第11章　其他搜索公式及其应用

有一些搜索技术的应用超出了为一个 agent选择动作的应用范围，这些应用包括为受限变量分配值和为解决最优问题等寻求解决方案。有一些专门的方法已经被开发用于这些应用中，虽然它们看上去并不直接与 agent设计有关，但它们也是重要的 AI技术，本章将讨论其中的一部分。

11.1　赋值问题

图搜索问题中的目标节点的条件可以定义为一个指定的数据结构或者标识它的状态描述，或者它可以根据对那个数据结构的条件和约束隐式地定义。在任一情况下，当问题是找到agent的一个动作序列时，标识目标节点的数据结构并不重要，重要的是到达目标的步骤序列。或者当目标节点不是由一个指定的数据结构定义而是由条件或约束定义时，问题可能就是要显示一些满足这些条件的数据结构，而用图搜索方法产生它的步骤可能是不相关的。我们称这类问题为约束满足(*constraint-satisfaction*)问题。这类问题中的一个著名例子是给受限变量赋值。它们被称为赋值问题(*assignment problem*)，下面将讨论这个问题。

我们能通过图搜索方法解决约束满足问题。一个目标节点是由满足约束的数据结构(或状态描述)标识的节点。算子将一个数据结构改变为另一个数据结构。开始节点是一些初始的数据结构。赋值问题的一个典型例子是 8皇后问题(*Eight-Queens problem*)。问题是这样的：把 8个皇后放在一个棋盘上，每行和每列只能有一个皇后，另外，一个皇后只能在任一行、列或对角线上 (也就是说，按照国际象棋规则，没有皇后能被放在一个位置以便它能抓住任何其他的皇后)。这个问题的一个解法参见图 11-1。由于这种问题具有从集合{第1行，第2行，...，第8行} 到变量{第1列的皇后位置，第2列的皇后位置，…，第8列的皇后位置}的赋值形式，所以称为赋值问题。

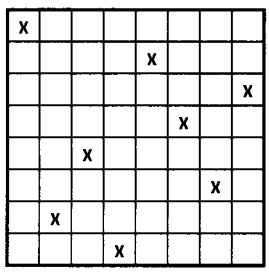


图11-1　8皇后问题的解决方案

把这个问题作为一个图搜索问题，一个明显的数据结构是一个8×8的数组，数组中的每个单元包含两个符号 (1和0)中的一个，1代表皇后，0为空。一个根据条件定义的隐式目标状态是保证 8个皇后都是安全的，不会被抓住。连接状态描述的算子和数组转换的方式相一致。例如，一个算子能把一个皇后加到还没有 8个皇后的数组中，或者它能把一个皇后移到另一个单元中。在赋值问题中，由于到达目标的路径并不重要，因此关于开始状态和算子是什么我们常有很多选择。

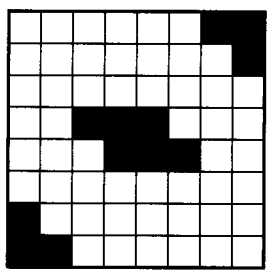


图11-2　纵横字谜的一个数组

作为赋值问题的另一个例子，考虑一下有很大状态空间的纵横字谜问题[⊖]。图11-2 显示了

⊖ 关于纵横字谜的产生问题， Matt, Ginsberg和他的学生已经测试了各种状态空间搜索技术［ Ginsberg et al. 1990。］

一个纵横字谜的数组。问题是根据纵横字谜规则用字母填充数组中的所有空白块，以使所有的行和列都是英语单词(没考虑这个问题的另一方面，即对企图解决纵横字谜的人创建提示)。在这个问题中，一个状态描述是字母和空格 (和已涂黑的单元一起)组成的任何数组，一个目标状态是一个合理纵横字谜解法的任何数组。连接状态描述的算子能将字母和空格的组合转变为另一种数组。例如，一个算子能把一个单词加到一个空行或空列上，或者它能把一个字母变为另一个字母。

