第六章 搜索方法

【本章目标】

- 理解和掌握搜索的概念和评价指标。
- 理解和掌握人工智能中常用的搜索方法。

前面章节介绍了人工智能中的知识表示方法，这是人工智能问题求解的基础内容。本章在此基础上主要介绍人工智能问题求解过程中用到的基本搜索方法，这些方法主要包括：图搜索策略、盲目搜索、启发式搜索和博弈搜索等内容。

6.1 搜索概念及其评价指标简介

搜索技术作为人工智能的主要技术之一，在专家系统、自然语言理解、自动程序设计、模式识别、机器人学、信息检索和博弈等人工智能领域得到了广泛的应用。人工智能在处理给定问题时，一般是找到能够达到所希望目标的动作序列，并使其所付出的代价最小、性能最好，而合适的搜索技术恰好能够找到人工智能既定目标的动作序列。

什么是搜索呢？实际上，搜索就是根据问题的实际情况，按照一定的策略或规则，从知识库中寻找可利用的知识，从而构造出一条使问题获得解决的推理路线的过程。其包含两层含义:

（1）要找到从初始事实到问题最终答案的一条推理路线。

（2）找到的这条路线是时间和空间复杂度最小的求解路线。

一个搜索方法的优劣直接影响到该算法在人工智能中解决问题的效果。通常情况下，搜索方法的执行效率通过下列 4 个指标来进行评价：

- **完备性**：如果存在一个解答，该算法是否保证能够找到这个解？
- **时间复杂性**：需要多长时间可以找到合适解？
- **空间复杂性**：执行搜索过程需要多大存储空间？
- **最优性**：如果存在多个解，该算法是否能够找到最高质量的解？

如果一个搜索方法具有完备性好、时空复杂度低，且具有最优解，那么该搜索方法在人工智能中能够取得较好解决问题的效果，本章将介绍常用的搜索方法。

6.2 图搜索策略

图搜索策略是一种在图中寻找解路径的方法，该搜索策略只记录状态空间那些被搜索过的状态，它们组成一个搜索图叫 G。G 由两张表内的节点组成：（1）OPEN 表：用于存放已经生成，且已用启发式函数作过估计或评价，但尚未产生它们的后继节点的那些结点，也称未考察结点，如图 6.1 所示。（2）CLOSED 表：用于存放已经生成，且已考察过的结点，如图 6.2 所示。

状态节点	父节点

图 6.1 Open 表的数据结构

编号	状态节点	父节点

图 6.2　Closed 表的数据结构

图搜索策略的一般搜索过程如下：

(1) 把初始节点 S0 放入 OPEN 表，并建立目前只包含 S_0 的图 G。

(2) 检查 OPEN 表是否为空，若为空则问题无解,退出。

(3) 取出 OPEN 表中第一个节点放入 CLOSED 表，该节点记作 n。

(4) 考察 n 是否为目标节点：如果是，则问题得解，退出。

(5) 扩展节点 n，生成一组子节点.把其中不是节点 n 先辈的那些节点记作集合 M，并把这些子节点作为节点 n 的子节点加入 G 中。

(6) 针对 M 中子节点的不同情况,分别进行处理：

a.对于那些未曾在 G 中出现过的 M 成员设置一个指向父节点(即 n)的指针，并把它们放入 OPEN 表。

b.对于那些先前已在 G 中出现过的 M 成员，确定是否需要修改它指向父节点的指针。

c.对于那些先前已在 G 中出现并且已经扩展了的 M 成员，确定是否需要修改其后继节点指向父节点的指针。

(7) 按某种搜索策略对 OPEN 表中的节点进行排序。

(8) 转第(2)步。具体搜索过程如图 6.3 所示。

实际上，图搜索策略就是如何从叶节点中选择一个节点扩展，以便尽快地找到一条符合条件的路径。不同的选择方法就构成了不同的图搜索策略。如果在选择节点时利用了与问题相关的知识或者启发信息，那么就称作启发式搜索，否则就称作盲目搜索。启发式搜索和盲目搜索是图搜索中比较常用的两种搜索方法，这两种搜索方法在后面两节内容中将做详细介绍[5][6]。

