一种面向作业的快速调度算法*

黄启春 陈 奇 俞瑞钊

(浙江大学人工智能研究所 杭州 310027)

摘要 面向作业的调度(job oriented scheduling,简称 JOS)在实际作业车间(job shop)调度中得到普遍的应用,它的基本思想是将作业一个个地安排到工作机器上.该文提出了一种基于计算机 JOS 系统的快速调度算法,该算法指定作业操作的可行调度起始和结束时间以正排工序或逆排工序方式将它们安排到有限能力的工作机器上.通过记录和修改每一机器有效时间槽的办法来减少操作在每一机器上搜索可行时间槽的时间,从而大大提高了计算效率.实际系统应用表明,此算法对于大规模调度具有很强的优越性.

关 键 词 Job oriented scheduling(JOS),启发式算法,时间槽,Job Shop 调度.

中图法分类号 TP393

如何有效地解决调度问题是许多计算机在生产管理领域应用的重要问题,Job Shop 调度是其中一类典型的困难问题.许许多多的调度研究表明,启发式算法是解决实际应用于作业车间调度的唯一有效的方法.遗传算法[1]、并行进化算法[2]等算法已有不少算例,这些方法现在有较多的研究,也已取得了较好的效果,但尚未在实际调度中得到应用.主流的启发式调度研究是采用面向操作的启发式方法[3,4],基于派送原则,每台机器上的调度是通过选择适合在该台机器上加工的下一操作来决定的,为了生成详细的生产调度计划,一般采用仿真方法.基于仿真的系统有能力处理实际生产环境的动态性,但是它存在许多需要解决的问题,如计算机处理时间耗费、模型仿真能力以及用户对仿真结果的接受程度等.

与面向操作的启发式方法不同,另一类启发式方法即面向作业的调度(job oriented scheduling,简称 JOS)方法[5,6],它以作业的方式进行调度,只有在某一作业的所有操作已经调度完成之后才考虑下一作业.JOS 不采用仿真模型,生成详细的生产调度计划是通过将作业一个接一个地安排到不同的机器上.这一方法同一般工业生产中实际应用的手工调度方法所采用的甘特图法非常相近.甘特图能有效地用来监视生产活动过程.然而手工方法生成甘特图非常耗费时间,特别是对于那些大规模的生产调度,手工方法无法适应适时调度,基于计算机的 JOS 系统却能很好地解决这一问题.

本文介绍了一般作业车间调度问题的 JOS 模型特征及其调度处理过程,并提出一种基于时间槽的面向作业快速搜索算法.

1 面向作业的调度模型

1.1 JOS模型

下面给出一个反映 Job Shop 调度问题一般特征的模型:

- 每个车间有多个加工中心;
- 每个加工中心有许多并行的机器;
- 每个加工中心必须处理许多作业;

---

* 本文研究得到国家自然科学基金资助.作者黄启春,1972 年生,博士生,主要研究领域为人工智能,ERP.陈奇,1963 年生,副教授,主要研究领域为人工智能,IDSS,ERP.俞瑞钊,1937 年生,教授,博士生导师,主要研究领域为人工智能,数据仓库,数据挖掘.

本文通讯联系人:黄启春,杭州 310027,杭州大学(玉泉校区)人工智能研究所

本文 1998-07-06 收到原稿,1998-11-10 收到修改稿

- 不同的机器有不同的能力约束;
- 每一作业由一个需求序列(操作)组成;
- 每种操作需要若干资源,这里仅考虑机器;
- 每台机器可用于不同类型的操作,每种操作可用若干台机器;
- 每种操作需占用某机器一段时间;
- 不同操作之间存在运送时间;
- 不同作业有不同的调度结束时间约束.

对于某些问题可能还有其他约束.调度问题就是决定每一作业的操作在指定机器上的可能调度起始时间和结束时间.调度的目标是,对给定作业,要么让它尽可能早完成,要么使它最靠近指定时间完成.

1.2 调度处理过程

调度过程首先是根据作业间的依赖关系决定作业的调度顺序,如管理上的优先级、结束时间先后、松弛时间、作业数量以及考虑装配关系的次序等.调度方法一般存在两种形式,即正排工序法和逆排工序法.正排工序法对每一作业的调度从第 1 道工序开始考虑调度的起始时间,根据操作过程间的相互关系,每一操作被安排到可行加工时间最早的机器上.正排工序法的目的是使作业最早完成.逆排工序法对每一作业的调度从最后一道工序开始考虑调度的结束时间,根据作业结束时间要求及操作过程的约束,每一操作被安排到可行加工时间最迟的机器上.逆排工序法的目的是使作业的完成时间最接近指定的作业结束时间.

