

一个基于动态代理次序的分布式约束求解算法

张剑鹏¹, 高 健²

(1. 建设部沈阳煤气热力研究设计院, 辽宁 沈阳 110026;

2. 东北大学 软件学院, 辽宁 沈阳 110004)

摘 要:最近分布式约束满足问题逐渐成为人工智能领域一个新的研究热点,它的提出将约束满足问题的应用范围扩展到复杂的分布式环境。并发搜索是求解分布式约束满足问题的一个高效算法。文中改进了并发搜索中的变量选择策略,将动态代理次序应用到其中,同时提出了一个适合于分布式条件下的基于动态代理次序的并发搜索算法。多组随机生成问题实验结果显示加入动态代理次序的并发回溯搜索在求解效率和通信量方面都表现出优异的性能。

关键词:分布式约束满足;动态代理次序;并发搜索

中图分类号:TP18

文献标识码:A

文章编号:1673-629X(2009)06-0152-04

A Dynamic Agent Ordering Based Algorithm for Distributed CSPs

ZHANG Jian-peng¹, GAO Jian²

(1. Shenyang Gas & Heating Research and Design Institute of Construction Ministry, Shenyang 110026, China;

2. College of Software, Northeast University, Shenyang 110004, China)

Abstract: Distributed CSPs have become a new hotspot in AI recently. It extends the application of CSPs to the complex distributed environment. Improves the agent ordering strategy in concurrent search for solving distributed CSPs, and combines dynamic agent ordering with concurrent search. At the same time, an algorithm based on dynamic agent ordering that is fit for distributed environment is proposed. Experiments of several random CSPs have been done, and the results show that the improved method performs better on efficiency and communication overhead.

Key words: distributed constraint satisfaction; dynamic agent ordering; concurrent search

0 引 言

作为人工智能领域中的一个重要问题,约束满足问题(CSP)是近年来一个非常活跃的研究方向,实践证明,大量现实问题都可转化为约束满足问题来求解,具体如调度、规划和产品配置等^[1,2]。约束满足问题可以描述为一个有限变量集合和一个定义在这个变量集合上的约束集合^[3]。问题的解是为每个变量赋一个相应论域中的值,使之满足问题中所有的约束。一个分布式约束满足问题是一个变量和约束都分布在多代理环境^[4]中的约束满足问题,每个代理包括一个局部约束网络,并和其他代理中的变量通过约束连接。代理对变量赋值,使其满足局部约束网,并与其它代理交互通信来满足代理间的约束。分布式约束满足问题在

现实生活中有着诸多的应用,如分布式资源分配、分布式调度、多代理真值维护等都可以化成分布式约束满足问题来求解^[5]。

近年来,一些分布式约束满足问题的求解算法相继被提出^[6]。集中式回溯是最基本算法,它将每个代理的所有信息都传送到一个代理上求解,这种方法通信负担大并且没有并行性;同步回溯算法规定了代理的实例化顺序,一个代理从它前一个代理接收一个部分赋值,并实例化所包含的变量再传给下一个代理,该算法虽然比集中式算法的通信量少,但同样不具备并行求解的能力;异步回溯克服了上述缺点,每个代理工作是并发且异步的,它们先把所包括变量实例化再与其他代理通信。并发搜索^[7]是多个搜索进程并发回溯的求解算法族,每个搜索进程对应一个部分赋值,部分赋值在代理间传递,并使其搜索空间没有交集。文献^[7]还提出了一个动态分裂部分赋值的并发搜索算法,算法可以根据问题的难易程度动态调节搜索进程使负载均衡,实验证明这类算法的效率要优于其它算法。

收稿日期:2008-10-05;修回日期:2009-01-11

基金项目:国家自然科学基金项目(60773097);吉林省青年科研基金项目(20080107)

作者简介:张剑鹏(1980-),男,辽宁沈阳人,助理工程师,从事计算机网络方面的研究。

动态变量次序启发式被广泛地应用在各种集中式回溯算法中,以最先失败原则为基础的最小值域优先实例化技术较大程度地提高了回溯算法的效率。而在分布式环境中,问题信息分布在各个代理中,受通信量和保密性等因素的限制,动态变量次序启发式的信息无法由某一个代理计算出来,这样就无法给出一个用于指导部分赋值在代理间传递的动态代理次序,因此影响了并发搜索的效率。文中提出了一个在分布式环境下依据最先失败原则计算动态代理次序的方法,这种方法通过计算代理中变量赋值后对其它代理中未赋值变量的支持数来估计未赋值变量论域的大小,从而确定变量实例化的次序和部分赋值在代理间的传递次序。实验结果表明,增加动态代理次序的并发搜索算法求解效率有明显提升。

