Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Pedro Pinheiro Borges

Relatório do Laboratório 2 - Busca Informada

1 Breve Explicação em Alto Nível da Implementação

1.1 Algoritmo Dijkstra

O Algoritmo Dijkstra foi implementado usando uma *priority queue* para armazenar os nós que ainda precisam ser analisados. Inicialmente, armazena-se o nó inicial na *priority queue* e define-se o custo de se chegar a ele como zero.

Em seguida, tira-se o nó de menor custo da fila (na primeira iteração sempre será o nó inicial), fato que determina que o nó já está com custo definido, e analisa-se os sucessores (nós adjacentes) desse nó e, caso o custo de se chegar até esse nó é maior do que o custo de se chegar até o nó anterior mais o custo da aresta que conecta eles dois, subsitui-se o custo original por essa soma citada e define-se como pai do nó em análise o nó anterior.

Finalmente, se esse nó ainda não teve seu custo definido, ou seja se ele não foi retirado da *priority queue*, ele é inserido na fila e repete-se esse processo para todos os nós que estão na fila até se chegar ao nó objetivo.

1.2 Algoritmo *Greedy Search*

O Algoritmo *Greedy Search* foi implementado usdando uma *priority queue* para armazenar os nós que ainda precisam ser analisados. Inicialmente, armazena-se o nó inicial na *priority queue* e define-se o custo de se chegar até o nó objetivo como a distancia entre o nó inicial e o nó objetivo.

Em seguida, tira-se o nó de menor custo da fila (na primeira iteração sempre será o nó inicial), fato que determina que o nó já está com custo definido, e analisa-se os sucessores (nós adjacentes) desse nó e, caso o custo desse nó é maior do que a distância desse nó até o nó objetivo, subsitui-se o custo original por essa distância citada e define-se como pai do nó em análise o nó anterior.

Finalmente, se esse nó ainda não teve seu custo definido, ou seja se ele não foi retirado da *priority queue*, ele é inserido na fila e repete-se esse processo para todos os nós que estão na fila até se chegar ao nó objetivo.

1.3 Algoritmo A*

O Algoritmo A* foi impledmentado usando uma *priority queue* para armazenar os nós que ainda precisam ser analisados. Inicialmente, armazena-se o nó inicial na *priority queue* e define-se o custo de se chegar a ele como zero e o custo de se chegar até o nó final passando pelo nó em questão como a distância entre o nó inicial e o nó objetivo.

Em seguida, tira-se o nó de menor custo da fila (na primeira iteração sempre será o nó inicial), fato que determina que o nó já está com custo definido, e analisa-se os sucessores (nós adjacentes) desse nó e, caso o custo de se chegar até o nó objetivo passando pelo nó em questão é maior do que o custo de se chegar até o nó anterior mais o custo da aresta que conecta eles dois mais a distância entre o nó em questão até o nó objetivo, subsitui-se o o custo de se chegar até o nó em questão pelo custo de se chegar até o nó anterior mais o custo da aresta que conecta eles dois, define-se o custo de se chegar até o nó objetivo pelo custo de se chegar até o nó em questão mais a distância até nó objetivo e define-se como pai do nó em análise o nó anterior.

Finalmente, se esse nó ainda não teve seu custo definido, ou seja se ele não foi retirado da *priority queue*, ele é inserido na fila e repete-se esse processo para todos os nós que estão na fila até se chegar ao nó objetivo.

2 Figuras Comprovando Funcionamento do Código

2.1 Algoritmo Dijkstra

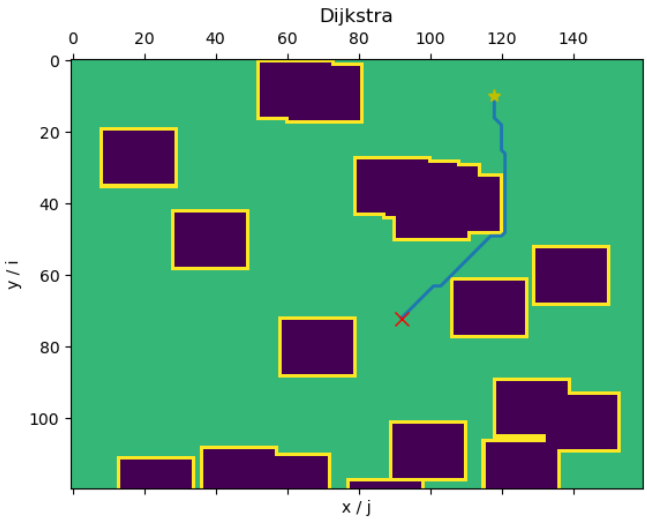

Figura 1: Caminho obtido pelo Algoritmo Dijkstra

2.2 Algoritmo *Greedy Search*

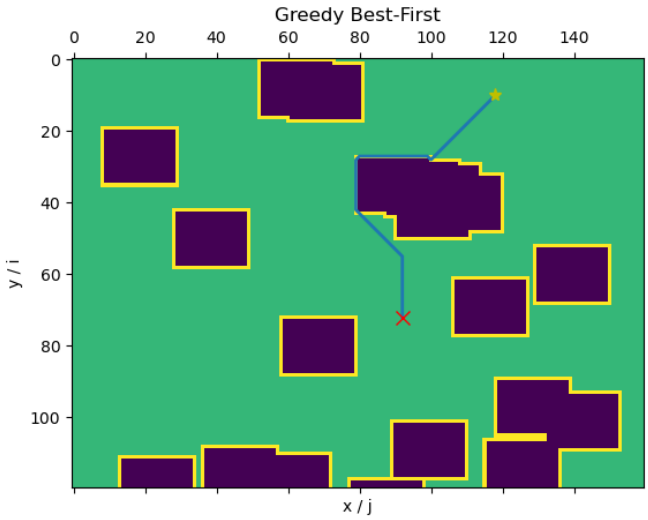

Figura 2: Caminho obtido pelo Algoritmo Greedy Search

2.3 Algoritmo A*

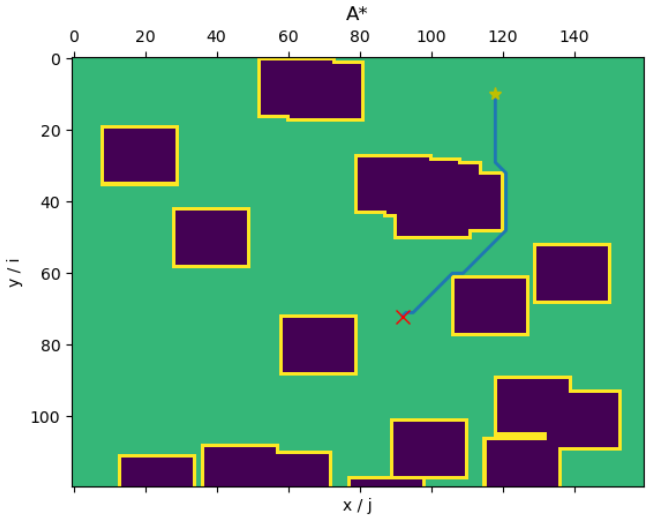

Figura 3: Caminho obtido pelo Algoritmo A*

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	0.1315	0.0740	79.8291	38.5709
Greedy Search	0.0047	0.0008	103.3420	59.4097
A*	0.0387	0.0380	79.8292	38.5710