



中国科学技术大学
University of Science and Technology of China

人工智能讲义

启发式搜索

March 20, 2018

ustc





- 1 启发式搜索基础
- 2 启发式函数的例子
- 3 启发式函数的设计
- 4 设计启发式函数的例子: A^* 算法





- 1 启发式搜索基础
- 2 启发式函数的例子
- 3 启发式函数的设计
- 4 设计启发式函数的例子: A^* 算法



搜索算法框架改动为启发式搜索算法

Input: G : 状态图; s_0 : 初态;

Output: $path$: 代表解的路径

$path \leftarrow (s_0), FRINGE \leftarrow \phi$ /* 初始化 */;

if ($GOAL(s_0) = T$ **then**

return $path = (s_0)$;

end

$INSERT(s_0, FRINGE)$;

while T **do**

if $empty(FRINGE) == T$ **then**

return failure /* 返回 failure, 表示无解 */;

end

$N \leftarrow REMOVE(FRINGE)$ /* 将未扩展节点中队头节点从队列移除到 N */;

$s \leftarrow STATE(N)$ /* 从节点 N 恢复为状态 s */;

 update $path$;

foreach s' in $successors(s)$ **do**

 为 s' 创建 N 的新子节点 N' ;

if $GOAL(s') = T$ **then**

return ($path, s'$) /* 找到目标节点, 返回解 */;

end

$INSERT(s', FRINGE)$;

end

end

FRINGE 的优先级次序, 表示了搜索的“策略”

修改该函数的功能, 使之完成
FRINGE 的优先级排序

最佳优先搜索：从 FRINGE 中选择最佳节点进行扩展

- 何为最佳节点？
- 如何获得最佳节点？

评估节点的优劣：利用状态/节点信息或描述来度量

- 设计评估函数 f ，将搜索树的节点 N 映射为一个非负实数 $f(N)$ ，表示从初始状态到达某个节点的路径耗散（越小越好！）
- 将整个未扩展节点集合 FRINGE 按增序排列，排在最前面的称为“最佳”（best），优先进行扩展（first），Best-first search
- 若两个节点 f 值相等，则可任意指定其次序，或者添加其他的信息进行进一步排序

注意“最佳”的概念，并非指最后获得的解是最佳的

- 局部贪婪的
- 什么时候能获得最优解？



目前为止，学习的路线/思路

- 搜索问题求解 \Rightarrow 算法框架 \Rightarrow 搜索策略 \Rightarrow 评估函数 f 的设计

设计节点评估函数 f 的两种方法：

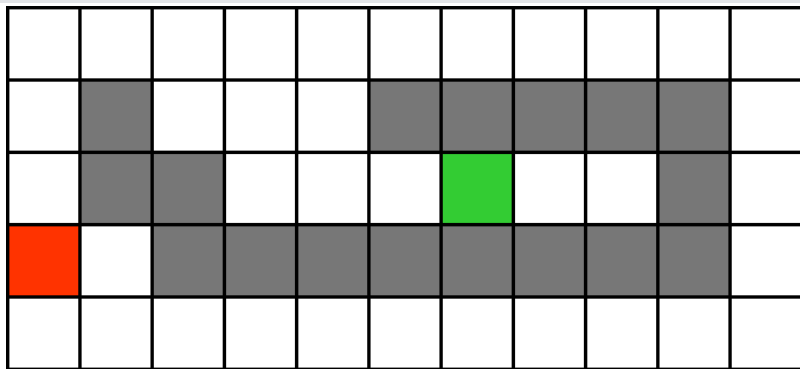
- $f(N) = g(N) + h(N)$ ，完整解的路径耗散，对应 A^* 算法
- $f(N) = h(N)$ ，从当前节点到目标节点的路径耗散，对应 贪婪算法

符号解释说明

- N ：当前待判决/估计/评价的节点
- $g(N)$ ：从初始节点到 N 的路径耗散
- $h(N)$ ：从 N 到目标节点的路径耗散，就是所谓的 启发式函数，估计值

函数 f 的形式因具体问题而异！





机器人导航：问题描述

- 灰格子表示障碍物；
- 红格子表示初态，绿格子表示终态。
- 寻找从绿格子到红格子的路径（路径规划问题）

8	7	6	5	4	3	2	3	4	5	6
7		5	4	3						5
6			3	2	1	0	1	2		4
7	6									5
8	7	6	5	4	3	2	3	4	5	6

