Relatório de Projeto

Problemas de mapas

Análise de algoritmos de busca

Universidade Federal de Juiz de Fora Departamento de Ciência da Computação
DCC014 - Inteligência Artificial, 2020.1

Wagno Leão Sergio - 201865555C
Davi Magalhães Pereira - 201865558C

Professor: Saulo Moraes Villela

1 Definição do problema

O problema de mapa consiste em encontrar o menor caminho entre dois pontos de um mapa que é representado por um grafo. Levando em consideração que as rotas que ligam um ponto ao outro possuem custos associados que influenciarão nas escolhas dos algoritmos informados.

2 Implementação

- O código-fonte do projeto foi escrito em python 3;
- Foi criado um gerador de instâncias (mapas) para realizar os experimentos, o mesmo retorna uma lista de tuplas contendo os ID's dos pontos conectados e o custo da rota, como por exemplo ("A", "B", 25.43). Durante a implementação tomou-se o cuidado de tentar gerar mapas consistentes e coerentes, como por exemplo a priorização de conexão entre pontos mais próximos além de especificar a quantidade máxima de vizinhos que cada nó pode ter. Foi adicionado também um ruído ao custo das arestas para garantir a consistência da heurística selecionada;
- Foi implementada uma classe Graph() baseada em lista de adjacência onde a expressão Graph[node] retorna uma lista dos vizinhos de 'node' e a expressão Graph[node1][node2] retorna peso associado a aresta 'node1'/'node2'. A classe tem como principal atributo um dicionário, onde os campos são acessados por NamedTuples que fazem a representação de pontos no mapa. As arestas também são representadas por NamedTuples;

- A classe Graph possui métodos para inserção de nós, inserção de arestas e de impressão do grafo no terminal, além de métodos adicionais para operações com a classe.
- Todo o projeto está hospedado em um repositório do GitHub.

Além disso foram criadas funções utilitárias para cálculo de heurística, formatação, registro das métricas executadas e criação de gráficos dos mapas gerados. A distância euclidiana entre os pontos foi escolhida como heurística.Foram utilizadas como métricas:

- Custo e profundidade da solução;
- Número de nós expandidos e visitados;
- Valor médio do fator de ramificação da árvore;
- Tempo de execução do algoritmo.

3 Divisão de tarefas

Os dois participantes implementaram a estrutura principal do grafo e as funções utilitárias. Em relação aos algoritmos de busca a divisão de tarefas foi a seguinte:

Davi

- Busca em largura (BFS)
- Busca em profundidade (DFS)
- Guloso (Greddy)
- A*

Wagno

- Backtracking
- Busca Ordenada (UCS)
- IDA*

4 Execuções

Executamos os algoritmos implementados para mapas gerados com 25, 50, 100 e 200 nós. O número de arestas foi regulado de acordo com o tamanho do mapa. Tomou-se o cuidado de sempre selecionar o par de nós mais distantes entre si do mapa. Cada algoritmo é executado 10 vezes para cada tamanho de mapa, gerando instâncias diferentes.

O computador utilizado nos experimentos possui um processador Ryzen 5 2400G de 3.9Ghz e com 8GB de memória RAM.

4.1 Tabela de dados

Para a geração dos dados foi levado em consideração apenas as execuções em que houve sucesso na busca da solução. A tabela gerada possui todas as variáveis mencionadas anteriormente e cada linha representa uma execução.

A tabela abaixo é um exemplo dos dados gerados.

	algorithm	solution	depth	cost	expanded_nodes	visited_nodes	average_branching_factor	execution_time	result	n
0	Backtracking	NewAnnette->'EastChristian'->'Bensonport'->...	7	289.28	7	8	1.0	6.45110012555960 6e-05	SUCCESS	25
1	BFS	NewAnnette->'Karenton'->'Cortezfurt'->...	3	175.07999999999996	15	22	4.533333333333333	8.162399899447337e-05	success	25
2	DFS	NewAnnette->'EastChristian'->'Karenton'->...	8	237.158	8	9	4.625	6.154500078991987e-05	success	25
3	UCS	NewAnnette->'Karenton'->'Cortezfurt'->...	7	166.813	109	25	4.32	0.0011675210007524586	SUCCESS	25
4	Greedy	NewAnnette->'Karenton'->'Cortezfurt'->...	3	175.07999999999996	3	4	5.333333333333333	9.51490001170896e-05	success	25

A tabela abaixo foi criada agrupando as execuções por algoritmo e calculando a média. Observando-a é perceptível que a média do tempo de execução do algoritmo IDA* é de 10 a 20 vezes maior que em relação aos outros algoritmos. Além disso a média de nós expandidos do IDA* distorce completamente a métrica, sendo em média 46 vezes maior do que os outros algoritmos. Isso é acontece pois são considerados os nós expandidos de todas as sub-árvores que ele gera para encontrar a solução.

