文章编号：2096-1472(2018)-02-30-04
DOI:10.19644/j.cnki.issn2096-1472.2018.02.010

# 基于启发式的约束满足问题AC系列算法改进研究

蒋李鸣，吕佳宇，何哲华，冯泽斌
(吉林大学软件学院，吉林 长春 130012)

**摘　要**：约束满足问题是人工智能领域中一个重要的研究方向，其研究结果在符号推理、系统诊断、真值维护系统、资源分配和产品配置等问题中有广泛的应用。局部相容性定义了约束满足问题在约束传播过程中必须满足的性质，是约束传播发展的主要方向。而对于较为复杂的相容性问题中的AC系列算法的改进可谓难上之难。本文围绕着以弧相容、Singleton弧相容为代表的相容性技术和求解算法展开，主要针对AC-2001算法、SAC算法等进行优化改进，重点基于启发式进行改进，使之获得了更快的筛选速度。尤其对于SAC算法，大大减少了约束检查次数，获得了较为成功的基于启发式的改进结果。

**关键词**：人工智能；约束满足问题；启发式；AC系列算法；约束检查次数
**中图分类号**：TP3-0　　**文献标识码**：A

# Heuristic Algorithm Based Improvement of AC Algorithms of Constraint Satisfaction Problems

JIANG Liming,Lv Jiayu,HE Zhehua,FENG Zebin
( *College of Software,Jilin University,Changchun* 130012,*China*)

**Abstract:**In the field of artificial intelligence,the Constraint Satisfaction Problem (CSP) is an important research direction,whose study results are widely used in Symbolic Reasoning,System Diagnosis,Truth Maintenance System,Resource Allocation,and Product Configuration and so on.Local consistency,which defines the properties that CSP must satisfy in the process of constraint propagation,is the main development direction of constraint propagation.It is much more difficult to improve the performance of the complex series of AC algorithms about consistency problems.Based on the algorithms and consistency technology represented by arc consistency and singleton arc consistency,the paper focuses on the improvement of AC-2001 algorithm and SAC algorithm.The improvement is based on the heuristic algorithm to achieve higher filtering speed.Especially,the improvement of SAC algorithm is quite successful because of its obvious reduction of constraint checking times.

**Keywords:**artificial intelligence;Constraints Satisfaction Problem;heuristic;AC algorithms;constraint checking times

## 1 引言(Introduction)

约束满足问题(Constraints Satisfaction Problem)[1]是人工智能领域中一个重要的研究方向，目前已广泛地应用于人工智能的各个领域，包括定性推理、基于模型的诊断、自然语言理解、景物分析、任务调度等方面。局部相容性[2]定义了约束满足问题在约束传播过程中必须满足的性质，是约束传播发展的主要方向。一方面，运用相容性技术对约束问题[3]进行预处理，删除大量局部不相容的值，可以极大地压缩问题的解空间[4]；另一方面，在搜索过程中保持局部相容性，可以有效地修剪搜索树[5]，以一定的时空代价换取合理的问题规模的压缩。

本文围绕以弧相容、Singleton弧相容[6]为代表的相容性技术和求解算法展开，研究各个相容性算法。弧相容算法具有相对较少的检查次数，但其运行时间还可以进一步降低。而Singleton弧相容的SAC[1]算法，运行时间较长，并且检查次数较多，尤其在检查次数方面有很大的改进空间。

本文主要针对AC-2001[7]算法、SAC算法等进行优化改进，重点基于启发式进行改进，使之获得更快的筛选速度。尤其对于SAC算法，我们在力图优化其运行时间的同时，大大减少其约束检查次数，获得较好的改进效果。

注：本论文属吉林大学“大学生创新创业训练计划”创新训练项目。由吉林大学软件学院资助。

## 2 相关背景及工作(Backgrounds and tasks)

一个约束满足问题(CSP)涉及对一个约束网络[8](Constraint Network，缩写为CN)进行求解，即满足CN中所有约束的那些参数论域赋值指派。约束确定了给定参数集的子集中被允许的赋值组合。

