doi:10.11959/j.issn.1000-436x.2015123

求解约束满足问题的改进蚁群优化算法

张永刚[1,2]，张思博[1,2]，薛秋实[1,2]
（1. 吉林大学 计算机科学与技术学院，吉林 长春 130012；
2. 符号计算与知识工程教育部重点实验室，吉林 长春 130012）

摘 要：为了克服传统的回溯算法在求解大型的约束满足问题时效率低，难以在合理的时间内求解这一问题。提出了基于启发式搜索的不完备性算法。结合不同算法特性，主要在蚁群优化元启发式约束求解算法的基础上提出了改进：一是在搜索之前用弧相容检查进行预处理以压缩搜索空间，二是提出了一种新的蚁群算法参数设置方案，提高算法的适应性。最后将改进后的算法应用于求解随机问题和组合优化问题。实验结果表明，改进后的算法求解效率得到大幅度提高。

关键词：约束满足问题；蚁群算法；弧相容；参数调节
中图分类号：TP18 **文献标识码**：A

Improved ant colony optimization algorithm for solving constraint satisfaction problem

ZHANG Yong-gang[1,2], ZHANG Si-bo[1,2], XUE Qiu-shi[1,2]
(1. College of Comprter Science and Technology, Jilin university, Changchun 130012, China;
2. Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China)

Abstract: The traditional backtracking algorithm was less efficient on solving large-scale constraint satisfaction problem, and more difficult to be solved within a reasonable time. In order to overcome this problem, many incompleteness algorithms based on heuristic search have been proposed. Two improvements based on ant colony optimization meta-heuristic constraint solving algorithm were presented: First, arc consistency checks was done to preprocess before exploring the search space, Second, a new parameter setting scheme was proposed for ant colony optimization to improve the efficiency of the search. Finally, the improved algorithm is applied to solve random problems and combinatorial optimization problems. The results of the experiment have showed its superiority.

Key words: constraint satisfaction problems; ant colony optimization; arc consistency; parameter adjustment

1 引言

在人工智能领域中，约束满足问题(CSP, constraint satisfaction problems)[1]是其重要组成部分，现今也被广泛应用于各领域的实际问题，如工作调度、资源配置、模式识别和机器视觉等。对约束满足问题进行求解主要包含：推理和搜索。推理技术以约束传播[1]为核心，目的是通过对某些特征节点进行传播来压缩搜索空间，从而提高搜索效率；搜索技术以回溯[2]为核心，在理论上回溯方法是一种完备算法，它能够求解所有约束满足问题。当求解实际问题时，如果 CSP 实例困难性较高，规模较大时，使用回溯方法求解的效率就会明显降低，出现难以在合理的时间内求解的情况。为了解决这一问题，人们提出了许多改进方法。这些方法力求在提高求解效率和降低求解成本之间得到较好权衡。希望不但可以实现最小的执行代价和最少的使用空间，又能达到提高效率的效果。本文进行的研究主要基于蚁群优化元启发式[3]的约束求解算法展开。蚁群优化算法是一种搜索算法，根据完备性可将其

收稿日期：2014-05-16；**修回日期**：2014-11-20
基金项目：国家自然科学基金资助项目（61170314, 61373052）
Foundation Item: The National Natural Science Foundation of China (61170314, 61373052)

定义为不完备搜索算法，即无法判定该类问题是否存在最优解。它的优点是能快速求出近似最优解（通常也能求出最优解），在实际应用中非常有效。另外，蚁群优化元启发式的效率不依赖于具体问题，所以有很强的通用性，称它为问题独立的启发式。蚁群算法通常在某个问题上求解效率可能不如专用的算法，但是在多种问题的平均求解效率上有很大优势。这一特点在事先不知道问题特性或不知道应该选择哪种专用算法时非常有意义，并且实现简单，在复杂的实际应用环境下可以极大地节省成本。

