课程《人工智能原理与技术》

同济大学 TONGJI UNIVERSITY

第三章搜索探寻与问题求解

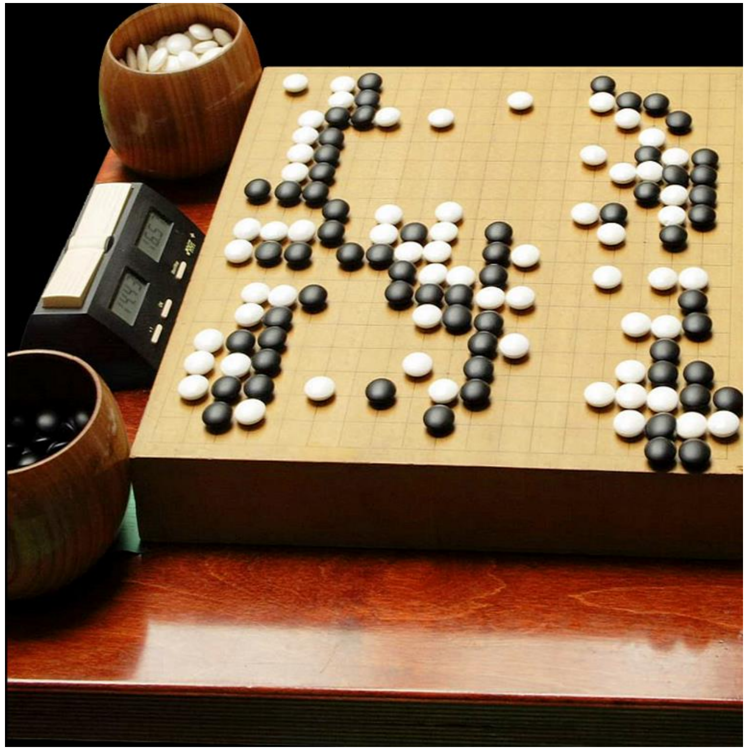
---对抗搜索

同濟大學
TONGJI UNIVERSITY

目录

中国古代博弈思想

- **子曰**：饱食终日，无所用心，难矣哉！不有博弈者乎？为之，犹贤乎已。 ——《论语•阳货》
- **朱熹**集注曰：“博，局戏；弈，围棋也。”；颜师古注：“博，六博；弈，围碁也。”
- 古语博弈所指下围棋，围棋之道蕴含古人谋划策略的智慧。
- 略观围棋，法于用兵，怯者无功，贪者先亡。——《围棋赋》
- **《孙子兵法》**等讲述兵书战法的古代典籍更是凸显了古人对策略的重视。



中国古代博弈例子

- ……齐将田忌善而客待之。忌数与齐诸公子驰逐重射。孙子见其马足不甚相远，马有上、中、下辈。于是孙子谓田忌曰：“君弟重射，臣能令君胜。”田忌信然之，与王及诸公子逐射千金。及临质，孙子曰：“今以君之下驷与彼上驷，取君上驷与彼中驷，取君中驷与彼下驷。”既驰三辈毕，而田忌一不胜而再胜，卒得王千金。于是忌进孙子于威王。威王问兵法，遂以为师。 ——《史记•孙子吴起列传》

表 7.1　齐威王与田忌赛马所采取的不同对局

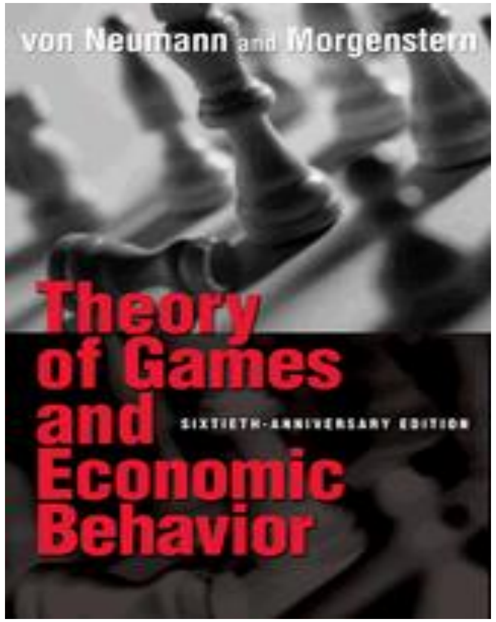
对局	齐王马	田忌马	结果
1	A+	A−	齐王胜
2	B+	B−	齐王胜
3	C+	C−	齐王胜

对局	齐王马	田忌马	结果
1	A+	C−	齐王胜
2	B+	A−	田忌胜
3	C+	B−	田忌胜

“以己之长，攻彼之短” 的博弈思想

现代博弈论的诞生

- **博弈论（game theory）**，又称对策论。
- 博弈行为：带有相互竞争性质的主体，为了达到各自目标和利益，采取的带有对抗性质的行为。
- 博弈论主要研究博弈行为中**最优的对抗策略及其稳定局势**，协助人们在一定规则范围内寻求最合理的行为方式。
- 1944年冯·诺伊曼与奥斯卡·摩根斯特恩合著**《博弈论与经济行为》**，以数学形式来阐述博弈论及其应用，标志着现代系统博弈理论的初步形成，冯·诺伊曼被称为现代博弈论之父。



John von Neumann(1903-1957), Oskar Morgenstern(1902-1977), *Theory of Games and Economic Behavior*, Princeton University Press, 1944

同濟大學 TONGJI UNIVERSITY

博弈论的相关概念

- **参与者或玩家（player）**：参与博弈的决策主体
- **策略（strategy）**：参与者可以采取的行动方案，是一整套在采取行动之前就已经准备好的完整方案。
- 某个参与者可采纳策略的全体组合形成了策略集（strategy set）。所有参与者各自采取行动后形成的状态被称为局势（outcome）。如果参与者可以通过一定概率分布来选择若干个不同的策略，这样的策略称为混合策略（mixed strategy）。若参与者每次行动都选择某个确定的策略，这样的策略称为纯策略（pure strategy）
- **收益（payoff）**：各个参与者在不同局势下得到的利益。混合策略意义下的收益应为期望收益（expected payoff）。
- 规则（rule）：对参与者行动的先后顺序、参与者获得信息多少等内容的规定
- **博弈论研究的范式：** 建模者对参与者（player）规定可采取的策略集(strategy sets)和取得的收益，观察当参与者选择若干策略以最大化其收益时会产生什么结果

两害相权取其轻，两利相权取其重

博弈论：囚徒困境（prisoner’s dilemma）

- 1950年，兰德公司的梅里尔·弗勒德和梅尔文·德雷希尔拟定了相关困境理论，后来美国普林斯顿大学数学家阿尔伯特·塔克以“囚徒方式”阐述：
  - 警方逮捕了共同犯罪的甲、乙两人，由于警方没有掌握充分的证据，所以将两人分开审讯：
  - 若一人认罪并指证对方，而另一方保持沉默，则此人会被当即释放，沉默者会被判监禁10年
  - 若两人都保持沉默，则根据已有的犯罪事实（无充分证据）两人各判半年
  - 若两人都认罪并相互指证，则两人各判5年

	乙沉默（合作）	乙认罪（背叛）
甲沉默（合作）	二人各 服刑半年	乙被释放， 甲服刑10年
甲认罪（背叛）	甲被释放， 乙服刑10年	二人各 服刑5年

- **参与者**：甲、乙
- **规则**：甲、乙两人分别决策，无法得知对方的选择
- **策略集**：认罪、沉默（纯策略）

- 局势及对应收益（年）
  - 甲认罪：0　　乙沉默：-10
  - 甲认罪：-5　　乙认罪：-5 （均衡解）
  - 甲沉默：-10　　乙认罪：0
  - 甲沉默：-0.5　　乙沉默：-0.5 （最优解）

- 在囚徒困境中，**最优解**为两人同时沉默，但是两人实际倾向于选择同时认罪（**均衡解**）

博弈的分类

- **参与者或玩家（player）合作博弈与非合作博弈**

  合作博弈（cooperative game）：部分参与者可以组成联盟以获得更大的收益

  非合作博弈（non-cooperative game）：参与者在决策中都彼此独立，不事先达成合作意向

- **静态博弈与动态博弈**

  静态博弈（static game）：所有参与者同时决策，或参与者互相不知道对方的决策

  动态博弈（dynamic game）：参与者所采取行为的先后顺序由规则决定，且后行动者知道先行动者所采取的行为

- **完全信息博弈与不完全信息博弈**

  完全信息（complete information）：所有参与者均了解其他参与者的策略集、收益等信息

  不完全信息（incomplete information）：并非所有参与者均掌握了所有信息

本章的博弈

- 本章所讲的博弈：主要指的是类似于象棋这样的游戏问题。
- 这类问题有以下一些特点：

① 双人对弈，对垒的双方轮流走步。

② 信息完备，对垒双方所得到的信息是一样的，不存在一方能看到，而另一方看不到的情况。

③ 零和。即对一方有利的棋，对另一方肯定是不利的，不存在对双方均有利、或均无利的棋。对弈的结果是一方赢，而另一方输，或者双方和棋。

本章的博弈

- 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，例如掷币等。
  - 例如：西洋双陆棋。

博弈的形式化描述

- S_0：**初始状态**，规范游戏开始时的情况。
- PLAYER(s)：定义此时该谁行动。
- ACTIONS(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**，游戏结束返回真，否则返回假。游戏结束的状态称为**终止状态**。
- UTILITY(s,p)：**效用函数**（也可称为目标函数或收益函数），定义游戏者 p 在终止状态 s 下的数值。在国际象棋中，结果是赢、输或平，分别赋予数值+1，0，或 1/2。有些游戏可能有更多的结果，例如双陆棋的结果是从 0 到+192。**零和博弈**是指在同样的棋局实例中所有棋手的总收益都一样的情况。国际象棋是零和博弈，棋局的收益是 0+1，1+0 或 1/2+1/2。“常量和”可能是更好的术语，但称为零和更传统，可以将这看成是下棋前每个棋手都被收了 1/2 的入场费。

博弈问题描述

- 两个玩家: MAX 和 MIN
- 任务：给MAX找一“最佳”行动
- 假定MAX 先行, 随后两方交替移动，直到游戏结束。
- 游戏结束时，给优胜者加分，给失败者罚分。
- MAX结点：下一步轮到MAX走子的状态
- MIN结点：下一步轮到MIN走子的状态

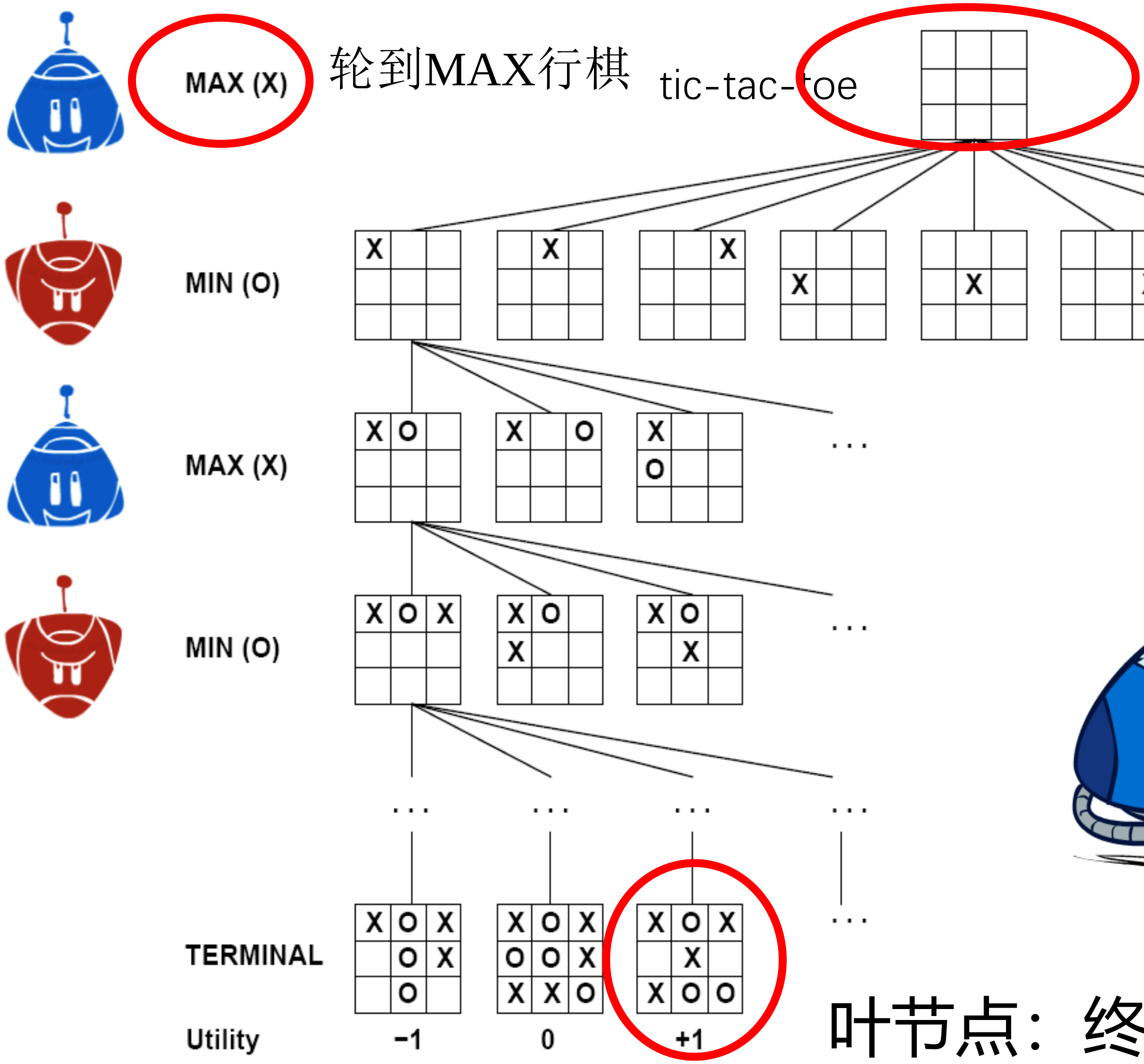
博弈树

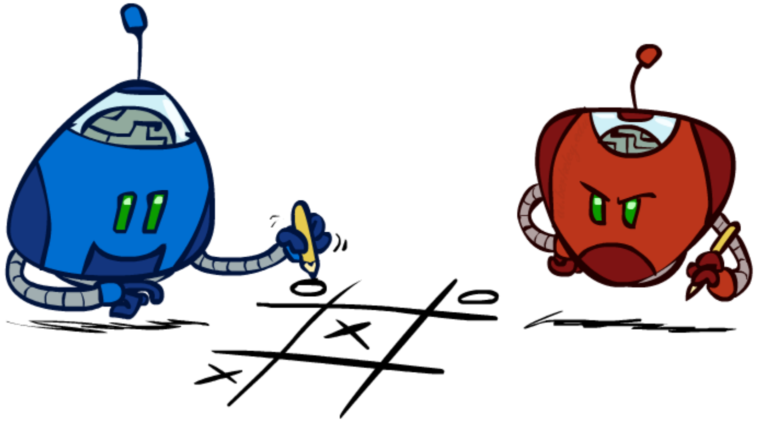
- 评估最佳第一步行棋
- 通常, 采用以下约定分析博弈树
  - 对MAX 有益的位置，对应的评估函数值为正
  - 对MIN 有益的位置，对应的评估函数值为负

博弈树

轮到MAX行棋

初始状态





叶节点：终止状态对于MAX的效用值

同济大学 TONGJI UNIVERSITY

博弈例子

- 分钱币博弈是一个分钱币的游戏。
  - 有一堆数目为N的钱币，由两位选手轮流进行分堆，要求每个选手每次只把其中某一堆分成数目不等的两小堆。
  - 例如，选手甲把N分成两堆后，轮到选手乙就可以挑其中一堆来分。
  - 如此进行下去，直到有一位选手先无法把钱币再分成不相等的两堆时就得认输。

