弧一致性符号ADD算法及在CSP求解中的应用

王腾飞 徐周波 古天龙
(桂林电子科技大学广西可信软件重点实验室 桂林 541004)

摘 要 约束满足问题(CSP)是人工智能领域中一个重要的研究课题,弧一致性(AC)技术是提高约束满足问题求解效率的一种有效技术。对传统弧一致性技术进行了改进,给出了弧一致性的符号代数决策图(ADD)算法并将其应用于CSP求解。传统弧一致性技术在压缩问题的搜索空间时,一次只能处理一条约束上的一个值对;而借助ADD技术来压缩问题搜索空间,可以一次处理多条约束。算法首先通过01编码将CSP问题描述成伪布尔函数,并由ADD进行表示。然后基于传统弧一致性技术的算法思想,利用ADD的交、并和提取操作来实现约束传播和变量域过滤。最后将弧一致性的符号ADD算法嵌入到BT搜索算法中来实现对CSP的求解。对标准库中的测试用例以及随机生成的测试用例进行了实验仿真,结果表明,该算法求解CSP的时间既优于带弧一致性维护的回跳算法MAC3+BJ和MAC2001+BJ,也优于采用传统数据结构进行预处理的CSP求解算法BT+MPAC和BT+MPAC*。

关键词 约束满足问题(CSP),代数决策图(ADD),弧一致性(AC)
中图法分类号 TP301 **文献标识码** A

Symbolic ADD Algorithms for Arc Consistency and Application in Constraint Satisfaction Problem Solving

WANG Teng-fei XU Zhou-bo GU Tian-long
(Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract Constraint satisfaction problem (CSP) is an important research topic in the field of artificial intelligence, and arc consistency (AC) is an effective technology to improve the CSP solving efficiency. The symbolic algebraic decision diagram (ADD) algorithm for arc consistency was proposed here to improve the traditional arc consistency technology and was applied in CSP solving. In this paper, ADD was used to compress the search space. It can process multiple constrains in one time, not as the traditional algorithm which can only deal with one value pair in a constraint per step. Fistly, CSP was described as pseudo-Boolean function by 01 coding and represented by ADD. Secondly, based on traditional arc consistency algorithm, the ADD operations intersection, union and abstraction were used to achieve constraint propagation and variable domain filtering. Finally, the symbolic algebraic decision diagram (ADD) algorithm for arc consistency was embedded in BT search algorithm to solve the CSP. The result of experimental simulation to solve some problems in benchmark and randomly generated test case shows that the efficiency is higher than backjumping algorithm maintained with AC, such as MAC3+BJ and MAC2001+BJ, meanwhile the performance is better than algorithms BT+MPAC and BT+MPAC* which is based on AC preprocess realized by traditional data structure in CSP solving.

Keywords Constraint satisfaction problem(CSP), Algebraic decision diagram(ADD), Arc consistency(AC)

1 引言

约束满足问题(Constraint satisfaction problem, CSP)[1]是人工智能的一个重要研究领域,它不仅可用来求解地图着色、N皇后问题等学术问题,还可广泛应用于现实场景,如调度中的资源分配、时序推理等。通常求解约束满足问题是NP难的,尤其当问题规模很大时,求解尤为困难。在约束满足问题的众多求解算法中,基于回溯的搜索算法(Backtracking algorithm, BT)[2]是一个有效完备的算法。该算法在实例化变量时,若发现当前取得的变量取值与由已实例化的变量组成的部分解存在冲突,则回溯到上一个变量重新实例化。原始的回溯算法效率较低,通过引入展望或者回顾模式,可有效提高搜索效率。基于冲突的回跳算法(Backjumping, BJ)[3]和动态的回溯算法[4]等是回顾模式类的搜索算法;基于弧一致性技术的算法是展望模式类的搜索算法。而弧一致性(Arc Consistency, AC)则是众多一致性算法中一种高效的一致性技术[5]。

