Algoritmos de busca na resolução do jogo dos 8

Anderson G. Moura - Daniel Mesquita - Hernandes Erick

[1]Universidade Federal do Piauí (UFPI) – Teresina, PI

1. Introdução

O jogo dos 8 consiste de uma matriz 9 por 9 onde se pode trocar as posições das células diretamente adjacentes de modo a chegar a um resultado desejado. Neste trabalho fazemos um levantamento comparativo entre 4 algoritmos de busca para a resolução desse puzzle. São eles:

- Busca Cega em Profundidade
- Busca Cega em Largura
- Busca Gulosa
- Busca A* (A-Estrela)

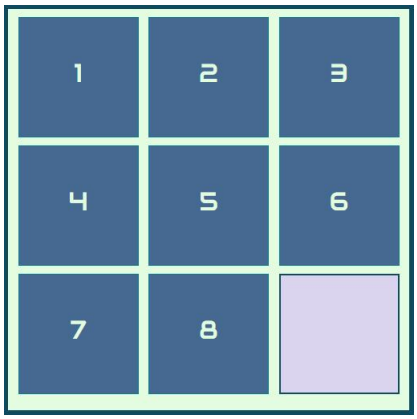

Figure 1. Solução desejada

Além disso, os quesitos de comparação definem-se como:

- **Tempo de Execução:** Relativo ao total de estados visitados
- **Memória Total:** Relativo ao total máximo de estados guardados em memória em qualquer momento da execução
- **Profundidade Máxima:** Nível máximo de profundidade alcançado pelo algoritmo
- **Profundidade da Solução:** Nível de profundidade da solução encontrada pelo algoritmo

Vale ressaltar que devido a natureza repetitiva do puzzle, todos os algoritmos mantêm um histórico de estados já percorridos, o que banaliza a informação de Memória Total, portanto trazemos esse valor como o máximo relativo ao total de estados no espaço de fronteira em vez do total real usado pelo algoritmo.

Foram utilizados os três estados iniciais de puzzle a seguir para a comparação:

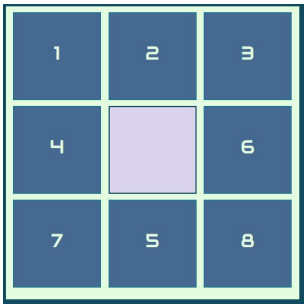

Figure 2. Problema 1

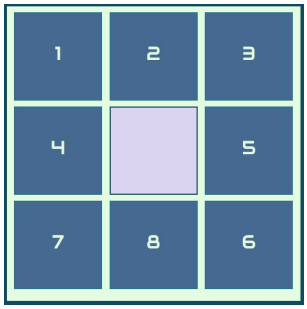

Figure 3. Problema 2

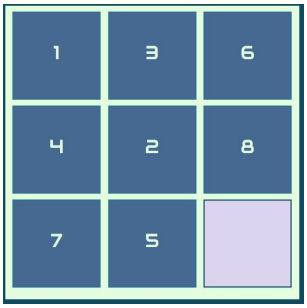

Figure 4. Problema 3

2. Busca Cega em Profundidade

A busca cega em profundidade consiste em escolher aleatóriamente um movimento do puzzle (nó), verificar se o mesmo pe a solução, e caso nsão seja, a partir dele, escolher o próximo nó, o que já configura também o próximo nível de profundidade.

Para esse algoritmo definiu-se um limite de profundidade adequado a cada problema (valor 4 para os problema UM e DOIS e valor 8 para o problema TRÊS), e manteve-se o histórico de estados visitados, de modo a impossibilitar contextos de loop infinito e/ou estouro de memória.

Por conta da natureza aleatória da escolha do próximo nó, diferentes execuções desse algoritmo devem dar resultados diferentes. Para efeitos de demonstração mostraremos o resultado de três execuções:

	Problema 1			Problema 2			Problema 3		
Tempo de Execução	34	25	30	25	21	7	22	22	96
Memória Total	5	5	5	5	5	5	8	8	8
Profundidade Máxima	5	5	5	5	5	5	8	8	8
Profundidade da Solução	3	3	3	3	3	3	7	7	7

Table 1. Resolução dos problemas utilizando Busca Cega em Profundidade

3. Busca Cega em Largura

A busca cega em largura consiste em verificar todos os nós de uma determinada profundidade da árvore afim de encontrar a solução, caso não encontre, todos os nós possíveis de cada estado desse nível são gerados e colocados em memória para que sejam então verificados pela solução.

Para esse algoritmo o limite de profundidade serve apenas para que seja identificado quando uma solução é impossível antes que todo nó seja de fato gerado. O histórico, no entanto, ainda é mantido, de modo a evitar a verificação de estados já verificados anteriormente.