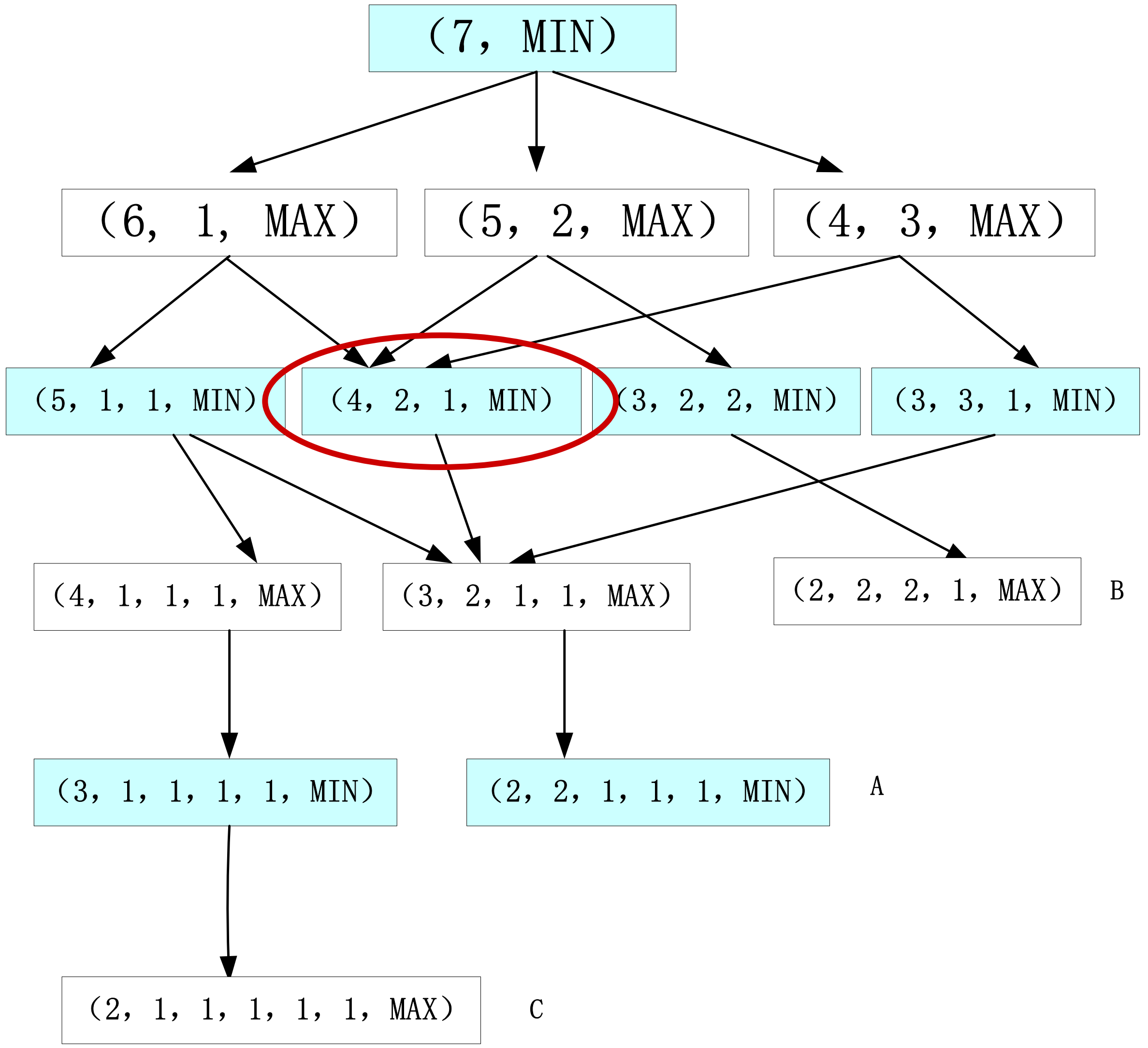
- 对于这样的简单博弈问题，可以生成出它的**状态空间图**。这样就有可能从状态空间图中找出取胜的策略。

博弈例子

- 当初始钱币数为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



博弈例子

- 搜索策略要考虑的问题：
    - 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，这是一个与的含意。
        - 因此含有MAX符号的节点可看成与节点。
    - 对MAX走步后的每一个MIN节点，只须证明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。
- 因此，对弈过程的搜索图就呈现出与或图表示的形式。

博弈例子

- 寻找MAX的取胜策略和求与或图的解图相对应。
  - MAX要取胜，必须对所有与节点取胜，但只需对一个或节点取胜，这就是一个解图。
- 因此，寻找一种取胜的策略就是搜索一个解图的问题，解图就代表一种完整的博弈策略。
- 对于分钱币这种较简单的博弈，或者复杂博弈的残局，可以用类似于与或图的搜索技术求出解图，解图代表了从开局到终局任何阶段上的弈法。
  - 显然这对许多博弈问题是不可能实现的。例如，中国象棋，国际象棋和围棋等。

博弈例子

- 对于复杂的博弈问题，因此即使用了强有力的启发式搜索技术，也不可能使分枝压到很少。
  - 因此，这种完全取胜策略（或和局）必须丢弃。
- 应当把目标确定为寻找一步好棋，等对手回敬后再考虑寻找另一步好棋这种实际可行的实用策略。
  - 这种情况下每一步的结束条件可根据时间限制、存储空间限制或深度限制等因素加以确定。
  - 搜索策略可采用宽度、深度或启发式方法。一个阶段搜索结束后，要从搜索树中提取一个优先考虑的"最好的"走步，这就是实用策略的基本点。

目录

Minimax搜索：对抗搜索（adversarial search）或博弈搜索（game search）

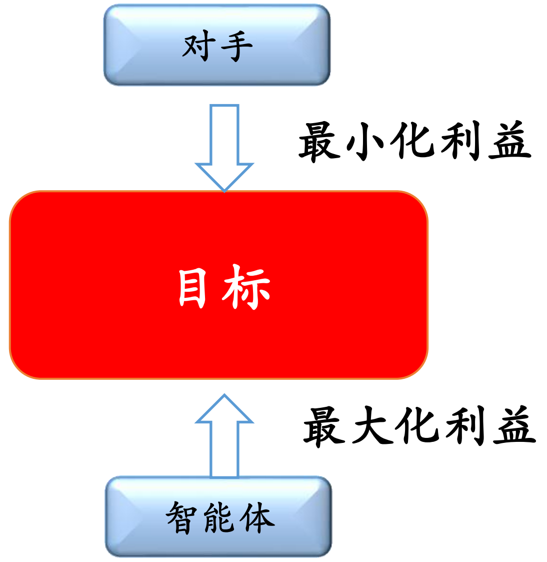
两个智能体同处于一个竞争的环境，智能体之间通过竞争来实现各自相反目标，即一方想**最大化自身的利益**，而另一方则想**最小化对手的利益**。通俗地说，对抗搜索的过程就是两个智能体各自选择对自己有利的策略。

狭路相逢勇者胜

勇者相逢智者胜

智者相逢德者胜

德者相逢**道者胜**



极小极大搜索过程

- 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一种应法，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

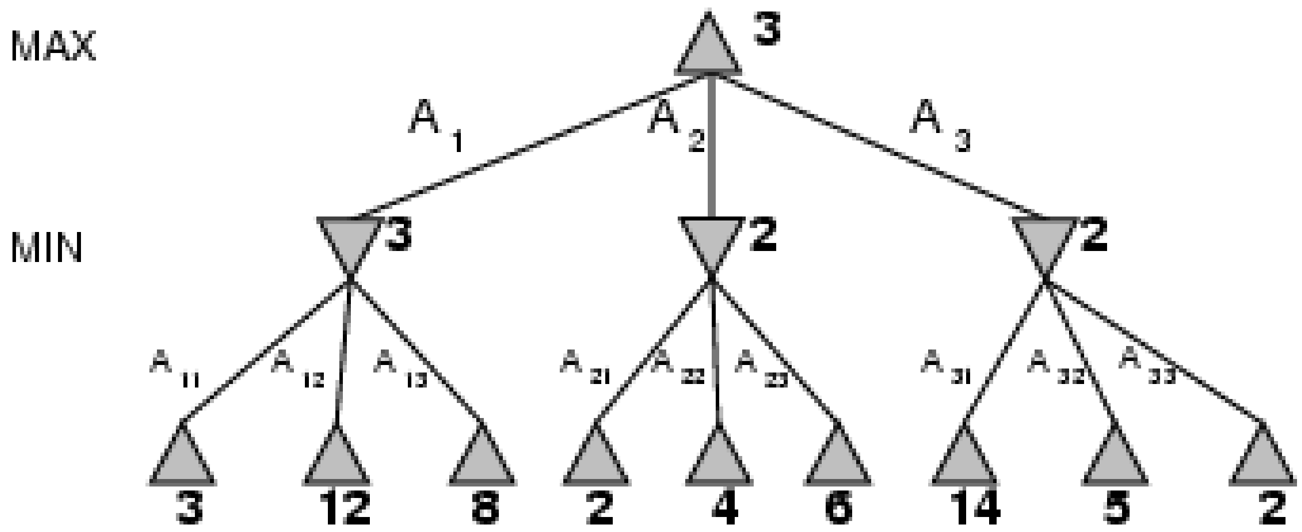
- 极小极大搜索策略是考虑双方对弈若干步之后，从可能的走步中选一步相对好棋的走法来走，即在有限的搜索深度范围内进行求解。

- 定义一个静态估计函数f，以便对棋局的势态（节点）作出优劣估值。
  - 这个函数可根据势态优劣特征来定义（主要用于对端节点的“价值”进行度量）。
  - 一般规定：有利于MAX的势态，f（p）取正值；有利于MIN的势态，f（p）取负值；势均力敌的势态，f（p）取0值。
  - 若f（p）＝＋∞，则表示MAX赢，若f（p）＝－∞，则表示MIN赢。

- 注意：不管设定的搜索深度是多少层，经过一次搜索以后，只决定了我方一步棋的走法。
  - 等到对方回应一步棋之后，需要在新的棋局下重新进行搜索，来决定下一步棋如何走。
- 极小极大过程是一种假定对手每次回应都正确的情况下，如何从中找出对我方最有利的走步的搜索方法。

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。



最后一层端节点给出的数字是用静态估计函数f（p）计算得到。

极小极大搜索算法

TONGJI UNIVERSITY 同濟大學

① T:＝（s，MAX），OPEN:＝（s），CLOSED:＝（ ）；开始时树由初始节点构成，OPEN表只含有s。

② LOOP1: IF OPEN＝（ ）THEN GO LOOP2；

③ n:＝FIRST(OPEN)，REMOVE(n，OPEN)，ADD(n，CLOSED)；

④ IF n可直接判定为赢、输或平局
THEN f（n）:＝∞∨－∞∨0，GO LOOP1
ELSE EXPAND（n）→{n_i}，ADD（{n_i}，T）
IF d（n_i）＜k THEN ADD（{n_i}，OPEN），GO LOOP1
ELSE 计算f（n_i），GO LOOP1；n_i达到深度k，计算各端节点f值。

⑤ LOOP2：IF CLOSED＝NIL THEN GO LOOP3
ELSE n_p:＝FIRST（CLOSED）；

⑥ IF（n_p∈MAX）∧（f（n_{ci}∈MIN）有值）
THEN f（n_p）:＝max{f（n_{ci}）}，REMOVE(n_p，CLOSED)；若MAX所有子节点均有值，则该MAX取其极大值。
IF（n_p∈MIN）∧（f（n_{ci}∈MAX）有值）
THEN f（n_p）:＝min{f（n_{ci}）}，REMOVE(n_p，CLOSED)；若MIN所有子节点均有值，则该MIN取其极小值。

⑦ GO LOOP2；

⑧ LOOP3：IF f（s）≠NIL THEN EXIT(END∨M(Move，T))；s有值，则结束或标记走步。

极小极大搜索算法

- 该算法分两个阶段进行:
  - 第一阶段②～④：用宽度优先法生成规定深度的全部博弈树，然后对其所有端节点计算其静态估计函数值。
  - 第二阶段⑥～⑧：从底向上逐级求非端节点的倒推估计值，直到求出初始节点s的倒推值f（s）为止，此时

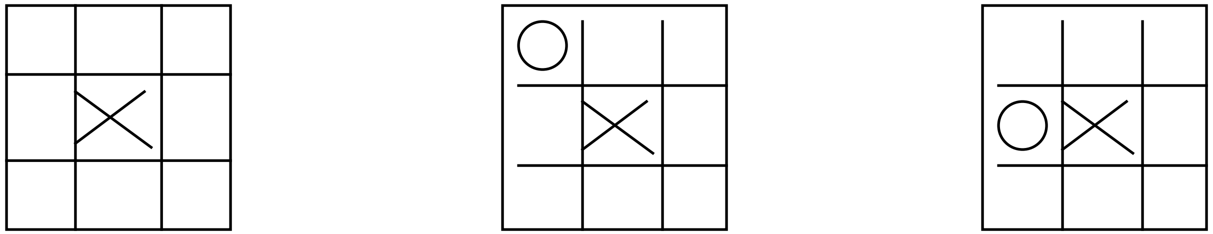
- 等对手响应走步后，再以当前的状态作为起始状态s，重复调用该过程。

极小极大搜索算法例子

- 3×3棋盘的一字棋：九宫格棋盘上，两位选手轮流在棋盘上摆各自的棋子（每次一枚），谁先取得三子一线的结果就取胜。
- 假设：
  - 程序方MAX的棋子用（×）表示；
  - 对手MIN的棋子用（○）表示；
  - MAX先走。

极小极大搜索算法例子

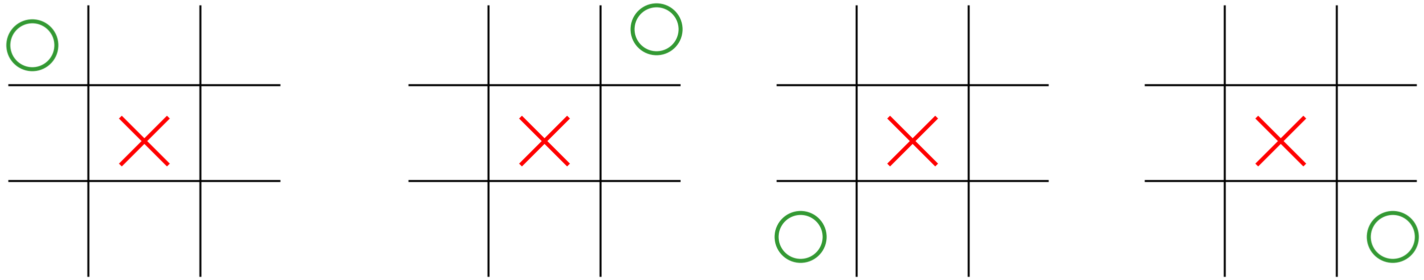
- 静态估计函数f（p）规定如下：
  - 若p是MAX获胜的格局，则f（p）＝∞；
  - 若p是MIN获胜的格局，则f（p）＝ - ∞；
  - 若p对任何一方来说都不是获胜的格局，则f（p）＝（所有空格都放上MAX的棋子之后，MAX的三子成线（行、列、对角）的总数 - （所有空格都放上MIN的棋子之后，MIN的三子成线（行、列、对角）的总数）。