O algoritmo de busca ordenada segue logo atrás, sendo cerca de 6 vezes maior do que os demais algoritmos, apesar de ter o menor valor médio de custo.

	algorithm	depth	cost	expanded_nodes	visited_nodes	average_branching_factor	execution_time
0	Astar	10.4	3071.9285	64.55	80.85	1.6906044568735574	0.0015237809749578447
1	BFS	5.9	3962.4754000000003	71.8	86.2	1.2314202894212223	0.0011541888000465405
2	Backtracking	37.125	13718.621949999997	50.475	51.475	4.09395832562936	0.0008982729750414365
3	DFS	41.475	15366.847100000003	86.625	62.175	2.545856824719528	0.001674916799947823
4	Greedy	9.825	3895.7099500000004	11.775	11.95	4.612746334617091	0.0003553083500037246
5	IDAstar	11.775	4311.506449999999	3326.35	6938.3	2.013095314933377	0.07765118637505566
6	UCS	10.4	3071.9285	594.4	93.325	5.128282145433931	0.00815758697497131

Na tabela seguinte os dados foram agrupados por algoritmo e número de nós e calculada a média. Ao analisar percebemos que, em um número baixo de nós, os algoritmos não se diferem drasticamente, situação que muda ao aumentarmos o número de nós.

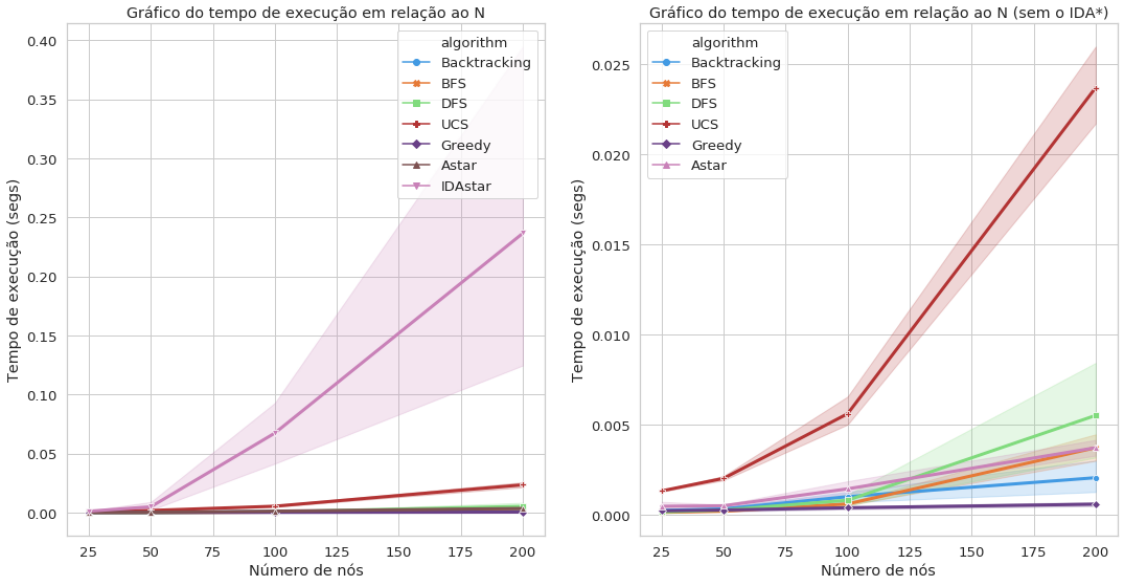
	algorithm	n	depth	cost	expanded_nodes	visited_nodes	average_branching_factor	execution_time
0	Astar	25	6.1	186.95550000000003	19.1	24.3	1.5722501033799658	0.0004618126999957894 4
1	Astar	50	8.4	2344.8496000000005	31.8	39.7	1.4611515225136913	0.0004925526000079117
2	Astar	100	11.1	4336.536100000001	79.3	90.6	1.5114298949740208	0.00142871779980851
3	Astar	200	16.0	5419.372799999999	128.0	168.8	2.2175863066265524	0.0037120408000191675
4	BFS	25	4.2	229.28580000000002	20.5	24.4	1.147921747715798	0.0001230947998919873 6
5	BFS	50	5.8	2589.3487	33.6	43.7	1.3189080291154796	0.0001892129999760072 6
6	BFS	100	6.9	5158.386799999999	75.4	89.8	1.2423700078316844	0.0005909178001274995
7	BFS	200	6.7	7872.8803	157.7	186.9	1.216481373021927	0.003713529600190668
8	Backtracking	25	10.6	376.3465	17.4	18.4	3.4085876563050475	0.0002332624000700889
9	Backtracking	50	16.3	4103.8153	27.5	28.5	2.9302589854640937	0.0003160958000989922
10	Backtracking	100	39.5	13889.631099999999	62.5	63.5	3.5754162025567915	0.000995297000190476
11	Backtracking	200	82.1	36504.694899999995	94.5	95.5	6.4615704581915026	0.0020484366998061885
12	DFS	25	12.3	408.30280000000005	20.5	18.3	2.340688009484699	0.0001455338000596384 6
13	DFS	50	17.1	4202.7992	40.5	33.9	1.6766432496075354	0.0002525927998249244 4
14	DFS	100	38.6	13314.927599999999	75.9	60.6	2.2789737892441506	0.000796179599910829
15	DFS	200	97.9	43541.3588	209.6	135.9	3.8871222505417267	0.005505360999995901
16	Greedy	25	7.6	269.96860000000004	7.9	8.8	3.4168398268398272	0.0002108604000568448
17	Greedy	50	7.4	2559.2086000000004	12.9	11.5	3.304806824000373	0.0002534211001147923 6
18	Greedy	100	12.0	5665.996799999999	14.0	14.2	3.8457456140350876	0.0003792688999055826 6
19	Greedy	200	12.3	7087.665800000001	12.3	13.3	7.883593073593073	0.0005776829999376787
20	IDAstar	25	5.1	202.2238	78.2	169.2	2.257307776147596	0.001353174199903151
21	IDAstar	50	8.7	2625.4957	398.1	593.8	1.597096616149373	0.0050912288001200064
22	IDAstar	100	16.5	6087.780799999999	4575.7	7110.4	1.6109621175000215	0.06749433320001116
23	IDAstar	200	16.8	8330.5255	8253.4	19879.8	2.58701474993652	0.2366660093001883
24	UCS	25	6.1	186.9555	107.9	24.9	4.333833333333334	0.0013159271999938938
25	UCS	50	8.4	2344.8496	170.1	49.1	3.46588110698825	0.002012795100108633
26	UCS	100	11.1	4336.5361	436.8	99.3	4.399414141414142	0.0056005626997830404
27	UCS	200	16.0	5419.3728	1662.8	200.0	8.314000000000002	0.023701062899999967

