# 多值传播的相容性技术

朱兴军 [1,2] 张永刚 [1,2] 李 莹 [1,2] 张长胜 [3]

**摘 要** 相容性技术是求解约束满足问题的重要手段. 本文针对目前已有相容性算法的单值传播特点, 提出多值传播理论, 证明出 $k$ 次单值传播与一次多值传播的等价性, 在此基础上, 给出多值传播的弧相容定理. 将该定理与目前流行的 Singleton 弧相容技术结合, 得到多值传播算法 SAC-MP, 并证明其完备性和正确性. 通过对随机问题、$N$ 皇后、鸽巢问题及基准用例的测试表明, 算法 SAC-MP 的执行效率是已有算法 SAC-SDS 和 SAC-3 的 2∼3 倍.

**关键词** 约束满足问题, 相容性技术, 多值传播, Singleton 弧相容
**中图分类号** TP18

## Consistency Technique of Multi-value Propagation

ZHU Xing-Jun[1,2] ZHANG Yong-Gang[1,2] LI Ying[1,2] ZHANG Chang-Sheng[3]

**Abstract** Consistency technique is an important method to solve constraint satisfaction problem (CSP) instances. Since all existing algorithms have the same feature of single propagation, we propose a multi-value propagation theory and prove that $k$-single propagation is equivalent to 1-multi-value propagation. Based on this, we present arc consistency (AC) theory of multi-value propagation and combine it with SAC (singleton AC) to form a new multi-value propagation algorithm SAC-MP. Besides, we also prove its soundness and completeness. In our experiments on random CSPs, $N$-queens problems, pigeon problems and benchmarks, the efficiency of our algorithm SAC-MP is 2∼3 times those of the existing SAC-SDS and SAC-3 algorithms.

**Key words** Constraint satisfaction problem (CSP), consistency technique, multi-value propagation, singleton arc consistency (SAC)

约束满足问题 (Constraint satisfaction problem, CSP) 是人工智能研究领域的一个重要问题, 现实中大量组合优化问题都可以通过约束满足问题来建模求解. 在求解过程中, 相容性技术 (Consistency technique) 所起的作用是十分巨大的, 该技术通过移走不相容的值和尽可能早地发现将来会产生冲突的值, 来降低搜索空间并加速求解过程. 现有的很多相容性技术和相容性算法都能够实现这一目的[1]. 目前的相容性技术可以分为以下两类: 一类广泛用于搜索过程中, 如: 弧相容维护 (Maintaining arc consistency, MAC)[2]. 另一类则用于求解前的预处理过程, 该类技术最近研究的较多, 如: 保守双向相容性算法 (Conservative dual consistency, CDC)[3], Singleton 弧相容技术 (Singleton arc consistency, SAC)[4−6], 都是时间和空间比较理想的相容性技术. 在 Debruyne 等于 1997 年提出的 SAC-1 算法[7] 基础上, 2004 年∼2005 年, Barták 和 Bessière 相继提出基于 AC4 和 AC 2001 的 SAC-2 算法和 SAC-SDS 算法[5,8], 这些算法都使用大量的数据结构来避免重复的约束传播, 通过维持这些数据结构可以在约束传播过程中降低时间复杂度, 提高算法 SAC 的效率. 2005 年, Lecoutre 等提出一种基于深度优先、贪婪搜索的 SAC-3 算法[4], 该算法利用一种贪婪的搜索和在搜索过程中递增的弧相容维护来避免重复的约束传播过程. 如果遇到一个死结点, 则重新搜索其他值对的组合. 这一系列算法是从执行效率上对原有算法的一种改进. 而文献 [9] 是对 SAC 在理论和技术方面的一些研究. 关于 SAC 技术的研究, 我们已经提出基于预处理技术的约束求解算法 BT + MPAC 和 BT + MPAC*[10], 以及基于完全独立相容的 SAC-ESC 技术[11]. 在原有 SAC 算法中, SAC-1 等都采用宽度优先策略, 并且都有重复的约束传播过程, 效率上不如采用深度优先的 SAC-3[4]. 但是由于 SAC-3 采用的是递增式弧相容维护, 即在每次实例化一个变量时, 都要做一次相容性传播的处理, 并且只有在搜索到死结点时才停止此次搜索, 这就做了大量的重复传播和延误了死结点的发现时间.

