计算机科学1994Vol. 21№. 5

# 约束满足问题的预处理方法研究

陈恩红　蔡庆生

（中国科学技术大学计算机系　合肥230026）

**摘　要**　Constraint satisfaction is a kind of technique which involves the assignment of values to the variables the problem contains, meanwhile, the assigned values are required to satisfy the constraints among the variables. It is extensively applied in many areas of artificial intelligence. This paper briefly introduces and analyzes the preprocessing method, i. e. local consistency method which can improve the efficiency of problem solving. Finally, several research directions for constraint satisfaction problems are given.

**关键词**　Constraint satisfaction problem, Constraint network, Arc. consistency, Path consistency, Ordered constraint graph.

搜索控制问题是大多数人工智能问题求解面临的一个根本问题，而约束满足是解决这一问题的常用方法之一。它源于机器视觉领域中的情景标识任务，如今在人工智能的众多领域（如规划、调度、时序推理）中获得了广泛的应用，受到了人工智能界的高度重视。在近几期的 IJCAI 和 AAAI 等国际人工智能会议上这方面的内容均占有一定的比重，《Artificial Intelligence》杂志曾于1992年出了一期约束满足问题的专辑，展现了这一领域在风范（paradigm）、可处理性（tractability）及应用等三个方面的研究内容。本文将简要介绍约束满足问题有关基本概念、约束预处理的基本方法及该领域的若干研究方向。

## 一、引　言

简单地讲，约束满足是对所求解的任务包含的变量进行赋值同时又满足变量间的关系或约束的一种技术，约束满足问题通常基于约束网络来处理。形式地说，约束网络由包含 n 个变量 $X_1, X_2, \cdots, X_n$ 的集合（对应的值域 $D_1, \cdots, D_n$）和约束集 $C_1, \cdots, C_t$ 组成。每个约束 $C_i$ 由两部分组成：变量集 $var(C_i)=\{X_{i_1}, \cdots, X_{i_j}\}$ 和关系 $rel(C_i)=rel(X_{i_1}, \cdots, X_{i_j}) \subseteq D_{i_1} \times \cdots \times D_{i_j}$。约束网络的全部解可表示成 $rel(R)=\{(X_1=x_1, \cdots, X_n=x_n) \mid \forall C_i, \Pi_{var(C_i)}rel(R) \subseteq rel(C_i)\}$，其中 $\Pi_U\rho=\{x_U=(x_{U_1}, \cdots, x_{U_j}) \mid \exists x \in \rho, x$ 是 $x_U$ 的扩充$\}$ 表示变量子集 $U=\{U_1, \cdots, U_j\}$ 在关系 $\rho$ 上的投影。求解约束满足问题包括确定约束网络是否有解、找出约束网络中单个或全部解、以及确定约束网络中部分变量的例化能否延拓至全局解等多方面的任务。

解决约束满足问题的一般方法是回溯搜索[1]，即在求解过程中，以某种顺序处理变量，每次给一个变量赋值后，都检查该变量与已赋值变量间的约束满足性。若满足，则继续对下一个变量赋值；若不满足，则选择其它的赋值。若该变量值域中的任何值都无法使其与已赋值变量间的所有约束都得到满足，则进行回溯。但这种方法极其低效，为此，人们对它作了若干改进，并提出相应的算法，如"选择回溯"、"面向从属关系的回溯"[2]。它们对回溯点的选择与标准的回溯方法不同，不再按变量例化的逆序盲目地进行，而是将回溯点直接定在与最新扩展的变量有关的变量上，显然，避免了很多回溯，提高了求解效率。但是，人们进一步发现，在实际开始搜索之前，通过一些预处理就可以消除回溯发生的许多因素。也就是说，这些预处理方法的着眼点不是放在回溯发生时如何去寻找回溯点，而是如何尽可能多地消除产生回溯的因素。

## 二、约束满足的预处理方法

目前已有的约束满足问题的预处理方法大多是针对二元约束满足问题，即每个约束至多涉及两个变量的问题，其原因有二。一是方便问题的研究，因为很多约束满足问题是 NP 类的，如图着色、n-皇后问题等，不可能有通用的有效解决方法。二是很多非二元约束满足问题可通过一种转换方法[3]被转换成与之等价的特殊形式的二元约束满足问题。因此，本文在介绍与分析约束满足问题求解方法时，是基于二元约

