第 33 卷 第 2 期 自 动 化 学 报 Vol. 33, No. 2
2007 年 2 月 ACTA AUTOMATICA SINICA February, 2007

约束问题求解

季晓慧 [1] 张健 [1]

摘 要 约束问题的求解涉及到人工智能、运筹学、计算机科学等领域. 它的应用范围也极为广泛，包括计算机科学、控制科学、生物学等方面. 本文对其发展历史、所要求解的问题以及它的基本方法等问题进行了综述.

关键词 约束求解，优化问题
中图分类号 TP301.6

Constraint Processing

JI Xiao-Hui[1] ZHANG Jian[1]

Abstract Constraint processing involves the techniques of artificial intelligence, operations research, computer science and so on. It has found many applications in computer science, control theory, biology and so on. This paper is a survey of its history, methods and the problems it encounters.

Key words Constraint processing, optimization

1 约束问题求解的发展

约束问题的求解研究始于上个世纪 70 年代的 Scene Labeling 以及 60 年代的 Interactive Graphics[1~3]. Scene Labeling 试图用二维的线条描绘三维的物体，它的成果在于产生了各种一致 (Consistent) 算法 [1, 3]. Interactive Graphics[4~6] 的目的是允许用户在计算机的显示屏上绘制和处理各种几何图形，它发展了约束求解技术中的局部推演 (Local propagation) 及约束编译 (Constraint compiling) 技术 [3, 6]. 约束求解的早期工作还包括 MIT 的电路分析与综合 [7]，这些工作开创了设计能够求解通用约束问题的语言的研究 [3, 7].

约束求解的重要进展在于 Gallaire[8] 及 Jaffar[9] 分别在 1985 年及 1987 年发现，逻辑程序设计 (Logic programming) 实质上是一种特殊的约束求解问题 [1~3]. 首先，逻辑程序设计语言的描述式 (Declarative) 特性与约束求解的 “描述问题而不是解决问题” 的思想很接近；其次，二者所使用的求解问题的回溯法也十分的接近. 更重要的是，逻辑程序设计中的等式项实质上是一种特殊的约束，而用于求解等式项的消解法也只是一种特殊的约束求解方法. 基于这种发现，约束逻辑程序设计 (Constraint logic programming)[9] 作为求解约束问题的新框架开始出现 [3]. 现在用于求解约束问题的工具大多都是基于约束逻辑程序框架的 [3]. 但这并不意味着约束求解仅限于此框架，约束问题也可以在类似于 C++，Java 的命令式语言框架下进行求解 [3]. 在此之后，随着约束求解技术的不断发展，求解混合约束问题的研究也开始发展起来.

收稿日期 2005-4-18 收修改稿日期 2006-6-9
Received April 18, 2005; in revised form June 9, 2006
国家自然科学基金 (60125207, 60421001) 资助
Supported by National Natural Science Foundation of P. R. China (60125207, 60421001)
1. 中国科学院软件研究所计算机科学国家重点实验室 北京 100080
1. State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080
DOI: 10.1360/aas-007-0125

2 约束问题

形式化地讲，约束问题由以下三个基本元素组成：变量 $\boldsymbol{V}$，变量的域 $\boldsymbol{D}$ 以及约束 $\boldsymbol{C}$. 变量的域 $\boldsymbol{D}$ 是变量可能取值的集合，变量 V_i 只能在它的域 D_i 中进行取值；约束 $\boldsymbol{C}$ 描述了变量 $\boldsymbol{V}$ 之间必须满足的关系. 求解约束问题就是在各变量的域 D 中找到一个 (或所有的) 值 $\boldsymbol{S}$ 使得各变量之间满足约束 $\boldsymbol{C}$[1]. 约束问题也可以表示为三元组 $<\boldsymbol{V},\boldsymbol{D},\boldsymbol{C}>$[2].

根据约束问题中变量的域 $\boldsymbol{D}$ 的不同，约束问题可以分为：布尔约束问题、有限约束问题、数值约束问题及混合约束问题等.

2.1 布尔约束问题

布尔约束问题要求 $\boldsymbol{V}$ 中的各变量只能在 0 或 1 上取值，即布尔约束问题的域 $\boldsymbol{D}$ 为 $\{0, 1\}$. 它的约束 $\boldsymbol{C}$ 实际上就是一组命题逻辑公式. 所谓命题逻辑公式是布尔变量与逻辑连接符按照如下的规则形成的组合体 [10]：1) 布尔变量是公式；2) 如果 φ 是公式，则 $\neg\varphi$ 也是公式；3) 如果 ϕ 和 φ 是公式，则 $\phi * \varphi$ 也是公式；4) 只有上面三条规则生成的表达式是公式. 这里 $\neg$ 是一元连接符 “非”，$*$ 可以是任何一个二元连接符，如 $\wedge$(与)、$\vee$ (或)、$\rightarrow$ (蕴含) 等.

