Algoritmos de busca para encontrar um caminho

NATHÁLIA HARUMI KUROMIYA *

*Engenharia da Computação - Graduação
E-mail: n175188@dac.unicamp.br

***Resumo* – O trabalho se concentra em comparar 3 algoritmos de busca: Busca em largura, busca em profundidade e A*. Além disso, no caso do A*, também foi testado o uso de 2 heurísticas diferentes, sendo uma admissível e outra não-admissível. Para esse intuito, o problema foi abstraído em um robô com estado inicial (X, Y) e que com um objetivo de chegar em (Xf, Yf). As ações modeladas se concentram em 8 direções para o robô andar e assim chegar em seu objetivo. Além disso, o plano em que o sujeito se encontra possui paredes que fazem os caminhos entre (X, Y) e (Xf, Yf) não triviais. Os algoritmos utilizados estão implementados na biblioteca AIMA.search e todo o trabalho foi implementado em Python 3. Os resultados mostram que o algoritmo de busca informada utilizado obteve estatísticas melhores do que os outros 2 métodos. Também foi possível ver como uma heurística não-admissível pode atrapalhar a obtenção de um resultado satisfatório.**

***Palavras-chave* – Busca sem informação, A*, comparação de buscas**

I. INTRODUÇÃO

A comparação dos algoritmos e seus funcionamentos tem a função de auxiliar a utilização deles e a escolha em cada contexto através de suas particularidades e vantagens.

Para a elaboração desse trabalho, foi necessário um estudo maior sobre a biblioteca utilizada AIMA.search[1] e todas as implementações utilizadas, principalmente porque a modelagem do problema deve ser feita de acordo com os parâmetros utilizados nas implementações. Para entender essas implementações, também foi necessário a utilização do livro[2] , dado que toda a biblioteca é baseada nos algoritmos e implementações apresentadas nele. Assim, a parte de busca, principalmente de "busca em largura", "busca em profundidade" e "A*" foram revisadas.

II. SEÇÕES

Esse trabalho está separado em tais seções:

- Trabalho Proposto
- Materiais e Métodos
  - Busca em largura
  - Busca em profundidade
  - A*
    - Heurísticas
- Resultado e Discussão
- Conclusões
- Referências

III. TRABALHO PROPOSTO [3]

Um robô encontra-se em estado inicial (X, Y) do mapa descrito na figura 1. As paredes (instranponíveis) estão representadas na imagem pela cor preta. As áreas em branco são livres para que o robô se locomova. O robô é capaz de perceber onde se encontra a cada instante de tempo e cada ação executada o leva para uma posição adjacente, de acordo com a ação executada. O ambiente é determinístico.

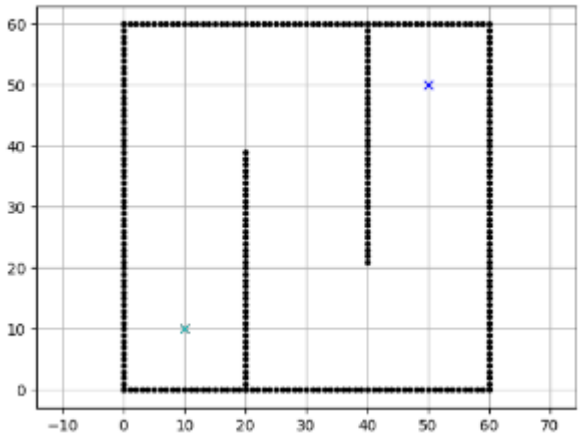

Figura 1. Mapa do ambiente em que o robô irá se locomover. Estão marcados um possível estado inicial e estado objetivo.

Dado todas essas características, a ideia é utilizar 3 algoritmos de busca para encontrar possíveis soluções para esse problema, sendo 2 não-informados e 1 informado com 2 heurísticas distintas. Esses algoritmos, então, serão comparados e mostrados de forma a melhorar a compreensão sobre cada um deles.

IV. MATERIAIS E MÉTODOS

O problema foi modelado através da classe "Problem" da biblioteca "AIMA.search"[1] em Python 3. Essa modelagem propôs que os estados (X, Y) do robô representariam o lugar que o robô se encontra dentro de um campo de 60x60. O ambiente foi modelado de forma fixa, com dimensões 60x60 (como citado acima) e 2 paredes internas colocadas como obstáculos (fig. 1)

Além disso, as ações possíveis para o robô se locomover inclui andar em 8 direções distintas (em ordem): norte, sul, leste, oeste, nordeste, noroeste, sudeste e sudoeste.