对弧一致性的应用主要有两个方面:一是在搜索的过程中保持弧一致性,对不满足弧一致性的分支不予扩展,因此可提高搜索效率;二是在搜索前根据弧一致性进行预处理,删除

到稿日期:2013-02-05 返修日期:2013-05-10 本文受国家自然科学基金(60963010,61100025,61262030),广西研究生教育创新计划(2011105950812M23)资助。

王腾飞(1985-),男,硕士生,主要研究方向为符号计算,E-mail:wtf1985@sina.com;徐周波(1976-),女,副教授,主要研究方向为符号计算、智能规划;古天龙(1964-),男,教授,博士生导师,主要研究方向为形式化方法、符号计算、知识工程等。

变量中肯定不参与解的值，通过缩减问题规模达到提高求解效率的目的。前人已对弧一致性技术及其实现方法做了大量研究和改进工作[15-17]，按算法单步执行单位进行分类，大体可分为针对弧进行连续修正的粗粒度算法和针对值进行连续修正的细粒度算法。

AC-1，AC-2，AC-3[6]和 AC-2001[7]属于粗粒度算法。AC-1 的实现思想最为简单，它根据弧一致性的定义，将所有约束放入约束队列 Q 中，对约束逐个进行检查，当发现变量 i 的取值 a 在约束的其他变量中没有支持，将 a 从变量 i 的论域中删除，并检查所有约束，任何变量论域的变化都将引起所有约束的重新检查，因而较低效；AC-2 和 AC-3 是对 AC-1 的改进，当变量 i 的论域发生变化时，它只重新检查有变量 i 参与的约束；AC-2001 是 AC-3 的改进，主要是通过维护一定的数据结构，减少约束传播后为变量论域中的值重新寻找支持时需要进行的一致性检查次数。

AC-4[8]，AC-6，AC-7 属于细粒度算法，它们维护的队列 Q 存储的是赋值 (i,a) 而不是约束，是已经被删除需要进行约束传播的变量的取值。AC-6 和 AC-7 都是基于 AC-4 的改进，通过增加更多数据结构记录支持的信息来减少一致性检查的次数。

前人已有实验证明，同样在搜索前进行弧一致性检查，细粒度算法在弧一致性检查次数上总体优于粗粒度算法，但是由于空间复杂度较高，细粒度算法并不适合于在搜索过程中保持弧一致性，而粗粒度算法则能较好适用[9]。以上两类传统 AC 算法尽管算法单步处理的对象不同，但都是逐个处理的，若能在一次算法循环内并行处理多条约束，势必可提高 AC 算法的执行效率，故而本研究尝试引入有序二叉决策图及其扩展形式。

有序二叉决策图(Ordered Binary Decision Diagram，OBDD)[10,11]是一种有效的图形、数学描述技术，是表述布尔函数的规范型，在硬件验证、模型检验、组合优化等多个领域有广泛应用[10-14]。在实际应用中，将问题转化为可运算的布尔函数表达形式并用 OBDD 描述，不仅表示方式更为紧凑，运算方式也相应地转化为简单易行的 OBDD 图形操作的组合。代数决策图(Algebraic Decision Diagram，ADD)[10]是 OBDD 的一种扩展结构，除具有 OBDD 的优点外，其适用范围更为广泛。在对约束满足问题的求解方面，OBDD 及其扩展结构高紧凑性的特点可以降低空间需求，集合操作的特点使得对于多条约束的处理可以并发执行，因而也是一种行之有效的技术。

本文的工作是用符号 ADD 实现 AC 算法，根据 ADD 数据结构的特点，首先，算法执行的单位是变量，与变量 i 有关的所有约束在算法的单步执行中被并发处理，是一种更粗粒度的算法；其次，变量论域中的每个取值寻找支持也是并发进行的，不需过多优化，基本不需要借助辅助结构；再者，要将 ADD 形式实现的 AC 算法运用在预处理和搜索过程两个阶段。综合考虑以上因素，本文算法选择 AC-3 作为理论基础。