	Problema 1	Problema 2	Problema 3
Tempo de Execução	10	9	54
Memória Total	8	8	40
Profundidade Máxima	3	3	7
Profundidade da Solução	3	3	7

Table 2. Resolução dos problemas utilizando Busca Cega em Largura

4. Busca Gulosa

A busca Gulosa se baseia em uma heurística para, a partir de um nó, escolher o seguinte o qual julga-se ser o melhor. Se tal heurística for escolhida de modo a não superestimar a situação em relação ao estado final, teremos uma heurística admissível que irá sempre nos trazer a melhor solução.

A heurística escolhida para esse algoritmo foi a Distância Manhattan, que consiste no somatório da distância perpendicular entre a posição atual e a final de cada um dos ponto da matriz, tal heurística é conhecida na literatura por ser admissível e portanto deve fornecer a melhor solução possível, como discutido anteriormente. Chamamos o resultado desse cálculo de ”fitness” do nó.

Da mesma forma como na busca em largura, o limite de profundidade serve apenas para que seja identificado quando uma solução é impossível antes que todo nó seja de fato gerado. O histórico, mais uma vez, é mantido, de modo a evitar eventuais loops que, por ventura, possam vir a acontecer.

Vale ressaltar ainda que foi considerado como nó visitado apenas aqueles nós onde de fato houve a verificação de solução, perceba que, de certa forma, outros nós são gerados para que seja calculado o fitness, mas não são escolhidos e portanto não visitados.

	Problema 1	Problema 2	Problema 3
Tempo de Execução	3	3	7
Memória Total	4	4	3
Profundidade Máxima	3	3	7
Profundidade da Solução	3	3	7

Table 3. Resolução dos problemas utilizando Busca Gulosa

5. Busca A*

Assim como a Busca Gulosa, a busca A* também se baseia em uma heurística, mas não escolhe um nó específico, mas os organiza pelo melhor fitness somado com seu custo de modo que o algoritmo possa voltar níveis na árvore caso o custo menor do nível acima compense a diferença pela melhor escolha. Ou seja, enquanto a busca gulosa escolhe o que julga melhor e segue, a busca A* não faz essa escolha de maneira definitiva, mas fica em constante avaliação do atual melhor valor de fitness + custo.

A heurística escolhida para esse algoritmo foi também a Distância Manhattan.

Novamente o limite de profundidade serve apenas para que seja identificado quando uma solução é impossível antes que todo nó seja de fato gerado. O histórico, mais uma vez, é mantido, de modo a evitar eventuais loops que, por ventura, possam vir a acontecer.

	Problema 1	Problema 2	Problema 3
Tempo de Execução	3	3	8
Memória Total	5	5	8
Profundidade Máxima	3	3	7
Profundidade da Solução	3	3	7

