UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

ALGORITMO A ESTRELA

Acadêmicos:	**RA:**
Vanessa Yukari Kajihara	**78605**
Vinícius Menossi	**108804**

Disciplina: Modelagem e Otimização de Algoritmos

Professor: Ademir Aparecido Constantino

Maringá, 12 de abril de 2021.

INTRODUÇÃO

Um algoritmo de busca é um procedimento de exploração do espaço de estados, em que a transição entre estados é definida por um operador definido como gerador de sucessores. Existem diferentes categorias de busca que visam solucionar problemas específicos.

A busca não-informada, ou busca cega, utiliza somente da definição do problema a ser solucionado. Nesta categoria, podem-se citar os algoritmos de busca em largura e busca em profundidade. Já a busca informada, ou busca heurística, utiliza informações adicionais, ou heurísticas, que são específicas para cada problema, definindo uma ordem de preferência para a expansão dos nós.

Um dos algoritmos de busca heurística de caminho mínimo mais conhecido é o chamado A* (ou A estrela), descrito originalmente por Hart, Nilsson e Raphael, em 1968. Ele avalia os nós por meio de custos de menor caminho assim definidos:

$$f(n) = g(n) + h'(n)$$

em que:

f(n) é o custo do caminho do estado final até o estado inicial passando pelo estado *n*;

g(n) é o custo do caminho mínimo entre o estado inicial e o estado *n*;

h'(n) é uma estimativa do custo do menor caminho entre o estado *n* e o estado final, dada por uma heurística.

Se *h'(n)* é uma função admissível, então existe prova de que o algoritmo A* sempre encontra o caminho ótimo.

Um dos problemas em que esse algoritmo pode ser aplicado é na computação do número mínimo de jogadas para a solução do jogo do tabuleiro de 15 peças. Ele consiste em um tabuleiro de dimensões 4x4, com 15 peças e 1 espaço vazio, que possibilita a movimentação das peças, e que tem como objetivo montar o tabuleiro em alguma configuração final.

Neste trabalho foi implementada uma versão do algoritmo A* apresentada em aula, utilizando a linguagem C++. Dada uma configuração inicial do tabuleiro, o programa retorna o número mínimo de jogadas para que se chegue na seguinte configuração final:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	0

Para tanto, foram implementadas 5 heurísticas admissíveis as quais foram comparadas utilizando 10 casos de teste.

HEURÍSTICAS

Neste trabalho foram implementadas e testadas 5 heurísticas admissíveis, sendo duas delas combinação das outras três. O resultado retornado por cada uma delas está descrito a seguir:

Heurística 1: número de peças fora de seu lugar na configuração final.

Heurística 2: número de peças fora de ordem numérica segundo o tabuleiro final.

Heurística 3: para cada peça, soma-se a quantidade de deslocamentos para colocá-la no seu lugar, considerando que o caminho esteja livre.

Heurística 4: é uma combinação do tipo $h'(n) = p_1 h'_1(n) + p_2 h'_2(n) + p_3 h'_3(n)$, tal que $p_1 + p_2 + p_3 = 1$.

Heurística 5: é uma combinação do tipo $h'(n) = max(h'_1(n), h'_2(n), h'_3(n))$.

Para a implementação da heurística 4 foram testadas todas as combinações de p_1, p_2 e p_3 variando de 0 a 1 em intervalos de 0,01, tal que sua soma fosse igual a 1. Foram utilizados para esse teste as entradas 1, 2, 3, 4, 8, 9 e 10, mostrados no tópico "**CASOS DE TESTE**".

Esta análise mostrou que os melhores valores dos pesos, tanto em termos de tempo quanto de memória, foram $p_1 = 0$, $p_2 = 0$ e $p_3 = 1$.