$e(p)=5-4=1$　　　$e(p)=6-4=2$

极小极大搜索算法例子

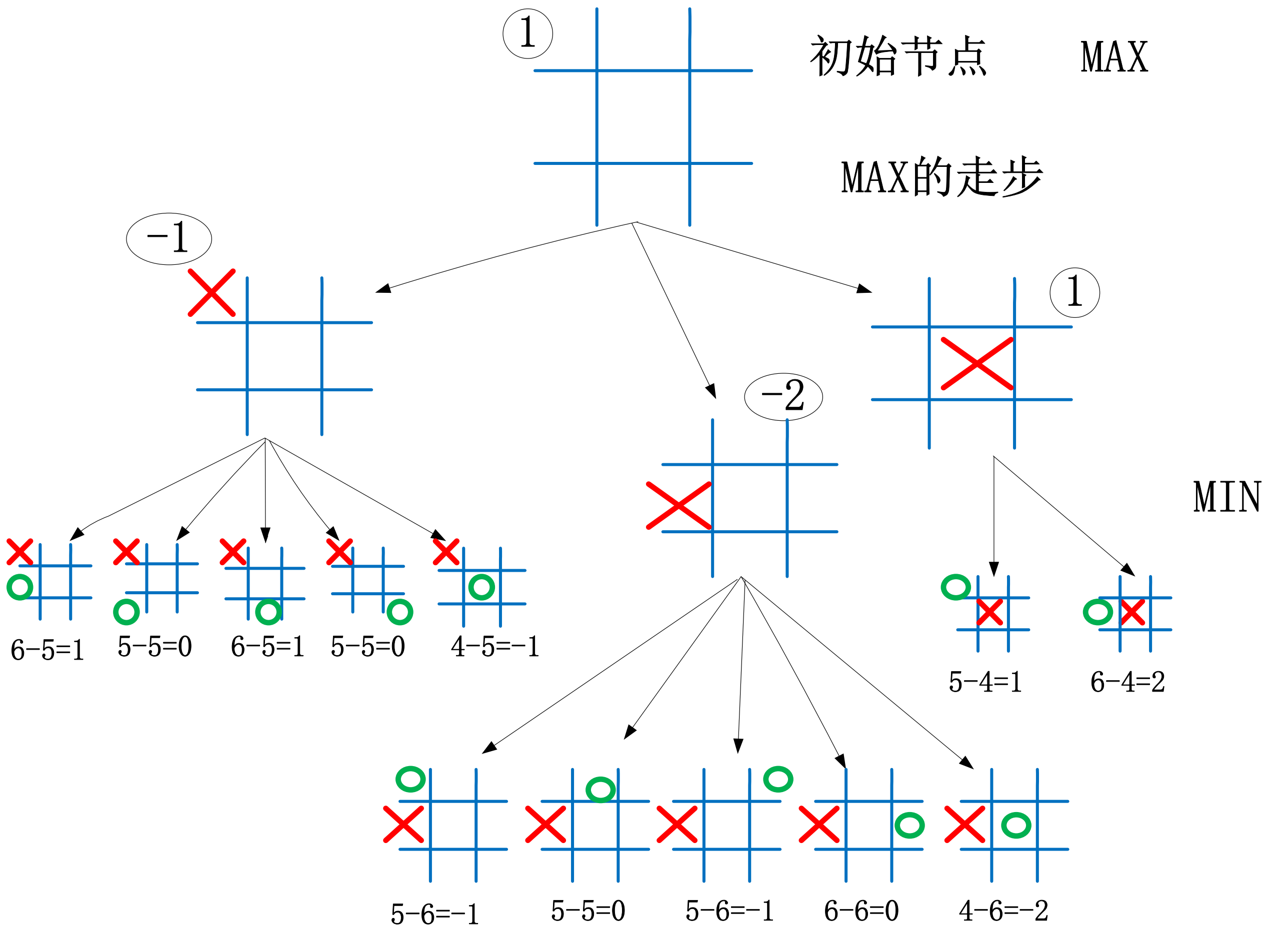
- 在搜索过程中，具有对称性的棋局认为是同一棋局，可以大大减少搜索空间。



对称棋局的例子

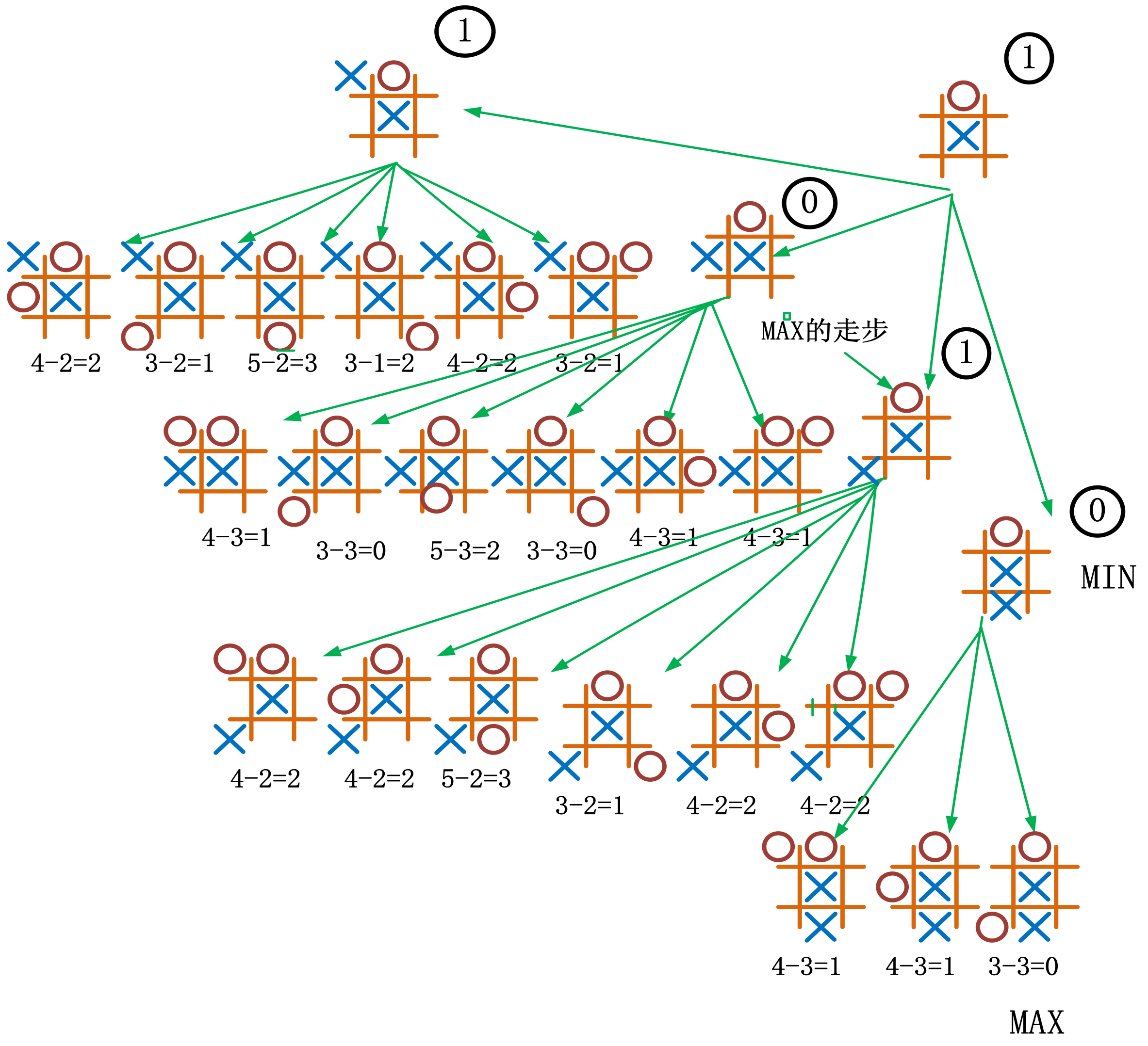
极小极大搜索算法例子

- 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：



极小极大搜索算法例子

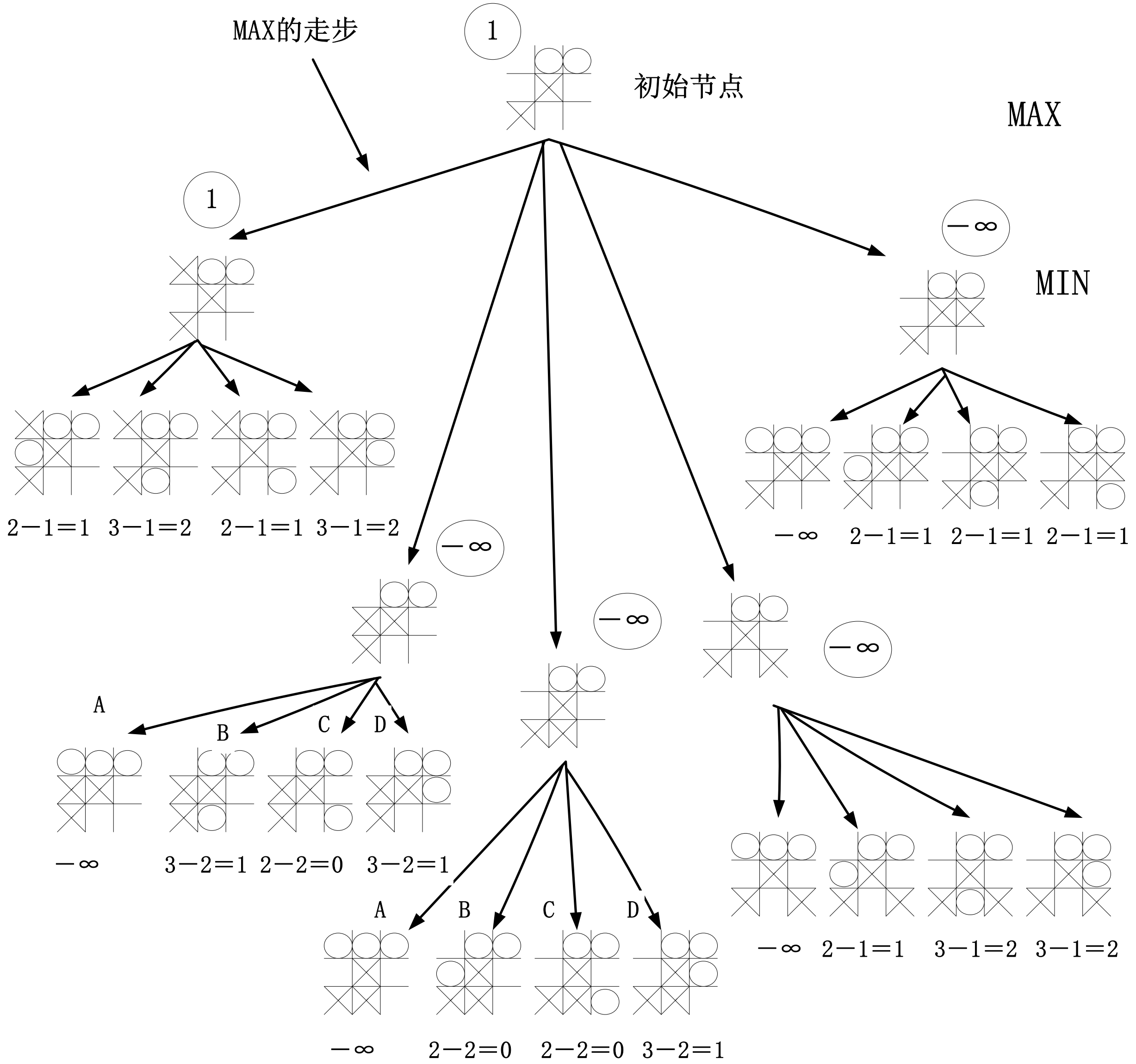
- 假设MAX走完第一步后，MAX的对手是在×之上的格子下棋子，这时MAX在新格局下调用算法，第二次产生的搜索树为：



极小极大搜索算法例子

- 类似地，第三次的搜索树为：

至此MAX走完最好的走步后，不论MIN怎么走，都无法挽回败局，因此只好认输了。



极小极大搜索算法性能

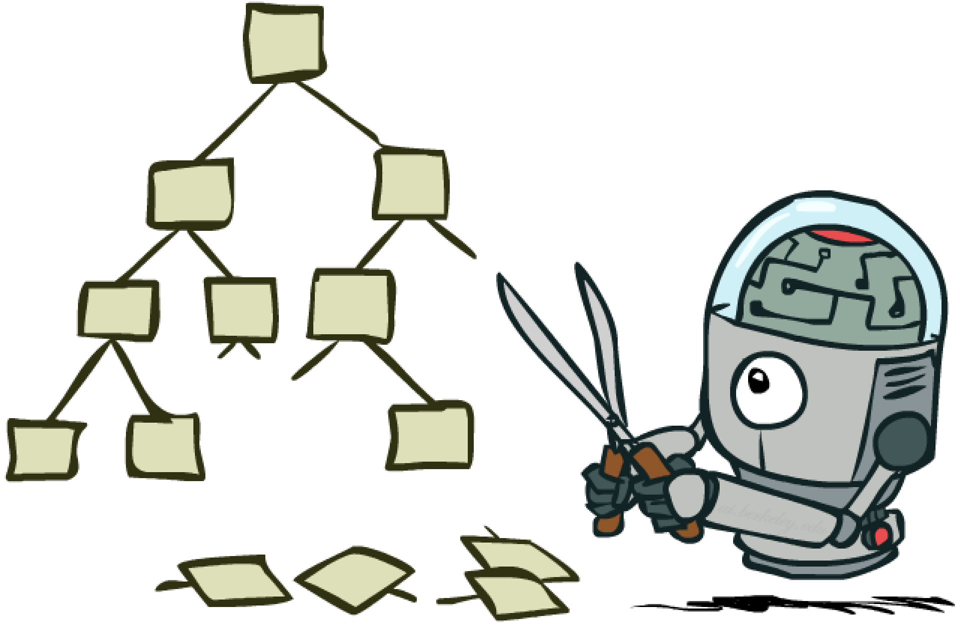
- 完备性? Yes (if tree is finite)
- 最优性? Yes (against an optimal opponent)
- 时间复杂度? $O(b^m)$ (b-legal moves; m- max tree depth)
- 空间复杂度? O(bm) (depth-first exploration)

  开销过大!

目录

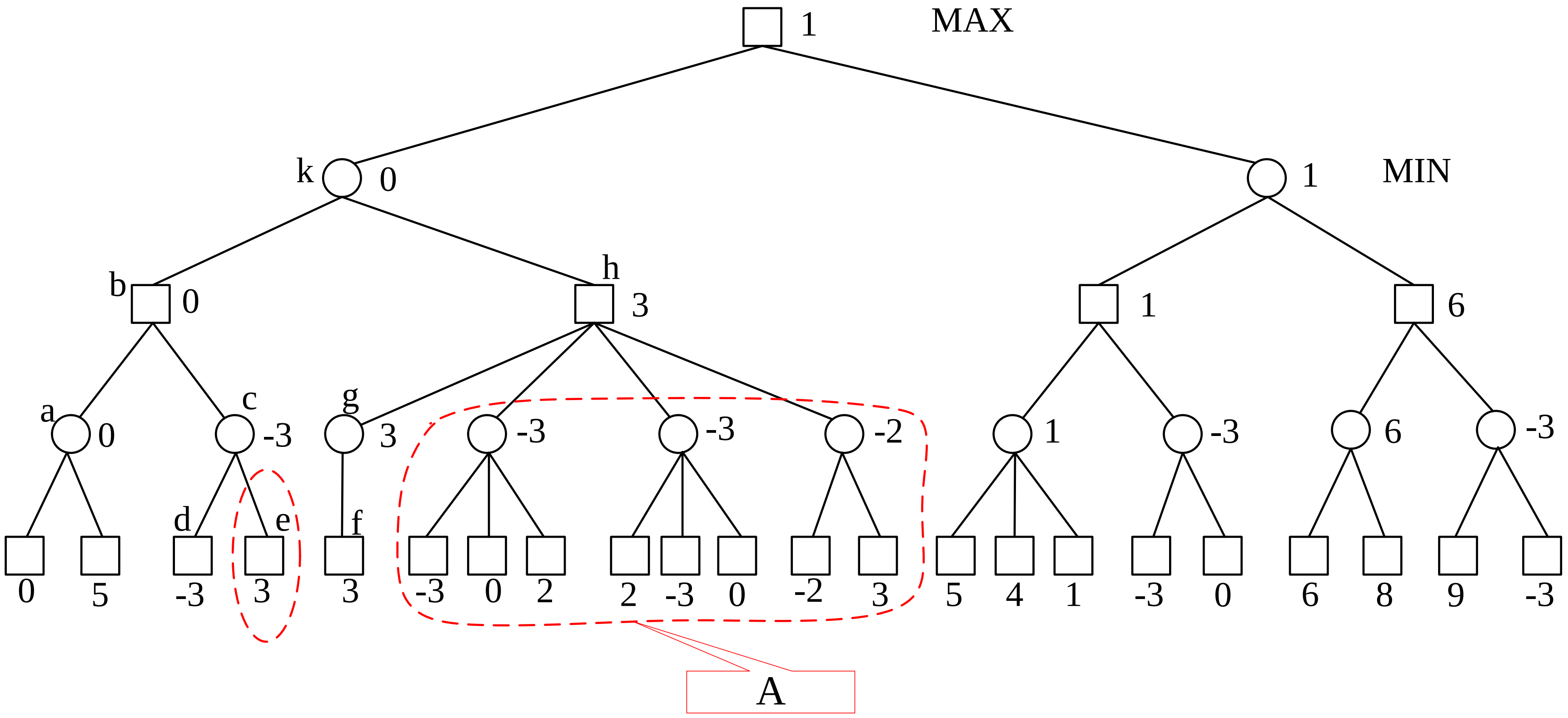
- 在极小极大搜索方法中，由于要生成指定深度以内的所有节点，其节点数将随着搜索深度的增加成指数增长。
  - 这极大地限制了极小极大搜索方法的使用。
- 能否在搜索深度不变的情况下，利用已有的搜索信息减少生成的节点数呢?