机器人导航：问题描述

- $f(N) = h(N)$
- $h(N)$ = Manhattan distance to the goal
- 格子中的数字标明了 $f(N)$ 的值

8+3	7+4	6+3	5+6	4+7	3+8	2+9	3+10	4	5	6
7+2		5+6	4+7	3+8						5
6+1			3	2+9	1+10	0+11	1	2		4
7+0	6+1									5
8+1	7+2	6+3	5+4	4+5	3+6	2+7	3+8	4	5	6

机器人导航：问题描述

- $f(N) = g(N) + h(N)$
- $h(N)$ = Manhattan distance to the goal
- 格子中的数字标明了 $f(N)$ 的值

5		8
4	2	1
7	3	6

N

1	2	3
4	5	6
7	8	

Goal

$h_1(N)$ = 错误放置的格子数 = 6

节点评估函数 f 的设计，核心在于启发式函数 h 的设计

- 一般情形下，启发式函数值由当前节点和目标节点二者所确定
- $h(N)$ 应保证的性质： $h(N) \geq 0$, $h(Goal) = 0$, $h(N)$ 越小，离目标越近
- 如上图所示 h_1



5		8
4	2	1
7	3	6

N

1	2	3
4	5	6
7	8	

Goal

数码问题，如上图所示，三种启发式函数的设计如下：

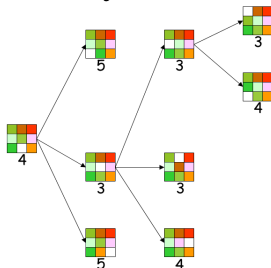
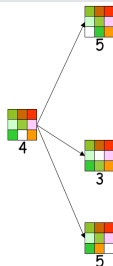
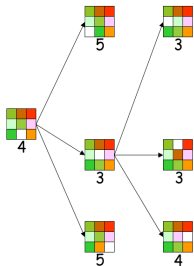
- $h_1(N)$ = 错误放置的格子数 = 6
- $h_2(N)$ = 所有数字到其正确位置的 Manhattan 距离之和 = $2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$
- $h_3(N)$ = 逆序数之和 = $n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6 = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16$



8 数码问题的搜索过程



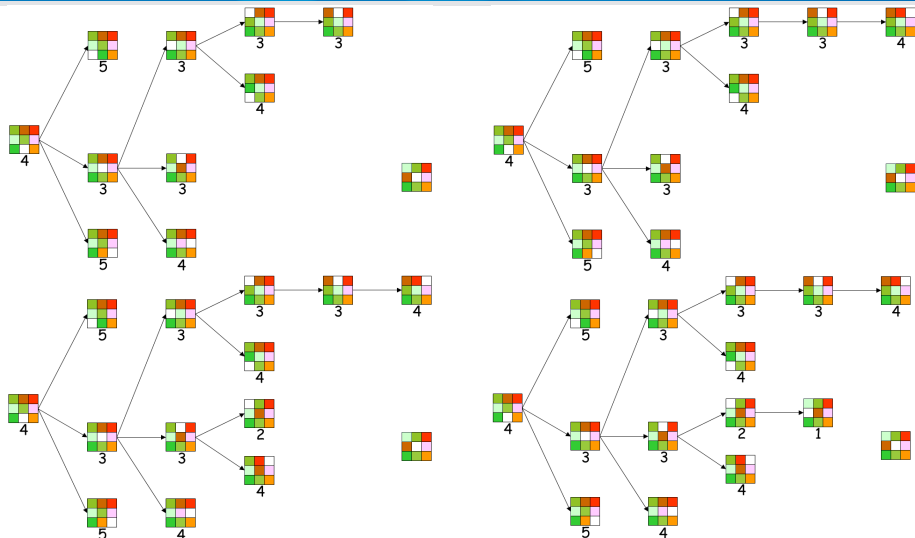
中国科学技术大学
University of Science and Technology of China