1 分布式约束满足问题

1.1 问题的定义

约束满足问题^[8](约束网络)被定义为一个三元组 $\langle X, D, C \rangle$, 其中 $X = \{X_1, X_2, \dots, X_n\}$ 是一个有限变量集合, 每一个 X_i 的取值范围都对应一个有限论域 D_{x_i} , 且有限论域集 $D = \{D_{x_1}, D_{x_2}, \dots, D_{x_n}\}$, 所有论域的笛卡尔乘积组成该问题的搜索空间, 记为 $D_s = D_{x_1} \times D_{x_2} \times \dots \times D_{x_n}$, C 是一个有限约束集合。集合 C 中的一个约束 c 限制了该约束中所包含的变量 $V(c)$ 容许取值的组合。如果对于 C 中的每个 c 有 $|V(c)| = 2$, 则问题为二元约束满足问题, 二元约束满足问题可以使用图结构来描述, 称之为约束图, 图中节点代表 CSP 中的变量, 连接两个节点的边代表变量间的约束关系。

一个赋值(标记)被定义为一个二元组 $\langle \text{var}, \text{val} \rangle$, var 代表一个变量, val 是赋给 var 的一个相应论域中值。一组赋值组成部分赋值(复合标记)。一个约束满足问题的解是一个包含所有变量的部分赋值, 且满足集合 C 中的所有约束。称 v_1 支持 v_2 , 如果 $\langle X_1, v_1 \rangle, \langle X_2, v_2 \rangle$, 其中 $v_1 \in D_{x_1}, v_2 \in D_{x_2}$, 满足 X_1 和 X_2 之间的约束, v_1 支持变量 X_2 的取值集合记为 $S(X_1, v_1, X_2) = \{\text{val} \mid \text{val} \in D_{x_2} \wedge v_1 \text{ 支持 } \text{val}\}$ 。

一个分布式约束满足问题^[9]由一个代理集合组成 $A = \{A_1, A_2, \dots, A_k\}$, 每个代理都包含一个变量集合 $X_{A_1}, X_{A_2}, \dots, X_{A_k}$, 且满足 $X_{A_1} \cup X_{A_2} \cup \dots \cup X_{A_n} = X, X_{A_i} \cap X_{A_j} = \emptyset (1 \leq i, j \leq k, i \neq j)$ 。代理内组成一个局部约束网络, 代理间通过全局约束连接^[10]。而分布式约束满足问题的解则需要满足所有代理中局部约束和代理间全局约束。文中在研究分布式约束满

足问题算法时, 作如下假设简化模型, 这些假设同样在文献[7]中使用。

- (1) 每个代理仅包含一个变量。
- (2) 问题中所有约束都是二元的。
- (3) 每个代理都可以与其他任何代理通信。
- (4) 每个代理都持有与其相关的约束信息。

1.2 并发搜索算法

并发搜索是多个搜索进程同时在各代理间进行求解的一族算法。在回溯算法中, 每个搜索进程对应一个当前部分赋值(CPA, Current Partial Assignment), CPA在各代理间传递, 且保证搜索子空间不相交。每个代理需要记录接收到的 CPA 信息, 如发送的代理 ID、变量值域等信息, 代理为每个 CPA 中所包含的变量赋值, 并传给其它代理。如果某个 CPA 中变量全部被赋值并且满足所有约束则宣布求解成功; 类似集中回溯算法, 如果某个变量的所有值都不能满足当前的部分赋值则回溯。在搜索过程中 CPA 被分散到各代理中, 减少了代理空闲时间, 提高了并行求解能力。

CPA 对应的搜索子空间是通过划分某一个变量的值域得到的, 若存在 n 个 CPA, 则可选取一个变量将其值域分为 n 份, 每一部分与其它变量组成一个子问题, 并作为一个 CPA 的搜索子空间。