收稿日期 2008-04-18 收修改稿日期 2008-12-29
Received April 18, 2008; in revised form December 29, 2008
国家自然科学基金 (60773097), 吉林省青年科研基金 (20080107) 资助
Supported by National Natural Science Foundation of China (60773097) and Young Scientific Research Foundation of Jilin Province (20080107)
1. 吉林大学计算机科学与技术学院 长春 130012 2. 吉林大学符号计算与知识工程教育部重点实验室 长春 130012 3. 东北大学信息科学与技术学院 沈阳 110004
1. College of Computer Science and Technology, Jilin University, Changchun 130012 2. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012 3. College of Information Science and Technology, Northeastern University, Shenyang 110004
DOI: 10.3724/SP.J.1004.2009.01296

本文通过对原有相容性技术和算法的研究, 发现它们都具有单值传播的特点. 为此, 我们提出多值传播理论, 给出相应定理, 证明出 $k$ 次单值传播与一次多值传播的等价性, 即每实例化 $k$ 个变量再进行约束传播, 这就避免了 $k-1$ 次的传播过程. 在此基础上, 又给出多值传播的弧相容定理, 并加以证明. 而后将该定理与目前流行的 SAC 技术相结合, 给出多值相容性算法 SAC-MP, 该算法利用多值传播性质降低约束检查, 提高了算法的执行效率. 不仅如此, SAC-MP 能够确定某些值对的约束传播是冗余的, 这样可以避免执行冗余的传播过程, 更早地迫使算法重新搜索其他值对的组合. 通过实验测试得出, 对于随机问题, 算法 SAC-MP 的执行效率是 SAC-SDS 和 SAC-3 的 2 倍; 对于鸽巢问题、$N$ 皇后问题以及基准用例, 算法 SAC-MP 的执行效率约是 SAC-SDS 和 SAC-3 的 2 ∼ 3 倍.

## 1 约束满足问题和 Singleton 弧相容技术

约束满足问题 $P$ 由三元组构成: 问题 $P$ 的变量集合 $var(P)$, 每个变量的论域 $dom(X)$ 以及关于这些变量的约束集合 $C(P)$, 问题 $P$ 的论域可用 $dom_P$ 表示. 特殊地, 如果对于约束集合中的任何一个约束 $c$, 所包含的变量个数都是 2, 则称该问题为二元约束满足问题. 问题 $P$ 的解 $\rho$ 是一个变量值对的集合, 使得任何一个变量实例化相应的值以后, 满足所有的约束关系, 通常用 $Val_X(\rho)$ 表示变量 $X$ 在解 $\rho$ 中的取值, 用 $sol(P)$ 表示问题 $P$ 解的集合. 称问题 $P$ 与 $Q$ 等价, 当且仅当 $sol(P) = sol(Q)$. 为了提高效率, 通常在求解前和求解过程中积极地使用约束删除变量中一些不参与解的值, 这个过程称为约束传播, 域过滤或者相容性技术. 而这一技术最核心的是弧相容技术, 其他众多相容性技术大多是基于此展开的. 对于二元 CSP 的约束图中的某一个弧 $(X, Y)$, 称它是弧相容的 (Arc consistency, AC), 当且仅当对于 $X$ 论域中能满足 $X$ 上一元约束的每一个值 $a$, 都在 $Y$ 的论域中存在一个值 $b$, 使得 $b$ 满足 $Y$ 上一元约束, 并且 $(a, b)$ 满足 $X$ 和 $Y$ 上的二元约束 $C(X, Y)$. 一个 CSP 是弧相容的, 当且仅当它的约束图中的每一条弧都是弧相容的. 对于问题 $P$ 在进行弧相容检查时, 如得到某一变量的论域为空, 则此问题 $P$ 无解, 这时记为: AC$(P) = \perp$. 文献 [12] 给出了其详细概念和相关术语.

在弧相容技术的基础上, 文献 [4−8] 引入 Singleton 弧相容的概念. 问题 $P$ 是 Singleton 弧相容 (SAC) 的, 当且仅当对于 $\forall X \in var(P), \forall a \in dom(X)$, $P|_{X=a}$ 是弧相容的. 其中 $P|_{X=a}$ 是将原问题 $P$ 中 $X$ 的论域 $dom(X)$ 用单独的 $a$ 进行替换.

## 2 多值传播

本节鉴于已有技术都是基于单值传播的特点, 首先提出多值传播理论, 即定理 1, 并证明 $k$ 次单值传播与一次多值传播是等价的. 利用定理 1, 我们可以得出在实例化 $k$ 个变量时, 能够避免 $k-1$ 次的约束传播过程. 证明定理 1 的过程中, 我们使用了 AC 算法执行过程的正确性这一性质[2].