Dada toda a modelagem, foi escolhido 3 algoritmos de busca distintos implementados na biblioteca para fazer a comparação.

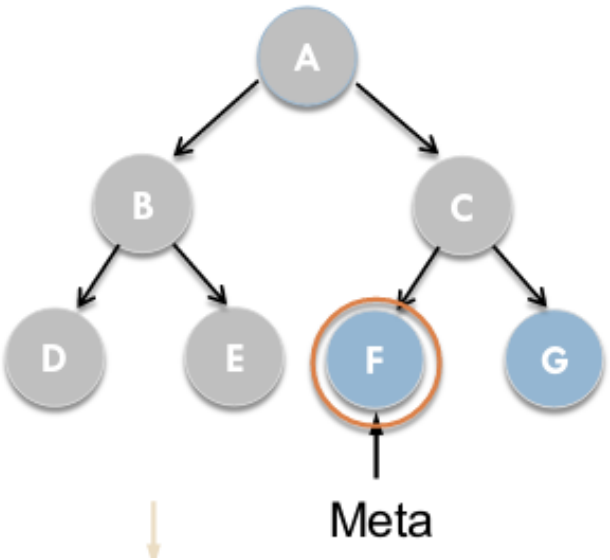

Figura 2. Um exemplo de grafo para ilustrar o funcionamento das buscas.

A. Busca em largura

A busca em largura em um grafo é feita de forma em que visita-se cada nó filho de um nó X e depois os filhos dos irmãos do nó X para só então olhar os nós netos do nó X. Por exemplo, na figura 2, teríamos a seguinte sequência de visita: ABCDEFG. A complexidade de tempo e espaço desse algoritmo se baseia em: $O(b^{d+1})$, onde b é o número médio de sucessores de cada nó e d é a profundidade da solução mais rasa encontrada. Além disso, esse algoritmo necessariamente encontra uma solução, se existir (sendo considerado completo, portanto) e essa solução é ótima, uma vez que ele seleciona a solução mais rasa possível.

B. Busca em profundidade

A busca em profundidade em um grafo se baseia em seguir um caminho até um nó folha. Por exemplo, visita-se o nó X e então o primeiro filho do nó X, Y. Após, é visitado o primeiro filho de Y e assim por diante até encontrar uma folha. Então, sobe-se na hierarquia do grafo para visitar os outros filhos do último nó não-folha encontrado. Seguindo a figura 2, a sequência de visita seria: ABDECFG. A complexidade de tempo desse algoritmo é $O(b^m)$, onde b é o número médio de sucessores de cada nó e m é a profundidade máxima do grafo. Porém, a complexidade de espaço é $O(bm)$, dado que após finalizar a busca em um dos caminhos do grafo, esse caminho pode ser removido da memória. Esse algoritmo não é completo, dado que se um caminho não tiver fim, a busca não termina. Também não acha a solução ótima.

C. A*

A busca A* em um grafo se baseia em seguir um caminho que engloba: o menor custo até o estado em questão e o valor associado de uma heurística naquele determinado estado. Essa heurística visa dar informações sobre a qualidade de um estado e, neste problema, ela indica o quão distante o estado atual está em relação ao estado objetivo. A complexidade de tempo e espaço, no pior caso, é $O(b^d)$, onde b é o número médio de sucessores e d é a profundidade do grafo. Caso a heurística seja admissível, esse algoritmo é completo e ótimo.

1) Heurísticas: As heurísticas utilizadas nesse problema foram:

- Distância em linha reta:
  $d = ((Xa - Xb)^2 + (Ya - Yb)^2)^{1/2}$
- 100x a distância Manhattan
  $d = 10 * (abs(Xa - Xb) + abs(Ya - Yb))$

A heurística de distância em linha reta é uma heurística admissível e, portanto, garante uma solução ótima do problema.

Já a heurística de distância Manhattan é não-admissível (pois superestima a distância entre o estado atual e o estado objetivo) e não garante uma solução ótima. Porém, caso encontre a solução, a execução do algoritmo provavelmente gerará menos estados e encontra uma solução em menos tempo.

O 100 vezes essa distância funciona apenas para evidenciar o comportamento de superestimação e obter resultados mais significantes para comparação.

V. Resultados e Discussão

Para a análise dos algoritmos, foram escolhidas 6 entradas (posição inicial do robô e estado objetivo) distintas e computadas estatísticas acerca das execuções, como mostra a tabela I, onde S é o número de sucessores, T é o número de testes de objetivo, E é o número de estados gerados e C é o custo do caminho encontrado.

