

MAC0425-Inteligência Artificial EP02.

Nome : Victor Chiaradia Gramuglia Araujo.

Nusp : 9793756.

Passo 1 :

Q1 : Em 100 jogos, com 1 fantasma a evaluationFunction obteve uma win-rate de 100% e uma média de 1330.93 pontos. Com 2 fantasmas temos uma win rate de 91% e uma média de 1215.21 pontos.

Passo 2 :

Q2: Como a cada segundo o PacMan perde pontos, se ele acredita que irá perder (pois ele presume estar jogando com adversários ótimos), encerrar o jogo o mais cedo possível é a decisão mais lógica a ser feita, por isso ele vai até o fantasma mais próximo.

Q3: O agente reativo só considera o presente e o futuro imediato, já o minimax considera um futuro maior e também considera que seu inimigo é ótimo.

Q4: Usar Dijkstra para achar a distância das “foods” ao contrário de uma BFS iria gastar menos memória RAM e potencialmente menos tempo de computação.

Passo 3 :

Q5: Para coletar os dados rodamos “\$ python2 pacman.py -p [Agent] -l trappedClassic -a depth=[x] -q -n 10” mudando Agent e x e rodando 10 vezes para poder notar melhor a diferença no tempo. Com relação ao número de estados, para x = 3 e 4, para o Minimax e o AlphaBeta respectivamente, foram explorados no total 770, 1070, 500, 560 estados com tempo igual a 0.105s, 0.120s, 0.092s, 0.066s.

Para uma depth igual a dois, houve diferenças no número de estados e score gerado pelo teste por tanto usamos “\$ python2 pacman.py -p [Agent] -l trappedClassic -a depth=[x] -q -n 100” e com obtivemos 13738 estados em 1.007s para o miniMax e 12803 estados em 0.944s.

Vale a pena notar que os dois agentes só ganhavam quando tinham sua depth limitada para 2.

Passo 4 :

Q6: Como no Expectimax os jogadores min não irão jogar otimamente, devido a característica aleatória, o PacMan pode acabar escolhendo ações que podem levar os min a terem ações melhor, pois a possibilidade de uma outra ação nesse caminho que dá um “reward” mais atraente. Devido a aleatoriedade dos fantasmas, o PacMan acaba por tomar mais riscos quando usa o Expectimax e não o MiniMax.

Passo 5 :

Q7: Para a betterEvaluationFunction primeiro vemos se o estado é de vitória ou derrota, retornando infinito e menos infinito respectivamente. Adiciona-se ao score o inverso da distância da “food” mais próxima ($1/\alpha$), assim favorecendo estados que estão mais próximos que os estados anteriores. Adiciona-se o inverso da quantidade de “food” multiplicadas por um número ($1000/\beta$), assim favorecemos os estados com menos “food” e como $1/\alpha$ é no máximo 1, favorecemos coletar “foods” ao invés de achar o lugar que minimiza a distância das “foods”. Subtraímos 100 pontos do score para todo fantasma que possui uma distância manhattan menor que 3, garantindo que estamos ficando longe do perigo imediato. Por fim fazemos com que comer os “power pellets” seja desejável.