一种动态约束优化调度的软件模型

姜大光　易军凯
(北京化工大学 信息科学与技术学院, 北京　100029)

摘　要: 基于静态约束调度及其组件结构,建立了一种动态约束优化调度的软件模型结构,可以适应于调度环境的动态变化。从软件模型上分析,动态约束优化调度的组件包括动态解析器、动态分配器和推理引擎。动态解析器规范了变元和约束条件,动态分配器解决了约束条件的重新分配和冲突,推理引擎实现了快速搜索。实验表明该模型可以满足实际动态逻辑约束问题的求解,并且把传统优化调度的封闭形式扩展为开放状态。

关键词: 动态约束优化调度; 动态解析器; 动态分配器; 推理引擎
中图分类号: TP391.41

引　言

约束调度是集成制造的主要研究方向之一,已经发展了多种约束优化调度语言和知识系统,环境动态变化使得这些工具的应用受到限制。Dechter 等人在 1988 年首次在全局搜索理论中描述了动态约束优化调度[1],动态调度的特点在于需要处理由动态环境带来的变元和约束条件[2]。Purvis 提出了采用局部满足的过滤器来实现动态调度[3]。Wallace 与 Freuder 采用维护优化结果的策略,一旦当前答案不满足新的约束就对优化结果进行修正[4]。Law 等人建议利用定义好的规则对新的结果尽量靠近上一次得到的结果[5]。Smith 等人和 Hurst 等人则提出重新利用上个结果做局部修改来产生新的结果[6-7]。所有这些方法在理论上做了很深入的讨论,在以上研究基础上,本文从系统软件模型结构方面来讨论动态约束优化调度的软件模型实现方式。

1　动态约束优化调度软件模型结构

1.1　动态约束优化调度模型要求分析

静态约束优化调度的核心是约束推理机,其搜索可以看作是在一棵树上进行,树上的节点分为 3 种:中间节点、错误节点和叶子(正确节点)[8]。

从静态约束优化调度设计思想出发,考虑以下几个方面的策略构造出动态约束优化调度软件模型。

解析策略　外部动态环境产生的变元和约束条件需要实时记录并解析进入约束优化调度系统。采用人机交互的方式来设置变元及其逻辑约束关系。如果能预知环境动态的变化,可以事先进行设置实现自适应解析。

分配策略　新变元和逻辑约束关系在解析进入搜索引擎前,需要和原来的变元和逻辑约束关系融合,重新分配。

内存处理策略　当搜索遇到错误节点时需回溯上个节点,恢复原来状态,这样就需对中间节点进行保存。大部分约束优化调度系统采用痕迹方法保留中间节点,还有一些系统采用全部拷贝和重新计算等内存处理方法。

搜索策略　树上节点的搜索,需要有次序地进行,即按照搜索策略进行搜索。最常用的搜索策略是深度优先,还有宽度优先,迭代深入等方法。

分支策略　这种策略能够对当前节点进行分支,构造出下一个需要搜索的节点。一般来说有自然分支法和最小有限域分支法。对最优节点的搜索,通常采用分支界定法。

1.2　动态约束优化调度模型结构分析

动态约束优化调度软件模型包括 3 个组件:动态解析器、动态分配器和推理引擎,组件结构框架如图 1 所示。环境变化产生的变元和约束逻辑关系经人机交互进入动态解析器,解析成为规范的变元集合和约束过滤器。规范格式后的变元和过滤器由动态分配器重新组合,形成新的变元和过滤器,送入推

收稿日期: 2008-04-29
基金项目: 国家"十一五"科技支撑计划(2006BAK31B04)
第一作者: 男,1971 年生,讲师
E-mail: jiangdg@mail.buct.edu.cn

理引擎进行推理，得到最新结果。

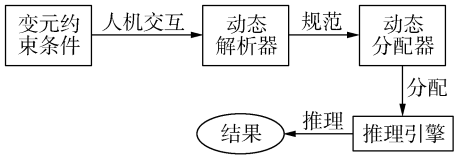

图 1　动态约束优化调度的软件模型结构

Fig. 1　Software model structure for dynamic constraint optimization scheduling

2　动态约束优化调度组件分析

2.1　动态解析器

实时接收动态的变元和约束条件，包括问题进行的初始和中间过程。一般来说，动态调度不能预测变元和约束条件突然发生变化，但是能实时记录这些变化。动态解析器就是要记录它们的变化，以规定的统一格式进行规范化。这里定义规范的动态变元集合 $V'=\{v'_1,v'_2,\cdots,v'_k\}$ 和动态逻辑约束关系集合 $C'=\{c'_1,c'_2,\cdots,c'_l\}$。动态解析器的工作过程如图 2 所示。