蚁群算法的约束求解器(ant-solver)[4]对于静态的组合优化问题沿用了经典蚁群优化算法。每一次循环构建一个完备的分配后将信息素进行更新。当已经找到了一种解决方案，或者当已经执行到最大循环次数时该迭代算法停止。通过信息素引导整个搜索过程，作为一种启发式引导变量选择值进行初始化。其中，通过设置信息素的上界和下界来尽量避免信息素表示大小和搜索空间实际大小之间的过度差异,从而试图使用较低的花销尽量搜索到最大的搜索空间。一个约束满足问题 CSP(X, D, C)，其中，X 为变量的集合，设 X 中变量数量是 n，关系图中顶点数为 q，则

$$q = \sum_{x_i \in X} \left| D(x_i) \right| \tag{1}$$

当构建一个完整的分配，所有转移概率的信息素值计算需要 $O(nq)$次操作。在每一次循环结束后，信息素的更新需要 $O(n^2)$次添加已访问节点的信息素浓度和 $O(q^2)$次挥发。对于其他不涉及到信息素的操作，如变量的选择、启发式因子的计算，时间复杂度主要根据约束是全局约束或者二元约束等有所不同。实验证明，这种不完备的仿生随机算法，在大规模实例的情况下，虽然无法保证最优解，但是比确定的算法更有效，可以在极短的时间内求出近似最优解，实际应用中更有意义。

本文在约束求解器 Ant-Solver 的基础上提出了改进：第一点改进是在整个搜索过程执行之前用弧相容检查[1]进行一次预处理；第二点改进是提出了一种新的蚁群优化算法参数调节方案。弧相容预处理可以删除搜索空间中的冗余节点，一方面可以在保持问题语义的同时降低搜索难度，另一方面可以减小启发式陷入局部最优的可能性；参数对蚁群算法的效率有关键性地影响，合理的参数设置是高效求解问题的必要条件，通过分析各参数在算法运行过程中所起到的作用，设计了一套参数的在线调解方案—预定参数调节。最后，将得出的算法应用于实际问题，将通过优化改进的 AC-AS(基于弧相容的蚁群算法约束求解器)算法应用于组合优化问题，并且与 Ant-Solver 算法进行了比较，通过实验得出结果：AC-AS 算法在收敛速度、稳定性及最优解的质量方面明显优于 Ant-Solver。

完备性算法[5]理论上可以完美求解所有组合优化问题，在存在最优解的情况下，一定可以求出有解，在无解的情况下可以证明无解。但是在实际应用中，很多大规模的复杂问题，用完备算法的求解代价很高，甚至不能在合理的时间内完成。因此，不完备算法被提出来。由于实际应用中，最优解并不是必须的，在短时间内求出近似最优解更有实际意义，这就体现出不完备算法的实用性。大量实验表明，蚁群算法是一种优秀的不完备算法，不仅具有求解速度快准确率高的优点，而且可以在较短时间内求出符合精度要求的近似最优解。本文是在蚁群算法上做出改进，进一步提升了算法的求解速度和准确性。

2　AC-AS 算法

AC-AS 算法是基于 Ant-Solver 而来的改进算法，主要加入了弧相容预处理和预定参数调节，并重新设计了数据结构。Solnon 在蚁群优化元启发式的基础上结合最大最小蚂蚁系统和局部搜索等技术设计了 CSP 求解器 Ant-Solver。Solnon 将 CSP 抽象为一个无向约束图，在抽象得到的约束图中：每一个顶点代表一个变量−值对，2 个顶点之间的连线（边）代表这 2 个顶点所代表的变量之间的约束关系。在约束图中对所有变量进行一次遍历的路径与 CSP 的候选解路径一一对应，这与使用蚁群算法求解最短路径的思路相吻合，因此可以利用这种思路，求出所有在约束图中的约束所满足的解路径，该解路径就是最终所求 CSP 的解。Ant-Solver 的优点是数据结构简单，容易实现且易于理解，求解效率较高；缺点是扩展性不足，所采用的不是 Benchmarks[1]中的数据结构，并且只能求解文献[4]中给出的实例或者自行设计测试用例，非常麻烦。本文重写了算法的数据结构，一方面使算法可以直接读取 Benchmarks 中的实例，方便测试和交流；另一方面也为加入弧相容预处理打下了基础。

