对抗搜索

张文生 孙正雅 研究员

中国科学院自动化研究所
中科院大学人工智能学院

2020年10月30日

讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完整的实时决策

五、随机博弈

六、部分可观察的博弈

七、博弈程序发展现状

博弈

- **双人完备信息博弈：**
  - ✓ 指两位选手对垒，轮流走步，这时每一方不仅都知道对方过去已经走过的棋步，而且还能估计出对方未来可能的走步；
  - ✓ 对弈的结果是一方赢（另一方则输），或者双方和局；
  - ✓ 这类博弈的实例有：一字棋、余一棋、西洋跳棋、国际象棋、中国象棋、围棋等。
- **机遇性博弈：**存在不可预测性的博弈，例如掷币、西洋双陆棋等。

博弈

- 考虑两人参与的游戏：MAX和MIN，可以形式化成含有下列组成部分的一类搜索问题
  - S_0：初始状态
  - PLAYER(s)：定义此时该谁行动
  - **ACTION(s)**：返回此状态下的合法移动集合
  - **RESULT(s,a)**：转移模型，定义行动的结果
  - TERMINAL-TEST(s)：终止测试，游戏结束的状态称为终止状态
  - UTILITY(s,p)：效用函数，定义游戏者在终止状态的数值

博弈

- 初始状态、ACTIONS函数和RESULT函数定义了游戏的博弈树——其中结点是状态，边是移动

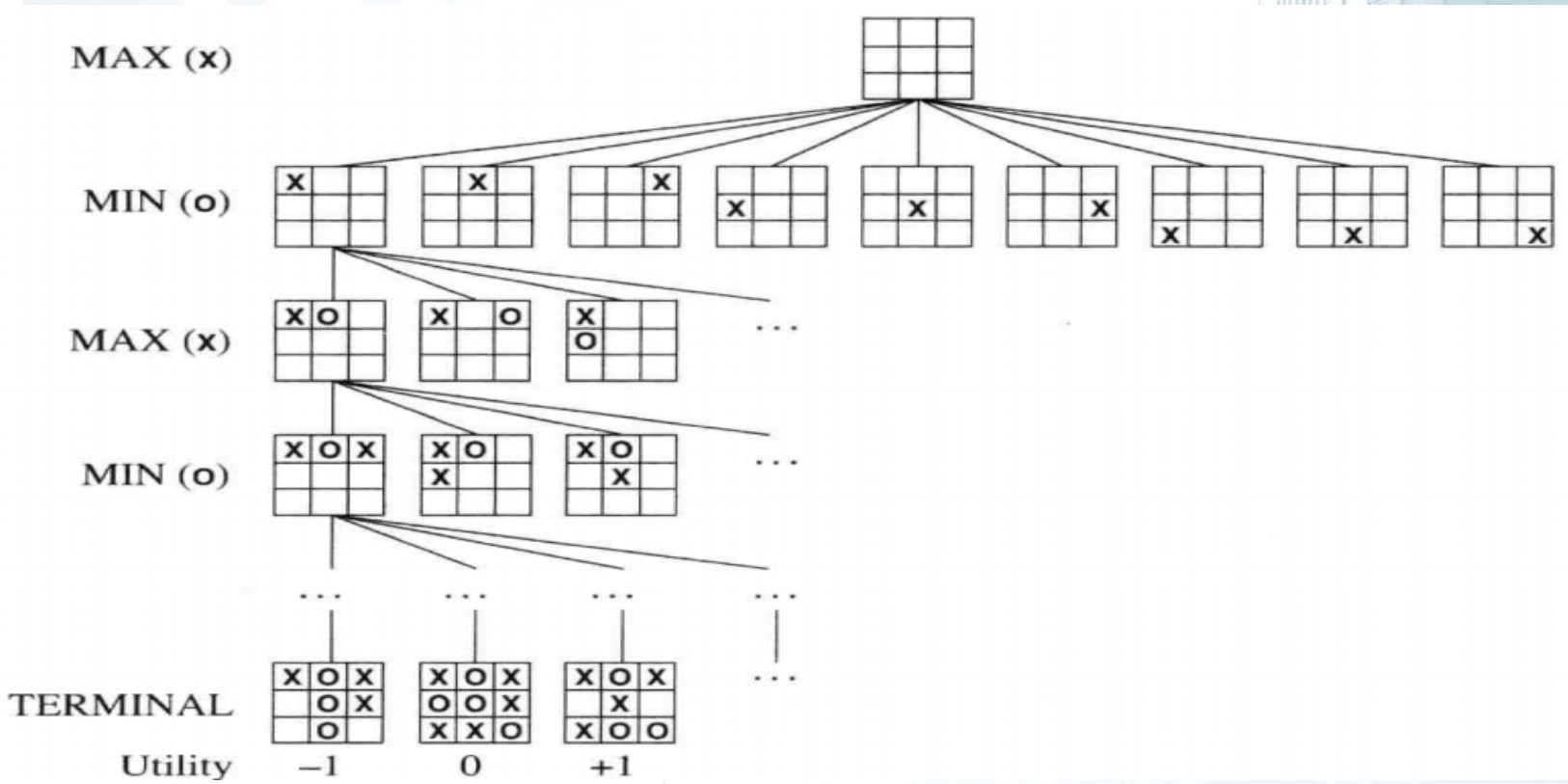


图 井字棋游戏的（部分）搜索树

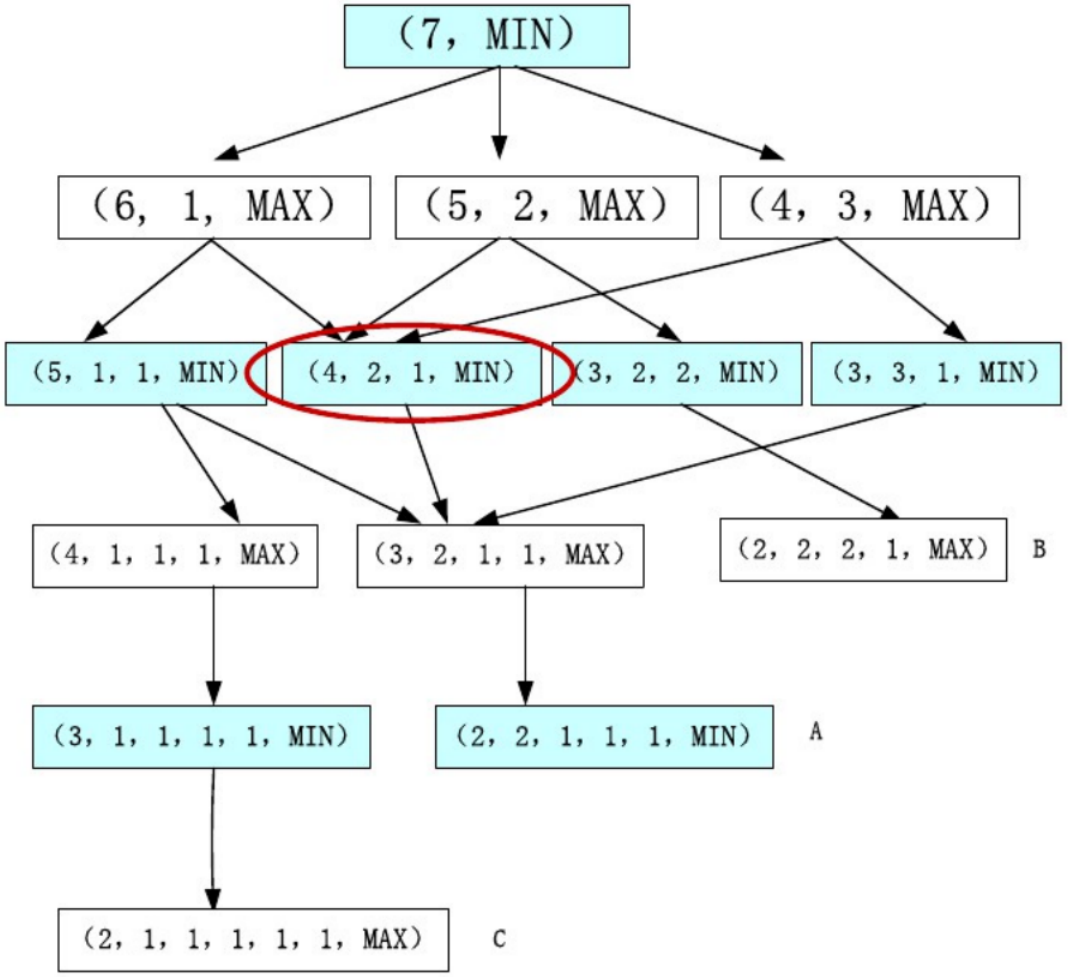
最上面的结点是初始状态，MAX 先走棋，放置一个 X 在空位上。图显示了搜索树的一部分，给出 MIN 和 MAX 的轮流走棋过程，直到到达终止状态，所有终止状态都按照游戏规则被赋予了效用值