由此可以看出,JOS 算法的关键在于将不同的操作正排或逆排至给定能力约束的机器上.计算每一操作在不同机器上的调度起始时间和结束时间是通过在给定的机器上寻找最早(正排工序)或最迟(逆排工序)的有效时间槽来完成的.对某一操作,要使给定机器上的时间槽有效,必须满足以下条件:(a) 该时间槽未被别的操作占用;(b) 时间槽的大小必须大于或等于该操作时间;(c) 该时间槽必须满足给定操作的所有约束条件.因此,调度过程的计算时间依赖于搜索有效时间槽的步数,一般情况下,在某一机器上调度的操作数量越大,随后调度的搜索步数就越多.

为了减少搜索时间槽的时间,我们对每一台机器均建立了一个时间槽缓冲区,在缓冲区中存储那些操作间隙大于 0 的时间槽值,调度过程就是不断搜索时间槽缓冲区,找到合适的时间槽后对缓冲区进行修改的过程.时间槽的修改过程存在 3 种情况:(a) 操作的起始时间和结束时间均不等于时间槽的起始结束时间,这时,时间槽数将增加 1;(b) 操作的起始时间和结束时间与时间槽的起始结束时间有一个是对应相等的,则时间槽的数量不会改变而只是时间槽值发生改变;(c) 操作的起始时间和结束时间均与时间槽的起始结束时间对应相等,这时对应的时间槽将消失.很显然,时间槽的数量不会随着调度数量的增长而增多,因此调度量越大,该算法所节省的时间就越多.

2 算 法

对于一般的作业车间,有 I 个作业需要调度,每一作业 $i(i=1,2,\ldots,I)$有 J_i 个操作,作业 i 的每一操作 $(i,j)(j=1,2,\ldots,J_i)$将要安排到可以加工该操作的加工中心 k 的第 $l(l=1,2,\ldots,L_k)$台机器上,其中 L_k 为该加工中心的机器数.调度的算法就是在机器上寻找最早或最迟的有效时间槽.i,j,k 之间的关系在调度之前已经确定,即操作(i,j)在指定的加工中心 k 上进行加工,而 i,j,k 与 l 的关系是通过调度过程计算来确定的.

我们首先给出正排工序算法,操作(i,j)调度到机器(k,l)上的调度起始和结束时间是通过计算以下时间段来实现的.

(1) 操作(i,j)可能的起始时间$(Pst_{i,j,k,l})$.它是由施与(i,j)上的约束决定的,主要包括:(a) 如果(i,j)为某一作业的第 1 道工序,则 $Pst_{i,j,k,l}$ 为作业的起始时间;(b) 如果(i,j)不是某一作业的第 1 道工序,则 $Pst_{i,j,k,l}$ 为上一道工序的结束时间;(c) 根据前一道工序的结束时间考虑优先级以及工序间的运输时间;(d) 机器(k,l)的开工时间.

(2) 操作(i,j)可能的结束时间$(Pft_{i,j,k,l})$.它是由 $Pst_{i,j,k,l}$ 加上操作(i,j)在(k,l)上的加工时间计算得到的.

(3) 操作(i,j)在机器$(k,l)(l=1,2,\ldots,L_k)$上加工的可行起始和结束时间$(Fst_{i,j,k,h},Fft_{i,j,k,l})$.它是根据时间段$(Pst_{i,j,k,h},Pft_{i,j,k,l})$与机器$(k,l)$上的时间槽进行比较后确定的.

(4) 操作(i,j)的调度结束时间$(SFT_{i,j,k,l})$是由以上不同机器上的计算结果取最小的可行结束时间.操作(i,j)的调度起始时间$(SST_{i,j,k,l})$是与调度结束时间相对应的最小可行起始时间,这将保证操作尽可能早完成.如果有两台或多台机器满足条件,则按顺序取最前面的一台.

