ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.10, October 2006, pp.2029−2039
DOI: 10.1360/jos172029

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

分布式约束满足问题研究及其进展*

王秦辉, 陈恩红+, 王煦法

(中国科学技术大学 计算机科学技术系,安徽 合肥 230027)

Research and Development of Distributed Constraint Satisfaction Problems

WANG Qin-Hui, CHEN En-Hong+, WANG Xu-Fa

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

+ Corresponding author: Phn: +86-551-3602824, Fax: +86-551-3603388, E-mail: cheneh@ustc.edu.cn

Wang QH, Chen EH, Wang XF. Research and development of distributed constraint satisfaction problems. *Journal of Software*, 2006,17(10):2029−2039. http://www.jos.org.cn/1000-9825/17/2029.htm

Abstract: With the rapid development and wide application of the Internet technology, many problems of Artificial Intelligence, for example scheduling, planning, resource allocation etc., are formally distributed now, which turn into a kind of multi-agent system problems. Accordingly, the standard constraint satisfaction problems turn into distributed constraint satisfaction problems, which become the general architecture for resolving multi-agent system. This paper first briefly introduces the basic concepts of distributed CSPs, and then summarizes the basic and the improved algorithms. Their efficiency and performance are analyzed and the typical applications of distributed CSPs in recent years are discussed. Finally, this paper presents the extensions of the basic formalization and the research trends in this area. Recent related work indicates that the future work will focus on the theoretical research to present the solid theoretical foundation for the practical problems.
Key words: constraint satisfaction; distributed AI; multi-agent system; search; asynchronous

摘 要: 近年来,随着网络技术的快速发展和广泛应用,人工智能领域中的诸多问题,如时序安排、计划编制、资源分配等,越来越多地以分布形式出现,从而形成一类多主体系统.相应地,求解该类问题的传统约束满足问题也发展为分布式约束满足问题,分布式约束满足已经成为多主体系统求解的一般框架.首先,简要介绍了分布式约束满足问题的基本概念,总结了该问题的基本算法及其改进算法,并对这些算法的效率和性能进行了比较分析.然后,讨论了近年来分布式约束满足问题的若干典型应用;最后,给出了分布式约束满足问题基本形式的扩展和今后的研究方向.分布式约束满足问题最新研究进展表明:今后的工作将着重于面向现实问题求解的理论研究,为实际应用提供坚实的理论基础.
关键词: 约束满足;分布式人工智能;多主体系统;搜索;异步
中图法分类号: TP301 文献标识码: A

自 1974 年 Montanari 在图像处理中首先提出了约束满足问题(constraint satisfaction problems,简称 CSPs)[1]

* Supported by the National Natural Science Foundation of China under Grant No.60573077 (国家自然科学基金); the Program for New Century Excellent Talents in University of China under Grant No.NCET-05-0549 (新世纪优秀人才支持计划)

Received 2006-03-09; Accepted 2006-05-08

以来,约束满足作为一种重要的求解方法在人工智能与计算机科学其他领域的很多问题中都得到了广泛的应用[2],从 n 皇后、图染色等经典问题到时序安排、计划编制、资源分配等大型应用问题,都可以形式化为约束满足问题进行求解.正因为在人工智能领域中的广泛适用,约束满足问题在理论、实验、应用上都得以深入研究,成为人工智能中很成功的问题解决范例之一.其相关成果一直是人工智能权威期刊《Artificial Intelligence》的热点,并有多个专题对此进行讨论;国内也有很多学者致力于约束满足问题的研究,主要的工作有约束程序理论、设计与应用的研究[3,4]、约束归纳逻辑程序设计等方面的研究[5,6],以及约束满足问题的求解研究[7−9]等等.约束满足问题是由一系列变量、变量相应的值域以及变量之间的约束关系组成,目标是为这些变量找到一组或多组满足所有约束关系的赋值.回溯搜索以及约束一致性检查两种基本思想和引入它们中的各种启发式方法构成了多种约束满足问题求解算法.

随着硬件和网络技术的发展,分布式计算环境快速、广泛地在各个领域中得到应用,很多人工智能问题也越来越多地处于分布式计算环境下,使得分布式人工智能成为一个十分重要的研究领域,特别是关系到人工自治 Agent 间需要相互协调影响的分布式问题.如在多智能体系统(multi-agent system,简称 MAS)中,处于同一环境下的 Agent 间通常存在着某种约束,此时,为各个 Agent 寻找一组满足它们之间约束的动作组合的分布式人工智能应用问题都可以看作是分布式约束满足问题(distributed CSPs).分布式约束满足问题是变量以及变量间的约束都分布在不同自治 Agent 中的约束满足问题,每个 Agent 控制一个或多个变量,并试图决定这些变量的值,一般在 Agent 内和 Agent 间都存在约束关系,对变量的赋值要满足所有这些约束.正因为不同的变量和约束是由不同的 Agent 控制,因此,在这种情形下,将所有 Agent 控制的变量及相关的约束等信息集中到一个 Agent,再用传统的集中式约束满足算法进行求解往往是不充分或者是不可能的,有如下几点原因[10]:

(1) 生成集中控制会带来额外开销.如类似于传感网络的约束满足问题很可能自然地分布在由一些同等 Agent 构成的集合中.这种情况下,对问题进行集中控制就需要增加不出现在原有结构中的额外元素.

(2) 信息传递的开销.在很多情况下,约束由复杂的决策过程产生,这些过程是内在于 Agent 并且不可能被集中控制的.集中式算法需要获得这些约束关系就要承担信息传递的开销.

(3) 隐私和安全的保证.在电子商务等情况中,常出现 Agent 之间的约束是不能泄露给竞争者甚至也不能泄露给集中控制的战略信息的情况.此时,隐私只能在分布式方法中得到很好的保护.

(4) 面对失败的鲁棒性.集中控制求解时的失败可能是致命的;而在分布式方法中,一个 Agent 的失败并不是致命的,其他 Agent 可以在忽略已失败 Agent 的情况下找到问题的解.比如在传感网络和基于网络的应用中,当约束求解过程正在进行而参与者可能离开时,都会产生这些问题.