定义2.1(约束)[9]一个约束c是定义在称为参数域的参数序列X(c)=($x_{i1}$,$x_{i2}$,…,$x_{i|X(c)|}$)上的关系，c是满足其参数值元组$\tau\in Z^{|X(c)|}$的$Z^{|X(c)|}$的子集。|X(c)|称为c的元数，而$Z^{|X(c)|}$是Z的|X(c)|次笛卡儿积。

定义2.2(解)[10]一个CSP的任务是为一个约束网络找到一个或多个解。一个解是使得CN中所有约束都得到满足的一组变量序列的赋值。一个解保证了对所有的约束都存在一个支持。

定义2.3(广义弧相容，(G)AC[11]给定一个约束网络N=(X,D,C)，一个约束c∈C和一个参数$x_i$∈X(c)。

(1)一个值$v_i\in D(x_i)$是与D中c相容的iff存在一个值元组$\tau(v_i=\tau[\{x_i\}])$满足c。值元组τ称为对c上$(x_i,v_i)$的一个支持。

(2)定义域D是c上$x_i$(广义)弧相容的iff $D(x_i)$中所有的值与D中c是相容的(也就是，$D(x_i)\subseteq\pi\{x_i\}(c\cap\pi X(c)(D))$)。

(3)网络N是(广义)弧相容的iff D对C中所有约束上的X中的所有参数是(广义)弧相容的。

定义2.4(Singleton弧相容，SAC)[1]标准约束网络[6]N=(X,D,C)是singleton弧相容的(SAC)iff对所有的$x_i\in X$，对所有的$v_i\in D(x_i)$，子问题$N|_{x_i=v_i}$是弧相容的。如果子问题$N|_{x_i=v_i}$是弧不相容的，则说$(x_i,v_i)$不是SAC的。

在此给出一些符号的定义：AC(P)表示问题P中所有弧不相容的值已被删除，即对P执行了弧相容后的问题；AC(P)=⊥表示P是弧不相容的；SAC(P)表示问题P中所有不是SAC的值已被删除，即P是SAC的。

## 3 几种AC系列经典算法的改进(Improvement of AC Algorithms)

本文主要针对AC-2001算法、SAC算法等进行优化改进，重点基于启发式进行改进，使之获得了更快的筛选速度。相关算法在Microsoft Visual C++ 6.0平台下给出其具体实现，并制作清晰、合理、友好的界面，显示测试结果，方便进行数据统计。保留实现原代码，并提出启发式改进和其他相关算法的改进技巧，优化原代码。生成大量的随机测试用例及标准的Benchmark测试用例对其进行性能测试。最后通过统计数据并画出相应柱状图对相关算法改进前后的性能进行对比，以评估算法改进的价值和成功与失败的方面。

### 3.1 数据结构队列优化原理

据我们观察，原程序中几乎所有的数据结构、存储结构都是用链表list实现的。然而，在代码实现中，大多只用到了对数据结构最前部和最尾部的操作。于是，我们比较了queue和list的性能和实现。代码调试过程中，debug退出后，输出面板会输出大量内存泄漏的信息。经过排查，我们发现原因是使用了std::list！进而，如果把list换成vector或者queue，所有内存泄漏的问题都消失了。list的push_back的实现会导致内部使用大量动态内存分配，而vector也会在动态增长连续内存长度的同时进行内存复制。

然后我们对频繁进行的插入操作进行测试，发现queue的性能明显优于list(无论是运行流畅度还是运行效率)，可见queue是更好的选择。我们又针对STL中的queue,list,vector频繁访问，即插入(尾部)与取出(头部)操作时的效率进行了对比。从时间消耗上来看，queue最少，list与vector消耗相对较大，二者之间差别不大。

查找了相关资料后，了解到queue的底层是对dequeue的适配，那么也就是分段内存实现存储，这样就不会像vector那样需要重新分配内存以及复制元素的操作了。但是list是双向链表，按说实现首尾的插入与删除应该也会很快，为何也相对queue而言很慢呢?