在布尔约束问题中，每一个命题逻辑公式也可

称为布尔约束条件. 布尔变量的取值只有 “真” 和 “假” 两种，而经由连接符连接而成的布尔约束条件也只能取 “真”、“假” 两个值. 求解布尔约束问题的目的就是为该问题中的布尔变量赋值，使得该问题中的每一个布尔约束条件的值为 “真”.

SAT 是一类特殊的布尔约束问题. 它要求约束的形式为: $(L_{11}\wedge\cdots\wedge L_{1n_1})\vee\cdots\vee(L_{m1}\wedge\cdots\wedge L_{mn_m})$. 其中，$L_{ij}$ 为一个变量或一个变量的非 [10].

2.2 有限约束问题

有限约束问题，顾名思义，就是变量只能在有限域上取值. 在有限约束问题中，通常不考虑约束的具体形式，而采用列出所有满足该条件的变量取值组合的形式 [10]. N 皇后问题、鸽笼问题、图着色问题等都属于有限约束问题.

实际上，布尔约束问题可以看作是有限约束问题的特例.

2.3 混合约束问题

混合约束问题中的变量可以在多个域中取值，比如变量可以取布尔值、可以在有限数值域及无限数值域上取值等.

混合约束问题实质上是对布尔约束问题的一种扩展. 为了清晰地描述该问题，先作如下定义.

定义 1. 数值约束：形如 $Exp1\ rop\ Exp2$ 的约束称为数值约束，其中 $Exp1$ 与 $Exp2$ 为数学表达式，如 $2x-yz$；rop 为一数学上的关系操作符，包括 $=,<,>,\leq$ 及 $\geq$.

定义 2. 混合约束条件：混合约束条件由布尔变量及数值约束与逻辑连接符组成，它们按照与命题逻辑公式同样的规则形成组合体. 例如 $a\wedge(x-y>3)\rightarrow(x+z=4)$.

定义 3. 混合约束问题：由一个或一组混合约束条件组成的问题称为混合约束问题.

一个数值约束有 “成立” 和 “不成立” 两种情况，分别对应于这个数值约束的值为 “真” 和 “假”，那么混合约束条件的值就如同布尔约束条件的值一样，也只能取 “真” 或者 “假”. 从而求解混合约束问题的目的就是为其中的数值和布尔变量赋值，使得该问题中的每个混合约束条件的值为 “真”. 如果能够找到这样的布尔变量值和数值变量值，则称该混合约束问题能够成立，或称是可行的 (Feasible)，否则称为不能够成立，或称是不可行的 (Infeasible).

下面举一个具体的例子说明混合约束问题.

例 1. $\begin{cases}(x^2+y^2=2)\wedge b\wedge((x^2+y<5)\wedge(y<-2))\\ \neg(y<-2)\wedge e\rightarrow f\end{cases}$

假设 x, y 的取值范围都为实数，那么容易验证当 $x=1$，$y=1$，$b=$“真”，$e=$“假”，$f=$“假” 时，原混合约束问题是能够成立的或者说是可行的. 同样可以验证当 $x=0$，$y=-\sqrt{2}$，$b=$ “真”，$e=$ “真”，$f=$“真” 时，原混合约束问题也能够成立. 一般情况下，使混合约束问题能够成立的解并不是唯一的，通常只需求出其中的一组解即可. 对于上例，如果要求 x 的取值范围为 “大于 2 的实数”，y 的取值范围仍为整个实数范围，那么上例这个混合约束问题就是不能够成立的，或者说是不可行的.

3 约束求解的基本方法

如第 2 节中所述，布尔约束问题可以看作是有限约束问题的特例，同时，数值约束是混合约束问题的子问题，即只包含数值约束，且约束间的关系都是 “与” 的混合约束，因此本节我们介绍有限约束问题、数值约束及混合约束问题的求解方法.

3.1 有限约束问题的求解方法

总的来说，用于求解有限约束问题的方法包括完备算法与不完备算法两种. 所谓完备算法是指能够完全确定某约束问题是有解的还是无解的算法；而不完备的算法则在未找到解时不能确定该约束问题无解.

1) 完备算法

完备算法中最常用的是回溯法 [11]. 该方法先为约束问题中的部分变量赋值，在这个部分赋值的基础上为其余的变量赋值. 如果在这个部分赋值的基础上，下一个待赋值的变量找不到使得约束成立的解，则需要改变这个部分赋值中所赋的值. 该算法可用递归形式表示为 (其中, $pSol$ 表示已有的部分赋值，初始时没有变量被赋值; $Cons$ 表示当前待求解的约束)：

```
int Sat(pSol, Cons)
{
  Propagate(pSol, Cons);
  if contradiction results
    return 0;
  if pSol assigns a value to every variable
    return 1;
  choose x that does not have a value in pSol;
  for (i = 1; i <= m, i++){
    if Sat(pSol ∪ {x = v_i}, Cons)
      return 1;
  }
  return 0;
}
```

在该算法中，Propagate 是用已有的部分赋值 $pSol$ 简化约束 $Cons$ 的过程. 例如，如果 $pSol$ 是

$x = TRUE, y = FALSE, Cons$ 是 $(x \vee z) \wedge (b \vee y)$，那么就能把这个约束简化为 b；但如果 $Cons$ 是 $(x \wedge y)$，矛盾就会产生了. m 表示变量 x 可能取值的个数，v_i 表示其中的第 i 个值.