11.2 构造性方法

我们可以用上一章讲的搜索方法解决赋值问题。最直接的方法是一步一步地构造所要求的赋值——尽管我们对那些步骤不是太感兴趣。开始（开始节点）不进行任何赋值，对 8皇后问题，相应的数据结构是全为 0的一个数组。每个算子加一个皇后到数组中，但要以满足约束的方式加入。因为在每一列必须有一个皇后，不失一般性，我们能保证在深度为 0的节点应用算子产生第1列皇后的放置，应用到深度为 1的节点的算子是第2列皇后的放置，等等。因为这个解法是一步步构成的，我们称这个方法为构造方法 (*constructive method*)（后面将提到一个与之相对应的方法）。

图11-3中给出了搜索树的一部分，它能用于 8皇后问题和纵横字谜问题的构造方法（为了易读性，用*X*而不是1和0来表示一个皇后的位置）。特别注意，算子用来从前一个节点产生节点。尤其对纵横字谜，很明显状态空间是巨大的，在每个节点我们有成千上万的算子可以应用。

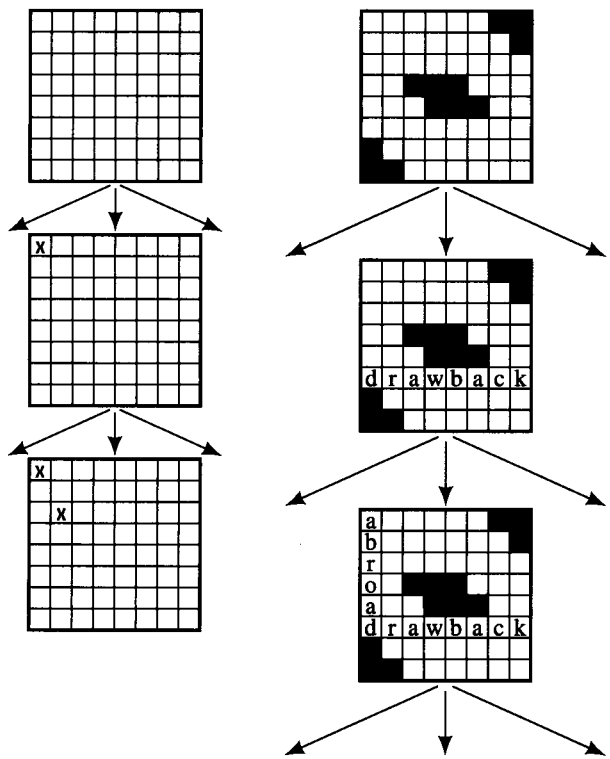


图11-3 构造性公式表示的状态空间

一种称为约束传播(*constraint propagation*)的计算技术有助于显著地减少搜索空间的大小。它和构造技术组合使用——依次给每一个变量赋值。我在如何应用8皇后问题的简化版本中描述了这个技术。考虑4个皇后放在一个4×4棋盘中，彼此不能互相抓住。在4皇后问题中，我们有4个变量q_1、q_2、q_3和q_4，分别代表4列中的某一列，一个皇后可以放在它们的其中之一上。每个变量可以是1、2、3和4这4个和行数相对应的值中的一个。例如，当q_3等于2时，一个皇后被放在第2行第3列。4皇后问题对这些变量的值提出了约束，因此如果q_1等于1，q_2就不能等于1或2。

约束被表示在一个叫约束图(*constraint graph*)的有向图中，在这个图中的每一个节点有一个变量和该变量的一组可能值标识。一个有向约束弧(*constraint arc*)（i，j），连结着节点i和j，条件是节点i的变量值受节点j的变量约束。图11-4表示了针对4皇后问题的这样一个图的例子。在这个问题中，每个变量约束着所有其他的变量，因此所有的节点之间都有弧存在。如果对弧尾的每一个变量值至少有一个弧头的变量值没有违反约束，我们就说有向弧（i，j）是一致的。图11-4中的弧是一致的，因为对每对q_i和q_j(i不等于j)，对q_i的每一个值，有一个q_j值没有违反约束。