Table 4. Resolução dos problemas utilizando Busca A*

6. Comparando os algoritmos

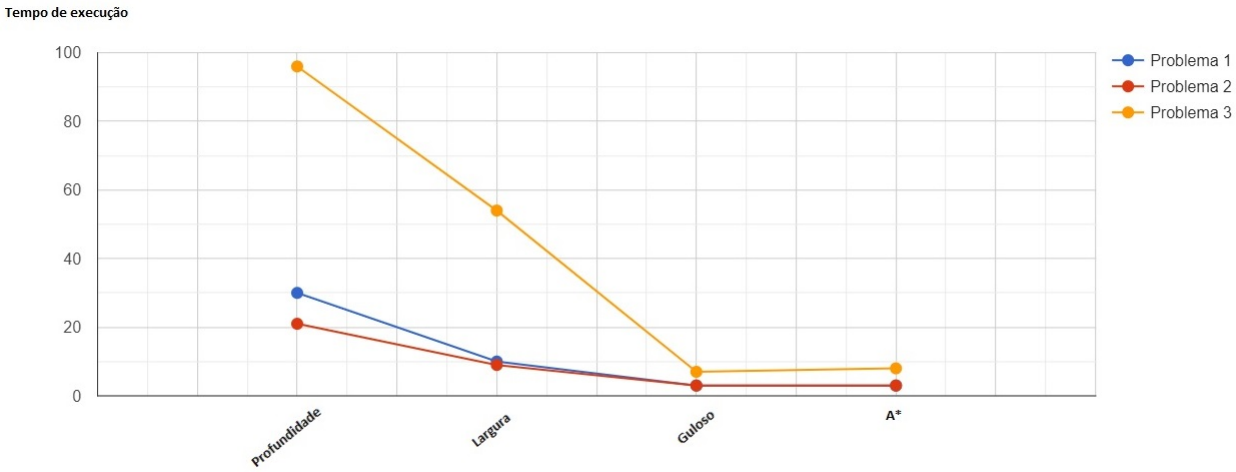

Figure 5. Em relação ao Tempo de Execução

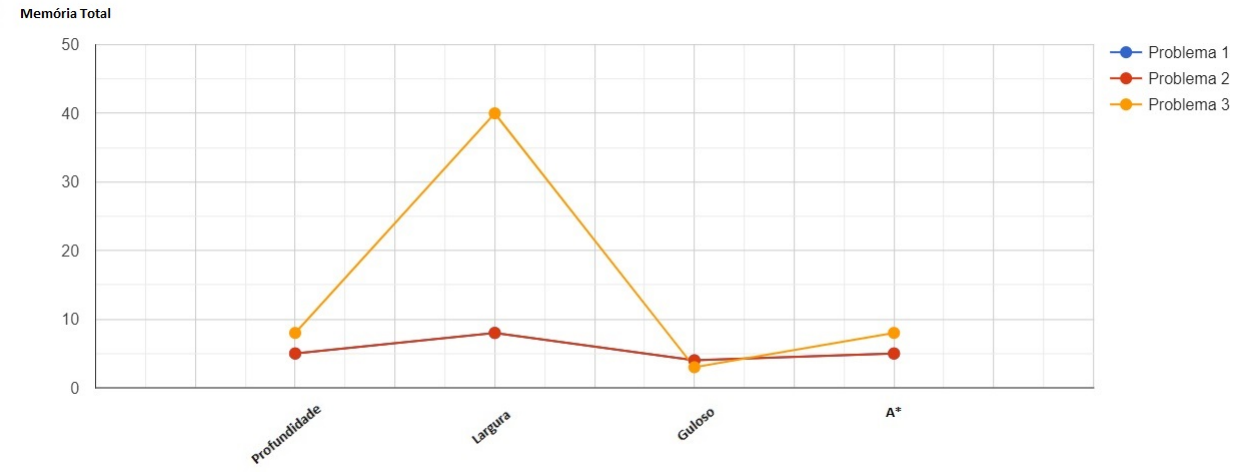

Figure 6. Em relação a Memória Total

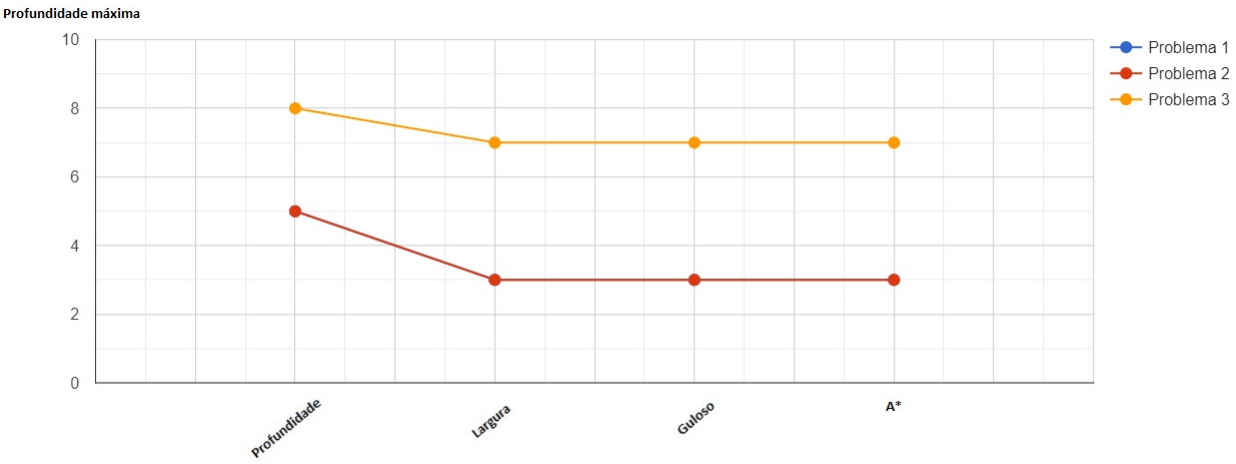

Figure 7. Em relação a Profundidade Máxima

7. Conclusão

Em relação ao tempo de execução, os algoritmos que se utilizam de heurísticas apresentam resultados consideravelmente melhores.

Quanto a memória utilizada, a busca cega em largura se mostra terrível como esperado, e temos uma pequena melhora do algoritmo guloso em relação ao A*.

A profundidade máxima dos algoritmos é a mesma exceto a busca cega em profundidade onde essa característica é claramente definido pelo valor de limite utilizado.

Todos os algoritmos encontraram a solução ótima, mas ressalta-se que o limite utilizado na busca em profundidade desempenha um papel fundamental nesse ocorrido, onde caso não utilizássemos esse limite, outra solução, não ótima, poderia ser obtida.