Trabalho de Inteligencia Artificial

Aluno: Cauan Halison Arantes de Oliveira - 554117

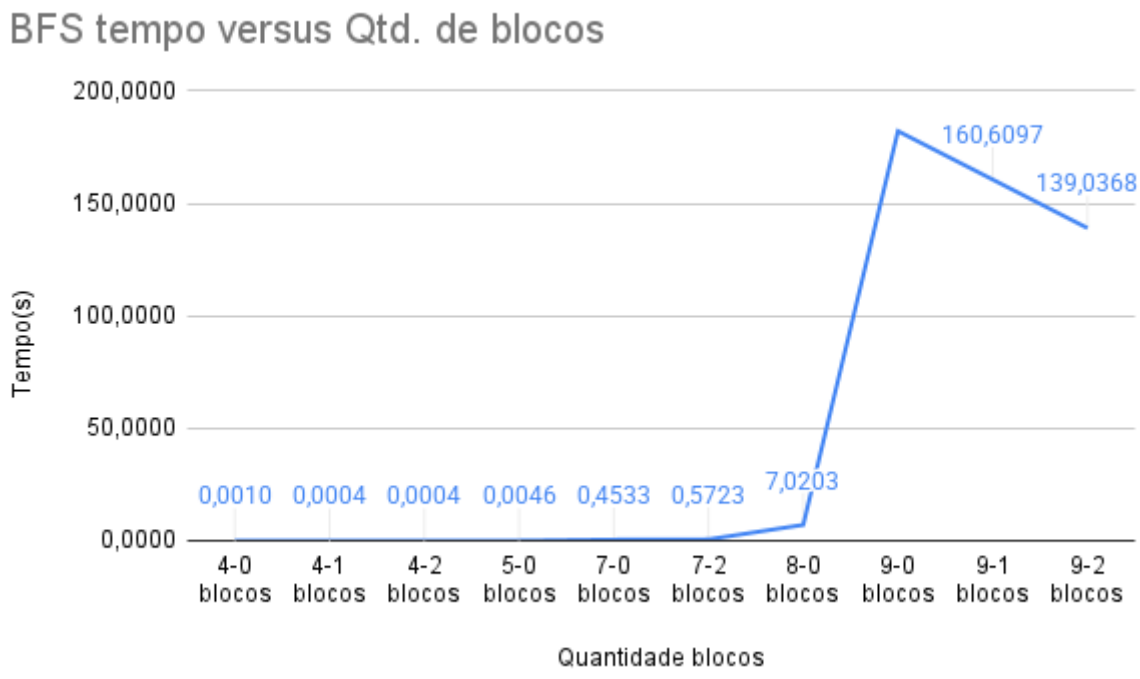
Máquina utilizada nos testes

- **CPU:** intel core i5-12450h
- **Memoria:** 8 Gb DDR5
- Instruções de uso: O código está configurado para ler a instância desejada a partir do arquivo instancias.txt. Para executar um algoritmo específico, basta descomentar a linha correspondente nas linhas finais do código.

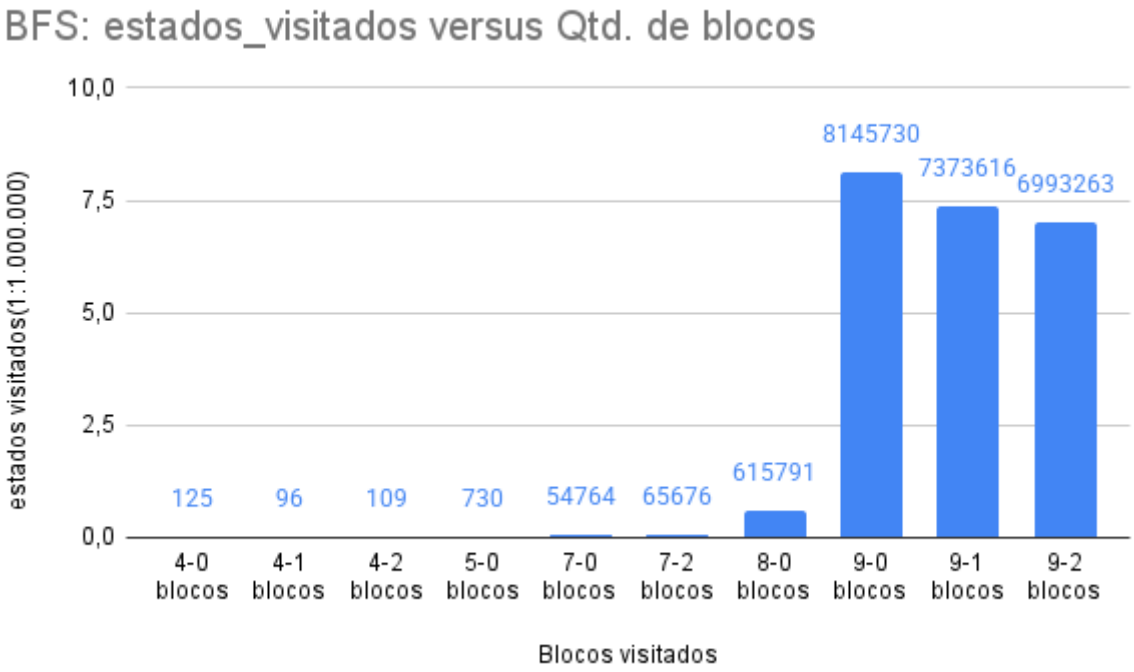
BFS

Este algoritmo apresentou um desempenho superior ao esperado, apesar de ser conhecido pelo alto consumo de memória. A implementação foi direta e permitiu resolver instâncias de até 9 blocos. Além disso, o algoritmo excedeu os 30 minutos de execução sem apresentar resultados.