事实上，list的push_back的实现会导致内部使用大量动态内存分配，这个过程也相对耗时，所以导致了相对queue的较慢的结果。

这个数据结构的优化可以运用于所有AC系列算法中。

### 3.2 针对AC_2001::revise2001算法的启发式算法改进

2001年，Bessiere和Regin提出了对AC-3的改进算法AC-2001。AC-2001遵循AC-3同样的框架，但是像AC-6[1]一样，通过对约束上每个值存储一个最小支持[12]实现了最优性。然而，信息存储和使用方式不同于AC-6，AC-2001不使用列表S[$x_j$,$v_i$]来存储以vj作为$c_{ij}$上最小支持的那些$(x_i,v_i)$，而是使用包含vj的指针Last[$x_i$,$v_i$,$x_j$]。

AC-2001仅在 Revise函数和初始化阶段不同于AC-3算法。初始化阶段，AC-2001将Last[$x_i$,$v_i$,$x_j$]初始化为某个比minD($x_j$)更小的虚拟值。在Revise2001中(算法3.4)，当D($x_j$)中一个值vj发现支持$c_{ij}$上的$(x_i,v_i)$时，AC-2001把$v_j$指派给Last[$x_i$,$v_i$,$x_j$](第4行)。下次$(x_i,c_{ij})$将被修改，仅当Last[$x_i$,$v_i$,$x_j$]不在D($x_j$)中时为$(x_i,v_i)$寻找支持(第2行)。更重要的是，最优性达到了因为D($x_j$)中小于Last[$x_i$,$v_i$,$x_j$]的值不再检查，它们以前已经在被调用Revise2001时被检查失败了(第3行)。

算法AC2001：function Revise for AC2001

function Revise2001(in $x_i$:variable; $c_{ij}$:constraint):Boolean;

begin

1 CHANGE false;

2 foreach $v_i$ $D(x_i)$ s.t.Last[$x_i$,$v_i$,$x_j$] $D(x_j)$ do

3 $v_j$ smallest value in $D(x_j)$greater than Last[$x_i$,$v_i$,$x_j$] s.t.($x_i$,$v_i$) $c_{ij}$;

4 if $v_j$ exists then Last[$x_i$,$v_i$,$x_j$] $v_j$;

5 else

6 remove vi from $D(x_i)$;

7 CHANGE true;

8 return CHANGE;

end

事实上，我们可以采用“每次记录上次处理到的值，下次从该值开始遍历”的思想，通过Last数据结构记录下标，下次遍历不再检查无用信息，而是“智能”地选择合适的位置开始搜索，是一种启发式算法的思想。

### 3.3 检查次数优化原理

经过进一步分析，我们发现，对于那些不满足某一xi的值vi，甚至不需要再第二重循环内部对其进行检查，我们可以在第一重循环之前对其进行预处理，将其存于新开的一个数组中(vector也可，新数组在效率上更佳)。这样，在值分布密度并不那么大的时候，可以极其明显地减少约束检查次数，大幅降低时间复杂度。但此时必须注意Last数据结构中记录的内容发生了改变，并不记录上一次检查到的值本身，而是记录其对应在新的数组中的位置，否则算法运行会出错。这个思想在第二重循环减少检查次数，使得该启发式算法更加“智能”地提高了运行效率。这个思想可以同时运用于AC-2001算法、SAC算法中。

### 3.4 二分启发式优化原理

另外，我们甚至还可以更进一步改进。在某些特定情况下，如约束满足偏序关系，如简单大于、小于关系时(事实上有时确实常遇到这种情况)，约束便满足了二分的性质，即“第一个满足约束的位置”前必然不会存在其他满足约束的位置，并且在之前的值未满足某一约束时，往后遍历的值必然不会满足之前较小的约束。于是这种情况下，我们在查找“下一个的第一个满足约束的位置”的时候，可以采用二分查找的思想，这样相对于普通遍历地检查可以大大减少检查次数，可以将时间复杂度从$O(d^2)$降低至$O(d*\log(d))$!