可以看出，回溯法中的两个关键操作是，选取下一个待赋值的变量及其值，以及在发现有约束不成立时回溯. 因此，针对这两个操作，回溯法常用的两种策略是：向前看 (Look-ahead) 和回跳 (Back-jumping)[11]. 向前看是指在为下一个变量赋值时，通常先进行约束推导 (Constraint propagation) 以及选择最合适的变量及变量的值. 约束推导是为了减少搜索空间，而选择最合适的变量及变量的值则是为了加快搜索. 向前看策略也称为可行性技术 (Consistency technique). 回跳则是在当下一个待赋值的变量找不到解的时候，不直接为刚赋过值的变量赋其它的值，而是经过分析，找到致使下一个变量无解的那个变量，然后为其改变所赋的值. 有关回溯法的具体策略及其它的求解约束问题的完备算法，参见文献 [11∼13].

2) 不完备算法

回溯法是一种全局搜索算法，即它要搜索遍整个问题空间. 因此，当问题空间很大时，这种算法是不可行的. 所以人们提出局部搜索法. 局部搜索法以损失解的完备性为代价来提高求解效率.

局部搜索法的基本思想是：从随机选定的解开始，不断对其进行微小的改变 (局部改进)，直到满意为止. 所谓的满意是指要么得到一个解，要么改进的次数大于设定的最大值. 对于后一种情况，我们就得不到该约束问题到底是有解还是无解的结论.

著名的局部搜索算法包括：顾钧的算法、贪心搜索过程 GSAT、禁忌搜索 (Tabu search) 和拟人拟物法等 [10]. 其它的还包括爬山法、模拟退火等 [14].

我们以 GSAT 为例说明局部搜索算法. GSAT 首先随机的为所有变量赋值，如果这组赋值使得所有的约束都成立，则算法结束；否则，GSAT 在该组赋值的邻点中选取一个使得约束成立最多的变量值为新的赋值. 此过程不断重复，直到找到使得所有约束都成立的解，或者改变次数大于预先设定的值. 在此，邻点是指如果两个赋值仅有一个变元的值不同.

为了克服 GSAT 容易陷入局部最小点的缺点，人们提出了各种改进措施，如给子句加权、随机游走等等. 具体可参见文献 [10].

关于完备算法与不完备算法间的详尽区别及各自的优缺点读者可参见文献 [14].

3.2　数值约束问题的求解方法

数值约束 (见第 2.3 节中的定义 1)，是形如 $Exp1\ rop\ Exp2$ 的约束，其中 $Exp1$ 与 $Exp2$ 为数学表达式，rop 为一数学上的关系操作符. 数值约束可化为如下的基本形式：$\begin{cases} f_i(\boldsymbol{x}) = 0, i = 1 \ldots p \\ h_j(\boldsymbol{x}) \le 0, j = 1 \ldots q \end{cases}$. 其中 $f_i(\boldsymbol{x}) = 0$ 称为等式约束，$h_j(\boldsymbol{x}) \le 0$ 称为不等式约束. 如果数值约束中的等式和不等式约束都是线性的，则称之为线性数值约束问题，否则称之为非线性数值约束问题. 线性数值约束问题的求解可用单纯形法 (Simplex method)[15]，它的求解相对已经比较成熟，故不多作介绍. 在此我们主要介绍非线性数值约束问题的求解.

1) 数值法

数值法的基础是高斯 - 牛顿法 [16]. 高斯 - 牛顿法是求解非线性方程 (即只有等式约束的数值约束) 的有力工具. 其迭代公式为：

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - [f'(\boldsymbol{x}_k)]^{-1} f(\boldsymbol{x}_k),\ k = 0, 1, 2 \ldots.$$

对于求解既包含等式又包含不等式的约束问题，数值法则将上述牛顿法扩展为如下的优化问题来进行求解 [17, 18].

$$\begin{aligned} \min \quad & ||\boldsymbol{x} - \boldsymbol{x}_n|| \\ \text{s.t.} \quad & f(\boldsymbol{x}_n) + f'(\boldsymbol{x}_n)(\boldsymbol{x} - \boldsymbol{x}_n) = 0 \\ & h(\boldsymbol{x}_n) + h'(\boldsymbol{x}_n)(\boldsymbol{x} - \boldsymbol{x}_n) \le 0 \end{aligned}$$

数值法的优点是速度快，它的收敛速度是超线性的. 但是它对于初始点的依赖极强，如果初始点的选取不接近于真实解，算法有可能找到错误的解 [19]，同时，如果原问题无解的话，数值法也不能肯定其就是无解的.