例 1. 一个二元约束满足问题如图 1 所示。该问题由五个变量和六个二元约束组成。若采用两个搜索进程, 可将 X_1 的论域分为两部分 $\{0, 1\}$ 和 $\{2, 3\}$, 分别对应 CPA₁ 和 CPA₂, 两个搜索进程对应的子空间如表 1 所示。上述方式保证划分只与一个变量有关, 划分操作可以在一个代理内完成, 而不影响其它代理中的数据和行为。

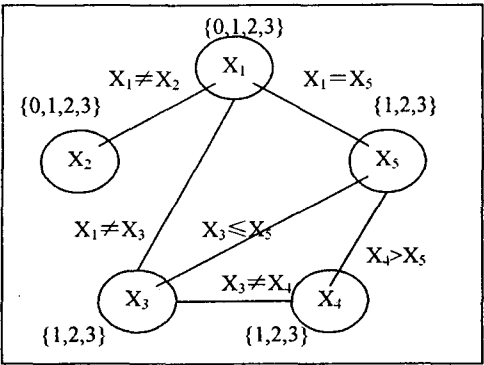


图 1 一个二元约束满足问题的例子

表 1 搜索空间的分割

SP ID	CPA ID	Search Space D_i
1	CPA ₁	$\{0, 1\} \times D_{x_2} \times D_{x_3} \times D_{x_4} \times D_{x_5}$
2	CPA ₂	$\{2, 3\} \times D_{x_2} \times D_{x_3} \times D_{x_4} \times D_{x_5}$

一个代理可以生成若干个搜索进程和其对应的

CPA, 在算法的最初由初始代理建立 CPA, 并启动搜索进程。为了平衡各个搜索进程的负载, 文献[7]提出了动态分裂 CPA 的思想, 算法使用一个简单的启发式计算 CPA 被赋值的次数来确定当前 CPA 是否需要分裂, 若 CPA-Step 超过某个阈值, 则将 CPA 对应的搜索空间划分成若干个子空间, 再为每个子空间创建新的 CPA。新的 CPA 遵循同样的规则在代理间传递, 直到找到一个解或搜索完整个子空间。

回溯算法、动态回溯等经典的约束满足问题求解算法都可以在并发搜索框架下使用, 且实验证明这些算法在效率、通信量等方面都表现出了出色的性能。

2 基于动态变量次序的并发搜索

2.1 动态次序启发式

变量实例化次序会对搜索空间大小产生很大的影响, 因此基于变量次序的启发式在回溯算法中得以应用。变量次序启发式分为静态和动态两种: 静态指在搜索前就确定变量实例化次序, 且在搜索过程中是不能改变的; 动态方式指在实例化一个变量后再确定下一个变量, 这种方式大都以最先失败原则作为基础。

根据最先失败原则, 应该选择约束最紧的变量作为下一个变量进行实例化。衡量变量受约束的紧密程度, 一种较为简单而有效的方式是使用变量值域的大小。由于求解过程中不同的赋值会使变量论域缩减的程度不同, 每个变量值域大小即可区分变量受约束的紧密程度, 它只需要在每次相容性检查后比较所有未实例化变量的值域大小, 从中选出具有最小值域的变量。如果最小值域的变量不唯一, 可以再使用其它的启发式规则。

2.2 启发式计算

在使用 DVO 启发式的回溯算法中, 变量的选择是以相容性检查为基础的, 如果未实例化变量值域没有得到有效的缩减, 动态变量次序启发式的效果也将受到影响。但是相容性检查需要在集中式环境下完成, 基于回溯的分布式约束满足问题求解算法无法通过相容性检查获得启发式信息, 严重地影响了求解效率。

文中提出一种近似的确定代理次序的方法, 这种方法通过近似估算变量值域大小来代替通过约束传播获得的变量值域精确值。这里引入了支持数的概念, 变量 X_i 在变量 X_j 的取值 val 下的支持数被定义为 $|S(X_i, \text{val}, X_j)|$, 并且算法为每个 CPA 中的未赋值变量提供一个值域大小计数器 $U(X_i)$, 初始值为 $|D_{x_i}|$ 。当一个变量被赋值时, 这个赋值可能对所有未赋值变量产生影响, 算法则需要更新这些未赋值变量值域大小的估计值。设变量 X_j 被赋值 val, X_i 为一个未赋值变