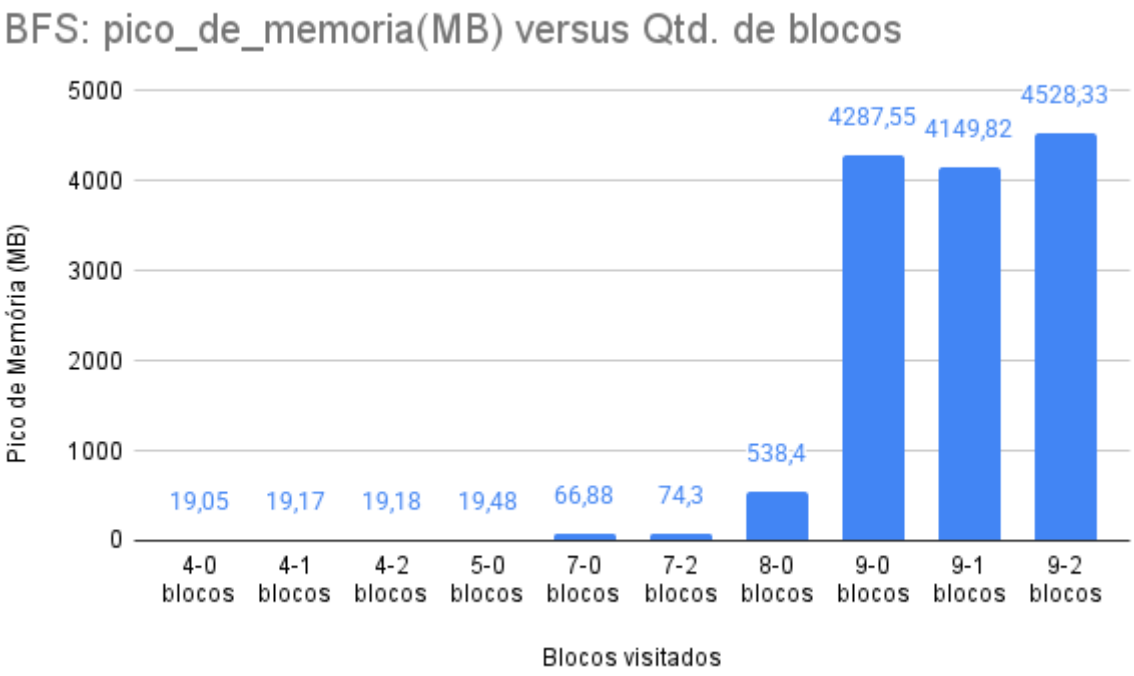
- Gráfico de tempo:



- Gráfico de Estados Visitados



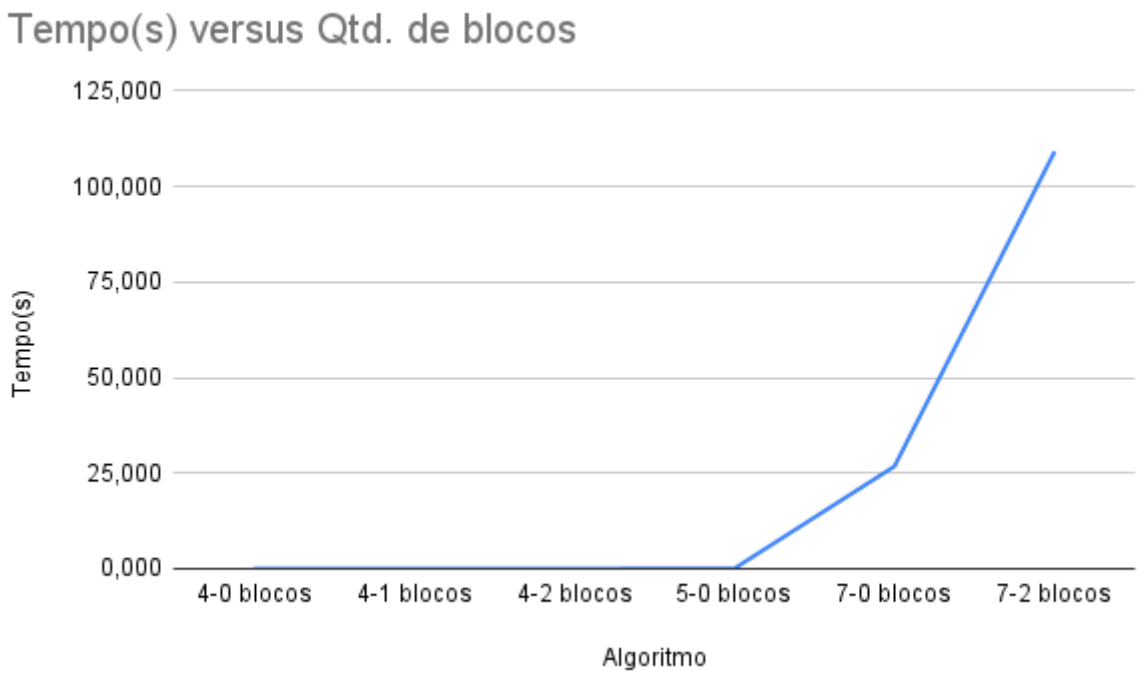
- Gráfico de Pico de Memória



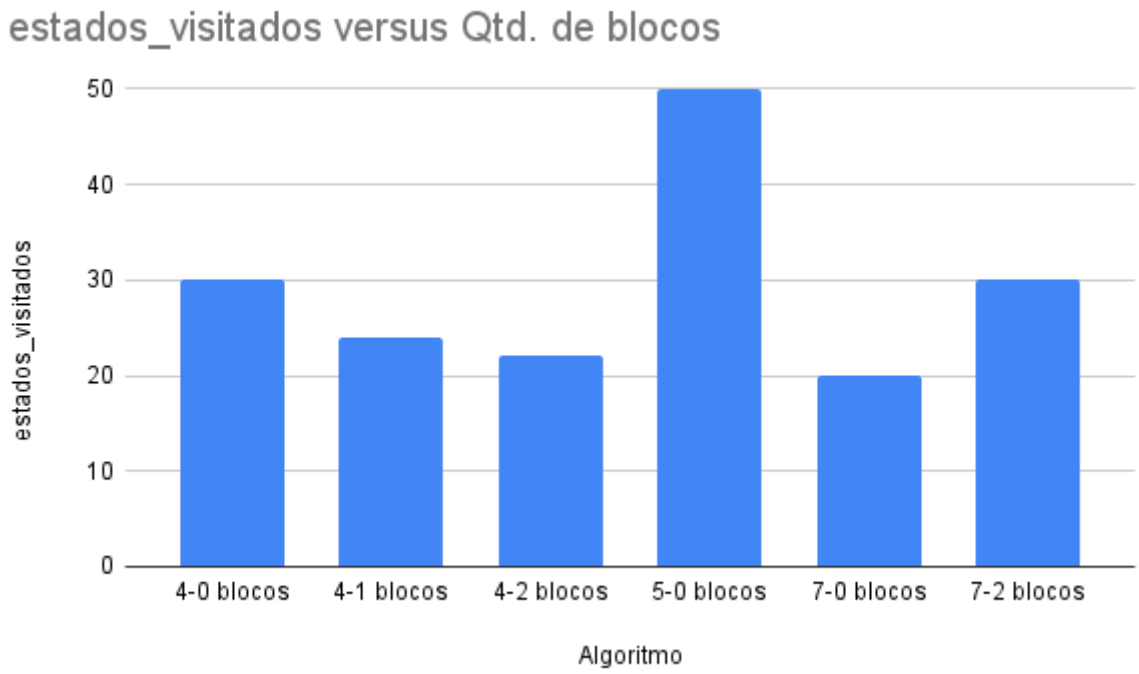
DLS

Neste algoritmo, a demanda de poder computacional foi mais acentuada. Observou-se que o algoritmo entrava em loops em certos cenários, possivelmente porque os estados visitados não eram armazenados, gerando redundância na exploração.

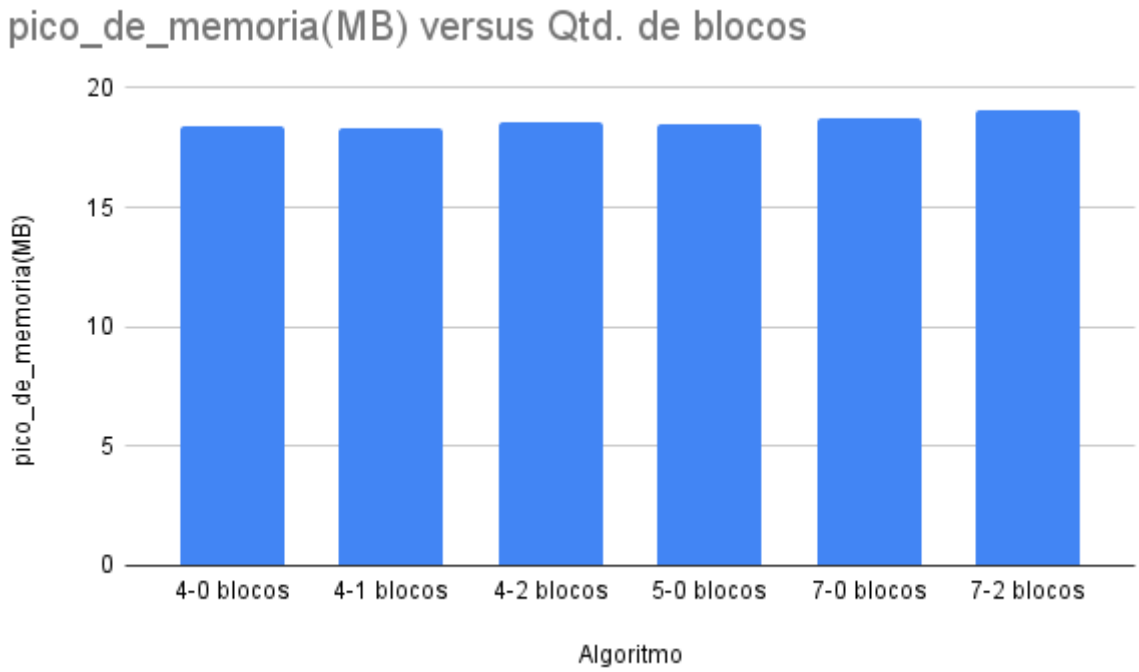
- Gráfico de tempo:



- Gráfico de Estados Visitados



- Gráfico de Pico de Memória

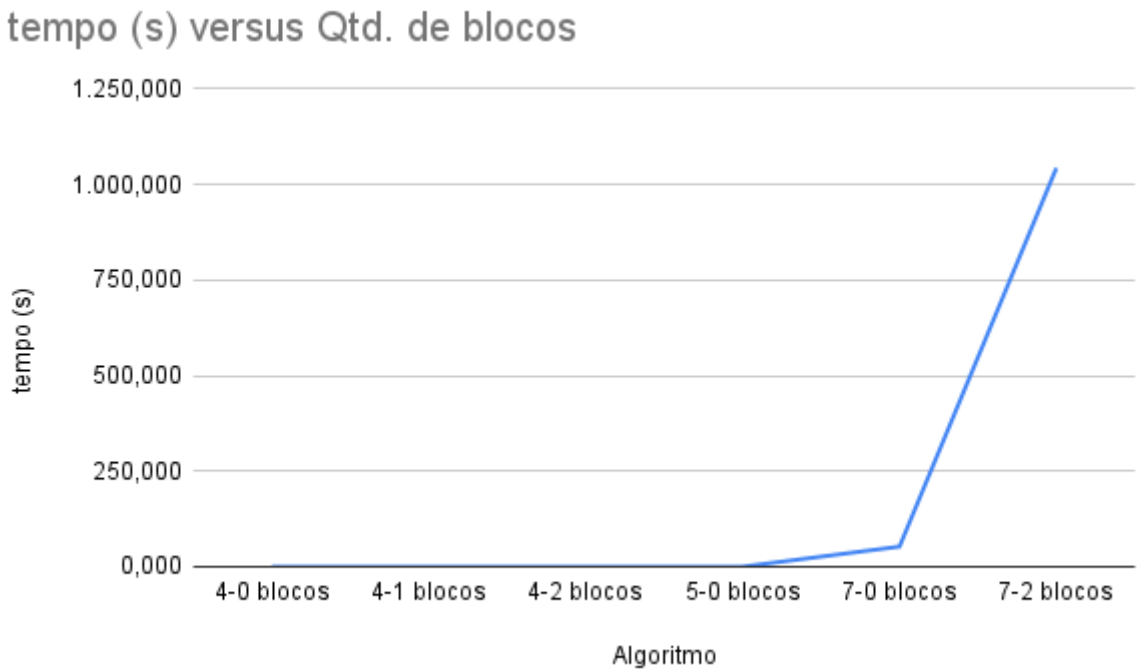


-

IDDFS

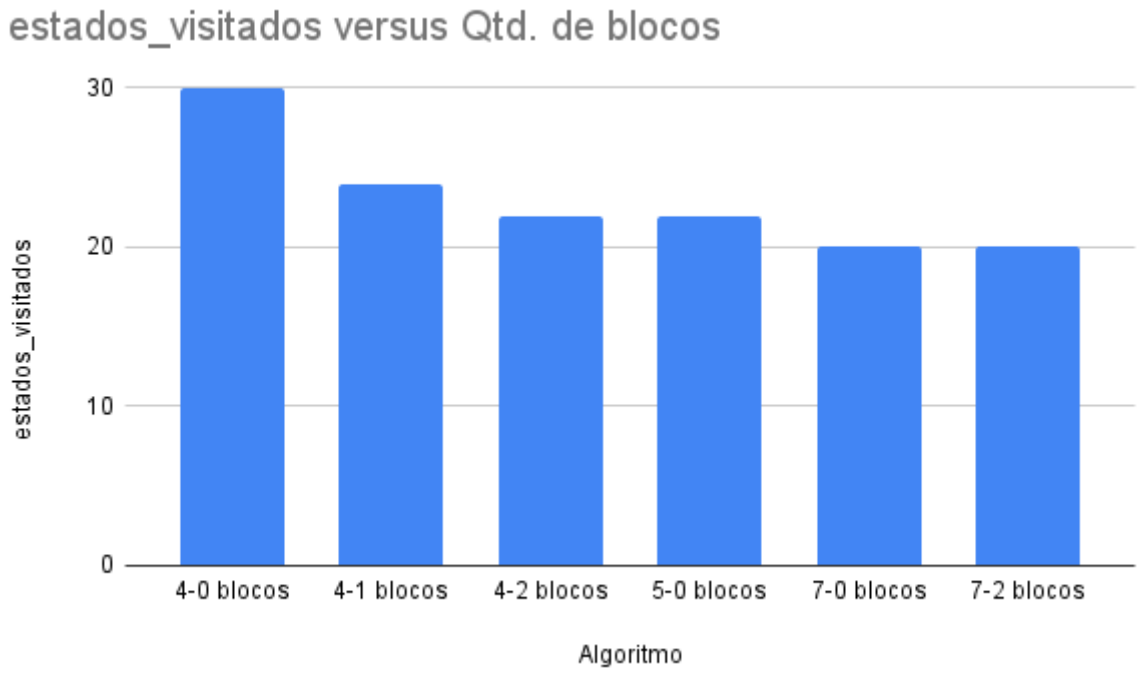
Assim como o DLS, o IDDFS exigiu bastante processamento. O tempo de execução foi superior ao do DLS para as mesmas instâncias, visto que o IDDFS realiza múltiplas iterações de buscas em profundidade com limites progressivos.