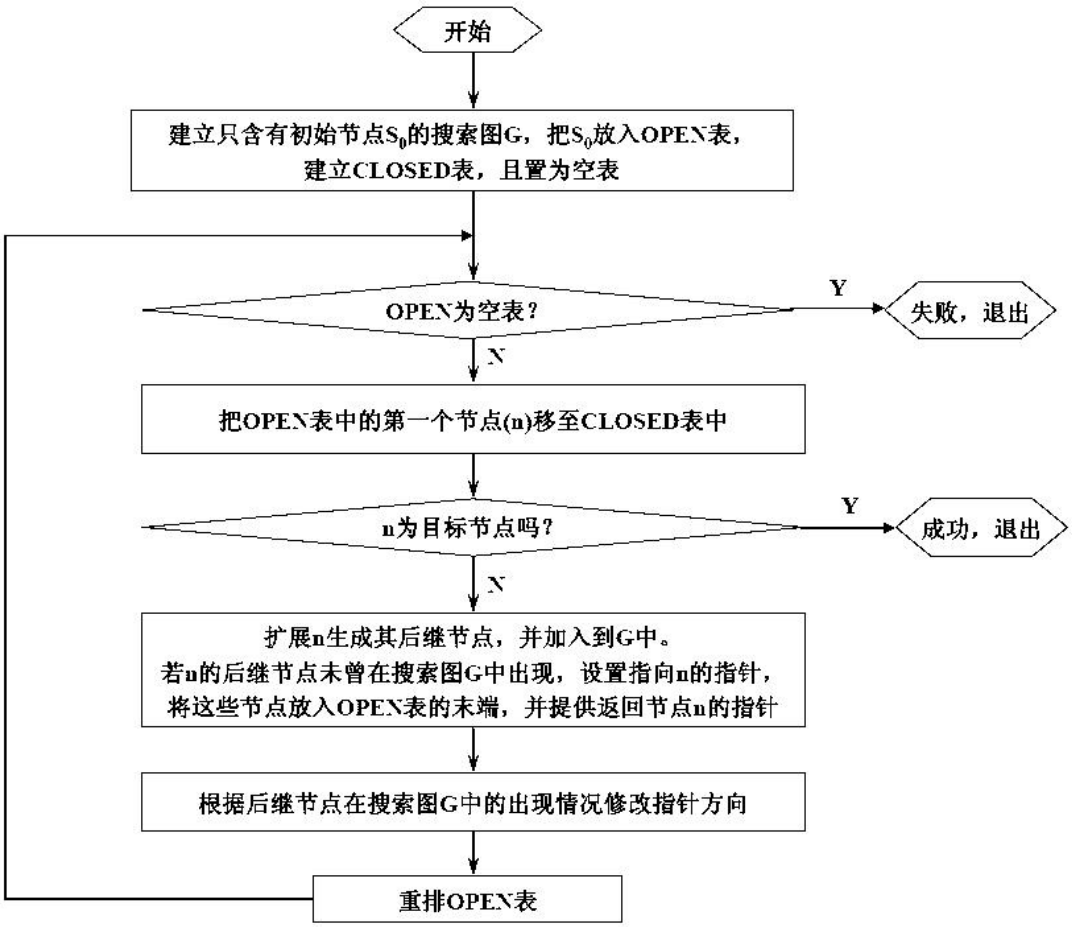


图 6.3 Open 表的数据结构

6.3 盲目搜索

盲目搜索方法又叫非启发式搜索，是一种不利用问题的有关信息，而根据事先确定好的搜索策略进行搜索的方法，也称作无信息搜索方法。该方法只适用于求解比较简单的问题，宽度优先搜索和深度优先搜索是常用的两种盲目搜索方法。

6.3.1 宽度优先搜索

宽度优先搜索算法（又称广度优先搜索算法）是最简单的图的搜索算法之一，这一算法也是很多重要的图的算法的原型。Dijksta 单源最短路径算法和 Prim 最小生成树算法都采用了与宽度优先搜索类似的思想。

宽度优先搜索的核心思想是：从初始结点开始，应用算符生成第一层结点，检查目标结点是否在这些后继结点中，若没有，再用产生式规则将所有第一层的结点逐一扩展，得到第二层结点，并逐一检查第二层结点中是否包含目标结点。若没有，再用算符逐一扩展第二层所有结点……，如此依次扩展，直到发现目标结点为止，如图 6.4 所示。

