Laboratório 2 - Busca Informada

Marcelo Buga Martins da Silva

CT-213 - Professor Marcos Ricardo Omena de Albuquerque Máximo

21/03/2021

1 Implementação

Três diferentes algoritmos foram implementados para o planejamento de um caminho entre um ponto inicial e um final. Os algoritmos utilizaram das posições em 8 conectado como os sucessores de cada nó, associando o custo de $\sqrt{2}$ para caminhos em diagonal e 1 para caminhos horizontais e verticais. Além disso, nós próximos de obstáculos possuem o dobro do custo para induzir os algoritmos a encontrarem caminhos mais seguros. Para a implementação, todos os algoritmos utilizaram uma lista de prioridades (*heapq* do *Python*). Os algoritmos foram implementados da seguinte forma:

1.1 Dijkstra

O algoritmo de Dijkstra foi implementado de forma a avaliar os custos de todas as células ao redor do ponto inicial, sem uma direção preferencial, até encontrar o ponto final. A fila de prioridades foi utilizada para que o nó de menor valor seja processado.

Nesse alogritmo foi necessário "fechar"os nós não só para evitar ciclos, mas também para evitar que alguns deles fossem processados mais de uma vez após serem atualizados. O algoritmo de Dijkstra garante que o caminho encontrado é um dos que possui o menor custo.

As Figuras 1 e 2 mostram o caminho planejado pelo algoritmo de Dijkstra para dois casos aleatórios.

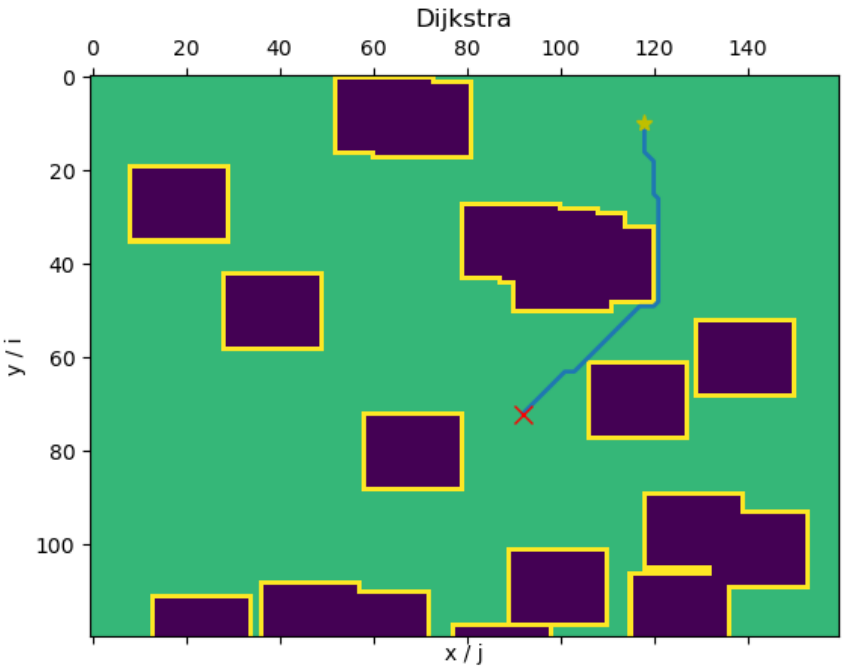

Figura 1: Planejamento de trajetória via Dijkstra - Caso 0

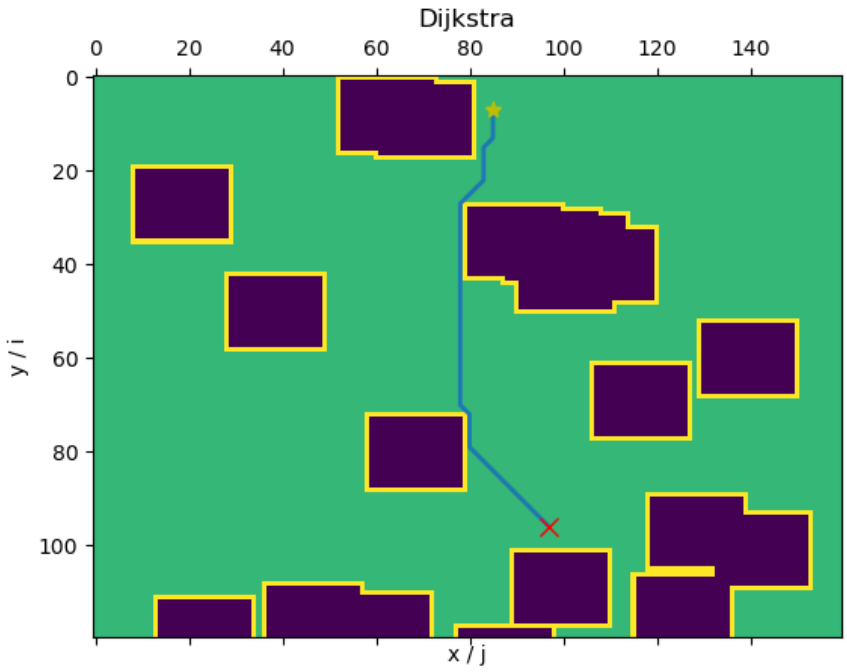

Figura 2: Planejamento de trajetória via Dijkstra - Caso 14

1.2 Greedy

O algoritmo Greedy utilizou como função base para custo uma função heurística dada pela distância real entre o ponto do nó atual e o destino. Aliando a utilização da função heurística com uma fila de prioridades que processa o nó com menor distância, o algoritmo tem execução muito rápida, embora não exista qualquer garantia que o caminho planejado é de fato o menos custoso.