Gu 和 Xu 已经在其论文中给出了加权约束满足问题的 ADD 描述[15]，即采用结点一致性技术进行预处理，并在搜索过程中决定变量 i 的某个取值是否可以扩展到当前部分解中时，对与 i 没有约束关系的未实例化变量采用有向弧一致性计数技术，进一步提高搜索下界，以及早发现不满足一致性的取值。该算法具有较好的求解性能，用实践证明了符号算法在求解 CSP 问题方面具有一定的优势。本文的思路是在 Gu 和 Xu 已经给出的约束满足问题的 ADD 表示基础上，给出弧一致性符号 ADD 算法，一方面，可对问题进行弧一致性预处理；另一方面，Gu 和 Xu 在搜索过程中对一个变量 i 的每个取值，都要对未实例化的变量进行有向弧一致性计算，而不管 i 的取值与这些变量是否满足弧一致性，若 i 的取值 a 对某变量 j 不满足弧一致性，这么做显然是多余的，在搜索中应用本算法进行弧一致性检查可避免这类操作。通过对大量随机生成的测试用例和一些标准库用例进行实验分析表明，本文算法在求解随机生成的测试用例时，性能上优于传统的 MAC3+BJ(Backjumping)搜索求解算法和 MAC2001+BJ 算法，对部分标准库的用例测试，求解时间也优于现有的用传统数据结构预处理的约束满足问题求解技术。

2 CSP 的弧一致性定义和 ADD 描述

一个二元约束满足问题(Constraint Satisfaction Problem，CSP)[1]通常表示成一个三元组 $P=(X,D,C)$，其中：

(1)$X=\{1,\cdots,n\}$是一组变量的有限集合；

(2)每个变量 $i\in X$ 都有一个有限的域 $D_i\in D$，这个域包含可以赋给 i 的取值，(i,a)就表示将值 $a\in D_i$ 赋给变量 i；

(3)C 是一组一元或者二元的约束，一个一元约束 C_i 是值域 D_i 的子集，包含变量 i 被允许的赋值，一个二元约束 C_{ij} 是包含于笛卡尔积 $D_i\times D_j$ 中的值对集合，表示变量 i 和 j 被允许同时赋予的值；

(4)元组 t 表示一组变量的赋值，实际上，t 是针对一组有序的变量集合 $X_t\subseteq X$ 的有序赋值，也就是说，t 中第 k 个元素代表给 X_t 中第 k 个变量的赋值。对 X_t 的一个子集 B 来说，t 在 B 上的映射记做 $t\downarrow_B$；

(5)一个约束包含的变量叫做约束的作用域。元组 t 满足一致性，就是说该元组满足所有作用域为 X_t 子集的约束。一个解就是一个满足一致性的完全赋值，即 $X_t=X$。

找一个 CSP 的解是 NP 难问题，可将其简化为实现弧一致性的过程。在搜索求解或预处理的过程中应用约束，去除变量论域中不会组成解的取值，这个过程通常称为约束传播或者一致性技术。这一技术中最有效的是弧一致性技术，将 CSP 用约束图的形式表示，变量 i 和 j 的约束就表示为一条弧(i,j)，称它是弧一致性的(Arc Consistency，AC)[3,14]，当且仅当变量 i 论域上能满足 i 上一元约束的每个取值 a，在变量 j 中都存在一个满足 j 上一元约束的取值 b，使得赋值(a,b)满足 i 和 j 上的二元约束 C_{ij}，(j,b)称为(i,a)在约束 C_{ij} 上的支持。若 CSP 约束图上的每条弧都是弧一致性的，则说这个 CSP 是弧一致性的。