Ant-Solver 中信息素因子代表评价某个赋值的

“经验”偏好，启发式因子代表局部评价赋值的效果。Ant-Solver 通过对信息素因子和启发式因子之间相关概率的随机赋值，为求解 CSP 提供了对索引进行引导的一个可通用的启发式。现在假设一只蚂蚁已经访问过的节点的集合为 A，下一个将被赋值的变量为 x_j。这只蚂蚁按照一定概率随机选择下一个节点，其中，节点$\langle x_j,v\rangle$称为目标节点，目标节点对其中所有候选节点$\langle x_j,\omega\rangle$（$\omega\in D(x_j)$）吸引力的比例即为转移概率[4]

$$P_A\left(\langle x_j,v\rangle\right)=\frac{\left[\tau_A\left(\langle x_j,v\rangle\right)\right]^{\alpha}\left[\eta_A\left(\langle x_j,v\rangle\right)\right]^{\beta}}{\sum_{\omega\in D(x_j)}\left[\tau_A\left(\langle x_j,\omega\rangle\right)\right]^{\alpha}\left[\eta_A\left(\langle x_j,\omega\rangle\right)\right]^{\beta}} \tag{2}$$

其中，$\tau_A\langle x_j,\omega\rangle$是$\langle x_j,v\rangle$的信息素因子，启发式因子是$\eta_A\left(\langle x_j,v\rangle\right)$，$\alpha$与$\beta$是相对权重的决定参数。当为$x_j$选择一个值时，在 A 中所有之前储存的赋值的重要程度都是相同的。因此，信息素因子$\tau_A\langle x_j,v\rangle$取决于$\langle x_j,v\rangle$和在 A 中所有节点间的边上面的信息素浓度

$$\tau_A\langle x_j,v\rangle=\sum_{<x_k,m>\in A}\tau_A(\langle x_k,m\rangle,\langle x_j,v\rangle) \tag{3}$$

启发式因子$\eta_A\left(\langle x_j,v\rangle\right)$也依赖于集合 A 中的所有节点。它与$x_j=v$时所产生的冲突数（不满足约束的值对数）增量成反比

$$\eta_A\left(\langle x_j,v\rangle\right)=\frac{1}{1+\mathrm{cost}\left(\left\{\langle x_j,v\rangle\right\}\cup A\right)-\mathrm{cost}(A)} \tag{4}$$

为了避免过早地收敛，使蚂蚁能够探索更多新的解路径，SACO 在每次迭代之后引入了挥发机制，信息素更新之前，边上的所有信息素会按照一定比例挥发

$$\tau_{ij}(t)\leftarrow(1-p)\tau_{ij}(t) \tag{5}$$

下面讨论 AC-AS 算法对 Ant-Solver 的第一个主要改进：预定参数调节。从式(1)可以看出，参数α,β对算法有直接影响，以下 2 个指标：成功率和运行时间，其中，成功率代表算法成功求解的问题占所有问题的百分比，运行时间代表算法从执行开始到执行终止所用的时间。当对这 2 个指标进行考察时，其中有 5 个参数α、β、ρ、$nbAnts$、q_0起到决定性作用。对参数进行设置也一直是蚁群算法研究中的核心问题。参数问题在方法上大致可以分为 2 种方法：离线调节方法和在线调节[6]方法。离线调节希望可以在算法还未真正开始工作以前找到一个针对于该问题较为适合的设置参数方案。这一过程相对较为耗时，且需要人为干预，整个算法主要依赖于开发人员的直觉和经验，容易出错，且难以通用和复用。比离线调节更有前景的另一个替代方案称为在线调节方法，该方法通常在问题实例的求解过程中持续不断地修正对参数的设置。在线调节的优点一是可以适应不均衡的实例，二是可以依据搜索的阶段特征来改变参数设置方案。本文提出的预定参数调节即是一种在线调节方案。方案的设计思路来自于对每个参数的具体作用分析。首先看α，假设对解图的搜索进行到了图中的某一节点v_i处，即已对v_i代表的变量进行了随机赋值操作，从v_i出发的候选节点集合（假设v_i的候选节点集合中元素不少于 2 个）中候选节点被选中的概率满足式(1)，继而得出任意 2 个节点v_i和v_k被选中的概率比为