2) 符号计算法

符号计算 [20∼23] 又称为计算机代数 (Computer algebra)，它是以现代计算机为工具，研究代数对象的一门新兴学科. 符号计算中用于计算非线性代数方程组的有力工具包括吴方法 [24∼26]、Grobner 基方法 [27∼29]、结式法等等. 它们的共同思想是，将待求解的非线性代数方程组化为一个或多个等价的三角型方程组

$$\begin{cases} f_1(x_1) = 0 \\ f_2(x_1, x_2) = 0 \\ \cdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases}$$

此时第一个方程只含有一个变量，那么就可以用符号或者数值的方法求出 x_1 的值；将求得的值代入第二个方程，从而将第二个方程化为了一元方程，再对其进行求解；以此类推，可求出方程组中所有变量的值.

柱形代数分解 (Cylindrical algebraic decomposition)[30∼32] 是判断不等式约束的有

效办法. 设约束 h 中出现的变元个数为 r，柱形代数的基本思想就是将 r 维欧氏空间剖分成 “块”，使得对 “块” 中个别点进行检验就可以判定 h 成立与否.

如前所述，符号计算的优点是可以求得问题的精确解，但是其代价就是求解的速度较慢 [33].

3) 区间分析法

区间分析 (Interval analysis)[34, 35] 也称区间数学 (Interval mathematics)，是为了解决计算机只能表示有限位数字，以及其它与此相关的实际问题而产生的. 它的基本思想是对实数的扩展，即它用一个区间而不是单独的一个数来表示数，例如数 a 用区间的表示方法为 $\underline{a} = [\underline{a_I}, \underline{a_S}]$，$\underline{a_I} \le a \le \underline{a_S}$，它的意义在于，区间 $[\underline{a_I}, \underline{a_S}]$ 中一定包含 a 的精确值，而且经过合理的数学运算，这个区间可以足够小. 我们称 $\underline{a_S}$ 为区间 $\underline{a}$ 的上界，$\underline{a_I}$ 为区间 $\underline{a}$ 的下界，同时 $\underline{a}$ 的宽度为 $w(\underline{a}) = \underline{a_S} - \underline{a_I}$[34~36].

如果将函数 $f(\boldsymbol{x})$ 中的变量用相应的区间变量代换，同时将函数中的运算用相应的区间运算替换，所得的函数 $F(\underline{\boldsymbol{x}})$ 称为 $f(\boldsymbol{x})$ 的自然区间扩展函数. 那么区间分析用于求解等式数值约束 $f(\boldsymbol{x}) = 0$ 的过程就是一个典型的分枝 – 限界过程. 该过程的限界规则是判断 0 是否包含于 $F(\underline{\boldsymbol{x}})$ 内. 如果不包含，就可以剔除这个区间向量 $\underline{\boldsymbol{x}}$；如果包含，但 $\underline{\boldsymbol{x}}$ 的分量中区间宽度最大的值大于所需的精度值时，则需要进行分枝.

对于不等式约束 $h(\boldsymbol{x}) \le 0$，当 $H(\boldsymbol{x})_I$ 大于所需的精度时，约束显然是不可行的，可剔除这个区间向量 $\underline{\boldsymbol{x}}$；当 $H(\underline{\boldsymbol{x}})_S$ 小于所需的精度时，约束显然是可行的，不需要对 $\underline{\boldsymbol{x}}$ 进行进一步的分枝 – 限界；否则，对 $\underline{\boldsymbol{x}}$ 进行分枝.

初始时，可以将 $\underline{\boldsymbol{x}}$ 取为可能包含解的值，如各变量区间都取为 $[-10^8, 10^8]$. 同时，分枝 – 限界过程保证了能搜索遍整个的这个初始区间. 因此用区间法求解数值约束问题能够保证解不丢失，从而保证了算法的完备性.

4) 启发式算法

启发式算法 (Meta-heuristic algorithms)[37] 是通过模拟或揭示某些自然现象或过程而得到发展的，其思想和内容涉及数学、物理学、生物进化、人工智能、神经科学和统计力学等方面，为解决复杂问题提供了新的思路和手段. 常用的启发式算法包括模拟退火 (Simulated annealing, SA)[38]、遗传算法 (Genetic algorithm, GA) [39, 40]、进化规划 (Evolution programming, EP)、人工神经网络 (Neural network, NN)、混沌、禁忌搜索 (Tabu search 或 Taboo search, TS) 及其混合策略等.

大部分启发式算法应该归为改进型算法 [37]，如 SA、GA、EP 和 TS. 改进型算法也称邻域搜索算法，它从任一解出发，对其邻域不断进行指导性搜索并替换当前解来实现求解，即利用一些指导规则来指导整个解空间中优良解的探索，但是当搜索超过了一定的时间或次数限制时，我们就得不到该约束问题到底是有解还是无解的结论. 可见，求解数值约束问题的启发式算法与求解有限约束问题的不完备算法是十分类似的.