**定理 1.** 设 $P$ 是一个约束满足问题, 若 $P_1 = AC(P|_{X_1=a_1}) \neq \perp$, $P_2 = AC(P_1|_{X_2=a_2}) \neq \perp$, $\cdots$, $P_k = AC(P_{k-1}|_{X_k=a_k}) \neq \perp$, 则

$$AC\left(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k}\right) \neq \perp$$

并且问题 $P_k$ 等价于

$$AC(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k})$$

**证明.** 设

$$Q = AC\left(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k}\right)$$

因为 $var(P_k) = var(Q)$, $C(P_k) = C(Q)$, 对任意 $i$, $i \in \{1, \cdots, k\}$, $X_i \in var(P_k)$, $X_i$ 的论域 $dom_{P_k}(X_i)$ 是 $\{a_i\}$, 而对于 $X_i \in var(Q)$, $X_i$ 的论域 $dom_Q(X_i)$ 也分别是 $\{a_i\}$, 故此, 由 $P_k \neq \perp$ 和 AC 算法的正确性得出

$$Q = AC(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k})$$

又因为 $P_1 \neq \perp$, $P_2 \neq \perp, \cdots, P_k \neq \perp$, 由 AC 算法的正确性容易证明:

$$sol(P_1) = \{\rho | \rho \in sol(P) \wedge Val_{X_1}(\rho) = a_1\}$$
$$sol(P_2) = \{\rho | \rho \in sol(P_1) \wedge Val_{X_2}(\rho) = a_2\}$$
$$\vdots$$
$$sol(P_k) = \{\rho | \rho \in sol(P_{k-1}) \wedge Val_{X_k}(\rho) = a_k\}$$

即有下式成立:

$$sol(P_k) = \{\rho | \rho \in sol(P) \wedge Val_{X_1}(\rho) = a_1 \wedge Val_{X_2}(\rho) = a_2 \wedge \cdots \wedge Val_{X_k}(\rho) = a_k\}$$

同理, 由 $AC(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k}) \neq \perp$ 和 AC 算法的正确性得出:

$$sol(Q) = \{\rho | \rho \in sol(P) \wedge Val_{X_1}(\rho) = a_1 \wedge Val_{X_2}(\rho) = a_2 \wedge \cdots \wedge Val_{X_k}(\rho) = a_k\}$$

故此, $sol(P_k) = sol(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k})$, 即, 问题 $P_k$ 等价于 $AC(P|_{X_1=a_1 \wedge X_2=a_2 \wedge \cdots \wedge X_k=a_k})$. □

从上面的证明过程中, 我们可以得出经过 $k$ 次约束传播以后得到的新问题 $P_k$ 和经过一次约束传

播得到的问题 $AC(P|_{X_1=a_1\wedge X_2=a_2\wedge\cdots\wedge X_k=a_k})$ 的解集合是相同的, 即这两个子问题是等价的. 下面我们给出多值传播的弧相容定理.

**定理 2.** 设 $P$ 是一个约束满足问题, 若

$$AC(P|_{X_1=a_1\wedge X_2=a_2\wedge\cdots\wedge X_k=a_k})\neq\perp$$

则

$$AC(P|_{X_1=a_1})\neq\perp\wedge AC(P|_{X_2=a_2})\neq\perp\wedge\cdots\wedge AC(P|_{X_k=a_k})\neq\perp$$

**证明.** 如若不然, 设 $AC(P|_{X_j=a_j})=\perp$, $P_1=P|_{X_j=a_j}$, $P_2=P|_{X_1=a_1\wedge X_2=a_2\wedge\cdots\wedge X_k=a_k}$, 由于 $var(P_1)=var(P_2)$, $C(P_1)=C(P_2)$, $dom_{P_2}\subseteq dom_{P_1}$, 又 $AC(P_1)=AC(P|_{X_j=a_j})=\perp$, 进而得: $AC(P_2)=AC(P|_{X_1=a_1\wedge X_2=a_2\wedge\cdots\wedge X_k=a_k})=\perp$, 与条件矛盾, 由此定理得证. □

## 3 SAC-MP 算法

本文将定理 1 和定理 2 与目前流行的 SAC 技术相结合, 形成多值传播的相容性算法 SAC-MP. 该算法与已有算法的不同之处在于: 已有算法在深度搜索时, 每实例化一个变量, 都要对其做一次约束传播. 若实例化 $k$ 个变量, 则做 $k$ 次约束传播. 这种递增式的检查, 其代价是高额的, 而由定理 1 得知每次实例化 $k$ 个变量, 只做一次约束传播与做 $k$ 次约束传播得到的问题是等价的. 据此, 算法 SAC-MP 可以节省 $k-1$ 次约束传播.

