Instituto Tecnológico de Aeronáutica - ITA
CT-213 - Inteligência Artificial aplicada à Robótica Móvel
Aluno: Ulisses Lopes da Silva

Relatório do Laboratório 1 - Busca Informada

---

1 Breve Explicação em Alto Nível da Implementação

1.1 Algoritmo *Dijkstra*

O algoritmo *Dijkstra*, como visto em aula, generaliza a ideia do Breadth-First Search (BFS) a fim de obter o caminho de menor custo para arestas com custo não uniforme. Seu princípio consiste em percorrer os nós com base em três estados que podem assumir: **não visitado**, **em exploração** e **explorado**. Para isso, o algoritmo utiliza uma fila de prioridades para alocar os nós em exploração, removendo-os sequencialmente à medida que são totalmente explorados e têm seus custos completamente definidos. A obtenção do caminho de menor custo varia de acordo com a linguagem utilizada, podendo-se empregar diferentes estruturas de dados, tais como pilhas, listas, entre outras.

Se a estrutura de *heap* for utilizada para a fila de prioridades, a complexidade do algoritmo reduz-se para $O(\log n)$, aumentando significativamente sua eficiência geral. Além disso, uma otimização empregada para evitar percorrer todos os nós do grafo consiste na inclusão das condições de **verificação de nó objetivo** e de **nó já explorado**. Dessa forma, caso o nó objetivo seja alcançado, o *loop* é encerrado imediatamente; se um nó já visitado for encontrado, ele é ignorado, passando-se diretamente para o próximo.

Assim, o algoritmo foi implementado e, conforme ilustrado na imagem e na tabela, encontra o caminho ótimo até o objetivo de maneira razoavelmente eficiente. No entanto, ainda percorre um número significativo de nós, pois segue apenas o parâmetro $g(n)$, que representa o custo do nó atual até o nó n.

1.2 Algoritmo *Greedy Search*

O algoritmo *Greedy*, por sua vez, baseia-se exclusivamente em uma heurística pré-determinada, representada por $h(n)$, que, no contexto deste laboratório, corresponde à distância euclidiana do nó até o nó objetivo. Embora semelhante ao algoritmo de *Dijkstra*, ele se diferencia pelo critério de escolha do nó sucessor: seleciona-se sempre o nó com menor distância euclidiana até o objetivo.

Essa abordagem torna o algoritmo consideravelmente mais rápido, pois, a cada iteração, ele tende a se aproximar mais do objetivo. No entanto, pode não encontrar o caminho ótimo em termos de custo. Para o algoritmo guloso, cuja principal função é alcançar o objetivo o mais

rapidamente possível, não são necessárias verificações de menor custo, o que o torna mais veloz, embora menos preciso. Tal fato é corroborado pela imagem do caminho obtido, que, apesar de menos eficiente em termos de custo, apresenta processamento mais rápido, sendo uma alternativa interessante para aplicações em robótica móvel.

1.3 Algoritmo A*

O algoritmo A* pode ser compreendido como uma combinação da otimalidade do *Dijkstra* com a velocidade do *Greedy*. Ele considera tanto o custo atual do nó, $g(n)$, quanto a heurística $h(n)$ para o cálculo de $f(n)$, garantindo não apenas a obtenção do caminho ótimo, mas também um tempo de execução inferior ao do *Dijkstra*.

Conforme ilustrado na imagem, o caminho obtido pode diferir ligeiramente daquele gerado pelo algoritmo padrão. No entanto, a otimalidade é preservada, uma vez que podem existir múltiplos caminhos ótimos para o objetivo. Observando a tabela, nota-se que o custo e o desvio padrão são exatamente os mesmos, justamente pelo fato de que, da mesma forma que o *Dijkstra*, o A* encontra, de fato, os caminhos ótimos, gerando os mesmos custos. Além disso, a tabela demonstra que o A* é ligeiramente mais rápido que o *Dijkstra*, tornando-se uma escolha mais eficiente para diversas aplicações, desde que a heurística utilizada seja adequada ao problema. No caso específico deste laboratório, cujo objetivo era encontrar o caminho mínimo, a distância euclidiana mostrou-se não apenas **admissível**, mas também **consistente**.

