 9 casos e mais uma regra para diminuir a probabilidade de valores iguais, que consiste em multiplicar o número de peças em sequência por 4 e o número de peças semisequência por 2. As heurística, seguidas da sua pontuação estão abaixo.

1º Heurística/Utilidade - Completar sequência de 5; – 500 pontos

2º Heurística - caso gere sequência de 4 peças e espaço para mais 1 peça; - 100 pontos

3º Heurística - caso gere sequência de 3 peças e espaço de 1 ou mais peças em cada extremidade da sequencia; - 100 pontos

4º Heurística - caso gere uma sequência de 2 peças e pelo menos mais 1 peça de semisequência, em uma linha com espaçamento de pelo menos 6 casas; - 100 pontos

5º Heurística - caso gere uma sequência de 3 peças e pelo menos 2 espaços vazios em uma das extremidades – 25 pontos

6º Heurística - caso gere uma sequência de 3 peças e espaço com mais de 2 casas vazias ou com peças próprias. - 20 pontos

7º Heurística - caso gere uma sequência de 3 peças e espaço com 2 casas vazias ou com peças próprias. - 15 pontos

8º Heurística - caso gere uma sequência de 2 peças e espaço com mais de 4 casas vazias ou com peças próprias. - 10 pontos

9º Heurística - caso gere uma sequência de 2 peças e espaço com mais de 2 casas vazias ou com peças próprias. - 5 pontos