第二部分　状态空间搜索

人无远虑，必有近忧。

——孔子

……[蚂蚁]知道必须采取一定的方案，但是它不知道如何去做。就像一个一只手拿着茶杯另一只手拿着三明治的人，想用一根火柴点燃一根香烟。这个人会想出一个办法，在拿起香烟和火柴前，先放下茶杯和三明治。而蚂蚁将会放下三明治，拿起火柴，然后放下火柴拿起香烟，又放下香烟拿起三明治，再放下茶杯拿起香烟，直到最后它放下三明治拿起火柴。这种方式倾向于依靠一系列的事件来达到目标，是一种不带任何思考的愚蠢做法……， Wart非常惊奇地观察了这个过程，它变得很着急，直到厌倦。他想问蚂蚁为什么不事先考虑一下呢……。

——T. H. White《The Once and Future King》，第13章

第7章　能计划的agent

7.1　存储与计算

响应型agent(图2-2、图5-1和图5-3)的动作功能几乎没有做任何计算。从本质上讲，这些agent执行的动作或者由它们的设计者、或者通过学习、或者通过演化过程、或者是由以上几方面的组合而选择给它们的。这些动作能够通过表、产生规则描述给定特征向量动作的组合逻辑电路来实现。在计算机科学中，这种实现倾向于经典的时空权衡的“空间”一方。它们是基于空间或存储的实现——对设计者知识的汇编。

一个能在复杂环境下执行复杂任务的反应型机器需要大量（也许是无法计算的）的存储。而且，这样一个反应型机器的设计者需要有超人类的预见能力，要为该机器能遇到的所有可能情况预期一个合适的反应。这启发我们可以考虑用时间换取空间，用适应性代替显式的设计。首先，考虑反应型机器设计者必须做的一些计算的动作函数。这些计算当然会需要时间，但是它们将减少agent的存储要求和设计者的负担。

我们要考虑的一些计算是推测在任何给定的情况下，某些动作的可能结果。确实，一个有能力的反应型机器设计者必须把它的设计构架在这些预期结果的基础上，设计者（或者是进化，或是学习过程）必须指定这些计算是什么，但是计算（在需要它们时）程序通常比计算出结果所需要的空间要少得多。而且指定计算比作出所有可能情况的结果对设计者来说也更容易。也许最重要的是如果预期计算的结果能被自动地学习或演化，那么使用该结果的 agent就能在那些连设计者都可能无法预见的情况下，也可以选择合适的动作来执行。

为了推测一个动作的结果，一个 agent必须有一个自身所处环境的模型和一些结果模型，这些结果模型是agent对其环境模型的动作结果。因此，真正的动作只有在模拟环境是安全和有效时才会发生作用。

7.2 状态空间图

作为一个例子，让我们考虑一个有*A*、*B*、*C*三个玩具积木的网格空间，开始时，三个积木都在地板上。假如机器人的任务是把它们堆起来以便*A*在*B* 的上面，*B*在*C*的上面，*C*在地板上。当然，对我们来说采取什么动作是很明显的，但对机器人来说就不一样了。假定机器人能够对其每一个动作对环境的建模结果，它可以通过一对环境模型 —— 一个代表动作执行前的环境状态，另一个代表动作执行后的环境状态 —— 来建模。为了达到此目的，假设机器人能够把其上没有任何其他积木的积木 *x*移到另一个地方*y*，*y*或是地板或是其上没有其他积木的积木。可以通过一个模式的实例对这些动作建模，该模式表示为 `move (x, y)`，其中*x*可以是*A*、*B*或C中的任何一个，*y*可以是*A*、*B*、*C*和地板中的任何一个。我们也知道这个方法中的一些实例（如`move(A,A)`）是不可执行的动作 。这个方法的实例，如`move(A,C)`被称为算子(operator)。因此，算子是动作的模型。