从上述原因可以看出:此类分布式环境中的问题需要更有效的解决方法.随着人工智能领域协作式分布问题研究的深入,Yokoo 等人在文献[11]中提出了分布式约束满足问题的框架和相应算法.作为一种新的技术,它特别适用于表示及求解规模大、难度高的组合问题.所以,分布式约束满足问题成为人工智能领域的一个研究热点.本文在文献[12]对分布式约束满足问题综述的基础上,不仅介绍了分布式约束满足的问题形式和一系列求解算法,还介绍了近年来在分布式约束满足问题基本形式上的扩展和多主体系统的应用.

本文第 1 节介绍分布式约束满足问题的定义.第 2 节详述一系列求解分布式约束满足问题的算法,比如异步回溯、异步 Weak-commitment 搜索、分布式逃逸算法等.第 3 节介绍相应的应用.最后总结该问题上的一些扩展和类似的工作,如开放式、分布式局部约束满足、隐私安全性等.

1 分布式约束满足

1.1 约束满足问题

约束满足问题是在一定的值域范围内为所有变量寻找满足它们彼此间约束关系的赋值的问题,由变量、变量的值域和变量之间的约束组成.

定义 **1**(约束满足问题). 约束满足问题可以形式化为一个约束网[13],由变量的集合、每个变量的值域的集合以及变量间的约束关系的集合来定义,表示为三元组(V,D,C),其中:

V 是变量的集合$\{v_1,\ldots,v_n\}$;

D 是所有变量的值域的集合,$D=\{D_1,\ldots,D_n\}$, D_i 是变量 v_i 的所有可能取值的有限域;

C 是变量之间的约束关系的集合 $C=\{C_1,\ldots,C_m\}$,其中每个约束包含一个 V 的子集$\{v_i,\ldots,v_j\}$和一个约束关系 $R\subseteq D_i\times\ldots\times D_j$.

约束满足方法是一种有效的问题求解方法,它为每个变量在其值域中寻找一个赋值,使得所有约束被满足.

定义 **2**(约束满足问题的解). 约束满足问题的解是分配给问题中所有变量的一组不违反任何约束的赋值.也即一组对所有变量的赋值 $S(v_1,\ldots,v_n)=\{d_1\in D_1,\ldots,d_n\in D_n\},\forall C_r\in C$ 都有 $S(v_{ri},\ldots,v_{rj})=\{d_{ri},\ldots,d_{rj}\}\in R_r$.

例如,n 皇后问题就是典型的约束满足问题.该问题描述为要在 $n\times n$ 的棋盘上摆放 n 个皇后,使得每一行、每一列和每条对角线上只能有一个皇后存在.图 1 是 4 皇后问题的示例以及相应的约束满足问题.

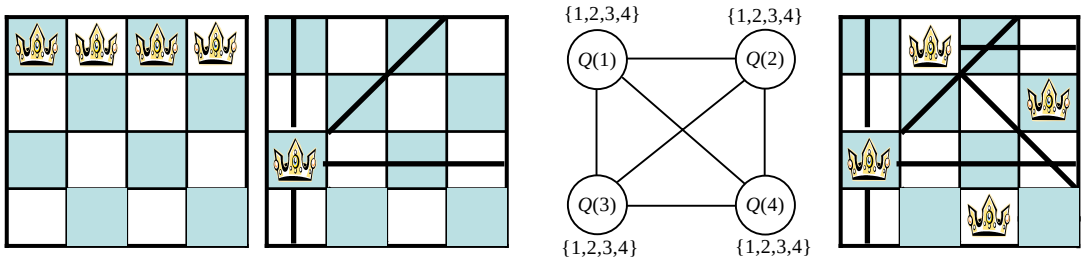

Fig.1 4 queens constraint satisfaction problem

图 1 4 皇后及相应的约束满足问题

图 1(a)表示在 4×4 的棋盘上放置 4 个皇后 Q_1~Q_4,即为变量集合;图 1(b)表示任意行、列、对角线上不能同时有两个皇后,即为约束关系;图 1(c)是相对应的约束满足关系网;图 1(d)是该问题的一个解.

1.2 分布式约束满足问题

分布式约束满足问题是变量和约束都分布在不同自治 Agent 中的约束满足问题.在约束满足问题定义的基础上,可如下定义分布式约束满足问题:

定义 **3**(分布式约束满足问题). n 个 Agent 表示为 $A_1,A_2,\ldots,A_n$,m 个变量为 $v_1,v_2,\ldots,v_m$,m 个变量的值域为 $D_1,D_2,\ldots,D_m$,变量间的约束仍用 C 表示;每个 Agent 有一个或多个变量,每个变量 v_j 属于一个 A_i,表示为 $belongs(v_j,A_i)$;变量间的约束关系分布在 Agent 内或 Agent 之间,当 A_l 知道约束关系 C_k 时,表示为 $Known(C_k,A_l)$.

分布在 Agent 内的约束称为局部约束,而 Agent 间的约束称为全局约束,局部约束可以通过 Agent 的计算来处理,全局约束不仅需要 Agent 的计算,更需要 Agent 间的通信来处理,因此需要如下的通信模式假设:

假设 **1**. Agent 间的通信通过传递消息完成,当且仅当一个 Agent 知道对方地址时才能够传递消息给其他 Agent.

假设 **2**. 传递消息的延时是随机但有限的,任何一对 Agent 间消息接收的顺序与消息发送的顺序是一致的.

假设 **3**. 每个 Agent 只知道整个问题的部分信息.

分布式约束满足中的 Agent 与多智能体系统(MAS)中的 Agent 有着细小的差别[14],分布式约束满足中的 Agent 是遵从协作机制来执行决策行为的计算实体;MAS 中的 Agent 自治地决定是否遵从特定的协作机制,并能以结构化的语义消息交换形式与其他 Agent 进行通信.在将 MAS 形式化为分布式约束满足问题进行求解时,并不考虑这些区别.