对一个 CSP $P=(V,D,C)$，n 为变量集合 V 中变量的个

数，n 个变量的下标分别用 0,1,2,…,$n-1$ 表示，d 为 D 中变量论域 D_i 的最大值。则 P 中的每个变量 i 可用长度为 $s=\lceil \log_2 n \rceil$ 的二进制串 $X=(x_0,x_1,\cdots,x_{s-1})$ 表示，每个变量论域 D_i 的取值 a 可用长度为 $t=\lceil \log_2 d \rceil$ 的二进制串 $V=(v_0,v_1,\cdots,v_{t-1})$ 表示，其中 $x_i,v_j\in\{0,1\}$。进而赋值 (i,a) 可表示为 $(X,V)=(x_0,x_1,\cdots,x_{s-1},v_0,v_1,\cdots,v_{t-1})$，二元约束 C_{ij} 表示 $(X,Y,V,W)=(x_0,x_1,\cdots,x_{s-1},y_0,y_1,\cdots,y_{s-1},v_0,v_1,\cdots,v_{t-1},w_0,w_1,\cdots,w_{t-1})$。

举例来说，如图 1 所示的 CSP 约束图 P，图中数字编号 0,1,2 代表变量，字母 a,b,c 为各变量的取值，它们之间的连线代表满足变量之间约束关系的值对集合。我们设定在二元约束的 ADD 表示中，有连线的边权值为 1，代表取值对满足约束关系，省略的边权值为 0，代表不满足约束关系的取值对；一元约束的 ADD 表示中权值的含义与二元约束的 ADD 表示中的相同。在变量序 $x_0<x_1<y_0<y_1<v_0<v_1<w_0<w_1$ 下 P 的二元约束和一元约束表示如图 2(a)、(b)所示。二元 CSP 可由符号 ADD 表示为 $P=(BC(X,Y,V,W),UC(X,V))$，其中 $BC(X,Y,V,W)$ 和 $UC(X,V)$ 分别为二元约束和一元约束。

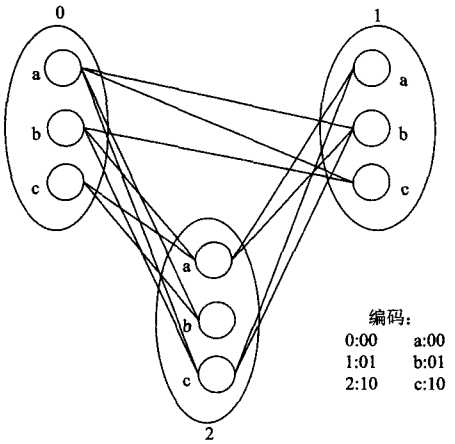
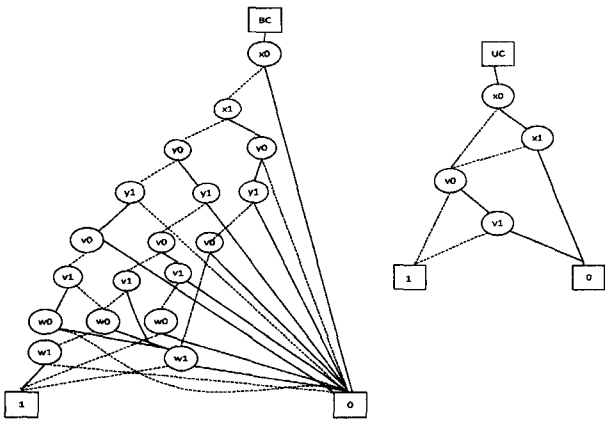


图 1 约束图 P



(a) 二元约束ADD (b) 一元约束ADD

图 2 约束图 P 的符号 ADD 表示

3 基于弧一致性符号 ADD 算法的 CSP 求解技术

基于前节给出的 CSP 的 ADD 描述，下面给出弧一致性符号 ADD 算法伪代码：

```
1. GetAC(BC(X,Y,V,W),UC(X,V),Q)
2. {
3.   Q={0,1,2,…,n-1};
4.   while(Q≠∅)
5.   {
6.     i=VariableSelect(Q);
7.     Q=Q-{i};
8.     tmpBC(X,Y,V,W)=i_Constraints(i,BC(X,Y,V,W));
9.     del(X,V)=∃y(\max_w tmpBC);
10.    i_Value=S(i)∩UC(X,Y,V,W);
11.    flag=del(X,V)∩i_Value;
12.    if(flag==zero)
13.    {
14.      约束满足问题无解;
15.      return;
16.    }
17.    else if (flag==i_Value) continue;
18.    UC(X,V)=UC(X,V)∩del(X,V)
19.    BC=BC-[(del(X,V)∩BC)∪(del(Y,W)∩BC)];
20.    Q=Q∪((∃x∃v∃w)tmpBC);
21.  }
22.}
```

