第12章 敌对搜索

12.1 双agent博弈

在第10章中提到的问题之一是在其他主动 agent参与的环境中如何计划并行动。如果不了解其他agent如何行动，就只能用感知/计划/动作体系结构，而这种结构不能更加深入地考虑到不可预测的将来。但是，当条件允许时，一个 agent建立的计划可以考虑到其他 agent的行为影响。以双agent的特殊情况为例，在理想的情况下，如这两个agent的行为是互相交替的，它们可以考虑到对方的行为。其中之一先行动，然后是另一个，接着如此反复。

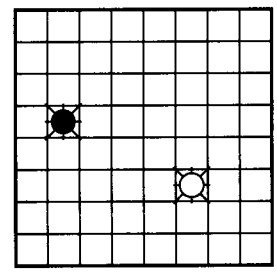


图12-1 在网格环境中的两个机器人

以图12-1中的网格为例，两个机器人，分别命名为“black”和“white”。它们可以向其所在的行或列中的相邻一格交替地移动（比如说，white先移动），而且轮到其中一个时，它必须移动。假设white的目标是与black在同一格，而black的目标是避免发生这种情况。white就可建立一棵搜索树，在交替的级别上，black可能的行动也被考虑进去。图12-2画出了这棵搜索树的一部分。

为了选取最佳的起始动作，white需要分析这棵树来决定可能的结果——即：考虑 black会阻止white实现此目标。在这种冲突的情况下，一个 agent可能发现一步移动，使得无论对方如何移动，它都可达到目标。不过更常见的情况是，由于计算和时间的限制，无论哪方都不能找

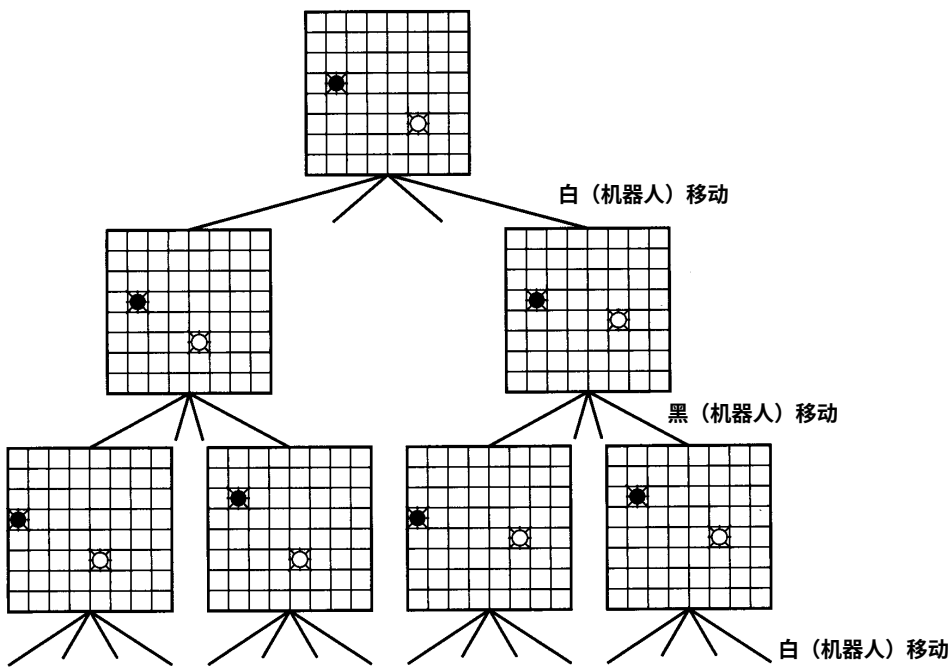


图12-2 两个机器人移动的搜索树

到一个动作以保证取得成功。本书将提供有限范围搜索方法，可用于在这种情况下找到合理的移动方式。在任何情况下，在决定了第一步之后，执行移动，考虑另一方的可能移动，然后在感知/计划/动作方式中重复计划过程。

这个网格例子是双agent、信息完全，零和（*zero-sum*）博奕的一个实例。此处所讨论的是，两个agent（称为博弈者）轮流移动，直到其中任何一方获胜（另一方因此失败），或双方和局。每个博弈者完全熟悉环境及自己和对方可能的移动方式和影响（尽管每个博弈者都不知道另一方在任何情况下究竟会怎样移动）。研究这种博弈可使我们深入了解当有多个 agent时，计划过程可能出现的更多的普遍问题——即使在这些agent的目标并不互相冲突时。

可以看出，有许多常见的博弈，包括国际象棋、西洋跳棋 (dranght)和围棋，都属于这种类型。而且它们的程序已经编写得相当好——有的甚至可以达到参赛的水平。但此处，以并不十分有趣的井字博弈（*Tic-Tac-Toe*）为例，因为它简单，有利于分析搜索技巧。有些博弈（例如西洋双陆棋，*Backgammon*）由于包含了概率因素而导致难以对它们进行分析。

对许多博弈，特别像国际象棋这一类，通常都用图标来描述自己的状态空间，我们用 8×8数组来记录黑白机器人在8×8网格中的不同位置[⊖]；用算子表示博弈的移动，算子把一种状态描述转换为另一种描述，由一个开始节点和每个博弈者的算子隐式定义博弈图，照前面章节的方法建立搜索树，但在选择第一步时要使用一些不同方法。下面将讨论这些方法。

