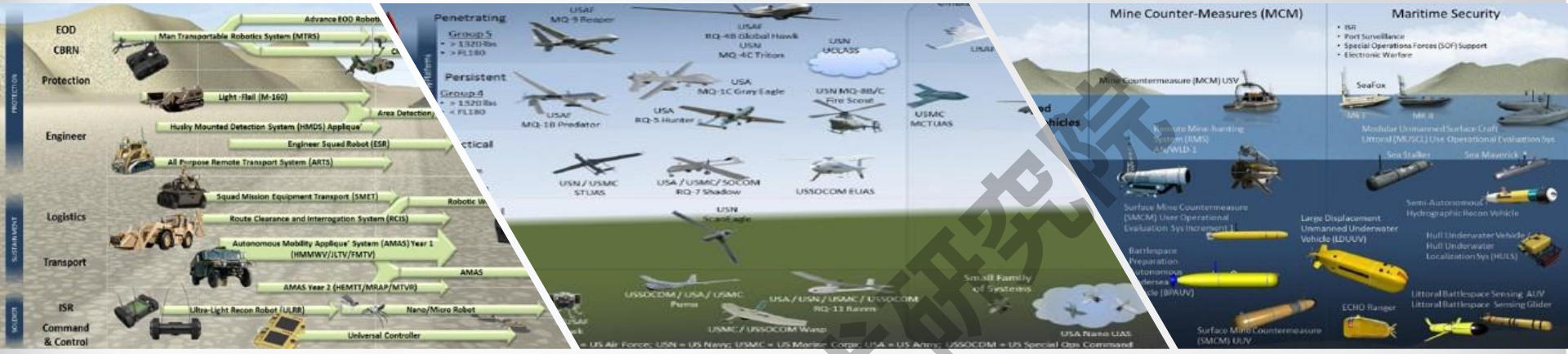


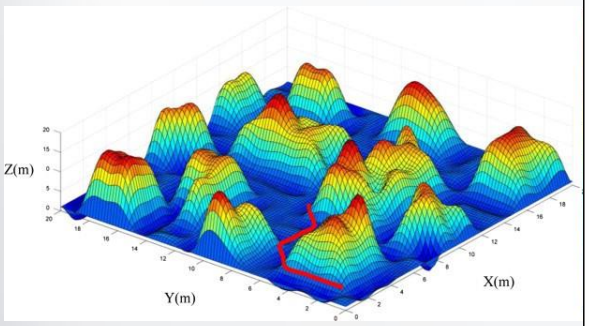
典型路径规划算法

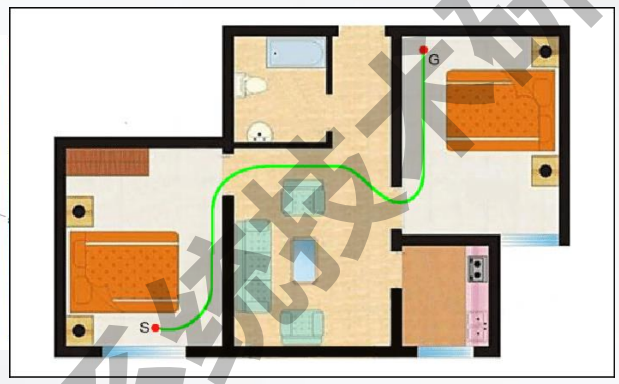
无人系统技术研究院
张勃 zhbo@nwpu.edu.cn

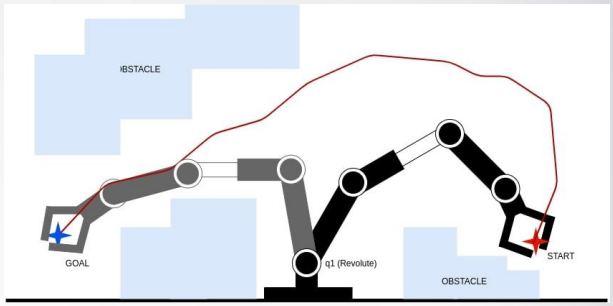
3 典型路径规划算法

什么是路径规划

路径规划是根据一定的评估准则，计算障碍环境中从起始点到目标点的该准则下的最优几何路径的过程。







路径规划的要素

1 障碍环境