$$\frac{P_{v_j}}{P_{v_k}}=\frac{(\tau_j{}^{\alpha}\eta_j{}^{\beta})}{(\tau_k{}^{\alpha}\eta_k{}^{\beta})}=\left(\frac{\eta_j{}^{\beta}}{\eta_k{}^{\beta}}\right)\left(\frac{\tau_j{}^{\alpha}}{\tau_k{}^{\alpha}}\right)\propto\frac{\tau_j{}^{\alpha}}{\tau_k{}^{\alpha}}=\left(\frac{\tau_j}{\tau_k}\right)^{\alpha} \tag{6}$$

即

$$\frac{P_{v_j}}{P_{v_k}}\propto\left(\frac{\tau_j}{\tau_k}\right)^{\alpha} \tag{7}$$

当$\tau_j/\tau_k\neq1$时，α的值越大，P_{v_j}与P_{v_k}相差的倍数越多，也就是v_j与v_k这 2 个后选节点分别被选中的概率相差就越大，同时信息素τ在转移概率中的权重也就越大。边界情况$\tau_j/\tau_k=1$属于极特殊情况，它对问题讨论的普遍规律并无意义。通过以上针对性的分析，在搜索过程中信息素τ起到的作用随着参数α的增大而变大。在算法中信息素代表了对已探路径的“记忆”，根据信息素就可以得出算法对“经验”依赖程度，即在搜索过程中信息素起到的作用越大就代表着算法对“经验”的依赖就越强，同时算法快速地收敛于信息素所引导的路径。

$$\frac{P_{v_j}}{P_{v_k}}\propto\left(\frac{\eta_j}{\eta_k}\right)^{\beta} \tag{8}$$

同理可得β对算法的影响，式(6)明显可以看出在搜索过程中，启发式η所起的作用随着参数β的

增大而变大。由于启发式与“经验”无关，仅代表 v_i 附近的局部信息，所以针对局部信息的依赖性可得出在算法中随着启发式所起作用的增大对局部信息的依赖增强，同时算法快速的收敛于局部最优组成的路径。阈值 q_0 满足 $0<q_0<1$，为当前节点选择后继节点时算法就会随机生成一个值 $q\sim U(0,1)$，当生成值 q 低于阈值 q_0 时算法就会直接将转移概率最大的节点取出，而当所选择的值高于 q_0 时算法则按照转移概率随机选取节点。阈值 q_0 的大小也影响着算法，算法在为当前节点选择后继节点时，q_0 越大，选取最大转移概率节点的可能性就越大。相对于采用转移概率随机选取的方法而言，该方法依赖过去的“经验”较多，而随机性较少。相对于参数 α 与 β 的设定，调节 q_0 属于对算法进行微调，原因是阈值 q_0 减小，过往解路径的子路径组合会受到影响，即阈值 q_0 越小就会使不同的经验解路径的交叉变得越频繁，同时也影响了算法的经验路径和探索路径，使得探索路径在经验路径附近加大了摆动振幅。

其中参数 ρ 用来对信息素的“挥发”速率进行控制，ρ 所起的作用随着自身值的增大而变小，即 ρ 越大，信息素就挥发得越快，那么它起到的作用就越小。由于调节 ρ 实际上是对信息素积累速度的调节，所以这也属于对算法的微调，间接影响信息素的引导作用。事实上，由于相对问题而言信息素是完全独立的，所以一般情况下不需对 ρ 进行调节，只要设定一个较为合理的初始值即可。参数 *nbAnts* 的作用是对信息素更新的频率进行控制，它决定了算法的循环周期和信息素在更新时所采用样本的解集，同时影响着收敛速度和探索宽度。当 *nbAnts* 变大，信息素的更新就会越慢，信息素所起到的作用就越小（极端情况下，$nbAnts\to\infty$，代表信息素永远不进行更新，算法变成完全的随机探查）。由此可见，*nbAnts* 也是间接影响信息素的引导作用，属于对算法的微调。特别地，在多次循环以后 *nbAnts* 的调节能力就会逐渐变小。由于信息素不断积累，在每一次循环中会得到解路径的样本集合，该样本集合和经验路径的相似度之间进行匹配，得出的相似度越高通过调节 *nbAnts* 产生的影响就越小。通常只在包含重置信息素的算法步骤时，对 *nbAnts* 调节才起作用。