给变量中的一个或几个赋值后，我们能用弧一致性概念排除其他变量的一些值。约束传播过程在图中的弧上进行迭代，试用加强弧一致性来排除弧尾上的变量值。当没有更好的值能被删除时这个过程终止。现用一个例子阐明这个过程。假如一个深度优先搜索过程通过给变量q_1赋值1开始（即我们把一个皇后放在第1行第1列）。应用到这个赋值的约束传播如下向前推进：

1) 考虑弧（q_2，q_1）：排除q_2=1和q_2=2，因为q_1值（我们刚分配的值）和q_2的这些值不一致。

2) 考虑弧（q_3，q_1）：排除q_3=1和q_3=3，因为q_1的值和q_3的这些值不一致。

3) 考虑弧（q_4，q_1）：排除q_4=1和q_4=4，因为q_1的值和q_4的这些值不一致。

在约束传播的这一阶段（不完全），我们有图11-5。在图随后的约束传播过程中，排除q_3的所有值，因此看到q_1=1时这个问题没有解法。在q_1=1的情况下不能再进行进一步的搜索，而是回溯到q_1=2的情况。

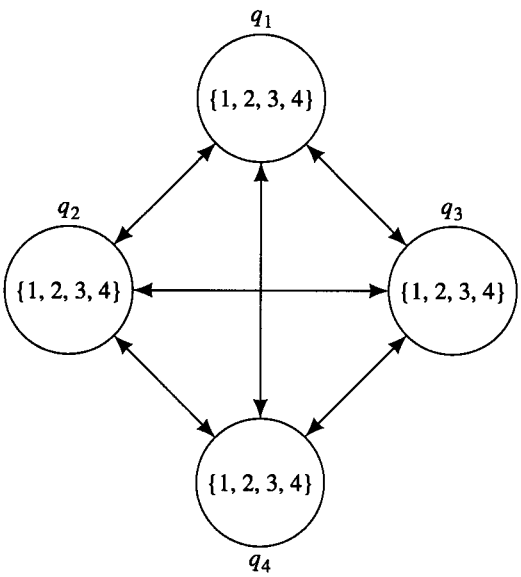


图11-4 4皇后问题的一个约束图

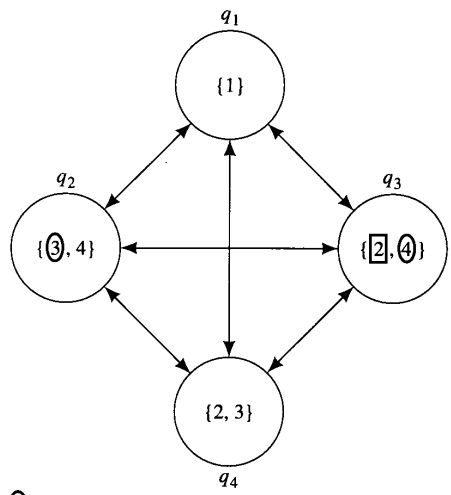


图11-5 q_1 = 1时的约束图

下面，我们假定q_1=2进行约束传播。开始几步如下：

1) 考虑弧（q_2, q_1）：排除q_2=1, q_2=2和q_2=3。

2) 考虑弧（q_3, q_1）：排除q_3=2和q_3 = 4。

3) 考虑弧（q_4, q_1）：排除q_4=2。

我们得到图11-6。连续的约束传播使每个节点变量只有一个值，使所有的弧保持一致。因此，我们看到在完成搜索前只有一个解法，约束传播已经找到了那个解法。

就像例子中一样，如果采用前面赋给变量的值，约束传播有时会排除几个变量的所有值——使得一个搜索问题没有解法。有时用先前给变量的赋值，可能只有一个一致的解法。在这些情况下，约束传播取消进一步搜索。约束传播的基本思想已扩展到包括对一致性更复杂的测试，但是它的大部分经济型应用可能涉及到刚才示例的一致性检查。约束传播已经应用到各种有趣的问题中，包括在可视地形分析中用 +、−、或→标识线和本书后面讨论的命题满足问题。为了更好地研究这个方法及其扩展和应用，可参考[Kumar 1992]。