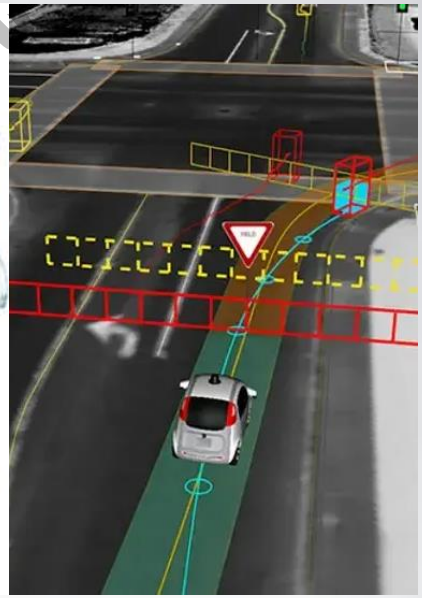
2 评估准则

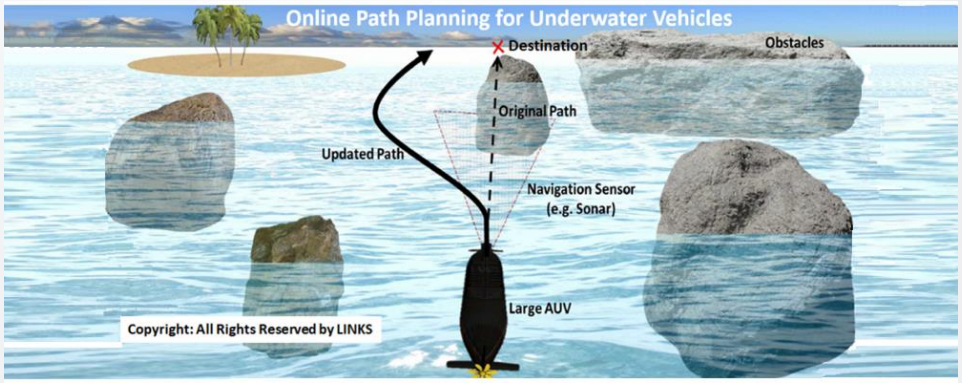
3 最优路径

典型路径规划算法

路径规划的应用

- 无人机
- 无人车
- 航天器
- 潜航器
- 机器人
- 机械臂

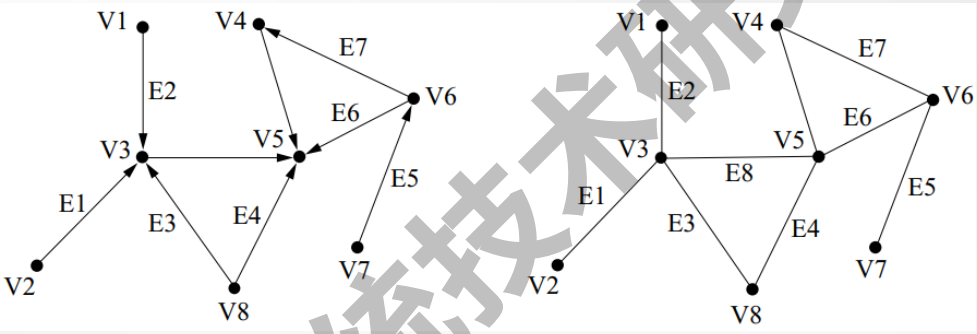




A*搜索算法

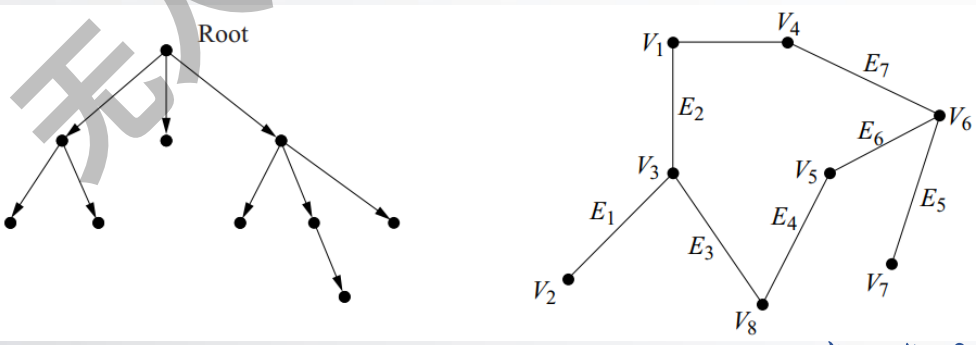
图（Graphs）

图是节点和边的集合。



树（Trees）

树是没有环的连通有向图。



