Trabalho 1 - MC906

GUSTAVO HENRIQUE STORTI SALIBI
Ciência da Computação - Graduação
E-mail: g174135@dac.unicamp.br

***Resumo* – As soluções de problemas com busca em grafos são amplamente utilizadas nas mais diversas aplicações, tal como em rotas de mapas. Existem diversos algoritmos passíveis de serem utilizados para se automatizar o processo de encontrar um caminho mínimo entre dois pontos. Assim, este trabalho visa comparar algumas dessas possibilidades.**

I. INTRODUÇÃO

Através da abstração de um problema e da discretização de suas ações e estados, é possível modelar uma solução com busca em grafos. Existem buscas não informadas e buscas informadas, onde utilizam-se o que chamamos de heurísticas, que ditam o quão bem estamos indo em direção à solução. Na estrutura de busca, temos uma definição dos nós, que possuem estado, nó-pai, custo do caminho e profundidade. Temos a definição do problema, que possui, dentre outras coisas, as ações válidas. Temos, também, a estrutura de fila e os algoritmos de busca propriamente ditos. Fazendo uso desses conhecimentos, é possível resolver o problema do robô descrito na seção seguinte, onde a modelagem é detalhada. A seção III trata do trabalho realizado a fim de se chegar a solução do problema descrito anteriormente. Já a seção IV diz respeito aos materiais e métodos utilizados e aos resultados esperados. A seção V mostra os dados obtidos pelo programa desenvolvido. A seção VI, por fim, discute e conclui sobre os resultados obtidos ante aos esperados.

II. MODELAGEM DO PROBLEMA

No problema dado, busca-se encontrar um possível caminho para o robô se locomover da sua posição inicial (x,y,θ) para sua posição destino (x2, y2), desviando dos obstáculos que se encontram no mapa. Dado que o robô é capaz de perceber onde se encontra a cada instante de tempo, que pode se locomover para uma posição adjacente e que o ambiente é determinístico, o problema foi modelado da seguinte maneira:

A. Representação do Ambiente

O ambiente é representado por um mapa 2D de 120x120 píxeis com saída em formato de imagem PNG, onde cada píxel (x,y) da imagem representa uma posição no ambiente.

B. Representação do Estado

O estado representa a posição do robô no mapa. Isso é feito através de uma tupla (x,y).

C. Conjunto de Ações

O robô pode se locomover para as suas posições adjacentes desde que elas não sejam barreiras. Abstraindo-se o conceito do ângulo θ em (x,y,θ) e o considerando como sendo de 45 graus * n, para n natural, podemos representar suas ações como movimentos para posições (x$\pm$1, y$\pm$1): (-1, -1), (0, -1), (1, -1), (-1, 0), (1, 0), (-1, +1), (0, +1) e (1, +1).

D. Teste do Estado Objetivo

Verifica se o estado atual, representado por (x,y), é igual ao par par ordenado que representa o estado objetivo.

E. Custo do Caminho g(x)

Para se calcular o custo de g(x), soma-se o custo de cada passo dado no caminho, sendo cada passo uma das ações previamente descritas. Para se calcular o custo de cada passo, utiliza-se uma função que retorna a distância em linha reta entre os dois pontos do passo dado.

III. TRABALHO PROPOSTO

A. Algoritmos de Busca

A princípio, uma função Best First Search foi implementada e, com base nela, todas as outras funções utilizadas na resolução do problema foram criadas. Foram utilizadas as seguintes funções de busca sem informação:

- Breadth First Search
- Uniform Cost Search

Foram utilizadas as seguintes funções de busca informada (foi acrescentada uma função além do pedido no enunciado para melhor comparação de resultados):

- A*
- Greedy

B. Heurísticas Utilizadas

Foram utilizadas as seguintes heurísticas junto aos algoritmos A* e Greedy:

- Distância em linha reta
- Distância Manhattan

C. Entradas Utilizadas

Uma vez definido o problema e criadas as funções responsáveis por implementá-lo, foram criadas instâncias para representar diferentes mapas e diferentes posições iniciais e finais. Foi criado, a princípio, um mapa m1 que representa a configuração sugerida no enunciado, Depois, foram criados outros nove mapas m2, m3, m4, m5, m6, m7, m8, m9 e m10, com posições iniciais e finais diferentes e com barreiras

geradas aleatoriamente, em quantidades diferentes para cada mapa, através de outra função.