5 Avaliação dos algoritmos

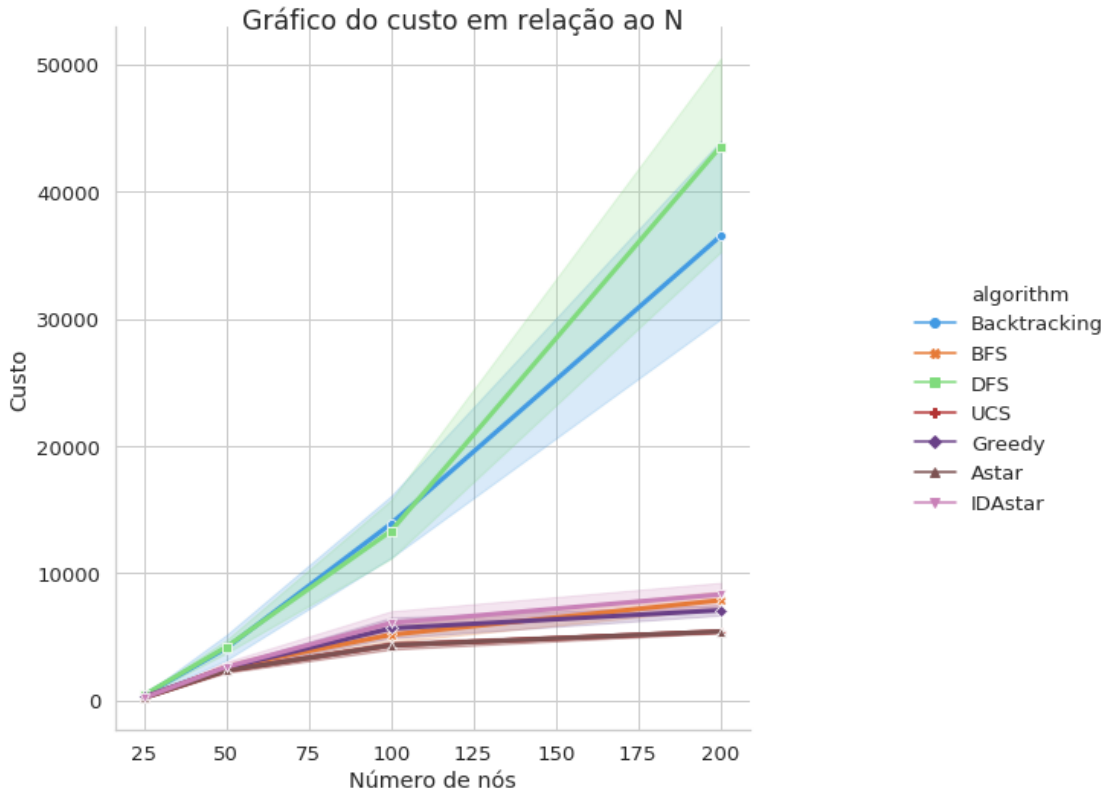
Para comparar o desempenho dos algoritmos criamos gráficos com os dados obtidos e calculamos o intervalo de confiança para as métricas de cada algoritmo com 95% de confiança.