典型路径规划算法

A*算法

深度优先搜索（Depth-first Search）

Start → A → D　　A → E → Goal

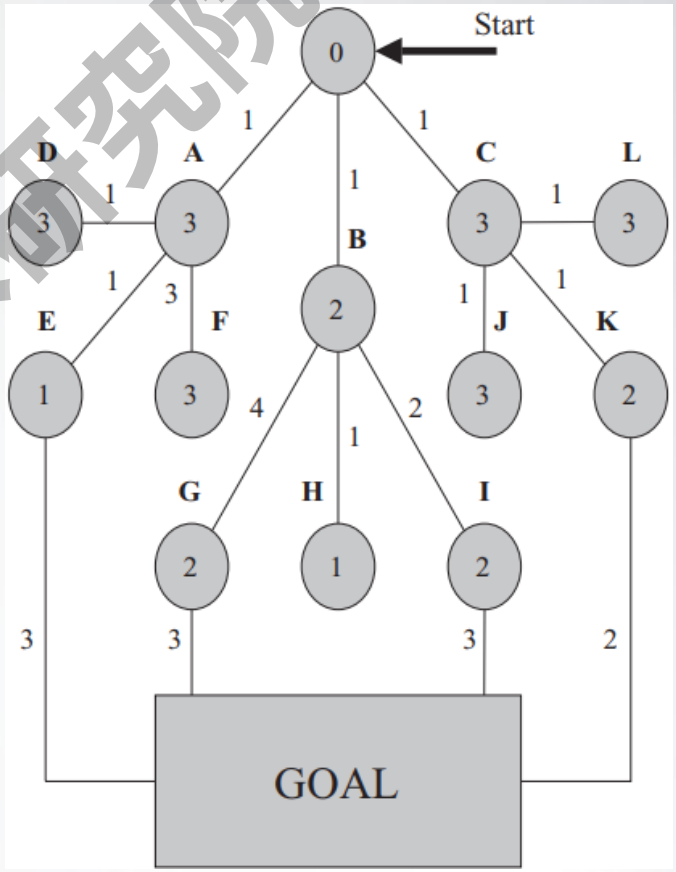
广度优先搜索（Breadth-first Search）

Start → A , B, C　　A → D, E, F

B → G, H, I　　C → J, K, L

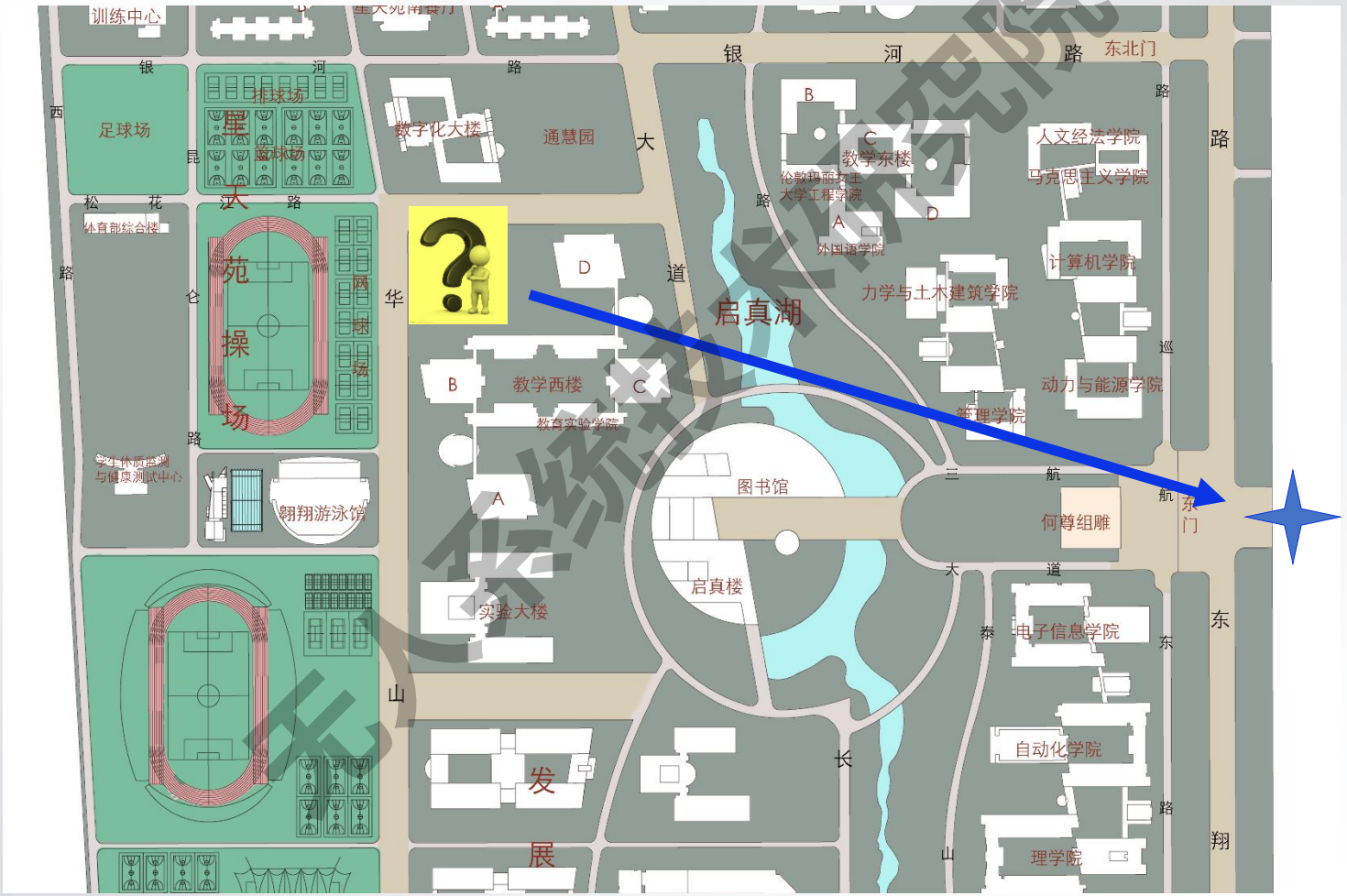
E → Goal

- 盲目搜索，且没有考虑路径代价



典型路径规划算法

A*算法