$$f(N) = h(N) = \text{错误放置的格子数}$$

ustc



8 数码问题的搜索过程

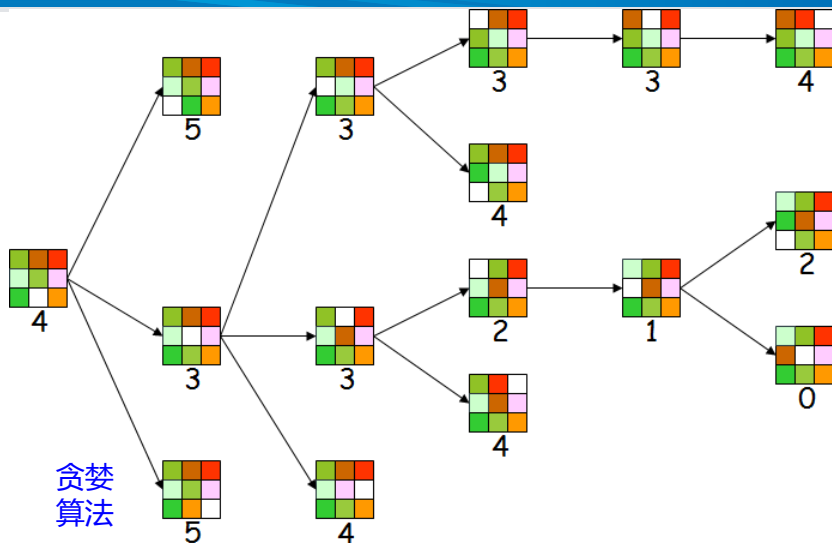


$f(N) = h(N) =$ 错误放置的格子数

ustc



启发式函数：应用到搜索过程



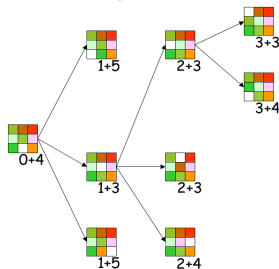
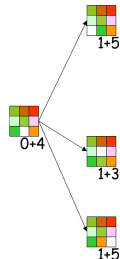
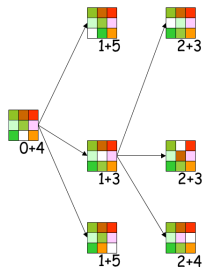
$h_1(N) =$ 错误放置的格子数