算法的效率决定于寻找可行的起始和结束时间$(Fst_{i,j,k,l},Fft_{i,j,k,l})$,即找到与$(Pst_{i,j,k,l},Pft_{i,j,k,l})$相匹配的时间槽,为此,我们给每一台机器$(k,l)$开设一个时间槽缓冲区存储不同的时间槽值.$Tss_{k,l,m}$表示机器$(k,l)$上第 m 个时间槽的起始时间,$Tfs_{k,l,m}$表示机器(k,l)上第 m 个时间槽的结束时间,其中 $m=1,2,\ldots,n$,n 为时间槽数.

下面,首先给出正排工序法调度操作(i,j)到机器(k,l)上的算法过程.

初始设置: $Pst_{i,j,k,l}=Pft_{i,j,k,l}=Fst_{i,j,k,l}=Fft_{i,j,k,l}=SFT_{i,j,k,l}=SST_{i,j,k,l}=0$

$Tss_{k,l,m}=0$,　$Tfs_{k,l,m}=t_{\max}$ ($t_{\max}$为一给定的足够大的值,根据系统调度规模来确定)

S1. 将机器$(k,l)(l=1,2,\ldots,L_k)$按一定的顺序排列.

S2. 根据操作(i,j)的约束条件确定 $Pst_{i,j,k,l}$.

S3. 通过以下步骤计算 $Fst_{i,j,k,l}$和 $Fft_{i,j,k,l}$.

S3.1. 计算 $Pft_{i,j,k,l}$由 $Pst_{i,j,k,l}$加上操作(i,j)在机器(k,l)上的操作时间确定.

S3.2. 如果 $l>1$ 且 $Pft_{i,j,k,l}\geq Fft_{i,j,k,l}(l\in\{1,2,\ldots,(l-1)\})$,说明该操作不可能在这台机器上加工,返回 S1,取下一台机器重新计算,直到 $l=L_k$,否则执行 S3.3.

S3.3. 寻找合适的时间槽.

S3.3.1. 将时间槽$(Tss_{k,l,m}\quad Tfs_{k,l,m})m\in(1,2,\ldots,n)$按时间顺序从小到大排序.

S3.3.2. 如果 $Pft_{i,j,k,l}>Tfs_{k,l,m}$或 $Pft_{i,j,k,l}-Pst_{i,j,k,l}>Tfs_{k,l,m}-Tss_{k,l,m}$,说明该时间槽不合适,取 $m=m+1$,返回 S3.3.1.

S3.3.3. 如果 $Pst_{i,j,k,l}\leq Tss_{k,l,m}$,则

$$Pst_{i,j,k,l}=Tss_{k,l,m},$$

$$Pft_{i,j,k,l}=Pst_{i,j,k,l}+\text{操作}(i,j)\text{的操作时间},$$

S4. 取 $Fst_{i,j,k,l}=Pst_{i,j,k,l}$,$Fft_{i,j,k,l}=Pft_{i,j,k,l}$.

S5. 重复执行 S1 到 S4 直到 $l=L_k$.

S6. 操作(i,j)的调度起始时间和调度结束时间由下式给出:

$$SFT_{i,j}=SFT_{i,j,k,p}=\min\{Fft_{i,j,k,l}\},\ l=1,2,\ldots,L_k,$$

$$SST_{i,j}=SST_{i,j,k,p}=Fst_{i,j,k,p}.$$

其中操作(i,j)被调度到机器(k,p)上,$p\in\{1,2,\ldots,L_k\}$

S7. 根据以下 4 种情况修改时间槽值:

(1) 如果 $Tss_{k,l,m}=Fst_{i,j,k,p}$ 且 $Tfs_{k,l,m}=Fft_{i,j,k,p}$,则删除时间槽$(Tss_{k,p,m}\quad Tfs_{k,p,m})$;

(2) 如果 $Tss_{k,l,m}=Fst_{i,j,k,p}$,则将时间槽$(Tss_{k,p,m}\quad Tfs_{k,p,m})$改为$(Fft_{i,j,k,p}\quad Tfs_{k,p,m})$;

(3) 如果 $Tfs_{k,l,m}=Fft_{i,j,k,p}$,则将时间槽$(Tss_{k,p,m}\quad Tfs_{k,p,m})$改为$(Tss_{k,p,m}\quad Fst_{i,j,k,p})$;

(4) 以上条件均不满足,首先将时间槽$(Tss_{k,p,m}\quad Tfs_{k,p,m})$改为$(Tss_{k,p,m}\quad Fst_{i,j,k,p})$,再增加一个时间槽$(Fft_{i,j,k,p}\quad Tfs_{k,p,m})$.