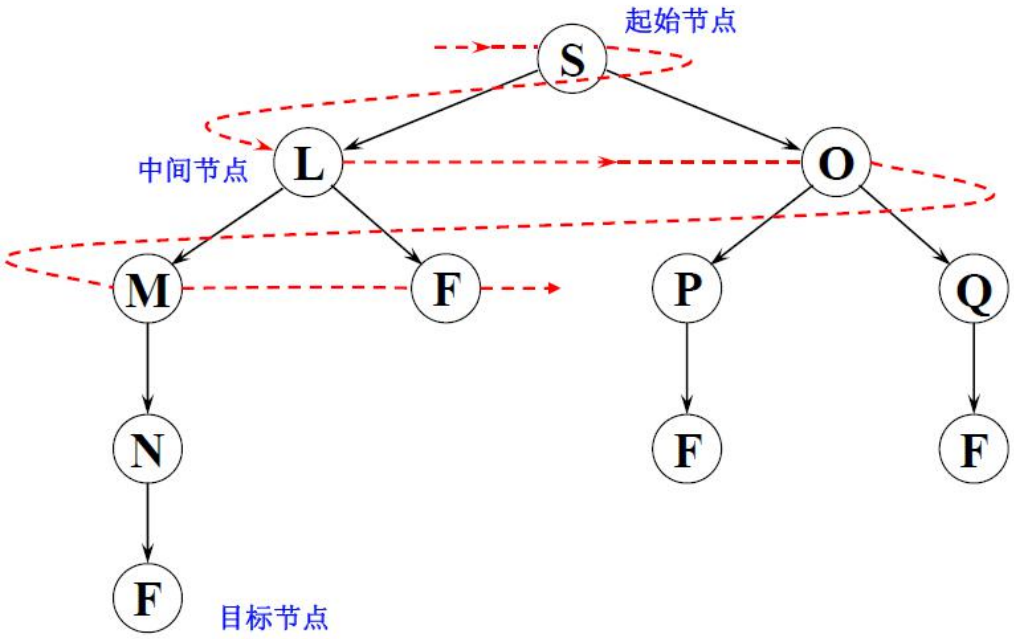


图 6.4 宽度优先搜索过程

宽度优先搜索算法的步骤如下：

（1）把初始节点 S_0 放入 OPEN 表中。

（2）若 OPEN 表为空，则搜索失败,退出。

（3）取 OPEN 表中前面第一个节点 N 放在 CLOSED 表中，并冠以顺序编号 n。

（4）若目标节点 Sg= N,则搜索成功，算法结束。

（5）若 N 不可扩展，则转到第（2）步执行。

（6）扩展 N, 将其所有子节点配上指向 N 的指针依次放入 OPEN 表尾部, 则转到第（2）步执行，如图 6.5 所示[5][6]。

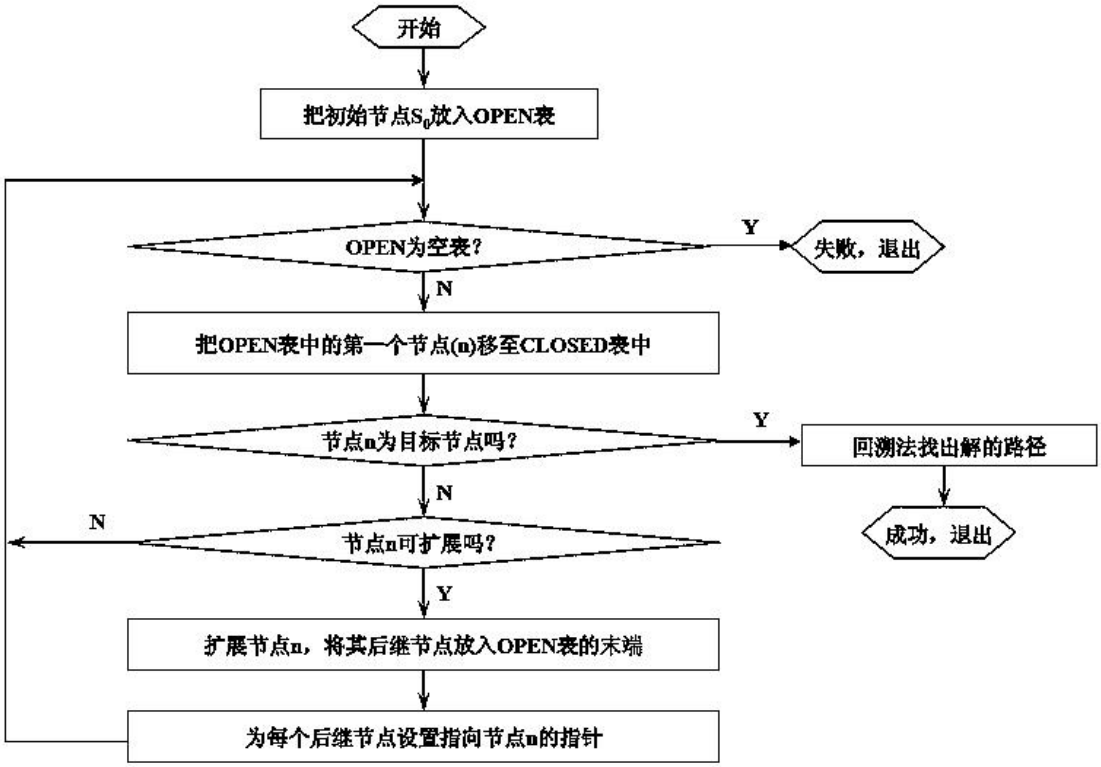


图 6.5 宽度优先搜索算法流程图

6.3.2 深度优先搜索

深度优先搜索所遵循的搜索策略是尽可能“深”地搜索图。深度优先搜索方法的核心思想是：从根节点开始，首先扩展最新产生的节点，即沿着搜索树的深度发展下去，一直到没有后继结点处时再返回，换一条路径走下去。就是在搜索树的每一层始终先只扩展一个子节点，不断地向纵深前进直到不能再前进（到达叶子节点或受到深度限制）时，才从当前节点返回到上一级节点，沿另一方向又继续前进，一直重复这个过程，直到发现目标结点为止，这种方法的搜索树是从树根开始一枝一枝逐渐形成的，如图 6.6 所示。

