第3章

搜索策略

搜索是人工智能中的一个基本问题，与推理密切相关。不管是符号智能还是计算智能，不管是解决具体应用问题还是智能行为本身，最终往往都归结为某种搜索，都要用某种搜索算法去实现。因此，美国人工智能专家尼尔逊将其列为人工智能研究中的4个核心问题之一。

本章将在讨论有关搜索的基本概念的基础上，重点研究状态图搜索策略、AND/OR图搜索策略和博弈树搜索策略。

3.1 搜索策略概述

搜索的本质就是一个从初始问题出发，不断寻求最优解的过程，因此在深入学习研究搜索的各种策略之前，有必要对搜索的基本概念有个明确的认识。

3.1.1 搜索概述

人工智能所研究的对象是多种多样的，其中，大多数属于结构不良或非结构化的问题。对于这样的问题，一般很难获得其全部信息，更没有现成的算法可以直接用来求解。因此，只能依靠经验，利用已有的知识一步一步地摸索求解。在此过程中既存在着问题的表示方法问题，也存在着知识的利用问题，即如何确定推理路线，在付出极小代价的前提下，尽快求得问题的解。另外，针对一个具体的问题有可能存在多个不同的解，这就涉及如何在推理过程中寻找最优解的问题。

1. 搜索的概念

根据问题的实际情况，不断寻找可利用的知识，从而构造一条代价最小的推理路线，使问题得以解决的过程称为搜索。

2. 搜索过程中需要解决的基本问题

搜索过程中需要解决的基本问题可以概括为以下几个方面：

(1)是否一定能找到一个解。

(2)是否能够终止运行或是否会陷入一个死循环。

(3)找到的解是否是最佳解。

(4)时间与空间的复杂性如何。

其中,时间与空间的复杂性是衡量一个搜索策略正确性与可用性的重要标准。例如,对一些结构性能较好、理论上有算法可依的问题,如果问题或算法的复杂性过高(如按指数形式增长),由于受到计算机在时间及空间上的限制,也是无法付诸实施的。64 阶梵塔问题有 3^{64} 种状态,仅从计算机的空间角度上来看,也是现有计算机所不能解决的。因此,算法的选择从某种程度上来讲,既要考虑到所用算法的正确性,又要与当前的计算机发展水平相结合,只有完全符合当前计算机软硬件发展现状的算法才是有效的算法。

3. 搜索的类型

对于搜索的类型,根据在问题求解过程中是否存在启发性信息,可以分为盲目搜索和启发式搜索两种。

1)盲目搜索

盲目搜索又称非启发式盲目搜索,是指在问题的求解过程中,不运用启发性信息,只按照预定的路线进行搜索,在搜索过程中获得的中间信息也不用来改进控制策略。显然,这种搜索具有盲目性,效率不高,不便于复杂问题的求解,而且容易出现“组合爆炸”的问题。

2)启发式搜索

启发式搜索是在搜索过程中,为了提高搜索效率,加入与问题有关的启发性信息,用于指导搜索朝着最有希望获得解的方向前进,从而加速问题的求解过程并找到最优解。

3.1.2 状态空间法

状态空间法是人工智能中最基本的问题求解方法,它所采用的问题表示方法称为状态空间表示法。状态空间法的基本思想是用“状态”和“算符”来表示和求解问题。其中,“状态”用以描述问题求解过程中出现的各种状况;“算符”用来表示对状态的基本操作,算符的每一次使用就是使问题从一种状态变换为另一种状态。从初始状态到目标状态所采用的算符序列,称为问题的一个解。

1. 状态

状态是指为了描述问题求解过程中不同时刻下状况(如初始状况、事实等叙述性知识)间的差异,而引入的最少的一组变量的有序组合,常用矢量的形式表示。

$$Q=[q_0,q_1,\cdots,q_n]^{\mathrm{T}}$$

其中,$q_i(i=0,1,2,\cdots,n)$称为分量,当给每个分量一个确定的值时,就得到了一个具体的状态。

2. 算符

算符也称操作符,表示引起状态中某些分量发生变化的一组关系或函数。算符用于

反映过程性知识。例如，产生式系统中的一条规则就是一个算符。

$$F=\{f_1,f_2,\cdots,f_m\}$$

3. 状态空间

状态空间是利用状态变量和算符表示系统或问题的有关知识的符号体系。状态空间常用一个三元组来表示：

$$(S,O,G)$$

其中，S 为问题的所有初始状态的集合；O 为所有操作算符的集合；G 为目标状态的集合，包括若干具体状态或满足某些性质的路径信息描述。这样，在状态空间表示法中，问题求解过程就转化为在图中寻找从初始状态出发到达目标状态的路径问题，也就是寻找操作序列的过程。状态空间的一个解，即是一个有限的操作算符序列。

$$S_0 \xrightarrow{O_1} S_1 \xrightarrow{O_2} S_2 \xrightarrow{O_3} \cdots \xrightarrow{O_k} G$$

其中，$O_1,O_2,\cdots,O_k$ 为状态空间的一个解，另外需要强调的是，解不一定是唯一的。

状态空间的图示形式称为状态空间图。其中，节点表示问题的状态，有向边（弧）表示算符。

【例 3-1】 重排九宫的状态空间表示法。

例 2-2 提到了重排九宫问题，下面应用状态空间法来解决该问题。

问题的状态可以用 3×3 的方格棋盘来表示，因此，状态集 S 是数字 1～8 的所有摆法；出于精简算符的目的，操作算符集可定义为空白方格的移动，共有 4 种基本的操作，即空格上移 Up、空格左移 Left、空格下移 Down、空格右移 Right。目标状态集 G 包含 S_g。

依据上述定义，可以得到如图 3-1 所示的重排九宫问题的状态空间。

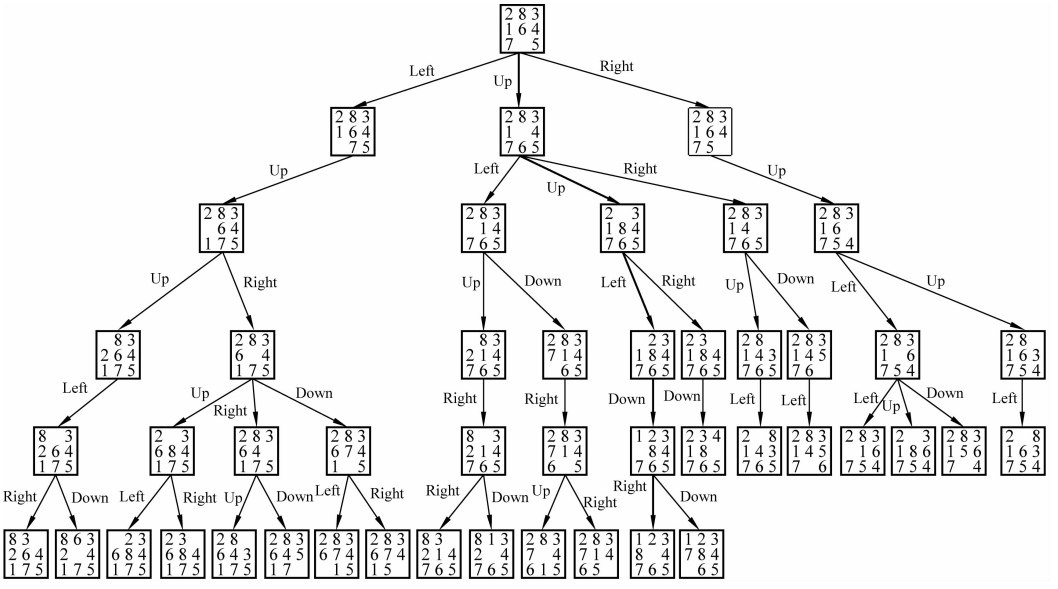

图 3-1 重排九宫问题的状态空间

从图 3-1 不难看出，由给定的初始状态 S_0 到目标状态 S_g 的最短路径为 5，解路径可以用如下的操作算符序列表示：

{Up，Up，Left，Down，Right}

【例 3-2】 猴子摘香蕉问题。

这一问题如图 3-2 所示，设房间里有一只猴子(机器人)位于 a 处，在 c 处上方的天花板上有一串香蕉，猴子想吃但摘不到，房间的 b 处还有一个箱子，如果猴子站到箱子上，就可以摸着天花板。请用状态空间表示法解决该问题。

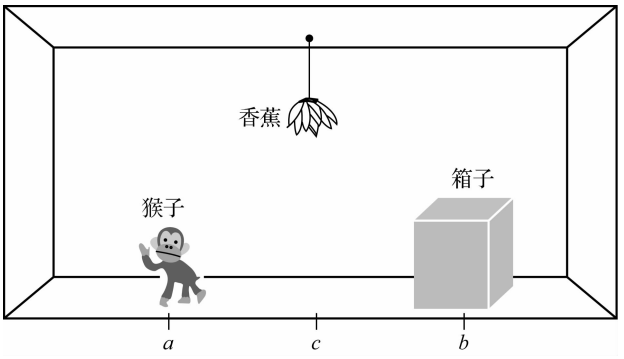

图 3-2 猴子摘香蕉问题图

针对该问题，其状态描述可以用一个四元组表示如下：

$$(w,x,y,z)$$

其中，w 表示猴子的水平位置；x 表示箱子的水平位置；y 表示猴子是否站到箱子上，当猴子站到箱子上时，y 取 1，否则取 0；z 表示猴子是否拿到香蕉，当猴子拿到香蕉时，z 取 1，否则取 0。

所有可能的状态为：

S_0：$(a,b,0,0)$　　初始状态

S_1：$(b,b,0,0)$

S_2：$(c,c,0,0)$

S_3：$(c,c,1,0)$

S_4：$(c,c,1,1)$　　目标状态

定义允许的操作为

Goto(u)：猴子走到位置 u，即

$$(w,x,0,0)\rightarrow(u,x,0,0)$$

Pushbox(v)：猴子推着箱子到水平位置 v，即

$$(x,x,0,0)\rightarrow(v,v,0,0)$$

Climbbox：猴子爬到箱子上，即

$$(x,x,0,0)\rightarrow(x,x,1,0)$$

Grasp：猴子拿到香蕉，即

$$(c,c,1,0)\rightarrow(c,c,1,1)$$

依据上述定义，可以得到如图 3-3 所示的状态空间。由图可知，从初始状态到目标状态的操作算符序列为

$$\{\text{Goto}(b),\text{Pushbox}(c),\text{Climbbox},\text{Grasp}\}$$

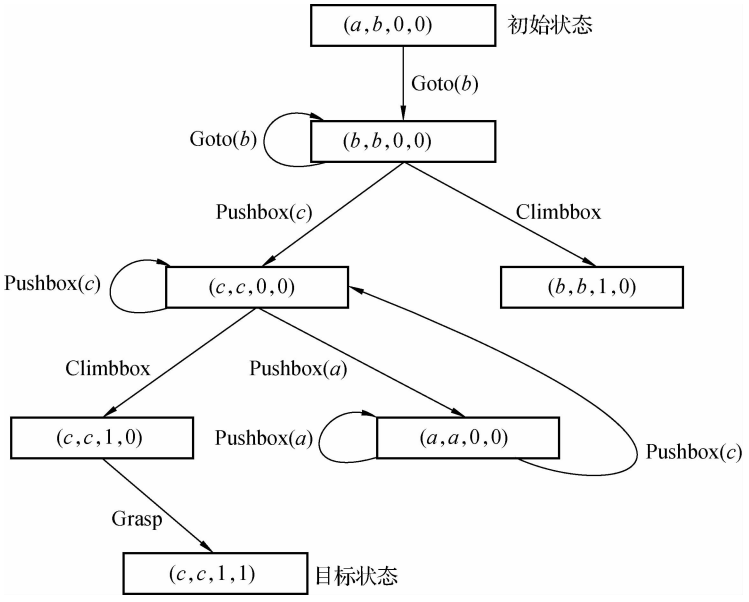

图 3-3　猴子摘香蕉问题的状态空间

3.1.3　问题归约法

1. 问题归约法的求解思路

问题归约法是不同于状态空间法的另一种形式化方法，通常用于表示比较复杂的问题，其基本思想是对问题进行分解或变换。当一个问题比较复杂时，通过一系列分解或变换把此问题变换为一个子问题集合，通过这些子问题的解可以直接得到本原问题，从而解决了原始问题。问题归约法的求解思路如图 3-4 所示。

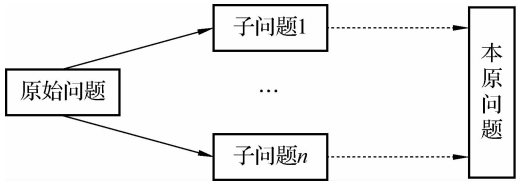

图 3-4　问题归约法的求解思路

把一个问题分解为一组子问题的集合，可以采用以下两种方法：

(1)分解。如果一个问题 P 可以归约为一组子问题 $P_1,P_2,\cdots,P_n$，并且只有当所有子问题都有解时，问题 P 才有解，任何一个子问题无解，都会导致问题 P 无解，则称此种归约为问题的分解，即分解所得到的子问题的“与”和问题等价。把一个问题分解为一组

子问题可用一个“与树”来表示。图 3-5 所示为问题 P 和与其对应的分解出来的 3 个子问题 P_1，P_2，P_3 之间的关系。其中，弧线表示节点 P_1，P_2，P_3 之间是“与”关系，节点 P 称为“与”节点。

(2)等价变换。如果一个问题 P 可以归约为一组子问题 P_1，P_2，…，P_n，并且这些子问题中只要有一个有解，则问题 P 就有解，只有当所有子问题均无解时，问题 P 才无解，则称这种归约为问题的等价变换，即等价变换所得到的子问题的“或”和问题等价。把一个问题等价变换为一组子问题可用一个“或树”来表示。图 3-6 所示为问题 P 和由它等价变换来的 3 个子问题 P_1，P_2，P_3 之间的关系。其中，节点 P_1，P_2，P_3 之间是“或”关系，节点 P 称为“或”节点。

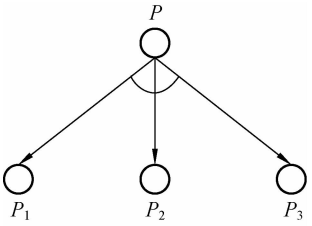

图 3-5　与树

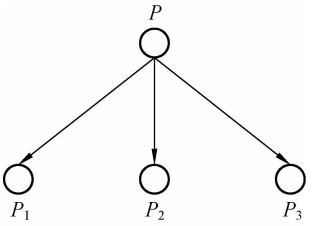

图 3-6　或树

在实际应用中，有可能会同时用到分解和等价变换两种方法，因此经常需要将“与树”与“或树”结合起来，称之为“与/或树”。图 3-7 显示了一个“与/或树”。

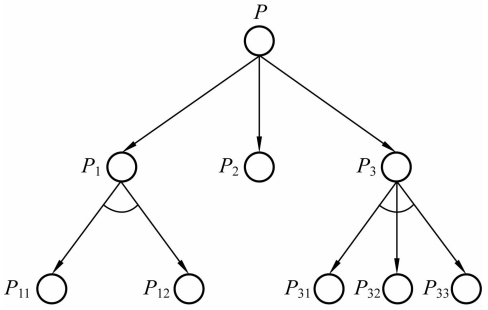

图 3-7　与/或树

与/或树中，没有子节点的节点称为端节点，本原问题所对应的节点称为终止节点。终止节点一定是端节点，但端节点不一定是终止节点。

2. 问题归约法的相关概念

为方便求解问题，问题归约法中引入以下几个基本的概念：

(1)可解节点与不可解节点。在与/或树中，满足下列条件之一的节点称为可解节点：

①任何终止节点都是可解节点。

②对“与”节点，只有当其子节点全部为可解节点时，该“与”节点才是可解节点。

③对“或”节点，当其子节点中至少有一个为可解节点时，该“或”节点就是可解节点。

不满足可解节点的 3 个条件的节点，称为不可解节点。

(2)解树。由可解节点构成，并且由这些可解节点能够推出初始节点为可解节点的子树，称为解树。解树中一定包含初始节点，它对应于原始问题。图 3-8 所示为一个与/或树的解树(图中用粗线标示)。

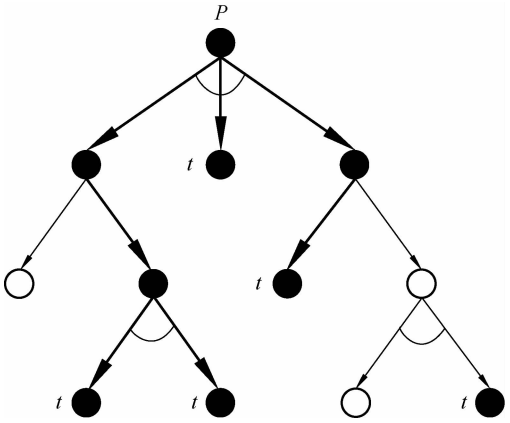

图 3-8　与/或树的解树

其中，P 表示原始问题，用 t 标出的节点代表终止节点。根据可解节点的定义，可知图 3-8 所示的与/或树对应的问题 P 为可解节点。

3. 问题归约法举例

问题归约的实质就是从目标(要解决的问题)出发，逆向推理，建立子问题及子问题的子问题，直到最后把原始问题归约为一个本原问题集合。

【例 3-3】 三阶梵塔问题。

设有 A,B,C 三个金片及 1,2,3 三根钢针，三个金片按自上而下、从小到大的顺序穿在 1 号钢针上，要求把它们全部移到 3 号钢针上，而且每次只能移动一个金片，任何时刻都不能把大的金片压在小的金片上面，如图 3-9 所示。

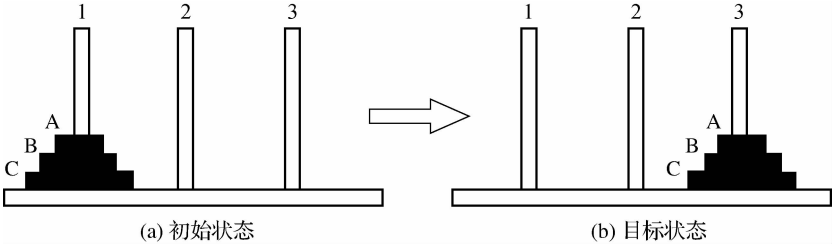

图 3-9　例 3-3 用图

这个问题既可以用状态空间法来求解，也可以用问题归约法来求解。如果用问题归约法来求解，则首先需要定义该问题的形式化表示方法。设用三元组

$$(i,j,k)$$

表示问题在任意一时刻的状态，用“→”表示状态的转换。在上述三元组中，i 代表金片 C

所在的钢针号，j 代表金片 B 所在的钢针号，k 代表金片 A 所在的钢针号。

利用问题归约法，原始问题可分解为以下 3 个子问题：

(1)把金片 A 和 B 移到 2 号钢针上的双金片移动问题，即

$$(1,1,1)\rightarrow(1,2,2)$$

(2)把金片 C 移到 3 号钢针上的单金片移动问题，即

$$(1,2,2)\rightarrow(3,2,2)$$

(3)把金片 A 和 B 移到 3 号钢针上的双金片移动问题，即

$$(3,2,2)\rightarrow(3,3,3)$$

其中，子问题(1)和子问题(3)都是一个二阶梵塔问题，它们都还可以继续进行分解；子问题(2)是一个本原问题，它已不需要再分解。

三阶梵塔问题的分解过程可用图 3-10 所示的与/或树来表示。

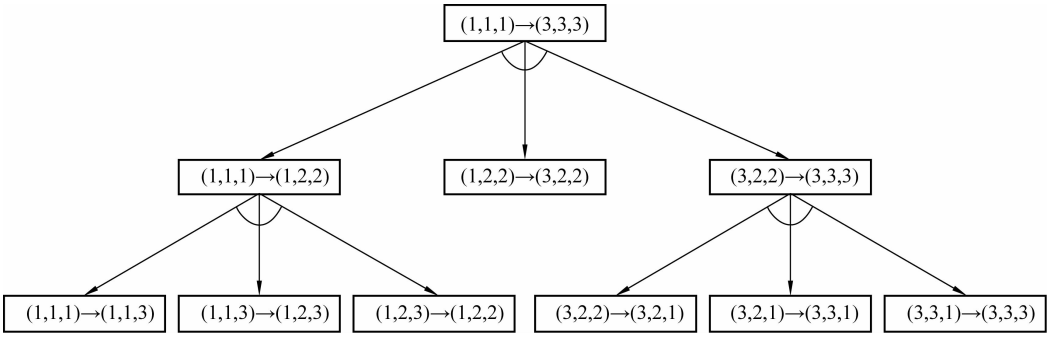

图 3-10　三阶梵塔问题的与/或树

在该与/或树中，有 7 个终止节点，它们分别对应着 7 个本原问题。如果把这些本原问题从左至右排列起来，即得到原始问题的解：

$(1,1,1)\rightarrow(1,1,3)$　$(1,1,3)\rightarrow(1,2,3)$　$(1,2,3)\rightarrow(1,2,2)$

