一种基于环切割的约束满足问题求解算法

李占山 李宏博 张永刚 王孜文

(吉林大学符号计算与知识工程教育部重点实验室 长春 130012)

(吉林大学计算机科学与技术学院 长春 130012)

摘 要 该文首先给出一种无环约束满足问题的无回溯搜索算法 Tree_Search,然后将环切割思想嵌入到目前最流行的 MAC3rm 算法中,给出一种新算法 CCS. CCS 将原回溯搜索过程分为两部分:第 1 部分通过回溯搜索求解环切割集中变量,将原问题化简成一个满足弧相容的无环问题;第 2 部分通过无回溯的 Tree_Search 算法求解化简后的无环问题,改进了 MAC3rm 算法. 证明了 MAC3rm 算法在环切割集上求得的局部解一定可以扩展为一个全局解,并且如果原问题无解,则 MAC3rm 算法在环切割集上找不到局部解. 实验结果显示,CCS 的效率在大多数情况下高于 MAC3rm. 在求解随机问题相变阶段的测试用例时,CCS 的效率最高可以达到 MAC3rm 的 140 倍. Benchmark 中几组问题的测试结果显示,CCS 在整体上效率高于 MAC,最高可以达到 MAC3rm 的 100 倍以上.

关键词 弧相容;无回溯搜索;环切割;MAC3rm

中图法分类号 TP18 **DOI 号**: 10.3724/SP.J.1016.2011.01528

An Approach of Solving Constraint Satisfaction Problem Based on Cycle-Cut

LI Zhan-Shan LI Hong-Bo ZHANG Yong-Gang WANG Zi-Wen

(*Key Laboratory of Symbolic Computation and Knowledge Engineering for Ministry of Education, Jilin University, Changchun* 130012)

(*College of Computer Science and Technology, Jilin University, Changchun* 130012)

Abstract Firstly a backtrack-free algorithm Tree_Search is given to solve the cycle-free CSPs. Secondly, by embedding the cycle-cut idea in the MAC3rm algorithm which is the most popular solver for binary CSPs in recent years, the new algorithm, called CCS, is presented. CCS separates the backtracking procedure into two steps, the former step searches the partial solution for the variables in the cycle-cut set and simplifies the problem into a cycle-free problem that is arc consistency, and the latter step solves the rest variables by Tree_Search algorithm. As a result, CCS has improved MAC3rm. It is proved that the partial solution which is found by MAC3rm can be extended to a global solution and MAC3rm can not find a partial solution on the cycle_cut set if the global solution does not exist. The experiments show that CCS is more efficient than MAC3rm. While solving the random CSPs which are at the phase transition, CCS can reach a maximum of 140 times more efficient than MAC3rm, and while solving some of the benchmarks, CCS can reach up to 100 times more efficient than MAC3rm.

Keywords arc consistency; backtrack free search; cycle cut; MAC3rm

---

收稿日期:2009-11-14;最终修改稿收到日期:2011-06-01. 本课题得到国家自然科学基金(60773097,60873148,60973089)和吉林省自然科学资金项目(20060532、20071106、20080107)资助. **李占山**,男,1966 年生,博士,教授,主要研究领域为基于模型的诊断、智能规划与决策、约束问题求解. **李宏博**,男,1985 年生,硕士,主要研究方向为约束问题求解. E-mail: lhb.jlu@163.com. **张永刚**,男,1975 年生,博士,副教授,主要研究方向为约束程序与约束推理. **王孜文**,1985 年生,硕士,主要研究方向为约束问题求解.