12.2 最小最大化过程

在以下讨论中，命名两个博弈者 *MAX*和*MIN*。我们的任务是为 *MAX*找最佳的移动。假设MAX先移动，然后两个博弈者轮流移动。因此，深度为偶数的节点，对应于 *MAX*下一步移动的位置，称为*MAX*节点；深度为奇数的节点对应于 *MIN*下一步移动的位置，称为 *MIN*节点（博弈树的顶节点深度为0）。*k*层包括深度为2*k*和2*k*+1的节点。通常用层数给出博弈树的搜索程度，它可以表示出向前预测的*MAX*和*MIN*交替运动的回合数。

正如我所说，完全的搜索（赢、输或和局）对于大多数博弈来说是不可行的。据估计完全的国际象棋博弈图解大约有 10^{40} 个节点。即使假设一个后继节点可在 1/3 ns[⊖] 内产生，也需要 10^{24} 年才能产生国际象棋博弈完全搜索图解（据推测，宇宙也只有大约 10^{10} 年的历史）。而且，启发式搜索方法并不会减少起作用的有效分枝因子。因此，对于复杂的博弈，必须认识到搜索到终点是不可能的（除了在博弈快结束时），所以，应该使用类似于在第 10章中描述的有限范围搜索方法。

我们可使用广度优先搜索、深度优先搜索或启发式搜索，除非必须马上修改终止条件。几个人为的终止条件为时间限制、存储空间限制以及在搜索树中最深节点的深度。例如，通常在国际象棋中，如果有任何叶节点代表可继续的好位置，即存在可以继续移动的好位置，就不终止。

搜索结束后，需从搜索树中选取一个最佳首次移动，这个选取方法可以对搜索树的叶节点

⊖ 然而，在这种情况下，当轮到 White移动时，一个更好的表示是基于机器人之间的距离是奇数还是偶数，这个留给读者自己思考。奇偶校验在一对移动后进行。

⊖ 相反，Schaeffer 估计西洋跳棋的完整游戏图只有大约 10^{18}个节点[Schaeffer & Lake 1996]。因为*MAX*只要在每一层出示到达 解法的一个移动就行了，这样，一个解法树将只有这个数的平方根即 10^{9}个节点。Schaeffer 和 Lake 认为“在不久的将来”，完美的机器西洋跳棋游戏将是可得到的。

采用静态评估函数。此评估函数衡量每一个叶节点位置的“值”。这种衡量基于影响这个值的许多不同特性；例如，在西洋跳棋中，一些有用的特性衡量相关部分优势、中心控制、王的中心控制等等。通常分析博弈树时，对*MAX*有利的位置，评估函数将赋于正值；对*MIN*有利的位置赋于负值，接近零的值表示该位置对*MAX*和*MIN*都一样。

一个最佳首步可以由一个最小最大化过程产生（为简单起见，在描述这个过程和基于它的其他过程时，把博弈图当作一棵树）。假设轮到*MAX*从搜索树的叶节点中选取，他肯定选择拥有最大值的节点。因此，*MIN*叶节点的一个*MAX*节点双亲的倒推值就等于叶节点的静态评估值中的最大值。另一方面，*MIN*从叶节点中选取时，必然选值最小的节点（即最负的值）。既然如此，*MAX*叶节点的*MIN*双亲节点被分配一个倒推值，它等于叶节点静态评估值的最小值。在所有叶节点的父节点被赋于倒推值后，开始倒推另一层，假定*MAX*将选择有最大倒推值的*MIN*后继节点，而*MIN*会选择有最小倒推值的*MAX*后继节点。

我们继续逐层对节点评估，直到最后开始节点的后继者被赋于倒推值。假定*MAX*首先移动，*MAX*将选择有最大倒推值的节点作为它的首步。

整个过程的有效性基于这样的假设：开始节点的后继的倒推值比直接从静态评估函数中得到的值更可靠。由于倒推值基于在博弈树中的预先推算，并且取决于在博弈结束时发生的一些特性，这些值往往更加切合实际。

井字博弈的简单例子阐述了最小最大化方法（在井字博弈中，博弈者在3×3数组中轮流标记，一个标记（X），一个标记（O）。先用标记填满一行、一列或一条对角线者便赢得博弈）。假设*MAX*标记（X），*MIN*标记（O），*MAX*先开始。在深度为2的范围内进行广度优先搜索，直到第二级节点全部产生，然后在这些节点代表的位置采用静态评估函数。位置 *p*的静态评估函数*e*（*p*）可如下给出：

假如对任何一方，*p*位置都不是取胜位置

e（*p*）=（对*MAX*开放的完整的行、列或对角线数）－（对 *MIN*开放的完整的行、列或对角线数）

假如对*MAX*来说，*p*是取胜位置，

e（*p*）=∞(用∞来表示一个非常大的正数)