对于以上弧一致性符号 ADD 算法，初始时等待队列 Q 为变量集 V，当等待队列不为空时，从 Q 中选取一个变量 i 进行处理，$S(X)$ 是变量 i 的 ADD 表示，首先取出与 i 有关的约束，利用算法第 8 行的 i_Constraints 操作，具体通过以下操作来完成：第 1 步取出约束集合 $U_{j\in X,j\neq i}C_{ij}$ 为 $p(X,Y,V,W)=S(X)\cap BC(X,Y,V,W)$；第 2 步取出约束集合 $U_{j\in X,j\neq i}C_{ji}$ 为 $q(X,Y,V,W)=S(Y)\cap BC(X,Y,V,W)$；第 3 步通过把 q 中变量集 X 替换为 Y，V 替换为 W 达到将约束 C_{ji} 也转化为 C_{ij} 形式的目的，$r(X,Y,V,W)=q(X,Y,V,W)[X\rightarrow Y,V\rightarrow W]$；第 4 步将 p 与 r 相或得到最终结果与 i 有关的约束集 $tmpBC$。

接下来对 $tmpBC$ 进行 $\backslash_w^{\max}$ 操作，此操作表示存在量化布尔变量 W，并从量化所得的函数中取最大值，通过这一操作就能取得 i 论域中每个取值在各个与 i 有关的约束上最大的权值，根据前节所定义的值对权值的表示含义，权值也即终结点为 1 的值对表示满足该二元约束，为 0 表示不满足，若取值 a 在某约束上所有取值对的最大权值为 0，则表明它在该约束上没有支持，可以删去。我们再对提取操作的结果进行存在布尔变量 Y 的操作得到 del，就能得到变量 i 中所有应删除的取值。根据我们对于终结点的指代含义的设计，删除操作也很简单，只需要将 del 与原 UC 相与即可。

但是从算法的第 11 行到第 17 行我们设定了一个变量 $flag$，它是通过将 del 与变量 i 的原论域相与得到的。这个变量可以指示 i 论域的变化情况，如果 $flag$ 为逻辑零，表明通过预处理得到的变量 i 的论域中的值都不满足弧一致性，都不可能参与组成解，那么我们可以直接得到这个 CSP 无解；如果 $flag$ 与变量 i 的原论域相同，也就是说这次对 i 的处理没有引发论域的变化，那么也就不需要对一元二元约束和

等待队列进行更新。设置这个函数的主要目的是提高算法的效率。

算法的最后，如果变量 i 的论域发生变化，那么要从一元约束 UC 中删除不满足一致性的取值，从二元约束 BC 中删除不满足一致性的取值有关的约束，同时还要把与变量 i 有约束关系的变量加至等待队列 Q 中，这与 AC-3 算法相同。

对于图 1 的约束问题，预处理缩减问题规模的过程如图 3 所示。

举例 1　图 3 是对图 1 中的 CSP 进行 AC 预处理的过程，图 3(a)为初始约束图，$Q=\{0,1,2\}$；图 3(b)表示处理与变量 0 有关的约束，$(0,c)$因在约束 C_{01} 上没有支持而被删除，$Q=\{1,2\}$；图 3(c)描述处理与 1 有关的约束，$(1,a)$和$(1,c)$分别因在约束 C_{10} 和 C_{12} 上没有支持而被删除，$Q=\{0,2\}$；图 3(d)为重新处理变量 0 有关的约束，$(0,b)$因在 C_{01} 上失去支持而被删除，$Q=\{1,2\}$；图 3(e)为重新处理变量 1 有关的约束，其论域没有变化，$Q=\{2\}$；图 3(f)为处理变量 2 有关的约束，$(2,a)$和$(2,b)$分别因在约束 C_{20} 和约束 C_{21} 上没有支持而被删除，$Q=\{0,1\}$。接下来算法会执行到 Q 为 $\emptyset$，但约束图不会再变化。