每一个 Agent 负责一些变量并决定它们的值,因为还存在着 Agent 间的内在约束,所以,赋值必须满足这些约束.分布式约束满足问题的解的形式化定义为:

定义 **4**(分布式约束满足问题的解). 当且仅当满足下述条件时,分布式约束满足问题找到了解:

$\forall A_i,\forall v_j$ 存在关系 $belongs(v_j,A_i)$,当 v_j 的赋值是 $d_j\in D_j$ 时,$\forall C_k,\forall A_l,Known(C_k,A_l)$都有 C_k 被满足.也即此时对问题中所有变量的赋值满足 Agent 间及 Agent 内的所有约束.

图 2 是一个分布式约束满足问题的示例.该问题为分布式图染色问题,从黑、白、灰这 3 种颜色中选一种分配给 Agent 中的节点变量,使得互相连接的节点颜色不同.图中每个 Agent 都各有 3 个变量,变量之间的边就表示彼此间存在着约束关系,该问题不仅 Agent 内而且 Agent 间都存在着约束关系.

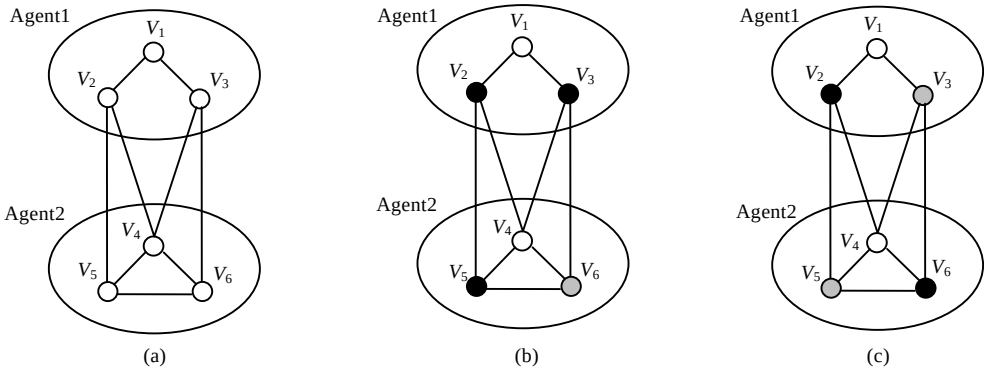

Fig.2 Example of distributed constraint satisfaction problem

图 2 分布式约束满足问题示例

图 2(a)表示该问题的约束关系网;图 2(b)为随机分配着色的初始状态;图 2(c)是该问题的一个解.

1.3 分布式约束满足与并行/分布式计算的区别

分布式约束满足问题看起来与求解约束满足问题的并行/分布式方法[15,16]虽然很相似,但它们从根本上是不同的.并行/分布式方法应用到约束满足问题求解中的目的是为了提高问题的求解效率,针对不同的约束满足问题可以选择任何一种合适的并行/分布式计算机体系结构将问题分而治之,取得较高的求解效率.

而在分布式约束满足中,问题的变量和约束等相关信息从问题给定时就既定地分布于各个自治 Agent 中,所以,研究的出发点是如何在这种固有的情形下有效地获得问题的解.比如一个大规模的 n 皇后问题,可以利用分布式并行计算获得更快的求解速度.而对应到分布式的 n 皇后问题,则是很多个不同的 Agent 拥有数量不同的皇后,通过自我决策和 Agent 间的通信协作来共同达到问题的解.

2 求解分布式约束满足问题的算法

在分布式约束满足问题提出的同时,Yokoo 就在文献[11]中提出了异步回溯算法.近些年来,其他的分布式约束满足求解算法也得到了进一步的研究,特别是异步 Weak-commitment 搜索[17,18]和分布式逃逸算法[19]等.这些算法基本上是由约束满足问题的求解算法而来,是这些传统算法的分布式扩展.但是,因为分布式约束满足问题中 Agent 之间也存在着约束关系,所以,Agent 间需要通信是与传统算法的最大区别.

分布式约束满足算法有两种最基本的消息需要通信,分别是 *ok*?和 *nogood* [11].

定义 **5**(***ok*?**消息). *ok*?是指 Agent 将当前的赋值信息传递给相邻 Agent 的消息.

定义 **6**(***nogood*** 消息). *nogood* 是用来传递约束是否发生冲突而产生新约束的消息.

在文献[12]中,对各种算法的描述都做了如下假设:(1) 每个 Agent 只控制一个变量;(2) 所有的约束都是二元的;(3) 每个 Agent 知道所有和自己变量相关的约束.因此,可以不加区分地使用相同标识 v_i 表示一个 Agent 及其变量,用约束网中的有向边表示约束关系,该有向边由发送消息 Agent 指向接收消息 Agent.

对假设 2 和假设 3 都可以自然地扩展到一般情形.下面分别介绍基于回溯的异步回溯算法、基于优化迭代的分布式逃逸算法和基于混合算法的异步 Weak-commitment 算法.

2.1 异步回溯(**AB**:**Asynchronous backtracking**)

异步回溯算法是由求解约束满足问题的回溯算法而来.所不同的是,异步回溯算法是分布式的、异步的.在异步回溯算法中,每个 Agent 都有一个优先顺序,该优先顺序是预先定义好的,一般由 Agent 标识的字母顺序来

决定,比如,按降序或升序来决定 Agent 的优先序的高低.在该算法中,每个 Agent 除了要发送接受 *ok*?和 *nogood* 消息以外,还要维护一个 *agent_view*,这是用来记录其他 Agent 的当前赋值的.当一个 Agent 接收到 *ok*?消息时,则检查其赋值与优先顺序高的 Agent 的当前赋值是否满足约束关系,如果不满足约束关系产生冲突而不一致就改变自己的赋值;如果该 Agent 值域中没有能与高优先序 Agent 的赋值相一致的值,就产生一个新的约束关系(也就是 *nogood*),并且将 *nogood* 传递给高优先序 Agent,这样,高优先序 Agent 就可以改变自己的赋值.

必须注意到:如果 Agent 不断地改变它们的赋值而不能达到一个稳定状态,那么它们就处于一种无限处理循环,当一个 Agent 的赋值导致其他 Agent 改变赋值而最终影响到自己时就可能产生这种无限循环.为了避免这种情况的发生,在算法中,按照标识的字母序为每个 Agent 定义了优先顺序,*ok*?只能从高优先序 Agent 发送给低优先序 Agent.当产生 *nogood* 时,也是 *nogood* 中的优先序最低的 Agent 接收到 *nogood* 消息.