用列表结构图标模型，可表示所有积木都在地面时能采用的所有动作的模型，参见图 7-1[⊖]。（为了清楚起见，每种情况包括一个图示化的略图。图画对人是直观的，但是对处理列表的agent，列表可能更直观）。在这些结果中的（(*AB*)（*C*)）和（(*A*)（*BC*)）在某些方面似乎比其他更接近于我们的目标（(*ABC*)）。因此，仅考虑单个动作的预期结果，机器人可能宁愿执行动作`move(A,B)`和`move(B,C)`。当然，机器人还没有采取真正的动作。

在一个模拟环境中，只向前看一步常常就能产生有用的预期效果，但是多看几步，也许直到任务完成的所有步骤都看到后就会发现一些捷径，从而避免走弯路。跟踪几个可选动作序列结果的最有用的结构是有向图。一个 agent通过它的动作产生的环境集合能用一个有向图表示。有向图的节点代表每个环境，弧代表算子（这里先使用一些不规范的图论术语；在本章的后面，将介绍一些我们要用到的图论概念的规范定义）。节点表示可以是基于图标的或基于特征的，尽管两种类型都会考虑，但先从图标表示开始。

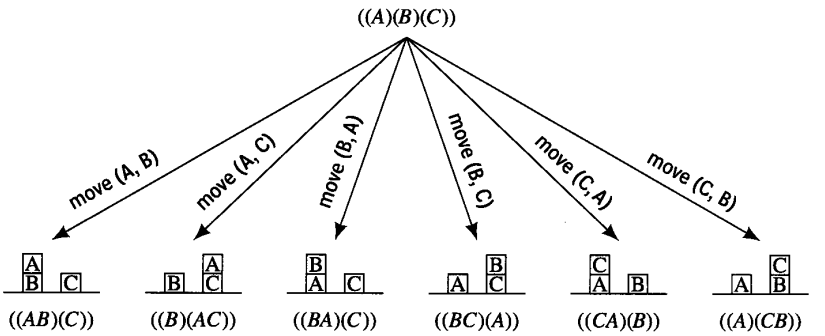


图7-1 移动一个积木的效果

如果大量可区分的环境状态足够小，那么一个代表所有可能动作和状态的图就能被显式地存储。例如，图7-2显示了所有的状态和相关移动来操作三个积木。这种环境模型和动作图被称为状态空间图(*state-space graph*)。需要注意的是每一个动作都是可逆的，为了使图不太混乱，仅仅标出了每对移动中的一个。从图中可以容易地看到，如果初始状态是（(*A*)（*B*)（*C*)），机器人的任务是获得（(*ABC*)）的状态，那么它应该执行{`move(B,C),move(A,B)`}的动作序列。

⊖ 为简单起见，继续使用一种抽象的表示，在这里各个积木的水平位置是不相关的。因此，它们没被表示出来。

图结构表示可能世界的优点之一是图中的任何节点能代表一个目标状态——也许是由诸如人为的一些外部源指定的目标。这种任务灵活性可以和我们前面学习过的简单目的的 agent形成对照。为了发现到达指定目标的一组动作，机器人只要能在图中发现一条代表初始状态节点到目标节点的路径就可以了。然后就能从该路径边上的标签读出到达目标的动作。例如，搭积木的机器人的任务是到达任务（(*CBA*)），如果初始状态是（(*ABC*)），动作序列将会是{move(A, floor), move(B,A), move(C, B)}。从图7-2中，可以通过视觉容易地找到路径，然而，为了发现路径，计算型agent要用各种图搜索过程。