O gráfico abaixo representa o tempo de execução em segundos em relação ao número de nós do mapa. Como já observamos, o algoritmo IDA* possui a maior no tempo de execução, tendo um comportamento exponencial. Como o IDA* distorce a escala do primeiro gráfico, foi criado um segundo sem ele para melhor visualização.

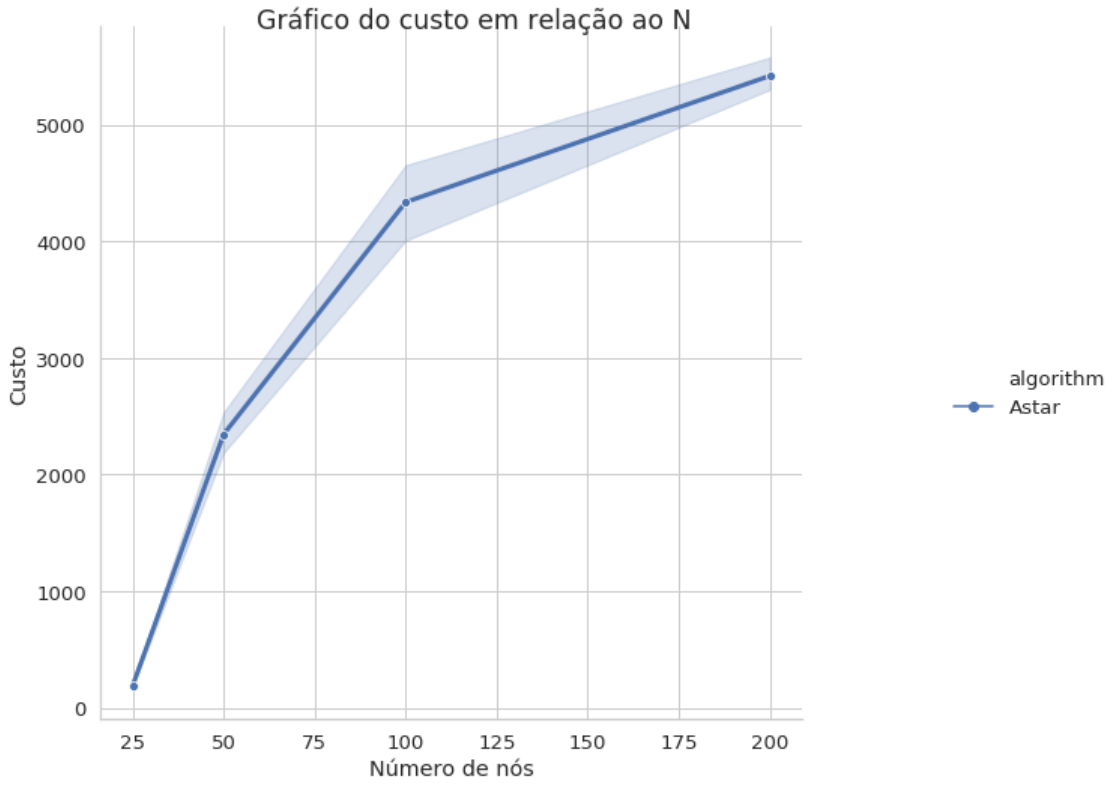
No segundo gráfico o algoritmo com maior tempo de execução é o de busca ordenada, sendo seguido logo abaixo pelo algoritmo de busca em profundidade. Os demais algoritmos tiveram um tempo de execução menor que 0.005 segundos (5 milisegundos). É fácil perceber também que o algoritmo guloso obteve o menor tempo de execução entre todos os outros.