另外,每个 Agent 的行动都是同时异步发生的,而且 Agent 间的通信是通过消息传递来进行的,所以,*agent_view* 中可能包含有已经无用的信息.因此,每个 Agent 都需要产生新的 *nogood* 进行通信,新 *nogood* 的接收方也必须检查在自己的 *agent_view* 的基础上与此 *nogood* 是否有冲突.

因为算法中最高优先序的 Agent 不会陷入无限处理循环中,文献[18]用归纳法证明了该算法是具备完全性的,也即:如果问题有解存在,那么一定能找到这个解;如果没有解存在,那么算法也会终止,不会陷入无限循环.

近年来,有很多工作都对异步回溯算法进行了改进.在对算法进行扩展时,文献[20]采用了 Agent 的动态重排序;文献[21]引入了一致性维护;文献[22]提出了不存储 *nogood* 消息的分布式回溯算法.这些算法与基本的异步回溯算法相比只是存储 *nogood* 消息的方式不同,它们都需要在未相连的 Agent 之间添加通信连接来检测已经无用的消息.而文献[23]中提出了一种新的异步回溯算法来避免在初始未相连的 Agent 之间动态地增添新的约束,这样就可以避免将一些信息传递给不需要知道的 Agent,从而提高效率.文献[24]从另一个角度提出了如何利用值聚集来减少信息阻塞以及如何利用弧一致维护来提高异步分布式下问题求解的有效性.

2.2 异步Weak-Commitment搜索(AWS:Asynchronous weak-commitment search)

异步回溯算法的局限在于 Agent 的优先顺序是预先定义好的,是静态的.如果高优先序 Agent 的赋值选择得不好,那么,低优先序 Agent 就要进行穷尽查找来修正不利的赋值.异步 Weak-commitment 搜索算法[17,18]的两个基本思想是:为了减少不利赋值的风险而引入了最小冲突启发;更进一步地,Agent 的优先级顺序是可以动态改变的,这样,不利赋值不需要穷尽搜索就能够得到更正.

最小冲突启发是指 Agent 选择值域中那些与其他 Agent 的赋值产生最少冲突的值作为自己的赋值.而为了使 Agent 的优先级顺序能够动态改变,特别地为 Agent 引入了优先值,优先值是非负整数,优先值大的 Agent 具有较高的优先顺序,优先值相等的 Agent 的优先顺序由它们所标识的字母序来决定.初始时,Agent 的优先值均为 0,当 Agent 的赋值与约束发生冲突而不一致时,该 Agent 的优先值就变为相邻 Agent 中的最大优先值再加 1.

与异步回溯算法相比,异步 Weak-commitment 搜索算法的不同在于:

(1) 异步回溯中每个 Agent 只将变量赋值发送给约束相关的低优先级 Agent;而异步 Weak-commitment 搜索中每个 Agent 将变量赋值发送给约束相关的所有 Agent,无论优先级的高低;

(2) *ok*?消息不仅用来传递 Agent 的当前赋值,还用来传递 Agent 的优先值;

(3) 如果当前的赋值与 *agent_view* 不一致,则 Agent 用最小冲突启发来改变赋值;

(4) 如果 Agent 不能找到与自己的 *agent_view* 一致的赋值,就发送 *nogood* 消息给其他 Agent,同时改变自己的优先值.如果 Agent 不能生成新的 *nogood*,那么就不改变自己的优先值,并等待下一条消息.

第 4 步的过程是保证算法的完全性所必需的.因为优先值的改变只有在新的 *nogood* 产生时才发生,而可能的 *nogood* 的数量是有限的,优先值不可能无限地改变,所以到了某个时间之后,优先值就稳定下来,此后,过程就与异步回溯算法一样,故而算法是完全的.为了保证算法的完全性,Agent 要记录所有目前已知的 *nogood*,实际操作时,可以限制记录 *nogood* 的数目,比如每个 Agent 只记录固定数目的最近发生的 *nogood* 消息.

正如前面所提到的假设,这些算法中的 Agent 都只含有一个变量,对于解决 Agent 含有多个变量的问题,无论是采用先让 Agent 找到自己局部问题的所有解后再将问题重新形式化为分布式约束满足问题来求解,还是让

Agent 为每个局部变量生成一个虚拟 Agent 再来模拟这些 Agent 的动作来求解,对大规模问题而言,既没有效率也不能扩展.文献[25]对异步 Weak-commitment 搜索算法进行了扩展,利用变量顺序来解决多个局部变量的问题,称为 Multi-AWS.它的特点是 Agent 按顺序来改变自己变量的值,当某个变量不存在满足所有与高优先序的变量有关的约束时,就增加该变量的优先值.不断反复该过程,当 Agent 中所有局部变量都与高优先序变量满足约束时,就传递值改变消息给相关的 Agent.

2.3 分布式逃逸(DB:Distributed breakout)

在最小冲突回溯等约束满足算法中的爬山(hill-climbing)搜索策略,有时会使求解过程陷入局部最小(local-minima)状态.local-minima 状态就是一些约束没有被满足从而出现了冲突,但是这些冲突的数目不能通过单独改变任何一个变量的值来减少.文献[26]中提出的逃逸算法是一种跳出 local-minima 状态的方法,算法中为每个约束定义了权值,所有冲突约束的权值总和作为一个评估值.当陷入 local-minima 状态时,逃逸算法增加当前状态中冲突约束的权值,使得当前状态的评估值高于其他邻接的状态,从而跳出 local-minima 状态,开始新的搜索.文献[19]在此基础上通过以下两个步骤来实现分布式逃逸:

(1) 始终保证评估值是逐步提高的:相邻的 Agent 对可能会提高的评估值进行交流,只有能够最大提高评估值的 Agent 才有权改变自己的值.如果两个 Agent 不相邻,那么它们有可能同时改变自己的值;

(2) 与检测整个 Agent 是否陷入 local-minima 不同的是,每个 Agent 检测其是否处于 quasi-local-minima 状态,这是比 local-minima 要弱一些的条件,并且能够通过局部通信而检测到.