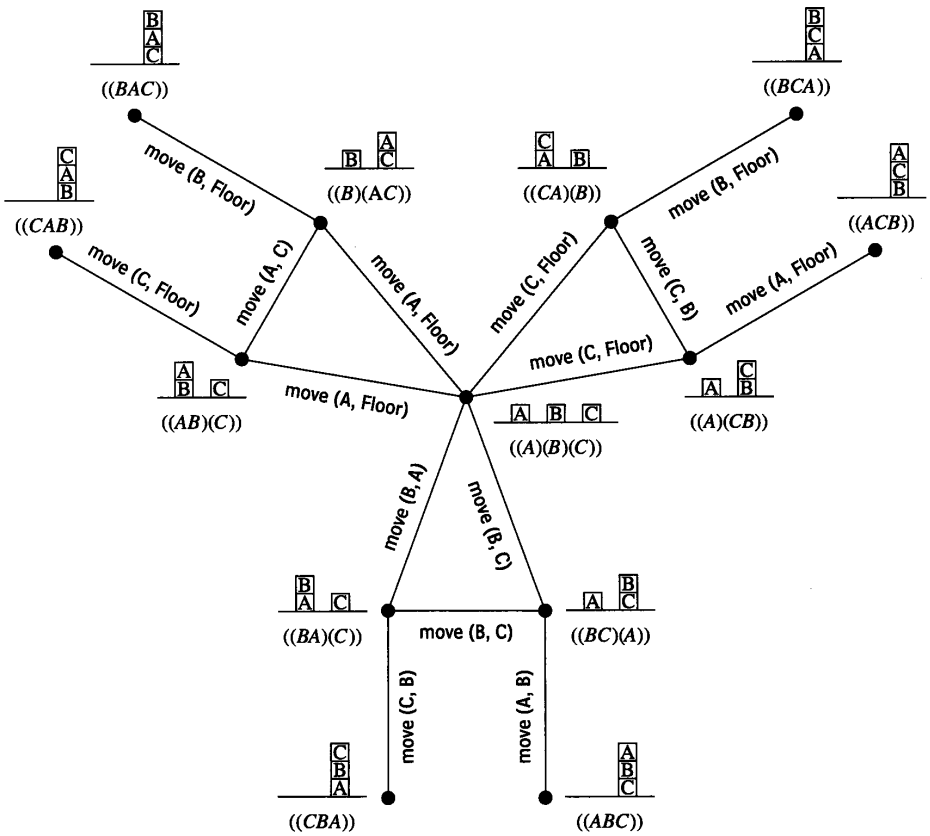


图7-2 状态空间图

顺着路径到达目标的所有弧的算子可以组合成称为一个序列的计划。搜索这个序列的过程称为规划。这种从一系列动作结果得到的世界状态的预测过程称为规划方案。当然，在这种方式下，为了获得目标而执行的一系列动作需要依靠若干假设。 agent必须能在图节点中表示所有相关的环境状态，它必须有在一对节点间如何动作的精确模型。动作必须总有其模型化的结果——也就是说，在agent的操纵系统中不能有错误或不确定性。 agent的感知系统必须精确地指定开始节点，并且没有任何其他的agent或动态过程会改变环境。如果所有的这些假设满足，且搜索到目标状态的时间允许，就能规划，并执行一个完整序列的动作（就像弹道学上的一样），不需要任何环境的信息反馈。

尽管这些假设在大多数实际应用中不能满足，图搜索计划还是一个相当有用和重要的思想。它既可以被一般化，以适应更少约束假设的设置，也能被作为一个适应这种设置的嵌入结构部

件。第10章会谈到这个主题。在本章中，通过描述一个相当简单的搜索过程来开始对图进行搜索处理，这适合于被搜索图能被显式存储的情况。在这种情况下，图一般都很小，因此不必太关心搜索方法的效率。在下面的两章中，将讨论应用到很大的图的图搜索方法，这些图只能被隐式地表示，而且必须被高效地搜索。

7.3 显式状态空间搜索

显式图搜索方法涉及到在图节点上传播“标记”。我们把开始节点标记为0,然后顺着图的边，连续传播更大的整数直至遇到目标节点。然后，顺着数字下降序列从目标点回溯到开始节点。顺着开始点到目标点路径上的动作序列就是获得目标应该采取的动作。这种方法需要 $O(n)$ 步，n是图中的节点数目（如果只有一个目标节点，这个过程也可以从相反方向实现——从目标节点开始，直到某个整数遇到了开始节点）。搜索过程中放在节点上的数字可以作为该节点上的一种人工式势函数，并且开始节点有一个全局最小值。相反路径（从目标到开始）顺着这个函数的“梯度”下降。

