第 17 卷第 4 期
2011 年 4 月

计算机集成制造系统
Computer Integrated Manufacturing Systems

Vol. 17 No. 4
Apr. 2011

文章编号:1006 - 5911(2011)04 - 0832 - 06

基于动态值启发式的约束满足求解算法

王孜文[1,2],李占山[1,2+],艾 阳[1,2],李宏博[1,2]
(1. 吉林大学 符号计算与知识工程教育部重点实验室,吉林 长春 130012;
2. 吉林大学 计算机科学与技术学院,吉林 长春 130012)

摘 要:为提高约束满足问题的求解效率,提出了一种基于动态值启发式的约束满足问题求解算法。该算法在求解过程中吸收了以往启发式算法的优点,充分利用了预处理和弧相容检查阶段的信息。不但加入了变量启发式,而且在实例化变量时,对所有值的优先级进行动态的改变,从而实现了动态值启发式。比较了静态值启发式和动态值启发式的效率,分析了该算法的优缺点。通过随机问题标准库用例测试表明,该算法比经典主流算法具有更好的效率优势。

关键词:动态值启发式;值排序;约束满足问题;弧相容技术;启发式算法
中图分类号:TP18 **文献标志码**:A

Algorithm for solving constraint satisfaction problems based on dynamic value ordering heuristic

WANG Zi-wen[1,2], *LI Zhan-shan*[1,2+], *AI Yang*[1,2], *LI Hong-bo*[1,2]
(1. Ministry of Education Key Laboratory of Symbolic Computation & Knowledge Engineering, Jilin University, Changchun 130012, China;
2. College of Computer Science & Technology, Jilin University, Changchun 130012, China)

Abstract: To improve the efficiency of solving constraint satisfaction problems, an algorithm for constraint satisfaction problems based on dynamic value ordering heuristics named Backtrack Dynamic Value ordering Heuristic algorithm (BT-DVH) was proposed. In the solving process, this algorithm absorbed the advantages of the previous heuristics, and made full use of the information generated in preprocess and arc consistency checking. To achieve dynamic value ordering heuristic, variable ordering heuristic was used, and the priorities of the values were changed dynamically to instantiate the variables. The efficiencies of the static and dynamic value ordering heuristics were compared, the advantages and shortcomings of the BT-DVH were analyzed. Tested by random problems and benchmark, the result showed this algorithm had a higher efficiency than classic mainstream algorithm.

Key words: dynamic value ordering heuristic; value ordering; constraint satisfaction problem; arc consistency; heuristic algorithm

0 引言

约束满足问题(Constraint Satisfaction Problem, CSP)是人工智能的一个重要研究方向,其研究结果在符号推理、系统诊断、真值维护系统、车间作业调度、资源分配和产品配置等问题中有广泛的应用。由于 CSP 一般都是 NP-hard 问题,为了更好地对其进行求解,1977 年 Mackworth 将弧相容概念引入到 CSP 中,提出了 AC-1 算法,继而又对其进行改进,提出了 AC-2 算法和 AC-3 算法[1],其中 AC-3 算法直到现在还被广泛使用。1986 年 Mohr 利用支持的概念对 AC-3 算法进行了改进,提出了 AC-4 算法[2]。

收稿日期:2010-03-05;**修订日期**:2010-07-07。Received 05 Mar. 2010; accepted 07 July 2010.
基金项目:国家自然科学基金资助项目(60773097,60873148,60973089);吉林省自然科学基金资助项目(20060532,20071106,20080107)。
Foundation items: Project supported by the National Natural Science Foundation, China (No. 60773097, 60873148, 60973089), and the Natural Science Foundation of Jilin Province, China (No. 20060532, 20071106, 20080107).