D. Processamento das Entradas

Cada um dos mapas criados foram submetidos aos seguintes métodos de busca:

- Breadth First Search
- Uniform Cost Search
- A* com heurística de distância em linha reta
- A* com heurística de distância Manhattan
- Greedy com heurística de distância em linha reta
- Greedy com heurística de distância Manhattan

Durante o processo de busca de cada método, foram coletadas informações para serem utilizadas posteriormente na análise dos resultados. Dentre as informações coletadas, estão: número de estados visitados, número de nós visitados, custo do caminho encontrado e quantidade de passos dados durante o caminho. Essas informações são disponibilizadas na saída do programa. Para representação visual do desempenho dos algoritmos, uma imagem é gerada para cada um deles.

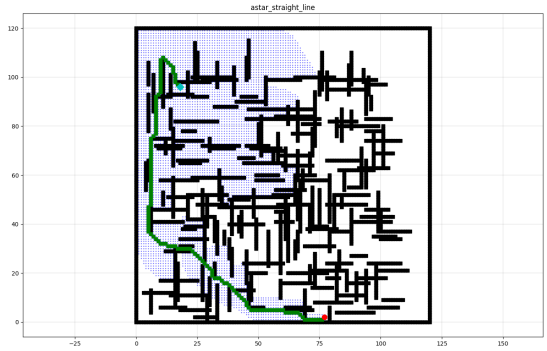

Figura 1. m6 com A* utilizando heurística de distância em linha reta.

Na imagem, o ponto azul representa a posição inicial; o ponto vermelho, a posição final; a linha verde, o caminho encontrado; os pontos azuis representam as posições exploradas no mapa; os pretos representam as barreiras; e, por fim, os pontos brancos são a área explorável no mapa.

IV. Materiais e Métodos

Para a implementação da solução, foram utilizados alguns arquivos modificados a partir da biblioteca disponibilizada pelo livro Artificial Intelligence - A Modern Approach (https://github.com/aimacode/aima-python).

A partir dessa biblioteca, foram criadas as classes e métodos que representam:

- Nós
- Fila de prioridade
- Problemas
- Custo de caminho
- Sequencia de estados para alcançar nó
- Os algoritmos de busca utilizados
- Gerador de mapas aleatórios
- Contador de chamadas
- Relatório com estatísticas de desempenho
- Plotagem dos mapas em imagens

Esperava-se encontrar nos resultados obtidos as características teóricas referentes a cada método de busca implementado. São elas:

- Breadth First Search: Completa; Ótima; Tempo e espaço $O(b^2 + 1)$
- Uniform Cost Search: Completa; Ótima; Tempo e espaço $O(b^{\lceil C*/e \rceil} + 1)$
- A* com distância em linha reta: Completa; Ótima; Tempo e espaço $O(b^d)$
- A* com distância Manhattan: Completa; Não Ótima; Tempo e espaço $O(b^d)$
- Greedy com distância em linha reta: Completa; Não Ótima; Tempo e espaço $O(b^d)$;
- Greedy com distância Manhattan: Completa; Não Ótima; Tempo e espaço $O(bm)$

V. Resultados

Junto a este aquivo, se encontram o código fonte do programa e as imagens obtidas na saída do programa. Para acessá-las, basta abrir a pasta ”imagens”. Para rodar o programa, basta seguir as orientações do arquivo ”REAME”também junto a essa submissão. As tabelas da próxima página apresentam os dados obtidos nos relatórios da saída do programa:

Tabela I
BREADTH FIRST SEARCH

Mapa	Estados	Nós	Custo	Passos
m1	13.280	109.762	197,1	164
m2	14.001	117.552	169,8	123
m3	9.431	86.457	138,1	105
m4	11.779	90.430	130,8	118
m5	10.013	73.994	164,6	136
m6	9.367	73.200	177,5	158
m7	8.722	63.966	293,1	255
m8	7.400	51.973	224,1	196
m9	9.566	74.649	134,6	111
m10	11.582	99.267	177,5	139
TOTAL	**105.141**	**841.250**	**1.806.5**	**1.505**

Tabela II
UNIFORM COST SEARCH

Mapa	Estados	Nós	Custo	Passos
m1	13.294	109.762	197,1	164
m2	14.001	115.593	169,8	123
m3	11.179	85.097	138,1	105
m4	11.056	79.430	130,8	118
m5	10.696	76.309	164,6	136
m6	9.275	66.615	177,5	158
m7	8.707	61.636	293,1	255
m8	7.396	51.250	223,3	196
m9	10.040	70.449	134,6	111
m10	11.582	84.067	177,5	139
TOTAL	**107.226**	**800.208**	**1.806,5**	**1.505**