博弈树剪枝



α-β剪枝的基本思想

- 设某博弈问题如下图所示：

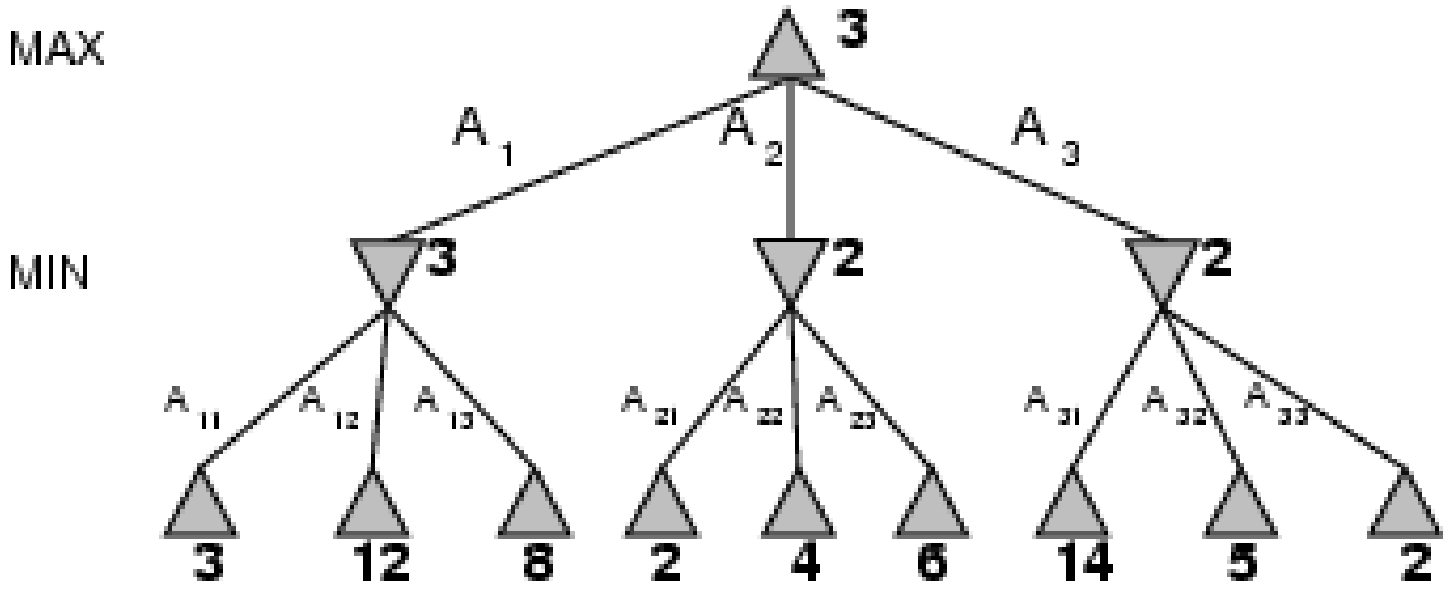


α-β剪枝的基本思想

- α-β搜索过程的基本思想：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。

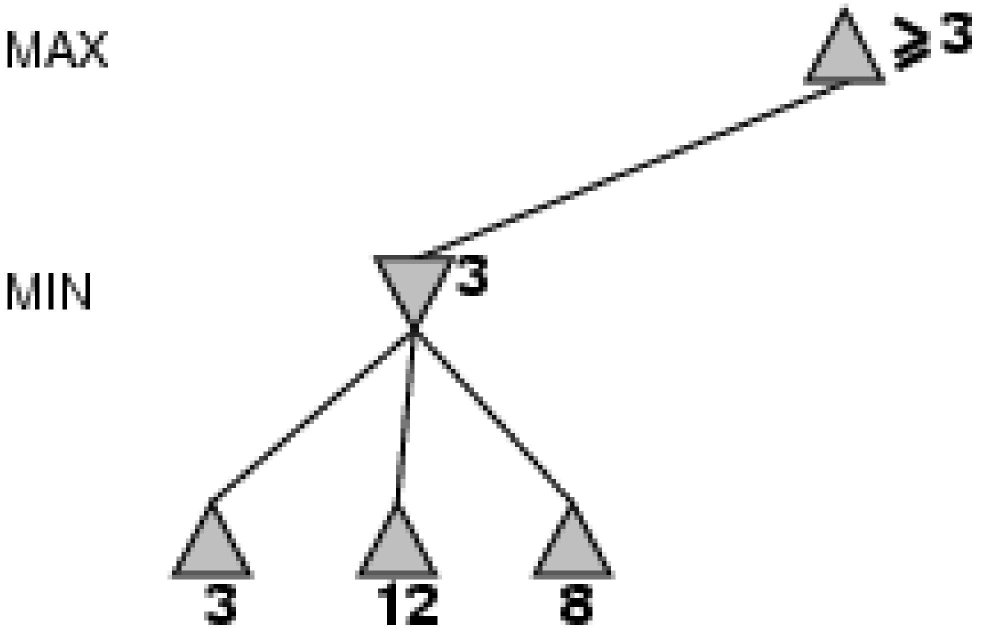
- 为了使生成和估值过程紧密结合，采用有界深度优先策略进行搜索。
- 当生成达到规定深度的节点时，就立即计算其静态估值函数，而一旦某个非端节点有条件确定其倒推值时就立即计算赋值。

