半在线调度中约束求解算法研究

张居阳 [1,2] 孙吉贵 [1,2] 杨轻云 [1,2]

摘 要 很多实际调度问题是半在线的. 尝试运用人工智能方法来求解半在线调度问题, 首先简要介绍了半在线调度问题并对其约束模型进行了分类, 通过引入单调性约束扩展的相关概念, 从约束建模角度形式化描述了一类动态约束扩展, 并在此基础上设计了一个完备动态约束求解算法, 最后给出该算法在半在线离散资源约束调度求解的应用算例. 测试结果表明, 该算法是可行有效的.

关键词 半在线调度, 约束求解, 单调性

中图分类号 TP301.6

A Constraint Solving Algorithm for Semi On-line Scheduling Problem

ZHANG Ju-Yang[1,2] SUN Ji-Gui[1,2] YANG Qing-Yun[1,2]

Abstract Most of real-life scheduling problems are semi on-line. Recently, how to solve such dynamic problems is a hot topic in the research of artificial intelligence. The semi on-line scheduling is introduced and the constraint models are analyzed and categorized. By defining the relevant concept of monotony about constraints that appear in general dynamic constraint models, a kind of constraint extending is formalized. Based on this dynamic constraint modeling, a sound dynamic constraint solving algorithm is designed to deal with the scheduling problems. Finally, an application example of semi on-line discrete resource scheduling problems is given. Experiments show that the algorithm is valid.

Key words Semi on-line scheduling, constraint solving, monotony

1 引言

根据调度者在调度时掌握工件 (任务) 信息的多少, 调度问题分为离线 (off-line) 和在线 (on-line) 两类[1]. 在实际问题中, 出现这两种情况的机会并不多, 大量存在着的问题是调度者一方面不可能知道工件的所有信息, 另一方面可能掌握着比经典在线模型更多的信息或有着更大的调度自由度, 因而使问题变得比离线相对 "困难" 一些, 但比在线相对 "容易" 一些, 即可能得到比在线算法性能更好的算法. 我们把不完全满足在线调度问题的两条基本假设, 难度介于在线和离线之间的模型称为半在线 (semi on-line) 的模型.

半在线概念于 1996 年被提出, 近十年来, 各种求解算法不断涌现 [2~4]. 何勇等在半在线问题研究领域取得了具有国际水平的研究成果[5~7]. 在人工智能领域, 约束程序 (Constraint programming, CP) 对于动态 (在线 / 半在线) 调度问题的关注最初源于配置问题. 约束程序作为一门新兴的软件技术, 是围绕关系约束这一数学概念建立起来的关于程序设计和问题求解的方法论, 是研究基于约束的组合优化问题的推理和计算系统. CP 具有说明特性而且特别适合高效地求解调度和规划领域的大规模约束满足问题. (静态) 约束满足问题 (Constraint satisfaction problem, CSP)[8] 可以清晰描述离线调度问题中的约束关系, 但无法满足动态调度问题的约束建模和求解. 对于动态约束建模, 文献 [9] 提出在线约束满足问题 (On-line CSP) 的概念; 针对复杂过程工业调度, 文献 [10] 引入动态约束满足问题 (Dynamic CSP, DCSP) 模型.

本文工作主要集中在对一般半在线调度问题[1] 中动态约束信息的建模和处理. 我们发现在一般半在线问题中存在一些约束的变化 (变量的增加或删减) 并不会引起当前解集的扩展, 我们称这样的约束和约束扩展具备单调性, 对于此类约束扩展只需对当前解集做局部修改, 而没有必要重新求解. 我们的策略可以更为容易地应用到现有的调度系统中去, 而无需转化成 DCSP 模型才能进行动态求解. 我们设计了一个针对单调约束扩展的求解算法, 并证明该算法对于单调约束求解是完备的. 通过应用算例的测试, 表明模型和算法对于半在线问题建模和求解是有效的.

2 动态约束建模和单调性

定义 1.[8] 约束满足问题 P 为一个三元组 (X, D, C), 其中变量集合 $X = x_1, \cdots, x_n$, 论域集合 $D = D(x_1), \cdots, D(x_n)$ (这里 $D(x_i)$ 是指对于变量 x_i 的可能取值的有限集合), 约束集合 $C = \{c_1, \cdots, c_m\}$. 一个在有序变量集合 X 上的约束 c_i, $X(c_i) = (x_{i1}, \cdots, x_{ik})$ 是变量论域笛卡儿乘积 $D(x_{i1}) \times \cdots \times D(x_{ik})$ 的子集.