运用二分的思想时，在数据规模较大、约束分布相对稀疏时，尤其奏效，可以大大减少检查次数，降低时间复杂度。另外，由于SAC算法用到了不少AC-2001算法的方法，事实上对AC-2001算法的优化也是对SAC算法的优化，而SAC算法中可以进一步加入数据结构队列优化和检查次数优化，进一步降低时间复杂度。而原本较慢的SAC算法，更加能体现出我们改进算法的成效，见后实验及结果分析算法改进前后的对比。

### 3.5 时空复杂性分析

我们针对AC-2001算法、SAC算法进行改进，综合运用上述四种改进方法。

对于这两种算法来说，空间复杂度并不会发生改变，都以O(ed)的空间复杂度在二元标准约束网络上实现弧相容性。

运用数据结构队列优化原理，可以减少大量动态分配内存的时间。尤其在插入删除操作比较频繁时，可以大大地降低时间复杂度。而检查次数优化原理，可以减少不必要的约束检查次数，仅针对满足某变量论域的值进行检查而不考虑无关的值。尤其对于变量论域中值的分布较为稀疏时尤其适用，可以超越常数优化地降低时间复杂度。接着，加入启发式算法改进后，至此总的时间复杂度为$O(ed^2)$，以此来实现二元标准约束网络上的弧相容性。而对于大多数实际情况，约束一般满足一定的偏序关系，此时我们可以再在这两个算法中加入二分启发式优化[13]，对于变量论域中的值的约束检查不再采用单纯的遍历，而用二分的方式去进行检查，可以使这一部分的时间复杂度从$O(d^2)$降低至$O(d*\log(d))$。此时，总的时间复杂度从$O(ed^2)$降为$O(ed*\log(d))$,实现了算法改进上的突破。

## 4 实验及结果分析(Experiment and Result Analysis)

### 4.1 实验测试说明

我们使用了两类测试用例，分别是随机生成的二元约束满足问题和可供用户选择的Benchmark测试用例组。

对于随机生成的数据的输入：<N,D,C/$p_1$,T/$p_2$>，其中N表示变量的个数，D表示所有变量论域中的最大值，C表示约束个数，T表示每个约束中不相容的值对，$p_1$表示约束的密度(Destiny)($p_1$=2C/N(N-1))，$p_2$表示每个约束的松紧度(Tightness)(p2=1-T/DD)。选用以下4个随机问题作为预处理过程中弧相容算法的例子。

P1：〈150，50，500 /0.045，1250 /0.5〉 P2：〈150，50，500 /0.045，2350 /0.06〉

P3：〈150，50，500 /0.045，2296 /0.082〉 P4：〈50，50，1225 /1.0，2188 /0.125〉

另外，合法的输入条件为：N！=0，　D！=0，0<Destiny<=1，0<Tightness<1。

### 4.2　AC-2001算法改进前后测试对比

AC−2001算法运行时间、检查次数改进前后对比如图1和图2所示。

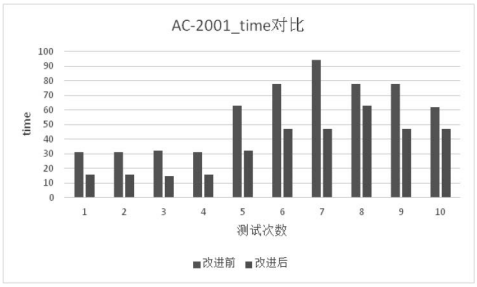


图1 AC−2001算法运行时间改进前后对比

Fig.1 Running time comparison of AC−2001 algorithm

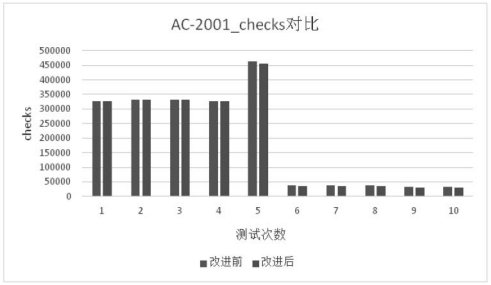


图2 AC−2001算法检查次数改进前后对比

Fig.2 Checking times comparison of AC−2001 algorithm

结果分析：AC−2001算法的改进，针对数据结构优化、启发式算法优化、二分优化、检查次数优化，运行时间明显减少，检查次数只有少量减少。但由于该算法本身优化水平已相当显著，此结果在预期范围之内。更为显著的优化结果，可参照后SAC算法的测试对比。