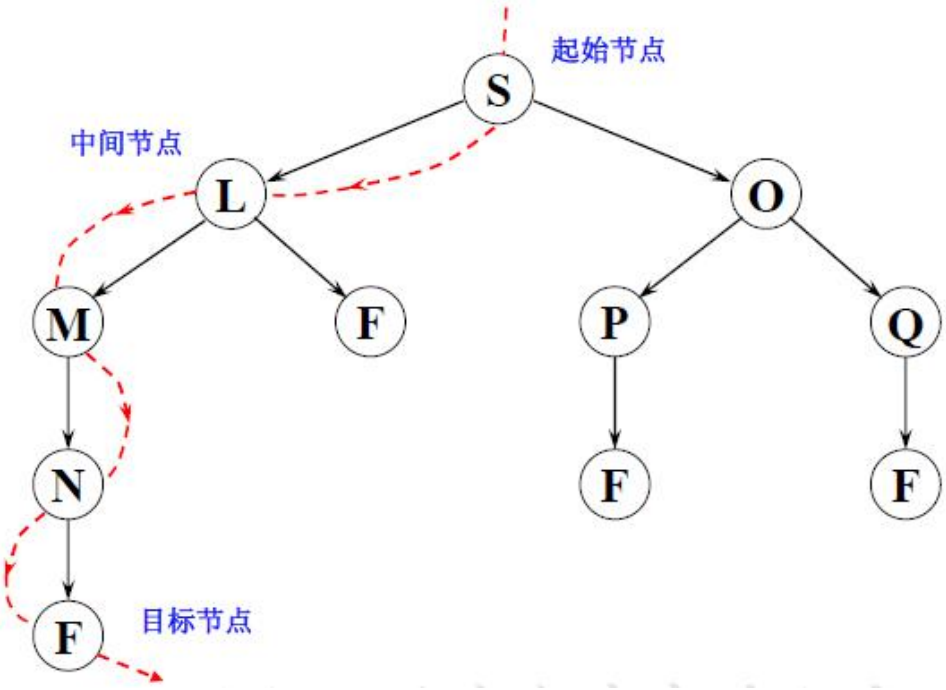


图 6.6 深度优先搜索过程

宽度优先搜索算法的步骤如下：

（1）把初始节点 S_0 放入 OPEN 表中。

（2）若 OPEN 表为空, 则搜索失败, 退出。

（3）取 OPEN 表中前面第一个节点 N 放入 CLOSED 表中,并冠以顺序编号 n。

（4）若目标节点 Sg=N, 则搜索成功，算法结束。

（5）若 N 不可扩展, 则转到第（2）步执行。

（6）扩展 N, 将其所有子节点配上指向 N 的指针依次放入 OPEN 表的首部, 则转到第（2）步执行，如图 6.7 所示。

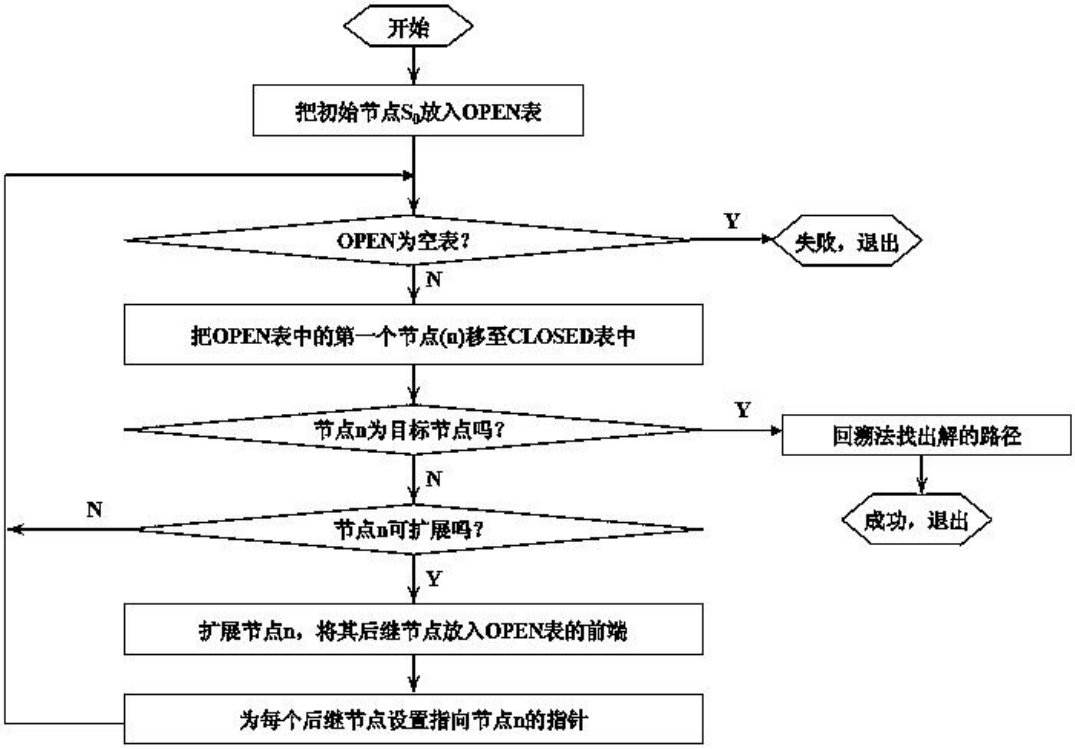


图 6.7 深度优先搜索算法流程图

宽度优先搜索算法的时间和空间复杂度都比较高，当目标节点距离初始节点较远时会产生许多无用的节点，搜索效率低。深度优先搜索比宽度优先搜索算法需要较少的空间，该算法只需要保存搜索树的一部分，它由当前正在搜索的路径和该路径上还没有完全展开的节点标志组成[5][6]。

6.4 启发式搜索

启发式搜索(Heuristically Search)方法又称为有信息搜索(Informed Search)方法，这种方法是一种利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的一种图搜索方法。启发式搜索方法核心思想是：利用一个评估函数对状态空间中的搜索中的每一个搜索位置的价值进行评估，决定先尝试哪一个方案，从而可省略大量无用的搜索路径，极大的优化普通的广度优先搜索。

与普通的广度优先搜索不同的是，启发式搜索会优先顺着有启发性和具有特定信息的结点搜索下去， 这些结点可能是达到目标状态空间的最好路径。在启发式搜索方法核心思想中最主要的是通过引入一个启发式函数(或称为评估函数)，来表示从初始节点经某个节点到达目标节点的最小代价的代价估计值。我们定义评估函数为：