Para melhor ilustrar esses dados, as figuras 3 e 4 mostram uma média dos dados da tabela.

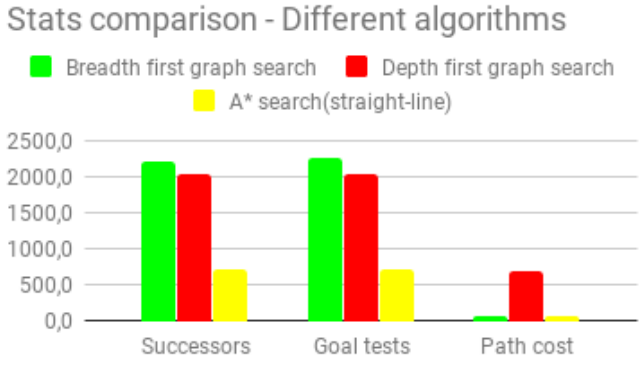

Figura 3. Gráfico da média das estatísticas obtidas para os 3 algoritmos testados.

Com esses dados, podemos perceber uma performance significativamente melhor do algoritmo informado, A*. Isso

Tabela I
ESTATÍSTICAS OBTIDAS POR ALGORITMO.

Algoritmo	S	T	E	C
	Robô: inicial (10, 10), final (50, 50)			
Busca em largura	3116	3135	23891	96,6
Busca em profundidade	1288	1289	9653	1419,9
A*	2082	2083	15985	96,6
	Robô: inicial (10, 10), final (50, 50)			
Busca em largura	3087	3107	23668	96,6
Busca em profundidade	1305	1306	9888	300,4
A*	2024	2025	15527	96,6
	Robô: inicial (10, 10), final (50, 50)			
Busca em largura	2024	2102	15616	56,6
Busca em profundidade	2093	2094	15896	819,8
A*	40	41	314	56,6
	Robô: inicial (10, 10), final (50, 50)			
Busca em largura	2005	2081	15464	56,6
Busca em profundidade	2866	2867	21955	262,2
A*	40	41	314	56,6
	Robô: inicial (10, 10), final (50, 50)			
Busca em largura	1058	1113	8123	40
Busca em profundidade	2845	2846	21760	301,6
A*	41	42	320	40
	Robô: inicial (10, 10), final (50, 50)			
Busca em largura	1942	2015	14969	40
Busca em profundidade	1761	1762	13261	1033,1
A*	41	42	320	40

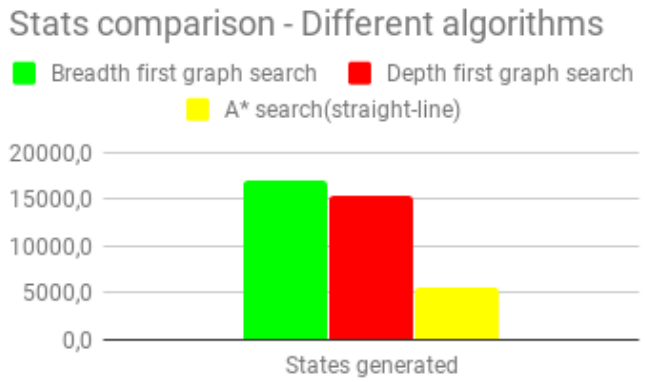

Figura 4. Gráfico da média dos estados gerados para cada um dos 3 algoritmos.

ocorre pois a heurística provê uma informação que avalia a qualidade dos estados e permite com que o algoritmo faça escolhas melhores de quais caminhos avaliar. Assim, menos estados são gerados e é possível encontrar um caminho ótimo.

Também é preciso salientar que os caminhos obtidos dependem da ordem em que as ações foram modeladas no código. Como citado anteriormente, as ações estão projetadas na ordem: norte, sul, leste, oeste, nordeste, noroeste, sudeste, sudoeste. A relação ordem das ações e caminhos fica muito evidenciada na figura 6, em que o algoritmo optou por um caminho não trivial e não eficiente. Dado que a estrutura de dados utilizada nessa busca é uma pilha, podemos ver que as ações executadas estão exatamente na ordem contrária de como foram projetadas, como se fossem colocadas na pilha. A busca em profundidade da figura 6 não possui compromisso com caminho ótimo e retorna, assim, o primeiro caminho encontrado.