假定 $(v_1, \cdots, v_n) \downarrow (x_{i1}, \cdots, x_{ik})$ 表示标准的投影操作, 即一组值 $(v_1, \cdots, v_n)$ 在变量 $(x_{i1}, \cdots, x_{ik})$ 上的投影: $(v_1, \cdots, v_n) \downarrow (x_{i1}, \cdots, x_{ik}) = (v_{i1}, \cdots, v_{ik})$. 假设有完全的变量实例化 $(v_1, \cdots, v_n)$, 如果 $\forall i = 1, \cdots, n, v_i \in D(x_i)$ 且 $j = 1, \cdots, m (v_1, \cdots, v_n) \downarrow X(c_j) \in C$, 则我们称之为有效赋值或有效解, 也就是说, 有效赋值是一个完全的变量实例化, 且满足所有的约束. $Sol(P)$ 是所有有效赋值组的集合, 我们称之为约束满足问题 P 的解集.

定义 2. 一般半在线调度问题 S 是一个六元组 T, R, X, C, D, f, 其中 T 是任务 (工件) 集合, R 是资源集合, X 是时间变量集合 (T 的属性集合), C 是约束集合, D 是变量 X 的论域集, f 为目标函数.

定义 3. 问题 $P' = (X', D', C')$ 被称为 $P = (X, D, C)$ 的扩展, 如果 $X \subseteq X'$, $C \subseteq C'$ 并且 $\forall x \in X, D(x) = D'(x)$. 也就是说, 如果向原始问题 P 中添加了新的变量和约束, 那么得到一个扩展问题 P'.

定义 4. 约束扩展是单调的, 如果 P' 是 P 扩展, 那么 $Sol(P') \downarrow X \subseteq Sol(P)$, 这里 X 是问题 P 的变量集合, 也就是说, 扩展问题 P' 的每一个解都是原始问题 P 的解.

$C(X)$ 表示一任意有限有序变量集合上的约束的组合. 这样的约束组合构成了约束满足问题的子问题. 注意, $C(x_1, \cdots, x_k)$ 中的约束所包含的变量并不止 $x_1, \cdots, x_k$, 那些变量可以是局部的或隐藏的. 对于一个在变量集合 X 上的 $C(X)$, 若 $X \subseteq Y$, 则称 $C(Y)$ 是其扩展.

收稿日期 2005-10-12 收修改稿日期 2006-8-23
Received October 12, 2005; in revised form August 23, 2006
国家自然科学基金 (60473003), 教育部 "新世纪优秀人才支持计划", 吉林省杰出青年基金 (20030107) 资助
Supported by National Natural Science Foundation of P. R. China (60473003), Program for New Century Excellent Talents in University, the Outstanding Youth Foundation of Jilin Province of China (20030107)
1. 吉林大学计算机科学与技术学院 长春 130012 2. 符号计算与知识工程教育部重点实验室 长春 130012
1. College of Computer Science and Technology, Jilin University, Changchun 130012 2. Key Laboratry of Symbolic Computation and Knowledge Engineering of Ministry of Education, Changchun 130012
DOI: 10.1360/aas-007-0765

[1] 一般指包括文献 [2, 3] 中所划分的第一、二、三类半在线调度, 而不具体指其中某一类.

定义 5. 设 $Sol(C(X), D(X))$ 是约束 $C(X)$ 和变量集合 X 的解集, 约束传播策略是单调的, 如果对于任意 X 和 Y 的变量集合 $Sol(C(X \cup Y), D(X \cup Y)) \downarrow X \subseteq Sol(C(X), D(X))$.

直观地说, 约束扩展的单调性意味着扩展一个约束, 即向约束中添加新变量, 但并不扩展解集. 约束传播策略的单调性是指传播算法无论开始于较大的变量论域或较小的变量论域都不会对它的传播过程造成影响. 即论域的削减程度是等效的.

3 通用动态约束求解算法

在半在线问题中, 不是所有的工件信息 (时间变量 X 及 $C(X)$) 都是预知的, 新工件是在调度过程中动态生成 (引入) 的, 即原始约束满足问题 P 在不断地被扩展. 因此, 传播约束过程只能处理当前所知的变量集 X 上的约束 $C(X)$. 当原始问题 P 被扩展为 P', 且这种扩展是单调扩展, 那么我们只需要在已知的变量集 X 上先进行约束传播, 等到新的约束变量集 Y 生成后, 触发新约束传播而解除旧约束传播, 这样的策略是单调的, 即 $Sol(C(X \cup Y), D(X \cup Y)) \downarrow X \subseteq Sol(C(X), D(X))$, 这就可以保证没有可行解被删除. 这是通用动态单调约束求解算法的基本策略. 下面详细描述 Dyna-Solver 算法及其完备性.