$f(N) =$

$u s t c$

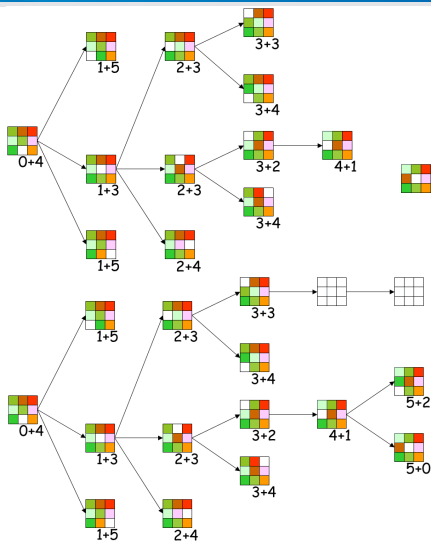


8 数码问题的搜索过程



$$f(N) = g(N) + h(N), h(N) = \text{错误放置的格子数}$$





ustc





$f(N) =$

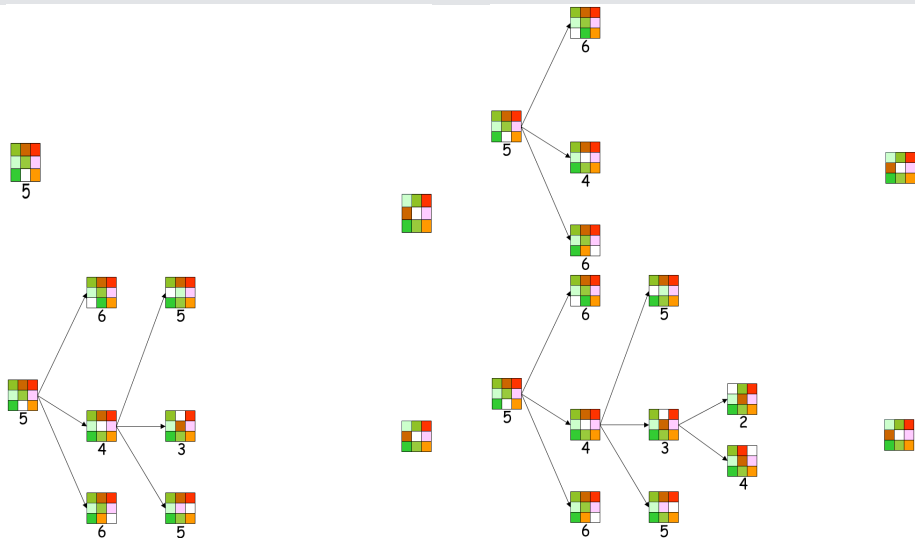
u s t c



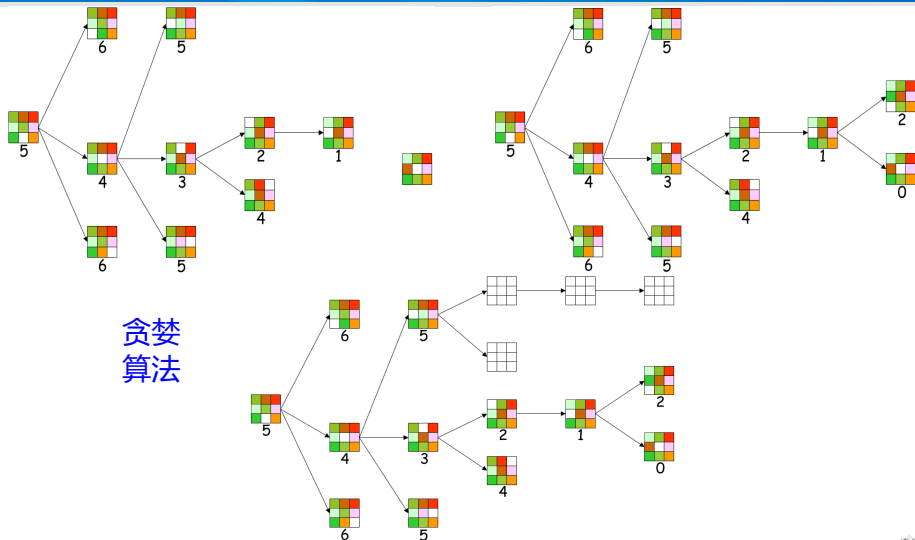
8 数码问题的搜索过程



中国科学技术大学
University of Science and Technology of China

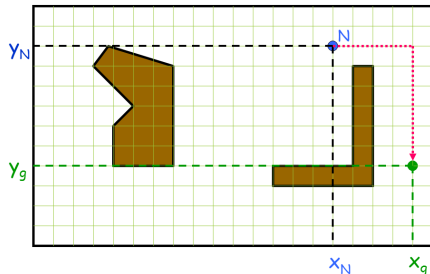
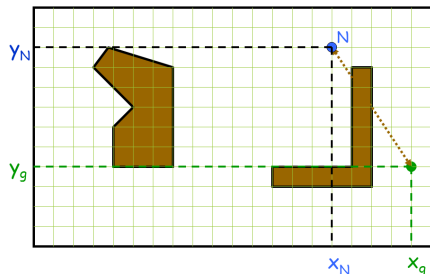

$$f(N) = h(N) = \text{所有数字到其正确位置的 Manhattan 距离之和}$$


8 数码问题的搜索过程



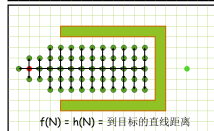
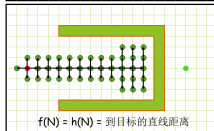
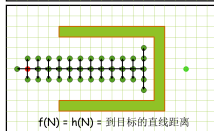
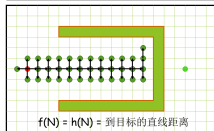
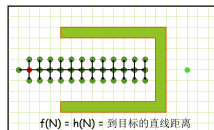
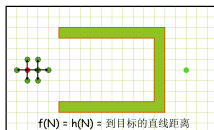
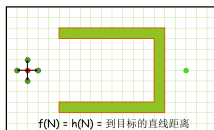
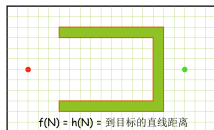
$f(N) = h(N) =$ 所有数字到其正确位置的 Manhattan 距离之和





路径规划问题的启发式函数：如上图， g 为目标状态

- Euclidean distance: $h_1(N) = \sqrt{(x_N - x_g)^2 + (y_N - y_g)^2}$
- Manhattan distance: $h_2(N) = |x_N - x_g| + |y_N - y_g|$



启发式函数： $f(N) = h(N) =$ 到目标的直线距离

- 最佳优先搜索存在的局部最小问题
- 引导搜索进入错误的方向
- 降低了搜索效率

基本要求

- $h(N) \geq 0, h(Goal) = 0, h(N)$ 越小, 离目标越近
- 更多的?

可采纳的/admissible 启发式函数

- 假设 $h^*(N)$ 是节点 N 到目标节点的实际最优路径耗散
- 启发式函数 $h(N)$ 是“可采纳的”, 当且仅当 $0 \leq h(N) \leq h^*(N)$
- 总是“**乐观地**”估计路径耗散!!!