典型路径规划算法

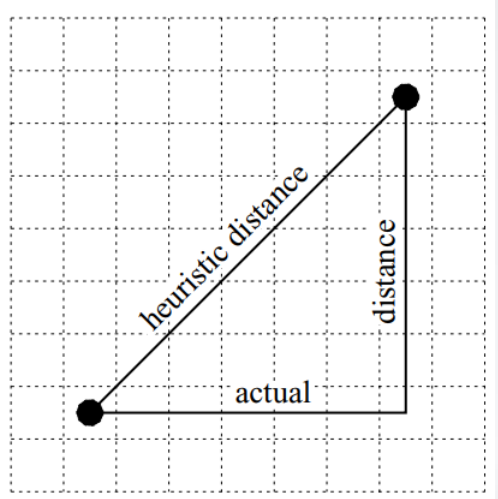
A*算法

- 如果能够利用目标的一些信息，则可以更高效地搜索出最优路径。

- 对于一个图，虽然可以获得一些关于目标的信息，但是搜索到目标之前最好的情况是能够给出一个反映期望的启发函数。

- 启发函数必须小于或等于从当前节点到目标节点的实际代价，否则可能使搜索效率降低，并且可能仅得到次优路径。

A*算法包含一个优先级列表，列表中的元素按优先级进行排序。优先级由已经耗费的代价和启发函数的估计代价之和决定。将每一次搜索得到的优先级最高的节点连接起来就可以得到条由起点到终点的通路。