1 引　言

约束满足问题(Constraint Satisfaction Problem，CSP)[1]在人工智能领域有着广泛的应用，许多实际应用问题都可以用 CSP 进行建模，如调度问题、配置问题、图着色问题等等. 约束满足问题求解一直是约束程序研究领域的难点，寻找一个 CSP 问题的解是一个 NP 完全问题. 深度优先的回溯搜索算法(Backtracking，BT 算法)[2]是 CSP 求解的一个完备的核心算法. 为了提高求解效率，通常在求解前和求解过程中使用约束关系删除变量值域中一些不参与解的值，这个过程称为约束传播[3]或者相容性技术. 常见的完备求解算法都是在 BT 算法搜索过程中加入约束传播技术，如向前检查算法(Forward Checking，FC)[4]、维持弧相容算法(Maintaining Arc Consistency，MAC)[5]. 其中 MAC 算法是应用的最广的，并且在求解大规模难解问题时有着明显的优势. MAC 算法是在 BT 算法中嵌入弧相容算法(Arc Consistency，AC)[6]，通过弧相容算法进行约束传播，达到高效的剪枝目的. MAC 算法的求解效率主要受两方面因素的影响，一是在求解过程中使用的 AC 算法本身的效率，另外一个因素是 AC 算法使用的数据结构的复杂程度，因为在求解过程中需要维护一部分数据结构，数据结构较复杂时，维持这些数据结构要消耗更多的时间. AC3[6]是早在 1977 年被提出的，它是最简单、最基本的 AC 算法，也是在求解中应用最广泛的. 在 AC3 的基础上，一些改进算法被提出，如 AC4[7]、AC6[8]、AC7[9]、AC2001[10]等. 这些算法虽然有着较优的最坏时间复杂度，但由于其本身数据结构复杂，因此在求解中的应用效果并不是很理想. 2007 年 Lecoutre 等人[11]提出了 AC3rm，AC3rm 使用和 AC3.1 类似的数据结构，但是在求解过程中只维护和 AC3 相同的最简单的数据结构，是目前普遍认为在 MAC 算法中应用效果最好的 AC 算法. 除 AC 算法以外，其它的约束传播技术如 Singleton Arc Consistency[12-13]、k-consistency[3]等，都可以实现比 AC 算法更高效的剪枝，但它们都比 AC 算法有着更高的时间复杂度和更复杂的数据结构，因此在求解中很少被应用. 除此以外，我国学者孙吉贵等人[14]曾于 2008 年在 MAC 算法基础上提出了一种基于预处理技术的约束满足问题求解算法 BT+MPAC. 该方法虽然没有 AC 算法剪枝效果好，但是本身的时间复杂度低于 AC，并且求解过程中维持的数据结构也很简单，在一些测试用例中，BT+MPAC 比 MAC 算法效率更高，其主要缺点在于需要在求解前执行预处理.

Freuder[15]于 1982 年提出无回溯搜索的充分条件以后，一些关于无回溯搜索的思想相继出现. 如文献[16-17]，但这些无回溯搜索算法主要是针对特定类型的问题，如树形结构、配置问题等. 普通约束满足问题的无回溯搜索需要在求解过程中维持高阶相容[15]，但是维持高阶相容本身要消耗大量时间，因此很少被采用. 除了无回溯思想以外，基于将问题分解进而提高求解效率的方法也相继被提出[18-19]，其中 Dechter[20]于 1990 年提出了环切割集的概念，但是 Dechter 关于环切割的方法，在为环切割集中变量找到局部解后，不能保证这个局部解一定可以扩展成一个全局解.

本文首先给出一种 Tree_Search 搜索算法，它应用于约束图是树形结构的并且满足弧相容的二元问题上可以实现无回溯搜索，然后将环切割集的思想与目前最流行的求解算法 MAC3rm[11]相结合，给出一种新的搜索算法(Cycle_Cut_Search，简称 CCS 算法). 我们还提出了环切割化简定理，证明了通过 MAC3rm 算法寻找到的环切割集上的局部解，一定可以扩展成一个全局解，并且如果原问题无解，则 MAC3rm 算法无法找到环切割集上的局部解. 基于环切割化简定理，CCS 算法将原 MAC3rm 算法的全局回溯搜索过程分为两部分：第 1 部分在环切割集上用 MAC3rm 算法回溯搜索局部解；第 2 部分通过 Tree_Search 无回溯搜索算法将局部解扩展为全局解. 通过环切割化简的方式改进了 MAC3rm 算法. 我们的实验结果显示，CCS 的效率在大多数情况下高于 MAC3rm. 在求解随机问题相变阶段的测试用例时，CCS 的效率最高可以达到 MAC3rm 的 140 倍. Benchmark 中几组问题的测试结果显示，CCS 在整体上效率高于 MAC，最高可以达到 MAC3rm 的 100 倍以上.

2 相关知识介绍

定义 1.　约束满足问题[1]. 一个约束满足问题 P 由 3 个部分组成：$P=\langle X, D, C\rangle$，其中 $X=\{x_1, x_2, \cdots, x_n\}$ 是一个有限的变量集合；$D=\{D_1, D_2, \cdots, D_n\}$，其中 $D_i \in D$ 对应 $x_i \in X$，是变量 x_i 的有限值域，$C=\{c_1, c_2, \cdots, c_k\}$ 是一个有限约束集合，其中任意 $c_j \in C$ 表示变量取值之间的制约关系. 如果 C 中所有约束包含的变量个数都小于等于 2，则称 P 为二元约束满足问题. 一个约束满足问题的解

是为每个变量在其值域中选择一个值构成集合 S，使得 S 中的所有变量取值满足 C 中所有约束. 本文中用 P 表示一个约束满足问题，S 表示 P 的一个解，$P.X$ 表示 P 的变量集，$P.C$ 表示 P 的约束集，(x,a)表示 x 的取值为 a，n 表示 P 中包含的变量个数，d 表示 P 中最大的变量值域的大小.