陈恩红　博士生，从事遗传算法、约束满足问题等领域研究。蔡庆生　教授，从事机器学习、专家系统等领域研究。

束满足问题的。

二元约束满足问题常用约束图来表示，图中节点 i 表示变量 $X_i$，节点 i 与 j 之间的边表示变量 $X_i$ 与 $X_j$ 的约束 $C_{ij}$（又称直接约束）。若 $X_i$ 与 $X_j$ 的任何取值均满足 $C_{ij}$，则 $C_{ij}$ 称泛约束[4]；若 $X_i$ 与 $X_j$ 的任何取值均不满足 $C_{ij}$，则称 $C_{ij}$ 为空约束。由连接节点 i 和 j 其长度为 m(m>1)的路径 $i_0=i,i_1,\cdots,i_m=j$ 导出的约束 $C_{i_0i_1\cdots i_m}$ 表示路径上全体约束的合成（称路径约束），若存在序列 $v_1\in D_{i_1},\cdots,v_{m-1}\in D_{i_{m-1}}$，使得 $C_{i_0i_1}(x,v_1)$，$C_{i_1i_2}(v_1,v_2)$，$\cdots$，$C_{i_{m-1}i_m}(v_{m-1},y)$ 都成立，则称值对 $x\in D_{i_0}$，$y\in D_{i_m}$ 满足路径约束。

### 2.1 节点、弧、路径一致性方法

如前所述，对于求解一般的约束满足问题没有一个有效的方法。但是人们发现，通过一些局部的一致性处理，在一定程度上能提高其中很多问题的求解效率。这些局部一致性处理方法依它所涉及的变量及约束（约束图中节点及边）的数目不同而分为节点、弧、路径一致性方法。虽然这些方法本身不能完全将解求出，但却能够消除部分不能延拓至全局的局部不一致性。

最简单的是节点一致性方法，该方法仅从节点对应的变量所取值域中删除那些不满足一元约束的值。弧一致性及路径一致性实际上是 i-一致性（i-consistency）的两种特殊情况。如果一个约束网络，其任何大小为 i−1的一致子网络在添加第 i 个变量且扩充相应的约束后而构成的子网络也是一致的，则称该网络是 i-一致的[5]。简言之，就是任意 i−1个变量的一致例化能够一致地扩充至 i 个变量。i=2,3就是上面所说的弧一致性和路径一致性。直观上看，弧一致性是基于支持这一概念的。例如，当给节点 i 一个标记 b（即给 $v_i$ 赋值 b）后，只要有弧与 i 相连的所有节点 j 至少存在一个标记支持 b（即满足 i 与 j 之间的约束 $C_{ij}$），则称 b 是节点 i 的有效标记。如果某个与 i 邻接的节点 j 的任何标记都不支持 i，则 b 应从 i 的可能标记中删除。用于保证弧一致性的算法 AC-1[6]就是根据这一思想提出的。但该标法中有许多冗余操作，因为只要有一条弧存在不一致性，就要对所有的弧作一致性处理，其最坏情况下的时间复杂度为 $O(a^3ne)$（$a=\max_i|D_i|$，n 为变量数，e 为约束图的边数）。AC-3对此作了改进，当某条弧存在不一致性时，仅对那些因该弧的一致性处理而受到影响的弧作一致性处理。具体算法如图1所示。其最坏情况下的时间复杂度为 $O(a^3e)$。Mohr 等的弧一致性算法 AC-4[7]在实现方法上做了改进，但基本思想相同。具体做法是：给每个弧-标记对指派一计数器，弧-标记对用[(i,j),b]表示，意为从 i 到 j 有弧且 i 的标记为 b。并为节点 j 的每个标记 c 构造集合 $S_{jc}=\{(i,b)|$ if 节点 i 的标记 b 受节点 j 的标记 c 支持$\}$。若节点 j 的标记 c 被删除，则所有的弧-标记对[(i,j),b]对应的计数器值均减一。其中 i 的标记 b 被 j 的标记 c 支持。该方法对约束的局部一致性传播的控制是通过跟踪删除的节点标记实现的，其实现效率高于 AC-3。最坏情况时间复杂度为 $O(ea^2)$。

```
Procedure AC-3
begin
  初始化 Q/*Q 包含图 G 的所有弧*/
  while(Q 非空)do
    从 Q 中选择一条弧(k,m)，并删除；
    若((k,m)不一致)则
    对(k,m)作一致性处理；/*删除 k 的某些标记
                          */
    将与 k 有关的其它弧添加至 Q 中
  end do
end
```