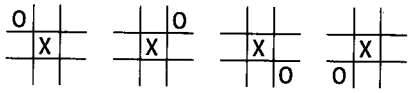
假如对*MIN*来说，*p*是取胜位置，

e（*p*）=－∞

所以，假如*p*为：

我们就得到*e*（*p*）= 6－4 = 2 。

在产生后继者位置时，采用对称法；因此，以下状态是相同的：



（博弈初期，井字博弈的分枝因子由于对称而很小；在后期，由于可用开放空间的数量而仍很小）。

图12-3说明了深度为2的搜索产生的树，静态评估表示在叶节点的右边，倒推值被圈起，

既然在

中有最大倒推值，它被选为首次移动（假如进行完全搜索，这同样也是 *MAX*的最佳首步）。

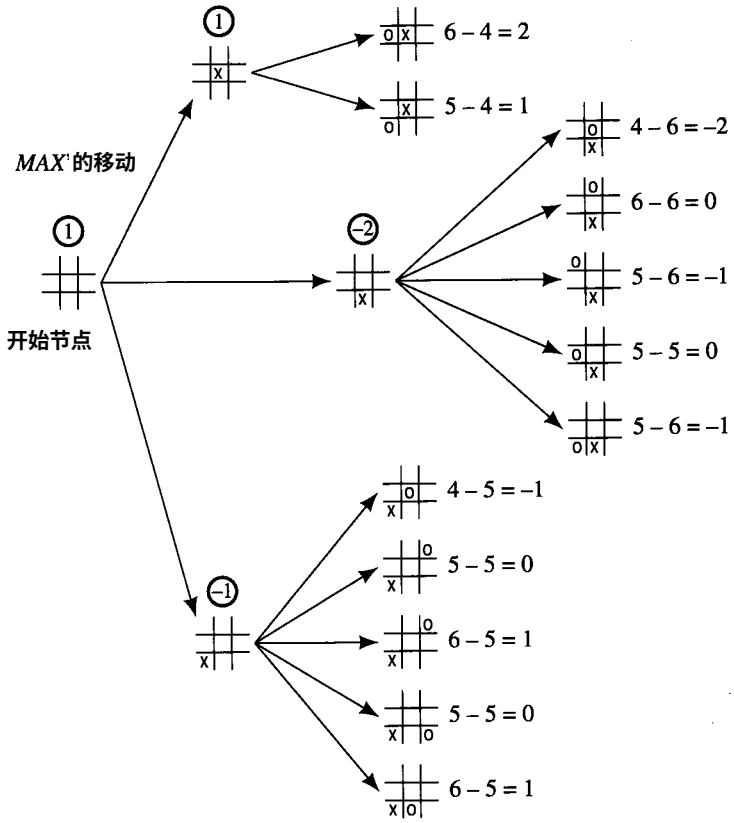


图12-3 井字博弈搜索的开始阶段

现在，按照感知/计划/动作的循环，假设*MAX*走了这一步而且*MIN*在（X）的左边做标记（O）（对*MIN*来说，这不是一个好的走法，它没有用一个好的搜索策略）。在这种布局下，*MAX*进行深度为2的搜索，产生如图12-4所示的树。现在有两个可能的最佳走法，假定*MAX*走其中之一。*MIN*移动以免自己马上失败，形成

*MAX*再次搜索，产生如图12-5所示的树。

该树的一些叶节点（如*A*）代表*MIN*的胜利，估计值为－∞。当倒推这些估计值时，可以看出*MAX*的最佳移动也是惟一可以避免马上失败的一步。现在*MIN*明白*MAX*下一步将取胜，于是只好投降。