CASOS DE TESTE

Dez diferentes entradas foram utilizadas para o teste do algoritmo para análise das heurísticas. São elas:

1. 0 2 9 13 3 1 5 14 4 7 6 10 8 11 12 15
2. 3 2 1 9 0 5 6 13 4 7 10 14 8 12 15 11
3. 2 1 9 13 3 5 10 14 4 6 11 15 7 8 12 0
4. 9 13 10 0 5 2 6 14 1 7 11 15 3 4 8 12
5. 4 3 2 1 8 10 11 5 12 6 0 9 15 7 14 13
6. 9 13 14 15 5 6 10 8 0 1 11 12 7 2 3 4
7. 10 6 2 1 7 13 9 5 0 15 14 12 11 3 4 8
8. 6 2 1 5 4 10 13 9 0 8 3 7 12 15 11 14
9. 10 13 15 0 5 9 14 11 1 2 6 7 3 4 8 12
10. 5 9 13 14 1 6 7 10 11 15 12 0 8 2 3 4

HARDWARE UTILIZADO

Para a realização dos testes do relatório foi utilizado uma instância de máquina virtual através da Google Cloud Platform com as seguintes configurações:

PROCESSADOR: Intel(R) Xeon(R) CPU @ 2.30GHz

MEMÓRIA: 128GB

DISCO: 10GB

SO: Ubuntu 16.04.7 LTS

ESTRUTURAS DE DADOS

Considerando o algoritmo apresentado em aula, para a representação do conjunto dos estados abertos foram utilizados duas estruturas em conjunto: um heap, a fim de obter o elemento de custo mínimo, e um hash para a maior eficiência nas buscas e o qual é responsável por guardar os elementos válidos do heap. Deste modo, melhora-se o tempo de execução da implementação em detrimento da memória.

Já o conjunto dos estados fechados foi representado por um hash, pois uma vez que todas as heurísticas testadas são admissíveis, as únicas operações realizadas sobre este conjunto são a busca e inserção.

EXECUÇÃO DOS CASOS DE TESTE

A Tabela 1 apresenta os resultados obtidos para cada caso de teste utilizando qualquer uma das heurísticas testadas.

Tabela 1 - Quantidade mínima de movimentos para cada caso de teste.

Caso de teste	**Resultado**
1	18
2	19
3	12
4	21
5	38
6	32
7	38
8	32
9	27
10	29

Os resultados de tempo de execução de cada caso, bem como os tamanhos de cada conjunto em cada teste executado, foram colocados em gráficos dispostos nas Figuras 1 a 4 a seguir. Uma vez que a ordem de grandeza dos resultados obtidos são bastante discrepantes entre si, todos os eixos verticais foram construídos em escala logarítmica para uma melhor visualização. Além disso, os resultados de tempo, embora mensurados em segundos, foram colocados no gráfico em microssegundos a fim de evitar valores negativos.

Os valores numéricos detalhados de todos os casos de teste dos resultados obtidos estão apresentados nas tabelas do Apêndice A.