图1 算法 AC-3

路径一致性方法的任务是，确保满足任一直接约束 $C_{ij}$ 的节点 i，j 的标记对(i,b)-(j,c)，满足所有从 i 到 j 的路径导出的路径约束。文[8]已指出，若一完全图是3-一致的，则它也是路径一致的。任何一个非完全约束图都可以通过在任何不存在弧的两节点间添加一条由泛约束对应的弧，转换成一个与之等价的完全约束图。所以，对任何图来说，要保证它是路径一致的，只要作3-一致性处理就足够了。算法 PC-1正是基于这一思想来实现路径一致性的，图2所示为 PC-1中所有冗余操作被删除后的改进算法 PC-2。该算法的最坏情况时间复杂度为 $O(a^5n^3)$。

```
Procedure PC-2
begin
  初始化 Q/*Q 中包含所有长度为2的路径*/
  while(Q 非空)do
    从 Q 中选择一条路径 i-j-k，并删除；
    若(i-k-j 不一致)则
    做 i-k-j 的一致性处理；
    将受 i-k-j 的一致性处理影响的路径添至 Q 中
  end do
end
```

图2 算法 PC-2

MOHR 等同样从实现方法上对此又作了进一步改进，对每一条长度为2的路径 i-k-j 的两节点 i 和 j 的任意标记 b 与 c 分配一计数器 Counter[(i,j),k,b,c]，计数值为 k 中与 b 和 c 一致的标记数。并为路径的中间结点 k 的每一标记 d 构造集合 $S_{kd}=\{((i,j),b,c)|$ if (b,d)满足 $C_{ik}$，(d,c)满足 $C_{kj}\}$。以此设计出改

算法 PC-3[7]在效率上有很大的提高，其最坏情况下的时间复杂度为 $O(n^3a^3)$。

### 2.2 基于变量排序的局部一致性方法

在求解约束满足问题时，变量的处理顺序往往对搜索空间的大小影响很大。例如，三元一次方程组 $x+y+z=10, x+2y=6, x=2$可看成是一个简单的约束满足问题。用搜索方法求解它，若变量的处理顺序为 x,y,z，问题很快便能得到解决，但若以其它顺序处理，问题的求解就复杂得多。所以，以何种顺序处理变量在约束满足问题求解中显得非常重要。本节我们主要就它与弧一致性、路径一致性等方法的结合来作介绍与分析。

实际上，基于变量排序的局部一致性处理依它所解决问题的复杂程度又分为两种方法：有向一致性（directional-consistency）和自适应一致性（adaptive-consistency），有向一致性方法是建立在弧一致性与路径一致性方法基础之上的，而自适应一致性是建立在 i-一致性基础上更为一般的方法。以上这两种方法对一类问题尤其是树型结构的问题的解决非常有效，问题经它处理之后可以无回溯地搜索求解，搜索过程中变量的处理顺序应与预处理方法相同。下面，首先介绍与变量排序有关的若干概念[8]。

有序约束图由一个约束图及其节点的一种排序 $v_1,\cdots,v_n$ 组成。有序约束图的节点宽度是排在该节点之前且有边与之连接的节点数，有序约束图的宽度是全体节点宽度的最大值，约束图的宽度是所有排序对应的全体约束图的宽度最小值。图3是一约束满足问题的六种可能的变量排序（自底向上）对应的有序约束图。最左图中节点 c 的宽度为2，而第二个图中 c 的宽度为1。第一个有序约束图的宽度为2，第二个有序约束图的宽度为1，因此约束图的宽度为1。有向一致性方法对宽度为1和2的约束图非常有效，经它处理后的问题一般可无回溯地搜索得到解。