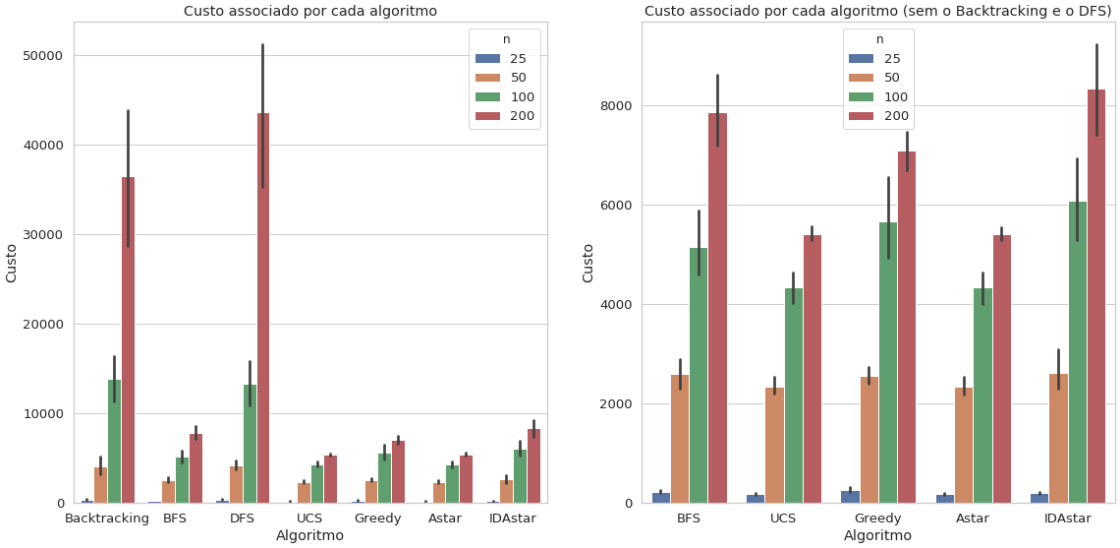
O gráfico seguinte mostra o custo médio de cada algoritmo em relação ao número de nós. Podemos perceber que tanto o Backtracking quanto a busca em profundidade apresentam comportamento exponinencial enquanto os demais algoritmos apresentam um comportamento logarítmico.



Como o algoritmo A* possui um custo médio semelhante ao da busca ordenada, onde o mesmo é escondido no gráfico. Por esse motivo foi criado um gráfico apenas para o A*, mostrando seu desempenho.



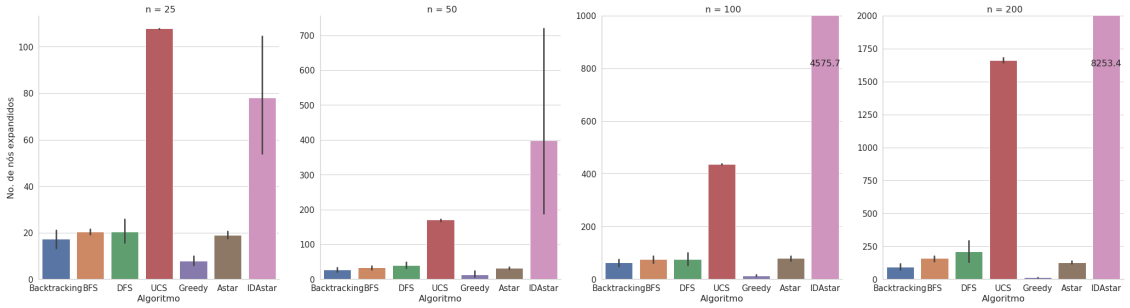
Os gráficos abaixo representam as mesmas informações que os gráficos anteriores e foram criados para melhor visualização.