Dyna-Solver 算法是一个动态约束求解算法, 它是静态约束求解算法 CB-Solver[11] 的改进. 算法输入为扩展约束集 $C(X \cup Y)$, 扩展论域集 $D(X \cup Y)$ 和一个在论域 $D'(X \cup Y)$ 上的新的变量集 Y, 解表示为 $Sol(C(X \cup Y), D(X \cup Y))$. 算法利用单调约束扩展的性质, 从先前的解 (论域集合) 出发再得到扩展后约束的解 (限制论域). 算法首先停止当前传播算法, 注意这并不是说把当前约束从约束槽中删除, 只是停止传播 (内部数据结构保存), 然后, 在堆栈中保存变量论域, 最后, 算法传播变量扩展后的新的约束.

$Pro(x_1, \cdots, x_n)$ 算法是约束集 $C(X)$ 上的约束传播算法 [11], 算法的执行由约束触发事件过程控制, 根据约束集 $C(X)$ 中不同约束触发事件所对应的不同的约束传播规则, $Pro(x_1, \ldots, x_n)$ 算法执行约束 $C(X)$ 的基本传播和推理. 规定变量要在回溯过程中才能够从约束中被添加或删除. 此外, 如果加入变量集合, 那么整个集合将在回溯中被删除. LIFO (last-in/first-out) 策略可以看作是一种限制, 很显然这样简单化了执行过程. Dyna-Solver 算法主要步骤如下:

步骤 1. 检测变量堆栈的长度, 即判断变量数是否变化;

步骤 2. 如果 $k \geq 1$, 停止传播算法 $Pro(x_1, \cdots, x_k)$; 把内部数据结构压入堆栈, 保存当前约束网络, 否则不做;

步骤 3. 把变量 $x_1, \cdots, x_k, x_{k+1}$ 论域压入变量堆栈, 其中 $x_{k+1} \in Y$;

步骤 4. 判断变量堆栈中是否存在未被实例化的变量. 若无, 则返回步骤 1;

步骤 5. 判断该变量的论域是否为空. 若真, 则至步骤 9;

步骤 6. 按变量论域最小、约束最多的启发式选择变量作为当前变量 $curVar$;

步骤 7. 判断当前变量论域是否有值. 若有, 则选择该值作为当前值; 否则转至步骤 9;

步骤 8. 检测当前变量的约束堆栈, 即看有无约束传播事件发生. 传播约束 $Pro(x_1, \cdots, x_k, x_{k+1})$, 若约束相容或满足, 返回 1 表示相容, 至步骤 4; 否则至步骤 5;

步骤 9. 回溯到上一个变量. 若成功, 至步骤 5; 否则程序终止, 返回 0.

如果要在回溯过程中消除一个变量, 只需要简单地颠倒算法. 也就是说, 从约束槽中删除扩展的约束 (这是真正的删除), 保存变量的论域 (这一操作在扩展约束从约束槽中删除后将自动完成), 激活以前的约束. 该动态算法是通用的, 它可以应用于任意的单调约束扩展.

接下来的定理保证了 Dyna-Solver 算法是完备的, 并且能够得到比静态算法好的解.

定理. 如果静态约束求解算法 CB-Solver 是完备和单调的, 并且约束 $C(X)$ 是单调的, 那么动态约束求解算法 Dyna-Solver 是完备的. 也就是说, 没有遗漏任何有效的赋值, 算法至少会得到和静态算法一样的结果. 形式化描述如下:

$Sol(C(X \cup Y), D(X \cup Y)) \subseteq CB - Solver(C(X \cup Y), D'(X) \cup D(Y)) \subseteq CB - Solver(C(X \cup Y), D(X \cup Y))$, 其中算法 Dyna-Solver 解为 $Sol(C(X \cup Y), D(X \cup Y))$, $D'(X) = CB - Solver(C(X), D(X))$.

4 半在线 Ship-loading 应用算例

为证实算法的有效性和可靠性, 这里选取一类离散资源调度问题作为应用算例. Shop-loading 调度是一类资源种类数为 1 的离散资源调度, 但仍是 NP 问题. 问题描述如下: 码头上有一批货物需要装卸, 称每次装卸为一次任务, 任务之间有时序关系, 任务有其各自执行时间; 任务执行需要一定量的资源 (装卸设备之类), 资源类型一致, 且资源量有限, 目标为最小化最终完成时间. 在实际任务执行过程中, 可能会有资源 (装卸设备) 出现故障而无法完成初始调度方案的任务执行计划, 即出现半在线 Shop-loading 调度情形. 此时需要对调度方案进行动态调整. 下面以该问题为例讨论应用本文针对半在线情形所设计的动态约束建模方法和求解算法.