典型路径规划算法

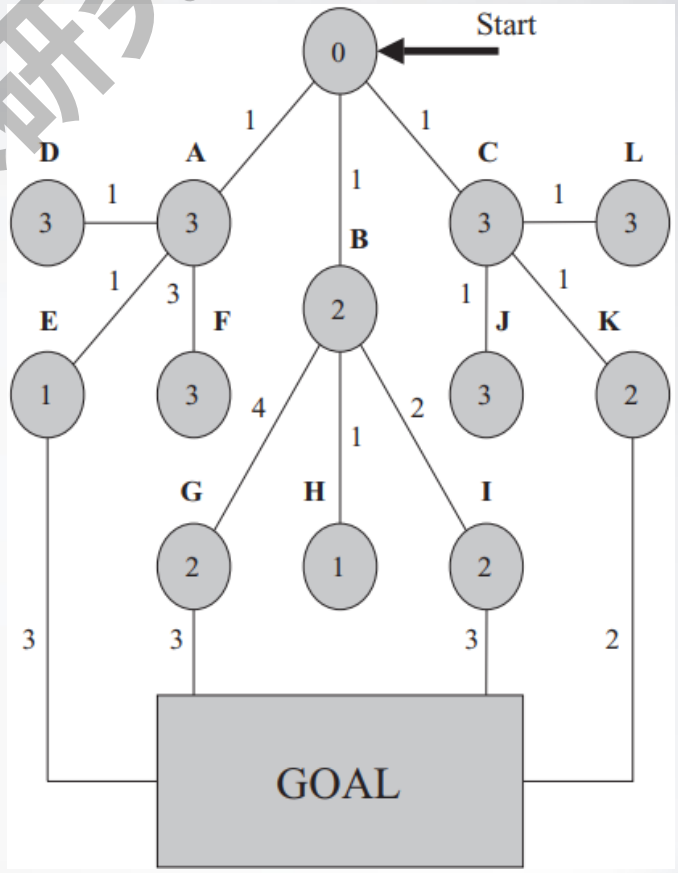
A*算法

已消耗路径代价与启发函数之和越小，优先级越高

① 将起始节点放入列表；

② 将起始节点取出，并将其相邻节点A、B、C放入列表，并按照优先级进行排序；

③ 将优先级最高的节点B取出，将与其相邻的节点G、H、I加入优先级列表，列表中的所有节点按优先级排序；

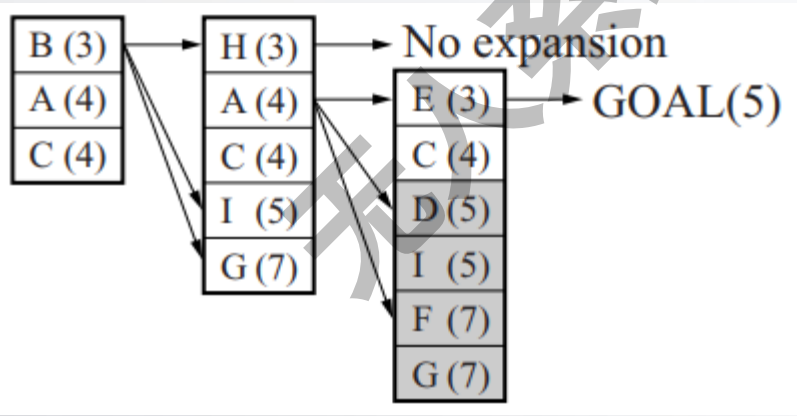


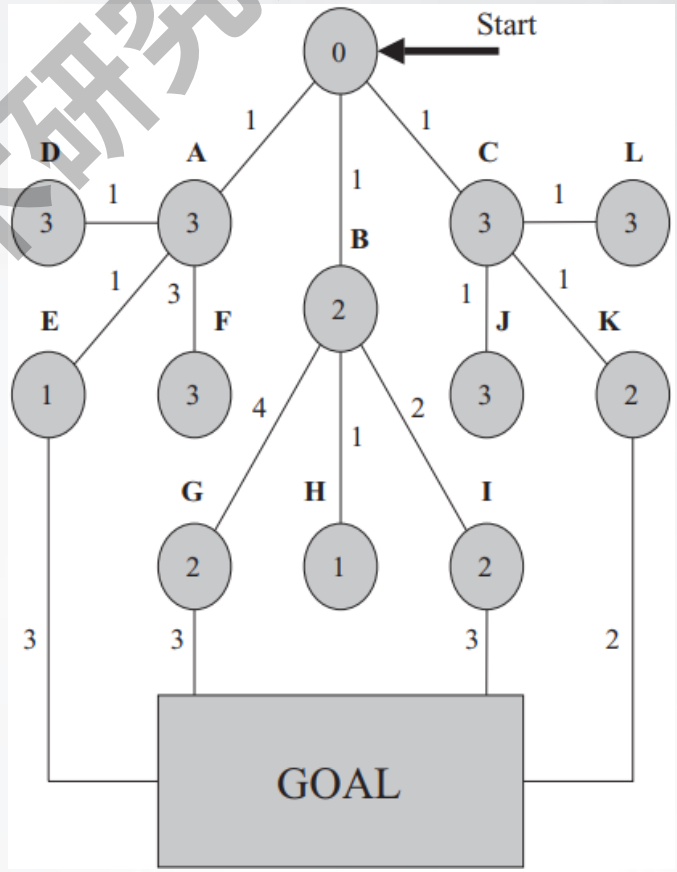
典型路径规划算法

A*算法

④ 按照步骤③，将优先级最高的节点H取出，加入相邻节点。但是节点H没有相邻节点，因此没有节点加入优先级列表；

⑤ 将列表中优先级最高的节点A取出，加入相邻节点，并按优先级排序；



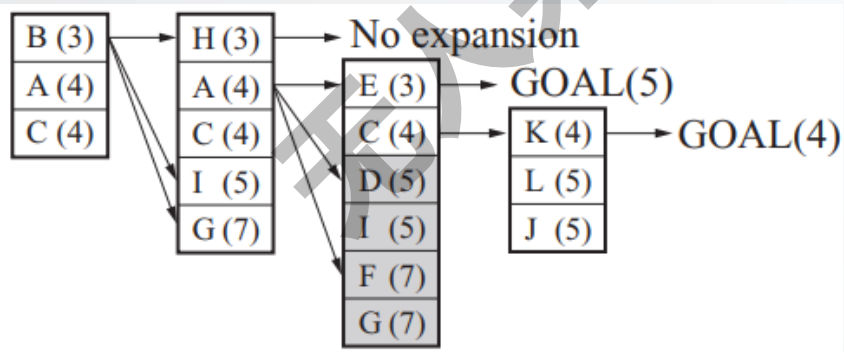


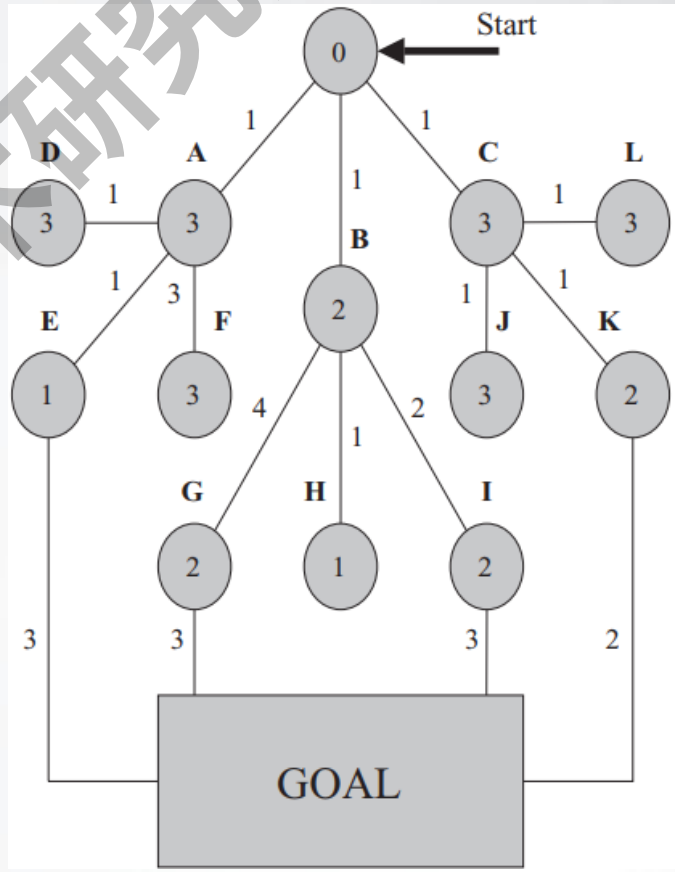
典型路径规划算法

A*算法

⑥ 取出节点E，将相邻节点加入优先级列表，得到由起始节点到目标节点代价函数为5的路径；

⑦ 但是，优先级列表中存在优先级大于5的节点，因此丢弃上一步骤得到的路径；按照类似方法对节点C进行扩展。





A*算法

为了描述A*算法，定义开集（open set）O，闭集（closed set）C。开集O为优先级列表，闭集C包含所有已经处理的节点。

- Star(n)表示与节点n相邻的节点的集合；
- $c(n_1,n_2)$表示连接节点n_1和节点n_2的路径代价；