定义“可采纳的”启发式函数的理由和方法

- 松弛问题: 放宽问题的限制 (约束), 减少行动限制, 获得更好的解, 原问题的解仍在松弛问题的可行域中。
- 用松弛问题来构造启发式函数是最常用的技术
- 加强问题: 增加问题限制, 把原来可行的路径 (解) 给剪枝掉, 可能无解;

ustc



5		8
4	2	1
7	3	6

N

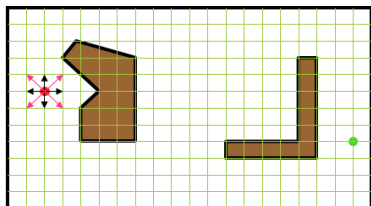
1	2	3
4	5	6
7	8	

Goal

数码问题，如上图所示，三种启发式函数的设计如下：

- $h_1(N)$ = 错误放置的格子数 = 6 **可采纳的**
- $h_2(N)$ = 所有数字到其正确位置的 Manhattan 距离之和 = $2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$ **可采纳的**
- $h_3(N)$ = 逆序数之和 = $n_5 + n_8 + n_4 + n_2 + n_1 + n_7 + n_3 + n_6 = 4 + 6 + 3 + 1 + 0 + 2 + 0 + 0 = 16$ **不可采纳的。反证法：找一个违背可采纳定义的状态**





$$h^*(I) = 4\sqrt{2}$$
$$h_2(I) = 8$$



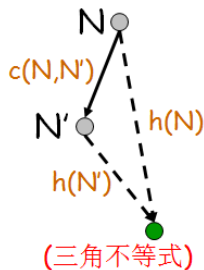
允许沿对角线移动，
则是不可采纳的

水平/垂直移动一格，路径耗散为1
对角线移动一格，路径耗散为 $\sqrt{2}$

路径规划问题，如上图所示，启发式函数设计如下：

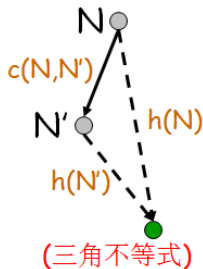
- Euclidean distance: $h_1(N) = \sqrt{(x_N - x_g)^2 + (y_n - y_g)^2}$, 可采纳的
- Manhattan distance: $h_2(N) = |x_N - x_g| + |y_n - y_g|$, 不允许沿对角线移动，则是可采纳的；否则是不可采纳的





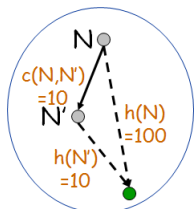
对一个可采纳的启发式函数 h

- 若它对所有节点都满足三角不等式, 即 $h(N) \leq c(N, N') + h(N')$, 其中 N' 是 N 的后继节点, 则 h 是一致的或单调的。
- $c(N, N')$ 是节点 N 到 N' 的单步路径耗散。



性质和理解：一致的/单调的启发式函数

- $h(N) \leq h^*(N) \leq c(N, N') + h^*(N')$
- $h(N) - c(N, N') \leq h^*(N')$
- $h(N) - c(N, N') \leq h(N') \leq h^*(N')$
- 随着搜索的深度越来越深，对深处节点估计的启发式函数值越来越准确!!!



不三角不等式

不满足三角不等式，不是一致的启发式函数

5		8
4	2	1
7	3	6

N

1	2	3
4	5	6
7	8	

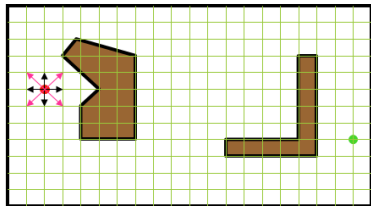
Goal

数码问题

- $h_1(N)$ = 错误放置的格子数 = 6, 可采纳的, 一致的
- $h_2(N)$ = 所有数字到其正确位置的 Manhattan 距离之和 = $2 + 3 + 0 + 1 + 3 + 0 + 3 + 1 = 13$, 可采纳的, 一致的

ustc





$$h^*(I) = 4\sqrt{2}$$
$$h_2(I) = 8$$



允许沿对角线移动，
则是不可采纳的

水平/垂直移动一格，路径耗散为1
对角线移动一格，路径耗散为 $\sqrt{2}$

路径规划问题，如上图所示，启发式函数设计如下：

- Euclidean distance: $h_1(N) = \sqrt{(x_N - x_g)^2 + (y_n - y_g)^2}$, 可采纳的，一致的
- Manhattan distance: $h_2(N) = |x_N - x_g| + |y_n - y_g|$, 不允许沿对角线移动，则是可采纳的，一致的；否则是不可采纳的，不一致的