从（(*BAC*)）转移到（(*ABC*)）的标记传播过程显示在图7-3中。这种方法和所谓的广度优先算法(*breadth-first search*)相一致，该算法由[Moore1959]首次提出。

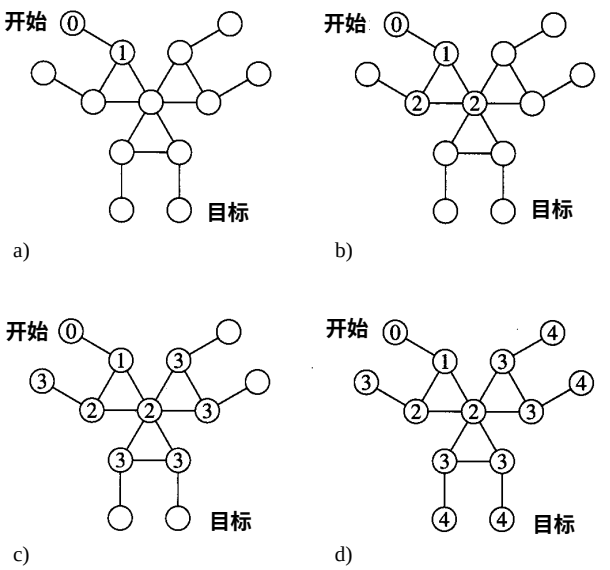


图7-3 搜索过程

把标记一个节点的后继节点的过程称为扩展 (*expansion*)。扩展将标记放在所有已标记过的节点的未标记的相邻节点上。应将哪一个已标记但还没有扩展的节点作为下一个扩展点是一个很重要的效率问题。在广度优先搜索中，下一个要扩展的节点是其节点标识数不大于任何其他没有扩展的节点标识数的节点。也就是说，在扩展标识为 j的节点之前，先要扩展标识为 i的所有节点，条件是$i < j$。其他搜索算法有不同的节点扩展选择，这些将在后面详细讨论。

7.4 基于特征的状态空间

用图标模型标识节点来解释状态空间是相当直接的——可以很容易地使状态上的动作结果

形象化。可以定义一个有特征标识的节点图，但此时，我们需要一种方法来描述一个动作是如何影响特征的。STRIPS [Fikes & Nilsson 1971, Fikes, Hartn, & Nilsson 1972]系统使用一种这种技术。其基本思想是定义一个有三个列表的算子，第一个是前提列表，指定了值为1和0的特征，以便可以应用动作。第二个是删除列表，列出其值将从1变为0的特征。第三个是值从0变为1的加入列表。在删除和加入列表中没有明确提到的特征值是不变的。这三个列表构成了一个STRIPS算子——一个动作结果模型。关于基于空间的STRIPS算子的讨论将推迟到本书的后面，在那之前会介绍关于特征计算的更为有效的技术。

我们也能训练一个神经网络，从它在$t-1$时刻的值和该时刻采取的动作来学习预测一个特征向量在时刻t时的值[Jordan & Rumelhart 1992]。如图7-4所示。虽然只显示了一层网络，但也可以使用具有隐藏单元的中间层。在训练后，预测网络能被用来计算来源于各种动作的特征向量。这些向量反过来又能作为网络的新输入来预测两步以后的特征向量，等等。一个相似的过程（基于统计群技术）被用来控制一个移动机器人[Mahadevan 1992, Connell & Mahadevan 1993a]。因为在预测过程中不可避免的错误,用这种方法企图作出更大的移动都证实是徒劳的。