2001 年 Bessière 等提出 AC-2000 算法和 AC-2001 算法[3]。后来相继提出了 AC-3.1[4],AC-3.2 和 AC-3.3 等算法[5],它们都是对 AC-3 算法的改进。将 AC 算法用于 BT(backtrack)框架之中,就是目前主流的 CSP 求解算法 BT + MAC[6]。在用 BT + MAC 算法处理问题之前,加入一个预处理过程,可以减小搜索空间,提高整个算法的效率。上面提到的弧相容算法都可以与 BT + MAC 相结合,使整个求解问题的效率得到提高。

启发式(heuristic)在问题的求解过程中扮演着十分重要的角色,常见的启发式有变量启发式和值启发式[7]。应用变量启发式的求解算法可显著提高算法的效率,使其能更快地找到解或者更早地发现该问题无解;值启发式则使算法优先在有解可能性大的空间中进行搜索,避免将不参与解的值赋予变量,从而达到提高搜索效率的目的。在一般 BT + MAC 算法中采用的启发式是一种静态启发式策略,但问题在求解过程中变量与变量的取值对求解的影响会发生变化,因此本文在吸取静态值启发式优点的基础上,利用预处理阶段和回溯过程相容检查中的有用信息,提出了一种动态值启发式算法——回溯动态值启发式(Backtrack-Dynamic Value Heuristic, BT-DVH)算法,通过随机问题和标准库(benchmark)测试,表明 BT-DVH 算法的效率是维持弧相容(Maintaining Arc Consistency, MAC)算法的数倍,在求解 CSP 算法上具有很大的优势。

1　约束满足问题和启发式的应用

定义 1　CSP 由三元组 (X, D, C) 构成。其中:$X = \{x_1, x_2, \cdots, x_n\}$ 表示问题中变量的集合;$D = \{d_1, d_2, \cdots, d_n\}$ 表示变量的值域,$d_i\ (i = 1, 2, \cdots, n)$ 是 $x_i\ (i = 1, 2, \cdots, n)$ 的值域;$C = \{c_1, c_2, \cdots, c_e\}$ 是约束集合,该集合作用于 X,使 X 取值受限。

二元 CSP 是指 C 中每一个约束都只涉及到两个变量,它可以用约束图表示。CSP 的一个解是对所有变量的一组赋值集合,这组赋值满足 C 中所有的约束。为更快地找到 CSP 的解,在算法进行过程中,剔除一些不能参与解的值,使算法更有效率地进行,该过程叫做约束传播或者相容性技术[8]。相容性技术包括弧相容(arc consistency)、路径相容等,而弧相容技术则是最主要的相容技术,文献[9]包含了对弧相容技术的一些研究。

定义 2　一个有向弧 (x_i, x_j) 是弧相容的,当且仅当 x_i 中每一个值 a,在 x_j 中都存在一个值 $b(b \in D(x_j))$,(a, b) 满足 x_i, x_j 上的二元约束 $C(x_i, x_j)$。一个 CSP 是弧相容的,当且仅当此 CSP 的约束图中每一条弧都是弧相容的。文献[10]给出了相关的详细概念,并对其进行了相关的研究和分析。

另外一类相容性技术——单弧相容(Singleton Arc Consistency, SAC)也值得关注,一个 CSP 问题 P 是 Singleton 弧相容的,当且仅当对于问题中任意变量 x, $\forall a \in D(x)$, $P|_{x=a}$ 是弧相容的。其中 $P|_{x=a}$ 是将原问题 P 中 x 的论域 $D(x)$ 用单独的 $\{a\}$ 进行替换。Singleton 弧相容是基于弧相容的,但是移走冗余值的能力远高于弧相容技术。2000 年,Prosser 等提出了 SAC-1 算法[11],2005 年 Lecoutre 等提出了 SAC-3 算法[12],这些单相容算法都可以参与求解 CSP。