$f(x)=g(x)+h(x)$，其中 $f(x)$ 表示节点 x 的估价函数，$g(x)$ 表示初始节点 S_0 到节点 x 已实际付出的代价，$h(x)$表示从节点 x 到目标节点最优路径的估计代价。搜索的启发信息主要由 $h(x)$来体现，所以把 $h(x)$称作启发函数。

A*算法和 A 算法是常用的启发式搜索算法。在状态空间图的一般搜索算法中，如果根据估价函数 $f(x)=g(x)+h(x)$对 OPEN 表中的节点进行排序，并且要求启发函数 $h(x)$的一个下界，即 $h(x)\leq h^*(x)$, 则这种状态空间图的搜索算法称为 A*算法。如果对启发函数 $h(x)$，不限制条件 $h(x)\leq h^*(x)$，则这种状态空间图的搜索算法称为 A 算法。这两种算法在本质上是非常类似的，所以下面重点对 A*算法的搜索思想及其流程进行介绍。

A*算法对状态空间图的一般搜索算法中的扩展节点选择方法做了一些限制，选择了一个比较特殊的估价函数。这时的估价函数 $f(x)=g(x)+h(x)$是对下列函数 $f^*(x)=g^*(x)+h^*(x)$的一种估计或近似（即：$f(x)$是对 $f^*(x)$的一种评估，$g(x)$是对 $g^*(x)$的估计，$h(x)$是对 $h^*(x)$的估计），其中，$f^*(x)$表示从节点 S_0 到节点 x 的一条最佳路径的实际代价加上从节点 x 到目标节点的一条最佳路径的代价之和；$g^*(x)$表示从节

点 S0 到节点 x 之间最小代价路径的实际代价。h*(x)表示从节点 x 到目标节点的最小代价路径上的代价；g(x)表示从初始节点 S_0 到节点 x 的路径代价，恒有 g(x)≥g*(x)。在 A*算法中要求启发函数 h(x)是 h*(x)的下界，即对所有的 x 均有 h(x)≤h*(x)，这保证了 A*算法能够找到最优解。

A*搜索算法的步骤如下：

（1）把初始节点 S_0 放入 OPEN 表中。

（2）若 OPEN 表为空，则搜索失败，算法结束。

（3）移出 OPEN 中第一个节点 N 放入 CLOSED 表中，并标以顺序号 n。

（4）若目标节点 Sg=N, 则搜索成功，算法结束。

（5）若 N 不可扩展，则转到第（2）步执行。

（6）扩展 N，生成一组子节点，对这组子节点作相应处理（即：删除重复节点和修改返回指针处理）后，放入 OPEN 表，按评价函数的升序重新排序 OPEN 表，转到第（2）步执行，如图 6.8 所示[5][6]。