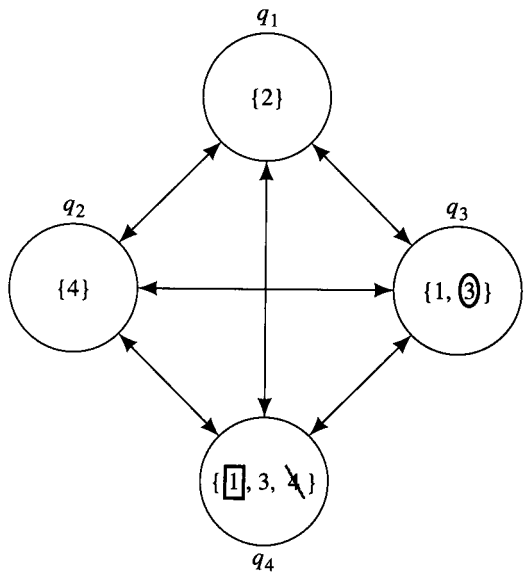


图11-6 q_1 = 2的约束图

11.3 后发式修补

存在另外的方法来为图搜索方法的解法设置一个问题。这个方法叫修补方法 (*repair approach*)，因为它用一个提议的解法开始，这个解法一般不满足约束条件，修补这个解法直到它满足约束条件。因此，初始节点一般由不满足所有约束的一个数据结构表示。算子产生一个新的数据结构，它相对应于一个不同的提议解法。

例如在8皇后问题中，开始时8个皇后每列一个，行位置是任意的，也许是随机的。我们通过移动一个皇后以违反更少的约束来修补有缺陷的解法。在所谓的最小冲突 (*min-conflict*)[Gu 1989, Minton, et al. 1992]修补方法中，我们依次考虑每一列（从第一列开始），用进攻那个单元的皇后数（在那一列的外面）标识出那一列的每个单元。然后，将那一列的皇后移到有最小进攻皇后数的单元（最小冲突数）。彼此间的联系被随机打破。这个算子产生一个后继节点，它由被提议算法的一个轻微修改所标识。这样依次通过每个列。在图 11-7中，用最小冲突示例了8皇后问题深度优先搜索的一部分。再一次，皇后位置用 *X*表示，单元中的数字表示进攻该单元的皇后数。类似的基于修补的方法 [Minton, et. al 1990]已被用于解决更大的问题，象百万皇后问题㊀。

当把修补方法应用于纵横字谜问题中时，开始节点可以是完全充满字母的任何组合。我们可以把一个组合中的某个字母换成其他的字母来修补一个有缺陷的解法，以使最终每行和每列

㊀ 尽管图标约束扩展和基于修补的方法是有趣的，但对 *N*皇后问题碰巧也是一个线性时间算法 [Abramson & Yung, 1989]