Grundy博弈

- 当初试钱币数为7时的状态空间图

MIN代表对方走，
MAX代表我方走

MAX存在完全取胜的策略



Grundy博弈

- 搜索策略需要考虑的问题：
  - ✓ 对MIN走步后的每一个MAX节点，必须证明MAX对MIN可能走的每一个棋局对弈后能获胜，即MAX必须考虑应付MIN的所有招法，这是一个“与”的含义
  - ✓ 含有MAX符号的节点可看成与节点
  - ✓ 对MAX走步后的每一个MIN节点，只需证明MAX有一步能走赢就可以，即MAX只要考虑能走出一步棋使MIN无法招架就成
  - ✓ 因此含有MIN符号的节点可看成或节点
- 对弈过程的搜索图就呈现出与或图表示的形式

讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完整的实时决策

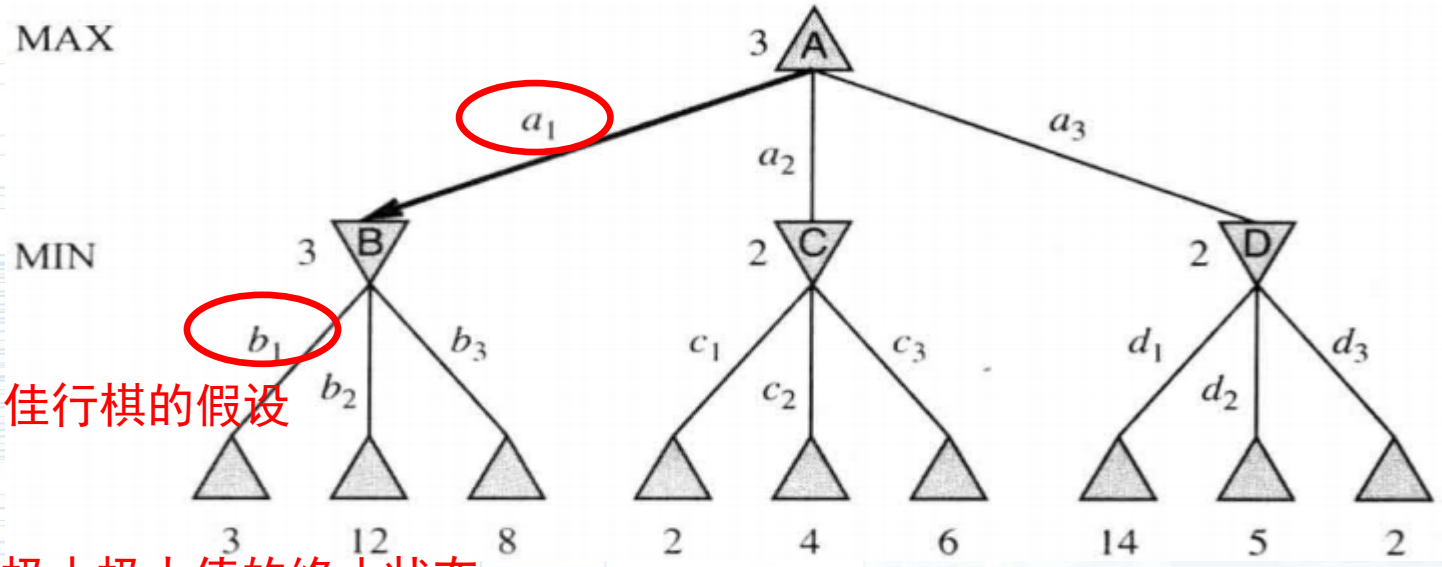
五、随机博弈

六、部分可观察的博弈

七、博弈程序发展现状

博弈中的优化决策

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & s \text{ 为终止状态} \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & s \text{ 为 MAX 结点} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & s \text{ 为 MIN 结点} \end{cases}$$



MIN也按最佳行棋的假设

指向有最高极小极大值的终止状态

可能有一些策略在对付非最优化对手方面做得比极小极大策略好，但是用这些策略对付最优化对手则会得到更差的结果

极小极大搜索过程

- 极小极大算法从当前状态计算极小极大决策。它使用了简单的递归算法计算**每个后继的极小极大值**
- 递归算法**自上而下**一直前进到树的叶结点，然后随着递归回溯通过搜索树把极小极大值回传

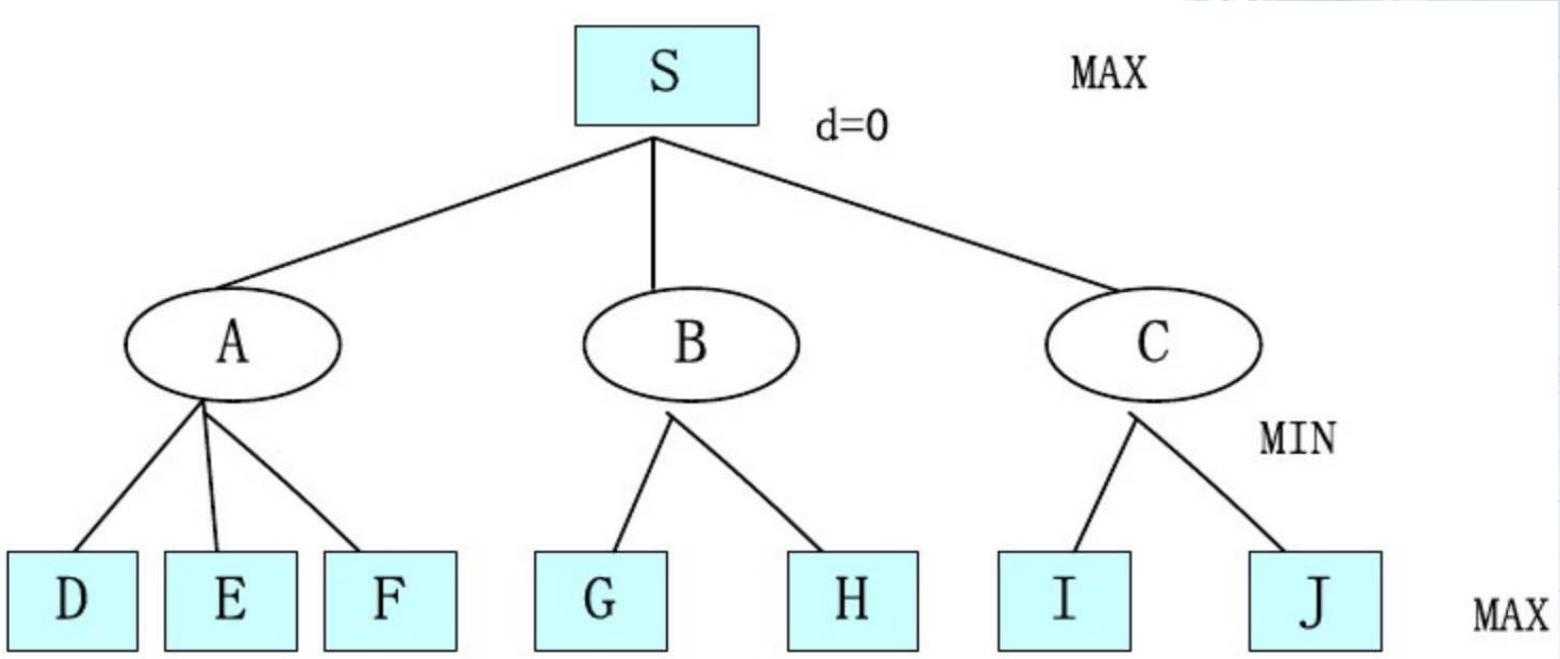
```
function MINIMAX-DECISION(state) returns an action
  return arg max_{a ∈ ACTIONS(s)} MIN-VALUE(RESULT(state, a))

function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s, a)))
  return v

function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← ∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s, a)))
  return v
```

极小极大搜索过程

- 对于人类下棋的方法：实际上采用的是一种试探性的方法
  - ✓ 首先假定走了一步棋，看对方会有那些应法，然后再根据对方的每-种应法，看我方是否有好的回应
  - ✓ 这一过程一直进行下去，直到若干步以后，找到了一个满意的走法为止
  - ✓ 初学者可能只能看一、两个回合，而高手则可以看几个，甚至十几个回合
- 极小极大搜索方法：模拟的就是人的这样一种思维过程