As Figuras 3 e 4 mostram o caminho planejado pelo algoritmo Greedy para os mesmos casos da seção 1.1.

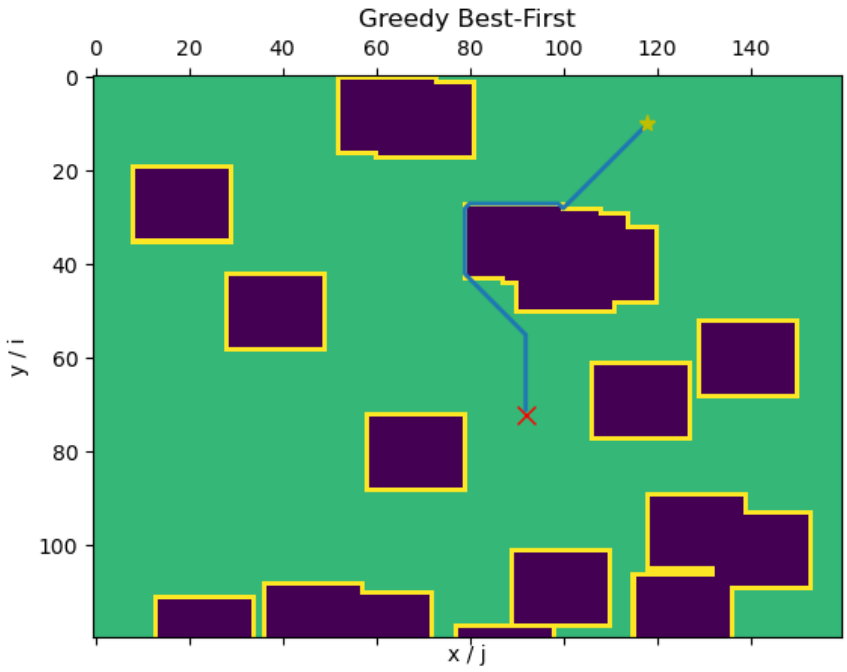

Figura 3: Planejamento de trajetória via Greedy - Caso 0

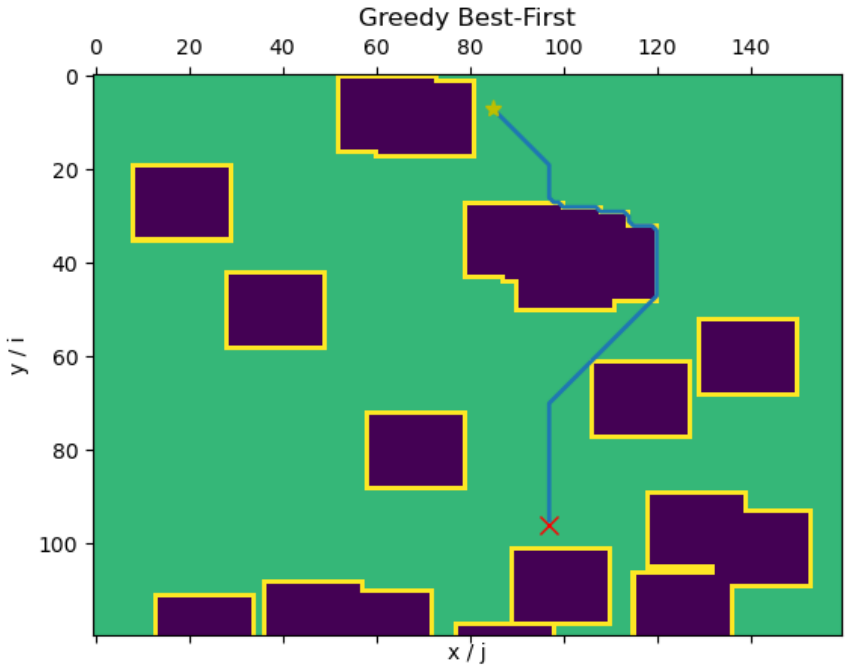

Figura 4: Planejamento de trajetória via Greedy - Caso 14

1.3 A*

Por fim, o algoritmo A* mescla a função heurística utilizada pelo algoritmo Greedy com o algortimo de Dijkstra para priorizar nós "promissores"e encontrar um dos caminhos. Sua implementação é idêntica à do algoritmo de Dijkstra, entretando a função "f"utilizada para a comparação dos custos tem o fator heurístico extra, sendo uma outra função, "g", utilizada para armazenar o custo total para chegar a cada nó.