定义 2. 局部解. 给定一个约束满足问题 $P=\langle X,D,C\rangle$，X'是 X 的一个子集，如果对于 X'中所有变量，都能找到对应的赋值，使这些赋值构成集合 S'，使得 S'满足 X'中所有变量相关的所有约束，则称 S'为 P 在 X'上的局部解.

在二元约束满足问题中，如果存在一条约束 c_{xy}，则(x,c_{xy})叫做一条弧，表示的是约束 c_{xy} 对 x 取值的制约. 在二元 CSP 中，一条约束可以表示两条弧.

定义 3. 弧相容(Arc Consistency, AC)[6]. 给定一个二元约束满足问题 P，对于 P 中的某一条弧(x,c_{xy})，如果 x 值域中能满足 x 上一元约束的每一个值 a，都能在 y 的值域中找到一个值 b 满足 y 上一元约束，并且(a,b)满足 c_{xy}，那么称它是弧相容的. 一个 CSP 是弧相容的，当且仅当它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，如果检查过程中将某一变量的值域删空，则此问题 P 无解，记为 $AC(P)=\perp$.

定义 4. Singleton 弧相容(Singleton Arc Consistency, SAC)[12]. 给定一个约束满足问题 P，其中一个变量 x_i，a 为 D_i中一个值，如果用单独的 $\{a\}$替换原问题 P 中 x_i的值域 D_i后得到的 $P\mid x_i=a$ 是弧相容的，即 $AC(P\mid x_i=a)\neq\perp$，则称(x_i,a)是 singleton 弧相容的. 如果 $\forall x_j\in X$，$\forall b\in D_j$，都有(x_j,b)是 singleton 弧相容的，则称 P 是 singleton 弧相容的.

本文主要使用单个值的 singleton 弧相容. 目前较流行的实现全局 singleton 弧相容的算法是 SAC-SDS[13].

维持弧相容算法(Maintaining Arc Consistency, MAC)[5]是一种高效的求解 CSP 问题的回溯搜索算法. MAC 算法是在 BT 算法框架下嵌入 AC 算法，在搜索过程中维持弧相容的状态，每次变量赋值后利用 AC 算法进行约束传播，如果得到一个弧相容的状态，则赋值成功，否则赋值失败，选择下一个赋值或者发生回溯. 目前最流行的 MAC 算法是 Lecoutre 于 2007 年提出的 MAC3rm 算法，即在 MAC 算法中通过 AC3rm 执行弧相容，MAC3rm 在求解效率上比其它 MAC 算法有着明显的优势. 由于篇幅有限，这里不介绍 MAC3rm 的算法描述，关于 MAC3rm 的一些细节可以参考文献[11].

给定一个二元 CSP，如果将其中的变量当成节点，其中的约束当做边，则可以得到一个无向图 G，我们称 G 为这个 CSP 的约束图. 我们称约束图中没有回路的二元 CSP 为无环约束满足问题或树形约束满足问题，约束图中有回路的称为有环约束满足问题.

定义 5. 环切割集(cycle-cut set)[20]. 一个约束图 G 的环切割集 M 是该约束图中变量集的一个子集，M 满足如下条件：将 M 中的变量从约束图 G 中移除后，G 中不存在环.

后面程序伪代码中的 $X(c)$返回约束 c 相关的变量集合. 本文所讨论的都是二元约束满足问题，因此 $X(c)$得到的只是一个约束相关的两个变量.

3 Tree_Search 搜索算法

Freuder[15]在 1982 年提出无回溯搜索的充分条件，指出满足弧相容的无环 CSP 在宽度为 1 的搜索序列下可以实现无回溯搜索，关于搜索序列宽度的更多细节，可以参考文献[15]. 我们给出无回溯搜索充分条件在树形问题上的具体描述.

定理 1. 给定一个无环约束满足问题 P，如果 P 满足弧相容，则 P 一定有解，并且可以用一种无回溯的算法求解.

树的深度优先遍历序列和宽度优先遍历序列都可以满足宽度为 1，我们给出一种基于深度优先顺序的无回溯搜索算法.