3.3 混合约束问题的求解方法

1) 区间分析法

在第 3.2 节中已经介绍了区间分析法在求解数值约束中的应用，但是它的应用不止如此. J. Cleary 首先在文献 [41] 中将区间分析引入到了约束逻辑程序设计中，但是他的模型有两个缺点，首先是只能处理区间凸 (Interval-convex) 约束 (即只能处理连续、单调函数)，其次是只能处理非逻辑变量，这就使得用区间分析进行的约束求解与整个约束逻辑程序设计的框架分离开来. 针对上述缺点，文献 [42] 提出了用区间分析求解混合约束问题的方法，从而将区间求解技术扩展到了逻辑变量的处理上. 它对布尔变量的处理，是把它当作仅能取值为 0, 1 的整数，并把逻辑连接符转化为相应的数学函数，如：

$$\begin{aligned}&\text{and=min}=\{(x,y,z)\in\mathcal{R}^3, \min(x,y)=z\}\\&\text{or=max}=\{(x,y,z)\in\mathcal{R}^3, \max(x,y)=z\}\\&\text{not}=\{(x,y)\in\mathcal{R}^3, y=1-x\}\end{aligned}$$

进一步，有：

$$\begin{aligned}\min = \ & (\{(x,y,z)\in\mathcal{R}^3, x\le y\}\bigcap\{(x,y,z)\in\mathcal{R}^3, z=x\})\bigcup\\ & (\{(x,y,z)\in\mathcal{R}^3, x>y\}\bigcap\{(x,y,z)\in\mathcal{R}^3, z=y\})\\ \max = \ & (\{(x,y,z)\in\mathcal{R}^3, x\ge y\}\bigcap\{(x,y,z)\in\mathcal{R}^3, z=x\})\bigcup\\ & (\{(x,y,z)\in\mathcal{R}^3, x<y\}\bigcap\{(x,y,z)\in\mathcal{R}^3, z=y\})\end{aligned}$$

文献 [42] 对数学函数的处理与第 3.2 节中介绍的方法不同，它是基于人工智能方法的，类似于第 3.1 节中介绍的可行性技术 (Consistency technique). 如对于加法 add=$\{(x,y,z), x+y=z\}$，设处理后的值分别为 u, v, w，则有：$u = x\bigcap(z-y)$；$v = x\bigcap(z-x)$；$w = z\bigcap(x+y)$.

但是，容易看出，这种求解数学函数的方法对于比较复杂的函数用起来是比较困难的，因为对某些函数来讲，求取它们的反函数是比较困难的.

2) 逻辑与数值相结合的方法

文献 [43] 介绍了一种用逻辑与数值相结合进行混合约束求解的方法. 为了清楚地描述该方法，在此将布尔约束条件及混合约束条件分别用二元组 $\{\beta, o\}$ 及三元组 $\{\beta, \chi, o\}$ 来表示，其中 β 表示约束中布尔变量的集合，χ 表示约束中数值约束的集合，o 表示约束中逻辑连接符的集合. 该方法主要分

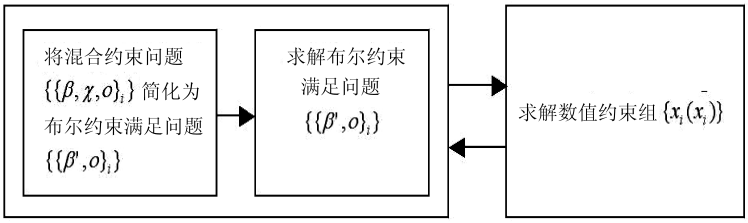

图 1　逻辑与数值相结合的方法
Fig. 1　Logic combinations of numerical method

为两步：替换和还原.

a) 替换：替换是指用一个新的布尔变量 β_{1i} 来替换混合约束问题中的每个数值约束 χ_i，从而将待解的混合约束问题的三元组集合 $\{\{\beta,\chi,o\}\}$ 简化为布尔约束问题的二元组集合 $\{\{\beta',o\}\}$. 显见，有 $\beta'=\beta+\beta_1$，然后对于该布尔约束问题，用回溯法进行求解.

b) 还原：还原是指将替换数值约束的布尔变量根据替换阶段结束时获得的值 (1 或者 0) 重新将其还原为该数值约束本身 x_i $(\beta_{1i}=1)$ 或者它的补 $\bar{x_i}$ $(\beta_{i1}=0)$，然后再用某种数学工具求解此时获得的数值约束组以验证其在数学上是否能够成立，如果有解，则其在数学上是成立的，原混合约束问题也是成立的，算法结束，否则需要重新回到替换阶段回溯混合约束转化而来的布尔约束问题，然后再进行还原.

该方法的求解流程如图 1 所示.

4　讨论

4.1　约束求解与优化问题的关系

优化问题是数学领域的重要分支，它研究如何在众多方案中找出最优的一个. 优化问题通常可以用下式进行描述：

$$\begin{aligned}\min\quad & f(\boldsymbol{x})\\ \text{s.t.}\quad & g_i(\boldsymbol{x})=0, i=1,\ldots,m\\ & h_j(\boldsymbol{x})\le 0, j=1,\ldots,n\end{aligned}$$

其中，$f(\boldsymbol{x})$ 称为目标函数，$g_i(\boldsymbol{x})$ 和 $h_j(\boldsymbol{x})$ 称为约束函数.