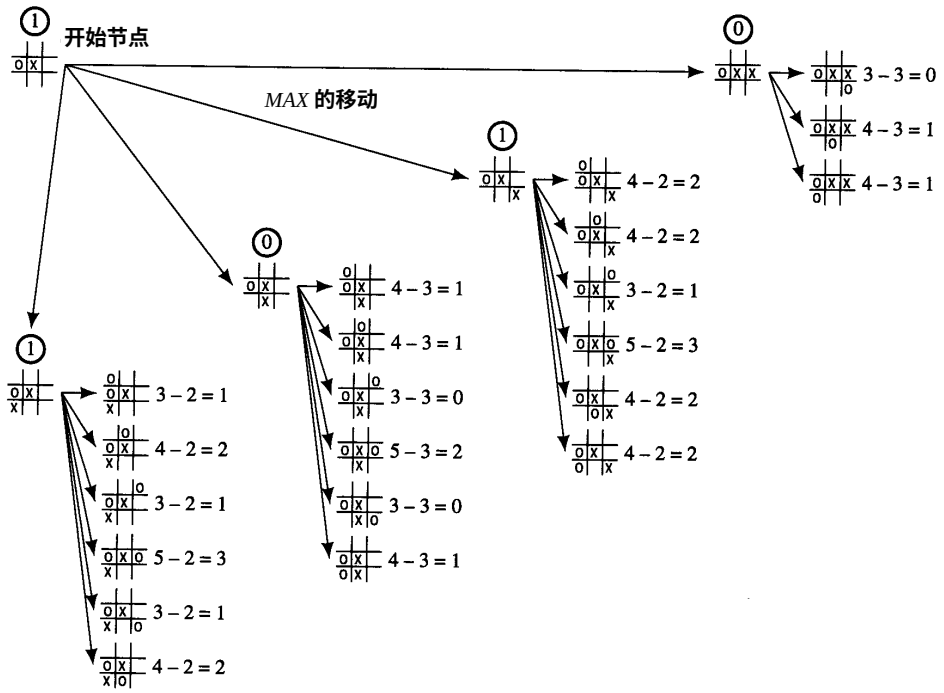


图12-4 井字博弈搜索的第二阶段

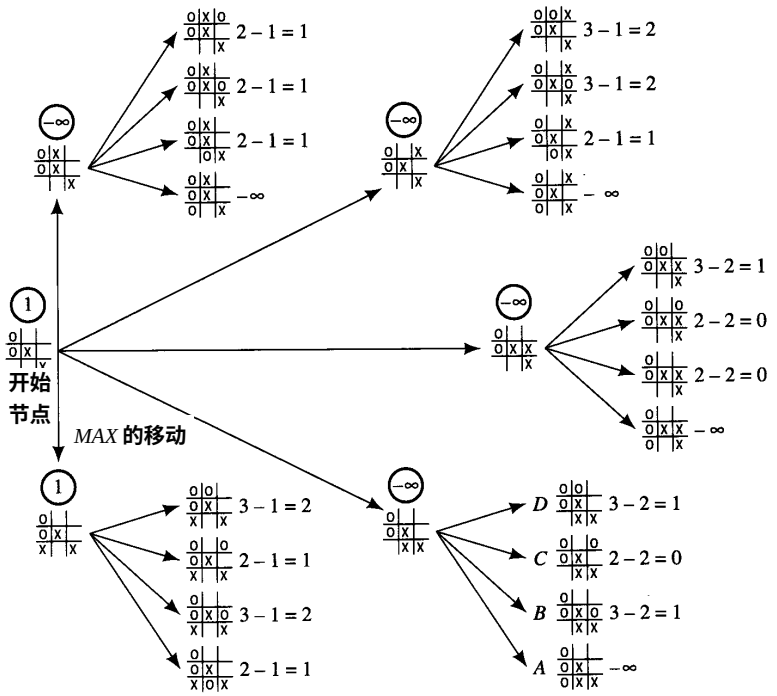


图12-5 井字博弈搜索的最后阶段

12.3 α-β过程

刚才叙述的搜索过程将搜索树的产生和位置评估完全分开，只有在树完全产生后才能对位

置评估。这种分离导致算法粗糙和效率低。假如将叶节点的评估、计算倒推值与树的产生同时进行，就可能大量减少所需搜索的数目（有时非常大），这个过程类似于第10章提到的α截断。

考虑井字博弈搜索中最后阶段的搜索树（图 12-5）。假定一个叶节点一产生便被评估，那么在产生节点A并评估后，便没有必要产生（并评估）节点 *B*、*C*和*D*；即由于*MIN*可以选*A*，并且只能选*A*，我们立即知道*MIN*肯定选择*A*。于是将*A*的父节点的倒推值赋于-∞，继续搜索，不再去费力地产生节点*B*、*C*、*D*并评估（注意，当做更深的搜索时，这种节省尤为明显，因为同样也不需产生节点*B*、*C*和*D*的后代）。重要的是要注意到不产生 *B*、*C*和*D*，决不会影响*MAX*的最佳第一步的产生。

在这个例子中，搜索节约取决于代表了 *MIN*获胜的节点*A*，然而，在搜索树中没有表示任何一方胜利的节点时，也能同样节省搜索操作。

考虑前面表示的井字博弈树的开始阶段（图 12-3），在图12-6中重复了这棵树的一部分。假定采用深度优先搜索，而且节点一产生，便被静态评估，并假定一旦一个位置可给于倒推值时，这个值便被计算出来。设想这样的情况： *A*节点及其所有后继都已产生，而 *B*节点还未产生，*A*节点的倒推值为－1。这时可知开始节点的倒推值范围大于等于－ 1。根据开始节点的其他后继的倒推值，开始节点的最终倒推值可能大于－ 1，但决不会小于－1，我们把这个下界称为开始节点的α值。

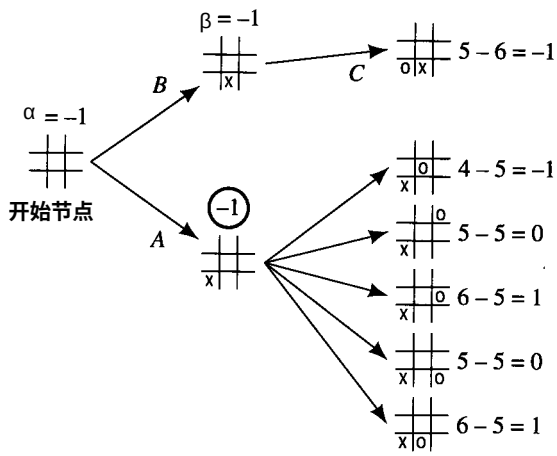


图12-6 井字博弈搜索开始阶段的一部分

继续深度优先搜索，直到节点 *B*及它的第一个后继节点 *C*生成，*C*节点的静态值为－ 1，现在可知*B*节点的倒推值限制在－ 1之下，*B*节点的其他后继节点的静态值可能会使 *B*的最终倒推值小于－1，但决不会大于－1。我们称这个上界为*B*节点的β值，这时可以看出*B*的最终倒推值不会超过开始节点的α值。因此，可以不再继续*B*节点以下的搜索，从而可以肯定*A*节点不会选择*B*节点为后继节点。

