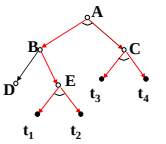
主要内容

- 3.1 问题规约和与或图
- 3.2 与或树的盲目式搜索
- 3.3 博弈树搜索

➢ 与或图搜索：在与或图上执行搜索的过程，其目的在于标明起始节点是可解的，即搜索不是去寻找到目标节点的一条路径，而是寻找一个解树。

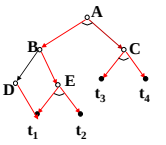
- 执行可解节点标志和不可解点标志过程。
- 解树：那些可解节点所构成的子树，这些节点能够证明问题的初始节点是可解的。



与或树

➢ 与或图/与或树

- 与或树：除初始节点，其余节点只有一个父节点
- 与或图：除初始节点，其余节点允许有多个父节点



与或图

与或树搜索

➢ 与或树的一般搜索过程：

(1) 把原始问题作为初始节点S_0，并把它作为当前节点；

(2) 应用分解或等价变换对当前节点进行扩展；

(3) 为每个节点设置指向父节点的指针；

(4) 选择适合的节点作为当前节点，反复执行第(2)步和第(3)步，在此其间要多次调用可解标志过程和不可解标志过程，直到初始节点被标为可解节点或不可解节点为止。

➢ 与或树的搜索的目的是寻找解树，从而求得原始问题的解.

与或树搜索

➢ 与或树搜索：

- 搜索目的：是证明起始节点是否可解，而可解节点是递归定义的，取决于后继节点是否可解，即搜索过程是能否找到可解的叶节点。
- 若初始节点被标志为可解，则搜索成功结束；若初始节点被标志为不可解，则搜索失败。

与或树搜索

与/或树的搜索策略

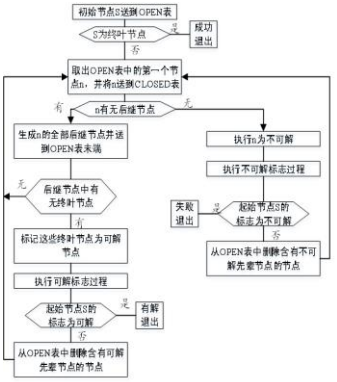
- ➢ 盲目式搜索
  - 宽度优先搜索
  - 深度优先搜索
- ➢ 博弈树搜索
  - Max-Min搜索
  - α–β剪枝

与或树的宽度优先搜索

➢ 与或树的宽度优先搜索的基本思想：

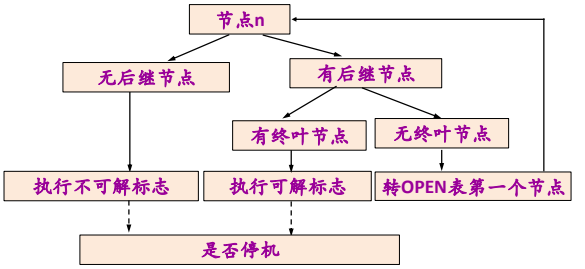
- 按照“先产生的节点先扩展”的原则进行搜索
- 搜索过程中要多次调用可解标志过程和不可解标志过程
- OPEN表、CLOSED表类似状态空间搜索
  - OPEN表：存放待扩展的节点
  - CLOSED表：存放已扩展的节点

与或树的宽度优先搜索流程图



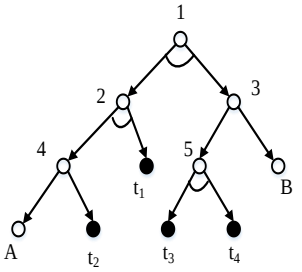
与或树的宽度优先搜索

➢ **关键步骤**：扩展节点n，生成其全部后继节点，送到OPEN表末端，并设置指向n的指针。

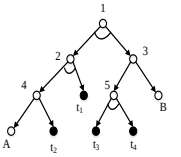


与或树的宽度优先搜索

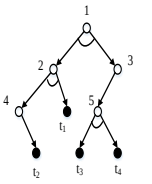
例：设有如图所示的与/或树，其中 t_1, t_2, t_3, t_4 均为终叶节点，A和B是不可解的端节点。用**宽度优先搜索法**求出解树。