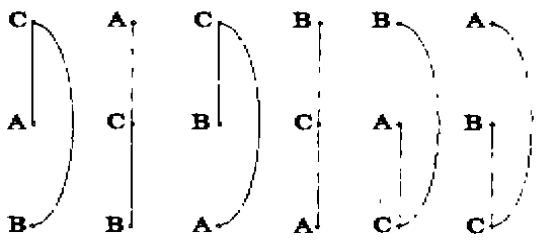


图3 一约束图的六种可能有序约束图

由于在弧一致性方法中，一条弧的一致性可能会因其它弧的一致性处理而受到破坏，因此，每条弧可能要被处理多次，但有向一致性方法克服了这种缺点。它对弧的处理是沿着变量的升序进行的，由此而得到的弧一致性称有向的弧一致性，每条弧只需处理一次。有向弧一致性与弧一致性有一个重要的差别：若一条弧 $(x_i,x_j)$ 是有向一致的且 $x_i$ 在 $x_j$ 之前，那么我们并不能保证有向弧 $x_j\rightarrow x_i((x_j,x_i))$ 也是一致的，即有向弧一致性仅能保证图中每条弧沿变量升序方向是一致的。用该方法对宽度为1即树形结构的约束满足问题作预处理后，再以升序来例化变量便可无回溯地求得问题的解。因此树结构约束满足问题在约束满足方法的研究中有非常重要的作用，据此可以将许多非树类的问题用某种方法转换成类树形的结构，以降低问题的复杂性。

对于宽度为2的约束满足问题，可以沿宽度为2的排序 d 的升序方向施实有向弧一致性和路径一致性方法。若对应变量排序 $d=(x_1,x_2,\cdots,x_n)$ 的有向约束图 R 的每一值对 $(x,y),x\in X_i,y\in X_j$，满足 $C_{ij}(x,y)$ 且对任意 $k>i,j$，均存在值 $z\in X_k$ 使得 $C_{ik}(x,z)$ 和 $C_{kj}(z,y)$ 成立，则称 R 是有向路径一致的。有向路径一致性方法在处理一条不一致的路径 $X_i\rightarrow X_k\rightarrow X_j$ 时，若该路径中 $X_i\rightarrow X_j$ 是泛约束对应的弧（即在约束图中 $X_i$ 与 $X_j$ 之间没有弧），则处理后 $X_i\rightarrow X_j$ 对应的不再是泛约束，从而在实际的约束图中就应将弧 $X_i\rightarrow X_j$ 加上。但这样处理后，原来宽度为2的问题的约束图结构就发生了变化，并有可能使排序宽度增大。宽度超过2后，就不能通过有向弧一致性及有向路径一致性方法实现无回溯搜索求解问题了。

对于宽度超过2的约束满足问题，一般采用一种称之为自适应一致性的方法来实现无回溯地搜索求解。它是以排序变量的降序来处理每个节点的，如同上面的有向路径一致性方法一样，在对节点进行处理后，可能会改变结点的宽度。因此该方法要动态地改变问题要求的一致性的级别，即 i-一致性的 i 应随节点宽度增大而增大。该方法对每个节点的处理，依排序中位于该节点之前且有弧与之相连的节点数 $i-1$ 的大小，做 i-一致性处理，亦即该节点将 i-一致性施实于 $i-1$ 个位于它之前且在约束图中有弧与之相连的节点集合之上。例如，图4(a)是一个有序约束图，图中变量的排序为 d=(E,D,C,A,B)，自适应一致性方法从节点 B 开始处理。由于节点 B 的宽度为1，所以 B 将2-一致性实施于节点 D，紧接着节点 A 将3-一致性施实于节点集(C,D)，即此节点 C,D 通过 A 联系起来，这意味着 C 与 D 之间的约束关系不再是泛约束，