- $g(n)$表示从节点n到起始节点q_{start}的路径总代价；
- $h(n)$为启发函数，输出从节点n到目标节点q_{goal}的期望代价的估计值；
- $f(n)=g(n)+h(n)$，从起始节点到目标节点的期望代价的估计值。

A*算法

Algorithm 1 A^* Algorithm

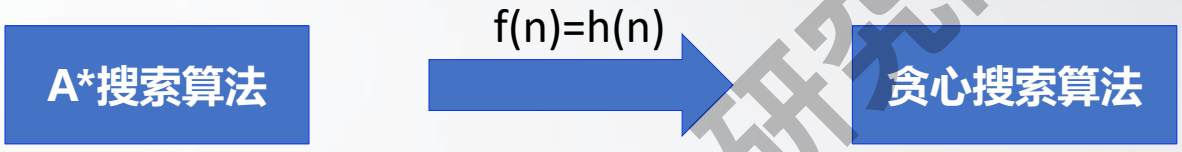
Input: A graph

Output: A path between start and goal nodes

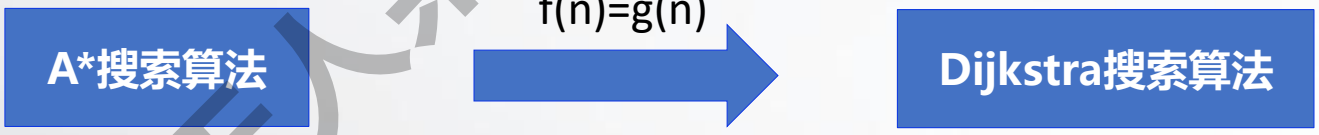
1: **repeat** 优先级最高
2: Pick n_{best} from O such that $f(n_{best}) \leq f(n), \forall n \in O$.
3: Remove n_{best} from O and add to C.
4: If $n_{best} = q_{goal}$, EXIT.
5: Expand n_{best}: for all $x \in \text{Star}(n_{best})$ that are not in C.
6: **if** $x \notin O$ **then**
7: add x to O.
8: **else if** $g(n_{best}) + c(n_{best}, x) < g(x)$ **then**
9: update x's backpointer to point to n_{best}
10: **end if**
11: **until** O is empty