2 Figuras Comprovando Funcionamento do Código

2.1 Algoritmo *Dijkstra*

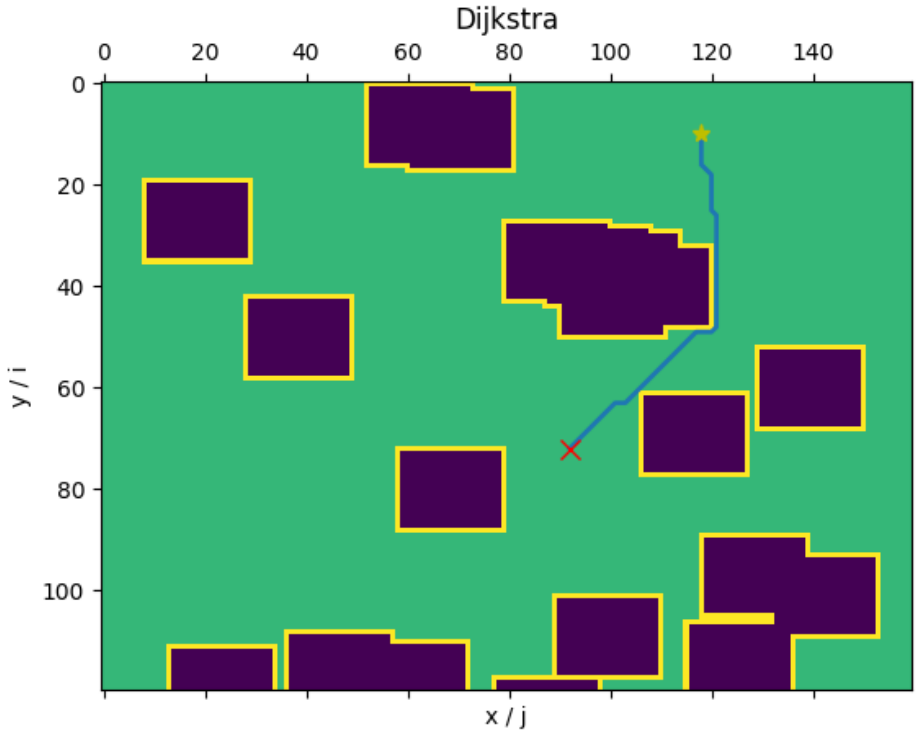

Fig. 1: Percurso obtido pelo algoritmo *Dijkstra*

2.2 Algoritmo *Greedy Search*

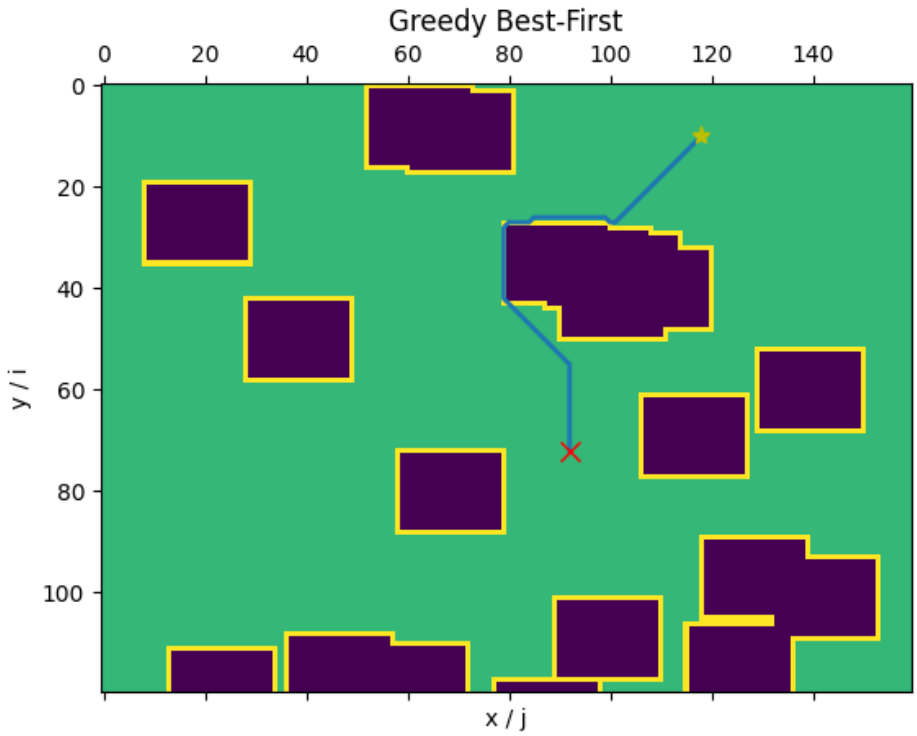

Fig. 2: Percurso obtido pelo algoritmo **Greedy Search**

2.3 Algoritmo A*

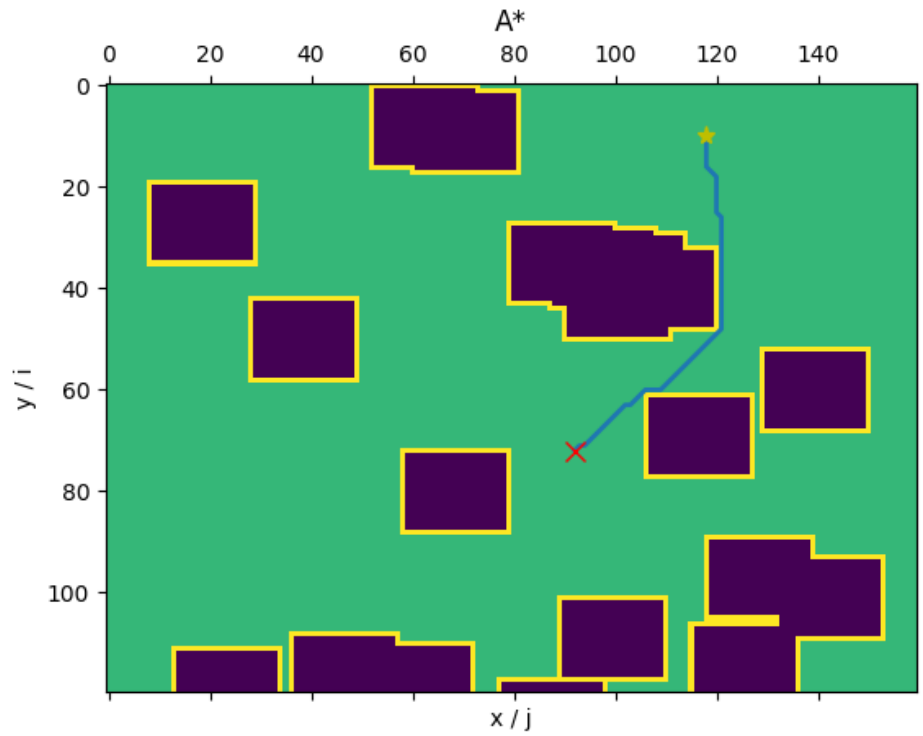

Fig. 3: Percurso obtido pelo algoritmo **A***

3 Comparação entre os Algoritmos

Tabela 1 com a comparação do tempo computacional, em segundos, e do custo do caminho entre os algoritmos usando um Monte Carlo com 100 iterações.

Tabela 1: tabela de comparação entre os algoritmos de planejamento de caminho.

Algoritmo	**Tempo computacional (s)**		**Custo do caminho**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Dijkstra	0.1252	0.0713	79.8292	38.5710
Greedy Search	0.0031	0.0012	103.1175	58.7940
A*	0.0206	0.0185	79.8292	38.5710