量, 在更新 $U(X_i)$ 时, X_i 值域的最大值不会超过 $|S(X_i, \text{val}, X_j)|$, 即 $U(X_i) \leq |S(X_i, \text{val}, X_j)|$, 所以 X_j 被赋值后, 更新后的 $U(X_i)$ 也应该小于等于当前 $U(X_i)$ 和 $|S(X_i, \text{val}, X_j)|$ 中的较小值, 根据这一原理, 算法采用如下公式更新 U , 其中 $U^r(X_i)$ 表示更新前变量 X_i 的域大小估计值, $U^{r+1}(X_i)$ 代表本次更新后的值。

$$U^{r+1}(X_i) = \min(U^r(X_i), |S(X_i, \text{val}, X_j)|)$$

算法更新值域大小计数器后, 在未实例化的变量集合中选出一个具有最小 U 的变量, 并将本次收到的 CPA 传递给该变量所在的代理。

因为每个代理可以直接访问它所包含变量相关的约束, 所以当变量赋值后, 所赋的值对相关变量论域的支持数可以在本地计算, 跟随 CPA 传递的仅是未实例化变量的估计论域大小。采用动态变量次序不同赋值的 CPA 在代理间的传递顺序不同, 同样可以达到随机变量选取时平衡代理间负载的效果。

2.3 算法

根据以上讨论的改进策略, 下面提出一个新的并发回溯搜索算法, 记作 ConcBT-DVO。其子过程变量赋值算法(如图 1 所示)尝试为当前变量赋值。同时, 计算近似论域大小的信息。当一个变量被赋值后, 近似论域大小被更新, 而更新前的信息会同 CPA 的基本信息一起被保存在该代理中。在一个 CPA 回溯时被保存的信息重新恢复以便下次赋值。每次选取变量时, 拥有最小近似值域的变量被选出, 作为下次赋值的变量。

算法 1 变量赋值算法

```

Procedure Assign Value
Input: received CPA, the variable  $X_a$  involved in this agent
for each untried value in  $D_{x_a}$ 
    set  $X_a$  to  $V_a$  in CPA
    if (CPA is consistent)
        if (all the variable in CPA is instantiated)
            announce solution & send(STOP)
        return
    else
        update domain sizes(CPA)
        find the variable  $X_m$  which has the smallest domain
        send(CPA, the agent that contains  $X_m$ )
    return
unassign  $X_a$  in CPA
if (initial agent)
    announce no solution & send(STOP)
else
    send(BACKTRACK, last assignee)
return
  
```

图 1 ConcBT-DVO 的赋值函数

算法2(如图2所示)描述了基于动态代理次序的并发搜索算法的主函数流程。当代理接收到一个消息后,首先确认信息的类型。若类型为CPA或BACK-TRACK,则对变量赋值。如果代理找到一个解,则向其它发送求解成功的消息并停止搜索。

算法2 基于动态代理次序的并发搜索算法 ConcBT-DVO

Procedure ConcBT I

Input: the variable X_a involved in this agent & local constraints set C

Initialize support degrees

stop ← false

if (initial agent)

 Create & initialize CPAs

while (not stop)

 mess ← wait & receive a message

 switch mess. type

 case: CPA

 store domain sizes (mess. CPA)

 Assign Value (mess. CPA, X_a)

 case: BACKTRACK

 restore domain sizes (mess. CPA)

 Assign Value (mess. CPA, X_a)

 case: STOP

 stop ← true

return

图2 ConcBT-DVO的主函数

3 实验结果

实验跟随文献[7]中使用的方法:采用随机生成的约束满足问题^[11]作为测试实例。随机问题可以产生大量的测试实例,并且问题的难度也可以控制,更重要的是随机问题提供一个无偏见的测试标准,针对某些特定问题的求解算法和启发式在求解随机问题时不起作用,所以绝大部分通用约束求解算法都采用这种测试方法。二元随机约束满足问题有四个描述参数 $\langle n, m, p_1, p_2 \rangle$, 它表示该约束问题含有 n 个变量,并且每个变量值域大小为 m , p_1 为约束的密度,代表约束图中边的个数和 $n \times (n-1)/2$ (完全图中边的个数)的比值, p_2 为每个约束的松紧度, $p_2 \times m^2$ 则为一个二元约束中不相容值对的个数。实验中计算非并发约束检查次数(Non-Concurrent Constraint Check, NCCC), 并作为效率比较数据。