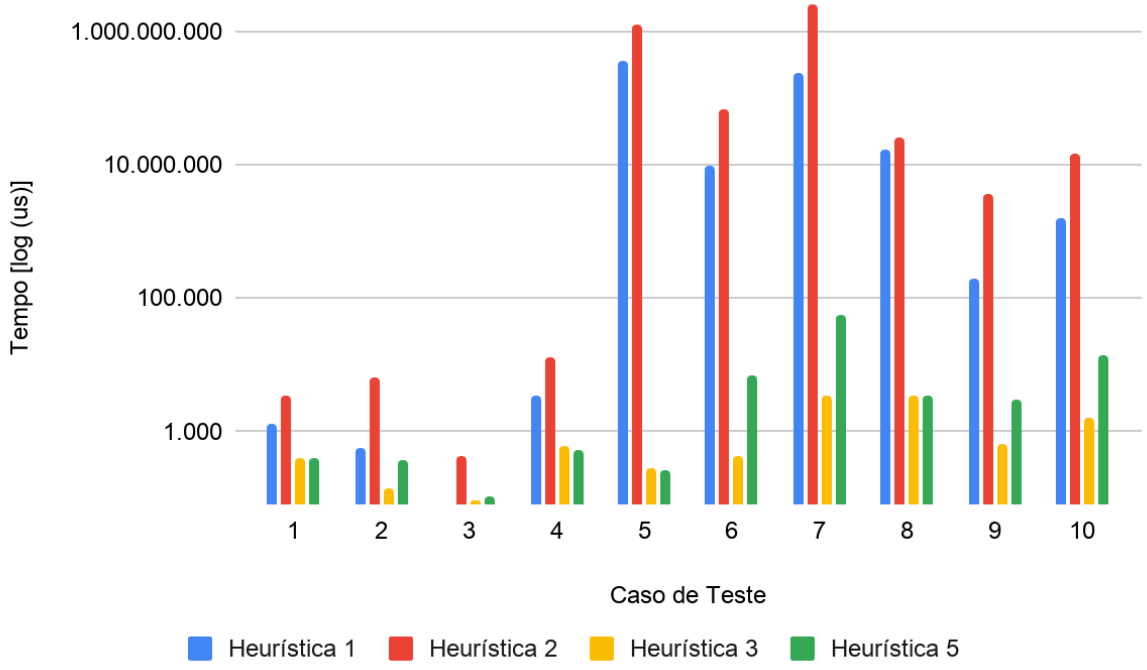

Figura 1 - Gráfico comparativo entre os tempos de execução dos testes.

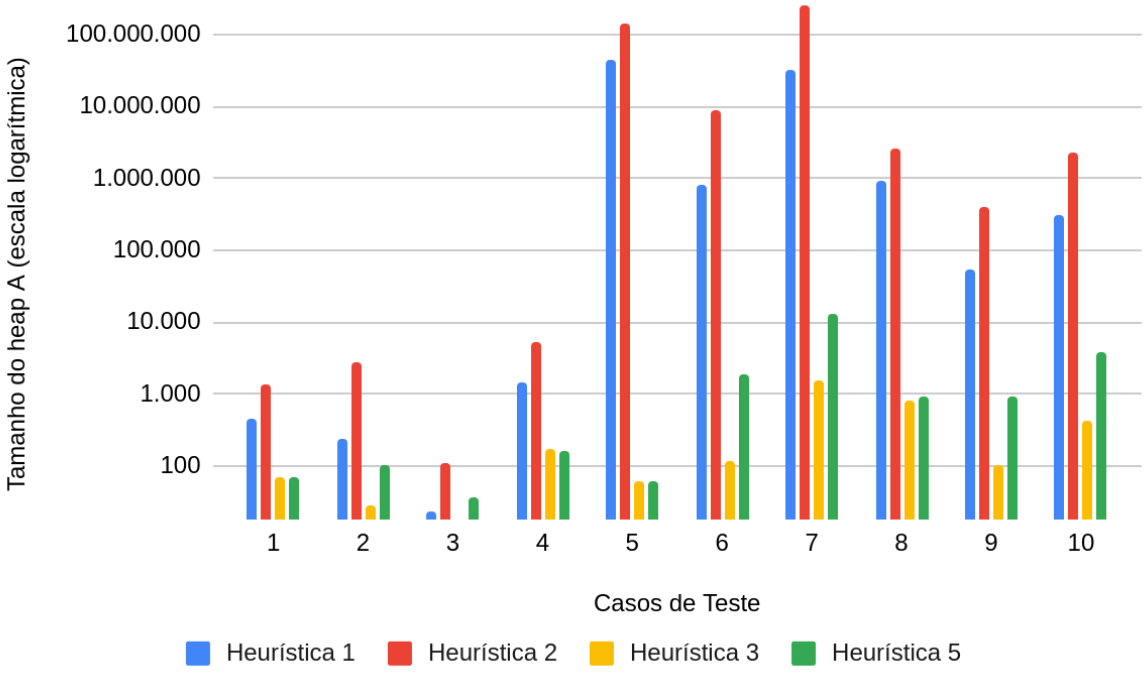

Figura 2 - Gráfico comparativo entre os tamanhos da estrutura heap do conjunto dos estados abertos.