它们之间应添加一条弧，表示C与D之间有非泛约束关系，如4(b)所示，节点C、D以同样方法一一处理。有序约束图经自适应一致性方法处理之后，可以无回溯地求解。但是，按照变量排序d对每个节点作i-一致性处理比较复杂，一是因为最佳排序d(即排序d对应的有序约束图之宽度与约束图的宽度相同)的寻找较困难，二是节点宽度较大时，一致性处理也较困难。若最佳排序d对应的有序约束图宽度为w(d)，则在最坏情况下完成自适应一致性方法所需时间为$O(n\exp(w^*(d)+1))$，其中$w^*(d)$为处理之后的有序约束图的宽度[3]。假设变量的排序$d=(X_1,X_2,\cdots,X_n)$，则相应的自适应一致性算法如图5所示。对宽度较小的约束满足问题即约束图较稀疏的问题来说，该方法还是比较有效的。

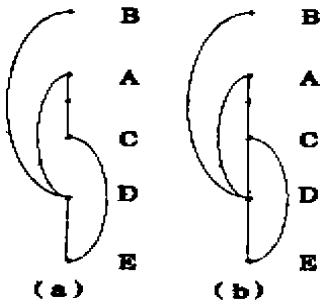


图4 自适应一致性方法处理前后的有序约束图
((a)处理前的有序约束图；(b)处理后的有序约束图)

```
Procedure Adaptive-Consistency(X1, X2, ···, Xn)
  begin
    for i==n to 1 by-1 / * 以降序处理节点 * /
    Compute PARENTS (Xi)/ * 若 X ∈ PARENTS
    (Xi)，则 X 与 Xi 之间有弧，且在序中 X 位于 Xi
    之前 * /
    connect all elements in PARENTS(Xi)
    perform consistency(Xi, PARENTS(Xi))
    find a solution using backtrack in the ordering(X1,
    ···, Xn)
  end for
end
```

图5 自适应一致性算法

## 三、若干进一步研究方向

1 *约束满足问题的并行处理研究* 近年来，随着并行计算技术的发展，从事约束满足问题的研究者已开始将其引入这一领域，以期进一步提高问题求解效率。文[9]在MIMD-SM及MIMD-DM等计算模型上设计并实现了用于约束满足问题的弧一致性处理的三种并行算法：SPAC-1，SPAC-2及L. M. Kirousis[10]在SIMD-EREW模型上设计出时间复杂度为$O(\log^2 n)$、处理器数为$O((m+n^3)/\log n)$(m为约数个数，n为变量数)的算法来求解约束符合某种条件(implicational约束)的约束满足问题，但由于其处理器数随问题规模的增大而增长过快，使得该算法难以实用。特别是随着新计算模型如BSP、LogP等的提出，如何在这些更加实际的模型上设计出有效的约束满足算法等将是非常有意义的研究内容。

2 *基于遗传算法的约束满足方法* 遗传算法是一种通用的搜索方法，它利用了"适者生存"这一自然选择机制，不断地对问题的中间解进行演化、最终生成问题的解。其基本思想是维持一定数量的知识结构(称population)，每个结构都表示问题的候选解，population通过"适者生存"的竞争机制和受控变异(包括重组和突变等遗传操作)不断进化。其优点在于它能够在搜索过程中兼顾新的空间搜索和将搜索的主要方向集中在高性能(high performance)空间部分。这种搜索技术对约束满足问题的解决是非常有益和重要的，如文[11]采用了遗传状态空间搜索方法来求解n-queen及Job Shop Scheduling两个典型的问题，显示出了相当好的性能。

3 *与基于解释的学习结合* 约束满足问题的基本解法是搜索，因此可能会出现组合爆炸。为了避免这一情况的发生，一种可能的方法是赋予问题求解器在搜索过程中获取有效的控制知识的能力。这种控制知识可以通过若干途径获得，如启发式评价函数的应用，或通过经验学习归纳并将其应用到新的问题中。但启发式评价函数的获得往往是困难的，而通过经验归纳得到的控制知识可能因为归纳跳步而失去其正确性。因此，作为克服这些缺点的一种可能方法——基于解释学习正被约束满足问题领域的研究者采用。这种学习方法的重要性在于，它能够利用领域理论去解释(或证明)某搜索过程低效的原因，再对此进行泛化(generalization)，并将其应用在以后的搜索过程，从而达到避免一类低效搜索的发生之目的。文[12]已在这一方向作了有益的探索并取得令人鼓舞的结果。

(参考文献共13篇略)