在以上分析的基础上，通过反复实验给出了预定参数调节方案。参数 $\alpha,nbAnts,\rho$ 通常情况下是与问题无关的，对于不同的问题可以使用同种调节方案。而参数 β 通常情况下则依赖于问题特征，原因是不同问题中的启发式因子不同，选择参数 β 的最优调节方案就不同。参数 α,β 共同决定着转移概率中信息素和启发式的权重，事实上对 2 个参数的调节是可以做到等效的，所以在参数 α,β 中，只需选择一个参数即可，而究竟选择哪个参数比较适合则要看两者的调节效率。通常人们更喜欢选择 β，因为实际应用中 α 远小于 β，做同样幅度的调节时，β 增加 1，而 α 可能只减少了零点几，所以出于方便简洁考虑，通常会选择 β。而对参数 *nbAnts* 和 ρ 只需要在开始的时候给出较为合理的初始值即可，所以也不予选择。综上，当 β 很大时，算法会迅速收敛于信息素浓度高的路径，反之，则会迅速收敛于局部最优解路径。从搜索的全局过程分析，初期要采用充分大的 β 值，这样用很小的计算量就可以迅速排除很多不可能的解；随着搜索的深入，减小 β 值，避免算法迅速收敛于次优解。同理，参数 q_0 在搜索初期应该充分大以提高搜索速度，随着搜索的深入逐渐减小，使搜索宽度增加来避免算法陷入局部最优。

AC-AS 算法对 Ant-Solver 的另一个主要改进是加入了弧相容预处理。弧相容检查是经典的约束传播技术，在用回溯搜索算法求解 CSP 时通常要结合约束传播对搜索空间进行压缩来简化问题。受这一思想的启发，本文在蚁群算法搜索之前采用弧相容检查对问题实例进行预处理。这样做有两点好处，第一是简化了问题，使蚁群算法能够更快地求解，第二是修剪掉了搜索空间中的许多无效值，减小了算法收敛于局部最优（这是启发式算法的通病）的可能性。显然，如果预处理过程的开销小于该过程为所有蚁群算法节省的开销，那么总体的效率就能得到提升。因此，在搜索权重和约束传播之间做较好的权衡即可得出较优的求解效率。如果在进行完相容性检测以后就得到了最优解那么算法就等同于约束传播算法，这种极端情况是存在的。

图 1 列出了几种主要约束传播技术的强弱关系[1]，越强的技术删除无效值的能力越强，相应的开销也越大。本文选择了删除能力最弱的弧相容检查用作预处理，这是为了避免在预处理阶段消耗过多的时间。实际上，对不同问题应该采用不同强度的约束传播技术，如何能让算法自适应地选择合适

的约束传播技术，是值得进一步研究的课题。至此，给出了 AC-AS 算法的主要思想，AC-AS 算法的具体描述如下。

1) 对问题进行弧相容性预处理；

2) 设置参数模式并初始化信息素；

3) Repeat until 存在 $i\in\{1\cdots nbAnts\}$ 使得 $const(A_i)=0$ 或达到最大迭代次数

4) for k in $1\cdots nbAnts$ do

5) 构建一组变量赋值 Ak；

6) End for ;

7) 利用 $\{A_1,\cdots,A_{nbAnts}\}$ 更行信息素；

8) End repeat;

