西安电子科技大学 XIDIAN UNIVERSITY

人工智能概论

刘若辰
西安电子科技大学 人工智能学院

第二章 状态空间表示以及问题求解

- 2.1 问题求解与状态空间表示法
- 2.2 图搜索策略
- 2.3 盲目式搜索
- 2.4 启发式搜索

2.4 启发式搜索

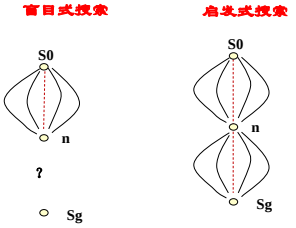
1. A算法
2. A*算法

引言

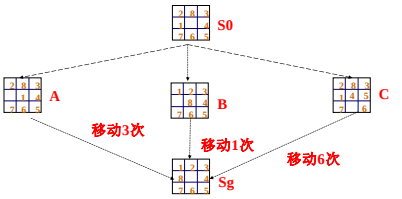
➢ 盲目式搜索的不足：

- 效率低，耗费过多的计算空间与时间
- 可能带来组合爆炸
- 没有利用搜索中的信息指导搜索

引言



引言



启发式信息

➢ 启发性信息是指那种与具体问题求解过程有关的，并可指导搜索过程朝着最有希望方向前进的控制信息。

➢ 启发式就是要猜测：

- 从节点n开始，找到最优解的可能性有多大?
- 从起始节点开始，经过节点n，到达目标节点的最佳路径的费用是多少?

启发式搜索

➢ 启发式搜索是利用与问题有关的启发性信息，并以这些启发性信息指导的搜索的问题求解过程。

- 需定义一个评价函数，对当前的搜索状态进行评估，找出一个最有希望的节点来扩展。
- 重排OPEN表，选择最有希望的节点加以扩展

➢ 种类：A、A*算法等

A算法

➢ 估价函数 $f(n)$：估算节点“希望”程度的量度

$$f(n) = g(n) + h(n)$$

$g(n)$：从起始状态到当前状态n的代价

$h(n)$：从当前状态到目标状态的估计代价（启发函数）

A算法
- 局部择优（尼尔逊，1964）
- 有序搜索（拉斐尔，1967）

局部择优搜索(瞎子爬山法)

➢ **搜索过程**如下：

- (1) 把初始节点S_0放入OPEN表，计算$f(S_0)$；
- (2) 如果OPEN表为空,则问题无解，退出；
- (3) 把OPEN表的第一个节点(记为节点n)取出，放入CLOSED表；
- (4) 考查节点n是否为目标节点，若是，则求得了问题的解，退出；
- (5) 若节点n不可扩展，则转第(2)步；
- (6) 扩展节点n，用估价函数$f(x)$计算每个子节点的估价值，并按估价值从小到大的顺序依次放入OPEN表的首部，并为每一个子节点都配置指向父节点的指针，转第(2)步。

有序搜索

➢ **有序搜索**，也称最好优先搜索/全局择优，选择OPEN表上具有最小f值的节点作为下一个要扩展的节点。

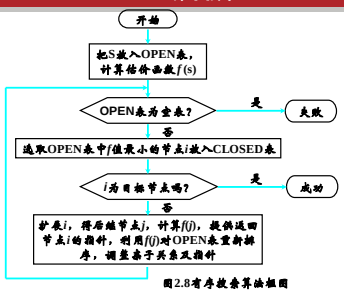


图2.8有序搜索算法框图

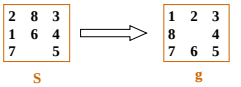
有序搜索的一个例子

定义**评价函数**：

➢ $f(n) = g(n) + h(n)$

➢ $g(n)$为从初始节点到当前节点的路径长度（**层数**）

➢ $h(n)$为当前节点“**不在位**”的将牌数

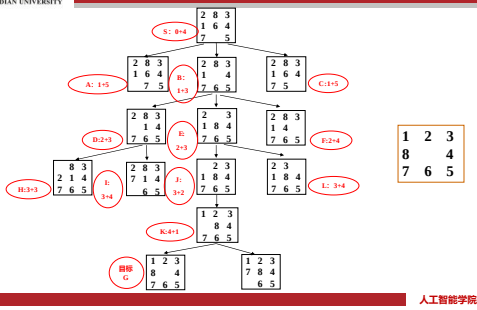


$g(S)=0$

$h(S)=4$

$f(S)=0+4=4$

有序搜索



1	2	3
8		4
7	6	5

有序搜索

2	8	3
1	6	4
7		5

⇒

1	2	3
8		4
7	6	5

	宽度搜索	深度搜索（界为5）	A算法
扩展节点	26	18	6
生成节点	46	34	13

思考：如果把宽度优先、深度优先、等代价搜索方法作为有序搜索的特例，那么它们的 f 函数如何计算?

