Relatório do Sistema Inteligente do Ligue 4

Dupla: Gustavo Roesler Brillinger & Osvaldo Miguel Junior

Controller: SI-Ligue4\public\scripts\controllers\homeController.js
Html principal: SI-Ligue4\public\views\home.html

Heurística: vai ser criado uma função que vai percorrer todo o tabuleiro e apartir de cada peça vai calcular as possíveis sequências de 4, exemplo:

considere:
um tabuleiro vazio, com uma peça na posição(6,4)
O = peça no tabuleiro
X = espaços em branco que podem ser usados para criar sequencia de 4

as seguintes sequencias são possíveis na horizontal:
XXXO
XXOX
XOXX
OXXX

Sequências com + peças no tabuleiro tem valor maior, considerando o exemplo a cima com mais uma peça na posição(6,5)

as seguintes sequencias são possíveis na horizontal:
XXXO +5 pontos
XXOO +50 pontos
XOOX +50 pontos
OOXX +50 pontos

OXXX não é possivel pois só existem dois espaços livres ao lado direito da peça(6,5), faltando um espaço para completar a sequencia de 4. O mesmo vai acontecer se invez de colidir com a parede colidir com peças adversárias

Cada uma dessas sequências vai ter um valor positivo se for uma sequencia da IA e negativo se for do humano, sendo o somatório de todas as possíveis sequências o custo do respectivo estado do tabuleiro.

Tabela de pontuação:
1 peça: 5 pontos.
2 peças: 50 pontos.
3 peças: 500 pontos.
4 peças: 20 000 pontos.

Ao final de todos os somatórios é removido do valor final 1% por peça presente no tabuleiro.
Em caso de empate o retorno é sempre 0.

Utilidade: Utiliza a função heurística.

Nodo Folha: A função de heurística tambem vai verificar se um nodo é nodo folha. Cada nodo folha retorna 20 000 e 0 no caso de empate, para os nodos que não vão executar a função da heurística, foi criado uma função com o objetivo de verificar se existe vencedor que ira retorna ""(quando não existe vencedor), "empate"(no caso do tabuleiro terminar sem vencedor) ou 1(humano)/0(computador)(sendo que quem retorna é o vencedor), essa função verifica respectivamente todas as celulas na vertical, horizontal e ambas as diagonais por sequências de 4, se nenhuma sequência de 4 for encontrada a função verifica ao final se deu empate.

Limitações da Implementação : O algoritmo deve limitar sua busca minmax mesmo com a poda alpha/beta pois sua árvore é muito grande. O limite de profundidade definido foi 6, mas quando aumenta a profundidade em apenas 1 nível tem se um drastico aumento no número de iterações.

Otimização:
- Utiliza-se poda alfa/beta tendo seus primeiros valores -Infinity/+Infinity respectivamentes, sempre passando os valores de alfa/beta do pai para os filhos, atualizando alfa nos niveis max e beta nos niveis min. Sempre que alfa>beta efetua poda
- Para à poda alfa/beta ser executada mais frequentemente da preferência na busca sempre olhando as jogadas ao centro do tabuleiro até as da borda, considerando o tabuleiro tendo a sequinte sequência de colunas [0,1,2,3,4,5,6] utilizando a sequinte sequência [3,4,2,5,1,6,0] na busca.
- Como o tabuleiro é um array de arrays contendo 42 valores e é passado muitas vezes por parámetro foi alterado o seu valor das Strings ""/"humano"/"computador" para null/1/0 respectivamente.

Principais Variáveis:
$scope.vencedor: só utilizada para mostrar na interface quem venceu, podendo receber null(sem vencedor),0(computador venceu),1(humano venceu) ou"empate"(empate).

$scope.tabuleiro: guarda todas as posições do tabuleiro atual, podendo receber null(nimguem), 0(computador) ou 1(humano).

Principais Métodos:
me.jogada(tabuleiro, coluna, jogadordavez, callback): efetua uma jogada e retorna o tabuleiro resultante ou "colunaEstaCheia".

me.alteraJogadorDaVez(): altera entre a jogada do humano e computador.

$scope.jogadaHumano(coluna): efetua jogada do humano, se possível cria novo tabuleiro e começa jogada do computador, quando retornar jogada do computador imprime na interface ambas as jogadas.

me.jogadaComputador(): executa jogada computador, a coluna é decidida pelo algoritmo.

me.calcMinMax(alfa, beta, davez, filhos, nivel, tabuleiro, callback): utilizando minmax calcula a melhor coluna a ser jogada, retornando {pontuacao: 'valor',coluna: 'coluna'}

me.calcValorTabuleiro(tabuleiro, callback): utilizando o calculo da heurística retorna valor do tabuleiro enviado.

me.verificaFimDeJogo(tabuleiro): sempre executado após cada jogada para saber se fim de jogo ou no caso do algoritmo para saber se nodo folha, retorna null(sem vencedor),0(computador),1(humano) ou "empate".