都是一个单词。因为在每个节点，有成千上万的其他算子可以应用，因此纵横字谜的搜索空间是巨大的。

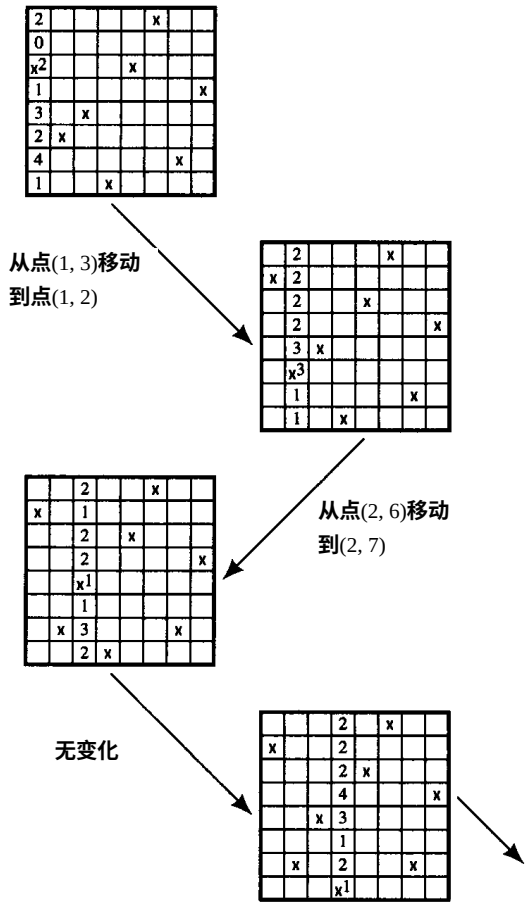


图11-7 8皇后问题的修补步骤

我们应该用构造方法还是修补方法，能使用的状态和算子是什么，所有这些对搜索的困难度都会有很大的影响。

11.4 函数优化

在某些问题中，我们没有一个显式的目标条件，而可能只有建立在一些数据结构上的函数 v，我们要找到一个结构，它有这个函数的最大（或最小）值。如果把这些数据结构当做空间中的一些“点”，这个函数就能被作为空间上的一个“地形”。一类方法就是遍历这个地形，在它当中寻找最高的点。由于不知道全局最大值，因此我们可能已找到了最大值时却不能肯定。

遍历一个空间的技术叫爬山方法(*hill-climbing*)，它从一个点遍历到“相邻”的有最高高度的点[⊖]。当相邻节点没有比当前点更高的高度时，爬山方法终止——因此，它们可以得到一个

⊖ 当寻找一个函数的最小值时，严格地讲，应该称其为下山方法。但一般地讲，将用爬山方法描述这两种活动。在任何情况下，通过乘一个因子－1，我们能把一个问题转化为另一个问题。

局部最大值。

我们能用图搜索方法解决爬山问题。同样，节点用数据结构标识。算子能把一个给定的数据结构改变为一个相邻的结构。爬山顺着一条路径（更像没有回溯的深度优先搜索）估计高度，它决不会下降到一个更低的点。下面是找到一个有（局部的）最大值的节点的爬山算法。

爬山算法：

1) 把当前节点n设为随机选择的节点n_0;

2) 产生n的后继 （用给这个问题定义的算子），选择后继n_b，它的值$v(n_b) = v_b$是这些后继中最大值。（任意断开联系）；

3) 如果$v_b < v(n)$，将n作为到目前发现的最好节点，并退出；

4) 否则，将n设为nb，转到第2步。

注意，除了我们总是扩展有最高值的后继（只要它的值不比它的父节点小）和没有提供回溯外，这个算法非常像深度优先搜索。爬山搜索中的移动是不能取消的。

在一个3×3的网格中用给单元填色示例了爬山方法（在这个问题中，山的高度其实在下降）。给出一个由任意的红色和兰色网格单元构成的网格，要找到一种配色以使得一个单元的颜色和其相邻单元颜色相同的数量达到最小。也就是说，在状态空间中我们要找到一个节点 n^*，对所有的n, ,$v(n^*) \le v(n)$, $v(n)$是节点n中有相同颜色的相邻单元数。对算子而言，允许任何单元的颜色可以从红色变为蓝色，反之也行。图11-8表示了该问题的一部分图和在下山过程中采用的路线。

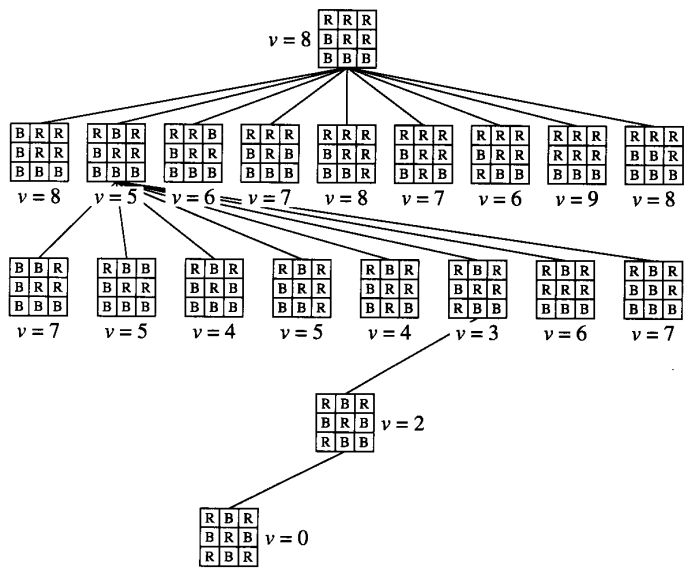


图11-8 两颜色问题解法

如果一次移动不能使v的值改变，我们就说在搜索空间中移到了一个高地。对于爬山算法，在高地上无休止地徘徊是可能的，在这种情况下，算法重复地遍历前面访问过的节点而不能爬到一个更高的地带。通过增加一个跟踪v值保持不变的次数计数器可以限制这种徘徊。以附加存储为代价，可以通过拒绝把以前访问过的有相同值的节点作为当前节点来改进高地问题。