定义 7(**Agent** A_i 处于 **quasi-local-minimum** 状态). A_i 的赋值使部分约束产生冲突,并且 A_i 和所有 A_i 邻居的可能提高值均为 0.

在分布式逃逸算法中,相邻的 Agent 之间有两种消息的通信:*ok*?和 *improve*. *improve* 消息用来对可能提高的评估值进行通信,Agent 在整个过程中处于 *wait_ok*?和 *wait_improve* 两种交替状态.

分布式逃逸算法有可能陷入无限循环当中,因而不能保证算法的完全性.文献[27]在对分布式逃逸算法进行扩展时,不仅提出了解决 Agent 有多个局部变量的 Multi-DB 算法,还对此算法引入了两种随机方式.一种是利用了随机跳出技术的 Multi-DB$^+$算法,另一种是在 Multi-DB$^+$中又引入随机行走的 Multi-DB^{++}算法.这些算法比其他异步方法有更好的扩展性,但有时也会有更差的性能.

2.4 几种算法的比较

上面介绍了几种求解分布式约束满足问题的基本算法,这些算法有各自的特点和适用性,也各有优点和局限性,表 1 是就上述算法在完全性及解决多局部变量方面的一个定性比较.

Table 1 Comparison of algorithms for solving distributed CSPs

表 **1** 几种基本分布式约束满足问题求解算法的定性比较

Algorithm	AB	AWS	Multi-AWS	DB	Multi-DB
Completeness	Yes	Yes	Yes	No	No
Multi local variables	No	No	Yes	No	Yes

文献[12]对基本的异步回溯(AB)、异步 Weak-commitment(AWS)和分布式逃逸(DB)算法进行了性能比较.通过离散的事件模拟来评估算法的效率,其中每个 Agent 维护自己的模拟时钟,只要 Agent 执行一个计算周期,其时间就增加一个模拟时间单元.一个计算周期包括读取所有消息、执行局部计算和发送消息.假设一个消息在时间 t 发布,则对接收者来说,在时间 t+1 时可用.最后,通过解决问题所需的计算周期数量来分析算法的性能.

表 2 是用分布式图染色问题来评测的结果,其中:Agent(变量)的数目 n=60,90 和 120;约束的数量为 $m=n\times 2$,可能的颜色数目为 3.总共生成 10 个不同的问题,对每个问题执行 10 次不同的初始赋值,并且限制周期最大为 1000,超过后就终止算法.表中列出了算法求解所需的平均周期和求解成功的比例.

明显地,AWS 算法要优于 AB 算法,因为在 AWS 算法中,不需要执行穷尽查找就能修正错误的赋值.

在表 3 和表 4 比较 AWS 与 DB 算法时,Agent(变量)的数目为 n=90,120 和 150,约束的数量分别为 $m=n\times 2$

和 $m=n\times2.7$ 两种情况,可能的颜色数仍然为 3.当 $m=n\times2$ 时,可认为 Agent 间的约束是比较稀疏的;而当 $m=n\times2.7$ 时,则被认为是能够产生阶段跳跃的临界状态[28].

Table 2 Comparison between AB and AWS

表 2 算法 AB 和 AWS 的比较

Algorithm		*n*		
		60	90	120
AB	Ratio (%)	13	0	0
	Cycles	917.4	—	—
AWS	Ratio (%)	100	100	100
	Cycles	59.4	70.1	106.4

Table 3 Comparison between DB and AWS ($m=n\times2$)

表 3 算法 DB 和 AWS 的比较($m=n\times2$)

Algorithm		*n*		
		90	120	150
DB	Ratio (%)	100	100	100
	Cycles	150.8	210.1	278.8
AWS	Ratio (%)	100	100	100
	Cycles	70.1	106.4	159.2

Table 4 Comparison between DB and AWS ($m=n\times2.7$)

表 4 算法 DB 和 AWS 的比较($m=n\times2.7$)

Algorithm		*n*		
		90	120	150
DB	Ratio (%)	100	100	100
	Cycles	517.1	866.4	1175.5
AWS	Ratio (%)	97	65	29
	Cycles	1869.6	6428.4	8249.5

从表 3 和表 4 中可以看出:当问题为临界困难时,DB 算法要优于 AWS 算法.而对于一般情形,AWS 算法要优于 DB 算法.因为在 DB 算法中,每个模式(*wait_ok*?或者 *wait_improve*)都需要一个周期,所以每个 Agent 在两个周期内至多只能改变一次赋值;而在 AWS 算法中,每个 Agent 在每个周期内都可以改变赋值.

如前所述,近年来的很多工作都对这些基本算法进行了改进或者扩展,在性能和效率方面也有各自的特点.

文献[20]中的实验表明:对 Agent 进行动态重排序的 ABTR(ABT with asynchronous reordering)算法的平均性能要优于 AB 算法,说明额外增加动态重排序的启发式消息实际上是可以提高算法效率的.文献[24]通过实验表明:采用值聚集的 AAS(asynchronous aggregation search)算法的效果要稍好一些.虽然在查找第一个解时,不利用值聚集的效果会更好,但是如果解不存在,那么 AAS 的性能总是要优于不采用值聚集的算法,因为此时需要扩展到整个搜索空间,AAS 可以减少消息序列,也就是减少存储 *nogoods* 消息的数目.更进一步地,使用了 bound-consistency 一致维护技术的 MHDC(maintaining hierarchical distributed consistency)算法在实验中的整体性能要比 AAS 有很大的提高.

通过图染色实验,文献[25]比较了算法 Multi-AWS,AWS-AP(Agent priority)和 Single-AWS,对于 Agent 有多个变量的情况,Multi-AWS 算法在执行周期以及一致性检查数目上都要优于其他两种算法,而且随着 Agent 或变量的增多,其性能就越优于 AWS-AP 算法和 Single-AWS 算法.这是因为在解决 Agent 有多个变量的问题时,AWS-AP 算法和 Single-AWS 算法需要增加额外的虚拟 Agents 来将多局部变量分布化,这样就增加了 Agents 间的通信,从而使性能降低.