实验采用的问题实例使用参数 $\langle 15, 10, 0.7, p_2 \rangle$ 生成。对于每个 p_2 , 这里生成 50 个问题实例, 分别测试算法 ConcBT 和 ConcBT-DVO。平均的 NCCCs、信息发送量(MSGs) 如表2所示。

从表2中可以看出改进后的算法求解效率明显提

高,尤其在相变区域(p_2 在 0.3 到 0.5 之间)NCCCs 提高的更明显。虽然对于简单的问题 MSGs 略微有些增加,但是当问题逐渐加难 MSGs 迅速降低。动态分裂 CPA 的方法被加入到算法 ConcBT 和 ConcBT-DVO 中,步长设为 35。问题参数选取与上组实验一致,测试结果如表3所示。

表2 ConcBT 和 ConcBT-DVO 比较的结果

P_2	NCCCs		MSGs	
	ConcBT	ConcBT-DVO	ConcBT	ConcBT-DVO
0.1	119.4	108	1087.4	1486.7
0.2	482.6	226.1	2163.1	1802.9
0.3	28268.8	2838.8	124780.2	18254.4
0.4	302532.8	31265.2	1424111.2	181994.1
0.5	31338.2	2980.2	172786.2	19436.7
0.6	8150.5	449.3	48126.7	4433.8
0.7	2409.7	135.9	15478.9	1601.4
0.8	1011.5	40.9	7349.2	522.0
0.9	336.3	5.8	2423.6	175

表3 基于动态论域分割的并发搜索比较的结果

P_2	NCCCs		MSGs	
	ConcBT	ConcBT-DVO	ConcBT	ConcBT-DVO
0.1	125.1	113.1	1022.8	1467.6
0.2	319.3	213.7	1805.2	1857.3
0.3	5245.6	1310.4	59531.2	15726.4
0.4	109783.6	12830.8	1502817	196057.3
0.5	12312.6	1846.1	180262.5	20257.3
0.6	3890.7	429.3	48073.7	4501.4
0.7	1483.8	138	16304.8	1601.4
0.8	738.2	42.2	7388.4	522
0.9	343	5.8	2638.5	175

4 结束语

启发式规则在约束满足问题求解算法中占据了举足轻重的地位,动态变量次序根据不同的部分赋值选择不同的变量实例化顺序,其中最有效的方法是基于最先失败原则的最小值域优先启发式。笔者把动态变量次序应用到求解分布式约束满足问题的并发回溯算法中,将赋值对变量的支持数引入到近似计算最小变量值域过程中,从而动态地确定部分赋值在代理间传递的次序。

还展示加入动态代理次序后的并发回溯算法在求解随机生成问题所表现出的效率。尤其在求解相变区域内的难问题时,使用 ConcBT-DVO 算法的非并发约束检查次数是 ConcBT 算法的 9 倍,通信量也比原算

(下转第 159 页)

最后得到了问题的最优解。使用 EPN 进行 Job-Shop 调度问题建模时,为便于理解,将库所细分为状态库所、资源(机器)库所和资源获得库所,变迁则分为工序变迁和资源分配变迁。

对于实际优化问题,在时间 Petri 网和着色 Petri 网的基础上,提出了一种新的扩展 Petri 网 (EPN) 模型,在 EPN 引入了库所颜色,以便于优化决策。并在 EPN 中引入了条件矩阵 Q 用于计算和判断变迁的激发,给出了 Petri 网与遗传算法相结合的优化方法和具体算法。在 Petri 网和遗传算法的基础上,提出了一种基于扩展时间 Petri 网的单亲遗传算法解决 Job-Shop 调度问题的新方法。由于算法中遗传算子是对 Petri 网模型中的元素进行操作,与问题空间的元素无关,因此,与其它调度算法相比,它有较强的通用性。