Agora nos gráficos seguintes temos a média de nós expandidos por cada algortimo. Foi

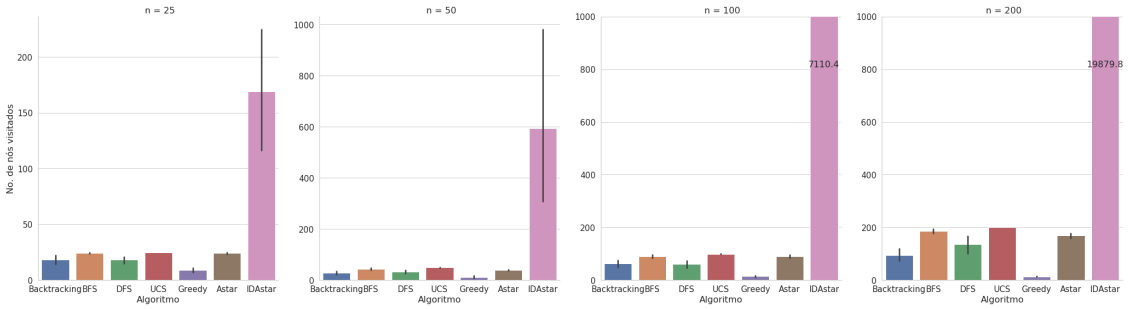
criado um gráfico para cada tamanho de mapa para melhor visualização. Como mencionado antes, o número de nós expandidos pelo IDA* é muito maior pois são contados os nós expandidos de todas as sub-árvores geradas na sua execução. A quantidade de nós expandidos por ele é tão grande que o gráfico é distorcido, por isso nós limitamos o tamanho e escrevemos o valor da sua média nos dois últimos gráficos.

O algoritmo de busca ordenada também expande mais nós por que ele tem a característica de só terminar a execução quando encontrar a solução com menor custo. Através desses gráficos vemos também que o algoritmo guloso teve o menor número de nós expandidos em todos os casos.



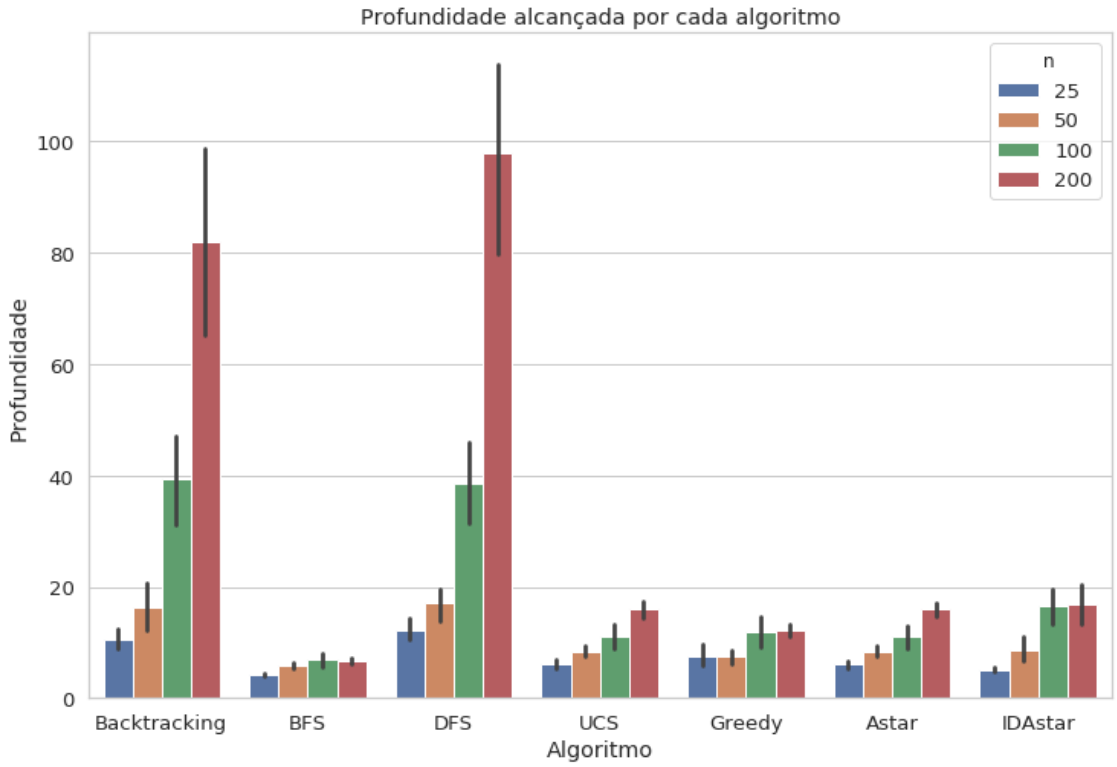
De forma similar ao gráfico anterior nós temos abaixo os gráficos do número de nós visitados por cada algoritmo, separados por número de nós. Podemos perceber que o padrão se repete como nos nós expandidos para o IDA*, já que também consideramos o número de nós visitados de todas as sub-árvores geradas por ele. É interessante notar que o comportamento da busca ordenada em relação aos nós visitados não se assemelha com os dos gráficos anteriores.