算法 1. Tree_Search(TS)无回溯求解算法，P 为一个满足弧相容的无环约束满足问题.

```
Tree_Search(P)
  Begin
1.  Q=P.X;
2.  while Q is not empty do
3.    select a x_i from Q;
4.    Depth_First_Search(x_i);
  End
Depth_First_Search(x_i)//深度优先遍历赋值
  Begin
1.  select an available value v for x_i;
2.  x_i=v;
3.  remove x_i from Q;
4.  while there is another constraint c involve x_i and
          c.dealed=false do
5.    x_j=X(c)-x_i;
6.    c.dealed=true;
7.    Depth_First_Search(x_j);
  End
```

定理 2.　算法 Tree_Search 的最坏时间复杂度为 $O(nd)$.

证明.　在深度优先搜索过程中，搜索到任意变量 x_i 时，若 x_i 为根节点，可直接在 D_i 中找到一个未删除的值为 x_i 赋值. 若 x_i 不是根节点，则只有它的父亲节点 x_j 已经被赋值(x_j,b)，因此只需在 D_i 中找到一个值 a 支持(x_j,b)即可，由于 P 是弧相容的，所以一定能找到一个(x_i,a)使得(a,b)满足 c_{xy}. 为每个变量寻找当前可用的赋值，最多寻找 d 个值，问题中一共有 n 个节点，所以算法 Tree_Search 的最坏时间复杂度为 $O(nd)$.　　证毕.

4　环切割化简

由于约束满足问题的众多求解算法都是建立在回溯搜索的基础上，而 Tree_Search 算法在处理无环约束满足问题时效率通常高于回溯搜索算法，所以如果能够将普通约束满足问题转化成一个满足弧相容的无环约束满足问题后应用 Tree_Search 算法进行求解，那么就会提高问题求解效率.

我们的环切割思想借鉴了单个值 singleton 弧相容的定义，下面举例说明：一个约束满足问题 $P=\langle X,D,C\rangle$，$X=\{x_1,x_2,x_3\}$，$D_i=\{1,2,3,4\}(i=1,2,3)$，$C=\{C_{12}:x_1+x_2>2,C_{23}:x_2<x_3,C_{31}:x_3\neq x_1\}$，$P$ 的约束图中存在环. 如果 $AC(P|x_1=1)\neq\perp$ 后得到新问题 P_1. 则 $P_1.D_1=\{1\}$，$P_1.D_2=\{2,3\}$，$P_1.D_3=\{3,4\}$，此时 $P_1.D_2$ 和 $P_1.D_3$ 中的所有值都支持$(x_1,1)$，以后的求解过程中，无论 x_2 赋值为 $P_1.D_2$ 中的哪个值，约束 C_{12} 都一定不会被违反，同理 x_3 的赋值也不会违反 C_{31}，因此以后的求解中可以不再考虑 x_1，C_{12}，C_{31}，这样就可以暂时将 x_1，C_{12}，C_{31} 从 P_1 中移除后得到 P_2，此时 P_2 的约束图中只含有 x_2，x_3，C_{23}，然后可以对 P_2 继续求解.

定义 6.　给定两个 CSP，P_1 和 P_2，如果 $P_1.X$ 和 $P_2.X$ 相同，并且 P_1 和 P_2 的解相同，则称 P_1 和 P_2 是等价的.

定理 3.　对于一个约束满足问题 P，如果 $AC(P|x_i=a)\neq\perp$ 后得到 P_1，则在 P_1 中暂时将 x_i 及所有和 x_i 相关的约束移除后得到化简后的问题 P_2 和 P_1 是等价的.

证明.　$AC(P|x_i=a)\neq\perp$ 得到 P_1 后，对于 P_1 中任意一个和 x_i 之间有约束关系的变量 x_j，当前 $P_1.D_j$ 中的所有值都一定支持(x_i,a)，并且(x_i,a)也一定支持 $P_1.D_j$ 中的所有值. 因此求解 P_1 时可以不再考虑这些 C_{ij}，而 $P_1.D_i$ 中只包含(x_i,a)一个值，因此在 P_1 和 P_2 的所有解中，x_i 的取值一定是 a. 因此将它们移除后得到的新问题 P_2 和 P_1 是等价的.

证毕.

值得注意的是定理 3 中提到的等价是指 P_1 和 P_2 等价，并不是和原问题 P 等价. 以下所提到的“化简”都是根据定理 3 的思想，通过为变量赋值后执行弧相容算法来判断化简是否成功. 当 $AC(P|x_i=a)\neq\perp$ 时，这一步化简成功，否则化简失败.

根据定理 3，我们可以在求解过程中逐步将原问题化简，但在化简过程中难免遇到 $AC(P|x_i=a)=\perp$ 的情况，针对这一问题，我们可以将化简过程嵌入到 BT 算法中，每次化简失败时启动回溯机制. 由于我们的化简是以弧相容执行结果判断是否化简成功，这和 MAC3rm 算法赋值后约束传播过程接近，因此我们可以将环切割化简的方法结合 MAC3rm 算法.