此外, 当 $AC(P|_{X_1=a_1\wedge X_2=a_2\wedge\cdots\wedge X_k=a_k})\neq\perp$ 时, 由定理 2 可以推出: $AC(P|_{X_1=a_1})\neq\perp\wedge AC(P|_{X_2=a_2})\neq\perp\wedge\cdots\wedge AC(P|_{X_k=a_k})\neq\perp$, 故它们也都无需进行约束传播; 而当 $AC(P|_{X_1=a_1\wedge X_2=a_2\wedge\cdots\wedge X_k=a_k})=\perp$ 时, 本文算法可以提前确定死结点, 以便重新搜索其他值对的组合. 将这些性质与原 SAC 算法相结合得出算法 SAC-MP. 该算法基于了深度优先及贪婪搜索的思想, 其伪码分别如算法 1、算法 2 和算法 3 所示.

**算法 1 (SAC-MP 算法).**

```
1  P ← AC(P);
2  repeat
3     P_before ← P;
4     Q_SAC ← {(X,a)|(X,a) ∈ P};
5     while Q_SAC ≠ ∅ do
6        buildBranch( );
7     end
8  until P = P_before
```

算法 1 通过 repeat 语句对问题 $P$ 中的所有值对进行检查, 直至问题 $P$ 不再发生变化. 而函数 buildBranch 的作用是对结构 $Q_{\rm SAC}$ 中的值对进行深度贪婪的 SAC 检查, 其伪码见算法 2.

**算法 2 (buildBranch 算法).**

```
1  br ← ∅;
2  P_before ← P;
3  repeat
4     d ← getpairs(k);
5     if not consistency(d) then goto 13;
6     P ← AC(P|_d);
7     if P ≠ ∅ then
8        foreach ∀(X_i,a_i) ∈ d do
9           add (X_i,a_i) to br;
10       end
11    end
12    else
13       if br ≠ ⊥ then
14          foreach ∀(X_i,a_i) ∈ d do
15             add (X_i,a_i) to Q_SAC;
16       end
17       goto 21;
18       end
19    end
20 until false
21 P ← P_before;
22 if br = ∅ then
23    foreach ∀(X_i,a_i) ∈ d ∧ i ≠ 1 do
24       add (X_i,a_i) to Q_SAC;
25    end
26    if AC(P|_{X_1=a_1}) = ⊥ then
27       remove a_1 from dom(X_1);
28       P ← AC(P);
29       delete (X_i,a_i) from Q_SAC s.t.
         (X_i,a_i) ∈ P_before ∧ (X_i,a_i) ∉ P
30    end
31 end
```

在算法 2 中, 首先实例化 $k$ 个变量, 然后作相容性判断. 如果相容性检查失败, 则重新搜索其他值对的组合; 否则, 进行深度优先、贪婪搜索. 在相容性检查过程中, 首先进行 consistency 检查, 然后再进行弧相容处理. 函数 consistency($d$) 的功能是: 判断部分解 $d$ 中变量的赋值是否都满足约束. 如果是, 则返回真; 否则返回假. 这样在存在冲突的情况下, 可以避免执行弧相容算法. 如果在第一次实例化 $k$ 个变量时就发现空论域, 则从这 $k$ 个变量中随机抽取一个变量值对进行弧相容检查, 将其余 $k-1$ 个变量放到初始队列中, 等待下一次的处理. 如果这个被选择变量进行约束传播后导致其他某个变量的论域为空, 则从原始问题 $P$ 中将该值删除, 否则, 不作处理. 这时无需再将该变量值对放入队列, 因为它已经 SAC 检查完毕, 并且成功.

在算法 3 中, 函数 getpairs 的主要功能是从队列 $Q_{\rm SAC}$ 中选择 $k$ 个变量值对, 且这些变量值对中

的变量彼此不同, 同时都没有在集合 $br$ 中出现. 如果得到这样的集合, 则将其返回; 否则, 返回当前所找到的部分变量值对的集合 $d$, 此时 $|d| < k$.