典型路径规划算法

贪心搜索算法



迪杰斯特拉（Dijkstra）搜索算法



典型路径规划算法

算例

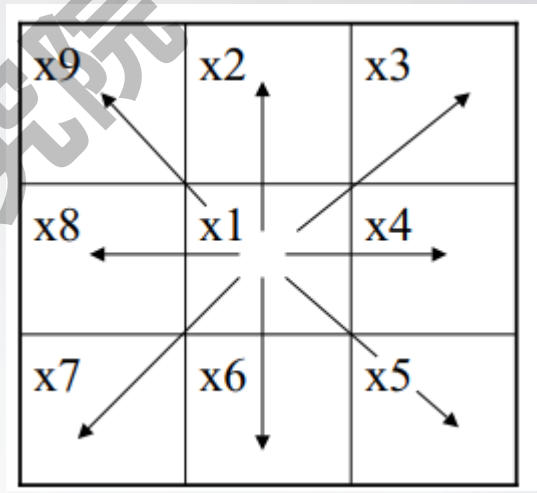
A*搜索

目标位置

6	h=6	h=5	h=4	h=3	h=2	h=1	h=0
5	h=6.4	h=5.4	h=4.4	h=3.4	h=2.4	h=1.4	h=1
4	h=6.8	h=5.8	h=4.8	h=3.8	h=2.8	h=2.4	h=2
3	h=7.2	h=6.2	h=5.2	h=4.2	h=3.8	h=3.4	h=3
2	h=7.6	h=6.6	h=5.6	h=5.2	h=4.8	h=4.4	h=4
1	h=8.0	h=7.0	h=6.6	h=6.2	h=5.8	h=5.4	h=5
	1	2	3	4	5	6	7

起始位置

障碍



$c(x1, x2) = 1$
$c(x1, x9) = 1.4$

$c(x1, x8) = 10000$, if x8 is in obstacle, x1 is a free cell

$c(x1,x9) = 10000.4$, if x9 is in obstacle, x1 is a free cell

典型路径规划算法

算例

6	**h=6**	h=5	h=4	**h=3**	**h=2** f=9.0 b=(6,5)	**h=1** f=7.6 b=(6,5)	h=0
5	**h=6.4**	h=5.4	h=4.4	**h=3.4** f=9.0 b=(5,4)	**h=2.4** f=7.6 b=(5,4)	**h=1.4** f=7.0 b=(5,4)	**h=1** f=7.6 b=(6,5)
4	**h=6.8**	h=5.8	h=4.8	**h=3.8** f=7.6 b=(4,3)	**h=2.8** f=7.0 b=(4,3)	**h=2.4** f=7.6 b=(5,4)	**h=2** f=9.0 b=(6,5)
3	**h=7.2**	h=6.2	h=5.2	h=4.2 f=7.0 b=(3,2)	**h=3.8** f=7.6 b=(4,3)	**h=3.4** f=9.0 b=(5,4)	**h=3**
2	h=7.6 f=9.0 b=(2,1)	h=6.6 f=7.6 b=(2,1)	h=5.6 f=7.0 b=(2,1)	h=5.2	**h=4.8** f=9.0 b=(4,3)	**h=4.4**	**h=4**
1	h=8.0 f=9.0 b=(2,1)	h=7.0	h=6.6 f=7.6 b=(2,1)	h=6.2 f=9.0 b=(3,2)	**h=5.8**	**h=5.4**	**h=5**
	1	2	3	4	5	6	7

						(4,3)	7
			(3,2)	7		(3,1)	7.6
			(3,1)	7.6		(2,2)	7.6
			(2,2)	7.6		(4,1)	9
			(1,2)	9		(1,2)	9
(2,1)	7		(1,1)	9		(1,1)	9