文献[27]首先比较了 Multi-DB 和 Multi-AWS 算法.Multi-DB 算法随着变量数目的增多,效率会越来越优于 Multi-AWS 算法,在很多情形下会有至少 1 个数量级的提高,但是成功率却较低.原因是由于 Multi-DB 算法在查找过程中固有的确定性使算法缺少了随机性.对于实验中 Multi-DB^{++}算法的效率优于 Multi-DB 算法,可以得出结论:多 Agent 搜索进程中的停滞可以通过添加随机行走来避免.

3 分布式约束满足问题的扩展

在很多工作都关注于分布式约束满足问题求解算法的同时,也有不少工作对分布式约束满足问题进行了扩展,比如分布式局部约束满足[29,30]、开放式约束求解[31]、分布式求解中的隐私安全[32,33]等等.

很多多主体系统的应用被形式化为分布式约束满足问题进行求解时,往往是过约束的,也就是问题实例的约束太多,以至于不存在一个完全满足所有约束的解.这种情况下,用上述算法进行求解就无法获得有用信息,如果算法完全,那么就找不到解;如果算法不完全,那么就会无限循环下去.这时候,我们宁愿得到一个满足尽可能多约束的不完全的解.解决过约束的分布式约束满足问题,被称为分布式局部约束满足(distributed partial CSPs).在分布式局部约束满足中,众多 Agent 试图通过放宽原始问题的约束条件来寻找一个可解的分布式约束满足问题及其解.原问题能够被放宽多少,由全局距离函数来衡量,当然越接近原问题越好.分布式局部约束满足分为两类:一类称为 distributed maximal CSPs[29],每个 Agent 试图寻找能够最小化冲突约束的最大数目的变量赋值;另一类称为 distributed hierarchical CSPs[30],每个 Agent 试图寻找能够最小化冲突约束的最大约束重要程度的变量赋值,即尽量满足更重要的约束关系.分布式局部约束满足的解,是可解的分布式约束满足问题与其解的组合,其中原问题与可解问题之间的全局距离函数要小于某个阈值.

同样,随着互联网络的发展,当很多应用被形式化为分布式约束满足问题时,问题的值域和约束条件有可能是从网络中不同的资源发现的.对这种情形建模时,文献[31]定义了开放式约束满足问题 OCSP(open CSP),其中的值域和约束条件是通过网络被逐渐发现的.在变量值域和约束关系以非降序出现这样一个假设情况下,OCSP 能够在变量值域的信息不完整的情况下进行求解.该算法的关键是用有效的方法来识别最小的不可解子问题,从而高效地进行信息收集,减少网络阻塞和服务器的载荷.

另外,保密性也是与一些多主体系统的应用息息相关的问题.如在会议日程安排或者电子商务中,Agent 可能不希望把问题解决中用到的信息泄漏出去.已有的分布式约束满足算法在搜索过程中会泄露一些信息,并没有去考虑处理隐私安全问题.文献[32]中提出了一个安全性的分布式约束满足算法,利用了不能识别的、同形的和随机的公钥加密方案.在该算法中,多个服务器从 Agent 接收加密的信息,然后协调地执行标准的回溯搜索过程,无论是 Agent 还是服务器,都不能获得属于其他 Agent 的变量赋值外的任何信息.因该算法的性能等同于标准的回溯算法,因而也具有相当的复杂性.对于这种情形,维护隐私安全和提高搜索效率之间就存在一个潜在的折衷.文献[33]表明,如何定量评估在分布式约束满足的求解中可以损失多少隐私,还指出了 Agent 如何能够从不需要其他隐私信息的通信中推断其他 Agent 的问题或子问题,这可以利用基于约束的推理来完成.这种对分布式约束满足中评估隐私丢失的扩展,可以让 Agent 在一些信息未知的条件下来动作以提高问题解决的效率.

4 分布式约束满足问题算法的应用

分布式约束满足特别适用于 Agent 间需要协作的问题.因为约束是可以很好地表示 Agent 间行为依赖性的一种形式,而且分布式约束满足算法的结构就是用来局部化 Agent 间的通信的,所以,很多以协作性为主的多主体系统中的应用问题都可以形式化为分布式约束满足问题.比如,多主体真值维护系统[34]中有多个 Agent,每个 Agent 都有自己的真值维护系统,都有不知是可信还是不可信的不确定数据,这其中也有一些数据是与其他 Agent 共享的,每个 Agent 必须保证数据的标记是一致的,共享数据也必须是相同标记.视那些不确定的数据为变量,视依赖关系为约束,就将真值维护系统形式化为分布式约束满足问题.

分布式约束满足的另一个应用领域是由多个单一相同 Agent 构成的网络,比如分布式传感网络.为了解决分布式传感网络中的有限资源和实时需求的限制,使用资源低开销和高实时性的分布式算法是相当必要的[35],因此,可将这类问题映射成分布式约束满足问题进行求解.

如图 3 为一个分布式传感网络的应用,为了检测到交通工具的方位,多个传感器必须一起来跟踪目标.如图 3 所示,每个传感器装配 3 个雷达头,雷达头的覆盖范围为 120°,并且一次只有一个雷达头处于活动状态.每个 Agent 控制感应方向,也就是相应的变量取值,3 个不同的 Agent 分配给一个目标,并且当每个 Agent 的感应方向正确时才能准确追踪目标的轨迹,每个传感器只能与最近的邻居互相通信.

另一类可形式化为分布式约束满足的问题是资源分配.资源分配中将任务或资源分配给 Agent,资源的有限性导致 Agent 间存在约束关系.在形式化为分布式约束满足问题时,将每个任务或资源看作变量,对任务或资源相应的指派看作赋值,它们彼此间的制约就是约束.文献[36]中给出了形式化的分布式资源配置问题,突出了问题的动态性和分布性,并提出了一般性的利用分布式约束满足技术的求解策略,Agent 利用有限的能力协同地分配共享资源.这种情况下,任务 Agent 不知道与自己的变量或任务相关联的约束关系,所以,任务 Agent 必须与约束 Agent 进行通信来确定相关的约束是否被满足.

图 4 是一个分布式通信网络中的资源分配问题,有一些通过通信线路相互连接的通信地点,这些地点因为地理原因分成几个组,每个组由一个 Agent 控制.这些 Agent 为连接要求协作地分配通信连接,使得所有要求能够同时得到满足.这个问题可以形式化为分布式约束满足问题,一个 Agent 包含多个变量表示需求,变量的值域是可能的线路分配计划,目标是在资源约束的条件下找到一个满足所有要求的计划组合.