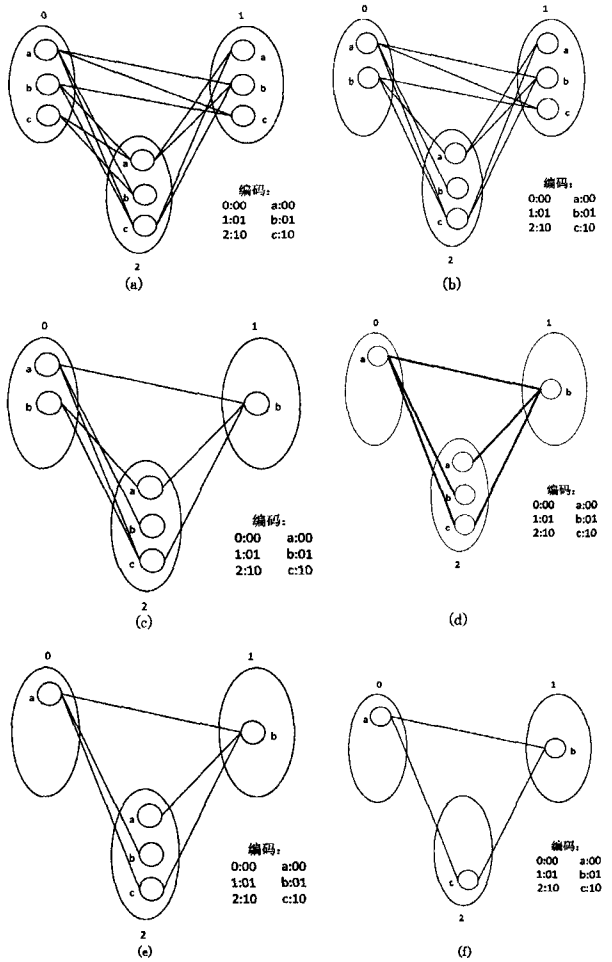


图 3　对约束图 P 进行弧一致性预处理的过程

对于一个已用 ADD 描述的约束满足问题 P_0，首先对 P_0 应用 GetAC 算法进行预处理，得到新的满足弧一致性的问题 P，P. BC 和 P. UC 分别表示二元约束和一元约束的 ADD。设未实例化的变量集合 F，初始时 F 为变量集 V。对 P 的递归求解过程如下：

```
1. int BT(CSP P,F)
2. {
3.   if(F==∅) return 1;//返回 1 代表求解成功
4.   i=VariableSelect(F);
5.   F=F-{i};
6.   while(D(i)! =∅)
7.   {
8.     a=ValueSelect(i);//对 i 实例化后将 i 的值域变为取值 a,得到新的子问题 P1
9.     D(i)=D(i)-a;
10.    P1. UC=(P. UC-S(i)∩P. UC)∪S(i,a);
11.    P1. BC=(P. BC--S(i)∩P. BC)∪(S(i,a)∩P. BC)
12.    P1. Q=P. Q;//如果子问题 P1 有解,继续递归求解,否则回溯
13.      if(GetAC(P1. BC,P1. BC,P1. Q)返回有解)
         return BT(P1,F);
14.      else return 0;
15.  }
16.  return 0;
17. }
```

嵌入弧一致性符号 ADD 算法的 BT 搜索算法流程如图 4 所示，将问题 P 的变量 i_0 初始化为 a，i_0 的值域固化为只有 a，对于其他和变量 i_0 有约束关系的变量，也只保留(i_0,a)参与的值对，这样就得到子问题 P_1，同样对 P_1 应用 GetAC 预处理得到 P_1'，发现无解就回溯，有解就继续递归求解。