**算法 3 (getpairs 算法).**

```
1  d ← ∅;
2  while |d| < k do
3     delete (X_i, a_i) from Q_SAC s.t.
      (X_i, a_i) ∈ Q_SAC ∧ X_i ∈ br ∧ X_i ∉ var(d);
4     add (X_i, a_i) to d;
5     if Q_SAC = ∅ then return d;
6  end
7  return d
```

**定理 3.** 算法 SAC-MP 维护 Singleton 弧相容是正确的.

**证明.** 假设存在 $(X, a)$ 是被算法 SAC-MP 删除的第一个 Singleton 弧相容值对, 则从算法 SAC-MP 中可以看出: 该值对只能从算法 2 中第 27 行被移走, 则此时条件 $AC(P|_{X=a}) = \perp$ 一定成立 (见算法 2 中第 26 行). 由 Singleton 的定义可以看出值对 $(X, a)$ 在问题 $P$ 中不是 Singleton 弧相容的, 故此与假设矛盾, 算法正确性可证. □

**定理 4.** 算法 SAC-MP 维护 Singleton 弧相容是完备的.

**证明.** 当算法 SAC-MP 终止时, 对于 $P$ 中任意一个值对 $(X, a)$, 如果在算法 2 中 repeat 语句执行的第一次时被检查 SAC, 则有 $AC(P|_{X=a}) \neq \perp$ 成立, 由 Singleton 弧相容定义得出该值对在 $P$ 中是 SAC 的. 否则, 该值对不是第一个被实例化的, 则有 $AC(P|_{X=a \wedge Y=b \wedge \cdots \wedge Z=c}) \neq \perp$, 由定理 1 和定理 2 可以得出 $AC(P|_{X=a}) \neq \perp$ 仍然成立. 综上, $(X, a)$ 在 $P$ 中是 SAC 的, 故此算法 SAC-MP 是完备的. □

对于算法 SAC-MP, 当 $k$ 次实例化成功时所做约束传播次数为 1, 否则需要做 $k+1$ 次约束传播. 根据算法 SAC-3 的最坏时间复杂度 $\mathrm{O}(en^2d^4)$ 的分析过程容易得出, 算法 SAC-MP 的最坏时间复杂度为

$$\mathrm{O}((\rho + (k+1)(1-\rho)) \times \frac{d}{k} \times en^2d^3)$$

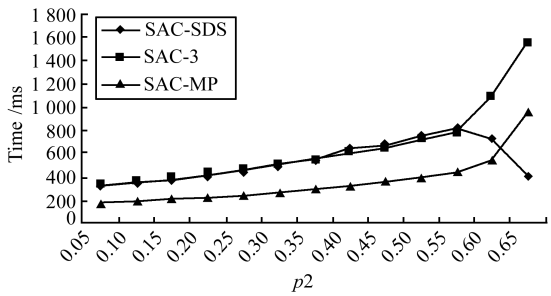
其中, $d/k$ 表示一次实例化 $k$ 个值所要做的约束传播次数, $e$ 表示约束网络中边的个数, $\rho$ 是每个变量一次实例化 $k$ 个值做相容性检查成功的平均概率. 当 $\rho = 0$ 时, 复杂度仍然是 $\mathrm{O}(en^2d^4)$, 而当 $\rho = 1$ 时, 复杂度是 $\mathrm{O}(en^2d^4/k)$, 时间效率可提高 $k$ 倍. 空间复杂度为队列 $Q$ 的大小, 即 $\mathrm{O}(nd)$.

## 4　实验结果

这里使用了四类测试用例, 分别是随机的约束满足问题、经典鸽巢问题和 $N$ 皇后问题, 还有一些标准的测试用例 (Benchmark). 从高效性和易实现的角度上考虑, 弧相容技术选择了含有启发式信息的 AC-3[13]. 本文用算法的执行效率和相容性检查次数来衡量算法的性能, 使用 C++ 语言在 "明月"[14] 平台上编写程序, 测试环境为: 硬件 DELL Intel P Ⅳ 3.0 GHz, CPU 512 MB RAM; 软件 Windows XP Professional SP2/Visual C++ 6.0, 其中每个用例测试 10 次, 然后取平均值作为最后结果.