### 4.3　SAC算法改进前后测试对比

由于SAC算法在某种程度上依赖于AC−2001算法的实现(可参考基础知识和其他相关资料介绍)，故对AC−2001算法的改进在SAC算法中体现得更为明显。可以说对SAC算法的改进与优化是最为成功的。

SAC算法运行时间、检查次数改进前后对比如图3和图4所示。

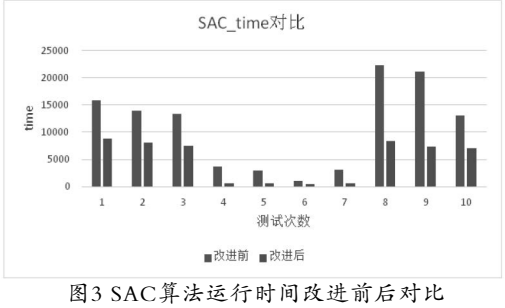


图3 SAC算法运行时间改进前后对比

Fig.3 Running time comparison of SAC algorithm

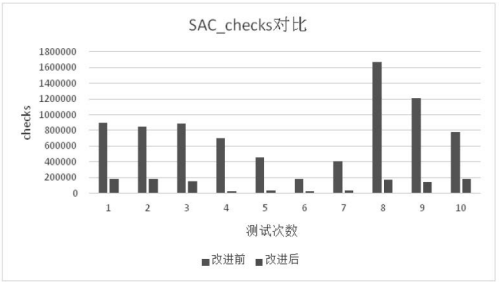


图4 SAC算法检查次数改进前后对比

Fig.4 Checking times comparison of SAC algorithm

结果分析：可见，本系统最为成功之处便在于对SAC算法的优化。实际上是对AC−2001算法的优化影响到了SAC算法。由于SAC算法对AC−2001算法的部分依赖性，导致对AC−2001算法的优化性能在此处得到放大。通过AC−2001算法的数据结构队列优化、启发式算法优化、二分优化、检查次数优化，以及对SAC算法自身的数据结构优化，使SAC算法本身的运行时间大大减少。而图表显示的结果反映了SAC算法不同于其他算法的一个更为明显的优化之处：修改后SAC算法的检查次数大致为修改前的1/10—1/3，最低甚至仅为修改前的1/20不到，平均情况下大约为修改前1/5！这是对当前已经较为完善的约束满足问题的AC系列算法改进的十分令人满意的结果！

## 5　结论(Conclusion)

本文围绕着以约束满足问题中弧相容、Singleton弧相容为代表的相容性技术和求解算法展开，研究各个相容性算法，并主要针对AC−2001算法、SAC算法等进行优化改进，重点基于启发式进行改进，使之获得了更快的筛选速度，运行时间大幅减少。尤其对SAC算法大大减少了约束检查次数，修改后SAC算法的检查次数大致为修改前的1/10—1/3，最低甚至仅为修改前的1/20不到，平均情况下大约为修改前1/5。另外，AC系列算法的效率都得到了大大的提高，获得了较为成功的基于启发式的改进结果。

多年来，众多专家和学者一直致力于提高相容性技术对于约束满足问题的求解效率，他们研究的重心大多是不断地提出局部相容性逐渐增强的技术。未来，随着计算机硬件的快速发展，多核并行是个相当可观的发展趋势，实现各种已有的相容性算法的并行算法对求解约束满足问题有很好的效率，这是个值得研究的新思路。

## 参考文献(References)

[1] Freuder E C,Mackworth A K.Constraint satisfaction:An emerging paradigm[J].Foundations of Artificial Intelligence,2006(2):13−27.

[2] Bessiere C.Constraint propagation[J].Foundations of Artificial Intelligence,2006(2):29-83.

[3] Kask K,Dechter R,Gogate V.Counting-based look-ahead schemes for constraint satisfaction[J].Principles and Practice of Constraint Programming CP 2004,2004:317-331.

[4] 朱兴军.约束求解的推理技术研究[D].吉林大学,2009.

[5] Boussemart F,Hemery F,Lecoutre C,et al.Boosting systematic search by weighting constraints[C].Proceedings of the 16th European Conference on Artificial Intelligence.IOS Press,2004:146-150.