启发式是一种寻找下一个的策略(look-ahead scheme),它能够引导并决定下一个该实例化的变量或者变量中该取的值,主要分为变量启发式和值启发式。在结合回溯框架的 CSP 求解算法中,需要实例化变量时,经常会用到启发式搜索。启发式搜索的关键是启发式规则的选择。相同的 CSP 选择的启发式规则不同,将导致搜索的效率和扩展过程大不相同。如图 1 所示是一个 CSP,其中 $C_1: X_1 < X_4$, $C_2: X_2 + X_4 = 4$, $C_3: X_3 + X_4 = 3$。如果按照变量的序号顺序赋值,其搜索树如图 2 所示,一共展开了 19 个节点。图 3 是加入了变量启发式的搜索树,先实例化约束最多的 X_4,可以看到一共展开了 10 个节点,极大地提高了效率。

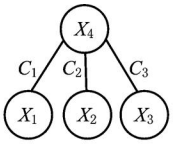

图1　一个简单的CSP

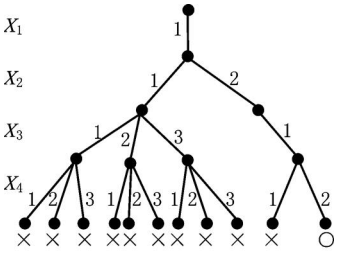

图2　未加入启发式的CSP搜索树

在变量启发式方面,1965 年 Golomb 和 Baumert 首先提出了一种动态变量排序算法,该算法优先实例化最小的变量[13]。1980 年,Haralick 和 Elliott 推广了值域启发式,证明了值域与向前检查算法结合的有效性[14]。1996 年,Bessière 和 Régin 提出另一种值域

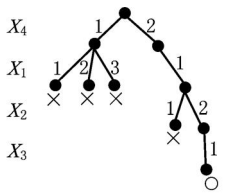

图3 加入启发式的CSP搜索树

的归纳方法,这种启发式称为 dom/ deg[15],它根据变量的值域大小或约束图中节点的度来对变量进行排序,然后选择值域最小的变量进行实例化。在值启发式方面,1988 年 Dechter 和 Pearl 提出了基于估算子问题解数的值启发式[16],但是他们对动态的值启发式的研究较少。

2 BT-DVH算法

本文首先介绍求解 CSP 传统的 BT + MAC 算法执行流程图。文献[17]曾对此类问题有过研究和改进。

如图 4 所示,BT + MAC 算法的执行过程如下:原问题 P_0 经过预处理,剔除一些不参与解的值,成为新问题 P_1;将 P_1 中的变量 X_1 实例化为其值域中的某个值 a,得到新的子问题 P_1';对 P_1' 进行局部相容性检查(lc-local consistency checking),使其在不影响 CSP 的解的正确性和完整性的前提下,尽量缩小问题的规模;这样就得到新的子问题 P_2,再对 P_2 进行实例化以及相容性检查,以此类推;如果遇到相容性检查失败,则回溯到上一个变量,继续执行算法,直到将所有变量都实例化后返回 CSP 的解,或者在问题无解的情况下算法失败退出。为提高效率,BT-DHV 算法在 BT + MAC 算法的基础上,加入了变量启发式和动态的值启发式。

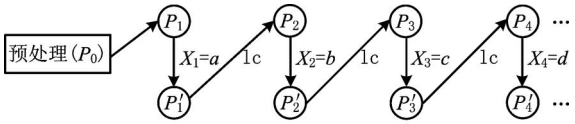

图4 BT+MAC算法的执行流程图

加入变量启发式是为了优先实例化能够尽早发现冲突或者利于减少回溯次数的变量。2007 年,孙吉贵等提出了基于最先失败原则(FFP-fail first principle)的约束传播算法[18]。该算法指出,应该选取具有最小论域的变量作为下一个要被实例化的对象,目的是尽可能早地发现空域。文献[18]提出了一种动态的计数方式:用参数 deleted 代表由变量 X_i 所引起的那次传播开始到本次传播为止,变量 X_i 被删除值的个数。在每次选择时,选取 deleted 计数最大的那个变量。初始化时 deleted = 0。具体算法如下:

```
Function getvariable
    index ←0;
    遍历所有的变量
      i ←0;
      if 变量 i 没有被实例化
        if 变量 i 的值域比变量 index 的值域小
            index ←i;
        else if 变量 i 的值域等于变量 index 的值域
            if 变量 i 的 deleted 大于变量 index 的 deleted
                index ←i;
    return index;
```