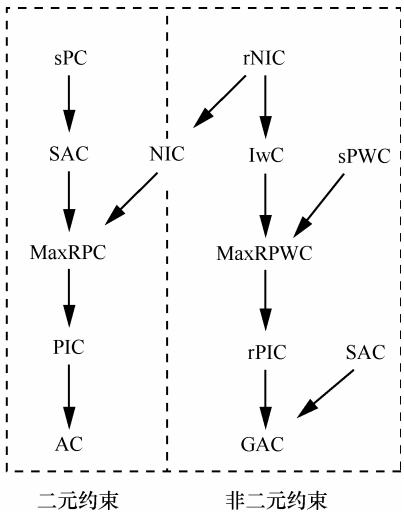


图 1 域过滤相容性检查的强弱关系

算法中的第 1)步进行了预处理操作，通过预处理减掉空间中的无效值，简化了问题，确保求解效率。实验中应用的是面向弧的粗粒度传播方法，主要使用其中的 AC3，因此在预处理阶段时间复杂度是 $O(ed^3)$，空间复杂度是 $O(e)$，其中，有 e 个约束 d 是论域的大小。第 2)步构建一个完整的分配，所有转移概率的信息素值计算需要 $O(nq)$次操作。在每一次循环结束后，信息素的更新需要 $O(n^2)$次添加已访问节点的信息素浓度和 $O(q^2)$次挥发。其后通过合理调节参数进一步提升了算法的求解速度和准确性。

3 AC-AS 算法在随机问题中的应用

在日常生活中所遇到的问题都是具有结构特点的，通过对不同结构问题的研究可以设计出针对不同结构的高效算法。当然有时针对某一问题结构设计的算法实际求解效率可能并不能达到预期效果。这一点不利人们设计出通用的求解算法。那么对无结构的随机 CSP 进行研究就显得尤为必要，并且随机 CSP 测试用例在评估求解器效能时更具说服力。

下面给出生成随机约束问题实例的经典模型[7]。

定义 1 B 模型：B 模型生成随机 CSP 实例为 $B(k,n,d,p_1,p_2)$，对其中的每个实例有：

$k\geqslant 2$ 表示其中每个约束的元数；

$n\geqslant 2$ 表示问题中变量的总数；

$d\geqslant 2$ 表示每个论域的一致大小；

$0\leqslant p_1\leqslant 1$ 决定约束的个数 $e=p_1\binom{n}{k}$；

$0\leqslant p_2\leqslant 1$ 决定每个关系中不满足的元组个数 $t=p_2d^k$。

定义 2 RB 模型： RB 模型生成的一类随机 CSP 实例表示为 $RB(k,n,\alpha,\gamma,p)$，对其中的每个实例有：

$k\geqslant 2$ 表示其中每个约束的元数；

$n\geqslant 2$ 表示变量的总数；

$\alpha>0$ 论域大小的决定参数： $d=n^{\alpha}$；

$\gamma>0$ 约束数量的决定参数： $e=rn\ln n$；

$0<p<1$ 决定每个关系中不满足约束关系的元组数 $t=pd^k$。

RB 模型是在 B 模型的基础上得来的，它不同于 B 模型的区别主要有 2 点：首先变量之间允许存在多个约束关系，即约束可以共享作用域；其次，RB 模型中每个变量的论域大小是随着变量多项式增长的，各自并不相同。本文中所使用的测试用例均来自相变区，因为该区域的问题难度相对较高，具有一定说服力与代表性。关于相变区在文献[8]中指出，欠约束区域与过约束区域之间称作相变区域，相变区域是最难求解的随机实例出现的区域。其中欠约束区域是指在该区域中的所有实例几乎都是可满足的，而过约束区域中实例则几乎都是不可满足问题。

除了上面所介绍的随机模型，还有包含少量结构的准随机问题，常用的有 Ehi、Geometric、Composed 3 种问题，实验所测问题来自于文献[1]中的 Benchmarks。以下所有的实验全部都在同一台 PC 上操作完成的，操作系统为 Microsoft Windows XP Professional Service Pack 3，处理器

为 Intel E7500 双核 CPU，主频为 2.93 GHz，内存为 2 GB。

Ant-Solver 作为实验的参照组，AC-AS 是本文改进后的最终算法。从表 1 和表 2 中可以清楚地看到，AC-AS 算法在随机实例上的表现明显优于 Ant-Solver 算法。

表 1　　B 模型随机 CSP 实例 100-8-14-p_2

随机问题	求解时间/s		冲突数/对	
	Ant-Solver	AC-AS	Ant-Solver	AC-AS
100-8-14-24	18.203	9.500	0	0
100-8-14-25	3.484	3.375	0	0
100-8-14-26	18.829	42.079	2	2
100-8-14-27	6.750	1.437	6	6
100-8-14-28	2.422	1.078	5	5