与或树的宽度优先搜索



OPEN表	CLOSED 表
{1}	{}
{2,3}	{1}
{3,4,t_1}	{1,2}
{4,t_1,5,B}	{1,2,3}
{ t_1,5,B,A,t_2}	{1,2,3,4}
{ 5,B,t_2}	{1,2,3,4,t_1}
{ B,t_2,t_3,t_4}	{1,2,3,4,t_1,5}



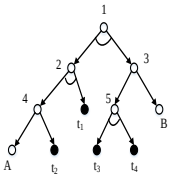
解树

与或树的宽度优先搜索

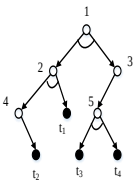
➢ “先产生的节点先被扩展”

➢ 如果问题有解，宽度优先搜索一定会求得一个解树，而且其最深的叶节点具有最小的深度。

- 初始节点深度为零
- 其他节点的深度是其父节点深度加1



与或树

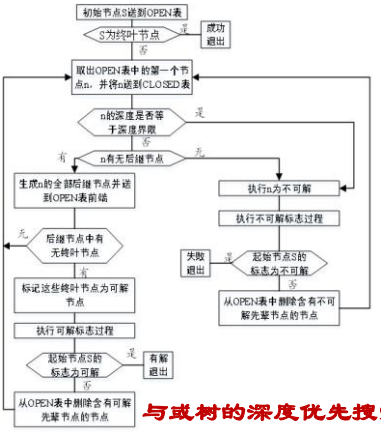


宽度优先搜索的解树

与或树的深度优先搜索

➢ 与或树的深度优先搜索

- 按照“新产生的节点先扩展”的原则进行搜索
- 设置深度界限
- 搜索过程中要多次调用可解标志过程和不可解标志过程



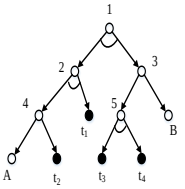
与或树的深度优先搜索流程图

与或树的深度优先搜索

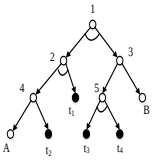
- 与宽度优先算法相比，深度优先算法的特殊之处：
  - 要判断从OPEN表取出来的节点的深度。如果等于深度界限，认定它为不可解节点；
  - 扩展节点n把其子节点放入OPEN表的前端，即新产生的节点先扩展；

与或树的深度优先搜索

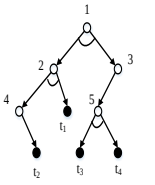
例：设有如图所示的与/或树，其中t_1,t_2,t_3,t_4均为终叶节点，A和B是不可解的端节点。采用深度优先搜索法求出解树。（规定深度界限为4）



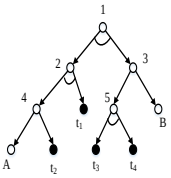
与或树的深度优先搜索



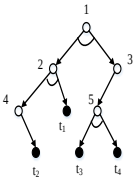
OPEN表	CLOSED 表
{1}	{}
{2,3}	{1}
{4,t_1,3}	{1,2}
{A,t_2,t_1,3}	{1,2,4}
{3}	{1,2,4,t_2,t_1}
{5,B}	{1,2,4,t_2,t_1,3}
{t_3,t_4,B}	{1,2,4,t_2,t_1,3,5}