$(1,2,2)\rightarrow(3,2,2)$　$(3,2,2)\rightarrow(3,2,1)$　$(3,2,1)\rightarrow(3,3,1)$

$(3,3,1)\rightarrow(3,3,3)$

3.2　状态图搜索策略

状态图搜索就是在状态图中寻找解路径的过程，即从初始节点出发，沿着与之相连的边试探地前进，寻找目标节点的过程，当目标节点找到后，路径也就确定了。

用计算机来实现状态图的搜索有两种最基本的方式：树式搜索和线式搜索。树式搜索就是以构造树的方式进行的搜索，即从树根(初始节点)出发，依次形成树的各层节点，搜索过程中要记录下所经过的所有节点和边。其记录的轨迹即是一棵树，这棵树也就是搜索过程中所产生的搜索树。线式搜索是指在搜索过程中只记录那些当前认为是处在所找路径上的节点和边。所以，线式搜索所记录的轨迹始终是一条线。

由此可见，树式搜索成功后，还需要再从搜索树中找出所求路径，而线式搜索只要搜

索成功，则搜索线即是所要寻找的路径，也就是问题的解。

3.2.1 状态图搜索的一般过程

状态图搜索的一般过程最早由尼尔逊提出，它是表达能力很强的一个搜索策略框架。由于搜索的目的是寻找初始节点到目标节点的路径，所以在搜索过程中就需要随时记录搜索的轨迹，这里引入 OPEN 表和 CLOSED 表两个动态数据结构。其中，OPEN 表用于存放刚生成的节点，这些节点将作为以后待考察的节点。节点进入 OPEN 表的顺序不同，决定了不同的搜索策略。CLOSED 表用来记录已经生成，并且已经考察过的节点。显然，对于树式搜索来说，CLOSED 表中存储的正是一棵不断成长的搜索树；而对于线性搜索来说，CLOSED 表中存储的是一条不断延长的折线，它可能本身就是所求的路径（如果能够找到目标节点的话）。

OPEN 表和 CLOSED 表的结构如图 3-11 所示。

节点	父节点编号

(a) OPEN 表

编号	节点	父节点编号

(b) CLOSED 表

图 3-11 OPEN 表与 CLOSED 表的结构

图搜索（GRAPHSEARCH）的一般过程如下：

（1）建立一个仅由初始节点 S_0 构成的搜索图 G，把 S_0 放到 OPEN 表中。

（2）建立 CLOSED 表，其初始状态为空表。

（3）LOOP。如果 OPEN 表是空表，则问题无解，以失败退出。

（4）在 OPEN 表中选出第一个节点，并将其删除，然后把它放到 CLOSED 表中，并标记此节点为节点 n。

（5）如果 n 是一个目标节点，则过程成功结束，解路径可通过追溯 G 中从 n 到 S_0 的指针获得。这些指针在步骤（7）中建立。

（6）扩展节点 n，同时生成不是 n 的祖先的后继节点的集合 M。把 M 的这些成员作为 n 的后继节点加入搜索图 G 中。

（7）对于 M 中那些既不在 OPEN 表中也不在 CLOSED 表中的成员设置一个通向 n 的指针，并把这些成员加到 OPEN 表中。对于 M 中那些已经在 OPEN 表中或 CLOSED 表中的成员，确定是否要调整它们的指针。对于 M 中那些在 CLOSED 表中的成员，确定是否需要调整它们在 G 中后裔节点的指针方向。

（8）按某一任意方式或根据探索信息重排 OPEN 表中的节点。

（9）转 LOOP。

该过程是最普通的一个搜索过程，可以包括各种各样特殊的图搜索算法。这个过程产生一个显式的图 G，称为搜索图，同时又产生 G 的一个支撑树 T，叫作搜索树。搜索树

用步骤(7)中建立的指针来确定。对于G中的每一个节点n,由搜索树T确定的由S_0到n的有向路的费用是所有从S_0到n的有向路的费用中最小的。

3.2.2 广度优先搜索

如果在3.2.1节GRAPHSEARCH过程的步骤(8)中对OPEN表中节点的排序不使用关于问题的搜索性信息,则必须任意规定一种排序方式,所得到的搜索过程叫作无信息的盲目搜索。下面介绍的广度优先搜索和深度优先搜索均属于无信息的图搜索过程。

广度优先搜索也称为宽度优先搜索,它采用的搜索策略是先生成的节点先扩展的思想。其搜索过程是:从初始节点S_0开始逐层向下扩展,在第n层节点还没有完全搜索完成之前,不进入第$n+1$层节点的搜索,即先按生成规则生成第一层节点,在该层全部节点中进行广度扫描,检查是否有目标节点出现。如果没有,则将所有第一层节点逐一进行扩展的,得到第二层节点,再检查是否有目标节点出现,如此反复,直到发现目标节点为止。

在广度优先搜索策略下,OPEN表中的节点总是按照进入的先后顺序进行扩展的,先进入OPEN表的节点排在前面,后进入OPEN表的节点排在后面。

广度优先搜索的过程如下:

(1)把初始节点S_0放入OPEN表。

(2)如果OPEN表为空,则搜索失败,退出。

(3)将OPEN表中的第一个节点n移出,并放入CLOSED表。

(4)如果节点n是目标节点,则求得问题的解,退出。

(5)如果节点n不可扩展,则转步骤(2)。

(6)如果节点n可扩展,则扩展节点n,将其所有子节点放入OPEN表的尾部,并为每个子节点配置指向父节点n的指针,然后转步骤(2)。

广度优先搜索的流程如图3-12所示。

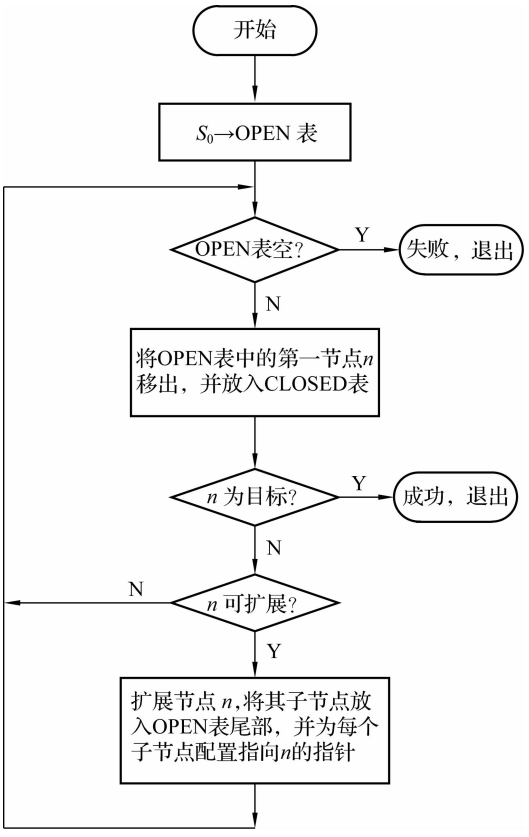

图3-12 广度优先搜索的流程

广度优先搜索是一种完备性搜索策略,即只要问题有解,它就一定可以找到解,并且广度优先搜索找到的解,也一定是最优解。这些都是广度优先搜索的优点。其主要缺点是盲目性较强,尤其是当目标节点距离初始节点比较远时,将产生许多无用的节点,因此广度优先搜索的效率比

较低。

【例 3-4】 用广度优先搜索策略解决重排九宫问题。

初始状态为 S_0，目标状态为 S_g，如图 3-13 所示。

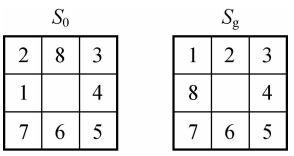

图 3-13 例 3-4 用图

可以使用的操作有 4 种：空格左移、空格上移、空格右移和空格下移。

依据广度优先搜索策略，可以在第四级得到问题的解，解路径为：S_0→3→8→16→26。其搜索树如图 3-14 所示。

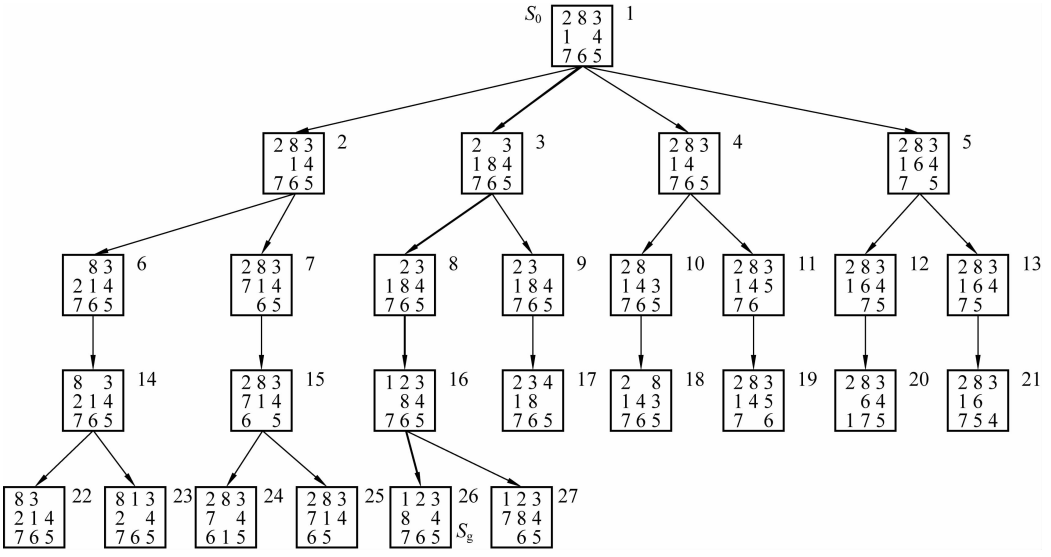

图 3-14 重排九宫问题的广度优先搜索树

3.2.3 深度优先搜索

与广度优先搜索对应的另外一种盲目搜索叫作深度优先搜索。深度优先搜索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，沿另一方向继续前进。其核心思想是优先扩展最新生成的（最深的）节点到表中，深度相等的节点可以任意排列。

深度优先搜索的过程如下：

（1）把初始节点 S_0 放入 OPEN 表。

（2）如果 OPEN 表为空，则搜索失败，问题无解，退出。

（3）将 OPEN 表中的第一个节点 n 移出 OPEN 表，并放入 CLOSED 表。

（4）如果节点 n 是目标节点，则求得问题的解，成功退出。

(5)如果节点 n 不可扩展,则转步骤(2)。

(6)如果节点 n 可扩展,则扩展节点 n,将其所有子节点放入 OPEN 表的首部,并为每一个子节点配置指向父节点 n 的指针,然后转步骤(2)。

通过对广度优先搜索过程和深度优先搜索过程的比较不难看出,两种搜索策略的区别仅在于步骤(6),即扩展节点 n 后,其子节点进入 OPEN 表的位置不同。广度优先搜索是将节点 n 的子节点放入 OPEN 表的尾部,而深度优先搜索是把节点 n 的子节点放入 OPEN 表的首部。

【例 3-5】 对于例 3-4 提出的重排九宫问题,应用深度优先搜索策略,可以得到如图 3-15 所示的搜索树。由于问题的目标尚未达到,因此该搜索还需要继续进行下去。

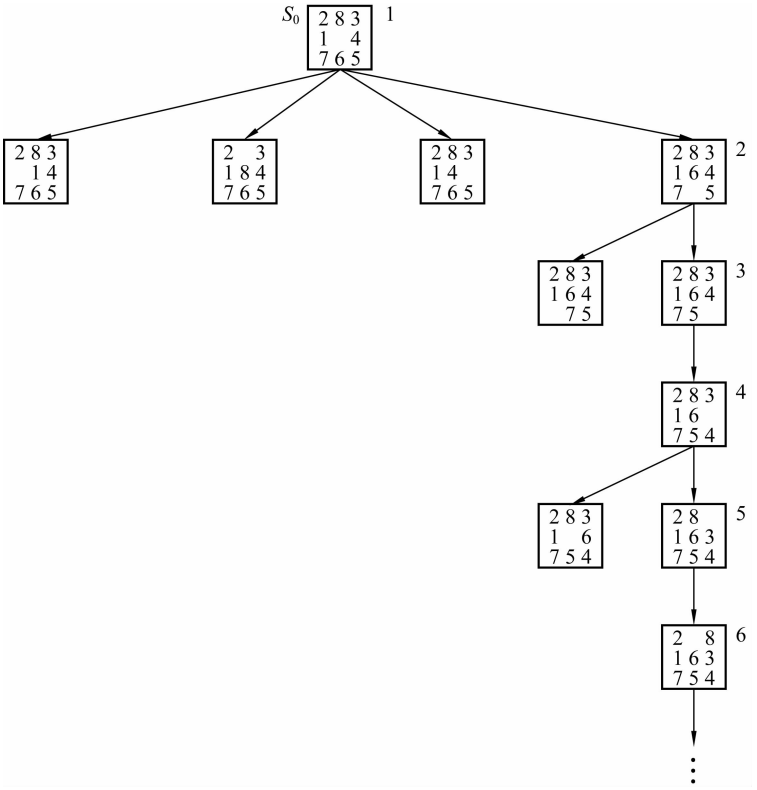

图 3-15 重排九宫问题的深度优先搜索树(部分)

从深度优先搜索的算法可以看出,搜索一旦进入某个分支,就将沿着这个分支一直进行下去,如果目标恰好在这个分支上,则它可以很快找到解。但是,如果目标不在这个分支上,且该分支是一个无穷分支,则搜索过程就不可能找到解。因此,深度优先搜索是一种不完备的搜索策略,即使问题有解,也不一定能够找到解。此外,即使在能找到解的情况下,按深度优先搜索找到的解也不一定是路径最短的解。深度优先搜索的最大优点是能够节省大量的时间和空间。

3.2.4 有界深度优先搜索

为了弥补广度优先搜索和深度优先搜索各自的不足，经常采用的是一种折中的办法，在深度优先搜索策略中引入深度限制，即采用有界深度优先搜索策略。有界深度优先搜索过程总体上按深度优先策略进行，但对搜索深度需要给出一个深度限制 d_m，当搜索深度达到 d_m，但还没有找到目标时，就停止该分支的搜索，换到另外一个分支继续进行搜索。

有界深度优先搜索的过程如下：

(1)把初始节点 S_0 放入 OPEN 表，设置 S_0 的深度 $d(S_0)=0$。

(2)如果 OPEN 表为空，则搜索失败，问题无解，退出。

(3)将 OPEN 表中的第一个节点 n 移出 OPEN 表，并放入 CLOSED 表。

(4)如果节点 n 是目标节点，则求得问题的解，成功退出。

(5)如果 n 的深度 $d(n)=d_m$(深度限制值)，或者 n 不可扩展，则转向步骤(2)。

(6)如果节点 n 可扩展，则扩展节点 n，将其所有子节点放入 OPEN 表的首部，并为每一个子节点配置指向父节点 n 的指针，然后转向步骤(2)。

【例 3-6】 设深度限制 $d_m=4$，用有界深度优先搜索策略求解例 3-4 提出的重排九宫问题。依据上述搜索过程，可以得到如图 3-16 所示的搜索树。其解路径为：S_0→20→25→26→28。

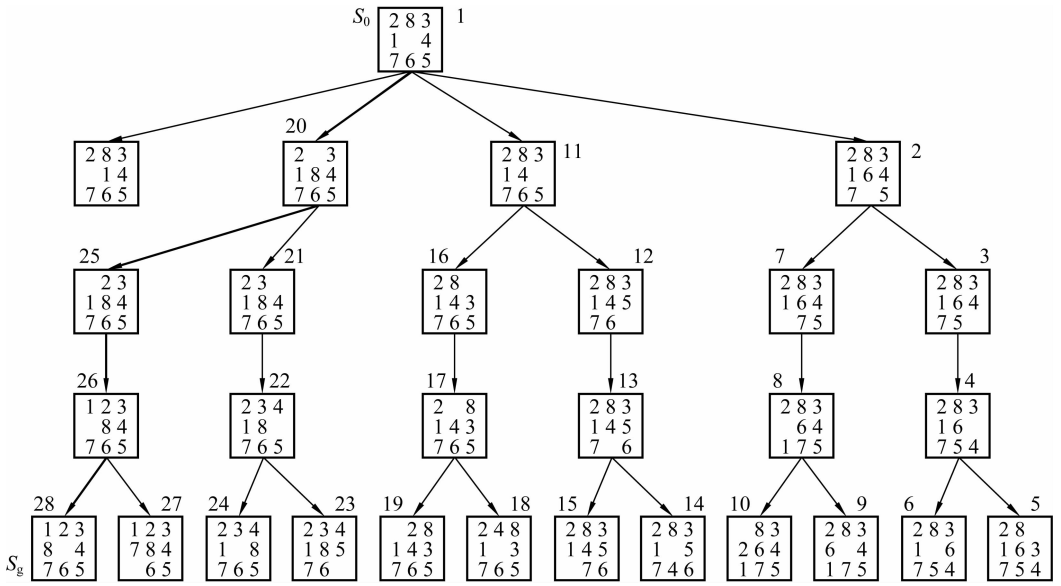

图 3-16 重排九宫问题的有界深度优先搜索树

3.2.5 启发式搜索

前面介绍的无信息搜索方式，从理论上讲，似乎可以解决任何状态空间的搜索问题，

但实践证明，该种方式需要产生大量的节点才能找到问题的解路径。而这些节点都需要在内存中保存起来，由此可见无信息搜索方式所需的存储空间是相当大的。例如，博弈问题，计算机为了取胜，需要将所有算法都试一遍，然后选择最佳走法。虽说找到这样的算法并不难，但计算机的时空消耗却大得惊人。例如，就可能有的棋局数而言，西洋棋是10^{78}，国际象棋是10^{120}，围棋是10^{761}，等等。而实际上计算机所能提供的资源却是极其有限的，因此必须寻找效率更高的搜索方法。

1. 启发性信息

对于某些问题，可以使用与问题有关的信息帮助减少搜索量，这样的信息称为启发性信息。使用启发性信息指导的搜索过程叫作启发式搜索。其中，启发性信息用在GRAPHSEARCH算法的步骤(8)中，即对OPEN表中的节点进行排序，把最有希望的节点排在最前面，使搜索图沿着有利于获得解的方向进行扩展。

就用途而言，启发性信息一般分为以下3类：

(1)用于扩展节点的选择，即用于决定应先扩展哪一个节点，以避免对于节点的盲目扩展。

(2)用于生成节点的选择，即用于决定应生成哪些后续节点，以避免盲目地生成过多无用节点。

(3)用于删除节点的选择，即用于决定应删除哪些无用节点，以避免造成进一步的时空浪费。

2. 估价函数

使用启发性信息的一种重要方法是采用估价函数，这是一个定义在所有状态描述上的实值函数。它定义了为从初始节点经过节点n到达目标节点的最小代价路径的代价估计值。在实际应用中，可以以各种不同的想法建立估价函数，如采用任意节点与目标节点的距离度量，棋盘上对局势的度量，一个节点处在最佳路径上的概率，等等。