表 2　　RB 模型随机 CSP 实例 frb

随机问题	求解时间/s		冲突数/对	
	Ant-Solver	AC-AS	Ant-Solver	AC-AS
35-17-1-bis	1.047	0.594	0	0
40-19-1-bis	1.656	1.016	0	0
45-21-1-bis	0.453	8.984	0	1
50-23-1-bis	12.781	3.015	1	0

表 3　　准随机 CSP 实例

随机问题	求解时间/s		冲突数/对	
	Ant-Solver	AC-AS	Ant-Solver	AC-AS
geom-50-20-d4-75	1.594	1.578	0	0
	0.296	0.313	0	0
	2.375	0.578	0	0
composed-25-10-20	1.250	0.375	0	0
	2.203	0.344	0	0
	0.610	0.343	0	0
ehi-85	8.375	1.704	9	9
ehi-90	6.312	2.238	9	9

以表 1~表 3 作为实例，表 1 统计了 AC-AS 和 Ant-Solver 求解 B(2, 100, 8, 0.14, p_2)的实验数据，p_2=0.19~0.28；表 2 统计了求解模型 RB 随机 CSP 实例的实验数据；表 3 统计了求解随机 CSP 实例 geom-50-20-d4-75、composed-25-10-20 和 ehi 的实验数据；从表 1 可以清楚看出，在实例 B(2,100, 8, 0.14, p_2)中 AC-AS 比 Ant-Solver 表现更好，只有 B(2, 100, 8, 0.14, 0.23)和 B(2, 100, 8, 0.14, 0.26)例外。后 3 个实例没有求出解，它们的冲突数在 2 种方案下对问题的无解猜测做出了进一步印证，但是仍然无法确定有无解。表 2 中实例的情况稍复杂，有一半的 AC-AS 比 Ant-Solver 结果要好，其中有些 Ant-Solver 明显好于 AC-AS，这说明 AC-AS 算法存在不足，在某些问题中需要进行微调。表 3 可以看出 AC-AS 表现普遍较好，但是 ehi-85 中不带 AC 预处理参数调节的 AC-AS 的时间为 1.500，低于 AC-AS 的 1.704。说明在极少数情况下预处理中删除的无效值可能会降低与它相关的一些解路径的转移概率。表中有些问题本身并不可满足，本文认为冲突数较多的问题从循环次数上推断（表中未列出）很有可能已得出了最优解。另外，认为结果中冲突数优先级高于求解时间，即冲突数少的解更优，在冲突数相同的情况下再比较求解时间。

4　AC-AS 算法在组合优化问题中的应用

组合优化问题中的应用比较广泛，例如工程、建筑、计算机科学、音乐、理财以及其他许多领域。例如，人们日常生活中所使用的电话，在其实现的过程中就要求通过成本低、路径堵塞小和智能化等系列优化问题来最终实现较高效能；汽车制造也有很多的优化目标：减小风阻、提高安全系数、人性化舒适体验等。组合优化问题可以用 CSP 来表示，因此可以用 AC-AS 来求解组合优化问题。另外，由于组合优化问题来自于现实中的应用，通常都带有一定的结构特点，求解组合优化问题可以考察 AC-AS 求解结构化问题的性能。以下实验均在同一台 PC 上完成：处理器为 Intel E7500 双核 CPU、主频为 2.93 GHz、内存为 2 GB、操作系统为 Microsoft Windows XP Professional Service Pack 3。

表 4 中全部实例取自文献[1]中的 Benchmarks，包括 QCP[9]、RLFAP[10]、JSSP[11]、BH[12]等，显然 AC-AS 算法的表现明显优于 Ant-Solver，仅在 bqwh 问题上 Ant-Solver 有微弱的优势。可见，AC-AS 算法对 Ant-Solver 的改进是非常显著的。可以预期，未来可以进一步改进算法，使算法能够更加自适应地选择合适的约束传播技术和参数调节方案，继续提高 AC-AS 算法的效率。