优化问题中的目标函数和约束函数都是线性的，称为线性优化问题；而优化问题中的目标函数或约束函数中含有非线性函数的称为非线性优化问题. 如果要求得到的解为整数或者部分为整数，那么该优化问题就是整数规划问题.

从问题形式上来看，约束问题与优化问题具有共同之处. 首先，数值约束就可以看作是目标函数为常数的优化问题；而有限约束问题就可看作是目标函数为常数的整数规划问题. 但通常约束求解属于计算机科学及人工智能的研究领域，而优化问题则是数学界尤其是运筹学界研究的对象.

虽然约束求解与优化技术分属于不同的领域，但是二者间的相互影响却由来已久. 20 世纪 50 年代就有人尝试用基于逻辑的方法求解优化问题，只是该方法没有像整数规划方法一样得到广泛的应用 [44]. 70 年代出现的隐枚举法则是基于逻辑的方法在求解整数规划问题上的成功应用 [44]. 随着整数规划求解方法的不断成熟，从 20 世纪 80 年代开始有人研究用整数规划方法来求解布尔约束问题. 这些学者包括 Hooker[45, 46]，顾钧 [47, 48] 等.

由于有限约束问题与整数规划问题的共同之处，结合二者的各自优势来解同一问题就是十分自然的. 事实上也的确有很多这类研究，如文献 [49∼51] 中采用了不同的方式将约束求解与优化技术结合起来.

在连续域上，约束求解与优化技术也有着广泛的结合. 用区间分析求解全局优化问题时就采用了约束求解的一致性 (Consistency) 技术 [19,52]. 同样在求解混合约束问题时，由于数值约束的存在，必然要用到优化问题的求解方法对其进行求解，如文献 [53,54] 采用了数值法求解混合约束问题中的数值问题，而文献 [55] 则采用了符号计算的方法求解混合约束问题中的数值约束.

4.2　分布式约束问题

随着网络和 Agent 技术的发展，分布式约束问题 (Distributed constraint satisfaction)[56∼58] 已成为约束问题研究中的一个重要领域. 分布式约束问题中的变量及约束分布存在于各 Agent 中，求解分布式约束问题就是为其中的各变量找到相应的值，使得各 Agent 以及各 Agent 之间的约束能够成立 [56]. 分布式约束问题并不是简单地并行求解约束问题，它能够求解传统约束求解技术不能够求解的问题，比如问题本身的结构是分布的，或者求解的

问题由于涉及安全或者隐私，各部分间是不透明的[56∼58].

4.3 约束问题生成

通过运行随机生成的约束问题来测试求解器的性能是约束求解中常用的手段.

在随机生成 SAT 问题实例时，常常需要事先固定好若干参数 (如子句长度、子句个数、变元个数)，然后根据它们随机地生成一些问题实例. 随机 SAT 问题实例的难度随着子句个数与变元个数之比 m/n 的变化而变化 [10]. 当 m/n 很小时，变元很多，子句很少 (比如说只有一个子句)，很容易满足；而当 m/n 很大时，变元很少，子句很多，很容易出现矛盾，子句集往往不可满足. 这两种极端情况都比较容易解决，而处于两者之间的情况则较难解决.

在随机生成有限约束问题上，文献 [59] 在 B 模型 [60] 的基础上提出了 RB 模型. RB 模型可以用下面的五元组 $< k, n, \alpha, r, p >$ 来表示，各参数的意义如下：$k \geq 2$ 表示每个约束中涉及的变量的个数；n 表示所得有限约束问题的变量个数；$\alpha > 0$ 表示通过计算 $d = n^{\alpha}$ 所得的每个变量的域，即各变量可取值的个数；$r > 0$ 表示通过计算 $m = rn \ln n$ 得到的约束的个数；$1 > p > 0$ 表示每个约束的难度 (Tightness). 为了产生 RB 模型的一个问题，RB 模型重复地选择 m 个约束，其中每个约束由互不相同的 k 个变量组成，并且这 k 个变量间不相容组合的个数为 $p * d^k$. 当 α 和 r 为常数时，它们表示随着 n 的增加，域的大小与约束的个数的增长. 可以证明，当 $p = 1 - e^{-\alpha/r}$ 时所产生的实例为最难解的实例 [59].