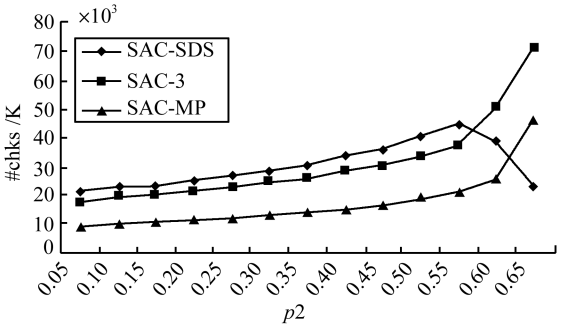
1) 随机约束满足用例

随机问题产生器随机地生成二元约束满足问题, 问题的难度和规模可以根据参数的设置来控制. 由于问题具有随机性, 所以绝大部分通用约束求解算法和通用启发式策略都采用这种测试方法[15]. 二元随机约束满足问题有四个描述参数 $\langle n, m, p1, p2\rangle$, 其中 $n$ 代表变量的个数, $m$ 代表变量论域的大小 (这里假设所有变量的论域大小相同), $p1$ 代表此约束满足问题的密度, 即约束图中边的个数和 $n \times (n-1)/2$ (完全图中边的个数) 的比值, $p2$ 为每个二元约束的松紧度. 本文选择参数为: $n = 100$, $m = 20$, $p1 = 0.05$[4−6]. 从图 1 (a) 中可以看出, 对于不同难度的随机问题, 算法 SAC-MP 的运行效率是算法 SAC-SDS 和 SAC-3 的 2 倍, 而从图 1 (b) 所示的相容性检查次数 (# chks) 来看, 算法 SAC-SDS 和 SAC-3




(a) CPU 时间

(a) CPU time




(b) 检查次数

(b) Check numbers

图 1　$\langle 100, 10, 0.05, p2\rangle$ 问题

Fig. 1　The problem $\langle 100, 10, 0.05, p2\rangle$

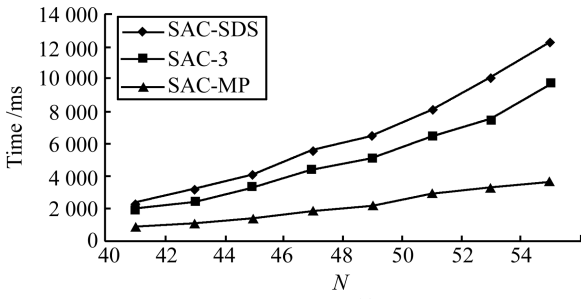
是算法 SAC-MP 的 2 倍. 当参数 $p2$ 变大时, 由于算法 SAC-SDS 使用大量数据结构避免冗余工作. 故此, 效率要高于本文提出的算法.

2) 鸽巢问题和 $N$ 皇后问题

鸽巢问题和 $N$ 皇后问题都是约束满足问题中极其具有代表性的问题, 它们分别代表着不同类别的问题. 鸽巢问题是一个不可满足问题的例子, 而 $N$ 皇后问题则是典型的可满足问题的一个用例. 首先对于鸽巢问题, 当鸽子数量由 40 逐渐变化到 54 时, 从图 2 (a) 中可以看出, 算法 SAC-MP 的执行效率始终是算法 SAC-SDS 和 SAC-3 的 3 倍. 从图 2 (b) 所示的相容性检查次数来看, 算法 SAC-SDS 和 SAC-3 所做的检查次数要明显高于 SAC-MP. 而对于另一类 $N$ 皇后问题而言, 皇后数量在 40～58 之间变化时, 算法 SAC-MP 的执行效率是算法 SAC-SDS 和 SAC-3 的 2 倍. 而当皇后数目为 61 时, SAC-MP 的执行效率达到 SAC-SDS 和 SAC-3 的 3 倍. 从图 3 (b) 所示的约束检查次数可以看出, 算法 SAC-MP 也具有明显的优势.

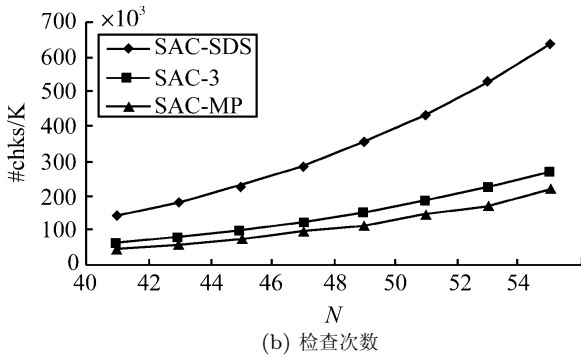
3) 标准库的测试用例 (Benchmark)

此外, 我们还测试了一些标准库中的测试样例 (http://cpai.ucc.ie/05/Benchmarks.html), 从中随机选取了两类问题, 一类是 frb 问题, 对这类问题进行求解一般来说比较困难, 文献 [16] 对这类问题进行了详细的研究; 另一类是 lsq-dg 问题. 测试结果见表 1, 对于 frb 问题, 算法 SAC-MP 的执行效率是 SAC-SDS 和 SAC-3 的 2 倍, 而从相容性检查次数来看, 算法 SAC-SDS 和 SAC-3 大约是算法 SAC-MP 的 2 倍. 对于另一类 lsq-dg-11 问题, 算法 SAC-MP 的执行效率是算法 SAC-SDS 和 SAC-3 的 3 倍. 从相容性检查次数来看, SAC-SDS 和 SAC-3 是 SAC-MP 的 3 倍左右, 而对于 lsq-dg-12, 算法 SAC-MP 的执行效率至少是 SAC-SDS 和 SAC-3 的 3 倍, SAC-SDS 和 SAC-3 的相容性检查次数是 SAC-MP 的 3 倍.