[6] Van Hentenryck P,Van Hentenryck P.Constraint satisfaction in logic programming[M].Cambridge:MIT press,1989.

[7] Lecoutre C,Saïs L,Tabary S,et al.Reasoning from last conflict(s)in constraint programming[J].Artificial Intelligence,2009,173(18):1592-1614.

[8] Grimes D,Wallace R J.Learning from failure in constraint satisfaction search[C].Learning for Search:Papers from the 2006 AAAI Workshop,2006:24-31.

[9] Tsang E.Foundations of constraint satisfaction[J].London: Academic,1993.

[10] 郭劲松.约束满足问题(CSP)的求解技术研究[D].吉林大学,2013.

[11] 徐均哲,孙吉贵,张永刚.弧相容算法性能比较[J].吉林大学学报:信息科学版,2007,25(2):177-182.

[12] Debruyne R,Bessiere C.From restricted path consistency to max-restricted path consistency[J].Principles and Practice of Constraint Programming-CP97,1997:312-326.

[13] 刘春晖.基于约束传播的约束求解方法研究[D].长春:吉林大学,2008.

## 作者简介：

蒋李鸣(1997-)，男，本科生.研究领域：计算机算法理论.

吕佳宇(1996-)，男，本科生.研究领域：计算机算法理论.

何哲华(1998-)，女，本科生.研究领域：计算机算法理论.

冯泽斌(1996-)，男，本科生.研究领域：计算机算法理论.

(上接第37页)

给出了振动数据的XML标识，设计了振动数据的XML架构(VibrationTestData.xsd)，对振动数据的XML应用和其他数据的XML架构设计具有借鉴作用。

## 参考文献(References)

[1] 王富海,韩引海,杨帆.基于XML的温盐深数据Schema设计[J].软件工程师,2013(10):59-60;58.

[2] 秦艳.基于XML的海洋水文调查数据交换研究[D].中国海洋大学,2008.

[3] 张学敏.XML设计方法研究[D].武汉理工大学,2006.

[4] 王霜.基于Schema文档的XML文档验证系统的设计[J].沈阳师范大学学报(自然科学版),2010,28(02):229-232.

[5] 张伟,苑迎春,王克俭.DTD与Schema简介[J].现代电子技术,2001(06):75-79.

## 作者简介：

王富海(1984-)，男，硕士，工程师.研究领域：数据库管理与振动测试工作.

李伟峰(1980-)，男，硕士，工程师.研究领域：地球空间信息可视化.

(上接第40页)

学学报,2016,39(02):25-29.

[10] LeCun Y,Bengio Y,Hinton G.Deep learning[J].Nature,2015,521(7553):436-444.

[11] 黄立威,刘艳博,李德毅.基于深度学习的推荐系统[J].计算机学报,2017:1-29.

[12] 王永固,邱飞岳,赵建龙,等.基于协同过滤技术的学习资源个性化推荐研究[J].远程教育杂志,2011,29(03):66-71.

[13] 朱夏,宋爱波,东方,等.云计算环境下基于协同过滤的个性化推荐机制[J].计算机研究与发展,2014,51(10):2255-2269.

[14] 吕学强,王腾,李雪伟,等.基于内容和兴趣漂移模型的电影推荐算法研究[J].计算机应用研究,2018(03):1-2.

[15] Wang S,Wang Y,Tang J,et al.What your images reveal:Exploiting visual contents for point-of-interest recommendation[C].Proceedings of the 26th International Conference on World Wide Web.International World Wide Web Conferences Steering Committee,2017:391-400.

[16] Huang W,Wu Z,Chen L,et al.A Neural Probabilistic Model for Context Based Citation Recommendation[C].AAAI,2015:2404-2410.

[17] Covington P,Adams J,Sargin E.Deep neural networks for youtube recommendations[C].Proceedings of the 10th ACM Conference on Recommender Systems.ACM,2016:191-198.

## 作者简介：

刘 灿(1993-)，女，硕士生.研究领域：深度学习.

任剑宇(1994-)，男，本科生.研究领域：软件工程.

李 伟(1996-)，男，本科生.研究领域：计算机应用.

张强强(1997-)，男，本科生.研究领域：计算机应用.