表 4 组合优化问题实例

组合优化问题		求解时间/s		冲突数/对	
		Ant-Solver	AC-AS	Ant-Solver	AC-AS
domino	1000-10	5.625	0.125	0	0
	100-100	3.578	1.438	0	0
	100-200	18.641	11.188	0	0
bqwh	15-106	0.297	0.688	0	0
	18-141	0.625	2.328	0	0
hanoi	6	0.125	0.109	0	0
	14	0.140	0.125	1	0
	30	1.610	0.266	1	0
qwh	10	0.203	0.141	0	0
	15	1.547	1.375	0	0
	20	12.687	13.469	2	0
	p10	0.187	0.140	0	0
	p15	0.938	0.328	0	0
qcp	10	0.203	0.157	0	0
	15	7.000	2.203	0	0
	20	51.328	43.547	0	0
	p10	0.218	0.172	0	0
	p15	5.813	0.375	0	0
	p20	26.797	6.984	0	0
qa	26	0.360	0.219	0	0
	37	8.734	0.937	0	0
	50	24.875	8.718	0	0
rlfap	mGraph	51.188	28.687	1	0
	mScen	22.797	12.734	1	0
bh	4-7	1.422	0.829	11	10
	4-13	27.39	16.922	14	14

5 结束语

本文工作主要体现在以下几个方面：①重新设计并实现了 Ant-Solver 的数据结构和接口，使得算法可以加入弧相容预处理，并可以处理所有 Benchmarks 中的问题实例；②提出了一种在线参数调节方案——预定参数调节，大幅提高了算法的性能并提升了算法的求解成功率；③为 Ant-Solver 算法加入了弧相容预处理步骤，进一步提高了算法的性能，最终得到了高效的 AC-AS 算法；④将此算法用于求解随机约束满足问题和组合优化问题，并与 Ant-Solver 进行了实验对比，验证了算法的有效性。

参考文献：

[1] LECOUTRE C. Constraint Networks: Techniques and Algorithms[M]. London: John Wiley & Sons, Inc, 2009.

[2] GOLOMB S W, BAUMERT L D. Backtrack programming[J]. Journal of the ACM, 1965,(12): 51-524.

[3] DORIGO M. Optimization, Learning and Natural Algorithms[D]. Politecnico di Milano, 1992.

[4] SOLNON C. Ants can solve constraint satisfaction problems[J]. IEEE Transactions on Evolutionary Computation, 2002,6(4):347-357.

[5] CORMEN T H, *et al*. Introduction to Algorithms[M]. Cambridge MIT Press, 2001.840-890.

[6] EIBEN A E, MICHALEWICZ Z, SCHOENAUER M, *et al*. Parameter Control in Evolutionary Algorithms[M]. Springer Berlin Heidelberg, 2007.

[7] SMITH B, DYER M. Locating the phase transition in binary constraint satisfaction problems[J]. Artificial Intelligence, 1996, 81(1): 155-181.

[8] CHEESEMAN P, KANEFSKY B, TAYLOR. Where the really hard problems are[A]. Proceedings of IJCAI[C]. 1991. 331-337.

[9] GOMES C, SHMOYS D. Completing quasigroups or latin squares: a structured graph coloring problem[A]. Proceedings of Computational Symposium on Graph Coloring and Generalization[C]. 2002. 22-39.

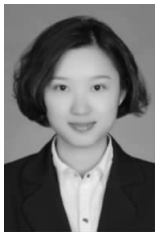
[10] CARBON B, *et al*. Radio link frequency assignment[J]. Constraints, 1999, (4): 79-89.

[11] SADEH N, FOX M S. Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem[J]. Artificial Intelligence, 1996, 86(1):1-41.

[12] PARLETT P. The Penguin Book of Patience[M]. Penguin Press, 1996.

作者简介：

张永刚（1975-），男，辽宁沈阳人，吉林大学副教授、硕士生导师，主要研究方向为约束求解与约束优化。



张思博（1989-），女，吉林长春人，吉林大学硕士生，主要研究方向为约束求解与约束优化。

薛秋实（1986-），男，吉林长春人，硕士，主要研究方向为约束求解与约束优化。