As figuras 5 e 7 mostram as buscas em largura e A*, respectivamente. Pode-se observar que ambos mostram caminhos ótimos, como os algoritmos se propõem a fazer.

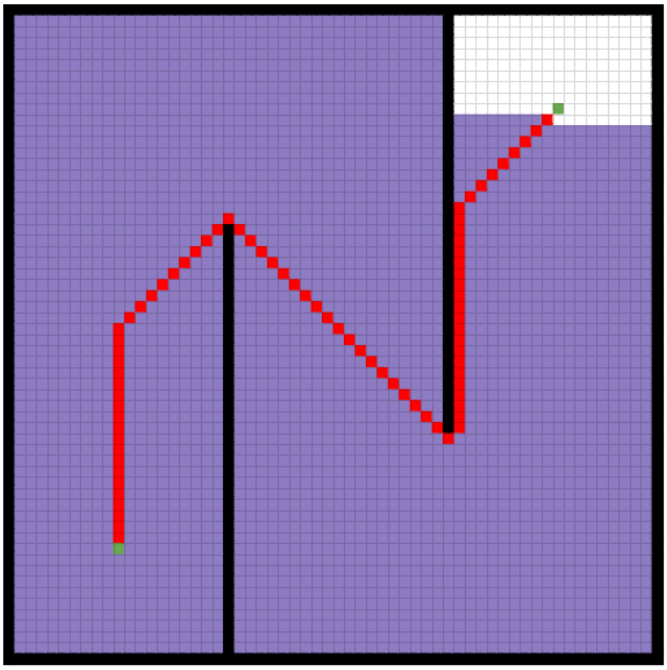

Figura 5. Mapa obtido com a busca em largura para a entrada Robô: inicial (10, 10), final (50, 50). O vermelho representa o caminho encontrado, o lilás mostra os estados gerados, o branco mostra estados não gerados e o preto mostra paredes intransponíveis.

Por outro lado, é possível perceber que o sucesso desse algoritmo está ligado a heurística escolhida e, por isso, foram testadas 2 distintas. A distância em linha reta é uma heurística que mostra a distância mínima entre um estado e outro, dado que uma linha reta é, matematicamente, a menor distância entre dois pontos. Como se trata de uma heurística admissível, ela foi a considerada para todos os dados explicitados anteriormente. A segunda heurística testada foi 100 vezes a distância Manhattan. A distância Manhattan em si é uma heurística não admissível para esse problema, dado que as ações foram modeladas com direções diagonais. Dessa forma, essa heurística pode dar um valor de caminho maior do que o caminho ótimo para partes do caminho ótimo. Isso faz com que, em alguns casos, ela não encontre o caminho ótimo.

A figura 8 mostra uma comparação entre o algoritmo A* com as duas heurísticas. A priori, pode-se erroneamente achar que a heurística não-admissível é a melhor escolha para o problema, dado que suas estatísticas se mostram melhores. Esse comportamento é esperado, pois o fato da heurística

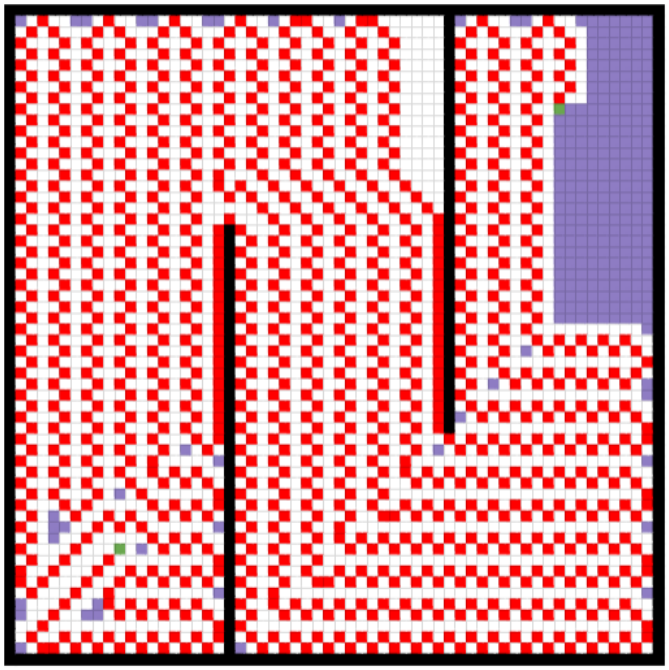

Figura 6. Mapa obtido com a busca em profundidade para a entrada Robô: inicial (10, 10), final (50, 50). O vermelho representa o caminho encontrado, o lilás mostra os estados gerados, o branco mostra estados não gerados e o preto mostra paredes intransponíveis.

