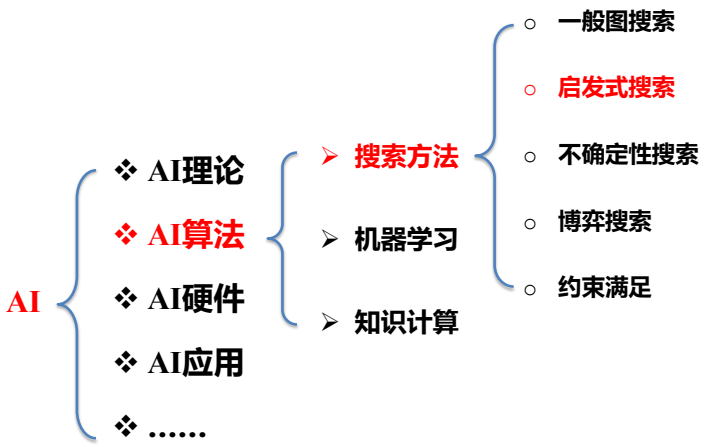
华东师范大学计算机科学与技术学院
School of Computer Science and Technology

《人工智能》

华东师范大学计算机科学技术系

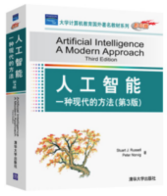
2020年春季学期

第3章 启发式搜索



第3章 启发式搜索

- **主要内容**
  - 3.1 A算法
  - 3.2 A*算法
  - 3.3 AO*算法
  - 3.4 本章小结
- 参考书目：
  - 《人工智能讲义》
  - 《人工智能》 朱福喜，清华大学出版社，2016. Ch4: pp43-63.
  - 《人工智能：一种现代的方法(第3版)》 拉塞尔、诺维格，清华大学出版社，2011. Ch3: pp. 64-120.

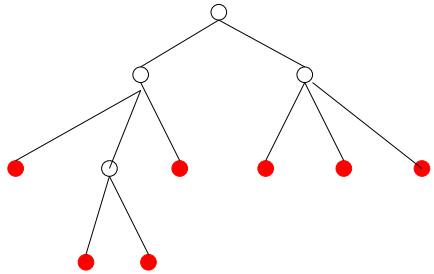


3.1 启发式图搜索

- **搜索算法**
  - 利用计算机的高性能来**有目的**的穷举一个**问题**解空间的部分或者所有的可能情况，从而求出问题的解的一种**计算机算法**
- **启发式信息**
  - **人**类有关求解问题的直观感觉或经验，往往无法得到证明
- **启发式搜索算法**
  - 利用**启发式信息**引导算法搜索，达到**减少搜索范围**，降低问题复杂度的目的
  - 将人类求解问题的经验固化到求解算法
- **启发信息的使用**
  - 强：降低搜索工作量，但可能导致找不到最优解
  - 弱：导致工作量加大，极限情况下变盲目搜索，但可能可以找到最优解
  - 关键点：如何利用启发式信息，尽可能有效地找到问题的近似最优解
  - 人：有关问题的求解经验+算法设计

3.1 启发式图搜索

- **算法思想**
  - 基于一般图搜索算法
  - 定义一个评价函数f，对当前的搜索状态进行评估，从Open表中找出一个最有希望的节点来扩展。（排序规则）



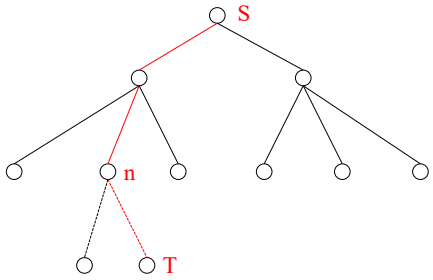
3.1 启发式图搜索

- **评价函数**

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

  f(n)：评价函数

  h(n)：启发函数
- **g*(n)**
  - 从起点S到当前点n的最短路径的耗散值
- **h*(n)**
  - 从当前点n到目标节点T的最短路径的耗散值
- **f*(n)=g*(n)+h*(n)**
  - 从S经过n到T的最短路径的耗散值
- g(n)、h(n)、f(n)分别是g*(n)、h*(n)、f*(n)的估计值



思考：

1. 搜索过程中，g(n)=g*(n)成立吗？如果成立，为什么？如果不成立，能想办法保证成立吗？

3.1 启发式图搜索

- **A算法**

1 $G=G_0$ (G_0=s), Open := (s), 计算f(s) := g(s) + h(s), Closed=();

2 Loop: If Open = ( ) Then Exit(Fail);

3 n := First(open), Remove(n, Open), Add(n, Closed);