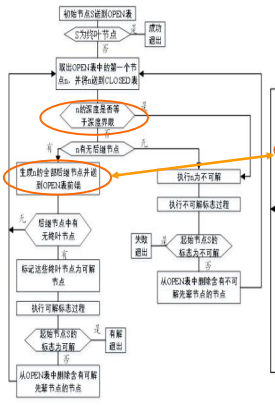
与或树的深度优先搜索



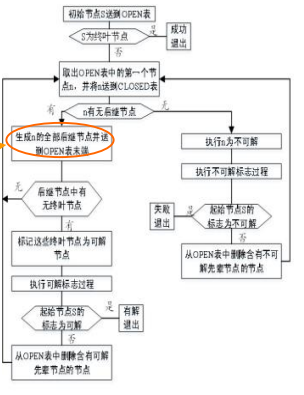
与或树



深度优先搜索的解树



与或树深度优先搜索算法



与或树宽度优先搜索算法

小结

- 与或树的宽度优先搜索算法
  - 要点：先产生的节点先被扩展
- 与或树的深度优先搜索
  - 要点：新产生的节点先被扩展

主要内容

- 3.1 问题规约和与或图
- 3.2 与或树的盲目式搜索
- 3.3 博弈树搜索

机器博弈

你可曾听说过“深蓝”？

1997年5月11日，IBM开发的“深蓝”击败了国际象棋冠军卡斯帕罗夫。

卡氏何许人也？

- *1980年他获得世界少年组冠军*
- *1982年他并列获得苏联冠军*
- *1985年22岁的卡斯帕罗夫成为历史上最年轻的国际象棋冠军*
- *积分是2849，这一分数是有史以来最高分。远远领先于第二位的克拉姆尼克的2770*

机器博弈

- 20世纪50年代，有人设想利用机器智能来实现机器与人的对弈。
- 1997年IBM的“深蓝”战胜了国际象棋世界冠军卡斯帕罗夫，惊动了世界。
- 加拿大阿尔伯塔大学的奥赛罗程序Logistello和西洋跳棋程序Chinook也相继成为确定的、二人、零和、完备信息游戏世界冠军。
- 西洋双陆棋这样的存在非确定因素的棋类也有了美国卡内基梅隆大学的西洋双陆棋程序BKG这样的世界冠军。
- 2017年5月，在中国乌镇围棋峰会上，AlphaGo与排名世界第一的世界围棋冠军柯洁对战，以3比0的总比分获胜。

博弈搜索

- **特征：智力竞技**

机器博弈，意味着机器参与博弈，参与智力竞技。

我们这里的博弈只涉及双方博弈，常见的是棋类游戏，如：中国象棋，军旗，围棋等。

- **目标：取胜**

取胜的棋局如同状态空间法中的目标状态。

与八数码游戏一样，游戏者需要对棋局进行操作，以改变棋局，使其向目标棋局转移。

然而，八数码游戏只涉及一个主体，不是博弈。

博弈涉及多个主体，他们按规则，依次对棋局进行操作，并且目标是击败对手。

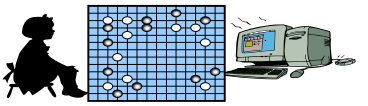
- **方法：Max-Min搜索、α–β剪枝**

博弈问题为什么可以用与或图表示呢？

当轮到我方走棋时，只需要从若干个可以走的棋中，选择一个棋走就可以了。从这个意义上说，若干个可以走的棋是“或”的关系。
而轮到对方走棋时，对于我方来说，必须能够应付对手的每一种走棋。这就相当于这些棋是“与”的关系。
因此博弈问题可以看成是一个与或图，但是与一般的与或图并不一样，是一种特殊的与或图。