α-β剪枝

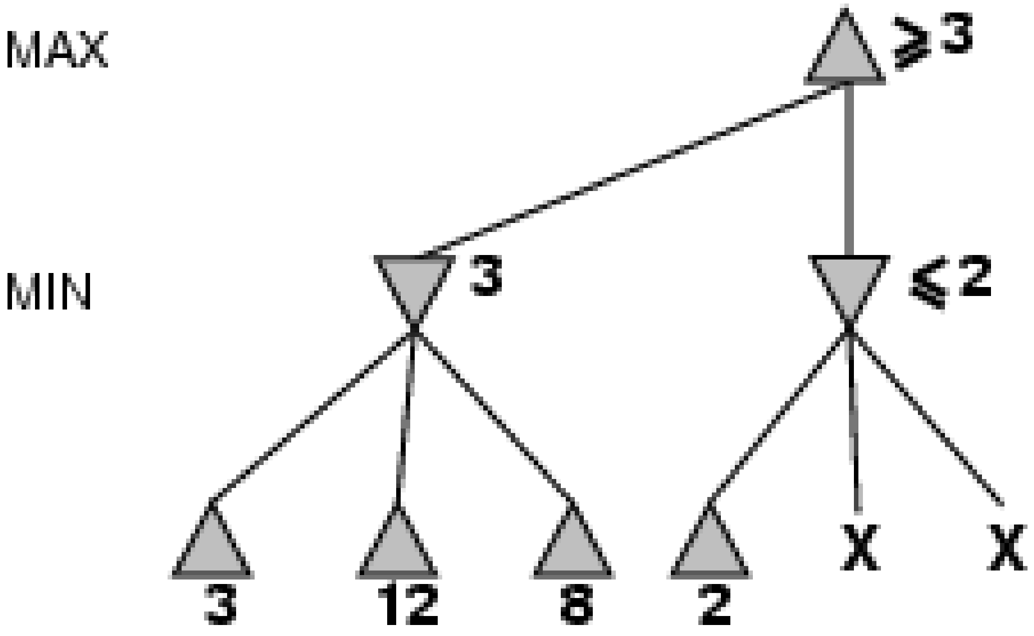


MAX
MIN
3
A 1
A 2
A 3
3
2
2
A 11
A 12
A 13
A 21
A 22
A 23
A 31
A 32
A 33
3
12
8
2
4
6
14
5
2

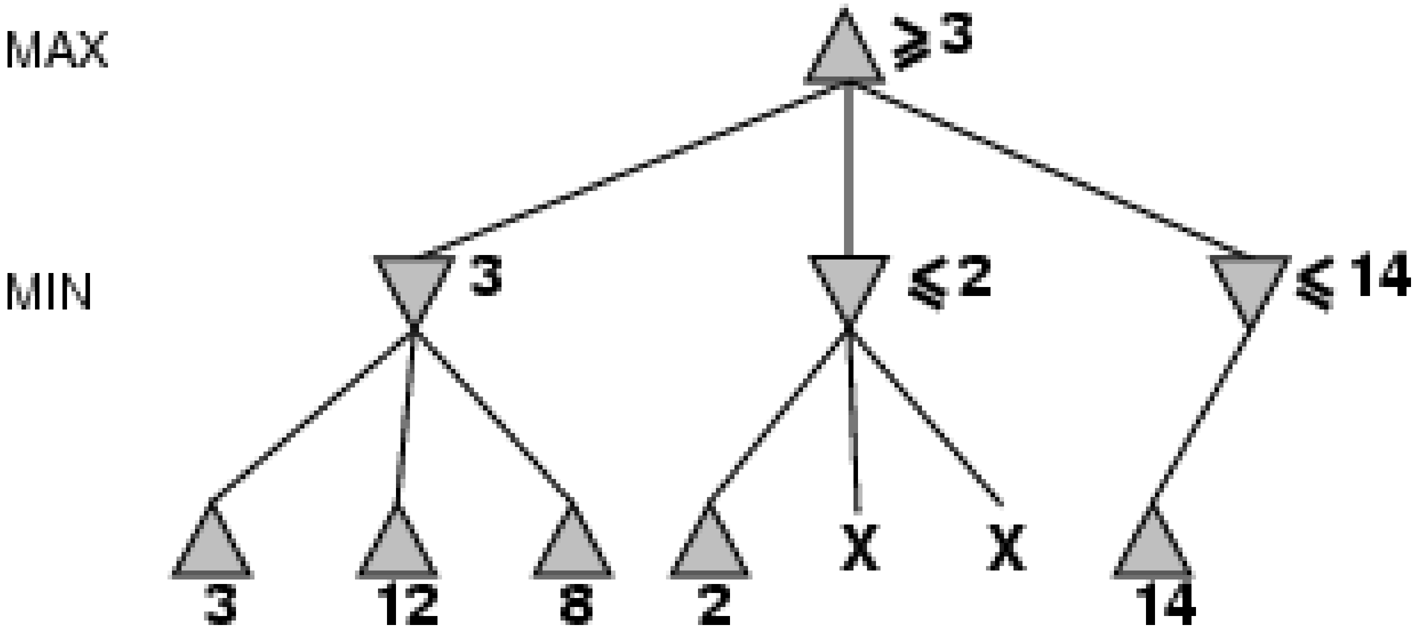
α-β剪枝



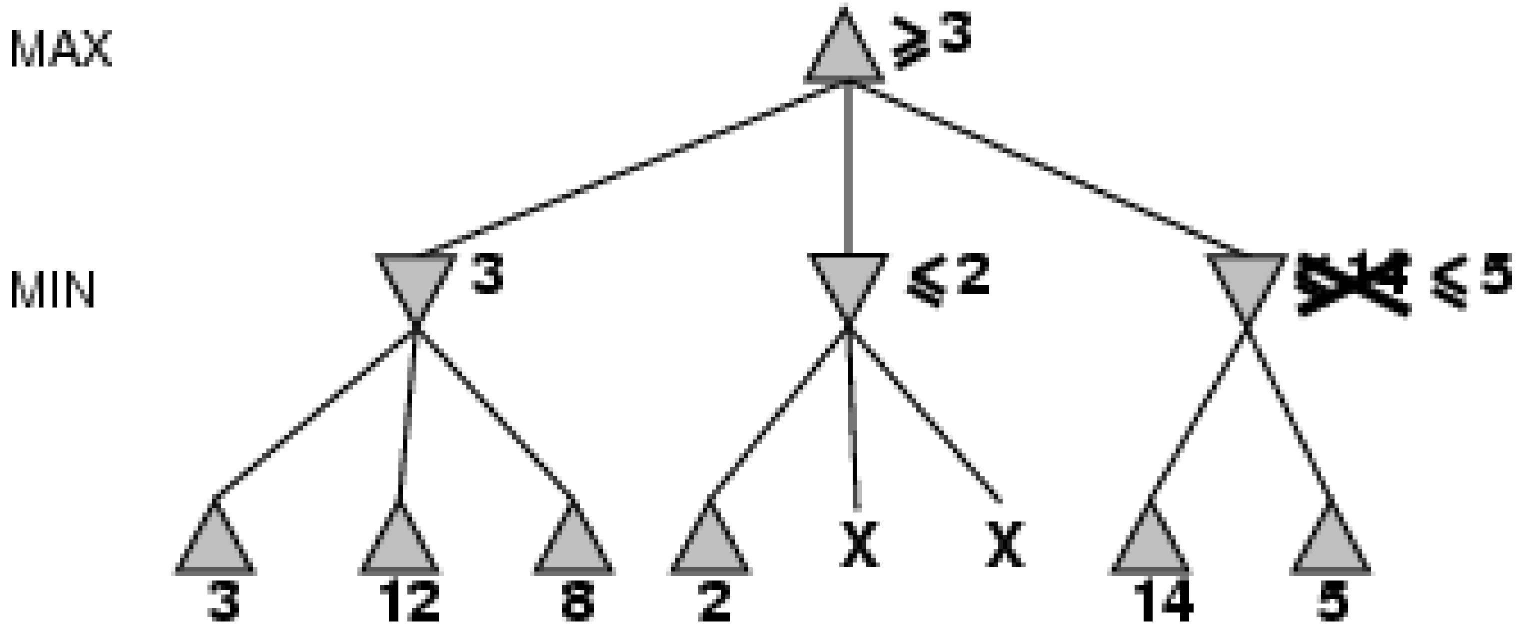
α-β剪枝



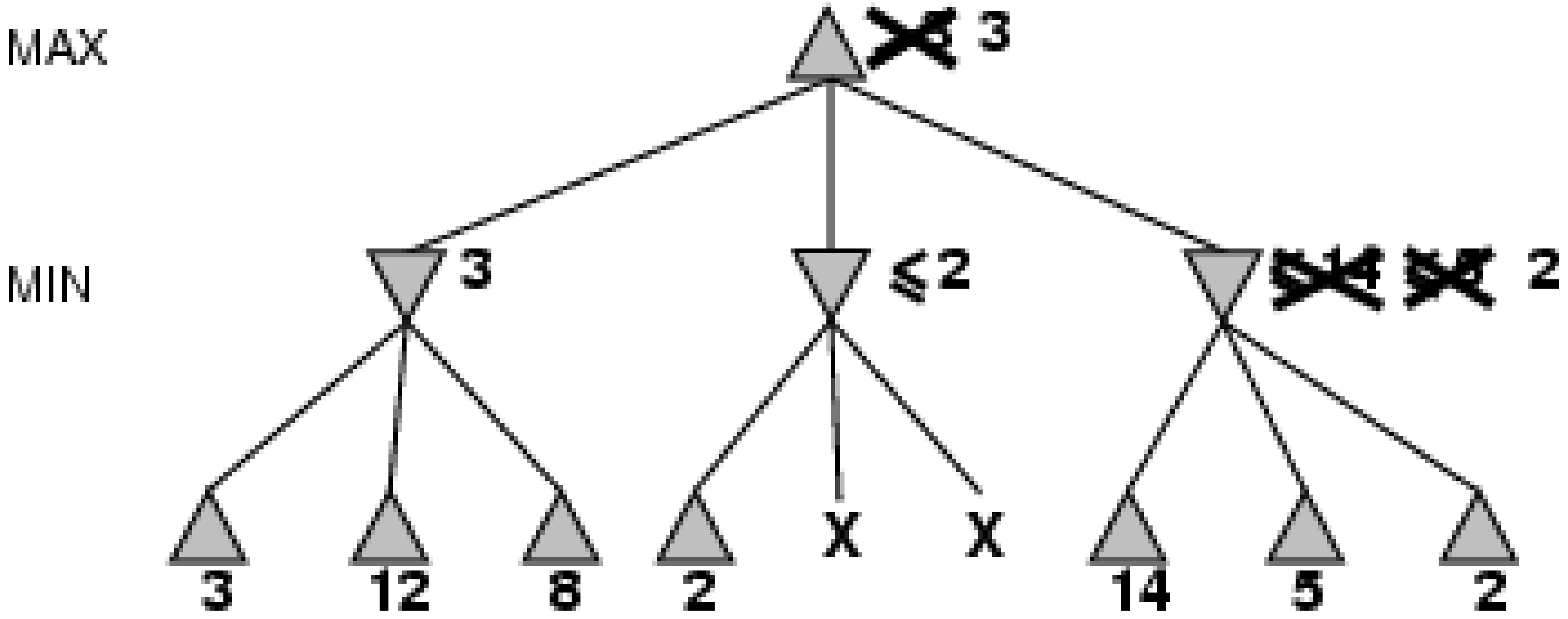
α-β剪枝



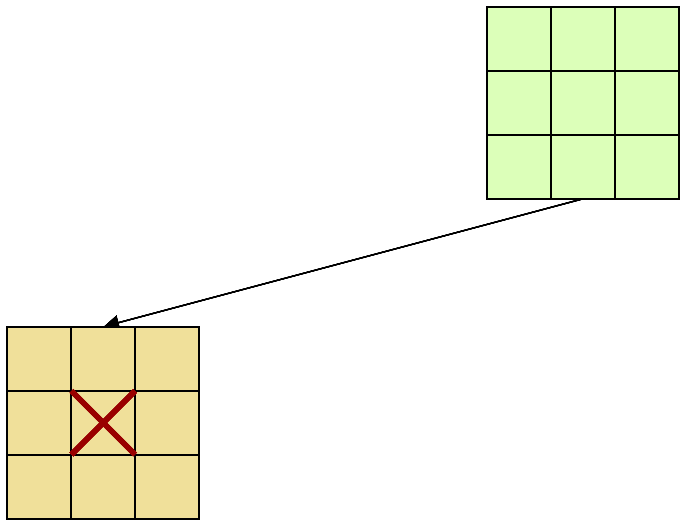
α-β剪枝

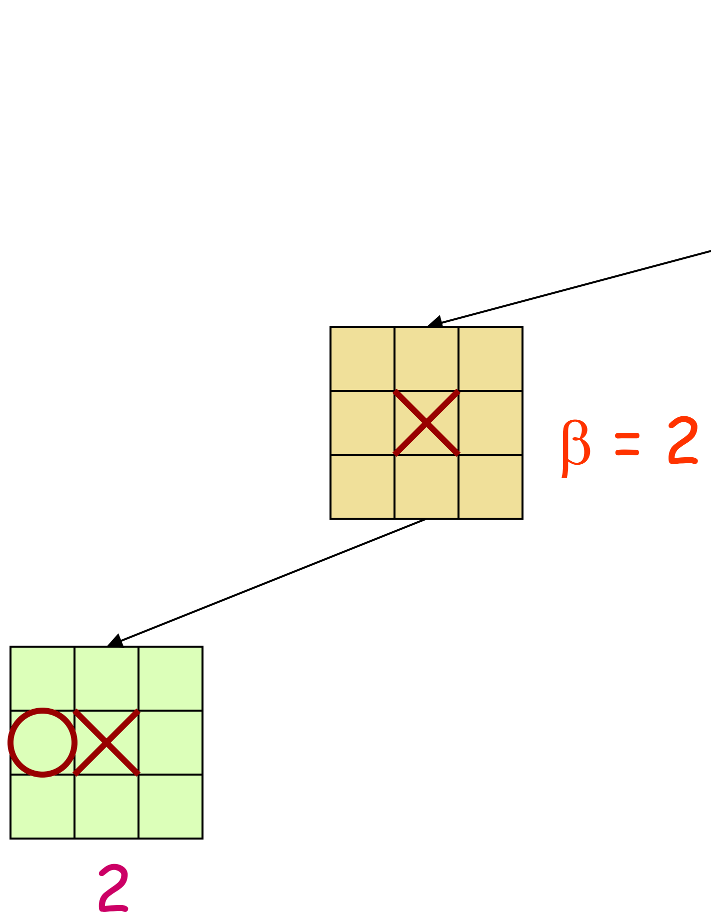


α-β剪枝



α-β剪枝

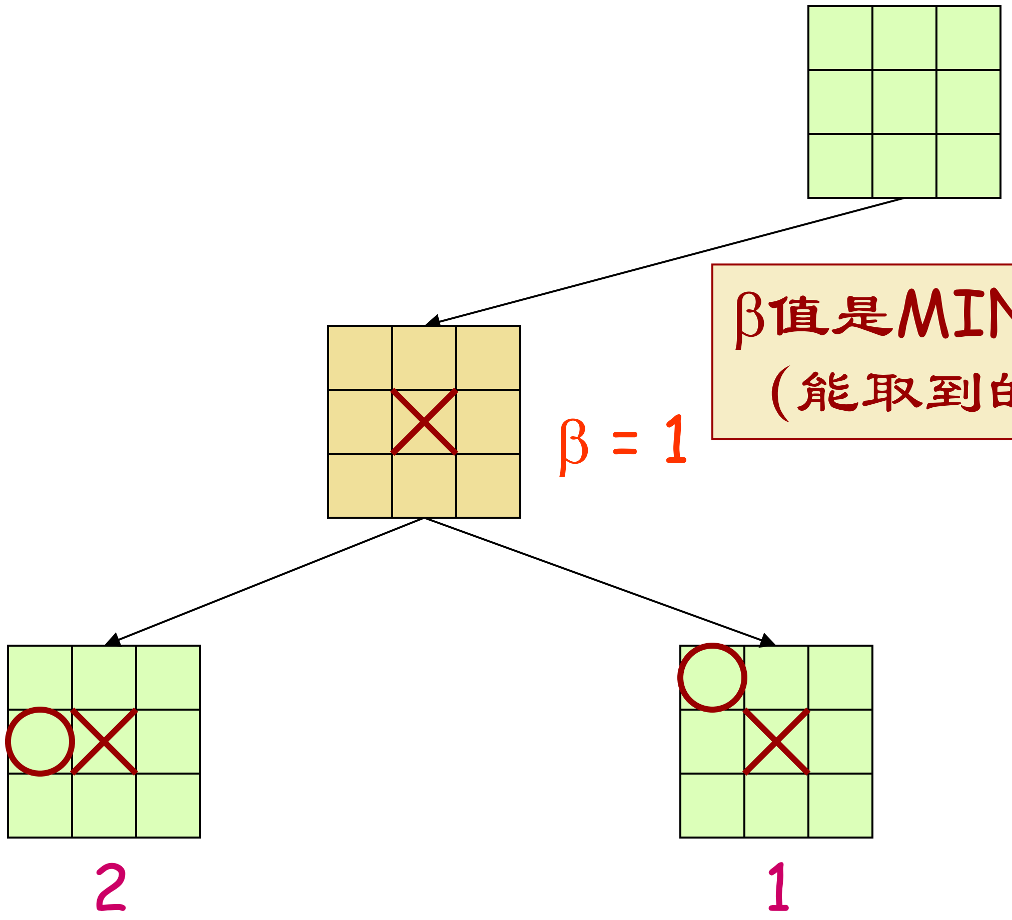




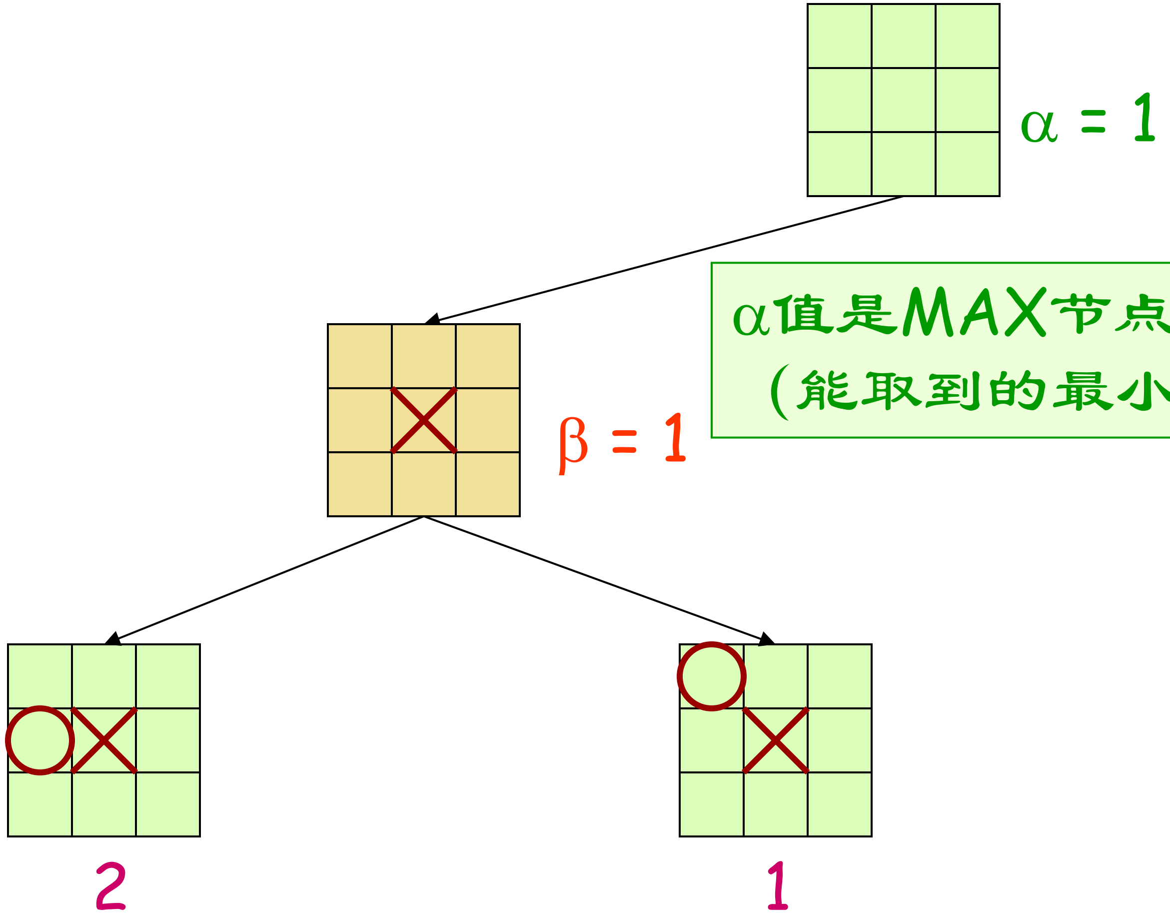
β值是MIN节点最后回推值的上限

（能取到的最大值，最糟糕情况）

α-β剪枝

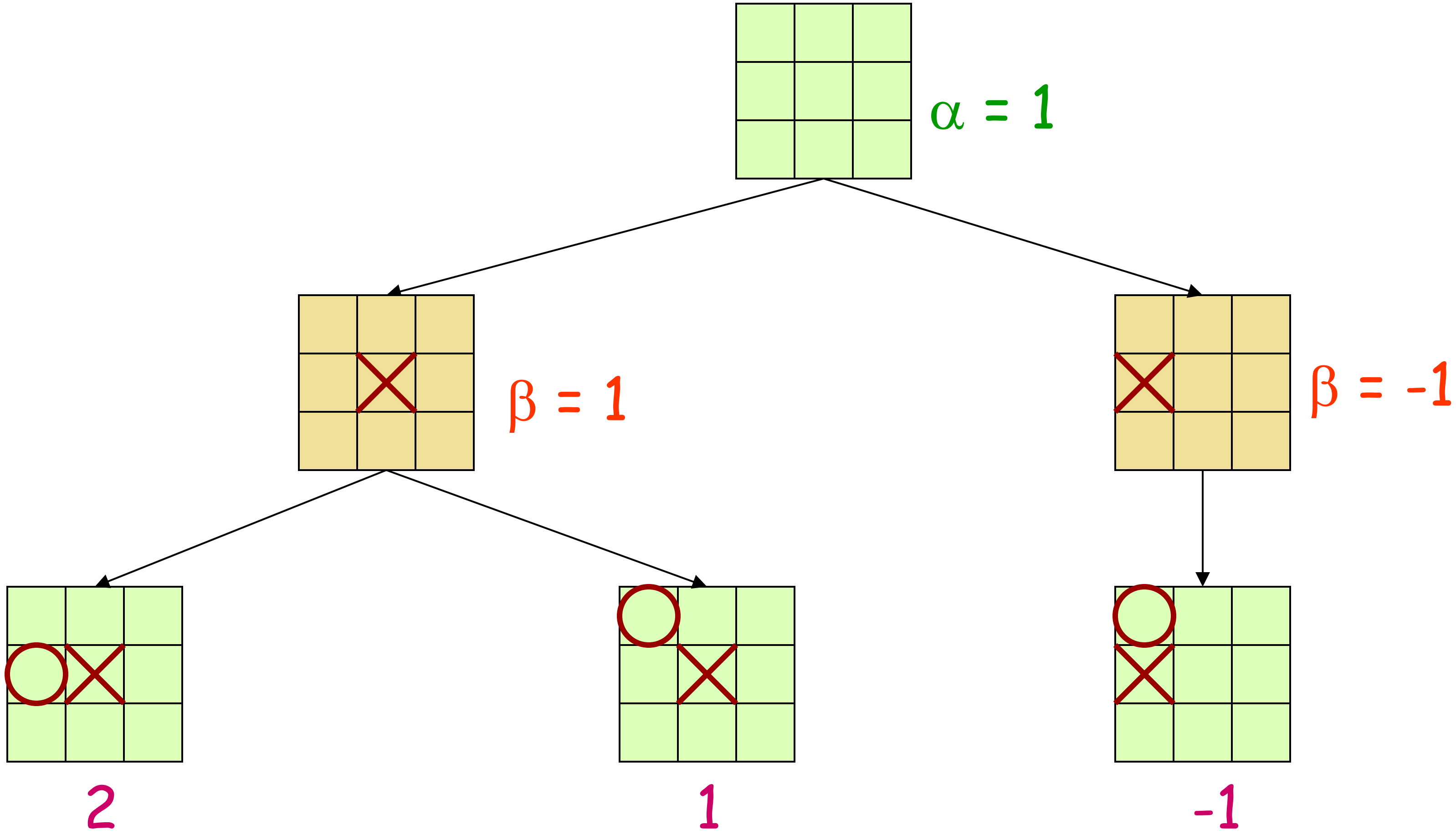


β值是MIN节点最后回推值的上限（能取到的最大值，最糟糕情况）

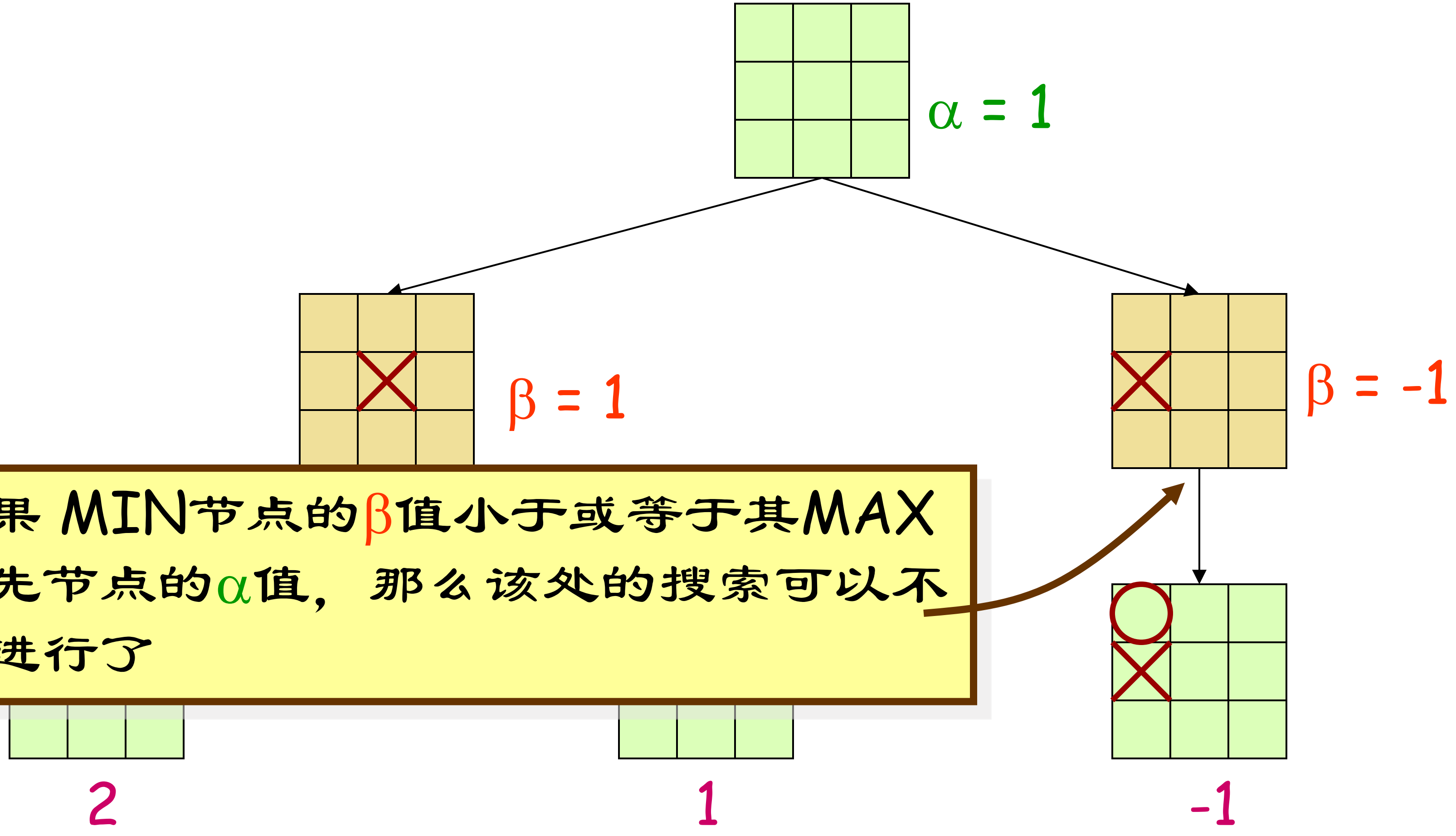


α值是MAX节点最后回推值的下限（能取到的最小值，最糟糕情况）

α-β剪枝



α-β剪枝



如果 MIN节点的β值小于或等于其MAX祖先节点的α值，那么该处的搜索可以不用进行了