getvariable() 函数优先实例化论域最小的变量,由算法第 6 行和第 7 行实现;如果两个变量论域相同,则比较 deleted 的大小,由算法第 9 行和第 10 行实现。最后,返回变量的标号 index。

加入值启发式有助于扩展出更大的解空间。目前,相对于其他的启发式,动态的值启发式的研究仍然较少,文献[19]是对静态值启发式的研究。本文提出的则是一种基于动态的值启发式算法。对于每一个变量,其值域中的每个值都设定一个参数 turnups,用来记录该值在约束 C 中出现的次数,一个值在约束 C 中出现的次数越多,说明将这个值赋予此变量后,扩展出来解的可能性就越小。在预处理过程中,先统计所有变量的值参数 turnups。在进行相容性检查时,对于剔除的值,找到其在约束 C 中的值对,分别将对应的变量中的对应值 turnups 减 1,从而实现动态的值启发式算法。因此,在每次赋值之前,需要把问题有关变量的 turnups 保存下来,如果遇到回溯,则可将环境恢复。具体实现如下:

```
Function getvalue
    if 此变量的值域不为 0
        index ←0;
        遍历此变量中所有的值
            if 值 i 的 turnups 小于值 index 的 turnups
                index ←i;
    return index;
```

现在,给出加入动态变量启发和动态值启发式的完整的 BT-DVH 算法。算法如下:

```
BT-DVH Algorithm
BT-DVH(CSP:P)
    P'←pre_process(P);          /*预处理*/
    if(P'无解)then return"无解";
    freevariables ←P.X;
    while(freevariables ≠∅)do
        x ←getvariable();       /*变量启发式*/
```

```
    v←getvalue();          /*动态值启发式*/
    x←v;
    freevariables←freevariables-{x};
    push(P');
    P'←consistency(P');          /*弧相容*/
    if(isnosolution(P'))then
        P'←pop(P');
      freevariables←freevariables∪{x};
return CSP的解;
```

算法输入一个 CSP,最后输出一个可行解或者无解退出。pre_process()是一个预处理过程,一般使用 SAC 算法,该算法可以剔除冗余值,缩短算法的执行时间。在实例化开始前,调用 getvariable()函数和 getvalue()函数分别选取一个变量和一个值,由算法第 5 行和第 6 行实现。保存现场后,对子问题进行相容性检查,目前效率较高的是弧相容技术,由算法第 10 行实现。

为体现 BT-DVH 算法的优越性,举一个简单的例子,分别用 BT-DVH 算法和传统的 BT + MAC 算法求解,比较它们的回溯次数。如图 5 所示为一个 CSP,X_1, X_2, X_3 的值域都是$\{1,2,3\}$。约束 C_1 是$(1,1)$,$(1,2)$,$(2,2)$;约束 C_2 是$(3,1)$,$(3,2)$;约束 C_3 是$(1,3)$。

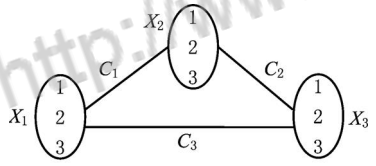

图5　一个CSP

按照 BT-DVH 算法,计算出的 turnups 分别为:

$(X_1,1)=3$,$(X_1,2)=1$,$(X_1,3)=0$;

$(X_2,1)=1$,$(X_2,2)=2$,$(X_2,3)=2$;