定理 4.　环切割化简定理. 给定一个有环约束满足问题 P，在 MAC3rm 求解过程中优先为环切割集中变量赋值，如果 P 有解，那么一定能将原问题化简成一个无环的问题 P_1，并且由环切割集上的局部解一定可以扩展出一个解；如果 P 无解，那么 MAC3rm 算法无法为环切割集中变量找到局部解.

证明.　如果 P 有一个解 S，由于 MAC3rm 算法是完备的，则 MAC3rm 算法至少能找到 S 在环切割集上的局部解，由于 MAC3rm 算法在执行过程中维持弧相容的状态，在每次赋值后要执行弧相容算法，MAC3rm 算法每次为变量赋值相当于一次化简. 因此在找到环切割集上的局部解后，问题仍是一个弧相容的状态，并且环切割集中的变量都已经化简成功，将环切割集中变量和相关约束全部删除后，P 化简成无环问题 P_1. 并且 P_1 满足定理 1 中条件，因此 P_1 一定有解. 由于在 P_1 中，环切割集中变量都已经找到局部解，因此 P_1 的解就是原问题 P 的解. 如果 P 无解，假设化简环切割集中所有变量成功后得到新问题 P_2. 由定理 3 可知，P_2 的约束图中无环，并且由于化简过程是维持弧相容的，则 P_2 满足弧相容，由定理 1 可知，P_2 一定有解，这与前提 P 无解矛盾，因此如果 P 无解，则化简一定失败.　　证毕.

计算最小环切割集是一个 NP 问题[18]，通常只能通过启发式算法计算一个近似最小的环切割集. 下面给出一种计算环切割集的算法 Compute_CutSet. 其中 *get_a_cycle*() 是基于深度优先遍历搜索 P 的约束图，如果找到一个环，记录这个环中所有变量并

返回 true,否则返回 false.

```
Compute_CutSet(in P:csp)
  Begin
1.  cut_set={};
2.  while get_a_cycle(P) do
3.    select a x_i variable from the cycle;
4.    cut_set=cut_set∪{x_i};
5.    remove x_i and all the constraints involve x_i from P;
6.  return cut_set;
  End
```

定理 5. 算法 Compute_CutSet 最坏时间复杂度为 $O(n^2)$.

证明. 每次 *get_a_cycle* 基于深度优先遍历约束图,最坏要遍历 k 个节点才能找到环(k 为当前约束图中的变量个数),因此寻找一个环需要最坏时间复杂度为 k,最坏情况是问题 P 的约束图是一个完全图的情况,此时任意 3 个变量都构成环,因此要将 $n-2$ 个变量加入环切割集,因此需要寻找 $n-2$ 个环,每次找到一个环后将一个变量从约束图中删除,k 递减 1,所以最坏情况下要访问 $\sum_{k=3}^{n} k=n\times(n-1)/2-3$ 个变量,所以最坏时间复杂度为 $O(n^2)$. 证毕.

下面给出基于环切割化简的约束满足问题求解算法 Cycle_Cut_Search(CCS).

```
Cycle_Cut_Search(in P:csp)
  Begin
1.  if Cycle_Cut_MAC3rm(P) then
2.    Tree_Search(P);
    else
3.    return nosloution;
  End

Cycle_Cut_MAC3rm(in P:csp):Boolean
  Begin
1.  cut_set=Compute_CutSet(P);
2.  if MAC3rm(P, cut_set) then //MAC3rm only assign
                               //values for the variables
                               //in cut_set
3.    for each x_i in cut_set do
4.      Remove x_i and all the constraints involve x_i from P;
5.    return true;
    else
6.      return false;
  End
```

定理 6. 当问题 P 的约束图是完全图时,算法 Cycle_Cut_Search 退化为 MAC3rm 算法.

证明. 当问题 P 的约束图是完全图时,任意 3 个变量都构成一个环,每次化简一个变量 x_i 后只能将 x_i 所在的所有环删除,其他未化简变量仍在环中,因此需要化简($n-2$)个变量才可以将问题化简为一个无环约束满足问题. 此时有($n-2$)个变量要通过 MAC3rm 算法求解,因此算法 Cycle_Cut_Search 退化为 MAC3rm 算法. 证毕.