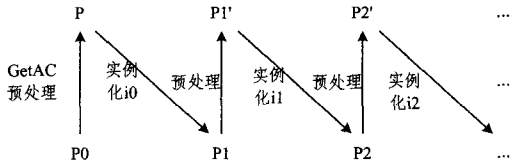


图 4　嵌入 GetAC 算法的 BT 求解过程

4　实验分析

对本文算法性能的检验采用了两类测试用例，分别是 CSP 模型生成的随机二元约束满足问题和一些标准的测试用例。对于前一类测试用例，实验使用本文算法和传统的 MAC3+BJ 以及 MAC2001+BJ 算法分别对问题求解并进行性能比较；对标准的测试用例，在使用本文算法求解得到求解时间后，与经典的在回溯中保持弧一致性技术(BT+MAC)和文献[17]中提出的用传统数据结构进行预处理，并在搜索中保持弧一致性的算法 BT+MPAC 和 BT+MPAC* 的求解时间进行比较。实验采用 C 语言，结合 Colorado 大学开发的 CUDD(CU decision diagram package release2. 3. 1. http://vls. iColorado. edu/fabio/CUDD/cuddIntro. html)软件包编写。测试环境为：硬件 P4 3GHz CPU，512MB RAM，软件 Windows XP Professional SP2/Visual C++ 6. 0。

(1)随机约束满足问题测试用例

二元随机约束满足问题有 4 个描述参数$\langle n,m,c,p\rangle$，其中 n 表示变量的个数，m 表示变量值域的大小，c 代表约束的

个数，$p2$ 为每个约束的松紧度，具体地说是存在冲突的值对个数与 $m\times m$ 的比值。在实验中选取参数为〈$n=40, m=5, c=55, p$〉的测试用例，对 $p=0.12$、0.2、0.24、0.32、0.36、0.42、0.48 的每个取值生成 5 组测试数据，分别用本文算法(GetAC+BT)和在搜索过程中保持 AC3(Maintain AC3, MAC3)弧一致性的回跳(Backjumping, BJ)算法 MAC3+BJ 以及保持 AC2001 弧一致性回跳算法 MAC2001+BJ(缩写意义类似 MAC3+BJ)对测试数据进行求解，求解时间为每组数据求解时间的平均值。3 种算法的求解时间对比折线图如图 5 所示。

从图 5 可以看出，当 p 小于 0.3 时，本文算法比两种传统的保持弧一致性的回跳搜索算法还稍慢一些，这是因为一方面对 CSP 进行 ADD 构造需要花费时间，另一方面在预处理过程中本文算法以变量为执行单位，即使在约束关系较松的情况下执行步数也是固定不变的；当 $p>0.3$ 时，本文算法的求解时间开始小于两种传统的保持弧一致性回跳搜索算法，且求解时间保持稳定，增加幅度较小，相比之下，两种传统算法的求解时间则随着约束强度的增大明显变长，这是因为本文算法对 CSP 中的约束是以集合方式进行的，约束强度 p 增大对算法的求解时间影响不大。

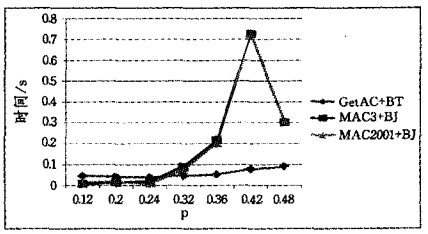


图 5　3 种算法在测试用例〈40,5,55,p〉上实验对比图

(2)标准库的测试用例

我们在标准库(http://cpai.ucc.ie/05/Benchmarks.html)的测试用例选取了两种规模的 frb 问题，文献[17]提出了两种用传统数据结构实现的、带预处理步骤并在搜索过程中保持 AC 和 AC* 的搜索算法 BT+MPAC 和 BT+MPAC*，用本文算法与该文献算法求解相同问题，得到的结果对比如表 1 所列。