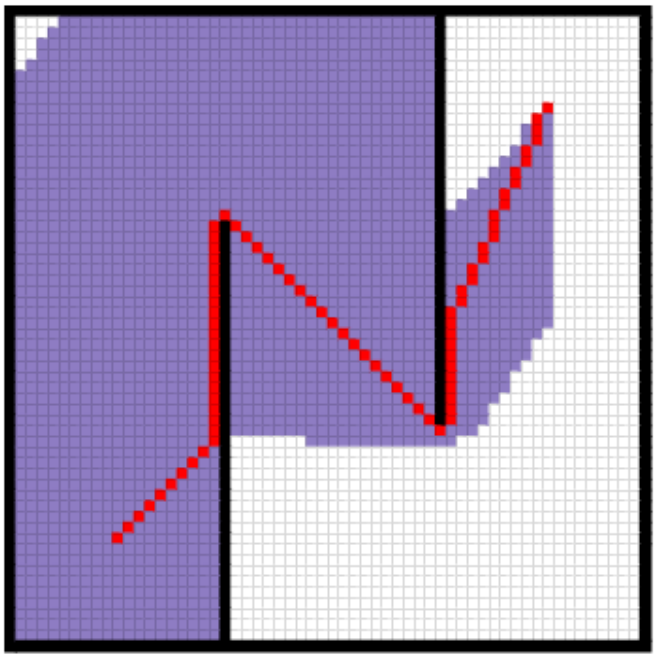

Figura 7. Mapa obtido com a busca A* para a entrada Robô: inicial (10, 10), final (50, 50). O vermelho representa o caminho encontrado, o lilás mostra os estados gerados, o branco mostra estados não gerados e o preto mostra paredes intransponíveis.

superestimar o custo do caminho faz com que ele gere menos estados e, portanto, encontre um resultado mais rapidamente (caso encontre). Por outro lado, através da tabela Y podemos perceber um comportamento que evidencia a heurística não admissível: há estados ”pai”que são computados com um custo maior do que estados filhos, o que não faz sentido.

Tabela II
ESTADOS E CUSTOS

	E(50,49)		E(50,50) - Objetivo	
Heurísticas	h()	Custo	h()	Custo
Linha reta	1	c + 1	0	c + 1
100*Manhattan	100	c + 100	0	c + 1

VI. CONCLUSÕES

A comparação dos métodos de buscas evidenciou que buscas informadas possuem maior eficiência para encontrar a solução esperada. Nesse caminho, o uso da biblioteca do AIMA foi imprescindível para obter os resultados e fazer as análises propostas.

Por outro lado, em relação às heurísticas, a ideia era encontrar um caso em que a heurística não-admissível não encontrasse a solução ótima do problema para ressaltar os problemas relacionados a tal heurística. Não sendo possível encontrar um caso de falha no problema proposto, a análise ficou limitada a inconsistência dos custos gerados pela heurística.

+————-+

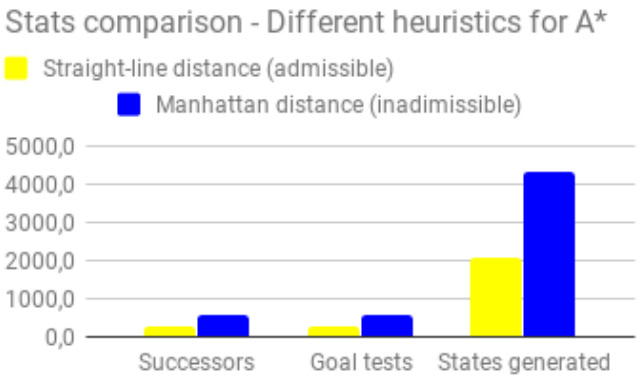

Figura 8. Um exemplo de figura.

REFERÊNCIAS

[1] “Aima-python: search.py.” [Online]. Available: https://github.com/aimacode/aima-python/blob/master/search.py 1

[2] P. Norvig and S. Russell, *Artificial Intelligence: A Modern Approach*, 3rd ed. Elsevier, 2013. 1

[3] E. L. Colombini, *Projeto 1*. [Online]. Available: https://www.ic.unicamp.br/~esther/teaching/2019s1/mc906/P1.pdf 1