O algoritmo guloso também obteve o menor número de nós visitados em todos os casos.



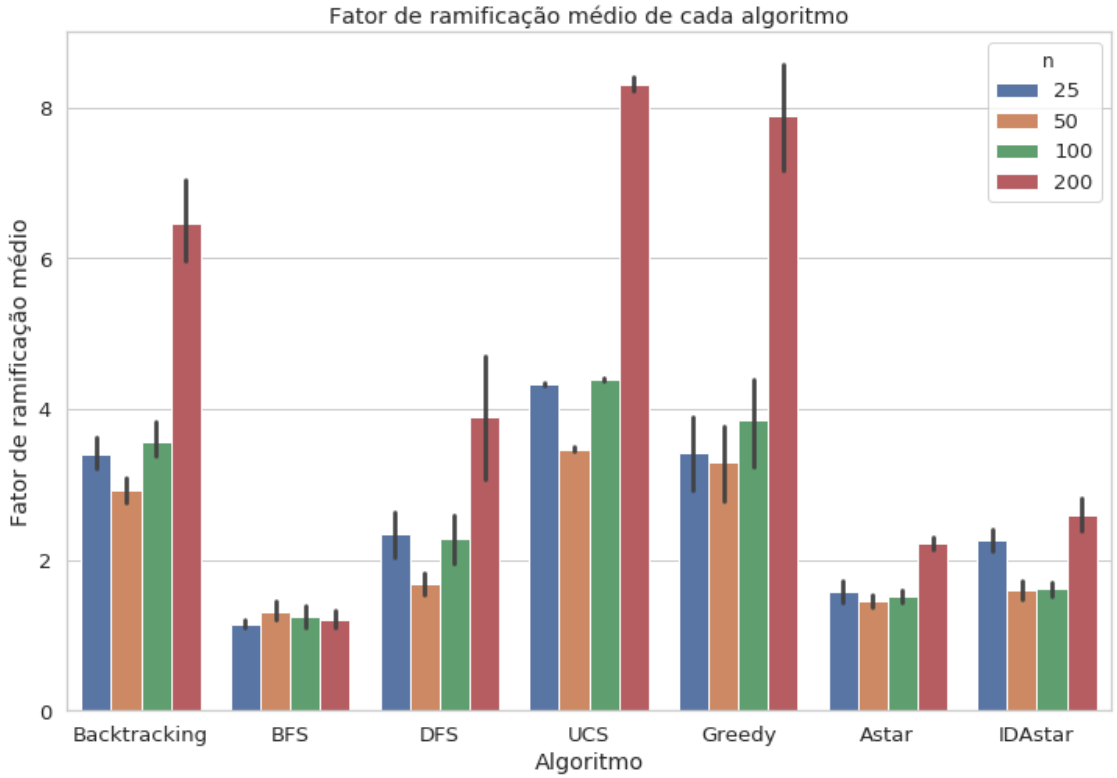
No gráfico seguinte temos a média da profundidade alcançada por cada algoritmo. De forma previsível notamos que o algoritmo de Busca em Profundidade possui a maior média.

É interessante notar que a segunda maior média está associada ao Backtracking. Isto se deve provavelmente ao seu comportamento de tentativa e erro na busca pelo nó de destino. Por causa dos intervalos de confiança do Backtracking e da Busca em Profundidade estarem englobando as médias um do outro nos casos com tamanhos de mapas de 50, 100 e 200 nós, podemos afirmar com 95% de confiança que eles são semelhantes nesses casos.



