UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO
CURSO DE SISTEMAS DE INFORMAÇÃO

GSI050 - INTELIGENCIA ARTIFICIAL

Profa. Márcia Aparecida

Dhiogo Pereira Santos – 12021BSI262

Ellen Christina Amaral Santana – 12011BSI208

Uberlândia, 07 de julho de 2022

- Busca Competitiva

Ambientes onde objetivos estão em conflito são ditos ambientes competitivos ou jogos. Nestes casos, são problemas de busca competitiva.
Em IA, os jogos mais comuns são aqueles de ambientes determinísticos, completamente observáveis em que existem dois agentes cujas ações se alternam e os valores de utilidade são iguais e simétricos. Por exemplo, se um jogador ganha, recebe +1, o outro que perde recebe -1.
A utilidade ou função de utilidade mapeia um estado em um valor que descreve o grau de "felicidade" com o estado.
Jogos envolvem tomar decisão, mesmo quando o cálculo da decisão ótima é inviável.
Considere jogos com dois jogadores, MAX e MIN.
MAX faz o primeiro movimento e os dois se revezam até o final do jogo. No final, pontos são dados ao vencedor e penalidades ao perdedor.

- **Algoritmo Minimax:**

Calcula a decisão minimax, a partir do estado atual. Recursivamente computa os valores minimax de cada estado sucessor, utilizando a equação anterior, desta forma, percorre todo caminho até às folhas, retornando os valores minimax na volta da recursão.

- **Algoritmo Alfa-Beta:**

A ideia é calcular a decisão Minimax sem examinar todos os nós na árvore do jogo, isto é, fazer uma poda.
Esta poda pode ser aplicada a árvores de qualquer profundidade, podendo podar subárvores inteiras e não apenas folhas.
Princípio: Considere um nó n em qualquer lugar da árvore e que o jogador tenha escolha de movimentos até este nó. Se o jogador tiver uma escolha melhor m no nó pai de n ou em qualquer outro ponto de escolha acima dele, então n nunca será alcançado.
É necessário lembrar valores Minimax já definidos em ramificações anteriores. Então, há dois parâmetros:

- α : o valor da melhor escolha, aquela de valor mais alto, encontrado até o momento em qualquer ponto de escolha para MAX.
- β : o valor da melhor escolha, aquela de valor mais baixo, encontrado até o momento em qualquer ponto de escolha para MIN.

Estes parâmetros são atualizados ao longo da busca, podando ramificações quando identifica que o valor do nó atual é o pior que os valores correntes destes parâmetros para MAX ou MIN.

- Descrição do problema (Jogo da Velha)

Inicia o jogo com uma base formada por três linhas e três colunas, resultando em 9 espaços vazios. Cada jogador, na sua vez, coloca a sua marca onde pretender (um joga com “0”, outro jogador com “X”).
O objetivo do jogo é fazer uma sequência de três símbolos iguais, seja em linha vertical, horizontal ou diagonal, enquanto tenta impedir que seu adversário faça o mesmo.
Quando um dos participantes faz uma linha, ganha o jogo;

- Principais elementos para a definição do algoritmo

Estado Inicial: especifica como jogo inicia.
Jogadores(s): define qual jogador move em um estado
Ações(s): retornam conjunto de movimentos válidos em um estado.
Resultado(s, a): o modelo de transição que define o resultado de um movimento “a” em um estado “s”.
Teste de término(s): é verdadeiro quando o jogo termina, caso contrário, é falso. Os estados em que o jogo termina são ditos estados terminais.
Utilidade(s, p): é uma função que define valor numérico para um jogo que termina no estado terminal s por um jogador p. A função é denominada função de utilidade. Por exemplo, podem ser dados os valores 1, 0 ou -1 se há vitória, empate ou derrota, respectivamente.

No nosso código, definimos as seguintes classes:

- Estado: nela temos o atributo tabuleiro, que é um vetor do tipo char de 9 posições, o atributo do tipo inteiro fator, que registra recursivamente a utilidade de cada estado, e a variável pai do tipo Estado, que recebe o estado a partir do qual foram gerados.

- Minimax: nesta classe que acontece a recursão, o método max verifica se está no estado terminal e caso não esteja, ele chama o min. O mesmo vale para o método min, que primeiramente verifica se tal estado é o estado terminal e caso não seja, chama max. Executa esse processo até se obter algum vitorioso, seja jogador min ou max ou então até resultar em um empate.

- AlfaBeta: essa classe contém o método minimax com a implementação de poda, tornando-o mais rápido e eficiente.

- Jogo: é a classe principal. Nela iniciamos as classes, chamamos os métodos até obter um resultado e fazemos as impressões dos estados.

- Conclusão relativa aos resultados

  Nos testes realizados executando o algoritmo configurado em máquina contra máquina, ele apresenta como resultado o empate, pois temos os jogadores max e min buscando simultaneamente a vitória atrapalhando um ao outro.
  Quanto ao tempo de execução, embora ambos tenham executado rapidamente, percebemos que o algoritmo da poda alfa-beta gastou praticamente a metade do tempo em relação ao algoritmo minimax.

- Referências

https://www.ic.unicamp.br/~ffaria/ia1s2017/class06/class06-Busca_competitiva.pdf

https://www.organicadigital.com/blog/algoritmo-minimax-introducao-a-inteligencia-artificial/