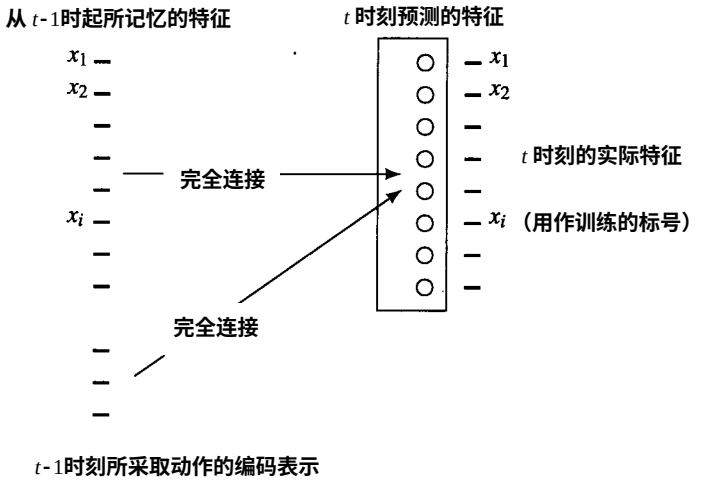


图7-4 在神经网络中预测一个特征向量

7.5 图记号

下一章我们将要处理大量的状态空间问题，在这之前，定义一些可以接受的、在讨论图和图搜索时要用到的术语是很有帮助的。在图7-5中，使用了一个图和树的示例来说明将要定义的术语。

一个图由一组节点（不一定是有限的）构成，节点对由弧连接，这些弧是从节点的一方指向另一方的有向弧，这样的图被称为有向图(*directed graph*)。为方便讨论，节点由环境状态模型标识，弧由动作名标识。如果弧是从节点n_i指向n_j，那么n_j就是n_i的后继（或者叫孩子），n_i是n_j的双亲。在我们研究的图中，一个节点只能有有限个后继。如果对节点中的每一个都可以是后继，在这种情况下，用边代替这一对节点。只包括边的图称为无向图。在随后有关图的表示中，弧带有箭头，而边没有。

一个（有限的）有向树是有向图的特殊情况。在有向树中（除了一个节点），每个节点只有一个父亲。没有父节点的节点被称为根节点。在树中没有后继的节点称为末端节点或叶节点。

我们称根节点的深度为0，树上任何其他节点的深度是其父节点的深度加1。一个（有根的）无向树（用边代替弧）是一个无向图，这里一对节点之间只有一条路径。

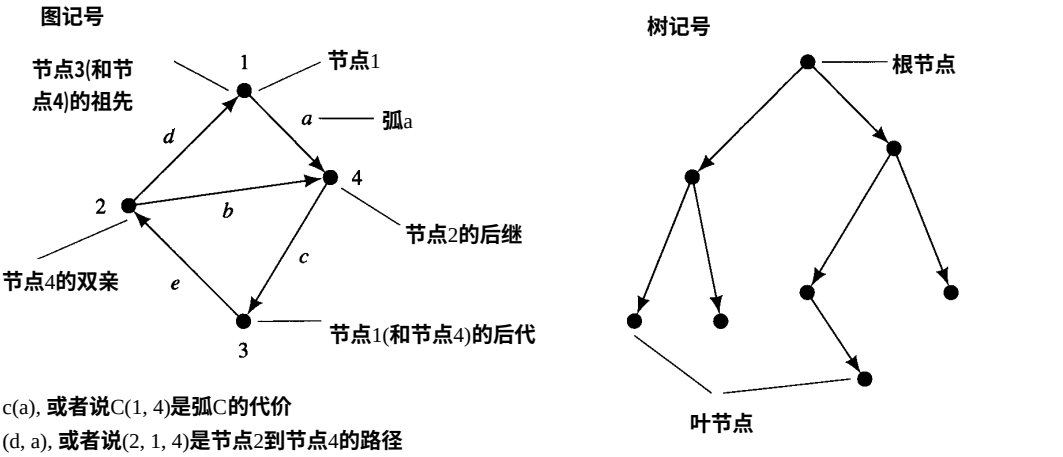


图7-5 图和树记号

在理论分析中，有一些树有这样的特征，即除了叶节点外，所有的节点都有相同数量 b 个后继。在这种情况下，b被称为这个树的分枝因子。

一个节点序列（$n_1, n_2, \ldots, n_k$），n_{i+1}是n_i的后继，$i=1,2,\ldots,k-1$被称为从节点n_1到n_k的长度为k的一条路径（另外,我们可以把连接节点的弧序列定义为一条路径）。如果存在从节点 n_i 到n_j的路径，那么就说从n_i可以访问n_j。n_j就是n_i的后代，n_i是n_j的祖先。