双方博弈实例

以围棋为例，竞技的双方分为黑方和白方，由黑方开棋，双方轮流行棋，最终，谁占据的地盘大，谁就成为获胜方。



Max-Min搜索

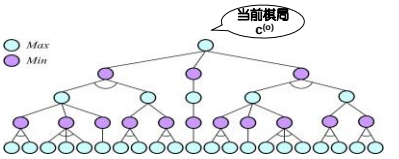
基本思想：

- 目的是为博弈的双方中的一方寻找一个最优行动方案；
- 要寻找这个最优方案，就要通过计算当前所有可能的方案来进行比较；
- 方案的比较是根据问题的特征来定义一个估价函数，用来估算当前博弈树端节点的得分；
- 当计算出端节点的估值后，再推算出父节点的得分（即计算倒推值）；
  - 对Max节点，选其子节点中一个最大得分作为父节点的得分
  - 对Min节点，选其子节点中一个最小得分作为父节点的得分
- 如果一个行动方案能获得较大的倒推值，则它就是当前最好的行动方案。

Max-Min搜索

Step1.生成k-步博弈树

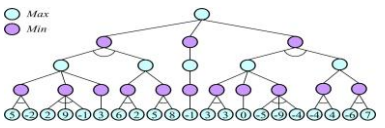
- Max 代表机器一方 / Min 代表敌方
- 设 Max 面对的当前棋局为 $c^{(0)}$，以 $c^{(0)}$为根，生成 k-步博弈树：



Max-Min搜索

Step2. 评估棋局(博弈状态)

- 估价函数
  - ✓ 为特定的博弈问题定义一个估价函数est(c)，用以评估k-步博弈树叶节点对应的棋局c
  - ✓ est(c)的值越大，意味着棋局c对Max越有利。

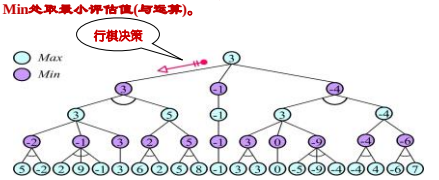


Max-Min搜索

Step3. 回溯评估

- 极大极小运算

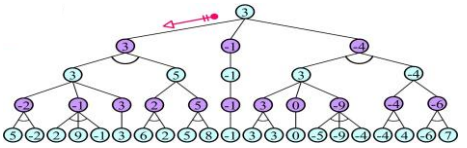
由叶节点向根节点方向回溯评估，在Max处取最大评估值(或运算)，在Min处取最小评估值(与运算)。



Max-Min搜索

Step3. 回溯评估

- 极大极小运算



- Max 按取最大评估值的方向行棋

Max-Min搜索

Step4. 递归循环

- Max 行棋后，等待 Min 行棋；
- Min 行棋后，即产生对于 Max 而言新的当前棋局 $c^{(0)}$；
- 返回 Step1，开始下一轮博弈。

Max-Min搜索

Max-Min搜索流程总结：

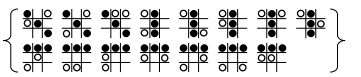
- **Step1：以 c(o) 为根，生成 k-步博弈树；**
- **Step2：评估博弈树叶节点对应的博弈状态(棋局)；**
- **Step3：进行极大极小运算 (Max-Min 运算)；**
- **Step4：等待 Min 行棋，产生新的 c(o)，返回 step1.**

Max-Min搜索

一字棋：

设有 3x3 棋格，Max 与 Min 轮流行棋，最先白后，先将 3 颗棋子连成一线的一方获胜。

- 一字棋博弈空间：共有 9！种可能的博弈状态
- 一字棋算子空间：博弈规则集合
- 一字棋博弈目标集合 (对 Max而言)：

$\Sigma c^{(g)}=$ 

Max-Min搜索

一字棋：

- 定义估价函数：*est*(c)