逆排工序算法与正排工序算法的基本思想是一样的,只是计算过程是从工序逆向进行.例如,调度过程首先是确定 $Pft_{i,j,k,l}$,它由以下几种情况确定:(a) 如果工序(i,j)为最后一道工序,则 $Pft_{i,j,k,l}$为作业的结束时间;(b) 如果工序(i,j)不是最后一道工序,则 $Pft_{i,j,k,l}$为后一工序的起始加工时间;(c) 根据后一工序的起始加工时间考虑优先级及运送时间计算得到;(d) 机器的工作结束时间.$Fst_{i,j,k,l}$和 $Fft_{i,j,k,l}$是每台机器上最迟的可行时间段$(Pst_{i,j,k,l},Pft_{i,j,k,l})$合适的时间槽搜索过程变为:

S3.3.1. 将时间槽$(Tss_{k,l,m}\quad Tfs_{k,l,m})(m\in(1,2,\ldots,n))$按时间顺序从大到小排序.

S3.3.2. 如果 $Pst_{i,j,k,l}<Tss_{k,l,m}$或 $Pft_{i,j,k,l}-Pst_{i,j,k,l}>Tfs_{k,l,m}-Tss_{k,l,m}$,说明该时间槽不合适,取 $m=m-1$,返回 S3.3.1.

S3.3.3. 如果 $Pft_{i,j,k,l}\geq Tfs_{k,l,m}$,则

$$Pft_{i,j,k,l}=Tfs_{k,l,m},$$

$Pst_{i,j,k,l}=Pft_{i,j,k,l}$—操作$(i,j)$的操作时间.

操作(i,j)的调度起始和结束时间由下式给定:

$$SFT_{i,j}=SFT_{i,j,k,p}=\max\{Fft_{i,j,k,l}\},\ i=1,2,\ldots,L_k,$$

$$SST_{i,j}=SST_{i,j,k,p}=Fst_{i,j,k,p}.$$

其中,操作(i,j)被调度到机器(k,p)上, $p\in\{1,2,\ldots,L_k\}$.

3 应用实例比较分析

在面向作业的调度算法中,以 Chung-HsingYeh 所提出的基于操作块的快速算法的效率最高(以下简称操作块算法)[6].其基本思想是将同一机器上时间间隙为 0 的操作以链表形式存储为一个块,这样,在搜索有效时间槽时,通过搜索操作链避免了对 0 时间槽的比较判断,从而缩短了搜索时间.其实现方式是通过记录每一操作的紧前工序$(dfo_{k,l}(i,j))$和紧后工序$(dpo_{k,l}(i,j))$来达到的.该算法随着操作链的增大,搜索时间虽然不像一般算法那样呈级数增长,但也必须耗费较多的时间.

本文提出的基于时间槽的快速算法(以下简称时间槽算法)以最直接的方式记录每一台机器上的有效时间槽值,从而大大减少了搜索时间.该算法已经在实际 MRP-II系统运行中应用.下面我们以某拖拉机厂金工车间的生产调度过程为例,对一般算法、操作块算法及时间槽算法的效率进行比较,比较程序以关系数据库存储调度信息数据,前端开发工具为 Powerbuilder,后端数据库采用 Sybase,在奔腾 200,内存为 32 兆的微机上运行.该金工车间的最大作业数为 99,总操作数为 1 084,机器数为 26,分别取 6 种不同的情况,采用 3 种方法进行计算比较,结果见表 1.

表 1 一般算法、操作块算法与时间槽缓冲区算法效率的比较

作业数	操作(工序)数	机器数	一般方法计算时间(s)	操作块算法计算时间(s)	时间槽算法计算时间(s)
13	100	13	16.781	13.423	26.762
22	200	20	34.316	26.397	33.851
41	400	23	82.266	60.938	68.748
56	600	23	138.202	102.372	101.739
71	800	25	234.130	161.469	135.576
99	1084	26	391.604	261.069	181.197

由表 1 可以看出,当调度作业数等于 13,操作数等于 100 时,操作块算法最快时间为 13.423s,其次为一般算法,耗费时间为 16.718s,而时间槽算法耗费时间最多为 26.762s,当操作数在 200~600 之间时,操作块算法与时间槽算法均比一般算法要快,但提高得不多.操作块算法与时间槽算法效率相差不大,而当操作数等于 800 和 1 084 时,操作块算法的效率比一般算法分别提高 45%和 50%,而时间槽算法的效率比一般算法分别提高 73%和 116%.这说明时间槽算法对大规模的调度具有很强的优越性.图 1 是调度过程中某车床的时间槽缓冲区变化情况,该车床上调度的总操作数为 330,而时间槽数最大值仅为 10,由此可见,调度过程中搜索有效时间槽的时间随着调度数量的增长不会有太大的增长.

