基于遗传算法的滚动调度策略*

方 剑 席裕庚
(上海交通大学自动化研究所·上海, 200030)

摘要: 本文研究了动态加工环境下的一类 Job-Shop 调度问题,提出了一种基于遗传算法的滚动调度策略,其要点是:1) 借鉴预测控制的思想,采用 time-based 和 job-based 的滚动调度策略适应动态环境和要求的多变性. 2) 以遗传算法和分派规则相结合,处理考虑与操作序列有关的工件安装时间和工件到期时间约束的复杂调度问题. 文中给出了在工件到期时间发生改变的动态环境中两种滚动调度算法的调度结果,并与静态调度结果作了比较.

关键词: Job Shop; 滚动优化; 遗传算法; 分派规则

1 引 言

实际 FMS 中的工件调度多为动态环境下的 Job Shop 调度问题,尽管已有学者提出了一些动态环境下的重调度策略[1~3],但工件调度中存在的两个主要问题,即适应环境的动态变化和问题求解规模过大,一直没有得到很好的解决. 本文借鉴连续系统中预测控制的基本原理,提出一种新的滚动调度策略. 通过调度区间的滚动不仅能够跟踪系统的动态变化,而且能得到满意的调度结果. 此外由于每次只对部分工件进行调度,使问题求解规模大大减少.

本文研究的 Job Shop 工件调度问题以系统加工时间最短为目标,其中考虑了与工序序列有关的工件安装时间和工件到期时间的约束,并且在加工过程中发生工件到期时间改变这样的突发事件. 文中的滚动调度策略分为 time-based 和 job-based 两种. 而调度算法则从问题分解角度用遗传算法决定各工件的每个工序应分配到哪台机器上加工,而对每台机器则运用分派规则来决定相应工件在此机器上加工的次序和开始加工时间. 文中分别给出了这两种调度策略的调度结果,将它们与一次性全局最优调度结果进行比较,并分析了这两种调度策略的差异.

2 Jop Shop 调度问题的描述

本文研究较为一般性的 Jop Shop 调度问题,具体地说它有以下的特点:

1) 有 M 台机器,每台机器可加工若干工序,不同的机器能加工的工序不全相同,且不同的机器完成相同工序的加工时间也不全相同.

2) 有 k 类工件,每类有若干数目的工件,每个工件有若干工序,各工序间的顺序不可改变,不同类工件所需的工序也不完全相同.

3) 工件在机器上加工时,工件的安装时间与此机器前一加工工序有关,这是因为在 FMS 中机器能加工不同的工序是通过更换刀具来实现的,恰当的工序序列能使机器减少刀具的更换,从而减少工件的安装时间.

4) 每个工件有到期时间,相同类工件的到期时间相同,不同类别的工件的到期时间不同. 工件的到期时间在加工过程中可以发生改变,当工件到期时间发生改变时,属此类的所有工件的到期时间变化相同.

* 上海市自然科学基金资助项目.
本文于 1995 年 6 月 26 日收到. 1995 年 12 月 26 日收到修改稿.

5) 在零时刻所有工件都可得,处于待加工状态.

本文要求工件的加工时间最短,同时要使工件尽量不超期,为此对超期工件在性能指标中加入惩罚项,这一性能指标可写为

$$\min C = \max_{i \in S_M} L_i + \alpha \sum_{j \in S_J} \max(0, t_{cj} - t_{dj}). \tag{2.1}$$

其中 α 为惩罚项加权系数,S_M 为机器集,S_J 为工件集,L_i 为第 i 台机器完成所有工序加工的时间,t_{cj} 为第 j 个工件完成加工的时间,t_{dj} 为第 j 个工件的到期时间. 要求合理地将工件分配到相应的机器上,并恰当安排每台机器上工件的加工次序和每个工序的开始加工时间,在满足工序顺序约束的同时,使性能指标得到优化.

3 滚动调度策略

实际的 FMS 加工环境中充满了随机和不确定因素,一次性全局最优的静态调度虽然可得到理想的最优结果,实际上并不能适应动态环境. 对照连续系统中预测控制取代最优控制在工业过程中取得广泛的应用,就是因为它放弃了全局最优的概念,通过在线优化每个滚动区间,使系统在此区间内达到最优,通过滚动而得到较为满意的控制结果. 这一思想同样可应用到动态环境下的 FMS 调度中. 通过滚动调度不仅可以克服不确定因素的影响,而且由于只对当前滚动窗口中的工件进行调度,将大大地减少问题的求解规模.