还有很多其他寻找 Agent 间行动决策的一致组合的问题,比如值班表安排[37]、分布式日程安排[38,39]等,都可以形式化为分布式约束满足问题来求解.

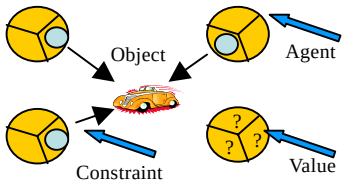

Fig.3 Distributed sensor networks

图 3 分布式传感网络

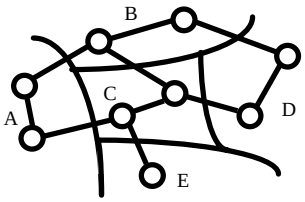

Fig.4 Distributed resource allocation problem

图 4 分布式资源配置问题

5 总 结

当多主体处于共享环境中时,这些主体的可能行为间通常存在着约束关系.分布式约束满足问题就是要找到满足这些主体间约束的一致行为组合.多主体系统中的多种应用问题都可以形式化为分布式约束满足问题进行求解.本文全面评述了关于分布式约束满足的现有研究,介绍了分布式约束满足问题的定义、算法,对分布式约束满足问题的扩展和应用也做了探讨.

自分布式约束满足问题提出以来,相当多的研究工作都着重于分布式约束满足的算法设计和性能度量等方面,所发表的文章也都围绕着查找算法应该同步还是异步、使用标准约束满足算法的一些策略是否可以加速分布式搜索以及如何才能更好地优化和度量算法性能等理论问题.另一方面,将分布式约束满足的算法应用到实际问题求解也得到了广泛的理论研究,国内的研究更着重于此方面,并取得了一些初步成果.

近年来,随着分布式约束满足问题研究取得的进展.算法的有效性允许人们更多地关注于该问题的实际应用,但由此也产生了更多的问题,如需要更好的算法来适应分布性从而减少通信需求和开销;当评估约束的代价太高时需要最小化约束评估的数量;根据实际要求动态增减计算实体和约束;平衡隐私安全性和算法效率,在两者间取得最佳折衷等等.这也是我们以后工作的方向.这些问题的有效解决,可以更好地应用于求解现实问题,也为前瞻性的应用提供了理论基础.

References:

[1] Montanari U. Networks of constraints: Fundamental properties and applications to picture processing. Information Sciences, 1974, 7(2):95−132.

[2] Kumar V. Algorithms for constraint satisfaction problems: A survey. AI Magazine, 1992,13(1):32−44.

[3] Sun JG, Jing SY. Solving non-binary constraint satisfaction problem. Chinese Journal of Computers, 2003,26(12):1746−1752 (in Chinese with English abstract).

[4] Sun JG, Zhang YG. Model generation and constraint programming. In: Lu RQ, ed. The Knowledge Science and Computation Science. Beijing: Tsinghua University Press, 2003.197−232 (in Chinese with English abstract).

[5] Zhang XZ, Liu CN. Synchronization of the inference engine with the constraint solver in a CLP system. Journal of Software, 1996,7(7):415−421 (in Chinese with English abstract).

[6] Liu CN. Constraint logic programming—current status and future. In: Lu RQ, ed. The Knowledge Engineering and Knowledge Science Faced to the 21st Century. Beijing: Tsinghua University Press, 2001. 251−279 (in Chinese with English abstract).

[7] Liu JM, Han J, Tang YY. Multi-Agent oriented constraint satisfaction. Artificial Intelligence, 2002,136(1):101−144.

[8] Han J, Chen EH, Cai QS. Multi-Level strategy for maintaining arc consistency in problem solving and its implementation. Journal of Software, 1998,9(8):622−627 (in Chinese with English abstract).

[9] Chen EH, Zhang ZY, Wang XF. A fast recognition algorithm of CRC constraint networks. Journal of Software, 2002,13(5):972−979 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/972.htm

[10] Faltings B, Yokoo M. Introduction: Special issue on distributed constraint satisfaction. Artificial Intelligence, 2005,161(1-2):1−5.

[11] Yokoo M, Durfee EH, Ishida T, Kuwabara K. Distributed constraint satisfaction for formalizing distributed problem solving. In: 12th Int'l Conf. on Distributed Computing Systems (ICDCS'92). Los Alamitos: IEEE Computer Society Press, 1992. 614−621. http://citeseer.ist.psu.edu/yokoo92distributed.html

[12] Yokoo M, Hirayama K. Algorithms for distributed constraint satisfaction: A review. Autonomous Agents and Multi-Agent Systems, 2000,3(2):185−207.

[13] Trevisan L. Approximating satisfiable satisfiability problems. In: Burkard RE, Woeginger GJ, eds. Proc. of the 5th Annual European Symp. on Algorithms. Berlin: Springer-Verlag, 1997. 472−485.

[14] Calisti M, Neagu N. Constraint satisfaction techniques and software agents. In: Proc. of the AIIA 2004 Workshop on Agents and Constraints. Perugia, 2004. http://www.whitestein.com/pages/downloads/publications.html

[15] Collin Z, Dechter R, Katz S. On the feasibility of distributed constraint satisfaction. In: Mylopoulos J, Reiter R, eds. Proc. of the 12th Int'l Joint Conf. on Artificial Intelligence. San Fransisco: Morgan Kaufmann Publishers, 1991. 318−324.

[16] Zhang Y, Mackworth A. Parallel and distributed algorithms for finite constraint satisfaction problems. In: Sahni S, ed. Proc. of the 3rd IEEE Symp. on Parallel and Distributed Processing. Los Alamitos: IEEE Computer Society Press, 1991. 394−397.

[17] Yokoo M. Asynchronous weak-commitment search for solving distributed constraint satisfaction problems. In: Montanari U, Rossi F, eds. Proc. of the 1st Int'l Conf. on Principles and Practice of Constraint Programming (CP'95). Berlin: Springer-Verlag, 1995. 88−102.