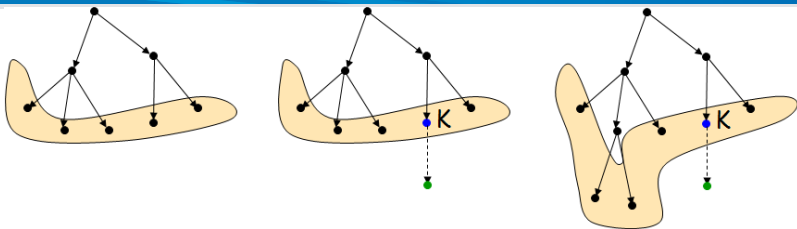


AI 中最著名的搜索算法

- $f(N) = g(N) + h(N)$, 其中
- $g(N)$ = 从初始节点到 N 的路径耗散
- $h(N)$: 从 N 到目标节点的路径耗散, 可采纳的启发式函数

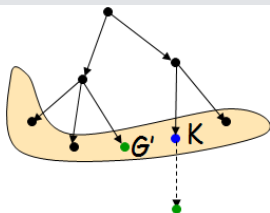
从 A* 算法中的启发式函数设计, 学习如何设计启发式函数





证明：完备性（若 A* 算法结束，问题有解，则一定会返回一个解）

- 如图所示
- 未扩展节点集合 FRINGE 中任意节点 N 满足 $f(N) = g(N) + h(N) \geq g(N) \geq d(N) \times \epsilon$ ，其中 $d(N)$ 是节点 N 的深度
- 只要 A* 算法没结束，在 FRINGE 中至少有一个节点 K 属于解路径
- 节点扩展会使得路径变长， K 将最终被扩展，除非在这之前找到一个到达目标节点的其他路径。
- 条件：状态被重复访问时，节点不被丢弃



证明: 最优性 (A* 选择某个目标节点扩展, 则到此目标节点的路径一定是最优的)

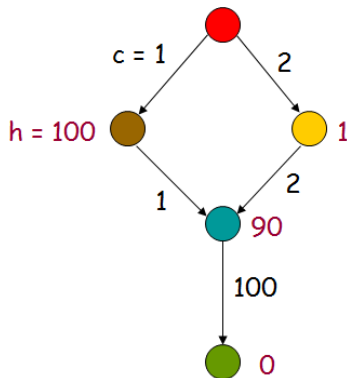
- 如上图所示
- C^* = 最优解的路径耗散
- G' : 未扩展节点集合中 “非最佳” 目标节点 (为什么只要考虑目标节点?) $f(G') = g(G') + h(G') = g(G') > C^*$
- 未扩展节点集合中的节点 K 位于最优解的路径上: $f(K) = g(K) + h(K) \leq C^*$
- 所以, G' 不会被选择扩展



问题无解时

- 若状态空间无限或允许状态重复访问，则 A^* 算法的搜索不会停止；
- 求解实际问题时，通常给一个停止时间/time limit，当停止时间达到时，算法停止运行；
- 因停止时间耗完而停止的 A^* 算法，无法判断其是否有解或无解，也无法说明更多的搜索时间可以得到解。

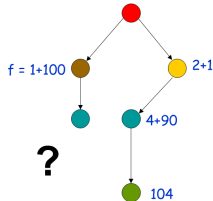
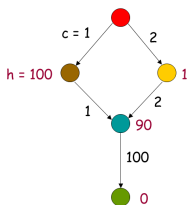
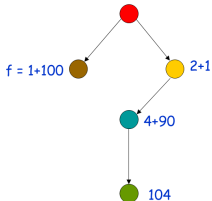
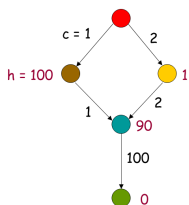
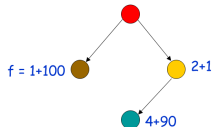
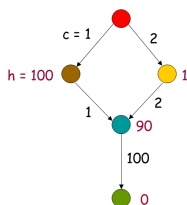
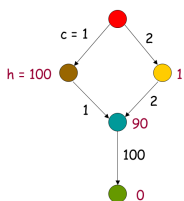