4 If Goal(n) Then Exit(Success);

6 Expand(n) →{m_i}, 计算f(n, m_i) := g(n, m_i) + h(m_i), G:=Add(m_i, G);

7 标记和修改指针：

Add(m_j, Open), 标记m_j到n的指针； //新节点

If f(n, m_k) < f(m_k) Then f(m_k) := f(n, m_k), //Open表中未扩展节点

标记m_k到n的指针；

If f(n, m_l)<f(m_l) Then f(m_l):=f(n, m_l), //Closed表中已扩展节点

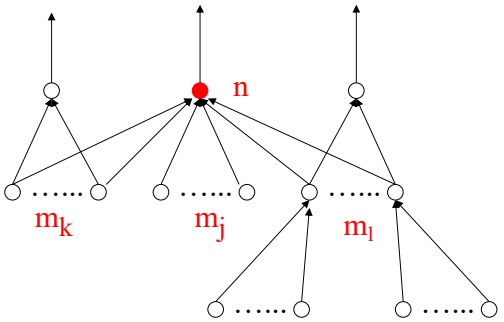
标记m_l到n的指针, Add(m_l, Open);

8 Open中的节点按f值从小到大排序；

9 Go Loop；

3.1 启发式图搜索

- **节点类型**



- m_k在Open表中
- m_j新扩展出来
- m_l在Closed表中

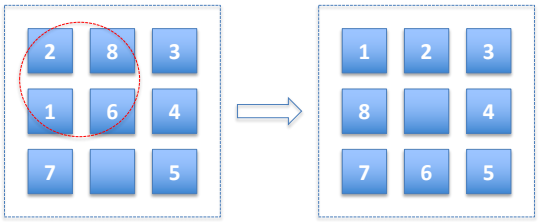
3.1启发式图搜索

- **九宫格游戏评价函数：**

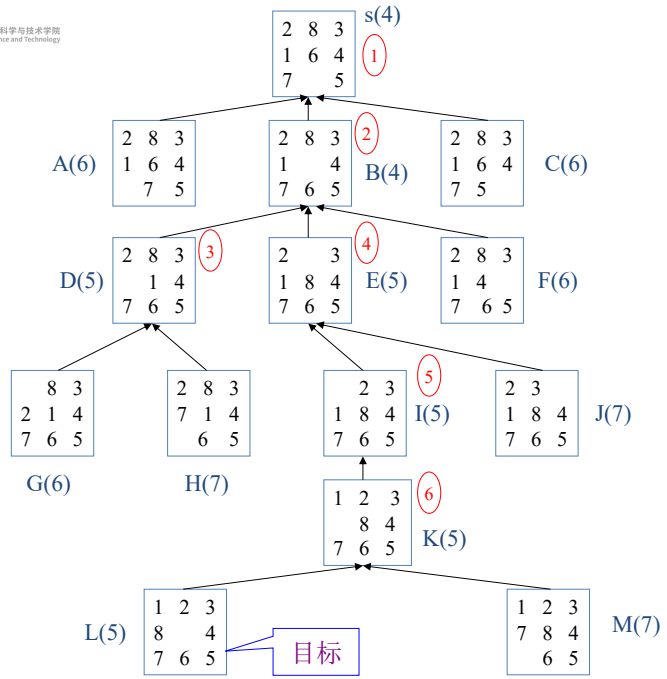
  f(n) = g(n) + h(n)

  g(n)为从初始节点到当前节点的耗散值

  h(n)为当前节点“**不在位**”的将牌数

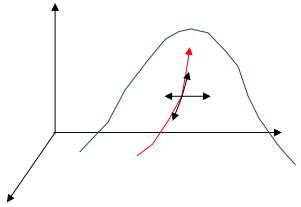


h(n) = 4



3.1启发式图搜索

- **爬山法**
  - 总是考虑与目标之间的差距
  - g(n) = 0，仅考虑h(n)