A*算法

- 1968年，彼得.哈特对A算法进行了很小的修改，并证明了当估价函数满足一定的限制条件时，算法一定可以找到最优解。
- 隶属于A算法，其特点在于对估价函数的定义上

彼得.哈特

- A*算法的限制条件

$$f(n) = g(n) + h(n)$$

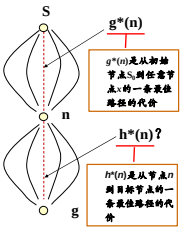
大于0　　要满足一定条件

A*算法

- 把OPEN表中的节点按估价函数的值从小到大进行排序

$$f(n) = g(n) + h(n)$$

- $g(n)$是对$g^*(n)$的估价，$g(n) >= g^*(n)$
- $h(n)$是$h^*(n)$的下界，即对所有n的，$h(n) <= h^*(n)$



A*算法

➢ **定义1**

在图搜索过程中，如果第8步的重排OPEN表是依据$f(n)=g(n)+h(n)$进行的，则称该过程为A算法。

➢ **定义2**

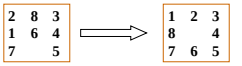
在A算法中，如果对所有的n存在$h(n)\le h^*(n)$，则称$h(n)$为$h^*(n)$的下界，它表示某种偏于保守的估计。

➢ **定义3**

采用$h^*(n)$的下界$h(n)$为启发函数的A算法称为A*算法。$h(n)=0$时，A*算法就变为等代价搜索算法。

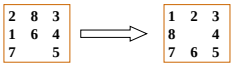
A*算法举例

➢ 对于如图所示的八数码问题，给出满足A*算法的启发式函数，并给出相应的搜索图。



S　　g

A*算法



S　　g

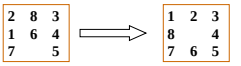
➢ **方案1**. $f(n)=g(n)+h_1(n)$,

$g(n)$：节点n的层数

$h_1(n)$：表示“不在位”的将牌数

显然$h_1(n)\le h^*(n)$

A*算法



S　　g

➢ **方案2**. $f(n)=g(n)+h_2(n)$,

$g(n)$：节点n在搜索树中的深度

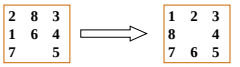
$h_2(n)$：节点n的每一数码与其目标位置之间的距离总和

A*算法

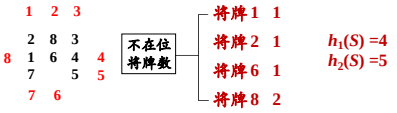
➢ 八数码问题

——$h_1(n)$ = “不在位”的将牌数

——$h_2(n)$ = 将牌“不在位”的距离和



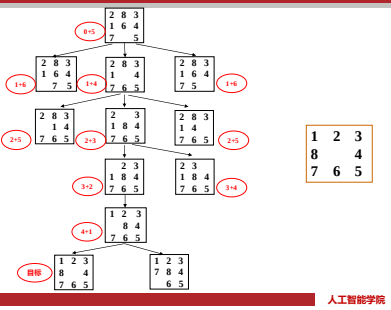
S　　g



$h_1(S)$ =4
$h_2(S)$ =5

显然 $0 \le h_1(n) \le h_2(n) \le h^*(n)$

A*算法



A*算法

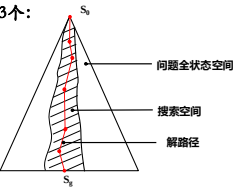
➢ 本题的启发函数的比较结果：

启发函数	$h(n)=0$	$h(n)=h_1(n)$	$h(n)=h_2(n)$
扩展节点	26	6	5
生成节点	46	13	11

➢ A*算法的搜索效率在很大程度上取决于$h(n)$，在满足$h(n)<=h^*(x)$的前提下，$h(n)$的值越大越好。

➢ 搜索方法好的标准，一般认为有3个：

- (1) 问题有解能否找到
- (2) 搜索空间小
- (3) 解最佳



➢ 状态空间的图搜索策略总结

状态空间搜索策略

- 盲目搜索
  - 宽度优先搜索
  - 深度优先搜索
  - 有界深度优先搜索
  - 等代价搜索

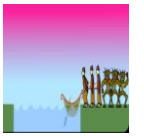
  搜索按规定的路线进行，不使用与问题有关的启发性信息，适用简单问题。

- 启发式搜索
  - 局部择优搜索
  - 有序搜索
  - A*算法

  使用与问题有关的启发性信息，指导搜索过程，可以求解结构复杂的问题。

练习一：修道士和野人问题

有三个传教士（m）和三个野人(c)过河，只有一条能装下两个人的船，在河的一方或者船上，如果野人的人数大于传教士的人数，那么传教士就会有危险.
你能不能提出一种安全的渡河方法呢?