状态图说明

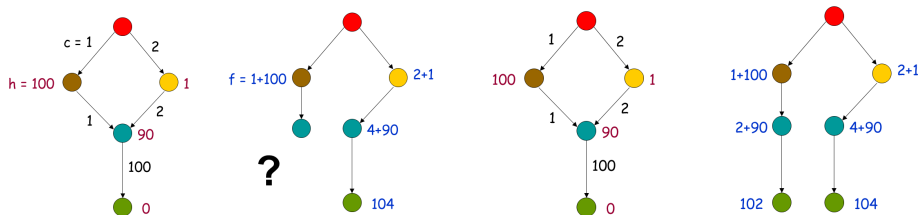
- 如左图
- h 是一个可采纳的启发式函数, 每个节点的 h 值 (估计值) 标记在节点附近;
- 每条边标记了路径耗散 c 值;
- 从红色状态节点找一条路径到绿色状态节点。





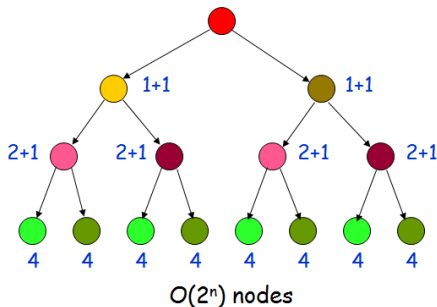
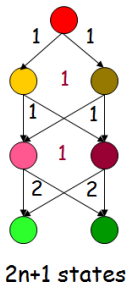
解释说明

- 蓝色节点被访问后，若丢弃该状态，会发生什么事情？
- 次优解（红-黄-蓝-绿），路径耗散 104



解释说明

- 若不丢弃访问过的状态，运行状态重复访问
- 我们可以得到最优解（红-褐-蓝-绿），路径耗散 102
- WHY? 付出什么代价了？



图搜索：允许状态重复访问

- 保证获得最优解
- 无解时可能永远停不下来

树搜索：避免状态重复访问

- 状态有限时，是完备的
- 但不保证解是最优的





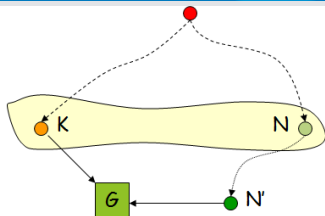
一个重要的事实

- 丢弃重复访问状态的节点，如果到该节点的新路径的耗散 $g(N)$ 比以前（访问该状态）的路径耗散更大

一致的/单调的启发式函数

- 来自上述重要事实
- 虽然采用一致启发式函数的搜索树的节点仍可能是指数增加的
- 但实际应用中，能有效地避免许多重复访问状态的节点扩展。





从第二次开始重复扩展的状态总是不如第一次扩展

证明：对一致的 h ，A* 算法扩展一个状态节点时，到该状态的路径一定是最优的

- 一致性意味着单调性：(考虑 N 及其后继 N')

$$f(N) = g(N) + h(N) \leq g(N) + c(N, N') + h(N') = f(N')$$
- 如上图， K 被选择进行扩展，我们要证明此时路径到 K 是最优的，即 $g(K)$ 最小。考虑在未扩展节点集合中存在一个其他节点 N ，经 N 到 K ， K 的状态此时扩展为节点 N' ，存在一条路径，那么有：(N' 和 K 是同一个状态不同的节点表示) $f(N') \geq f(N) \geq f(K)$ 以及 $h(N') = h(K)$ 所以， $g(N') \geq g(K)$



处理状态的重复访问

- 节点被扩展，则状态进入 CLOSED 表
- 当一个新节点 N 产生了
 - 若 N 表示的状态在 CLOSED 表中，则丢弃节点 N
 - 若 N 表示的状态不在 CLOSED 表中，但是在未扩展节点集合 FRINGE 中（不妨设为 N' ），则去掉 $f(N), f(N)$ 中较大的节点，等价于去掉 $g(N)$ 和 $g(N)$ 中较大的节点





特殊的一致启发式函数: $h \equiv 0$

- 一致的, 当然也是可采纳的
- 等价于单步路径耗散相同
- A^* 退化为宽度优先搜索, 代价一致搜索

为 A^* 设计什么样的启发式函数更好?