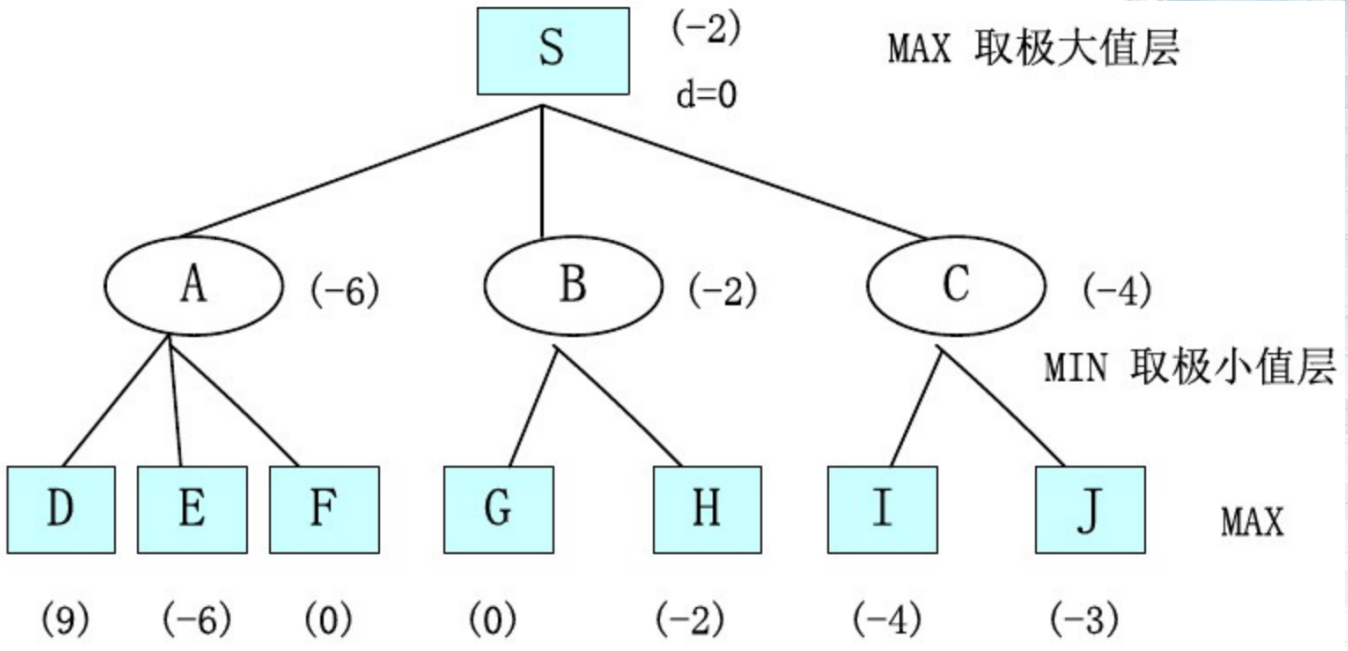
极小极大搜索过程

- 规定:顶节点深度d=0, MAX代表程序方，MIN代表对手方，MAX先走
- 例：一个考虑2步棋的例子



极小极大搜索过程

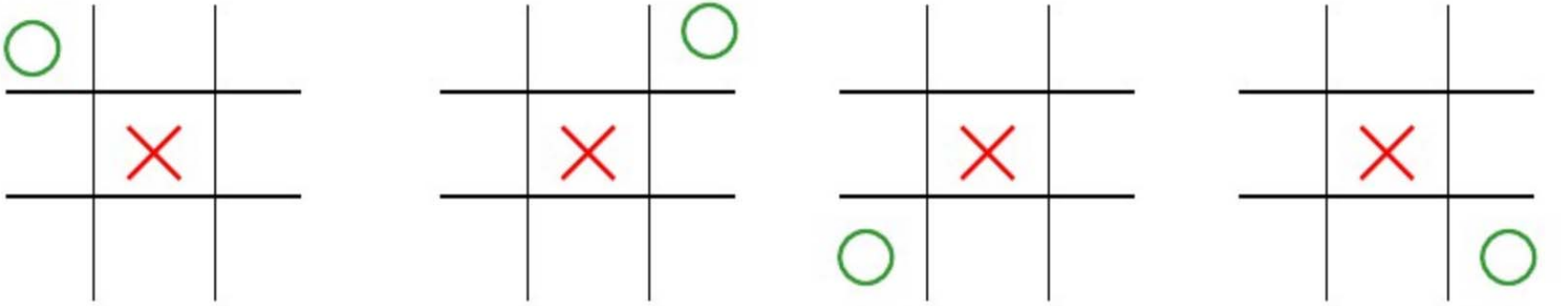
- 规定:顶节点深度d=0, MAX代表程序方，MIN代表对手方。MAX先走
- 例，一个考虑2步棋的例子



