MAC425/5739 - Inteligência Artificial

Vítor Kei Taira Tamada - 8516250

Exercício-programa 1 - Relatório

a) Compilar em tabelas as estatísticas das buscas implementadas

	Size	Nós	Custo
DFS	Tiny	15	10
	Medium	146	130
	Big	390	210
BFS	Tiny	15	8
	Medium	269	68
	Big	620	210
IDS - 1 [1]	Tiny	64	8
	Medium	8699	68
	Big	60211	210
IDS - 10 [1]	Tiny	11	10
	Medium	982	68
	Big	6069	210
UCS	Tiny	15	8
	Medium	269	68
	Big	620	210
A*	Tiny	14	8
	Medium	221	68
	Big	549	210

Tabela 1: Estatísticas das buscas implementadas em termos de nós expandidos e custo em cada labirinto

■

b) Discutir os méritos e desvantagens de cada método, no contexto dos dados obtidos

I) Busca em profundidade (DFS): A busca em profundidade tem a sorte como fator importante: dependendo de qual caminho ela decidir percorrer primeiro, pode obter uma solução (não necessariamente a melhor) em pouco ou muito tempo. No caso dos testes, ela retornou uma solução rapidamente (poucos nós expandidos), mas não as melhores (custo mais alto nos labirintos pequeno e médio). A vantagem é que os únicos nós armazenados na memória são os do caminho encontrado.

II) Busca em largura (BFS): A busca em largura retorna o menor caminho entre a origem e o destino caso ele exista. Para isso, ela verifica todos os nós a uma distância i da origem, sendo $i = 1, 2, ...$ O maior problema deste método de busca é o uso excessivo da memória e a expansão de muitos nós que podem até mesmo estar se distanciando da solução.

III) Busca de aprofundamento iterativo (IDS): A busca de aprofundamento iterativo combina a economia de memória da DFS (os únicos nós armazenados são os do caminho encontrado entre a origem e o destino) com a completudo e menor caminho da BFS. O maior problema deste método de busca é o fato de ele verificar os nós mais próximos da origem diversas vezes, havendo certo gasto de tempo, como é evidente pelo número muito maior de nós expandidos se comparar com os resultados das outras buscas.

[1] Nos testes de busca de aprofundamento iterativo, o limite de cada iteração aumentou em 1 na IDS - 1 e em 10 na IDS - 10. O intuito foi verificar a influência do aumento do limite da busca a cada iteração no número de nós expandidos

IV) Busca de custo uniforme (UCS): A busca de custo uniforme é idêntica a uma busca em largura se todas as ações tiverem o mesmo custo. Por outro lado, se os custos das ações variarem entre si, a UCS garante encontrar o caminho mais barato, não sendo esse necessariamente o mais curto. A desvantagem deste método é o de haver a possibilidade de ele agir da mesma forma que uma BFS, evidenciado pelos números desta busca terem sido idênticos aos da BFS, já que os labirintos testados tem o mesmo custo para qualquer ação.

V) Busca heurística (A*): A busca A* procura o menor caminho (se as ações tiverem o mesmo custo) ou o mais barato, expandindo o menor número de nós possível.

Comparando os resultados das buscas, o algoritmo A* é claramente o melhor (considerando que a busca em profundidade encontrou um caminho correto rapidamente por acaso): ele conseguiu encontrar caminho de menor custo com menos nós expandidos.

Para que esta busca seja, de fato, a melhor, é necessário desenvolver uma heurística para que o problema seja resolvido pelo caminho ótimo. Logo, diferente das buscas mencionadas anteriormente, ela não é uma fórmula pronta e, portanto, é mais complicada de se implementar corretamente.

■

c) Responder as questões teóricas

Questão 1 - Busca em profundidade: Apesar de haver diversos nós expandidos no labirinto, o Pac-Man percorre apenas aqueles necessários para alcançar o objetivo. Como eu não tenho certeza de como o método `getSuccessors()` itera sobre os estados sucessores, posso apenas afirmar que o resultado em `tinyMaze` foi conforme o esperado enquanto no `mediumMaze` e no `bigMaze` os resultados foram melhores que o esperado.

Questão 2 - Busca em largura: A busca em largura encontra um caminho de custo mínimo contanto que todas as ações tenham o mesmo custo, uma vez que o caminho de menor custo é o mais curto nessas circunstâncias. Se o custo das ações variar, a busca em largura não necessariamente encontrará o caminho de custo mínimo.

Questão 3 - Busca de custo uniforme: Uma vez que a busca de custo uniforme sempre executa a ação de menor custo em cada estado, o agente `StayEastSearchAgent` mantem-se do lado leste do mapa pois a ação de ir para a esquerda (oeste) é a mais cara e a de ir para a direita (leste) é a mais barata (seja (x, y) a coordenada do estado resultado da ação a ser tomada, a função de custo é 0.5^x, ou seja, ir para a esquerda - reduzir o valor de x - tem custo maior) dentre as quatro (norte, sul, leste e oeste). Analogamente, `StayWestSearchAgent` tem a ação de ir para a direita (leste) como a mais cara e a de ir para a esquerda (oeste) como a mais barata (neste caso, a função de custo é 2^x, ou seja, ir para a direita - aumentar o valor de x - tem custo maior).

Questão 4 - Busca de aprofundamento iterativo: Sem a modificação, a busca de aprofundamento iterativo não é ótima pois pode acabar expandindo menos nós, uma vez que um nó n pode ser de profundidade máxima por um caminho a e profundidade não-máxima por um outro caminho b explorado depois de a. Se isso ocorrer sem a modificação, não haverá a devida expansão que passe por n.

Questão 5 - Busca heurística: O algoritmo A* encontra uma solução mais rapidamente pois considera não apenas o custo de cada ação como também um conhecimento prévio (heurística) que ajuda a determinar qual ação é de fato melhor tomar e não apenas aparenta ser.

A razão para se implementar uma heurística consistente é garantir que um determinado estado seja alcançado pela primeira vez utilizando o caminho ótimo.

Questão 6 - Agentes adversariais: O agente reativo tem mais problemas para ganhar porque move-se pensando apenas em se manter longe dos fantasmas e próximo da comida. Como ele também não enxerga muito longe, é mais facilmente encurralado. O agente reativo jogaria melhor se tivesse um campo de visão e conhecimento maior sobre o mapa, e soubesse dar a devida prioridade entre ficar longe do fantasma e ir atrás da comida de acordo com as respectivas distâncias.

■

d) Sugerir possíveis melhorias ao sistema e relatar dificuldades: As maiores dificuldades enfrentadas para se resolver o exercício envolveram a extração e uso

das informações necessárias para resolver o exercício, tais como receber as coordenadas atuais do Pac-Man e dos fantasmas, e as ações possíveis, e como manipular essas informações para programar o comportamento dos agentes. Minha principal sugestão é maior clareza em como manipular os dados disponíveis para construir as buscas e comportamentos uma vez que imagino que essa não deveria ser a parte desafiadora do problema, mas sim a parte lógica.