(a) CPU 时间

(a) CPU time

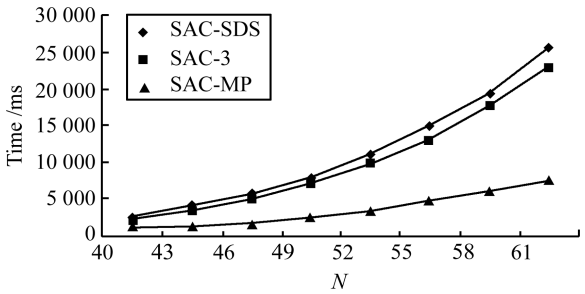



(b) 检查次数

(b) Check numbers

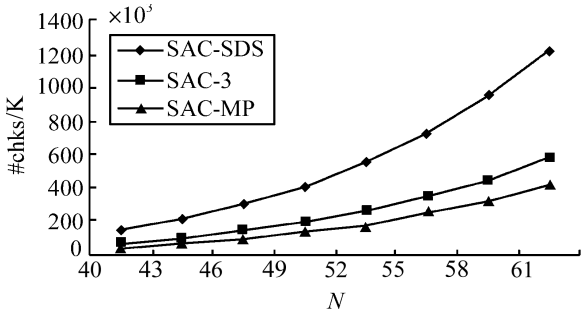
图 2 鸽巢问题

Fig. 2 Pigeon problems




(a) CPU 时间

(a) CPU time




(b) 检查次数

(b) Check numbers

图 3 $N$ 皇后问题

Fig. 3 $N$ queens problems

表 1 标准库中的测试用例

Table 1 The results of Benchmarks

| 问题 | | SAC-SDS | SAC-3 | SAC-MP |
|---|---|---|---|---|
| frb53-24-1 | Time (ms) | 688 | 703 | 390 |
| | chks (K) | 44 987 | 37 297 | 21 260 |
| frb56-25-1 | Time (ms) | 907 | 859 | 515 |
| | chks (K) | 55 856 | 46 334 | 26 676 |
| frb59-26-1 | Time (ms) | 1 266 | 1 312 | 766 |
| | chks (K) | 67 784 | 55 389 | 31 376 |
| lsq-dg-11 | Time (ms) | 1 234 | 1 250 | 407 |
| | chks (K) | 87 657 | 74 660 | 27 010 |
| lsq-dg-12 | Time (ms) | 2 156 | 2 234 | 703 |
| | chks (K) | 156 222 | 134 606 | 46 393 |

## 5 结论

相容性技术是求解约束满足问题的一个重要技术, 而作为一种特殊的相容性算法 — Singleton 弧相容最近已被大量研究. 本文通过对原有算法的研究, 提出多值传播理论, 给出多值传播定理和弧相容的多值传播性质. 将其与 SAC 技术相结合, 提出一种新相容性算法 SAC-MP, 并证明其正确性和完备性. 该算法能够在搜索过程中, 每实例化 $k$ 个变量, 仅需一次弧相容的处理, 所需时间仅为 $O(ed^2)$[2]. 而对于此过程, 算法 SAC-SDS 和 SAC-3 则需要做 $k$ 次弧相容处理, 耗用时间为 $k \times O(ed^2)$. 通过对随机问题、经典的鸽巢、$N$ 皇后问题以及基准测试用例的测试可以看出, 新算法 SAC-MP 与原有算法 SAC-SDS 和 SAC-3 相比具有极为明显的性能优势. 但本文的算法 SAC-MP 和现有 SAC 算法一样, 都没有考虑问题的固有性质, 如问题本身的对称性. 我们相信结合问题固有信息的 Singleton 弧相容算法将会有更加显著的性能优势.