Tabela III
A* COM DISTÂNCIA EM LINHA RETA

Mapa	Estados	Nós	Custo	Passos
m1	8.955	101.952	197.1	164
m2	2.029	15.585	169.8	123
m3	4.422	52.273	138.1	105
m4	4.620	41.238	130.8	118
m5	3.468	35.151	164.6	136
m6	4.550	44.039	177.5	158
m7	7.412	72.973	293.1	255
m8	5.628	50.150	223.3	196
m9	2.869	23.014	134.6	111
m10	4.341	46.253	177.5	139
TOTAL	**48.294**	**482.628**	**1806.5**	**1,505**

Tabela IV
A* COM DISTÂNCIA MANHATTAN

Mapa	Estados	Nós	Custo	Passos
m1	7.788	17.,725	197,1	164
m2	572	956	173,9	130
m3	994	8.653	138,1	105
m4	2.708	26.395	130,8	118
m5	1.824	17.613	164,6	136
m6	4.004	56.508	181,6	165
m7	3.959	37.750	295,6	262
m8	4.425	42.477	224,1	193
m9	1.711	13.834	134,6	111
m10	659	1.972	186,6	153
TOTAL	**28.644**	**377.883**	**1.826,9**	**1.537**

Tabela V
GREEDY COM DISTÂNCIA EM LINHA RETA

Mapa	Estados	Nós	Custo	Passos
m1	3.299	405.181	197,1	164
m2	572	956	173,9	130
m3	507	1.804	143,6	110
m4	556	4.773	155,7	135
m5	916	10.718	214,5	176
m6	1.711	101.797	251,9	213
m7	3.218	83.905	316,5	278
m8	2.351	29.408	275,3	221
m9	545	2.007	178,9	140
m10	569	1.099	188,8	147
TOTAL	**14.244**	**641.648**	**2.096,3**	**1.714**

Tabela VI
GREEDY COM DISTÂNCIA MANHATTAN

Mapa	Estados	Nós	Custo	Passos
m1	3.418	222.169	197,1	164
m2	572	956	173,9	130
m3	529	1.771	154,6	126
m4	669	7.304	144,2	121
m5	968	10.489	216,9	180
m6	2.250	24.362	343,2	274
m7	3.121	49.571	316,8	277
m8	1.839	24.519	252,8	194
m9	861	6.721	180,9	142
m10	571	1.143	188,2	146
TOTAL	**14.798**	**349.005**	**2.168,6**	**1.754**

VI. Conclusões

Como o tempo de processamento em cada computador varia bastante, iremos analisar apenas as saídas produzidas pelo programa. Elas mostram a quantidade para cada uma das operações realizadas listadas nas tabelas acima. Como podemos verificar, os métodos de busca informada obtiveram resultados muito melhores em termos uso de memória e processamento. Consideraremos o custo de memória e de processamento iguais aqui, dada a implementação dos algoritmos.

Como esperado, os métodos Breadth First Search e Uniform Cost Search nos trouxeram os caminhos de menor custo nos resultados. O método A* com heurística de distância em linha reta também nos trouxe caminhos ótimos. Isso se deu por conta da heurística em questão nunca superestimar o custo para alcançar o objetivo, já que a linha reta sempre será a menor distância possível. A* com heurística de distância Manhattan, Greedy com heurística de distância em linha reta e Greedy com heurística de distância Manhattan, entretanto, não apresentaram caminhos de menor custo em todos os casos. Isso se dá, no caso de A*, por conta da heurística, por vezes, superestimar o custo para alcançar o objetivo, já que é permitido movimento em diagonal e não apenas em quadrado, como nesse tipo de distância. Os algoritmo de Greedy, por natureza, nunca tem otimicidade garantida.

O desempenho em termos de custo de memória e processamento dos algoritmos não ótimos, entretanto, são significativamente superiores. Chegando, no mapa m2, a 14.001 estados visitados pelo Breadth First Search contra apenas 572 (4%) pelo Greedy com heurística de distância Manhattan. Já a quantidade de nós visitados varia, nesse caso, de 117.552 no BFS para apenas 956 (0,8%) no Greedy. Por outro lado, o custo da soma dos caminhos encontrados é de 2.168,6 no Greedy, enquanto que o caminho mínimo tem custo de 1.806,5.

Por fim, o algoritmo A* com heurística de distância Manhattan apresenta um bom meio termo. Onde, no total dos 10 mapas, são visitados apenas 28.644 estados e o caminho mínimo total não é muito maior que o caminho mínimo: 1.826,9 contra 1.806,5. Assim, é sempre importante avaliar o que se precisa mais em uma busca em grafos para que se encontre a solução mais adequada para a situação em questão.