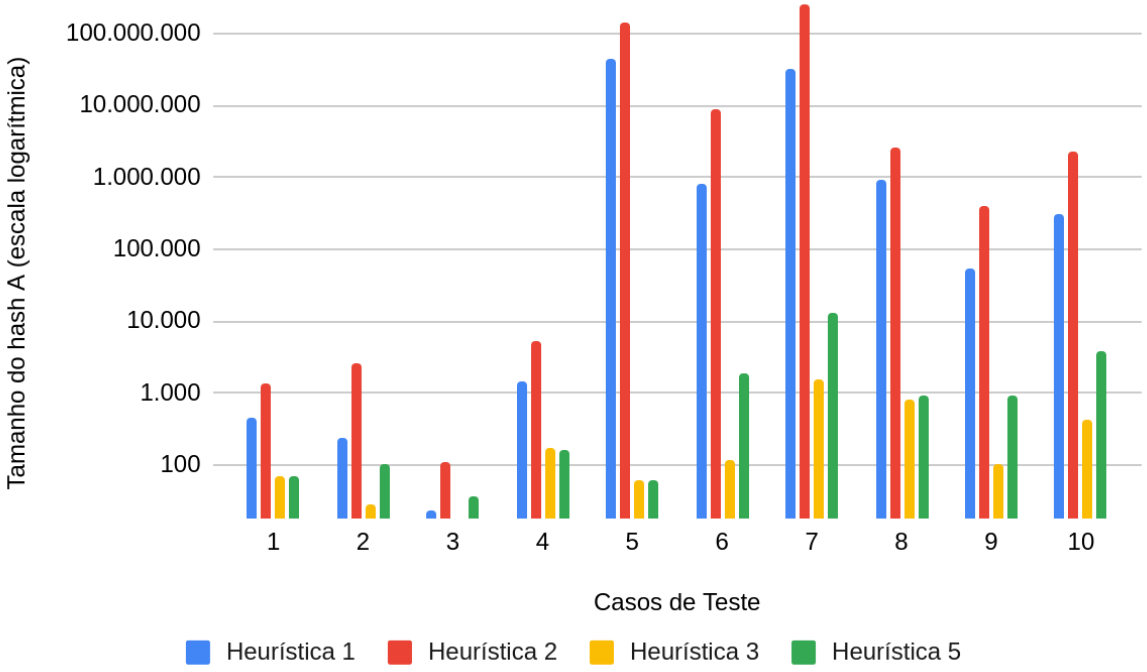

Figura 3 - Gráfico comparativo entre os tamanhos do hash do conjunto dos estados abertos.