估价函数综合考虑了两个方面的因素，即已付出的代价和将要付出的代价。其一般定义形式为

$$f(n)=g(n)+h(n)$$

其中，$f(n)$表示从初始节点经过节点n到达目标节点的总代价，称为估价函数。它的作用是对OPEN表中的节点进行评估，根据它们对于解的重要程度来决定它们在OPEN表中的先后次序。$g(n)$表示从初始节点到达节点n已经实际付出的代价。$h(n)$是从节点n到目标节点将要付出的估计代价，称为启发函数，它体现了问题的启发性信息。

例如，对于重排九宫问题，可以定义如下的估价函数：

$$f(n)=d(n)+w(n)$$

其中，$d(n)$表示节点n在搜索树中的深度，$w(n)$是与n对应的状态描述中偏离目标的数字个数。对于例3-6给出的初始状态描述，应用定义的上述估价函数，可以求得$f(S_0)=0+3=3$。

使用估价函数求解重排九宫问题的搜索树如图3-17所示，圆圈中的数字是估价函数

值，节点上方的数字表示节点的扩展次序。由图 3-17 不难看出，使用估价函数得到的解路径与无信息搜索得到的解路径是相同的，但搜索过程所产生的节点数却要少得多。

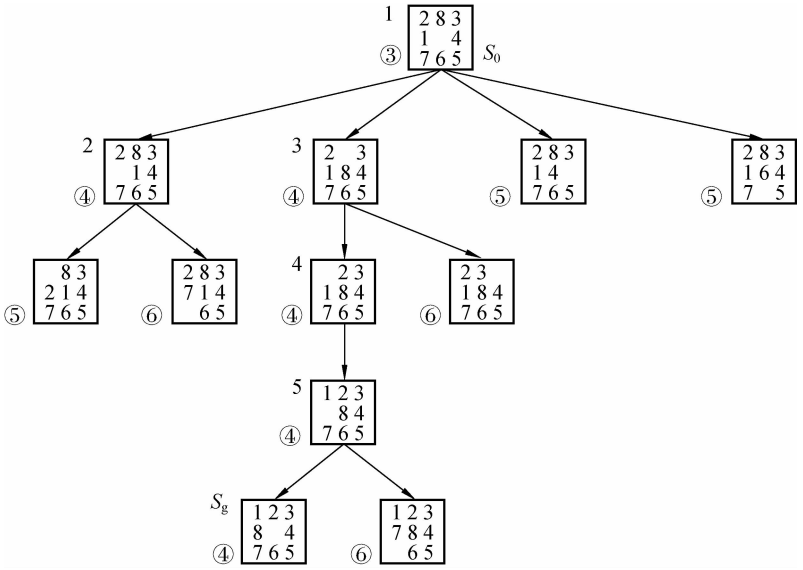

图 3-17　使用估价函数求解重排九宫问题的搜索树

3. A 算法

启发式搜索要用启发函数来导航，其搜索算法就要在状态图一般搜索算法的基础上再增加启发函数值的计算与传播过程，并且由启发函数值来确定节点的扩展顺序。在图搜索算法中，如果能在搜索的每一步都利用估价函数 $f(n)=g(n)+h(n)$ 来对 OPEN 表中的节点进行排序，则该搜索算法称为 A 算法。另外，由于估价函数中带有问题自身的启发性信息，因此，A 算法也被称为启发式搜索算法。

根据搜索过程中选择的扩展节点的范围，启发式搜索算法可以分为局部择优搜索算法和全局择优搜索算法。

1）局部择优搜索算法

对于局部择优搜索算法，每当需要扩展节点时，总是从刚生成的子节点中选择一个估价函数值最小的节点进行扩展。

局部择优搜索算法流程如下：

（1）把初始节点 S_0 放入 OPEN 表，并计算 $f(S_0)$。

（2）如果 OPEN 表为空，则搜索失败，问题无解，退出。

（3）将 OPEN 表中的第一个节点 n 移出并放入 CLOSED 表。

（4）如果节点 n 是目标节点，则求得问题的解，成功退出。

（5）如果节点 n 不可扩展，则转向步骤（2）。

（6）如果节点 n 可扩展，则扩展节点 n，用估价函数 $f(n)$ 计算每个子节点的估价函数值，并按估价函数值从小到大的顺序依次放入 OPEN 表的首部，并为每个子节点配置指

向父节点 n 的指针，然后转向步骤(2)。

局部择优搜索的流程如图 3-18 所示。

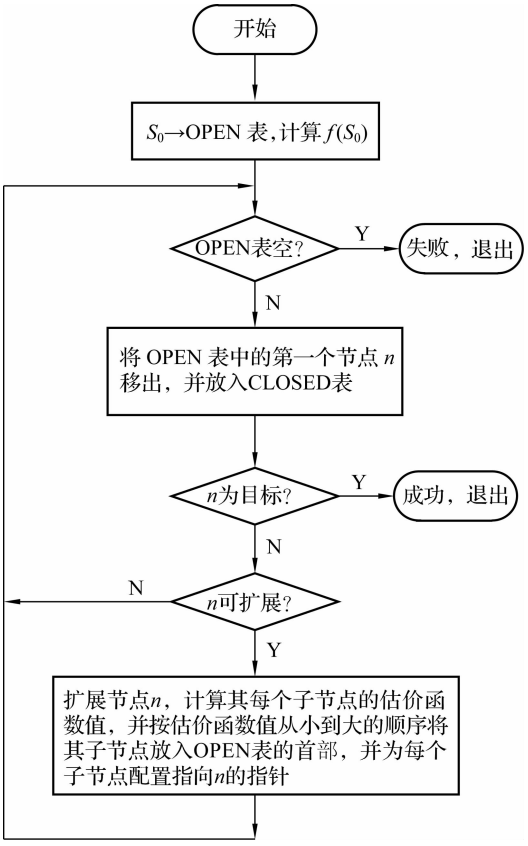

图 3-18　局部择优搜索算法流程

2)全局择优搜索算法

对于全局择优搜索算法，每当需要扩展节点时，总是从 OPEN 表的所有节点中选择一个估价函数值最小的节点进行扩展。

全局择优搜索算法流程如下：

(1)把初始节点 S_0 放入 OPEN 表，并计算 $f(S_0)$。

(2)如果 OPEN 表为空，则搜索失败，问题无解，退出。

(3)把 OPEN 表中的第一个节点 n 移出，并放入 CLOSED 表中。

(4)如果节点 n 是目标节点，则求得问题的解，成功退出。

(5)如果节点 n 不可扩展，则转步骤(2)。

(6)如果节点 n 可扩展，则扩展节点 n，生成其子节点 $n_i(i=1,2,\cdots)$，计算每个子节点的估价函数值，并为每个子节点配置指向父节点 n 的指针，然后将这些子节点放入 OPEN 表。

(7)根据各节点的估价函数值，对 OPEN 表中的全部节点按从小到大的顺序重新进

行排序，然后转向步骤(2)。

由于上述算法中的步骤(7)要对 OPEN 表中的全部节点按其估价函数值从小到大进行排序，这样从步骤(3)中取出的节点就一定是 OPEN 表中所有节点中估价函数值最小的一个节点，因此，它是一种全局择优搜索算法。

3.2.6 A* 算法

如果对上述 A 算法再限制其估价函数中的启发函数 $h(n)$ 满足：对所有的节点 n_i 均有

$$h(n)\leqslant h^*(n)$$

其中，$h^*(n)$ 是从节点 n 到目标节点的最小代价，即最佳路径上的实际代价(若有多个目标节点，则为其中最小的一个)，则它就称为 A* 算法。

在 A* 算法中限制 $h(n)\leqslant h^*(n)$ 是为了保证取得最优解。理论分析证明，如果问题存在最优解，则这样的限制可以保证找到最优解。虽然这个限制可能产生无用的搜索。实际上，当某一节点 n 的 $h(n)>h^*(n)$ 时，该节点就可能失去优先扩展的机会，导致得不到最优解。A* 算法也称为最佳图搜索算法。它由人工智能学者 Nilsson 提出。

下面来讨论 A* 算法的有关特性。

1. A* 算法的可采纳性

一般来说，对任意一个状态空间图，当从初始节点到目标节点有路径存在时，如果搜索算法能在有限步内找到一条从初始节点到目标节点的最佳路径，并在此路径上结束，则称该搜索算法是可采纳的。下面证明 A* 算法是可采纳的。

定理 3.1 对有限图，如果从初始节点 S_0 到目标节点 S_g 有路径存在，则 A* 算法一定成功结束。

证明：首先证明算法必然会结束。由于搜索图为有限图，如果算法能找到解，则会成功结束；如果算法找不到解，则必然由于 OPEN 表变空而结束。因此，A* 算法必然会结束。

然后证明算法一定会成功结束。由于至少存在一条由初始节点到目标节点的路径，因此设此路径为

$$S_0=n_0,n_1,\cdots,n_k=S_g$$

算法开始时，节点 n_0 在 OPEN 表中，而且路径中任意一节点 n_i 离开 OPEN 表后，其后继节点 n_{i+1} 必然进入 OPEN 表，这样，在 OPEN 表变空之前，目标节点必然出现在 OPEN 表中，因此，算法一定会成功结束。

引理 3.1 对无限图，如果从初始节点 S_0 到目标节点 S_g 有路径存在，且 A* 算法不终止，则从 OPEN 表中选出的节点必将具有任意大的 f 值。

证明：设 $d^*(n)$ 是 A* 算法生成的从初始节点 S_0 到节点 n 的最短路径长度，由于搜索图中每条边的代价都是一个正数，令这些正数中最小的一个数是 e，则有

$$g^*(n)\geqslant d^*(n)\cdot e$$

因为 $g^*(n)$ 是最佳路径的代价，故有

$$g(n)\geqslant g^*(n)\geqslant d^*(n)\cdot e$$

又因为 $h(n)\geqslant 0$，故有

$$f(n)=g(n)+h(n)\geqslant g(n)\geqslant d^*(n)\cdot e$$

如果 A* 算法不终止，则从 OPEN 表中选出的节点必将具有任意大的 $d^*(n)$ 值，因此，它将具有任意大的 f 值。

引理 3.2 在 A* 算法终止前的任何时刻，OPEN 表中总存在节点 n'，它是从初始节点 S_0 到目标节点的最佳路径上的一个节点，且满足 $f(n')\leqslant f^*(S_0)$。

证明：设从初始节点 S_0 到目标节点 t 的一条最佳路径序列为

$$S_0=n_0,n_1,\cdots,n_k=S_g$$

算法开始时，节点 S_0 在 OPEN 表中，当节点 S_0 离开 OPEN 表进入 CLOSED 表时，节点 n_1 进入 OPEN 表。因此，在 A* 算法没有结束以前，在 OPEN 表中必存在最佳路径上的节点。设这些节点中排在最前面的节点为 n'，则有

$$f(n')=g(n')+h(n')$$

由于 n' 在最佳路径上，故有 $g(n')=g^*(n')$，从而

$$f(n')=g^*(n')+h(n')$$

又由于 A* 算法满足 $h(n')\leqslant h^*(n')$，故有

$$f(n')\leqslant g^*(n')+h^*(n')=f^*(n')$$

因为在最佳路径上的所有节点的 f^* 值都应相等，因此有

$$f(n')\leqslant f^*(S_0)$$

定理 3.2 对无限图，若从初始节点 S_0 到目标节点 t 有路径存在，则 A* 算法必然会结束。

证明：(反证法)假设 A* 算法不结束，由引理 3.1 知，OPEN 表中的节点有任意大的 f 值，这与引理 3.2 的结论相矛盾，因此，A* 算法只能成功结束。

推论 3.1 OPEN 表中任一满足 $f(n)<f^*(S_0)$ 的节点 n，最终都被 A* 算法选中作为扩展的节点。

定理 3.3 A* 算法是可采纳的，即若存在从初始节点 S_0 到目标节点 S_g 的路径，则 A* 算法必能结束在最佳路径上。

证明：分两步证明。

(1)证明 A* 算法一定能够终止在某个目标节点上。

由定理 3.1 和定理 3.2 可知，无论是对有限图还是对无限图，A* 算法都能够找到某个目标节点而结束。

(2)证明 A* 算法只能终止在最佳路径上(反证法)。

假设 A* 算法未能终止在最佳路径上，而是终止在某个目标节点 t 处，则有

$$f(t)=g(t)>f^*(S_0)$$

但由引理 3.2 可知，在 A* 算法结束前，必有最佳路径上的一个节点 n' 在 OPEN 表中，且有

$$f(n')\leqslant f^*(S_0)<f(t)$$

这时，A* 算法一定会选择 n' 来扩展，而不可能选择 t，从而也不会去测试目标节点 t，这就与假设 A* 算法终止在目标节点 t 相矛盾。因此，A* 算法只能终止在最佳路径上。

2. A* 算法的最优性

A* 算法的搜索效率很大程度上取决于估价函数 $h(n)$。一般来说，在满足 $h(n)\leqslant h^*(n)$ 的前提下，$h(n)$ 的值越大越好。$h(n)$ 的值越大，说明它携带的启发性信息越多，A* 算法搜索时扩展的节点就越少，搜索效率就越高。A* 算法的这一特性也称为信息性。

定理 3.4 设 A_1 和 A_2 是两个 A* 算法，它们分别使用如下两个估价函数：

$$f_1(n)=g_1(n)+h_1(n)$$

$$f_2(n)=g_2(n)+h_2(n)$$

如果 A_2 比 A_1 有更多的启发性信息，即对所有非目标节点均有

$$h_2(n)>h_1(n)$$

则在搜索过程中，被 A_2 扩展的节点也必然被 A_1 扩展，即 A_1 扩展的节点不会比 A_2 扩展的节点少。

证明(用数学归纳法)：

(1)对深度 $d(n)=0$ 的节点，即 n 为初始节点 S_0，如果 n 为目标节点，则 A_1 和 A_2 都不扩展 n；如果 n 不是目标节点，则 A_1 和 A_2 都要扩展 n。

(2)假设对 A_2 搜索树中 $d(n)=k$ 的任意节点 n，结论成立，即 A_1 也扩展了这些节点。

(3)证明 A_2 搜索树中 $d(n)=k+1$ 的任意节点 n 也要由 A_1 扩展(用反证法)。

假设 A_2 搜索树上有一个满足 $d(n)=k+1$ 的节点 n，A_2 扩展了该节点，但 A_1 没有扩展它。根据第(2)条的假设，知道 A_1 扩展了节点 n 的父节点。因此，n 必定在 A_1 的 OPEN 表中。既然节点 n 没有被 A_1 扩展，则有

$$f_1(n)\geqslant f^*(S_0)$$

即

$$g_1(n)+h_1(n)\geqslant f^*(S_0)$$

但由于当 $d=k$ 时，A_2 扩展的节点，A_1 也一定扩展，故有

$$g_1(n)\leqslant g_2(n)$$

因此有

$$h_1(n)\geqslant f^*(S_0)-g_2(n)$$

又由于 A_2 扩展了 n，因此有

$$f_2(n)\leqslant f^*(S_0)$$

即

$$g_2(n)+h_2(n)\leqslant f^*(S_0)$$

即

$$h_2(n)\leqslant f^*(S_0)-g_2(n)$$

所以有

$$h_1(n)\geqslant h_2(n)$$

这与最初假设的 $h_2(n)>h_1(n)$ 矛盾，因此反证法的假设不成立。

3. $h(n)$ 的单调限制

在 A* 算法中，每当要扩展一个节点时都要先检查其子节点是否已在 OPEN 表或 CLOSED 表中，有时还需要调整指向父节点的指针，这就增加了搜索的代价。如果能够保证每扩展一个节点，就能够找到通往这个节点的最佳路径，则没有必要再去检查其后继节点是否已在 CLOSED 表中。为满足这一要求，需要对启发函数 $h(n)$ 增加单调性限制。

定义 3.1 如果启发函数满足以下两个条件：

(1) $h(S_g)=0$。

(2) 对任意节点 n_i 及其任一子节点 n_j，都有

$$0\leqslant h(n_i)-h(n_j)\leqslant c(n_i,n_j)$$

其中，$c(n_i,n_j)$ 是节点 n_i 到其子节点 n_j 的边代价，则称 $h(n)$ 满足单调限制。

上式也可以写成

$$h(n_i)\leqslant c(n_i,n_j)+h(n_j)$$

它说明从节点 n_i 到目标节点最小代价的估值不会超过从节点 n_i 到其子节点 n_j 的边代价加上从 n_j 到目标节点的最小代价估值。

定理 3.5 如果 h 满足单调条件，则当 A* 算法扩展节点 n 时，该节点就已经找到了通往它的最佳路径，即 $g(n)=g^*(n)$。

证明：设 A* 正要扩展节点 n，而节点序列

$$S_0=n_0,n_1,\cdots,n_k=n$$

是由初始节点 S_0 到节点 n 的最佳路径。其中，n_i 是这个序列中最后一个位于 CLOSED 表中的节点，则上述节点序列中的 n_{i+1} 节点必定在 OPEN 表中，故有

$$g^*(n_i)+h(n_i)\leqslant g^*(n_i)+c(n_i,n_{i+1})+h(n_{i+1})$$

由于节点 n_i 和 n_{i+1} 都在最佳路径上，故有

$$g^*(n_{i+1})=g^*(n_i)+c(n_i,n_{i+1})$$

所以

$$g^*(n_i)+h(n_i)\leqslant g^*(n_{i+1})+h(n_{i+1})$$

一直推导下去可得

$$g^*(n_{i+1})+h(n_{i+1})\leqslant g^*(n_k)+h(n_k)$$

由于节点在最佳路径上，故有

$$f(n_{i+1})\leqslant g^*(n)+h(n)$$

因为这时 A* 扩展节点 n，而不扩展节点 n_{i+1}，则有

$$f(n)=g(n)+h(n)\leqslant f(n_{i+1})\leqslant g^*(n)+h(n)$$

即

$$g(n)\leqslant g^*(n)$$

但是，$g^*(n)$ 是最小代价值，应当有

$$g(n)\geqslant g^*(n)$$

所以有

$$g(n)=g^*(n)$$

定理 3.6 如果 $h(n)$满足单调性限制，则 A* 算法扩展的节点序列的 f 值是非递减的，即 $f(n_i)\leqslant f(n_{i+1})$。

证明：假设节点 n_{i+1}在节点 n_i 之后立即扩展，由单调限制条件可知

$$h(n_i)-h(n_{i+1})\leqslant c(n_i,n_{i+1})$$

即

$$f(n_i)-g(n_i)-f(n_{i+1})+g(n_{i+1})\leqslant c(n_i,n_{i+1})$$

亦即

$$f(n_i)-g(n_i)-f(n_{i+1})+g(n_i)+c(n_i,n_{i+1})\leqslant c(n_i,n_{i+1})$$

所以

$$f(n_i)-f(n_{i+1})\leqslant 0$$

即

$$f(n_i)\leqslant f(n_{i+1})$$

【例 3-7】 利用 A* 算法解决修道士和野人问题。

设在河的左岸有 3 个野人、3 个修道士和 1 条船，修道士想用这条船把所有的野人运到河对岸，但受到以下条件约束：

(1)修道士和野人都会划船，但每次船上至多可载两个人。

(2)在河的任一岸，如果野人数目超过修道士数目，修道士就会被野人吃掉。

(3)如果野人会服从任何一次过河安排，请规划一个确保修道士和野人都能过河，且没有修道士被野人吃掉的安全过河计划。

设 m 表示左岸的修道士人数，c 表示左岸的野人数，b 表示左岸的船数，用一个三元组 (m,c,b) 来表示问题的状态。

对 A* 算法，首先需要确定估价函数。设 $g(n)=d(n)$，$h(n)=m+c-2b$，则有

$$\begin{aligned}f(n)&=g(n)+h(n)\\&=d(n)+m+c-2b\end{aligned}$$

其中，$d(n)$表示节点的深度。通过分析可知，$h(n)\leqslant h^*(n)$，满足 A* 算法的限制条件。

修道士和野人问题的搜索树如图 3-19 所示，其中每个节点旁边标注的是该节点的 h 值和 f 值。

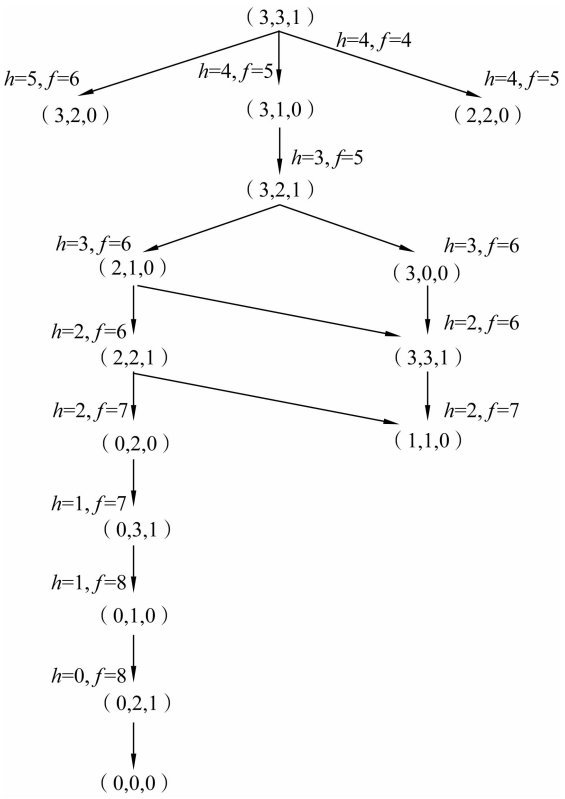

图 3-19 修道士和野人问题的搜索树

3.3 AND/OR 图搜索策略

AND/OR 图表示问题的求解过程与状态空间法类似，也是通过搜索来实现对问题求解的。AND/OR 图的搜索策略分为盲目搜索和启发式搜索两大类。其中 AND/OR 图的广度优先搜索和深度优先搜索均属于盲目搜索策略。