$(X_3,1)=1$,$(X_3,2)=1$,$(X_3,3)=1$。

在变量 X_1 中,由于值 3 的 turnups 最小,则先把 X_1 实例化为 3,再把 X_2 实例化为 turnups 值最小的 1,同理 X_3 也实例化为 1。这样就得到了该 CSP 的解{3,1,1}。在此过程中,回溯次数为 0。如果按照传统的 BT + MAC 算法,X_1 实例化为 1,X_2 和 X_3 都只能实例化为 3,则导致冲突,产生了回溯,直到最后得出解{2,1,1},回溯次数为 2。由此可见,BT-DVH 算法比传统的 BT + MAC 算法具有更少的回溯次数和更高的效率。

3　复杂性和正确性分析

为更好地衡量动态值启发式(Dynamic Value ordering Heuristic, DVH)的性能,本章给出了带有此启发式的相容性检查(以弧相容算法 AC-3 为例)的时间复杂度和空间复杂度的分析,然后证明其正确性。

定理 1　带有 DVH 的相容性技术(AC-3 算法)的最坏时间复杂度是 $O(e^2d^4)$。

证明　在最坏的情况下,当值域中的一个值被删除,就遍历所有的约束,这一过程所用的时间是 $O(e)$。当此约束涉及的变量包含这个被删除的值时,就遍历此约束中的所有值对,该过程所用的时间是 $O(d)$。而传统的 AC-3 算法的时间复杂度是 $O(ed^3)$,因此带有 DVH 的 AC-3 算法的最坏时间复杂度是 $O(e^2d^4)$。

虽然最坏时间复杂度大于传统的 AC-3 算法,但是启发式使整个求解算法的效率有所提高。

定理 2　带有 DVH 的相容性技术(AC-3 算法)的最坏空间复杂度是 $O(e+nd)$。

证明　每个变量中的每个值都保存着一个 turpups 参数,因此它消耗的空间是 $O(nd)$。而传统 AC-3 算法的空间复杂度为 $O(e)$,则带有 DVH 的 AC-3 算法的最坏空间复杂度是 $O(e+nd)$。

下面对算法 BT-DVH 的正确性进行证明。在传统的 BT + MAC 算法中,选择实例化的变量和值具有随机性,但是无论先选择哪个变量和值进行实例化,都不影响 BT + MAC 算法的正确性。BT-DVH 算法在不改变传统 BT + MAC 算法结构的前提下,增加了 turpups 数据结构,这并不能影响 BT 框架算法的执行过程。在启发式选择优先实例化的值时,根据 turpups 参数的大小进行选择。但是如前文分析,无论先选择哪个值进行实例化,都能令 BT 框架的算法正常进行。因此,BT-DVH 算法和传统的 BT + MAC 的正确性是等价的,由于 BT + MAC 算法是正确的,BT-DVH 算法也是正确的。

4　实验结果

为说明算法具有正确性和比较理想的效率,笔者采用了常用的随机问题和标准库问题对算法进行测试。通过对这两种例子的测试,可以看出加入动态值启发式的算法比原有 MAC 在效率上有不同程度的提高。在表 1～表 5 和图 6～图 8 中:BT + MAC 代表在经典的回溯算法中采用了弧相容技术,FFP 代表在搜索过程中仅加入了失败优先原则的启发式。为体现动态值启发式相对于静态值启发式的优势,将两者分别进行测试。静态值启发式表示在 FFP 的基础上又加入了值启发式,但无动态变

化,即启发参数在预处理之后恒定;动态值启发式即 BT-DVH 算法,在每次相容检查后将启发参数优化。使用 Java 语言编写程序,测试环境为:硬件 Intel P Ⅳ 3.0 GHz CPU,512 MB RAM;软件 Windows XP Professional SP3,Eclipse 3.5.1。其中每个用例测试 50 次,取平均值作为最后结果。