α-β剪枝

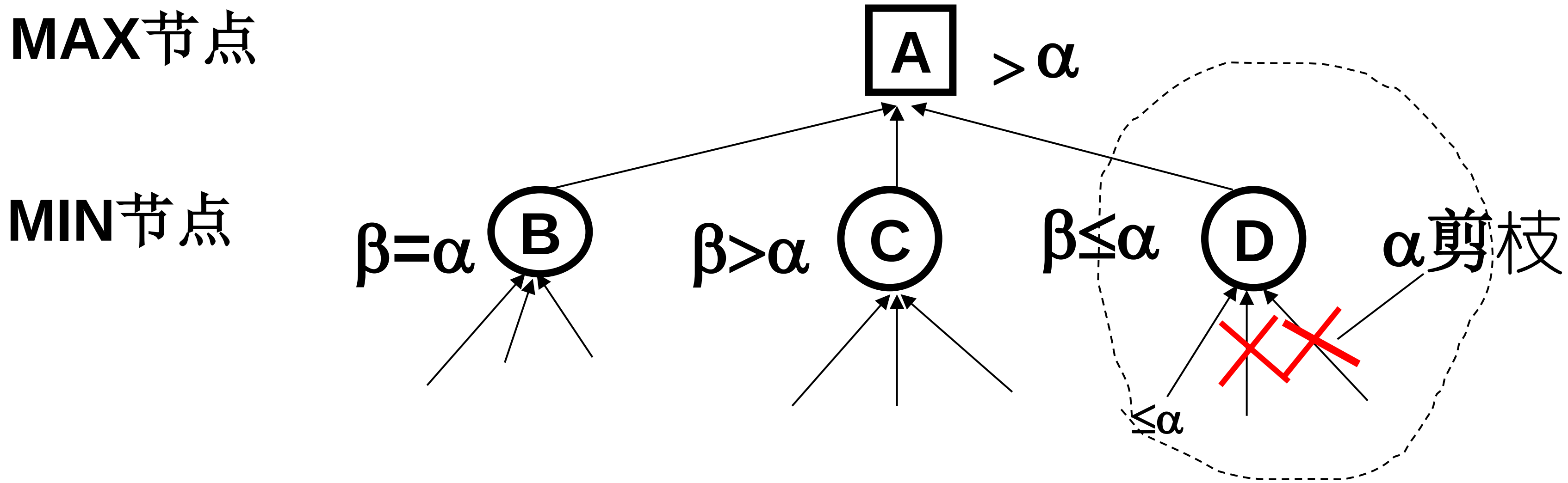
- 采用深度优先方法探索博弈树
- 只要有可能就回推 α和 β 的值
- 把那些不会改变最终决策的分支剪去

α-β剪枝

- 当节点N以下的搜索完成（或剪枝中止）时，更新N的父节点的α或β值
- 当一个MAX节点N的α值≥ N 的MIN祖先节点的β值，则节点N 以下的搜索中止
- 当一个MIN节点N的β值≤ N 的MAX祖先节点的α 值，则节点N 以下的搜索中止

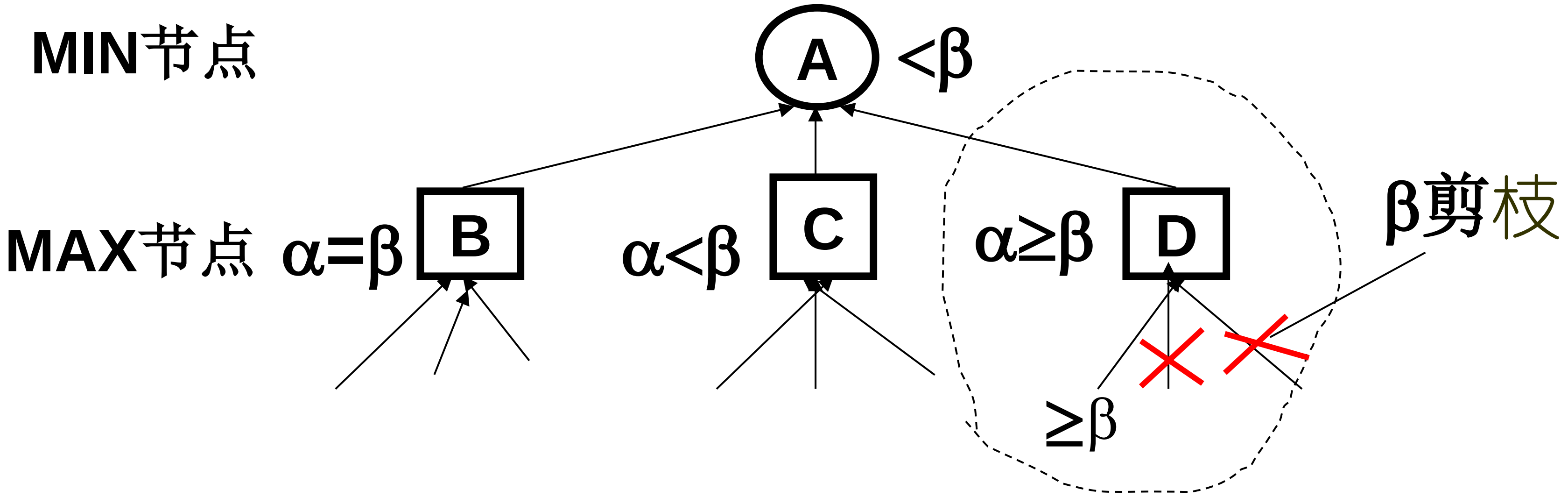
- α剪枝法

设MAX节点的下限为α，则其所有的MIN子节点中，其评估值的β上限小于等于α的节点，其以下部分的搜索都可以停止了，即对这部分节点进行了α剪枝。

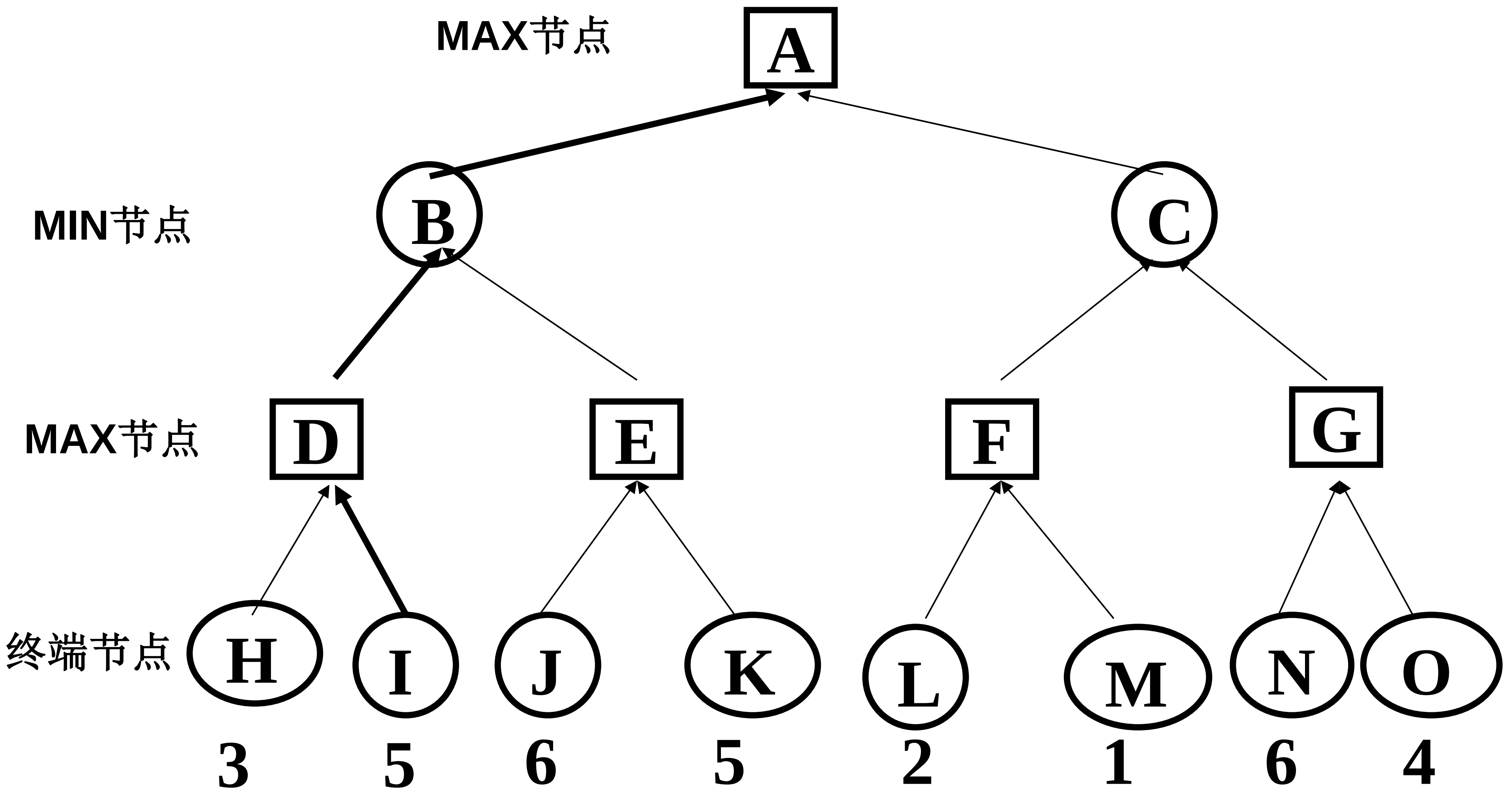


■β剪枝法

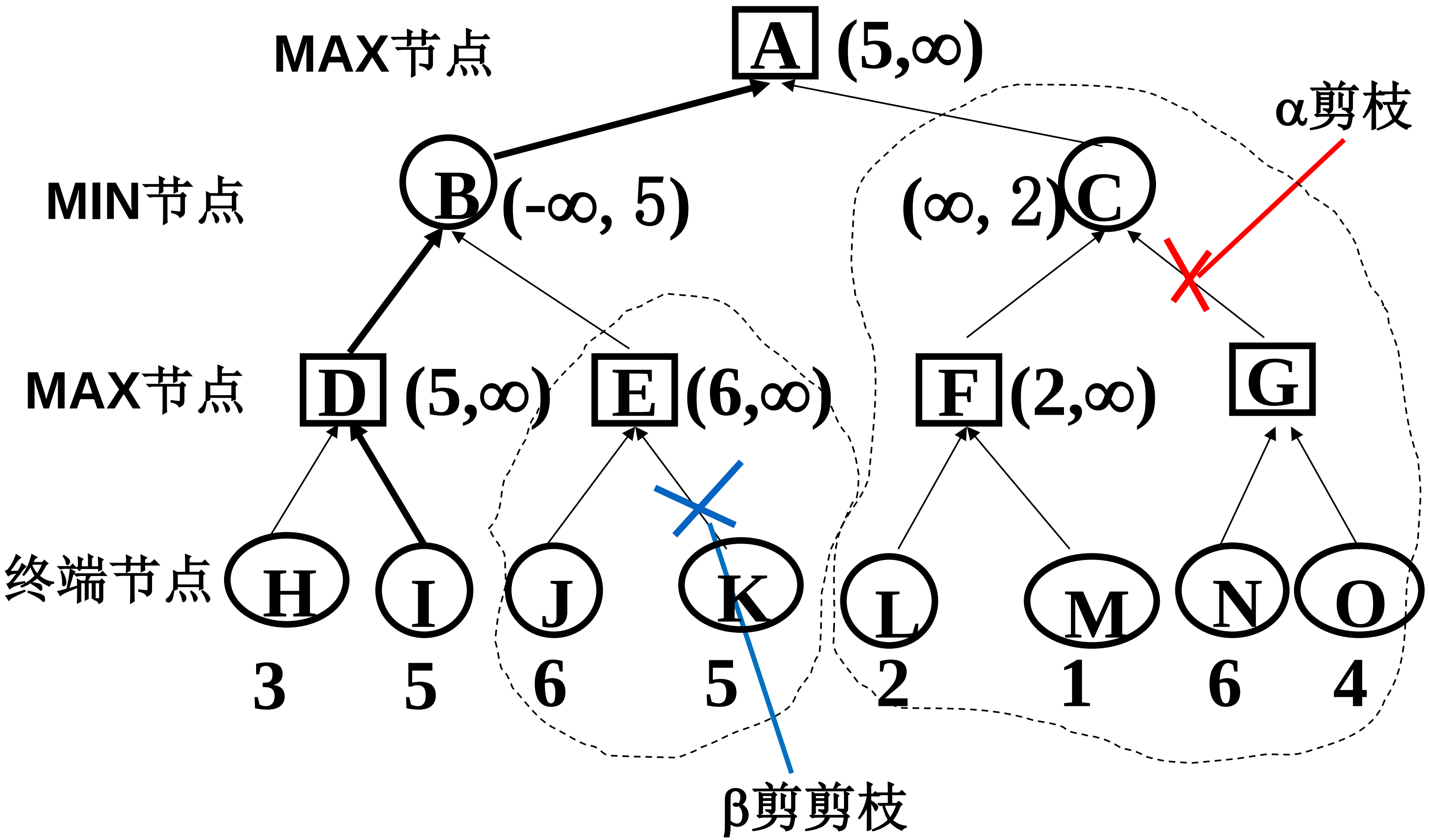
设MIN节点的上限为β，则其所有的MAX子节点中，其评估值的α下限大于等于β的节点，其以下部分的搜索都可以停止了，即对这部分节点进行了β剪枝。



α-β剪枝过程



α-β剪枝过程


MAX节点
A (5,∞)
α剪枝
MIN节点
B (-∞, 5)
(∞, 2) C
MAX节点
D (5,∞)
E (6,∞)
F (2,∞)
G
终端节点
H
I
J
K
L
M
N
O
3
5
6
5
2
1
6
4
β剪剪枝

α-β剪枝性能

- 剪枝不影响最终结果
- 好的行棋排序，可以提高剪枝效率
- 拥有好的后继顺序，时间复杂度= O(b^m/2)