- **分支界限法**
  - 总是扩展具有最小耗散值的路径
  - 使用g(n)，h(n)=0

- **动态规划法**
  - 总是扩展具有最小耗散值的路径
  - 总是删除明显不可能的路径
  - 使用g(n)，h(n)=0

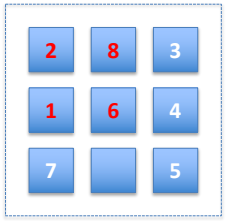
3.2 最佳图搜索A*算法

- 在A算法中，如果满足条件：

  $h(n) \le h^*(n)$

  则A算法称为A*算法。

- **九宫格例子**
  - h1(n) = “不在位”的将牌数
  - h2(n) = 将牌“不在位”的距离和



h1(n) = 4
h2(n) = 5

3.2.1 A*算法的性质

- **A*算法的假设**

  设n_i、n_j是任意两个节点，有：

  $C(n_i, n_j) > \varepsilon$

  其中ε为大于0的常数

- **几个等式**

  $f^*(S) = f^*(T) = h^*(S) = g^*(T) = f^*(n)$

  其中S是初始节点，T是目标节点，n是S到T的最佳路径上的节点。

3.2.1 A*算法的性质

- **定理1：**

  对有限图，如果从初始节点S到目标节点T有路径存在，则算法A一定成功结束。

- **引理1：**

  对无限图，若有从初始节点s到目标节点T的路径，则A*不结束时，在Open表中即使最小的一个f值也将增到任意大，或有f(n)>f*(S)。

- **引理2：**

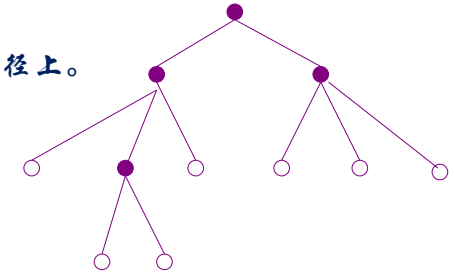
  A*结束前，Open表中必存在f(n)≤f*(S)。

3.2.1 A*算法的性质

证明：

存在一个节点n，n在最佳路径上。

$$
\begin{aligned}
f(n) &= g(n) + h(n) \\
&= g^*(n) + h(n) \\
&\le g^*(n) + h^*(n) \\
&= f^*(n) \\
&= f^*(S)
\end{aligned}
$$



3.2.1 A*算法的性质

- **定理2：**

  对无限图，若从初始节点S到目标节点T有路径存在，则A*一定成功结束。

证明：

由引理1.1：A*如果不结束，则Open中所有的n有$f(n) > f^*(S)$

由引理1.2：在A*结束前，必存在节点n，使得 $f(n) \le f^*(S)$

所以，如果A*不结束，将导致矛盾。

3.2.1 A*算法的性质

- **推论1：**

  Open表上任一具有f(n)<f*(S)的节点n，最终都将被A*选作扩展的节点。

证明：

由定理1.1和1.2，知A*一定结束，由A*的规则，OPEN表中f(T)最小时才被选择。而

$$f(T) \geq f^*(T) = f^*(S)$$

所以f(n)<f*(S)的n，均被扩展。得证。

3.2.1 A*算法的性质

- **定理3 (可采纳性定理)：**

  若存在从初始节点S到目标节点T有路径，则A*必能找到最佳解结束。

证明：

由定理1.1和1.2知，A*一定找到一条路径结束

设找到的路径s→t不是最佳的(t为目标),则

$$f(t) = g(T) > f^*(S)$$

由引理1.2知结束前OPEN中存在f(n)≤f*(S)的节点n，所以

$$f(n) \leq f^*(S) < f(T)$$

因此A*应选择n扩展，而不是T。与假设A*选择t结束矛盾，得证。(注意：A*的结束条件)

3.2.1 A*算法的性质

- **推论2：**

  A*选作扩展的任一节点n，有f(n)≤f*(s)。

证明：

由引理2.2知在A*结束前，OPEN中存在节点n'：f(n')≤f*(S)

设此时A*选择n扩展。

如果n = n' ，则f(n)≤f*(S)，得证。

如果n≠ n' ，由于A*选择n扩展，而不是n' ，所以有f(n) ≤ f(n')≤f*(S)。得证。

3.2.1 A*算法的性质

- **f(n)越接近f*(n)，则扩展的节点越少。**
- **应用的启发式信息越多，扩展越少。**
- **定理1.4**
  - **设对同一个问题定义了两个A*算法A_1和A_2，若A_2比A_1有较多的启发信息，即对所有非目标节点有$h_2(n) > h_1(n)$，则在具有一条从s到t的路径的隐含图上，搜索结束时，由A_2所扩展的每一个节点，也必定由A_1所扩展，即A_1扩展的节点数至少和A_2一样多。**
  - 简写：**如果$h_2(n) > h_1(n)$（目标节点除外），则A_1扩展的节点数≥A_2扩展的节点数**
  - **注意：**在定理1.4中，评价指标是“扩展的节点数”，也就是说，同一个节点无论被扩展多少次，都只计算一次。