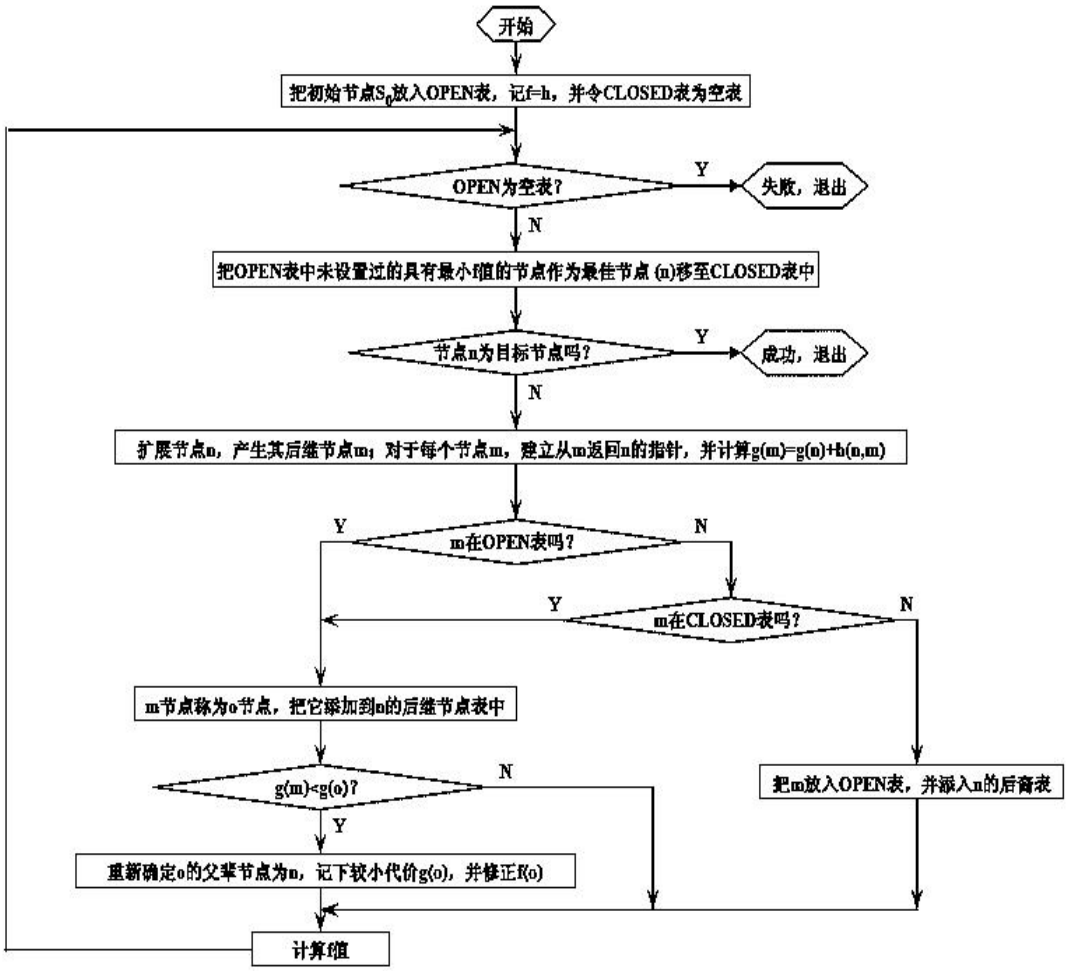


图 6.8　A*搜索算法流程图

启发式搜索方法弥补盲目搜索方法的不足，提高了图的搜索效率，该搜索方法的应用非常广泛，在现代社会的很多领域均有应用，例如：图像边缘提取的启发式搜索算法、基于启发式图搜索的最小测点集优选新算法和模糊图的启发式搜索算法等都是在启发式搜索方法的基础上发展起来的。

6.5 博弈搜索

6.5.1 博弈和博弈树的概念

博弈是一类富有智能行为的竞争活动，如下棋、打牌、战争等，是启发式搜索的一个重要应用领域。最简单的博弈是双人完备信息博弈。双人完备信息博弈是指两位选手对垒，轮流走步，每一方不仅知

道对方已经走过的棋步，而且还能估计出对方未来的走步，对弈的结果是一方赢，另一方输，或者双方和局[1][2] [3][4]。

在博弈过程中, 任何一方都希望自己取得胜利。因此，当某一方当前有多个行动方案可供选择时, 他总是挑选对自己最为有利而对对方最为不利的那个行动方案。此时，如果我们站在 A 方的立场上，则可供 A 方选择的若干行动方案之间是“或”关系, 因为主动权操在 A 方手里，他或者选择这个行动方案, 或者选择另一个行动方案, 完全由 A 方自己决定。当 A 方选取任一方案走了一步后，B 方也有若干个可供选择的行动方案, 此时这些行动方案对 A 方来说它们之间则是“与”关系，因为这时主动权操在 B 方手里，这些可供选择的行动方案中的任何一个都可能被 B 方选中，A 方必须应付每一种情况的发生。如果站在某一方把上述博弈过程用图表示出来, 则得到的是一棵“与或树”，我们把上述的“与或树”称作博弈树,它有如下特点：

(1) 博弈的初始格局是初始节点。

(2) 在博弈树中, “或”节点和“与”节点是逐层交替出现的。自己一方扩展的节点之间是“或”关系, 对方扩展的节点之间是“与”关系。双方轮流地扩展节点。

(3) 所有自己一方获胜的终局都是本原问题, 相应的节点是可解节点；所有使对方获胜的终局都是不可解节点。

6.5.2 极大极小值分析法

极大极小值分析法本质上是一种为博弈过程中一方找到最优行动方案的方法。为寻找当前最优行动方案，需要分析对比每个行动方

案会带来的后果。也就是说，，就是要考虑每一方案实施后对方可能采取的所有行动，并计算可能的分值。为了计算分值，需要定义一个估值函数 f，并对当前博弈树叶节点分值进行估算。叶节点分值计算出后，据此估算把父节点分值。估算方法的基本思想是：针对“或”节点，选择子节点内最高分值作为父节点的分值，其目的是为自身选择一种当前最优的行动方案；针对“与”节点，选择子节点内最低分值作为父节点的分值，这是对最坏情况的考虑。如果一个行动方案获得的倒推值比较高，那么它就可以看做是当前最优的行动方案。通常情况下，将取得极大值的一方称作 Max 方，另一方称作 Min 方，在实际博弈过程中，首先生成某深度的博弈树，然后在利用极大极小方法获得当前最优的行动方案，如图 6.9 所示，在图 6.9 中，□表示 MAX，○表示 MIN，节点上的数字表示它对应的估价函数值。在 MIN 处用圆弧连接，用 0 表示其子节点取估值最小的格局[1][2] [3][4]。