叶结点给出的数字是用效用函数计算得到

例：3×3棋盘的一字棋

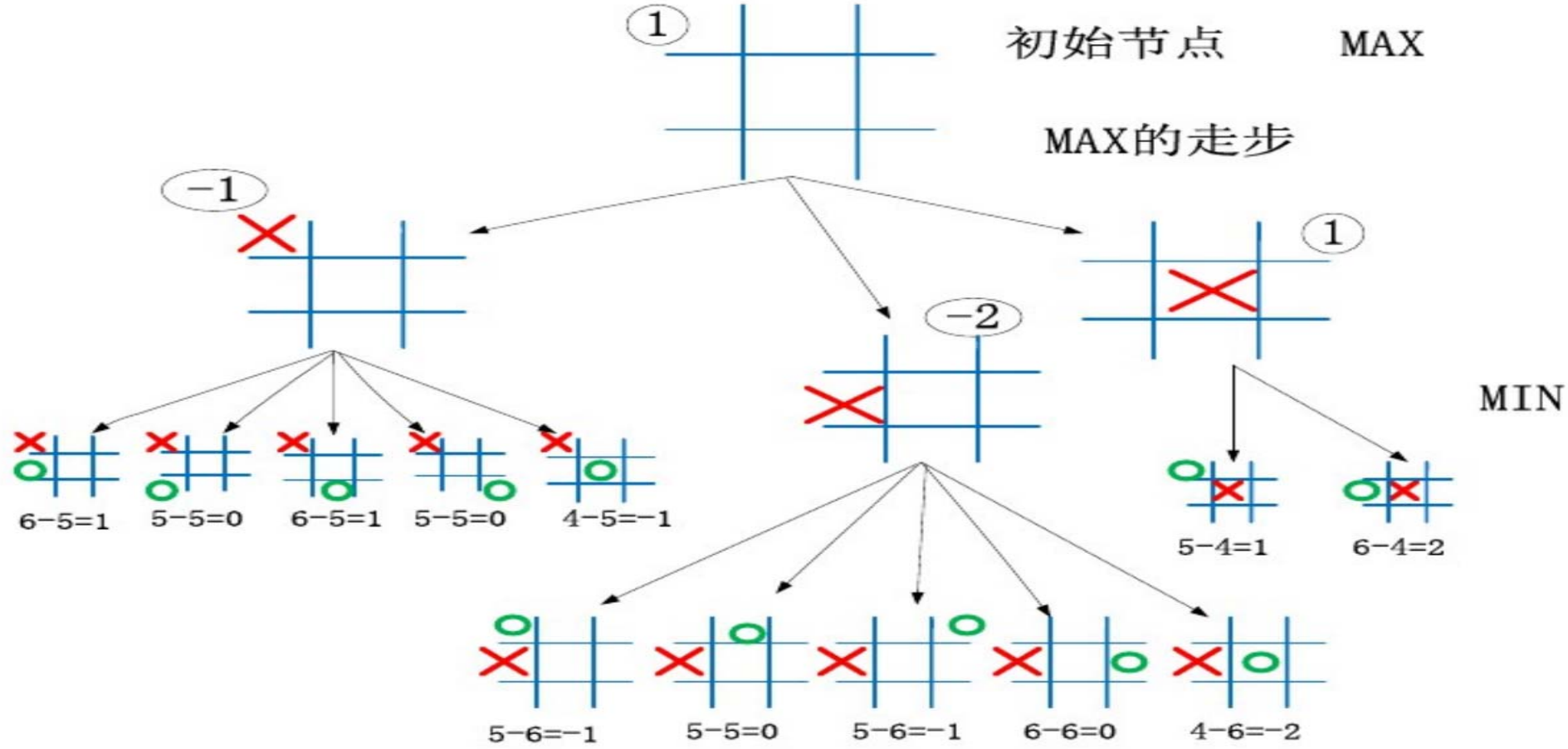
- 在搜索过程中，具有对称性的棋局认为是同一棋局，可以大大减少搜索空间。



对称棋局的例子

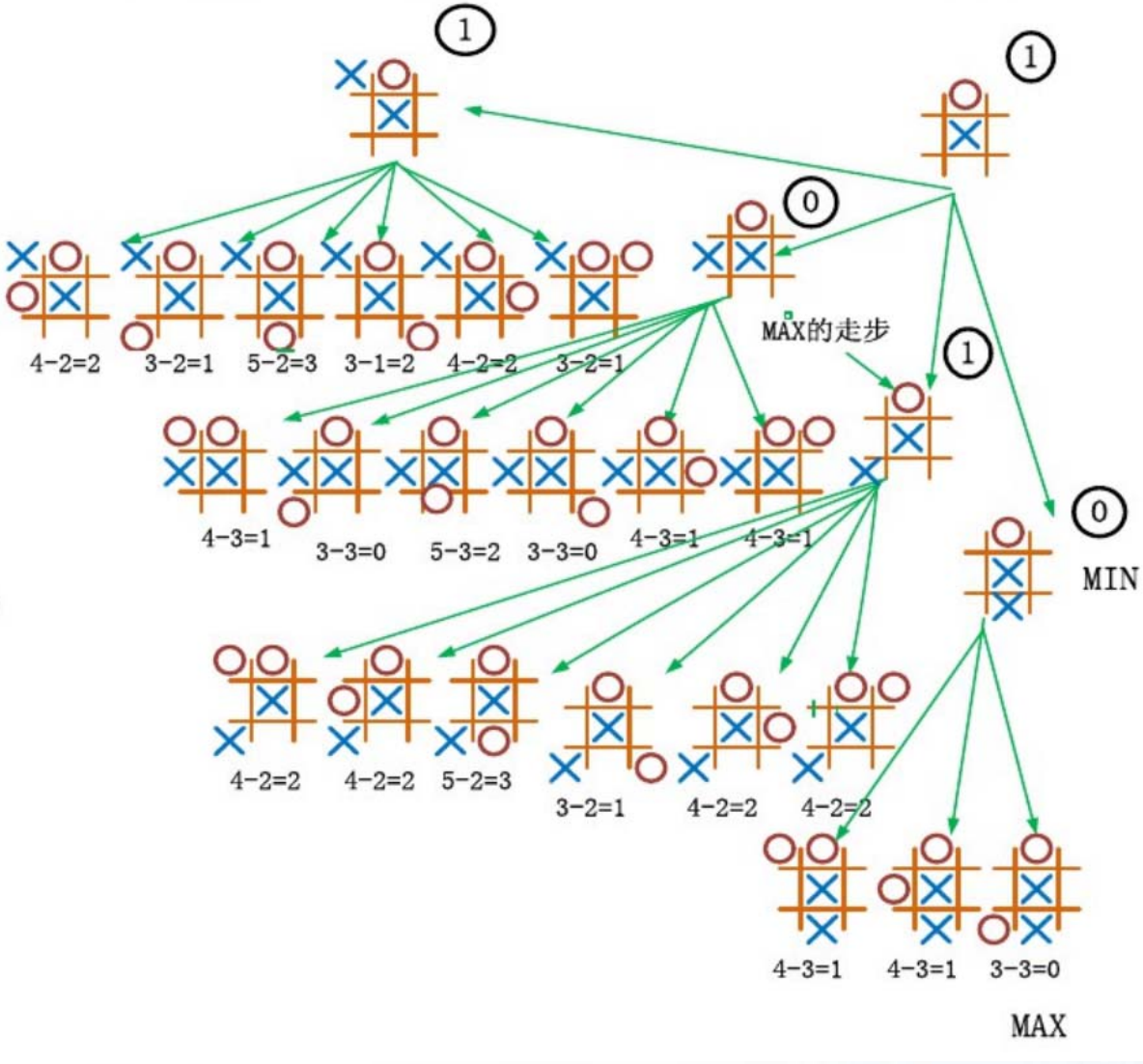
例：3×3棋盘的一字棋

- 假定考虑走两步的搜索过程，利用棋盘对称性的条件，则第一次调用算法产生的搜索树为：



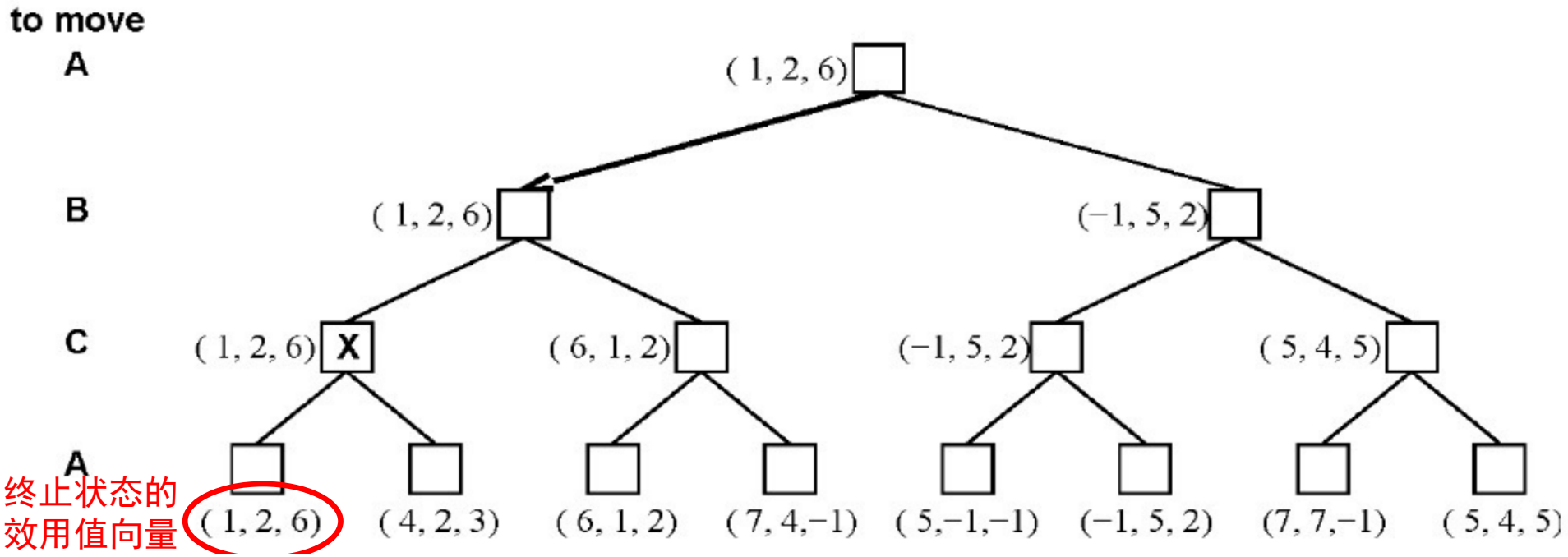
例：3×3棋盘的一字棋

- 假设MAX走完第一步后，MAX的对手是在X之上的格子下棋子，这时MAX在新格局下调用算法,第二次产生的搜索树为：



多人游戏中的最优决策

- 考虑非终止状态，考虑图中博弈树上标为X的结点
- 一般来讲，节点n的回传值是该游戏者在结点n选择的效用值最高的后继者的效用值向量



讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完整的实时决策

五、随机博弈

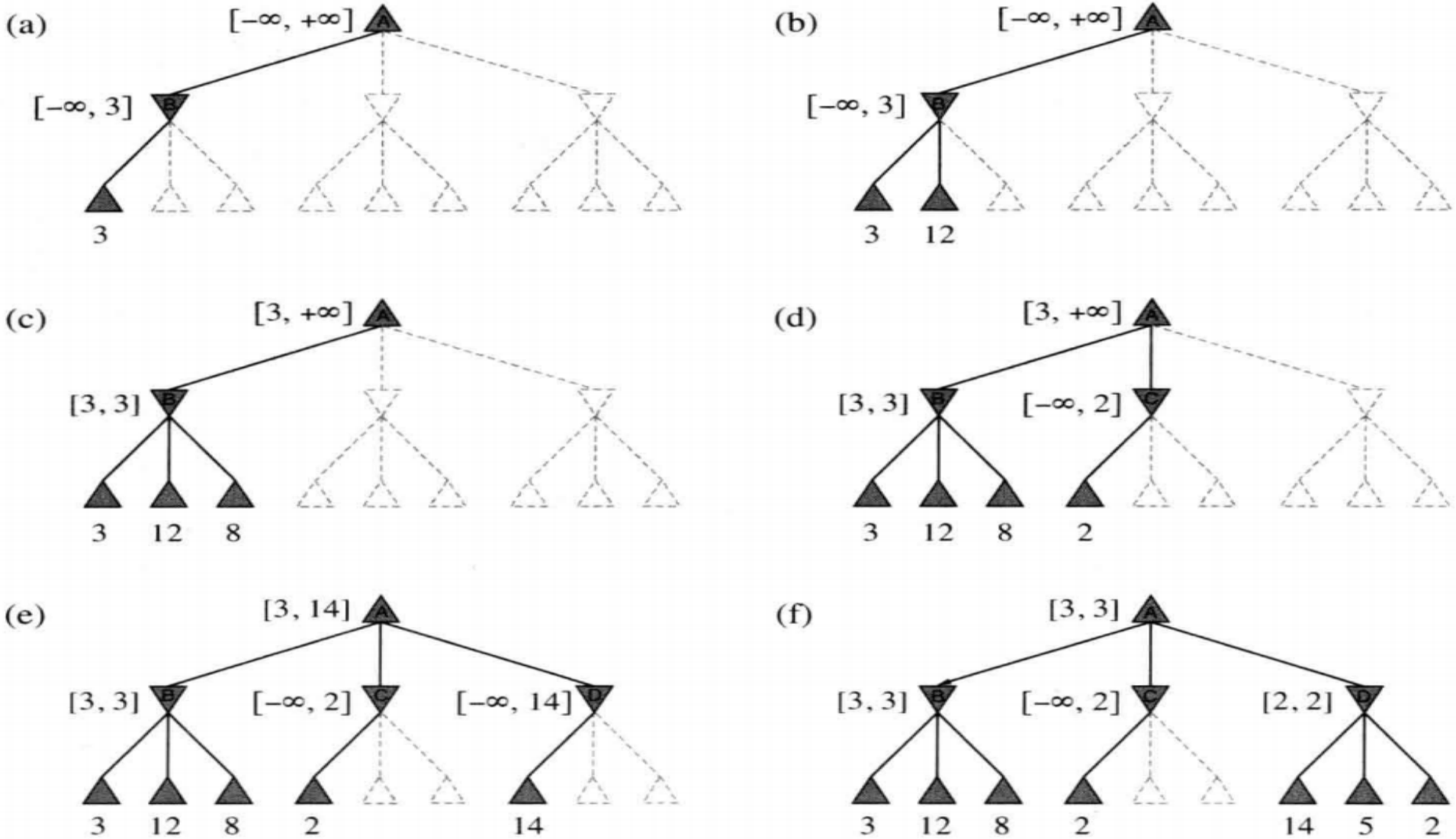
六、部分可观察的博弈

七、博弈程序发展现状

α-β剪枝的基本思想

- 把博弈树生成和倒推估值结合起来进行，再根据一定的条件判定，有可能尽早修剪掉一些无用的分枝
