UNIVERSIDADE DE SÃO PAULO

Escola de Artes, Ciências e Humanidades

ACH2016 - Inteligência Artificial

Karina Valdivia Delgado

Ivo de Andrade de Deus 8075238

Relatório do Exercício Programa 1
Busca

São Paulo
2019

Estatísticas

Cenário 1 - Encontrando um ponto fixo de comida

Busca Implementada	Nós Expandidos	Tamanho do Plano
Busca em Profundidade (DFS)	15 146 390	Pequeno Médio Grande
Busca em Largura (BFS)	15 269 620	Pequeno Médio Grande
Busca de custo uniforme	15 269 620	Pequeno Médio Grande
Busca heurística	14 221 549	Pequeno Médio Grande

Cenário 2 - Encontrando os cantos do labirinto

Busca Implementada	Nós Expandidos	Tamanho do Plano
Busca em Largura (BFS)	253 1967 7950	Pequeno Médio Grande
Busca heurística	200 1137 4381	Pequeno Médio Grande

Comparações

Cenário 1 - Encontrando um ponto fixo de comida

No primeiro cenário, a busca em profundidade (DFS), apesar da maior eficiência em nós expandidos em relação aos outros métodos de buscas, não tem demonstrada sua ineficácia de obtenção de solução ótima, uma vez que está devolve o primeiro caminho a ser encontrado, ou seus problemas de eficiência, uma vez que algoritmo,

a priorizar profundidade, pode efetuar diversos cálculos necessários até encontrar a solução (por exemplo, em uma árvore de profundidade muito alta, onde o algoritmo verifica ramo demasiadamente grandes antes de encontrar a solução a poucos nós da origem).

Colocando em foco os resultados similares entre a busca em largura (BFS) e a busca de custo uniforme (UCS), confirma-se a teoria em que ambas são idênticas quando o custo dos passos são iguais entre si, uma vez que a USC é estruturada de forma similar a BFS com a prioridade em caminhos de custos de menor valor a cada passo. Porém, com todos os passos sendo idênticos, essa prioridade perde toda a influência no algoritmo baseado no BFS. A única diferença entre as buscas, portanto, é o fato que a busca em largura se encerra assim que esta alcança seu objetivo, enquanto a busca de custo uniforme não, uma vez que, numa eventualidade dos custos serem diferentes, a solução encontrada pela BFS pode não ser ótima.

Por fim, a busca heurística (ou busca A*) aperfeiçoa o processo de ambas a BFS e UCS, uma vez que sua heurística, ao satisfazer suas condições de otimalidade, visa uma busca ao objetivo com maior eficiência. A função de heurística h(n), que tem por si só estimar o custo do caminho de menor custo de um dado nó n para o estado objetivo, quando aliada ao custo do caminho já encontrado do estado inicial até o nó n, chamado de g(n) (que pela construção do algoritmo é sempre ótimo quando o nó n passa a ter seus vizinhos analisados) oferece uma função f(n) = g(n) + h(n) que melhor estima o custo total do caminho resposta que passe por n. Dado que a heurística h(n) para esse exercício se encontra em condições de otimalidade (h(n) é consistente, uma vez que h(n) <= c(n,a,n') + h(n') para um nó n' que venha após de um nó n; g(n) é consistente, quando a encontrar o caminho ótimo do nó de início até o nó n quando este é analisado; e, por consequência, f(n) é crescente ao longo dos nós expandidos por A*), a busca heurística sucede em otimizar a busca por uma solução ótima, como demonstrada pelas estatísticas adquiridas.

Cenário 2 - Encontrando os cantos do labirinto

No segundo cenário, de forma similar, comparam-se a busca em largura (BFS) e a busca heurística (A*) onde, como mencionado antes, ambas são buscas completas, mas a BFS entrega a solução mais rasa possível (a primeira solução com o menor número de nós possível), o que não é necessariamente a solução ótima ao problema de forma geral. Enquanto isso, não só a heurística verificaria os casos em que a solução ótima não seria a solução rasa (o que seria o caso se o problema envolvesse custos de passo não-semelhantes entre si), mas também estrutura seu algoritmo para priorizar os caminhos com a melhor função f(n) de custo total estimado a partir de uma função de heurística h(n) (sob condições de otimalidade), assim adquirindo uma resposta com mais eficiência em relação ao BFS, como exposto pelas estatísticas do cenário.