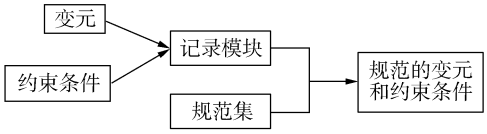

图 2　动态解析器工作过程

Fig. 2　Structure of the dynamic analyzer

2.2　动态分配器

动态分配器需要对新的变元和约束条件进行重新整合和分配。由于动态环境产生的变元和约束可以与已有变元和约束发生关系，这就需要重新确定变元和约束。如何确定整合后的变元 V^N 和约束 C^N 集合是动态分配器的主要任务。定义动态变化前的变元集合和逻辑约束关系集合分别为 $V=\{v_1,v_2,\cdots,v_m\}$ 和 $C=\{c_1,c_2,\cdots,c_n\}$，又以 $f(\theta)$ 表示约束关系涉及的变元中有 θ。可对动态解析器产生的变元集合进行分类。

$$V'_1=\{v'_1\in V'\mid\theta\neq v'_i,\theta\in V\}$$
$$V'_2=\{v'_i\in V'\mid\exists\,\theta=v'_i,\theta\in V\}\qquad(1)$$

对于 V'_2，有

$$V''_2=\{v''_i=v'_i\cap\theta\mid v'_i\in V'_2,\theta=v'_i,\theta\in V\}\qquad(2)$$

对 V 中其他和 V' 没有关系的变元在 C 和 C' 中对应的约束有

$$C'_1=\{c'_i=f(\theta)\mid c'_i\in C',\theta\notin V'\}\cup\{c_i=f(\theta)\mid c_i\in C,\theta\notin V'\}$$

集合 C'_1 可以分为两类：已经完成的和没有完成的约束，表示为

$$C'_1=C'_{11}\cup C'_{12},C'_{11}=\{c_i\mid c_i\in C'_1,c_i\uparrow\},C'_{12}=\{c_i\mid c_i\in C_1,c_i\downarrow\}$$

其中 $\uparrow$ 和 $\downarrow$ 表示为该约束完成和没有完成。在没有完成的约束中涉及的变元是

$$V'_3=\{v_j\notin V'\mid\exists\,c_i=f(v_i),c_i\in C'_{12}\}\qquad(3)$$

综合(1)、(2)和(3)式得到

$$V^N=V'_1\cup V''_2\cup V'_3$$

根据同样分类可以对动态解析器产生的约束集合和已有的约束集合进行融合得到新的约束集合 $C^N=C_1^N\cup C_2^N$，其中 $C_1^N=C'$，$C_2^N=\{c_i=f(\theta)\mid c_i\in C,\theta\in V^N\}$。动态分配器的工作过程如图 3 所示。

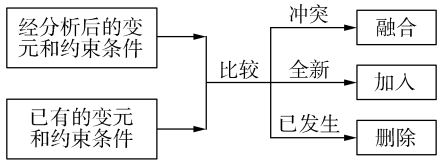

图 3　动态分配器工作过程

Fig. 3　Structure of the dynamic allocator

2.3　推理引擎

推理引擎借鉴了静态约束优化调度中的约束推理机，由临时容器、变元、约束传播器和约束过滤器组成。动态分配器处理过的变元和约束条件传送到推理引擎，推理引擎在选定节点回溯算法、分支策略和节点探索方向后开始工作，直到完成搜索得到答案。推理引擎搜索策略如图 4 所示。

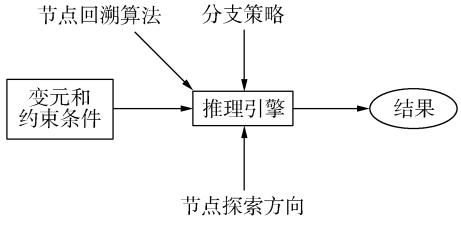

图 4　推理引擎搜索策略图

Fig. 4　Strategy chart for the inference search engine

推理引擎是动态约束优化调度的核心，按照组件设计思想，有 4 个组件：临时容器、变元组、约束传播器和约束过滤器。图 5 表示出了推理引擎组件结构及其相互间影响。

变元组是为描述问题而抽取出的，最后又需要得到答案的不可知变元。每个变元都有自己的离散范围，称为有限域。约束过滤器可以压缩变元有限域，对树进行剪枝。约束传播器对约束过滤器进行

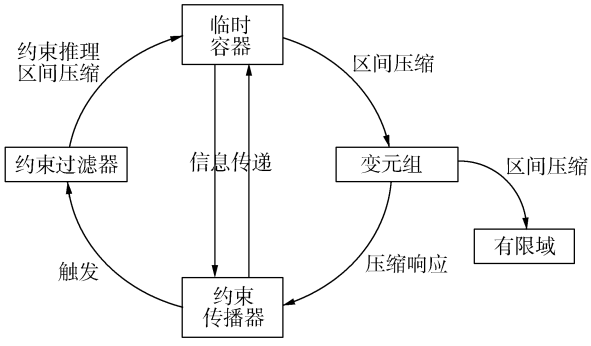

图 5 推理引擎组件结构及其相互影响图

Fig. 5 Flow chart of the inference engine structure and its internal relations

管理,包括约束加入、删除和触发等动作。临时容器可以和其他 3 组件进行信息沟通,是中间媒介。当某变元有限域发生区间压缩时,与该变元有关的约束约束过滤器加入约束传播器传播行列,约束传播器触发约束过滤器,压缩有关变元,同时新的变元压缩可能引发其它过滤器的加入。在搜索策略的驱动下,变元压缩信息在约束传播器和临时容器之间发生传递,引发下一次约束传播。搜索在循环进行,把 4 个组件协作成统一体。