- Gráfico de tempo:

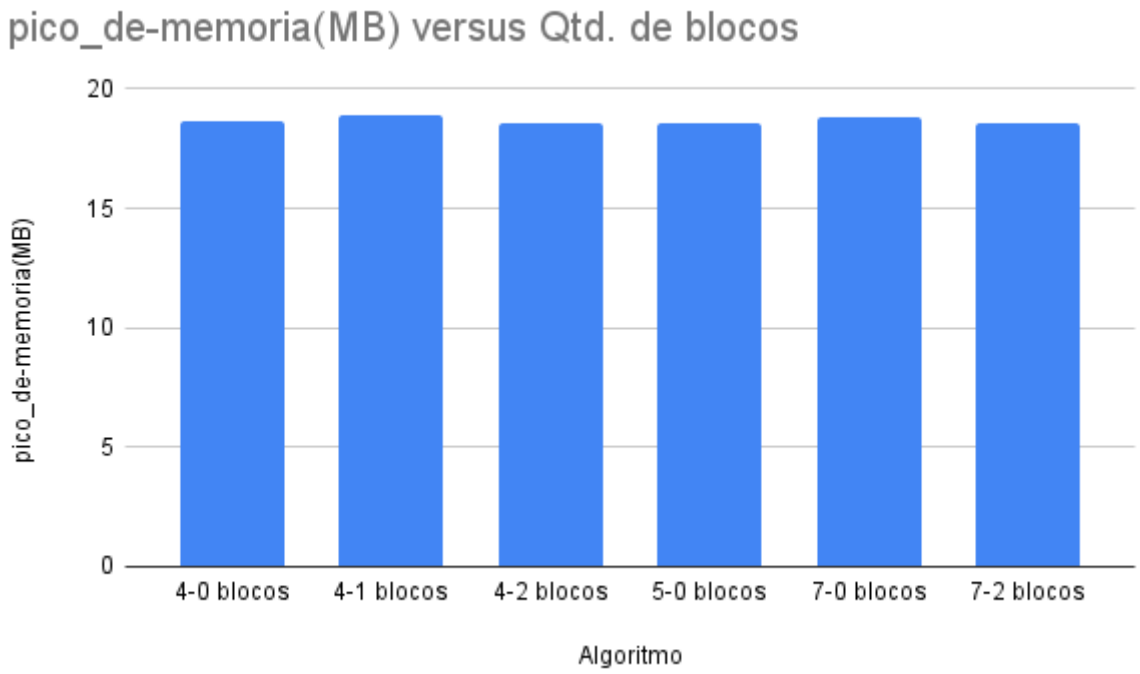


-

- Gráfico de Estados Visitados



- Gráfico de Pico de Memória

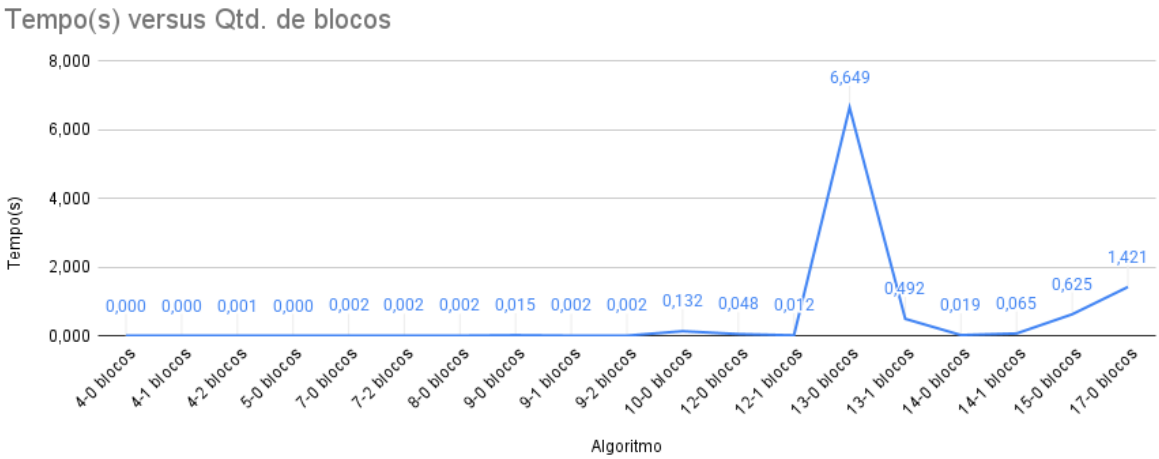


A*

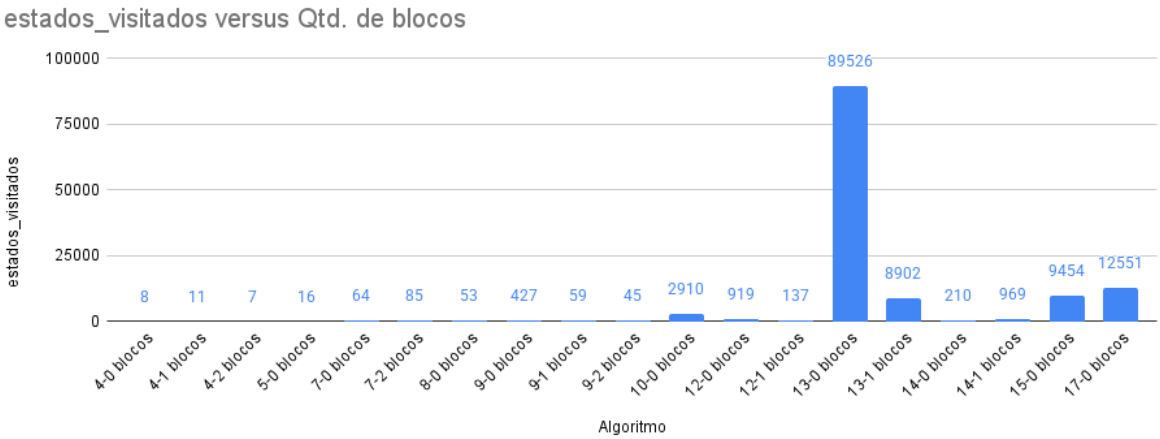
O algoritmo A* utiliza uma heurística própria: primeiro, conta-se quantos blocos estão fora da posição final esperada. Em seguida, verifica-se a base de cada pilha no estado atual em relação ao objetivo. Blocos