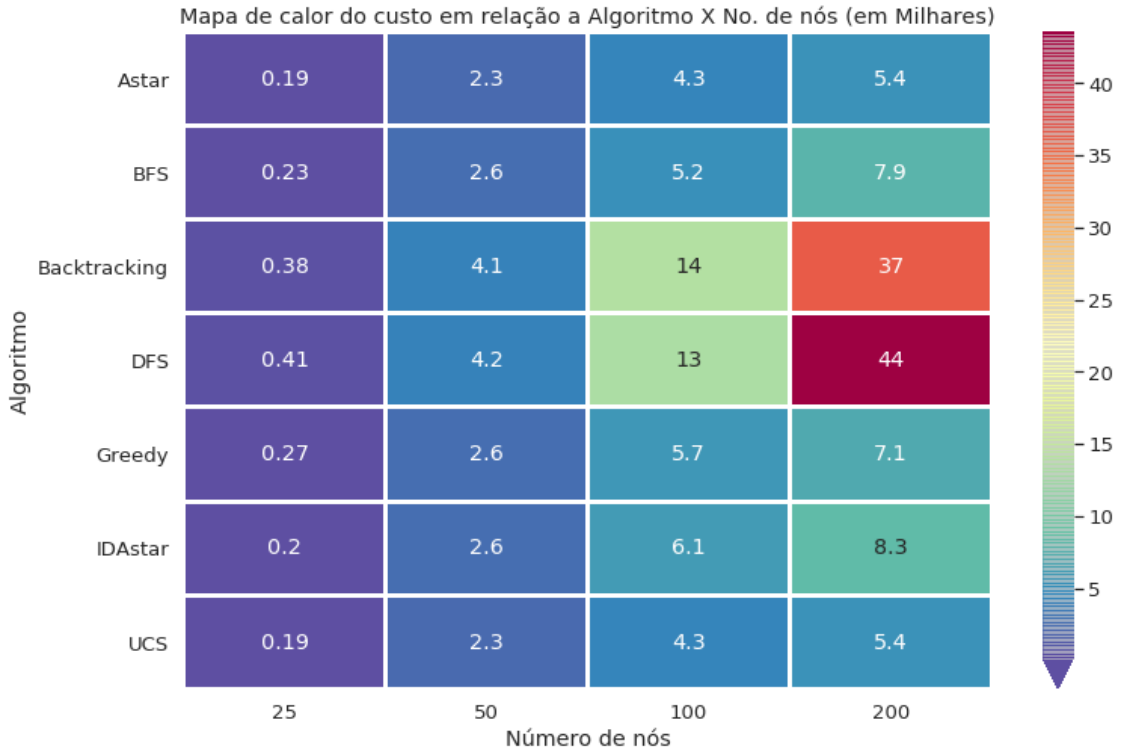
Agora abaixo temos o gráfico do fator de ramificação médio de cada algoritmo agrupados por número de nós no mapa. Podemos interpretar o fator ramificação médio como sendo a média do número de filhos que cada nó da árvore de busca de um algoritmo tem.

Podemos observar que, em tamanhos de mapa com 25 nós, o Backtracking possui uma ramificação maior. Conforme o número de nós aumenta, a busca ordenada tomo lugar tendo o maior fator de ramificação médio. É fácil observar também que o algortimo com menor fator de ramificação médio é a Busca em Largura.



Como forma de melhor visualização do dados obtidos, foi feito um mapa de calor dos custos de cada algoritmo em relação ao tamanho do mapa gerado. Como já foi observado, os algoritmos que possuem o maior custo médio são a busca em profundidade e o Backtracking. Podemos ver aqui também que a maioria do algortimos tem aumento de custo suave em relação ao tamanho do mapa, com exceção do dois algoritmos mencionados anteriormente.

Neste mapa de calor também podemos ver que a média de custo do A* e da busca ordenada são exatamente iguais. Isto provavelemente acontece pois, quando uma heurística consistente é aplicada, o algoritmo A* sempre vai encontrar a solução ótima, que no caso é a solução da busca ordenada. O nosso gerador de mapas foi implementado de forma que a consistência da heurística sempre seja garantida, gerando então o resultado mostrado abaixo.



6 Conclusão

Podemos analisar através dessas comparações que cada algoritmo possui as suas próprias caracterísicas e, consequentemente, seus próprios desempenhos. Fica a critério do programador escolher o algortimo que melhor se encaixa nos requisitos do seu problema. Vale a pena ressaltar que as condições criadas para avaliação dos algoritmos foram geradas artificialmente, fazendo com que os resultados observados neste relatório não reflitam em condições mais rigídas e/ou específicas.

É também importante notar que o algoritmo A* obteve resultados satisfatórios em todas as métricas avaliadas, de modo que é perceptível o fato dele ser o principal algoritmo escolhido para descoberta de caminhos entre pontos.