3 软件模型实际应用结果及分析

引入变量事件后,搜索过程激发约束过滤器次数明显减少。软件模型实际应用中,约束逻辑程序设计的计算机平台及软件选择:主频 1.8 G;内存 512 M;操作系统 Redhat Linux 7.2;约束优化系统 FLY 0.1.0 版本。

实验验证,FLY 在实际应用中其求解速度优于大多数约束优化调度系统。为便于和静态系统进行对比,该文采用经典问题做约束优化调度求解,并设定环境变元和约束条件改变 2 次,静态情况为重新计算,实验结果如表 1 所示,其中运行时间包括了屏幕输出时间,实际求解时间可能更少。这种动态约束优化调度的搜索速度有很大的提高,减少了不必要的约束过滤器激发次数。

表 1 静态和动态模型实验结果对比

Table 1 Comparison tests between static and dynamic models

样例	运行时间/s	
	静态	动态
Nqueen(100)	0.53	0.45
Magic_square(5)	0.15	0.14
Hamilton	22.13	15.26
Photo	26.05	18.46
Robin	12.47	5.75
Larry	3.98	2.54
Bridge	1.14	0.67

环境变化次数(2 次)

4 结束语

动态约束优化调度问题广泛存在于实际的调度和排程中,但是实际应用系统还不是很多,主要原因在于优化调度实际就是 NP 完全问题,而且变元和约束又处于动态环境中。本文讨论的该动态约束优化调度软件模型已经实现了部分优化调度问题,该模型建立在组件基础上,由组件搭建而成,包含 3 个主要组件:动态解析器、动态分配器和推理引擎,解决了变元和约束条件的规范化以及约束条件的重新分配和冲突,利用启发式和一致性理论实现了快速搜索。组件的设计也采用子组件方式,各个组件相互作用,形成一个协调的统一体。在实际的动态优化调度软件模型研究过程中,组件的重复利用得到了充分发挥。

参考文献:

[1] Dechter R, Dechter A. Belief maintenance in dynamic constraint networks[C]// In Proceedings of 6th National Conference on Artificial Intelligence, 1988: 37-42.

[2] 翟桥柱, 管晓宏, 郭燕, 等. 具有混合动态约束的生产系统优化调度新算法[J]. 自动化学报, 2004, 30(4):539-546.

[3] Purvis L. Dynamic constraint satisfaction using case-based reasoning techniques[C/OL]// International Conference on Constraint Programming, 1997 [2008-04-20]. http://home.rochester.rr.com/thepurvises/CP97.ps.

[4] Wallace R J, Freuder E C. Stable solutions for dynamic constraint satisfaction problems[C/OL]// International Workshop on Constraint Solving and Constraint Logic Programming, 2003 [2008-04-20]. http://citeseer.comp.nus.edu.sg/cache/papers/cs/12284/ftp:zSzzSzftp.cs.unh.eduzSzpubzSzcspzSzPaperszSzcp98-stable-rjw-ecf.pdf/stable-solutions-for-dynamic.pdf.

[5] Law Y C, Lee J H M, Smith B M. Automatic generation of redundant models for permutation constraint satisfaction problems[J]. Constraints, 2007, 12(4):469-505.

[6] Smith B M, Bistarelli S, O'Sullivan B. Constraint symmetry for the soft CSP[C/OL]// Proceedings of the International Symmetry Conference, 2007[2008-04-20].

http://4c.ucc.ie/web/upload/publications/inProc/softsymms.pdf.

[7] Hurst N, Marriott K, Moulder P. Dynamic approximation of complex graphical constraints by linear constraints [C/OL]// Proceedings of the ACM Symposium, 2002 [2008-04-22]. http://www.csse.monash.edu.au/~marriott/HurMarMou03.pdf.

[8] Henz M, Müller T, Boon N K. Figaro: Yet Another Constraint Programming Library[C/OL]// Workshop on Parallelism and Implementation Technology for Constraint Logic Programming at ICLP, 1999[2008-05-06]. http://citeseer.comp.nus.edu.sg/cache/papers/cs/12564/http:zSzzSzwww.comp.nus.edu.sgzSz~henzzSzpublicationszSzpszSzfigaro.pdf/figaro-yet-another-constraint.pdf.

A software model for dynamic constraint optimization scheduling

JIANG DaGuang　YI JunKai

(College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)

Abstract: This paper describes the construction of a software model for dynamic constraint optimization scheduling, based on the framework of static constraint optimization scheduling, which can adapt to dynamic changes in the environment. The components of the software model include a dynamic generator, a dynamic allocator and an inference engine. The dynamic generator can normalize variables and constraints, the dynamic allocator can deal with constraint reallocation and conflicts and the inference engine can realize fast searching. The model, which changes the development status from closed to open, can effectively solve practical dynamic constraint problems.

Key words: dynamic constraint optimization; dynamic generator; dynamic allocator; inference engine