Relatório do Trabalho de Inteligência Artificial

Gissely Souza

8 de setembro de 2019

1 Desempenho do Algoritmo de Busca em Profundidade

A Tabela 1 mostra o desempenho deste algoritmo nos casos de teste.

O algoritmo funciona percorrendo o grafo de possibilidades de posições de modo que ele sempre tenta expandir os próximos filhos de ada vértice encontrado, e só abandona um vértice depois de se certificar de que nenhum de seus filhos leva ao objetivo esperado.

A implementação usa uma pilha para armazenar os vértices e sempre escolhe expandir o último que foi colocado.

Nota-se intuitivamente ao observar o comportamento do algoritmo nos testes que o caminho escolhido muitas vezes é bem longe do ótimo. Isso ocorre porque o algoritmo muitas vezes ignora completamente um caminho somente por não ser a primeira opção que aparece, e insiste em caminhos piores somente porque mesmo quando eles são ruins, se não levam a locais sem saída, eles não são abandonados.

2 Desempenho do Algoritmo de Busca em Largura

A Tabela 2 mostra o desempenho deste algoritmo nos casos de teste.

Este algoritmo demonstra um desempenho melhor que o anterior, pois ele sempre tenta encontrar soluções primeiro em locais mais próximos para só então partir para posições mais distantes. Isso faz com que ele não tome decisões que soam intuitivamente ruins. Seu maior problema é não ser um método heurístico e que por isso pode demorar muito para encontrar a solução e não lidar com casos nos quais existem custos diferentes de movimentação.

Tamanho	Pontuação	Custo	Expansões	Tempo (s)
Pequeno	500	10	15	0.0
Médio	380	130	146	0.0
Grande	300	210	390	0.2

Tabela 1: Algoritmo de Busca em Profundidade nos casos de teste

Tamanho	Pontuação	Custo	Expansões	Tempo (s)
Pequeno	502	8	15	0.0
Médio	442	68	269	0.1
Grande	300	210	620	0.1

Tabela 2: Algoritmo de Busca em Largura nos casos de teste

Tamanho	Pontuação	Custo	Expansões	Tempo (s)
Pequeno	502	8	15	0.0
Médio	442	68	268	0.1
Grande	300	210	620	0.2

Tabela 3: Algoritmo de Busca Uniforme nos casos de teste

A implementação é muito semelhante à anterior, mas utiliza uma fila ao invés de uma pilha.

A desvantagem deste método frente ao anterior está na quantidade maior de expansões que precisaram ser feitas em todos os casos para se achar o caminho. Ele certamente consome mais memória. Nos casos de teste o tempo não mostrou grande diferença, mas talvez em exemplos maiores ainda, possa-se notar uma diferença entre os dois métodos.

3 Desempenho do Algoritmo de Busca Uniforme

A Tabela 3 mostra o desempenho deste algoritmo nos casos de teste.

Este algoritmo é muito semelhante ao anterior, sendo a principal diferença o fato de que ele leva em conta o custo de percorrer o grafo de busca. Se todas as arestas tem o mesmo custo, ele se comporta exatamente como a busca em largura. Se não, ele sempre escolhe para expandir o próximo vértice cujo custo total para se chegar nele for o menor.

Sua implementação não usa nem uma fila nem uma pilha, mas sim uma lista que sempre mantemos ordenada de acordo com o custo para se chegar em cada vértice. Escolhemos sempre expandir a primeira posição.

Para o caso dos testes envolvendo o labirinto do Pacman, não foi notada muita diferença de desempenho. Em outros tipos de problema, onde há uma maior diferença de custo em cada aresta, a diferença pode ser mais pronunciada.

4 Desempenho do Algoritmo de Busca A*

A Tabela 4 mostra o desempenho deste algoritmo nos casos de teste.

Esse é o algoritmo que apresentou o melhor desempenho do ponto de vista da quantidade de vértices expandidos e o pior do ponto de vista do tempo em labirintos muito grandes. O desempenho de tempo pode ser atribuído devido à necessidade de cálculo da heurística para cada vértice e à particularidade desta implementação que pode ainda ser otimizada.

Tamanho	Pontuação	Custo	Expansões	Tempo (s)
Pequeno	502	8	14	0.0
Médio	442	68	217	0.1
Grande	300	210	542	0.4

Tabela 4: Algoritmo de Busca A* nos casos de teste

O algoritmo funciona como o anterior, com a diferença de que soma o custo para se chegar a cada vértice com uma heurística que prevê o custo para se chegar de cada vértice até o objetivo. No caso do problema de teste, que consiste em um movimento em grid que permite 4 movimentos diferentes, a heurística Manhattan é reconhecidamente a melhor e consiste na soma da distância horizontal e vertical de cada vértice até o objetivo.

5 Conclusão

Dos algoritmos apresentados, o A* foi o que apresentou o melhor desempenho e o de Busca em Profundidade foi o pior. O segundo é o pior principalmente por apresentar resultados e soluções insatisfatórias. A Busca em Largura, por outro lado, resolve tal problema, mas o custo para obter as soluções ótimas dela é muito alto em termos de vértices expandidos. A Busca Uniforme não é algo que faz muito sentido de ser usado em problemas como este, mas pode tornar-se importante em labirintos nos quais existam custos diferentes associados a cada quadrante a ser percorrido.