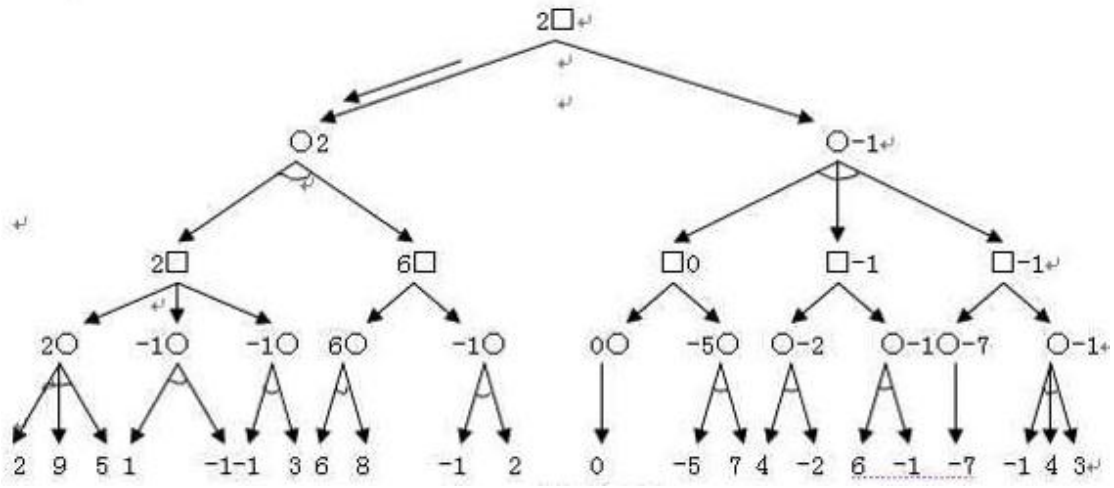


图 6.9　极大极小值方法示例

6.5.3 α - β剪枝技术

极大极小值方法在执行时把搜索树的生成和估值这两个过程完全分开，只有在已经生成博弈树之后才开始估值计算，这一分离导致

了搜索策略的效率低下。α - β剪枝技术对极大极小方法进行了改进，是一种提高博弈树搜索效率的方法，该技术在生成博弈树节点时，计算估值和倒推值，同时及时停止扩展那些已无必要再扩展的子节点，剪去某些分枝，从而节约了机器开销，提高了搜索效率。

在α - β剪枝技术中，α和β值的定义和计算如下：

（1）β值计算：在博弈树中，一个“与”节点取它当前子节点中的最小倒推值作为它的倒推值上界，我们称该值为β值。也就是说，β值等于其后继节点当前最小的最终倒推值。“与”节点的β值是永远不会增加的。

（2）α值计算：在博弈树中，一个“或”节点取它当前子节点中的最大倒推值作为它的倒推值下界，我们称该值为α值。也就是说，α值等于其后继节点当前最大的最终倒推值。“或”节点的α值是永远不会减少的。

α - β剪枝技术的基本思想：一边生成博弈树，一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。

具体的剪枝方法如下：

(1) 对于一个与节点 MIN，若能估计出其倒推值的上确界 β，并且这个β值不大于 MIN 的父节点(一定是或节点)的估计倒推值的下确界α，即α≥β，则就不必再扩展该 MIN 节点的其余子节点(因为这些节点的估值对 MIN 父节点的倒推值没有任何影响)，这一过程称为α剪枝。

(2) 对于一个或节点 MAX，若能估计出其倒推值的下确界 α，并且这个α值不小于 MAX 的父节点(一定是与节点)的估计倒推值的上确界β，即α≥β，则就不必再扩展该 MAX 节点的其余子节点(因为这些节点的估值对 MAX 父节点的倒推值没有任何影响)，这一过程称为β剪枝[1][2] [3][4]。

注意：在进行α - β剪枝时，α和 β值必须以某个节点的静态估，值为依据，至少必须使某一部分的搜索树生长到最大深度，所以该剪枝过程都要使用某种深度优先的搜索方法来实现，具体示例如图 6.10-6.11 所示。