References

1 Apt K. *Principles of Constraint Programming*. Cambridge: Cambridge University Press, 2003. 9∼10

2 Dechter R. *Constraint Processing*. San Francisco: Elsevier Science, 2003. 1∼25

3 Hentenryck P. Strategic directions in constraint programming. *ACM Computing Surveys*, 1996, **28**(4): 701∼726

4 Sutherland I. A Man-Machine Graphical Communication System[Ph.D. dissertation]. Massachusetts Institute of Technology, 1963. 8∼24

5 Sutherland I. *Sketchpad: A Man-Machine Graphical Communication System*. New York: Garland Publishing, 1963

6 Borning A. The programming language aspects of thinglab, a constraint-oriented simulation laboratory. *ACM Transactions on Programming Languages and Systems*, 1981, **3**(4): 252∼387

7 Jaffar J, Maher M. Constraint logic programming: A survey. *Journal of Logic Programming*, 1994, **19/20**: 503∼581

8 Gallaire H. Logic programming: future developments. In: IEEE Symposium on Logic Programming. Boston, USA: IEEE Press, 1985. 88∼96

9 Jaffar J, Lassez J. Constraint logic programming. In: Proceeding of The ACM Symposium on Principles of Programming Languages. Munich, USA: ACM Press, 1987. 111∼119

10 Zhang Jian. *Deciding the Satisfiability of Logical Formulas - Methods, Tools and Applications*. Beijing: Science Press, 2000. 2∼48
(张健. 逻辑公式的可满足性判定 – 方法、工具及应用. 北京：科学出版社, 2000. 2∼48)

11 Dechter R, Frost D. Backtracking Algorithms for Constraint Satisfaction Problems - A Tutorial Survey[Online], available: http://citeseer.ist.psu.edu/correct/298588, June 1, 2006

12 Tsang E. *Foundations of Constraint Satisfaction*. San Diego: Academic Press Limited, 1993. 119∼188

13 Kumar V. Algorithms for constraint satisfaction problems: A survey. *Artificial Intelligence Magazine*, 1992, **13**(1): 32∼44

14 Freuder E, Dechter R, Ginsberg M, Selman B, Tsang E. Systematic versus stochastic constraint satisfaction. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. Montr é al, USA: Morgan Kaufmann, 1995. 2027∼2032

15 Dantzig G. *Linear Programming and Extensions*. New Jersey: Princeton University Press, 1963. 94∼111

16 Lin Cheng-Sen. *Numerical Computing Methods*. Beijing: Science Press, 1998. 160∼165
(林成森. 数值计算方法 (下册). 北京：科学出版社, 1998. 160∼165)

17 Robinson S. Extension of Newton′s method to nonlinear functions with values in a cone. *Numerical Mathematics*, 1972, **19**: 341∼347

18 Hentenryck P, Michel L, Deville Y. *Numerica: A Modeling Language for Global Optimization*. Cambridge: The MIT Press, 1997. 1∼16

19 Maranas C, Floudas C. Finding all solutions of nonlinearly constrained systems of equations. *Journal of Global Optimization*, 1995, **7**(2): 143∼182

20 Yang Lu, Zhang Jing-Zhong, Hou Xiao-Rong. *Nonlinear Algebraic Equations and Theorem Mechanical Proving*. Shanghai: Shanghai Science, Technology and Education Press, 1996. 14∼116
(杨路，张景中，侯晓荣. 非线性代数方程组与定理机器证明. 上海：上海科技教育出版社, 1996. 14∼116)

21 Wang Dong-Ming. *Selected lectures in Symbolic Computation*. Beijing: Tsinghua University Press, 2003. 1∼56
(王东明，杨路等. 符号计算选讲. 北京：清华大学出版社, 2003. 1∼56)

22 Wang Dong-Ming, Xia Bi-Can. *Computer Algebra*. Beijing: Tsinghua University Press, 2004. 1∼12
(王东明，夏壁灿. 计算机代数. 北京：清华大学出版社, 2004. 1∼12)

23 Liska R, Drska L, Limpouch J, Sinor M, Wester M, Winkler F. Computer Algebra, Algorithms, Systems and Applications[Online], available: http://kfe.fjfi.cvut.cz/ liska/ca/, June 1, 2006

24 Wu Wen-Tsun. Elementary geometry decision problems and mechanical proving. *Chinese Science*, 1977, **20**: 507∼516
(吴文俊. 初等几何判定问题与机械化证明. 中国科学, 1977, **20**: 507∼516)

25 Wu W T. Basic principles of mechanical theorem proving in elementary geometries. *Journal of Automated reasoning*, 1986, (2): 221∼252

26 Wu W T. *Mechanical Theorem Proving in Geometries: Basic Principles*. New York: Springer, 1994. 22∼42

27 Buchberger b. Grobner Bases: An algorithmic method in polynomial ideal theory. In: Multidimensional Systems Theory & Applications. New York: Institute of Electrical and Electronics Engineers, 1979. 184∼232

28 Adams W, Loustaunau P. *An Introduction to Grobner Bases*. USA: American Mathematical Society, 1994. 1∼50

29 Becker T, Weispfenning V. *Grobner Bases: A Computational Approach to Commutative Algebra*. New York: Springer, 1993

30 Arnon D, Collins G, McCallum S. Cylindrical algebraic decomposition I: The basic algorithm. *SIAM Journal of Computing*, 1984, **13**(4): 865∼877

31 Arnon D, Collins G, McCallum S. Cylindrical algebraic decomposition II: An adjacency algorithm for the plane. *SIAM Journal of Computing*, 1984, **13**(4): 878∼889