问题 CSP 建模: 时间变量集 $X = \{st_1, st_2, \ldots, st_n\}$, 时间变量 st_i 表示每个任务 T_i 的开始时间; 论域集 $D = D_1, D_2, \ldots, D_n$, D_i 为变量 st_i 论域, 初始化为 $[0, \sum_{i=1}^{n} Dur_i]$; 约束集 $C(X)$. 时态优先约束: $Successors(T_i) = T_j$, 表示不等式约束 $st_j \geq st_i + Dur_i$; 资源容量约束: $Cumulative((st_1, st_2, \ldots, st_n), Dur_1, Dur_2, \ldots, Dur_n), (T_1(R), T_2(R), \ldots, T_n(R), h)$, 表示在任意时刻 $t(st_i \leq t \leq st_i + Dur_i)$ 有约束关系 $\sum_{i=1}^{n} T_i(R) \leq h$, 该约束为全局约束.

表 1 Ship-loading 参数

Table 1 The parameters for ship-loading scheduling

T_i	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}
$Successors$	T_2T_4	T_3	T_5T_7	T_5	T_6	T_8	T_8	T_9	T_{10}	$T_{11}T_{12}$	-	T_{13}	-
Dur_i	3	4	4	6	5	2	3	4	3	2	3	2	1
$T_i(R)$	4	4	3	4	5	5	4	3	4	8	4	5	4

例. 半在线 Shop-loading 调度问题 S, 预知任务数 $n = 13$, 资源容量 $h = 8$. 时态优先约束, 任务加工时间和资源需求量如上页表 1 所示. 假定在时刻 $t = 20$ 生成新任务 T_{14}(参数 $Dur_{14} = 3, T_{14}(R) = 1$), 则新时间变量集为 $Y = \{st_{14}\}$. 扩展约束为: $Successors(T_4) = T_{14}$, 即 $st_{14} \geq stst_4 + Dur_8$, $Cumulative(X \cup Y, Dur \cup Dur_{14}, T(R) \cup T_{14}(R), h)$.
测试环境如下:
硬件环境：IBM Netvista P4 2.0G CPU/256M RAM
软件环境：Windows 2000 Server SP4/Sun j2sdk1.4.0

情形 1. CB-Solver 求解

用 CB-Solver 算法求解, 即在 T_{14} 生成后, 停止当前算法执行, 重新执行算法. 调度解的甘特图表示如图 1, $makespan = 35, CPU(s) = 3.152$.

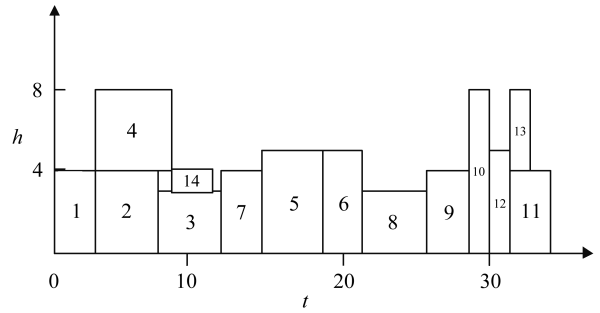

图 1　CB-Solver 求解结果

Fig. 1　The solution of CS-Solver

情形 2. Dyna-Solver 求解

用 Dyna-Solver 算法进行求解. 调度解的甘特图表示如图 2, $makespan = 35, CPU(s) = 1.875$.

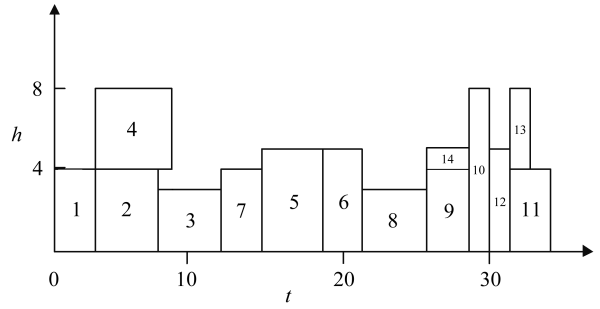

图 2　Dyna-Solver 求解结果

Fig. 2　The solution of Dyna-Solver

分析结果, 可见新约束 $Cumulative$ 中新时间变量 st_{14} 的引入并没有对初始调度目标值 ($makespan$) 产生影响. 至于新任务 T_{14} 开始时间变量 T_{14} 的实际取值在不影响调度目标值和满足所有约束的情形下, 对调度者来说并没有太大意义. 而两个算法在执行时间上的差距却值得关注. 静态修正全局解和动态修正局部解的系统 CPU 开销差异在本文的小规模测例中还不明显, 但是对于任务、资源在线变化频繁且数量大的大规模复杂工业问题, 那么每次都进行重新静态求解显然是不现实的. 当然以上的动态局部修正解是以单调约束扩展为前提的.