Dechter[20] 的方法中只是通过为变量赋值进行切割,但是并没有用弧相容算法进行约束传播,因而不能保证环切割集上找到的局部解一定能扩展成全局解. 除此以外,Sabin 等人[21] 在 1997 年曾提出一种和 CCS 接近的算法 MACE,两者主要区别在于,MACE 是建立在 MAC-7ps 基础上的,而 CCS 是建立在 MAC3rm 基础上的,并且 MACE 在求解过程中不断地将不在环中的变量加入到另一个变量集合中,在求解过程中变量所属的集合是动态变化的. 而 CCS 在 MAC3rm 开始求解前已经将变量所属集合确定,在求解过程中不再改变. MACE 通过约束检查次数来衡量算法效率,虽然约束检查次数是一个不随编程环境改变的硬性指标,但是文献[9,22]都指出了减少约束检查次数并不一定提高求解效率.

5 实验结果

MAC3rm 算法是目前最流行的二元约束满足问题求解算法,我们将 CCS 算法和 MAC3rm 进行比较. 我们的一些实验细节如下:二元约束通过二维数组表示,这样的约束检查代价最小. CCS 算法计算环切割集时选择当前环中连接约束最多的变量加入环切割集. 每次测试以一小时为界限,超时则停止测试,记为—. 测试环境为:Intel Core(TM)2 Duo CPU T6570, 双核 2.10 GHz,3 GB DDR800 内存,Windows XP SP3,JDK1.5.0.

5.1 随机问题测试

我们选择二元随机约束满足问题经典模型 Model B[24] 对算法 Cycle_Cut_Search 进行测试. 使用 Dom/Ddeg[23] 作为变量赋值顺序的启发式策略. Model B 问题模型由 4 个参数控制,$\langle n,m,p_1,p_2\rangle$ 其中 n 是问题中变量个数,m 是每个变量的值域大小,其中每个变量的值域大小都是相同的,p_1 表示约束图中约束的密度,即在约束图 G 中均匀选择 $p_1\times n\times(n-1)/2$ 条约束,p_2 表示约束的松紧度,是用来描述约束中存在冲突值对多少的参数,对于每条约束均匀地选 $p_2\times m\times m$ 个值对作为存在冲突的值对. 我们选取 $n=50$,$m=30$,$d=0.1$ 的问题进行测试,p_2 在 0.1～0.9,每组连续测试 50 次求平均值,

其中 p_2 在 0.1～0.69 和 p_2 在 0.78～0.9 的随机问题，都是容易解的问题，耗时均在几十毫秒左右，并且两者差异不大，这里并没有给出这两个区间的测试结果，只给出了处于相变阶段[24]的难解问题即 p_2 在 0.69～0.78 区间内两者耗时比较，统计后结果如下.

图 1 中 x 轴表示 p_2 在 0.69～0.78，每个数据点的 p_2 递增 0.002. 由图 1 中数据可知，在求解随机问题 $n=50, m=30, d=0.1$ 时，CCS 的效率明显高于 MAC3rm，我们的测试数据显示，CCS 的效率最多可以达到 MAC3rm 的 140 倍.

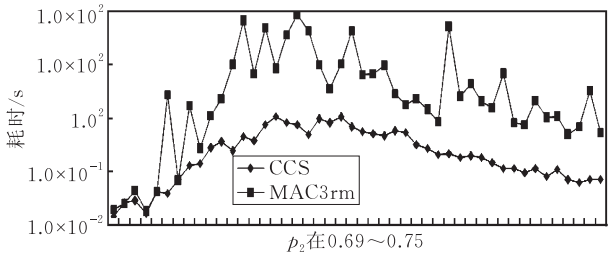

图 1　CCS 与 MAC3rm 的耗时比较

5.2　benchmark 测试

Benchmark 测试用例是用来测试 CSP 求解算法各方面性能的一个标准，本文采用的 benchmark 用例全部来自 2005 年国际约束程序竞赛. 我们选择其中的几组问题对 CCS 进行测试，分别是 rlfap、bqwh15_106、frb35-17、frb40-19、Domino、composed-25-10-20. 测试结果见表 1～5.