α-β剪枝注意点

- 在进行α-β剪枝时，应注意以下几个问题：

1. 比较都是在极小节点和极大节点间进行的；极大节点和极大节点的比较，或者极小节点和极小节点间的比较是无意义的。

2. 在比较时注意是与“先辈层”节点比较，不只是与父辈节点比较。当然，这里的“先辈层”节点，指的是那些已经有了值的节点。

3. 只有一个节点的值“固定”以后，其值才能够向其父节点传递。

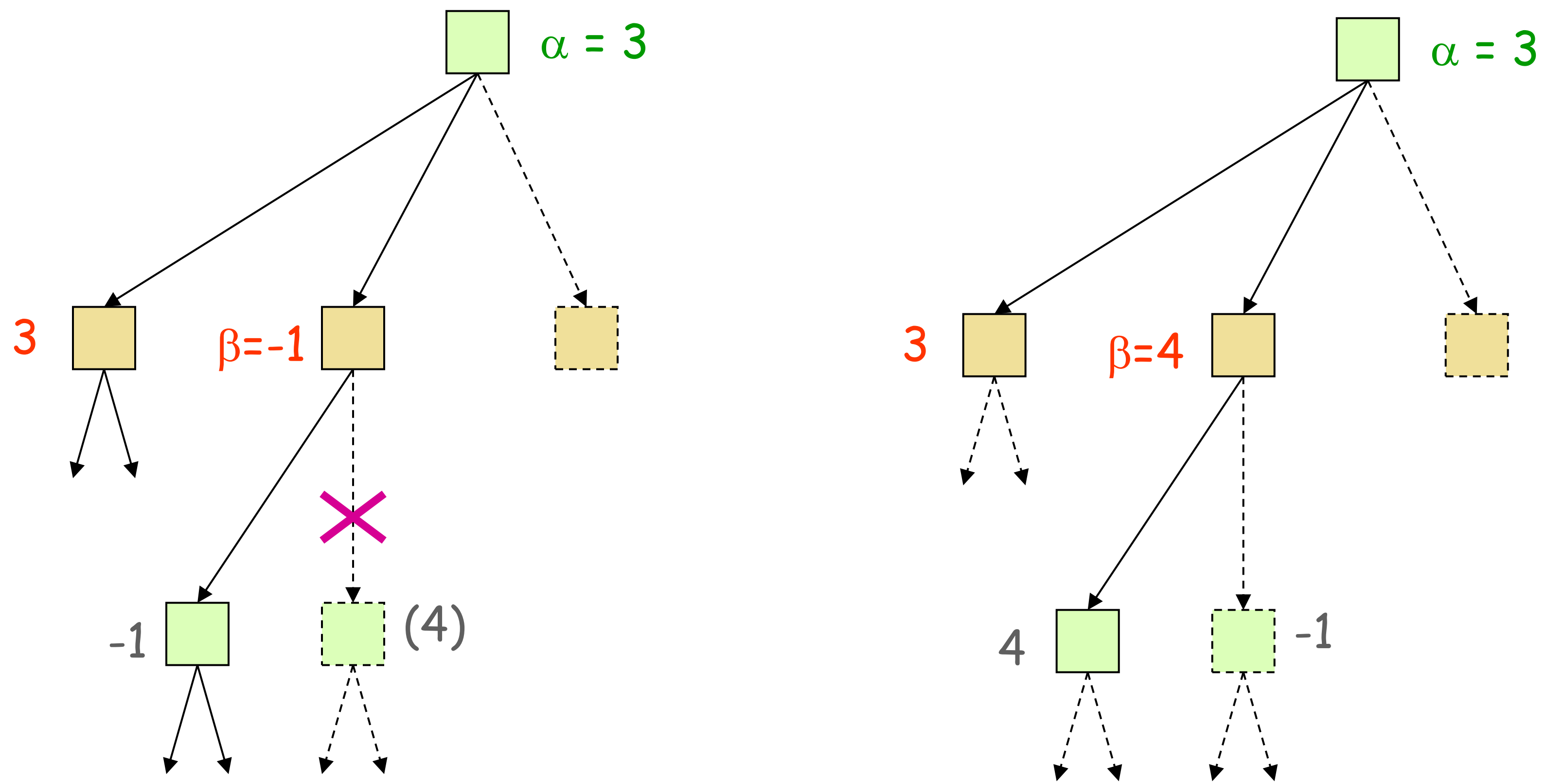
α-β剪枝注意点

- 在进行α-β剪枝时，应注意以下几个问题：

4. α-β剪枝方法搜索得到的最佳走步与极小极大方法得到的结果是一致的，α-β剪枝并没有因为提高效率，而降低得到最佳走步的可能性。

5. 在实际搜索时，并不是先生成指定深度的搜索图，再在搜索图上进行剪枝。
   - 如果这样，就失去了α-β剪枝方法的意义。
   - 在实际程序实现时，首先规定一个搜索深度，然后按照类似于深度优先搜索的方式，生成节点。在节点的生成过程中，如果在某一个节点处发生了剪枝，则该节点其余未生成的节点就不再生成了。

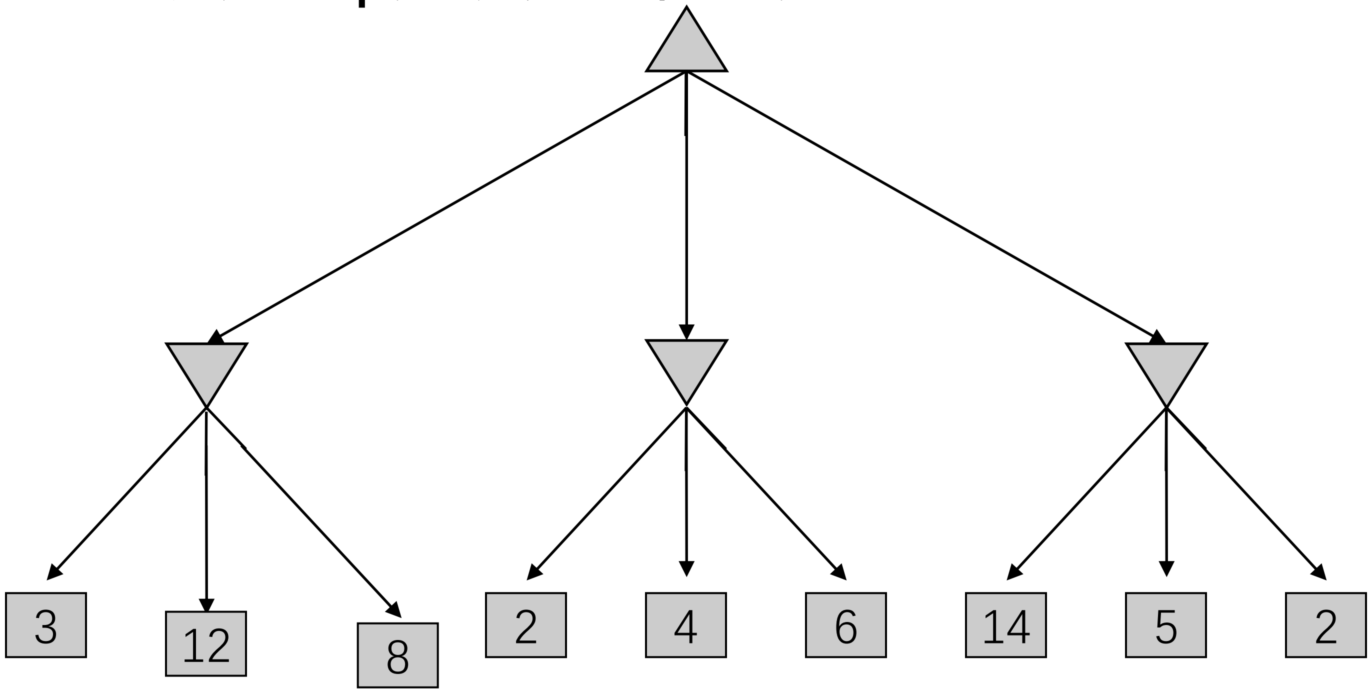
考虑以下两种情况



α-β剪枝例子

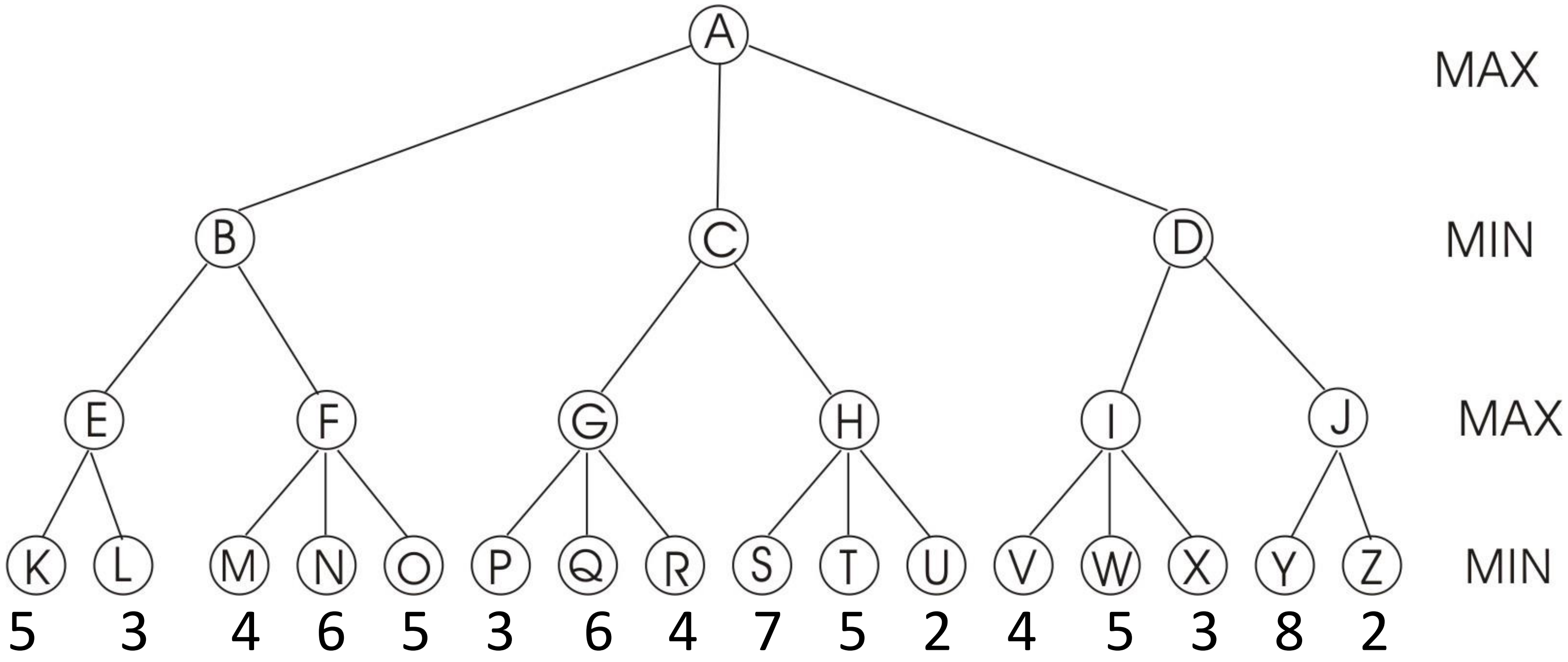
(a) 通过最大最小化算法计算结点倒推值

(b) 圈出使用α-β剪枝算法将被剪掉的结点

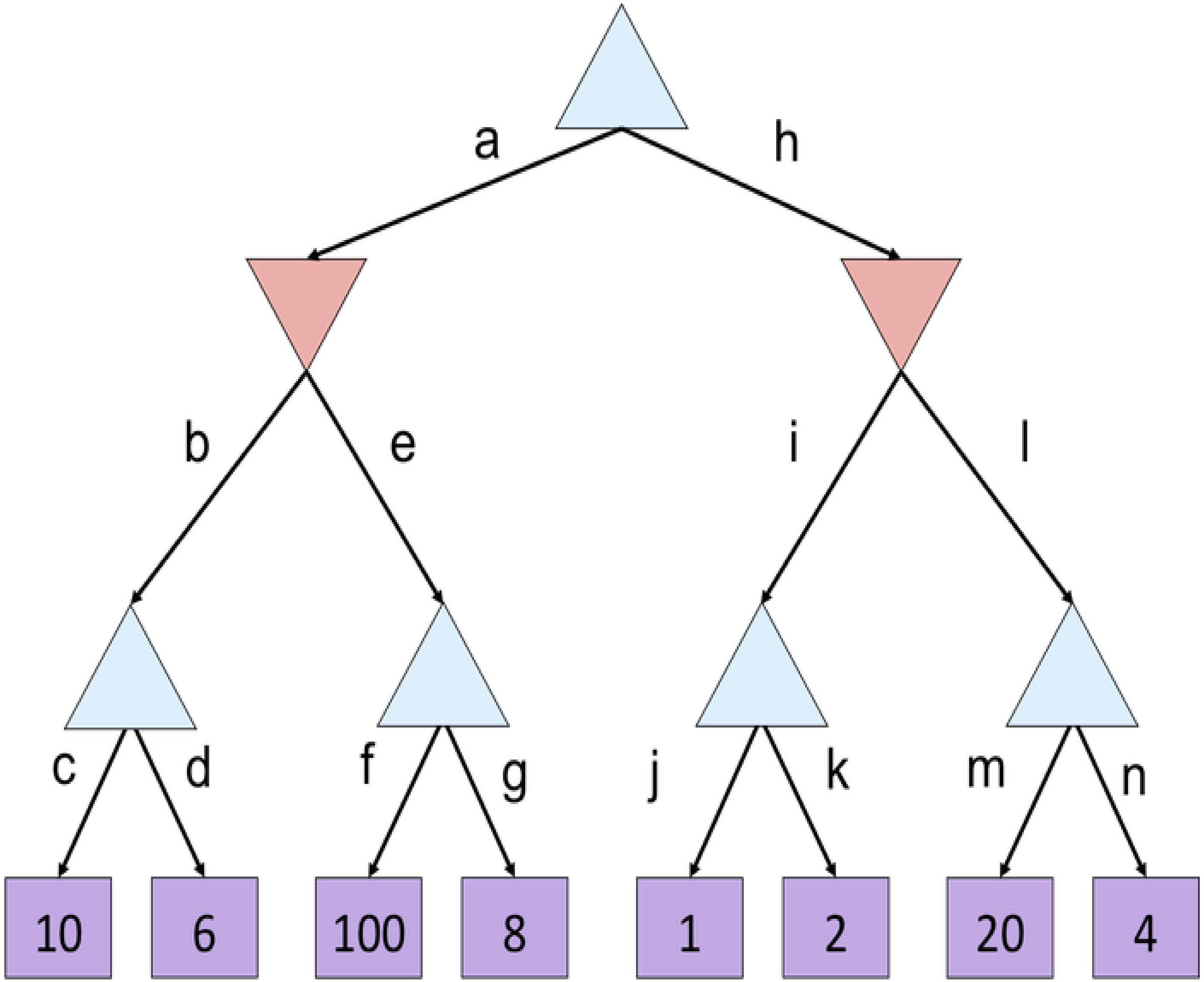


α-β剪枝例子

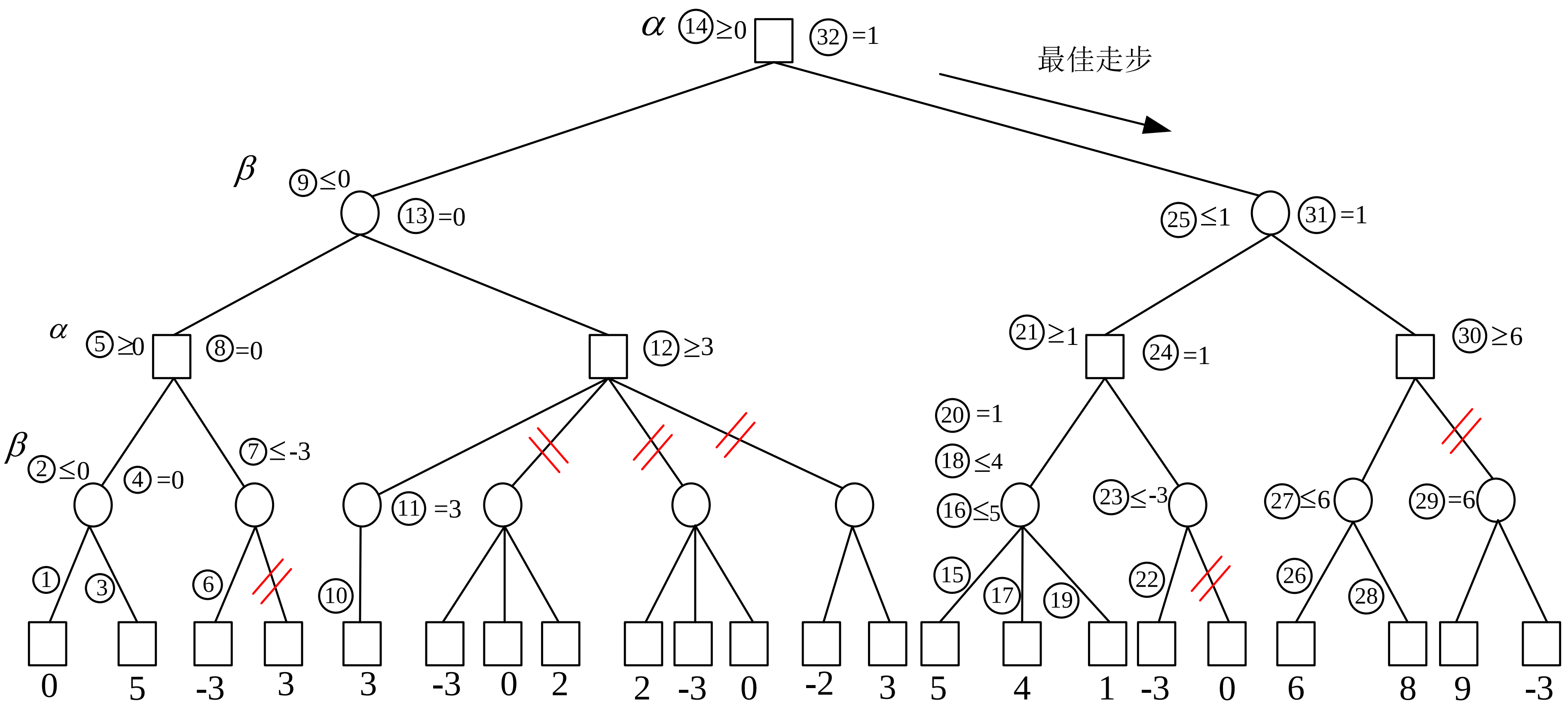
- (a) 通过最大最小化算法计算结点倒推值
- (b) 圈出使用α-β剪枝算法将被剪掉的结点