3.3.1 AND/OR 图搜索

AND/OR 图的搜索过程实际上是一个不断寻找解树的过程，但它不像在或图中那样只是简单地寻找目标节点，而是边扩展节点边进行逻辑判断，以确定初始节点是否可解。一旦能够确定初始节点是可解的，则搜索停止。这时根据返回指针便可从搜索树中得到一个解树(图)。所以，准确地说，解树实际上是由可解节点形成的一个子树，这个子树的根为初始节点，叶子为终止节点，且这个子树一定是与树。

AND/OR 图的一般搜索过程可概括为以下步骤：

(1)把初始节点 S_0 放入 OPEN 表。

(2)将 OPEN 表中第一个节点 n 移出，并放入 CLOSED 表。

(3)若节点 n 可扩展，则做下列工作：

①扩展节点 n，将其子节点配上指向父节点的指针后放入 OPEN 表。

②考察在这些子节点中是否有终止节点。若有，则标记它们为可解节点，并将它们也放入 CLOSED 表，然后由它们的可解反向推断其先辈节点的可解性，并对其中的可解节点进行标记。如果初始节点也被标记为可解节点，则搜索成功，结束。

③删去 OPEN 表中那些具有可解先辈的节点(因为其先辈节点已经可解，故已无再考察这些节点的必要)，转步骤(2)。

(4)若节点 n 不可扩展，则做下列工作：

①标记 n 为不可解节点，然后由它的不可解反向推断其先辈节点的可解性，并对其中的不可解节点进行标记。如果初始节点也被标记为不可解节点，则搜索失败，退出。

②删去 OPEN 表中那些具有不可解先辈的节点(因为其先辈节点已不可解，故已无再考察这些节点的必要)，转步骤(2)。

另外，AND/OR 图的广度优先搜索和深度优先搜索的区别仅在于上述算法的步骤(3)子节点进入 OPEN 表的顺序不同。在扩展节点时，AND/OR 图的广度优先搜索过程总是将刚生成的节点放在 OPEN 表的尾部，而深度优先搜索过程则是将刚生成的节点放在 OPEN 表的首部。

【例 3-8】 设有一个 AND/OR 图如图 3-20 所示，其中 1 号节点为初始节点，t_1、t_2、t_3、t_4 均为终止节点，A 和 B 为不可解节点。

采用 AND/OR 图的广度优先搜索策略，其搜索过程如下：

(1)扩展 1 号节点，得到 2 号和 3 号节点，依次放入 OPEN 表的尾部。由于这两个节点都是非终止节点，所以接着扩展 2 号节点，此时 OPEN 表中只有 3 号节点。

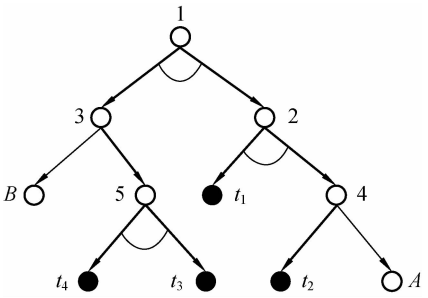

图 3-20　例 3-8 用图

(2)扩展 2 号节点,得到 4 号节点和 t_1 节点。此时 OPEN 表中依次有 3 号、4 号和 t_1 节点。由于 t_1 是终止节点,故标记它为可解节点,并将其放入 CLOSED 表,再判断其先辈节点的可解性,但 t_1 的父节点 2 是一个与节点,故仅由 t_1 的可解性还不能确定 2 号节点的可解,所以继续搜索。

(3)扩展 3 号节点,得到 5 号节点和 B 节点。两者均非终止节点,所以继续扩展 4 号节点。

(4)4 号节点扩展后得到 A 节点和 t_2 节点。由于 t_2 是终止节点,故标记为可解节点,并放入 CLOSED 表。这时其先辈节点 4 和 2 也为可解节点,但 1 号节点还不能确定。这时从 OPEN 表中删去节点 A,因为其父节点 4 已经可解。

(5)扩展 5 号节点得到 t_3 节点和 t_4 节点。由于 t_3 和 t_4 都为终止节点(放入 CLOSED 表),故可推得节点 5、3、1 均为可解节点。搜索成功,结束。

此时,由 CLOSED 表便得到由节点 1、2、3、4、5、t_1、t_2、t_3、t_4 构成的解树(图 3-20 中的粗线部分)。

3.3.2　AND/OR 图的启发式搜索

AND/OR 图的广度优先搜索和深度优先搜索均属于非启发式搜索,其共同点在于:搜索从初始节点开始,先自上而下地进行搜索,寻找终止节点和端节点,然后再自下而上地进行可解性标记,一旦初始节点被标记为可解节点或不可解节点,搜索就不再继续进行;搜索都是按确定路线进行的,当要选择一个节点进行扩展时,只是根据节点在 AND/OR 图中所处的位置,而没有考虑要付出的代价,因而求得的解树不一定是代价最小的解树,即不一定是最优解树。

为求得最优解树就需要在搜索过程中加入启发性信息。下面就来讨论 AND/OR 图的启发式搜索问题。

1. 解树的代价

要寻找最优解树(代价最小的那棵解树),首先需要计算解树的代价。在 AND/OR 图的启发式搜索过程中,解树的代价可按如下规则计算:

(1)若 n 为终止节点,则其代价 $h(n)=0$。

(2)若 n 为或节点,且其子节点为 $n_1,n_2,n_3,\cdots,n_k$,则 n 的代价为

$$h(n)=\min_{1\leqslant i\leqslant k}\{c(n,n_i)+h(n_i)\}$$

其中,$c(n,n_i)$是节点 n 到其子节点 n_i 的边代价。

(3)若 n 为与节点,且其子节点为 $n_1,n_2,n_3,\cdots,n_k$,则 n 的代价可用和代价法或最大代价法计算。

①和代价法,其计算公式为

$$h(n)=\sum_{i=1}^{k}[c(n,n_i)+h(n_i)]$$

②最大代价法,其计算公式为

$$h(n)=\max_{1\leqslant i\leqslant k}\{c(n,n_i)+h(n_i)\}$$

(4)若 n 是端节点,但不是终止节点,则 n 不可扩展,其代价定义为

$$h(n)=\infty$$

(5)根节点的代价即为解树的代价。

【例 3-9】 设图 3-21 是一个 AND/OR 图,其中包括两棵解树,左边的解树由 S_0,A,t_1,C 及 t_3 组成;右边的解树由 S_0,B,t_2,D 及 t_4 组成。在此 AND/OR 图中,t_1,t_2,t_3,t_4 为终止节点;E,F 是端节点;边上的数字是该边的代价。请计算解树的代价。

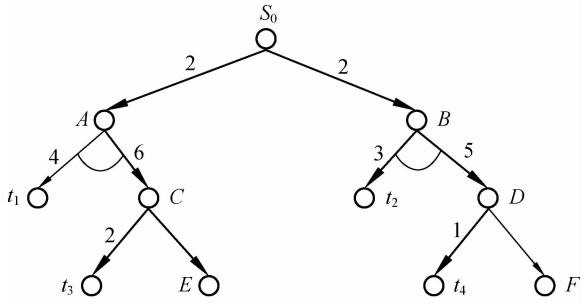

图 3-21 例 3-9 用图

由左边的解树可得:

①按和代价法:$h(S_0)=2+4+6+2=14$。

②按最大代价法:$h(S_0)=8+2=10$。

由右边的解树可得:

①按和代价法:$h(S_0)=1+5+3+2=11$。

②按最大代价法:$h(S_0)=6+2=8$。

显然,针对例 3-19,无论是按和代价法还是按最大代价法,右边的解树都是最优解。但有时候,当采用的代价法不同时,找到的最优解树有可能是不同的。

2. 希望树

为了找到最优解树,搜索过程的任何时刻都应该选择那些最有希望成为最优解树一

部分的节点进行扩展。由于这些节点及其父节点所构成的 AND/OR 图最有可能成为最优解树的一部分，因此称它为希望解树，简称希望树。下面给出希望树的定义。

定义 3.2 希望树 T：

(1)初始节点 S_0 在希望树 T 中。

(2)如果 n 是具有子节点 $n_1, n_2, n_3, \cdots, n_k$ 的或节点，则 n 的某个子节点 n_i 在希望树 T 中的充分必要条件是

$$h(n_i)=\min_{1\leqslant i\leqslant k}\{c(n,n_i)+h(n_i)\}$$

(3)如果 n 是与节点，则 n 的全部子节点都在希望树 T 中。

3. AND/OR 图的启发式搜索过程

AND/OR 图的启发式搜索过程是一个不断选择、修正希望树的过程，如果问题有解，则经此搜索过程必将找到最优解。其搜索过程如下：

(1)把初始节点 S_0 放入 OPEN 表，并计算 $h(S_0)$。

(2)计算希望树 T。

(3)依次在 OPEN 表中取出 T 的端节点，放入 CLOSED 表，并记该节点为 n。

(4)如果节点 n 为终止节点，则做下列工作：

①标记节点 n 为可解节点。

②在 T 上应用可解标记过程对 n 的先辈节点中的所有可解节点进行标记。

③如果初始节点 S_0 能够被标记为可解节点，则 T 就是最优解树，成功退出。

④否则，从 OPEN 表中删去具有可解先辈的所有节点。

⑤转步骤(2)。

(5)如果节点 n 不是终止节点，但可扩展，则做下列工作：

①扩展节点 n，生成 n 的所有子节点。

②把这些子节点都放入 OPEN 表，并为每个子节点设置指向父节点 n 的指针。

③计算这些子节点及其先辈节点的 h 值。

④转步骤(2)。

(6)如果节点 n 不是终止节点，且不可扩展，则做下列工作：

①标记节点 n 为不可解节点。

②在 T 上应用不可解标记过程对 n 的先辈节点中的所有不可解节点进行标记。

③如果初始节点 S_0 能够被标记为不可解节点，则问题无解，失败退出。

④否则，从 OPEN 表中删去具有不可解先辈的所有节点。

⑤转步骤(2)。

上述过程可用图 3-22 所示的流程来描述。

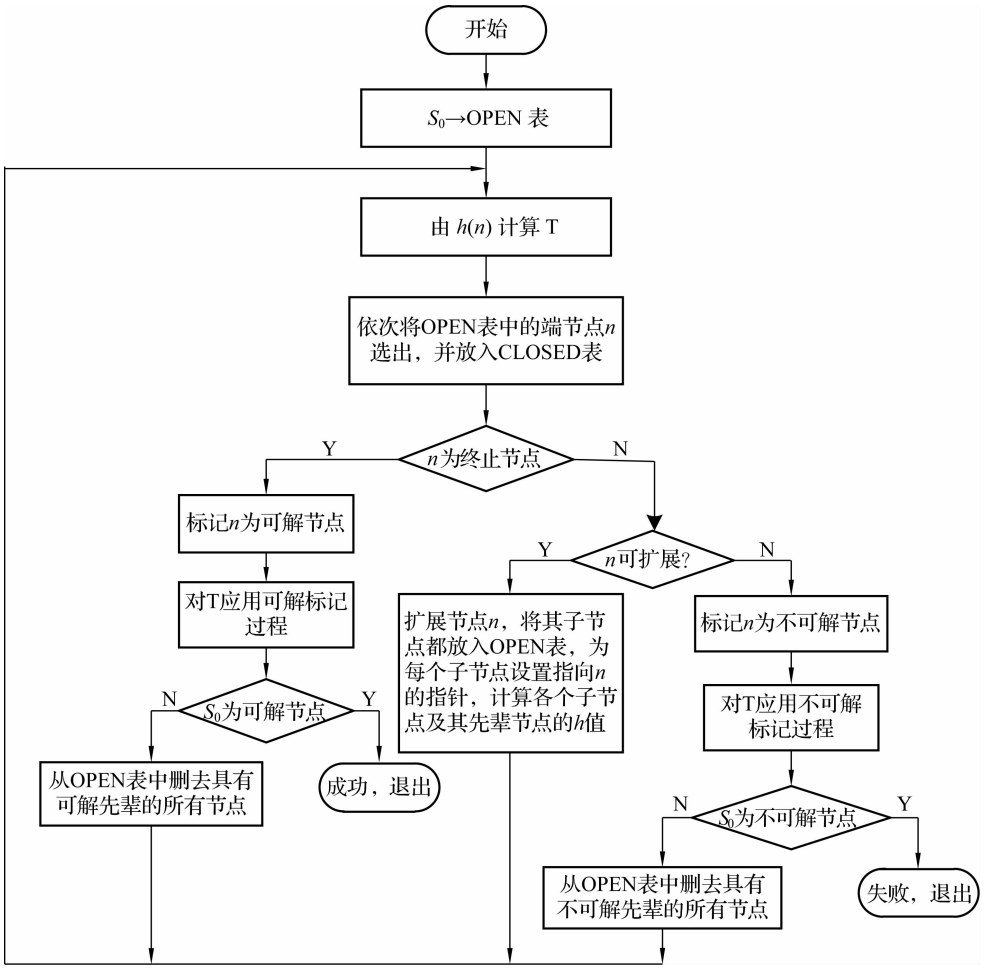

图 3-22　AND/OR 图的启发式搜索流程

【例 3-10】 举例说明 AND/OR 图的启发式搜索过程。

设初始节点为 S_0，每次扩展两层，并设 S_0 经扩展后得到如图 3-23(a)所示的 AND/OR 图，其中子节点 B、C、E、F 用启发函数估算出的 g 值分别是

$$g(B)=3, g(C)=3, g(E)=3, g(F)=2$$

若按和代价法计算，则得到

$$g(A)=8, g(D)=7, g(S_0)=8$$

其中，边代价一律按 1 计算。此时，S_0 的右子树是希望树。下面将对此希望树的节点进行扩展。

设对节点 E 扩展两层后得到如图 3-23(b)所示的 AND/OR 图，节点旁的数字为用启发函数估算出的 g 值，则按和代价法计算得到

$$g(G)=7, g(H)=6, g(E)=7, g(D)=11$$

此时，由 S_0 的右子树算出的 $g(S_0)=12$。但是，由左子树算出的 $g(S_0)=9$。显然，左子树的代价小，所以现在改取左子树作为当前的希望树。

假设对节点 B 扩展两层后得到如图 3-23(c)所示的 AND/OR 图，节点旁的数字是对相应节点的估算值，节点 L 的两个子节点是终止节点，则按和代价法计算得到

$$g(L)=2, g(M)=6, g(B)=3, g(A)=8$$

由此可推算出 $g(S_0)=9$。这时，左子树仍然是希望树，继续对其扩展，扩展节点 C。

假设节点 C 扩展两层后得到如图 3-23(d)所示的 AND/OR 图，节点旁的数字是对相应节点的估算值，节点 N 的两个子节点都是终止节点。按和代价法计算得到

$$g(N)=2, g(P)=7, g(C)=3, g(A)=8$$

由此可推算出 $g(S_0)=9$。另外，由于 N 的两个子节点都是终止节点，所以 N 和 C 都是可解节点。再由前面推出的 B 是可解节点，可推出 A 和 S_0 都是可解节点。这样就求出了代价最小的解树，即最优解树(图中粗线部分)。该最优解树是用和代价法求出来的，解树的代价为 9。

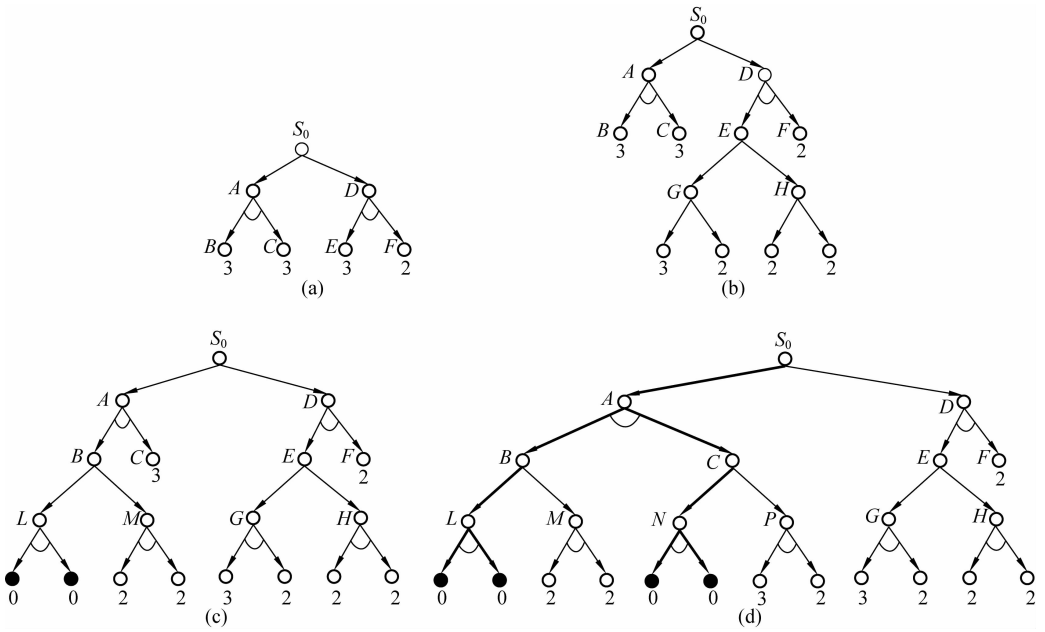

图 3-23　AND/OR 图启发式搜索

3.4　博弈树搜索策略

博弈自古以来都被认为是最富有挑战性的智力游戏，具有吸引人的非凡魅力。早期的博弈通常是人与人之间的对垒，自从计算机诞生以后，便出现了计算机博弈的概念。计算机博弈是人工智能领域的一个重要研究方向，是一个挑战无穷、生机勃勃的研究课题。