## References


1 Barták R. Theory and practice of constraint propagation. In: Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control. Gliwice, Poland: Springer, 2001. 7−14

2 Bessière C, Regin J C, Yap R H C, Zhang Y L. An optimal coarse-grained arc consistency algorithm. *Artificial Intelligence*, 2005, **165**(2): 165−185

3 Lecoutre C, Cardon S, Julien V. Conservative dual consistency. In: Proceedings of the 22nd Conference on Artificial Intelligence. California, USA: AAAI Press, 2007. 237−242

4 Lecoutre C, Cardon S. A greedy approach to establish singleton arc consistency. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, UK: Professional Book Center, 2005. 199−204

5 Bessière C, Debruyne R. Optimal and suboptimal singleton arc consistency algorithms. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, UK: Professional Book Center, 2005. 54−59

6 van Dongen M R C. Beyond singleton arc consistency. In: Proceedings of the 17th European Conference on Artificial. Riva del Garda, Italy: IOS Press, 2006. 163−167

7 Debruyne R, Bessière C. Some practicable filtering techniques for the constraint satisfaction problem. In: Proceedings of the 15th International Joint Conference on Artificial Intelligence. Aichi, Japan: Morgan Kaufmann, 1997. 412−417

8 Barták R, Erben R. A new algorithm for singleton arc consistency. In: Proceedings of the 17th FLAIRS Conference. Florida, USA: AAAI Press, 2004. 257−262

9 Bessière C, Romuald D. Theoretical analysis of singleton arc consistency. In: Proceedings of the 16th European Conference on Artifical Intelligence. Valencia, Spain: Springer, 2004. 20−29

10 Sun Ji-Gui, Zhu Xing-Jun, Zhang Yong-Gang, Li Ying. An approach of solving constraint satisfaction problem based on preprocessing. *Chinese Journal of Computers*, 2008, **31**(6): 919−926
(孙吉贵, 朱兴军, 张永刚, 李莹. 一种基于预处理技术的约束满足问题求解算法. 计算机学报, 2008, **31**(6): 919−926)

11 Zhu Xing-Jun, Sun Ji-Gui, Zhang Yong-Gang, Li Ying. A new preprocessing technique based on entirety singleton consistency. *Acta Automatica Sinica*, 2009, **35**(1): 71−76
(朱兴军, 孙吉贵, 张永刚, 李莹. 一种新的基于完全独立相容性的预处理技术. 自动化学报, 2009, **35**(1): 71−76)

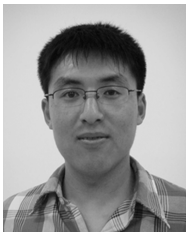
12 Tsang E. *Foundations of Constraint Satisfaction*. London: Academic Press, 1993

13 Sun Ji-Gui, Zhu Xing-Jun, Zhang Yong-Gang, Gao Jian. Constraint propagation on fail first principle. *Mini-Micro Systems*, 2008, **29**(4): 678−681
(孙吉贵, 朱兴军, 张永刚, 高健. 基于最先失败原则的约束传播算法. 小型微型计算机系统, 2008, **29**(4): 678−681)

14 Sun Ji-Gui, Jing Shen-Yan. Solving non-binary constraint satisfaction problem. *Chinese Journal of Computers*, 2003, **26**(12): 1746−1752
(孙吉贵, 景沈艳. 非二元约束满足问题求解. 计算机学报, 2003, **26**(12): 1746−1752)

15 Gent I P, MacIntyre E, Prosser P, Smith B M, Walsh T. Random constraint satisfaction: flaws and structure. *Constraints*, 2001, **6**(4): 345−372

16 Xu K, Boussemart F, Hemery F, Lecoutre C. A simple model to generate hard satisfiable instances. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, Scotland: Professional Book Center, 2005. 337−342


**朱兴军** 吉林大学硕士研究生. 主要研究方向为约束程序.
E-mail: zhu_xing_jun@163.com
(**ZHU Xing-Jun** Master student at Jilin University. His main research interest is constraint programming.)



**张永刚** 吉林大学副教授, 博士. 主要研究方向为约束程序. 本文通信作者.
E-mail: ygzhang@163.com
(**ZHANG Yong-Gang** Associate professor, Ph. D. at Jilin University. His main research interest is constraint programming. Corresponding author of this paper.)



**李 莹** 吉林大学硕士研究生. 主要研究方向为自动推理. E-mail: liying_21020910@email.jlu.edu.cn
(**LI Ying** Master student at Jilin University. Her main research interest is automated reasoning.)



**张长胜** 东北大学信息科学与工程学院副教授. 主要研究方向为智能信息处理.
E-mail: zcs820@yahoo.com.cn
(**ZHANG Chang-Sheng** Associate professor at the College of Information Science and Technology, Northeastern University. His main research interest is intelligent information processing.)