表 1　标准库测试用例的实验对比

算法	Frb30-15 求解时间/s	Frb35-17 求解时间/s
BT+MAC	10.203	19.64
BT+MPAC	3.234	5.50
BT+MPAC*	3.766	6.64
GetAC+BT	0.4472	0.7658

从表 1 可以看出对于两种规模的 frb 问题，本文算法的求解效率既优于现有的用传统数据结构实现预处理并且在 BT 搜索中保持弧一致性的算法(BT+MPAC 和 BT+MPAC*)，也优于传统带弧一致性维护的回跳算法。

结束语　在 CSP 求解前应用弧一致性对问题进行预处理，理论上可以达到精简问题规模、提高搜索效率的目的；在搜索的过程中保持弧一致性也能有效缩减解的搜索空间。本文研究首先将弧一致性符号 ADD 算法应用于 CSP 预处理，接着利用该算法在搜索的过程中始终保持由未实例化变量所组成的子问题的弧一致性，形成约束满足问题基于 ADD 实现的弧一致性预处理与保持的回溯求解算法。实验结果表明，不管是对随机约束满足问题还是标准库测试用例的求解，本文算法相比传统的保持弧一致性的回溯算法，求解效率提高较多，且随着求解问题规模的增大求解时间增长趋势平缓，与其他带预处理的弧一致性保持回溯算法相比求解效率也有明显提高，这证明了将 ADD 与弧一致性技术相结合求解 CSP 有一定的优势。下一步工作将要用 ADD 实现软约束满足问题的弧一致性，验证这一优势在软约束满足问题范围是否继续被保持。

参 考 文 献

[1] Marriot K, Stuckey P. Programming with constraints[M]. Cambridge: MIT Press, 1998

[2] Apt K. Principle of Constraint Programming [M]. Cambridge: Cambridge Press, 2003

[3] Prosser P. Hybrid algorithms for the constraint satisfaction problems[J]. Computational Intelligence, 1993, 9(3): 268-299

[4] Ginsberg M L. Dynamic backtracking[J]. Artificial Intelligence, 1993, 1: 25-46

[5] Larrosa J, Schiex T. Solving Weighted CSP by Maintaining Arc Consistency[J]. Artificial Intelligence, 2004, 159(1/2): 1-26

[6] Mackworth A K. Consistency in networks of relations[J]. Artificial Intelligence, 1977, 8(1): 99-118

[7] Christian B, Charles R J. Refining the basic constraint propagation algorithm[C]// Proceedings of the 17th International Joint Conference on Artificial Intelligence. Seattle WA: Morgan Kaufmann, 2001: 309-315

[8] Mohr R, Henderson T C. Arc and path consistency revised[J]. Artificial Intelligence, 1986, 28(2): 225-233

[9] 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

[10] 古天龙，徐周波. 有序二叉决策图及应用[M]. 北京：科学出版社，2009

[11] Bryant R E. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams[J]. ACM Computing Surveys, 1992, 24(3): 293-318

[12] Bachar R I, Frohm E A, Gaona C M, et al. Algebraic Decision Diagrams and Their Applications[J]. Formal Methods in Systems Design, 1997, 10(2/3): 171-206

[13] Hachtel G D, Somenzi F. A Symbolic Algorithm for Maximum Flow in 0-1 Networks[J]. Formal Methods in System Design, 1997, 10(2/3): 207-219

[14] 徐周波，古天龙，赵岭忠. 网络最大流问题的一种新的符号 ADD 求解算法[J]. 通信学报，2005, 26(2): 1-8

[15] 徐周波，古天龙，常亮. 加权约束满足问题的符号 ADD 求解算法[J]. 模式识别与人工智能，2011, 24(1): 14-21

[16] Larrosa J. Node and Arc Consistency in Weighted CSP[C]// Proc. of the AAAI-02. Alberta, Canada, Aug. 2002: 48-53

[17] 孙吉贵，朱兴军，张永刚，等. 一种基于预处理的约束满足问题求解算法[J]. 计算机学报，2008, 31(6): 919-926