Questões

Questão 1 - Busca em Profundidade

"A ordem de exploração do espaço de estados seguiu conforme esperado? O Pac-Man de fato se move para todos os estados explorados em sua deliberação para encontrar uma solução para a meta? A sua implementação de busca em profundidade encontra uma solução de custo mínimo? Por quê?"

Não, o Pac-Man segue as rotas encontradas pelos algoritmos de busca, uma vez que são estes algoritmos que visitam os espaços do labirinto a fim de construir o caminho a ser percorrido pelo Pac-Man. Já a busca em profundidade não encontra uma solução de custo mínimo, uma vez que esta percorre em profundidade, o que abre a possibilidade de cálculos desnecessários ao se calcular ramos por sua completa extensão antes de encontrar o estado final em poucos níveis num ramo adjacente (condição que piora quanto mais níveis a árvore ou grafo apresentar).

Questão 2 - Busca em Largura

"A sua implementação de busca em largura encontra uma solução de custo mínimo? Por quê?"

A busca em largura, neste caso, encontra uma solução de custo mínimo, mas apesar pelo fato de todos os custos de passo serem semelhantes entre si. Numa condição em que esses passos fossem de valores diferentes, a BFS só encontraria a solução mais rasa (a primeira solução com o menor número de nós possível), o que não necessariamente seria a solução ótima quanto ao custo.

Questão 3 - Busca de Custo Uniforme

"Execute os comandos abaixo e explique sucintamente o comportamento dos agentes StayEastSearchAgent e StayWestSearchAgent em termos da função custo utilizada por cada agente.

$ python pacman.py -l mediumDottedMaze -p StayEastSearchAgent

$ python pacman.py -l mediumDottedMaze -p StayWestSearchAgent"

Os agentes StayEastSearchAgent e StayWestSearchAgent são agentes de busca de custo uniforme que tem funções de penalização de passos para posições ao Oeste e Leste, respectivamente, para um labirinto médio que contém 'pellets' normais, que devem ser coletadas antes do Pac-Man chegar ao ponto final do labirinto, uma 'power pellet'. A função de custo por entrar numa posição (x, y) é de 0.5 vezes x (para StayEastSearchAgent) ou de 2 vezes x (para StayWestSearchAgent). Uma vez que o cenário passa a ter custos de passos diferentes comparados aos cenários testados no EP, a UCS efetua a busca com preferência horizontais de movimentação (como o próprio nome das funções sugerem, em inglês) e, com isso, faz que o Pac-Man foque um percurso a direita da tela em StayEastSearchAgent, o que possibilita a conclusão do labirinto com pontos extra, e um percurso a esquerda da tela em StayWestSearchAgent, o que impossibilita a conclusão do labirinto uma vez que a UCS só está programada para achar o ponto final.

Questão 4 - Busca Heurística

"Você deve ter percebido que o algoritmo A encontra uma solução mais rapidamente que outras buscas. Por quê? Qual a razão para se implementar uma heurística consistente para sua implementação da busca A*?"*

A busca heurística encontra uma solução mais rapidamente que as outras buscas porque sua função heurística h(n) para um dado nó n, aliada à análise do caminho ótimo g(n) do nó de início até o nó n, gera uma função f(n), de custo total estimado do caminho que passa por n, que otimiza a busca ao estado final por meio do estudo de custos de caminho, algo que não é considerado na busca em largura, uma vez que esta não apenas entrega a solução mais rasa (que pode não ser ótima quando os custos de casos são diferentes) mas também busca todos os elementos de um dado nível antes de prosseguir ao próximo, e na busca de custo uniforme, uma vez que esta considera os custos de passos e, uma vez que todos sejam idênticos, a UCS opera de forma similar ao BFS (exceto o seu ponto de parada, além da primeira solução).