4.1 随机约束满足问题测试

随机 CSP 的测试用例由产生器随机产生,这里只产生二元随机问题。二元随机 CSP 中含有四个参数<n, m, p_1, p_2>,其中 n 表示变量的个数,m 表示变量论域的大小,每个论域的大小相同。p_1 表示该 CSP 的密度,即约束图中边的个数和 $n\times(n-1)/2$ 的比值,p_2 为每个二元约束的松紧度。本文选取参数为<20,30,0.4,p_2>的这组用例进行测试。

p_2 先从 0.05 测到 0.95,取 FFP 和动态值启发式进行对比,如图 6 所示。可以看出,0.5～0.7 之间是问题最复杂的区间。

为了更细微地对比,绘制参数 p_2 在 0.5～0.56 区间的运行时间和回溯次数的图表。当松紧度 p_2 > 0.56 时,问题易无解。由于值启发式的目的是提前实例化变量的优先值,使其更有可能扩展出 CSP 的解,当问题无解时加入值启发式的意义不大,有时会导致反效果,降低算法的效率。测试所得到的回溯次数如表 1、表 2 和图 7 所示,其运行时间如表 3、表 4 和图 8 所示。

表 1 <20,30,0.4,p_2>在 0.50～0.53 之间的回溯次数

	0.5	0.505	0.51	0.515	0.52	0.525	0.53
BT+MAC	32.50	43.20	78.00	82.34	91.12	135.52	288.20
FFP	15.98	23.98	31.64	33.80	37.10	85.22	100.94
静态值启发	9.38	14.24	14.04	23.94	33.88	35.50	57.94
BT-DVH	5.96	8.12	11.86	18.28	21.48	24.46	43.84

表 2 <20,30,0.4,p_2>在 0.50～0.53 之间的运行时间 ms

	0.5	0.505	0.51	0.515	0.52	0.525	0.53
BT+MAC	808	1 018	1 295	1 496	1 653	2 427	4 695
FFP	761	867	956	911	1 079	1 694	2 078
静态值启发	679	799	775	861	872	1 084	1 320
BT-DVH	703	732	760	845	862	943	1 184

表 3 <20,30,0.4,p_2>在 0.535～0.560 之间的回溯次数

	0.535	0.54	0.545	0.55	0.555	0.56
BT+MAC	318.3	334.1	774.44	941.8	1 116.34	2 625
FFP	136.2	200.3	264.58	327	716.44	936
静态值启发	73.3	111.2	211.14	229	421.56	704
BT-DVH	55	90.5	131.36	160	343.46	407

表 4 <20,30,0.4,p_2>在 0.535～0.560 之间的运行时间 ms

	0.535	0.54	0.545	0.55	0.555	0.56
BT+MAC	5 874	6 228.7	15 061	20 150	26 523	60 763
FFP	2 667	3 846	5 347	7 105	16 138	22 018
静态值启发	1 584	2 238	4 043	4 565	8 455	15 316
BT-DVH	1 366	1 871	2 659	3 342	7 247	8 849