通过记录倒推值的界限可减少搜索。通常，当一个节点的后继节点有了倒推值之后，倒推值的界限就会被修改，但我们注意到：

- *MAX*节点的α值（包括开始节点）决不会减小。
- *MIN*节点的β值决不会增加。

由于有这些限制，可以用以下规则来减少搜索量：

1) 当任何*MIN*节点的β值小于等于它的*MAX*的祖先节点的α值，则可中止该*MIN*节点以下的

搜索。该*MIN*节点的最终倒推值即为它的β值。该值与完全最小化搜索得到的值不等，但同样可以选择到相同的最佳移动方式。

2) 当任何*MAX*节点的α值大于等于它的*MIN*的祖先节点的β值时，则可中止该*MAX*节点以下的搜索，该*MAX*节点的最终倒推值等于它的α值。

在搜索中，如下计算α和β值：

- *MAX*节点的α值等于它的后继节点中的最大最终倒推值。
- *MIN*节点的β值等于它的后继节点中的最小最终倒推值。

当按规则1停止搜索时，称产生α剪枝；当按规则2停止搜索时，称产生β剪枝。保存α、β值，并且当可能时进行剪枝的整个过程通常称为α-β过程。当开始节点的所有后继节点都得到倒推值后，过程终止。最佳首步移动是产生有最大倒推值的后继节点的移动。该过程产生的移动方案与简单使用极大极小过程搜索相同深度得到的方案是相似的。惟一的不同是， α-β过程通常用少得多的搜索就可找到最佳首步移动。

上述α-β过程可用一个简洁的伪码递归算法表示。下面的例子评估了节点 *n*相对于α和β截断值的最大最小值，摘自[Pearl 1984，P. 234]:

AB（*n*; α, β）

1) 如果*n*在深度约束上，返回AB(*n*) = *n*的静态值。否则，令n_1，…，n_k，…，n_b为*n*（按顺序）的后继节点，令*k*←1，如果*n*是一个*MAX*节点，转第2步：否则，到第2'步。

2) 令α← max[α，AB(n_k; α，β)]。

3) 如果α≥β，返回β;否则继续。

4) 如果*k* = *b*，返回α；否则前进到 n_{k+1}，例如，让*k* ← *k*+1，转到第2步。

2') β← min[β，AB(n_k; α，β)].

3') 如果β≤α，返回α;否则继续。

4') 如果*k* = *b*，返回β; 否则前进到n_{k+1}例如，让*k* ← *k*+1，转到第2'步

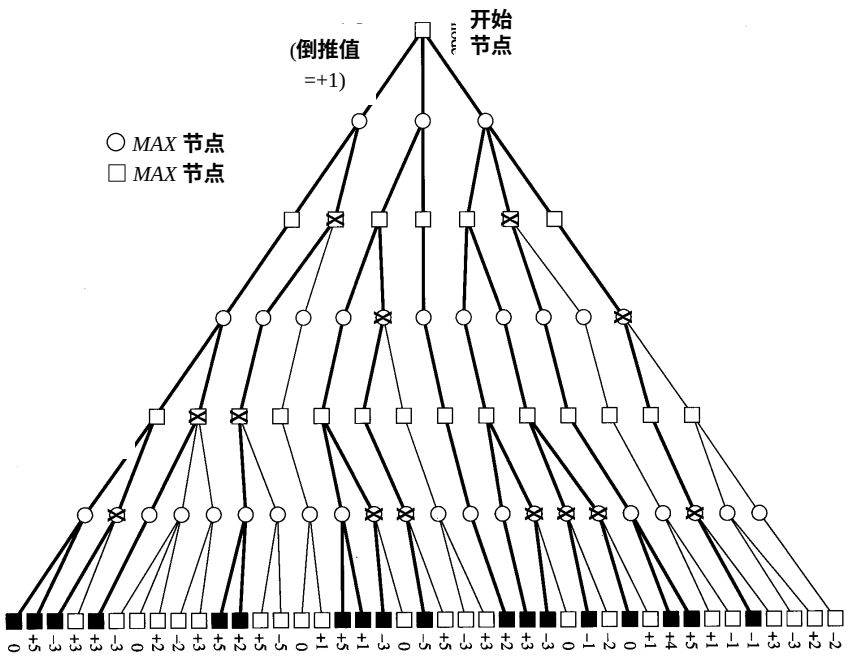


图12-7 α−β搜索的一个例子

通过调用AB（s；- ，+ ）来开始α-β过程，s为开始节点。在整个算法中，α<β。正如我们将会看到的，算法第1步中节点的顺序对算法的效率有很大影响。

图12-7说明了α-β过程的应用之一。画出一棵6层的搜索树，方形表示*MAX*节点，圆圈表示*MIN*节点，叶节点注明静态值。假定用α−β过程进行深度优先搜索（我习惯先产生最底层节点）。α−β过程产生的子树用黑色分枝标明，剪枝节点中画 X，原先41个叶节点中只有18个需评估（可以试着重复此例中α−β搜索来检测对该过程的理解）。

12.4 α-β过程的搜索效率

为了完成α和β剪枝，至少部分搜索树要产生至最大深度值，因为 α、β值需要基于叶节点的静态值。因此，使用 α-β过程时通常用到深度优先搜索，而且，搜索中可以得到的剪枝数取决于早期的α、β值与最终倒推值的近似程度。

假设一棵树深度为d，并且每个节点（除了叶节点）都有b个后继节点，这样一棵树大约有b^d 个叶节点。假定α-β过程产生的后继节点按其倒推值大小排列 —— 对*MIN*节点按从小到大顺序，对*MAX*节点按从大到小顺序（当然，它们的倒推值是不可能在后继节点产生前知道的，因此，除非偶然，否则该顺序不可能得到）。