通过分配一个正值给弧来代表执行相应的动作的代价常常是方便的。用符号 $c(n_i, n_j)$（或者有时是$c(a)$）指示弧a（从n_i指向n_j）的代价。在后面的讨论中，假定这些代价比任意小的正数都大，这是很重要的。两个节点之间的路径代价是连接路径上节点间所有弧的代价的总和。在某些问题中，我们想找到两个节点之间的最小代价路径，这个路径被称为最佳路径。

最简单的问题是，在一个给定的代表初始状态的节点 n_0和另一个代表其他状态的节点 n_g之间寻找一条路径（也许有最小代价）。然而，经常遇到的问题是在节点 n_0和一组节点中任意成员之间找到一条路径，这些节点都代表着满足一些目标条件的状态。称这个节点集为目标集，它里面的每个节点都是目标节点。

给定一个图，有时我们可能想找到从图中其他各个节点到某个节点 n_0的一条路径，这样的路径集构成以n_0为根节点的图的一个生成树（注意，为了使一棵生成树是一棵树，对树的定义必须做一点改变，见习题7.2）。

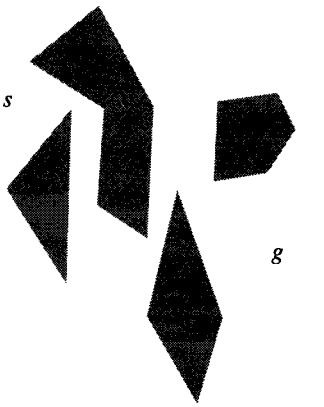
由于用图表示状态空间，因此有时可能交替地使用节点这个词，它既可以表示图中的一个节点，也可以代表环境状态，还可以表示那个节点的状态（数据结构或特征集）。同样，用单词“arc（弧）”代表图中的一条弧，或弧代表的动作，或对动作建模的算子（在状态描述中）。

7.6 补充读物和讨论

关于图的计算和算法的更多内容，可以参考[Cormen, Leiserson, & Rivest 1990, 第6章]。在图中寻找路径的任务可以引出很多问题，因此在这里和下面几章描述的方法还可应用在 agent 计划的研究范围之外的几个领域中。

习题

7.1 在下图由多边形障碍构成的二维空间中，一个机器人必须找到从开始点 s到目标点g之间的最短路径。假定机器人是无穷小，路径可以与障碍相邻（或接触到）但不能交叉。



1) 复制上图，画出s和g之间的最短路径。
2) 在这个二维空间中，虽然有无限个点，但在任意的 s和g点之间搜索一条最短路径时必须考虑的最小点集是什么？
3) 在刚找到的最小点集中给定一点，在相应的搜索图中描述产生该点后继的方法。图中s点的后继点是什么？

7.2 生成树：给定有向图S中的一个节点n, 从S中的任何一个节点到n至少有一条路径，阐述一下以n为根的S的生成树的精确定义。然后，考虑一下什么是最小生成树。

7.3 “传教士和食人者”问题是人工智能中的一个著名问题。下面是该问题的描述：

三个传教士和三个食人者来到一条河边。河边只有一条每次最多可供两个人过河的小船。传教士如何用这条小船过河才能使河两边的食人者决不会超过传教士的数量？

指定状态描述的格式、开始状态和该问题的目标状态。画出整个状态描述图，标出其节点（只要画出“合法”的状态即可——也就是说，只要画出河两边食人者不超过传教士数量的状态就行了）。

7.4 参考习题5.3中的三圆盘汉诺塔问题。假定我们不知道移动圆盘的 CW/CCW方法，但必须找到一个解决方法。用算子描述一下由方法 move(x, y, z)给出的动作，x可以是圆盘D_1、D_2或D_3中的任何一个，y和 z可以是A、B、C三个桩中任何不同的一对。定义这个问题的状态描述，确定开始状态和目标状态，画出包含该问题的所有可能状态的完整搜索空间。用适当的算子标出弧（每个移动都可反向进行；你只要标出每对可逆移动中的一个即可）。