- 为了使生成和估值过程紧密结合，采用有界深度优先策略进行搜索
- 当生成达到规定深度的节点时，就立即计算效用函数，而一旦某个非端节点有条件确定其倒推值时就立即计算赋值

α-β剪枝


(a)
[−∞, +∞]
A
[−∞, 3]
B
3
(b)
[−∞, +∞]
A
[−∞, 3]
B
3
12
(c)
[3, +∞]
A
[3, 3]
B
3
12
8
(d)
[3, +∞]
A
[3, 3]
B
[−∞, 2]
C
3
12
8
2
(e)
[3, 14]
A
[3, 3]
B
[−∞, 2]
C
[−∞, 14]
D
3
12
8
2
14
(f)
[3, 3]
A
[3, 3]
B
[−∞, 2]
C
[2, 2]
D
3
12
8
2
14
5
2

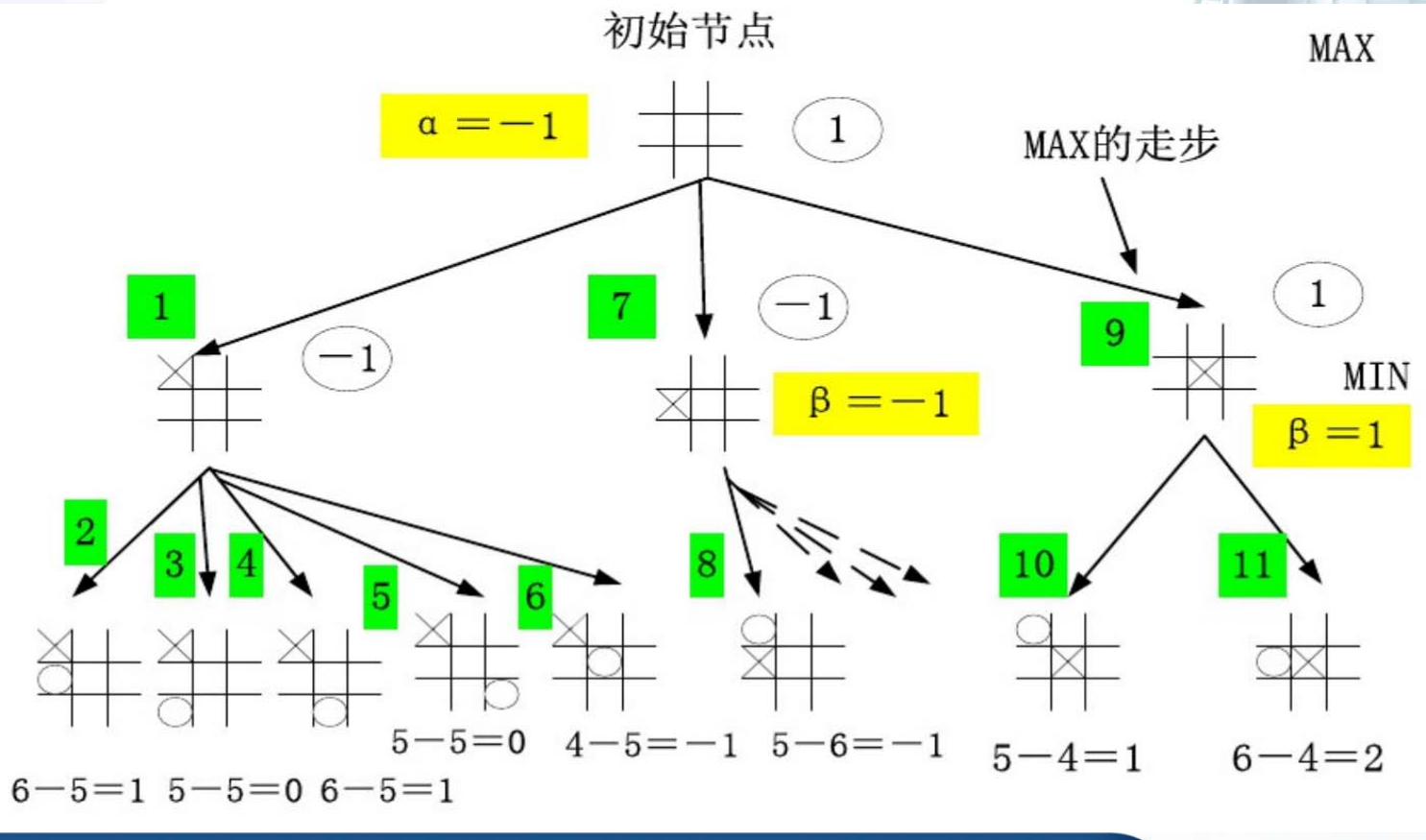
α-β剪枝算法

- α=到目前为止路径上发现的MAX的最佳（即极大值）选择
- β=到目前为止路径上发现的MIN的最佳（即极小值）选择
- 不断更新α和β值，当某个结点的值分别比目前的MAX的α或者MIN的β值更差，裁剪此结点剩下的分支（即终止递归调用）

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a), α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