滚动调度的主要思想是滚动优化. 在初始时刻,从所有待加工工件中选取一定数目的工件,称之为工件窗口,对这些工件进行调度,然后按调度结果对工件进行加工. 过一定时间之后,如果某工件的所有操作都完成加工,则将它从工件窗口中移去,再选择一些待加工工件到工件窗口中使工件窗口中的工件数目保持衡定,重新对工件窗口内的工件进行调度. 这样的过程重复进行,直到所有工件都完成加工. 滚动调度的要点如下:

3.1 工件窗口

在滚动调度中首先要定义一个滚动窗口,本文中的滚动窗口为工件窗口,即一定数量的工件集,每次调度只对当前工件窗口中的工件进行,并依照调度结果进行加工. 通过从工件窗口中移去已完成加工的工件和向其中添加待加工工件来实现工件窗口的滚动. 工件窗口中的工件数目和待加工工件的选择规则是决定工件窗口的两个要素,它们会影响工件调度的整体效率. 工件窗口中的工件越多,调度结果越接近全局最优解,但所需的时间也越长. 在本文的指标函数中对超期工件要加入惩罚项,因此文中从待加工工件中选取具有最早到期时间的工件到工件窗口中. 在工件加工接近结束时,工件窗口中的工件将逐渐减少,直到其中的工件数目为零时加工完成.

3.2 滚动机制

滚动调度按其机制可分为 time-based 和 job-based 两种,time-based 滚动调度是指在完成一次调度之后,按调度结果进行加工,经过一定时间 ΔT 之后,将所有已完成加工的工件都从工件窗口中移去,再从待加工工件中选择若干工件,保持工件窗口中工件数目衡定,然后进行重调度. 而 job-based 滚动调度则是在工件按调度结果进行加工中不断地检查工件窗口中的工件的加工状态,一旦某一工件完成加工,就将它从工件窗口中移去,从待加工工件中选取一个工件到工件窗口,再进行重调度.

在 time-based 和 job-based 调度中,如果取工件窗口中工件数目为所有待加工工件,则它就分别对应于一般的周期性调度和连续性调度.

3.3 调度问题的描述

滚动调度是优化当前工件窗口中工件的加工性能指标，与全局调度中的性能指标有些不同. job-based 和 time-based 滚动调度这两种问题的描述也不一样. job-based 滚动调度问题可描述为：

$$\begin{cases} \min C = \max\limits_{i\in S_M}(L_i - t_{w,n}) + \alpha\sum\limits_{i\in S_n}\max(0, t_{cj} - t_{dj}), \\ L_i = t_{ir} + \sum\limits_{k\in K_{in}}(t_{wik} + t_{sik} + t_{pik}) + t_{w,n}, \quad i \in S_M, \\ t_{w,n+1} = \min\limits_{j\in S_n} t_{cj}, \\ t_{fj,l} \leqslant t_{sj,l+1}, \quad j \in S_n, \quad l = O_{jn},\cdots,O_j - 1. \end{cases} \tag{3.1}$$

其中 S_n 为第 n 次调度时工件窗口中的工件集，K_{in} 为第 n 次调度时机器 i 上工件的未加工工序集，$t_{w,n}$ 为第 n 次调度的开始时间，t_{ir} 为机器 i 上正在加工工件的剩余加工时间，O_{jn} 为工件 j 在此工件窗口中加工的第一个工序，其它的符号定义与前面的定义相同.

而 time-based 滚动调度问题可描述为：

$$\begin{cases} \min C = \max\limits_{i\in S_M}(L_i - t_{w,n}) + \alpha\sum\limits_{j\in S_n}\max(0, t_{cj} - t_{dj}), \\ L_i = t_{ir} + \sum\limits_{k\in K_{in}}(t_{wik} + t_{sik} + t_{pik}) + t_{w,n}, \quad i \in S_M, \\ t_{w,n+1} = t_{w,n} + \Delta T, \\ t_{fj,l} \leqslant t_{sj,l+1}, \quad j \in S_n, \quad l = O_{jn},\cdots,O_j - 1. \end{cases} \tag{3.2}$$

其中 ΔT 为调度间隔，其余符号的定义与前面相同.