谷歌研制的人工智能系统阿尔法围棋 2016 年与韩国围棋九段棋手李世石及 2017 年与中国围棋九段棋手柯洁的比赛，更是将计算机博弈推向新的高度。

目前，我国由中国人工智能学会举办的中国大学生计算机博弈大赛成为该领域的一项重要赛事，每年都吸引着众多的在校大学生积极参与。

3.4.1 博弈树的概念

博弈分为双人完备信息博弈和机遇性博弈两种，这里主要研究的是双人完备信息博弈。其特点是：双人零和、非偶然和全信息。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，而且能估计出对方未来的走步。对垒的结果是一方赢、另一方输或者双方和局。这一类博弈的典型例子是跳棋、象棋、围棋等。

在双人完备信息博弈过程中，双方都希望自己能够获胜。因此，当任何一方走步时，都是选择对自己最为有利，而对另一方不利的行动方案。假设博弈的一方为 MAX，另一方为 MIN。在博弈过程的每一步，可供 MAX 和 MIN 选择的行动方案都可能有多种。从 MAX 方的观点看，可供自己选择的那些行动方案之间是“或”关系，原因是主动权掌握在 MAX 手中，选择哪个方案完全是由自己决定的；而那些可供对方选择的行动方案之间是“与”关系，原因是主动权掌握在 MIN 手里，任何一个方案都有可能被 MIN 选中，MAX 必须防止那种对自己最为不利的情况发生。

若把双人完备信息博弈过程用图表示出来，就可得到一棵与/或树，这种与/或树被称为博弈树。在博弈树中，那些下一步该 MAX 走步的节点称为 MAX 节点，而下一步该 MIN 走步的节点称为 MIN 节点。博弈树具有如下特点：

(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)整个博弈过程始终站在某一方立场上，所有能使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；所有能使对方获胜的终局都是不可解节点。

3.4.2 极小极大搜索过程

对于简单的博弈问题，可以生成整个博弈树，找到必胜的策略。但对于复杂的博弈，如西洋跳棋完整的博弈树约有 10^{40} 个节点，国际象棋大约有 10^{120} 个节点，可见要生成整个搜索树显然是不切实际的。一种可行的办法是用当前正在考察的节点生成一棵部分博弈树，由于该博弈树的叶节点一般不是哪一方的获胜节点，因此，需要利用估价函数 $f(n)$ 对叶节点进行静态估值。一般来说，那些对 MAX 有利的节点，其估价函数取正值；那些对 MIN 有利的节点，其估价函数取负值；那些使双方均等的节点，其估价函数取接近于零的值。

为了计算非叶节点的值，必须从叶节点向上倒推。对于 MAX 节点，由于 MAX 方总是选择估值最大的走步，因此，MAX 节点的倒推值应该取其后继节点估值的最大值。对于 MIN 节点，由于 MIN 方总是选择使估值最小的走步，因此 MIN 节点的倒推值应取其后继节点估值的最小值。这样一步一步地计算倒推值，直至求出初始节点的倒推值。另外，由于是站在 MAX 方的立场上，因此应选择具有最大倒推值的走步。这一过程称为极小极大搜索过程。

图 3-24 给出了一个计算倒推值的示例。其中，□代表 MAX 方，○代表 MIN 方。

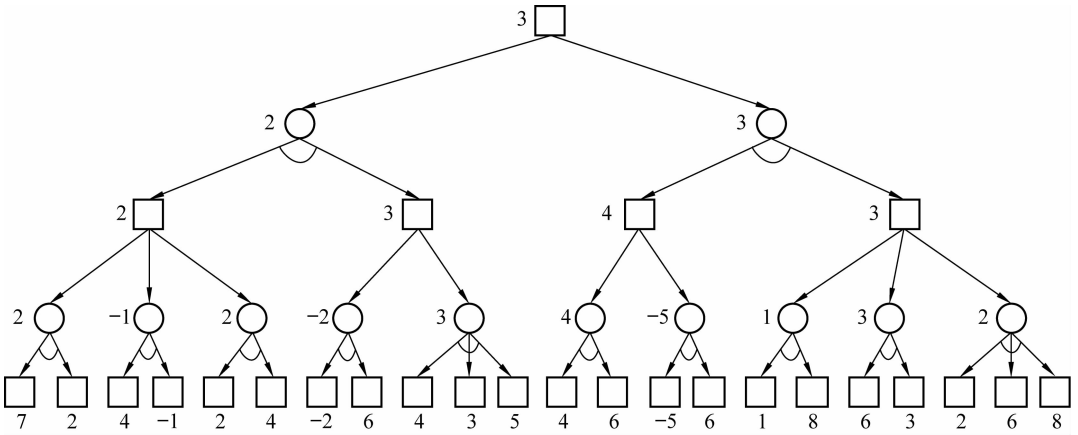

图 3-24　计算倒推值的示例

【例 3-11】　一字棋游戏。

设有一个三行三列的棋盘，如图 3-25 所示，两个棋手轮流走步，每个棋手走步时往空格上摆一个自己的棋子，谁先使自己的棋子成三子一线为赢。

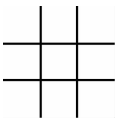

图 3-25　例 3-11 用图

设 MAX 方的棋子用×标记，MIN 方的棋子用○标记，并规定 MAX 方先走步。为了对叶节点进行静态估值，规定估价函数 $e(p)$ 如下：

(1)若 p 是 MAX 的必胜棋局，则 $e(p)=+\infty$。

(2)若 p 是 MIN 的必胜棋局，则 $e(p)=-\infty$。

(3)若 p 是胜负未定的棋局，则 $e(p)=e(+p)-e(-p)$。

其中，$e(+p)$表示棋局 p 上有可能使×成三子一线的数目，$e(-p)$表示棋局 p 上有可能使○成三子一线的数目。

另外，在搜索过程中，具有对称性的棋局被认为是同一棋局。例如，图 3-26 所示的 4 个棋局可以被认为是同一个棋局，这样可以大大减小搜索空间。

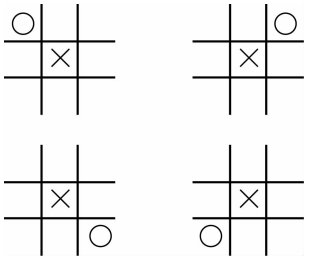

图 3-26　对称棋局

图 3-27 给出了第一着走棋后生成的博弈树。图中叶节点下面的数字是该节点的静态估值，非叶节点旁边的数字是计算出的倒推值。从图中可以看出，对 MAX 来说，S_2 是一着最好的走棋，它具有较大的倒推值。

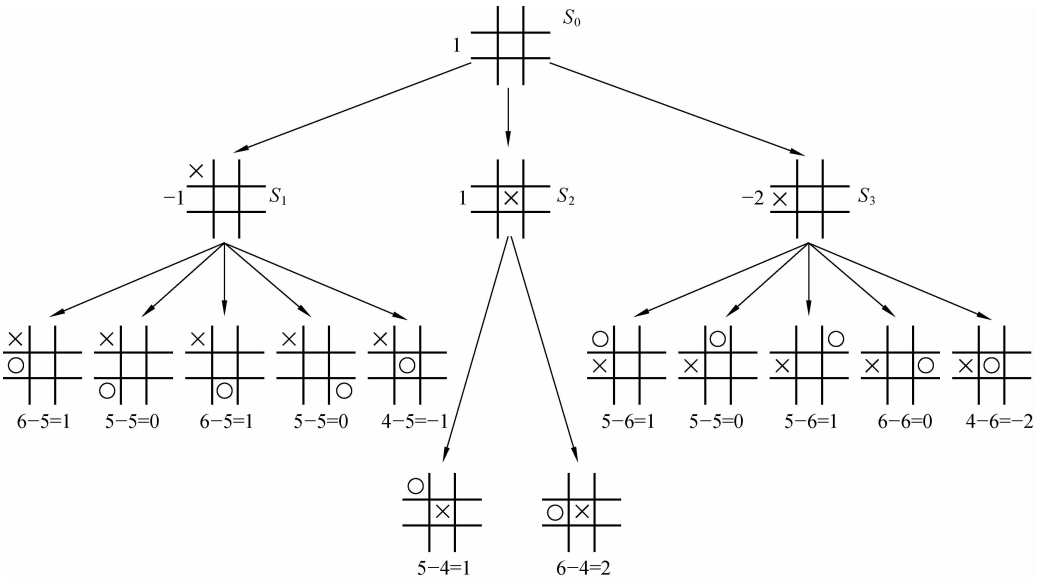

图 3-27　第一着走棋后生成的博弈树

3.4.3　α-β 过程

上述极小极大搜索过程是先生成一棵博弈树，然后计算其各节点的估值，这种生成节点和计算估值相分离的搜索方式，需要生成规定深度内的所有节点，因此搜索效率较低。

将生成节点与估值计算同时进行的博弈搜索过程称作 α-β 过程。在 α-β 过程中，计算 MAX 节点的返回值时，不是等它的所有后继 MIN 节点都取得返回值之后再取它们的极大值，而是在它的第一个后继节点取得返回值之后，确定 MAX 的最终返回值的一个下界，这个下界称为该节点的 α 值，以后每当它的后继节点的返回值确定时就考虑修改这个下界。如果后继节点的返回值比这个下界高，就把 α 改成这个返回值作为新的下界。如果已经能够看出某后继节点的返回值比这个下界低，则在此后继节点之下的其余节点就可以不必生成及进行静态估值了。α-β 过程一边生成节点，一边进行静态估值和 α 值的修改，当这个节点的所有后继节点都生成以后，α 值修改完毕，这也就是该 MAX 节点的最终返回值。

计算 MIN 节点的返回值的过程与计算 MAX 节点的返回值的过程类似。下面给出 α-β 过程的具体剪枝规则。

α-β 过程的剪枝规则由以下两条规则构成：

(1)α 剪枝。如果一个 MIN 节点的 β 值小于或等于它的某一个 MAX 祖先节点的 α 值，则剪枝发生在该 MIN 节点之下，终止这个 MIN 节点以下的搜索过程，这个 MIN 节

点的倒推值就可确定为 β 值。

(2)β 剪枝。如果一个 MAX 节点的 α 值大于或等于它的某一个 MIN 祖先节点的 β 值，则剪枝发生在该 MAX 节点之下，终止这个 MAX 节点以下的搜索过程，这个 MAX 节点的倒推值就可确定为 α 值。

在搜索过程中，α 值和 β 值按如下方式确定：

①令 MAX 节点的 α 值等于它的现有后继节点返回值中的最大者。

②令 MIN 节点的 β 值等于它的现有后继节点返回值中的最小者。

【例 3-12】 图 3-28 所示为采用了 α-β 剪枝技术的博弈树搜索，其中，最下层端节点旁边的数字是假设的估值。

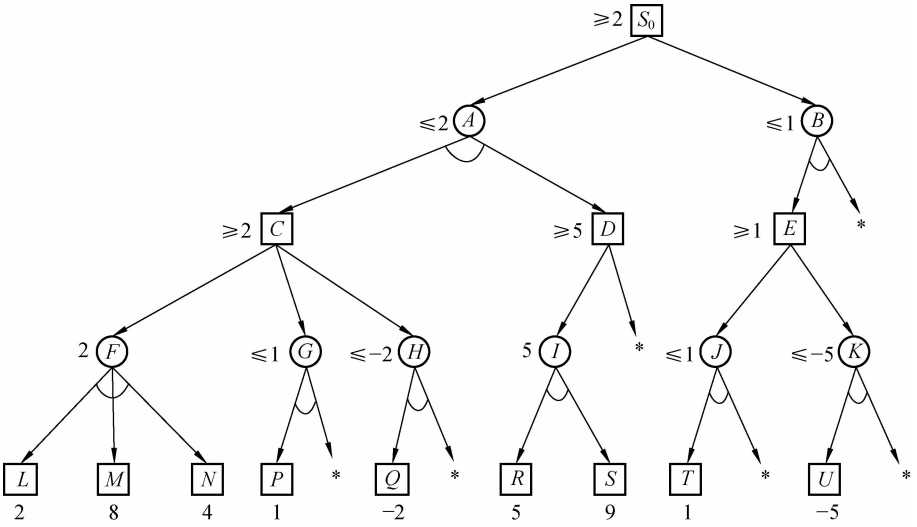

图 3-28　例 3-12 用图

在图 3-28 中，由节点 L，M，N 的估值推出节点 F 的倒推值为 2，即 F 的 β 值为 2，由此可推出节点 C 的倒推值大于或等于 2，即 C 的倒推值的下界为 2，不可能比 2 小，故 C 的 α 值为 2。由节点 P 的估值推知节点 G 的倒推值小于或等于 1，无论 G 的其他子节点的估值是多少，G 的倒推值都不可能比 1 大。事实上，随着子节点的增多，G 的倒推值只可能是越来越小，因此 1 是 G 的倒推值的上界，所以 G 的 β 值为 1。另外，已经知道 C 的倒推值大于或等于 2，G 的其他子节点又不可能使 C 的倒推值增大，因此对 G 的其他分枝不必进行搜索，这相当于把这些分枝剪去。同理可知，节点 H 的 β 值为 -2，节点 H 除 Q 之外的其他分枝也被剪去。由节点 F，G，H 的倒推值可推出节点 C 的倒推值为 2，再由 C 可推出节点 A 的倒推值小于或等于 2，即 A 的 β 值为 2。由节点 R，S 推出节点 I 的倒推值为 5，此时可推出 D 的倒推值大于或等于 5，即 D 的 α 值为 5。此时 D 的其他子节点的倒推值无论是多少都不能使 D 的倒推值减小或 A 的倒推值增大，所以 D 的其他分枝被剪去，并可确定 A 的倒推值为 2。用同样的方法可推出其他分支的剪枝情况，最终推出 S_0 的倒推值为 2。

最后，讨论 α-β 过程的搜索效率。为了实现 α-β 剪枝，搜索树的某一部分必须生成到

最大深度。另外，对一个节点来说，如果它的 α 值或 β 值越早接近它的最终返回值，则它的后裔节点产生剪枝的机会就越多。树的起点的返回值与某一个终端节点的静态估值是相同的。如果在宽度优先搜索的方式下这个终端节点能够先达到，则剪枝数将是最大的，当树的剪枝数最大时，需要生成并进行静态估值的终端节点的个数是最少的。

假设树的深度是 D，每个节点(终端节点除外)有 B 个后继节点，则这样一棵树有 B^D 个终端节点。假设 α-β 过程是以各节点最终的返回值的顺序产生后继节点的，即对 MIN 节点来说，先产生具有最小值的后继节点；对 MAX 节点来说，先产生具有最大值的后继节点(当然，在产生节点的时候一般不知道返回值，因此，上述节点排序实际上是不存在的，为了产生最大的剪枝数，此处人为地安排了上述排序)。

按如上假设的排序方式，α-β 搜索所产生的剪枝数量最多，需要产生的终端节点数量最少。设在这种情况下需要产生的终端节点数是 N_D，则可以证明

$$N_D=\begin{cases}2B^{D/2}-1 & (D\text{ 为偶数})\\ B^{(D+1)/2}+B^{(D-1)/2}-1 & (D\text{ 为奇数})\end{cases}$$

由此可见，在最好的情况下，使用 α-β 过程搜索深度为 D 的树所产生的终端节点数大约等于不用 α-β 过程搜索深度为 $D/2$ 的树所产生的节点数，因此，在使用相同存储空间的条件下，α-β 过程能把搜索深度扩大一倍。

习　题　3

1. 什么是搜索？有哪两大类不同的搜索方法？两者的区别是什么？
2. 什么是状态空间？用状态空间法表示问题时，什么是问题的解？
3. 什么是与/或图？什么是解树？
4. 简述状态空间的一般搜索过程。
5. 广度优先搜索和深度优先搜索有何区别？
6. 什么是估价函数？在估价函数中 $g(n)$和 $h(n)$各起什么作用？
7. 画出图 3-29 中与/或树的 4 棵解树。

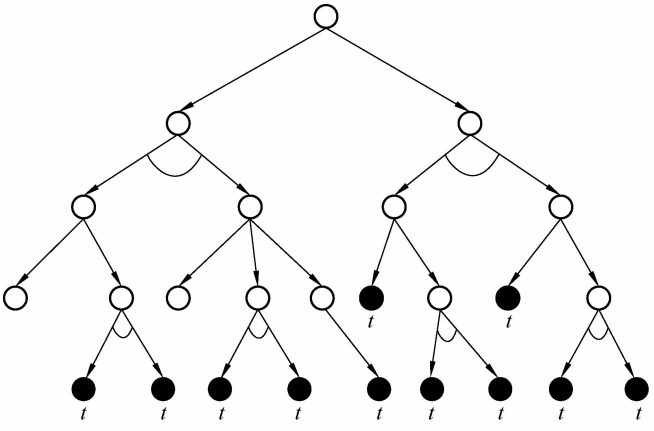

图 3-29　习题 3 第 7 题用图

8. 设有如图 3-30 所示的与/或树，请分别用与/或树的广度优先搜索和深度优先搜索求出解树。

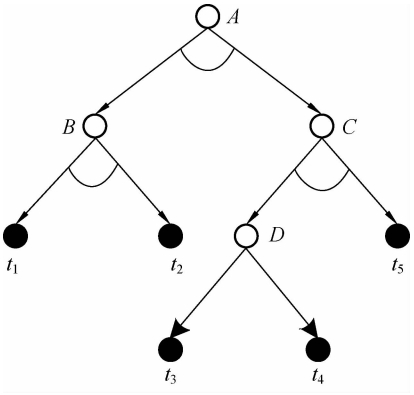

图 3-30　习题 3 第 8 题用图

9. 图 3-31 是 5 个城市的交通图，城市之间连线旁边的数字是城市之间路程的费用，要求从 A 城出发，经过其他各城一次且仅一次，最后回到 A 城，请找出一条最优路线图。

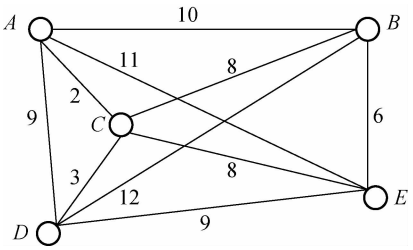

图 3-31　习题 3 第 9 题用图

10. 设有如图 3-32 所示的博弈树，其中最下面的数字是假设的估值，请对该博弈树做如下工作：

(1)计算各节点的倒推值。

(2)利用 α-β 剪枝技术剪去不必要的分支。

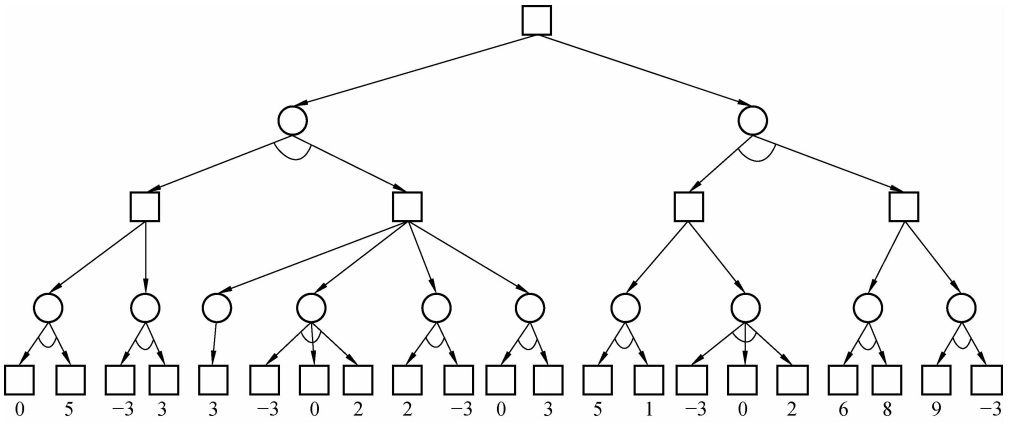

图 3-32　习题 3 第 10 题用图

第4章 不确定性知识的表示与推理

现实世界中，人们通常是在信息不完善、不精确的情况下，运用不确定性知识进行思维、求解问题的，推出的结论也并不总是随着知识的增加而单调地增加。因此，研究不确定性知识的表示、处理及推理方法非常重要。