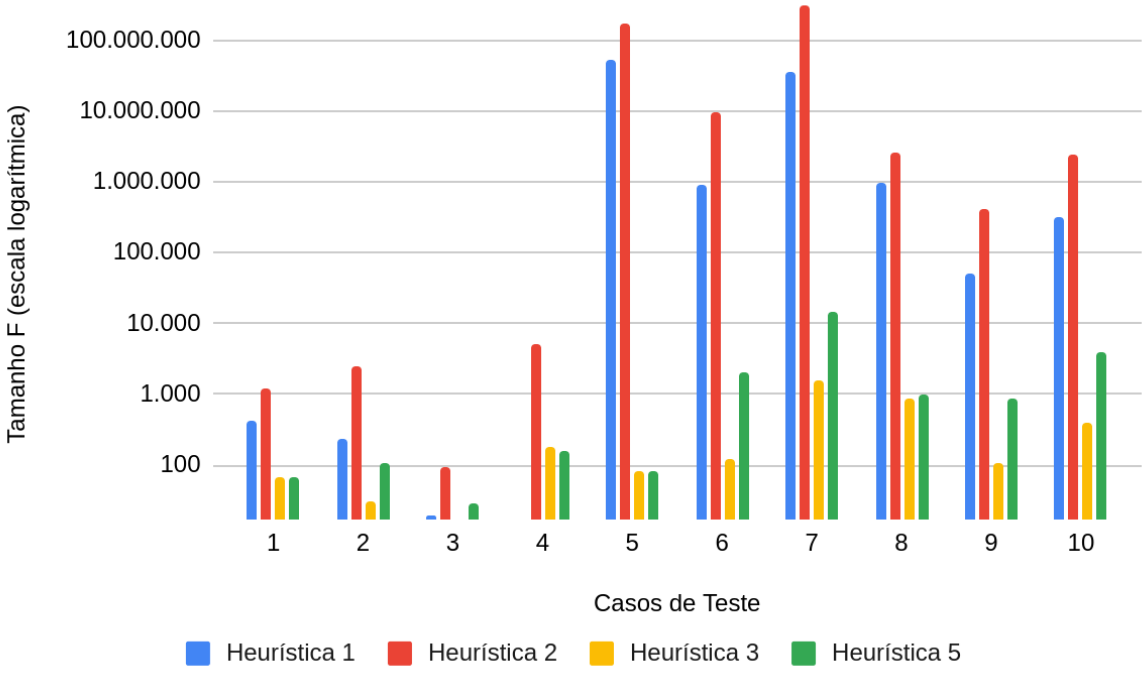

Figura 4 - Gráfico comparativo entre os tamanhos do conjunto dos estados fechados.

A partir das Figuras 1 a 4, pode-se perceber que a heurística mais eficiente, tanto em termos de tempo de execução quanto em memória alocada, é a heurística 3. Por outro lado, a

heurística que apresentou maior tempo de execução e maior memória alocada foi a de número 2. Os valores da heurística 4 foram ocultados pois como já mencionado anteriormente, obtiveram os mesmos valores da heurística 3.

Análise das linhas 9 e 10 do algoritmo

As linhas 9 e 10 responsáveis pela verificação de que sendo *m'* um novo sucessor gerado de um nó, se *m'* obteve um valor menor de *g(m')* do que *g(m)* sendo *m* já presente no conjunto dos Abertos, atualiza-se *m* para *m'*. Para a análise da influência das linhas 9 e 10 todos os testes foram novamente executados sem esta condição e os resultados estão tabelados no Apêndice B. Os dados mostraram que, para casos mais simples, o número de elementos no conjunto F, ou seja, a quantidade de nós computados, diminuiu sem a realização desta verificação. Consequentemente, houve uma pequena diminuição do tempo de execução para estes testes. Já para testes mais complexos com caminhos mínimos maiores é observado um aumento no número de nós a serem computados.

Nem todos os testes retornaram o resultado mínimo esperado. Sem essa verificação não há garantia de que o algoritmo retorne a quantidade mínima de movimentos para que se chegue ao estado final.

REFERÊNCIAS

CONSTANTINO, ADEMIR APARECIDO. **Algoritmos de busca na resolução de problemas**. 2021. Notas de aula.

ZANCHIN, Betina Carol. **Análise do algoritmo A* (A estrela) no planejamento de rotas de veículos autônomos**. 2018. 63 f. TCC (Graduação) - Curso de Engenharia Eletrônica, Departamento Acadêmico de Engenharia Eletrônica, Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2018.

APÊNDICE A

Abaixo estão apresentadas as tabelas com os valores numéricos dos resultados obtidos. Todos os casos de teste retornaram os resultados apresentados na Tabela 1.

Tabela A.1 - Resultados da heurística 1.

Caso de Teste	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	469	469	428	0,00130
2	246	245	231	0,00058
3	24	24	20	0,00008
4	1.488	1.485	1 398	0,00347
5	43.403.162	43.354.990	51.552.470	365,37171
6	832.837	830.832	908.689	9,53473
7	31.385.587	31.344.898	36.216.865	243,24919
8	920.548	919.004	974.824	16,30663
9	52.462	52.386	51.786	0,19049
10	308.259	307.558	323.114	1,55693

Tabela A.2 - Resultados da heurística 2.