3.4 调度算法

本文采用将遗传算法和分派规则相结合的调度算法. 鉴于这里考虑的是有与工序序列有关的工件安装时间和工件到期时间约束的复杂调度问题，故从问题分解的角度将 Job Shop 中工件调度的两个要素分开考虑，即将各工件的每个工序分配到相应能加工此操作的机器上，以及决定各工件在相应机器上的加工次序和开始加工时间. 对第一部分，解的搜索空间是非常巨大的，用常规方法难以求解. 而遗传算法是通过选择、交换和变异等遗传操作使群体进化来进行全局优化搜索的，它本身虽然不能减小搜索空间的大小，但由于群体搜索的独特机制使得它能有效地搜索较大的解空间. 因而本文用遗传算法来解决工件的各个工序分配到合适的机器上的问题. 而分派规则简单易行，容易满足工件工序顺序的约束，因而可采用分派规则来决定每台机器上待加工工件的加工次序和开始加工时间，由此可决定相应个体的适合度值. 由于遗传算法中每个个体相应于一种调度方式，通过遗传算法中的选择，交换和变异等操作将使群体中的个体逐步进化，最后使工件的各个工序在各加工机器上的分配达到最优. 当然这个最优解与所采用的分派规则有关，采用不同的分派规则所得到的最优值有可能不一样. 文中对过期工件加惩罚项，因此采用最早到期分派规则 EDD，即从各机器的工件缓冲区中选择最早到期的工件加工. 关于遗传算法优化的原理可参考文献[4].

4 仿真算例

在本文的仿真系统中有四台机器，它们各自能加工的工序为：m1：A，B，C，F；m2：B，C，D，E；m3：B，D，F；m4：E，F. 工件有三类，各类工件所需的工序为：J1：A，C，B，D；J2：B，D，E；J3：B，A，F. 四台机器加工工件时的安装时间和加工相应工序的时间因篇幅所限不再列出. 三类工

件的到期时间分别为:300,400 和 500,且每类工件各有 10 个工件要加工. 工件的编号如下:1 ～ 10 号为 J1 类工件,11 ～ 20 号为 J2 类工件,而 21 ～ 30 号则是 J3 类工件.

加工过程中在时刻 $t = 150$,J3 类工件的到期时间发生改变,由 500 变为 350. 为评价动态调度结果,把式(2.1) 中的整体指标函数作为衡量调度结果的标准,取目标函数中的惩罚项系数 $\alpha = 0.65$,遗传算法中有 30 个个体,交换概率 $P_c = 0.765$,变异概率 $P_m = 0.0875$. 群体经过 300 代进化算法收敛. 采用静态调度算法得到四台机器调度结果的 Gantt 图如图 1 所示,其中各方框表示相应工序的加工时间(包括工件安装时间),方框内的字符代表所加工的工序,方框上的数字表示该工序所属的工件号. 这时总的加工时间 $L = 521.17$,代价值 $C = 1412.89$. 在 job-based 滚动调度中取工件窗口中的工件数目为 14,得四台机器加工结果的 Gantt 图如图 2 所示,相应的总加工时间 $L = 530.52$,代价值 $C = 1295.08$. 在 time-based 滚动调度中取工件窗口中的工件数为 14,调度间隔 $\Delta T = 54$,可得四台机器调度结果的 Gantt 图如图 3 所示,此时总加工时间 $L = 526.44$,代价值 $C = 1307.43$.