约束条件:

① 任何时刻两岸、船上都必须满足传教士人数不少于野人数（m=0时除外，既没有传教士）:m≧c

② 船上人数限制在2以内: m+c≦2

解：m：左岸的修道士人数，

c：左岸的野人数，

b：左岸的船数

用三元组(ml, cl, bl)表示问题的左岸状态。

其中：0≦m≦3;0≦c≦3; b={0,1}

初始状态:（3,3,1），目标状态:(0, 0, 0)

状态空间：32 种状态

Pm：如果船在左岸(b=1)，从左岸划向右岸(P10:从左岸移一个传教士到右岸)

Qm：如果船在右岸(b=0)，从右岸划向左岸（Q01:从右岸移一个传教士到左岸）

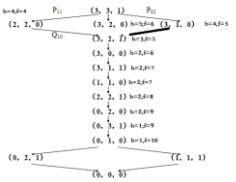
对A*算法，首先需要确定估价函数（还有其他定义？？？）。

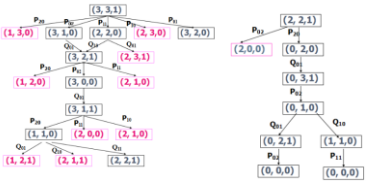
设: g(n)=d(n)，h(n)=m+c-2b，

f(n)=g(n)+h(n)=d(n)+m+c-2b

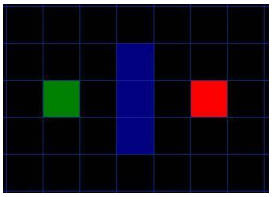
其中，d(n)为节点的深度。通过分析可知h(n)≤h*(n)，满足A*算法的限制条件。

M-C问题的搜索过程如下图所示。





练习二：迷宫寻路

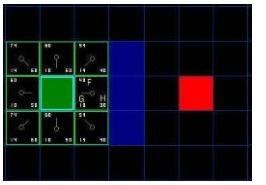


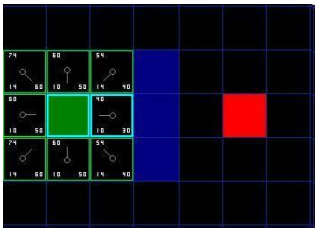
https://www.cnblogs.com/yanlingyin/archive/2012/01/15/2322640.html
https://blog.csdn.net/myjoying/article/details/7648247?utm_medium=distribute.pc_relevant.none-task-blog-title-1&spm=1001.2101.3001.4242

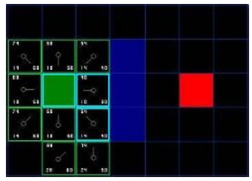
G: 沿路径从起点到当前点的移动耗费；

H:可以用不同的方法估算。这里使用的方法称为曼哈顿方法，它计算从当前格到目的格之间水平和垂直的方格的数量总和，忽略对角线方向。然后把结果乘以10。它忽略了一切障碍物。这是对剩余距离的一个估算，而非实际值。

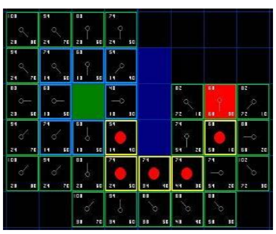
水平或者垂直移动的耗费为1，对角线方向耗费为1.4







此时，当我们检测与其相邻的方格时发现其正右方的方格是墙壁，忽略它。同样忽略其上方的方格。另外，我们忽略墙壁下方的方格，为什么呢？因为你无法不横切旁边的墙壁方格到达那里。你首先需要向下移动一格然后再移动到那里，以绕过墙壁方格。



1、将起点方格（节点）加入到closed链表中；

2、重复以下步骤：

a) 在open链表中找寻有最小F值的方格，将其标记为当前方格；

b) 将其加入到closed链表中；

c) 对其所有的8个相邻方格做如下操作：

- 如果该方格是不可通行的或者已经在closed链表中，则忽略他，否则，按以下步骤；
- 如果该方格不在open链表中，将其加入到open链表中，将当前所选的方格标记为其父方格，并计算F、G和H值；
- 如果该方格已经在open链表中，检测新路径到该方格的G值是否更优，更小的G值表明路径更好，如果是，则将该方格的父方格标记为所选方格，并重新计算F和G值。如果你的open链表是按F值排序的，此时就需要重新对链表进行排序；

d) 当以下情况发生则停止：

- 目标方格被加入到了closed链表中，此时路径被找到（注意以下的说明），或者
- 没有找到目标方格，并且open链表为空。此时，没有可行路径；

3、保存路径。从终点回溯，沿着箭头所指方向移动到父方格直到回到起点方格，则得到了路径。

谢　谢！