3 结束语

介绍了 workflow 模型验证和工作流优化方法,在优化算法方面做了探讨,对进一步研究基于 Web 服务方式的工作流执行优化有一定的参考价值。模型验证是模型证实和性能分析的基础。使用新的扩展 Petri 网结合遗传算法优化方法,对典型案例生产车间作业 (Job-Shop) 调度问题进行求解,建立了该 Job-Shop 调度的 EPN 模型,采用遗传算法对这个 Petri 网模型进行优化,体现建模和优化方法的有效性。在实际的企业业务过程可能要复杂得多,在工作流过程优化方面的研究还值得进一步深入。

参考文献:

- [1] 周江波,凌 鸿,胥正川. 基于 Petri 网的工作流优化分析[J]. 中国管理科学, 2005, 13(3): 50-55.
- [2] Goldratt E M, Cox J. The Goal[M]. Aldershot, England: Gower, 1993.
- [3] Reijers H A. Design and Control of Workflow Processes[M]. Berlin: Springer-Verlag, 2003.
- [4] Poyssick G, Hannaford S. Workflow Reengineering[M]. Mountain View, California: Adobe Press, 1996.
- [5] Rupp R O, Russell J R. The Golden Rules of Process Redesign[J]. Quality Progress, 1994, 27(12): 85-92.
- [6] Seidmann A, Sundararajan A. The Effects of Task and Information Asymmetry on Business Process Redesign[J]. International Journal of Production Economics, 1997, 50(2-3): 117-128.
- [7] 时亚娟. 面向工作流的优化问题求解方法研究[D]. 天津: 河北工业大学, 2006.
- [8] 王小平, 曹力明. 遗传算法-理论、应用与软件实现[M]. 西安: 西安交通大学出版社, 2002: 10-12.
- [9] 肖志娇, 常会友, 衣 杨. 成本约束下工作流时间最小化的资源配置优化[J]. 系统仿真学报, 2006, 18(11): 3320-3323.
- [10] van der Aalst W M P. The application of Petri nets to Workflow Management[J]. The Journal of Circuits, Systems, and Computers, 1998, 8(1): 21-66.
- [11] 朱庆生, 付 新. 基于 Web Service 的工作流系统及优化[D]. 重庆: 重庆大学, 2005.
- [12] 周卫东, 杨加敏, 贾 磊, 等. 一种 Petri 网结合遗传算法的优化方法及应用[J]. 山东大学学报, 2005, 35(4): 59-67.

(上接第 155 页)

法降低。

参考文献:

- [1] Amilhastre J, Fargier H, Marquis P. Consistency restoration and explanations in dynamic CSPs - Application to configuration[J]. Artificial Intelligence, 2002, 135: 199-234.
- [2] 邵伟平, 刘永贤, 郝永平, 等. 基于分布式约束满足的产品配置研究[J]. 东北大学学报: 自然科学版, 2007, 28(1): 103-106.
- [3] 孙吉贵, 景沈艳. 非二元约束满足问题求解[J]. 计算机学报, 2003, 26(12): 1746-1752.
- [4] 董甲东, 郑春香. 分布式系统的时间同步容错机制研究[J]. 计算机技术与发展, 2008, 18(3): 99-101.
- [5] 王秦辉, 陈恩红, 王煦法. 分布式约束满足问题研究及其进展[J]. 软件学报, 2006, 17(10): 2029-2039.
- [6] Yokoo M, Hirayama K. Algorithms for Distributed Constraint

Satisfaction: A Review[J]. Autonomous Agents and Multi-Agent Systems, 2000(3): 185-207.

- [7] Zivan R, Meisels A. Concurrent search for distributed CSPs[J]. Artificial Intelligence, 2006, 170: 440-461.
- [8] 孙吉贵, 高 健, 张永刚. 一个基于最小冲突修补的动态约束满足求解算法[J]. 计算机研究与发展, 2007, 44(12): 2078-2084.
- [9] Yokoo M, Durfee E H, Ishida T, et al. Distributed constraint satisfaction for formalizing distributed problem solving[C]//in Proc. 12th IEEE Int. Conf. Distributed Comput. Syst. Los Alamitos: IEEE Computer Society Press, 1992: 614-621.
- [10] Faltings B, Yokoo M. Introduction: Special Issue on Distributed Constraint Satisfaction[J]. Artificial Intelligence, 2005, 161: 1-5.
- [11] Gent I P, MacIntyre E, Prosser P, et al. Random constraint satisfaction: flaws and structure[J]. Constraints, 2001, 6(4): 345-372.