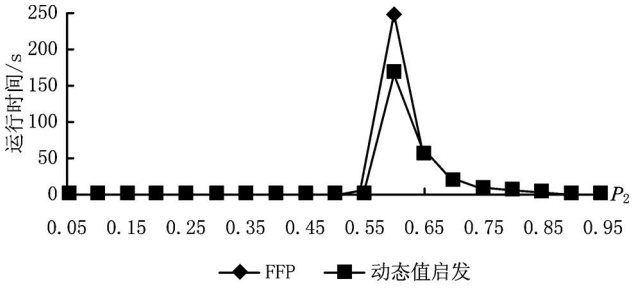

图6 <20,30,0.4,p_2>在0.05～0.95之间的运行时间

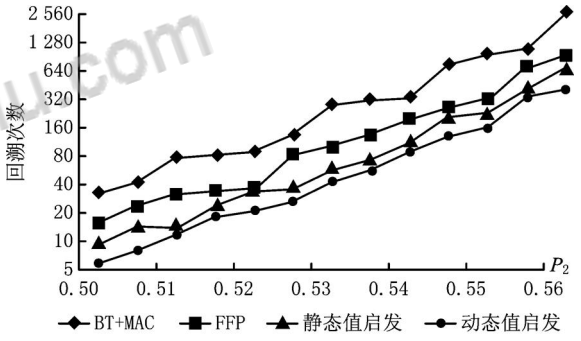

图7 <20,30,0.4,p_2>在0.50～0.56之间的回溯次数对比

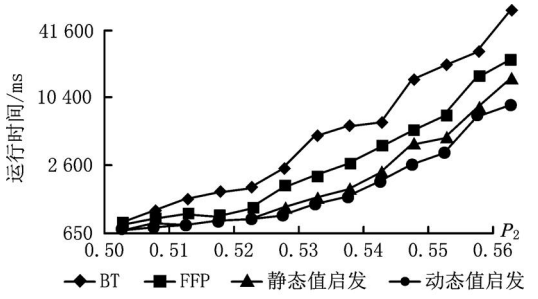

图8 <20,30,0.4,p_2>在0.50～0.56之间的运行时间对比

由表 1～表 4 和图 7 可以看出,加入静态值启发式的算法效率相对于没有加值启发式的算法有一定的提高。当加入动态值启发式之后,回溯次数明显减少,效率进一步提高,能达到经典 BT+MAC 算法的 2～10 倍。

5.2 标准库问题测试

为了进一步说明 BT-DVH 算法的高效性,本文测试了标准库问题。随机选取了几组 frb 问题,将 BT+MAC,FFP 和动态值启发式三个算法进行了

测试。测试结果如表 5 所示。

表 5　几组标准库问题的回溯次数对比

	frb30-15-2	frb30-15-3	frb30-15-5	frb35-17-1
BT + MAC	11 133	102 929	11 917	> 10 000
FFP	9 102	40 382	12 317	65
BT-DVH	8 836	49 290	8 305	35

结果表明,对于测试的大部分问题,BT-DVH 算法都能有效减少回溯次数,从而达到提高效率的目的。随机 CSP 和标准库问题中的约束,均以值对的形式出现,但是在很多情况下,问题用表达式或其他形式来表示约束。因此,如果碰到其他形式的约束表示,可以先将约束转化成值对的形式再求解,亦可取得满意的效果。

6　结束语

在求解 CSP 的过程中,启发式具有不可替代的作用,但是目前研究的重点仅在于变量启发式和静态的值启发式。本文吸收了静态值启发式的优势,利用预处理阶段和回溯过程的相容检查中的有用信息,提出了一种动态的值启发式算法——BT-DVH 算法。该算法随着每次相容性检查对启发参数进行动态更新,从而达到先将更有可能成为解的值赋予变量,对有解的 CSP 求解效率的提高具有明显的效果。测试结果表明,无论是随机 CSP 还是标准库问题,此算法的执行效率都高于目前主流算法,能达到其 2～10 倍。但是本文所提出的动态值启发式主要针对有解的 CSP,下一步工作将针对全部的 CSP 进行研究,重点放在无解的问题上,以求得到更优的求解算法。

参考文献:

[1] MACKWORTH A K. Consistency in networks of relations [J]. Artificial Intelligence, 1977, 8(1): 118-126.

[2] MOHR R, HENDERSON T C. Arc and path consistency revisited[J]. Artificial Intelligence, 1986, 28(2): 225-233.

[3] BESSIERE C, REGIN J C. Refining the basic constraint propagation algorithm[C]// Proceedings of the 17th International Joint Conference on Artificial Intelligence. San Frarcisco, Cal., USA: Morgan kaufmann Publishers Inc., 2001: 309-315.