Caso de Teste	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	1.368	1.364	1.206	0,00346
2	2.688	2.680	2.499	0,00652
3	107	107	94	0,00042
4	5.339	5.326	4.983	0,01320
5	144.609.422	144.300.766	170.918.241	1.298,31991
6	8.684.906	8.670.288	9.466.379	69,01040
7	260.551.190	259.778.201	317.527.377	2.604,56958
8	2.510.701	2.507.971	2.640.959	24,87312
9	405.562	404.792	406.194	3,76231
10	2.281.187	2.276.813	2.395.293	14,49535

Tabela A.3 - Resultados da heurística 3.

Caso de Teste	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	70	70	67	0,00040
2	28	28	30	0,00014
3	18	18	17	0,00009
4	175	175	175	0,00060
5	60	60	80	0,00028
6	120	120	122	0,00044
7	1.527	1.511	1.585	0,00352
8	827	818	844	0,00352
9	104	104	104	0,00063
10	414	414	405	0,00166

Tabela A.4 - Resultados da heurística 5.

Caso de Teste	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	70	70	67	0,000411
2	107	106	105	0,00037
3	36	36	28	0,00011
4	161	160	155	0,00054
5	60	60	80	0,00026
6	1.854	1.844	2.034	0,00693
7	13.144	12.889	14.243	0,05667
8	938	926	967	0,00334
9	914	904	871	0,00294
10	3.767	3.717	3.960	0,01367

APÊNDICE B

A seguir estão apresentados os resultados dos testes da execução do algoritmo A*, sem a verificação das linhas 9 e 10. O marcador "-" nos resultados indica que o retorno foi idêntico ao resultado apresentado na Tabela 1.

Tabela B.1 - Resultados da heurística 1, sem as linhas 9 e 10.

Caso de teste	Resultado	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	-	444	444	405	0,00120
2	-	245	245	231	0,00060
3	-	24	24	20	0,00008
4	-	1.478	1.478	1.402	0,00354
5	-	43.749.162	43.749.162	52.050.757	410,76997
6	-	902.807	902.807	989.388	10,02681
7	-	31.724.316	31.724.316	366.57.566	308,86757
8	-	898.717	898.717	950.292	15,71223
9	-	52.370	523.70	51.658	0,21094
10	-	306.110	306.110	320.454	1,62295

Tabela B.2 - Resultados da heurística 2, sem as linhas 9 e 10.

Caso de teste	Resultado	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	-	1.363	1.363	1.202	0,00345
2	-	2.699	2.699	2.502	0,00670
3	-	107	107	94	0,00043
4	-	5.123	5.123	4.806	0,01301
5	-	143.935.474	143.935.474	170.120.241	1.417,92846
6	-	13.633.425	13.633.425	15.056.542	145,99492
7	-	387.921.832	387.921.832	474.270.352	4.076,37819
8	-	2.740.248	2.740.248	2.889.303	257,13190
9	29*	1.904.957	1.904.957	1.971.733	13,84875
10	-	2.286.622	2.286.622	2.402.990	15,52994

*o resultado ótimo esperado era 27

Tabela B.3 - Resultados da heurística 3, sem as linhas 9 e 10.

Caso de teste	Resultado	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	-	70	70	67	0,00039
2	-	28	28	30	0,00011
3	-	18	18	17	0,00007
4	-	175	175	175	0,00057
5	-	60	60	80	0,00031
6	-	120	120	122	0,00039
7	-	1.474	1.474	1.530	0,00533
8	-	742	742	749	0,00262
9	-	104	104	104	0,00042
10	-	414	414	405	0,00129

Tabela B.4 - Resultados da heurística 5, sem as linhas 9 e 10.

Caso de teste	Resultado	Tamanho do heap (conjunto A)	Tamanho do hash (conjunto A)	Tamanho do hash (conjunto F)	Tempo de execução (segundos)
1	-	70	70	67	0,00042
2	-	106	106	105	0,00038
3	-	36	36	28	0,00011
4	-	159	159	153	0,00056
5	-	60	60	80	0,00027
6	-	1.878	1.878	2 055	0,00710
7	-	12.098	12.098	13.118	0,05222
8	-	924	924	953	0,00331
9	-	884	884	846	0,00303
10	-	3.648	3.648	3.846	0,01341