本章研究的主要内容是不确定性知识的表示、处理及推理方法。在此基础上，介绍5种常用的不确定性推理方法，包括概率推理、主观Bayes方法、可信度方法、证据理论和模糊推理。

4.1 不确定性知识的表示与推理概述

许多比较复杂的人工智能系统往往含有不确定性、不确切性、不完全性和不一致性。

当采用产生式系统或专家系统的结构时，要求设计者建立某种不确定性问题的代数模型及其计算和推理过程。

4.1.1 不确定性推理的概念

所谓推理，就是从已知事实出发，运用相关的知识（或规则）逐步推出结论或者证明某个假设成立或不成立的思维过程。其中，已知事实和知识（规则）是构成推理的两个基本要素；已知事实是推理过程的出发点及推理中使用的知识，称为证据；知识（或规则）是推理得以向前推进，并逐步达到最终目标的根据。

但是，现实世界中的事物以及事物之间的关系是极其复杂的，由于客观上存在的随机性、模糊性以及某些事物或现象暴露的不充分性，导致人们对它们的认识往往是不精确、不完全的，具有一定的不确定性。这种认识上的不确定性反映到知识以及由观察所得到的证据上来，就分别形成了不确定性的知识及不确定性的证据。

1.（狭义）不确定性

不确定性（uncertainty）就是一个命题（所表示的事件）的真实性不能完全肯定，而只能对其为真的可能性给出某种估计。例如：

如果乌云密布并且电闪雷鸣，则很可能要下暴雨。

如果头痛发烧，则大概是患了感冒。

这两个命题就是含有不确定性的命题。当然，它们描述的是人们的经验性知识。

2. 不确切性（模糊性）

不确切性（imprecision）就是一个命题中所出现的某些言辞的含义不够确切，从概念角度讲，也就是其代表的概念的内涵没有硬性的标准或条件，其外延没有硬性的边界，即边界是软的或者说是不明确的。例如：

小王是个高个子。

张三和李四是好朋友。

如果向左转，则身体就向左稍倾。

这几个命题就含有不确切性，因为其中的言辞“高”“好朋友”“稍倾”等的含义都是不确切的。不妨称这种含义不确切的言辞所代表的概念为软概念（soft concept）。

注意：在模糊集合（fuzzy set）的概念出现以后，可以将这里的不确切性称为模糊性（fuzziness），将含义不确切的言辞所代表的概念称为模糊概念。

3. 不完全性

不完全性就是对某事物来说，关于它的信息或知识还不全面、不完整、不充分。例如，在破案的过程中，警方所掌握的关于罪犯的有关信息，往往就是不完全的。但就是在这种情况下，办案人员仍能通过分析、推理等手段而最终破案。

4. 不一致性

不一致性就是在推理过程中得出了前后不相容的结论；或者随着时间的推移或范围的扩大，原来一些成立的命题变得不成立、不适合了。例如，牛顿定律对于宏观世界是正确的，但对于微观世界和宇观世界却是不适合的。

4.1.2 不确定性推理中的基本问题

在不确定性推理中，知识和证据都具有某种程度的不确定性，这就使推理机的设计、实现的复杂度和难度增大。它除了必须解决推理方向、推理方法及控制策略等问题外，一般还要解决证据和知识的不确定性的度量及表示问题、不确定性知识（或规则）的匹配问题、不确定性传递算法以及多条证据同时支持结论的情况下不确定性的更新或合成问题。

1. 不确定性的表示

不确定性的表示包括知识不确定性的表示和证据不确定性的表示。

1）知识不确定性的表示

知识不确定性的表示方式是与不确定性推理方法密切相关的一个问题。在选择知识不确定性的表示方式时，通常需要考虑以下两个方面的因素：一是要能够比较准确地描述问题本身的不确定性；二是要便于推理过程中不确定性的计算。对于这两个方面的因素，一般是将它们结合起来综合考虑的，只有这样才能得到较好的表示效果。

知识的不确定性通常是用一个数值来描述的，该数值表示相应知识的确定性程度，也称为知识的静态强度。知识的静态强度可以是该知识在应用中成功的概率，也可以是该知识的可信程度等。如果用概率来表示静态强度，则其取值范围为[0,1]，该值越接近于1，说明该知识越接近于“真”；其值越接近于0，说明该知识越接近于“假”。如果用可信度来表示静态强度，则其取值范围一般为[－1,1]。当该值大于0时，值越大，说明知识越接近于“真”；当其值小于0时，值越小，说明知识越接近于“假”。在实际应用中，知识的不确定性是由领域专家给出的。

2)证据不确定性的表示

证据可有多种不同的分类方法。如果按照证据的组织形式，证据可分为基本证据和组合证据。所谓基本证据，就是指那种单一证据和单一证据的否定。所谓组合证据，就是指那种将多个基本证据组织到一起形成的复合证据。

如果按照证据的不同来源，证据可分为初始证据和中间结论。所谓初始证据，就是指在推理之前由用户提供的原始证据，如患者的症状、检查结果等，其可信度是由提供证据的用户给出的。所谓中间结论，就是指在推理过程中所得到的中间结果，它将被放入综合数据库，并作为以后推理的证据来使用，其可信度是在推理过程中按不确定性更新算法计算出来的。

证据不确定性的表示应包括基本证据的不确定性表示和组合证据的不确定性计算。

(1)基本证据的不确定性表示。其表示方法通常应该与知识的不确定性表示方法保持一致，以便推理过程能对不确定性进行统一处理。基本证据的不确定性常用的表示方法有可信度方法、概率方法、模糊集方法等。

(2)组合证据的不确定性计算。多个基本证据的组合方式可以是析取关系，也可以是合取关系。当一个知识的前提条件是由多个基本证据组合而成时，各个基本证据的不确定性表示方式同上，组合证据的不确定性可在各基本证据的基础上由最大最小方法、概率方法和有界方法等计算得到。

2. 不确定性的匹配

推理过程实际上是一个不断寻找和运用可用知识的过程。所谓可用知识，就是指其前提条件可与综合数据库中的已知事实相匹配的知识。只有匹配成功的知识才可以被使用。

在不确定性推理中，由于知识和证据都是不确定的，而且知识所要求的不确定性程度与证据实际所具有的不确定性程度不一定相同，那么，怎样才算匹配成功呢？这是一个需要解决的问题。目前，常用的解决方法是：设计一个用来计算匹配双方相似程度的算法，并给出一个相似的限度，如果匹配双方的相似程度落在规定的限度内，则称匹配双方是可匹配的；否则，称匹配双方是不可匹配的。

3. 不确定性的更新

在不确定性推理中，由于证据和知识均是不确定的，因此就存在以下两个问题：一是在推理的每一步如何利用证据和知识的不确定性去更新结论(在产生式规则表示中也称

为假设)的不确定性;二是在整个推理过程中如何把初始证据的不确定性传递给最终结论。

对于第一个问题,一般做法是按照某种算法由证据和知识的不确定性计算出结论的不确定性,至于如何计算,不同不确定性推理方法的处理方式各有不同。

对于第二个问题,不同不确定性推理方法的处理方式基本相同,都是把当前推出的结论及其不确定性作为新的证据放入综合数据库,供以后推理使用。由于推理第一步得出的结论是由初始证据推出的,该结论的不确定性当然要受初始证据的不确定性的影响,当把它放入综合数据库作为新的证据进一步推理时,该不确定性又会传递给后面的结论,如此进行下去,就会把初始证据的不确定性逐步传递给最终结论。

4. 不确定性结论的合成

在不确定性推理过程中,很可能会出现由多个不同知识推出同一结论,并且推出的结论的不确定性程度又各不相同的情况。对此,需要采用某种算法对这些不同的不确定性进行合成,求出该结论的综合不确定性。

以上问题是不确定性推理中需要考虑的一些基本问题,但也并非每种不确定性推理方法都必须包括这些内容。实际上,不同的不确定性推理方法所包括的内容可以不同,并且对这些问题的处理方法也可以不同。

4.1.3 不确定性推理的分类

目前,关于不确定性推理有多种不同的分类方法,如果按是否采用数值来描述不确定性,可将其分为数值方法和非数值方法两大类型。数值方法是一种用数值对不确定性进行定量表示和处理的方法。人工智能对它的研究和应用较多,目前已形成了多种不确定性推理模型。非数值方法是指除数值方法以外的其他各种对不确定性进行表示和处理的方法,如非单调推理、不完备推理等。

对于数值方法,又可按其所依据的理论分为两大类型:一类是在概率理论的基础上形成的模型,另一类是在模糊逻辑的基础上形成的模型。第一类模型又可分为基于概率的推理模型和直接概率推理(以下简称“概率推理”)模型。其中,基于概率的推理模型包括确定性理论、主观 Bayes 方法、证据理论等,它们的共同特点是需要用到概率,但又不能在概率论的框架内直接推理。而概率推理模型却不同,它可以在概率论的框架内直接推理,如贝叶斯(Bayes)网络推理方法。

4.2 不确定性推理方法

由于现实问题中普遍含有种种的不确定性,大大提高了对不确定性推理技术的需求,因此不确定性推理技术引起了人们的广泛重视。

4.2.1 概率推理

概率推理也称 Bayes 推理,是以 Bayes 法则为基础的不确定性推理方法,具有处理

“事物发生与否不能确定”这样的不确定性的能力。而这样的不确定性在现实问题中是普遍存在的。因此，概率推理有着广泛的应用。为了便于说明，以下将以多源目标身份信息融合处理这一当前十分热门的应用为例对这种推理方法加以介绍。

在一个特定的目标身份融合问题中，目标身份的可能种类的集合称为假设空间，可以抽象地表示为一个有限集合。假定这个集合是确定的，且该集合中的每个元素（每种类型）的先验概率是已知的。现有若干信息源（如传感器）分别能够从某一角度对所关注的目标进行观察，并给出目标身份为假设空间中每一类型的条件下得到这一观察结果的条件概率，则利用概率论中著名的 Bayes 法则就能够得出融合所有信息源观察信息后的目标各种可能身份的验后概率。这就是概率推理的基本原理。

假定目标身份的类型为 $O_1, O_2, \cdots, O_m$，即假设空间为

$$\Theta = \{O_1, O_2, \cdots, O_m\} \tag{1}$$

同时假定传感器（信息源）的数量为 n，它们在某一次观察中得到的关于目标身份信息的观察报告分别为 $D_1, D_2, \cdots, D_n$，即观察报告集为

$$D = \{D_1, D_2, \cdots, D_n\} \tag{2}$$

按照前面的说法，若已知各先验概率 $P(O_i)(i=1,2,\cdots,m)$，且按照概率论的要求，满足

$$\sum_{i=1}^{m} P(O_i) = 1 \tag{3}$$

同时，每个信息源给出条件概率 $P(D_j \mid O_i)(i=1,2,\cdots,m)(j=1,2,\cdots,n)$。按照 Bayes 公式，有

$$P(O_i \mid D) = \frac{P(D \mid O_i)P(O_i)}{\sum_{i=1}^{m} P(D \mid O_i)P(O_i)} \tag{4}$$

$P(D_j \mid O_i)(i=1,2,\cdots,m)$ 是综合了各传感器报告结果后各目标身份的概率。这就是概率推理的推理结果。可以看出，这个概率推理过程也是一个信息融合过程。如果各传感器的报告相互独立，则

$$P(D \mid O_i) = \prod_{j=1}^{n} P(D_j \mid O_i) \tag{5}$$

将式(5)代入式(4)，有

$$P(O_i \mid D) = \frac{\prod_{j=1}^{n} P(D_j \mid O_i)P(O_i)}{\sum_{i=1}^{m}\prod_{j=1}^{n} P(D_j \mid O_i)P(O_i)} \tag{6}$$

概率推理最大的优点在于它的理论严密性。概率推理以概率论为基础，而概率论是建立在完整的公理体系之上的，因此，概率推理是严密的。只要有关先验概率和条件概率可信，则概率推理的结果就是可信的。概率推理的这种理论严密性也正是被广泛应用的根本原因。因此，尽管新的推理技术不断出现，但人们对这种十分经典的推理技术还是偏爱有加的。

然而，概率推理也面临一些问题。首先是要求知道各先验概率 $P(O_i)$ 和条件概率 $P(D_j|O_i)$，且这些概率的分配互斥而完备。这种要求对许多应用问题来说显得苛刻。实际中的许多问题给出的信息要么不互斥，即存在“分不清”的信息；要么存在不完备，即存在“不知道”的信息。面对这样的问题，概率推理是无能为力的。

此外，概率推理是比较值得信赖的，原因还是在于它的严密性。相比之下，有的推理技术在根基上就难以服人，推理结果自然就没有保障了。至于先验概率的获取问题，现有的一些办法还是很有效的。这些办法主要包括：根据大量试验样本分布得出的试验概率；利用等可能性事件概率相等的原则得出的古典概率；基于人的主观感觉得出的主观概率；当没有任何先验信息时，可以将各可能事件的先验概率设为相等的“不区分原则”；等等。这些办法从某种角度讲都有较为充分的合理性。

4.2.2 主观 Bayes 方法

主观 Bayes 方法又称主观概率论，是由美国斯坦福大学国际研究所的研究员杜达(R. O. Duda)等人于 1976 年提出的一种不确定性推理模型，它是对概率论中基本 Bayes 公式的改进，是一种基于概率逻辑的方法。该方法在地矿勘探专家系统 PROSPECTOR 中得到了成功的应用。

1. 知识不确定性的表示

在主观 Bayes 方法中，知识(规则)的不确定性是以一个数值对(LS，LN)来进行描述的。若以产生式规则的形式表示，则具体为

$$\text{IF}\quad E\quad \text{THEN}\quad (\text{LS},\text{LN})\quad H\quad (P(H))$$

其中，

(1)(LS，LN)是为度量产生式规则的不确定性而引入的一组数值，LS 表示规则成立的充分性，用于指出证据 E 对结论 H 为真的支持程度；而 LN 则表示规则成立的必要性，用于指出证据 E 对结论 H 为真的必要性程度。它们的定义如下：

$$\text{LS}=\frac{P(E/H)}{P(E/\neg H)}$$

$$\text{LN}=\frac{P(\neg E/H)}{P(\neg E/\neg H)}=\frac{1-P(E/H)}{1-P(E/\neg H)}$$

LS 和 LN 的取值范围为$[0,+\infty)$。它们的具体取值由领域专家根据实际经验给出。

(2)E 是该条知识的前提条件。它既可以是一个简单条件，也可以是用 AND 或 OR 把多个简单条件连接起来的复合条件。

(3)H 是结论。$P(H)$是 H 的先验概率，它指出在没有任何专门证据的情况下结论 H 为真的概率。$P(H)$的值由领域专家根据以往的实践及经验给出。

2. 证据不确定性的表示

1)单个证据不确定性的表示方法

对于初始证据 E，其先验概率为 $P(E)$，也可以由用户根据观察 S 给出它的后验概率 $P(E/S)$。但由于后验概率 $P(E/S)$的给出比较困难，因而在 PROSPECTOR 系统中引

进了可信度 $C(E/S)$的概念。给了 $C(E/S)$就相当于给了证据的概率 $P(E/S)$。

$$P(E/S)=\begin{cases}\dfrac{C(E/S)+P(E)[5-C(E/S)]}{5} & 0\leqslant C(E/S)\leqslant 5\\ \dfrac{P(E)[C(E/S)+5]}{5} & -5\leqslant C(E/S)<0\end{cases}$$

这样，用户只要对初始证据给出相应的可信度 $C(E/S)$，就可由系统将它转换为相应的 $P(E/S)$。

2)组合证据不确定性的确定方法

当证据 E 是由多个单一证据的合取组合而成时，则

$$P(E/S)=\min\{P(E_1/S),P(E_2/S),\cdots,P(E_n/S)\}$$

当证据 E 是由多个单一证据的析取组合而成时，则

$$P(E/S)=\max\{P(E_1/S),P(E_2/S),\cdots,P(E_n/S)\}$$

对于“非”运算，用下式计算：

$$P(\neg E/S)=1-P(E/S)$$

3. 不确定性的推理计算

主观 Bayes 方法推理计算的任务是根据证据 E 的概率 $P(E)$及影响结论的知识的规则强度(LS,LN)，把 H 的先验概率 $P(H)$更新为后验概率 $P(H/E)$或 $P(H/\neg E)$。

下面就确定性证据和不确定性证据两种情况分别讨论结论 H 后验概率的推理计算方法。

1)确定性证据

确定性证据是指证据的出现与否是肯定的。

当证据 E 肯定出现时，$P(E)=P(E/S)=1$，可根据 LS 和 $P(H)$求出 $P(H/E)$。

$$P(H/E)=\frac{\mathrm{LS}\cdot P(H)}{(\mathrm{LS}-1)P(H)+1}$$

当证据 E 肯定不出现时，$P(E)=P(E/S)=0$，$P(\neg E)=1$，可根据 LN 和 $P(H)$求出 $P(H/\neg E)$。

$$P(H/\neg E)=\frac{\mathrm{LN}\cdot P(H)}{(\mathrm{LN}-1)P(H)+1}$$

2)不确定性证据

在现实中，出现确定性证据的情况是不多的，更多的是介于肯定出现和肯定不出现两者之间的不确定情况。对于不确定证据 E，$0<P(E)<1$，问题可转化为：观察 S 之下，证据 E 的概率 $P(E/S)$，再根据 $P(H)$和 $P(E/S)$确定 H 的后验概率 $P(H/S)$。在这种情况下，可使用杜达等人于 1976 年证明的公式进行后验概率的计算，包括 EH 公式和 CP 公式。

EH 公式：

$$P(H/S)=\begin{cases}P(H/\neg E)+\dfrac{P(H)-P(H/\neg E)}{P(E)}P(E/S) & 0\leqslant P(E/S)<P(E)\\ P(H)+\dfrac{P(H/E)-P(H)}{1-P(E)}[P(E/S)-P(E)] & P(E)\leqslant P(E/S)\leqslant 1\end{cases}$$

CP 公式：

$$P(H/S)=\begin{cases}P(H/\neg E)+[P(H)-P(H/\neg E)]\left[\frac{1}{5}C(E/S)+1\right] & C(E/S)\leqslant 0\\ P(H)+[P(H/E)-P(H)]\cdot\frac{1}{5}C(E/S) & C(E/S)>0\end{cases}$$

这里，$C(E/S)$为用户对初始证据给出的可信度。

4. 结论不确定性的合成算法

若有 n 条知识都支持相同的结论，而且每条知识的前提条件所对应的证据 $E_i(i=1,2,\cdots,n)$都有相应的观察 S_i 与之对应，此时只要先对每条知识分别求出 $O(H/S_i)$，然后就可运用下述公式求出 $O(H/S_1,S_2,\cdots,S_n)$和 $P(H/S_1,S_2,\cdots,S_n)$：

$$O(H/S_1,S_2,\cdots,S_m)=\frac{O(H/S_1)}{O(H)}\cdot\frac{O(H/S_2)}{O(H)}\cdot\cdots\cdot\frac{O(H/S_m)}{O(H)}\cdot O(H)$$

$$P(H/S_1,S_2,\cdots,S_m)=\frac{O(H/S_1,S_2,\cdots,S_m)}{1+O(H/S_1,S_2,\cdots,S_m)}$$