这种排序可以使剪枝的数目最大化，并使产生的末端节点数最小化。我们用 N_d表示叶节点的最小值，可如下表示[Slagle & Dixon 1969, Knuth & Moore 1975]：

$$N_d = \begin{cases} 2b^{d/2} - 1 & d \text{ 为偶数} \\ b^{(d+1)/2} + b^{(d-1)/2} - 1 & d \text{ 为奇数} \end{cases}$$

也就是说，优化的α-β搜索产生的深度为d的树，其叶节点的数目等同于不用 α-β搜索时深度为$d/2$ 产生的叶节点数。因此，当不用 α-β的搜索进行到$d/2$ 深度时，α-β搜索（最佳排序）将进行到d 深度，换而言之，用最佳排序的α-β搜索，减少了大约从b到 $\sqrt{b}$ 的有效分枝因子。

当然，最佳节点排序是不可实现的（假如可实现，就完全不需要搜索过程了）。最糟糕的情况下，α-β搜索不会产生剪枝，有效分枝因子不变化。 Pearl指出，假如后继节点随机排列，α-β搜索深度会增加约4/3倍，即平均分枝因子减少到大约 $\sqrt[4]{b^3}$ [Pearl 1982b]（见[pearl 1984，第9章]详细分析及注解）。实际上，假如将好的试探方法用于排列后继节点， α-β过程通常可以最大程度地减少有效分枝因子。

排列后继节点最直接的方法是使用静态评估函数。节点排序的另一种技巧来自迭代加深的副效应——迭代加深首次运用在chess博弈程序CHESS4.5中[slate & Atkin 1977]。程序首先搜索一层，然后评价最佳移动；然后又搜索两层，评价最佳移动；如此继续。这种多重搜索看似浪费的主要原因是博弈程序通常有时间限制，由于可用时间资源的关系，向更深层次的搜索可能随时终止，搜索得到被认为最佳的移动。这种副效应在节点排序中体现为，在 K层搜索中被认为最佳的节点，可用于在K+1层搜索中排列节点。

12.5 其他重要问题

对于大多数博弈，必须在找到终止点之前结束搜索（有限范围内），这就导致了许多困难。其一，搜索可能终止在使*MAX*（或*MIN*）能实现大的移动的位置。因此，许多博弈搜索方法确保一个位置为静止，直到在这个位置上搜索结束。如果一个位置的静态值与它前面一两步之内的倒推值相差不大时，称之为静止的。

即使沿一定分枝在终止搜索前保持静止，在搜索范围外，还存在许多情况难以预料。有些博弈树存在一些情形，会不可避免地产生*MAX*获胜，也就是说，无论*MIN*如何走，一定范围内都不会影响*MAX*的胜利。这种所谓的范围效应就是博弈树面临的基本困难之一。

最大最小法和α-β方法都假设对手会选最佳位置移动。在有些情况下这是不合适的。例如，假设*MAX*看起来要输了 —— 假定*MIN*走最佳步，然而，*MAX*依然可以摆脱困境，只要*MIN*犯个错误。最大最小法不会推荐这种走法，然而，由于博弈看来无论如何都会输，那么猜测 *MIN*会出错也不会带来任何损失。假如博弈一方有另一方的策略模型，最大最小法也不合适，也许，另一方并非是对手而只是另一个变化的agent。

12.6 概率博弈

有些博弈，如西洋双陆棋（*Backgammon*），包含了概率因素。例如，一方允许的移动取决于骰子的投掷结果，图12-8表示了这种博弈的博弈树（为了使这棵树简单些，只表示出投掷模型的6种可能的不同结果）。*MAX*和*MIN*的每次移动都牵涉到投掷一次骰子。假想在每一次投掷后，一个想像的第三方博弈者，骰子，也做了一次移动。这种移动由概率决定，投掷时， 6种结果都是等可能的，但概率因素也可能涉及任意的一种概率分布。