3.2.1 A*算法的性质

证明：使用数学归纳法，对节点的深度进行归纳

(1)当d(n)＝0时，即只有一个节点，显然定理成立。

(2)设d(n)≤k时定理成立。（归纳假设）

(3)当d(n)=k+1时，用反证法。

设存在一个深度为k＋1的节点n，被A2扩展，但没有被A1扩展。而由假设，A1扩展了n的父节点，即n已经被生成。因此当A1结束时，n将被保留在OPEN中。

一方面，由假设： $f1(n) \geq f^*(S)$

即： $g1(n) + h1(n) \geq f^*(S)$

所以： $h1(n) \geq f^*(S) - g1(n)$

另一方面，由于A2扩展了n，有$f2(n) \leq f^*(S)$

即： $h2(n) \leq f^*(S) - g2(n)$ (A)

由于d(n)=k时，A2扩展的节点A1一定扩展，有

$g1(n) \leq g2(n)$ (因为A2的路A1均走到了，而且A1路径更多)

所以： $h1(n) \geq f^*(S) - g1(n) \geq f^*(S) - g2(n)$ (B)

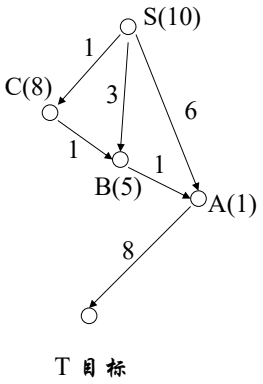
比较A、B两式，有 $h1(n) \geq h2(n)$ ，与定理条件矛盾。故定理得证。

3.2.2 A*算法的改进

- **A*算法的改进**
- **问题的提出：**
  - 因A算法第6步对m_i类节点可能要重新放回到OPEN表中，因此可能会导致多次重复扩展同一个节点，导致搜索效率下降。
  - $g(n) \neq g^*(n)$

3.2.2 A*算法的改进

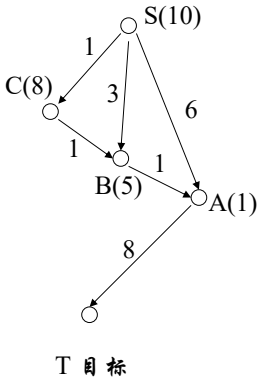
- **一个例子**



Open表	Closed表
S(10)	S(10)
A(7) B(8) C(9)	A(7) S(10)
B(8) C(9) G(14)	B(8) S(10)
A(5) C(9) G(14)	A(5) B(8) S(10)
C(9) G(12)	C(9) A(5) S(10)
B(7) G(12)	B(7) C(9) S(10)
A(4) G(12)	A(4) B(7) C(9) S(10)
T(11)	

3.2.2 A*算法的改进

- **出现多次扩展节点的原因**
  - 在前面的扩展中，并没有找到从初始节点到当前节点的最短路径，如节点A。



3.2.2 A*算法的改进

- **解决的途径**
  - 对h加以限制：对h增加适当的限制，使得第一次扩展一个节点时，就找到了从S到该节点的最短路径。
  - 对算法加以改进：对算法加以改进，避免或减少节点的多次扩展。
- **改进的条件**
  - 可采纳性不变
  - 不多扩展节点
  - 不增加算法的复杂性

对h加以限制

- 定义：一个启发函数h，如果对所有节点n_i和n_j，其中n_j是n_i的子节点，满足

  $h(n_i) - h(n_j) \le c(n_i, n_j)$

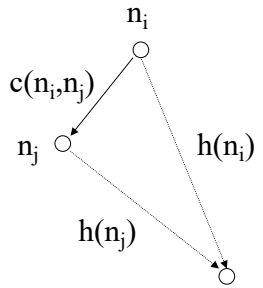
  $h(t) = 0$

  或

  $h(n_i) \le c(n_i, n_j) + h(n_j)$

  $h(t) = 0$

  则称**h是单调**的。



估计越来越准确！

三角不等式！

h单调的性质

- **定理5：**

  若h(n)是单调的，则A*扩展了节点n之后，就已经找到了到达节点n的最佳路径。即：当A*选n扩展时，有g(n)=g*(n)。

证明：

设n是A*扩展的任一节点。当n = S时，定理显然成立。下面考察n ≠S的情况。

设P = (n_0=s, n_1, n_2, …, n_k=n)是s到n的最佳路径，则P中一定有节点在Closed中，设P中最后一个出现在Closed中的节点为n_j，则n_{j+1}在Open中。