α-β剪枝的基本思想

- 极大值层的下界值称为α
- 极小值层的上界值称为β

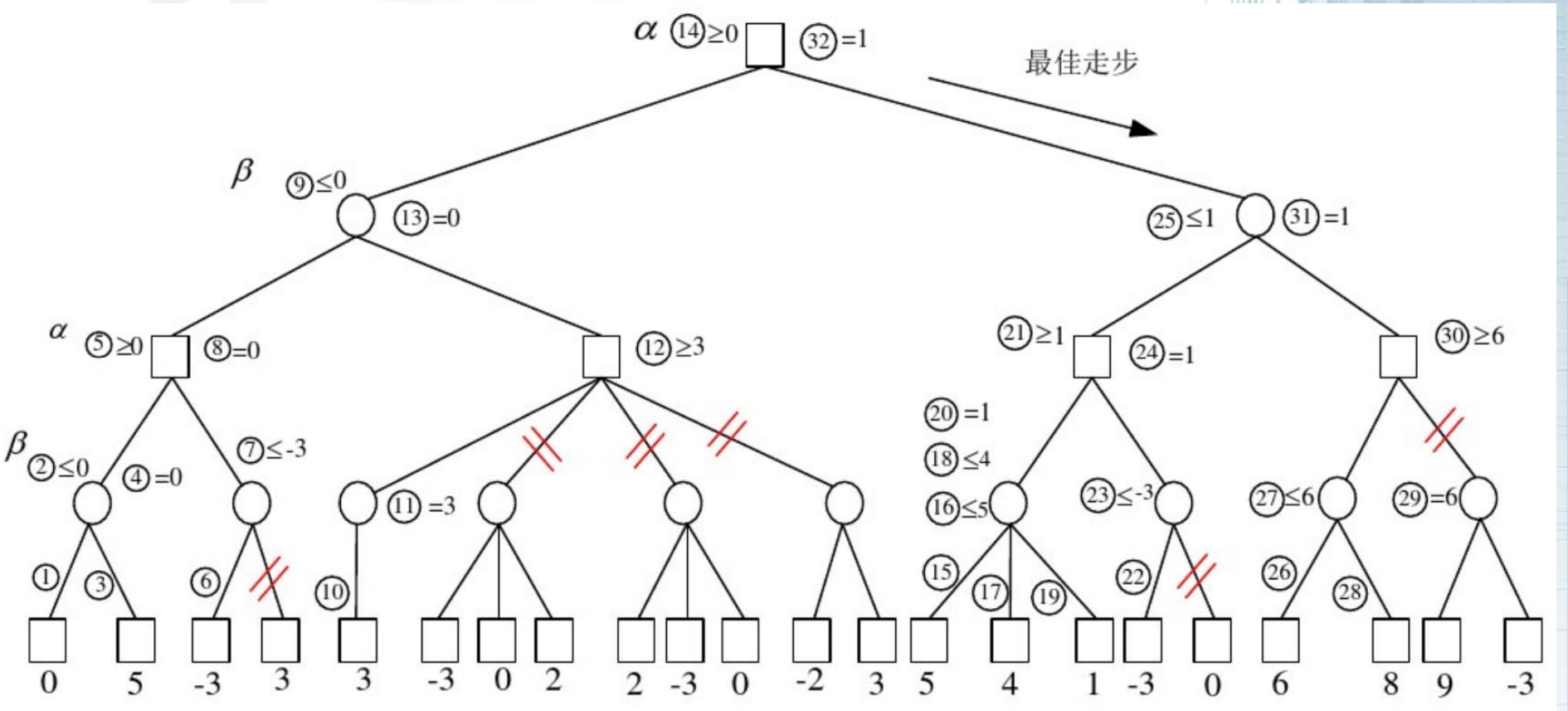


α-β搜索过程的剪枝规则

- α剪枝：若任一极小值层节点的β值小于或等于它任一先辈极大值层节点的α值，即α（先辈层）>= β（后继层），则可中止该极小值层中这个MIN节点以下的搜索过程。这个MIN节点最终的倒推值就确定为这个β值
- β剪枝：若任一极大值层节点的α值大于或等于它任一先辈极小值层节点的β值，即α（后继层）>= β（先辈层），则可以中止该极大值层中这个MAX节点以下的搜索过程。这个MAX节点的最终倒推值就确定为这个α值

α-β搜索过程的博弈树

- α-β剪枝举例



剪枝的效率问题

- 比较后得出最佳α-β搜索技术所生成深度为D处的端结点数约等于不用α-β搜索技术所生成深度为D/2处的端结点数

  ✓ 这就是说，在一般条件下使用α-β搜索技术，在同样的资源限制下，可以向前考虑更多的走步数，这样选取当前的最好优先走步，将带来更大的取胜优势

讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完整的实时决策

五、随机博弈

六、部分可观察的博弈

七、博弈程序发展现状

不完整的实时决策

- 启发式极小极大值，s为状态，d为最大深度

$$\text{H-MINIMAX}(s, d) = \begin{cases} \text{EVAL}(s) & \text{如果 CUTOFF-TEST}(s,d)\text{为真} \\ \max_{a \in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s, a), d+1) & s \text{ 为 MAX 结点} \\ \min_{a \in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s, a), d+1) & s \text{ 为 MIN 结点} \end{cases}$$

- 回顾极小极大值

$$\text{MINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & s \text{ 为终止状态} \\ \max_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & s \text{ 为 MAX 结点} \\ \min_{a \in Actions(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & s \text{ 为 MIN 结点} \end{cases}$$

评估函数

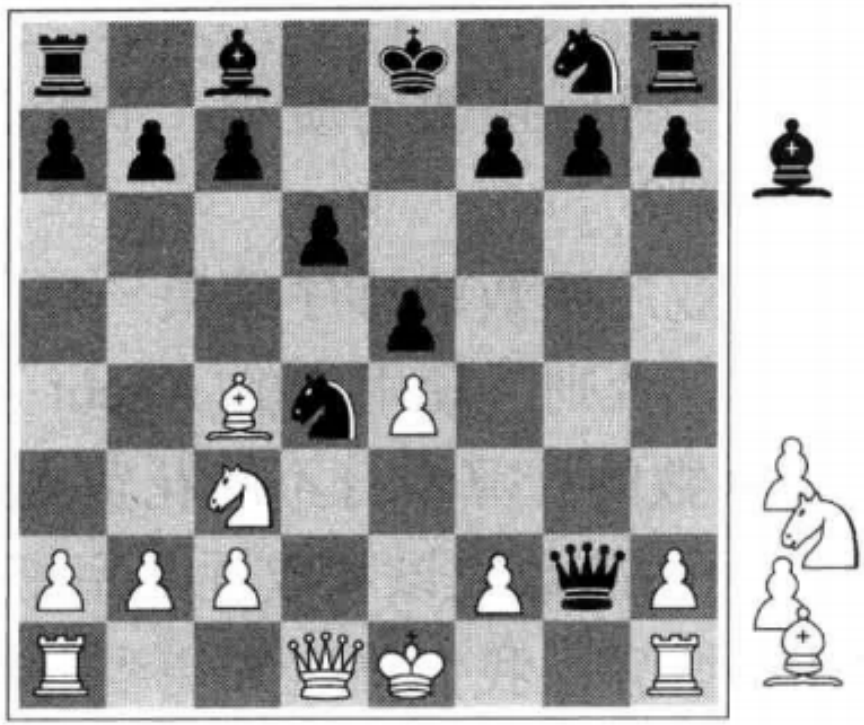
- 在实际应用中，往往有太多分类，需要太多的经验去估计所有取胜可能。所以，大多数评估函数会分别计算每个特征单独的数值贡献，然后把它们结合起来找到总数值
- 评估函数表示如下，数学上称为加权线性函数

$$EVAL(\mathrm{s}) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s) = \sum_{i=1}^{n} w_i f_i(s)$$

其中，w_i是权值，f_i是棋局的某个特征。对于国际象棋，f_i可能是棋盘上每种棋子的数目，w_i可能是每种棋子的价值（如兵为1，象为3，等等）

评估函数

例如：黑棋多1个马2个兵，应该取胜



(a) White to move

评估函数

$$EVAL(s) = \sum_{i=1}^{n} w_i f_i(s)$$

- 加权线性评价函数的假设：每个特征的贡献独立于其它特征的值
  - ✓ 假设太强
  - ✓ 例如，象赋予3分忽略了象在残局中能够发挥更大作用的事实
- 当前国际象棋和其它游戏的程序也采用非线性的特征组合