[4] ZHANG Y, YAP R H C. Making AC-3 an optimal algorithm [C]// Proceedings of the 17th International Joint Conference on Artificial Intelligence. San Frarcisco, Cal., USA: Morgan kaufmann Publishers Inc., 2001: 316-321.

[5] LECOUTRE C, BOUSSEMART F, HEMERY F. Exploiting multidirectionality in coarse-grained arc consistency algorithms [C]// Proceedings of CP'03. Berlin, Germany: Springer-Verlag, 2003: 480-494.

[6] SABIN D, FREUDER E C. Contradicting conventional wisdom in constraint satisfaction[C]// Proceedings of the 11th ECAI. Berlin, Germany: Springer-Verlag, 1994: 125-129.

[7] VAN BEEK P. Backtracking search algorithms[M]// Handbook of Constraint Programming. Amsterdam, The Netherlands: Elsevier, 2006: 85-134.

[8] BARTAK R. Theory and practice of constraint propagation [C]// Proceedings of the 3rd Workshop on Constraint Programming in Decision and Control. 2001: 7-14. DOI: 10.1.1.86.5504.

[9] SABIN D, FREUDER E. Contradicting conventional wisdom in constraint satisfaction[C]// Proceedings of CP. Berlin, Germany: Springer-Verlag, 1994: 10-20.

[10] TSANG E. Foundations of constraint satisfaction[M]. Salt Lake City, Utah, USA: Academic Press, 1993.

[11] PROSSER P, STERGIOU K, WALSH T. Singleton consistencies[C]// Proceedings of CP'00. Berlin, Germany: Springer-Verlag, 2000: 353-368.

[12] LECOUTRE C, CARDON S. A greedy approach to establish singleton arc consistency[C]// Proceedings of IJCAI'05. San Francisco, Cal, USA: Morgan Kaufmann Publishers Inc., 2005: 199-204.

[13] GOLOMB S W, BAUMERT L D. Backtrack programming [J]. Journal of the ACM, 1965, 12(4): 516-524.

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

[15] BESSIERE C, REGIN J C. MAC and combined heuristics: Two reasons to forsake FC (and CBJ?) on hard problems [C]// Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming. Berlin, Germany: Springer-Verlag, 1996: 61-75.

[16] DECHTER R, Pearl J. Network-based heuristics for constraint satisfaction problems [J]. Artificial Intelligence, 1988, 34(1): 1-38.

[17] SUN Jigui, ZHU Xingjun, ZHANG Yonggang, et al. An approach of solving constraint satisfaction problem based on preprocessing[J]. Chinese Journal of Computers, 2008, 31(6): 919-926 (in Chinese). [孙吉贵,朱兴军,张永刚,等,一种基于预处理技术的约束满足问题求解算法[J]. 计算机学报, 2008, 31(6): 919-926.]

[18] SUN Jigui, ZHU Xingjun, ZHANG Yonggang, et al. Constraint propagation based on fail first principle[J]. Journal of Chinese Computer Systems, 2008, 29(4): 678-681 (in Chinese). [孙吉贵,朱兴军,张永刚,等,最先失败原则的约束传播算法[J]. 小型微型计算机系统, 2008, 29(4): 678-681.]

[19] LI Bo, ZHU Xingjun, ZHANG Changsheng. Value ordering for solving CSP based on preprocessing[C]// Proceedings of the 4th International Conference on Natural Computation. Washington, D.C., USA: IEEE Computer Society, 2008: 214-216.

作者简介:

王孜文(1985 -),男,山东淄博人,硕士研究生,研究方向:约束程序,E-mail: wang_ziwen@qq.com;

+李占山(1966 -),男,吉林公主岭人,教授,博士,研究方向:基于模型的诊断、智能规划与决策、约束问题求解,E-mail: zslizsli@163.com;

艾　阳(1985 -),女,吉林吉林人,硕士研究生,研究方向:模型诊断;

李宏博(1985 -),男,吉林公主岭人,硕士研究生,研究方向:约束程序。