5　结论

基于约束的调度已成为人工智能领域的研究热点. 本文从半在线问题动态约束建模入手, 通过引入约束扩展单调性概念, 设计了一个通用的半在线调度动态单调约束求解算法, 并证明此算法的完备性, 最后通过算例测试证实了算法的有效性. 当然不是所有半在线问题中的约束扩展都具有单调性, 对于非单调性约束的建模和动态求解势必更加困难, 这有待于进一步的研究.

References

1 Fiat A, Woeginger G. *Online Algorithms: The State of Art. (Lecture Notes in Computer Sciences)*. Berlin: Springer-Verlag, 1998

2 He Yong, Yang Qi-Fan, Tan Zhi-Yi. Semi on-line problem on parallel machines (I). *Applied Mathematics - A Journal of Chinese Universities*, 2003, **18**(1): 105∼114
(何勇, 杨启帆, 谈之奕. 平行机半在线排序问题研究 (I). 高校应用数学学报, 2003, **18**(1): 105∼114)

3 He Yong, Yang Qi-Fan, Tan Zhi-Yi. Semi on-line problem on parallel machines (II). *Applied Mathematics - A Journal of Chinese Universities*, 2003, **18**(2): 213∼222
(何勇, 杨启帆, 谈之奕. 平行机半在线排序问题研究 (II). 高校应用数学学报, 2003, **18**(2): 213∼222)

4 Kellerer H, Kotov V, Speranza M R, Tuza Z. Semi on-line algorithm for the partition problem. *Operations Research Letters*, 1997, **21**(5): 235∼242

5 He Y, Zhang G S. Semi on-line scheduling on two identical machines. *Computing*, 1999, **62**(3): 179∼187

6 He Y. Semi on-line scheduling problem for maximizing the minimum machine completion time. *Acta Mathematica Sinica*, 2001, **17**: 107∼113

7 Tan Z Y, He Y. Semi on-line problem with ordinal data on two uniform machines. *Operations Research Letters*, 2001, **28**(5): 222∼231

8 Tsang E. *Foundations of Constraint Satisfaction*. London: Academic Press, 1993. 9∼11

9 Fages F, Fowler J, Sola T. A reactive constraint logic programming scheme. In: Proceedings of the International Conference on Logic Programming. Tokyo, Japan: MTT Press, 1995. 149∼163

10 Zmittal S, Falkemhainer B. Dynamic constraint satisfaction problems. In: Proceedings of National Conference on Artificial Intelligence. Boston, Massachusetts: AAAI Press, 1990. **2**: 25∼32

11 Zhang Ju-Yang, Li Xin, Sun Ji-Gui. Research on constraint-based scheduling and its implementation. In: Proceedings of CNCC'03. Beijing: Tsinghua University Press, 2003. 80∼85
(张居阳, 礼欣, 孙吉贵. 基于约束的调度研究和实现. 中国计算机大会. 北京: 清华大学出版社, 2003. 80∼85)

张居阳　吉林大学博士研究生. 主要研究方向为约束程序, 智能调度, 离散优化. 本文通信作者. E-mail: zjy2003@vip.sina.com
(**ZHANG Ju-Yang**　Ph. D. candidate at College of Computer Science and Technology, Jilin University. His research interest covers constraint programming, intelligent scheduling, and discrete optimazation. Corresponding author of this paper.)

孙吉贵　吉林大学教授. 主要研究方向为人工智能, 约束程序, 智能信息处理. E-mail: jgsun@jlu.edu.cn
(**SUN Ji-Gui**　Professor at College of Computer Science and Technology, Jilin University. His research interest covers artificial intelligence, constraint programming, and intelligence information processing.)

杨轻云　吉林大学博士研究生. 主要研究方向为约束程序, 计算智能, 群智能. Email: qyyang_jlu@yahoo.com.cn
(**YANG Qing-Yun**　Ph. D. candidate in College of Computer Science and Technology, Jilin University. His research interest covers constraint programming, computatial intelligence, and swarm intelligence.)