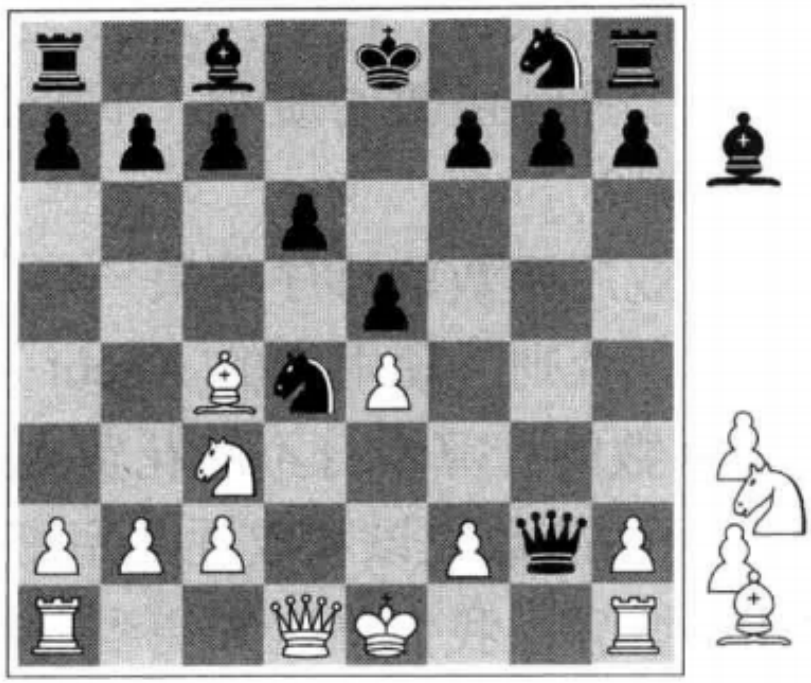
截断搜索

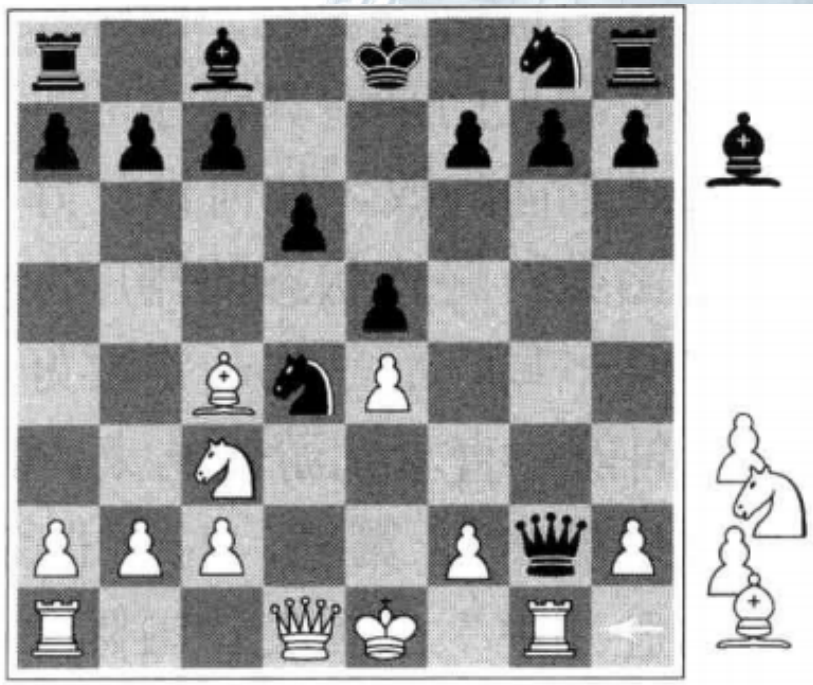
- 下一步是修改ALPHA-BETA-SEARCH，当适合阶段搜索时调用启发式函数EVAL
- 用程序替换 `if CUTOFF-TEST(state, depth) then return EVAL(state)`

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v

function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a), α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

截断搜索

- 假设棋局到达了深度限制，此时黑方有一马两兵的优势，根据这个状态的启发式函数值，认为这个状态导致黑方获胜
- 其实下一步白方可以毫无意外地吃掉黑方皇后，因此这个棋局时白棋赢，需要向前多看一步才能预测





截断搜索

- 评估函数只适用于那些静态棋局
  - ✓ 静态棋局：评估值在很近的未来不会出现大的摇摆变化棋局
  - ✓ 例如，在国际象棋中，有很好的吃招的棋局对于只统计子力的评估函数来说就不是静态的
- 非静态的棋局可以进一步扩展直到变为静态棋局，这种额外的搜索称为静态搜索
  - ✓ 有时候只考虑某些类型的招数，诸如吃子，能够快速地解决棋局的不确定性

向前剪枝

- 假设已经实现：
  - ✓ 国际象棋的评价函数
  - ✓ 使用静止搜索的合理截断测试
  - ✓ 一个很大的调换表（存储以前见过的棋局的哈希表一般被称做调换表)
- 若在“最新的PC”上可每秒生成和评估约一百万个节点，则允许我们在标准的时间控制下（每步棋3分钟）对每步棋可搜索约2亿个结点
  - ✓ 国际象棋的分支因子平均为35，35^5约为5000万
  - ✓ 如果使用极大极小搜索，只能向前预测5层，很容易被平均水平的人类棋手欺骗
  - ✓ 如果使用alpha-beta搜索，可以预测大约10层，接近专业棋手水平

中国科学院
自动化研究所
INSTITUTE OF AUTOMATION
CHINESE ACADEMY OF SCIENCES

讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完整的实时决策

五、随机博弈

六、部分可观察的博弈

七、博弈程序发展现状

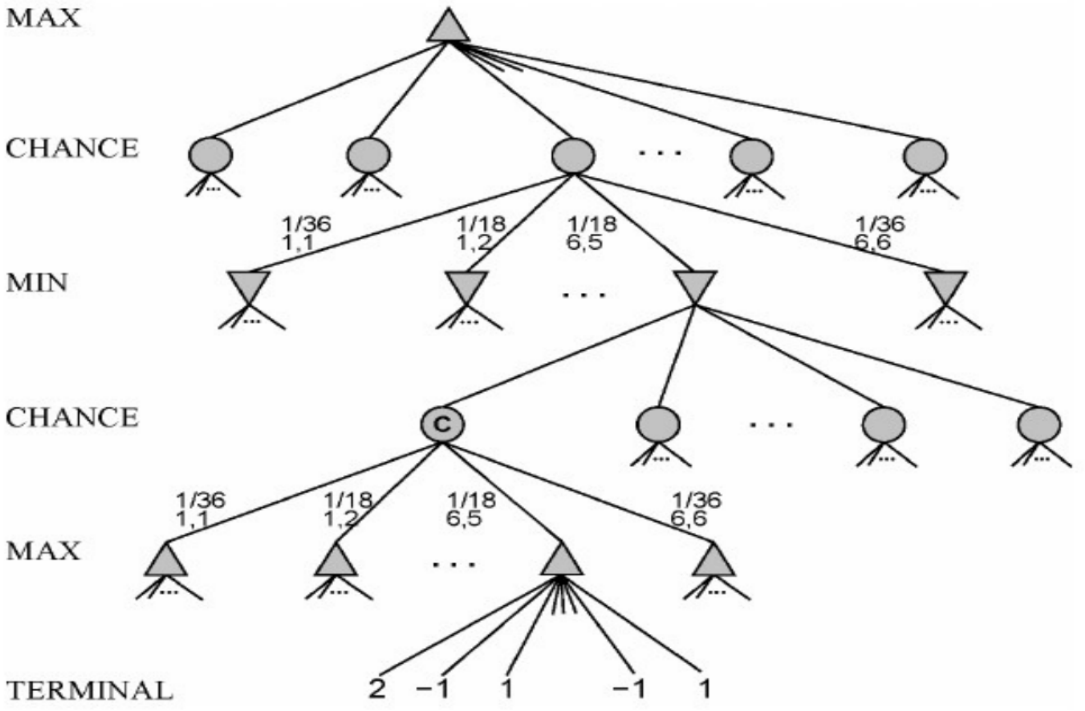
随机博弈

- 现实生活中，很多不可预知的外部事件会把我们推到无法预计的情景中
- **随机博弈**：许多博弈用随机因素，如掷骰子，来反映这种不可预测性
  - 西洋双陆棋就是组合运气和技巧的游戏

双陆棋的博弈树

- 博弈树中除了MAX和MIN结点之外，还必须包括机会结点
  - 子结点代表可能的掷骰子结果：骰子数及概率

- 两个骰子总共有21种骰法，其中:
- 6个有相同骰子数的组合(1/36概率)
- 15个不同骰子数的组合(1/18概率)