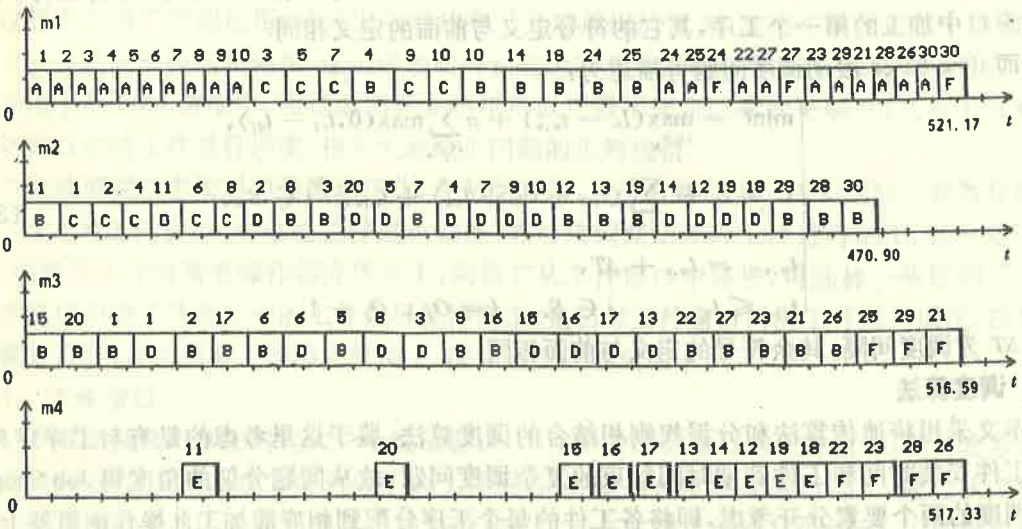

图 1 静态调度结果

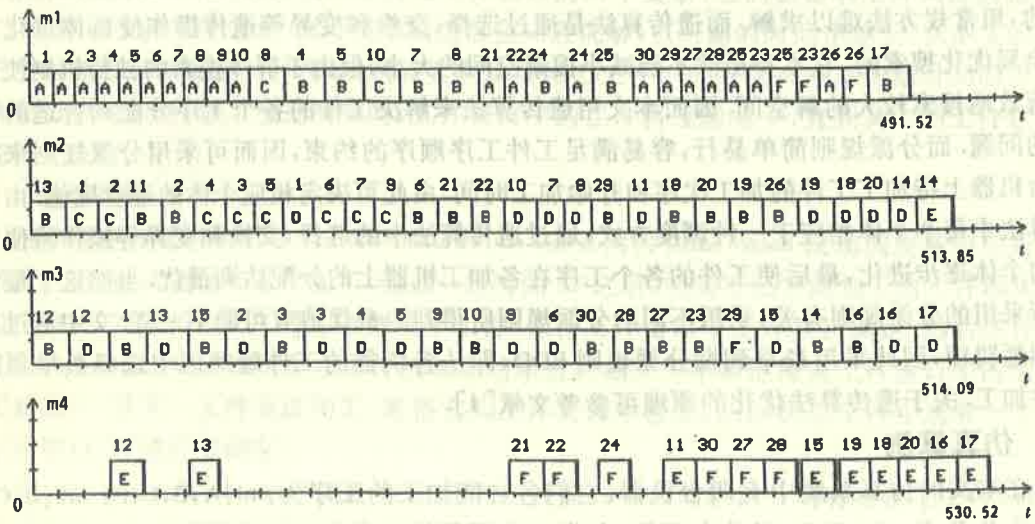

图 2 工件窗口中工件数目为 14 时,四台机器的 job-based 滚动调度结果

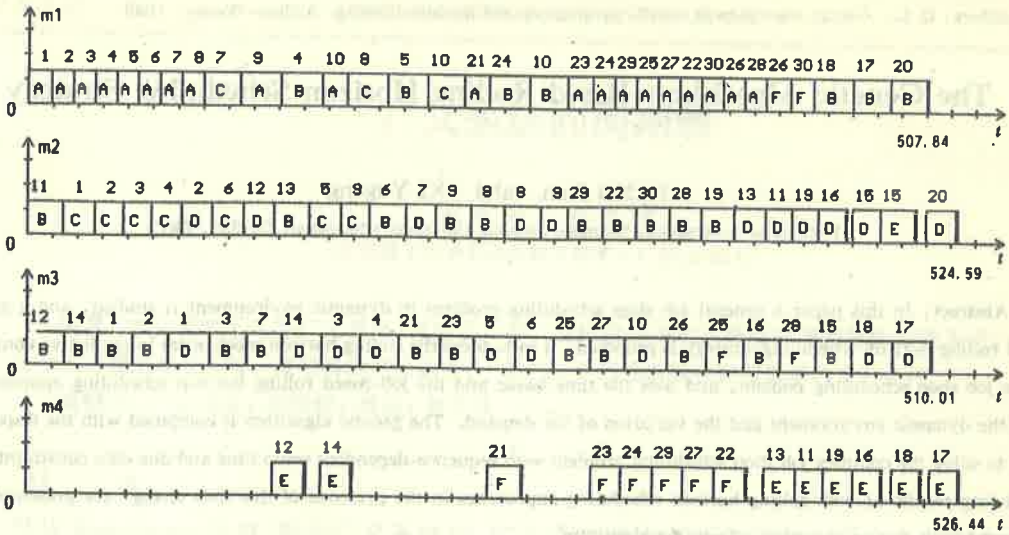

图 3　工件窗口中工件数目为 14，$\Delta T = 54$ 时，四台机器 time-based 滚动调度结果

由此可得出下面的结论：

1）如果仅考虑加工时间，则静态调度得到的时间较短，但完成一次调度所需的计算量最大，而 time-based 和 job-based 滚动调度可得到相当接近静态调度的结果，且每次调度求解的问题从 4 台机器 30 工件问题降为 4 台机器 14 工件，计算量大大减少.

2）考虑加工环境的变化，则静态调度的总体性能较差，而 time-based 和 job-based 滚动调度可得到较好的调度性能. 例中在加工开始时 J2 类工件的到期时间比 J3 类工件早，应比 J3 类工件早加工. 但在时刻 $t = 150$ 时 J3 类工件的到期时间由 500 变为 350，J3 类工件应提前加工. 图中可看出，静态调度在 J3 类工件的到期时间发生改变后 J3 类工件仍在 J2 类工件之后加工，而在 time-based 和 job-based 滚动调度的结果中 J3 类工件的加工时间提前，比 J2 类工件早加工，这说明了滚动调度的确能跟踪环境的变化. 因此，滚动调度所得的代价值比静态调度明显减小.

3）在 job-based 滚动调度算法中，重调度的次数等于工件的总数，重调度的时间间隔不确定，与工件和调度算法有关. 而在 time-based 滚动调度中，ΔT 大小决定了重调度的次数和调度算法对突发事件的反应能力，ΔT 越小则重调度越频繁，对突发事件的反应能力就越强，但计算量也越大.

5　结　　论

本文研究了工件到期时间发生改变的动态加工环境下 Job Shop 调度问题，采用了滚动调度策略，不仅使问题的求解规模大大下降，而且使调度系统能跟踪环境的变化，获得较好的调度结果.

参 考 文 献

1　Yamamoto, M. and Nof, S. Y.. Scheduling/Rescheduling in the manufacturing operating system environment. Int. J. Prod. Res., 1985, 23(4): 705—722

2　Rong Kwei LI, Yu Tang ShYu and Sadashiv Adiga. A heuristic rescheduling algorithm for computer-based production scheduling systems. Int. J. Prod. Res., 1993, 31(8): 1815—1826

3　Laurak. Church and Reha Uzsoy. Analysis of periodic and event-driven rescheduling policies in dynamic shops. Int. J. Computer Integrated Manufacturing, 1992, 5(3): 153—163