α-β剪枝例子

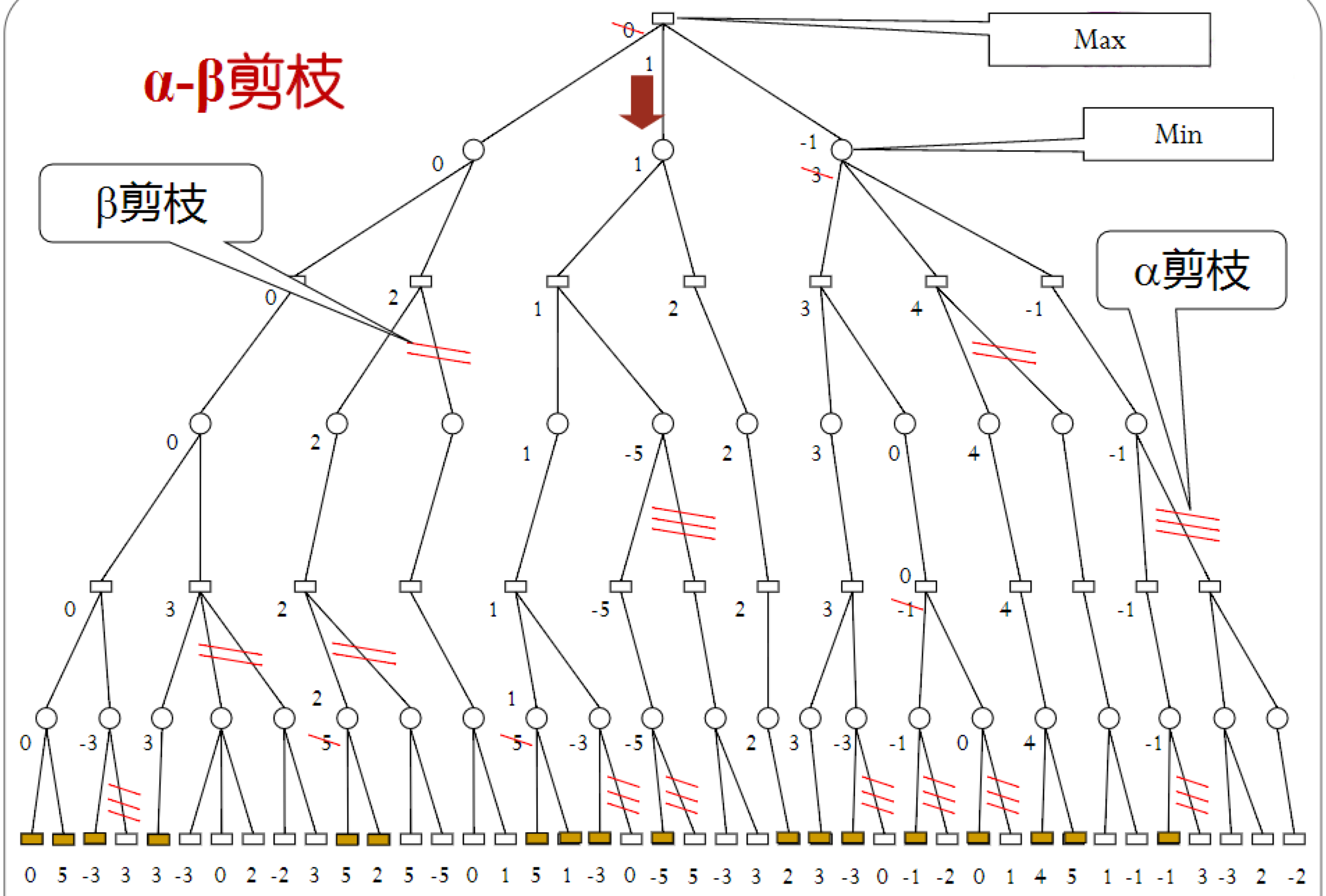


α-β剪枝例子



α 14≥0
32 =1
最佳走步
β 9≤0
13 =0
25≤1
31 =1
α 5≥0
8=0
12≥3
21≥1
24 =1
30≥6
20 =1
18≤4
β 2≤0
4 =0
7≤-3
11 =3
16≤5
23≤-3
27≤6
29=6
1
3
6
10
15
17
19
22
26
28
0 5 -3 3 3 -3 0 2 2 -3 0 -2 3 5 4 1 -3 0 6 8 9 -3

α-β剪枝


Max
Min
β剪枝
α剪枝
0 5 -3 3 3 -3 0 2 -2 3 5 2 5 -5 0 1 5 1 -3 0 -5 5 -3 3 2 3 -3 0 -1 -2 0 1 4 5 1 -1 -1 3 -3 2 -2

资源限制

资源限制

- 问题: 在真实游戏中，难以搜索到叶子结点。
- 解决: 深度受限搜索
- 将搜索限制在给定深度内
- 使用启发评估函数代替效用函数
- 无法保证最优决策
- 针对任意时间，使用迭代加深算法

目录

不完美的实时决策

同济大学 TONGJI UNIVERSITY

- MINIMAX或α–β剪枝算法
  - 理想情况：算法一直搜索，直到至少一部分空间到达终止状态，从而对端结点做出准确评价。
  - 这样的搜索不现实。

- 实用方法：
  - 用可以估计棋局效用的启发式评价函数评价非终止结点。
  - 用可以决策什么时候运用评价函数的截断测试取代终止测试。

- 评价函数的设计：

① 应该以和真正的效用函数同样的方式对终止状态进行排序。

- 效用函数（又称目标函数或者收益函数）：对终止状态给出一个数值。例如，在国际象棋中，结果是赢、输或平局，分别赋予＋1，－1或0。

② 评价函数的计算不能花费太多的时间！

③ 对于非终止状态，评价函数应该和取胜的实际机会密切相关。

评价函数

- 在计算能力有限情况下，评价函数能做到最好的就是猜测最后的结果。
  - 例如，国际象棋并不是几率游戏，而且也确切知道当前状态；但计算能力有限，从而导致结果必然是不确定的。

- 大多数评价函数的工作方式是计算状态的不同特征。
  - 例如，国际象棋中兵的数目、象的数目、马的数目等等。
  - 这些特征一起定义了状态的各种类别或者等价类。
  - 但因类别太多而几乎不可能去估计取胜概率。

评价函数

- 大多数评价函数计算每个特征单独的数值贡献，然后把它们结合起来找到一个总值。

  - 加权线性评价函数

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

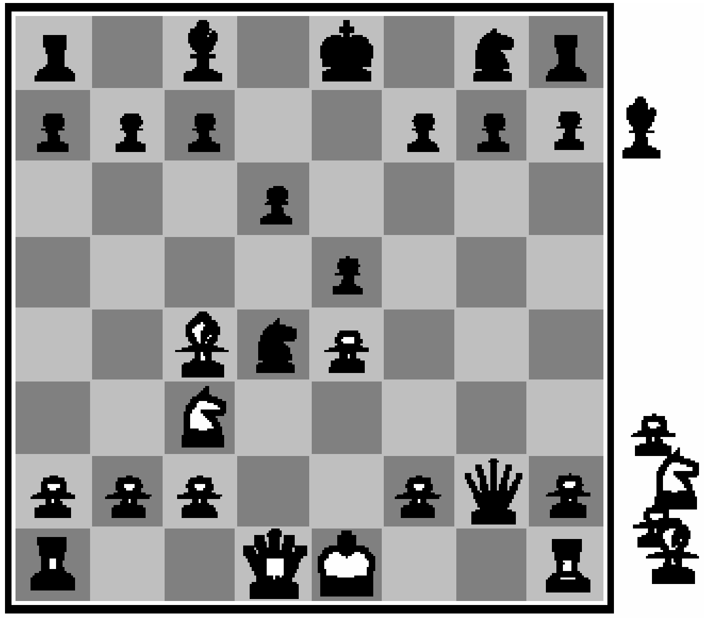
每个w_i是一个权值，f_i是棋局的某个特征。

- 例如，国际象棋的入门书中给出各个棋子的估计子力价值。
  - 兵值1分；
  - 马、象值3分；
  - 车值5分；
  - 后值9分。
  - 其它特征。例如，“好的兵阵”和“王的安全性”可能值半个兵。
- 把这项特征值简单相加就得到了一个对棋局的估计。
- 经验表明，如果其它都一样，则
  - 在领先超过1分的可靠子力优势下很可能取得胜利；
  - 3分的优势几乎足以肯定取胜。

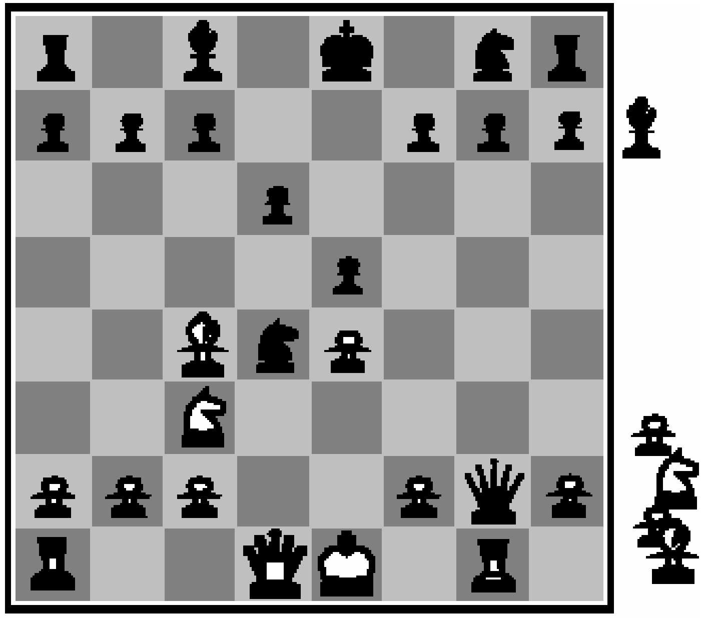
评价函数

（a）黑棋有1个马、2个兵的优势，能够取胜。

（b）黑棋会被白棋吃掉皇后，从而失败。



(a) White to move



(b) White to move

评价函数

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

- 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。

评价函数

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

- 注意：特征和权值并不是国际象棋规则的一部分。
  - 它们只是人类下棋的经验总结。

- 在很难归纳这样的经验规律的游戏中，怎么办？
  - 评价函数的权值可以通过机器学习来估计。

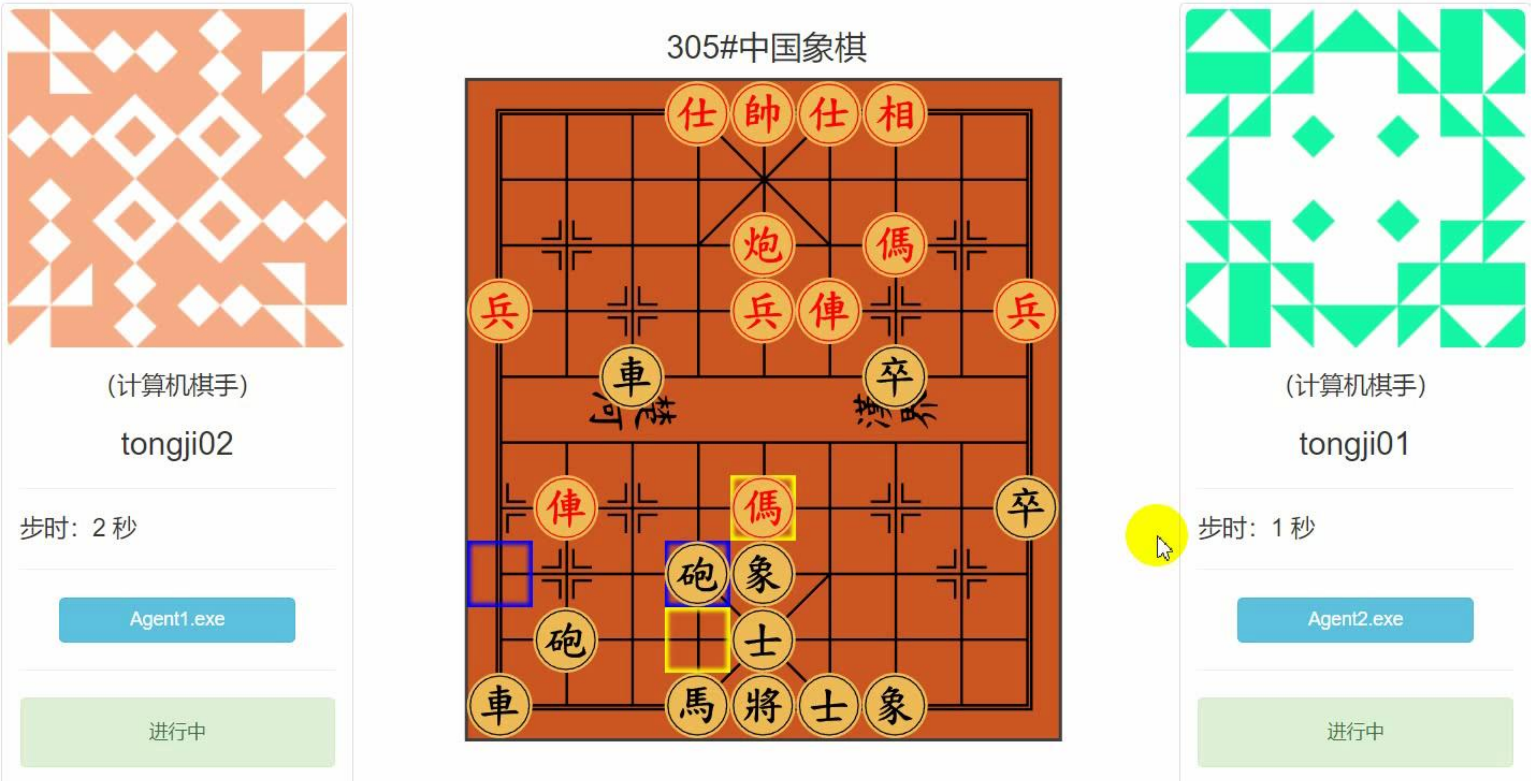
截断搜索

- 最直接的控制搜索次数的方法：设置一个固定的深度限制。
- 更鲁棒的方法：使用迭代深入搜索。
  - 具体实现：不断加大深度优先限制。首先为0，接着为1，然后为2，依此类推。
  - 当时间用完时，程序就返回目前已完成的最深的搜索所选择的招数。

截断搜索

- 由于评价函数的近似性，这些方法可能导致错误。
  - 需要更为复杂的截断测试。
- 评价函数应该只用于那些静止的棋局。
  - 静止的棋局：评价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价函数来说就不能算时静止的。
- 非静止的棋局可以进一步扩展直到静止的棋局，这种额外的搜索称为静止搜索。
  - 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。

中国象棋博弈


(计算机棋手)
tongji02
步时：2 秒
Agent1.exe
进行中
305#中国象棋
(计算机棋手)
tongji01
步时：1 秒
Agent2.exe
进行中

总结

- 博弈是项很有趣的任务!
- 博弈阐述了人工智能的几个重要方面
- 完美决策是难以达到的→ 只能近似

第三章作业

- 7、8、9