这里，$O(x)$是几率函数，它与概率函数 $P(x)$的关系为

$$O(x)=\frac{P(x)}{1-P(x)},P(x)=\frac{O(x)}{1+O(x)}$$

【例 4-1】 假定证据 A_1，A_2 必然发生，而规则

R1： $A_1\rightarrow B$ LS=20 LN=1

R2： $A_2\rightarrow B$ LS=300 LN=1

已知 B 的先验概率为 0.03，计算 B 的概率变化。

【解】 (1)依据 B 的先验概率 $P(B)=0.03$ 便可得到 B 的先验几率。

$$O(B)=\frac{0.03}{1-0.03}=0.030\ 928$$

(2) 依据 R1，根据主观 Bayes 方法有

$$O(B|A_1)=\mathrm{LS}\cdot O(B)=20\times 0.030\ 928=0.618\ 56$$

$$P(B|A_1)=\frac{0.618\ 56}{1+0.618\ 56}=0.382$$

可见，由于 A_1 的必然发生，使 B 的概率由先验概率 $P(B)=0.03$ 增到后验概率 $P(B|A_1)=0.382$。

(3)依据 R2，有

$$O(B|A_1A_2)=300\times O(B|A_1)=185.568$$

$$P(B|A_1A_2)=\frac{185.568}{1+185.568}=0.994\ 64$$

注意：按照概率传播，B 由 R1 得到的后验几率 $O(B|A_1)$，在 R2 中变成 B 的先验几率。因此，由于 A_2 的必然发生，使 B 的概率又由 0.382 增到 0.994 64。

4.2.3 可信度方法

可信度方法是美国斯坦福大学数学家爱德华·汉斯·肖特利夫(E. H. Shortlifie)等

人在确定性理论(theory of confirmation)的基础上,结合概率论等提出的一种不确定性推理方法。可信度方法于1976年在专家系统MYCIN中首先应用。可信度是指人们在实际生活中根据自己的经验或观察对某一事件或现象为真的相信程度。知识的不确定性以可信度表示。

1. 知识不确定性的表示

在基于可信度的不确定性推理模型中,知识是以产生式规则的形式表示的。其一般形式为

$$\text{IF}\quad E\quad \text{THEN}\quad H\quad (\text{CF}(H,E))$$

其中,CF(H,E)是该条知识的可信度,称为可信度因子(certainty factor)或规则强度。

在专家系统MYCIN中,CF(H,E)被定义为

$$\text{CF}(H,E)=\text{MB}(H,E)-\text{MD}(H,E)$$

这里,MB为信任增长度(measure belief),它表示因与前提条件E匹配的证据的出现,对结论H为真的信任增长度。当MB(H,E)>0时,有$P(H/E)>P(H)$。这里,$P(H)$表示H的先验概率;$P(H/E)$表示在前提条件E所对应的证据出现的情况下,结论H的条件概率。MD为不信任增长度(measure disbelief),它表示因与前提条件E匹配的证据的出现,对结论H为真的不信任增长度,当MD(H,E)>0时,有$P(H/E)<P(H)$。显然,一个证据不可能既增加对H的信任程度,又同时增加对H的不信任程度,因此MB(H,E)与MD(H,E)是互斥的。即

当MB(H,E)>0时,MD(H,E)=0。

当MD(H,E)>0时,MB(H,E)=0。

MB和MD的值域为[0,1]。

根据CF(H,E)的定义及MB(H,E)与MD(H,E)的互斥性,可得到CF(H,E)的计算公式为

$$\text{CF}(H,E)=\begin{cases}\text{MB}(H,E)-0=\dfrac{P(H/E)-P(H)}{1-P(H)} & P(H/E)>P(H)\\ 0 & P(H/E)=P(H)\\ 0-\text{MD}(H,E)=\dfrac{P(H/E)-P(H)}{P(H)} & P(H/E)<P(H)\end{cases}$$

其中,$P(H/E)=P(H)$表示E所对应的证据与H无关。CF(H,E)的取值范围是[−1,1]。

2. 证据不确定性的表示

如果支持结论的证据只有一条,则证据可信度值的确定分以下两种情况:

(1)证据为初始证据,其可信度的值一般由提供证据的用户直接指定,指定的方法也是用可信度因子对证据不确定性进行表示。例如,CF(E)=0.8表示证据E的可信度为0.8。

(2)用先前推出的结论作为当前推理的证据。对于这种情况的证据,其可信度的值在推出该结论时通过不确定性传递算法计算得到(传递算法将在下面讨论)。证据E的可信度CF(E)也是在[−1,1]上取值的。

如果支持结论的证据有多个,当多个证据间的关系是合取时,即$E=E_1\wedge E_2\wedge E_3\wedge\cdots\wedge E_n$,

则 $CF(E)=\min\{CF(E_1),CF(E_2),\cdots,CF(E_n)\}$；当多个证据间的关系是析取时，即 $E=E_1 \vee E_2 \vee E_3 \vee \cdots \vee E_n$，则 $CF(E)=\max\{CF(E_1),CF(E_2),\cdots,CF(E_n)\}$。

3. 不确定性的推理计算

不确定性的推理计算是从不确定的初始证据出发，通过运用相关的不确定性知识，最终推出结论并求出结论的可信度值。

1）只有单条知识支持结论时，结论可信度的计算方法

如果支持结论的知识只有一条，且已知证据 E 的可信度 $CF(E)$ 和规则（知识）（IF E THEN H）的可信度为 $CF(H,E)$，则结论 H 的可信度计算公式为

$$CF(H)=CF(H,E)\cdot\max\{0,CF(E)\}$$

若 $CF(E)<0$，即相应证据在某种程度上为假，则 $CF(H)=0$，说明在该模型中没有考虑证据为假时对结论 H 所产生的影响。当证据为真，即 $CF(E)=1$ 时，有 $CF(H)=CF(H,E)$，说明结论 H 的可信度即为规则的可信度 $CF(H,E)$。

2）多条知识支持同一结论时，结论不确定性的合成计算方法

若由多条不同知识推出了相同的结论，但可信度不同，则可用合成算法求出结论的综合可信度。由于对多条知识的综合可通过两两的合成实现，因此下面只考虑两条知识的情况。

设有如下两条知识：

$$\text{IF}\quad E_1\quad \text{THEN}\quad H\quad (CF(H,E_1))$$
$$\text{IF}\quad E_2\quad \text{THEN}\quad H\quad (CF(H,E_2))$$

则结论 H 的综合可信度可先分别应用上述方法计算出 $CF_1(H)$ 和 $CF_2(H)$，再应用以下公式求出 E_1 与 E_2 对 H 的综合影响所形成的可信度 $CF_{1,2}(H)$：

$$CF_{1,2}(H)=\begin{cases}CF_1(H)+CF_2(H)-CF_1(H)CF_2(H) & CF_1(H)\geqslant 0,CF_2(H)\geqslant 0\\ CF_1(H)+CF_2(H)-CF_1(H)CF_2(H) & CF_1(H)<0,CF_2(H)<0\\ \dfrac{CF_1(H)+CF_2(H)}{1-\min\{|CF_1(H)|,|CF_2(H)|\}} & CF_1(H)\text{与}CF_2(H)\text{异号}\end{cases}$$

这实际上是著名的专家系统 MYCIN 中所使用的结论不确定性计算公式。

3）已知结论原始可信度时结论可信度的更新计算方法

前面两个公式都是假设在对结论的初始可信度不知或为 0 的前提下计算结论 H 的可信度的方法。在某些情况下，如果已知证据 E 对结论 H 有影响，且知识（IF E THEN H）的可信度为 $CF(H,E)$，同时结论 H 原来的可信度为 $CF(H)$，那么如何求在证据 E 下结论 H 可信度的更新值 $CF(H/E)$ 呢？即已知规则［IF E THEN H $(CF(H,E))$］及 $CF(H)$，求 $CF(H/E)$。

这时分 3 种情况进行讨论：

（1）当 $CF(E)=1$，即证据肯定出现时。

$$CF(H/E)=\begin{cases}CF(H)+CF(H,E)-CF(H,E)CF(H) & CF(H)\geqslant 0,CF(H,E)\geqslant 0\\ CF(H)+CF(H,E)+CF(H,E)CF(H) & CF(H)<0,CF(H,E)<0\\ \dfrac{CF(H,E)+CF(H)}{1-\min\{|CF(H)|,|CF(H/E)|\}} & CF(H)\text{与}CF(H,E)\text{异号}\end{cases}$$

(2)当 $0<CF(E)<1$ 时。

$$CF(H/E)=\begin{cases}CF(H)+CF(H,E)CF(E)-CF(H)CF(H,E)CF(E) & CF(H)\geqslant 0,CF(H,E)\geqslant 0\\ CF(H)+CF(H,E)CF(E)+CF(H)CF(H,E)CF(E) & CF(H)<0,CF(H,E)<0\\ \dfrac{CF(H,E)CF(E)+CF(H)}{1-\min\{|CF(H)|,|CF(H,E)CF(E)|\}} & CF(H)\text{与}CF(H,E)\text{异号}\end{cases}$$

(3)当 $CF(E)\leqslant 0$ 时,规则 IF E THEN H 不可使用,对结论 H 的可信度无影响。实际上,在 MYCIN 系统中就规定,当 $CF(E)\leqslant 0.2$ 时,规则 IF E THEN H 不可使用。

【例 4-2】 设有如下一组知识:

R1:IF E_1 THEN H (0.9)

R2:IF E_2 THEN H (0.6)

R3:IF E_3 THEN H (−0.5)

R4:IF E_4 AND (E_5 OR E_6) THEN E_1 (0.8)

已知 $CF(E_2)=0.8$,$CF(E_3)=0.6$,$CF(E_4)=0.5$,$CF(E_5)=0.6$,$CF(E_6)=0.8$,求 $CF(H)$。

【解】 由 R4 得

$$\begin{aligned}CF(E_1)&=0.8\times\max\{0,CF(E_4\ \text{AND}\ (E_5\ \text{OR}\ E_6))\}\\&=0.8\times\max\{0,\min\{CF(E_4),CF(E_5\ \text{OR}\ E_6)\}\}\\&=0.8\times\max\{0,\min\{CF(E_4),\max\{CF(E_5),CF(E_6)\}\}\}\\&=0.8\times\max\{0,\min\{CF(E_4),\max\{0.6,0.8\}\}\}\\&=0.8\times\max\{0,\min\{0.5,0.8\}\}\\&=0.8\times\max\{0,0.5\}\\&=0.4\end{aligned}$$

由 R1 得

$$\begin{aligned}CF_1(H)&=CF(H,E_1)\times\max\{0,CF(E_1)\}\\&=0.9\times\max\{0,0.4\}\\&=0.36\end{aligned}$$

由 R2 得

$$\begin{aligned}CF_2(H)&=CF(H,E_2)\times\max\{0,CF(E_2)\}\\&=0.6\times\max\{0,0.8\}\\&=0.48\end{aligned}$$

由 R3 得

$$\begin{aligned}CF_2(H)&=CF(H,E_3)\times\max\{0,CF(E_3)\}\\&=-0.5\times\max\{0,0.6\}\\&=-0.3\end{aligned}$$

根据结论不确定性的合成算法得

$$\begin{aligned}CF_{1,2}(H)&=CF_1(H)+CF_2(H)-CF_1(H)CF_2(H)\\&=0.36+0.48-0.36\times0.48\\&=0.84-0.17\\&=0.67\end{aligned}$$

$$\begin{aligned}CF_{1,2,3}(H)&=\frac{CF_{1,2}(H)+CF_3(H)}{1-\min\{|CF_{1,2}(H)|,|CF_3(H)\}}\\&=(0.67-0.30)\div(1-\min\{0.67,0.3\})\\&=0.37\div(1-0.3)\\&=0.37\div0.7\\&=0.53\end{aligned}$$

因此，所求的综合可信度 $CF(H)=0.53$。

4.2.4 证据理论

证据理论又称 D-S 理论，在该理论中，知识是用产生式的形式表示的，而证据和结论则是以集合的形式表示的。知识的不确定性通过一个集合形式的可信度因子来表示，而证据和结论的不确定性度量则采用信任函数和似然函数来表示。为此，引入概率分配函数、信任函数和似然函数的概念。

1. 概率分配函数、信任函数和似然函数的概念

概率分配函数的作用是把样本空间 D 上的任意一个子集 $A(\in 2^D)$ 都映射为 $[0,1]$ 上的一个数 $M(A)$。当 $A(\in 2^D)$ 对应一个命题时，$M(A)$ 即是对相应命题不确定性的度量。如果定义函数 $M(x)$ 为集合 2^D 到区间 $[0,1]$ 上的一个映射函数，其满足下列条件：

$$M(\varphi)=0$$

$$\sum_{A\subseteq D}M(A)=1$$

则称 $M(x)$ 为 2^D 上的概率分配函数。$M(A)$ 称为命题 A 的基本概率数。

信任函数是用来对命题 A 的不确定性进行度量的。如果定义函数 $Bel(x)$ 为将集合 2^D 映射到区间 $[0,1]$ 上的一个函数，即 $0\leqslant Bel(A)\leqslant 1$，并且满足下列条件：

$$Bel(A)=\sum_{B\subseteq A}M(B)\quad(\text{对所有的 }A\in 2^D)$$

则称 $Bel(x)$ 为信任函数，或称下限函数。

似然函数也是一个从集合 2^D 到区间 $[0,1]$ 的映射函数，也将其称为上限函数。设函数 $Pl(x)$ 是从集合 2^D 到区间 $[0,1]$ 的映射函数，且

$$Pl(A)=1-Bel(\neg A)=\sum_{B\cap A\neq\varnothing}M(B)\quad(\text{对所有的 }A\in 2^D)$$

则称 $Pl(x)$ 为似然函数。

显然，A 不为假，并不代表 A 一定为真，也就是说对 A 不为假的信任程度应该大于对 A 为真的信任程度，即有

$$Pl(A)\geqslant Bel(A)$$

那么，$Pl(A)-Bel(A)$ 就是对 A 既不为假又不为真的信任程度。或者说表示了既不

信任 A 也不信任 $\neg A$ 的一种度量。这种既不为假、又不为真的情况，就是前面所说的“不知道”的情况。证据理论就能处理这种“不知道”引起的不确定性。

命题的不确定性需要用信任函数和似然函数来度量，而信任函数和似然函数的定义又依赖于概率分配函数，所以，概率分配函数是对一个命题的不确定性度量的基础。然而在有的情况下，对同样的证据，由于数据的来源不同，会得到两个不同的概率分配函数。为了计算信任函数和似然函数，就必须将两个概率分配函数合并成一个概率分配函数。组合的方法就是求概率分配函数的正交和。

2. 特定概率分配函数

推理模型是建立在概率分配函数的基础之上的，所选取的概率分配函数的复杂性直接影响着推理模型的复杂性，进而影响着不确定性计算的复杂性。这里给出一个具体的概率分配函数，并做了一些约束限制，以便简化不确定性的推理模型。

设样本空间 $D=\{S_1,S_2,\cdots,S_n\}$，领域内的命题都用 D 的子集表示，则定义 2^D 上的概率分配函数 $M(x)$ 满足如下条件：

(1) $M(\{S_i\})\geqslant 0$（对任何 $S_i\in 2^D$）。

(2) $\sum_{i=1}^{m} M(\{S_i\})\leqslant 1$。

(3) $M(D)=1-\sum_{i=1}^{m} M(\{S_i\})$。

(4) 当 $A\subset D$ 且 $|A|>1$ 或 $|A|=0$ 时，$M(A)=0$。

其中，$|A|$ 表示命题 A 对应的集合中所包含的元素的个数。

在这一特定的概率分配函数中定义了只有单个元素构成的子集和样本空间 D 本身的基本概率数才有可能大于 0，其他子集的基本概率数均为 0。

3. 基于特定概率分配函数的不确定性推理模型

在证据理论中，信任函数 Bel(A) 和似然函数 Pl(A) 分别表示对命题 A 的信任程度的下限和上限。所以，证据及结论的不确定性以及不确定性知识的规则强度都可以由区间 (Bel(A)，Pl(A)) 内的某个数值来进行度量。为了推理计算的方便，可以利用 Bel(A) 和 Pl(A) 来构造一个函数，并以它来度量命题的不确定性，该函数的值应落在区间 (Bel(A)，Pl(A)) 内。由于 Bel(A) 是下限，所以该函数可以定义如下：

$$f(A)=\text{Bel}(A)+\frac{|A|}{|D|}\cdot[\text{Pl}(A)-\text{Bel}(A)]$$

其中，$|A|$ 和 $|D|$ 分别表示 A 和 D 中元素的个数。由于该函数是由信任函数构造而来的，故把它称为信任度函数。

4.2.5 模糊推理

许多命题在人工智能、生态系统、计算机、自动控制等实例中反映的是不精确、不确定、不完备的信息，推理利用的也是不精确、不确定、不完备的知识或规则，无法直接使用经典的推理模式得到真或假的结论。于是，必须仅凭借经验和不完备信息进行近似推理

或不确定性推理。这种基于模糊数学方法处理由模糊性引起的不确定性推理称为模糊推理。

1. 模糊命题

在现实生活中，人们常常遇到诸如“她很漂亮”“电动机的转速稍偏高”“主回路电流太大”这样的陈述句，其特点是：含有模糊概念，如“漂亮”“稍偏高”“太大”等对应着所讨论的论域上的某个模糊集合；无法像二值逻辑命题那样明确地判断“真”与“假”，判断的结果处于真假之间的模棱两可状态。也就是说，它们与二值逻辑中的命题相仿，但具有一定的模糊性。一般地，称这种含有模糊概念的陈述句为模糊命题。

虽然模糊命题不能像二值逻辑命题那样单纯地判断“真”和“假”，但人们仍然希望对模糊命题的判断进行度量。为此，利用模糊集合的隶属函数将二值逻辑命题的真值由{0,1}推广到[0,1]，用0和1之间的连续值来度量模糊命题的“真伪程度”。

设U为模糊命题的集合，令$T:U\to[0,1]$，使得$T(P)=\alpha\in[0,1]$，$\forall P\in U$，则称$T(P)$为模糊命题P的真值。具体地，若模糊命题$P\in U$的形式为“P：x is A”，其中x是变量，A是某个模糊概念对应的模糊集合，则模糊命题P的真值取值为变量x对模糊集合A的隶属度，即$T(P)=A(x)$。

当$A(x)=1$时，模糊命题P为全真；当$A(x)=0$时，模糊命题P为全假；当$T(P)=A(x)$介于0和1之间时，$T(P)$表征模糊命题真假的程度：越接近于1，真的程度越大，越接近于0，假的程度越大。

显然，模糊命题的取值虽不是单纯的“真”和“假”，但它却反映了真或假的程度，比二值逻辑中的命题更能符合人脑的思维。例如，上述的模糊命题“她很漂亮”“电动机的转速稍偏高”“主回路电流太大”，若用“漂亮”“稍偏高”“太大”的程度来描述其真伪程度，将更为贴切合理，也更为深刻。

与二值逻辑命题类似，形如“P：x is A”的模糊命题称为简单模糊命题或原子模糊命题。用命题联结词联结起来的模糊命题称为复合模糊命题，用联结词联结简单模糊命题的过程称为模糊命题的逻辑演算。复合模糊命题就是某些模糊命题逻辑演算的结果。

同样地，由命题联结词“析取$\vee$”联结起来的复合模糊命题“$P\vee Q$”称为选言模糊命题；由命题联结词“合取$\wedge$”联结起来的复合模糊命题“$P\wedge Q$”称为联言模糊命题；由命题联结词“蕴含$\to$”联结起来的复合模糊命题“$P\to Q$”称为假言模糊命题或条件模糊命题。

2. 模糊语言定义

1)定义

模糊语言是带有模糊性的语言。如“妈妈很年轻”“今天天气不错”等。其定义如下：

由4个参数U、T、E、N描述的系统，即

$$L=(U,T,E,N)$$

其中，

(1)U 为语言主体的全体，即论域。

(2)T 为词或项的模糊集合，称为项集合，分为原子词与合成词。原子词如“人”“狗”“黑”“快”“美丽”等不可再分解；合成词如“红花”，可分解成“红”“花”两个原子词。