4 Goldberg, D. E.. Genetic algorithms in search, optimization and machine learning. Addison-Wesley, 1989

The Genetic Algorithms-Based Rolling Horizon Scheduling Strategy

FANG Jian and XI Yugeng

(Institute of Automation, Shanghai Jiaotong University • Shanghai, 200030, PRC)

Abstract: In this paper a general job shop scheduling problem in dynamic environment is studied, and a genetic-based rolling horizon scheduling strategy is proposed, It introduces the rolling horizon mechanism in predictive control into the job shop scheduling poblem, and uses the time-based and the job-based rolling horizon scheduling approaches to meet the dynamic environment and the variation of the demand. The genetic algorithm is combined with the dispatching rules to solve the complex job shop scheduling problem with sequence-dependent setup time and due date constraints. The scheduling results of both rolling horizon scheduling approaches in the presence of due date change are presented, and compared with that of the static scheduling algorithm.

Key words: job shop; rolling horizon scheduling; genetic algorithms; dispatching rules

本文作者简介

方 剑 1969年生. 分别于1991年,1993年在上海交通大学信息与控制工程系获得学士和硕士学位. 现为上海交通大学自动化系博士生. 主要研究方向为:FMS调度、神经网络以及进化算法等.

席裕庚 1946年生. 1968年毕业于哈尔滨军事工程学院,1984年在德国慕尼黑工业大学获工学博士学位. 现为上海交通大学教授,博士生导师. 目前主要研究方向为复杂工业过程与智能机器人的控制理论和方法.