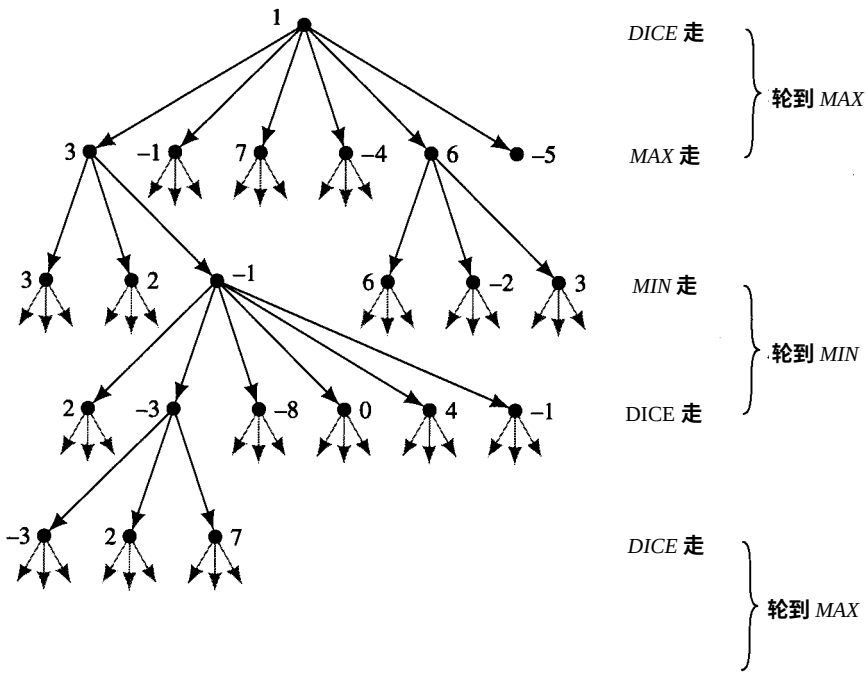


图12-8 一个掷骰子的博弈树

在涉及到概率的博弈树中，也可以倒推值，除了有时在有概率移动的节点，我们取后继节点值的平均值而非最大值或最小值[⊖]。图12-8中节点边上的数值为倒推值。我们倒推以下值：代表*MIN*移动的节点的后继节点值的最小值；代表*MAX*移动的节点的后继节点值的最大值；代表骰子移动的节点的后继节点的期望值。这种修改过的倒推程序有时称为期望的最大值(*expectimaxing*)。

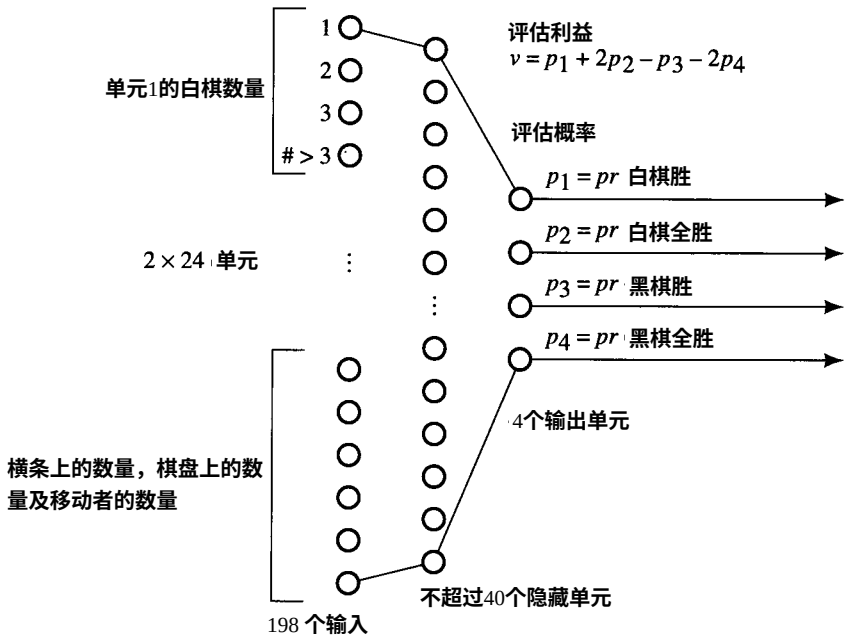
⊖ 当然，如果我们对概率移动的结果非常保守，无论什么时候只要骰子移动，我们就倒推 *MAX*的最坏值。

引入概率移动会使博弈树搜索产生过多的分枝，此时，一个好的静态评估函数对限制搜索的深度是相当重要的。假如可找到一个足够好的静态评估函数，*MAX*在一些位置选取移动时只需向前预测一步，然后选择可使这些位置的静态值最大的移动方式。下面我们将讨论学习动作策略的技术，以及在有很大的分枝因子博弈中学习静态评估函数以允许搜索的技术。

12.7 学习评估函数

有些博弈（如围棋）的每一步都存在着多种移动，以至于不能进行深层搜索⊖。在不进行搜索的情况下，可通过后继节点的静态值或各种模式识别技术对当前布局作出反应来选择移动。例如对围棋，就曾经试图寻找一些对位置评估并做出反应的方法 [Zobrist 1970，Reitman & Wilcox 1979，Kierulf，Chen，& Nievergelt 1990]。

有时，可通过神经网络来了解好的静态评估函数，西洋双陆棋博弈就是这方面的一个好例子。一个叫TD-GAMMON的程序[Tesauro 1992，Tesauro 1995]通过一个具有层次结构且向前反馈的神经网络来操作西洋双陆棋博弈，神经网络的结构和编码如图 12-9所示。198种输入（代表西洋双陆棋格局）被全部连接到隐藏单元，隐藏单元全部连接到输出单元。隐藏单元与输出单元都是Sigmoid。输出单元根据输入格局产生的结果的可能值产生4个估计值p_1、p_2、p_3和p_4。布局位置的全局值由一个估算的性能指标给出：$v = p_1+2p_2-p_3-2p_4$。在用网络操作西洋双陆棋时，投掷骰子，现有布局的任何可能的移动产生新的布局，并由网络来评估骰子的变化。选取含有最好v值的布局，进行产生这种格局的移动（假如是白棋移动，最大的 v值最好；假如是黑棋移动，最小的v值最好）。



隐藏和输出单元是S型的学习率：c=0.1；随机选取的初始权值在－0.5和+0.5之间。

图12-9 TD-GAMMON 网络

在实际使用网络操作博弈时，会有时间差异。每移动一步，通过反向传播调整网络权值，

⊖ 下围棋的人显然可以对布局的孤立子集进行搜索，一些有能力的程序下围棋也是可能的。

使预期指标从初始位置更接近结果位置。为简单起见，解释这种方法时，假定它只有时间差（实际上用到一个变量，但这里该变量与我们无关），假如v_t是t时刻网络博奕结果的估计值（一步移动之前），v_{t+1}是时间$t+1$时的估计值（一步移动之后），标准的时间差的权修正值规则为：

$$\Delta \mathbf{W}_t = c(v_{t+1} - v_t)\frac{\partial v_t}{\partial \mathbf{W}}$$