由单调限制条件，对P中任意节点n_i有：

$$h(n_i) \le C(n_i, n_{i+1})+h(n_{i+1})$$

$$g^*(n_i)+h(n_i) \le g^*(n_i)+C(n_i, n_{i+1})+h(n_{i+1})$$

由于n_i、n_{i+1}在最佳路径上，所以：

$$g^*(n_{i+1}) = g^*(n_i)+C(n_i, n_{i+1})$$

带入上式有：

$$g^*(n_i)+h(n_i) \le g^*(n_{i+1})+h(n_{i+1})$$

h单调的性质

从i=j到i=k-1应用上不等式，有：

$$g^*(n_{j+1})+h(n_{j+1}) \le g^*(n_k)+h(n_k)$$

即：$f(n_{j+1}) \le g^*(n)+h(n)$（注意：$n_j$在Closed中，$n_{j+1}$在Open中）

重写上式：

$$f(n_{j+1}) \le g^*(n)+h(n)$$

另一方面，A*选n扩展，必有：

$$f(n) = g(n)+h(n) \le f(n_{j+1})$$

比较两式，有：

$$g(n) \le g^*(n)$$

但已知g*(n)是最佳路径的耗散值，所以只能有：g(n) = g*(n)。得证。

h单调的性质

- **定理6：**

若h(n)是单调的，则由A*所扩展的节点序列其f值是非递减的。即$f(n_i) \le f(n_j)$。

- h单调的例子
  - 九宫格问题：(n_j为n_i的后继节点)

    h为“不在位”的将牌数

    $h(n_i) - h(n_j) = \begin{cases} 1 & \text{不在位->在位} \\ 0 & \text{不在位->不在位} \\ -1 & \text{在位->不在位} \end{cases}$

    $h(t) = 0$

    $c(n_i, n_j) = 1$

    满足单调的条件。

定理6的证明

证明：由单调限制条件，有：

$h(n_i) - h(n_j) \le C(n_i, n_j)$

$= f(n_i)-g(n_i)$　　$= f(n_j)-g(n_j)$

$$f(n_i)-g(n_i) - f(n_j)+g(n_j) \le C(n_i, n_j)$$

$= g(n_i)+C(n_i, n_j)$

$$f(n_i)-g(n_i) - f(n_j)+ g(n_i)+C(n_i, n_j) \le C(n_i, n_j)$$

$f(n_i) - f(n_j) \le 0$，得证。

3.2.2 A*算法的改进

- **一些结论：**
  - **Open表上任以具有f(n) < f*(S)的节点定会被扩展。**
  - **A*选作扩展的任一节点，定有f(n)≤f*(S)。**
- **改进的出发点**

Open = (… … f*(S)… …)

f值小于f*(S)的节点　　f值大于等于f*(S)的节点

f_m：到目前为止已扩展节点的最大f值，用f_m代替f*(S)

3.2.2 A*算法的改进

- **改进A算法**

1 G=G_0 (G_0=s), Open := (s), 计算f(s) := g(s) + h(s), Closed=(), f_m:=0;
2 Loop: If Open = ( ) Then Exit(Fail);
3 Next:={n_i|$f(n_i)<f_m$}
If Next ≠ ( ) Then n:=Next中g最小的节点
Else n:=First(Open);
f_m:=max{f_m, f(n)};
Remove(n, Open), Add(n, Closed);
4 If Goal(n) Then Exit(Success);
6 Expand(n) →{m_i}, 计算f(n, m_i) := g(n, m_i) + h(m_i), G:=Add(m_i, G);

3 n := First(open), Remove(n, Open), Add(n, Closed);

3.2.2 A*算法的改进

- **改进A算法**

7 **标记和修改指针：**

Add(m_j, Open), 标记m_j到n的指针； //新节点

If f(n, m_k) < f(m_k) Then f(m_k) := f(n, m_k), //Open表中未扩展节点

标记m_k到n的指针；

If f(n, m_l)<f(m_l,) Then f(m_l):=f(n, m_l), //Closed表中已扩展节点

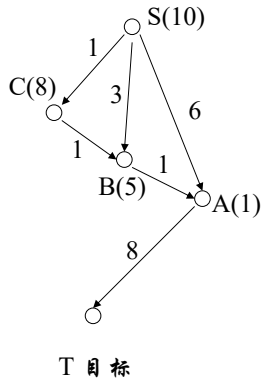
标记m_l到n的指针, Add(m_l, Open);

8 **Open中的节点按f值从小到大排序；**

9 **Go Loop；**

3.2.2 A*算法的改进

- **例子**



OPEN 表	CLOSED 表	f_m
S(0+10)	S(0+10)	10
A(6+1) B(3+5) C(1+8)	S(0+10) C(1+8)	10
A(6+1) B(2+5)	S(0+10) C(1+8) B(2+5)	10
A(3+1)	S(0+10)C(1+8)B(2+5)A(3+1)	10
T(11+0)		