表 1　rlfap 测试结果

	子问题	时间/s	
		CCS	MAC3rm
rlfap	Graph12_w1	0.664	1.825
	Graph13_w1	0.066	0.245
	Graph12_w0	0.003	0.170
	Graph13_w0	0.004	0.319
	scen7_w1_f4	0.118	0.210

表 2　bqwh15_106 测试结果

	子问题	时间/s	
		CCS	MAC3rm
bqwh15_106	14	4.729	9.713
	24	1.331	3.349
	27	1.049	2.113
	32	2.092	2.384
	38	1.149	1.571
	46	2.066	1.667
	49	2.511	1.247
	71	1.079	1.412
	86	5.665	8.992
	91	1.570	3.353
	96	1.467	3.568

表 1 中给出了 rlfap 问题中约束较稀疏的几个子问题的测试结果. bqwh15_106 这组问题中共 100 个子问题，我们测试了全部 100 个子问题，将其中两者耗时都在 1 s 以上的子问题测试结果显示在表 2 中. 表 3 中列出了这两组 frb 问题的全部 10 个子问题测试结果，frb 问题的约束相对比较稠密，因此环切割化简的优势并不明显. Domino 问题的特点是约束稀疏且每条约束都非常紧，表 4 中列出了 Domino 问题的测试结果. 表 5 中列出了 composed25-10-20 的测试结果，从表 5 中数据可知，部分子问题中，MAC3rm 超时，但是 CCS 可以在有效时间内求解.

表 3　frb35-17、frb40-19 测试结果

	子问题	时间/s			子问题	时间/s	
		CCS	MAC3rm			CCS	MAC3rm
frb35-17	1	6.587	7.678	frb40-19	1	34.209	37.158
	2	12.382	12.203		2	12.286	15.210
	3	2.064	3.110		3	19.576	13.690
	4	4.755	4.476		4	21.628	15.077
	5	16.741	20.523		5	44.256	30.356

表 4　Domino 测试结果

	子问题	时间/ms	
		CCS	MAC3rm
Domino	100-100	12.9	16.6
	100-300	107.3	138.3
	300-100	39.1	114.7
	300-300	331.7	449.1
	500-100	67.2	214.1
	500-300	276.3	565.6
	1000-10	2.9	247.4

表 5　composed-25-10-20 测试结果

	子问题	时间/s	
		CCS	MAC3rm
composed-25-10-20	0	—	—
	1	0.008	—
	2	441.481	714.471
	3	1892.927	—
	4	0.002	0.003
	5	389.908	299.292
	6	—	—
	7	0.002	0.003
	8	0.003	0.003
	9	14.568	0.090

5.3　实验结果分析

由于变量赋值顺序对算法的效率有影响，因此我们的测试结果在个别情况下 CCS 效率低于 MAC3rm，这可能是由于 Dom/Ddeg 启发式策略在全局变量中寻找到一个比优先赋值环切割集更高效的变量赋值顺序. 我们的实验结果显示，CCS 在绝大多数情况下效率高于 MAC3rm，在随机问题相变阶段，最高可以达到 MAC3rm 效率的 140 倍，在我

们选择的 benchmark 测试用例中，CCS 的效率可以达到 MAC3rm 的 100 倍以上.

6 结 论

本文首先给出一种无环约束满足问题的无回溯搜索算法 Tree_Search，然后将环切割集的思想与目前最流行的求解算法 MAC3rm 相结合，给出一种新算法 Cycle_Cut_Search. CCS 将问题中变量集分为两部分：环切割集中的变量和环切割集以外的变量，通过 MAC3rm 算法为环切割集中变量赋值. 给出了环切割化简定理和证明，根据环切割化简定理，CCS 算法通过 MAC3rm 算法在环切割集上找到局部解后，可将原本有环约束满足问题化简成满足弧相容的无环约束满足问题，然后用 Tree_Search 算法执行无回溯求解，改进了 MAC3rm 算法的求解效率. 我们实验结果显示，CCS 的效率在大多数情况下高于 MAC3rm. 在求解随机问题相变阶段的测试用例时，CCS 的效率最高可以达到 MAC3rm 的 140 倍. Benchmark 中几组问题的测试结果显示，CCS 在整体上效率高于 MAC，最高可以达到 MAC3rm 的 100 倍以上.

参 考 文 献

[1] Freuder E C, Mackworth A K. Constraint satisfaction: An emerging paradigm//Rossi F, van Beek P, Walsh T. Handbook of Constraint Programming. Amsterdam: Elservier, 2006: 13-23

[2] van Beek P. Backtracking search algorithms//Rossi F, van Beek P, Walsh T. Handbook of Constraint Programming. Amsterdam: Elservier, 2006: 86-118

[3] Bessiere C. Constraint propagation//Rossi F, van Beek P, Walsh T. Handbook of Constraint Programming. Amsterdam: Elservier, 2006: 29-83

[4] Haralick R M, Elliott G L. Increasing tree search efficiency for constraint satisfaction problems. Artificial Intelligence, 1980, 14(3): 263-313

[5] Sabin D, Freuder E C. Contradicting conventional wisdom in constraint satisfaction//Cohn A G. Proceedings of the 11th European Conference on Artificial Intelligence. Amsterdam: John Wiley and Sons, 1994: 125-129