$\mathbf{W}_t$是t时刻网络中所有权值的向量，$\frac{\partial v_t}{\partial \mathbf{W}}$是在权空间中对$v_t$的梯度（对一个具有多层结构并向前的网络，如TDGAMMON中的网络，每一层权向量中分量的变化都可用第3章中的方式描述）。训练网络以便对所有的t，一步移动前的输入产生的输出v_t总是趋向于一步移动之后输入产生的输出值v_{t+1}（就像在数值迭代中）。在用这种训练方法的测试实验中，使用网络对成千上万种博弈进行了操作（大约包括了150万种博弈）。

经过完善训练，网络的性能几乎是一流的。在用向前搜索程序操作了 40个博弈后，Bill Robertie（前西洋双陆棋世界冠军）估计 TD-GAMMON 2.1的水平处于顶级高手——极其接近（相差只有几个百分点）世界上最好的人类选手[Tesauro 1995]。

12.8 补充读物和讨论

如同谜宫一样，博弈对完善和测试人工智能技术也很重要。例如，[Russell & Wefald 1991，第4章]提出并评估了博弈树搜索算法（MGSS*和MGSS2），它使用了元级计算（涉及继续搜索的期望值）来减小博弈树，比α-β过程更有效。Berliner's B*算法使用间隔范围[Berliner 1979]，可进行更有效的剪枝。[korf 1991]把α-β过程扩展到多人博弈。

有些书中没有使用数值评估函数，而利用模式识别技巧来判断位置的好坏，这种技术已被用于国际象棋残局博弈的程序中[Huberman 1968，Bratko & Michie 1980]。

博弈操作方面早期成功的著作来自于 Arthur Samuel ，他开发了西洋跳棋博弈的机器学习方法[Samuel 1959， Samuel 1967]。Samuel的 西洋跳棋操作程序达到冠军级水平，Alberta大学的Jonathan Schaeffer的CHINOOK程序[Schaeffer，et al. 1992，Schaeffer 1997]现在被认为是世界西洋跳棋冠军。 1997年，IBM程序“深蓝”在一场冠军赛中击败了国际象棋冠军 Garry Kasparov（盖利·卡斯帕洛夫）。

[Newborn 1996]是一本关于计算机下棋的书，它将历史追溯到Garry kasparov战败“深蓝”的1996年。要了解这本书及关于象棋博弈作为人工智能研究平台方面的评论，见[McCarthy 1997]。McCarthy认为如果象棋程序采用类似人的推理方法，用较少的搜索就能有更好的性能。他认为假如研究者们用它测试基于知识的推理方法，象棋又会成为人工智能的“果蝇”(*Drosophila.*)

[Michie 1966]提出了期望最大(*expectimax*)一词，并进行了实验。

[Lee & Mahajan 1988] 对奥赛罗博弈（Othello）采用加强学习方法，[Schraudolph，Dayan & Sejnowski 1994]将时间差方法运用于围棋博弈。

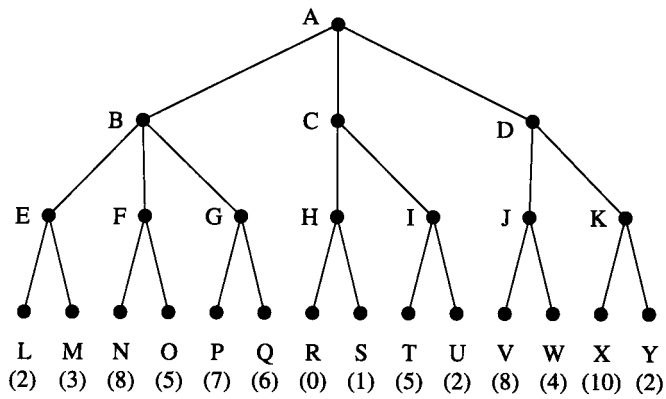
要更多地了解博弈搜索方法，见[Pearl 1984，第9章]

习题

12.1 为什么在博弈的程序中搜索总是从当前位置向前而不是从目标向回搜索？

12.2 考虑下面的博弈树，静态值（在叶节点的圆括号中）都是从第一个博弈者的角度得出

的，假设第一个博弈者为*MAX*一方。



1) 第一个博弈者将选择什么移动？

2) 假如使用α-β算法，哪些节点无须检验？——假设节点按从左到右顺序检验。

12.3 假设用α-β截断来决定博弈树中的移动值，而且已断定节点*n*及其子女可被截断。假如博弈树实际上是一个图，并且有另一条路可到达节点 *n*，你是否仍认为应删除该节点？如果应该删除，证明它，否则举出一个反例。

12.4 许多博弈的程序在跳向下一步时不存储搜索结果，而是每次轮到机器移动时，它都完全重新开始，为什么？

12.5 谈谈（限于一页半以内）你将如何改进最小最大化策略，使之可用于在三人的完全信息博弈中为其中一方选择移动（一个完全信息博弈指任何一个博弈者都完全了解博弈状况，象棋即此类博弈，而桥牌不是）。假设有三个博弈者*A*、*B*和*C*，轮流开始。假设博弈可以平局结束或以任何一方胜利结束。还可假设博弈树不可搜索至终点，但为了选择移动，非终点位置的评估需用某个类似最小最大化策略的过程来倒推。