3.2.2 A*算法的改进

- **h的单调化方法**
  - 如果令：

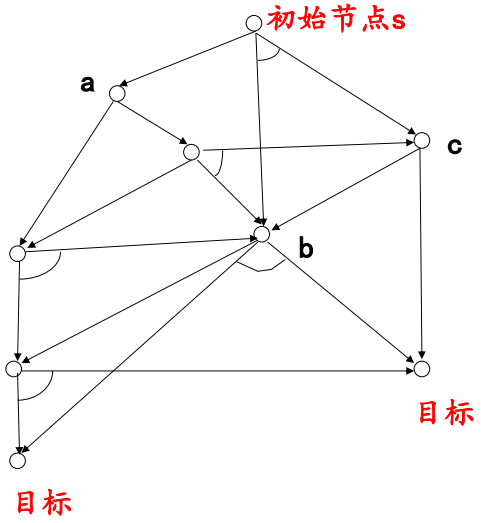
    f(n) = max(f(n的父节点), g(n)+h(n))

    则容易证明，这样处理后的h是单调的。

3.3 与或图搜索

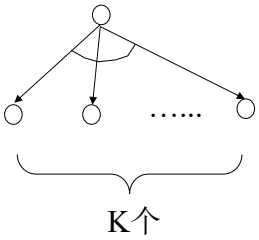
- **如何求解一类问题，该问题可以分解为一系列子问题或转换为等价问题？**
- **A算法求解问题==在图中寻找一条（最优）路径**
- **与或图搜索==在图中寻找一个（最优）子图**
- **内容：**
  - 3.3.1 与或图的基本定义
  - 3.3.2 AO*算法
  - 3.3.3 AO*算法求解例子

3.3.1 与或图基本定义



3.3.1 与或图基本定义

- **与或图是一个超图，节点间通过连接符连接。**
- **K-连接符：**



- **耗散值的计算**

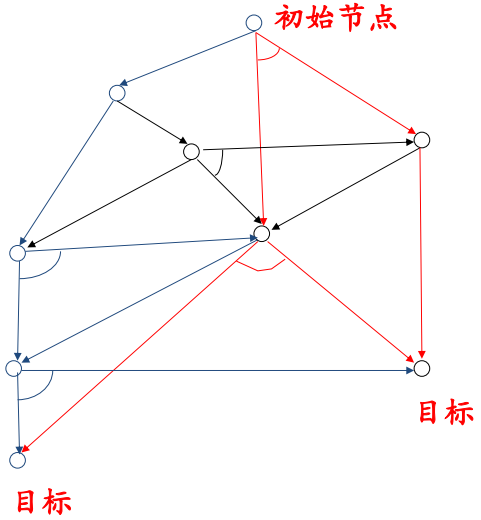
  $k(n, N) = C_n + k(n_1, N) + \ldots + k(n_i, N)$

  其中：N为终节点集

  C_n为连接符的耗散值

3.3.1 与或图基本定义

- **解图：**

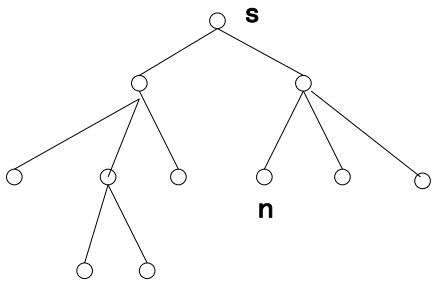


3.3.1 与或图基本定义

- **能解节点**
  - 终节点是能解节点
  - 若非终节点有“或”子节点时，当且仅当其子节点至少有一能解时，该非终节点才能解
  - 若非终节点有“与”子节点时，当且仅当其子节点均能解时，该非终节点才能解
- **不能解节点**
  - 没有后裔的非终节点是不能解节点。
  - 若非终节点有“或”子节点，当且仅当所有子节点均不能解时，该非终节点才不能解
  - 若非终节点有“与”子节点时，当至少有一个子节点不能解时，该非终节点才不能解