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

[7] Mohr R, Henderson T C. Arc and path consistency revised. Artificial Intelligence, 1986, 28(1): 225-233

[8] Bessière C. Arc-consistency and arc-consistency again. Artificial Intelligence, 1994, 65(1): 179-190

[9] Bessière C, Freuder E C, Régin J C. Using constraint metaknowledge to reduce arc consistency computation. Artificial Intelligence, 1999, 107(1): 125-148

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

[11] Lecoutre C, Hemery F. A study of residual supports in arc consistency//Proceedings of the IJCAI'07. Hyderabad, India, 2007: 125-130

[12] Debruyne R, Bessiere C. Some practicable filtering techniques for the constraint satisfaction problem//Proceedings of the IJCAI'97. Nagoya, Japan, 1997: 412-417

[13] Bessière C, Cardon S, Debruyne R, Lecoutre C. Efficient algorithms for singleton arc consistency. Constraints, 2011, 16(1): 25-53

[14] 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(in Chinese)
(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)

[15] Freuder E C. A sufficient condition for backtrack-free search. Journal of ACM, 1982, 29(1): 24-32

[16] Dechter R, Pearl J. Network-based heuristics for constraint-satisfaction problems. Artificial Intelligence, 1987, 34(1): 1-38

[17] Subbarayan S, Jensen R M, Hadzic T et al. Comparing two implementations of a complete and backtrack-free interactive configurator//Proceedings of the CP'04 Workshop on CSP Techniques with Immediate Application. Toronto, Canada, 2004: 97-111

[18] Dechter R. Constraint networks//Proceedings of the Encyclopedia of Artificial Intelligence. 2nd Edition. Wiley, New York, 1992: 276-285

[19] Rina Dechter. Tractable structures for constraint satisfaction problems. Chapter in the "Handbook of Constraint Programming". Rossi F, Walsh T, van Beek P eds. Elsevier, 2006: 209-245

[20] Dechter R. Enhancement schemes for constraint processing: Backjumping, learning and cutset decomposition. Artificial Intelligence, 1990, 41(3): 273-312

[21] Sabin D, Freuder E C. Understanding and improving the MAC algorithm//Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming. Linz, Austria, 1997: 167-181

[22] van Dongen M R C. Saving support checks does not always save time. Artificial Intelligence Review, 2004, 21(3-4): 317-334

[23] Bessière C, Régin J-C. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems//Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming. Cambridge, Mass, 1996: 61-75

[24] Smith B M, Dyer M E. Locating the phase transition in binary constraint satisfaction problems. Artificial Intelligence, 1996, 81(1-2): 155-181

LI Zhan-Shan, born in 1966, Ph.D., professor. His research interests include model-based diagnosis, planning, decision support systems and constraint programming.

LI Hong-Bo, born in 1985, master. His main research interest is constraint programming.

ZHANG Yong-Gang, born in 1975, Ph. D., associate professor. His research interests include constraint programming and constraint-based reasoning.

WANG Zi-Wen, born in 1985, master. His research interest is constraint programming.

Background

Constraint satisfaction problem plays an important role in artificial intelligence area. It is one of the foundations of AI. To solve a CSP is NP-hard. Maintaining arc consistency during backtracking search is the algorithm that embeds arc consistency technique in backtracking. There are some MAC algorithms, the differences between them are the arc consistency algorithms used in them. There are two factors impacting the efficiency of MAC algorithms, on the one hand, the efficiency of the arc consistency algorithm used in it, on the other hand, the data structures that must be maintained during search. If the data structures used in the corresponding arc consistency algorithm are simple, it would cost less time to maintain them during search. AC3 is widely used in constraint solvers in the last 30 years although it is not optimal, for it is simple. AC3rm, the optimal version of AC3, proposed in 2007, is widely accepted as the most successful arc consistency used in MAC. AC3rm improves AC3 and the MAC3rm algorithm maintains the same data structures as MAC3.

This work improves MAC3rm by combine the cycle cut idea with MAC3rm. The cycle cut idea was proposed by Dechter in 1990, but Dechter did not use it in MAC. Her work did not guarantee the partial solution in cut set can be extended to a global solution. We use it in MAC3rm, and proved that the partial solution which is found by MAC3rm can be extended to a global solution and MAC3rm can not find a partial solution on the cycle_cut set if the global solution does not exist. The experimental results show that CCS is more efficient than MAC3rm. While solving the random CSPs which are at the phase transition, CCS can reach a maximum of 140 times more efficient than MAC3rm, and while solving some of the benchmarks, CCS can reach up to 100 times more efficient than MAC3rm.