(1) 对于非终局的博弈状态c，估价函数为：
est(c)=(所有空格都放上黑色棋子之后，3颗黑色棋子连成的直线总数) − (所有空格都放上白色棋子之后，3颗白色棋子连成的直线总数)

est(c)=3−2=1　　例如：c= 

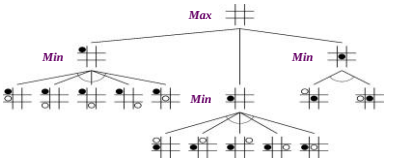
(2) 若 c 是 Max 的胜局，则：
est(c) = +∞　　例如：c= 

(3) 若 c 是 *Min* 的胜局，则：
est(c) = −∞　　例如：c= 

需要说明的是，等价的(如具有对称性的)棋局被视为相同棋局。

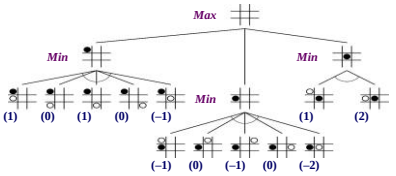
Max-Min搜索

step1. 以 $c^{(o)}$ = 为根，生成2-步博弈树：



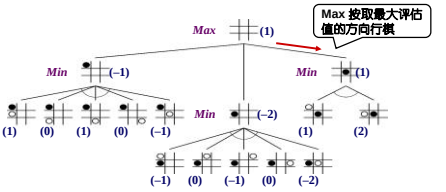
Max-Min搜索

step2. 评估博弈树叶节点对应的博弈状态



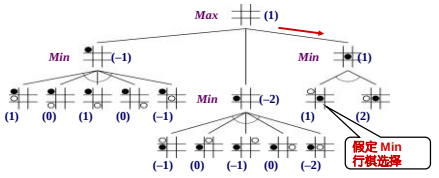
Max-Min搜索

step3. 进行极大极小运算(Max-Min运算)



Max-Min搜索

step4. 等待 Min 行棋，产生新的 $c^{(\omega)}$，返回 step1.



则下一轮：$c^{(\omega)}$ =

α–β 剪枝

首先分析极大极小法效率，上述的极大极小法，实际是先生成一棵博弈树，然后再计算其倒推值，至使极大极小法效率较低。于是在极大极小法的基础上提出了α-β剪枝技术。

α-β剪枝技术的基本思想，边生成博弈树边计算评估各节点的倒推值，并且根据评估出的倒推值范围，及时停止扩展那些已无必要再扩展的子节点，即相当于剪去了博弈树上的一些分枝，从而节约了机器开销，提高了搜索效率。

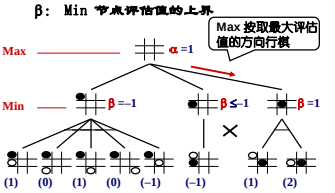
α–β 剪枝

- 实际上，就博弈而言，人类棋手的思维模式更多地表现出深度优先的特征，而不是宽度优先。
- 因此，采用深度优先搜索策略进行k-步博弈搜索，更符合AI模拟人类智能的原则，这里的k是深度优先搜索的一个自然的深度界限。
- 深度优先搜索策略产生的k-步博弈树是可以剪枝的，因此，搜索空间较小。重要的是，这正是人类棋手约束搜索空间的特征。

α–β 剪枝

一字棋：

- 搜索策略：k-步博弈；深度优先；每次扩展一个节点；一边扩展一边评估。
- 在 α -β 算法中：
  - α：Max 节点评估值的下界
  - β：Min 节点评估值的上界



一字棋的α–β搜索过程

α–β 算法的剪枝规则

对于一个Min节点来说，它取当前子节点中的最小倒推值作为它倒推值的上界，称此为β值；（β <= 最小值 ）

对于一个Max节点来说，它取当前子节点中的最大倒推值作为它倒推值的下界，称此为α值. （α >= 最大值）

规则一（α 剪枝规则）：

任何Min节点x的β值如果不能升高其父节点的α值，则对节点x以下的分支可停止搜索，并使x的倒推值为β

规则二（β 剪枝规则）：

任何Max节点x的α值如果不能降低其父节点的β值，则对节点x以下的分支可停止搜索，并使x的倒推值为α

由规则一形成的剪枝被称为 “α 剪枝”，而由规则二形成的剪枝被称为 “β 剪枝”。

习题

- 设有如下图所示的博弈树，其中最下面的数字是假设的估值。请计算各节点的倒推值。