Atividade Prática de Inteligência Artificial de 04/04/2022

Mundo do Pacman Multiagente

Prof. Luan Garcia

Introdução

Continuaremos implementando técnicas de Inteligência Artificial no mundo do Pacman. Desta vez, projetaremos agentes para a versão clássica do Pacman, com fantasmas. Ou seja, trabalharemos em um mundo multiagente.

Lembrando que este projeto foi desenvolvido na universidade de Berkeley para disciplina de Inteligência Artificial (http://ai.berkeley.edu).

O projeto está disponível no portal Ulife, dentro da Aula 5, em um arquivo zipado com o nome de multiagent.zip. Dentro deste zip vocês encontrarão diversos arquivos Python que compõe o jogo.

Vocês modificarão apenas um arquivo:

multiAgents.py	Arquivo onde ficarão todos agentes de busca em mundos multi-agente.

Para entender melhor como o jogo funciona, você também deve olhar os seguintes arquivos (sem modificar!)

pacman.py	O arquivo main responsável por executar o jogo. Este arquivo descreve os estados do Pacman no tipo GameState, que será utilizado para desenvolver os seus algoritmos.
game.py	Onde a lógica por trás do mundo Pacman funciona. Aqui existem vários tipos utilizados no jogo, como AgentState, Agent, Direction e Grid.
util.py	Arquivo com diversas implementações úteis que podem ser aproveitadas, como estruturas de fila, pilha, fila de prioridade, etc.

Leia os comentários dos arquivos com atenção, pois você precisa entender a lógica por trás das chamadas do jogo para poder implementar seus algoritmos.

O restante dos arquivos apenas dá suporte aos gráficos do jogo e outras facilidades e podem ser ignorados.

Executando o jogo

Para executar o jogo pela primeira vez, dentro de um terminal que esteja na pasta do projeto, utilize o seguinte comando:

```
python pacman.py
```

Este comando executa um jogo em que o agente é você mesmo!

O agente mais simples no arquivo *multiAgents.py* é um o **ReflexAgent**. É uma implementação básica de um agente simples. Você pode testar ele executando:

```
python pacman.py -p ReflexAgent
```

Ele possui um desempenho ruim até mesmo em layouts muito simples. Utilize o comando abaixo no terminal para executar o agente de reflexo com um layout simples:

```
python pacman.py -p ReflexAgent -l testClassic
```

Sua tarefa será desenvolver um agente de busca adversária utilizando o algoritmo minimax.

Você deve completar a classe **MinimaxAgent**, que está contida dentro do arquivo *multiAgents.py*

Seu código deve ser genérico a ponto de aceitar um número arbitrário de fantasmas. Isto vai requerer uma pequena modificação no algoritmo que vimos em aula.

Seu código também deve ser capaz de expandir a árvore do jogo até uma profundidade arbitrária. Você pode atribuir valores para os nodos "folha" da sua árvore minimax utilizando a função **self.evaluationFunction**, que utiliza por padrão a função **scoreEvaluationFunction**. O **MultiAgentSearchAgent** dá acesso às funções **self.depth** (profundidade) e **self.evaluationFunction** que você precisará utilizar.

Importante: uma camada da árvore considera uma jogada do Pacman e mais uma jogada de cada fantasma. Ou seja, uma busca com profundidade 2 envolve o Pacman e cada fantasma se movendo 2 vezes.

O seguinte comando executará o jogo com seu agente minimax e uma profundidade de árvore 4:

```
python pacman.py -p MinimaxAgent -l mediumClassic -a depth=4
```

Dicas e Observações

- Implemente o algoritmo recursivamente, utilizando funções auxiliares.
- O layout padrão para a chamada é o próprio mediumClassic, mas você pode experimentar outros layouts como por exemplo minimaxClassic e openClassic.

- Mesmo uma implementação correta do minimax levará o Pacman a perder alguns jogos, isto não é um problema.
- O Pacman é sempre o agente 0, enquanto os fantasmas serão os demais índices do vetor de agentes.
- Todos estados no minimax devem ser **GameStates**, sejam passados para a função **getAction,** seja gerados por **GameState.generateSuccessor.**
- Se implementar corretamente o algoritmo, você vai perceber que o Pacman será muito bom em não morrer, mas ruim em ganhar. Geralmente vai ficar andando em voltas sem fazer progresso.

Pseudo código para o Minimax

```
function  minimax(nodo, profundidade, player) is
    if profundidade == 0 or nodo é terminal then
        return valor do nodo
    if player == false then /*maximizacao*/
        valor := −∞
        for each filho de nodo do
          valor := max(valor, minimax(filho,profundidade−1, FALSE))
        return value
    else /*minimizacao*/
        value := +∞
        for each filho de nodo do
          valor := min(value, minimax(filho,profundidade− 1, TRUE))
        return valor
```