3.3.1 与或图基本定义

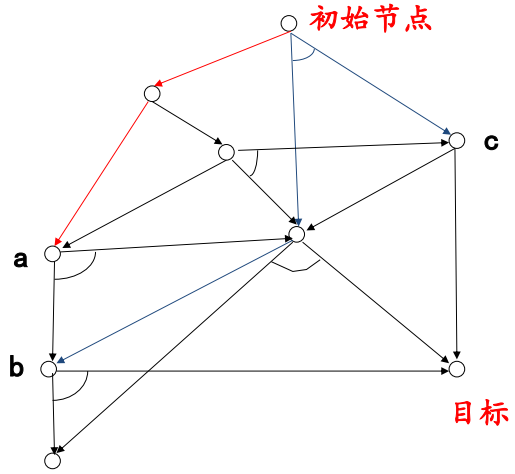
- 普通图搜索的评价



f(n) = g(n) + h(n)
对n的评价实际是对从s到n这条路径的评价

3.3.1 与或图基本定义

- 与或图: 对局部图的评价



3.3.2 AO*算法

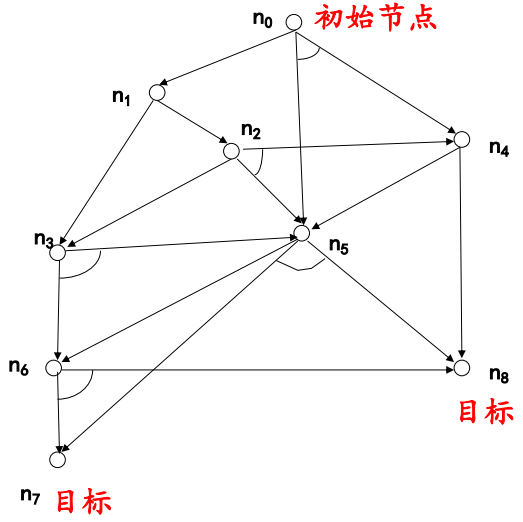
1 建立搜索图G:=S,计算q(S)=h(S), If Goal(S) Then M(S, Solved)
2 Until S 被标记为Solved, Do:
3 Begin //扩展
4 G':= Find(G) //G'为根据连接符找到的待扩展局部解图
5 n := G'中任一非终结点
6 $\{n_j\}$:=Expand(n), 计算$q(n_j)=h(n_j)$,If Goal(n_j) Then M(n_j,solved)
7 N:={n} //回溯,修改指针和耗散值
8 Until N为空, Do:
9 Begin
10 Remove(m, N) If m的子结点不在N中 //子结点耗散值已确定

11 修改m的耗散值：
对m的每个连接符i $\{n_{1i},n_{2i},\ldots,n_{ki}\}$,计算$q_i(m)=C_i+q(n_{1i})+\ldots+q(n_{ki})$
$q(m):=\min q_i(m)$
修改m的指针到$\min q_i(m)$对应的连接符上
If(n_{ji},Solved) THEN M(m,Solved) //某一个连接符已解决
12 If M(m,Solved) or q(m)被修改 Then Add(m_a,N), m_a为m的所有先辈节点
13 End
14 End

3.3.2 AO*算法

- **AO*包含两个过程**
  - 图生成过程，即扩展节点
    - 从最优的局部途中选择一个节点扩展
  - 计算耗散值的过程
    - 对当前的局部图从新计算耗散值

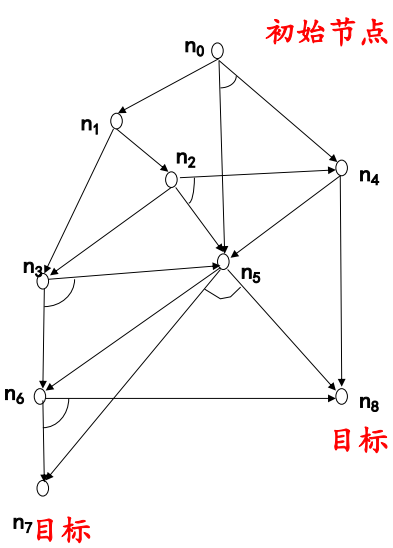
3.3.3 AO*算法举例

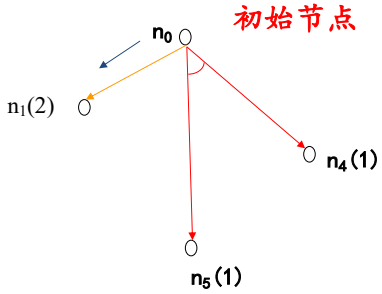


其中：

$h(n_0)=3$
$h(n_1)=2$
$h(n_2)=4$
$h(n_3)=4$
$h(n_4)=1$
$h(n_5)=1$
$h(n_6)=2$
$h(n_7)=0$
$h(n_8)=0$