[18] Yokoo M, Durfee EH, Ishida T, Kuwabara K. The distributed constraint satisfaction problem: Formalization and algorithms. IEEE Trans. on Knowledge Data Engineering, 1998,10(5):673−685.

[19] Yokoo M, Hirayama K. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In: Weiß G, ed. Proc. of the 2nd Int'l Conf. on Multiagent Systems (ICMAS'96). Cambridge: MIT Press, 1996. 401−408.

[20] Silaghi MC, Sam-Haroud D, Faltings B. Hybridizing ABT and AWC into a polynomial space, complete protocol with reordering. Technical Report, No.364, Lausanne: Swiss Federal Institute of Technology (EPFL), 2001.

[21] Silaghi MC, Sam-Haroud D, Faltings B. Consistency maintenance for ABT. In: Walsh T, ed. Proc. of the 7th Int'l Conf. on Principles and Practice of Constraint Programming (CP 2001). Berlin: Springer-Verlag, 2001. 271−285.

[22] Hamadi Y, Bessière C, Quinqueton J. Backtracking in distributed constraint networks. In: Prade H, ed. Proc. of the 13th European Conf. on Artificial Intelligence (ECAI'98). Chichester: John Wiley and Sons, 1998. 219−223.

[23] Bessière C, Maestre A, Brito I, Meseguer P. Asynchronous backtracking without adding links: A new member in the ABT family. Artificial Intelligence, 2005,161(1-2):7−24.

[24] Silaghi M, Faltings B. Asynchronous aggregation and consistency in distributed constraint satisfaction. Artificial Intelligence, 2005, 161(1-2):25−53.

[25] Yokoo M, Hirayama K. Distributed constraint satisfaction algorithm for complex local problems. In: Demazeau Y, ed. Proc. of the 3rd Int'l Conf. on Multi-Agent System. Los Alamitos: IEEE Computer Society Press, 1998. 372−379.

[26] Morris P. The breakout method for escaping from local minima. In: Fikes R, Lehnert W, eds. Proc. of the 11th National Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 1993. 40−45.

[27] Hirayama K, Yokoo M. The distributed breakout algorithms. Artificial Intelligence, 2005,161(1-2):89−115.

[28] Cheeseman P, Kanefsky B, Taylor W. Where the really hard problems are. In: Mylopoulos J, Reiter R, eds. Proc. of the 12th Int'l Joint Conf. on Artificial Intelligence. San Fransisco: Morgan Kaufmann Publishers, 1991. 331−337.

[29] Hirayama K, Yokoo M. Distributed partial constraint satisfaction problem. In: Smolka G, ed. Proc. of the 3rd Int'l Conf. on Principles and Practice of Constraint Programming (CP'97). Berlin: Springer-Verlag, 1997. 222−236.

[30] Yokoo M. Constraint relaxation in distributed constraint satisfaction problem. In: IEEE, ed. Proc. of the 5th Int'l Conf. on Tools with Artificial Intelligence. Los Alamitos: IEEE Computer Society Press, 1993. 56−63.

[31] Faltings B, Macho-Gonzalez S. Open constraint programming. Artificial Intelligence, 2005,161(1-2):181−208.

[32] Yokoo M, Suzuki K, Hirayama K. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. Artificial Intelligence, 2005,161(1-2):229−245.

[33] Wallace RJ, Freuder EC. Constraint-Based reasoning and privacy/efficiency tradeoffs in multi-agent problem solving. Artificial Intelligence, 2005,161(1-2):209−227.

[34] Huhns MN, Bridgeland DM. Multiagent truth maintenance. IEEE Trans. on Systems, Man and Cybernetics, 1991,21(6): 1437−1445.

[35] Zhang W, Wang G, Xing Z, Wittenburg L. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. Artificial Intelligence, 2005,161(1-2):55−87.

[36] Modi PJ, Jung H, Tambe M, Shen WM, Kulkarni S. A dynamic distributed constraint satisfaction approach to resource allocation. In: Walsh T, ed. Proc. of the 7th Int'l Conf. on Principles and Practice of Constraint Programming (CP 2001). Berlin: Springer-Verlag, 2001. 685−700.

[37] Solotorevsky G, Gudes E. Solving a real-life nurses time tabling and transportation problem using distributed CSP techniques. In: Proc. of the CP'96 Workshop on Constraint Programming Applications. 1996. 123−131. http://citeseer.ist.psu.edu/65211.html

[38] Liu JS, Sycara KP, Multiagent coordination in tightly coupled task scheduling. In: Tokoro M, ed. Proc. of the 2nd Int'l Conf. on Multi-Agent Systems. Menlo Park: AAAI Press, 1996. 181−188.

[39] Sycara KP, Roth S, Sadeh N, Fox MS. Distributed constrained heuristic search. IEEE Trans. on Systems, Man and Cybernetics, 1991,21(6):1446−1461.

附中文参考文献:

[3] 孙吉贵,景沈艳.非二元约束满足问题求解.计算机学报,2003,26(12):1746−1752.

[4] 孙吉贵,张永刚.模型生成与约束求解.见:陆汝钤,主编.知识科学与计算科学.北京:清华大学出版社,2003.197−232.

[5] 张秀珍,刘椿年.CLP 系统中推理机与约束求解器的协调技术.软件学报,1996,7(7):415−421.

[6] 刘椿年,约束逻辑程序设计 CLP——现状与未来.见:陆汝钤,主编.世纪之交的知识工程与知识科学.北京:清华大学出版社,2001. 251−279.

[8] 韩靖,陈恩红,蔡庆生.求解过程中约束一致性维护的多层次策略研究.软件学报,1998,9(8):622−627.

[9] 陈恩红,张振亚,王煦法.相接行凸约束网络的快速识别算法.软件学报,2002,13(5):972−979. http://www.jos.org.cn/1000-9825/13/972.htm

王秦辉(1977－),男,浙江绍兴人,博士生,主要研究领域为约束满足问题,知识发现.

王煦法(1948－),男,教授,博士生导师,CCF 高级会员,主要研究领域为智能信息处理.

陈恩红(1968－),男,博士,教授,博士生导师,主要研究领域为约束满足问题,知识发现,机器学习.