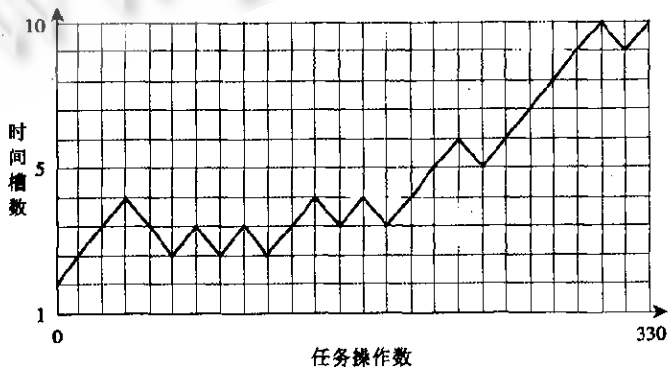

图 1 时间槽数随调度作业数的增长情况

以上表明,分析显示基于时间槽缓冲区的快速算法从本质上提高了一般 JOS 算法的效率.虽然随着调度规模的增大,算法所需耗费的时间随之增长,但它不再是呈级数式增长,这一算法完全可以应用于适时调度环境和大规模的调度系统.

4 结　论

作业车间调度问题的计算复杂性发展了许多针对实际工业生产调度的有效方法,随着调度规模的增大以及考虑作业车间环境的动态性计算机辅助调度成为必然.基于计算机的 JOS 算法能够提供高质量的、可行的调度计划而在实际生产中得到应用.本文提出的快速 JOS 算法大大减少了计算处理时间,对大规模的调度问题有更明显的效果.

参考文献

1 陈恩红,刘贵全,蔡庆生.基于遗传算法的 Job-Shop 调度问题求解方法.软件学报,1998,9(2):139~143
(Chen En-hong, Liu Gui-quan, Cai Qing-sheng. A genetic algorithm based job-shop scheduling problem solving method. Journal of Software, 1998,9(2):139~143)

2 方剑,席裕庚.应用并行进化规划求解 Job-Shop 调度问题.模式识别与人工智能,1997,10(4):344~350
(Fang Jian, Xi Yu-geng. Using parallel evolutionary programming to solve the job-shop scheduling problem. Pattern Recognition and Artificial Intelligence, 1997,10(4):344~350)

3 Sun D, Batta R, Lin L. Effective job shop scheduling through active chain manipulation. Computers and Operations Research, 1995,22(2):159~172

4 Kim Y D. A backward approach in list scheduling algorithms for multi-machine tardiness problems. Computers and Operations Research, 1995,22(3):307~319

5 Hastings N A J, Marshall P H, Willis R J. Scheduled based MRP: an integrated approach to production scheduling and material requirements planning. Journal of the Operational Research Society, 1982,33(11):1021~1029

6 Yeah Chung-Hosing. A fast finite loading algorithm for job oriented scheduling. Computers and Operations Research, 1997, 24(2):193~198

A Fast Job Oriented Scheduling Algorithm

HUANG Qi-chun　CHEN Qi　YU Rui-zhao

(Institute of Artificial Intelligence　Zhejiang University　Hangzhou　310027)

Abstract　Job oriented scheduling (JOS) has been the most commonly used technique in actual job shop scheduling. It loads jobs one by one onto machines. In this paper, the authors present a fast scheduling algorithm of computer-based JOS system, the algorithm assigns feasible schedule start and finish times to the operations of a job by loading them forward or backward onto the capacity constrained machines. The computation time to find the feasible time slot on the machine is reduced by log and modify each machine's feasible time slot. Thus, the computational efficiency is substantially improved. Experimental testing shows that the algorithm has significant merits for large size problems.

Key words　Job oriented scheduling (JOS), heuristics algorithm, time slot, job shop scheduling.