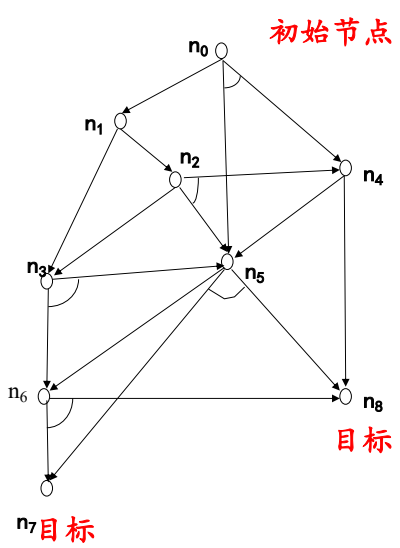
设：K连接符的耗散值为K

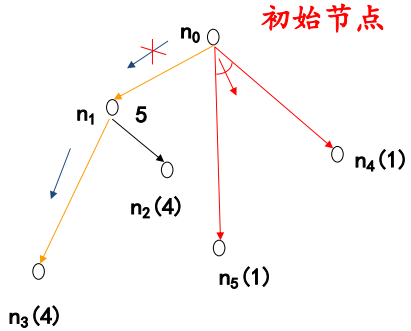




红色：4

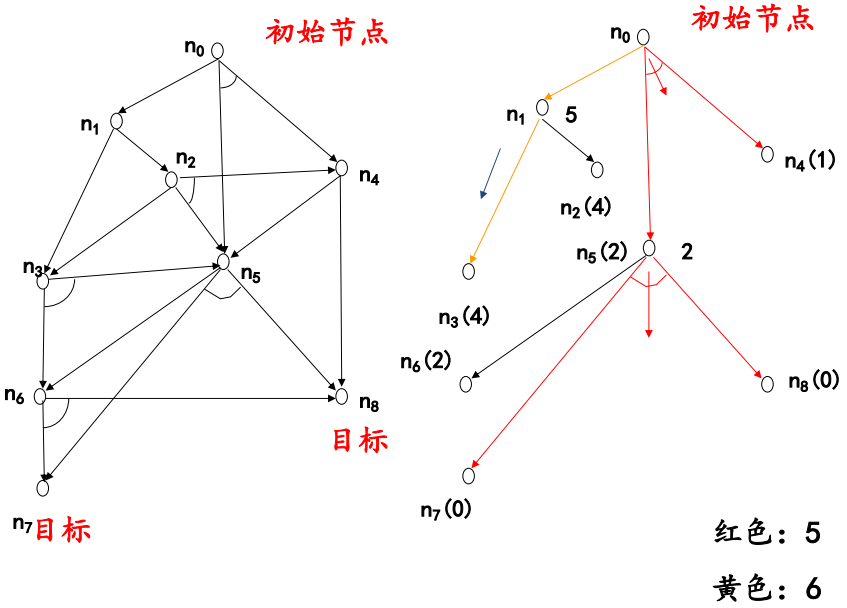
黄色：3

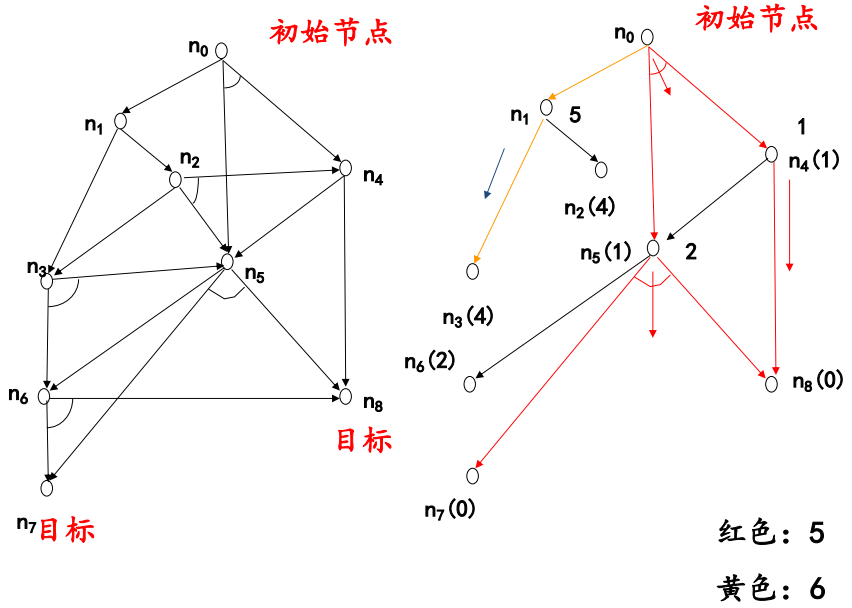




红色：4

黄色：6





3.4 本章小结

- **A算法是利用了启发式信息的一般图搜索算法**
  - 无信息搜索 vs 有信息搜索
- **A*算法是一类特殊的A算法**
  - 满足条件：h(n)≤h*(n)
  - 启发信息使用越多，算法效率越高
  - 单调性质对算法改进
- **AO*算法是A*算法的图版本**
  - 解是一个子图

谢谢