32 Collins G, Hong H. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 1991, **12**(3): 299∼328

33 Granvilliers L, Monfroy E, Benhamou F. Symbolic-interval cooperation in constraint programming. In: Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation. Hawaii, USA: ACM Press, 2001. 150∼166

34 Moore R. *Methods and Applications of Interval Analysis*. Philadelphia: SIAM Publishers, 1979. 9∼17

35 Hansen E. *Global Optimization Using Interval Analysis*. New York: Marcel Dekker, 1992

36 Kearfott R. *Rigorous Global Search: Continuous Problems*. Netherlands: Kluwer Academic Publishers, 1996. 1∼3

37 Wang Ling. *Meta-heuristic Optimization Algorithms and Their Applications*. Beijing: Tsinghua University Press, 2001. 1∼14
(王凌. 智能优化算法及其应用. 北京: 清华大学出版社, 2001. 1∼14)

38 Kirkpatrick S, Gelatt C, Vecchi M. Optimization by simulated annealing. *Science*, 1983, **4598**: 670∼680

39 Tomassini M. A survey of genetic algorithms. *Annual Reviews of Computational Physics*, 1995, (3): 87∼118

40 Vose M. *The Simple Genetic Algorithm: Foundations and Theory*. Cambridge: The MIT Press, 1999. 21∼46

41 Cleary J. Logical arithmetic. *Future Computing Systems*, 1987, **2**(2): 125∼149

42 Benhamou F, Older W. Applying interval arithmetic to real, integer and Boolean constraints. *Journal of Logic Programming*, 1997, **32**: 1∼24

43 Zhang J. Specification analysis and test data generation by solving Boolean combination of numeric constraints. In: The First Asia-Pacific Conference on Quality Software. HongKong, Los Alamitos: IEEE Computer Society, 2000. 267∼274

44 Hooker J. Logic, optimization and constraint programming. *INFORMS Journal on Computing*, 2002, **14**: 295∼321

45 Hooker J. A quantitative approach to logic inference. *Decision Support Systems*, 1988, (1): 45∼69

46 Hooker J. Resolution *vs.* cutting plane solution of inference problems: Some computational experience. *Operations Research Letters*, 1988, (1): 1∼7

47 Gu J. Global optimization for satisfiablility (SAT) problem. *IEEE Transactions on Knowledge and Data Engineering*, 1994, **6**(3): 361∼381

48 Gu J, Gu Q, Du D. On optimizing the satisfiability (SAT) problem. *Journal of Computer Science and Technology*, 1999, **14**(1): 1∼17

49 Hentenryck P. Constraint and integer programming in OPL. *INFORMS Journal on Computing*, 2002, **14**(4): 345∼372

50 Hooker J, Ottosson G, Thorsteinsson E, Kim H. A scheme for unifying optimization and constraint satisfaction methods. *Knowledge Engineering Review*, 2000,**15**(1): 11∼30

51 Darby-Dowman K, Little J, Mitra G, Zaffalon M. Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem. *Constraints*, 1997, **1**(3): 245∼264

52 Neumaier A. Complete Search in Continuous Global Optimization and Constraint Satisfaction[Online], available: http://www.mat.univie.ac.at/ neum/ms/glopt03.pdf, June 1, 2006

53 Ji Xiao-Hui, Zhang Jian. Solving Boolean combinations of nonlinear numerical constraints. *Journal of Software*, 2005, **16**(5): 659∼668

54 Ji Xiao-Hui, Zhang Jian. An efficient and complete method for solving mixed constraints. *Journal of Computer Research and Development*, 2006, **43**(3): 551∼556
(季晓慧, 张健. 一种求解混合约束问题的快速完备算法. 计算机研究与发展, 2006, **43**(3): 551∼556)

55 Castro C, Monfroy E. Basic operators for solving constraints via collaboration of solvers. *Lecture Notes in Artificial Intelligence*, 2000, **1930**: 142∼156

56 Yokoo M. *Distributed Constraint Satisfaction*. Berlin: Springer-Verlag, 2001. 47∼54

57 Yokoo M, Edmund Durfee, Toru Ishida, Kazuhiro Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Trans. on Data and Kn. Eng.*, 1998, **10**(5): 673∼685

58 Yokoo M, Katsutoshi Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 2000, **3**(2): 189∼212

59 Xu K, Li W. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 2000, (12): 93∼103

60 Gent I, MacIntyre E, Prosser P, Smith B, Walsh T. Random constraint satisfaction: Flaws and structure. *Constraints*, 2001(4): 345∼372

季晓慧 博士. 研究方向为约束求解、优化问题等. E-mail: xhji@cse.cuhk.edu.hk
(**JI Xiao-Hui** Ph. D.. Her research interests include constraint processing and optimization.)

张健 中国科学院软件研究所研究员, 博士生导师. 研究方向为自动推理、约束满足、程序测试及形式化方法等. 本文通信作者. E-mail: zj@ios.ac.cn
(**ZHANG Jian** Research professor. His research interests include automated reasoning, constraint satisfaction, program testing, and formal methods. Corresponding author of this paper.)