5		8
4	2	1
7	3	6

STATE(N)

1	2	3
4	5	6
7	8	

Goal state

可采纳的启发式函数 h_1, h_2 对任何节点, $h_1 \leq h_2$, 则称 h_2 比 h_1 准确

- 如上图
- $h_1(N)$ = 错误放置的格子数目
- $h_2(N)$ = 每个数字到对应目标位置的 Manhattan 距离
- h_2 比 h_1 更准确/精确/更富含信息



解释说明

- 当解存在时，较精确启发式函数导致的被扩展节点集合包括在“不精确”启发式函数导致的被扩展节点集合中，除了 f 值相同且等于最优解的路径耗散的那些节点。

证明：精确的启发式函数扩展的节点更少一些

- 任何 $f(N) < C^*$ 的节点都将会被扩展（在获得最优解之前）。可用绘制等 $g(N)$ 值线的方式来逐步扩展初始状态为核心的等值线系统，在最优解路径耗散 C^* 围成的等值线包括了所有 $f(N) < C^*$ 的节点。（完备性容易由此被证明）
- 所以由 $h_1(N) \leq h_2(N)$ 可知，在找到最优解之前， h_2 扩展的节点， h_1 都会进行扩展
- 除了最后达到最优解时，可能获得的是不同的最优目标节点！

不精确的启发式函数会访问更多的节点！



d	搜索代价			有效分支因子		
	IDS	$A^*(h_1)$	$A^*(h_2)$	IDS	$A^*(h_1)$	$A^*(h_2)$
2	10	6	6	2.45	1.79	1.79
4	112	13	12	2.87	1.48	1.45
6	680	20	18	2.73	1.34	1.30
8	6384	39	25	2.80	1.33	1.24
10	47127	93	39	2.79	1.38	1.22
12	3644035	227	73	2.78	1.42	1.24
14	-	539	113	-	1.44	1.23
16	-	1301	211	-	1.45	1.25
18	-	3056	363	-	1.46	1.26
20	-	7276	676	-	1.47	1.27
22	-	18094	1219	-	1.48	1.28
24	-	39135	1641	-	1.48	1.26

有效分支因子 b^*

- b^* 可以度量启发式函数的有效性
- 通过隐函数来定义: $n = b^* + (b^*)^2 + \dots + (b^*)^d$, 其中 n 是被扩展的节点数目, d 是解的深度
- 如上图, 8 数码问题, 考虑无信息的迭代深入搜索、 h_1 和 h_2 , 随机产生 1200 个 instances
- 被扩展节点数目 n
 - $d = 12, h_1 \rightarrow 227, h_2 \rightarrow 73$
 - $d = 24, h_1 \rightarrow 39135, h_2 \rightarrow 1641$

经验总结

- 基本思路：求解原问题的松弛问题，获得灵感
- 例如 8 数码问题
 - h_1 的设计：假设错了数字，可以一次就放到正确位置，忽略其它所有因素
 - h_2 的设计：假设数字可以水平和垂直任意移动，忽略移动目标位置是否被占据
 - 更复杂有效的启发式函数设计：假设数字可以水平和垂直移动，使得 1, 2, 3, 4 移动到目标位置，忽略 5, 6, 7, 8 的阻挡，计算移动次数 d_1 ；使得 5, 6, 7, 8 移动到目标位置，忽略 1, 2, 3, 4 的阻挡，计算移动次数 d_2 ； $h = d_1 + d_2$ 。（产生大约 3024 个状态节点）
- 其他方法：从经验中学习，归纳学习，不同启发式函数的集成等



目标：精确的、一致的启发式函数

- 保证了完备性、最优性，不需要重复访问同一个状态
- 实际上，问题并没有解决。问题规模大时，因为时空要求都是解长度的指数函数
- 算法时间限制及其设置
- 其它实用的，不可采纳的启发式函数，不能保证最优性和完备性，但是能快速获得最优解或近似最优解



迭代深入 A* 算法: IDA*

- 思想: 设置 f 的阈值, 超过 f 的阈值, 节点不再扩展; 迭代执行 A*, 降低 A* 算法对内存的需求
- 要求: 一致的启发式函数

算法思想

- 初始化 f 的阈值 t 为 $f(N_0)$
- 重复执行下述两步:
 - 执行深度优先搜索, 扩展 $f(N) \leq t$ 的节点 N
 - 重新设置 t 为未扩展节点中 f 的最小值

优点

- 完备的、最优的
- 比 A^* 要求的内存少
- 避免了未扩展节点集合的排序开销

不足

- 无法充分利用内存，用的内存太少，两次迭代之间只保留阈值 t
- 无法避免重复访问不在路径上的节点

IDA* 改进为 SMA*

- IDA* 的改进：把内存用光，不能保存节点了，丢掉保存的一个高耗散，最旧的节点，再插入新的节点，这个算法称为：SMA*