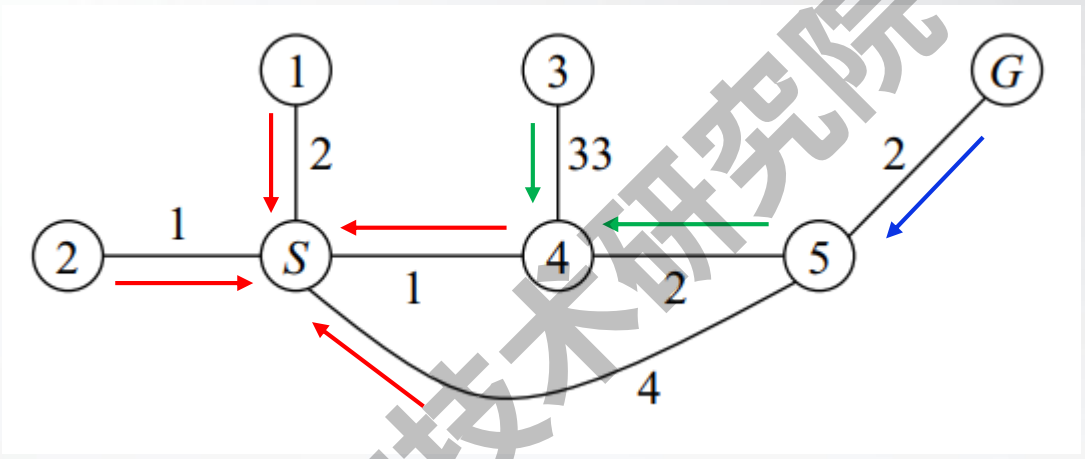
						(7,6)	7
						(6,6)	7.6
						(7,5)	7.6
						(5,5)	7.6
			(6,5)	7		(6,4)	7.6
			(5,5)	7.6		(5,3)	7.6
			(6,4)	7.6		(4,4)	7.6
			(5,3)	7.6		(3,1)	7.6
(5,4)	7		(4,4)	7.6		(2,2)	7.6
(5,3)	7.6		(3,1)	7.6		(4,1)	9
(4,4)	7.6		(2,2)	7.6		(1,2)	9
(3,1)	7.6		(4,1)	9		(1,1)	9
(2,2)	7.6		(1,2)	9		(5,2)	9
(4,1)	9		(1,1)	9		(6,3)	9
(1,2)	9		(5,2)	9		(4,5)	9
(1,1)	9		(6,3)	9		(7,4)	9
(5,2)	9		(4,5)	9		(5,6)	9

(2,1) → (3,2) → (4,3) → (5,4) → (6,5) → (7,6)

典型路径规划算法

迪杰斯特拉算法

算例



① O={S}

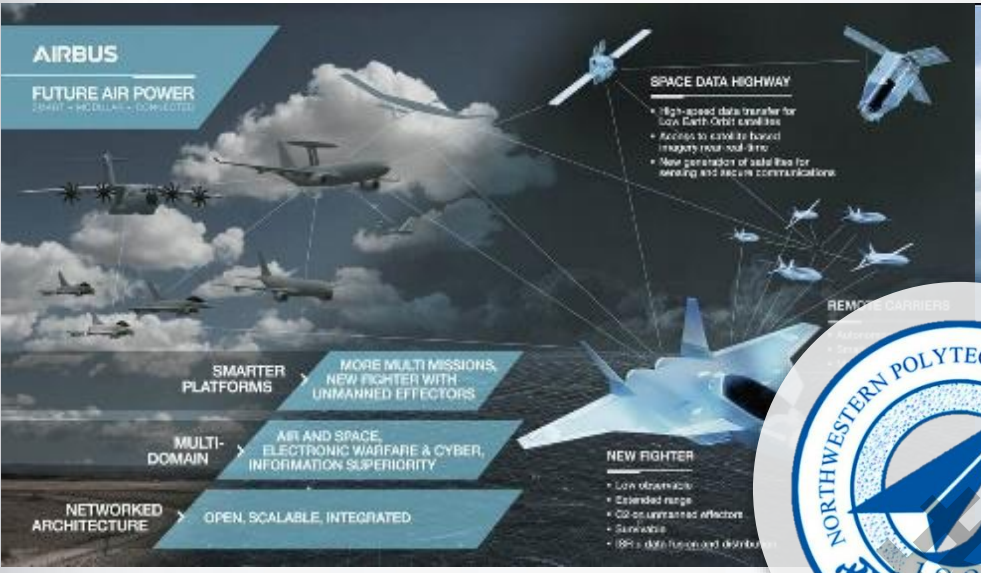
② O={2(1), 4(1), 1(2), 5(4)}； C={S} (1,2,3,5 all point back to S)

③ O={4(1), 1(2), 5(4)}; C={S, 2} (there are not adjacent nodes not in C)

④ O={1(2), 5(3), 3(34)}; C={S, 2, 4} (1,2,4 point to S; 5 point to 4)

⑤ O={5(3), 3(34)}; C={S, 2, 4, 1}

⑥ O={G(5), 3(34)}; C={S, 2, 4, 5} (goal points to 5 which points to 4 which points to S)



课 程 结 束 ， 欢 迎 提 问

THANK YOU FOR WATCHING