fora de lugar recebem um multiplicador (peso) 2 para influenciar a prioridade da busca.

Durante os testes, notou-se que, embora o algoritmo chegue próximo ao ideal, em instâncias complexas ele pode divergir do caminho ótimo. Aumentar o peso para 3 ou 4 aumentou a eficácia, mas optei por manter o peso 2 para evitar que o algoritmo se tornasse excessivamente "guloso".

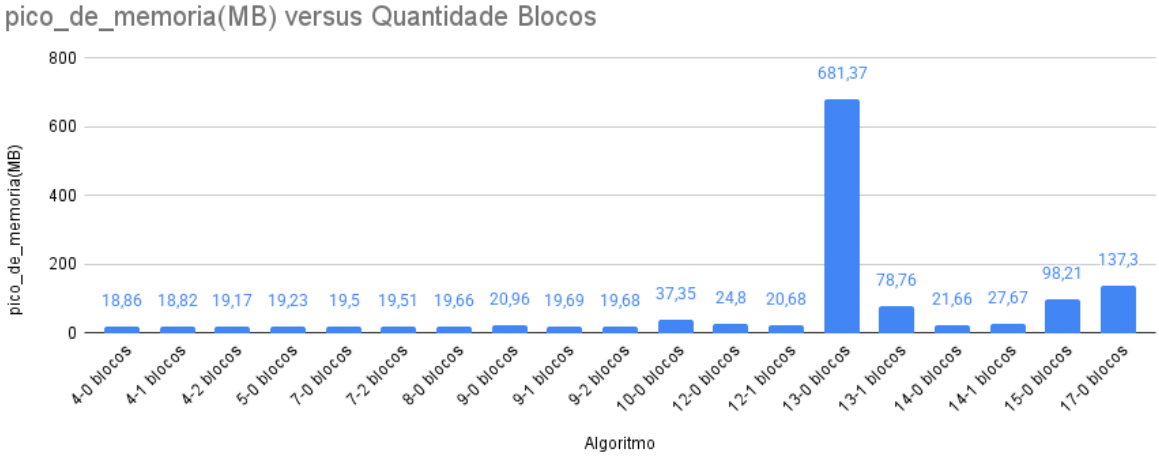
- Gráfico de tempo:



- Gráfico de Estados Visitados



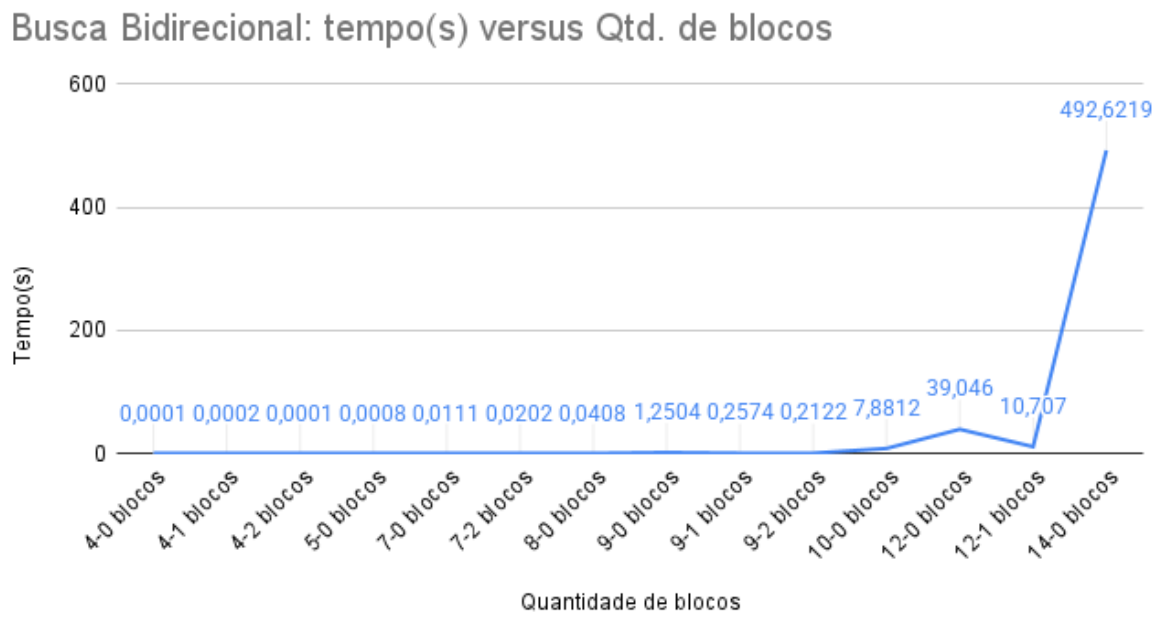
- Gráfico de Pico de Memória



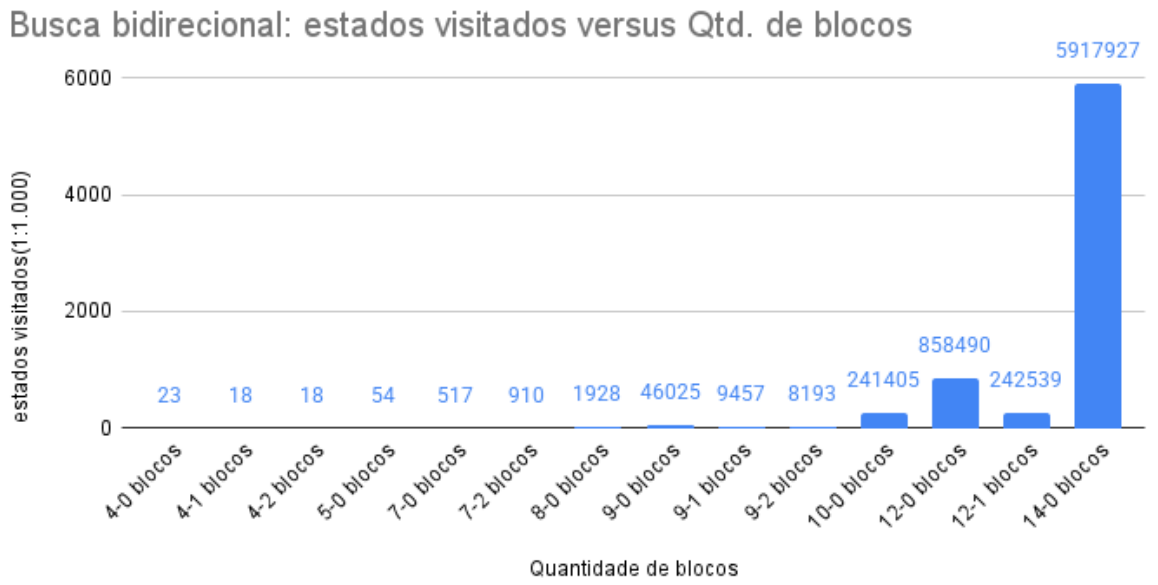
BUSCA BIDIRECIONAL

A maior dificuldade nesta implementação foi estruturar a busca reversa (do objetivo para o início). Utilizei a lógica do BFS para a expansão dos nós, garantindo que o algoritmo não gerasse estados inconsistentes.

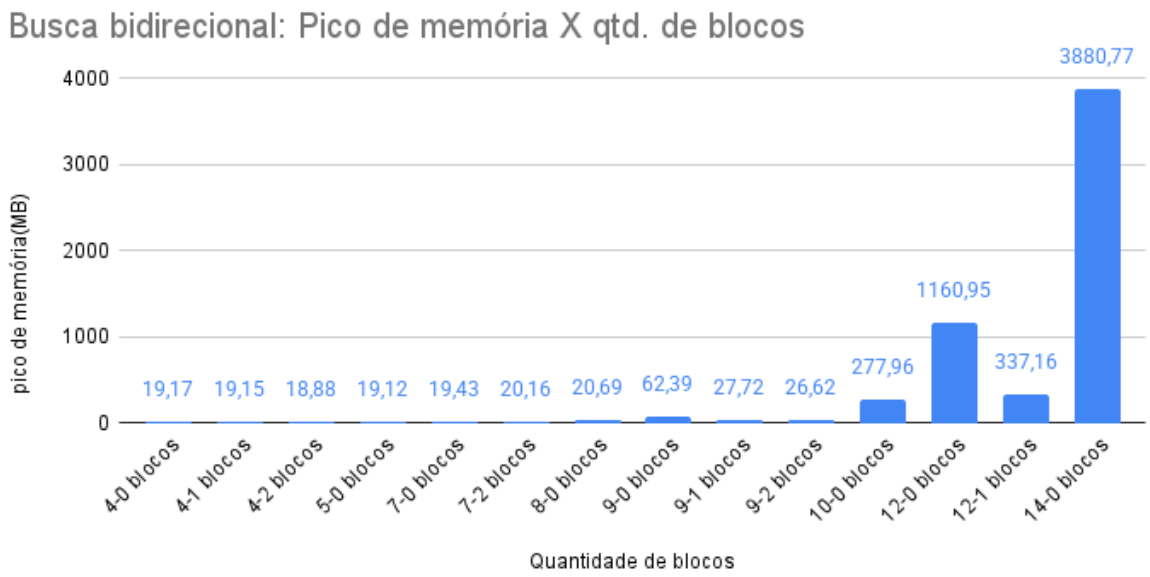
- Gráfico de tempo:



- Gráfico de Estados Visitados



- Gráfico de Pico de Memória



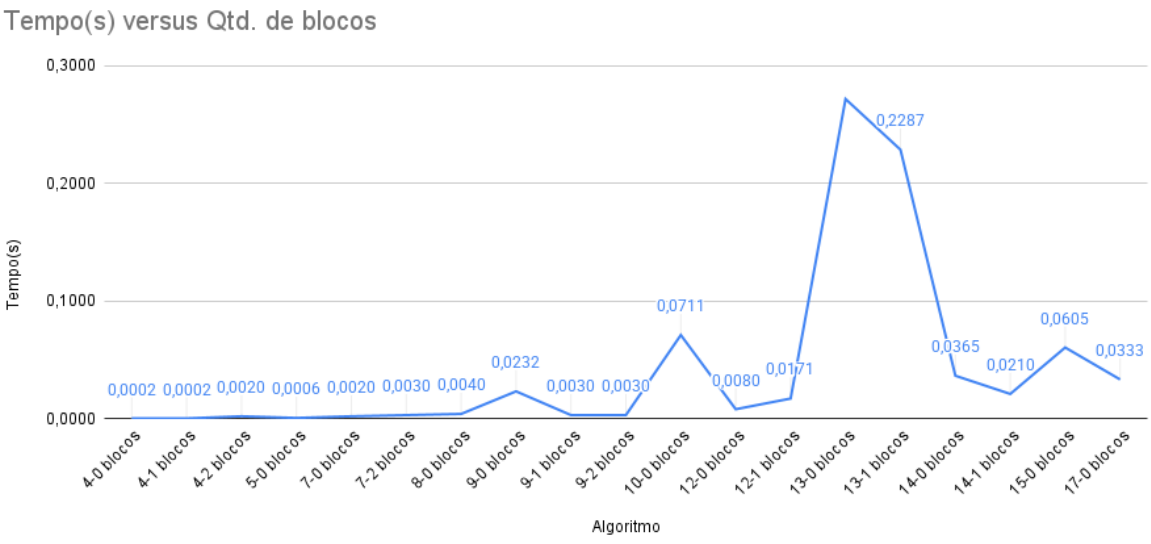
Considerações

Assim como nos outros métodos, houve dificuldade em resolver instâncias com 13 blocos, onde o consumo de memória excedia o limite disponível no sistema.

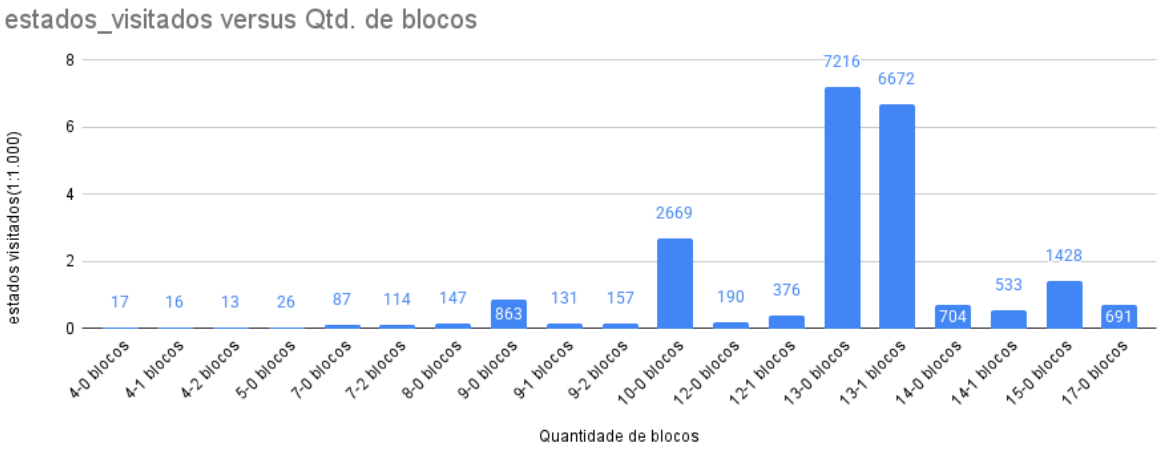
Busca bidirecional com heuristica

Implementei uma variação experimental combinando a busca bidirecional com a heurística desenvolvida para o A*. O objetivo foi unir a eficácia do A* com a velocidade característica da busca bidirecional.

- Gráfico de tempo:



- Gráfico de Estados Visitados



- Gráfico de Pico de Memória