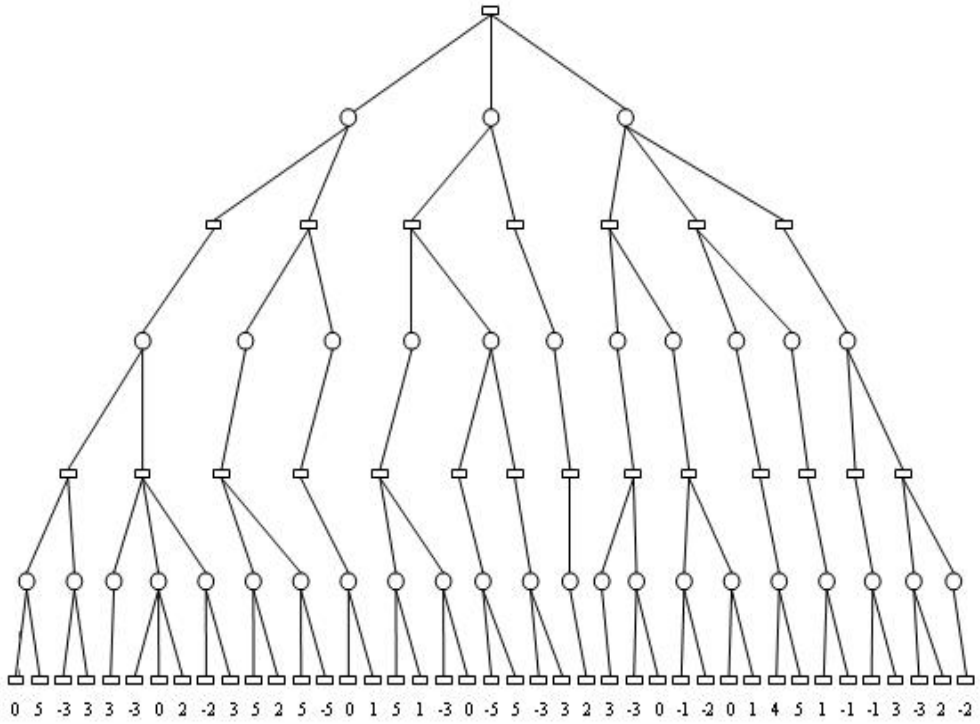


图 6.10 α - β剪枝前博弈树初始状态示例

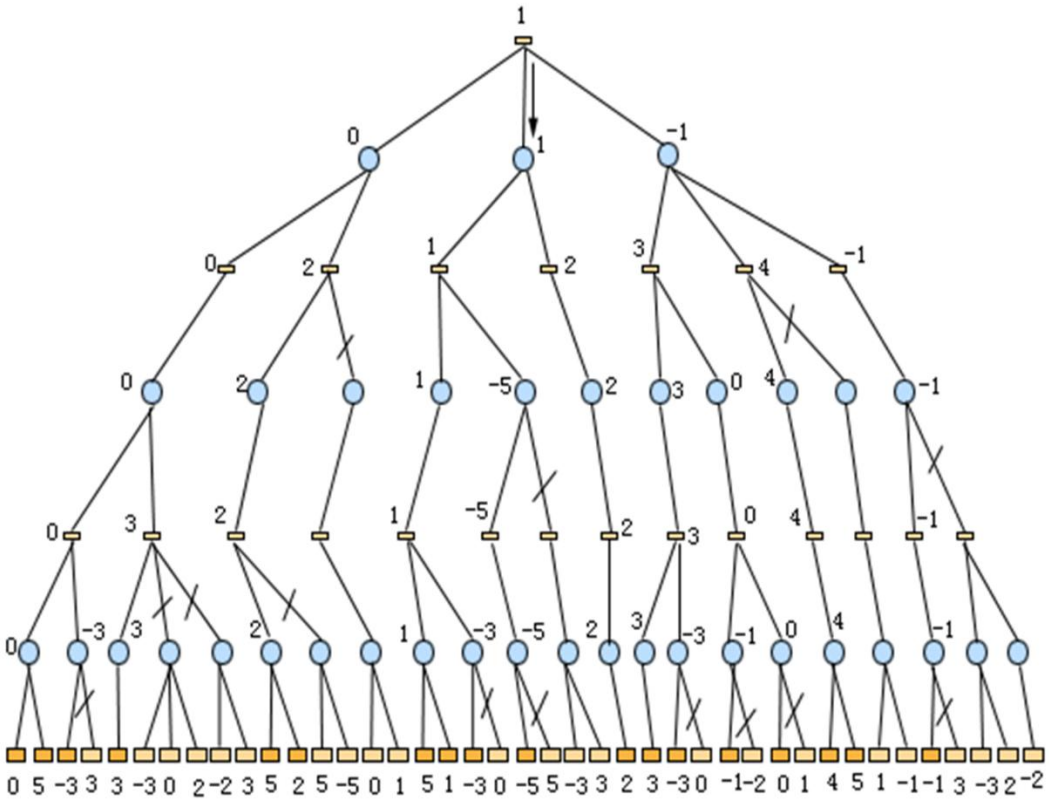


图 6.11　α - β剪枝后博弈树最终状态示例

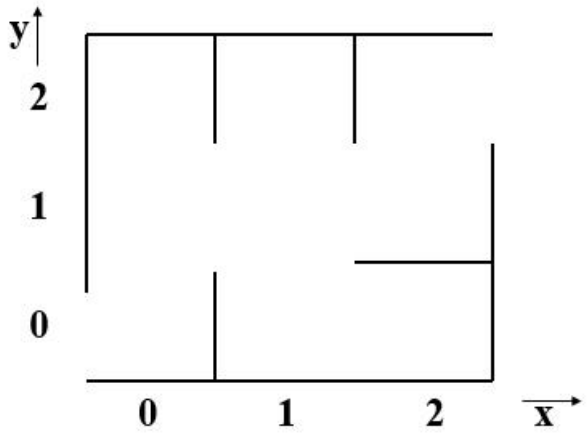
6.6 本章小结

搜索技术在人工智能中起着重要作用，搜索方法的好坏直接关系到智能系统的性能与效率，这也成为人工智能领域的核心问题之一。本章较详细地介绍了搜索的概念、搜索的评价指标以及常用的搜索方法（包括图搜索、盲目搜索、启发式搜索和博弈搜索等内容）。读者通过本章的学习，应该对人工智能中与搜索技术相关的重要概念、常用搜索方法的原理和步骤有更深入的理解和认识，并在今后人工智能相关领域的研究和学习中，能够对这些概念、原理和常用搜索方法达到学以致用的效果。

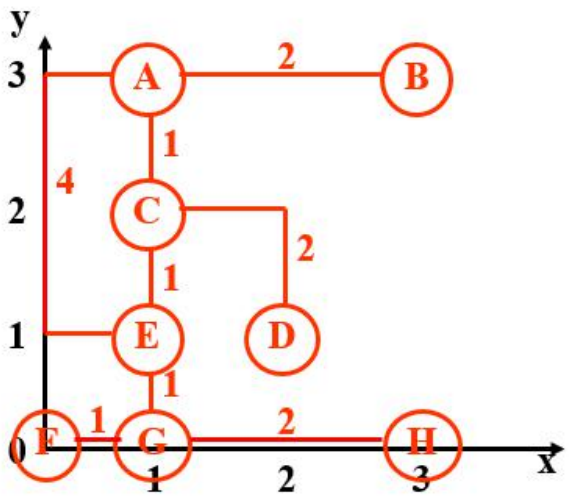
习题

1. 如下图所示的迷宫问题，分别采用横向(宽度)搜索算法、纵向（深

度）搜索算法求出从入口(0,0)到出口(2,2)的一条路径。



2. 迷宫问题如下：F 是入口，B 是出口，试采用纵向搜索算法、横向搜索算法进行求解。



3. 用 A 算法求解下列八数码魔方，启发函数 h (n)分别采用：

（1）h=0；（2）h 为放错的棋子数；（3）h 为用曼哈顿距离的和。