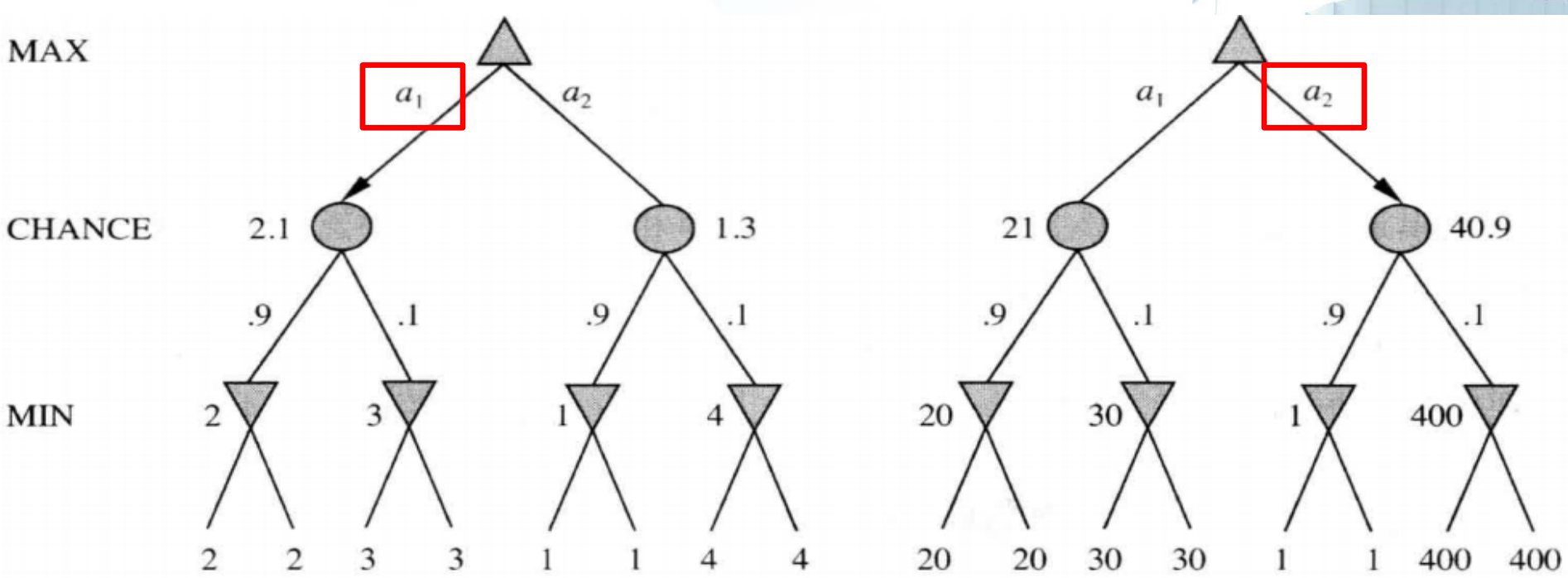
极小极大值->期望极小极大值

- 把确定性博弈中的极小极大值一般化为包含机会结点的博弈的期望极小极大值

$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & s \text{ 为终止状态时} \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & s \text{ 为 MAX 结点时} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & s \text{ 为 MIN 结点时} \\ \sum_r P(r)\,\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & s \text{ 为机会结点时} \end{cases}$$

其中，r表示可能的掷骰子结果(或其他偶然事件)，RESULT(s,r)仍是状态s，附加了掷骰子结果r

机会博弈中的评估函数



- 在保持顺序不变的情况下，叶结点赋值的变换导致了最佳招数的改变
- 评估函数与棋局获胜概率（期望效用值）成线性变换

期望极小极大值的复杂度

- 算法的时间复杂度为 $O(b^m n^m)$ ，其中n为不同的掷骰子结果的数目
- 由于额外代价的存在使得向前考虑得很远是不现实的
- 可以对机会博弈树适用类似**α-β**剪枝技术？
  - ✓ 限制效用函数的取值范围->不用检查机会结点的所有子结点就可以设置机会结点的上界

讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完美的实时决策

五、随机博弈

六、部分可观察的博弈

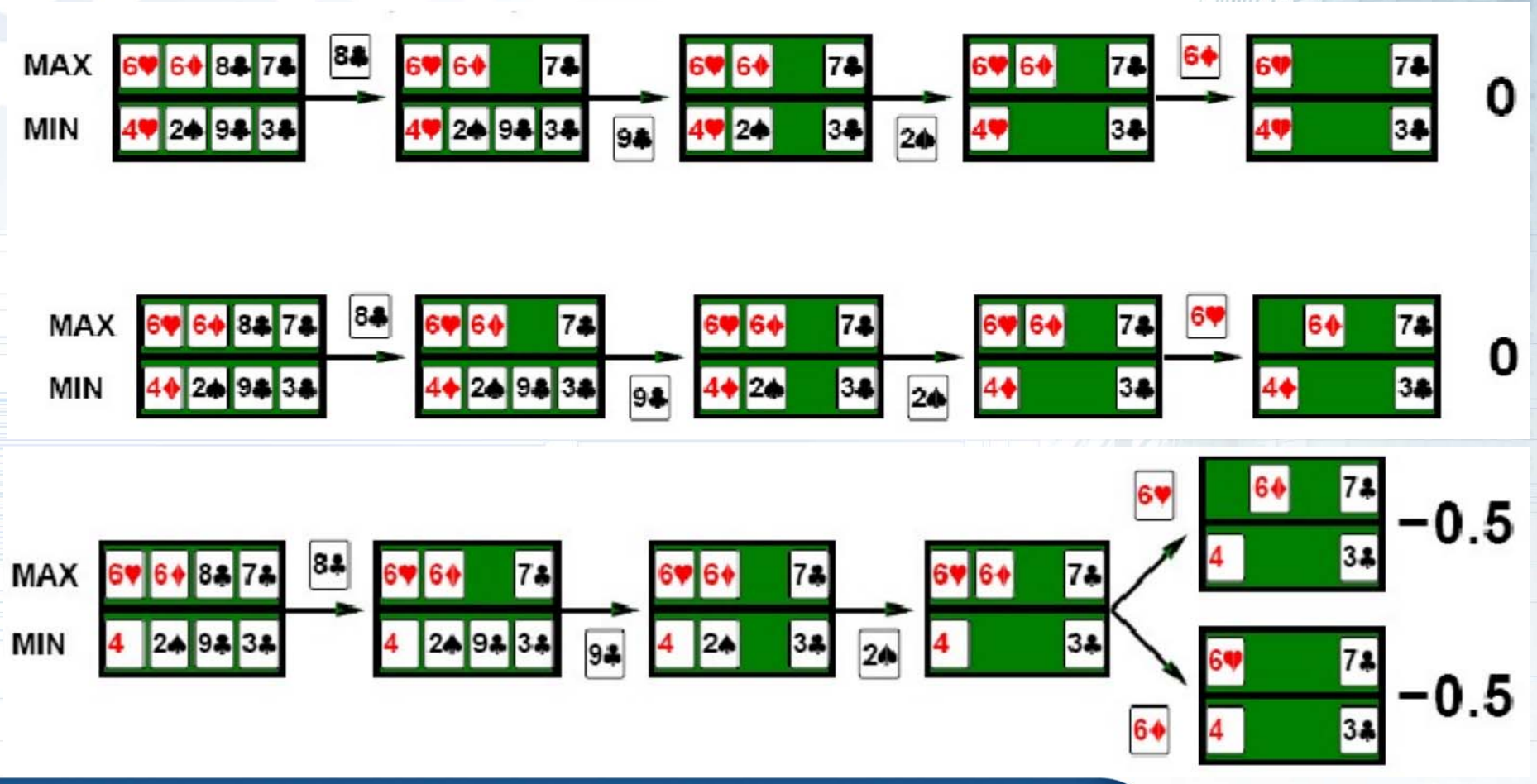
七、博弈程序发展现状

部分可观察的博弈

- 国际象棋通常被视为微型战争，但与现实战争相比，至少缺乏一点特征，部分可观察性
  - 敌方的存在和部署通常是不知道的
  - 战争中通常会有侦察兵和间谍收集信息
  - 隐瞒信息或虚张声势搞乱敌方
- 军旗：国际象棋的部分可观察变种，棋子可以移动但是对方看不见棋子是什么

牌类游戏

- 发牌是随机的，选手拿到的牌别人无法知道



牌类游戏

- 随机算法，考虑未知牌的所有可能应对策略。假设应对策略s发生的概率是$P(s)$

$$\arg\max_a \sum P(s)MINIMAX(RESULT(\mathrm{s},a))$$

- 大多数牌类游戏，可能的应对策略数目相当大，使用蒙特卡洛近似：不考虑所有可能组合，随机采样N个样本，设组合s在样本中出现的概率与 $P(s)$成正比

$$\arg\max_a \frac{1}{N}\sum_{i=1}^{n} P(s)MINIMAX(RESULT(s,a))$$

讲课内容

一、博弈

二、博弈中的优化决策

三、α-β剪枝

四、不完整的实时决策

五、随机博弈

六、部分可观察的博弈

七、博弈程序发展现状

博弈程序发展现状

- 国际象棋：IBM深蓝打败了国际象棋大师
- 西洋跳棋：Chinook在1990年的简化比赛中战胜了长久以来的人类冠军
- 奥赛罗，也叫翻转棋：1997年的Logistello程序，以6比0击败了人类冠军
- 西洋双陆棋：TD-GAMMON稳定的排在世界前列（强化学习与神经网络结合）
- 围棋：2016年阿尔法围棋（AlphaGo）击败人类职业围棋选手,战胜围棋世界冠军

感谢同学们聆听

欢迎讨论与交流