As Figuras 5 e 6 mostram o caminho planejado pelo algoritmo A* para os mesmos casos da seção 1.1 e 1.2.

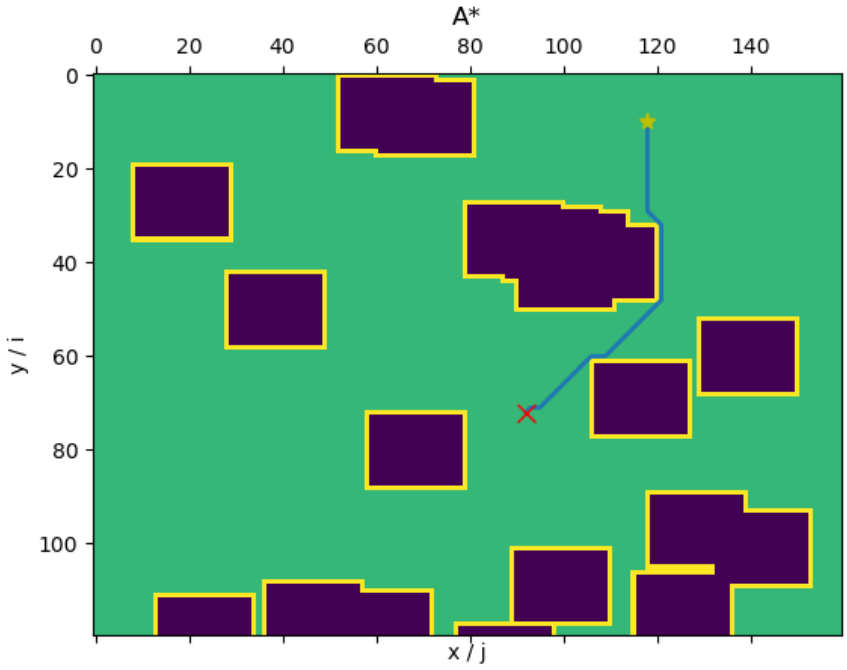

Figura 5: Planejamento de trajetória via A* - Caso 0

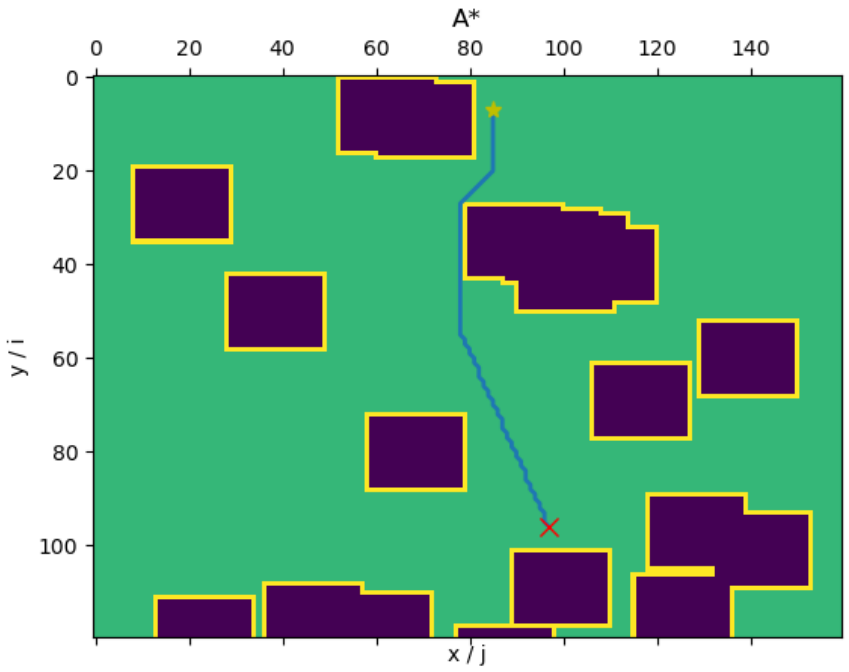

Figura 6: Planejamento de trajetória via A* - Caso 14

2 Comparação

A Tabela 1 foi gerada ao rodar 100 casos iguais para os três algoritmos. O processador utilizado foi um Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.

Tabela 1: Comparação de performance dos algoritmos

Algoritmo	**Tempo Computacional(s)**		**Custo do caminho(s)**	
	Média	**Desvio Padrão**	**Média**	**Desvio Padrão**
Dijkstra	0,1547	0,0872	79,8292	38,5709
Greedy	0,0061	0,0009	103,3420	59,4097
A*	0,0373	0,0327	79,8292	38,5709

Pode-se preceber que o algoritmo de Greedy é muito mais rápido do que os demais, entretanto, possui custo significativamente superior aos encontrados pelos outros algoritmos (evidenciado pela Figura 4). Tanto Dijkstra como A*, assim como esperado, tiveram resultados iguais para os custos dos caminhos (ainda que os caminhos encontrados não sejam exatamente os mesmos, como evidenciado pelas Figuras 2 e 6). Porém, A* se provou muito mais eficiente, tendo tempo computacional significativamente menor que Dijkstra devido à informação fornecida pela função heurística.