在搜索空间中，另一个困难由“山脊”（或它的反义词“山沟”）引起。在这种情况下，任何存在的移动都会把我们带到一个更低的节点，但是顺序执行的两次移动将会增加高度。该问题如图11-9所示。每一个可能的移动都使我们离开山脊到达一个更低的高度，但是两个移动序列会使

我们移动到一个更高的山脊。山脊问题有时可通过将两个或更多的移动组合成一个“宏移动”来避免，或者通过一个有限数量的“向前”搜索来避免。

我们可以通过从不同的位置开始执行几个独立的爬山搜索（并行或顺序的）来解决这个问题，以找到局部最大值。这些搜索中的每一个都可能结束在不同的局部最大值，我们可以用这些值中的最大值。第 4章讨论的GP方法是一种随机的爬山方法。在这个方法中，几个“爬山者”并行地工作，通过建立后代爬山者进行移动，被优化的函数是适应性函数。可以用各种普通爬山方法[Juel & Wattenberg 1996，O' Reilly & Oppacher 1994] 来比较GA和GP方法的效率。

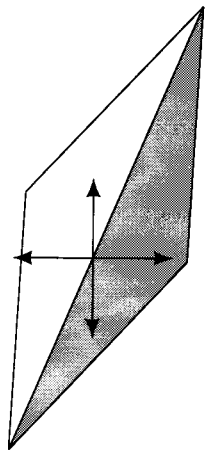


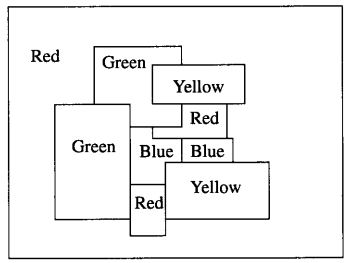
图11-9 山脊问题

一个叫做模拟退火 (*simulated annealing*)的过程[Kirkpatrick，Gelatt，& Vecchi 1993]也有助于解决极端的局部问题。这个过程有多种版本。在一种版本中，根据可用移动的概率分布选择移动，分布促成向较低高度（下山）的节点移动。用一个几乎可以忽略偏差的分布开始这个过程，使它向着这些偏好的节点移动，逐步增加偏差，直到最后移动可能以极大的概率向着一个较低高度的节点进行。最终的结果是：在过程的开始，我们在这个地形上随机地移动。最终，如过程保证的话，我们会下降到一个山谷之中。如果这个山谷不是非常深，也不是非常宽，不久一个后继的随机移动将把我们从这个山谷中推出。不可能从一个宽阔的（因此可能很深）山谷移出，在这个过程结束时（没有任何随机移动），我们下降到它的最深点。这个过程因类似于冶金中的退火过程而得名，在冶金过程中一个材料的温度被逐渐降低以允许它的晶体结构达到一个最小能量状态。在模拟退火过程中，在概率分布中控制宽度的参数常叫做温度。

习题

11.1 参照第6章习题6.4中的图。解释一下如何用一个约束图和约束传播为这个图找到一个一致的标识。

11.2 用启发式修补方法解决下图中有 4种颜色的着色问题，以便没有任何相邻的两个区域有相同的颜色。用下面的图开始。



11.3 用最小冲突方法解决 4皇后问题。开始时， 4个皇后都在主对角线上。随机地打破连接！（手工完成这个问题可能是冗长乏味的，但是这很值得做）。

11.4 如果你还没有做习题7.3（传教士和食人者问题），现在就做它。假设函数f = 在河的开始一侧的总人数。这个问题能用基于f的下山方法解决吗？能或不能，为什么？