(3)E 为名词记号间的连接总和，称其为对 T 的嵌入集合。T 是 E 的模糊子集；E 对 T 有 $\mu_T: E \to [0,1]$，即词 $x(x \in E)$ 对 T 的隶属函数 $\mu_T(x)$ 定义在闭区间 $[0,1]$ 上。

(4)N 为 E 对 U 的模糊关系，称其为命名关系，有 $\mu_N: E \times U \to [0,1]$，即隶属函数 $\mu_N(x,y) \in [0,1]$ 是 $x \in E, y \in U$ 两个变量的函数。

例如，设 x 为单词“高个子”，y 为成年男子的身高(cm)，则有

$$\mu_N(\text{高个},140)=0.1$$
$$\mu_N(\text{高个},167)=0.5$$
$$\mu_N(\text{高个},172)=0.8$$
$$\mu_N(\text{高个},190)=1$$

2)语言变量

以自然语言中的字或句，而不是以数为值的变量，如年龄、大小、高低、快慢等。语言变量由一个五元体$(N,U,T(N),G,M)$来表征，其中的量的具体含义如下：

(1)N 是语言变量名称，如年龄、大小等。

(2)U 是 N 的论域。

(3)$T(N)$是语言变量值 X 的集合，其中每个 X 都是论域 U 上的模糊集合。例如：

$$\begin{aligned} T(N) &= T(\text{年龄}) \\ &= \text{“很瘦”}+\text{“瘦”}+\text{“标准”}+\text{“微胖”}+\text{“很胖”} \\ &= X_1+X_2+X_3+X_4+X_5 \end{aligned}$$

(4)G 是语法规则，用于产生语言变量 N 的值 X 的名称，研究原子词构成合成词后词义的变化，并求取其隶属函数。例如：

否定词“非”的隶属函数：$\mu_{\bar{A}}=1-\mu_{\tilde{A}}$。

联结词“或”的隶属函数：$\mu_{\tilde{A}\cup\tilde{B}}=\mu_{\tilde{A}} \vee \mu_{\tilde{B}}$。

联结词“与”的隶属函数：$\mu_{\tilde{A}\cap\tilde{B}}=\mu_{\tilde{A}} \wedge \mu_{\tilde{B}}$。

修饰词“极”“非常”“相当”“比较”“略”“稍微”的隶属函数：$\mu_{\text{极}\tilde{A}}=\mu_{\tilde{A}}^4$、$\mu_{\text{非常}\tilde{A}}=\mu_{\tilde{A}}^2$、$\mu_{\text{相当}\tilde{A}}=\mu_{\tilde{A}}^{1.25}$、$\mu_{\text{比较}\tilde{A}}=\mu_{\tilde{A}}^{0.75}$、$\mu_{\text{略}\tilde{A}}=\mu_{\tilde{A}}^{0.5}$、$\mu_{\text{稍微}\tilde{A}}=\mu_{\tilde{A}}^{0.25}$。

上述加重或减弱语气的词可视为一种模糊算子，其中“极”“非常”“相当”称为集中化算子，“比较”“略”“稍微”称为散漫化算子，二者统称为语气算子。

例如，查德在论域 $U=[0,100]$ 岁内给出了“年龄”的语言变量值“老”的模糊子集隶属函数为

$$\mu_{\tilde{\text{老}}}(x)=\begin{cases} 0 & x<50 \\ \dfrac{1}{1+\left(\dfrac{x-50}{5}\right)^{-2}} & x \geqslant 50 \end{cases}$$

现以 60 岁为例，通过隶属函数分别计算它属于“极胖”“非常胖”“相当胖”“比较胖”

“略胖”“稍微胖”的程度如下：

$$\mu_{\widetilde{极胖}}(60)=[\mu_{\widetilde{胖}}(60)]^4=0.8^4=0.410$$

$$\mu_{\widetilde{非常胖}}(60)=[\mu_{\widetilde{胖}}(60)]^2=0.8^2=0.64$$

$$\mu_{\widetilde{相当胖}}(60)=[\mu_{\widetilde{胖}}(60)]^{1.25}=0.8^{1.25}=0.757$$

$$\mu_{\widetilde{比较胖}}(60)=[\mu_{\widetilde{胖}}(60)]^{0.75}=0.8^{0.75}=0.846$$

$$\mu_{\widetilde{略胖}}(60)=[\mu_{\widetilde{胖}}(60)]^{0.5}=0.8^{0.5}=0.894$$

$$\mu_{\widetilde{稍微胖}}(60)=[\mu_{\widetilde{胖}}(60)]^{0.25}=0.8^{0.25}=0.946$$

(5)M 是语义规则，根据语义规则给出模糊子集 X 的隶属函数。

3. 模糊语句

1)模糊直言语句

句型：“$\widetilde{A}$ 是 $\widetilde{A}_1$”。

$\widetilde{A}$ 是对象的名称，$\widetilde{A}_1$ 是论域 U 上的一个模糊子集。

例如，“$\widetilde{A}$ 是非常瘦”是一个模糊直言语句，其中模糊子集“非常瘦”可由论域 $U=\{1,2,3,4,5\}$ 上的模糊子集$\widetilde{瘦}=\frac{1}{1}+\frac{0.8}{2}+\frac{0.6}{3}+\frac{0.4}{4}+\frac{0.2}{5}$求得，即

$$\widetilde{非常瘦}=(\widetilde{瘦})^2=\widetilde{瘦}=\frac{1}{1}+\frac{0.64}{2}+\frac{0.36}{3}+\frac{0.16}{4}+\frac{0.04}{5}$$

2)模糊条件语句

模糊条件语句常用句型如下：

(1)若为“$\widetilde{A}$ 则 $\widetilde{B}$”型，则记为 IF $\widetilde{A}$ THEN $\widetilde{B}$。$\widetilde{A}$ 和 $\widetilde{B}$ 为不同论域上的模糊集合。

(2)若为“$\widetilde{A}$ 则 $\widetilde{B}$ 否则 $\widetilde{C}$”型，则记为 IF $\widetilde{A}$ THEN $\widetilde{B}$ ELSE $\widetilde{C}$。

(3)若为“$\widetilde{A}$ 且 $\widetilde{B}$ 则 $\widetilde{C}$”型，则记为 IF $\widetilde{A}$ AND $\widetilde{B}$ THEN $\widetilde{C}$。其反映双输入单输出的一种控制策略。

(4)复杂控制策略。

①双输入双输出。若为“$\widetilde{A}$ 且 $\widetilde{B}$ 则 $\widetilde{C}$ 则 $\widetilde{D}$”型，则记为 IF $\widetilde{A}$ AND $\widetilde{B}$ THEN $\widetilde{C}$ ELSE $\widetilde{D}$。

②双输入三输出。若为“$\widetilde{A}$ 且 $\widetilde{B}$ 则 $\widetilde{C}$ 则 $\widetilde{D}$ 则 $\widetilde{E}$”型，则记为 IF $\widetilde{A}$ AND $\widetilde{B}$ THEN $\widetilde{C}$ ELSE $\widetilde{D}$ ELSE $\widetilde{E}$。

4. 模糊推理方式

1)假言推理

假言推理属于演绎推理，是模糊推理最常用的一种方法。

基本规则：若 A，则 B；

如今 A；

结论 B。

例如，A 为“奶奶住院”，B 为“奶奶生病”，如今“奶奶住院”为真，结论“奶奶生病”也真。

2)模糊假言推理

上述命题中的 A、B 是指精确事件，若在模糊情况下，则 $\widetilde{A}$ 和 $\widetilde{B}$ 为模糊命题，代表模

糊事件，不能用传统的形式逻辑中的假言推理方法进行推理，查德提出了以下近似推理理论：

若为“$\widetilde{A}$ 则 $\widetilde{B}$”型，则记为 IF $\widetilde{A}$ THEN $\widetilde{B}$。

设 X 和 Y 是两个各自具有基础变量 x 和 y 的论域，其中模糊集合 $\widetilde{A}\in X$ 和 $\widetilde{B}\in Y$ 的隶属函数分别为 $\mu_{\widetilde{A}}(x)$ 及 $\mu_{\widetilde{B}}(y)$。又设 $\widetilde{R}_{\widetilde{A}\to\widetilde{B}}$ 是 $X\times Y$ 论域上描述模糊条件语句“若 $\widetilde{A}$ 则 $\widetilde{B}$”的模糊关系，其隶属函数为

$$\mu_{\widetilde{A}\to\widetilde{B}}(x,y)=[\mu_{\widetilde{A}}(x)\wedge\mu_{\widetilde{B}}(y)]\vee[1-\mu_{\widetilde{A}}(x)]$$

模糊关系 $\widetilde{R}_{\widetilde{A}\to\widetilde{B}}$ 可写成

$$\widetilde{R}_{\widetilde{A}\to\widetilde{B}}=[\widetilde{A}\cdot\widetilde{B}]\cup[\overline{\widetilde{A}}\cdot E]$$

其中，E 为代表全域的全称矩阵。

近似推理情况下的假言推理逻辑结构为

若 $\widetilde{A}$ 则 $\widetilde{B}$；

如今 $\widetilde{A}_1$；

结论 $\widetilde{B}_1=\widetilde{A}_1\circ\widetilde{R}_{\widetilde{A}\to\widetilde{B}}$。

其中，$\widetilde{B}_1=\widetilde{A}_1\circ\widetilde{R}_{\widetilde{A}\to\widetilde{B}}$ 表征合成推理规则，算符“$\circ$”代表合成运算。推理合成规则是假言推理的近似推广。

【例 4-3】 设论域 $X=a_1+a_2+a_3+a_4+a_5$ 和 $Y=b_1+b_2+b_3+b_4+b_5$ 上的模糊子集 $\widetilde{A}=\widetilde{小}=\dfrac{1}{a_1}+\dfrac{0.5}{a_2}$ 及 $\widetilde{B}=\widetilde{大}=\dfrac{1}{b_4}+\dfrac{1}{b_5}$，$X\times Y$ 上的模糊关系为“若 x 小，则 y 大”(若 $\widetilde{A}$ 则 $\widetilde{B}$)。求通过模糊假言推理确定，与“x 为较小”$\left(\text{与模糊集合 }\widetilde{A}_1=\widetilde{较小}=\dfrac{1}{a_1}+\dfrac{0.4}{a_2}+\dfrac{0.2}{a_3}\right)$ 对应的模糊集合 $\widetilde{B}_1$。

【解】 (1)计算模糊关系 $\widetilde{R}_{\widetilde{小}\to\widetilde{大}}$，即

$$\widetilde{R}_{\widetilde{小}\to\widetilde{大}}=[\widetilde{A}\times\widetilde{B}]\cup[\overline{\widetilde{A}}\times E]=\begin{bmatrix}1\\0.5\\0\\0\\0\end{bmatrix}[0\quad 0\quad 0\quad 0.5\quad 1]\vee\begin{bmatrix}0\\0.5\\1\\1\\1\end{bmatrix}[1\quad 1\quad 1\quad 1\quad 1]$$

$$=\begin{bmatrix}0&0&0&0.5&1\\0&0&0&0.5&0.5\\0&0&0&0&0\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}\vee\begin{bmatrix}0&0&0&0&0\\0.5&0.5&0.5&0.5&0.5\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

$$=\begin{bmatrix}0&0&0&0.5&1\\0.5&0.5&0.5&0.5&0.5\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

(2)由已知 $\widetilde{A}_1$ 及模糊关系 $\widetilde{R}_{\widetilde{A}\to\widetilde{B}}$ 的合成运算,求 $\widetilde{B}_1$。

$$\widetilde{B}_1=\widetilde{A}_1\circ\widetilde{R}_{\widetilde{\pi}\to\widetilde{\mathrm{大}}}=[1\quad 0.4\quad 0.2\quad 0\quad 0]\circ\begin{bmatrix}0&0&0&0.5&1\\0.5&0.5&0.5&0.5&0.5\\1&1&1&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix}$$

$$=[0.4\quad 0.4\quad 0.4\quad 0.5\quad 1]$$

与$\widetilde{\mathrm{大}}=[0\quad 0\quad 0\quad 0.5\quad 1]$相比较,可得出 $\widetilde{B}_1=\widetilde{\text{较大}}$。

3)模糊条件推理

(1)若为"$\widetilde{A}$ 则 $\widetilde{B}$"型,则模糊条件语句"IF $\widetilde{A}$ THEN $\widetilde{B}$"的模糊条件推理即为上述推理过程。

(2)若为"$\widetilde{A}$ 则 $\widetilde{B}$ 否则 $\widetilde{C}$"型,则模糊条件语句"IF $\widetilde{A}$ THEN $\widetilde{B}$ ELSE $\widetilde{C}$"的模糊条件推理如下:设 $\widetilde{A}$ 是论域 X 上的模糊子集,$\widetilde{B}$ 及 $\widetilde{C}$ 是论域 Y 上的模糊子集,则"IF $\widetilde{A}$ THEN $\widetilde{B}$ ELSE $\widetilde{C}$"在论域 $X\times Y$ 上的模糊关系 $\widetilde{R}$ 为

$$\widetilde{R}=[\widetilde{A}\times\widetilde{B}]\cup[\overline{\widetilde{A}}\times\widetilde{C}]$$

基于推理合成规则,根据模糊关系 $\widetilde{R}$ 求得与已知模糊集合 $\widetilde{A}_1$ 对应的模糊集合 $\widetilde{B}_1$ 为

$$\widetilde{B}_1=\widetilde{A}_1\circ\widetilde{R}$$

(3)若为"$\widetilde{A}$ 且 $\widetilde{B}$ 则 $\widetilde{C}$"型,则模糊条件语句"IF $\widetilde{A}$ AND $\widetilde{B}$ THEN $\widetilde{C}$"的模糊条件推理如下:设 $\widetilde{A}$,$\widetilde{B}$ 和 $\widetilde{C}$ 分别是论域 X,Y 和 Z 上的模糊子集(一般 $\widetilde{A}$ 和 $\widetilde{B}$ 是模糊控制器的输入模糊集合,$\widetilde{C}$ 是其输出模糊集合),双输入单输出系统如图 4-1 所示。

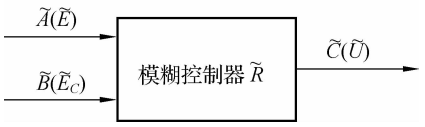

图 4-1　双输入单输出系统

例如,$\widetilde{A}$ 是过程误差信号论域上的模糊子集 $\widetilde{E}$,$\widetilde{B}$ 是其误差变化率信号论域上的模糊子集 $\widetilde{E}_C$,$\widetilde{C}$ 是其模糊控制器输出信号论域上的模糊子集 $\widetilde{U}$,则"IF $\widetilde{A}$ AND $\widetilde{B}$ THEN $\widetilde{C}$"所决定的为三元模糊关系 $\widetilde{R}$,即

$$\widetilde{R}=[\widetilde{A}\times\widetilde{B}]^{T_1}\times\widetilde{C}$$

其中,$[\widetilde{A}\times\widetilde{B}]^{T_1}$ 为由模糊关系矩阵$[\widetilde{A}\times\widetilde{B}]_{n\times m}$构成的 nm 维列向量,n 和 m 分别为模糊集合 $\widetilde{A}_1$ 和 $\widetilde{B}_1$ 的论域元素数目。

基于推理合成规则,根据模糊关系 $\widetilde{R}$ 求得与给定输入模糊集合 $\widetilde{A}_1$ 和 $\widetilde{B}_1$ 对应的输出模糊集合 $\widetilde{C}_1$,即

$$\widetilde{C}_1=[\widetilde{A}_1\times\widetilde{B}_1]^{T_2}\circ\widetilde{R}$$

其中，$[\widetilde{A}_1\times\widetilde{B}_1]^{T_2}$ 为由模糊关系矩阵 $[\widetilde{A}_1\times\widetilde{B}_1]_{n\times m}$ 构成的 nm 维行向量。

注意：T_1 和 T_2 分别是将二维矩阵 $[\widetilde{A}\times\widetilde{B}]$ 变成一维列向量、行向量的标记符号。

【例 4-4】 设论域 $X=\{a_1,a_2,a_3\}$，$Y=\{b_1,b_2,b_3\}$，$Z=\{c_1,c_2\}$，已知模糊集合：

$$\widetilde{A}=\frac{0.5}{a_1}+\frac{1}{a_2}+\frac{0.1}{a_3},\widetilde{A}\in X$$

$$\widetilde{B}=\frac{0.1}{b_1}+\frac{1}{b_2}+\frac{0.6}{b_3},\widetilde{B}\in Y$$

$$\widetilde{C}=\frac{0.4}{c_1}+\frac{1}{c_2},\widetilde{C}\in Z$$

试确定模糊条件语句“IF $\widetilde{A}$ AND $\widetilde{B}$ THEN $\widetilde{C}$”所决定的模糊关系 $\widetilde{R}$，并计算由给定输入模糊集合 $\widetilde{A}_1=\frac{1}{a_1}+\frac{0.5}{a_2}+\frac{0.1}{a_3}$ 和 $\widetilde{B}_1=\frac{0.1}{b_1}+\frac{0.5}{b_2}+\frac{1}{b_3}$ 决定的输出模糊集合 $\widetilde{C}_1$。

【解】 计算 $\widetilde{A}$ 与 $\widetilde{B}$ 的笛卡尔积(取小原则)。

$$(\widetilde{A}\times\widetilde{B})^{T_1}=\begin{bmatrix}0.1\\0.5\\0.5\\0.1\\1\\0.6\\0.1\\0.1\\0.1\end{bmatrix}_{9\times1}$$

由 $(\widetilde{A}\times\widetilde{B})^{T_1}$ 和 $\widetilde{C}$ 计算模糊关系 $\widetilde{R}$。

$$\widetilde{R}=[\widetilde{A}\times\widetilde{B}]^{T_1}\times\widetilde{C}=\begin{bmatrix}0.1\\0.5\\0.5\\0.1\\1\\0.6\\0.1\\0.1\\0.1\end{bmatrix}[0.4\quad 1]=\begin{bmatrix}0.1&0.1\\0.4&0.5\\0.4&0.5\\0.1&0.1\\0.4&1\\0.4&0.6\\0.1&0.1\\0.1&0.1\\0.1&0.1\end{bmatrix}$$

基于推理合成规则，求取 $\widetilde{C}_1$。

将 $\widetilde{A}_1\times\widetilde{B}_1$ 写成行向量：

$$[\widetilde{A}_1\times\widetilde{B}_1]^{T_2}=[0.1\quad 0.5\quad 1\quad 0.1\quad 0.5\quad 0.5\quad 0.1\quad 0.1\quad 0.1]_{1\times9}$$

由 $(\widetilde{A}_1\times\widetilde{B}_1)^{T_2}$ 和模糊关系 $\widetilde{R}$ 计算输出模糊集合 $\widetilde{C}_1$。

$$\tilde{C}_1=[\tilde{A}_1\times\tilde{B}_1]^{T_2}\circ\tilde{R}=\begin{bmatrix}0.1 & 0.5 & 1 & 0.1 & 0.5 & 0.5 & 0.1 & 0.1 & 0.1\end{bmatrix}\begin{bmatrix}0.1 & 0.1\\ 0.4 & 0.5\\ 0.4 & 0.5\\ 0.1 & 0.1\\ 0.4 & 1\\ 0.4 & 0.6\\ 0.1 & 0.1\\ 0.1 & 0.1\\ 0.1 & 0.1\end{bmatrix}$$

$$=[0.4 \quad 0.5]$$

即

$$\tilde{C}_1=\frac{0.4}{c_1}+\frac{0.5}{c_2}$$

习　题　4

1. 什么是不确定性推理？不确定性推理中需要解决的基本问题是什么？
2. 主观 Bayes 方法中知识的不确定性是如何表示的？请说明其中 LS 与 LN 的意义。
3. 什么是可信度？可信度方法中证据不确定是如何表示的？
4. 模糊语言定义中有哪几个主要参数？各代表什么？
5. 模糊推理有几种方式？